Dobrovolni Tibor — Herboly Miklós — Tuba Péter

Számítógéprendszerek architektúrája

KSH NEMZETKÖZI SZÁMÍTÁSTECHNIKAI OKTATÓ ÉS TÁJÉKOZTATÓ KÖZPONT

Számítógéprendszerek architektúrája

Computer Systems Architecture

T. Dobrovolni – M. Herboly – P. Tuba

Computer Systems Architecture

CENTRAL STATISTICAL OFFICE
INTERNATIONAL COMPUTER EDUCATION AND INFORMATION CENTRE
Budapest, 1977

Dobrovolni Tibor – Herboly Miklós – Tuba Péter

Számítógéprendszerek architektúrája

KÖZPONTI STATISZTIKAI HIVATAL
NEMZETKÖZI SZÁMÍTÁSTECHNIKAI OKTATÓ ÉS TÁJÉKOZTATÓ KÖZPONT
Budapest, 1977

Szerkesztette: DOBROVOLNI TIBOR

Edited by TIBOR DOBROVOLNI

Lektorálta: KELLNER DÉNES

Supervised by DÉNES KELLNER

© *Dobrovolni Tibor–Herboly Miklós–Tuba Péter, 1977*

ISBN 963 7552 464

Felelős kiadó: Faragó Sándor igazgató
Felelős szerkesztő: Huba Zoltánné. Műszaki vezető: Bauer Géza.
Műszaki szerkesztő: Salamon István. A rajzokat Csaklós György készítette.
Megjelent 40,5 (A/5) ív terjedelemben, 6000 példányban.
Statisztikai Kiadó Vállalat, Nyomdaüzem – 77–1041–9
Felelős vezető: Kecskés József igazgató
Printed in Hungary

Tartalom

Előszó ---------- 11

1. Bevezetés ---------- 13

1.1 A leegyszerüsitett gépi modellek ---------- 18

1.2 A jegyzet használata ---------- 22

2. A számitógépek adatábrázolásának alapjai ---------- 25

2.1 Adatszintek ---------- 25

2.2 Számrendszerek ---------- 26

2.2.1 10-es (decimális) számrendszer. Néhány alapfogalom ---------- 26

2.2.2 2-es (bináris) számrendszer ---------- 30

2.2.3 8-as (oktális) számrendszer ---------- 31

2.2.4 16-os (hexadecimális - szedecimális) számrendszer ---------- 32

2.3 Aritmetikai müveletek a különböző számrendszerekben. Komplementálás --- 34

2.3.1 Bináris összeadás ---------- 34

2.3.2 Bináris kivonás ---------- 35

2.3.3 Bináris szorzás ---------- 36

2.3.4 Bináris osztás ---------- 37

2.3.5 Komplemensképzés ---------- 37

2.3.6 Oktális összeadás ---------- 39

2.3.7 Oktális kivonás ---------- 40

2.3.8 Hexadecimális összeadás ---------- 41

2.3.9 Hexadecimális kivonás ---------- 42

2.4 Logikai müveletek. Boole-algebra ---------- 42

2.4.1 A logikai NEM müvelet ---------- 44

2.4.2 A logikai ÉS müvelet ---------- 46

2.4.3 A logikai VAGY müvelet ---------- 47

2.4.4 A KIZÁRÓ VAGY müvelet ---------- 48

2.4.5 A Boole-algebra alaptételei ---------- 49

2.5 Konvertálás. Számrendszerek közötti átalakitás ---------- 51

2.5.1 Bináris, oktális, hexadecimális számrendszerbeli szám átalakitása decimálissá ---------- 52

2.5.2 Decimális számrendszerből bináris, oktális, hexadecimális számrendszerbe való átalakitás ---------- 53

2.5.3 Bináris számrendszerbeli szám átalakitása oktálissá és hexadecimálissá, illetőleg viszont ---------- 57

2.6 Kódolás. Kódolt decimális számábrázolás (BCD) ---------- 58

2.7 Adatábrázolás ---------- 61

2.7.1 Numerikus adatok ábrázolása ---------- 62

2.7.2 Alfanumerikus adatok ábrázolása ---------- 83

2.7.3 Logikai adat ábrázolása ---------- 92

3. A központi számitógép ---------- 95

3.1 A központi számitógépről általában ---------- 95

3.1.1 A központi számitógép felépitése ---------- 95

3.1.2 Az operativ tár felépitése. Tárrészek és cimzésük ---------- 96

3.1.3 Az aritmetikai-logikai egység felépitése, funkciói ---------- 96

3.1.4 A vezérlő egység felépitése. A mikroprogramozott és hardware vezérlés fogalma ---------- 96

3.1.5 A csatornarendszer feladata. Csatornafajták. Az önálló input/output vezérlés alapgondolata. A csatorna-interface jelentősége, funkciói -- 97

3.1.6 A kezelőpult szerepe, felépitése. Az autonóm üzemmód jelentősége -- 98

3.2 Az operativ tár ---------- 98

3.2.1 Mágneses és elektronikus információtárolási elvek ---------- 98

3.2.2 A ferritgyürüs tároló felépitése, működése. A tárciklus lezajlása. A ferritgyürüs tároló regiszterei. Példák tármüveletekre ---------- 99

3.2.3 Az elektronikus tárolóelem felépitése, működése ---------- 105

3.2.4 A kétféle tárolási elv összehasonlitása ---------- 106

3.2.5 Az operativ tár részei. Lokális, multiplex tár, főtár, kulcstár, általános és lebegőpontos regiszterek és cimzésük ---------- 107

3.2.6 A főtároló állandóan felosztott területe ---------- 108

3.2.7 A tárvédelem mechanizmusa. Tárvédelmi kulcsok ---------- 110

3.3 A processzor. Az aritmetikai-logikai egység ---------- 112

3.3.1 Az aritmetikai müveletek visszavezetése összeadásra ---------- 112

3.3.2 A félösszeadó felépitése és működése ---------- 113

3.3.3 A teljes összeadó felépitése és működése ---------- 113

3.3.4 Példák az 1 bites teljes összeadó működésére ---------- 114

3.3.5 Soros, párhuzamos és "vegyes" aritmetikai müveletvégzők ---------- 116

3.3.6 Példa konkrét aritmetikai-logikai egységre ---------- 120

3.4 A processzor. A vezérlő egység ---------- 120

3.4.1 A mikroprogramozott és hardware vezérlési mód ---------- 120

3.4.2 Bevezetés a mikroprogramozás elméletébe ---------- 121

3.4.3 Példa konkrét mikroprogramozási rendszerre. A mikroutasitás szerkezete ---------- 123

3.4.4 A mikroprogram futásának folyamata. A mikroprogram-felfüggesztések rendszere ---------- 125

3.4.5 Példa mikroutasitásra. Mikroprogram-lista értelmezése ---------- 126

3.4.6 Bevezetés a gépi kódu programozás elméletébe. 4, 3, 2, 1+1 és 1 cimes számitógépek ---------- 127

3.4.7 A 2 cimes számitógépek utasitástipusai. Cimzési módok ---------- 130

3.4.8 A gépi kódu program futása. A program állapotának nyilvántartása. A programállapotszó felépitése ---------- 133

3.4.9 A programmegszakitási rendszer müködése. A megszakitások prioritása és a feldolgozási sorrend ---------- 136

3.4.10 Példa egyszerü gépi kódu program futására. A vezérlő egység ciklusa 139

3.5 A csatornarendszer ---------- 141

3.5.1 A csatornarendszer felépitése. Alcsatornák ---------- 141

3.5.2 A szelektorcsatorna müködése ---------- 143

3.5.3 A multiplex csatorna müködése. Üzemmódok ---------- 144

3.5.4 Az input/output müveletet vezérlő információ szerkezete. A csatornacimszó, csatornaparancsszó, csatornaállapotszó felépitése ---------- 144

3.5.5 Az input/output müvelet lefolyása gépi szinten ---------- 147

3.5.6 A csatorna-interface strukturája és müködése. Példa a perifériával való kapcsolatteremtésre szabad, üzemképes és foglalt periféria esetén ---------- 150

3.6 A kezelőpult ---------- 153

3.6.1 A kezelőpult felépitése ---------- 153

3.6.2 A kijelzések és a kezelőszervek szerepe ---------- 154

3.6.3 A kezelőpult mikroprogramjainak müködése ---------- 154

3.6.4 A kezdeti programbetöltés folyamata. A számitógéprendszer üzembe helyezése ---------- 156

4. Perifériális egységek ---------- 157

4.1 Kötegelt feldolgozás ---------- 159

4.1.1 Lyukkártya ---------- 159

4.1.2 Lyukszalag ---------- 163

4.1.3 Lyukszalag vagy lyukkártya? ---------- 166

4.2 Kötegelt feldolgozás input eszközei ---------- 168

4.2.1 Lyukkártyaolvasó ---------- 168

4.2.2 Lyukszalagolvasó ---------- 172

4.2.3 Bizonylatolvasók ---------- 174

4.3 Kötegelt feldolgozás output eszközei ---------- 181

4.3.1 Sornyomtató ---------- 181

4.3.2 Lyukkártyalyukasztó ---------- 189

4.3.3 Lyukszalaglyukasztó ---------- 192

4.3.4 Rajzgép (plotter) ---------- 193

4.4 Háttértárak ---------- 201

4.4.1 Mágnesszalagos tár ---------- 203

4.4.2 Mágneslemezes tár ---------- 218

4.4.3 Mágnesdobos tár ---------- 228

4.4.4 Mágneskártyás tár ---------- 230

4.5 Interaktiv feldolgozás input/output eszközei ---------- 233
4.5.1 Karakternyomtató (konzolirógép) ---------- 233
4.5.2 Vizuális megjelenitő (display) ---------- 237

5. Az operációs rendszer ---------- 247
5.1 Az operációs rendszer kialakulása ---------- 247
5.1.1 A gépi kódu programozásról ---------- 247
5.1.2 A szimbolikus programozásról ---------- 248
5.1.3 A programfordítás fogalma és szükségessége ---------- 249
5.1.4 Assembly szintü és magasszintü programozási nyelvek ---------- 251
5.1.5 Az automatizált programfuttatás megjelenése. Monitorok, executiv programok, felügyelő programok ---------- 253
5.1.6 A multiprogramozás gyakorlata ---------- 253

5.2 Az operációs rendszer felépitése ---------- 255
5.2.1 Az operációs rendszer vezérlő programjai ---------- 257
5.2.2 Az operációs rendszer feldolgozó programjai ---------- 258

5.3 Az operációs rendszer vezérlő programja ---------- 258
5.3.1 A Supervisor felépitése és funkciói ---------- 258
5.3.2 Job Control. Az automatizált üzemeltetés megvalósitása ---------- 266
5.3.3 SPI. Egyedi programinditás ---------- 272

5.4 Az operációs rendszer feldolgozó programjai ---------- 273
5.4.1 Forditóprogramok. A programfordítás menete. A forrásprogram fogalma, szerkezete ---------- 273
5.4.2 Példa programfordításra. A fordítási lista értelmezése ---------- 278
5.4.3 A szerkesztőprogram. A programszerkesztés menete. A modulprogram fogalma, szerkezete ---------- 279
5.4.4 Példa programszerkesztésre. A fázisprogram fogalma, szerkezete -- 284
5.4.5 Programkönyvtárak. A könyvtárkezelő programok funkciói -------- 286
5.4.6 Példa a katalogizálásra a fáziskönyvtárban ---------- 291

5.5 Az operációs rendszer egyéb szolgáltatásai ---------- 292
5.5.1 AUTOTEST. Programok belövése ---------- 292
5.5.2 Rendezőprogramok ---------- 293
5.5.3 Utility programok ---------- 294

5.6 Az operációs rendszer generálása ---------- 295

5.7 Az ESZR/OS operációs rendszerről ---------- 298
5.7.1 Az ESZR/OS felépitése ---------- 298
5.7.2 A programok feldolgozása ---------- 300
5.7.3 A programok szerkezete. Dinamikus programbetöltés ---------- 301
5.7.4 Az ujrabelépő programok fogalma és használata ---------- 302
5.7.5 A vezérlő program üzemmódjai ---------- 304

6. Számitógépes hálózatok ---------- 307
6.1 A távolról történő feldolgozások kifejlődése, felhasználási területek ------ 307
6.2 A felhasználók karakterisztikája ---------- 308
6.3 A számitógépes adatátvitel alapjai ---------- 310
6.3.1 Hibakorlátozás ---------- 311
6.3.2 Analóg és digitális átviteli eljárások ---------- 312
6.3.3 Az adatátvitel vezérlési eljárásai ---------- 313
6.3.4 Az adatcsatornák kihasználása ---------- 314
6.4 Számitógépes hálózatok tipusai ---------- 316
6.4.1 Hagyományos számitógépes hálózatok ---------- 316
6.4.2 A csomagkapcsolás elvén müködő hálózatok ---------- 318
6.4.3 A csomagrádiózáson alapuló számitógépes hálózatok ---------- 320

IRODALOM ---------- 323

Contents

1. INTRODUCTION

2. PRINCIPLES OF COMPUTER DATA REPRESENTATION

3. COMPUTER MAINFRAME

4. PERIPHERAL EQUIPMENTS

5. OPERATING SYSTEMS

6. COMPUTER NETWORKS

Előszó

Ez a könyv elsősorban a KSH Nemzetközi Számitástechnikai Oktató és Tájékoztató Központ szakképzettséget nyujtó (Rendszerszervező, Folyamatszervező, Számitógép Programozó, Számitógép Müszaki és Számitógép Operátor) tanfolyamainak hallgatói számára készült.

Azon tul, hogy a hetvenkét órás Rendszertechnika tantárgy anyagát hivatott támogatni az elkövetkezendő néhány oktatási év során – a számitástechnikában nem jártas, de az iránt érdeklődő olvasóknak is segitséget nyujt a legfontosabb ismeretek megszerzéséhez.

Több éves oktatási tapasztalataink és a nagyobb ESZR modellekben megvalósitott elvek felhasználásával a könyv átfogó képet nyujt a számitógépekről.

Ebben a tankönyvben a számitógépet mint komplex automatát tárgyaljuk. Az olvasó rendszertechnikai szinten megismeri a számitógépek két alkotóelemét, a fizikai és az absztrakt gépet. Ez a tárgyalásmód ujszerü, várhatóan segit majd a számitógépek világában történő tájékozódásban. Az anyag, miután rendszertechnikai alapokat ad, nem foglalkozik az absztrakt gép konstruálásával, tehát a programozással. A fizikai gép részletes bemutatásán tul a fizikai és az absztrakt gép közötti interface (az operációs rendszer) ismertetésére is sor kerül.

Felhasználói szemlélettel és összefoglaló jelleggel tárgyaljuk a géptermen és annak közvetlen környezetén kivül, pontosabban távol, de azzal on-line kapcsolatban levő rendszereket is.

Ezuton mondok köszönetet a társszerzőknek és minden munkatársnak, akik munkájukkal lehetővé tették, hogy ez a tankönyv a szokásosnál rövidebb átfutási idő alatt megjelenhessen.

Budapest, 1976

A szerkesztő

1. Bevezetés

A számitástechnika olyan interdiszciplináris terület, amelynek jelentősége évről évre erőteljesen növekszik. Sok neves gondolkodó, feltaláló évszázadokkal ezelőtti kutatásai, sikerei és balsikerei ötvöződtek egybe az 1951-ben kereskedelmi forgalomba hozott első tároltprogram-vezérlésü számitógépben (UNIVAC). A fizikai munka könnyitését célzó egyszerü szerkezetek (emelők) mellett már mintegy kétezer évvel ezelőtt gondoltak a rutinszerü szellemi tevékenység támogatására is (ABACUS, "golyós számológép"). Az első számitógép megszületését az alábbi tudományos és technikai eredmények előzték meg:

1623. Schickard bemutatja az első mechanikus, fogaskerekekkel müködő összegező gépet (az összeadás és kivonás müveletének gépi megvalósitása).

1642. Pascal megvalósitja az automatikus átvitelképzést az egyes helyi értékek között, saját konstrukcióju fogaskerekekkel müködő mechanikus összeadó gépen.

1671. Leibniz olyan mechanikus összeadó és szorzó gépet szerkesztett, amelyben a szorzás müveletét ismételt összeadással valósitotta meg.

1679. Leibniz javasolta elsőként a kettes számrendszer alkalmazását az addig használatos tizes számrendszer helyett a mechanikus összeadó és szorzó gépekben.

1786. Muller bevezette a regiszterek közötti müveletvégzést.

1833. Babbage mechanikus számológépet konstruált, amely ha elkészült volna, 1000 db 50 helyi értékes számjeggyel, lyukkártyás vezérléssel egy összegzést egy másodperc alatt, egy szorzási müveletet pedig egy perc alatt tudott volna végrehajtani.

1847. Boole: "A logika matematikai analizise" cimü munkája megjelent, amelyet lényegében csak száz év mulva lehetett igazán hasznositani.

1889. Hollerith kidolgozta a lyukkártya mai formáját és a lyukkártyás adatfeldolgozás elvét. Az 1890-es népszámlálás adatainak feldolgozását "Hollerith gépeken" végezték el az USA-ban.

1936. Zuse bináris aritmetikai egységet épitett jelfogók felhasználásával.

1941. Zuse megépiti az első lebegőpontos aritmetikával rendelkező jelfogós külső programvezérlésü gépet.

1942-45. A második világháboru alatt kifejlesztették az első, elektronikus elemekből (elektroncsövek) felépitett külső programvezérlésü kalkulátorokat (MARK-1, ENIAC).
Ezek felhasználási területe a müszaki tudományos számitások gyors elvégzésére szoritkozott, pl. a lőelemképző táblázatok készitése. Az első elektroncsöves kalkulátorok müveleti sebességére jellemző a mintegy 5000 elemi müvelet/sec-os sebesség, amely jelentős ugrást hozott az elektromechanikus gépekhez képest.

1946. A magyar származásu Neumann János a világon elsőként javasolja a tároltprogram koncepció alkalmazását. Ezen elv alkalmazásának első példája a vezetésével épitett JONIAC. A gépnek külön táregysége volt, amelyben a programok és adatok egyaránt helyet kaptak. Ennek eredményeként - szemben az addig alkalmazott külső programvezérlési elvekkel - a programok és adatok könnyen megváltoztathatók voltak, egy adott program másik program segitségével átirathatóvá vált, sőt egy program önmagát is megváltoztathatta stb. Ez a gondolat tette lehetővé, hogy a következő évtized során a gépi kódu programozás szintje fölé autokódok, assemblerek és egy magasszintü, müszaki-tudományos célra kifejlesztett nyelv (FORTRAN) épülhessen. Mondhatjuk, hogy a Neumann-féle alapelvek ma is meghatározó jelentőségüek a legkorszerübb óriás- és mikrogépek fejlesztésében egyaránt.

Az ötvenes évek második felében kialakultak a világ legnagyobb számitógépgyártói (IBM, UNIVAC, CDC, Honeywell, Borroughs stb.). Erre az időszakra esik a félvezetők (tranzisztorok, diódák) alkalmazásának robbanásszerü elterjedése. Ez lökésszerüen megnövelte az üzembiztos, megbizható gépek tömeggyártását és jelentős teljesitménynövekedést, illetve méretcsökkentést váltott ki.

Az elektroncsövek alkalmazása (10-20 ezer db egy gépen belül) nem tette lehetővé üzembiztos gépek konstruálását, mert a csövek élettartama legfeljebb néhány ezer óra volt, és ez sorozatos meghibásodáshoz vezetett. Ezzel szemben a félvezetők élettartama üzemi körülmények között szinte végtelen. Az egyre gyorsabb és nagyobb teljesitményü gépek kihasználásának fokozására a gyártók nemcsak a hardware-t (az egyes gépek látható, megérinthető és fizikai formában létező berendezéseit, komponenseit) és a néhány programnyelv használatához szükséges forditóprogramokat adták a felhasználónak, hanem a számitógépek optimális üzemét megszervező monitorokat, később operációs rendszereket is. Igy a számitógépek valóban komplex automatákká válnak, hiszen a fizikailag eszközként létező berendezések munkájának aktuális szervezését egy absztrakt komponens (több programból álló összetett szellemi termék), az operációs rendszer végzi. Összefoglalva, a hardware-re ráépül a software (az operációs rendszereket, sőt még általánosabban a programok halmazát nevezzük igy) és a két szint funkcionális elemei együttesen alkotják az elektronikus digitális számitógépeket.

Az ötvenes évek végén és a hatvanas évek elején elterjedt magasszintü nyelvek (FORTRAN, ALGOL, COBOL) lehetővé tették a számitógépek széleskörü alkalmazását a müszaki-tudományos és az adatfeldolgozási területeken egyaránt. Uj tipusu géphasználat jött létre: a programozók, akik a kötegelt feldolgozás térhóditása miatt a géptermekből kiszorultak,az időosztásos rendszerekben ujra gépközelbe kerültek az interaktiv (párbeszéd jellegü) terminálok használatán keresztül.

A hatvanas évek első felében megjelentek az integrált áramköri elemek, ez ismét nagy előrelépést jelentett. A hagyományos tárkonstrukcióhoz képest egy nagyságrend sebességnövekedést lehetett elérni.

Érdemes megjegyezni, hogy már 1964-ben a CDC óriásgépei hagyományos (ferritgyürüs) tárral, de jó szervezéssel harminc millió elemi müvelet/sec elvégzésére voltak képesek. A gyors fejlődés egyik jellemzője, hogy az egyes géptipusok, konstrukciók néhány év alatt teljesen elavultak. Ez a tendencia ma is érvényesül, sőt bizonyos irányokban gyorsul.

1963-ban az IBM bejelentette a 360-as sorozat inditását. Az IBM óriási sulyával (a világon gyártott gépek több mint hatvan százalékát még ma is az IBM gyártja) a kisebb és nem önálló utat járó gyártókat saját - elsősorban adatfeldolgozási célokra konstruált - gépeihez hasonló fejlesztési irányokra késztette. Az IBM produktumaként született a PL/I nyelv, amely önmagában próbálja egyesiteni a müszaki - tudományos és adatfeldolgozás orientált nyelvek előnyeit.

A hatvanas évek második felében egyre nőttek az igények a gépek széleskörü felhasználását illetően. Nagyszámu felhasználó igényelte egyidőben problémáinak lehetőleg azonnali megoldását. Multiprogramozásu és több processzoros gépeket konstruáltak és ezekkel párhuzamosan számitógépes hálózatok létrehozására törekedtek. A hálózatokkal lehetővé vált a legkülönbözőbb távoli helyről érkező igények kielégitése is. A felhasználók (pl. biztositó társaságok, bankok, trösztök stb.) hivatali helyiségeibe a kötegelt távoli és az interaktiv tipusu feldolgozást lehetővé tevő terminálok kerültek. Az itt dolgozó felhasználók ugy érezték, hogy rendelkezésükre áll egy teljes számitógép, sőt talán csak velük foglalkozik. 1968-ban mutatták be kisérleti jelleggel a világ első csomagkapcsolt számitógépes hálózatát. Talán elsősorban a hálózatok fejlesztésével kapcsolatban megnőtt a miniszámitógépek jelentősége, uj tőkeerős gyártók kerültek az élvonalba (pl. DEC).

A hetvenes évek elején az IBM megjelent a 370-es sorozattal, amely a 360-as család továbbfejlesztésének tekinthető.

A szocialista országok összefogásaként kiemelkedő jelentőségü közös fejlesztési program indult, amely az IBM 360/370-es család megfelelő tagjaival software kompatibilis gépcsalád létrehozását tüzte ki célul. Az ESZR család legkisebb tagját,az R 10-et hazánk gyártja. Ezt a gépet elsősorban ipari folyamatszabályozási célokra tervezték, de a nagyobb R modellek mellett front-end funkciók megvalósitására, valamint csomóponti processzorként is alkalmazzák. Az R10 nem kompatibilis a nagyobb modellekkel, sőt még a továbbfejlesztett R 12 sem. Az ESZR-sorozat nagyobb modelljei közül az R20-as gépek üzemelnek legnagyobb számban hazánkban. Müveleti sebességük 20 ezer müvelet/sec. Ezt a modellt ESZ/DOS operációs rendszerrel szállitja a gyártó. Maximális tárkapacitása, 256 Kbyte. Megfelelő adatátviteli vezérlő berendezésekkel már alkalmas on-line távfeldolgozó rendszer kiépitésére is. Jelenleg már létezik a gyorsabb változata is R22 néven, azonban Magyarországon még nem installáltak ilyen modellt. Az R 30 a közepes teljesitményü ESZR gépek közé tartozik. Müködési sebessége 100 ezer müvelet/sec. Alkalmas többprocesszoros rendszerek kiépitésére is. Maximális tárkapacitása 512 Kbyte. Megfelelő konfigurációhoz már az OS/ESZ operációs rendszer alkalmazható, amely jelentősen kiszélesiti a számitógép alkalmazási lehetőségeit. Ma már létezik az R 32-es és R 33-as,az R 30 továbbfejlesztett változata. Az R 40 összetett müszaki-gazdasági problémák megoldására szolgáló közepes teljesitményü tagja a családnak. Átlagos müveletvégzési sebessége 320 ezer müvelet/sec. Az R 40 konfigurációja a felhasználási területtől függően változhat. Maximális tárkapacitása 1024 Kbyte. Az ESZ/OS operációs rendszer használata ajánlott. Az R50 az ESZR gépek első sorozatának legnagyobb tagja. Átlagosan félmillió müveletet hajt végre másodpercenként. Maximális tárkapacitása 1024 Kbyte. ESZ/OS operációs rendszerrel fut.

Az első sorozat tagjai az IBM 360-as család megfelelő modelljeivel software kompatibilisek. A hardware döntő része integrált áramkörökből épül fel, sőt ezen tulmenően a változtatható tartalmu "csak olvasható tárak" alkalmazásával lehetőség van a fizikai gép architekturájának bizonyos foku megváltoztatására is.

Intenziv ütemben folyik az ESZR-II. család fejlesztése, amelynek tagjai az IBM/370-es sorozat megfelelő modelljeivel lesznek software kompatibilisek. A második sorozatban a fejlesztés alatt álló modellek tipusszámai a növekvő teljesitmények sorrendjében az alábbiak:

R15, R25, R35, R45, R55, R60 és R65.

A legkisebb modellt, az R15-öt hazánk fogja gyártani. Az ESZR-II. sorozaton belül alulról felfelé is megvalósul a teljes software kompatibilitás, tehát pl. egy R15-ön futó program minden további nélkül lefuttatható R25-ön, vagy bármelyik nagyobb gépen is.

Jelenleg hazánkban az ESZR gépek száma valamivel több mint 25 %-ot tesz ki a teljes számitógép állományból. A negyedik ötéves tervben indult meg a hazai gyártás felfuttatásával egyidőben a nagyobb modellek importja is. A tervek szerint 1980-ra mintegy 150 darab hazai gyártmányu számitógép és körülbelül ugyanennyi nagyobb modell üzembe állitásával lehet számolnunk.

Az alkalmazási területek bővitésére rendelkezésre áll néhány IBM programcsomag, az AIR keretében pedig több mint harminc programcsomag került elfogadásra a KGST országok között. Ezek a programcsomagok térítésmentesen, cseremegállapodás keretében válnak hozzáférhetővé. Ugyancsak hozzáférhető több hazai fejlesztésü programcsomag, amelyek központilag támogatott kutatás-fejlesztés eredményeként születtek. Sor kerül néhány programcsomag nyugati software-gyártó cégektől történő megvásárlása is.

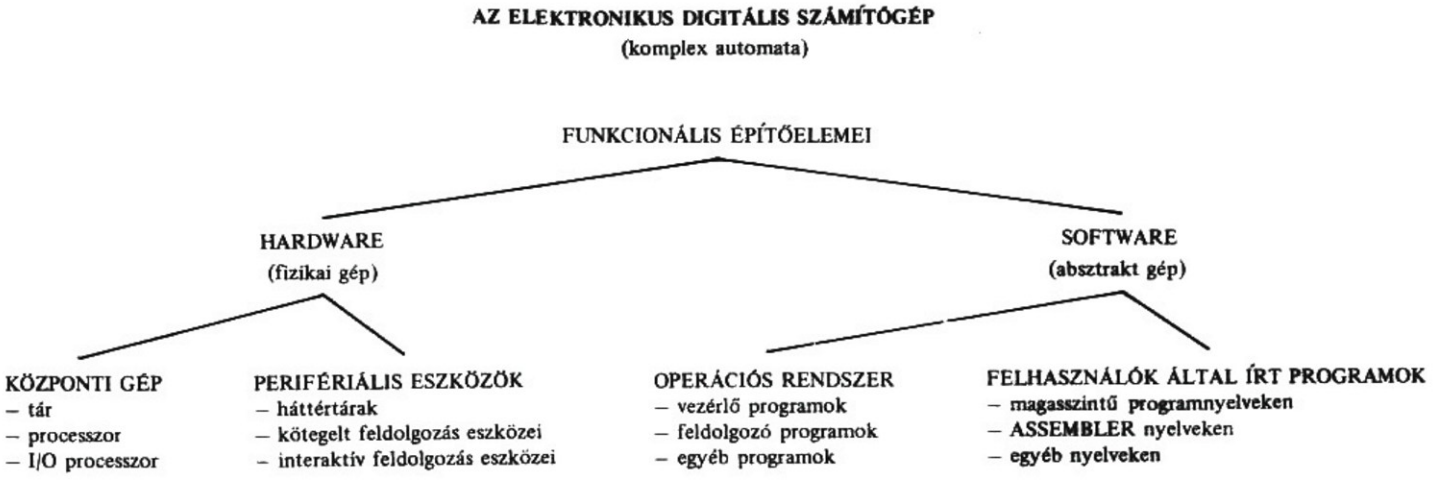
A hazai számitástechnikai ipar termékstrukturája az 1976-80 közötti időszakban az elmult ötéves tervhez képest módosulni fog. Az össztermelési érték mintegy huszmilliárd forintot tesz majd ki. A gyártás jelenleg is már három területet ölel fel. Kisszámitógépgyártás, perifériagyártás, távadatfeldolgozási eszközök gyártása. Az első két termékcsoporton belül a kisszámitógépek részaránya növekedni fog, a távadatfeldolgozási eszközök gyártása is a jelenlegi részarány négy-ötszörösére emelkedik.

Az elmult néhány év során felgyorsult technológiai fejlődés eredményeként megjelent mikroprocesszorok rendkivüli jelentőségü változást hoznak a számitástechnika széleskörü alkalmazása terén. A világsikert aratott kisszámitógépek (elsősorban a DEC PDP sorozat) architekturáját követve egyetlenegy, néhány négyzetcentiméter alapterületü tokban - a tömeggyártás eredményeként - már husz-ötven dollárért megvásárolhatók a különböző cégek által piacra dobott nyolc bites mikroprocesszorok. Ezzel párhuzamosan már léteznek a programozható input/output processzorok ugyancsak egy áramköri tokban. A magas integráltsági foku tárak (néhány Kbyte) szintén egy-egy tokban foglalnak helyet. Ezek felhasználásával és természetesen további szükséges áramköri elemek (integrált áramkörös kapuk, tároló elemek, tápegység stb...) alkalmazásával mintegy három-ötszáz dollárért akár a felhasználó is konstruálhat jól funkcionáló mikroszámitógépet. A legelterjedtebb mikroprocesszorokból épithető mikrógépek tárkapacitása elérheti a 64 Kbyte-ot, átlagosan egy müvelet elvégzéséhez néhány mikroszekundumra van szükségük. Széleskörben alkalmazzák az ipari berendezésekben, programozható terminálvezérlő egységekben, adatátviteli berendezésekben, automatikus irányitó rendszerekben, elosztott intelligenciáju számitógépes hálózatokban stb. Az egyszerübb változatokat várhatóan fel fogják használni a

gépkocsiktól kezdve még a bonyolultabb háztartási gépekben is. A software fejlesztés, a programok belövése hagyományos számitógépeken cross-assemblerek segitségével történik. A "virtuális mikroprocesszor koncepció" bevezetésével könnyen megoldható a különböző cégek által gyártott azonos kategóriáju mikroprocesszorok közötti software-kompatibilitás. Az uj eszközök megjelenése "egyenszilárdságu" szakembereket igényel és termel ki a hardware-software problémák összefonódásán keresztül.

Előzetes tervek szerint az ESZR távlati fejlesztések során az ESZR-III. sorozat gépeiben már megjelennek a mikroprocesszorok, funkcionális épitőelemei lesznek a nagygépeknek is.

A ma használatos terminológiát alkalmazva az első ábrán látható az elektronikus digitális számitógépek hierarchikus felépitése.



1. ábra. Elektronikus digitális számitógépek funkcionális felépitése

A fizikai gépen belül a processzorok rendelkeznek utasitáskészlettel, igy csupán ezek az eszközök képesek a tárolt programok lefuttatására. Ez a kép várhatóan megváltozik a mikroprocesszorok széles körü alkalmazásával, hiszen a fizikai gép minden épitőeleme processzor szintre emelkedhet. Ez a tény többek között fel fogja gyorsitani a számitógépes hálózatok kialakitását még a kevésbé fejlett országokban is.

Néhány mondatban a következőképpen fogalmazható meg az absztrakt és fizikai gép egymásrautaltsága.

A felhasználók egy-egy adott, valós probléma megoldására absztrakt gépeket konstruálnak. Egy-egy absztrakt gép konstruálásához minimálisan az alábbiakra van szükség:

- a probléma megoldásához szükséges módszerek, eljárások, adatok, eredmények ismerete,
- a feladat megoldásának leirására szolgáló programnyelv ismerete,
- a fizikai gép használatát lehetővé tevő operációs rendszer felhasználói szintü ismerete.

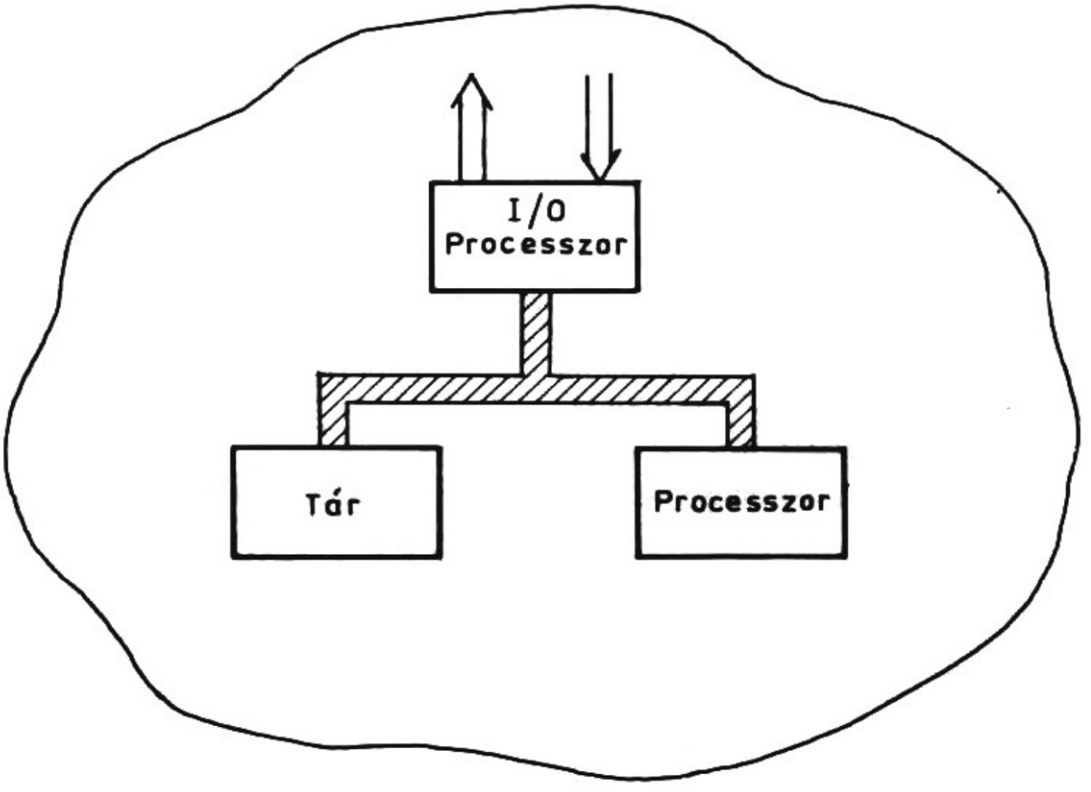
Itt érdemes megjegyezni, hogy általában egy felhasználónak nem kell szükségszerüen ismerni a fizikai gépet, hiszen az operációs rendszer megvalósitja az absztrakt és fizikai gép közötti kapcsolatot. Az absztrakt gép tehát a felhasználó által az adott terület ismerete alapján valamilyen probléma megoldására irt programból, adatokból, az operációs rendszer számára irt parancsokból és magából az operációs rendszerből áll. A program, az adatok, valamint az operációs rendszer számára szóló pa-

rancsok valamilyen adathordozóra kerülnek, leggyakrabban lyukkártyára. A fizikai gép perifériális berendezésein és az I/O processzoron keresztül az aktuális absztrakt gépnek a felhasználó által definiált komponense háttértárra kerül. Az operációs rendszer - megfelelő moduljainak aktivizálásával - értelmezi a felhasználó által leirt eljárásokat, adatokat a fizikai gép számára. Egyuttal gondoskodik a futtatásról, amely több felhasználó esetén általában szimultán jelleggel történik, igy lehetővé válik az egyetlen fizikai processzor jobb kihasználása, A program futása után az eredmények a tárból a háttértárra kerülnek, majd az I/O processzoron, és valamilyen perifériális berendezésen keresztül egy adathordozón (pl. sornyomtató papir) megjelennek az absztrakt gép alkotója előtt. Ezen tulmenően az operációs rendszer a felhasználó számára könnyen érthető formában tájékoztatást ad a futási időről, a program helyfoglalásáról és más, a fizikai gépre vonatkozó eseményekről is.

Az absztrakt gép létezése és használata rendkivül nagy jelentőségü. Jellemző, hogy manapság egy-egy számitógép árának (a közepes és nagyobb gépek ára egy-két millió dollár körül mozog) több mint felét az operációs rendszer képezi. Ez az arány a software javára tovább növekszik, különös tekintettel a mikroprocesszorokra épitendő sokprocesszoros gépekre, ugyanakkor viszont a tömeggyártásban levő különböző "csak olvasható" tipusu tárak lehetővé teszik, hogy a monitorok és operációs rendszerek funkcióinak egyre nagyobb része a fizikai gép részévé váljék.

1.1 A LEEGYSZERŰSÍTETT GÉPI MODELLEK

Az elektronikus digitális számitógépekkel történő ismerkedést célszerü a központi géppel kezdeni, hiszen a fizikai szintü gépi müködés alapelveibe ezen keresztül láthatunk bele. A központi gép modelljét a második ábra mutatja.



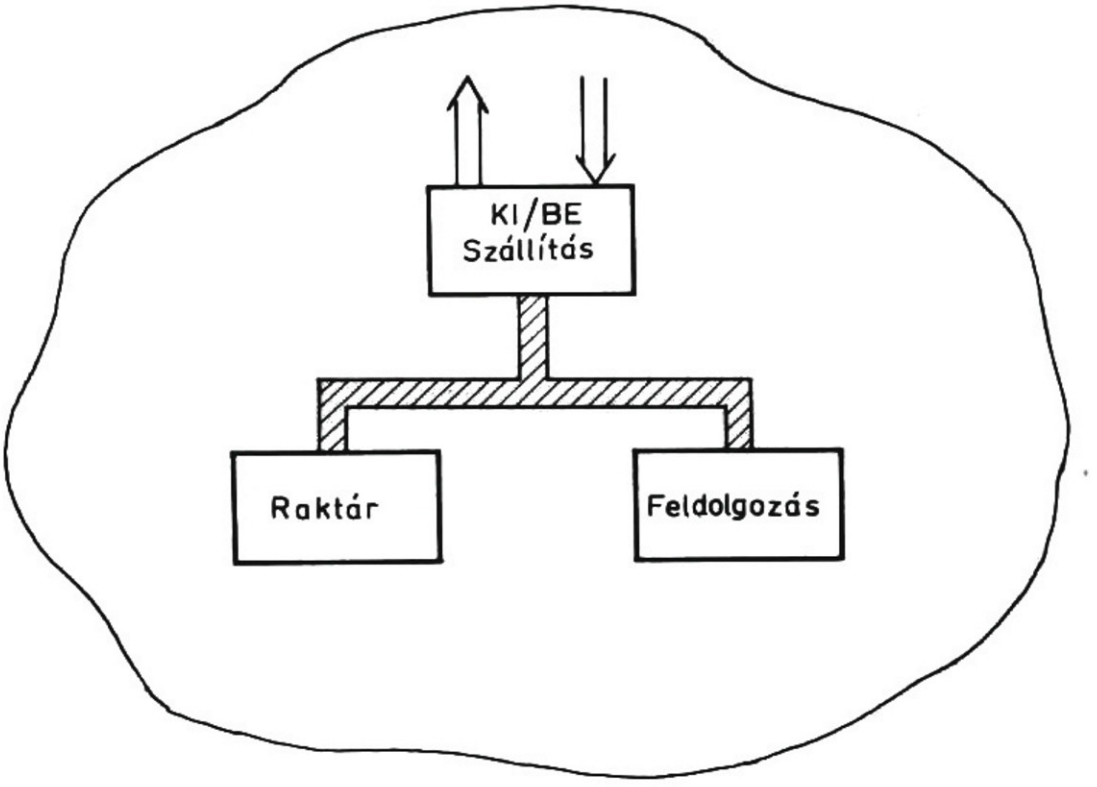
2. ábra. A központi gép modellje

Mielőtt azonban az I/O processzor - tár - processzor összjátékot elvi szinten elemeznénk, általában hasznos, ha egy az élet valamilyen területéről vett leegyszerüsitett hasonlattal elősegitjük a jobb megértést. A harmadik ábrán egy kisüzem modelljét látjuk.

A hasonlóság szembetünő, sőt az egyes üzemegységek és a központi gép megfelelő egységeinek funkciója is nagy vonalakban azonos jelleget mutat. A szállitási részleg feladata a nyersáru raktárba szállitása és természetesen a termelési folyamat során létrejött késztermék elszállitása a raktárból a külső felhasználókhoz. A raktár funkciója egyértelmü, de a termelési folyamat során tar ılma dinamikusan változik. Ez azt jelenti, hogy pl. óráról órára változó a nyersáru és a késztermék által lefoglalt, és a fel nem használt terület a raktáron belül. Az természetes, hogy a raktár lehető legjobb kihasználására törekszünk, de igy is előfordulhat, hogy vagy tul nagy a fel nem használt terület, vagy esetleg a termelőegységen kivül az egyenletesebb ütemü termelés biztositása érdekében további raktárakat (háttérraktárakat) kell létesiteni. A raktárak általában a késztermékek és a nyersáruk halmazán tul más eszközök, eljárások (pl. gyártási dokumentáció) tárolását is végezhetik. A feldolgozás, fizikai szintü termelőmunka a műhelyben folyik. Ez a legfontosabb szervezeti egység ebben a modellben, igy ehhez rendeljük hozzá a mindhárom egység összehangolt irányitását végző főnököt.

A termelés leegyszerüsitett folyamata ugy történik, hogy a főnök magához kéreti az adott termék gyártására vonatkozó dokumentációt (eljárások, parancsok, utasitások) és azt lépésenként értelmezve végrehajtatja a műhellyel ugy, hogy közben szükség szerint utasitja a raktárat a nyersáruk műhelybe szállitására, a késztermékek fogadására, valamint a raktárkészlet változásainak alapján feladatokkal látja el a szállitási részleget is.

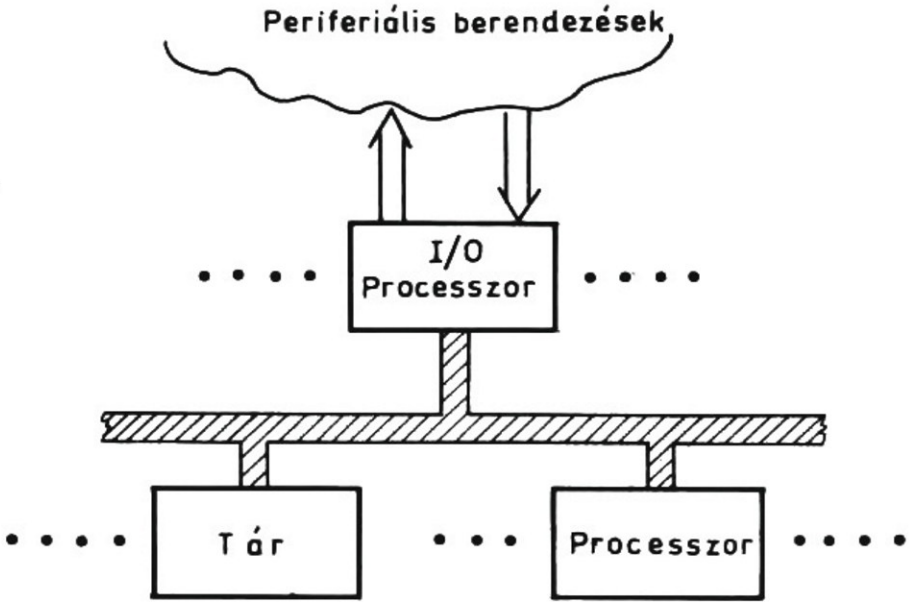
A központi gép modelljében is a feldolgozó egységet két funkcionális részre osztjuk, nevezetesen az aritmetikai - logikai egységre (a műhely), valamint a vezérlőegységre (főnök).



3. ábra. Egy kisüzem modellje

A fizikai gép működésének leegyszerüsitett folyamata: tételezzük fel, hogy egy adott problémát megoldó program (utasitások sorozata) futásra kész állapotban a tárban van a hozzátartozó adatokkal együtt. A vezérlőegység lehivja a tárból a program első utasitását. Értelmezi (dekódolja) az illető parancsot, majd az utasitás tartalmától függően, szükség szerint vezérli az aritmetikai és logikai egységet és a tárat az illető utasitásban foglaltak végrehajtása céljából (pl. egy adat átvitele a tárból az aritmetikai és logikai egységbe). A program utasitásainak végrehajtása az "állj" parancs végrehajtásával fejeződik be.

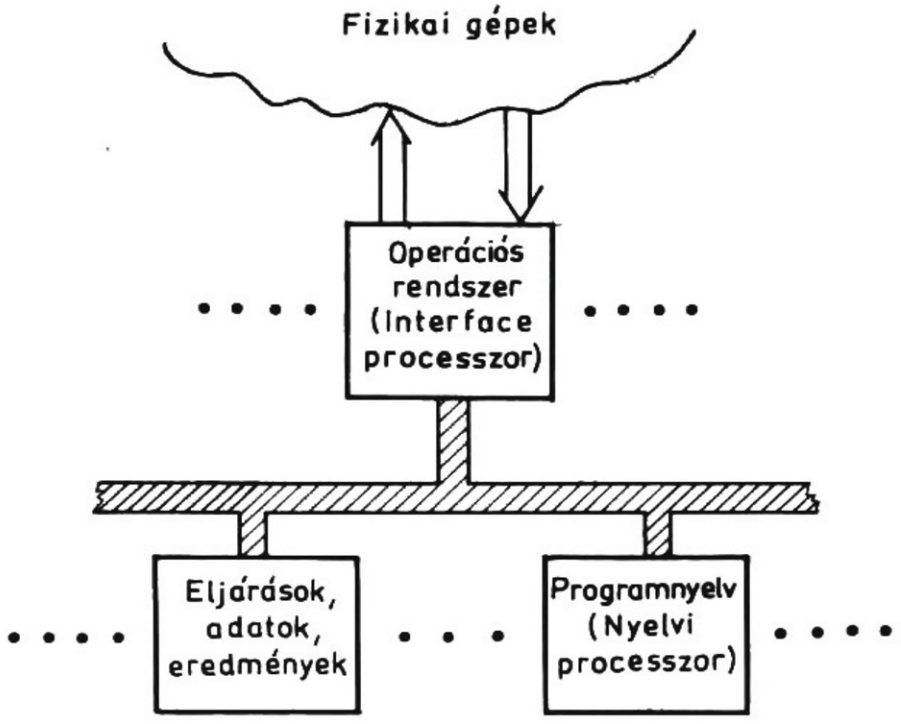
A központi gép modelljének általánosabb értelmezése: a számitógépek rendelkezhetnek több processzorral, több I/O processzorral és több táregységgel is.
Ennek megfelelően az általánositott modell a 4. ábrán látható. Ezt az architekturális képet mutatják az óriásgépek és a mikroszámitógépek egyaránt.



4. ábra. A központi gép általánositott modellje

Az absztrakt gép modelljét, sőt általánositott módositott modelljét az 5. ábrán láthatjuk.

Az absztrakt gépnek létezik egy állandó és egy változó komponense. Az állandó komponens az operációs rendszer, amelyet ugyancsak programozók (hardware ismeretekkel rendelkező rendszerprogramozók) konstruáltak több tiz, esetleg száz emberévnyi munkával. Az operációs rendszer képezi a kapcsot az adott fizikai gép és a felhasználók között. Igy egy-egy fizikai gépre, vagy gépcsaládra kifejlesztett operációs rendszer csakis az adott paraméterekkel rendelkező fizikai gépre, vagy gépcsaládra installálható. Az operációs rendszerek elsősorban a programmegszakitási rendszer által nyujtott információk elemzésén keresztül tájékozódnak a fizikai gépben zajló események normális vagy abnormális alakulásáról. Az adott müszaki paraméterekkel rendelkező fizikai gép munkáját ütemezi, optimalizálja az operációs rendszer, a fizikai gép igénybevételével megoldja a különböző programnyelvek szintje és a gépi kód közötti transzformációt, gondoskodik a futtatásról, a vezérkártyákon keresztül kapcsolatot tart a felhasználókkal stb.



5. ábra. Az absztrakt gép általánositott modellje

Az absztrakt gép változó komponensét a felhasználó konstruálja. Vegyünk például egy programozót, aki vagy rendelkezik megfelelő ismeretekkel egy adott probléma megoldásához, vagy pedig az adott probléma megoldásához szükséges eljárásokat, módszereket és adatokat egy magasabb szintről,a rendszerszervezőktől készen kapja. Ezek után az a feladata, hogy az eljárások,módszerek és adatok birtokában egy problémaorientált nyelven (pl. adatfeldolgozási feladat esetén COBOL) leirja (megkonstruálja) a feladat megoldását és ellássa az igy elkészült programot az operációs rendszer számára lényeges parancsokkal. Tehát a programozás nem más, mint egy-egy aktuális absztrakt gép konstruálása. Egy-egy absztrakt gép konstruálása a feladat összetettségétől függően néhány perctől kezdve akár néhány emberév munkát is igényelhet. A programozó féléves-egyéves munkája eredményeként elkészit egy hibátlan, belőtt programot, ugyanakkor a fizikai gép számára ez lehetséges, hogy csupán néhány perc vagy óra terhelést jelent, annak teljesitményétől függően. A felhasználó által konstruált absztrakt gép nem tiszavirág életü. Megmarad kártyaköteg formájában, vagy például mágnesszalagon, vagy lemezen tárolva, bármikor ujra futtatható és módositható. Igy a számitógépek felhasználási területe a különböző és ujabb absztrakt gépkonstrukciókon keresztül széles határok között bővithető a fizikai géptől függetlenül is.

Az általánositott modell azt mutatja, hogy egy-egy gép többféle operációs rendszerrel is üzemelhet (pl. ESZ/DOS és ESZ/OS), egy operációs rendszer több gépre is installálható (pl. az ESZ/DOS R20-as, R22-es, R30-as stb. gépekre), valamint az eltérő müszaki paraméterekkel rendelkező fizikai gépek üzemeltetéséhez különböző operációs rendszerekre van szükség.

1.2 A JEGYZET HASZNÁLATA

Az előszóban emlitett koncepciónak megfelelően ez a jegyzet a fizikai gépet tárgyalja részletesen, valamint a fizikai és absztrakt gép közötti interface-t a fizikai gép oldaláról megközelitve. Nem tárgyaljuk az absztrakt gép konstruálásának egyetlen szintjét sem, tehát nem esik szó a programozásról. Az absztrakt gépre vonatkozó ismeretek a jelenleg már kiadás alatt levő PROGRAMOZÁS ALAPJAI cimü jegyzetben találhatók majd meg.

Mind a fizikai, mind az absztrakt gép müködésének megértéséhez elengedhetetlenül szükséges néhány alapvetően fontos témakör előismerete. A második fejezetben megtalálható az adatszintek leirása, a számitástechnikában használatos számrendszerek tárgyalása, a számrendszerek közötti konvertálás, az aritmetikai és logikai müveletek, valamint a számábrázolás és az adatábrázolás. Az itt felsorolt előismeretek megszerzéséhez elegendő a középiskolában tanult matematika egy része, továbbá elengedhetetlenül szükséges az információegység fogalmának ismerete, az utóbbi azonban e néhány sor olvasásával megszerezhető. Az információ egysége a bit. Egy bit információ keletkezik bármely kétállapotu esemény bekövetkezésekor. Keressünk egy kézenfekvő kétállapotu eseményt. Az egyszerüség kedvéért vegyünk a kezünkbe egy kétforintost. Zárjuk ki azt a lehetőséget, hogy egy feldobott kétforintos az élére is eshet. Most dobjuk fel a kétforintost, a kapott eredmény kizárólag fej vagy irás lehet. Igy bekövetkezett egy kétállapotu esemény és a kapott eredmény információtartalma egy bit. Egy kis általánositással mondjuk azt, hogyha az eredmény fej, akkor ezt ugy irjuk le egy papirlapra hogy "nem", amenynyiben az eredmény irás, irjuk le azt, hogy "igen". Ezután dobjunk fel több kétforintost és a kapott eredményeket irjuk egymás mellé a következő röviditések bevezetésével: fej = F, irás = I, valamint nem = n, igen = i. Legyen a kapott eredmény például az alábbi:

FIFIFFII

n in in n ii

A továbbiakban vezessünk be egy ujabb jelölést is, legyen a nem = 0, az igen = 1. Az előbbi eredményt irjuk fel ebben a formában:

01010011

Látható, hogy a több elemi kétállapotu esemény bekövetkezéséből kapott egy-egy bit információ együttesen egy bitkombinációt alkot, ráadásul ez a bitkombináció egy számjegy formájában leirható.

A fizikai gépek általában nyolc, tizenhat és harminckét bitből álló szavakat kezelnek. A szóhosszuságot a bitek száma határozza meg. A különböző tipusu fizikai gépek szóhossza általában nem azonos. A nyolc bit hosszuságu szót elterjedten byte-nak (ejtsd: bájt) nevezzük.

A harmadik fejezetben a központi gép tárgyalását a tárral kezdjük. Ismertetjük a ferritgyürüs tár felépitését és müködését. Összehasonlitásra kerül a ferritgyürüs és az elektronikus tárolási elv. Az ESZR sorozat R20-as gépén keresztül tárgyaljuk a tár felosztását, a tár részeit és a tárvédelem megvalósitását.

A processzoron belül először az aritmetikai-logikai egység tárgyalására kerül sor. Látni fogja a kedves olvasó, hogy az aritmetikai müveletek számrendszertől függetlenek. Ebből következik, hogy a négy alapmüveletet bárki el tudja végezni tetszőleges alapu számrendszerben. További jelentős tény, hogy a négy alapmüvelet bármelyike visszavezethető az összegzés és komplementálás müveletére.

A processzor másik modulja a vezérlőegység. Tárgyalásra kerül a mikroprogramozott és a hagyományos vezérlési mód. Ismertetésre kerül az ESZR R20-as gép mikroprogram rendszere. A gépi kódu program futásán keresztül bemutatjuk a processzor és a tár közötti leegyszerüsitett összjátékot. Itt ismerheti meg a kedves olvasó a fizikai gép automatikus müködésének alapelvét. A programmegszakitási rendszer ismerete ugyancsak rendkivül jelentős. Az I/O processzor (Input/Output = Bemeneti/Kimeneti) feladata a központi gép és a perifériális egységek közötti kapcsolat megvalósitása.

Az I/O processzort más néven csatornarendszernek nevezzük. A központi gép a külvilággal a szelektor és multiplextipusu csatornákon keresztül érintkezik. Ismertetjük ugyancsak az ESZR R20-as gép alapján az I/O müveleteket vezérlő információ szerkezetét. Igy tárgyalásra kerül a csatornacimszó, a csatornaparancsszó és a csatornaállapotszó felépitése, valamint az I/O müveletek fizikai szintü lefolyása.

A harmadik fejezeten belül végül néhány oldalon összefoglaltuk a vezérlőpult funkcióinak ismertetését is.

A negyedik fejezet tárgyalja a perifériális berendezések nagy családját. Először a kötegelt feldolgozási mód, ezen keresztül a legnépszerübb két adathordozó,a lyukkártya és a lyukszalag kerül ismertetésre. Sorrendben ezután a kötegelt feldolgozás input eszközei következnek, tehát a lyukkártyaolvasó, lyukszalagolvasó és a közvetlen bizonylatolvasásra alkalmas berendezések. A kötegelt feldolgozás output eszközeinek tárgyalását a sornyomtatóval kezdtük, majd lyukkártyalyukasztók, lyukszalaglyukasztók és a rajzgépek kerültek sorra. Ujabb csoportot képeznek a perifériális egységek táborában a háttértárak. A "Háttértárak" cimü alfejezetben ismertetésre kerülnek a mágnesszalagos, mágneslemezes, mágnesdobos és mágneskártyás tárak. Végül a perifériális egységek sorát az interaktiv (dialógus tipusu) feldolgozás eszközei zárják. Ezek között tárgyalásra kerülnek a különböző konstrukcióju karakternyomtató és display terminálok.

Az ötödik fejezetben először az operációs rendszer fejlődését, majd pedig felépitését ismerheti meg a kedves olvasó. Sorra kerül a vezérlőprogramok, a feldolgozóprogramok és egyéb programok funkcióinak tárgyalása. Ezeken keresztül megérthető az absztrakt gép és a fizikai gép közötti kapcsolat realizálása ESZ/DOS környezetben, majd az ESZ/OS operációs rendszerről adunk egy összefoglaló jellegü áttekintést.

A hatodik fejezet a számitógépes hálózatok témakört tárgyalja. Röviden ismertetésre kerülnek a számitógépes adatátvitel alapjai, a hálózattipusok, igy a hagyományos, a csomagkapcsolt és a csomagrádiózáson alapuló hálózatok.

2. A számítógépek adatábrázolásának alapjai

2.1 ADATSZINTEK

Induljunk ki egy példából! Képzeljünk el egy kimutatást, ami egy megye személygépkocsiállományát tartalmazza! Más szóval egy táblázatot, amelynek minden sora más és más rendszámu személygépkocsiról ad felvilágositást. Egy-egy ilyen sorban szerepelni fog például az üzemben tartó (tulajdonos) neve, születési éve és helye, lakcime, a gépkocsi rendszáma, a gépkocsi márkája, tipusa, évjárata, hengerürtartalma, alvázszáma stb., stb.

Fal Kornél, 1946. Szombathely, Bp., Trafik u. 22., ZC 82-53, Lada, 1300, 1976, 1294 cm^3... .

Ebben a példában az adatoknak az összetartozásuk alapján különböző szintjeit lehet megfigyelni. Nézzük ezt végig!

Az egész kimutatás, táblázat, ami a feldolgozáshoz szükséges összes adatot tartalmazza, alkot egy file-t. A táblázat sorokból áll, mindegyikben egy-egy gépkocsihoz kapcsolódó különböző adatok vannak. Ezek az összetartozó adatok a logikai rekordok. Adott soron belül nevek, számok szerepelnek egymás után, az oszlopok (mezők) tartalmának megfelelően. Mindegyik jelent valamit, van valami értelme. Ezek az ugynevezett elemi adatok. Ezek még mindig tovább bonthatók kisebb részekre! A nevek, adatok betükből, számokból állnak, de ezeknek már önálló jelentésük nincs! Ezek a karakterek. Ilyen módon a feldolgozáshoz szükséges táblázatból egy logikai sorrend alapján eljutottunk egészen a legkisebb egységig, az irásjelig. Közben végighaladtunk az adatok különböző összetettségü szintjein. Ezek a következők voltak:

- karakter
- elemi adat (mező)
- logikai rekord
- file

Karakter: Az adatok legkisebb része. A jelkészletbe tartozó betü, számjegy vagy speciális jel ábrázolására szolgál. A számitógéppel kapcsolatban ez az ábrázolás jelsorozat segitségével egy byte-on (8 pozición) történik.

Elemi adat (mező): A feldolgozandó információ legkisebb önállóan is értelmes része. A lyukkártyás feldolgozással kapcsolatban gyakran "mezőnek" is hivják. Egy vagy több különféle karakterből áll. A karakterek fajtái alapján lehet numerikus vagy alfanumerikus.

- Numerikus (elemi) adat: csak számokat és esetleg előjelüket tartalmazza. Pl.: születési évszám, évjárat, tipus stb.
- Alfanumerikus (elemi) adat: betüket, számjegyeket és különleges karaktereket is tartalmazhat. Pl.: név, lakcim, rendszám stb.
- Ezek az elemi adatok alkotják a különböző müveletekben szereplő operandusokat.
- Egyes esetekben az adatmezők tartalma, az elemi adat is részekre bontható, pl. a lakcim: város, utca, házszám, emelet stb. Ezeket tételnek hivják.

Logikai rekord: elemi adatok összetartozó halmaza. Például egy gépkocsi nyilvántartásához szükséges összes adat. Lásd a bemutatott példában felsoroltakat!

File: Ez valamilyen rendezési elv, logikai szempont alapján összetartozó rekordok halmaza. Itt már a felhasználás szabja meg, hogy milyen szempont alapján történjen a rendezés. Például lehet
- a megye gépkocsijainak adatait a tulajdonosok neve alapján összegyüjteni és felsorolni;
- lehet a rendszám alapján sorrendbe rakni;
- lehet a hengerürtartalom alapján a különféle márkákat csoportositani stb.

A számitógép esetében is az adatoknak ezeket a "szintjeit" használjuk a tárolás, feldolgozás különböző fázisaiban.

De térjünk csak vissza egy kicsit még a karakterhez! Ez - amint már mondottuk - a betük, számok megjelenitésére, irásbeli kifejezésére szolgál. A számitógépnek is az információt a mindennapi gyakorlathoz hasonlóan elemi részekből kell összerakni, felépiteni. A gép viszont egy karakter kifejezéséhez, megjelenitéséhez nem tud csak egy jelet használni. Ahogy már a bevezetőben is emlitettük, ezt csak a kettes számrendszer segitségével képes megoldani. Itt minden egyes karakterhez egy bitsorozat (0-k és 1-esek sorozata) szükséges. Ilyen alapon tehát a gépben a karakternek, mint irásos szimbólumnak egy jelsorozat fog megfelelni. A gép ezekből a jelsorozatokból viszont mindig csak megadott hosszt (megadott darabszámu bitet) képes egyszerre felhasználni.

Az ily módon a gépben egységként kezelt információ az ugynevezett gépi szó vagy röviden csak szó. Ennek nagysága a gépekre jellemző adat. Általában 32 bit egy szó hossza, de vannak gépek, ahol ez 16, 24, sőt 48 bit is lehet.

2.2 SZÁMRENDSZEREK

2.2.1 10-ES (DECIMÁLIS) SZÁMRENDSZER. NÉHÁNY ALAPFOGALOM

Mindennapi munkánk során állandóan ezt használjuk, ennek ellenére nagyon sokszor csak automatikusan, a lényegét nem is tudatositva magunkban. Hogy az általunk használt számrendszer 10-es számrendszer, ez azt jelenti, hogy az alapszáma, idegen szóval radix*-a 10. A radix jele, a nagy vagy kis "r" betü, ezért ezt is irhatjuk: R(=r) = 10.

* radix (latin) = alap, bázis, gyökér, gyök
radix (angol) = alapszám

Egy számrendszer alapszámát a bármely adott helyi értéken kifejezhető, egymástól különböző értékek száma (a 0-t is beleértve) adja meg. A lehetséges különböző értékek a mi esetünkben a következők:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Mivel ezek bármelyike szerepelhet egy helyi értéken (pozición), az összesen 10 számjegy miatt hívják számrendszerünket decimális, vagyis 10-es számrendszernek. A számrendszer alapszámát a számot követő index jelzi, például: $3589_{(10)}$.

A decimális számrendszer annyira általános, hogy rendszerint nem tesszük ki a 10-es alapszámra utaló indexet, de mi a továbbiakban jelölni fogjuk, mivel különböző alapszámu számrendszereket fogunk használni.

Az eddig elmondottakból még két dolog következik. Az egyik, hogy alapszámról csak a helyi értékes számrendszerek esetében lehet beszélni, de ezeknél minden esetben!

A másik megjegyzés az egy adott helyi értéken előforduló számértékhez, az un. együtthatóhoz (a_k) kapcsolódik. Ez a decimális számrendszerben a 0 és a 9 közötti értéket veheti fel, vagyis rá az alábbi összefüggés irható fel:

$$0 \leq a_k \leq (R - 1)$$

Ebből az is következik, hogy "0" és "1" szimbólum (jegy) minden számrendszerben van! Egy számrendszer alapszámául (R) bármely 1-nél nagyobb pozitiv egész szám választható, és igy tetszőlegesen sok számrendszer képezhető. Ezek közül kell majd kiválasztani azt (vagy azokat), amelye(ke)t a digitális számitógépek a legkedvezőbben használhatnak.

Térjünk most vissza a decimális (10-es) számrendszer egy másik jellemzőjére, arra, hogy helyi értékes! Vegyük az alábbi számjegyeket:

3, 5, 8, 9

Ebből a négy számjegyből sokféleképpen lehet decimális számot képezni.

3589
9853
3895
8395
stb.

Az egyes számok értéke attól függően más és más, hogy egy adott pozición, "helyi értéken" melyik számjegy szerepel. A szám értékét ugyanis a következőképpen kapjuk meg:

a) Minden egyes pozicionak meghatározzuk (vagy már "fejből" tudjuk) az értékét, az un. helyi értéket. Ez 10-nek, vagyis az alapszámnak megfelelő hatványozásával nyerhető. Másképpen megfogalmazva, minden egyes pozició értéke a tőle jobbra álló pozició értékének alapszámszorosa!

... tizezres,	ezres,	százas,	tizes,	egyes,	tized,	század...
10000	1000	100	10	1	$\frac{1}{10}$	$\frac{1}{100}$
10^4	10^3	10^2	10^1	10^0	10^{-1}	10^{-2}

b) Az egyes helyi értékeket annyiszor vesszük, amennyit az adott pozicióra irt együttható mutat (a helyi értékeket szorozzuk az adott együtthatóval)

c) Az igy kapott értékeket (részletszorzatokat) összegezzük, összeadjuk.
Ennek alapján nézzünk egy példát!

$5\ 0\ 7\ 6\ ,\ 1\ 8_{(10)}$

5	0	7	6 ,	1	8
10^3	10^2	10^1	10^0	10^{-1}	10^{-2}

$$
\begin{aligned}
5 \cdot 10^3 &= 5 \cdot 1000 &&= 5000 \\
0 \cdot 10^2 &= 0 \cdot 100 &&= 0 \\
7 \cdot 10^1 &= 7 \cdot 10 &&= 70 \\
6 \cdot 10^0 &= 6 \cdot 1 &&= 6 \\
1 \cdot 10^{-1} &= 1 \cdot 0,1 &&= 0,1 \\
8 \cdot 10^{-2} &= 8 \cdot 0,01 &&= \underline{+\ 0,08} \\
& &&\ 5076,18
\end{aligned}
$$

Ez a módszer azonban nemcsak a tizes számrendszer esetén alkalmazható! Bármely számrendszerrel felirt szám értékét megkaphatjuk a tizes számrendszerben a fentiek alapján. Természetesen más számrendszer esetén az egyes helyi értékek meghatározásánál az illető számrendszer alapszámát kell hatványozni! Általános megfogalmazásban:

$$N = \pm \sum_{k=-m}^{j-1} a_k \cdot R^k =$$

$$= \pm (a_{j-1} \cdot R^{j-1} + a_{j-2} \cdot R^{j-2} + \ldots + a_1 \cdot R^1 + a_0 \cdot R^0 + a_{-1} \cdot R^{-1} + \ldots a_{-m} \cdot R^{-m})$$

ahol: N — egy tetszőleges, R alapszámu számrendszerbeli szám

R — radix, az illető számrendszer alapszáma

a_k — az egyes helyi értékeken szereplő együttható értéke

k — kitevő, az egyes helyi értékek meghatározásánál szükséges a hatványozáshoz

Σ — görög betü (ejtsd: szumma) az ismételt összeadás jele. Az összeadást mindazokra a tagokra el kell végezni, amelyek indexe a k szumma alatti értéke és k szumma feletti értéke között változik.

Vagyis egy tetszőleges szám egyenlő a radix hatványaival sulyozott együtthatók összegével.

A következő fogalom megértéséhez képzeljük el egy gépkocsi kilométeróráját (kilométer-számlálóját)! Ezen, az általánosan előforduló öt "ablak", öt pozició segitségével $00000_{(10)}$ értéktől $99\ 999_{(10)}$-ig jelenithetők meg a megtett kilométerek.

Ez összesen $100\ 000_{(10)}$ különböző értéket, számot jelent.
Egy eszköz modulusán (M) az eszközzel kifejezhető vagy megjelenithető összes, egymástól különböző mennyiség, állapot számát értjük. A modulust, mint az eszköz által használt számrendszer valamilyen kitevőre (n) emelt alapszámát (R) lehet megadni.

$$M = R^n$$

A kitevő (n) értékét az eszköz pozicióinak, helyi értékeinek darabszáma adja meg. Ez a gépkocsi kilométerórája esetében 5, mivel 5 pozició (ablak) mutatja a számjegyeket. A használt számrendszer alapszáma, R = 10, igy az eszköz (kilométeróra) modulusa:

$$M = R^n = 10^5 = 100\ 000$$

Az előbbi példában a számolás felső határa (az ábrázolható legnagyobb szám) $99\ 999_{(10)}$ volt. Ez eggyel kisebb, mint az eszköz modulusa (100 000), hiszen a megjelenithető állapotok számába beletartozott a $00\ 000_{(10)}$ alak is. Ez utóbbi megint minden számrendszer esetében igy van, vagyis az adott eszközzel ábrázolható legnagyobb egész szám (N_{max}) az eszköz modulusából (M) meghatározható:

$$N_{max} = M - 1 = R^n - 1$$

Megjegyzés: ez csak egész számok esetén van igy, mert ha az eszközzel törtjegyeket is ki akarunk fejezni, akkor $N_{max} = R^j - R^{-m}$
ahol: j - az egész jegyek ábrázolására szolgáló pozíciók darabszáma
m - a tört jegyek ábrázolására rendelkezésre álló pozíciók száma

Ilyen esetben az eszközzel ábrázolható legkisebb értékes szám (a számolás pontossági korlátja)

$$N_{min} = R^{-m}$$

A számitógépekben az egyes helyi értékeken levő számjegyek mindegyikét valamilyen fizikai mennyiség (leginkább feszültség) képviseli: a decimális számrendszer 10 különböző számjegyének igy 10 különböző feszültségszintet kellene stabilan megfeleltetni. Ez müszakilag roppant bonyolult feladat lenne, ezért a decimális számrendszer (általános elterjedtsége ellenére) nem használatos a számitógépekben.

2.2.2 2-ES (BINÁRIS) SZÁMRENDSZER

Elektromos áramköröknél a "bekapcsolt", illetőleg "kikapcsolt" állapotot megbizhatóan, stabilan lehet regisztrálni. Ezért az ilyen áramkörökből felépithető digitális számitógépeknél olyan számrendszert kellett keresni, amelynek csak két diszkrét számjegye van, hasonlóan az áramkör két állapotához. A megoldást a 2-es, más néven bináris számrendszer jelenti. Ennek tehát az alapszáma kettő

$$R = 2.$$

Az egyes poziciókon előforduló jegyek, az együtthatók pedig a 0 és az 1.

$$0 \leq a_k \leq 1$$

Ezeket az angol "binary digit" (kettes számrendszerbeli számjegy) szavak betüiből összerakva "bit"-nek hivják.

A decimális számrendszernél elmondottak segitségével határozzuk meg az alabbi kettes számrendszerbeli szám értékét!

$$1\,0\,1\,1\,1\,0\,1\,,\,0\,1_{(2)}$$

2^6	2^5	2^4	2^3	2^2	2^1	$\boxed{2^0}$	2^{-1}	2^{-2}
64	32	16	8	4	2	$\boxed{1}$	$\frac{1}{2}$	$\frac{1}{4}$

$$\begin{aligned}
1 \cdot 2^6 &= 1 \cdot 64 = 64 \\
0 \cdot 2^5 &= 0 \cdot 32 = 0 \\
1 \cdot 2^4 &= 1 \cdot 16 = 16 \\
1 \cdot 2^3 &= 1 \cdot 8 = 8 \\
1 \cdot 2^2 &= 1 \cdot 4 = 4 \\
0 \cdot 2^1 &= 0 \cdot 2 = 0 \\
1 \cdot 2^0 &= 1 \cdot 1 = 1 \\
0 \cdot 2^{-1} &= 0 \cdot \frac{1}{2} = 0 \\
1 \cdot 2^{-2} &= 1 \cdot \frac{1}{4} = +\ 0{,}25 \\
\hline
& \qquad\qquad 93{,}25_{(10)}
\end{aligned}$$

$$1011101{,}01_{(2)} = 93{,}25_{(10)}$$

A tizes számrendszernél már emlitettük, hogy minden egyes pozició értéke a tőle jobbra levő pozició értékének alapszámszorosa. Ez azt jelenti, hogy a kettes számrendszerben a pozíciók értékét, a helyi értéket az "egyes" pozicióból (lásd az ábrán bekeretezve) kiindulva kettővel való szorzással, illetőleg osztással kapjuk meg.

Még egy könnyebbség adódik abból, hogy a kettes számrendszerben egy pozíción vagy 0, vagy csak 1 állhat. Ha bármely pozíción 0 áll, akkor ez a pozíció 0-t, ha pedig 1 áll, akkor kettő megfelelő hatványát képviseli. Ez azt jelenti, hogy például: a 2^2 pozíció csak 0-ás vagy 4-es értéket képviselhet, semmi mást. Hasonlóan a 2^1 pozíció is csak 0-t vagy 2-t, a 2^0 pozíció pedig 0-t vagy 1-et jelenthet.

Az előbbi példánál valószínűleg mindenkinek feltűnt, hogy a decimálisan négy helyi értéken felírható szám binárisan 9 pozíciót igényelt. Vagyis lényegesen hosszabban, terjedelmesebben lehetett csak ugyanazt az értéket kifejezni. Az ilyen hosszu számokat, eredményeket nehéz megjegyezni, kimondani, vagy akár csak lemásolni. Ezért a számítógépekkel kapcsolatban használatos még két - az alábbiakban bemutatásra kerülő - másik számrendszer is, amelyek csak az eredmények megjelenítésére, a könnyebb kezelésére, kiértékelésére szolgálnak. A számítógépen belüli ábrázolás és műveletvégzés azonban továbbra is bináris marad!

2.2.3 8-AS (OKTÁLIS) SZÁMRENDSZER

$R = 8$

$0 \leq a_k \leq 7$

A nyolcas alapszámból következően a bármelyik pozíción előforduló legnagyobb együttható a 7. A felírt szám értékét decimálisan így lehet meghatározni:

$$5\ 7\ 0\ 3\ ,\ 2\ 1_{(8)} = ?_{(10)}$$

8^3	8^2	8^1	8^0	8^{-1}	8^{-2}
512	64	8	1	$\frac{1}{8}$	$\frac{1}{64}$

$$5 \cdot 8^3 = 5 \cdot 512 = 2560$$
$$7 \cdot 8^2 = 7 \cdot 64 = 448$$
$$0 \cdot 8^1 = 0 \cdot 8 = 0$$
$$3 \cdot 8^0 = 3 \cdot 1 = 3$$
$$2 \cdot 8^{-1} = 2 \cdot \frac{1}{8} = \frac{16}{64}$$
$$1 \cdot 8^{-2} = 1 \cdot \frac{1}{64} = + \frac{1}{64}$$
$$3011\frac{17}{64}_{(10)}$$

$$5703,21_{(8)} = 3011,266_{(10)}$$

Az eredmény alapján látható, hogy a két különböző számrendszerben ábrázolt szám "hossza" és a körülbelüli nagyságrendje között lényeges különbség nincs. (Ennek a két alapszám közti kis eltérés az oka.) Ezt az oktális számrendszerbeli számot már ki lehet könnyen olvasni, meg lehet jegyezni és a nagyságrendjét is nagyjából látni lehet. A korábban már ismertetett bináris számrendszerbeli szám pedig könnyüszerrel felírható oktálisan, mint ahogy ez a továbbiakban (lásd: 2.5.3 pont) majd ki is derül.

2.2.4 16-OS (HEXADECIMÁLIS-SZEDECIMÁLIS) SZÁMRENDSZER

Ez az oktálishoz hasonlóan a bináris szám rövidebb leirásához használatos. Az alapszáma 16, vagyis egy helyi értéken 16 különböző állapotot lehet kifejezni. Ezek leirásánál problémát csak az jelent, hogy a 16 különböző állapotot nem tudjuk mind számjegyekkel jelölni. Egy pozición itt is csak egy jegy fejezhet ki egy állapotot, és erre a tizes számrendszerben levő 10 különféle érték kevés. Segitségül a hiányzó 6 állapot leirásához az A B C első hat betüjét használjuk. Igy a hexadecimális számrendszerben az egy helyi értéken kifejezhető 16 különféle állapot közül hatot (10-től 15-ig) szám helyett betüvel jelölünk.

$R = 16$

$0 \leq a_k \leq 15$

$10_{(10)} = A_{(16)}$
$11_{(10)} = B_{(16)}$
$12_{(10)} = C_{(16)}$
$13_{(10)} = D_{(16)}$
$14_{(10)} = E_{(16)}$
$15_{(10)} = F_{(16)}$

Az eddigiekben alkalmazott "helyi értékek szerinti kifejezés"-t felhasználva, irjuk fel decimálisan az alábbi szám értékét!

$$2\ A\ 3,\ D\ 1_{(16)} = ?_{(10)}$$

16^2	16^1	16^0	16^{-1}	16^{-2}
256	16	1	$\frac{1}{16}$	$\frac{1}{256}$

$$2 \cdot 16^2 = 2 \cdot 256 = 512$$
$$A \cdot 16^1 = 10 \cdot 16 = 160$$
$$3 \cdot 16^0 = 3 \cdot 1 = 3$$
$$D \cdot 16^{-1} = 13 \cdot \frac{1}{16} = \frac{13}{16}$$
$$1 \cdot 16^{-2} = 1 \cdot \frac{1}{256} = + \frac{1}{256}$$
$$675\,\frac{208}{256}{}_{(10)}$$

$$2A3,D1_{(16)} = 675,81_{(10)}$$

Az oktális számrendszernél elmondott előnyök (azonos poziciószám, kiolvashatóság, nagyságrendi értékelés) az eredmény alapján láthatóan itt is fennállnak. A bináris szám felirása hexadecimálisan pedig a 2.5.3 pontban ismertetett módon szintén nagyon egyszerü.

Ezek után számoljunk egyesével a négy megismert számrendszerben, hogy néhány általános szabályra felhivhassuk a figyelmet!

R = 10 (decimális)	R = 2 (bináris)	R = 8 (oktális)	R = 16 (hexadecimális)
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27	11011	33	1B
⋮	⋮	⋮	⋮

Bármelyik számrendszerben érvényesek az alábbi számolási szabályok:

- A számolást a legkisebb helyi értéken kezdjük, és ott a 0-tól kezdve sorozatosan növeljük az értékeket mindaddig, amig az adott számrendszerben az egy helyi értéken kifejezhető legnagyobb számot el nem érjük.
  (Ez minden számrendszerben az alapszámmal megegyező darabszámu számjegyet jelent.)
- Ha a legkisebb helyi értéken elérjük a maximális értéket, ezen a pozicióon további növelésre már nincs lehetőség. Egy ennél nagyobb szám kifejezéséhez már 2 számjegyre van szükségünk. Igy ezen a helyi értéken kezdjük elölről (0-tól) a számolást és a következő, magasabb helyi értéken levő számot pedig eggyel megnöveljük.
- Ez a szabály akkor is érvényes, ha egyszerre két vagy több egymás utáni helyi értéken is elérjük a maximális értéket. Ilyenkor mindegyiket helyettesitjük nullával, és a következő, magasabb helyi értékhez adunk egyet.

⋮	⋮	⋮	⋮
$99_{(10)}$	$11_{(2)}$	$77_{(8)}$	$FF_{(16)}$
$100_{(10)}$	$100_{(2)}$	$100_{(8)}$	$100_{(16)}$

- Vegyük észre, hogy az oktálisan egy helyi értéken kifejezhető számok bináris leirásához három pozició szükséges (ezt majd a bináris - oktális átalakitásnál fogjuk felhasználni).
- A hexadecimálisan egy helyi értéken kifejezhető számok bináris leirásához pedig négy pozició szükséges (ezen fog alapulni majd a bináris - hexadecimális átalakitás).

2.3 ARITMETIKAI MŰVELETEK A KÜLÖNBÖZŐ SZÁMRENDSZEREKBEN. KOMPLEMENTÁLÁS

A digitális számitógépek csak binárisan irt operandusokat képesek feldolgozni. A gép ugynevezett bináris aritmetikai egysége ezekkel aritmetikai müveleteket tud végezni. A leglényegesebb ilyen müvelet az összeadás, illetőleg kivonás, mivel a szorzás és osztás müvelete ezekre már visszavezethető. (Az összeadás és kivonás megfelelő sokszor ismételt alkalmazásával elvégezhető tulajdonképpen a szorzás és osztás is.)

Az ugynevezett komplemensképzés pedig lehetővé teszi, hogy minden müveletet vissza lehessen vezetni komplementálással vagy összeadásra vagy kivonásra. Igy a gép számára elég, ha vagy csak összeadó vagy csak kivonó egységgel rendelkezik.

A programozási feladatok megkönnyitésére viszont a már megismert oktális és hexadecimális számrendszer alkalmas. Ezeken belül a program ellenőrzéséhez azonban csak összeadásra és kivonásra van szükség. Az esetleg előforduló szorzáshoz és osztáshoz célszerübb az operandusokat decimálissá visszaalakitani!

2.3.1 BINÁRIS ÖSSZEADÁS

A gépek egyszerre csak két operandussal képesek müveletet végezni, több szám ("számoszlop") összeadását is igy "kettesével" végzik el. Igy egy helyi értéken - összeadásnál - az alábbi esetek fordulhatnak elő:

0 + 0, 0 + 1, 1 + 0, 1 + 1

Ezek <u>összeadásakor</u> az alábbi <u>szabályok</u> érvényesek:

$$\begin{array}{r} 0 \\ +\,0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ +\,1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\,0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\,1 \\ \hline 1\,0 \end{array}$$

Az eredmény számjegye

Átvitel az eggyel nagyobb helyi értékre

A négy eredmény közül bővebb magyarázatra csak az utolsó szorul. Mivel a bináris számrendszerben az egy helyi értéken kifejezhető legnagyobb szám az 1, az ennél eggyel nagyobb szám (összeg) ábrázolásához már két helyi értékre van szükség. Az adott helyi értéken a számolás kezdődik elölről (0-tól),

és a nagyobb helyi értéken pedig 1-gyel meg kell növelni az ott levő értéket. Ezt a nagyobb helyi értékre átkerülő 1-est hivják átvitelnek (v.ö. 2.2.4 pontban a különböző számrendszerekben egyesével történő számolásnál elmondottakkal!).

Hosszabb számok bináris összeadásánál is minden egyes pozición a fenti négy eset valamelyikét kell használni az adott bitpárositásnak megfelelően.

```
    1 1 1 <------ átvitelek az alacsonyabb helyi értékekről
    ↖ ↖ ↖
  0 0 1 0 1 0
+ 1 0 1 1 1 0
-------------
  1 1 1 0 0 0
```

Igy például a 3. oszlop külön figyelmet érdemel. Itt eredetileg 1 + 1 = 0 lett volna az eredmény és 1 átvitel keletkezett. Mivel az alacsonyabb helyi értékről is volt átvitel, az oszlop eredménye 0 + 1 = 1 és továbbra is van 1 átvitel.

2.3.2 BINÁRIS KIVONÁS

A lehetséges négy bitpárositás esetén a kivonásra az alábbi szabályok érvényesek:

```
                          Áthozat az eggyel magasabb helyi értékről
                                  ↓
  0        1        1        1 0
- 0      - 0      - 1      - 1
---      ---      ---      ---
  0        1        0        1
```

Hosszabb bináris operandusok esetén is az előbbi négy eset valamelyikét kell alkalmazni. Végezzük el az alábbi müveletet:

```
  1 1 1 0
- 0 1 0 1
---------
```

A legkisebb helyi értéken a 0-1 müveletet csak ugy lehet elvégezni, ha egy magasabb helyi értékről kölcsönveszünk, "áthozunk" arra a helyi értékre. Ahonnan a kölcsönt vesszük, ott a pozició értéke 0 lesz (ezt bekarikázással jelöltük). A kisebb pozición pedig ez a "kölcsönvett 1-es" már két darab 1-est "ér"! Ebből a két 1-esből kell a müveletben kijelölt egy 1-est levonni. Igy lesz az eredmény bitje 1.

```
        1
      ↗ 1
  1 1 (1) 0
- 0 1 0 1
---------
  1 0 0 1
```

Ha az áthozat adott esetben nem a szomszédos pozicióról történik, akkor a következő helyi értékre mindig csak egy 1-est kell pozíciónként továbbvinni, és az ott két 1-est fog érni.

```
  1 0 0 0
- 0 0 1 1
---------
```

A legelső pozíciónál tudunk csak kölcsönvenni. Ott 0 lesz helyette, a következő pozícióra pedig két 1-es kerül. Ezek közül csak az egyiket visszük tovább, a másikat ott hagyjuk. Az átvitt 1-es a kisebb helyi értéken megint két 1-est fog érni, amiből megint csak egyet viszünk tovább. Igy a legkisebb helyi értékre két 1-es került, amiből már egyet le lehet vonni!

```
    1 1 1
  ↗ (1)(1) 1
 ↗ (1) 0 0 0
- 0 0 1 1
---------
  0 1 0 1
```

2.3.3 BINÁRIS SZORZÁS

A szorzás tulajdonképpen összeadások sorozata. Képezni kell az egyes részletszorzatokat és ezeket össze kell adni. A számitógép azonban ilyen módon nem képes a szorzás elvégzésére, mert egy adott időben csak két szám összeadására képes. Ehelyett a szorzást a következő lépések ismételt végrehajtására vezeti vissza.

- A szorzandót addig a helyi értékig lépteti valamelyik irányba, amig azon a helyi értéken a szorzóban 1-est nem talál.
- A megelőző részletszorzathoz hozzáadja az ilyen módon elléptetett szorzandót, vagyis ezzel előállitja a következő részletszorzatot. (A kiindulási állapotban a részletszorzat végig 0.)

Kisérjük végig az elmondottakat az alábbi példán!

```
0 1 1 1 1 0 · 0 1 0 0 1 1
szorzandó     szorzó

                          0 1 1 1 1 0 · 0 1 0 0 1 1
Kiindulási állapotban a
részletszorzat ———————→   0 0 0 0 0 0          ↑ ↑ ↑ első szorzóbit
                        + 0 1 1 1 1 0          |   |   második szorzóbit
Első részletszorzat ——→   0 1 1 1 1 0          |     harmadik szorzóbit
                      + 0 1 1 1 1 0
Második részlet-
szorzat ——————————→       1 0 1 1 0 1 0
                    + 0 1 1 1 1 0
Szorzás ered-
ménye ————————→           1 0 0 0 1 1 1 0 1 0
```

2.3.4 BINÁRIS OSZTÁS

Az osztást a számitógép nulláig folytatott sorozatos kivonásokkal valósitja meg, feltéve, hogy az aritmetikai egysége kivonómüvet tartalmaz.

Abban az esetben, ha a kisebbitendő (az osztandónak az osztóval megegyező hosszuságu része) nagyobb, mint a kivonandó (osztó), a hányados jegye 1-es lesz. Ha a kisebbitendő kisebb, mint a kivonandó, a hányados jegye 0 lesz. Az uj kisebbitendő értékét ugy kapjuk, hogy az osztandó következő jegyét leirjuk a maradékhoz. Amikor az osztandónak már nincs több számjegye, az utolsó kivonás eredménye adja az osztás maradékát.

```
0 1 1 1 0 1 0 1 : 1 0 1 0 1
    osztandó          osztó

  0 1 1 1 0 1 0 1 : 1 0 1 0 1 = 0 1 0 1
  0 1 1 1 0 1                   hányados
-   1 0 1 0 1
  0 0 1 0 0 0 0
      1 0 0 0 0 1
    -   1 0 1 0 1
      0 0 1 1 0 0
       maradék
```

Ha az aritmetikai egységben összeadómü van, az osztást a komplemensek hozzáadásával végzi el a gép.

2.3.5 KOMPLEMENSKÉPZÉS

A számitógép megjelenése szükségessé tette, hogy az alapvető aritmetikai müveleteket másféleképpen oldjuk meg. A hétköznapi gyakorlatban különböző előjelü számokat könnyedén össze tudunk vonni ugy, hogy külön összeadjuk a pozitiv és külön a negativ előjelüeket, megállapitjuk a két összeg közti különbséget és a nagyobb abszolut értékü szám (összeg) előjelét tulajdonitjuk az eredmény előjelének. A számitógép azonban nem tud ilyen módon előjeles számokkal müveletet végezni. Helyette az ugynevezett komplemens aritmetikát használja.

A komplemens szó két, egymást kölcsönösen kiegészitő dolog egyikét jelenti. Emlékezzünk vissza a modulusra! Ezt mint az eszközzel megjelenithető összes egymástól különböző állapot számát definiáltuk. Vegyük például ismét a gépkocsi kilométeróráját, ami mutasson pillanatnyilag 42 698 km-t! Az eszköz (kilométeróra) modulusa 100 000. ($M = 10^5$ - feltéve, hogy a kilométeróra decimálisan számol és öt helyi értéken jeleniti meg az eredményt.) A pillanatnyilag leolvasott szám komplemensét ugy kapjuk meg, ha értékét a modulusból levonjuk:

100 000 - 42 698 = 57 302

A gépkocsi tulajdonképpen még ennyi kilométert futhat, amig az óra megint elölről nem kezd számolni.

A komplemens tehát egy adott eszközön még kifejezhető különböző állapotok számát jelenti.

Például egy helyi értéken történő ábrázolásnál (M = 10) a $4_{(10)}$ komplemense

$10 - 4 = 6.$

Két helyi értéken (M = 100) ugyanennek a számnak a komplemense

$100 - 4 = 96$

Egyezzünk meg, hogy a negativ számokat a komplemensükkel ábrázoljuk! Hogy minden esetben egyértelmüen el lehessen dönteni, vajon az adott szám egy tényleges pozitiv számot jelent-e, vagy hogy egy eredetileg negativ szám komplemense-e, az adott hosszon kifejezhető számtartományt felezzük meg! Pozitiv számot csak a számtartomány feléig ábrázolunk, a számtartomány másik felével pedig a pozitivnak megfelelő negativ számok komplemenseit fejezzük ki!

Két helyi értéken például igy csak + 49-ig ábrázolhatók a pozitiv számok. A 49-től a 99-ig terjedő számértékek a -1-től a -49-ig terjedő, de negativ számok komplemenseinek kifejezésére szolgálnak.

$-27_{(10)}$ ábrázolása komplemensével $73_{(10)}$

Ez már a számtartomány "felső felébe" esik, és ebből lehet tudni, hogy egy eredetileg negativ szám komplemenséről van szó.

De hogyan ábrázoljuk a +50-et vagy az annál nagyobb pozitiv számokat? Ehhez a két pozició már kevés, ilyenkor legalább három pozicióra van szükség, ahol az ábrázolható számtartomány ismét felezésre kerül, de az "alsó felében" most már 499-ig lehet a pozitiv számokat ábrázolni.

Például a +65 ábrázolása csak 065-ként lehetséges, mert a két pozicion megjelenő 65 az elmondottak értelmében a -35-öt jelentené, annak komplemenseként.

A komplementálási müvelet segitségével a kivonási müvelet visszavezethető összeadásra, feltételezve, hogy az előbbiek betartásán tul a "számolásra" rendelkezésre álló pozíciók számát is ismerjük! Nézzünk erre példát!

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

Hagyományos módon, kivonással kaptuk meg az eredményt. Most az "1 levonása" helyett adjuk hozzá a 4-hez az 1 komplemensét (egy helyi értéken számolva!)

$1_{(10)}$ komplemense $9_{(10)}$

$$\begin{array}{r} 4 \\ +\ 9 \\ \hline 13 \end{array}$$

Az előre megállapitott helyi értéken az eredményt helyesen kaptuk meg. Ugyanezt a müveletet két pozíciót felhasználva is elvégezhetjük. Az eredmény akkor is helyes lesz!

$$\begin{array}{r} 04 \\ \underline{-\ 01} \\ +\ 03 \end{array}$$

$01_{(10)}$ komplemense $99_{(10)}$

$$\begin{array}{r} 04 \\ \underline{+\ 99} \\ 103 \end{array}$$

(A megadott poziciószám előtt keletkező 1-es az ugynevezett átvitel - lásd még a 2.7.1.1 fejezetben!)
Végezzük most el az alábbi müveletet!

3 - 7

Az előbb elmondottak értelmében ezt már csak két pozíción tudjuk elvégezni, mert a +7 és igy a komplemense is már csak ott ábrázolható.

$$\begin{array}{r} 03 \\ \underline{-\ 07} \\ -\ 04 \end{array} \quad \text{komplemensképzéssel} \quad \begin{array}{r} 03 \\ \underline{+93} \\ 96 \end{array}$$

Az eredmény a számtartománynak abba a felébe esik, amely már a negativ számok komplemenseit tartalmazza. Igy tehát még nem a tényleges eredményt kaptuk meg! Ahhoz ezt még "rekomplementálni" kell!

100 - 96 = 04

Ezt az értéket viszont a rekomplementálás miatt negativ előjellel kell venni, és akkor már tényleg a helyes eredményhez jutottunk!

Ilyen módon a kivonást egyértelmüen vissza lehetett vezetni összeadásra, és hasonló módon az összeadás is visszavezethető kivonásra! Ennek az az előnye, hogy a számitógépnek nem kell külön összeadó és kivonó egységgel rendelkeznie (nem kell tudnia összeadni és kivonni is), minden müveletet el tud végezni vagy csak összeadó vagy csak kivonó egységgel (elég csak valamelyik müveletet és a komplementálást tudni).

2.3.6 OKTÁLIS ÖSSZEADÁS

Az összeadást az alábbiaknak megfelelően kell elvégezni:

- Az adott helyi értéken levő számjegyeket decimálisan összeadjuk.
- A részösszeget osztjuk 8-cal, a maradék az eredmény számjegye lesz az adott pozicióban, a hányados pedig a következő helyi értékre kerülő átvitelt adja.

Adjuk össze oktálisan a következő két számot:

$$\begin{array}{r} 3457_{(8)} \\ +\ 1640_{(8)} \\ \hline \end{array}$$

Akár a bináris összeadásnál, itt is célszerű a kisebb helyi értékről származó átvitelt az adott oszlop fölé írni, ahol azt figyelembe kell venni!

$$\begin{array}{r} 1\ 1\ \\ 3\ 5\ 4\ 7_{(8)} \\ +\ 1\ 6\ 4\ 0_{(8)} \\ \hline 5\ 4\ 0\ 7_{(8)} \end{array}$$

Több számból álló számoszlop összeadása ritkán fordul ugyan elő, de gyakorlásképpen nézzünk egyet!

$$\begin{array}{r} 6\ 5\ 2\ 7_{(8)} \\ 1\ 7\ 3\ 5_{(8)} \\ 3\ 7\ 2\ 6_{(8)} \\ +\ 4\ 0\ 1\ 7_{(8)} \\ \hline \end{array}$$

Ilyen esetben a kisebb helyi értékről származó átvitel már nemcsak 1 lehet!

$$\begin{array}{r} 2\ 2\ 1\ 3\ \\ 6\ 5\ 2\ 7_{(8)} \\ 1\ 7\ 3\ 5_{(8)} \\ 3\ 7\ 2\ 6_{(8)} \\ +\ \ 4\ 0\ 1\ 7_{(8)} \\ \hline 2\ 0\ 4\ 3\ 1_{(8)} \end{array}$$

2.3.7 OKTÁLIS KIVONÁS

A kivonás elvégzésénél két eset fordulhat elő:

- A kivonandó számjegye kisebb, mint a kisebbitendő számjegye, ilyenkor a kivonást a decimálisnak megfelelően végrehajtjuk.
- A kivonandó számjegye nagyobb, mint a kisebbitendőé, így a kivonást közvetlenül elvégezni nem tudjuk. A binárishoz hasonlóan ilyen esetben itt is kölcsön kell vennünk, egy nagyobb helyi értékről. Ahonnan a kölcsönt vesszük, ott a kisebbitendő számjegye eggyel csökken, ahová pedig a kölcsönt átvisszük, ott az 8-at ér! Ezt hozzáadva decimálisan a kisebbitendő ott levő számjegyéhez, a kivonást decimálisan már elvégezhetjük.

$$\begin{array}{r} 5\ 4\ 0\ 6_{(8)} \\ -\ 3\ 5\ 4\ 7_{(8)} \\ \hline \end{array}$$

Első lépésként az utolsó helyi értékre kell kölcsönt "szerezni". Ez csak a harmadik oszlopból sikerül. Ott a 4 helyett marad 3, a következő oszlopba viszont átkerülve a kölcsön 8-at ér. Ebből egyet megint tovább kell vinni, tehát itt 7 marad, a legkisebb helyi értékre pedig 8 kerül. Tehát itt összesen 8 + 6 = 14 lesz. Ebből kell levonni 7-et.

$$
\begin{array}{r}
7 \\
3\ \cancel{8}\ 8 \\
5\ \cancel{4}\ 0\ 6_{(8)} \\
-\ 3\ 5\ 4\ 7_{(8)} \\
\hline
?\ 3\ 7_{(8)}
\end{array}
$$

A harmadik oszlopban a kisebbitendő számjegye ismét kisebb, mint a kivonandóé, tehát megint kölcsönre van szükség!

$$
\begin{array}{r}
8\ 7 \\
4\ 3\ \cancel{8}\ 8 \\
\cancel{5}\ \cancel{4}\ 0\ 6_{(8)} \\
-\ 3\ 5\ 4\ 7_{(8)} \\
\hline
1\ 6\ 3\ 7_{(8)}
\end{array}
$$

2.3.8 HEXADECIMÁLIS ÖSSZEADÁS

Az oktális összeadásnál emlitetteknek megfelelően kell ezt is elvégezni, de a részösszeget itt 16-tal kell osztani!

$$
\begin{array}{r}
6\ B\ 8\ A_{(16)} \\
+\ 7\ 9\ D\ C_{(16)} \\
\hline
\end{array}
$$

A hexadecimális számrendszerben levő betűket természetesen decimális értékükkel kell figyelembe venni!

$$
\begin{array}{r}
1\ 1\ 1 \\
6\ B\ 8\ A_{(16)} \\
+\ 7\ 9\ D\ C_{(16)} \\
\hline
E\ 5\ 6\ 6_{(16)}
\end{array}
$$

Adjuk össze a következő hexadecimális számoszlopot!

$$
\begin{array}{r}
F\ 9\ 8\ A_{(16)} \\
7\ D\ B\ 2_{(16)} \\
B\ C\ E\ 5_{(16)} \\
+\ 9\ 0\ F\ F_{(16)} \\
\hline
\end{array}
\quad \Longrightarrow \quad
\begin{array}{r}
2\ 2\ 3\ 2 \\
F\ 9\ 8\ A_{(16)} \\
7\ D\ B\ 2_{(16)} \\
B\ C\ E\ 5_{(16)} \\
+\ \ 9\ 0\ F\ F_{(16)} \\
\hline
2\ C\ 5\ 2\ 0_{(16)}
\end{array}
$$

2.3.9 HEXADECIMÁLIS KIVONÁS

Ugyanaz a két eset lehetséges itt is, mint amit az oktálisnál már megismertünk. A különbség csak az, hogy a kölcsön a kisebb helyi értéken itt most 16-ot ér!

$$\begin{array}{r} B\ 4\ A\ 9_{(16)} \\ -\ 8\ A\ F\ 5_{(16)} \\ \hline \end{array}$$

Megint több oszlopban van szükség kölcsönre, igy azt lépésről lépésre elvégezve:

$$\begin{array}{rccc} & 16 & & \\ & 10 & 3 & 16 \\ & \cancel{B} & \cancel{4} & A\ 9_{(16)} \\ - & 8 & A & F\ 5_{(16)} \\ \hline & 2 & 9 & B\ 4_{(16)} \end{array}$$

2.4 LOGIKAI MŰVELETEK. BOOLE-ALGEBRA

A számitógépek széles körü felhasználása érdekében követelmény velük szemben, hogy ne csak numerikus adatokat, hanem eseményeket, fogalmakat, jelenségeket is kezelni tudjanak, fel tudjanak dolgozni. Ez a képesség az emberi gondolkodás utánzását, modellezését teszi lehetővé. A gondolkodás folyamata ugyanis nem más, mint az eseményekkel, jelenségekkel végzett következtetések, logikai müveletek sorozata. Ezekkel matematikai formában az ugynevezett logikai algebra (Boole-algebra) foglalkozik. Az egyes eseményeket, "itételeket" a logikai algebrában logikai változónak hivják. Jelölésükre az ABC valamelyik nagy betüjét szokás használni

A - süt a nap

D - szabad szombat van

F - éhes vagyok stb.

Minden logikai változónak - eseménynek - két lehetséges állapota van: vagy bekövetkezik, amit állitottunk, vagy nem (és ilyenkor szükségszerüen pont az ellenkezője következik be!). Ha az esemény, amit logikai változónak tekintünk (és egy nagy betüvel jelölünk) bekövetkezik, akkor azt mondjuk, hogy logikai értéke IGAZ, és jelölése "1". Ha az esemény nem az állitásnak megfelelően következett be (hanem pont az ellenkezője), a logikai értéke HAMIS, és jelölése "0". Általában az algebrában a változóknak, betüszimbólumoknak numerikus jelentésük van és különféle értékeket vehetnek fel. A Boole-algebrában a változóknak csupán logikai jelentésük van és csak a fenti két lehetséges értéket tulajdonithatjuk nekik.

A logikai függvény megadja a bemenő és kimenő események közti kapcsolatot.

$$\boxed{\text{Bemenő esemény}} \xrightarrow[\text{logikai függvénykapcsolat}]{\text{következtetés(ek)}} \boxed{\text{Kimenő esemény}}$$

Például:

$X = X\ (A, B, C)$

X - a kimenő eseményt jelöli

A, B, C - a bemenő eseményt jelölik

A fenti jelölésmód azt jelenti, hogy az X logikai változó általánosságban az A, B, C-vel jelölt logikai változók (események) függvénye. A köztük levő kapcsolatot, a tényleges összefüggést csak akkor tudjuk meg, ha feltüntetjük az egyes logikai változók között elvégzendő un. logikai műveleteket. A fenti példa esetében például:

$$X = \overline{A} + B \cdot C$$

A logikai műveletek az események (logikai változók) logikai értékeivel végzendő műveletek és eredményük, értékük csak azok logikai értékétől függ. A Boole-algebra három alapművelete a NEM, az ÉS és a VAGY művelet.

A Boole-algebra sajátosságait egy labdarugó mérkőzéssel kapcsolatos példa keretében mutatjuk be. Logikai változóként az alábbi eseményeket vesszük csak figyelembe (a számtalan, még tekintetbe vehető feltétel közül). Ezek határozzák meg számunkra, hogy elmegyünk-e végül mérkőzésre vagy sem.

1. Az első szempont, "esemény", hogy van-e 20.- Ft költeni való pénzünk.

   Ha van 20.- Ft költeni valónk, ezt az esetet jelöljük "A"-val, hogy ne kelljen ezt a mondatot ujra és ujra leirnunk.

   Tehát

   A = van 20 forintunk

   Ha nincs 20 forintunk, akkor az A-val jelölt állitásunkat tagadjuk, tehát a "NEM A" jelölésére "$\overline{A}$" szolgál

   NEM A = $\overline{A}$ = nincs 20 forintunk.

2. A következő tényező, szempont, hogy szombat van-e? Ezt az eseményt, állitást jelöljük "B"-vel!

   B = szombat (van)

   ($\overline{B}$ = nincs szombat)

   Az "F" jelölje azt az eseményt, hogy elmegyünk mérkőzésre! Azt mondjuk, hogy "elmegyünk a mérkőzésre (F), ha van 20 forintunk (A), ÉS szombat van (B). A logikai ÉS * művelet jelölésére a szorzás jele használatos, de mint a matematikában, itt is gyakran elhagyjuk a · jelét. A fenti mondat, állitás tehát szimbolikus formában a következőképpen néz ki:

   $F = A \cdot B$, illetve még rövidebben $F = AB$

3. Hogyan vonható be az egyenletbe az a további feltétel, hogy akkor megyünk el mérkőzésre, ha kupatalálkozó van? Jelölje "C" azt az eseményt, hogy kupatalálkozó van! Az állitás tehát igy fog hangzani:

   "Elmegyünk a mérkőzésre (F), ha van 20 forintunk, ÉS szombat van, ÉS kupatalálkozó van."

   Röviden ez igy néz ki:

   $F = ABC$

---

* Az ÉS-t nagybetűvel irjuk, hogy megkülönböztessük a logikai műveletet az "és" kötőszótól. Ugyanigy járunk el a VAGY és a NEM esetében is.

4. Ha esik az eső, NEM megyünk el!

Azt az eseményt, hogy "esik" az eső, jelöljük "D"-vel!

"$\overline{D}$" igy azt jelenti, hogy "NEM esik" az eső.

$F = ABC\overline{D}$

Tehát „elmegyünk a mérkőzésre, ha van 20 forintunk, ÉS szombat van, ÉS kupatalálkozó van, ÉS NEM esik az eső".

5. Hogyan alakulna a kifejezés, ha azt mondanánk, hogy "elmegyünk a mérkőzésre, ha van 20 forintunk, ÉS szombat van, ÉS kupatalálkozó van, ÉS NEM esik az eső, VAGY ha van 20 forintunk, ÉS szombat van, ÉS kupatalálkozó van, ÉS esik az eső, ÉS van esőkabátunk.

A VAGY kapcsolatot az összeadás jele (+) jelenti.

"E" jelentse azt, hogy van esőkabátunk!

$F = ABC\overline{D} + ABCDE$

Ez tehát egy olyan logikai függvénykapcsolat, amely a figyelembe vett bemenő események (A, B, C, D, E) és a kimenő esemény (F) között a logikai müveletek (NEM, ÉS, VAGY) segitségével kapcsolatot teremt. De, hogy tényleg elmegyünk-e mérkőzésre, az még ebből nem derül ki. F logikai értéke lehet "1" (IGAZ - tehát valóban elmegyünk) és lehet "0" is (HAMIS - tehát nem igaz, hogy elmegyünk a mérkőzésre). Ez attól fog függeni, hogy a figyelembe vett események, a logikai változók milyen értéket vesznek fel.

Most nézzük ugy, hogy az egyes logikai változók tényleges logikai értékéből hogy lehet logikai müveletek segitségével a függvény (a kimenő esemény) logikai értékére következtetni.

2.4.1 A LOGIKAI NEM MÜVELET

Ez a legegyszerübb logikai müvelet, egyetlen bemenő eseményen végzett müveletvégzést jelent. Hivják még NEGÁCIÓ-nak, TAGADÁS-nak, INVERTÁLÁS-nak is.

A NEM müvelet jelölése a logikai változó betüjelének felülvonásával történik. Ha például azt az eseményt, hogy "esik az eső" "A"-val jelöltük, az ellentétes állitást, a "tagadott eseményt" ("NEM esik az eső") "$\overline{A}$"-val jelöljük.

A logikai müveletben szereplő bemenő és kimenő logikai változók logikai értékei közötti összefüggést az ugynevezett igazságtáblázat adja meg.

A NEM müvelet igazságtáblázata:

	bemenő esemény	kimenő esemény
	A	$\overline{A}$
az események logikai értékei	0 →	1
	1 →	0

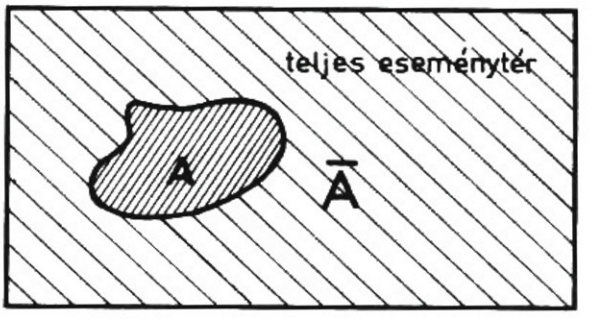
Gondoljuk végig ennek helyességét egy példán! Jelölje ismét "A" azt az eseményt, hogy "esik az eső", igy "$\overline{A}$" ennek tagadottját, ellentétjét jelenti, vagyis azt, hogy "NEM esik az eső". Tegyük fel, hogy NEM esik az eső! Ekkor az előbbi jelölések alapján az "A"-val jelölt állitás HAMIS, tehát logikai

értéke "0", az "$\overline{A}$"-val jelölt állitás logikai értéke viszont IGAZ, tehát "1". Ha esik az eső, "A" logikai értéke lesz IGAZ (1), "$\overline{A}$" logikai értéke pedig HAMIS (0).

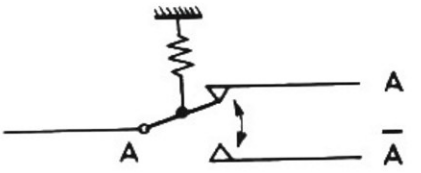
Bármelyik esetet is tekintjük, az ellentett esemény ($\overline{A}$) logikai értékét a bemenő esemény (A) értékének invertálásával, ellenkezőre változtatásával kapjuk meg. Tehát valamely logikai változó negálása, tagadása, eredeti logikai értékének ellentétesre változtatását jelenti. A logikai tagadást a halmazelméletben a következőképpen értelmezik (6. ábra).

Vagyis az összes figyelembe vett esemény halmazán belül (teljes eseménytér) az ellentett eseményhez ($\overline{A}$) tartoznak azok az események, amelyek kivül esnek a kiinduló esemény (A) halmazán.

Az elektronikában a "Morse" érintkezős kapcsolóval lehet a NEM kapcsolatot modellezni (7. ábra).



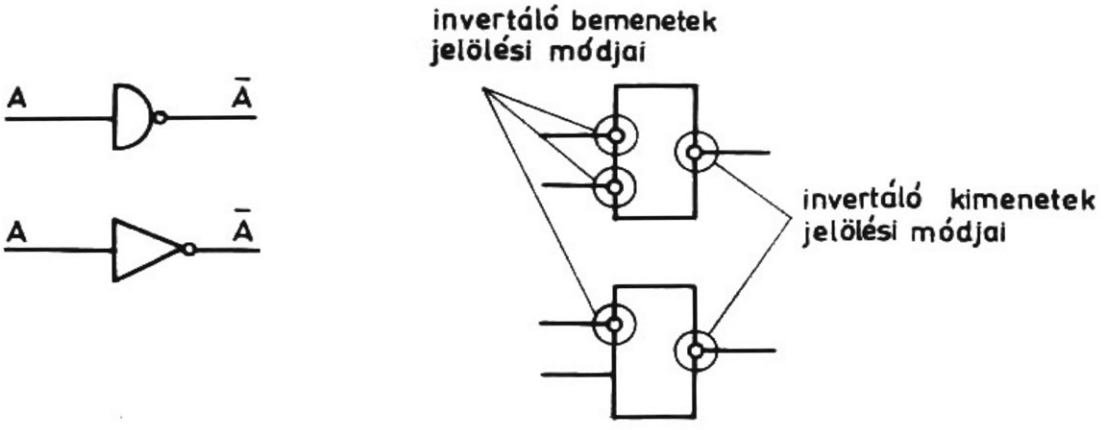
6. ábra.
"A", "$\overline{A}$" halmazelméleti értelmezése



7. ábra.
NEM kapcsolat áramköri értelmezése

A gyakorlati kapcsolásokban a tranzisztoros vagy integrált áramkörös "inverterekkel", NEM kapukkal realizálják a logikai NEM müveletet. Ezek rajzbeli szimbóluma (a tényleges kiviteltől függetlenül) az alábbi (8. ábra).

A bal oldaliak széles körben elterjedt jelölési módok. Irodalomban, gyakorlatban nálunk is sokfelé megtalálhatók. A jobb oldali szimbólumot a KGST-n belül szabványositották és hazánkban is kötelezően ezt kell(ene) használni 1974. október 1. óta.



8. ábra. A NEM kapu rajzjelei

2.4.2 A LOGIKAI ÉS MÜVELET

A logikai ÉS müveletben kettő vagy több bemenő esemény és egy kimenő esemény szerepel. Jelölésére a szorzás (·) jele (amit gyakran az egyszerüség kedvéért el is hagyunk) vagy a $\wedge$, illetve $\cap$ jel szolgál (utóbbiak a halmazelméletben használt jelölések).

$$X = A \cdot B = AB = A \wedge B = A \cap B$$

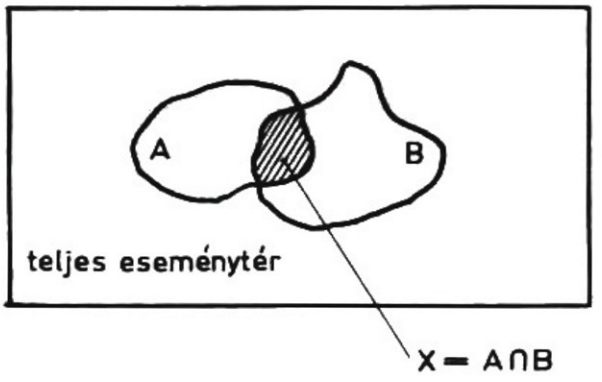
A fejezet elején bemutatott példára visszatérve, akkor megyünk mérkőzésre, ha van 20 forintunk, ÉS szombat van, ÉS stb. Tehát az egymással ÉS kapcsolatban levő bemenő események együttes bekövetkezésére van szükség, hogy a kimenő esemény bekövetkezzen. Ha az ÉS müveletben szereplő logikai változók közül csak egynek az értéke is 0 (HAMIS), a kimenő esemény logikai értéke is 0 (HAMIS) lesz. Ennek megfelelően az igazságtáblázat a következőképpen néz ki:

Az események logikai értékei

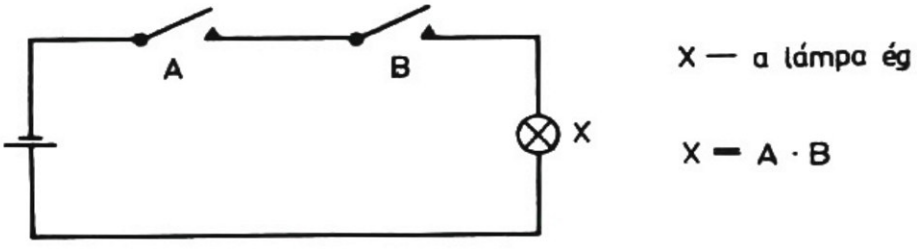
bemenő események		kimenő esemény
A	B	X = A·B
1	1	1
1	0	0
0	1	0
0	0	0

A logikai ÉS müveletet a halmazelméletben két halmaz közös részeként (metszeteként) értelmezik (9. ábra).

Az elektronikus eszközök területén kapcsolók sorbakötésével modellezhető az ÉS kapcsolat (10. ábra). Megegyezés alapján a nyitott kapcsolóhoz logikai nullát, a zárthoz pedig logikai egyet rendelünk!



9. ábra. Az ÉS kapcsolat halmazelméleti szemléltetése



10. ábra. Az ÉS kapcsolat áramköri értelmezése

A logikai kapcsolásokban az ÉS müvelet megvalósitására az ugynevezett ÉS kapuk szolgálnak. Ezek rajzjele a következő (11. ábra):

11. ábra. Az ÉS kapu rajzjelei

2.4.3 A LOGIKAI VAGY MÜVELET

Az ÉS müvelethez hasonlóan itt is kettő vagy több bemenő esemény és egy kimenő esemény szerepel. Jelölésére az összeadás jele (+), illetőleg a halmazelméletben a $\vee$ vagy $\cup$ szolgál.

$X = A + B = A \vee B = A \cup B$

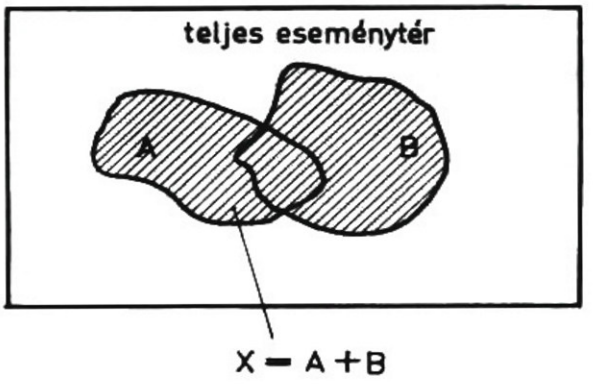
Visszatérve a korábbi példára, elmegyünk a mérkőzésre, ha nem esik az eső, VAGY ha van esőkabátunk. Elég a két feltétel közül valamelyiknek bekövetkeznie ahhoz, hogy a "kimenő esemény létrejöjjön" és a VAGY kapcsolatban szereplő logikai esemény logikai értéke már IGAZ (1) lesz.

A VAGY kapcsolat igazságtáblázata a következő:

Az események logikai értékei

bemenő események		kimenő esemény
A	B	X = A + B
1	1	1
1	0	1
0	1	1
0	0	0

A halmazelméletben két halmaz egyesitett területét (unióját) tekintik a VAGY kapcsolat megvalósitásának (12. ábra).

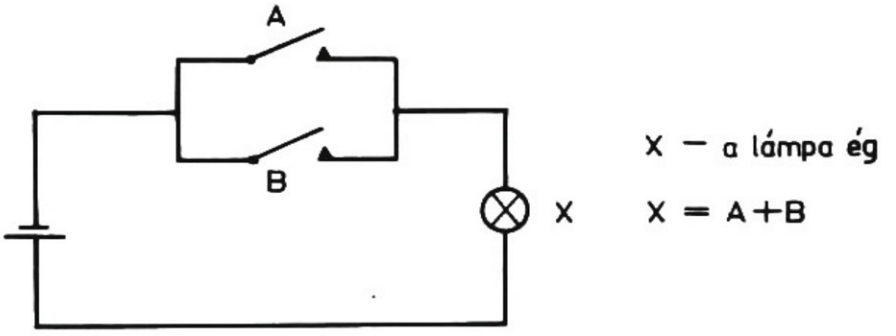


12. ábra.
A VAGY kapcsolat halmazelméleti szemléltetése

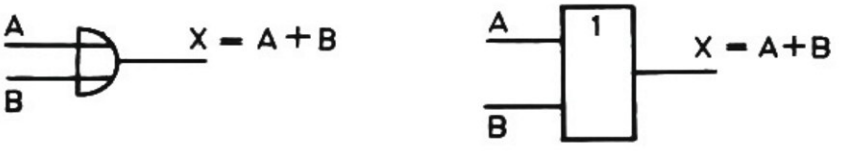
Elektronikában kapcsolók párhuzamos összekötésével lehet a VAGY kapcsolat működését elképzelni (13. ábra).

Az ugynevezett VAGY kapu a VAGY kapcsolat létrehozására szolgál, rajzjele a következő (14. ábra).

Ezek voltak tehát a logikai alapműveletek, amelyekből bármilyen kombinált logikai függvénykapcsolat felépíthető.



13. ábra. A VAGY kapcsolat áramköri értelmezése



14. ábra. A VAGY kapu áramköri rajzjelei

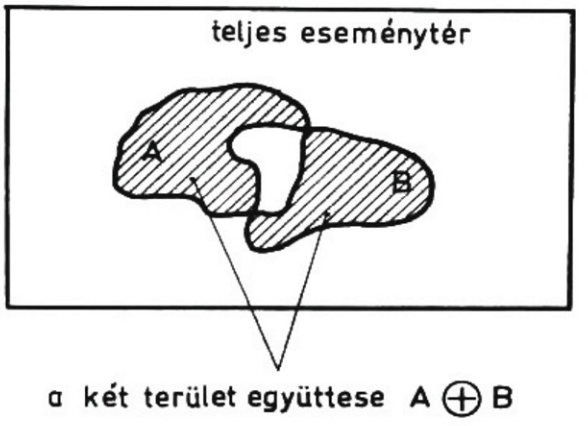
2.4.4 A KIZÁRÓ VAGY MÜVELET

Ez már nem tartozik szorosan az alapműveletek közé, mégis megemlítjük, mivel igen gyakran használják. A VAGY művelettől csak annyiban különbözik, hogy a bemenő események egyidejű IGAZ (1) állapota esetén a kimeneten HAMIS (0) logikai eredményt ad. (Kizárja a "bemenő események" egyidejű bekövetkezését.) A jelölése $\oplus$

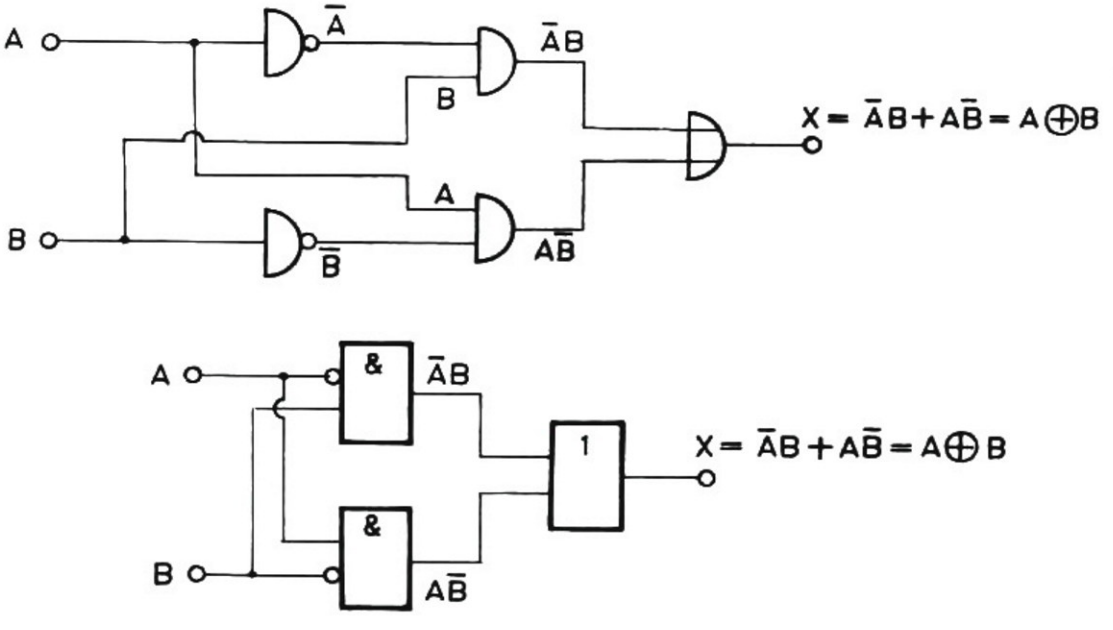
Az igazságtáblázata ennek megfelelően

bemenő események		kimenő esemény
A	B	$X = A \oplus B$
0	0	0
1	0	1
0	1	1
1	1	0

A számitógépek aritmetikai egységén belül az ugynevezett félösszeadó (lásd később!) megvalósitására használják, mivel egy helyi értéken a bináris összeadást valósitja meg tulajdonképpen. Halmazelméletileg a következőképpen képzelhető el (15. ábra).
Gyakorlatban ÉS, VAGY, illetve NEM kapukból lehet összeállitani, illetőleg ezeket tartalmazza, mivel ez már felépithető a logikai alapmüveletekből (16. ábra).



15. ábra.
A KIZÁRÓ VAGY halmazelméleti szemléltetése



16. ábra. A KIZÁRÓ VAGY megvalósitása ÉS, VAGY, NEM kapukkal
(kétféle rajzjellel bemutatva)

2.4.5 A BOOLE-ALGEBRA ALAPTÉTELEI

A Boole-algebrának sok olyan alaptétele van, amelyek lehetővé teszik a bonyolult kifejezések átrendezését és egyszerüsitését. A következőkben ezek közül sorolunk fel néhányat, a teljesség igénye nélkül. Valamennyi érvényességét halmazelméleti példák segitségével be lehet látni!

a) De Morgan tétel

A logikai sorrendben nem ez volna a legelső, de a programozási gyakorlatban ez fordul elő leggyakrabban, ezért a jegyzet szempontjából ez a legfontosabb. A nevét is ezért tartjuk célszerűnek kiirni, holott a többi tételnél ezt nem tesszük meg. Eszerint:

Bármely negált (tagadott) Boole-kifejezés egyszerüsithető, ha az ÉS müveletet VAGY-ra változtatjuk vagy forditva, és a változókat a negált értékükkel helyettesitjük.

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

Ez a tétel egyszerűen ugy is megfogalmazható, hogy a felülhuzás vonalát megszakitjuk, ha a müveleti jelet megváltoztatjuk.

b) A "nullák" (HAMIS események) közötti kapcsolattól függetlenül az eredmény mindig 0.

$$0 \cdot 0 = 0$$

$$0 + 0 = 0$$

Egy olyan tényezőt, amelyik sohasem IGAZ, mindig HAMIS (0), elektromos kapcsolásbeli szakadásnak lehet elképzelni. Ezt akár sorosan, akár párhuzamosan kötjük egy másik szakadással, az eredmény továbbra is szakadás (nyitott áramkör!).

c) A csupa "egyesből" álló bármely kombináció mindig "egyet" eredményez.

$$1 \cdot 1 = 1$$

$$1 + 1 = 1$$

Egy olyan tényezőt, amelyik mindig IGAZ (1) elektromos kapcsolásbeli rövidzárnak tekinthetünk.

d) Bármely olyan esetben, amikor az egymáshoz ÉS müvelettel kapcsolt elemek sorozatában legalább egy elem 0, a teljes kifejezés értéke is 0.

$$0 \cdot 1 = 0$$

Ha a VAGY müvelettel egymáshoz kapcsolt elemek sorozatában előfordul egy 1-es, akkor az egész kifejezés 1-gyel egyenlő, mivel az elemek közül legalább egy IGAZ, s ez kielégiti már a VAGY követelményeit.

e) Egy kifejezés logikai 1-gyel való szorzata (ÉS függvény) egyenlő önmagával:

$$A \cdot 1 = A$$

f) Egy kifejezés logikai 0-val való szorzata egyenlő nullával.

$$A \cdot 0 = 0$$

g) Egy kifejezés logikai 1-gyel való összege (VAGY függvény) egyenlő 1-gyel.

$$A + 1 = 1$$

h) Egy kifejezés logikai 0-val való összege egyenlő önmagával.

$$A + 0 = A$$

i) Egy kifejezésnek a negáltjával való szorzata egyenlő nullával.

$$A \cdot \overline{A} = 0$$

j) Egy kifejezésnek a negáltjával való összege egyenlő eggyel.

$$A + \overline{A} = 1$$

k) Egy kifejezésnek önmagával való szorzata egyenlő önmagával.

$$A \cdot A = A$$

l) Egy kifejezésnek önmagával való összege egyenlő önmagával.

$$A + A = A$$

m) Egy negált kifejezés negálása (kettős tagadás) az eredeti kifejezést eredményezi.

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

n) A két vagy több változón alkalmazott logikai VAGY, illetve ÉS művelet eredménye független a változók sorrendjétől:

$$A \cdot B = B \cdot A \; ; \; A + B = B + A$$

o) Bármely olyan kifejezés, amely ÉS és VAGY függvényt is tartalmaz, egyszerüsithető, ha vannak olyan változók, amelyek minden tényezőben, illetőleg tagban szerepelnek.

$$A\,(A + B) = A \; ; \; A + AB = A$$

Ha egy tényező és negáltja szerepel:

$$A\,(\overline{A} + B) = AB \; ; \; A + \overline{A}B = A + B$$

p) Egy kifejezés, amelyben ÉS és VAGY művelet is szerepel, egyenlő a két műveletben szereplő változók kombinációjával.

$$A \cdot (B + C) = AB + AC$$
$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

q) A két vagy több változón alkalmazott ÉS, illetve VAGY művelet eredménye független a változók csoportositásától.

$$A \cdot (B \cdot C) = A \cdot B \cdot C$$
$$A + (B + C) = A + B + C$$

2.5 KONVERTÁLÁS, SZÁMRENDSZEREK KÖZÖTTI ÁTALAKÍTÁS

Megismertük idáig a kettes, nyolcas és tizenhatos számrendszert, mint a számitógépek esetében (a decimális helyett) leggyakrabban alkalmazott számrendszereket. A különböző rendszereknek sajátos és a többi rendszerektől eltérő előnyeik vannak.

Ebben a fejezetben az átalakítások különböző fajtáival fogunk foglalkozni, hogy bármelyik számrendszer használata esetén is át tudjunk térni más számrendszerbe. Az alkalmazott módszerek hasonlósága miatt az alábbi csoportosításban célszerű az átalakításokkal foglalkozni:

- bináris, oktális, hexadecimális számrendszerbeli szám átalakitása decimálissá,
- decimális számrendszerbeli szám átalakitása binárissá, oktálissá, hexadecimálissá,
- bináris számrendszerbeli szám átalakitása oktálissá és hexadecimálissá, illetőleg viszont.

2.5.1 BINÁRIS, OKTÁLIS, HEXADECIMÁLIS SZÁMRENDSZERBELI SZÁM ÁTALAKÍTÁSA DECIMÁLISSÁ

Ez a módszer tulajdonképpen már ismert is, hiszen az egyes számrendszerek bevezetésénél már volt szó az illető számok tizes számrendszerbeli értelmezéséről. Az ott alkalmazott "helyi érték szerinti kifejtést", amelyet "polinomikus kifejtés"-nek is hivnak, általánosan használhatjuk bármely számrendszerből decimálissá való átalakitásnál. A módszer lényege, hogy a bináris, az oktális, illetőleg a hexadecimális vesszőtől kiindulva meghatározzuk az egyes pozíciók értékét (a helyi értéket), mind az egész, mind a tört jegyekét. Az egyes helyi értékek és az illető pozíción levő együttható szorzataként keletkező részletszorzatokat, részeredményeket összegezzük, és igy megkapjuk az illető számrendszerben kifejezett szám decimális megfelelőjét. Nézzünk egy-egy példát az átalakitásokra!

1. példa

$$(2) \xrightarrow{\text{egész, tört}} (10)$$

$$1011010,01_{(2)} = ?_{(10)}$$

$$1011010,01_{(2)} = 1\cdot 2^6 + 0\cdot 2^5 + 1\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2} =$$

$$64 + 0 + 16 + 8 + 0 + 2 + 0 + 0 + \frac{1}{4} = 90,25_{(10)}$$

2. példa

$$(8) \xrightarrow{\text{egész, tört}} (10)$$

$$673,25_{(8)} = ?_{(10)}$$

$$673,25_{(8)} = 6\cdot 8^2 + 7\cdot 8^1 + 3\cdot 8^0 + 2\cdot 8^{-1} + 5\cdot 8^{-2} = 6\cdot 64 + 7\cdot 8 + 3\cdot 1 + 2\cdot\frac{1}{8} + 5\cdot\frac{1}{64} =$$

$$= 443\frac{21}{64}(10) = 443,328_{(10)}$$

3. példa

$$(16) \xrightarrow{\text{egész, tört}} (10)$$

$$5BC,\ 1A_{(16)} = ?_{(10)}$$

$$5BC,\ 1A_{(16)} = 5\cdot 16^2 + 11\cdot 16^1 + 12\cdot 16^0 + 1\cdot 16^{-1} + 10\cdot 16^{-2} = 5\cdot 256 + 11\cdot 16 + 12\cdot 1 + 1\cdot\frac{1}{16} + 10\cdot\frac{1}{256} =$$

$$= 1468\frac{26}{256}_{(10)} = 1468,1016_{(10)}$$

2.5.2 DECIMÁLIS SZÁMRENDSZERBŐL BINÁRIS, OKTÁLIS, HEXADECIMÁLIS SZÁMRENDSZERBE VALÓ ÁTALAKÍTÁS

Az átalakitás módja más az egész és más a törtszámok esetében. Igy, ha az átalakitandó decimális számnak egész és törtrésze is van, külön végezzük el az átalakitást az egész és külön a törtrészre. A részeredményeket egyesitjük és igy kapjuk meg az adott decimális szám uj számrendszerbeli megfelelőjét. A módszert az alábbi decimális számon kisérjük végig!

1. példa

$183,79_{(10)} = ?_{(2)}$

Először tehát az egész részt alakitjuk át binárissá.

$183_{(10)} \xrightarrow{\text{egész}} (2)$

Az egész szám átalakitásánál a következő "ökörszabály" használatos. Az egész részt osztjuk kettővel, a hányadost az egész rész alá irjuk, a maradékot pedig egy külön oszlopba (erre lesz tulajdonképpen szükségünk). Az előbb kapott hányadost megint osztjuk kettővel és a maradékot az előbbi maradék alá irjuk. Ezt mindaddig csináljuk, amig hányadosként 0-t nem kapunk! Ezek az egymás alá irt maradékok (0 vagy 1) adják alulról felfelé összeolvasva az eredeti decimális számnak megfelelő bináris számot. (A "0" hányadossal egyidőben kapott "1"-es maradék lesz a bináris szám legnagyobb helyi értékü bitje!)

	183	:2	
	91	1	
	45	1	↑
	22	1	
	11	0	kiolvasás iránya
	5	1	
	2	1	
	1	0	
utolsó hányados	0	1	utolsó maradék
hányadosok		maradékok	
(ezeket osztjuk 2-vel)		(a bináris egész jegyei)	

$183_{(10)} = 10110111_{(2)}$

A törtrész átalakitásánál az "ökörszabály" a következő: A törtrészt szorozzuk kettővel és külön irjuk egymás alá a szorzat egész és külön a törtrészét. (A szorzat törtrésze mindig annyi jegyből fog állni. ahány jegyből a szorzandó törtrésze állt. Ha a szorzat ennél több jegyet tartalmaz, azok már az egész részt alkotják.) A következő lépésben az előzőleg megkapott törtrészt szorozzuk tovább, és az eredményt az előbbieknek megfelelően irjuk le. Ezt mindaddig folytatjuk, amig a szorzandó törtrésze "0" nem lesz, vagy a kivánt pontosságot el nem érjük! Az egymás alá irt részletszorzatok egész részeit felülről lefelé összeolvasva, megkapjuk a decimális törtrész bináris megfelelőjét.

$0,79_{(10)} \xrightarrow{\text{tört}} (2)$

Mi az átalakitást csak 8 bináris jegyig végezzük el!

	0,	79 · 2
	1	58 · 2
	1	16 · 2
	0	32 · 2
a kiolvasás iránya ↓	0	64 · 2
	1	28 · 2
	0	56 · 2
	1	12 · 2
	0	24
a szorzatok egész részei (a bináris tört jegyei)		a szorzatok tört részei (mindig csak ezeket szorozzuk kettővel)

$0,79_{(10)} = 0,11001010_{(2)}$

A két konvertálás eredményét, a bináris egész, illetőleg tört részeket most már csak egymás után kell irni, vagyis

$183,79_{(10)} = 10110111,11001010_{(2)}$

Igy tehát megkaptuk az eredeti szám értékének megfelelő bináris számot.

A bemutatott módszer helyességét az alábbiakban láthatjuk be. (Egyszerüség kedvéért csak decimális egészek binárissá való átkonvertálását mutatjuk be általános esetben.)

Tulajdonképpen adott egy N decimális szám és ennek értékét akarjuk kifejezni a kettes számrendszerben.

$$N_{(10)} = \sum_{k=0}^{n} b_k \cdot 2^k$$

$$N_{(10)} = b_n \cdot 2^n + b_{n-1} \cdot 2^{n-1} + \ldots + b_2 \cdot 4 + b_1 \cdot 2 + b_0$$

Az eredeti számot csak akkor tudjuk felirni binárisan, ha a b_k együtthatók értékét sorra meghatározzuk. Kezdjük a b_0-nél. Osszuk végig az egyenletünket 2-vel!

$$\frac{N_{(10)}}{2} = b_n \cdot 2^{n-1} + b_{n-1} \cdot 2^{n-2} + \ldots + \frac{b_2 \cdot 4}{2} + \frac{b_1 \cdot 2}{2} + \frac{b_0}{2}$$

Mivel a b_0 együttható értéke vagy 0, vagy 1 lehet csak, a $\frac{b_0}{2}$ müvelet hányadosa nem lesz egész szám, az osztást az általános képletben csak jelölni tudjuk. A b_0 az osztás maradéka.

$$\frac{N_{(10)}}{2} = b_n \cdot 2^{n-1} + b_{n-1} \cdot 2^{n-2} + \ldots + b_2 \cdot 2 + b_1 + \boxed{\frac{b_0}{2}}$$

Az előbbi számszerű példában a 2-vel való először osztás maradéka tulajdonképpen ugyanigy a b_0-t adja, vagyis a bináris egész szám utolsó helyi értéken levő együtthatóját. A bináris szám következő jegye (a következő együttható) a b_1. Ezt megint ugy kapjuk meg, hogy az előbb megmaradt egyenletet 2-vel végigosztjuk.

$$\frac{N_{(10)}}{2\cdot 2} = b_n \cdot 2^{n-2} + b_{n-1} \cdot 2^{n-3} + \ldots + b_2 + \boxed{\frac{b_1}{2}}$$

A b_1-et nem lehet osztani 2-vel, ez az osztás maradéka. Ez adja a bináris szám eggyel nagyobb pozicióban levő együtthatóját. A 2-vel való osztogatást mindaddig lehetne folytatni, amig a b_n-et kapnánk maradékul. Az eközben adódó maradékok a bináris szám jegyeit (együtthatóit) adják a növekvő helyi értékeknek megfelelően.

A decimális törtrész átalakításánál alkalmazott módszert az előbbiek alapján szintén igazolni lehet. Ezt azonban már a kedves olvasóra bizzuk!

2. példa

$183,79_{(10)} = ?_{(8)}$

Az eljárás lényege megegyezik a binárissá való átalakitásnál bemutatottal.

$183_{(10)} \xrightarrow{\text{egész}} (8)$

	183	: 8
	22	7
	2	6
utolsó hányados	0	2

a kiolvasás iránya

hányadosok (ezeket osztjuk 8-cal) | maradékok (az oktális egész jegyei)

$183_{(10)} = 267_{(8)}$

A törtrész átalakitását megint külön végezzük el. A szorzást mi csak az első négy oktális jegy eléréséig alkalmazzuk!

$0,79_{(10)} \xrightarrow{\text{tört}} (8)$

	0,	79 · 8
a kiolvasás iránya	6	32 · 8
	2	56 · 8
	4	48 · 8
	3	84

a szorzatok egész részei (az oktális tört jegyei) | a szorzatok tört részei (mindig csak ezeket szorozzuk nyolccal)

$0,79_{(10)} = 0,6243_{(8)}$

A decimális szám értéke az oktális számrendszerben tehát a következőképpen fejezhető ki:

$183,79_{(10)} = 267,6243_{(8)}$

3. példa

$183,79_{(10)} = ?_{(16)}$

Ezt az előbbiek értelemszerű alkalmazásával ugy alakithatjuk át, ha az egész részt osztjuk 16-tal.

$$183_{(10)} \xrightarrow{\text{egész}} (16)$$

	183	: 16
	11	5
utolsó hányados	0	$11_{(10)} = B_{(16)}$ ↑ a kiolvasás iránya
hányadosok (ezeket osztjuk 16-tal)		maradékok (hexadecimális számrendszerben kifejezve a hexadecimális egész jegyeit adják)

$$183_{(10)} = B5_{(16)}$$

A tört rész átalakitását pedig 16-tal való szorzással végezhetjük el. Az átalakitást az első két jegyig folytassuk!

$$0,79_{(10)} \xrightarrow{\text{tört}} (16)$$

	0,	79 · 16
A kiolvasás iránya ↓	$C_{(16)} = 12_{(10)}$	64
	$A_{(16)} = 10_{(10)}$	24
	a szorzatok egész részei (hexadecimálisan kifejezve a törtrész jegyei)	a szorzatok tört részei (ezeket szorozzuk csak 16-tal)

$$0,79_{(10)} = 0,CA_{(16)}$$

A decimális szám hexadecimálisan tehát igy néz ki:

$183,79_{(10)} = B5,CA_{(16)}$

A kettővel való osztás, szorzás általában nem okoz nagy problémát! Nyolccal és főleg tizenhattal osztani és szorozni már hálátlanabb feladat, különösen sok jegyü egész és tört részek esetén! Ezért a decimálisból oktálisba, illetőleg hexadecimálisba való átalakitás előbb bemutatott módját ritkán használjuk. (Ezzel inkább csak a módszer általánosságát akartuk szemléltetni!) Helyette a decimális számot minden esetben átalakitjuk binárissá, és ezt alakitjuk azután át oktálissá, illetve hexadecimálissá a következőkben bemutatott eljárással.

$(10) \longrightarrow (2) \longrightarrow (8)$

$(10) \longrightarrow (2) \longrightarrow (16)$

2.5.3 BINÁRIS SZÁMRENDSZERBELI SZÁM ÁTALAKÍTÁSA OKTÁLISSÁ ÉS HEXADECIMÁLISSÁ, ILLETŐLEG VISZONT

A bináris, az oktális és a hexadecimális számrendszerek nagyon egyszerűen átalakithatóak egymásba, mivel a nyolc és a tizenhat kettő egész kitevőjű hatványai ($8 = 2^3$; $16 = 2^4$). Ennek alapján 3 bináris jegy egy oktális, 4 bináris jegy pedig egy hexadecimális számjeggyel helyettesithető. Az átalakitásnál a bináris vesszőtől kiindulva kell a hármas, illetőleg a négyes csoportokat létrehozni. (A hármas csoportokat triád-nak, a négyes csoportokat tetrád-nak is hivják.) A csoportoknak megfelelő oktális, illetőleg hexadecimális számokat ez után már csak egymás mellé kell irni! Nézzünk erre példát!

$$1110111010,1010111_{(2)} = ?_{(8)}$$

$$001\ 110\ 111\ 010,\ 101\ 011\ 100_{(2)}$$

$$1\quad 6\quad 7\quad 2,\quad 5\quad 3\quad 4_{(8)}$$

A nem teljes triádokat nullákkal töltöttük fel, hogy az oktálisba való átirásuk egyértelmü legyen. Tehát

$$1110111010,1010111_{(2)} = 1672,534_{(8)}$$

Ha ugyanezt a bináris számot hexadecimálisan akarjuk felirni, akkor a bináris vesszőtől kiindulva négyes csoportokat, tetrádokat kell képezni, és azokat kell a megfelelő 16-os számrendszerbeli alakjukkal helyettesiteni.

$$1110111010,1010111_{(2)} = ?_{(16)}$$

$$11\ 1011\ 1010,\ 1010\ 111_{(2)}$$

A legelső és legutolsó tetrád hiányzó jegyeit 0-kkal kell pótolni!

$$0011\ 1011\ 1010,\ 1010\ 1110_{(2)}$$

$$3\quad B\quad A\ ,\ A\quad E_{(16)}$$

Vagyis:

$$1110111010,1010111_{(2)} = 3BA,AE_{(16)}$$

Oktálisból binárissá történő átalakitás ugyanezen az elven történik, csak akkor az oktális számrendszerben felirt minden számjegyet egy triáddal kell helyettesiteni.

$$703,62_{(8)} = ?_{(2)}$$

$$7\quad 0\quad 3\ ,\ 6\quad 2_{(8)}$$
$$111\ 000\ 011\ ,\ 110\ 010_{(2)}$$

Ezeket a triádokat ezután csak egymás mellé kell irni!

$$703,62_{(8)} = 111000011,110010_{(2)}$$

A hexadecimális alakból binárissá való áttérésnél pedig minden hexadecimális jegyet egy tetrádon kell felirni és ezeket a tetrádokat "összeolvasni".

$A81,FC_{(16)} = ?_{(2)}$

$$\begin{array}{ccccc} A & 8 & 1 & , & F \quad C_{(16)} \\ 1010 & 1000 & 0001 & , & 1111 \quad 1100_{(2)} \end{array}$$

$A81,FC_{(16)} = 101010000001,11111100_{(2)}$

Ezt a módszert nemcsak akkor alkalmazhatjuk, ha oktális, illetőleg hexadecimális számrendszerben felirt számot binárissá akarunk alakitani! A legnagyobb gyakorlati haszna abban van, hogy az ilyen módon binárissá alakitott oktális, illetőleg hexadecimális szám értékét kényelmesen meghatározhatjuk a tizes számrendszerben. Ugyanis kettő hatványait "fejből" tudni, és azokat 1-gyel vagy 0-val szorozni sokkal egyszerübb, mint nyolc, illetőleg tizenhat hatványait kiszámitani és az adott együtthatókkal még szorozgatni azt.

Igy - bár kerülő uton, de gyorsabban és kényelmesebben jutunk célhoz, mint a korábban már bemutatott "helyi értékes" módszerrel!

(16) ⟶ (2) ⟶ (10)

(8) ⟶ (2) ⟶ (10)

2.6 KÓDOLÁS. KÓDOLT DECIMÁLIS SZÁMÁBRÁZOLÁS (BCD)

Ahhoz, hogy a digitális számitógépeken adatfeldolgozást is lehessen végezni, szükséges, hogy numerikus adatokon kivül a gép számára érthető formában közölni tudjunk karakterenként szöveges információt is. Mivel a számitógép az információt bináris elemek, a 0 és az 1 segitségével ábrázolja és dolgozza fel, az általunk feldolgozásra szánt karaktereket is ilyen módon kell megadni. Ezt ugy végezhetjük el, hogy minden egyes karakternek megfeleltetünk egy 0-kból és 1-esekből álló "bit"-sorozatot. A bitsorozat hosszának legalább akkorának kell lenni, hogy az általunk használni kivánt összes karaktert egyértelmüen ki tudjuk rajta fejezni. Vagyis e bitsorozat modulusa vagy ugyanannyi, vagy nagyobb, mint ahány karaktert ábrázolni akarunk. Egy bitsorozatnak az általunk használt karakterekhez való hozzárendelése az un. kódolás.

Általános értelemben kódoláson egy fogalomnak, egy közölnivalónak szemléletes, tömör, mindenki számára érthető formában való kifejezését értjük. Gondoljunk itt csak a közuti jelzőtáblákra! Vagy a 17. ábrán látható, nemzetközileg elfogadott egységes képjelekre! A mi esetünkben a közölnivalók az egyes karakterek, a könnyü érthetőséget pedig a gép szempontjából kell figyelembe venni. Minden egyes karakterhez többféleképpen rendelhetünk bitsorozatot. A lényeg azonban az, hogy ha egyszer már eldöntöttük ezt a "bitsorozat hozzárendelés"-t, a kódolást, akkor minden esetben ezt is kell alkalmazni, akár mi közlünk valamilyen információt a géppel, akár a gép válaszol. A kódolás egyértelmüvé, mindenki számára megismerhetővé tétele érdekében a különféleképpen kódolt karaktereket ugynevezett "kódtáblázatok"-ban rögzitik (lásd: EBCDIC, ASCII stb. táblázatok a 2.7.2 fejezetben).

17. ábra. Kódok

Vannak olyan számitógépek, amelyek nemcsak az előző fejezetben leirt bináris müveletvégzésre alkalmasak, hanem képesek decimális számokkal is dolgozni. Ehhez azonban a decimális számokat a gép számára érthető formában, kódolva kell megadni. A decimális számrendszerben előforduló 10 különféle számjegy (0...9) bináris leirásához legalább négy pozició szükséges. A négy pozición a decimális számok kódolása többféleképpen történhet. A kódolás egyes fajtáinak elemzése, előnyeik, hátrányaik felsorolása már nem feladatunk, de nézzük meg, hogy a négy bit segitségével hogyan lehet decimális számokat kódolva kifejezni.

4 bittel kifejezhető számok	8-4-2-1 kód (BCD kód)	2-4-2-1 kód (Aiken-kód)	3 többletes kód (Stiebitz-kód)
0000	0	0	
0001	1	1	
0010	2	2	
0011	3	3	0
0100	4	4	1
0101	5		2
0110	6		3
0111	7		4
1000	8		5
1001	9		6
1010			7
1011		5	8
1100		6	9
1101		7	
1110		8	
1111		9	

A kódok elnevezésében szereplő számok az egyes pozíciók értékeit jelentik.

Egy több jegyü decimális szám kódolt felirása ugy történik, hogy minden egyes számjegyet az adott kódnak megfelelő négy bites csoporttal külön-külön fejezünk ki, és ezeket a csoportokat azután egymás után irjuk.

$$\begin{array}{lcl} & & 2^3 2^2 2^1 2^0 \;\; 2^3 2^2 2^1 2^0 \\ 28_{(10)} & \Longrightarrow & \underbrace{0\;0\;1\;0}_{2}\;\;\underbrace{1\;0\;0\;0}_{8} \quad \text{(BCD 8-4-2-1 kód)} \end{array}$$

Mivel ebben az esetben a decimális számnak csak a jegyeit irjuk fel binárisan és az eredeti szám értékét nem alakitjuk át a neki megfelelő bináris értékké, ezt az ábrázolást binárisan kódolt decimális vagy más néven B C D (Binary Coded Decimal) számábrázolásnak hivjuk. A bemutatott néhány kód közül általában a 8-4-2-1 kód használatos és ha nincs külön utalás rá, akkor a BCD elnevezés alatt ilyen módon kódolt számot kell érteni.

Vigyázat! A BCD számábrázolás nem azonos a bináris ábrázolással! Bár ugyanolyan számjegyekből (0 és 1) áll, de a használt számrendszer alapszáma itt nem változik meg, továbbra is a 10-es számrendszeren belül maradunk. Csak a számjegyek felirási módja, a kódolása lett más! A bináris számrendszerben a $2^3 = 8$ helyi értékü pozicióról balra a $2^4 = 16$ helyi értékü pozició áll. A BCD ábrázoláskor viszont ismét a $2^0 = 1$ értékü pozició következik (az eggyel magasabb helyi értékü decimális számjegy legalsó bitje)!

$$\begin{array}{lcl} & & 2^4 2^3 2^2 2^1 2^0 \\ 28_{(10)} & \Longrightarrow & 1\;1\;1\;0\;0_{(2)} \\ & & 16+8+4 = 28 \end{array}$$

A decimális aritmetika a decimális helyi értékeknek megfelelően csoportonként kezeli a kódolt számot, azaz csoportonként elvégzi a kijelölt bináris müveletet!

$$85 + 29 = ?$$

$$\begin{array}{rcl} 85_{(10)} & = & 1000\;0101_{(BCD)} \\ \underline{+\;29_{(10)}} & = & \underline{+0010\;1001_{(BCD)}} \\ 114_{(10)} & & 1010\;1110 \end{array}$$

A kapott eredmények azonban a használt BCD kódban (8-4-2-1 kódban) nem értelmezhetőek! Ilyen bitkombinációkhoz ott nem tudunk decimális értéket rendelni. Az 1010 .... 1111 alakokat a decimális érték számolása során ki kell hagyni, "át kell ugorni". Mivel 6 bitkombinációhoz nem rendeltünk értéket, ezért minden olyan eredményt, amely 1001-nél (9) nagyobb, 0110-val (6-tal) korrigálni kell (18. ábra). Ezt a korrekciót a fenti példában is elvégezve, már helyes eredményt kapunk. (Természetesen ilyenkor a keletkező átvitelt is figyelembe kell venni!)

$$\begin{array}{c} 1 \quad\;\; 1 \\ 1010 \;\; 1110 \\ +0110 \; +0110 \\ \hline \underbrace{1}_{1}\;\underbrace{0001}_{1}\;\underbrace{0100}_{4_{(10)}} \text{(BCD)} \end{array}$$

Decimális számok		BCD számok	
0		0 0 0 0	
1		0 0 0 1	
2		0 0 1 0	
3		0 0 1 1	
4		0 1 0 0	
5		0 1 0 1	
6		0 1 1 0	
7		0 1 1 1	
8		1 0 0 0	
9		1 0 0 1	
		~~1 0 1 0~~	BCD számoláskor tiltott bináris értékek
		~~1 0 1 1~~	
		~~1 1 0 0~~	
		~~1 1 0 1~~	
		~~1 1 1 0~~	Összeadáskor ezeket ki kell hagyni, 6-tal többet kell lépni!
		~~1 1 1 1~~	
1 0	0 0 0 1	0 0 0 0	
1 1	0 0 0 1	0 0 0 1	
1 2	0 0 0 1	0 0 1 0	
1 3	0 0 0 1	0 0 1 1	
1 4	0 0 0 1	0 1 0 0	

18. ábra. Számolás BCD kódban

2.7 ADATÁBRÁZOLÁS

A különféle információk ábrázolására rendelkezésre álló hely hossza és az adott hosszon az ábrázolás szabálya más és más különféle adatfajták esetén. Az adatok, fajtájuk szerint, attól függően hogy mit ábrázolnak, három csoportba sorolhatók. A csoportok a következők:

numerikus adatok (operandusok)

- csak előjeles számokat ábrázolnak + 23; 0; -273,5; 72 stb.

alfanumerikus adatok (operandusok)

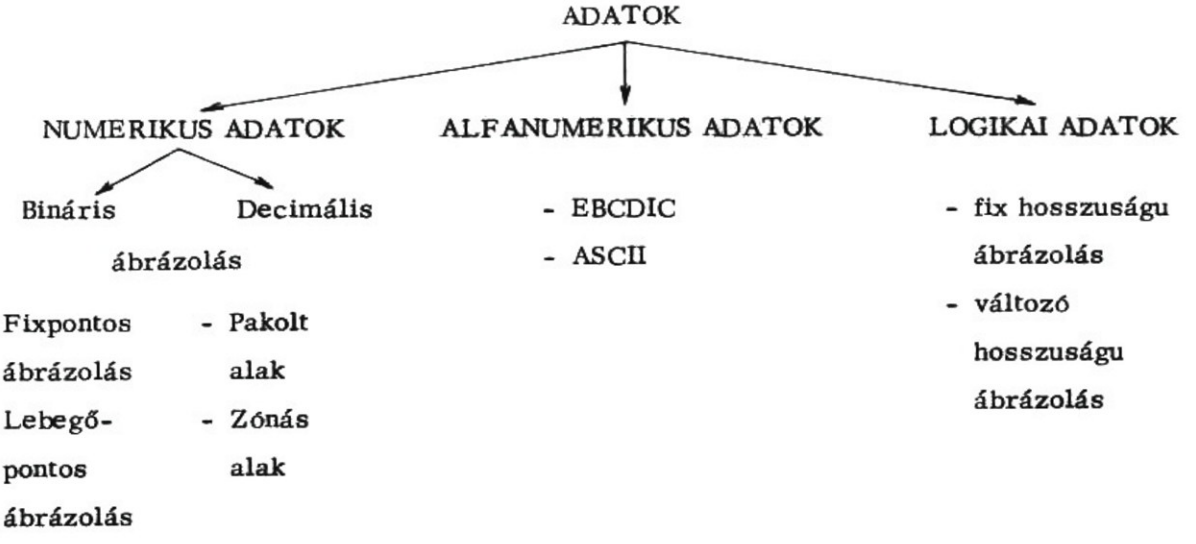
- számjegyeket, betűket, különféle írásjeleket ábrázolnak A; 8; ?.,; $ ± %

logikai adatok (operandusok)

- logikai változót, illetve logikai művelet eredményét ábrázolják

igen ⟶ 1

nem ⟶ 0



2.7.1 NUMERIKUS ADATOK ÁBRÁZOLÁSA

Numerikus adatok ábrázolása, tehát amikor csak számokat és az előjelüket akarjuk ábrázolni, kétféleképpen történhet:

- binárisan és
- decimálisan

2.7.1.1 Bináris ábrázolás

Számok bináris ábrázolásakor a tizes számrendszerről a kettes - bináris - számrendszerre térünk át, vagyis a teljes tizes számrendszerbeli szám értékét konvertáljuk, azaz a szám értékét kettő hatványaiból rakjuk össze.

$$
\begin{array}{lccccc}
 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
20_{(10)} \longrightarrow & 1 & 0 & 1 & 0 & 0_{(2)} \\
 & | & & | & & \\
 & 16 & + & 4 & = & 20
\end{array}
$$

A bináris ábrázolás kétféleképpen valósitható meg:

- fixpontosan és
- lebegőpontosan.

1. Fixpontos számábrázolás

Ez az ábrázolási forma a legelterjedtebb. Fixpontos aritmetikai egysége majdnem minden számitógépnek van. Ez minden fixpontos (fixvesszős) operandust egész számként kezel, és ennek megfelelően dolgoz is fel. A fixpontos ábrázoláskor tehát a numerikus adatokat egész számoknak kell tekinteni, mert csak azok ábrázolhatók, törtek nem. Tizedespont vagy tizedesvessző ábrázolására és a számolás közben figyelemmel kisérésére nincsen lehetőség. Az elnevezés is ebből következik, vagyis, hogy a tizedesvesszőt (tizedespontot) az operandus egy rögzitett, fix helyére kell képzeletben tenni. Ez a fix pont (vessző) általában a legutolsó ábrázolt számjegy után következik, oda "képzeli" azt a gép. Törtrészt is tartalmazó számból tulajdonképpen kétféleképpen is készithető fixpontos ábrázoláshoz szükséges egész. Például vegyük a 328,59-et. A legegyszerübb, hogy a törtrészt egyszerüen elhagyják, igy eleve tekintélyes hiba is jelentkezik és:

328,59 ⟶ 328 kerül ábrázolásra.

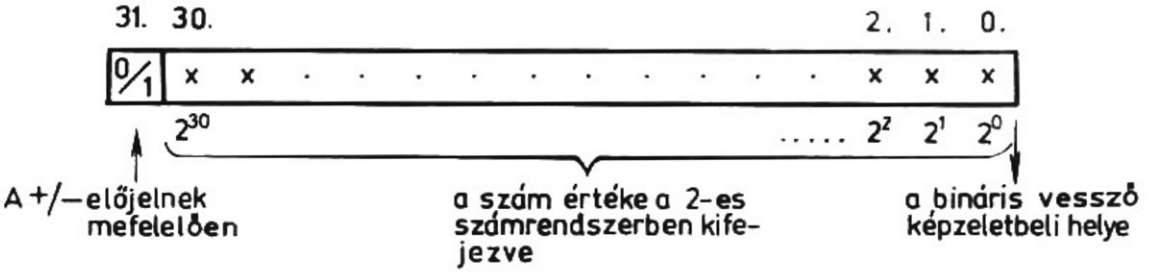
A másik módszer, hogy a tizedesvesszőt hagyják el, és az eredeti szám utolsó ábrázolandó jegye után képzelik:

328,59	32859(,)
0,072	0072(,)
15,80	1580(,)
659,0	6950(,)

Ez a gyakorlatban elterjedt módszer azt eredményezi, hogy az eredmény számjegyeit tekintve helyes lesz, de a nagyságrend az már nem! A felhasználónak vagy a programozónak kell tehát az eredményt megfelelően korrigálni. (El kell dönteni, hogy milyen nagyságrendü számról, eredményről van szó, hová kell elképzelni, vagy hová kell kinyomtatás előtt elhelyezni a tizedesvesszőt. Például a 7180-at mint 7180-as sorszám, vagy 71,80 Ft, vagy 718,0 m, vagy 7.180 t értelmezze és nyomtassa ki!)

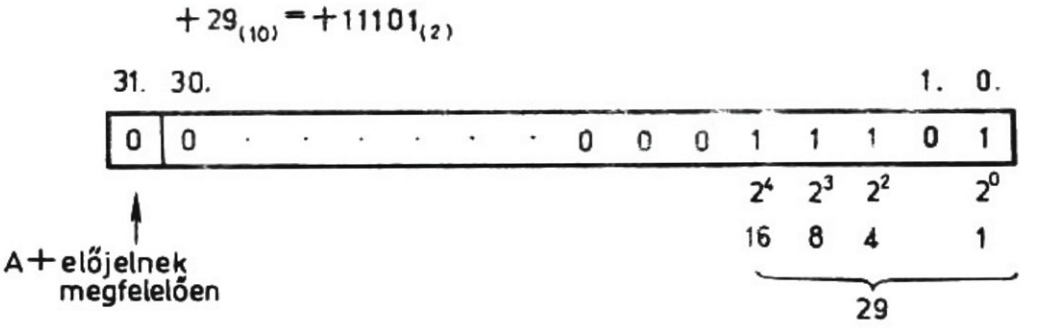
Az ábrázoláshoz rendelkezésre álló hosszuság kötött, minden gépnél előre (gépkönyvben) adott. Ez általában a gép által használt szó hosszával egyezik meg. Egyes esetekben lehetőség van a szóhossz felének (félszavas ábrázolás) vagy kétszeresének (duplaszavas ábrázolás) használatára is. Igen sok gépnél használatos a 32 bites (szavas) ábrázolási forma (32 bit = 1 szó).

A 32 bitből egy, a legnagyobb helyi értékü, az előjel ábrázolására van fenntartva, a többi pozicióhoz jobbról kezdve a kettes számrendszer helyi értékei vannak hozzárendelve. Ezek segitségével történik a szám értékének ábrázolása a kettes, vagyis bináris számrendszerben!



19. ábra. Fixpontos ábrázolási forma (32 pozicion = 1 szóhosszuságon)

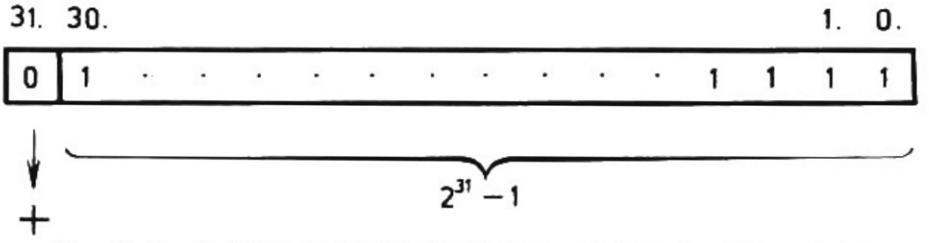
Az előjelábrázolás a ma használatos gépek legtöbbjénél az un. 2-es komplemens vagy más néven komplemens kódban történik. Pozitiv szám esetén ez azt jelenti, hogy az egész szám abszolut értékét kifejező bináris szám jegyei töltik ki a pozíciókat jobbról kezdve, mintha a képzeletbeli bináris vessző az ábrázolásra szolgáló pozíciók után következne. A bináris szám jegyei előtt fennmaradó összes pozícióba $0_{(2)}$ kerül. Igy automatikusan a 31., az előjelet ábrázoló pozicion is $0_{(2)}$ bit fog szerepelni, ami itt megegyezés szerint a pozitiv előjelet jelenti.



20. ábra. Pozitiv szám ábrázolása fixpontosan 32 pozicion

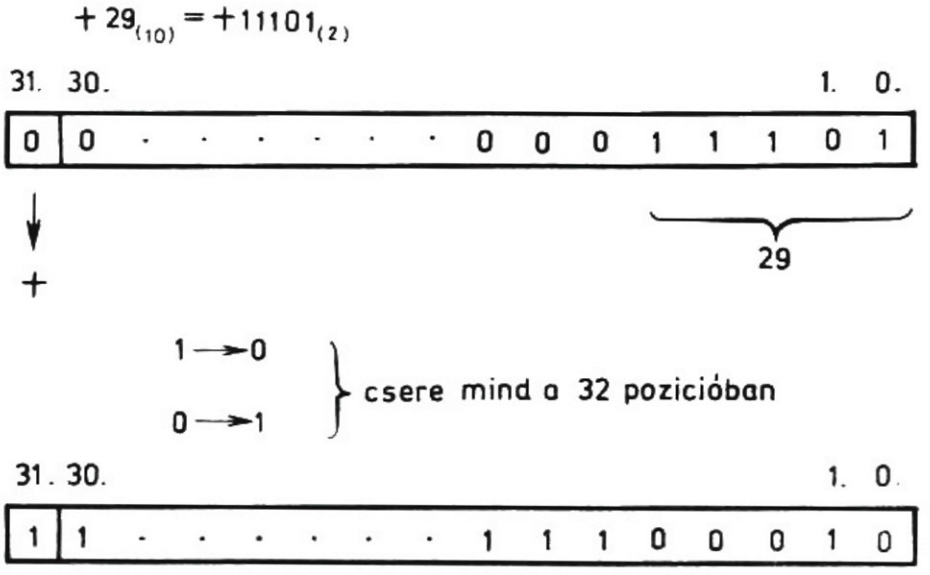
A legnagyobb pozitiv szám, amit ilyen módon 32 biten, fixpontosan ábrázolni tudunk az, aminek bináris alakja 31 db $1_{(2)}$-est tartalmaz. Ennek értéke pedig ismert módon eggyel kisebb, mint a 31 pozíciós eszköz modulusa.

$$N_{max} = M - 1 = R^n - 1 = 2^{31} - 1 = 2\ 147\ 483\ 647$$



21. ábra. A legnagyobb ábrázolható pozitiv szám 32 pozición

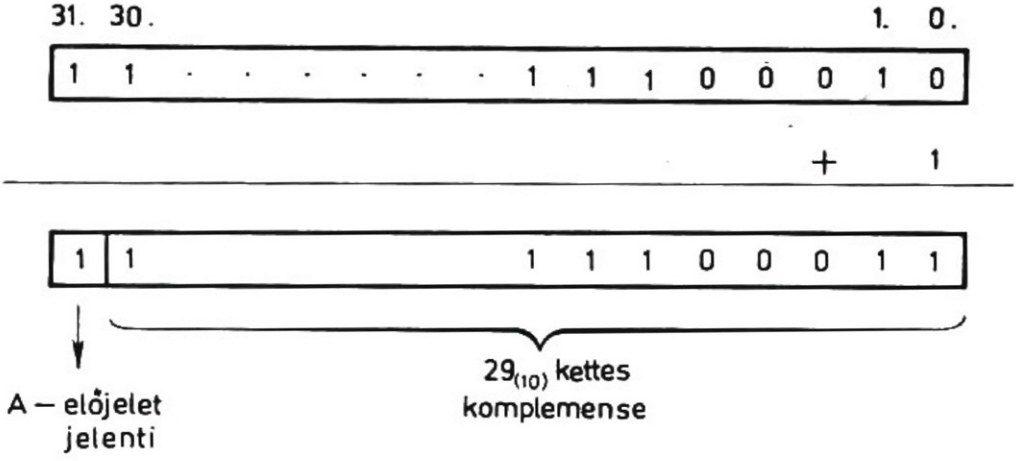
Negativ szám ábrázolásakor első lépésként a számnak megfelelő pozitiv számot ábrázoljuk. Utána invertáljuk az egészet, vagyis minden egyes pozicióban az ott levő bitet ellenkezőjére változtatjuk. Ábrázoljuk e szerint a $-29_{(10)}$-et!



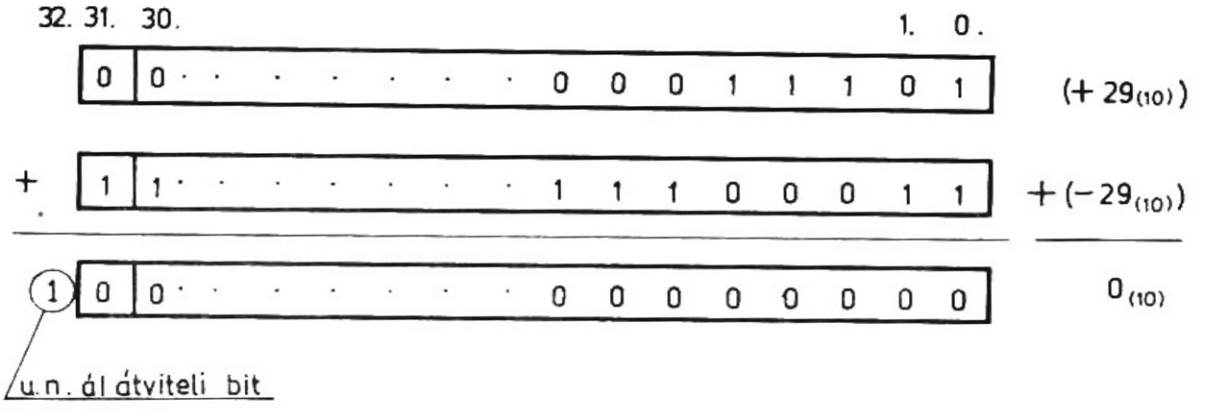
22. ábra. Negativ szám fixpontos ábrázolásának első lépése

Ezután a legalacsonyabb helyi értéken 1-et binárisan hozzáadunk, és igy megkapjuk az eredeti szám un. kettes komplemensbeli megfelelőjét (23. ábra).

A legnagyobb helyi értéken, a 31-ediken levő 1-es bit jelzi az ábrázolt fixpontos operandus negativ előjelét. Ezt az invertálás minden esetben biztositja, hiszen a 31-edik pozición csak előjel lehet és oda előzőleg a pozitiv szám ábrázolása miatt $0_{(2)}$ került. Ha próbaképpen összeadjuk a $+29_{(10)}$ és a $-29_{(10)}$ kettes komplemens kóddal kifejezett fixpontos alakját, a matematikailag helyes eredményt, a $0_{(10)}$-át kapjuk meg (24. ábra).



23. ábra. Negativ szám ábrázolási formája fixpontosan (második lépés)



24. ábra. A $0_{(10)}$ keletkezése fixpontos ábrázoláskor

A gép által még használt legelső pozició előtt, a 32-diken keletkező $1_{(2)}$-es, az un. "ál átviteli bit" kettes komplemens kód esetén figyelmen kivül hagyható, mivel az eredmény helyes.

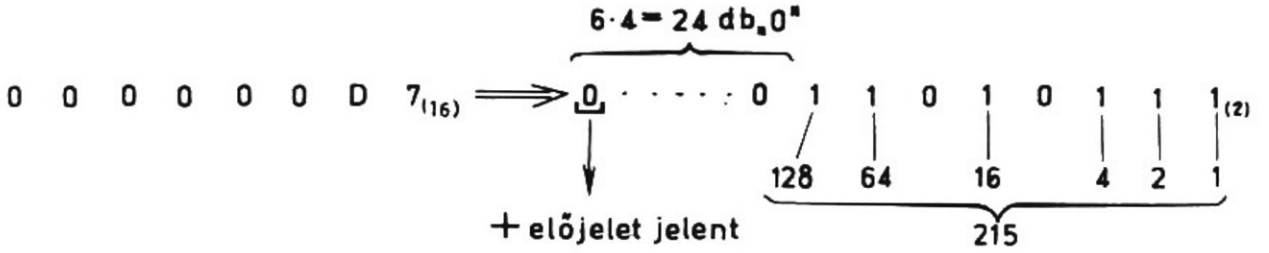
A bináris formában levő fixpontos operandust az egyszerüség, a könnyebb kezelhetőség érdekében felirhatjuk rövidebben is, a hexadecimális számrendszer segitségével. Igy nem kell állandóan 32 bináris jegyet kiolvasni, kimondani vagy leirni, hanem csak 8 hexadecimálisat. (Ezt a rövidebb felirási módot a gép is használja és meg is érti.) A hexadecimális alakba történő átalakitás itt is a tetrádok segitségével történik. A $+29_{(10)}$ fixpontos alakja tehát röviden igy irható:

$$0000001D_{(16)}$$

A $-29_{(10)}$-é pedig az alábbi módon:

$$FFFFFFE3_{(16)}$$

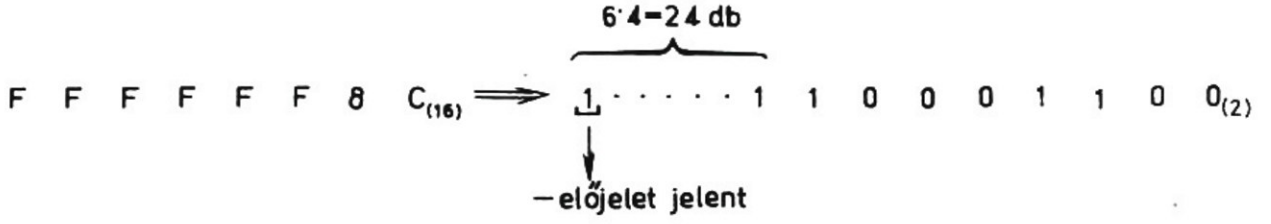
Ha a fixpontos operandust, eredményt kell értelmezni, a bináris alak 31. pozíciójában levő bit egyrészt megadja a szám előjelét, másrészt megszabja a visszaalakitás módját is. Ugyanis, ha ott $0_{(2)}$ szerepel, pozitiv számról lévén szó, a pozíciókhoz rendelt helyi értékek alapján kell a szám decimális értékét meghatározni. (Vagyis 2 megfelelő hatványait össze kell adni.)



25. ábra. Fixpontosan ábrázolt pozitiv szám

Tehát $+215_{(10)}$ volt ábrázolva 32 biten, fixpontosan, kettes komplemens kódban.

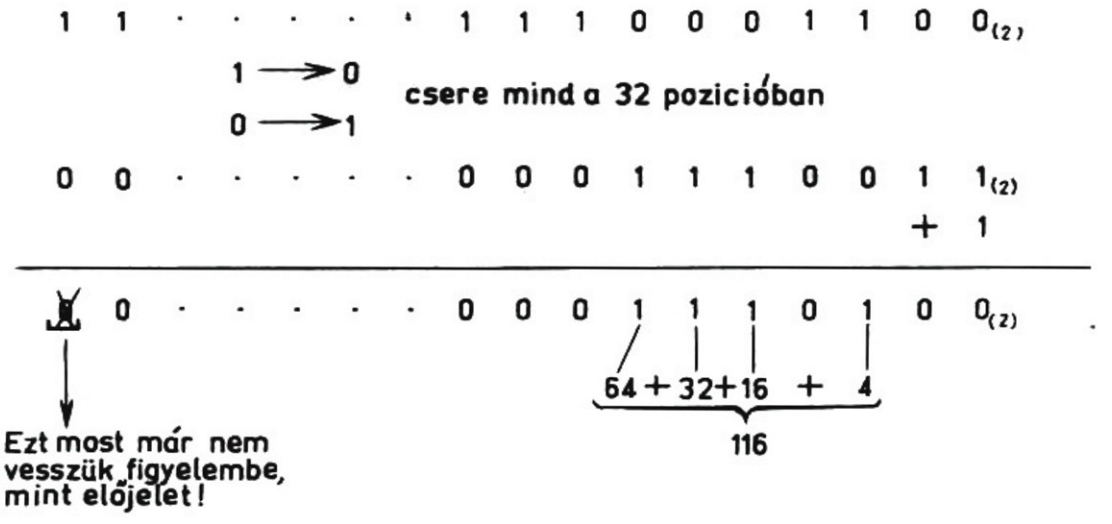
Ha a 31. pozición $1_{(2)}$-es bit áll, az negativ számra utal. Legyen az értelmezendő eredményünk a következő:



26. a. ábra. Fixpontosan ábrázolt negativ szám

Kettes komplemens kódban történt ábrázolás esetén (ami egy gépnél adott, tehát ismert dolog) a negativ szám visszaalakitása teljesen hasonló módon történik, mint az ábrázolás! Vagyis először invertálni kell minden egyes pozició tartalmát, utána pedig a legkisebb helyi értéken 1-et hozzá kell adni (tömören: $\overline{A} + 1$). Az oda- és visszaalakitásnál is egyaránt alkalmazható módszer a kettes komplemens egyik előnye. Az invertálás és az 1 hozzáadása után kapott abszolut érték (tulajdonképpen egy pozitiv szám) nagyságát a megfelelő pozíciók helyi értékeinek összeadásával kapjuk meg.

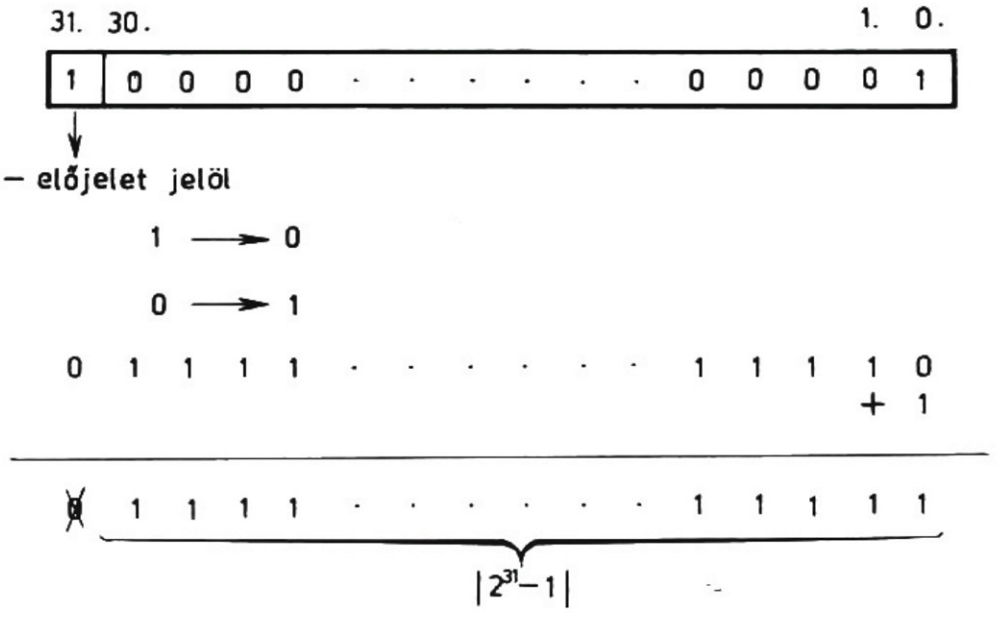
Az ábrázolt szám ezek szerint $-116_{(10)}$ volt.



26. b. ábra. Fixpontosan ábrázolt negativ szám értelmezése

Most nézzük meg, melyik az a szám, amelyik a következőképpen néz ki:

$80000001_{(16)} = ?$



27. ábra. A legnagyobb ábrázolható negativ szám 32 pozición

Ez pedig nem más, mint a 32 biten ábrázolható legnagyobb abszolut értékü szám, negativ előjellel. Fixpontosan, 32 biten, tehát azok az egész számok fejezhetők ki, amelyek az alábbi tartományba esnek:

$$-(2^{31} - 1) \leq N \leq +(2^{31} - 1)$$

$$-2\ 147\ 483\ 647 \leq N \leq +2\ 147\ 483\ 647$$

Egyes gépek, mint már korábban is szó volt róla, ettől eltérő hosszat szabnak meg a fixpontos ábrázolás számára. Ekkor természetesen megváltozik a maximálisan és minimálisan kifejezhető számok, vagyis a számtartomány értéke. Például a 16 bites - félszavas - ábrázolás esetén az előjel mellett már csak 15 bináris számjegy adható meg, az ezzel kifejezhető érték igy $-(2^{15} - 1) = -32767$ és $+(2^{15} - 1) = 32767$ közé kell, hogy essen. Az előjeles számok komplemens kódban történő ábrázolásának előnye, hogy igy minden müvelet visszavezethető az előjelbittel együtt elvégzett összeadásra, illetve kivonásra.

A +, - müveleti jel és az előjelbit ismert szabályok szerinti felcserélhetősége miatt a számitógépben vagy csak összeadó vagy csak kivonó áramkörre van szükség. Csak összeadó áramkör esetén a kivonás helyett végzett összeadást az alábbi módon értelmezhetjük:

$$18-5 = (+18) - (+5) \longrightarrow (+18) + (-5)$$

$$-23-2 = (-23) - (+2) \longrightarrow (-23) + (-2)$$

ezeket kell előjelükkel együtt ábrázolni és velük az összeadást elvégezni.

Példaképpen végezzük el a fenti egyik müveletet 32 biten, kettes komplemens segitségével, összeadó egységet feltételezve!

Először ábrázolni kell a $+18_{(10)}$-at és a $-5_{(10)}$-öt.

$+18_{(10)} = +\ 10010_{(2)}$ ábrázolása 32 biten

00.....010010

$-5_{(10)} = ?$

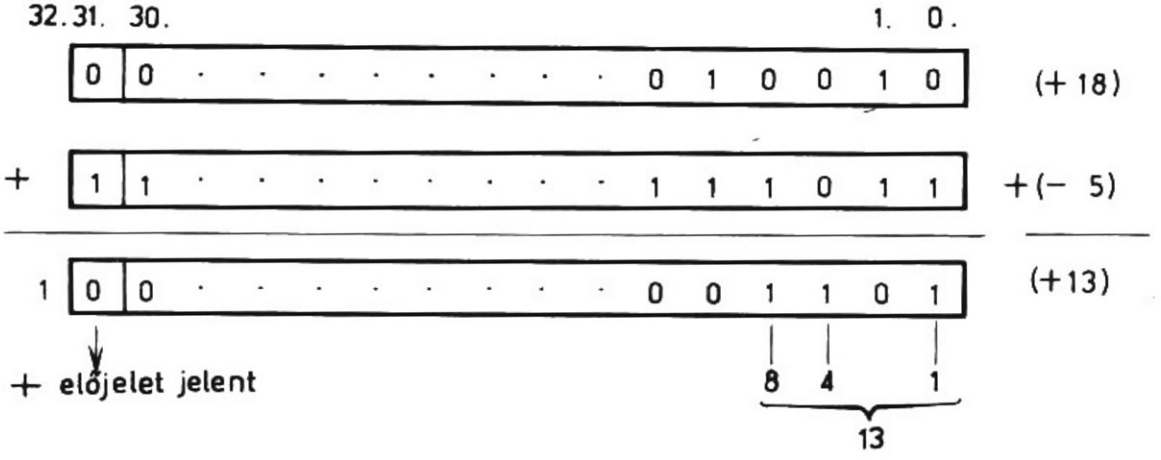
$+5_{(10)} = 101_{(2)}$ ábrázolása 32 biten

00.....000101

invertálás (1 ⟶ 0; 0 ⟶ 1 csere) után a legkisebb helyi értéken $1_{(2)}$-et hozzáadunk:

```
11.........111010
                + 1
-----------------
11.........111011
```

Ez jelenti kettes komplemens kódban a $-5_{(10)}$-öt. A két fixpontos operandus között a kijelölt összeadást elvégezve:



28. ábra. Müveletvégzés komplemens kódban ábrázolt fixpontos operandusokkal

A 32 biten keletkező eredmény algebrailag helyes, a 32. pozicióban megjelenő 1-es bit kettes komplemens esetén figyelmen kivül hagyható!

Ha két olyan nagy számot vonunk össze, hogy az eredmény már nem fér el (nem ábrázolható) az adott hosszon, pl. 31 biten, un. tulcsordulás lép fel. Ez az eset csak azonos előjelü (pozitiv vagy negativ) számok összevonásakor léphet fel, hiszen az eredmény csak akkor lesz nagyobb, mint bármelyik összevonandó. A tulcsordulást az előjelbit indokolatlan megváltozása jelzi:

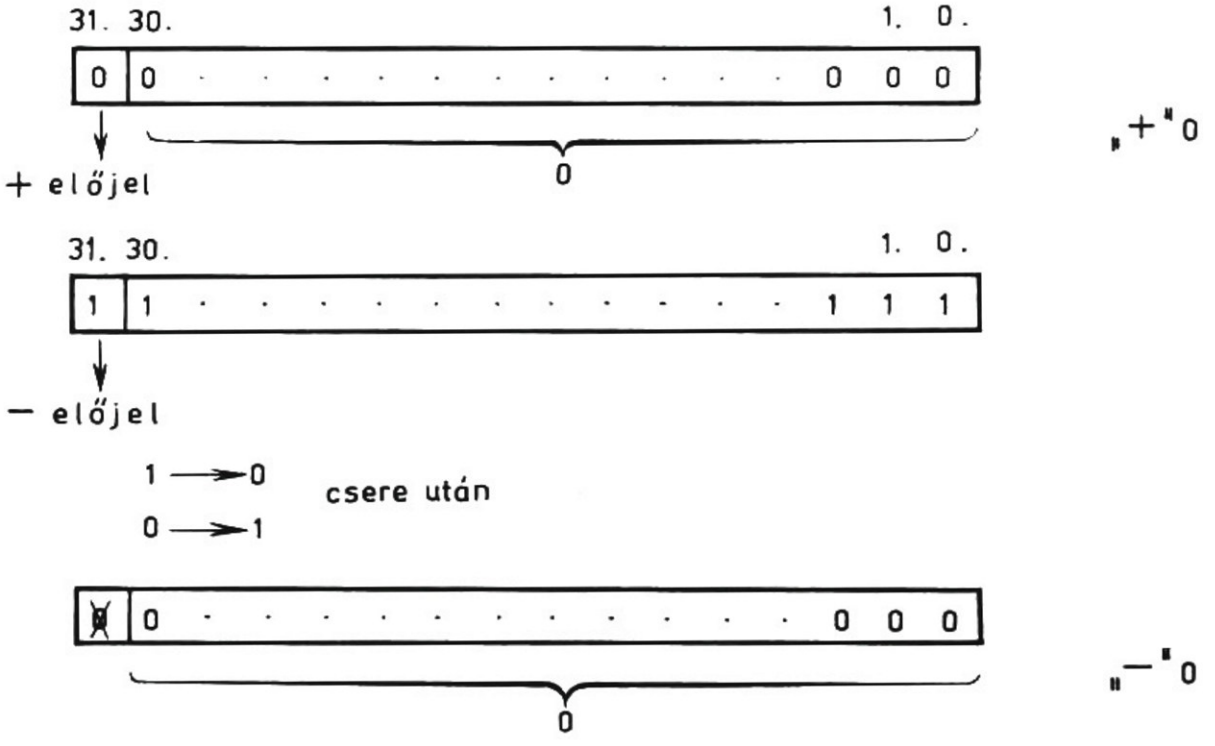
```
 0010......0100101        (+.....)
+0110......0010010      + (+.....)
----------------
 1000......0110111
 ↓
 - előjelet jelent.
```

A kapott eredmény negativ, holott két pozitiv operandus összeadásakor ez nem léphetne fel. Az oka, hogy az eredmény bináris jegyei már nem fértek el 31 biten, "tulcsorultak" a 32. pozicióra, és ez okozta az előjel indokolatlan megváltozását. Az előjelbitek azonosságát a gép figyelemmel kiséri, és ha tul-

csordulás miatt az megváltozik, jelez és leáll. Erre egyébként a felhasználónak kell ügyelnie, tudván, hogy az adott hosszon mekkora (rész)eredmény ábrázolható még tulcsordulás nélkül. Ha mégis szükség volna az ábrázolhatóság határát tullépő (rész)eredményre, akkor az operandust lebegőpontosan kell rögziteni.

Korábban előjeles számok fixpontos ábrázolásához az un. 1-es komplemens vagy más néven inverz kód is használatos volt. Ez pozitiv számok esetében semmiben sem tér el a 2-es komplemenssel történő ábrázolástól, vagyis a legnagyobb helyi értéken $0_{(2)}$ bit jelzi a pozitiv előjelet és utána a szám értékét kettő hatványai segitségével fejezzük ki.

Negativ számok ábrázolásánál is nagy a hasonlóság! Csak a 2-es komplemenssel ellentétben a legkisebb helyi értéken nem kell még $1_{(2)}$-et hozzáadni és emiatt a visszaalakitásnál először $1_{(2)}$-et levonni! A képzési és visszaalakitási folyamat igy rövidebb és egyszerübb ugyan, de van egy jelentős hátránya, mégpedig az, hogy két alak is ugyanazt, a 0-t fejezi ki.

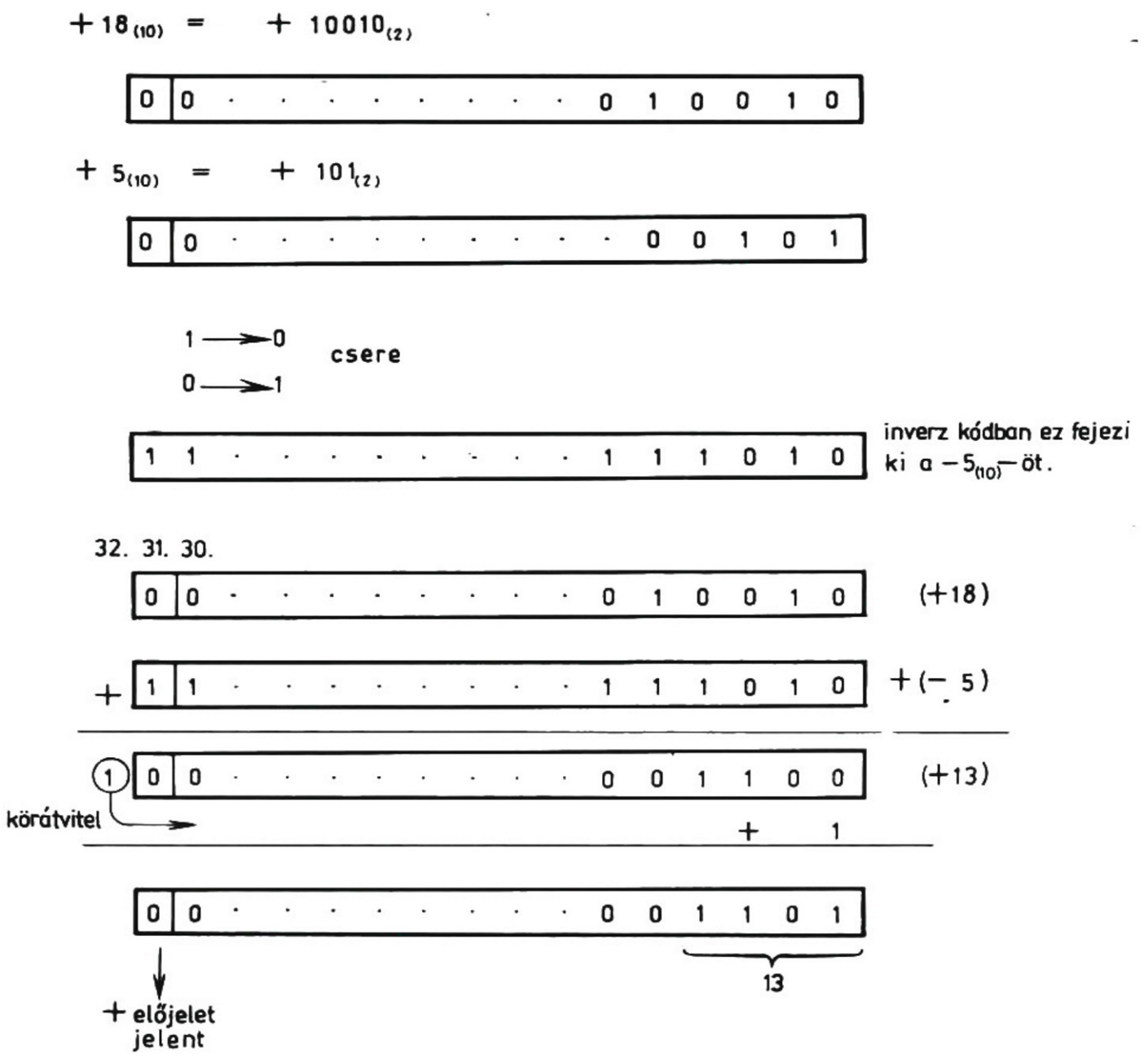


29. ábra. A 0 kétfajta ábrázolási formája inverz kódban

Tehát a "+" 0 ábrázolása nem egyezik meg a "-" 0 alakjával. Ez az oka, hogy inverz kódban ábrázolt számokkal végzett müveleteknél a 0-n való áthaladáskor a helyes értéknél 1-gyel kisebb eredményt kapunk. Az eredmény korrigálását az ilyen esetekben a legmagasabb helyi érték előtt automatikusan keletkező un. átviteli bit hozzáadásával végzi el a gép. Ezt a pozíciót ilyenkor külön kell a gépnek figyelnie. Az itt megjelenő $1_{(2)}$-essel történő korrigálást hivják körátvitelnek.

Példaképpen végezzük el inverz kód segitségével a kettes komplemensnél már bemutatott összevonást!

$$18 - 5 = (+18) - (+5) \longrightarrow (+18) + (-5)$$



30. ábra. Müveletvégzés inverz kódban ábrázolt fixpontos operandusokkal

A körátvitel elvégzése előtt az eredmény már előjelhelyes, de értéke eggyel kisebb, mint ténylegesen és ezt kell korrigálni a 32. pozicióban megjelenő "átvitel bit"-tel. Inverz kódban történő müveletvégzésnél is fellép tulcsordulás, ha az ábrázolható számtartományt átlépjük.

A 32. pozició külön figyelésére és az esetleges körátvitel kiküszöbölésére hozták létre és alkalmazzák elterjedten a 2-es komplemens kódot, ahol már csak egy alak szolgál a 0 kifejezésére. Ennek igazolását az olvasóra bizzuk!

2. Lebegőpontos számábrázolás

A fixpontos operandusoknak azt a hátrányát, hogy az ábrázolás adott hossza miatt csak korlátozott nagyságu és csak egész számok ábrázolhatók, a lebegőpontos számábrázolás küszöböli ki. Lebegőpontos ábrázolással a számtartomány lényegesen megnő és tört szám is ábrázolható. Ezért elsősorban

müszaki-tudományos számitásoknál használják. Tulajdonképpen minden szám felirható egy együttható és egy hatvány szorzataként, ahol az együttható tartalmazza a szám jegyeit, a hatvány pedig a szám értékét adja meg. Általános formában ez a következőképpen irható fel:

$$A = X \cdot B^y$$

A - az eredeti szám

X - az együttható, ennek tört része az un. mantissza

B - a hatvány alapja

y - a hatvány kitevője, az un. karakterisztika

Például a 419 is felirható sokféleképpen:

$$\begin{aligned} 419 &= 419 \cdot 10^0 \\ &= 41,9 \cdot 10^{+1} \\ &= 4,19 \cdot 10^{+2} \\ &= 0,419 \cdot 10^{+3} \ * \\ &= 4190 \cdot 10^{-1} \\ &= 41900 \cdot 10^{-2} \end{aligned}$$

Ez az un. féllogaritmikus felirási mód bármelyik számrendszerben használható. Ha az alakitásnál az eredeti számot az együtthatóban valódi törtként (tehát 0 egészet tartalmazó számként) fejezzük ki ugy, hogy a törtrész értékes jeggyel kezdődjön, megkapjuk az eredeti szám un. normálalakját.** (Lásd a fenti példában a *-gal megjelölt formát!)

A szám eredeti értékét, vagyis a nagyságrend megváltozását, az utána irt hatvány segitségével tüntetjük el. A hatvány alapja attól függően változik, hogy melyik számrendszerben - 10, 2, 16 - végeztük el a normalizálást, vagyis "milyen vesszőt" - tizedest, binárist vagy hexadecimálist - toltunk-e el. A hatvány kitevőjének nagysága a vessző eltolásának mértékét, "a lépések számát", az előjele pedig az eltolás irányát adja meg. Az eredeti vessző balra léptetésénél a kitevő előjele pozitiv, a jobbra léptetésnél a kitevő előjele negativ.

$419_{(10)}$ normálalakja $0,419_{(10)} \cdot 10^{+3}$
3 léptetés balra

$4,19_{(10)}$ normálalakja $0,419_{(10)} \cdot 10^{+1}$
1 léptetés balra

$0,00419_{(10)}$ normálalakja $0,419_{(10)} \cdot 10^{-2}$
2 léptetés jobbra

$0,419_{(10)}$ normálalakja $0,419_{(10)} \cdot 10^0$

** Ez a normálalak csak a nevében egyezik meg az elemi matematikában szereplő normálalakkal, ahol az együttható 1 és 10 közé esik (pl. a tizes alapu logaritmusnál).

További példák a normalizálásra a kettes és a tizenhatos számrendszerben:

$$111001010_{(2)} = 0,111001010_{(2)} \cdot 2^{+9}$$

$$10,00110_{(2)} = 0,1000110_{(2)} \cdot 2^{+2}$$

$$-0,1000110_{(2)} = -0,1000110_{(2)} \cdot 2^{0}$$

$$+0,000111011_{(2)} = +0,111011_{(2)} \cdot 2^{-3}$$

$$E6AD_{(16)} = 0,E6AD_{(16)} \cdot 16^{+4}$$

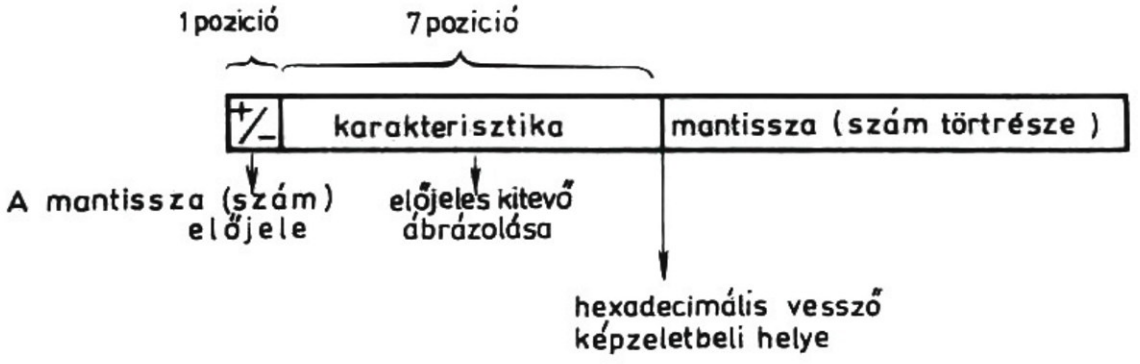
$$7D,82_{(16)} = 0,7D82_{(16)} \cdot 16^{+2}$$

$$-0,1BC3_{(16)} = -0,1BC3_{(16)} \cdot 16^{0}$$

$$+0,000F03A_{(16)} = \underbrace{+}_{\text{mantissza előjele}}0,\underbrace{F03A}_{\text{mantissza}}{}_{(16)} \cdot 16^{\underbrace{-3}_{\text{karakterisztika}}}$$

Lebegőpontos ábrázolásnál mindig ebből a normálalakból indulunk ki. Ha a normalizálásnál használt számrendszer alapszámát előre meghatározzuk, felesleges a hatvány alapjának ábrázolása, az eredeti "A" számot a mantissza előjelével, a mantisszával és a karakterisztikával egyértelmüen megoldhatjuk. A ma használatos, korszerü gépeknél általában 16-os alapra normalizálnak, vagyis a hexadecimális vesszőt tolják el ugy, hogy a mantissza törtrésze értékes jeggyel kezdődjön, természetesen a hexadecimális alakban megadott számértéknek megfelelően.

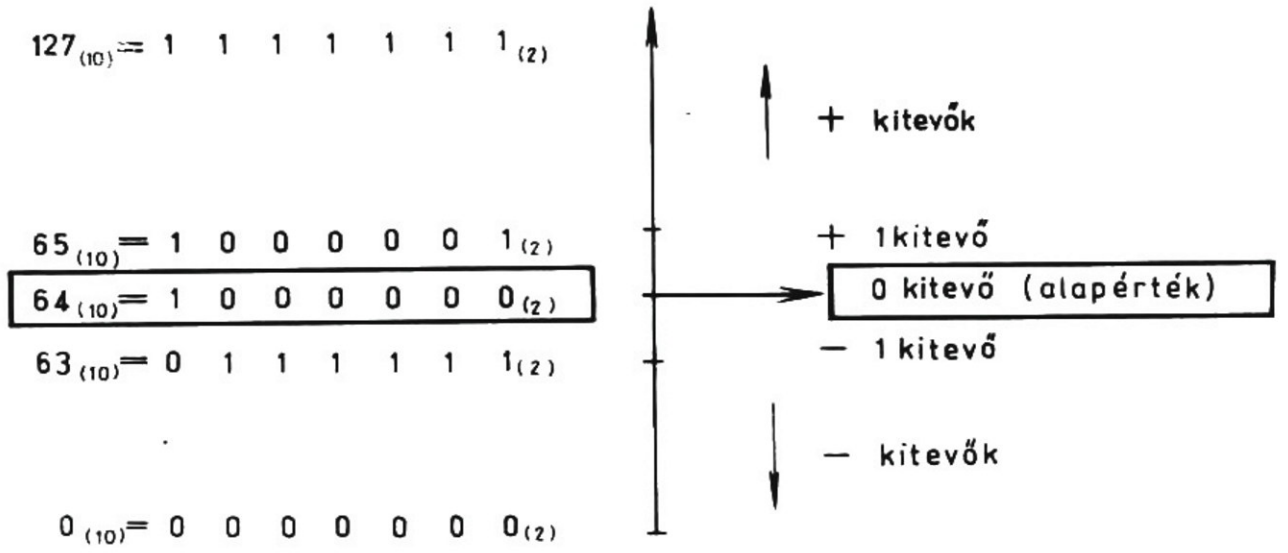
A szokásos ábrázolási forma:



31. ábra. Lebegőpontos ábrázolási forma

A mantissza (szám) előjelének ábrázolása megegyezés szerint ugy történik, hogy pozitiv mantissza (szám) esetén 0-ás, negativ mantissza (szám) esetén 1-es bit kerül a legelső pozicióba. Ezt a előjelábrázolási módot hivják egyenes kódnak. A karakterisztika (előjeles kitevő) ábrázolását az un. alapértékes vagy feszitett előjeles kitevő módján végzik. Szokták még "64 többletes" kitevőábrázolásnak is hivni.

A rendelkezésre álló 7 pozicióban a 0 kitevőt megegyezés szerint az "alapérték", az $1000000_{(2)} = 64_{(10)}$ segitségével ábrázolják. Ez tulajdonképpen ugy keletkezett, hogy a 7 biten kifejezhető $2^7 = 128_{(10)}$ különféle állapot "közepét", a $64_{(10)}$-et önkényesen nullának, alapértéknek nevezték el. A pozitiv kitevő értékét ehhez az alapértékhez hozzáadják, a negativ kitevő értékét pedig ebből az alapértékből levonják.



32. ábra. A karakterisztika ábrázolási módja alapértékesen, 7 biten

Például a +3 kitevő ábrázolása alapértékesen a következőképpen történik:

alapérték	$1000000_{(2)}$	$= 64_{(10)}$
kitevő 3 értékének hozzáadása	$+ \quad 11_{(2)}$	$= + 3_{(10)}$
A +3 kitevő ábrázolási formája (7 bittel)	$1000011_{(2)}$	$67_{(10)}$

A kitevő alapértékes ábrázolásához a számitást természetesen akár a bináris, akár a decimális számrendszeren belül (lásd a fenti példa jobb oldalán) elvégezhetjük, de az ábrázolás mindig a bináris alakban történik.

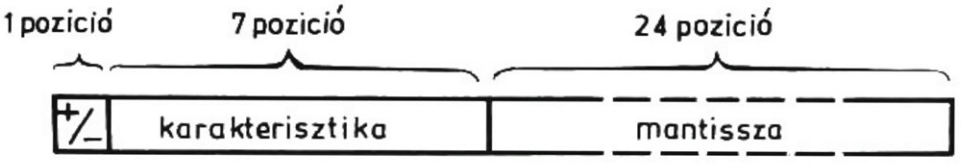
A -3 kitevő ábrázolása alapértékesen:

alapérték	$1000000_{(2)}$	$= 64_{(10)}$
kitevő 3 értékének levonása	$- \quad 11_{(2)}$	$= - 3_{(10)}$
A -3 kitevő ábrázolási formája (7 bittel)	$0111101_{(2)}$	$61_{(10)}$

Az alapértékes (feszitett előjeles) karakterisztika-ábrázolás előnye, hogy nem kell külön feltüntetni valamilyen formában a kitevő előjelét és ez ránézésre mégis azonnal megállapitható a 7 bitből. Amennyiben a 7 pozición levő bitsorozat $1000000_{(2)}$-nél nagyobb, vagyis 1-essel kezdődik, az ábrázolt szám karakterisztikája pozitiv (mert valami „hozzá lett adva" az alapértékhez), amennyiben $1000000_{(2)}$-nál kisebb, vagyis 0-val kezdődik, az ábrázolt szám karakterisztikája negativ (valami "le lett vonva" az alapértékből).

A mantissza rész ábrázolása bináris formában történik, tehát a 16-osra normalizált törtrészt még át kell alakitani kettes számrendszerbe. A bináris vesszőt a 8. pozició után kell képzelni, és igy a mantissza - a törtrész - jegyei a kilencediktől kezdődhetnek. A fixpontoshoz hasonlóan ez az ábrázolás is rögzitett hosszon történik. Az un. rövid, más néven egyszeres pontosságu lebegőpontos alakban az ábrázolás hossza a gép által használt szó hosszával megegyezően általában 32 bit. Az un. hosszu lebegőpontos vagy duplapontos alakban az ábrázolás hossza a gép által használt szó hosszuságának kétszerese, vagyis általában 64 bit. Ez azonban csak az ábrázolt szám pontosságát növeli, mivel igy a mantissza rész hosszabb lehet. Az ábrázolható nagyságrend nem változik, mivel a karakterisztikát minden esetben 7 bittel ábrázoljuk.

Rövid lebegőpontos alak (32 bit):



33. ábra. Rövid lebegőpontos alak

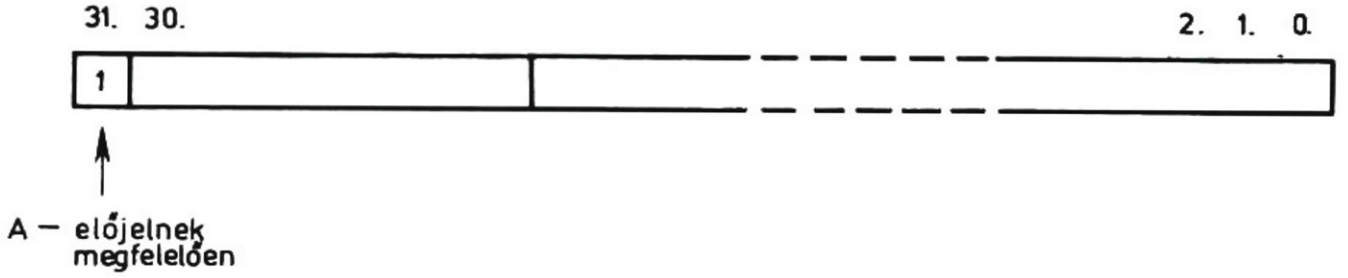
Hosszu (duplapontos) lebegőpontos alak (64 bit):

34. ábra. Hosszu (duplapontos) lebegőpontos alak

Ábrázoljuk most rövid lebegőpontos alakban (32 biten), 16-osra normalizálva a $-91_{(10)}$-et!

$$-91_{(10)} \longrightarrow -1011011_{(2)} \longrightarrow -5B_{(16)} \longrightarrow \underbrace{-0,5B_{(16)} \cdot 16^{+2}}_{\text{ábrázolandó}}$$

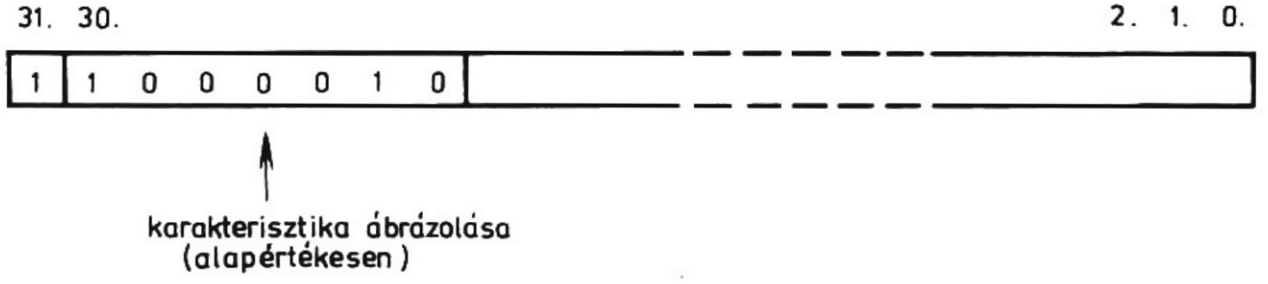
A negativ előjel miatt az ábrázolás legelső pozициójába (a megegyezésnek megfelelően) $1_{(2)}$-es bit kerül.



35. ábra. Negativ előjel ábrázolása lebegőpontosan

A karakterisztikát alapértékesen ábrázoljuk. Mivel a kitevő előjele pozitiv, a kitevő értékét hozzáadjuk az alapértékhez.

alapérték	$1000000_{(2)} = 64_{(10)}$
kitevő 2 hozzáadása	$+ \quad 10_{(2)} = +2_{(10)}$
a kitevő ábrázolási formája (7 biten)	$1000010_{(2)} = 66_{(10)}$



36. ábra. Előjel- és alapértékes karakterisztika-ábrázolás lebegőpontosan

A normalizált szám értékes jegyeit, a mantissza tört részét bináris formában kell ábrázolni, ezért a 16-os számrendszerben levő törtrészt tetrádok segitségével vissza kell irni a 2-es számrendszerbe. A mantissza rész ábrázolása után fennmaradó pozíciókat (jobb oldalt) 0-kkal töltjük ki.

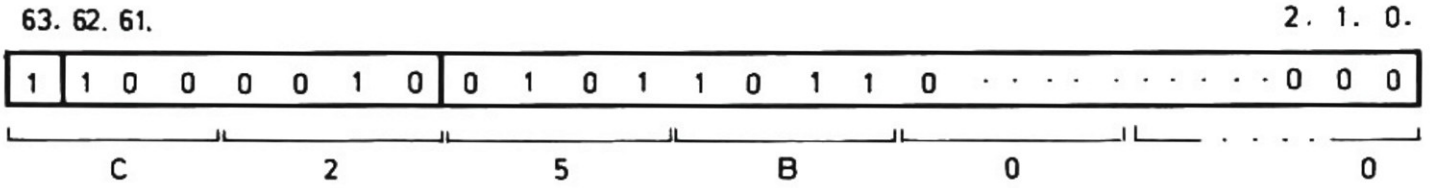
5 B$_{16}$
0101 1011

37. ábra. $-91_{(10)}$ lebegőpontosan ábrázolva, 16-osra normalizálva

A könnyebb kezelhetőség érdekében a lebegőpontosan ábrázolt operandust is felirhatjuk rövidebben a 16-os számrendszer segitségével. A gép is ugyanezt az egyszerüsitést alkalmazza, ha eredményt vagy tártartalmat akar kiirni. A $-91_{(10)}$ 16-osra normalizált, rövid lebegőpontos alakja tehát

$C25B0000_{(16)}$

A $-91_{(10)}$ 16-osra normalizált hosszu lebegőpontos alakja csak a mantisszarészben tér el az előbbitől (nagyobb rész áll a mantissza rendelkezésére):



38. ábra. Az előző ábrán szereplő operandus duplapontos alakban

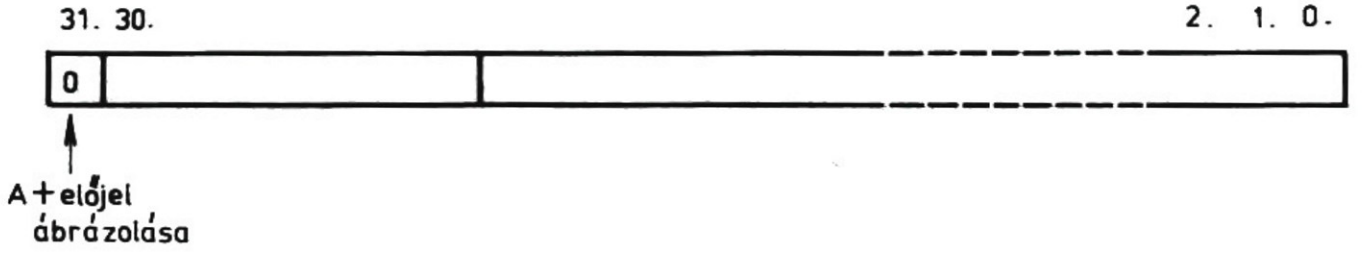
Ábrázoljuk még gyakorlásképpen a $+0,00125_{(10)}$-öt, ugyancsak rövid, lebegőpontos formában, 16-osra normalizálva. A decimális-bináris konverziót 12 bináris törtjegyig végezzük!

$$+0,00125_{(10)} \Longrightarrow +0,\underbrace{000000000101}_{0\ \ 0\ \ 5}{}_{(2)} \Longrightarrow$$

$$+0,005_{(16)} \Longrightarrow +0,5_{(16)} \cdot 16^{-2}$$

ábrázolandók

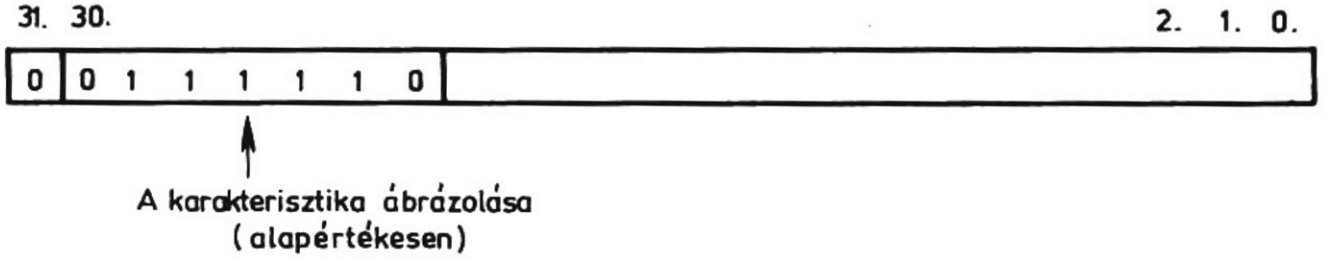
A pozitiv előjel miatt a legelső pozicióba a $0_{(2)}$ kerül.



39. ábra. Pozitiv előjel ábrázolása lebegőpontosan

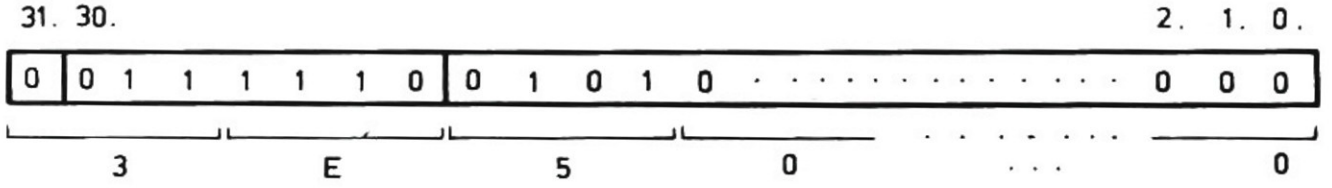
A karakterisztika alapértékes ábrázolásakor a kitevő értékét le kell vonni az alapértékből, mivel az előjele negativ

alapérték	$1000000_{(2)} = 64_{(10)}$
kitevő 2 levonása	$- \quad 10_{(2)} = -2_{(10)}$
a kitevő ábrázolt formája (7 biten)	$0111110_{(2)} = 62_{(10)}$



40. ábra. Előjel és alapértékes karakterisztika lebegőpontos formában

Ezután a mantissza tört részét ábrázoljuk, természetesen most is binárisan:



41. ábra. Lebegőpontosan ábrázolt operandus ($+0,00125_{(10)}$)

Hexadecimálisan ez a következőképpen néz ki:

$3E500000_{(16)}$

Lebegőpontosan az a maximálisan ábrázolható szám, amelynek normál alakjában olyan kitevő szerepel, ami a 7 pozicióm elfér (amit "még hozzá lehet adni" az alapértékhez). A 7 pozicióm megjelenithető legnagyobb szám a hét db "1"-es.

$1\ 111\ 111_{(2)} = 127_{(10)}$

Ez az alapértéknél $0\ 111\ 111_{(2)} = 63_{(10)}$-mal nagyobb. Ez tehát az alapértékesen ábrázolható kitevő maximális értéke.

Legkisebb számként pedig az irható fel lebegőpontosan, amelynek normálalakjában olyan kitevő szerepel, ami a 7 pozicióm még ábrázolható (vagyis amennyit az alapértékből még "le lehet vonni"). A 7 pozicióm megjelenithető legkisebb szám a hét db "0".

$0\ 000\ 000_{(2)} = 0_{(10)}$

Ezt pedig akkor kapjuk, ha az alapértékből vele egyező nagyságu számot, tehát $1\ 000\ 000_{(2)} = 64_{(10)}$-et levonunk. Az alapértékesen ábrázolható legnagyobb negativ kitevő értéke tehát ennyi lehet.

A 16-osra normalizált, lebegőpontosan kifejezett szám (normálalakbeli) nagyságrendje tehát e közé a két érték közé kell hogy essen.

$16^{-64} \leq N \leq 16^{+63}$

$10^{-78} \leq N \leq 10^{+76}$

Vannak gépek, amelyek az előzőektől kissé eltérnek és nem 16-os, hanem 2-es alapra normalizálnak. Vagyis nem a hexadecimális, hanem a bináris vesszőt tolják el, az első értékes jegy elé. Az adott hosszon (általában itt is 32 vagy 64 pozición) az ábrázolható szám nagyságrendje igy kisebb lesz.

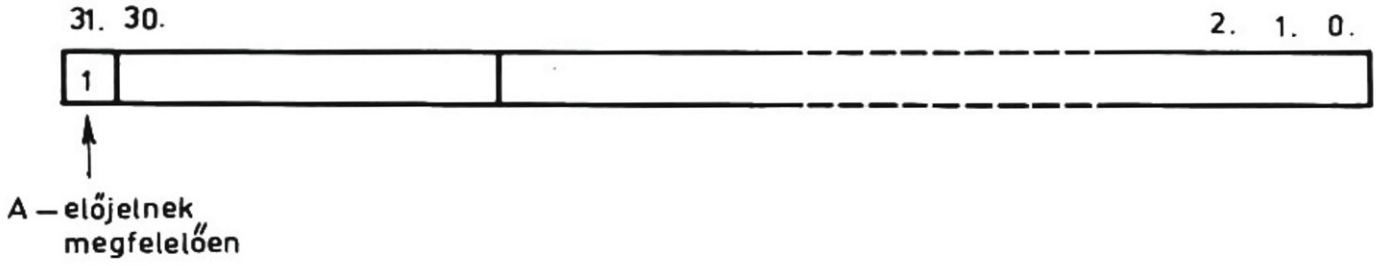
$$2^{-64} \leq N \leq 2^{+63}$$

Az ábrázolás pontosságát viszont ez nem befolyásolja, az, amint már emlitettük, csak a mantissza hosszától függ. A korábban már ábrázolt $-91_{(10)}$-et most az elmondottak szemléltetésére alakitsuk át lebegőpontossá, kettes alapra normalizálva!

$$-91_{(10)} \Longrightarrow -1011011_{(2)} \Longrightarrow -0,\underline{1011011}_{(2)} \cdot 2^{\underline{+7}}$$

ábrázolandók

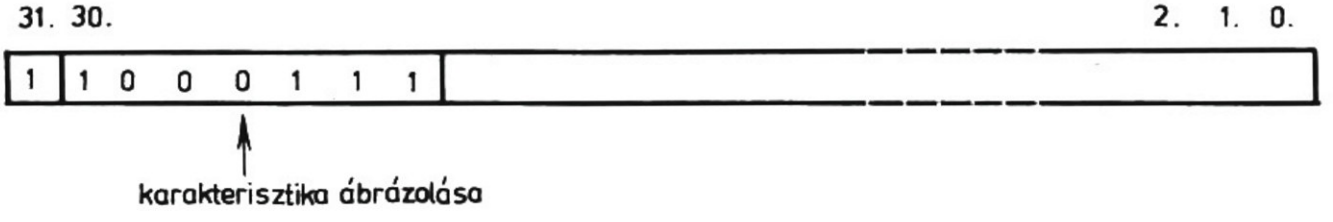
A negativ előjelet a megegyezés szerint a legelső pozición, az 1-es bit jelenti:



42. ábra. Negativ előjel ábrázolása lebegőpontosan

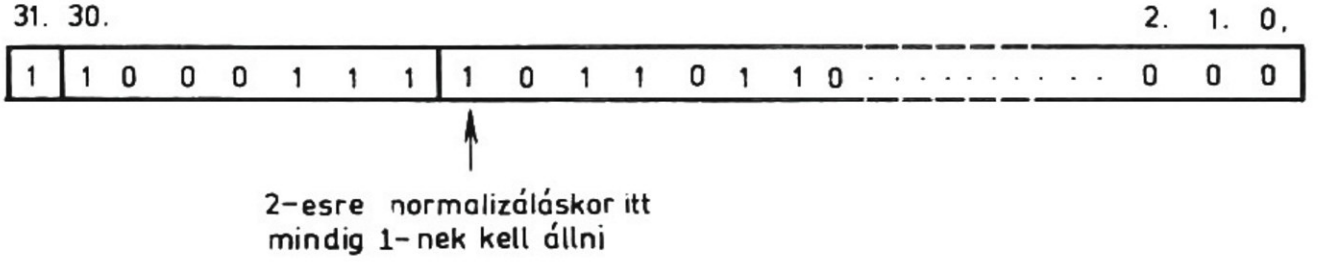
A karakterisztika alapértékes ábrázolása az elmondottaknak megfelelően történik, de mivel a normalizálást, a "vessző eltolását" a kettes számrendszerben végeztük el, a kitevő értéke természetesen nagyobb lesz, mintha a 16-os számrendszerben végeztük volna, igy az ábrázolása is több pozíciót igényel:

alapérték	$1000000_{(2)} = 64_{(10)}$
kitevő 7 hozzáadása	$+ \quad 111_{(2)} = +7_{(10)}$
a kitevő ábrázolása alapértékesen	$1000111_{(2)} = 71_{(10)}$



43. ábra. Előjel és karakterisztika ábrázolása lebegőpontosan

A mantissza ábrázolása: mivel a normalizáláskor a bináris vesszőt az első értékes jegy elé viszszük, és az igy keletkező törtrészt kell ábrázolni, a 2-esre történt normalizáláskor a mantissza rész mindig értékes jeggyel (1-essel) kezdődik! (A 16-os alapra történő normalizálásnál ez csak akkor fordul elő, ha a hexadecimális alakban levő mantissza rész $8_{(16)}$ vagy annál nagyobb jeggyel kezdődik, egyébként $0_{(2)}$ áll a mantissza rész első pozíciójában.)



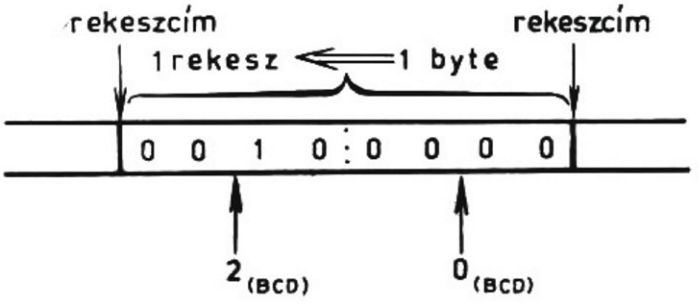
44. ábra. $-91_{(10)}$ lebegőpontosan ábrázolva, kettesre normalizálva

2.7.1.2 Decimális ábrázolás

Decimális ábrázoláskor a numerikus operandusokat BCD számként ábrázoljuk. A gépnek ilyenkor kevesebb számrendszerek közötti konverziót kell végeznie, bár a müveletvégzés ideje hosszabb lesz, mint a fixpontos aritmetikai müveleteknél. Ezért ez a számábrázolási mód adatfeldolgozási feladatoknál a legelterjedtebb, ahol csak az alapmüveleteket kell elvégezni. Az ilyen operandusokat a decimális aritmetikai müveleteknél használják és feldolgozásukat az un. decimális aritmetika végzi el.

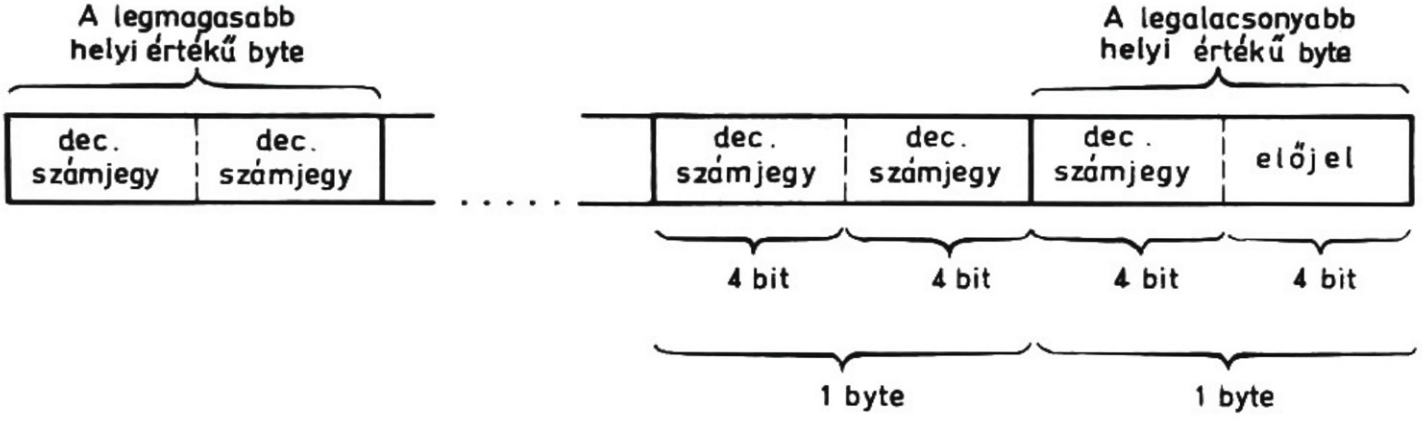
1. Pakolt alak

Az operativ tárban a cimezhető egységek, a rekeszek 1 byte-ot, vagyis általában 8 bitet képesek tárolni. Igy egy ilyen "táregységben", rekeszben egyidejüleg két binárisan kódolt decimális (BCD) számjegyet helyezhetünk el.



45. ábra. Egy rekeszben két BCD szám fér el

Az igy ábrázolt decimális operandusokat tömöritett vagy más néven pakolt operandusoknak nevezzük. Az előjel ábrázolása a számmező legalacsonyabb helyi értékü byte-jának jobb felén, a jobb félbyteon történik.



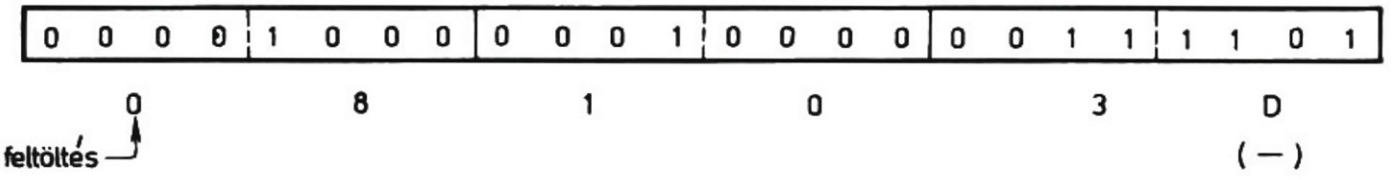
46. ábra. Pakolt (tömöritett) operandus

Az előjelek kódja megegyezés szerint:

+ előjel $\longrightarrow 1100_{(2)} = C_{(16)}$

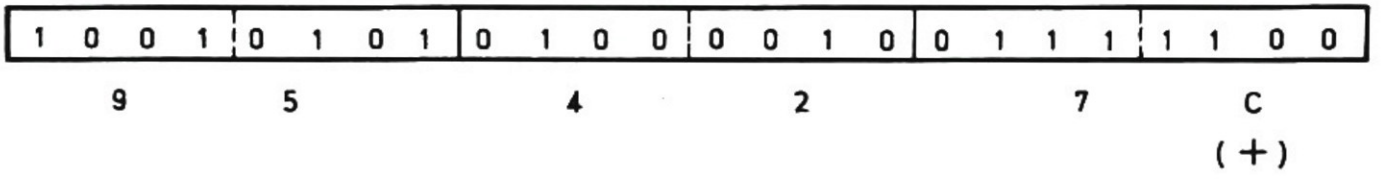
- előjel $\longrightarrow 1101_{(2)} = D_{(16)}$

Az ábrázolt operandus mindig egész számu byte-ot foglal el, mégpedig annyit, amennyire éppen szükség van. A maximális hosszuk a számitógépek utasitástipusai miatt általában 16 byte lehet, amelyen összesen 31 decimális számjegy és az előjel adható meg. (Ennek okát a byte-szervezésü gépek utasitásformátumainál ismertetésre kerülő SS2 tipusu utasitásnál lehet megérteni!) A $-8103_{(10)}$ ábrázolása pakolt operandusként 5 db félbyte-ot igényel, mivel 4 db számjegy és egy előjel van. Mivel páratlan számu félbyte-ot a gép nem tud tárolni, ezért az operandus összesen 3 byte-ot foglal le. A legmagasabb félbyte-ot a gép 0-kkal tölti fel ("balra, kötötten nullákat ir be").



47. ábra. A $-8103_{(10)}$ pakolt formában

A $+95427_{(10)}$ ugyancsak 3 byte-on fér el, de itt már nem marad üresen félbyte, tehát nincs szükség feltöltésre sem.



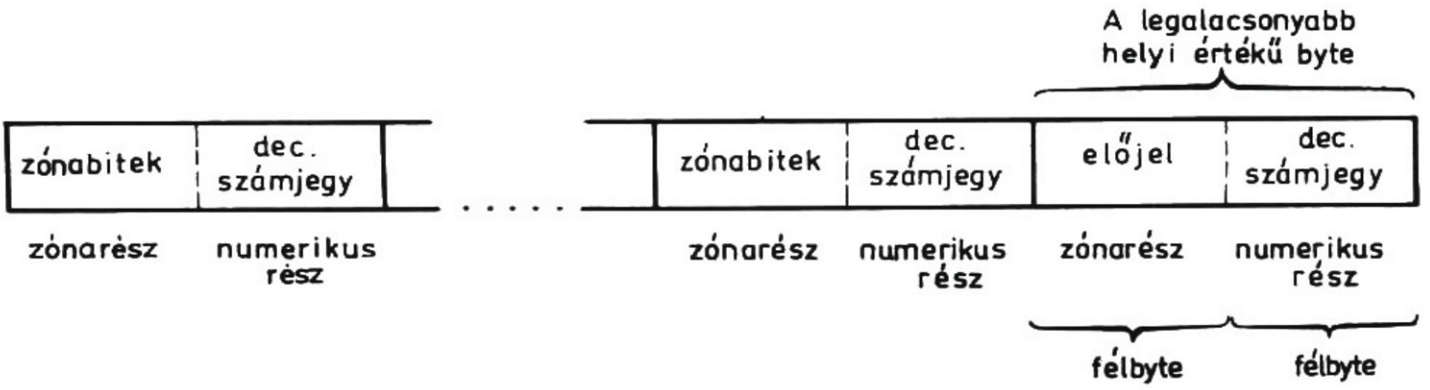
48. ábra. A $+95427_{(10)}$ ábrázolása pakolt operandusként

A pakolt operandusokkal a decimális aritmetikai egység müveleteket végezhet, hiszen az előjelet tartalmazó utolsó félbyte és a számjegyeket ábrázoló többi félbyte könnyen különválasztható egymástól. Az input-output müveleteknél viszont problémát okoz az ilyen módon megadott operandus. Ott ugyanis minden egyes byte-nak egy karaktert kell megfeleltetni. Ez pedig lehetetlen, ha egy byte-on két különböző karakter van (decimális számjegy, illetve előjel).

A megoldást az jelenti, ha az input-output müveleteknél a pakolt operandusokat "felhigitjuk", "széthuzzuk" ugy, hogy minden egyes byte-on csak egy decimális számjegy legyen. A decimális számoknak ez a fajta ábrázolása az un. zónás vagy nem tömöritett alak.

2. Zónás alak

Ebben az esetben a decimális számjegyeket 1 byte-on karakterenként ábrázoljuk. Kivétel a legalacsonyabb helyi értékü byte, ahol a számjegy mellett az előjel van. A byte-ok jobb oldali felét, a jobb oldali félbyte-ot ebben az esetben numerikus résznek hivják, és itt helyezkednek el a decimális számjegyek, binárisan kódolva. A bal oldali félbyte az un. zónarész. A legalacsonyabb helyi értékü byte zónarészében szerepel az előjel, a többi zónarészben az un. zónabitek helyezkednek el.



49. ábra. Zónás (nem tömöritett) operandus

zónabitek ⟶ $1111_{(2)} = F_{(16)}$

+ előjel ⟶ $1010_{(2)} = A_{(16)}$

$\boxed{1100_{(2)} = C_{(16)}}$

$1110_{(2)} = E_{(16)}$

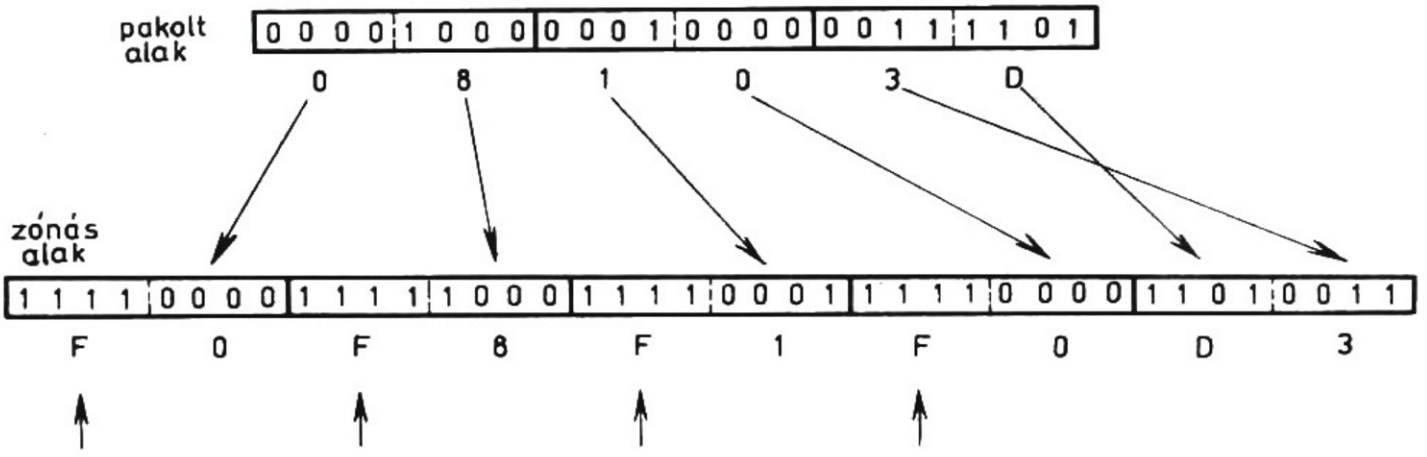
$1111_{(2)} = F_{(16)}$

- előjel ⟶ $1011_{(2)} = B_{(16)}$

$\boxed{1101_{(2)} = D_{(16)}}$

A decimális számok zónás alakjában a zónabitek azt jelzik, hogy a következő félbyte-on számjegy található, értékük $1111_{(2)} = F_{(16)}$.

Az előjel ábrázolására a legutolsó byte zónarészében ugyanaz a két kód szolgál, mint a pakolt alaknál (lásd fent bekeretezve!). Ha a decimális számjegyet a pozitiv előjel feltüntetése nélkül ábrázoltuk, akkor az előjel helyére is $1111_{(2)} = F_{(16)}$ kerül. A gépek forditóprogramjai ezért nemcsak az $1100_{(2)} = C_{(16)}$, hanem az $1111_{(2)} = F_{(16)}$ bitkombinációt is pozitiv előjelnek forditják. Sőt ugyanigy pozitivnak értelmezik a számot egyéb kódrendszerekkel való kompatibilitás érdekében, ha az előjelet tartalmazó zónarészben az $1010_{(2)} = A_{(16)}$ vagy az $1110_{(2)} = E_{(16)}$ szerepel. Ehhez hasonlóan a negativ előjel "szabványos" ábrázolásán - $1101_{(2)} = D_{(16)}$ - kivül a zónarészben szereplő $1011_{(2)} = B_{(16)}$ alakot is negativ előjelnek fogja tekinteni a gép. Alakitsuk át az előbbi pakolt formából zónássá a -8103-at! Legelőször is a pakolt alak utolsó két félbyte-ja helyet cserél. Utána minden további BCD szám egy-egy byte numerikus részébe (jobb oldali félbyte-jába) kerül. A zónarészekbe (bal oldali félbyte-ok) pedig a gép $1111_{(2)} = F_{(16)}$-eket ir.



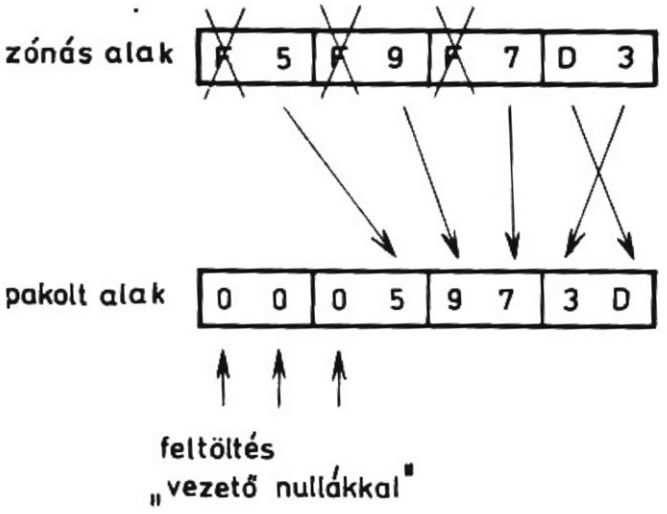
50. ábra. Átalakitás pakoltból zónás operandussá

Figyelem! Az átalakitás a megfelelő utasitás hatására automatikusan történik. Tehát a gép nem ellenőrzi, hogy a pakolt alakban értékes jegy szerepel-e minden egyes félbyte-on. Ezért a legelső félbyte-ra esetleg feltöltés miatt odakerült értéktelen 0-kat is átalakitja zónás alakban egy külön byte-on. Igy 5 byte-ra van szükség a -8103 zónás alakban történő ábrázoláshoz.

A zónás alakból tömöritetté való visszaalakitás pedig az alábbiak szerint megy végbe. A kiindulási szám legyen $-5973_{(10)}$.

Zónás alakban: F 5 F 9 F 7 D 3

Először a jobb szélső byte zóna és numerikus része felcserélődik, a többi byte zónarészét azonban nem veszi figyelembe a gép. A többi byte numerikus részének tartalmát jobbról balra haladva sorban átviszi a legkisebb helyi értékü byte mellé. Ha a legnagyobb, még lefoglalt byte bal fele üres maradt, akkor oda $0_{(16)}$-t ir be. Ha ugyanolyan hosszon akarjuk a pakolt alakot is ábrázolni, mint a zónásat, akkor az üres byte-okat is feltölti $0_{(16)}$-kkal (un. "vezető nullákkal").



51. ábra. Zónás operandus átalakitása pakolttá

A zónás alakban ábrázolt decimális adatokkal perifériális müveletet már tud végezni a gép, aritmetikai müveletet viszont nem! Könnyen beláthatjuk ezt, ha pl. az összeadásnál jelentkező átvitelre gondolunk. A számjegyek között levő zónabitek, illetve előjelbitek a helyes müveletvégzést lehetetlenné teszik!

Decimális operandussal végezhető müvelet

pakolt alakban	zónás alakban
- input-output müvelet	- input-output müvelet
- aritmetikai müvelet	- aritmetikai müvelet

Ezért a két alakot felváltva, a müveletvégzésnek megfelelően kell használni. A két alak közötti átalakitást utasitás (PACK, illetve UNPK) segitségével irhatjuk elő, és azt a gép önállóan elvégzi.

2.7.2 ALFANUMERIKUS ADATOK ÁBRÁZOLÁSA

Alfanumerikus adatábrázolásról akkor beszélünk, amikor számokat, betüket és különféle karaktereket önállóan, egymástól függetlenül, karakterenként kivánunk rögziteni. A központi egységben történő ilyen ábrázoláshoz tulajdonképpen két 8 bites kód is rendelkezésre áll:

EBCDIC — Extended Binary Coded Decimal Interchange Code

ASCII-8 — American Standard Code for Information Interchange

1. táblázat

ASCII-8 kód (bővitett ASCII)

Bitpoziciók

								0				0				1				1											
								0				1				0				1											
								0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1								
								0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1								
8	7	6	5	4	3	2	1																								
				0	0	0	0	NUL	DLE			SPA	0						P				p								
				0	0	0	1	SOH	DC1			1	1					A	Q			a	q								
				0	0	1	0	STX	DC2			"	2					B	R			b	r								
				0	0	1	1	EXT	DC3			#	3					C	S			c	s								
				0	1	0	0	EOT	DC4			$	4					D	T			d	t								
				0	1	0	1	ENQ	NAK			%	5					E	U			e	u								
				0	1	1	0	ACK	SYN			&	6					F	V			f	v								
				0	1	1	1	BEL	ETB			'	7					G	W			g	w								
				1	0	0	0	BS	CAN			(	8					H	X			h	x								
				1	0	0	1	HT	EM			)	9					I	Y			i	y								
				1	0	1	0	LF	SS			.	:					J	Z			j	z								
				1	0	1	1	VT	ESC			+	;					K	[			k	{								
				1	1	0	0	FF	FS			,	<					L	\			l	/								
				1	1	0	1	CR	GS			-	=					M	]			m	}								
				1	1	1	0	SO	RS				>					N	^			n	¬								
				1	1	1	1	SI	US			/	?					O	–			o	DEL								

Az ASCII-8 kód (1. táblázat) az eredetileg 7 bites ASCII kód 8 bitesre kiegészitett, továbbfejlesztett változata. (Ez a Nemzetközi Szabványügyi Szervezet, az ISO által javasolt 7 bites kódtól (2. táblázat) csak némely különleges karakter és irásjel tekintetében tér el.) Az ESZ rendszerben mindegyiknek megvan a cirillbetükkel és a cirillbetüs vezérlőkarakterekkel kiegészitett változata. Ezek az un. KOI-8, illetve a KOI-7. Mi a gyakrabban használt KOI-8 karakterkészletét mutatjuk be kódtáblázat formájában (3. táblázat).

2. táblázat

ISO 7 bites kód (CCITT Nr. 5. ABC)
(Csaknem azonos az ASCII-7 kóddal)

7	6	5	4	3	2	1								
							0	0	0	0	1	1	1	1
							0	0	1	1	0	0	1	1
							0	1	0	1	0	1	0	1
			0	0	0	0	NUL	DLE	SP	0	@*	P	*	p
			0	0	0	1	SOH	DC_1	!	1	A	Q	a	q
			0	0	1	0	STX	DC_2	" *	2	B	R	b	r
			0	0	1	1	ETX	DC_3	£*	3	C	S	c	s
			0	1	0	0	EOT	DC_4	$*	4	D	T	d	t
			0	1	0	1	ENQ	NAK	%	5	E	U	e	u
			0	1	1	0	ACK	SYN	&	6	F	V	f	v
			0	1	1	1	BEL	ETB	' *	7	G	W	g	w
			1	0	0	0	FE_0	CAN	(	8	H	X	h	x
			1	0	0	1	FE_1	EM	)	9	I	Y	i	y
			1	0	1	0	FE_2	SUB	*	: *	J	Z	j	z
			1	0	1	1	FE_3	ESC	+	; *	K	[*	k	*
			1	1	0	0	FE_4	IS_4	,	<	L	*	l	*
			1	1	0	1	FE_5	IS_3	–	=	M	]*	m	*
			1	1	1	0	SO	IS_2	.	>	N	^*	n	–.*
			1	1	1	1	SI	IS_1	/	?	O	_	o	DEL

Megjegyzés: a * -gal jelölt kódkombinációk jelentése változhat.

A központi egységben a karakterábrázoláshoz azonban általában az EBCDIC (4. táblázat) használatos, szinte valamennyi gyártó cég ezt alkalmazza. A továbbiakban mi is csak ezt fogjuk ismertetni. Ennek cirillbetükkel és ilyen formában irt vezérlőjelekkel kiegészitett változata az un. DKOI (5. táblázat). A táblázatokban szereplő vezérlőjelek, az un. szabványos technikai karakterek jelentése a 6. táblázatban található. Az EBCDIC kódban minden egyes karakter (betü, szám, irásjel) egy külön byte-ot foglal el. Az egy byte-on, azaz 8 biten kifejezhető összes különféle állapot száma (a modulus) $2^8 = 256$.

3. táblázat

KOI-8 kódtáblázat

	Zónacsoport															
a	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
a	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
a	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
a	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

a	a	a	a	a	a	a	a		00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
				0	0	0	0	0	ПУС	АР1	Проб	0	@	P		p	Д00	Д16			ю	п	Ю	П
				0	0	0	1	1	НЗ	СУ1	!	1	A	Q	a	q	Д01	Д17				я		Я
				0	0	1	0	2	НТ	СУ2	"	2	B	R	b	r	Д02	Д18			б		Б	
				0	0	1	1	3	КТ	СУ3	#	3	C	S	c	s	Д03	Д19			ц		Ц	
				0	1	0	0	4	КП	СТП	¤	4	D	T	d	t	Д04	Д20			д	т	Д	
				0	1	0	1	5	КТМ	НЕТ	%	5	E	U	e	u	Д05	Д21						у
				0	1	1	0	6	ДА	СИН	&	6	F	V	f	v	Д06	Д22			ф	ж	ф	Ж
				0	1	1	1	7	ЗВ	КБ	'	7	G	W	g	w	Д07	Д23			г	в	Г	
				1	0	0	0	8	ВШ	АН	(	8	H	X	h	x	Д08	Д24				ь		Ь
				1	0	0	1	9	ГТ	КН	)	9	I	Y	i	y	Д09	Д25			и	ы	И	Ы
				1	0	1	0	10	ПС	ЗМ	*	:	J	Z	j	z	Д10	Д26			й	з	Й	З
				1	0	1	1	11	ВТ	АР2	+	;	K	[	k	{	Д11	Д27			к	ш		Ш
				1	1	0	0	12	ПФ	РФ	,	<	L	\	l	|	Д12	Д28			л	э	Л	Э
				1	1	0	1	13	ВК	РГ	-	=	M	]	m	}	Д13	Д29			м	щ		Щ
				1	1	1	0	14	РУС	РЗ	.	>	N	^	n	~	Д14	Д30			н	ч		Ч
				1	1	1	1	15	ЛАТ	РЭ	/	?	O	_	o	ЗБ	Д15	Д31				ъ		ЗБ

EBCDIC kódtáblázat

Bitpoziciók

					0				0				1				1		
					0				1				0				1		
					0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
					0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
8	7	6	5	4	3	2	1													
				0	0	0	0	NUL	DLE	DS		SP		–										0
				0	0	0	1	SOH	DC1	SOS				/		a	j			A	J			1
				0	0	1	0	STX	DC2	FS	SYN					b	k	s		B	K	S		2
				0	0	1	1	ETX	TM							c	l	t		C	L	T		3
				0	1	0	0	PF	RES	BYP	PN					d	m	u		D	M	U		4
				0	1	0	1	HT	NL	LF	RS					e	n	v		E	N	V		5
				0	1	1	0	LC	BS	ETB	UC					f	o	w		F	O	W		6
				0	1	1	1	DEL	IL	ESC	EOT					g	p	x		G	P	X		7
				1	0	0	0		CAN							h	q	y		H	Q	Y		8
				1	0	0	1	RLF	EM							i	r	z		I	R	Z		9
				1	0	1	0	SMM	CC	SM		¢	!		:									≈
				1	0	1	1	VT	CU1	CU2	CU3	.	$	,	#									
				1	1	0	0	FF	IFS		DC4	<	*	%	a					⌠		⌡	
				1	1	0	1	CR	IGS	ENQ	NAK	(	)	–	'									
				1	1	1	0	SO	IRS	ACK		+	;	>	=					⊤				
				1	1	1	1	SI	IUS	BEL	SUB	/	¬	?	"									EO

5. táblázat

DKOI kódtáblázat

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

0	1	2	3	4	5	6	7	Numerikus csoport
				0	0	0	0	0
				0	0	0	1	1
				0	0	1	0	2
				0	0	1	1	3
				0	1	0	0	4
				0	1	0	1	5
				0	1	1	0	6
				0	1	1	1	7
				1	0	0	0	8
				1	0	0	1	9
				1	0	1	0	A
				1	0	1	1	B
				1	1	0	0	C
				1	1	0	1	D
				1	1	1	0	E
				1	1	1	1	F

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ПУС	АР1	Д00	Д16	Проб	&	-		ц	й	я	ъ	{	}	\	0
1	НЗ	СУ1	Д01	Д17			/		a	j	-	ы	A	J	Д31	1
2	НТ	СУ2	Д02	СИН					b	k	s	З	B	K	S	2
3	КТ	СУ3	Д03	Д19					c	l	t	щ	C	L	T	3
4	Д28	Д29	Д04	Д20					d	m	u	э	D	M	U	4
5	ГТ	Д05	ПС	Д21					e	n	v	щ	E	N	V	5
6	Д06	ВШ	КБ	Д22				ю	f	o	w	ч	F	O	W	6
7	ЗБ	Д07	АР2	КП					g	p	x	ь	G	P	X	7
8	Д23	АН	Д08	Д24				б	h	q	y	Ю	H	Q	Y	8
9	Д13	КН	Д09	Д25				◥	i	r	z		I	R	Z	9
A	Д14	Д18	Д10	Д26	[	]	|	:	д	к		Б				З
B	ВТ	Д15	Д11	Д27	.	¤	,	#		л		Ц	И		У	Ш
C	ПФ	РФ	Д12	СТП	<	*	%	@	ф	м	т	Д	Й	П	ж	Э
D	ВК	РГ	КТМ	НЕТ	(	)	_	◤	г	н				Я		Щ
E	РУС	РЗ	ДА	Д30	+	;	>	=			ж	ф	Л		ь	Ч
F	ЛАТ	РЭ	ЗВ	ЗМ	!	^	?	"	и	п	в	Г			ы	ЗБ

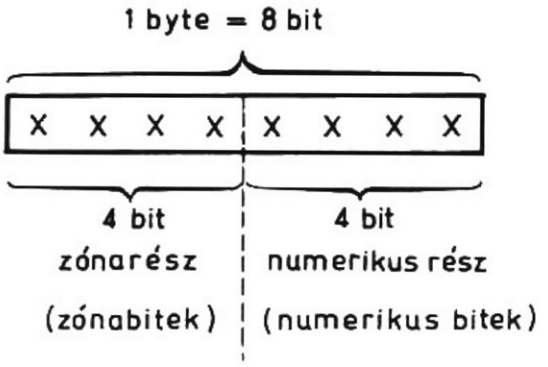
6. táblázat

Szabványos vezérlőjelek

Röviditések		Jelentésük
Orosz	Nemzetközi	
КБ	ETB (EOB)	az egység, a blokk vége
АР2	ESC	autóregiszter 2.
КТМ	ENQ	azonositás lekérdezése
ДА	ACK	pozitiv visszajelzés
ЗВ	BEL	csengő
СИН	SYN	szinkronizálás
КП	EOT	átvitel vége
СТП	STOP (DC4)	stop
НЕТ	NAK	negativ visszajelzés
ЗМ	SUM	csere
ВЫКР	PF	lyukasztó kikapcsolása
СБ	LC	kisbetük
ВСТ	RES	helyreállitás
НС	NL	uj sor
ОЖО	IL	várakozás
УУК	CC	a jelző (kurzor) vezérlése
ВЧФ	DS	számkiválasztás
НЗН	SOS	szignifikáns kezdete
РФ	FS	elválasztójel
БК	BYP	blokkolás
УР	SM	üzembe állitás
ВК	PN	a lyukasztás bekapcsolása
ОУ	RS	az olvasó leállitása
ЛБ	US	nagybetü

Röviditések		Jelentésük
Orosz	Nemzetközi	
ПУС	NUL	nulla (kitöltőjel)
НЗ	SOH	fej kezdete
НТ	STX	szöveg kezdete
КТ	ETX	szöveg vége
ГТ	HT	vizszintes tabulálás
ЗБ	DEL	törlés
ВТ	VT	függőleges tabulálás
ПФ	FF	formátum váltás
ВК	CR	kocsi vissza
РУС	SO	cirill betük
ЛАТ	SI	latin betük
АРІ	DLE	autóregiszter 1.
СУІ	DC 1	vezérlőjelek
СУ2	DC 2	vezérlőjelek
СУ3	DC 3	vezérlőjelek
ВШ	BS	visszalépés
АН	CAN	törlés (nullázás)
КН	EM	hordozó vége
РФ /РИ4/	FS (IS4)	file-ok elválasztó jele
РГ /РИ3/	GS (IS3)	csoportok elválasztó jele
РЗ /РИ2/	RS (IS2)	beirás elválasztójel
РЭ /РИІ/	US (IS1)	elem elválasztójel
ПС	LF	soremelés

Ennyi karakter ábrázolására egy rendszerben sincs szükség, tehát lehetőség van a karakterkészlet csoportokba sorolására és a csoportok megkülönböztetésére. Az egyes csoportokba közel azonos számu karakter tartozik. Igy csoportot alkotnak például a számjegyek (0....9) vagy a nagybetük és a kisbetük közel azonos darabszámu tagjai. Az EBCDIC a karakterek ábrázolására a byte-ot két részre, két 4 bites tetrádra osztja, ezek alkotják az ugynevezett zónarészt, illetve numerikus részt.



52. ábra. A byte felosztása zónarészre és numerikus részre

A zónarészben levő ugynevezett zónabitek szolgálnak a karakterek egyes csoportjának kijelölésére:

karaktercsoportok	zónabitek
0 - 9	$1111_{(2)} = F_{(16)}$
A - I	$1100_{(2)} = C_{(16)}$
J - R	$1101_{(2)} = D_{(16)}$
S - Z	$1110_{(2)} = E_{(16)}$
a - i	$1000_{(2)} = 8_{(10)}$
j - r	$1001_{(2)} = 9_{(16)}$
s - z	$1010_{(2)} = A_{(16)}$

Például a számjegyek EBCDIC-kódja a következőképpen néz ki:

	zóna-rész	numerikus rész	
0 =	1111	0000	$= F0_{(10)}$
1 =	1111	0001	$= F1_{(16)}$
2 =	1111	0010	$= F2_{(16)}$
3 =	1111	0011	$= F3_{(16)}$
4 =	1111	0100	$= F4_{(16)}$
5 =	1111	0101	$= F5_{(16)}$

6 = 1111 0110 = $F6_{(16)}$

7 = 1111 0111 = $F7_{(16)}$

8 = 1111 1000 = $F8_{(16)}$

9 = 1111 1001 = $F9_{(16)}$

A zónabitek itt a számok csoportját jelölték ki, a numerikus részben pedig az illető szám értéke szerepel.

A betük a csoporton belüli sorszámokkal adhatók meg könnyen. Igy ezek EBCDIC-kódjában a numerikus bitek az illető csoporton belüli sorszámot mutatják. Igy a bitsorozat által ábrázolt karakter gyorsan visszakereshető, könnyen megjegyezhető. Néhány példa a betük EBCDIC-kódu ábrázolására

A = 1100 0001 = $C1_{(16)}$

B = 1100 0010 = $C2_{(16)}$

⋮

I = 1100 1001 = $C9_{(16)}$

(-1) J = 1101 0001 = $D1_{(16)}$

(-2) K = 1101 0010 = $D2_{(16)}$

(-3) L = 1101 0011 = $D3_{(16)}$

⋮

(-9) R = 1101 1001 = $D9_{(16)}$

S = 1110 0010 = $E2_{(16)}$!

T = 1110 0011 = $E3_{(16)}$

⋮

Z = 1110 1001 = $E9_{(16)}$

Az EBCDIC-kódban ábrázolt számokkal a gép számitást nem tud végezni! Főként a szöveges információhoz tartozó (tárolandó, kinyomtatandó) numerikus adatok ábrázolására szolgál. Az előjel, mint különleges karakter, külön kerül ábrázolásra! Az EBCDIC ezért alapvetően csak előjel nélkül (pozitiv) számok ábrázolására alkalmas. Ha valamilyen oknál fogva mégis negativ számok ábrázolására van feltétlenül szükség, akkor a második betücsoport kódjait használják fel. De a tárolt információ (tártartalom) alapján sem a gép, sem a felhasználó nem tudja megkülönböztetni, hogy az adott bináris jelsorozat betüt vagy negativ előjelü számot jelöl-e? A programozó feladata, hogy megjegyezze, hol és mit rögzitett és ennek megfelelően megkülönböztesse a tár szövegmezőit és számmezőit.

Az utolsó betücsoportba már csak 8 betü jut, ezért itt a numerikus bitek eggyel nagyobb sorszámot adnak meg. (A csoporton belüli sorszámozás 2-től kezdődik!)
Alfanumerikus ábrázolás esetén az üres pozíciót, az "üres karaktert" (blank) a $0100\ 0000_{(2)} = 40_{(16)}$ kód jelöli.

Számok ábrázolása esetén az elöl álló értéktelen nullák ($0000\ 0000_{(2)} = 00_{(16)}$ elhagyásakor helyettük is blank kerül tárolásra. (A kijelölt, de el nem foglalt pozíciókban.)

Ábrázoljunk alfanumerikus adatokat egy 4 byte-os területen (mezőben) az egyszerüség kedvéért a rövidebb, a hexadecimális felirásmódot használva.

KI␣?	D2	C9	40	6F
(51)	4D	F5	F1	5D
GOLF	C7	D6	D3	C6
015M	00	F1	F5	D4
DS21	C4	E2	F2	F1
0000	F0	F0	F0	F0
␣-17	40	60	F1	F7
␣␣␣W	40	40	40	E6
$␣10	5B	40	F1	F0
+0,5	4E	F0	6B	F3

␣ jel az üres pozíciókat mutatja

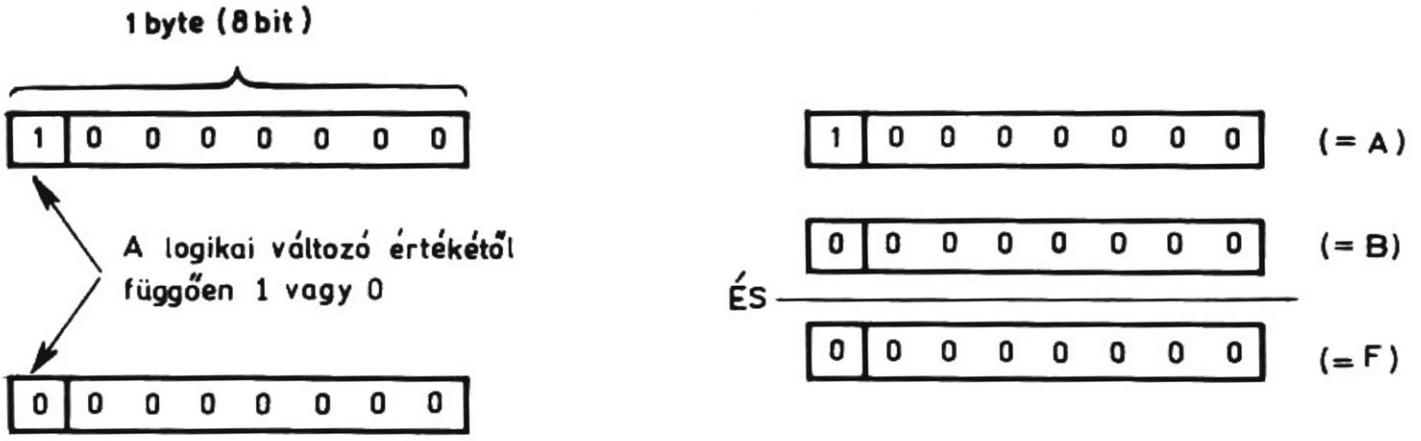
Megjegyzés: Az alfanumerikus operandus hossza a szükségnek megfelelően változhat, de maximálisan 256 byte lehet (a magyarázatot lásd a SS1 tipusu utasitásban). (Lásd 3.4.7 pont.)

A perifériális berendezésekben (háttértárak, input-output eszközök) alfanumerikus adatok kifejezésére léteznek még másfajta kódok is. Például a 6 bites BCD kód (főleg mágnesszalagoknál és II. generációs gépeknél használatos), lyukkártyakódok, lyukszalagkódok. Az ASCII kód is ezeknél és az ESZR gépeknél terjedt el.

2.7.3 LOGIKAI ADAT ÁBRÁZOLÁSA

Logikai adatnak a logikai müveletben szereplő logikai változót tekintjük. Ennek ábrázolása egy bitet igényel, mivel ez vagy $0_{(2)}$ vagy $1_{(2)}$ a logikai változó értékétől, illetőleg a vele végzett müvelet eredményétől függően. A gép ezt az adatot is minimálisan egy byte-on ábrázolja. (Ez a gép által kezelhető legkisebb információmennyiség.) A logikai adat rögzitése a byte legnagyobb helyi értékü bitjén történik, a többi bit-pozicióban pedig $0_{(2)}$ van. Ez a fix vagy rögzitett hosszuságu (1 byte-os) ábrázolás. (53. ábra.)

Mivel bármilyen adaton végezhetünk logikai müveletet, igy bármilyen adatot tekinthetünk logikai változónak is (karaktersorozatot, bitsorozatot, numerikus operandust stb.). Ilyenkor a gép egymástól függetlennek tekinti az operandus egyes bitjeit, és mindegyikkel külön-külön végzi el a kijelölt müveletet. A logikai adat hossza ilyenkor a bitsorozattól függően változó, de mindig egész számu byte-ot



53. ábra. Logikai adat ábrázolása fix hosszon (egy byte-on)

igényel. (A könnyebb kezelhetőség miatt a bitsorozatokat itt is hexadecimálisan irjuk fel.) Tekintsük logikai adatnak az alább hexadecimális formában megadott két bitsorozatot (operandust) és végezzünk közöttük logikai VAGY müveletet!

$$
\begin{array}{lll}
 & F\ 5\ A\ 9_{(16)}\ ;\ 4\ D\ B\ 2_{(16)} & \\
 & 1111\ 0101\ 1010\ 1001_{(2)} & (=\ F\ 5\ A\ 9) \\
\text{VAGY} & \underline{0100\ 1101\ 1011\ 0010_{(2)}} & (=\ 4\ D\ B\ 2) \\
 & 1111\ 1101\ 1011\ 1011_{(2)} & \\
 & \quad F \quad D \quad B \quad B &
\end{array}
$$

Igy a logikai VAGY müvelet eredménye $FDBB_{(16)}$.

3. A központi számítógép

3.1 A KÖZPONTI SZÁMÍTÓGÉPRŐL ÁLTALÁBAN

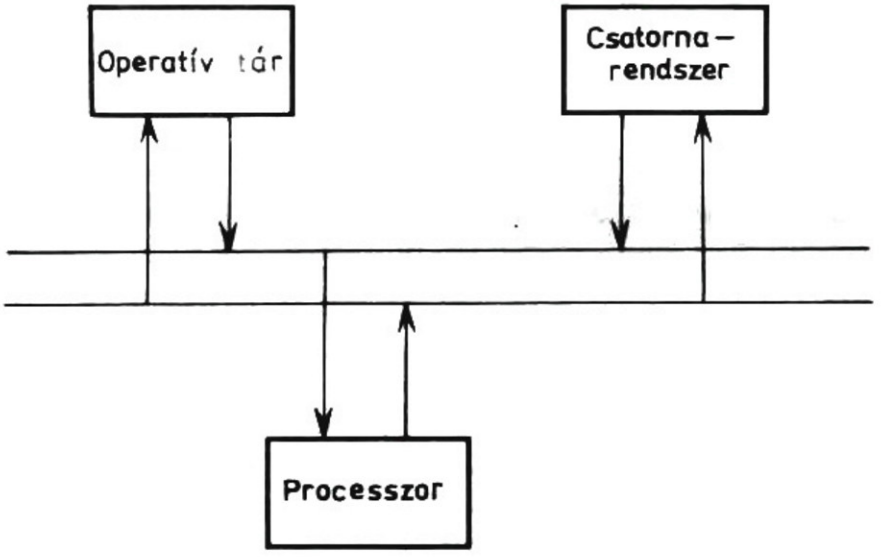
3.1.1 A KÖZPONTI SZÁMÍTÓGÉP FELÉPÍTÉSE

A központi számitógép a teljes rendszernek az a része, amely helyzeténél és szerepénél fogva egyaránt központi helyet foglal el. Itt van az egész rendszer müködését irányitó vezérlő egység, a számitási müveleteket végző aritmetikai-logikai egység és az adatok és végrehajtandó programok tárolását szolgáló operativ tár. A vezérlő egységet és aritmetikai-logikai egységet együttesen processzornak nevezzük.

A régebbi számitógépeknél a központi gép és a perifériakészlet nem vált el olyan élesen egymástól, mint a mai korszerü komputereknél. Ma már több központi számitógépet is össze lehet kapcsolni egymással akár közvetlenül, akár valamilyen adatátviteli vezérlő egységen keresztül. Minthogy ebben az esetben több processzor müködik együtt egy közös rendszerben, az ilyen géprendszereket multiprocesszoros (többprocesszoros) rendszereknek nevezzük, a rájuk jellemző feldolgozási mód pedig az ugynevezett multiprocessing. A központi számitógép vázlatos felépitését az 54. ábrán láthatjuk.

Ez az ábra nagyon általánosan ábrázolja a központi gépet; elemei adatátviteli sinekkel kapcsolódnak egymáshoz.

Többprocesszoros rendszerekben minden központi számitógépnek lehet saját operativ tára, de használhatják egymás operativ tárát is.



54. ábra. A központi számitógép felépitése

3.1.2 AZ OPERATÍV TÁR FELÉPÍTÉSE. TÁRRÉSZEK ÉS CÍMZÉSÜK

Az operativ tár a központi gépnek az a része, amelyben a futó programok és a hozzájuk tartozó adatok találhatók. A tárat rekeszekre osztják fel, minden rekesznek van egy sorszáma. A sorszámozás 0-tól folyamatos. A rekeszek sorszámát a rekesz címének nevezzük. A rekeszek mérete géptipusonként változó lehet. Általában 8 bittől 64 bitig terjed. A 8 bites rekeszekből felépitett tárat byteszervezésünek nevezzük.

Valamely rekeszbe információt beirni vagy kiolvasni ugy lehetséges, hogy előbb megcimezzük az illető rekeszt. A cimzés azonos a rekesz megkeresésével. Mindez természetesen szimbolikusan értendő, a megcimzés és információelérés is villamos impulzusok segitségével történik.

Funkcionális szempontból az operativ tár nem egységes, hanem több önálló részre oszlik. Egy rész a gép felhasználója számára is hozzáférhető, cimezhető. Ezt főtárnak nevezzük. A tár többi része általában nem vagy csak részben cimezhető a programból; ezek a tárterületek a központi számitógép funkcionális egységeinek munkaterületei. Igy például külön területet használ a csatornarendszer az input/output müveletek lebonyolitásához, megint más területről dolgozik a tárvédelmi mechanizmus stb. Az elérendő tárterület kiválasztása a mikroutasitás feladata, ennek módját később részletesen is megismerjük.

3.1.3 AZ ARITMETIKAI-LOGIKAI EGYSÉG FELÉPÍTÉSE, FUNKCIÓI

Az aritmetikai-logikai egység a központi gép müveletvégzője. Ez végzi el az összes számitási, logikai, léptetési müveletet, és általában minden programutasitás végrehajtásában részt vesz. Itt történik az operanduscimek kiszámitása is. Ahhoz tehát, hogy adatokkal müveleteket végezhessünk, az illető adatokat előbb be kell vinni az operativ tárból az aritmetikai-logikai egységbe. Itt azután több regiszter használható az operandusok, részeredmények, végeredmények tárolására. A kiszámitott eredményeket ez az egység azonnal továbbadja, például az operativ tárnak vagy a csatornarendszernek.

Az aritmetikai-logikai egység felépitése és bonyolultsága géptipusonként nagyon eltérő lehet. A teljes géprendszer müködési sebességét nagymértékben befolyásolja, a gép árába is jelentékenyen beleszól.

3.1.4 A VEZÉRLŐ EGYSÉG FELÉPÍTÉSE, A MIKROPROGRAMOZOTT ÉS HARDWARE VEZÉRLÉS FOGALMA

A vezérlő egység a hardware architektura legmagasabb pontján áll; ez irányitja a rendszer összes többi elemének müködését. A vezérlő egység az, amely lépésről lépésre "megvalósitja" a felhasználó elképzelését, követi a feladat megoldás algoritmusát. Mindezt a bevitt program utasitásai alapján teszi. Mivel programunk futás alatt az operativ tárban található, a vezérlő egység rendszeresen az operativ tárhoz fordul, és onnan egy vagy több programutasitást, ugynevezett gépi utasitást "kiolvas". Az utasitások szerkezete olyan, hogy meghatározza a végzendő müveletet, valamint azt, hogy a müveletvégzéshez

szükséges adatok (számok) az oprativ tár mely rekeszeiben találhatók. A vezérlő egység tehát "értelmezi", megfejti a gépi utasításokat, kiszámitja belőlük az operanduscimeket, az operandusokat előkeresi, majd a megadott müveletet végrehajtatja az aritmetikai-logikai egységgel. Ez a folyamat ismétlődik mindaddig, mig a programvégrehajtás be nem fejeződik. Ugyancsak a vezérlő egység az, amely megvizsgálja minden utasitás végrehajtása előtt, hogy van-e ok a program megszakitására. Programmegszakitásra azért van szükség, hogy futás alatt is biztositsuk a programmal való kommunikációt bizonyos események részére. Ezek az események a megszakitás során juthatnak a program tudomására, igy a programnak lehetősége van megfelelően intézkedni ezen események kezeléséről. Például ilyen esemény az, ha egy input/output müvelet befejeződött, a programban hiba van stb.

A vezérlő egység két, alapvetően különböző módon dolgozik: mikroprogramozottan vagy hardwarevezérelten. Egy adott vezérlő egység természetesen vagy mikroprogramozott, vagy hardware vezérlést valósit meg. Némely számitógépnél keveredik a kétféle vezérlési mód: bizonyos funkciók mikroprogramozottak, mások nem.

Mi a lényege a mikroprogramozott és hardware vezérlésnek?

Mikroprogramozás esetén a vezérlő egységben létezik egy olyan program, amely a gépi utasitások végrehajtását végzi. Azért indokolt a program elnevezés, mert az egyes gépi utasitások általában több elemi lépésből tevődnek össze; ilyen például az utasitás kiolvasása a tárból, operanduscimek kiszámitása, operandusok bevitele az aritmetikai-logikai egységbe stb. Egy-egy ilyen elemi lépés egyenértékü a mikroprogram egyetlen utasitásának, a mikroutasitásnak végrehajtásával. A mikroprogramok általában rögzitettek, de a legkorszerübb számitógépeknél kismértékben módosithatók is.

Hardware vezérlés esetén a gépnek nincs mikroprogramja; ilyenkor a vezérlő egység a meglevő áramköri kapcsolatok alapján dolgozik. Ilyenkor nem egy mikroprogram utasitásainak végrehajtása valósitja meg a gépi utasitásokat, hanem az áramköri elemek egymást követő állapotai. Ezek bonyolult módon össze vannak huzalozva egymással, és a sorozatos állapotváltoztatások segitségével hajtódnak végre a gépi utasitások. Ebből kitünik, hogy a mikroprogramozott vezérlési mód jóval egyszerübb a hardware-nél, utóbbi viszont jóval gyorsabb, drágább. Ezért hardware vezérlést olyan gépeknél alkalmaznak, amelyek müveleti sebességének a lehető legnagyobbnak kell lennie (főleg nagy számitógépeknél, illetve kis célszámitógépeknél).

3.1.5 A CSATORNARENDSZER FELADATA. CSATORNAFAJTÁK. AZ ÖNÁLLÓ INPUT/OUTPUT VEZÉRLÉS ALAPGONDOLATA. A CSATORNA-INTERFACE JELENTŐSÉGE, FUNKCIÓI

A számitógéprendszer müködése során rengeteg input/output müveletet kell elvégezni. Az adatok beolvasása, eredmények kiirása, állandó kapcsolat a háttértárakkal, mind-mind input/output müveletet igényel. Ha ezt mind a vezérlő egység irányitaná, akkor jóformán nem is maradna ideje a program többi utasitásának végrehajtására! Indokolt tehát az, hogy az input/output vezérlést a vezérlő egységről leválasszuk, őt ez alól tehermentesitsük. Erre alakitották ki a csatornarendszert. A csatornarendszer tehát kisebb vezérlő egységekből áll, és feladata csupán az input/output müveletek lebonyolitása.

Kétféle csatorna terjedt el a gyakorlatban: a multiplex és a szelektorcsatorna.

A multiplex csatorna időosztásosan több perifériával dolgozik; ez azt jelenti, hogy mig például a sornyomtatón soremelés történik, azalatt egy másik perifériáról lehet adatot olvasni stb. Mindez azért lehetséges, mert a multiplex csatorna perifériái "lassuak", müködési sebességük nem mérhető össze a csatorna és az adatátvitel sebességével. Ugy tünik a szemlélőnek, hogy az összes periféria "egyszerre" megy.

A szelektorcsatorna ezzel szemben a "gyors" perifériákat szolgálja ki, és egyszerre csak eggyel foglalkozik. Itt ugyanis nincs ideje a csatornának arra, hogy "körbejárja" valamennyi perifériáját, mert az adatátvitel sebessége igen nagy.

A csatornához kapcsolt perifériák egy mindkét végén illesztő áramkörökkel ellátott szabványos kábelen keresztül csatlakoznak a csatorna áramköreire. Ezt nevezzük illesztő egységnek, csatorna-interface-nek. Az interface-en haladnak az olvasott vagy kiirandó byte-ok, különféle kódok, ellenőrző villamosjelek stb. Ez valósitja meg a csatorna-periféria kapcsolatot. Adatátvitel közben a csatorna állandó kapcsolatban áll a perifériával, és minden egyes adat a kisérő és a megérkezést nyugtázó jelekkel együtt halad át az interface-en. Erre azért van szükség, hogy növeljük az input/output müveletek biztonságát, és elkerüljük az adatvesztést, adattorzulást.

Mivel az interface szabványositott, minden csatornához minden periféria csatlakoztatható. Az ajánlott csatlakoztatási módok betartása azonban növeli a géprendszer teljesitményét, sebességét.

3.1.6 A KEZELŐPULT SZEREPE, FELÉPÍTÉSE. AZ AUTONÓM ÜZEMMÓD JELENTŐSÉGE

A kezelőpult a központi számitógépnek az a' része, amelyről a gépkezelő közvetlenül beavatkozhat a rendszer bármely területére. Elérheti a nem cimezhető tárterületeket, regisztereket stb. A pulton kapcsolók, nyomógombok és indikátorlámpák találhatók. Ezek értesitik a gépkezelőt a gép állapotáról, müködéséről. A pultról lehetőségünk van egy programot utasitásonként, sőt mikroutasitásonként is végrehajtani. Erre akkor kerülhet sor, ha hibát keresünk a hardware-ben, vagy egyéb tesztelési célunk van vele. Normális müködés esetén a pulttal a gépkezelőnek nem sok dolga akad; müszakkezdéskor azonban ennek segitségével lehet a tárba tölteni az operációs rendszert, amely az automatikus üzemeltetést végzi.

3.2 AZ OPERATÍV TÁR

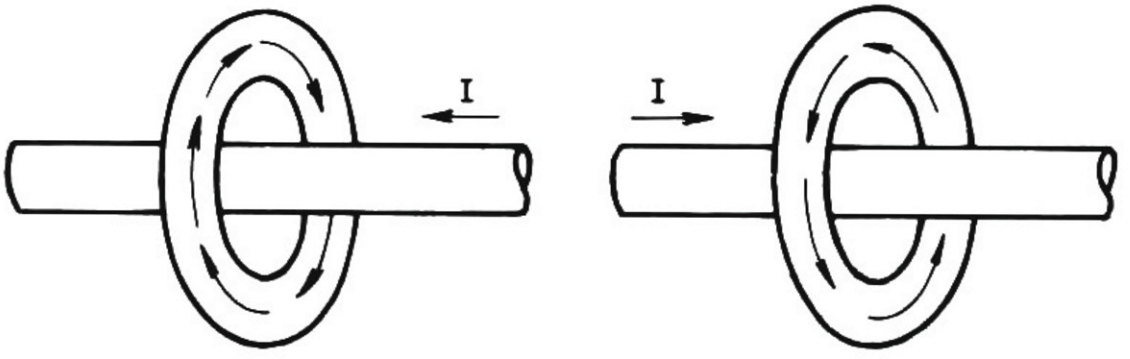
3.2.1 MÁGNESES ÉS ELEKTRONIKUS INFORMÁCIÓTÁROLÁSI ELVEK

A mai korszerü számitógépek operativ tárát alapvetően kétféle fizikai információtárolási elv alapján tervezik: a mágneses és az elektronikus tárolási elv alapján. Mágneses elvü információtárolásnál az anyagoknak azt a tulajdonságát használjuk ki, hogy egyes anyagok - a mágnesezhetők - a felmágnesezés után még hosszu ideig megtartják mágneses állapotukat. Igy, ha egyszer a tárba valamit beirtunk, akkor az mindaddig változatlanul fennmarad, mig ujabb információt nem irunk a helyére. Ilyen anyagok

például a ferritek, amelyek viszonylag könnyen átmágnesezhetők. Elektronikus tárolási elv alkalmazása esetén az információt elektronikus alapáramkörök tárolják. Ezeknek állandó energiautánpótlásra van szükségük, mert az információt elektromos feszültség megléte vagy hiánya jelenti. Általában azt mondhatjuk, hogy az elektronikus müködésü tárak gyorsabbak, mint a mágneses müködésüek. Mégis, az operativ tárak nagy részét mágneses tárolóelemekből épitik fel, mert az elektronikus tárolóelemek előállitása költségesebb. A tármüködés gyorsitására különböző szervezési megoldásokat alkalmaznak.

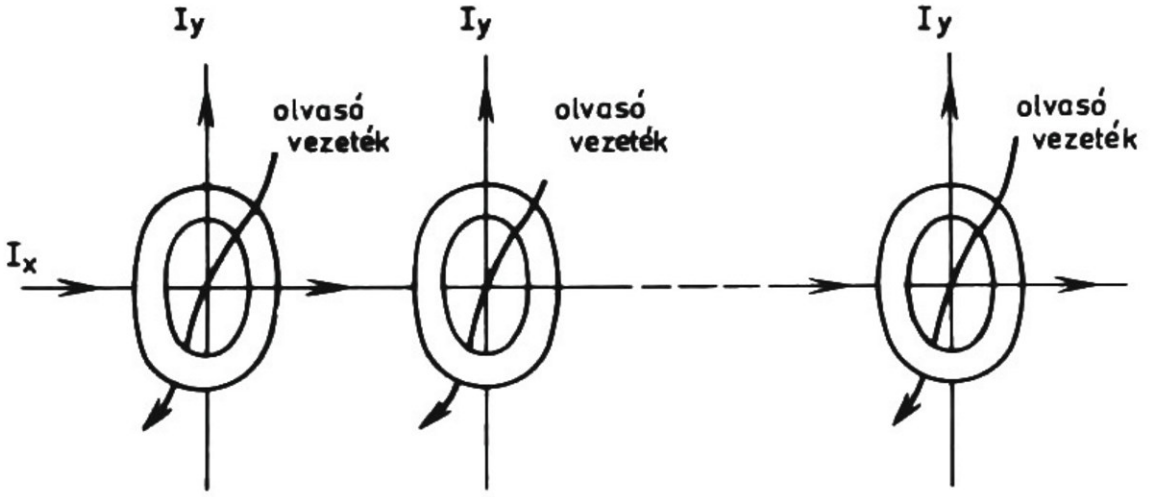
3.2.2 A FERRITGYŰRŰS TÁR FELÉPÍTÉSE, MŰKÖDÉSE. A TÁRCIKLUS LEZAJLÁSA. A FERRITGYŰRŰS TÁR REGISZTEREI. PÉLDÁK TÁRMŰVELETEKRE

A ferritgyürüs tárolónál a remanens (maradandó) mágnesezettség jelenségét használjuk ki. Minden egyes bitnek megfeleltetünk egy ferritgyürüt, amely kétféle mágnesezettségi állapotban lehet. Ugyanigy egy bit információtartalma is csak "1" vagy "0" lehet. Ez képezi a megfeleltetés alapját: az egyik irányu mágnesezettséget 1-nek, az ezzel ellentétes irányut pedig 0-nak tekintjük. A lehetséges mágneses irányokat az 55. ábra tünteti fel.



55. ábra. A ferritgyürüs információ-tárolás elve

A mágnesezettség beállitását ugy képzelhetjük el, mint a mágnesezhető anyag rendezetlen mágneses állapotának rendezetté való átalakitását. Ezt az átalakitást a ferritgyürün keresztülvezetett villamos vezetéken folyó áram végzi, saját mágneses tere segitségével. A mágnesező áram megszünése után a ferritgyürü továbbra is megtartja beállitott mágneses állapotát. Most, hogy elvileg megoldottuk egy bit információ tárolását, arról kell csupán gondoskodnunk, hogy nagyobb információegységeket is tudjunk cimezni. Ebből a célból ugy alakitjuk ki az operativ tárat, hogy az 1-1 bit tárolására használt ferritgyürüket valamilyen huzalozási rendszerbe épitjük be. A szokásos gyürükiválasztási módszer az ugynevezett koordinátaáramok módszere. Ennek alapgondolata az, hogy a ferritgyürü átmágnesezéséhez meghatározott nagyságu áramerősségre van szükség, ez alatt a gyürü nem változtatja meg állapotát. Az átmágnesező áramot két részre osztjuk: két különálló vezetékben folyó áram együttesen hozza létre a szükséges mágneses teret. Egymaga egyik áram sem képes a gyürüt átmágnesezni. Ezt az elvet a féláramu kiválasztás elvének nevezzük. Az átmágnesezésnél lényeges, hogy a két vezeték mágneses tere összeadódjon, ne pedig közömbösitse egymást. Ezért ugy választják meg az áramirányokat, hogy azok azonos értelemben mágnesezzenek. Az elmondottakat szemlélteti az 56. ábra.



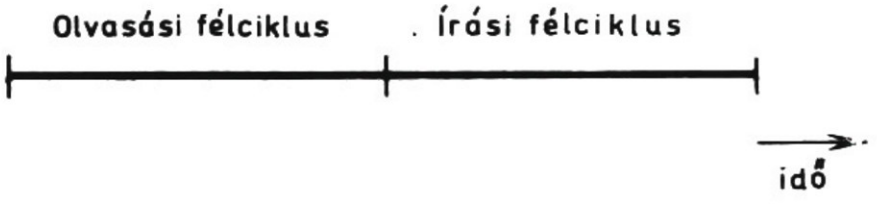
56. ábra. Ferritgyürük féláramu kiválasztása

Az ábrán több gyürüt füztünk fel az I_x jelü vezetékre; ezek az x irányu áramot valamennyien megkapják. Ez azonban csak "féláram", nem elég az átmágnesezéshez. Ezért áthuztuk a gyürükön I_y jelü vezetékeiket is; az $I_x + I_y$ áram együttesen már elég az átmágnesezéshez. Tehát a kiválasztás y irányban szelektiv.

Az x és y irányok bevezetésével tehát már tudjuk cimezni az egyes ferritgyürü-csoportokat. A cimzés legtöbbször egy byte-ra vonatkozik, vagyis egy adott memóriacim egy byte-nyi információ elérését teszi lehetővé. Nagyobb gépeken a byte többszöröseit szokták egy cimmel kiválasztani; ilyenkor azonban a cimben valamilyen utalás található arra, hogy a több byte-os mező mely byte-jai kivánatosak.

Megoldásra vár még az a feladat, hogy a tárba most már egy megcimzett területre tetszőleges információt beirjunk, illetve e terület tartalmát kiolvassuk. Ezt a tárak időben két önálló, de egymással szorosan összefüggő lépésben oldják meg, a lépések neve: félciklus. Kétféle félciklus van: olvasási és irási félciklus. Ezek sorrendje mindig a fenti, és közöttük semmilyen tármüvelet nem megengedett. A két félciklus együtt alkotja a tárciklust, amelynek időtartama a tár fontos jellemző adata. A tárciklus ideje mai gépeknél a μs nagyságrendjébe esik.

A tárciklus lezajlását mutatja az 57. ábra:



57. ábra. A tárciklus lezajlása

Az olvasási félciklus során a megcimzett ferritgyürüket definiciószerüen "0" állapotba hozzuk. Ez nagyon egyszerüen megoldható, ha megállapodunk például abban, hogy $-I_x$ és $-I_y$ áramok hatására a gyürü "0" állapotba kerül. Ekkor az egy byte-on átmenő I_x vezetékre, és valamennyi bitjéhez tartozó I_y vezetékre "negativ irányu" áramot adunk. Ezzel valamennyi gyürü azonos, "0" állapotba

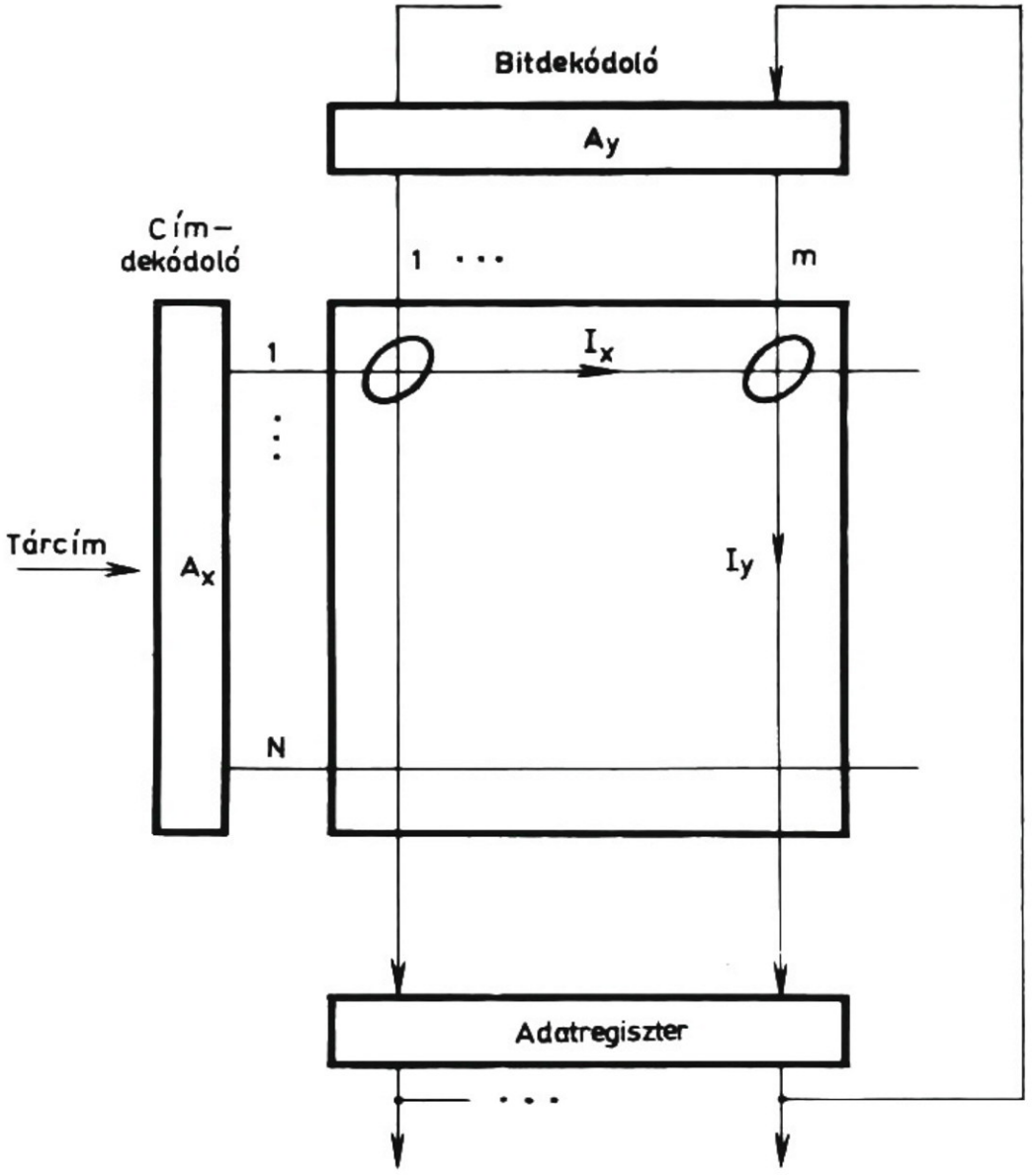
kerül. Az átmágnesezés alatt csupán azt kell figyelnünk, hogy melyik gyürü váltott irányt mágnesezettség szempontjából. Azok ugyanis, amelyek "billentek", korábban "1" állapotban voltak, amelyek nem billentek, azok korábbi állapota "0" volt. Meg kell tehát őriznünk a 0-ba billenés eseményét, erre szolgál a gyürükön átvezetett olvasóvezeték. Ez arra való, hogy jelezze a megcimzett gyürük esetleges állapotváltását. Ahogy az áram mágneses teret hoz létre maga körül a gyürü átmágnesezésekor, ugyanugy a mágneses tér megváltozása áramot fejleszt az olvasóvezetékben. Egy byte kiolvasásakor tehát 8 db olvasóvezetékkel dolgozunk; amelyik vezetéken áramlökés jelentkezik, annak megfelelő bit információtartalma "1" volt. Az olvasóvezetékeket közvetlenül a tár adatregiszterébe vezetjük, és ott tároljuk ideiglenesen a kiolvasott információt. Ha erre szükségünk van, akkor továbbitjuk a rendeltetési helyére, például valamelyik csatornába, a központi számitógép regisztereibe stb. Ha nincs rá szükségünk, akkor "veszni hagyjuk".

Az irási félciklus során a tár adatregiszterének tartalmát irjuk be a megcimzett területre. Ez lehet a kiolvasott információ, de lehet uj információ is. Ha a kiolvasott információt irjuk vissza, akkor tárciklus során kiolvasás-regenerálás tármüveletet végzünk. Regenerálásra azért van szükség, mert az olvasási félciklus alatt törlődik a tárterület tartalma, és esetleges későbbi felhasználásnál szükség lehet az ott tárolt információra. Ha a kiolvasott információ helyett uj információt irunk be, akkor törlés-beirás tármüveletet végzünk, ilyenkor a kiolvasott információt nem tároljuk, hiszen éppen annak felülirása a cél. Speciálisan, ha csupa zérust irunk be, ez a törlés. Ehhez is szükség van azonban arra, hogy a megcimzett tárterület tartalmát az olvasási félciklus alatt kiolvassuk!

Összefoglalva: minden tármüvelet egy teljes tárciklust jelent. A tármüvelet tipusát az dönti el, hogy az irási félciklus során milyen információ kerül vissza a tárba. A tárciklus alatt a cimregiszter tartalmát nem lehet megváltoztatni, mert ellenkező esetben a visszairás máshova történhetne, mint a kiolvasás.

A tárak gyakorlati megvalósitásában három fő tipus terjedt el: a kétdimenziós (szószervezésü), a három dimenziós (bitszervezésü) és a 2,5 dimenziós tár. Előnyös tulajdonságai miatt leginkább a 2,5 dimenziósat használják, ezért ezt mutatjuk be egy példán. Speciális esetként adódik a 2 dimenziós tár. A 2,5 dimenziós tárban a ferritgyürüket sakktáblaszerüen, ugynevezett mátrix alakban rendezzük el. Ez azt jelenti, hogy több gyürüt egy memóriaszóba fogunk össze, és ezeket felfüzzük az x irányu ugynevezett szóvezetékre. Y irányban minden gyürün 1-1 ugynevezett bitvezetéket huzunk át, a harmadik az olvasóvezeték. Ha feltételezzük, hogy N darab memóriaszót kell cimeznünk, és egy-egy x irányu szóvezetékre c darab szó ferritgyürüit füzzük fel, akkor összesen $\frac{N}{c}$ darab szóvezetékre van szükség. Ha minden egyes memóriaszó m bites, akkor $m \cdot c$ darab bitvezeték kell. A c=1 eset felel meg a 2 dimenziós tárnak. Ennek felépitése az 58. ábrán látható.

Általában c értéke nagyobb, mint 1, ez már "valódi" 2,5 dimenziós elrendezéshez vezet. Itt egy-egy szóvezetéken c db szó található, összesen tehát $c \cdot m$ bit. A c-edik bitek tehát ugyanazt a helyértéket tárolják, csak más-más memóriaszóban. Összesen m darab ilyen memóriaszelet van, mindegyiken szóvezetékenként c darab ferritgyürüvel. X irányban tehát c darab szó kiválasztása történne meg, de a féláramu kiválasztás miatt y irányu áramokra is szükség van. Y irányban oldják meg a c darab szóból az egyetlen szónak a cimtől függő kiválasztását, a többi (c-1) szóra az x irányu kiválasztás egymagában hatástalan. Az elmondottakat az 59. ábra szemlélteti.

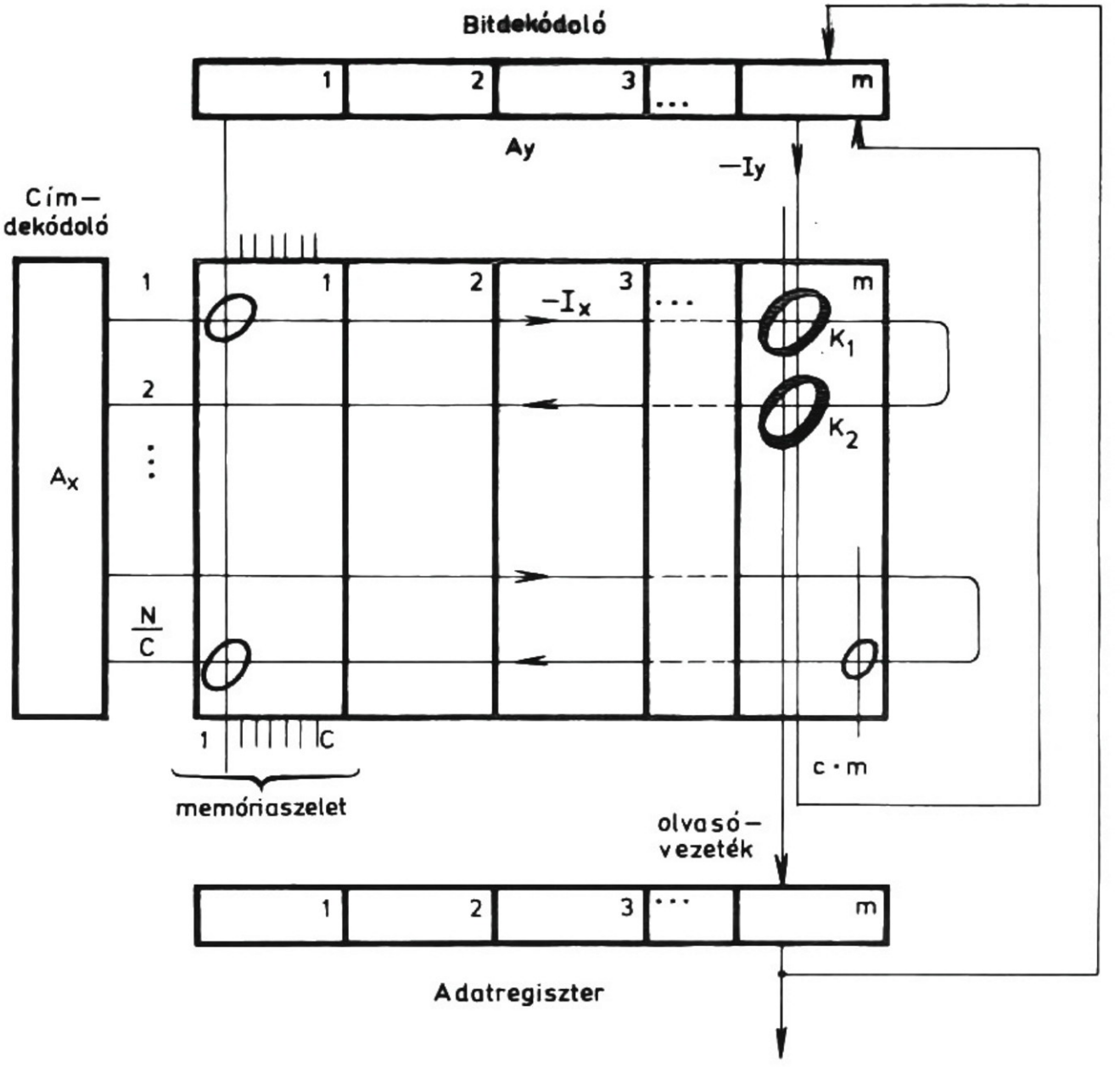


58. ábra. A kétdimenziós tár felépitése

Az olvasási félciklus alatt $-I_x$ és $-I_y$ áramok hatására megtörténik az 1. szövezetéken található ferritgyürük kiválasztása, igy például a K_1 jelü gyürü állapota "0"-ra változik. Mivel azonban K_2 gyürüt az x irányu áram ellentétes irányban éri, ennek állapota nem változik! Ha K_2-t akarjuk "0"-ba billenteni, akkor $+I_x$ és $-I_y$ áramokat kell rajta átvezetnünk. Az irási félciklusban $+I_x$ vagy $-I_x$ segitségével ismét kiválasztjuk a megfelelő memóriaszókat, de a beirandó információtól függően csak azok az I_y vezetékek kapnak áramot, ahová "1"-et kell irni. Ezért kell többek között beirás előtt is mindig "0" állapotba hozni az érintett tárterület ferritgyürüit, ugyanis "0"-t beirni nem tudunk, csak "1"-et! A beirandó információt az adatregiszter tartalmazza.

Nézzünk meg két gyakran előforduló tármüveletet: a kiolvasás-regenerálást és a törlés-beirást.

Ennél a tármüveletnél a cimregiszterbe betöltött cim hatására megtörténik a megcimzett ferritgyürük kiválasztása (1). A tárrekesz információtartalma az adatregiszterbe kerül (2), ezzel a rekeszt nulláztuk. A kiolvasott adatot innen továbbvezetjük feldolgozásra, egyidejüleg azonban vissza is irjuk ugyanarra a cimre, ahonnan kiolvastuk (3). Látható, hogy a visszairás függ az adatregiszter tartalmától.

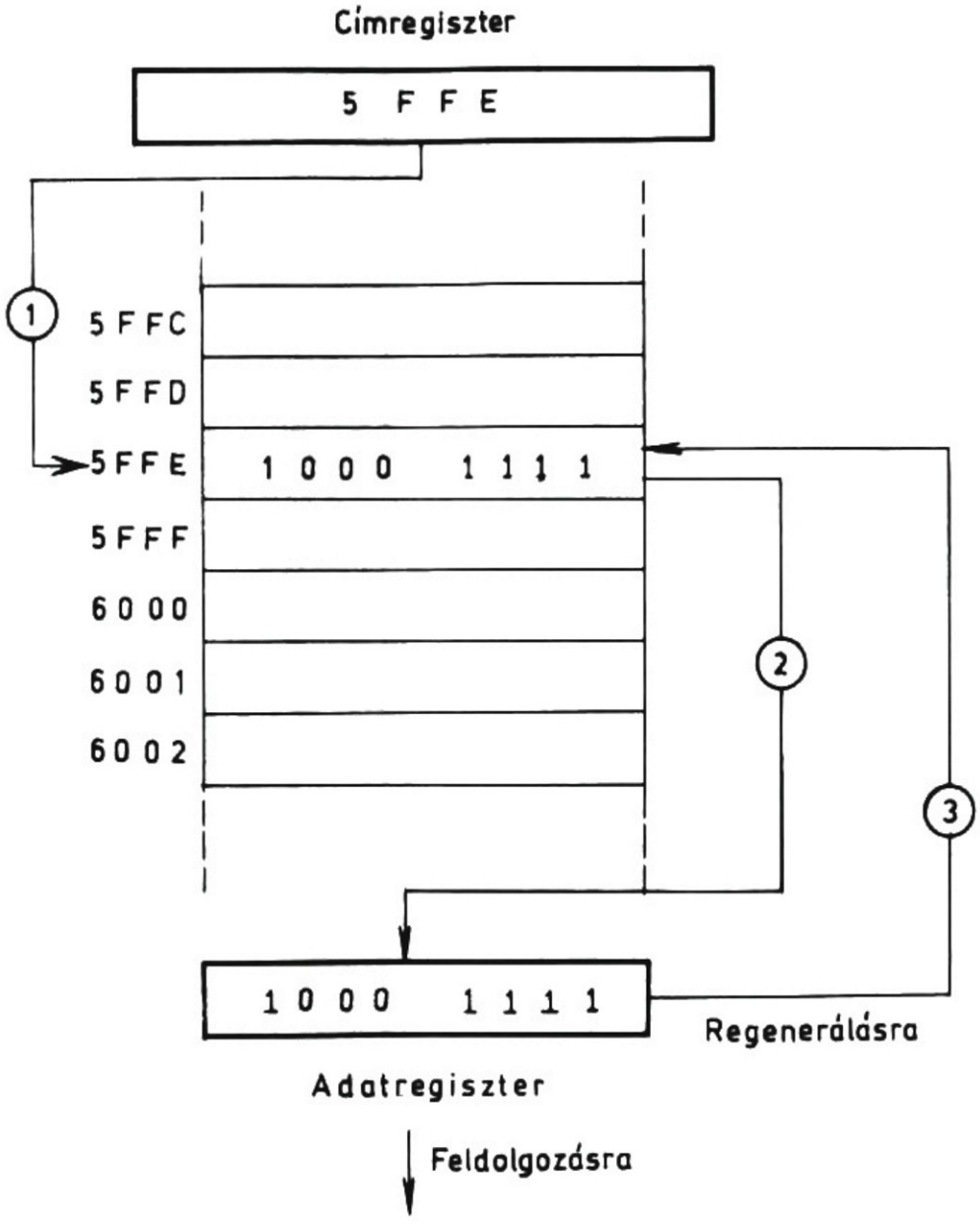


59. ábra. A 2,5 dimenziós tár felépitése

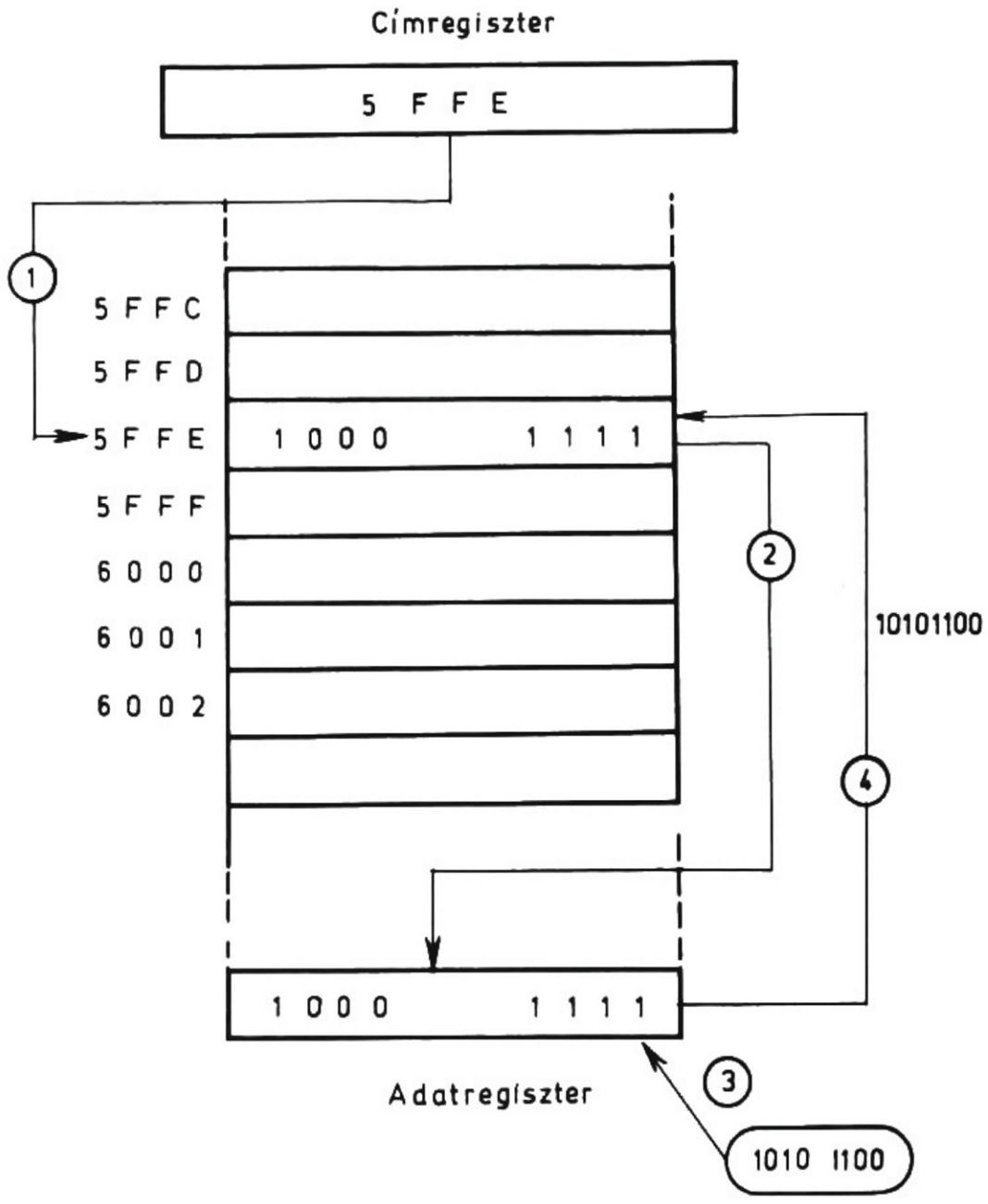
Törlés-beirás tármüveletnél uj információt akarunk a tárba bevinni.

Itt is először kiválasztjuk a megcimzett rekeszt a cimregiszter tartalma alapján (1). Az olvasási félciklus alatt a rekesz tartalma az adatregiszterbe kerül (2), ahonnan azonban nem vezetjük el feldolgozásra! Ehelyett az adatregiszterbe betöltjük az uj információt (3), és az irási félciklus alatt ezt irjuk be a tárba (4). Látjuk, hogy a két tármüvelet csak abban tér el egymástól, hogy melyik információt irjuk vissza.

Az operativ tárak nagysága, sebessége géptipusonként eltérő lehet. Ugyanigy különböző az egyszerre kiolvasható memóriaterület nagysága is. Például az R20 számitógép egy tárciklus alatt 2 byte-ot olvas ki és ir vissza, az R40 modell viszont 8 byte-ot. Ha csupán egyetlen byte-ra van szükségünk, akkor a cimben utalás található arra, hogy a kiolvasott byte-ok közül hányadikra van szükség (például az R40 tárcimében 3 bit szolgál arra, hogy eldöntse, a 8 kiolvasott byte közül hányadikra van szükségünk).



60. ábra. Kiolvasás-regenerálás tármüvelet

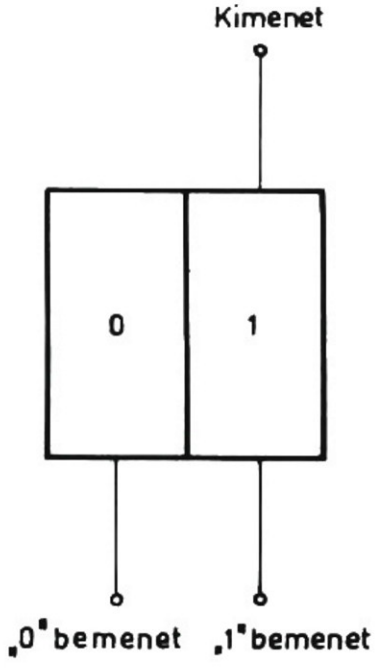


61. ábra. Törlés-beirás tármüvelet

3.2.3 AZ ELEKTRONIKUS TÁROLÓELEM FELÉPÍTÉSE, MŰKÖDÉSE

A mágneses tárolóelemtől eltérően, itt nem az anyagok valamilyen fizikai tulajdonságát használjuk az információ megjelenitésére, hanem olyan elemi áramköröket épitünk fel, amelyek két egymástól jól megkülönböztethető állapotban lehetnek. Az állapotot ugy értelmezzük, hogy az áramkörök kimenetén van vagy nincs villamos feszültség. Ezután például azt az esetet, amikor a kimeneten van feszültség, megfeleltetjük az "1", amikor nincs feszültség, azt pedig a "0" állapotnak. Igy tudunk bináris számokat tárolni. Mivel azonban az információhordozó itt feszültség, és ez az anyagnak nem maradandó tulajdonsága, az elektronikus tárolóelem csak akkor müködik, ha állandó energia utánpótlást biztositunk számára. Más szavakkal: a tápfeszültség kikapcsolása után az elektronikus tároló elvesziti információtartalmát, és bekapcsolás után előre meg nem jósolható állapotot vesz fel. Ezért van az, hogy a számitógép bekapcsolása után valamennyi elektronikus regisztert nullázzuk, hogy meghatározott állapotba hozzuk.

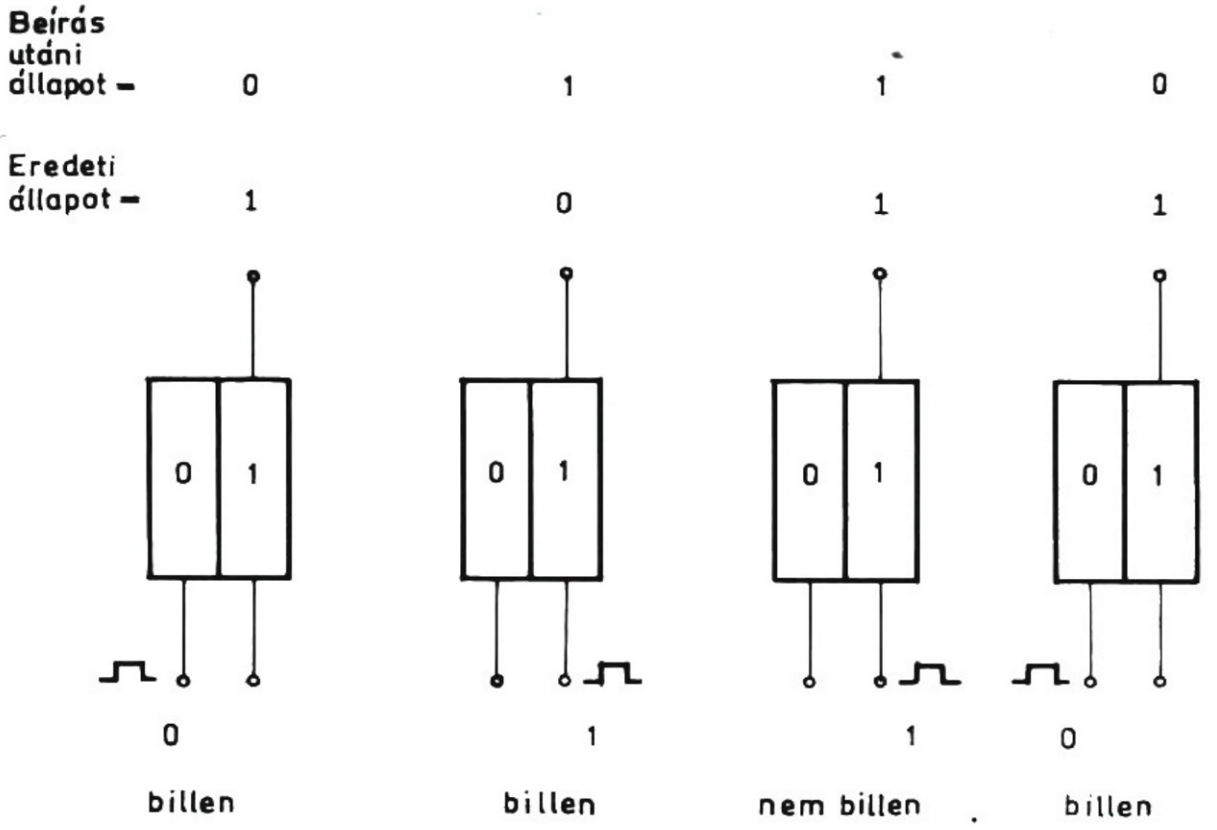
Az elektronikus tárolóelemet flip-flopnak nevezzük. Vázlatos jelölése a 62. ábrán látható.



62. ábra. Elektronikus tárolóelem

A tároló állapotát a kimenetén érzékelhetjük. A kivánt információtartalmat a bemenetekre adott áram- (feszültség-) impulzus segitségével hozhatjuk létre. Ha az "1" bemenetre adunk impulzust, akkor a tároló "1" állapotba kerül, ha a "0" bemenetre, akkor "0"-ba. Természetesen, egyidejüleg mindkét bemenetre impulzust adni értelmetlen dolog. Igen nagy előnye a tárolónak, hogy a kiolvasás nem törli információtartalmát, hiszen csak a kimeneti feszültséget kell "megmérni". Azok a flip-flopok, amelyek uj állapota megegyezik a beállitandó állapottal, nyugalomban maradnak.

A 63. ábra egy 4 bites elektronikus tároló müködését mutatja.



63. ábra. Beiró impulzusok hatása az elektronikus tárolóelemre

Az eredeti állapot 1011 volt, az uj állapot 0110. Ennek érdekében az első és negyedik flip-flopot nullázni kell, a második és harmadikat pedig "1"-be kell állitani. Mivel a beállitás független a korábbi információtartalomtól, a második és harmadik flip-flop "1" bemenetére, és az első, negyedik flip-flop "0" bemenetére adunk impulzust. Látjuk, hogy a harmadik flip-flop nem változtatja meg állapotát, erre nézve a beiró impulzus hatástalan. A többi flip-flop állapotot vált. Az állapotváltás ideje néhányszor 10 ns körül van.

3.2.4 A KÉTFÉLE TÁROLÁSI ELV ÖSSZEHASONLÍTÁSA

Jóllehet mindkét információtárolási elvnek megvan a maga előnye és hátránya, nem lehet egyértelmüen állást foglalni az egyik vagy másik elv mellett. A különböző tulajdonságok más-más területen az egyik, illetve másik tárolási elv alkalmazását teszik indokoltabbá. Általában azt mondhatjuk, hogy a ferritgyürüs tárak alkotják a számitógéppark nagyobb részénél az operativ tárat. Hátrányuk, hogy minden tármüveletnél a teljes tárciklust le kell folytatni, és ez meghatározza az elérhető maximális müködési sebességet. Jó ferritgyürüs tárnak tekinthető az, amelynek ciklusideje 1 μs körüli értékü. Mivel a programok és adatok közvetlenül az operativ tárból kerülnek a vezérlő egységbe feldolgozásra, a vezérlő egység sebességét főleg az operativ tár sebessége korlátozza. Előnye viszont az ilyen tárnak, hogy a feszültségkimaradás nem vezet információvesztéshez. Az elektronikus tárolási elvet általában ugy használják ki, hogy az operativ tár egy részét - gyorsregisztereket, gyorstárat - elektronikus elemekből épitik fel. A leggyakrabban szükséges információkat azután itt helyezik el, igy a

hozzáférési idő jelentősen javul. Müszaki hiba esetére gondoskodnak a regiszterek automatikus elmentéséről, igy hidalják át azt a nehézséget, hogy az elektronikus tár érzékeny a feszültségkimaradásra. Flip-flopokból épülnek fel a vezérlő egység, a csatornák regiszterei is, mert itt igen lényeges a gyors müködés. A kétféle tárolási elv néhány tulajdonságát a 7. táblázatban foglaltuk össze.

7. táblázat

Ferritgyürüs és elektronikus tárolási elv összehasonlitása

Szempontok	Ferritgyürüs tároló	Elektronikus tároló
Ár	olcsóbb	drágább
Sebesség	lassubb	gyorsabb
Olvasás töröl-e	igen	nem
Feszültségkimaradás töröl-e	nem	igen

3.2.5 AZ OPERATÍV TÁR RESZEI. LOKÁLIS TÁR, MULTIPLEX TÁR, FŐTÁR, KULCSTÁR, ÁLTALÁNOS ES LEBEGŐPONTOS REGISZTEREK ÉS CIMZÉSÜK

Funkcionális szempontból az operativ tár általában nem egységes, hanem több részre osztható. Ennek az a magyarázata, hogy a tárat nemcsak a gépen futó programok használják, hanem a számitógéprendszer más elemei is, például a csatornák. A programok számára fenntartott tárrészt szokták főtárnak nevezni, a többit pedig kiegészitő tárnak. Példaképpen az R20 számitógép operativ tárát vizsgáljuk meg. Itt a tár négy fő részre osztható: főtár, lokális tár, multiplex tár és kulcstár. Az egyes tárrészek közös cimregisztert használnak, igy egyszerre csak az egyikhez lehet hozzáfordulni. A tárterület kijelölését a mikroutasitás Tárterület nevü 2 bites mezője végzi; ez dönti el, hogy a beállitott cim melyik tárrészbe mutasson.

A főtár szolgál a futó programok és adataik tárolására. Ebben foglal helyet a számitógéprendszert vezérlő operációs rendszer egy része is. A főtár byte - szervezésü; minden egyes byte-nak saját cime van. A cimzés 0-tól kezdődik. A főtár első 256 byte-jának rögzitett funkciója van, ezt állandóan felosztott területnek nevezzük. Erről a 3.2.6 pontban részletesen beszélünk. A főtár bármely byte-jához ugyanannyi idő alatt lehet hozzáférni; ebből a szempontból a tár "eleje" és "vége" között nincs különbség. Müködése mágneses elvü; ferritgyürükből áll.

A lokális tár 256 byte terjedelmü, és egy része a programból is cimezhető, más részét a vezérlő egység használja munkaterületként. Innen dolgoznak a szelektorcsatornák is. A programból cimezhető rész 16 darab általános célu és 4 darab lebegőpontos regiszterből áll. Az általános célu regiszterek szolgálnak fixpontos bináris, logikai operandusok, cimek, indexértékek tárolására. A lebegő-

pontos regiszterek a lebegőpontos operandusokat és eredményeket tárolják. Az általános regiszterek cimzése 4 bites számokkal történik: 0000-tól 1111-ig. Ennek megfelelően vannak sorszámozva a regiszterek is: 0-tól 15-ig. A regiszter használatra bizonyos megkötések vonatkoznak, ezeket a 3.4.7 pontban ismertetjük. Minden általános célu regiszter hossza 4 byte. A lebegőpontos regiszterek lehetséges cimei: 0, 2, 4, 6. Ettől eltérő cimmel lebegőpontos regiszterre hivatkozni nem lehet, a vezérlő egység ezt programhibának tekinti. A lebegőpontos regiszterek 8 byte hosszuságuak, és rövid lebegőpontos operandusok tárolása esetén az operandus a regiszter "bal felében" foglal helyet, a 32-63. bitek nincsenek kihasználva. Olyan regisztercimek esetén, amelyek egyaránt jelenthetnek általános célu és lebegőpontos regisztereket, az illető utasitás müveleti kódja dönti el, hogy az operandust honnan kell venni.

A lokális tárban található még a szelektorcsatornák átvitelt vezérlő információja. Igy a lokális tár egy része alcsatornának is tekinthető.

A multiplex tár nem egyéb, mint a multiplex csatorna összes alcsatornája egybegyüjtve. Egy-egy alcsatorna 16 byte hosszuságu, ezzel a multiplex tár nagysága - a gép kiépitettségétől függően - 768 vagy 1792 byte lehet. A multiplex tár programból nem cimezhető, de a kezelőpultról, ahonnan a mikroprogramok akár mikroutasitásonként is lefuttathatók, be lehet irni a multiplex tárba is.

A kulcstár feladata, hogy a memóriavédelmi kulcsokat tárolja. Programból közvetlenül szintén nem cimezhető, de a memóriavédelem gépi utasitásaival ki lehet olvasni a kulcsok értékét, illetve be lehet állitani tetszőleges kulcsértékeket. A kulcstár elektronikus elemekből épül fel, mert itt igen lényeges a nagy müveleti sebesség, a gyors müködés.

Végül megjegyezzük, hogy nem feltétlenül ebben a formában, de valahogyan minden számitógéptipusnál megtalálhatók a fenti tárterületek. Lényeges, hogy a főtár és a kiegészitő tár fogalmát tisztán lássuk, a konkrét megoldások már igen sokfélék lehetnek.

3.2.6 A FŐTÁR ÁLLANDÓAN FELOSZTOTT TERÜLETE

A 64. ábrán felrajzoltuk a főtár első 128 byte-jának felosztását.

Állandóan felosztott területre azért van szükség, hogy bizonyos kötött formátumu információk helyét egyértelmüvé tegyük, ezek cime ne függjön semmilyen programtól. Hangsulyozzuk, hogy az állandó felosztást hardware szempontból kell érteni, azaz a központi számitógépet eleve ugy tervezték, hogy az itt található információkat mindig ugyanitt keresi, és mindig ugyanide helyezi el. Ezért a főtár elejét nem szabad tetszőlegesen felhasználni.

Az első három duplaszó a kezdeti programbetöltés vezérlését végzi. Ennek a területnek a számitógép bekapcsolásakor van szerepe, illetve olyan hibák esetén, amikor az operációs rendszert ujból be kell tölteni a tárba.

A következő öt duplaszó a programmegszakitásoknál elmentett információt tárolja. Ezek alapján lehet visszatérni a megszakitott programba, ha az folytatható.

A 64-75. cimek tartalmát a csatornarendszer használja az input/output müveletek lebonyolitásánál.

Az óraszót időmérésre használhatjuk. Ez tulajdonképpen egy bináris számláló, amelynek értékét bizonyos időközönként aktualizáljuk. Igy programok futási ideje stb. mérhető.

Tárcím	A tárterület tartalma
0	A kezdeti programbetöltés programállapotszava
8	A kezdeti programbetöltés 1. csatornaparancsszava
16	A kezdeti programbetöltés 2. csatornaparancsszava
24	Külső megszakítások régi programállapotszava
32	SVC eredetű megszakítások régi programállapotszava
40	Programhiba eredetű megszakítások régi programállapotszava
48	Géphiba eredetű megszakítások régi programállapotszava
56	Input/output megszakítások régi programállapotszava
64	Csatornaállapotszó
72	Csatornacímszó
80	Óraszó
88	Külső megszakítások új programállapotszava
96	SVC eredetű megszakítások új programállapotszava
104	Programhiba eredetű megszakítások új programállapotszava
112	Géphiba eredetű megszakítások új programállapotszava
120	Input/output megszakítások új programállapotszava
128 ⋮	Diagnosztikai terület

64. ábra.
Az állandóan felosztott főtárterület információtartalma

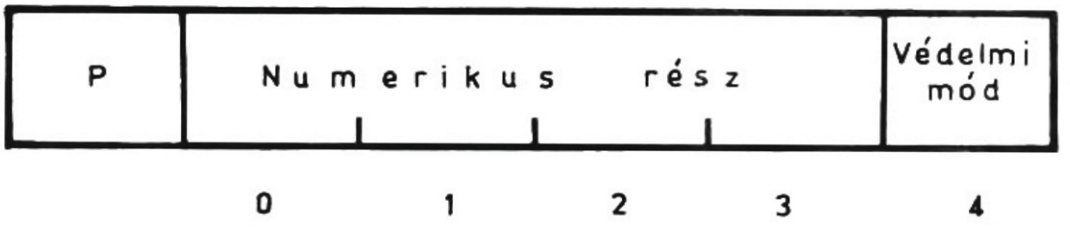
A következő öt duplaszó a programmegszakitások kezelésére használatos. Itt található mindaz a vezérlő információ, amely a megszakitáskezelő rutinok futásához kell. A vezérlési információ szerkezetére, tartalmára a 3.4.8 és 3.5.4 pontban részletesen kitérünk.

A 128-255. byte-ok tartalma változó lehet. Ez a mező géphiba eredetü kódokat tartalmazhat. A kódok kiértékelése révén következtetni lehet a géphiba fajtájára, okára. A 256. byte-tól kezdődő főtárterület már szabadon használható, erre semmilyen hardware megkötés nincs.

3.2.7 A TÁRVÉDELEM MECHANIZMUSA. TÁRVÉDELMI KULCSOK

A mai számitógépeken általában nem egyetlen program fut egy adott időben, hanem több. Ezt nevezzük multiprogramozásnak, illetve multiprogramozott üzemmódnak. Több program egyidejü futtatása ugy oldható meg, hogy az egyes programok között elosztjuk a számitógép erőforrásait. Például, az egyik program a központi számitógépet használja, a másik a perifériakészlet bizonyos elemeit stb. Természetesen, két program egy és ugyanazon pillanatban nem használhatja például a vezérlő egységet, de az már elképzelhető, hogy a vezérlő egység és a csatornarendszer időben egyszerre dolgozzon. Ekkor valamennyi program használhatja az operativ tárat is: onnan adatokat olvashat ki, illetve oda adatokat irhat be. Fontos, hogy az egyes programok ne zavarhassák egymást, azaz ne irhassák felül a tárban egymás adatait, valamint szükség esetén ne férhessenek hozzá egymás adataihoz. Ennek biztositására szolgál a tárvédelmi mechanizmus, amely csakis a főtároló védelmét oldja meg. A kiegészitő tárak védelmére azért nincs szükség, mert minden, ami a kiegészitő tárakban található, a főtárból került oda! Elegendő tehát ezt védenünk.

A főtár információvédelme kettős: irás elleni védelem és irás-olvasás elleni védelem. Csak olvasás elleni védelem nincs. A tárvédelem eszközei a tárvédelmi kulcsok. Ezek 5 bites bináris számok, felépitésüket a 65. ábra mutatja.



65. ábra. Tárvédelmi kulcsok felépitése

A négy numerikus biten összesen 16 különböző érték állitható be. Ebben a rendszerben tehát legfeljebb 16 program védhető egymással szemben. A védelmi mód bitje azt határozza meg, hogy irás vagy irás-olvasás elleni védelemről van szó. P a kulcs paritásbitje. A tárvédelmi kulcsok a kulcstárban vannak. Ezt ugy képzelhetjük el, hogy a kulcstár az operativ tárnak egy olyan speciális része, amely csak kulcsokat tartalmaz. A kulcsok kiolvasására, illetve beállitására 1-1 gépi utasitás szolgál.

A tárvédelmi mechanizmus az alábbiak szerint müködik: minden egyes 2 Kbyte-os tárterülethez tartozik a kulcstárban egy rekesz. Ez a hozzárendelés rögzitett. Minden egyes tárhivatkozásnál

- mielőtt a főtárbeli adat elérése megtörténne - a vezérlő egység kikeresi az adott tárterülethez tartozó kulcsot a kulcstárból, és összehasonlitja annak numerikus részét az illető program vagy csatornaprogram tárvédelmi kulcsával. A programok tárvédelmi kulcsa a programállapotszóban, a csatornaprogramok tárvédelmi kulcsa pedig a csatornacimszóban található. Ha a kulcsok numerikus része egyezik, akkor a program "jogosult" a megcimzett tárterület kiolvasására vagy tartalmának felülirására. Ha a numerikus részek nem egyeznek, akkor a védelmi mód bitje dönt arról, hogy a vezérlő egység mit tegyen. Ezt a 8. táblázatban foglaltuk össze.

8. táblázat

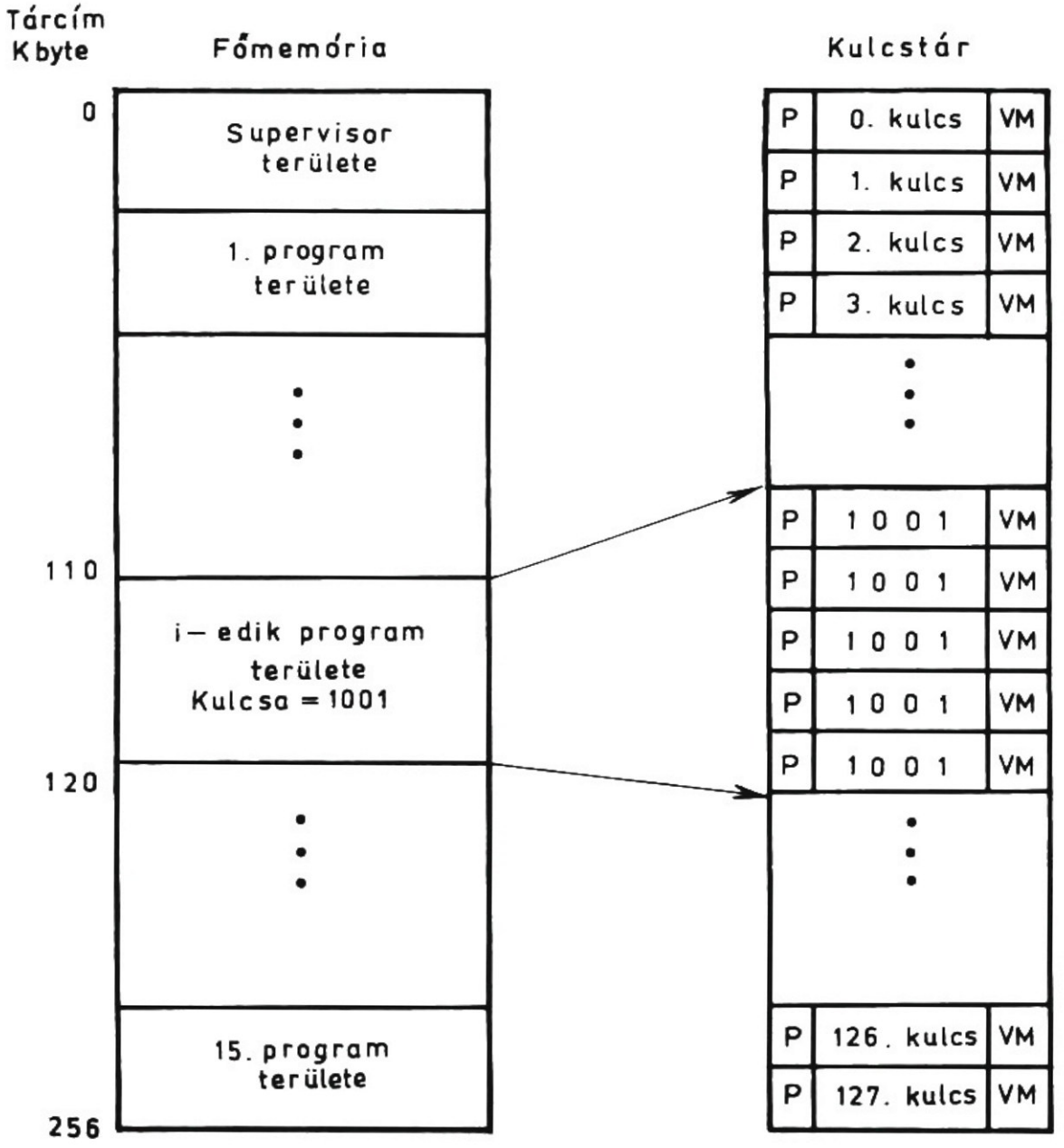
A tárvédelmi mechanizmus müködése

Védelmi mód bit	Numerikus részek egyeznek	Numerikus részek nem egyeznek
0	irás-olvasás engedélyezett	olvasás engedélyezett, irás letiltva
1	irás-olvasás engedélyezett	irás-olvasás letiltva

Látjuk, hogy a numerikus részek egyezése esetén a védelmi mód bit értéke közömbös: a program jogosult a tárhivatkozásra. Idegen program - amelyik nem ismeri a védelmi kulcs értékét - tárhozáfordulásánál a védelmi mód bitje meghatározó jelentőségü. Ezzel egy program akár csak irás, akár irás és olvasás ellen is védheti magát. A védelmi mód bit beállitását a kulcs beállitásánál kell eldönteni. Titkos adatok feldolgozásánál például kivánatos lehet az olvasást is letiltani: idegen programok ekkor sehogyan sem juthatnak hozzá a védett területhez.

A számitógépek operativ tára általában nagyobb annál, hogy minden egyes 2 Kbyte-os blokkhoz eltérő értékü kulcsot rendelhessünk hozzá. Erre azonban nincs is szükség, hiszen egy-egy program területe általában jóval nagyobb 2 Kbyte-nál. Emiatt azt tesszük, hogy a programot lefedő főtárterület valamennyi blokkjához ugyanazt a kulcs értéket rendeljük, és azt mondjuk, hogy ez a tárterület az adott programé. Ez a program e területen bármelyik tárcimhez hozzáférhet, hiszen a saját kulcsát használja. Egy lehetséges főtárfelosztást mutat a 66. ábra.

Itt az i-edik program a főtárban a 110.-120. Kbyte-os területet foglalja el, összesen 10 Kbyte-ot. Kulcsa 1001, ez az érték áll a kulcstárnak azokban a rekeszeiben, amelyek a 110.-120. Kbyte-os főtár - területhez tartoznak. Látjuk, hogy minden rekeszben 1001 érték áll, ez a terület tehát teljes egészében hozzáférhető az i-edik program számára. Megjegyezzük, hogy a 0000 értékü kulccsal bárhonnan lehet olvasni, de irni nem.



66. ábra. A tárvédelmi kulcsok használata

3.3 A PROCESSZOR. AZ ARITMETIKAI-LOGIKAI EGYSÉG

3.3.1 AZ ARITMETIKAI MŰVELETEK VISSZAVEZETÉSE ÖSSZEADÁSRA

Mint tudjuk valamennyien, a legkézenfekvőbb, legközismertebb aritmetikai müveletek az összeadás és a kivonás. Lépten-nyomon találkozunk velük a mindennapi életben, munkahelyünkön, bevásárláskor stb. Azt viszont már sokkal kevesebben sejtik, hogy milyen szoros kapcsolatban áll a korszerü nagyteljesitményü elektronikus számitógépek bonyolult müködése ezekkel a látszólag egyszerü matematikai alapmüveletekkel. A dolog magyarázata az, hogy a digitális számitógépek az összetett müveleteket elemi összegzések sorozatára bontják le, valóban igaz az, mint erről a 2. fejezet tanulmányozásánál

meggyőződtünk, hogy "összeadással és negálással minden megoldható". Ennek igen fontos gyakorlati következménye az, hogy a számítógépek műveletvégző egysége tulajdonképpen egy vagy több elemi összeadóműből áll. Természetesen géptipustól függően igen sokféle konkrét megoldással találkozhatunk, a működési elv azonban végső soron mindenütt ugyanaz.

3.3.2 A FÉLÖSSZEADÓ FELÉPÍTÉSE ÉS MŰKÖDÉSE

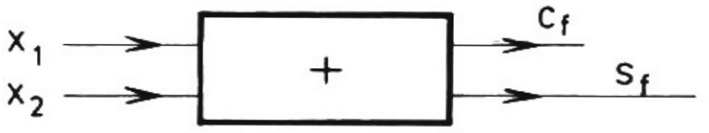
A 67. ábrán lerajzoltuk a félösszeadó leegyszerüsitett vázlatát.

Ez az elemi összeadó mindössze arra képes, hogy két bináris szám - X1 és X2 - egy-egy bitjét összeadja, és a keletkezett részösszeg (S_f) az átvitellel együtt (c_f) a kimenetén megjelenjen. Nevét onnan kapta, hogy egyetlen bit összeadását is csak "félig" végzi el, mert nem veszi figyelembe az előző helyértékről származó átvitelt. Az igazságtáblázatból kiolvashatjuk, hogy

$S_f = X1 \oplus X2$, és $(\bar{x}_1 x_2 + x_1 \bar{x}_2)$

$c_f = X1 \cdot X2$,

vagyis egyszerü logikai függvényekkel realizálható az összeadás. Két vagy több félösszeadóból már felépithetjük azokat a bonyolultabb müveletvégzőket, amelyek az átvitelkezelést is elvégzik.

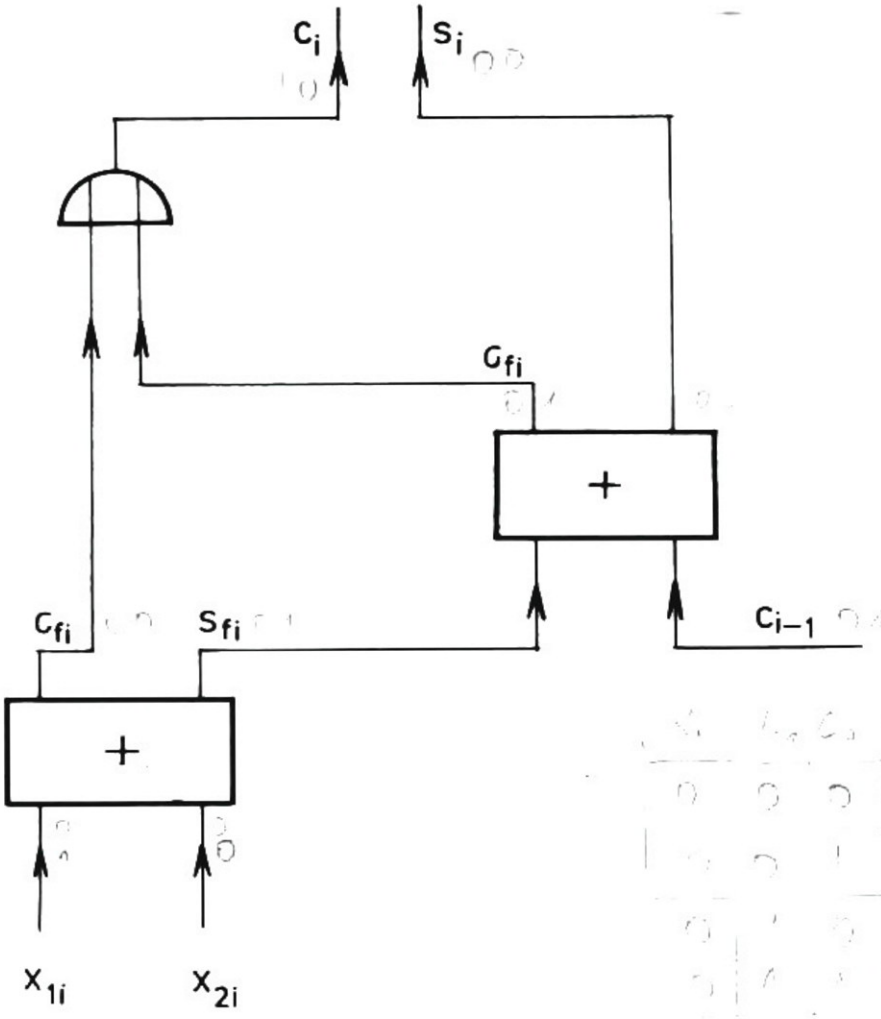


X_1	X_2	S_f	C_f
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

67. ábra. Félösszeadó vázlata és igazságtáblázata

3.3.3 A TELJES ÖSSZEADÓ FELÉPÍTÉSE ÉS MŰKÖDÉSE

Két félösszeadó megfelelő összekapcsolásával könnyen épithetünk olyan 1 bites aritmetikai müveletvégzőt, amely minden körülmények között helyes részösszeget szolgáltat, és az előző helyértékről származó átvitelt is figyelembe veszi. A teljes összeadó felépitését a 68. ábra mutatja.

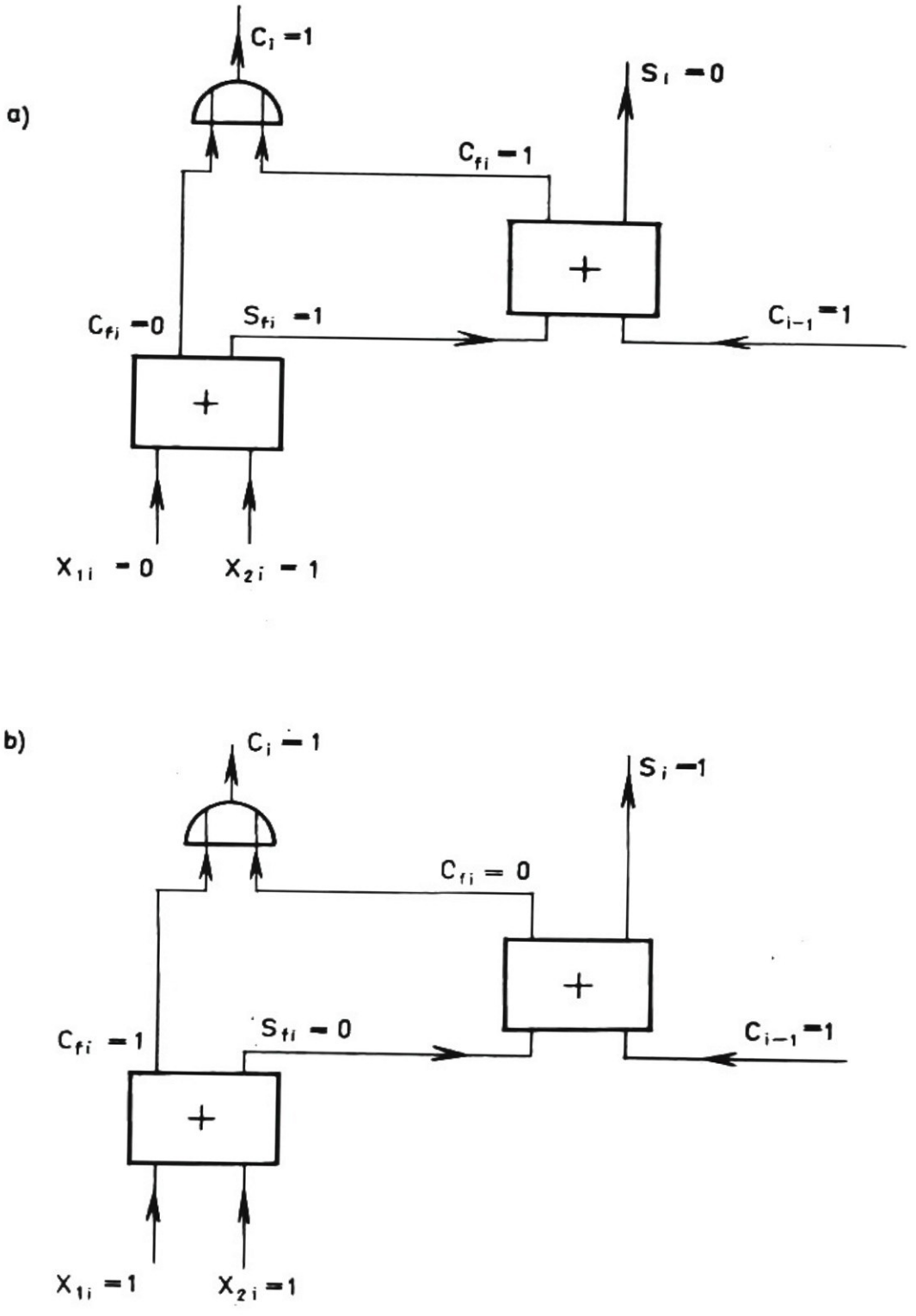


68. ábra. 1 bites teljes összeadó vázlata

Az egyik félösszeadó összeadja az X1 és X2 operandus i-edik bitjét, a másik félösszeadó pedig az előző (i-1-edik) helyértékről származó átvitellel (c_{i-1}) korrigálja a részösszeget. A teljes összeadó kimenetén megjelenő S_i összeg és c_i átvitel $X1_i$ és $X2_i$ minden lehetséges kombinációja mellett helyes eredményt jelent; az S_i részösszeg lesz a végeredmény i-edik bitje, a c_i átvitelt pedig az $X1_{i+1}$, $X2_{i+1}$ bitek összeadásánál vesszük figyelembe. Ebből a gondolatmenetből már kitünik, hogy a szokásos hosszuságu operandusok (1 byte, 1 félszó, 1 gépi szó) összeadása egyetlen problémát vet fel csupán: valamilyen módon mindkét operandus minden egyes bitpárját rendre össze kell adnunk. Ennek gyakorlati megvalósitására számos eljárás terjedt el. Mielőtt ezek közül a fontosabbakat megvizsgálnánk, tekintsünk néhány példát a teljes összeadó müködésére.

3.3.4 PÉLDÁK AZ 1 BITES TELJES ÖSSZEADÓ MŰKÖDÉSÉRE

Kövessük nyomon most egy teljes összeadó müködését különböző bemeneti bitkombinációkra! Ehhez segitséget nyujt a 69. ábra, amelyen az X1 és X2 operandus i-edik bitjének összeadását mutatjuk be. Mindkét esetben feltételeztük, hogy az előző (i-1-edik) helyértékről átvitelt "örököltünk".



69. ábra. Példák az 1 bites teljes összeadó müködésére

Az a) példában $X1_i = 0$ és $X2_i = 1$, ennek az összeadásnak az eredménye $S_{fi} = 1$, és az első félösszeadó átvitelt nem szolgáltat ($c_{fi} = 0$). Ezután a második félösszeadó bemenetére kerül az S_{fi} részösszeg és a feldolgozandó átvitel: c_{i-1}. Mivel ezek értéke 1, a második félösszeadó már a helyes részösszeget (S_i) szolgáltatja, az átvitelképzőn pedig megjelenik a $c_i = 1$ érték.

A b) példa ettől mindössze annyiban különbözik, hogy itt $X1_i = X2_i = 1$, és a keletkezett c_i átvitel az első félösszeadóról jut az átvitelképző bemenetére.

Könnyen ellenőrizhetjük, hogy mindkét elrendezés a félösszeadó már megismert igazságtáblázatát valósitja meg. Ez nem véletlen: a gyakorlatban is ilyen jellegü logikai táblázatok segitségével tervezik a bonyolult számitógép-rendszereket. Annak ellenőrzését, hogy a teljes összeadó az összes többi $X1_i$, $X2_i$, c_{i-1} bitkombinációra is helyesen müködik, az olvasóra bizzuk.

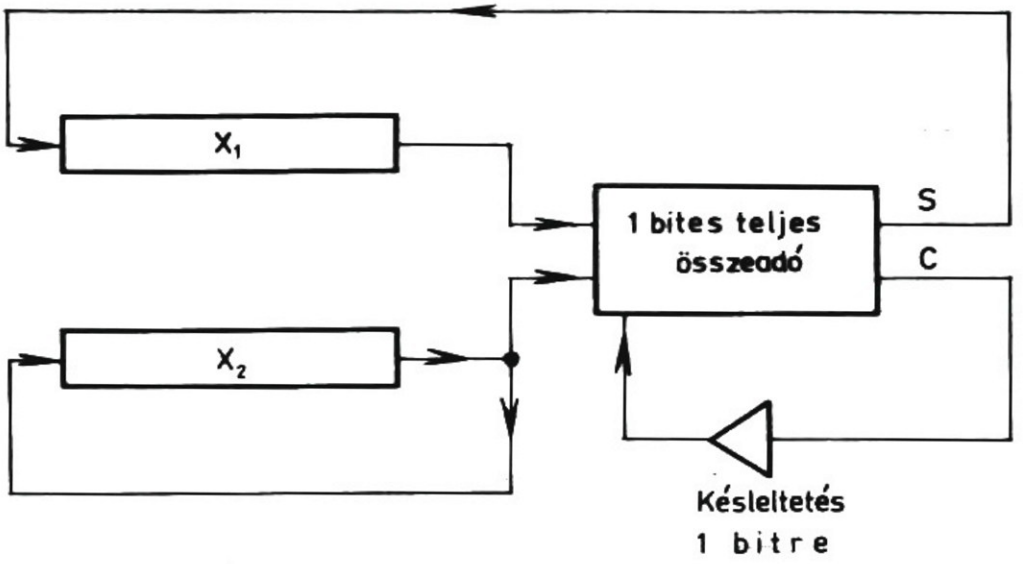
3.3.5 SOROS, PÁRHUZAMOS ÉS "VEGYES" ARITMETIKAI MŰVELETVÉGZŐK

Az előzőekben megismert müködési elvek alapján számos konkrét müveletvégzőt fejlesztettek ki a számitógépgyártó cégek. Az egyes összeadómüvek - különböző tulajdonságaik alapján - jelentősen eltérhetnek egymástól, azonos müködési elv mellett is. A különbségek főleg a müveletvégzés sebességében és az operandusok szélességében mutatkoznak. Ha meggondoljuk, hogy a számitógépnek másodpercenként több ezer müveletet kell végeznie, nem közömbös az aritmetikai egység sebessége. Az operandusszélesség közvetve szintén a gyorsaságra hat ki: a "keskeny" (pl. 1 byte-os) aritmetikák a szélesebb (pl. 1 gépi szó méretü) operandusokat csak részletekben tudják feldolgozni, igy a teljes müvelet több lépés időtartamára lefoglalja az aritmetikát. Megállapithatjuk tehát, hogy a legelőnyösebb felépitésü az az aritmetikai müveletvégző, amely viszonylag széles operandusok összeadását rövid idő alatt elvégzi. Ez azonban jelentősen növelheti az előállitási költséget, ezért kénytelenek vagyunk középutat keresni az ár és a müveleti sebesség között. Ennek alapján háromféle müveletvégző-konstrukció terjedt el a gyakorlatban: a soros, párhuzamos és "vegyes" aritmetika.

Soros aritmetikai müveletvégző

Soros müködésünek nevezzük azt az összeadómüvet, amely az operandusokat a legalacsonyabb helyértéktől kezdve bitenként egymás után adja össze. Más szavakkal: az első lépésben összeadja a két operandus 0-dik bitpárját, ezután az 1. bitpárját, vagyis az összeadásnál "jobbról balra" - növekvő bináris helyértékek irányába - halad. Világos, hogy egy összeadási müveletet annyi lépésben végez el, ahány bitesek az operandusok. Ha a két összeadandó szám hossza nem egyezik meg, akkor a lépésszámot a hosszabb operandus határozza meg, ui. a gép ilyenkor automatikusan kiegésziti a rövidebb operandust értéktelen un. vezető nullákkal. Eltérő hosszuságu operandusok elsősorban tömör formáju decimális számok összeadásakor fordulnak elő. A 70. ábrán felrajzoltuk a soros müködésü összeadó leegyszerüsitett vázlatát.

Az elrendezés legfontosabb eleme az 1 bites teljes összeadó. Ennek felépitését már ismerjük, müködését két szemléletes példán keresztül mutattuk be. Figyelmet érdemel az a körülmény, hogy a soros összeadóban csupán egyetlen teljes összeadó van. Az X1 és X2 operandust egy-egy önálló regiszterben tároljuk a müveletvégzés kezdetéig. Ez más szavakkal azt jelenti, hogy az összeadás megkezdése előtt ezt a két regisztert fel kell tölteni a megfelelő operandusokkal. A regisztereket ugy alakitják ki, hogy alkalmas vezérlő jelek segitségével tartalmukat egy bittel jobbra lehet léptetni. Az összeadás az alábbi módon megy végbe: az első léptető jel hatására X1 és X2 legalacsonyabb helyértékü (jobbszélső) bitje belép az 1 bites teljes összeadóba, amely S kimenetén a helyes részösszeget, c kimenetén pedig az esetleges átvitelt szolgáltatja. Figyeljük meg, hogy az S részösszeget közvetlenül visszavezetjük X1 helyére; az első lépés után tehát a felső regiszter első bitje már nem az X1 operandushoz, hanem a



70. ábra. Soros összeadó müködési vázlata

végeredményhez tartozik. Az alsó regiszterbe a második operandust vezetjük vissza bitenként. Az 1 bites teljes összeadóhoz csatolt késleltető addig tárolja a c átvitelt, mig esedékessé nem válik a következő bitpár összeadása. Ekkor a teljes összeadó a már megismert módon hozzáadja az átvitelt az aktuális "félig kész" részösszeghez.

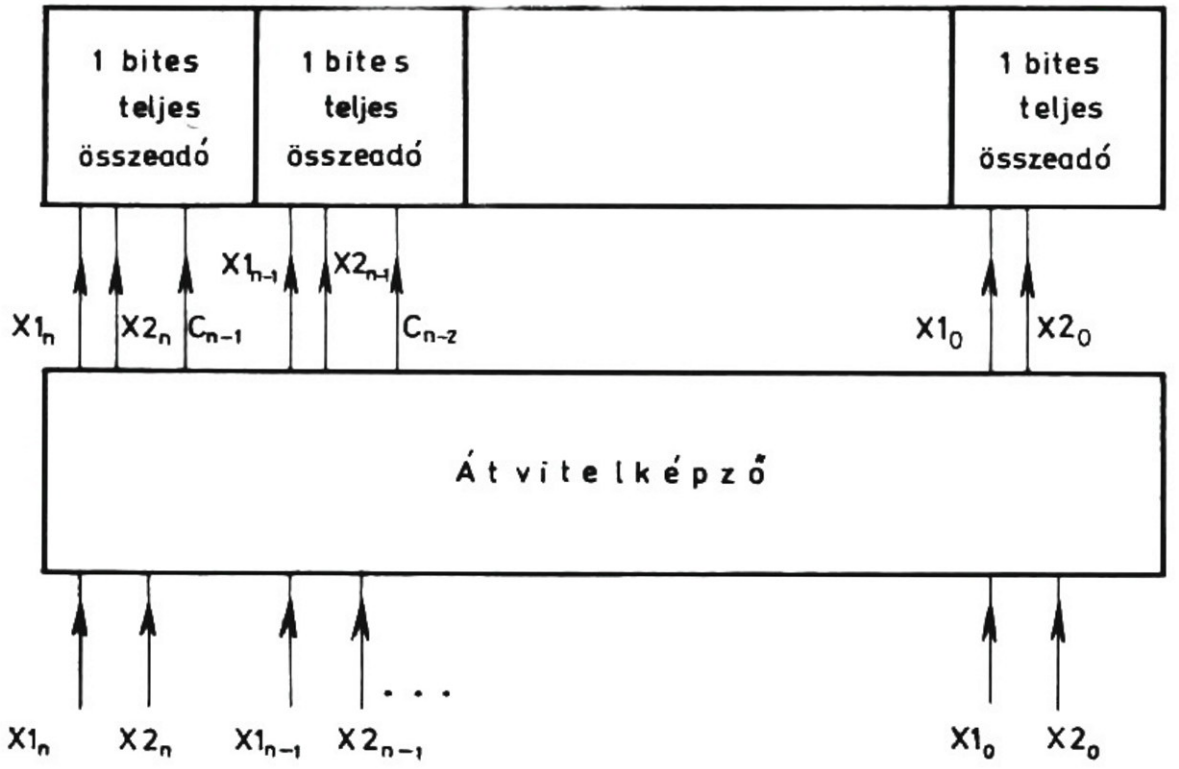
Az összeadó müködését tanulmányozva fontos következtetést vonhatunk le, hogy milyen információ vész el, illetve őrződik meg az összeadás során. Mivel X1 operandusunkat az eredmény felülirja, az összeadás elvégzése után X1 itt többé nem áll rendelkezésünkre. Az X2 operandust azonban lépésenként visszavezettük saját korábbi helyére, ez tehát ismét felhasználható. Tulajdonképpen X1 sem veszett el véglegesen, hiszen megtalálható az operativ tárban vagy annak valamelyik regiszterében. Ahhoz azonban, hogy X1-et ismét valamilyen aritmetikai müveletben szerepeltessük, ujból be kell töltenünk az aritmetikai-logikai egységbe.

Összefoglalva a soros összeadó ismérveit, mondhatjuk, hogy ez a legegyszerübb felépitésü összeadómü, hiszen mindössze egy db 1 bites teljes összeadóból áll. Hátránya viszont, hogy hosszabb operandusok összeadását - a soros müködés miatt - viszonylag lassan tudja csak elvégezni.

Párhuzamos aritmetikai müveletvégző

A párhuzamos aritmetikai müveletvégző (párhuzamos összeadó) müködését a 71. ábrán mutatjuk be.

Mint látjuk, itt annyi 1 bites teljes összeadóra van szükség, amilyen hosszuságu operandusokat össze akarunk adni. Az összeadást ez az aritmetika egyetlen lépésben végzi el. Először az X1 és X2 operandust teljes egészében betöltjük az összeadómübe (ez azt jelenti, hogy az X1 operandus 0-dik, első stb. bitje rendre megjelenik az $X1_0$, $X1_1$ stb. bemeneteken, és ugyanez megtörténik X2-re nézve is). Ezután a párhuzamos összeadóban levő átvitelképző X1 és X2 valamennyi bitjét figyelembe véve megállapitja az összes átvitelt. Most már dolgozhatnak a teljes összeadók, és a kimenetükön valamennyi helyértéken egyszerre jelenik meg az X1 + X2 összeg.



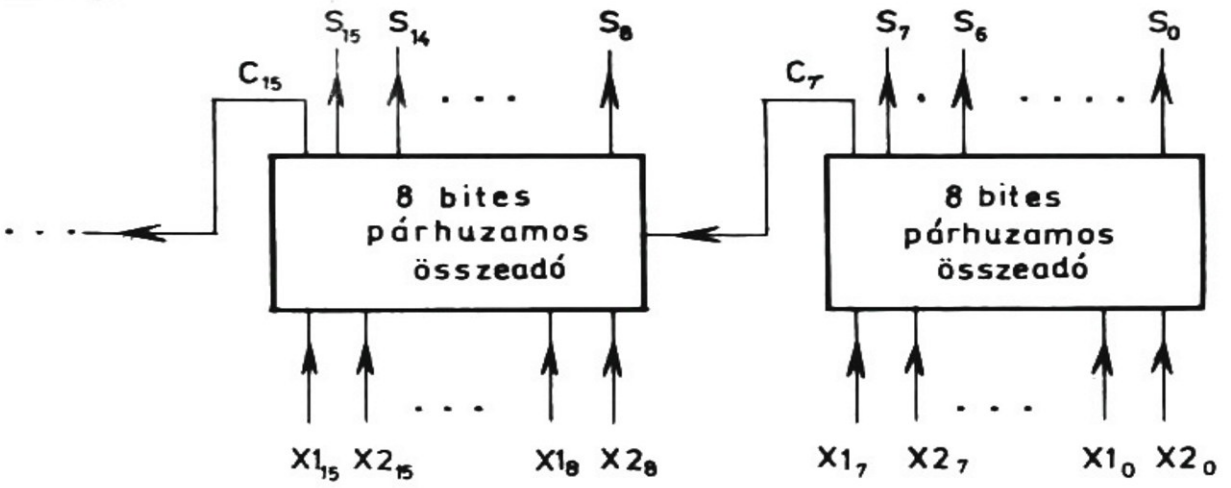
71. ábra. Párhuzamos összeadó működési vázlata

A soros működésű összeadótól ez a berendezés lényegesen eltér, ugyanis egy lépésben valósitja meg ugyanazt, amire a soros összeadónak több lépésre volt szüksége. Ebből következik, hogy a párhuzamos összeadó jóval gyorsabb a sorosnál, bár jóval bonyolultabb is nála. Különösen az átvitelképző megvalósitása nehéz, és a nagyobb méretű párhuzamos aritmetika előállitási árát jelentősen emeli. Emiatt főleg csak drágább, nagy számitógépekben alkalmazzák; ott, ahol a gyors müveletvégzés elsőrendű követelmény.

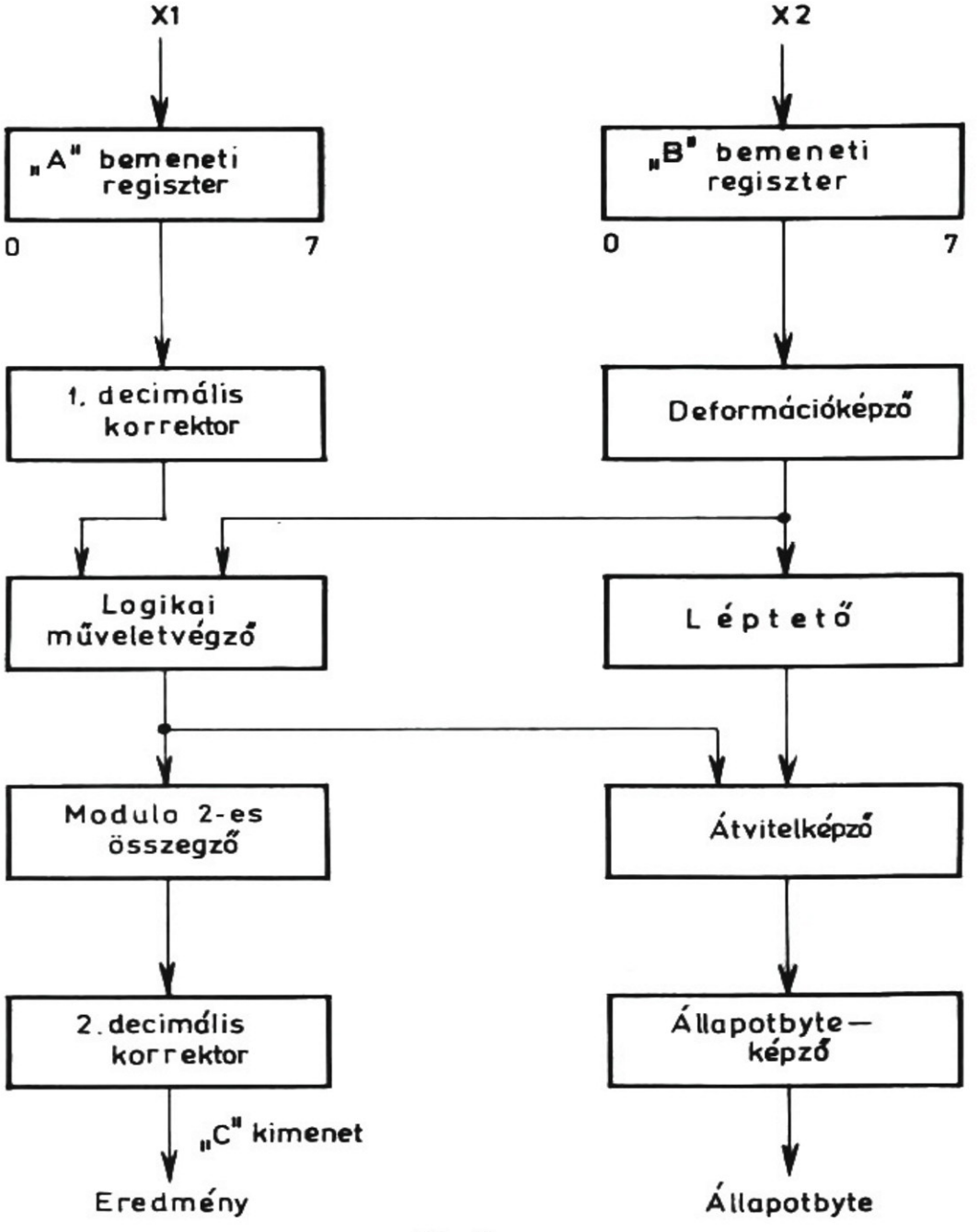
"Vegyes" aritmetikai müveletvégző

A soros és párhuzamos összeadó előnyös tulajdonságait egyesiti az ugynevezett "vegyes" összeadó. Nevét onnan kapta, hogy az operandusokat néhány bites csoportokra tördelve kezeli, az egyes bitcsoportokon belül azonban párhuzamosan dolgozik.

Az ábrán egy olyan vegyes összeadó részletét rajzoltuk fel, amely 8 bites egységekre tördelve dolgozza fel az operandusokat. Annyi db 8 bites párhuzamos összeadóból épül fel, ahányszor 8 bitesek az operandusok. Ezután az első párhuzamos összeadó kiszámitja az $S_0S_1 \ldots S_7$ összeget, és továbbadja a c_7 átvitelt. A második párhuzamos összeadó (c_7 figyelembe vételével) kiszámitja az $S_8S_9 \ldots S_{15}$ összeget stb. Előnye ennek az elrendezésnek, hogy működése nem teljesen soros, és a kisebb méretű 8 bites párhuzamos összeadó átvitelképzője jóval egyszerübb felépitésü, mint a nagyméretü párhuzamos összeadóé. Elképzelhető olyan eset is, hogy egy számitógépbe csak 1 db 8 bites párhuzamos összeadót épitenek be, és ez végzi el az operandusok összeadását 8 bitenként. Ilyenkor az operandusokat 8 bites egységekre tördelve több lépésben "keresztülvezetjük" az összeadón, amely elvégzi a c_7, c_{15} stb. átvitelek ideiglenes tárolását is.



72. ábra. Vegyes összeadó működési vázlata



73. ábra.
Az R20 számítógép aritmetikai-logikai egységének egyszerüsitett vázlata

3.3.6 PÉLDA KONKRÉT ARITMETIKAI-LOGIKAI EGYSÉGRE

A 73. ábrán felrajzoltuk az R20 számitógép aritmetikai-logikai egységének egyszerüsitett vázlatát. A berendezés legfontosabb része az aritmetikai-logikai müveletvégző, amely 8 bites operandusokat tud kezelni. Az összeadáson kivül kivonást és több logikai müveletet végezhet a két operandus között. Az X1 és X2 8 bites operandus az "A" és "B" bemeneti regiszterből kerül a müveletvégzőre. A decimális korrekciós blokkok akkor lépnek müködésbe, ha BCD összeadás esetén érvénytelen részösszeg képződött. A modulo 2-es összegző az eredmény helyességét ellenőrzi. Végül a keletkezett átvitelt és az eredményre vonatkozó egyéb információt (negativ, pozitiv eredmény stb.) az átvitelképző és az állapotbyte-képző tárolja.

A berendezés párhuzamosan dolgozik: 8 bites operandusokra nézve, tehát például egy gépi szó feldolgozása 4 lépést igényel. A 8 bites részeredmények a "C" kimeneten jelennek meg. Ezekből rakható össze a teljes végeredmény.

3.4 A VEZÉRLŐ EGYSÉG

3.4.1 A MIKROPROGRAMOZOTT ÉS HARDWARE VEZÉRLÉSI MÓD

Mint a központi számitógép vázlatos ismertetésénél láttuk, a vezérlő egység feladata a teljes számitógéprendszer müködésének központi irányitása, az egységek müködtetése. Ilyen értelemben a vezérlő egység fölérendelt helyzetben van, más szavakkal: a számitógéprendszer elemei - az aritmetikai-logikai egység, a csatornarendszer, a perifériák - "kötelesek" azt cselekedni, amit számukra a vezérlő egység meghatároz. Ez a fölérendeltség nem véletlenszerü: a számitógéprendszer tervezésekor már eldől. Ennek fontosabb következményeit a programozás ismertetésénél megvilágitjuk.

Azért, hogy a vezérlő egység hatékonyan irányithassa az egész számitógéprendszert, fel kell ruháznunk megfelelő vezérlési sajátosságokkal. Kétféle vezérlési mód terjedt el napjainkban: a mikroprogramozott és a hardware (huzalozott) vezérlés. Mindkettő célja közös: olyan belső - a felhasználó számára nem észlelhető - vezérlő jelek kialakitása, amelyek - a rendszer többi eleméhez eljutva - a kivánt funkciót váltják ki. Például tekintsük azt az esetet, amikor egy regiszter tartalmát át kell irni egy másik regiszterbe. Ehhez az átirás tartamára kapcsolatot kell létesiteni a két regiszter között, elvégezni az információátvitelt, majd a kapcsolatot megszüntetni.

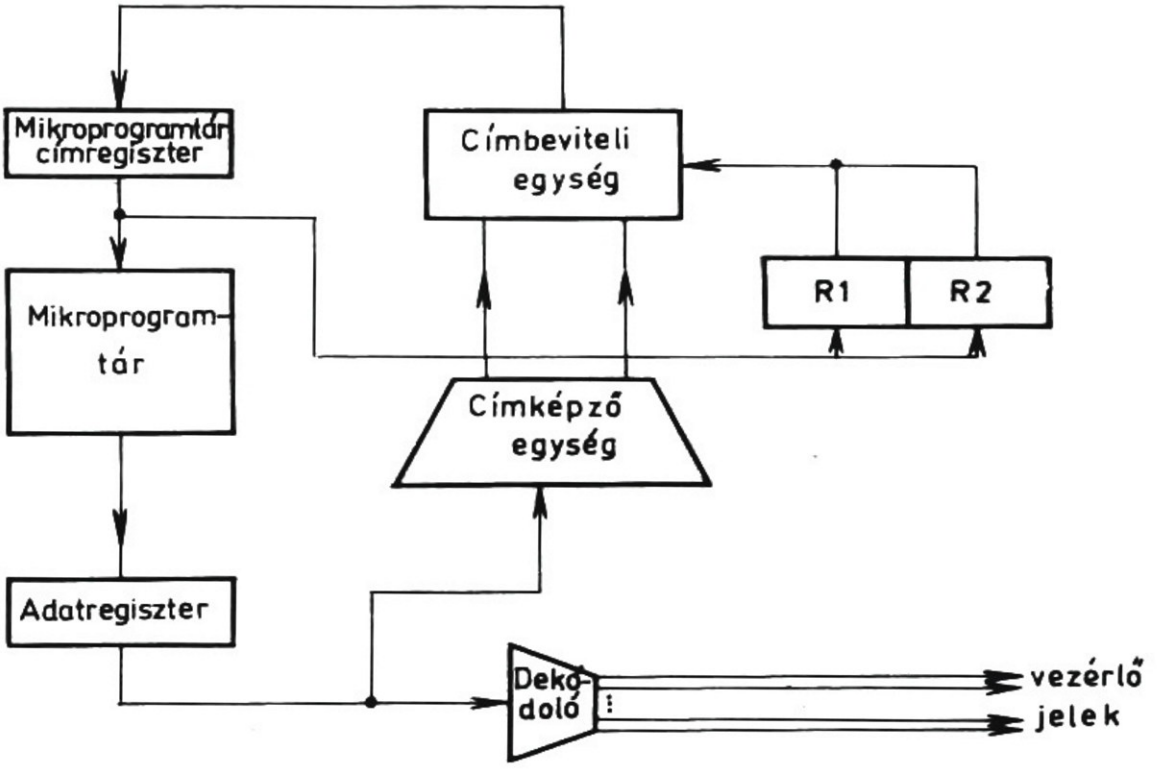
Egy másik példa: két szám összeadásához az operandusokat be kell tölteni az aritmetikai-logikai egység bemeneti regisztereibe, az összeadást el kell végeztetni az aritmetikai egységgel, majd az eredményt a kivánt helyre (például a főmemória egy megadott cimére) kell vinni stb. Látjuk, hogy viszonylag egyszerü müveletek is sok elemi lépésre bomlanak, ha végrehajtásukat közelebbről szemléljük. Minden egyes ilyen elemi lépés egy-egy vezérlő jel kialakulásával egyenértékü. Ebből jól érzékelhetjük, hogy a vezérlő egységnek igen sok helyre, igen nagy sebességgel, igen sokféle vezérlő jelet kell kiküldenie ahhoz, hogy a rendszer gyorsan müködjön.

A mikroprogramozott és a hardware vezérlési mód közötti fő különbség éppen e vezérlő jelek kialakitásának módjában áll: mig mikroprogramozott vezérlés esetén a belső vezérlő jeleket egy speciális

program, az ugynevezett mikroprogram futása szolgáltatja, addig a hardware vezérlésnél ezek a jelek pusztán az összehuzalozott áramkörök révén keletkeznek, természetesen a megfelelő időbeli sorrendiség betartásával. A két módszer összevetéséből érezhető, hogy a hardware vezérlés gyorsabb, mint a mikroprogramozott. A nagyobb gépeknél azonban komoly nehézségeket okoz a tisztán hardware vezérlésü központi számitógép tervezése és kivitelezése, igy ezt csak a drágább, kifejezetten nagyteljesitményü modelleken alkalmazzák. Vegyes vezérlésüek többnyire a középkategóriáju gépek; itt a sebességre érzékeny funkciókat hardware uton realizálják, a többi müvelet mikroprogramozott. A gyakorlatban azonban többnyire tisztán mikroprogramozott gépekkel találkozhatunk, és ez az oka annak is, hogy elsősorban ezek müködését vizsgáljuk meg közelebbről.

3.4.2 BEVEZETÉS A MIKROPROGRAMOZÁS ELMÉLETÉBE

A mikroprogramozás lényegét a 74. ábra segitségével világitjuk meg.
Az ábrán látható mikroprogramtár lényegében az operativ tárhoz hasonló müködésü tároló. Feladata a mikroprogramok tárolása. Mivel azonban a mikroprogramok változtatására általában nincs szükség, a mikroprogramtár tartalmát a gyártó cég egyszer s mindenkorra meghatározza, a felhasználó azon nem változtathat. Megjegyezzük, hogy egyes géptipusoknál a mikroprogramtár részben vagy egészen cserélhető; ez bizonyos lehetőséget nyujt a felhasználónak a mikroprogramtár módositására. Ez a helyzet például az R40 számitógépnél: itt szükség esetén speciális teszt-mikroprogramtár használható bizonyos fajta hardware hibák felderitésére. A mikroprogramtár tartalmának állandóságát - a tárba



74. ábra. A mikroprogramozott vezérlés vázlata

való beirás lehetetlenségét - ugy juttatjuk kifejezésre, hogy a mikroprogramtárat "csak olvasható tárnak" nevezzük. A mikroprogramtár tartalma nem egyéb, mint a megengedett összes vezérlési funkció kialakitásához szükséges vezérlő jel kódolt formában. Ahhoz tehát, hogy a mikroprogramtár tartalmából vezérlő jelek keletkezhessenek, a vezérlő egységnek meg kell fejtenie, dekódolnia kell a mikroprogramtárban elhelyezett információt.

A mikroprogramtár - az operativ tárhoz hasonlóan - rekeszekből épül fel. Egy-egy ilyen rekeszben egy-egy mikroutasitás foglal helyet. A mikroutasitás a mikroprogram elemi része. Minden egyes mikroutasitás végrehajtása bizonyos számu vezérlő jel kialakitását eredményezi. A mikroutasitások bitekben kifejezett hossza modellenként változó, általában 64-130 bit közötti.

A mikroprogramtár minden egyes rekeszéhez egy cimet rendelünk hozzá; ez tulajdonképpen az illető rekesz sorszáma a tárban. Az egyes mikroutasitások cimük alapján kereshetők elő. Mindegyik mikroutasitásban vannak olyan mezők, néhány bites részek, amelyek részt vesznek a következő mikroutasitás cimének kialakitásában. Ily módon a mikroutasitások egymáshoz láncolódnak, és sorozatos végrehajtásuk - a mikroprogram futása - folyamatossá válik. Ha most szerzett ismereteink birtokában betekintenénk egy működő vezérlő egység belsejébe, azt tapasztalnánk, hogy minden időpillanatban valamelyik mikroutasitás végrehajtása folyik - vagyis fut a mikroprogram - és ez igy megy végestelen végig, amig csak a vezérlő egység működése szándékos leállitás vagy valamilyen sulyosabb hiba következtében megszakad.

A mikroprogram feldolgozása a következő lépésekből áll:

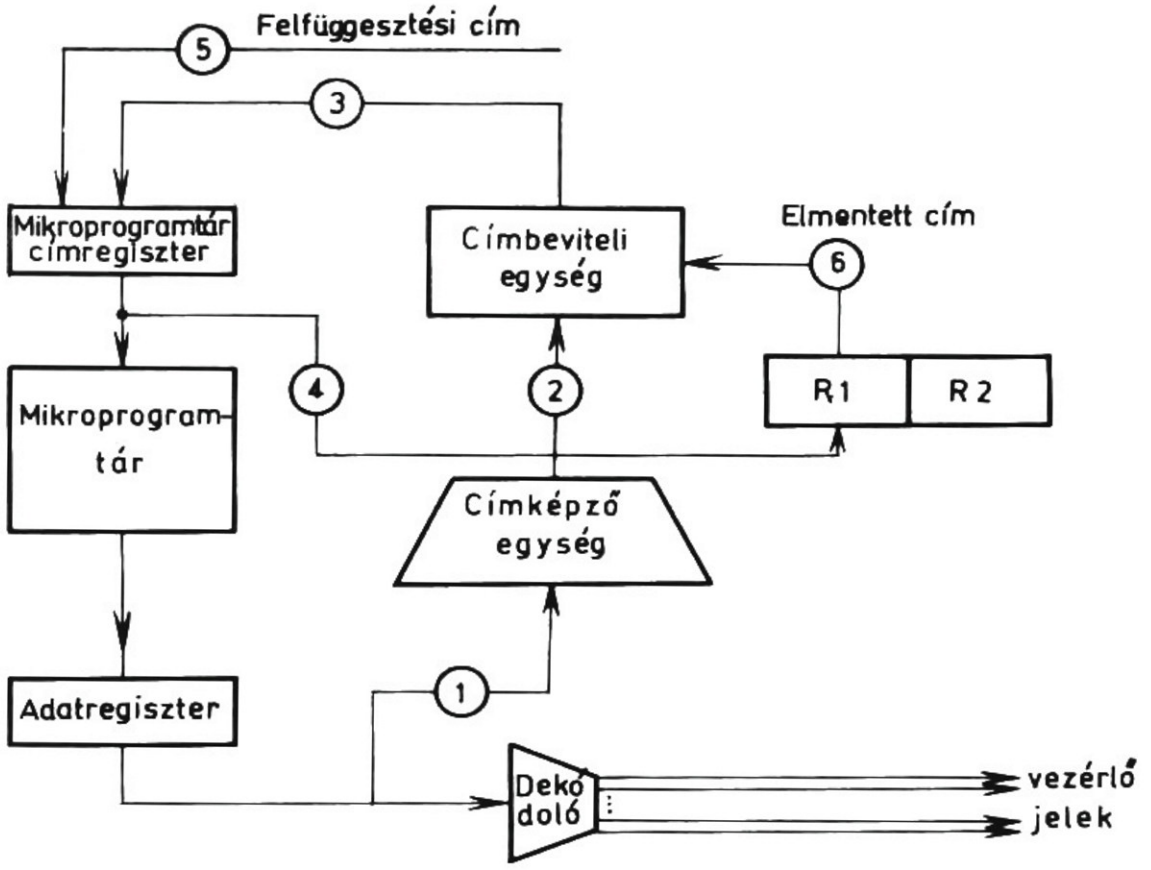
1. a végrehajtandó mikroutasitás cimének betöltése a mikroprogramtár cimregiszterébe;
2. a megcimzett mikroutasitás kiolvasása a mikroprogramtár adatregiszterébe;
3. az adatregiszter tartalmának dekódolása (vezérlő jelek kiküldése) és a következő mikroutasitás cimének kiszámitása.

A mikroprogram futása alatt tehát általában a fenti lépések ciklikus, adott sorrendü ismétlődését értjük.

A 74. ábrán feltüntettünk még két segédregisztert is. Ezek arra szolgálnak, hogy a kiszámitott mikroutasitás-cimet tárolják olyankor, ha fel kell függeszteni a mikroprogramfutás normális menetét. Ilyen jellegü felfüggesztésre például akkor kerülhet sor, ha az adott mikroprogram szempontjából, annál "fontosabb", "elsőbbséget élvező" esemény történik a számitógépben. Ilyen esemény az, ha valamelyik csatorna adatátvitelt akar kezdeményezni az operativ tárral, valamelyik speciális mikroprogramot akarjuk szándékosan elinditani stb. A mikroprogramfelfüggesztés időtartama néhány μs-tól néhányszor 100 μs-ig terjedhet.

A felfüggesztés lefolyását a 75. ábra mutatja.

Abban a pillanatban, amikor valamilyen felfüggesztési kérelem érkezik a vezérlő egységhez, az megvárja a futó mikroutasitás végrehajtását. Miután kiszámitotta a következő mikroutasitás cimét - itt folytatnánk normális esetben a mikroprogramot, ha felfüggesztés nem lenne - ezt a cimet betöltjük valamelyik segédregiszterbe. Ezután a mikroprogramtár cimregiszterébe annak a mikroprogramnak a kezdőcime kerül, amely a felfüggesztéskérést kiszolgálja, például ha a multiplex csatorna akar az operativ tárhoz fordulni, akkor a multiplex csatornát kiszolgáló mikroprogram kezdőcime stb. Miután ez a mikroprogram lefutott, a megfelelő segédregiszterből visszatöltjük a kimentett cimet a mikroprogramtár cimregiszterébe, és a félbeszakitott mikroprogramot ott folytatjuk, ahol abba-



75. ábra. A mikroprogram felfüggesztésének folyamata

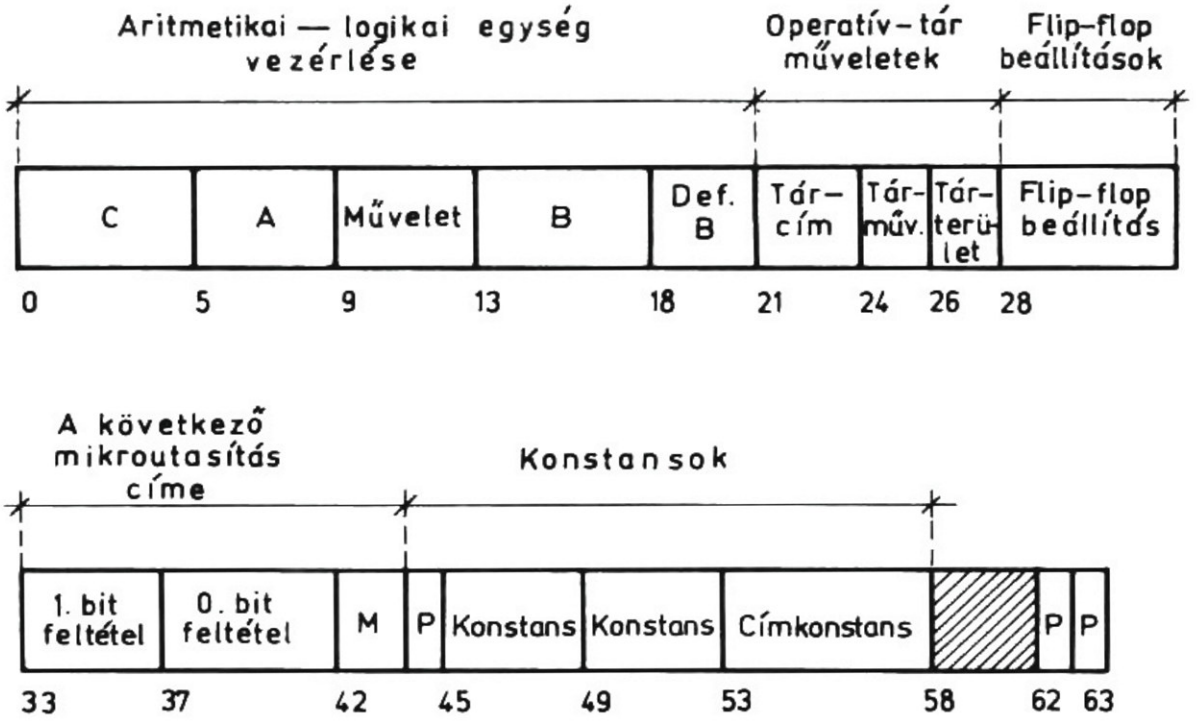
hagytuk. Mindez igen gyorsan játszódik le; előző megfigyelésünket tehát annyiban tehetjük pontosabbá, hogy a müködő számitógéprendszer, amelyben a központi számitógép és a perifériák szimultán üzemelnek, nem egyetlen mikroprogramot futtat folyamatosan, hanem a futó mikroprogram mindig felfüggesztődik egy-egy "pillanatra", vagyis nagy a valószinüsége annak, hogy még egy rövidke kis mikroprogram is csak több "részletben", többszöri felfüggesztések közbejöttével tud lefutni. Ha meggondoljuk, ez nem jelent hátrányt, sőt: ellenkezőleg, éppen ennek révén válik lehetővé a teljes számitógéprendszer szimultán üzemeltetése.

Végül ejtsünk néhány szót arról, miért nevezzük ezt a programot "mikro" programnak. Az elnevezés azzal indokolható, hogy a mikroutasitások a számitógép müködésének legaprőbb, legelemibb lépéseit valósitják meg. A mikroprogramtár tartalma pedig annyiban tekinthető programnak, hogy egy előre elgondolt müveletsorozatot hajt végre, és ez nagyjából megfelel a program értelmezésének.

3.4.3 PÉLDA KONKRÉT MIKROPROGRAMOZÁSI RENDSZERRE. A MIKROUTASÍTÁS SZERKEZETE

A 76. ábra az R20 számitógép mikroutasitásának felépitését mutatja.

Vizsgáljuk meg, mi a szerepe az egyes bitcsoportoknak! Azt már tudjuk, hogy a mikroutasitás egyes mezői tulajdonképpen kódolt vezérlő jelek; ezek "megfejtése" alapján adja ki a vezérlő egység a tényleges vezérlő jeleket.



76. ábra. Az R20 számitógép mikroutasitása

A mikroutasitás öt fő részre osztható:

a) Az aritmetikai-logikai egység müködését vezérlő mikromüveletek,

b) Operativ-tár müveletek,

c) A vezérlő egység belső flip-flopjait beállitó müveletek,

d) A következő mikroutasitás cimének kiszámitása,

e) Konstansok és paritásbitek.

Vegyük sorra az egyes fő részek információtartalmát és annak jelentését !

a) Az aritmetikai-logikai egység müködését vezérlő mikromüveletek egyértelmüen meghatározzák az elvégzendő aritmetikai vagy logikai müveletet. Ennek kódja a Müvelet mezőben áll, és 16 különböző értéket vehet fel. Eszerint az aritmetikai-logikai egység 16 elemi müveletre képes; ezekből rakja össze a bonyolultabb számitásokat. Ilyen müvelet például a bináris, decimális összeadás, kivonás, logikai léptetés 1 bittel, VAGY-kapcsolat, komplementálás stb. Az A és B mező tartalma azt határozza meg, hogy az aritmetikai-logikai egység bemeneti regiszterei honnan kapják az operandusokat, C mező pedig megadja, hogy a kiszámitott eredményt melyik regiszterbe kell továbbitani. Az eredmény maradhat a vezérlő egységen belül, de továbbitható közvetlenül valamelyik csatornán is. A B regiszter deformációja azt jelenti, hogy a B regiszter tartalmának bal és jobb felét (4-4 bitet) meg kell-e cserélni, nullázni kell-e, vagy változatlanul kell hagyni stb. Ezzel tehát mindent megadtunk, amit az aritmetikai-logikai egységnek "tudnia" kell: honnan vegye az operandusokat, milyen müveletet hajtson végre velük, és hova továbbitsa az eredményt.

b) Az operativ-tár müveletekre akkor kerül sor, ha a mikroutasitásban tárhivatkozást is akarunk végezni. Ekkor a Tárcim mező tartalmazza annak a regiszternek a kódját, amelyből a tár cimregiszterét kell feltölteni, a Tármüvelet mező megadja, hogy kiolvasás, beirás vagy egyéb tármüvelet ese-

dékes, a Tárterület mező pedig kijelöli az operativ tárnak azt a fő részét, amelyre az utasitásban hivatkozunk (főmemória, lokális tár, multiplex tár vagy kulcstár). Természetesen nem minden mikroutasitás tartalmaz operativ-tár hivatkozást; ezeknél a Tármüvelet mező tartalma zérus.

c) A vezérlő egység belső flip-flopjainak beállitását vezérlő bitcsoport szerepe abban áll, hogy az egyik mikroutasitás valamilyen feltételtől függően "1" vagy "0" értéküre állitja be az illető flip-flopot, és valamelyik rákövetkező mikroutasitás lekérdezi ennek az állapotát. Ezen keresztül tehát az egyes mikroutasitások feltételeket adhatnak át egymásnak, és e feltételek alapján elágaztatható a mikroprogram. Ilyen feltétel például az, hogy keletkezett-e átvitel két operandus összeadásánál stb.

d) A következő mikroutasitás cimének kiszámitása igen bonyolult módon történik. A cimképzésben részt vesznek egyrészt a számitási eredményektől függő, tehát előre be nem tervezhető feltételek, valamint előre megadott konstansok. Általában a következő mikroutasitás cimének 0. és 1. bitjét közvetlen vagy közvetett módon az 1. bit, illetve 0. bit feltételek határozzák meg — ezek számitásoktól függő feltételek, tehát előre nem határozhatók meg. Az M mező a cimképzés módját határozza meg; ez szabályozza azt is, hogy a konstansokat hogyan kell felhasználni a cim kiszámitásában. Mindezzel azt akartuk érzékeltetni, hogy a mikroprogramtár jóllehet rögzitett információtartalmu, a mikroprogramnak cim szerinti sokrétü elágaztatása révén rendkivül rugalmassá válik.

e) A mikroutasitás végén vannak összegyüjtve azok a konstansok, amelyek részint a cimképzésben, részint a mikroprogram futásakor mint állandó értékek szerepelnek. Itt vannak azok a fix cimek is, amelyek valamelyik speciális mikroprogram kezdőcimei, például a kezdeti programbetöltés mikroprogramja stb.

3.4.4 A MIKROPROGRAM FUTÁSÁNAK FOLYAMATA. A MIKROPROGRAM-FELFÜGGESZTÉSEK RENDSZERE

Tételezzük fel, hogy valaki elinditotta a számitógépet. Ekkor - mint korábban már láttuk - hacsak a gép valamilyen hiba miatt le nem állt, folyamatosan fut a központi számitógépben valamelyik mikroprogram. Mit jelent az, hogy fut a mikroprogram? Ha végiggondoljuk az eddig elmondottakat, könnyen elképzelhetjük magunknak, hogyan zajlik le ez a folyamat.

Az első mikroutasitás végrehajtására ugy került sor, hogy annak cimét valahogyan betöltöttük a mikroprogramtár cimregiszterébe. Ennek a cimnek a dekódolása révén jutottunk el a mikroprogramtár egy adott rekeszéhez, és onnan a megcimzett mikroutasitást a vezérlő egység kiolvasta a mikroprogramtár adatregiszterébe. Ez a folyamat a mikroutasitás lehivása. Ezek után a vezérlő egység hozzálátott a mikroutasitás "megfejtéséhez", az egyes mezők - bitcsoportok - dekódolásához. Mivel ezek a bitcsoportok egymástól függetlenek, dekódolásuk időben egyszerre történhet. A dekódolt mezők alapján most már kiadhatja a vezérlő egység a szükséges vezérlő jeleket. Ezt nevezzük a mikroutasitás végrehajtásának. Figyeljük meg, hogy a vezérlő jelek egy része éppen a következő mikroutasitás cimét szolgáltatja! Ez a cim ismét bekerül a cimregiszterbe, és a leirt folyamat elölről kezdődik. A mikroprogram futása tehát nem más, mint az egyes mikroutasitások egymás utáni lehivása és végrehajtása. Mivel minden mikroutasitás tartalmazza az utána következőnek a cimét, nem szükséges,

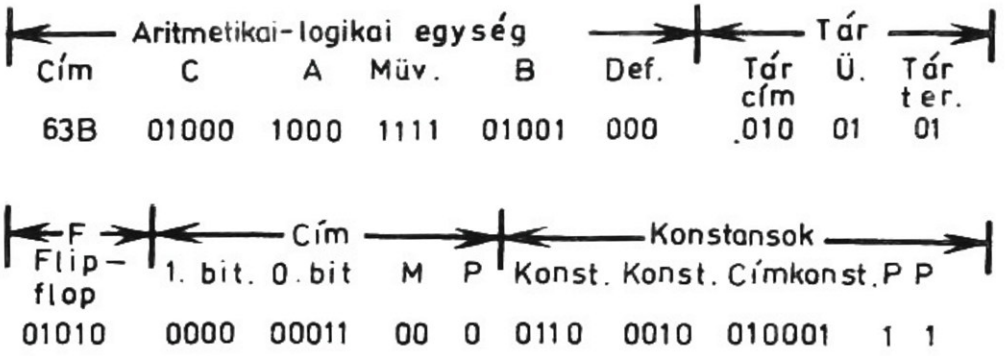
hogy a mikroutasitások a mikroprogramban logikailag is egymás után helyezkedjenek el. A tárban "ugrálhatunk" egyikről a másikra.

Mindez azonban csak arra az esetre vonatkozik, ha a mikroprogram futását semmilyen külső körülmény nem zavarja meg. Ha ez nem áll fenn, akkor el kell térnünk a "normális" mikroprogramfutástól, rövid időtartamokra fel kell függesztenünk a mikroprogramot. Ilyen külső, "zavaró" feltétel lehet például az, hogy valamelyik csatorna adatátvitelt akar kezdeményezni az operativ tárral. Mivel az összes tárművelet mikroprogramozott, a csatorna is ugyanazt a mikroprogramozott vezérlő egységet használja, ha a tárhoz fordul! Emiatt arra van szükség, hogy a csatornát soron kivül kiszolgáljuk, hiszen a kötött sebességviszonyok miatt a csatorna "nem várhat". Ezt biztositják azok a mikroprogramok, amelyek a csatornák kiszolgálására vannak, és csupán a csatorna adatátvitelét szervezik. A felfüggesztett mikroprogram azonban csak akkor folytatható, ha elmentjük a következő mikroutasitás cimét, ahova majd vissza kell térnünk. Ennek az elmentett cimnek a tárolására szolgál az R1 és R2 segédregiszter. R1-be akkor kerül a cim, ha a multiplex csatorna kért felfüggesztést, R2-be pedig akkor, ha valamelyik szelektorcsatorna. A szelektorcsatornák elsőbbséget élveznek a multiplex csatornával szemben: felfüggeszthetik a multiplex csatornát kiszolgáló mikroprogramot is! Ekkor mind R1, mind R2 egy-egy visszatérési cimet tartalmaz. A mikroprogramfelfüggesztés időtartama változhat: 2-3 us-tól 200-300 us-ig attól függően, milyen tipusu adatátvitelt kezdeményezett a csatorna. A 75. ábra egyben a multiplex csatorna felfüggesztés kérésének folyamatát is feltünteti. Az 1-3 esemény a normális mikroprogramfutást mutatja. Közben azonban felfüggesztéskérés érkezett, tehát a mikroutasitás cimét el kell menteni (4) az R1 segédregiszterbe. Ugyanakkor a cimregiszterbe be kell tölteni a multiplex csatornát kiszolgáló mikroprogram kezdőcimét (5), a felfüggesztési cimet. Miután megtörtént a csatorna kiszolgálása, folytatható a felfüggesztett mikroprogram: az elmentett cimet (6) alatt visszatöltjük a cimregiszterbe.

Ha most figyelembe vesszük az eddig elmondottakat, és ugy tekintünk bele a vezérlő egység működésébe, azt látjuk, hogy minden időpillanatban valamely mikroutasitás feldolgozása folyik, de a mikroprogram futása egyáltalán nem "folytonos": tele van rövid időtartamu felfüggesztésekkel! A segédregiszterek tehát állandóan dolgoznak: a csatornák közel folyamatos működése miatt igen gyakran van szükség cimmentésre. Mindez rendkivül szoros hasonlóságot mutat a később tárgyalandó gépi programmegszakitással, valamint a software szintü megszakitáskezeléssel. Egyelőre elég, ha itt a legalacsonyabb szintü megszakitást, az ugynevezett mikroprogramfelfüggesztést jól átgondoljuk; az analógiákra később is mindenütt rámutatunk majd.

3.4.5 PÉLDA MIKROUTASÍTÁSRA. MIKROPROGRAM-LISTA ÉRTELMEZÉSE

A mikroprogramoknak kétféle dokumentálása terjedt el: a gépi mikroprogram-lista és a szimbolikus mikroprogram-lista. A kettő között az a különbség, hogy a szimbolikus listán a cimek, konstansok, müveletek decimálisan, illetve szövegesen szerepelnek, mig a gépi lista a mikroutasitások közvetlen "másolata", bináris alakban. Nyilvánvaló, hogy a mikroprogramok "olvasására" a szimbolikus irásmód a kényelmesebb; a mikroprogramok pontos felépitését viszont a gépi listából ismerhetjük meg. Alább egyetlen mikroutasitást ismertetünk vázlatosan; a gépi lista egy kiragadott elemét.



77. ábra. A gépi mikroutasitás-lista értelmezése

A listán elkülönitve jelennek meg a mikroutasitás fő mezői. Az aritmetikai-logikai egységet vezérlő mezőkben álló kódokat táblázatból visszakeresve megtudjuk, hogy honnan veszi a gép az operandusokat, melyik regiszterbe viszi az eredményt, és milyen elemi müveletet végez. A tárra vonatkozó rész közli, hogy milyen tármüveletet kell végezni, honnan kell venni a tárcimet, és melyik memóriaterülethez történik hozzáfordulás. A flip-flop beállitás mezőjében szereplő kód azt dönti el, hogy melyik flip-flopot kell beállitani, illetve törölni. A következő mikroutasitás cimét meghatározó mezők eldöntik, hogy a cim 0. illetve 1. bitjét közvetlenül az utasitásból kell meghatározni, vagy ehhez fel kell használni valamelyik belső flip-flop állapotát. A mező végén egy paritásbit is található. Végül a mikrokonstansok vannak kigyüjtve: a két első mező a 8 bites konstans bal és jobb felét tartalmazza, az ezt követő 6 bites mezőben pedig mikroutasitások cimkonstansa áll. Ezt a következő utasitás cimének kiszámitásánál esetleg figyelembe kell venni. Az utolsó két bit szintén paritásbit.

Felhivjuk a figyelmet arra, hogy egy adott mikroutasitás áttanulmányozása után nem feltétlenül tudjuk meg, hogy hol folytatódik a mikroprogram! Elég emlékeztetnünk arra, hogy a mikroprogram futása során minden egyes mikroutasitás végrehajtása után szükség lehet arra, hogy a keletkezett részeredmény figyelembe vételével folytassuk a mikroprogramot. Ez pedig egyértelmü a mikroprogram elágaztatásával, esetleg sokféle irányba! A mikroutasitás cimszámitásából tehát általában azt tudhatjuk meg, hogy mely cimeken folytatódhat a mikroprogram; a tényleges folytatásról az adott körülmények döntenek.

3.4.6 BEVEZETÉS A GÉPI KÓDÚ PROGRAMOZÁS ELMÉLETÉBE. 4, 3, 2, 1+1 ÉS 1 CÍMES SZÁMÍTÓGÉPEK

A mikroprogramozáshoz teljesen hasonlóan képzelhetjük el a gépi kódu programozást is. A leglényegesebb különbség a kettő között az, hogy tetszőleges gépi kódu programot irhatunk anélkül, hogy a számitógép hardware tulajdonságait megváltoztatnánk, mig a mikroprogramtár átirására általában nincs lehetőségünk. A mikroprogramokhoz hasonlóan a gépi kódu programok is elemi részekből állnak, ezek a gépi utasitások. Gépi utasitáson meghatározott terjedelmü bitcsoportokat értünk, ezek különböző müveleteket jelenthetnek, vagy pedig a központi számitógép regisztereire, operativ tárára stb. hivatkozhatnak. Mivel minden gépi utasitás bitekből épül fel, a gépi kódu program tisztán bináris formáju, 1-esekből és 0-kból áll. Egy-egy számitógépre vonatkozólag a gépi utasitások összességét a számitó-

gép gépi nyelvének nevezzük. A gépi nyelv általában kötött, de szabályai betartása mellett is számos lehetőséget rejt magában a programozás számára. Egy gépi kódu program csak akkor futtatható egy adott számitógépen, ha a gép "érti" ezt a programot, vagyis azt az illető gép "nyelvén" irták. Nagyobb gépcsaládok esetén egységesiteni szokták a gépek nyelvét, igy egy adott gépre irt program a gépcsalád többi tagján is futtatható. Ez vonatkozik például a szocialista országok Egységes Számitógép Rendszerére (ESZR), ahol az R20, R30 stb. számitógépek gépi nyelve azonos.

Vizsgáljuk meg most közelebbről is, milyen felépitésüek a gépi program elemei, a gépi utasitások! Annyiban mindenesetre nagyon hasonlitanak a mikroutasitásokhoz, hogy valamilyen formában tartalmazzák az elvégzendő müveletet kódolt formában, valamint az operandusok és az eredmény cimét. Ezenkivül utalnak a következőként végrehajtandó gépi utasitás cimére. A gépi utasitások válfajait a 78. ábra foglalja össze.

Műveleti kód	1. operandus címe	2. operandus címe	Eredmény címe	Következő utasítás címe

a) 4-címes utasítás (3+1 címes) utasítás

Műveleti kód	1. operandus címe	2. operandus címe	Eredmény címe

b) 3-címes utasítás

Műveleti kód	1. operandus címe	2. operandus címe

c) 2-címes utasítás

Műveleti kód	Operandus címe	Következő utasítás címe

d) 1+1-címes utasítás

Műveleti kód	Operandus címe

e) 1-címes utasítás

78. ábra. Gépi utasitások általános felépitése

a) A 4 cimes (3 + 1 cimes) utasitás a müveleti kód mellett megadja az operandusok cimét, az eredmény cimét és a következő utasitás cimét. Ezek a cimek vagy az operativ tárba mutatnak, vagy egy regisztert jelölnek ki a központi számitógépben. Az ilyen tipusu utasitásokból felépitett program futása hasonló a már megismert mikroprogram futásához: a vezérlő egység egymás után lehivja és végrehajtja a gépi utasitásokat, és ezeknek nem kell egymás "közelében" lenniük az operativ tárban, hiszen minden egyes utasitás tartalmazza az utána következőnek a cimét. Látszólag igen rugalmasan lehet igy programozni, ugyanakkor azonban a program igen sok helyet foglal a tárban, mert hosszuak az utasitásai! A mai korszerü számitógépeken 4 cimes utasitásokkal már nem dolgozhatunk; ezeket a korai számitógépeken használták csupán.

b) A 3 cimes utasitás annyiban jelent egyszerüsödést az előbbihez képest, hogy itt nincs semmiféle utalás arra, hogy hol kell folytatni a programot. Ez hallgatólagos "megállapodást" takar: a 3 cimes utasitások használatánál mindig feltételezzük, hogy a vezérlő egység olyan sorrendben foglalkozik az utasitásokkal, ahogyan azok az operativ tárban egymás után következnek. Az ilyen 3 cimes utasitásokat szintén nem használhatjuk a mai számitógépeken; elavultnak tekinthetők.

c) A 2 cimes utasitás a ma leginkább elterjedt utasitásfajta. Itt még többet kell feltételeznünk a vezérlő egységről: az eredményt általában az 1. operandus cimére teszi. Vagyis az 1. operandus a müveletvégzés után elvész, felüliródik. Ez ellen legtöbbször ugy védekezünk, hogy az 1. operandust "két példányban" tároljuk - például az operativ tárban és egy regiszterben - és a müveletvégzés után valamelyik 1. operandus változatlanul megmarad. Ha a program olyan, hogy később nincs szükség az 1. operandusra, akkor nyugodtan felülirhatjuk az eredménnyel.

d) Az 1+1 cimes utasitásban már csak egyetlen operandust cimezhetünk, a másik rendszerint már eleve a vezérlő egység valamelyik regiszterében áll. Természetesen, külön utasitással kell oda betölteni a másik operandust, mielőtt müveletvégzésre sor kerülhetne. Emiatt az 1+1 cimes utasitásokból felépitett programok hosszabbak a 2 cimes gépek programjainál, de ez nem jelenti azt, hogy futási idejük is nagyobb! Az eredmény általában a vezérlő egység regisztereiben képződik. A program végrehajtásakor "ugrálhatunk" utasitásról utasitásra, hiszen a következő cimét tároltuk.

e) Az 1 cimes utasitásban szintén egyetlen operandust cimezhetünk, az eredmény a vezérlő egység regisztereiben képződik. Ugyanakkor feltételezzük, hogy az utasitások feldolgozása előfordulásuk sorrendjében történik. Az 1 cimes számitógépek szintén igen elterjedtek a mai számitógépparkban.

Mindezek ismeretében már semmi nehézséget nem jelent az, hogy egy gépi kódu programot összeállitsunk. Pusztán azt kell végiggondolnunk, hogy milyen müveleteket akarunk egymás után elvégezni, és ennek megfelelően kell az egyes gépi utasitásokat összeválogatnunk. A korai számitógépfelhasználók igy is dolgoztak, de a gépi kód bináris alakja sok nehézséget jelentett az ember számára, és ezért más, szimbolikus programozási rendszerek terjedtek el. Ezek közös alapja azonban minden esetben a gépi kód, amit a gép közvetlenül "megért".

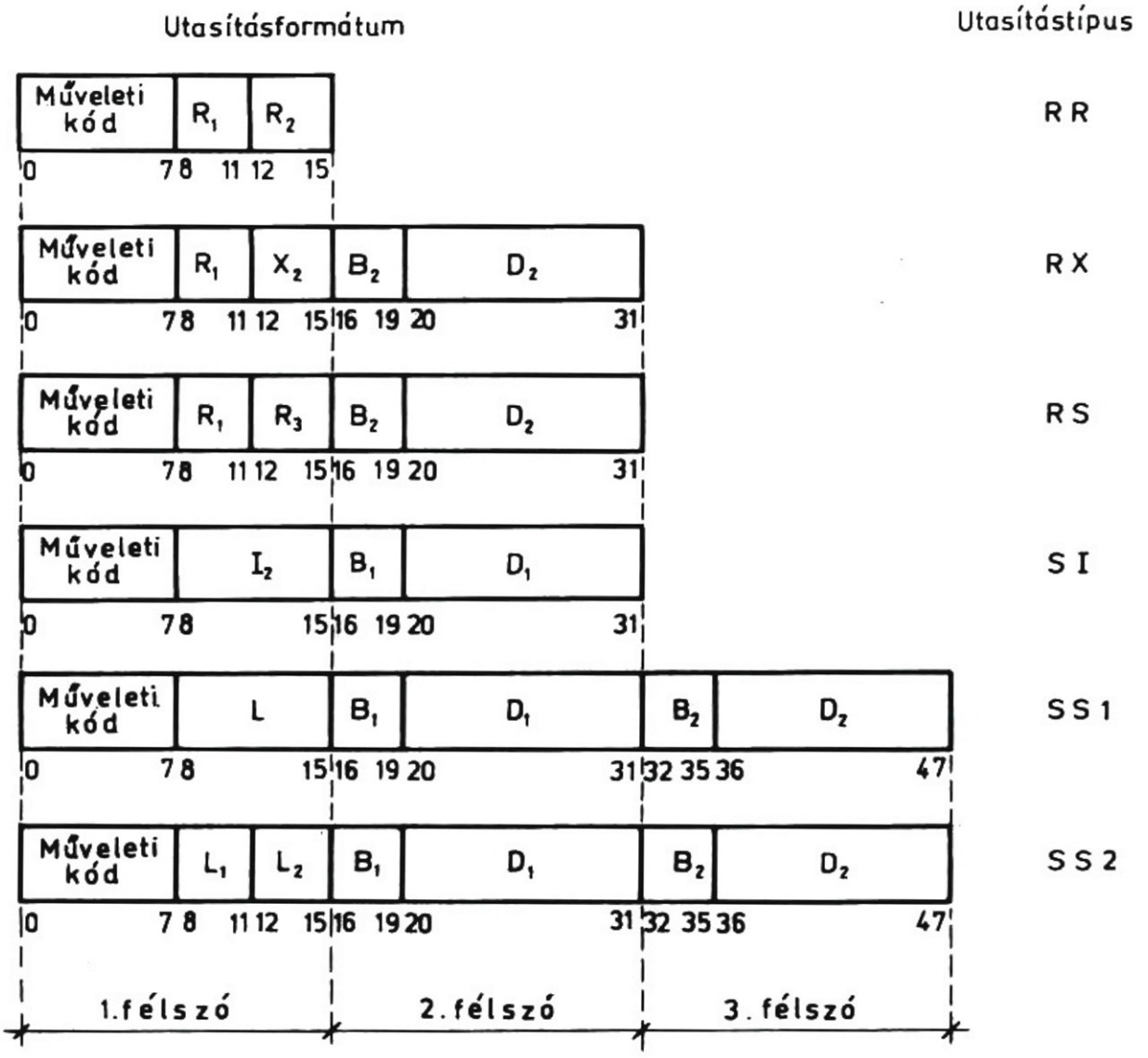
Végül ejtsünk néhány szót a mikroprogram és a gépi kódu program kapcsolatáról. A mikroprogramozásnál láttuk, hogy a vezérlő egység az összetettebb müveleteket elemi lépésekre, ugynevezett mikromüveletekre bontja le, illetve ezekből rakja össze a bonyolultabb müveleteket. Mig az ember szempontjából két szám összeadása egyetlen elemi müveletnek tünik, addig a vezérlő egységnek igen sok apró lépést kell megtennie, mig az eredményig eljut. Igy például ki kell számitania az operandusok

cimét, be kell töltenie őket az aritmetikai-logikai egység bemeneti regisztereibe, el kell végeztetnie az előirt müveletet, az eredményt előirt helyre kell továbbitania, gondoskodnia kell a program folytatásáról stb. Minden ilyen elemi müvelet egy-egy mikroutasitás végrehajtásával egyenértékü. Azt mondhatjuk tehát, hogy egy gépi utasitás végrehajtása megfelel a hozzá tartozó mikroutasitások vagy mikroprogram lefuttatásának. Más szavakkal: minden gépi utasitásnak megvan a maga mikroprogramja, és a vezérlő egység ugy hajtja végre a gépi utasitást, hogy lefuttatja annak mikroprogramját.

3.4.7 A 2 CÍMES SZÁMÍTÓGÉPEK UTASÍTÁSTÍPUSAI. CÍMZÉSI MÓDOK

Foglalkozzunk most kissé részletesebben a legelterjedtebb, 2 cimes utasitások belső felépitésével! Hogyan adhatjuk meg az operandusok cimét, milyen lehetőségeket nyujtanak a különféle cimzési módok? A 2 cimes utasitástipusokat az ESZR gépek utasitásain szemléltetjük.

A számitógép teljes gépi nyelve az itt feltüntetett 6 féle utasitásból kerül ki. Látható, hogy a gép az utasitás első byte-ját minden esetben müveleti kódként értelmezi, az utasitás további része már ti-



79. ábra. Szabványos ESZR utasitások felépitése

pusonként különböző lehet. Az egyes tipusok elnevezése utal az utasitás jellegére is, például RR tipusuak a regisztermüveletek, RX tipusuak az indexelt cimes utasitások stb. Az 1. illetve 2. operandusra vonatkozó információ azonositására szolgál az 1 illetve 2 számjegy, például R_1 az 1. operandusra, B_2 és D_2 a 2. operandusra vonatkozik. Vegyük sorra, hogyan értelmezi a vezérlő egység az utasitások egyes bitcsoportjait, és hogyan számitja ki az operandusok cimét!

a) Az RR tipusu utasitásokban R_1 és R_2 egy-egy általános vagy lebegőpontos regiszter cimét jelenti. Mivel 4-4 bit szolgál a regiszterek cimzésére, összesen 16 regisztert tudunk cimezni ugy, hogy az R_1 illetve R_2 mezőbe beirjuk az illető regiszter számát. Például R_1 = 0011 és R_2 = 1001 esetén 1. operandusunk a 3-as, 2. operandusunk pedig a 9-es regiszterben van. Az ilyen tipusu utasitással tehát nem tudunk az operativ tárhoz fordulni, csak a vezérlő egység regisztereihez. Ugyanakkor ez a fajta utasitás kevés helyet foglal a tárban - mindössze 2 byte-ot - és végrehajtási ideje is igen gyors. Mielőtt végrehajthatnánk, operandusainkat be kell töltenünk a használt regiszterekbe.

b) Az RX tipusu utasitásokban R_1 egy általános vagy lebegőpontos regiszter cimét jelenti. X_2 és B_2 a 2. operandus indexregisztere, illetve bázisregisztere. A 2. operandus cimét tehát nem közvetlenül tartalmazza az utasitás, hanem csupán utalást ad a cim összetevőinek helyére. A 2. operandus cimét az alábbi összefüggés alapján számitja ki a vezérlő egység:

2. operandus cime = $(X_2) + (B_2) + D_2$.

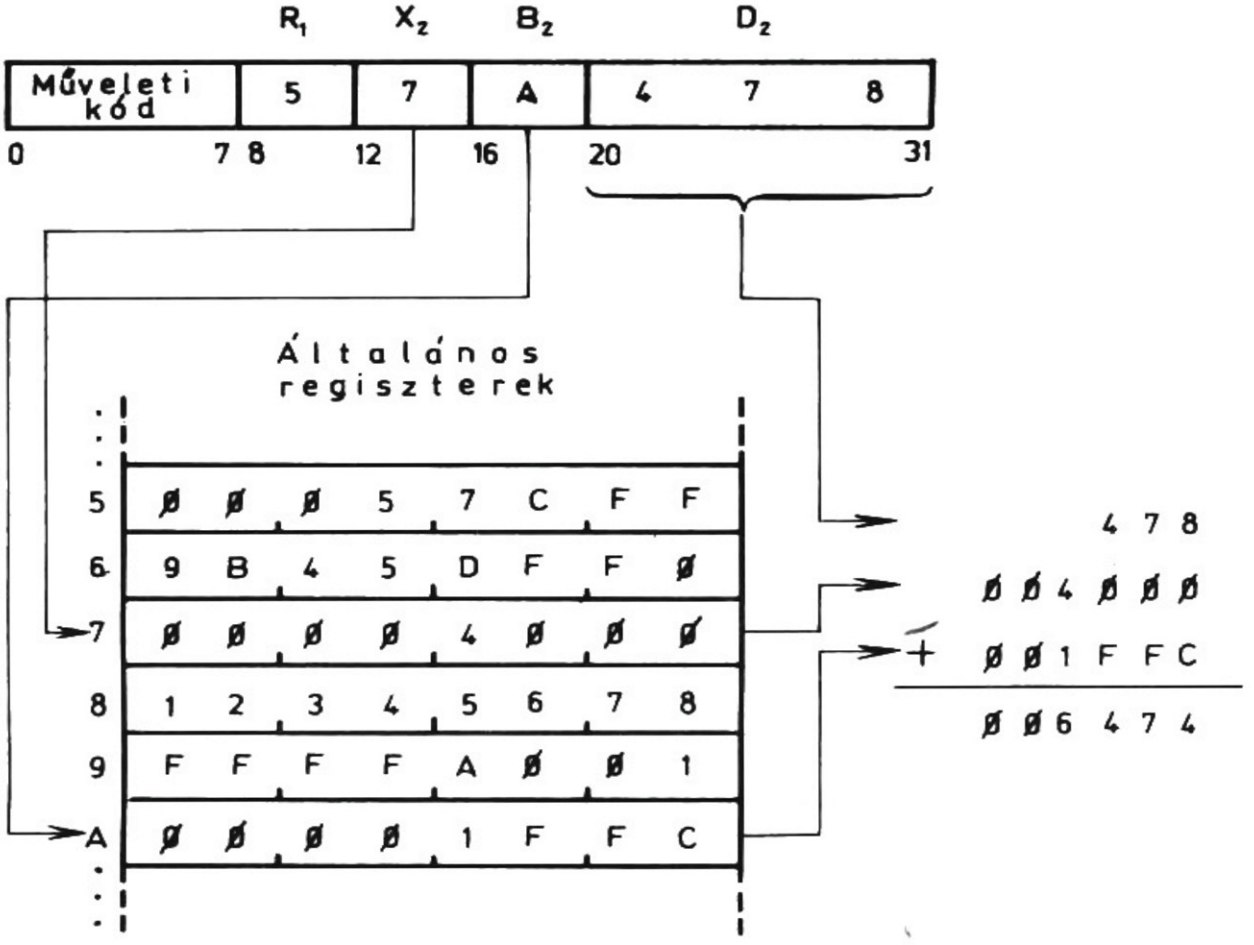
Szavakban: veszi az X_2 és B_2 mezőben megadott általános regiszter tartalmának összegét, és ehhez hozzáadja az utasitásban tárolt D_2 jelü ugynevezett relativ cimet. A zárójelek azt jelentik, hogy az illető regiszter tartalma vesz részt a cimszámitásban, nem maga a regiszter száma. Ez nem vonatkozik a D_2 relativ cimre; itt közvetlenül az utasitásban tárolt 12 bites D_2 szám a cim komponense. A cim kiszámitását szemlélteti a 80. ábra. A számjegyek hexadecimálisak. A 2. operandus cime 6474.

Látjuk, hogy elég bonyolult módon számitható az indexelt cim. Mi lehet az előnye mégis, hogy ezt használjuk? Képzeljük el, hogy a tárban 4 byte-os operandusok helyezkednek el egymás után. Ilyenkor egy-egy operandus feldolgozása után a tárcimet 4-gyel kell növelnünk ahhoz, hogy a következő operandusra mutasson. Ha az utasitás közvetlenül az operandus cimét tartalmazná, minden ujabb cim kiszámitásához át kellene irni magát az utasitás cimrészét! Mivel azonban a cim egyik komponense az indexregiszter tartalma, elegendő ebbe a regiszterbe esetenként 4-gyel nagyobb értéket tölteni, tartalmát 4-gyel növelni és máris kész az uj operanduscim. Igy egész tárterületeken lehet végighaladni egyetlen utasitással.

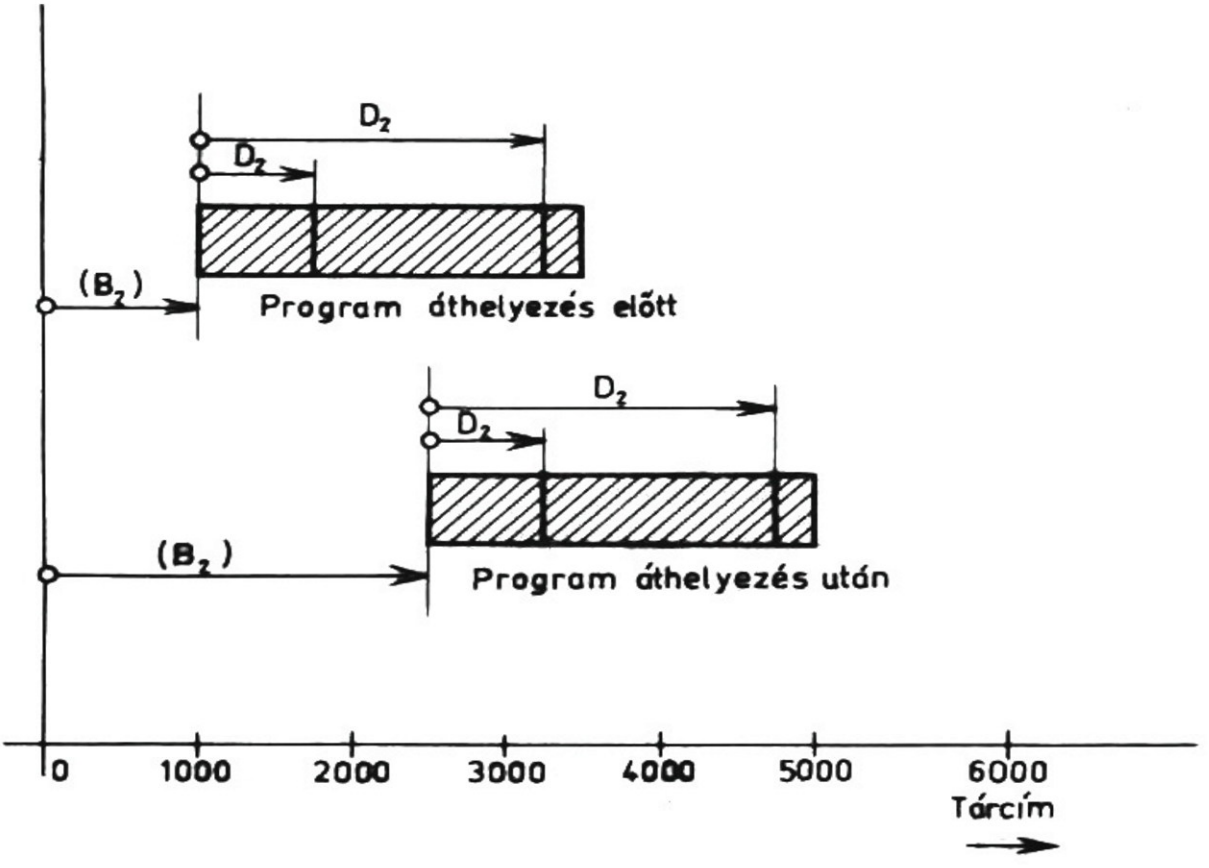
A bázisregiszter és relativ cim használata azért indokolt, mert általában nem tudjuk előre, hogy hol fog tartózkodni gépi kódu programunk a futás során: általában ott, ahol szabad hely van az operativ tárban. Ezért a programot ugy irjuk meg, hogy kezdőcimét a bázisregiszterbe töltjük, és a program "belsejében" a D_2 relativ cim segitségével cimzünk.

Ha a programnak valamilyen okból kifolyólag másik kezdőcimtől kell indulnia, mint előzőleg, akkor most ezt az uj cimet töltjük be a bázisregiszterbe, és az összes belső cim ennek értelmében módosul. Tehát áthelyezhető a program anélkül, hogy át kellene irnunk!

A relativ cim ezenkivül igen jól használható még arra, hogy segitségével több byte hosszuságu operandusok belsejét elérjük. Képzeljük el azt, hogy 4 byte-os operandusaink egymás után helyezkednek el az operativ tárban. Ugyanakkor azonban minden 4 byte-os mezőnek csak a "jobb felét" akarjuk



80. ábra. Indexelt cim kiszámitása



81. ábra. Bázis- és relativ cimek használata

feldolgozni. Az indexregiszterrel megoldhatjuk azt, hogy "végiglépegessünk" az operandusokon 4 byte-onként, a relativ cimet pedig 2-re állitjuk (D_2=0000 0000 0010). Igy minden operandus 3-4. byte-ját cimezzük, és az indexeléssel automatikusan az operandusok jobb felén haladunk végig.

Mivel a relativ cim maximális értéke 4095 (2^{12}-1), egyetlen bázisregiszterrel maximálisan 4096 byte-ot cimezhetünk. Ha a program ennél nagyobb méretü, több bázisregiszterre van szükség.

c) Az RS tipusu utasitásokban R_1 és R_3 egy-egy általános regisztert jelöl ki, B_2 és D_2 pedig a 2. operandus bázisregiszterét és relativ cimét adja meg. Látjuk, hogy itt három cim szerepel; emiatt az RS tipusu utasitásokat 3 cimesnek tekinthetjük. Néhány RS tipusu utasitásban azonban az R_3 mező nincs kihasználva. Ide tartoznak azok az utasitások is, amelyek egy vagy két regiszter tartalmát adott számu bittel jobbra-balra léptetik.

d) Az SI tipusu utasitások érdekessége, hogy mig az 1. operandus cime bázis- és relativ cim alakjában adott (B_1 és D_1 mező), addig a 2. operandus mindig 1 byte hosszuságu, és közvetlenül az utasitásban foglal helyet (I_2). Ez különösen előnyös akkor, ha az operandusok hossza 1 byte (például egy karakter 8 bites kódjának feldolgozásánál), mert nem kell külön tárolni a 2. operandus cimét. Ehhez a tipushoz tartoznak az input/output utasitások is.

e) Az SS1 és SS2 tipusu utasitások mindkét operandusa az operativ tárban van, és ezek cimét bázis- és relativ cim alakjában tároljuk. Az utasitások második byte-ja vagy az operandusok megegyező hosszára (L) utal, vagy pedig az 1. és 2. operandus hosszát külön-külön tárolja (L_1 és L_2). Az operandushosszt mindig byte-okban mérjük, és az utasitás a tényleges hossznál eggyel kisebb értékü bináris számot tárol. Igy például az SS1 tipusu utasitások operandusainak hossza 1,2,..,256 byte lehet, és ennek megfelelően L=0, 1,...,255. Hasonlóképpen az SS2 tipusu utasitások operandusainak hossza 1,2,...,16 byte lehet, és L_1=0, 1,...,15, L_2=0, 1,...,15. Itt azonban nem szükséges az, hogy az operandusok egyenlő hosszuságuak legyenek. Ide tartoznak a decimális aritmetikai utasitások, az SS1 tipushoz pedig általában karaktermozgató, logikai utasitások sorolhatók.

Bizonyos esetekben szükség lehet arra, hogy egy operandus cime ne függjön a bázisregiszter és indexregiszter tartalmától, hanem állandó, rögzitett értékü legyen. Ilyen cimet ugy tudunk előállitani, hogy az illető utasitás B_1, B_2 illetve X_2 mezőjébe zérust irunk. Ezt a vezérlő egység nem ugy értelmezi, hogy a megfelelő cimkomponenst a 0-ás regiszterből kell vennie, hanem speciálisan ugy, hogy a cimkomponenst a cimszámitásnál el kell hagyni. Az olyan RX tipusu utasitásban tehát, amelyben X_2 = 0000 és B_2=0000, a 2. operandus cime = D_2, és igy a relativ cimből abszolut cim lett. Ilyen abszolut cimekkel az operativ tár első 4096 byte-ját tudjuk csupán elérni, efölött bázisregiszteres cimzést kell használnunk. Ez a megkötés nem vonatkozik az R_1, R_2 és R_3 mezőkre.

3.4.8 A GÉPI KÓDÚ PROGRAM FUTÁSA. A PROGRAM ÁLLAPOTÁNAK NYILVÁNTARTÁSA. A PROGRAMÁLLAPOTSZÓ FELÉPÍTÉSE

A mikroprogramozásnál láttuk, hogy a központi számitógépben - hacsak a gép valamilyen okból le nem állt - állandóan fut egy mikroprogram: a vezérlő egység "dolgozik". Mivel a vezérlő egység összes tevékenysége mikroprogramozott, nem biztos, hogy egy adott időpillanatban éppen egy gépi utasitáshoz tartozó, azt megvalósitó mikroprogram fut! Gondoljunk például arra az esetre, amikor

valamelyik csatorna mikroprogramozott kiszolgálása folyik; ezzel egyidőben gépi utasitások végrehajtása nem lehetséges, hiszen a vezérlő egység foglalt. Mindezek alapján azt mondjuk a gépi kódu programról, hogy feldolgozása folyik, ha a benne foglalt gépi utasitások mikroprogramjai futnak. Mivel a gépi utasitás több mikroutasitásnak felel meg, és a mikroprogram minden mikroutasitás végrehajtása után felfüggeszthető, a gép kódu program futása is szaggatott, időben nem folytonos. Ha visszagondolunk a mikroprogram futására, és azt összehasonlitjuk a gépi kódu program futásával, azt tapasztaljuk, hogy a két folyamat igen sokban hasonlit egymáshoz. A vezérlő egység lehivja a soron következő gépi utasitást, értelmezi azt (megfejti a müveleti kódját), kiszámitja a szükséges operanduscimeket, majd a müveleti kód által meghatározott müveletet végrehajtja vagy végrehajtatja a számitógéprendszer valamelyik elemével (például az aritmetikai-logikai egységgel, csatornákkal stb.). Ezután következik az ujabb gépi-utasitás lehivás, utasitásvégrehajtás, és ez igy megy mindaddig, amig a gépi kódu program végrehajtása be nem fejeződik.

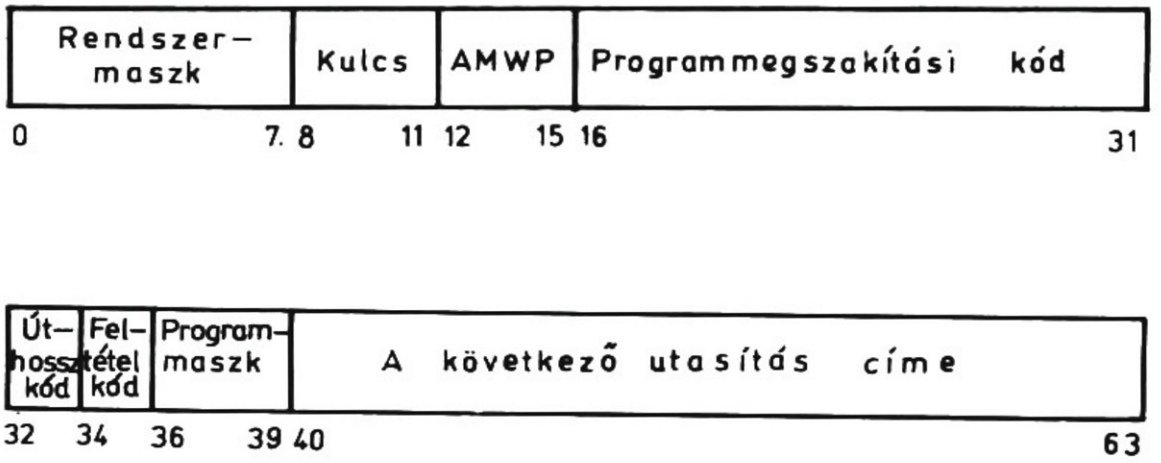
Lényeges különbség a mikroprogram és a gépi kódu program futása között, hogy a mikroutasitásokat a fix mikroprogramtárból veszi a vezérlő egység, a gépi kódu program utasitásait pedig az operativ tárból. Ezenkivül a kétféle program külön cim- és adatregisztereket használ; a mikroprogramot a gépi kódu program "kifejtésének", részletesebb leirásának, lépésenkénti megvalósitásának tekinthetjük.

Emlékezzünk arra, hogy a mikroprogramozásnál is meg kellett engednünk, hogy a futó mikroprogram tudomást szerezzen bizonyos külső feltételekről. Ezek hatással voltak a mikroprogram futására, és általában annak felfüggesztését eredményezték. Hasonlóan kell eljárnunk a gépi kódu program esetében is: itt is létre kell hoznunk olyan rendszert, amely a gépi kódu programot függeszti fel külső feltételek hatására. A felfüggesztést gépi kódu program esetén programmegszakitásnak vagy egyszerüen csak megszakitásnak nevezzük. Minden programmegszakitás alkalmával el kell mentenünk a program aktuális állapotát, végre kell hajtanunk egy olyan gépi kódu programot, amely a megszakitási feltételt kezeli, és végül vissza kell térnünk a megszakitott programba oda, ahol annak feldolgozását abbahagytuk. Mindez azt teszi szükségessé, hogy a gépi kódu program állapotát lépésről lépésre nyilvántartsuk, figyelemmel kisérjük. Emlékeztetünk arra, hogy a mikroprogram-felfüggesztés során az elmentett információ a következő mikroutasitás cime volt, ennek alapján lehetett visszatérni. A gépi kódu program esetén - mivel ez összetettebb, mint a mikroprogram - nem elegendő a következő gépi utasitás címének elmentése; itt részletesebb információra van szükség a megszakitáskezeléshez és a visszatéréshez. Ezt az információt 64 bites egységbe foglalták össze, és programállapotszónak nevezzük, utalva arra, hogy ebben kisérjük figyelemmel a gépi kódu program futását.

A programállapotszó felépitését a 82. ábra mutatja.

Funkció szempontjából háromféle programállapotszót különböztetünk meg: aktuális, régi és uj programállapotszót. Az aktuális az, amely lépésről lépésre követi a program futását, a régi programállapotszó tartalmazza a megszakitáskor elmentett információt, az uj programállapotszó pedig a megszakitáskezelő program állapotának nyilvántartását végzi. A programmegszakitás tehát minden esetben programállapotszó-cserével jár együtt:

aktuális programállapotszó ⟶ régi programállapotszó
uj programállapotszó ⟶ aktuális programállapotszó.



82. ábra. A programállapotszó felépitése

Tekintsük most a programállapotszó felépitését!

a) Rendszer maszk. A rendszer maszk arra szolgál, hogy a csatornarendszertől érkező, nem kivánatos megszakitási feltételek érvényre jutását elkerüljük, továbbá letiltsuk az ugynevezett külső megszakitásokat. A bitek értelmezése az alábbi:

0. bit . . . . . multiplex csatorna maszk
1. bit . . . . . 1. szelektorcsatorna maszk
2. bit . . . . . 2. szelektorcsatorna maszk
3-6. bit . . . . . 3-6. szelektorcsatorna maszk
7. bit . . . . . külső megszakitás maszk.

Ha egy megszakitási feltételt el akarunk nyomni, akkor a megfelelő bitet 0-ra állitjuk. Például a 00100000 rendszer maszk csupán a 2. szelektorcsatorna megszakitási feltételeit engedélyezi, a többit letiltja. A maszkolás jelentőségére a 3.4.9 pontban még visszatérünk.

b) Kulcs. Ez a 4 bites mező tartalmazza a futó program tárvédelmi kulcsának numerikus részét. Ezt az értéket használja a vezérlő egység akkor, amikor a tárvédelmi mechanizmust müködteti.

c) "A" - ez a bit jelzi, hogy a gép belső kódja ASCII vagy EBCDIC. Általában EBCDIC kódban dolgozunk.

d) "M" - géphiba maszk. Ezzel a bittel lehet letiltani a géphiba miatti programmegszakitásokat.

e) "W" - ez a bit jelzi, hogy a vezérlő egység gépi kódu programot futtat, vagy pedig várakozik. A várakozási állapotban is futhatnak mikroprogramok, de ezek nem lehetnek gépi utasitások mikroprogramjai! A felhasználó szemszögéből nézve várakozási állapotban a "gép áll".

f) "P" - ez a bit jelzi, hogy a vezérlő egység problémaállapotban vagy supervisorállapotban van-e. A két állapot abban különbözik egymástól, hogy problémaállapotban bizonyos utasitások nem hajthatók végre. Ezek az ugynevezett privilegizált utasitások, és általában a rendszer normális müködését érintik (például tárvédelmi kulcsok beállitása, input/output müveletek stb.). Ezért tartja fenn ezeket magának az operációs rendszer vezérlőprogramja, a supervisor.

g) A programmegszakitási kódnak az aktuális programállapotszóban nincs jelentősége. Ide irja be a vezérlő egység azt a bináris kódot, amely jellemző a megszakitás okára, forrására; ezt elemzi a megszakitáskezelő program, hogy a szükséges intézkedéseket megállapitsa.

h) Az utasitáshosszkód tárolja a végrehajtás alatt álló gépi utasitás hosszát félszavakban kifejezve. Erre akkor lehet szükség, ha egy gépi utasitást ujból végre kell hajtani például hibás müködés miatt. Ekkor innen tudjuk, hogy hány félszót kell visszaszámlálni az utasitás elejéig.

i) A feltételkód tárolja egyes gépi utasitások végrehajtásának eredményét. Például az összehasonlitó müveletek eredménye lehet az, hogy az 1. és 2. operandus egyenlő, az 1. operandus kisebb, vagy a 2. operandus kisebb stb. A feltételkód lekérdezésével lehet elágazni a gépi kódu programban. A gépi utasitások egy része állitja a feltételkódot, más része változatlanul hagyja azt.

j) A program-maszkban 4 programhiba eredetü megszakitási feltételt lehet letiltani. A bitek jelentése:

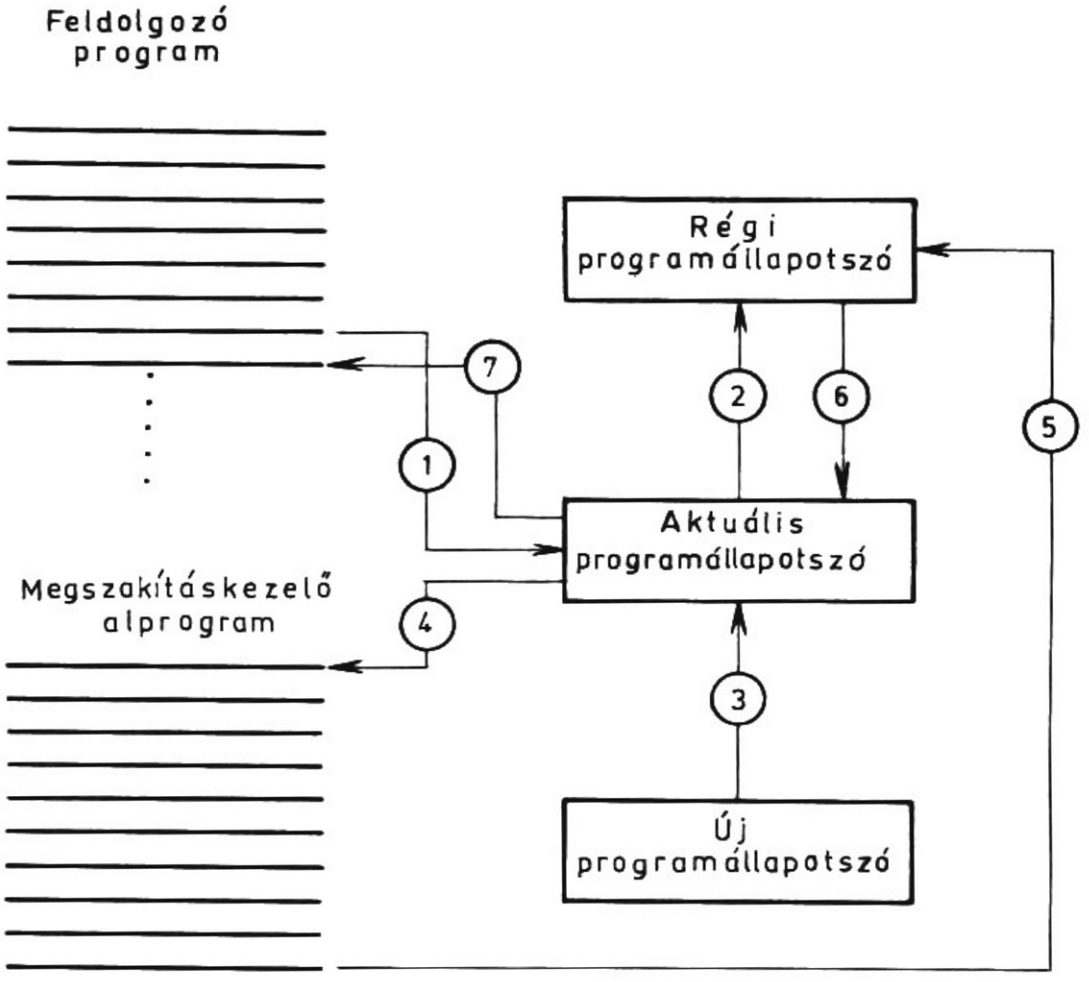
36. bit . . . . . Fixpontos tulcsordulás maszk
37. bit . . . . . Decimális tulcsordulás maszk
38. bit . . . . . Kitevőeltünés maszk
39. bit . . . . . Mantissza nulla maszk

k) A 40-63. bitek tartalmazzák a következőként végrehajtandó utasitás cimét. Közönséges utasitásoknál ez a cim a sorrendben következő utasitás kezdetére mutat, ugróutasitásoknál pedig a programnak arra a helyére, ahová a vezérlést át kell adni. A megszakitás kezelése után ez a cim lesz az, ahova vissza kell térni a megszakitott programba. Ez az értelme annak, hogy megszakitás alkalmával a következő utasitás cimét mentjük: igy a visszatérési cim közvetlenül rendelkezésre áll.

3.4.9 A PROGRAMMEGSZAKÍTÁSI RENDSZER MŰKÖDÉSE. A MEGSZAKÍTÁSOK PRIORITÁSA ÉS A FELDOLGOZÁSI SORREND

A gépi kódu programot általában igen sok feltétel szakithatja félbe; ezek valamennyien olyan külső körülmények, amelyek figyelembe vételére szükség van ahhoz, hogy a program az előirásnak megfelelően futhasson tovább. A sokféle megszakitási feltételt rendszerbe foglalták, igy a számitógépeknél megszakitási rendszerről beszélünk. Ennek szabályai rögzitik az egyes megszakitások kezelését, valamint adott sorrendet állapitanak meg az egyidejüleg fellépő megszakitási feltételek érvényre jutása között. Minden programmegszakitás programállapotszó-cserével kezdődik, majd ezek után az uj (most már aktuális) programállapotszó vezérlése alatt lefut a megszakitáskezelő program. Ennek utolsó utasitása általában visszatölti az elmentett programállapotszót aktuálisnak, igy biztositja a visszatérést a megszakitott programba. Egyetlen megszakitási feltétel esetére ezt a folyamatot szemlélteti a 83. ábra.

A jelzett helyen programmegszakitási feltétel keletkezik, ez a vezérlő egységet az aktuális programállapotszóra irányitja (1). Megtörténik a vezérlő információ elmentése (2), és az uj vezérlő információ aktualizálása (3). Az itt előre kitöltött utasitáscim hatására a megszakitáskezelő alprogramot inditja a vezérlő egység (4). Ennek utolsó utasitása az elmentett programállapotszóhoz fordul (5) és azt visszatölti aktuálisnak (6). Az ebben elmentett utasitáscim a megszakitott feldolgozó program következő uta-



83. ábra. A programmegszakitás folyamata

sitására mutat (7), itt folytatódik annak végrehajtása. Vegyük észre, hogy a leirt folyamat nem változtatja meg az uj programállapotszót; a következő megszakitás kezeléséhez ismét felhasználható.

Az ilyen lefolyásu programmegszakitás csak akkor érvényes, ha egy adott időpillanatban csupán egy megszakitási feltétel keletkezett. Több megszakitási feltétel együttes kezelése már némileg bonyolultabb feladat elé állitja a vezérlő egységet. Kérdés: melyik megszakitási feltétellel foglalkozzon, és mi legyen a többinek a sorsa? Erre ad megoldást a megszakitási feltételek sorbaállitása, a prioritás.

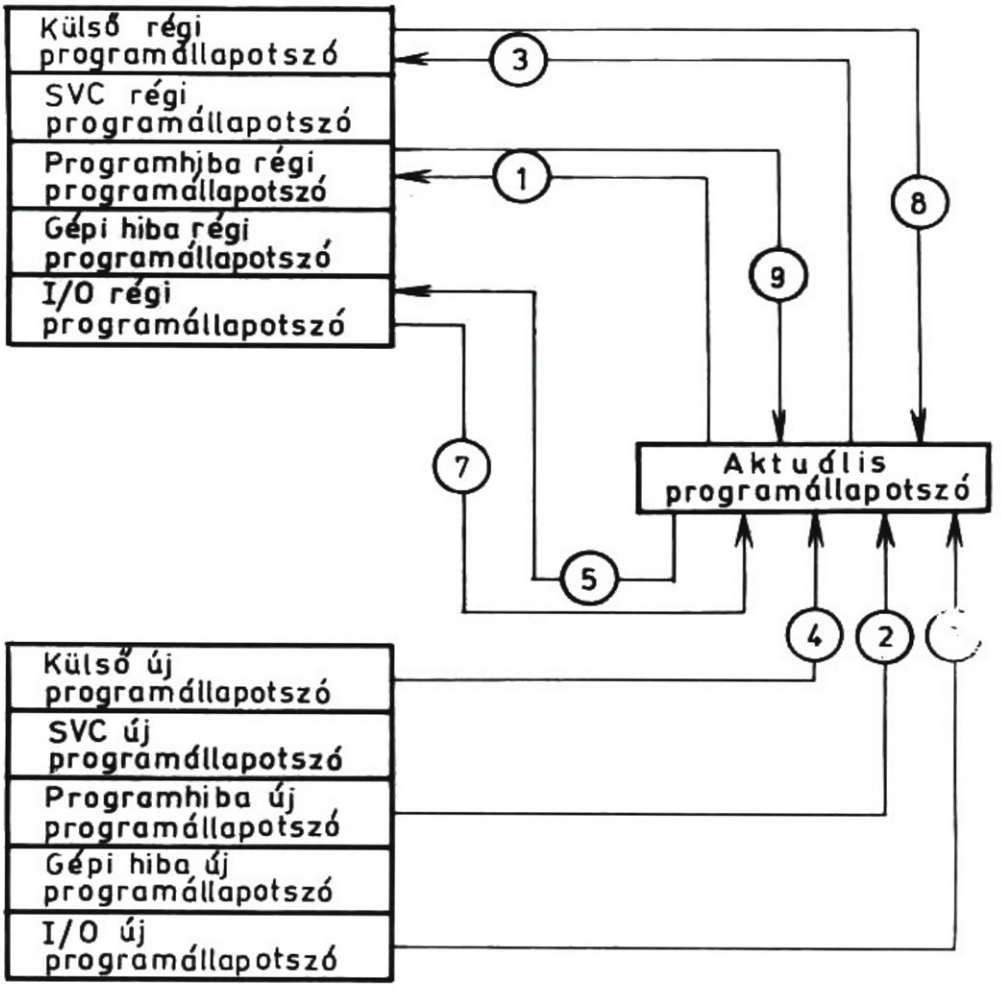
A számitógéprendszerben legfeljebb öt különböző forrása lehet a programmegszakitásoknak: géphiba, programhiba, SVC (Supervisor Call = Supervisor-hivás), külső feltétel és input/output feltétel. Ezek általában egyidőben is felléphetnek, ezért mindegyikük azonnali programállapotszó-cserét szeretne végezni. A megszakitáskérések kiszolgálásánál elsőbbségi sorrendet állapitottak meg az egyes feltételek között. Ezt nevezzük a megszakitások prioritásos rendszerének. A prioritási sorrend csökkenő prioritás szerint az alábbi:

1. géphiba
2. programhiba vagy Supervisor-hivás*
3. külső feltétel
4. input/output feltétel.

* Egymást kölcsönösen kizárják.

A prioritási sorrend azt jelenti, hogy az egyes megszakitási feltételek ennek értelmében vezetnek programállapotszó-cseréhez. A géphibának ezenkivül abszolut prioritása van; ha nincs maszkolva, ez mindig azonnal megszakitást okoz. Az elmondottakat szemlélteti a 84. ábra is. Itt feltételeztük, hogy három megszakitási feltétel futott be egyszerre.

A megszakitáskérések forrásai: programhiba, külső feltétel és input/output feltétel. Az adott prioritási sorrend értelmében először az aktuális programállapotszót tároljuk a programhiba rekeszébe (1), majd ennek helyére kerül aktuálisként a programhibakezelés uj programállapotszava (2). Ez azonban csak egy pillanatig marad aktuális; valójában el sem indul a programhibát kezelő alprogram futása, máris tároljuk ezt a programállapotszót a külső feltételek rekeszébe (3), és cseréljük a külső uj programállapotszóra (4). Ez ismét nem a végleges helyzet: ezt is tárolnunk kell az input/output feltételek rekeszében (5), és aktualizáljuk az input/output uj programállapotszót. Mivel a legalacsonyabb prioritási szintre jutottunk, további csere nem lehetséges, (6) hatására elindul az input/output feltételt kezelő alprogram. Ha ez véget ért, (7) alatt visszatölti a külső feltételek programállapotszavát, a külső feltételeket kezelő alprogram - lefutása után - (8) alatt visszatölti a programhibát kezelő alprogram programállapotszavát, végül pedig a programhibát kezelő alprogram visszatölti az eredetileg megszakitott program programállapotszavát (9). Ezzel térünk vissza a kiindulási feldolgozó programba.



84. ábra. Több megszakitási feltétel együttes kezelése

Ahhoz, hogy a leirt folyamat ilyen lefutásu legyen, szükséges, hogy minden uj programállapotszó maszkolja a vele azonos vagy nála nagyobb prioritásu megszakitási feltételeket. Ha ez teljesül, akkor a megszakitások feldolgozása éppen ellentétes sorrendben zajlik le, mint a programállapotszó-csere. Ez érthető, ha meggondoljuk, hogy az input/output megszakitások kiszolgálása a legsürgősebb, a többi feltétel "várhat egy kicsit". Hasonlóképpen nem okoz gondot az, hogy a 2. prioritási szinten két feltétel áll: programhiba és Supervisor-hivás. Ezek ugyanis sohasem fordulhatnak elő egyszerre.

Most már látjuk, mi volt az értelme annak, hogy minden egyes megszakitás-forrás számára külön régi és uj programállapotszó-rekeszt tartottunk fenn az operativ tár állandóan felosztott területén. Igy minden megszakitáskezelő alprogram tudhatja, hogy melyik fix cimről kell visszatöltenie az elmentett programállapotszót. Ha egy programállapotszó nem maszkolja a saját és a magasabb prioritási szintek megszakitásait, akkor végtelen programállapotszó-csere ciklusba kerülhetünk, és mindig felülirjuk az elmentett információt.

3.4.10 PÉLDA EGYSZERŰ GÉPI KÓDÚ PROGRAM FUTÁSÁRA. A VEZÉRLŐ EGYSÉG CIKLUSA

A gépi kódu program futását egyetlen utasitásból álló programon mutatjuk be. Elegendő egy utasitást vizsgálnunk, mert minden utasitás végrehajtása jellegét tekintve azonosan történik. A 85. ábrán egy 6 byte hosszuságu utasitás végrehajtását rajzoltunk fel. Legyen ez például két decimális szám összeadása. Kövessük nyomon, milyen lépésekből építi fel a vezérlő egység a számunkra elemi utasitás végrehajtását!

(1) Első lépésként betölti az utasitászámláló regiszterbe a végrehajtandó utasitás cimét. Esetünkben ez 100. Az utasitásszámláló azonos a programállapotszó 5-7. byte-jának tartalmával.

(2) Az utasitásszámláló tartalma átkerül az operativ tár cimregiszterébe, ezzel megcimeztük az utasitást.

(3) Ezt követi az utasitás elérése. Az elérés történhet például 2 byte-onként, de nagyobb gépeknél akár 8 byte-onként is egy lépésben.

(4) Az utasitást kiolvassuk a tárból, először a tár adatregiszterébe. Ennek mérete is géptipustól függ; az egyszerüség kedvéért tételezzük fel, hogy egy utasitás mindig elfér az adatregiszterben.

(5) A tárból kiolvasott utasitás a vezérlő egység utasitásregiszterébe kerül. Ezzel befejeződött az első szakasz: az utasitás lehivása.

(6) A vezérlő egység ezután azonnal beállitja az utasitásszámláló tartalmát a következő utasitás elejére. Erre azért van szükség, hogy programmegszakitás esetén kéznél legyen az elmentendő cim. Esetünkben egy 6 byte-os utasitás hatására az uj cim 106 lesz.

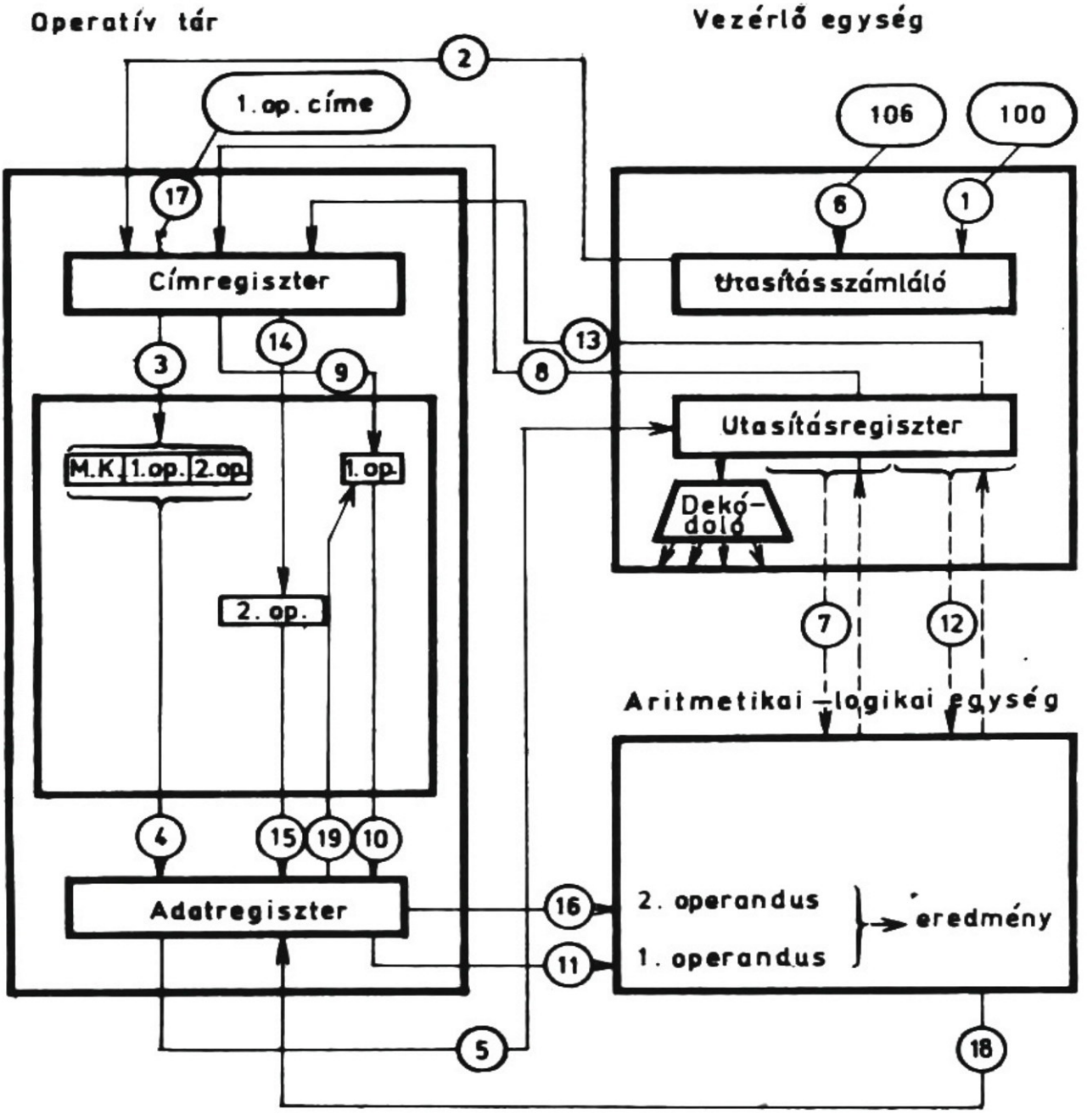
(7) A vezérlő egység kiszámittatja az 1. operandus cimét; a bázis- és relativ cimből abszolut tárcim lesz.

(8) Ennek felhasználásával megcimzi az 1. operandust, vagyis annak cimét beviszi a tár cimregiszterébe.

(9) Most következik az 1. operandus elérése.

(10) Az 1. operandus bekerül a tár adatregiszterébe.

(11) A tár átadja a kiolvasott 1. operandust az aritmetikai-logikai egységnek müveletvégzés céljából.



85. ábra. A vezérlő egység ciklusa

(12) A vezérlő egység kiszámittatja a 2. operandus cimét.

(13) Megcimzi a 2. operandust.

(14) Ezt követi a 2. operandus elérése a tárban.

(15) Kiolvassa a 2. operandust az adatregiszterbe.

(16) A tár átadja a kiolvasott 2. operandust az aritmetikai-logikai egységnek müveletvégzés céljából.

(17) Most már indítható az aritmetikai-logikai egység az utasitás müveleti kódja által meghatározott müvelettel. Ennek elvégzése után az eredményt - a 2 cimes gép sajátosságainak megfelelően - az 1. operandus cimére kell tenni. Ujból megcimezzük tehát az 1. operandust.

(18) Az aritmetikai-logikai egység átadja a kiszámitott eredményt a tár adatregiszterébe beirásra.

(19) Az adatregiszterből megtörténik az eredmény beirása az 1. operandus cimére. Ezzel véget ért a folyamat második része: az utasitásvégrehajtás.

Minthogy a következő utasitás cimét már beállitottuk az utasitásszámlálóban, következhetne a soronkövetkező utasitás lehivása és végrehajtása. Előzőleg azonban megvizsgálja a vezérlő egység, hogy van-e közben beérkezett megszakitási feltétel, amely oda vezethet, hogy a program nem folytatható a 106-os cimen. Ha megszakitásra van szükség, akkor a már megismert programállapotszócsere játszódik le, és valamelyik megszakitáskezelő alprogram indul el. Ha nincs megszakitáskérés, akkor a 106-os cimen folytatódik a program.

Az itt leirt folyamatot nevezzük a vezérlő egység ciklusának, amelynek két fő része: utasitás-lehivás és utasitásvégrehajtás.

3.5 A CSATORNARENDSZER

3.5.1 A CSATORNARENDSZER FELÉPÍTÉSE. ALCSATORNÁK

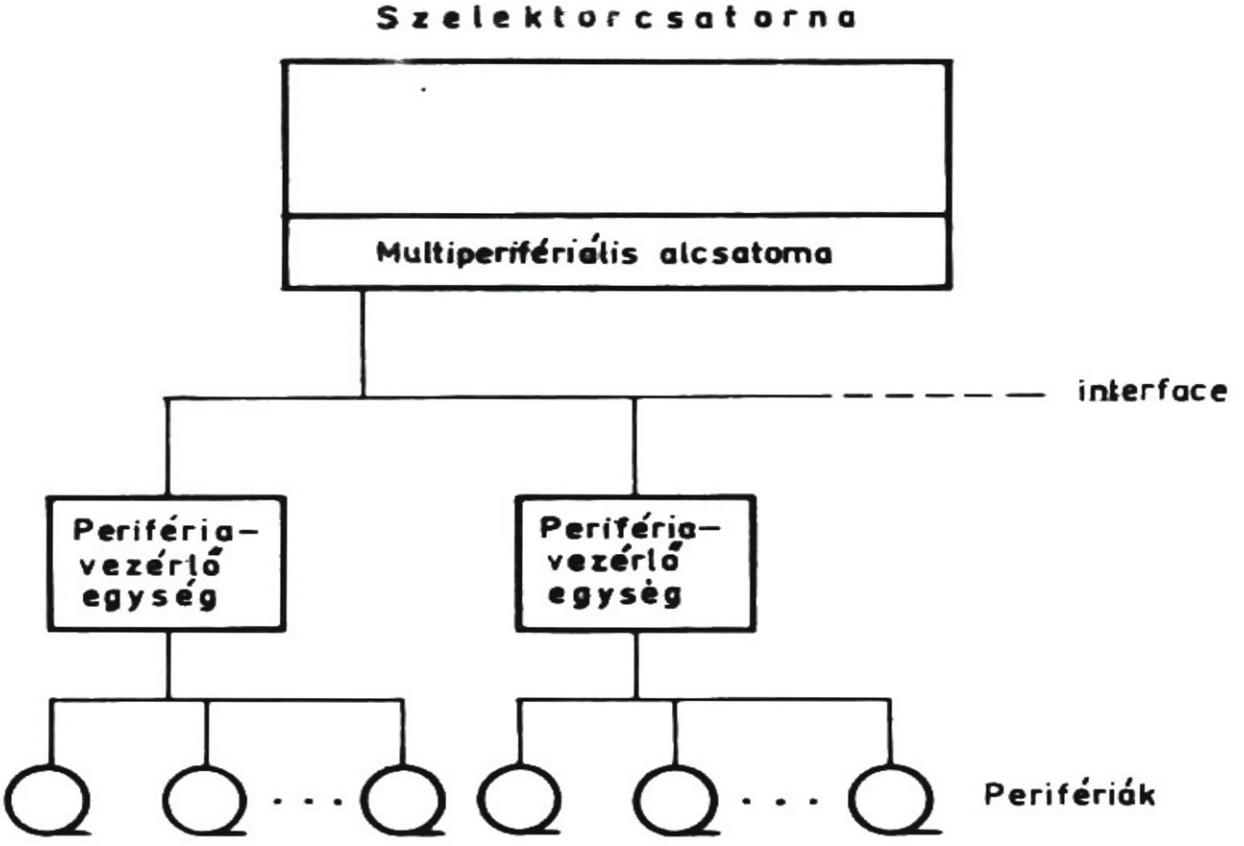
A mai korszerü számitógépek nagy müveleti sebessége és a perifériális egységek viszonylagos lassusága közötti feszültség nélkülözhetetlenné tette azt, hogy a rendszerben folyó input/output müveletek önálló vezérlést kapjanak. Igy tehermentesitettük a központi számitógépet az input/output müveletek felügyeletétől, és áthidaltuk a sokféle periféria különböző müveleti sebességéből fakadó problémákat. A csatornarendszer bonyolultsági foka hasonló a központi számitógépéhez, müködése lehet mikroprogramozott vagy huzalozott, és maguk a csatornák lehetnek a központi számitógéppel egybeépitettek vagy különállók. Fontos megfigyelnünk azt, hogy a csatornák elsősorban nem adatátevesztő egységek, hanem olyan vezérlő egységek, amelyek kizárólag input/output müveletek lebonyolitásával foglalkoznak.

A követelményekhez igazodóan kétféle csatornatipus terjedt el: a szelektorcsatorna és a multiplex csatorna. A szelektorcsatorna általában gyorsabb a multiplex csatornánál, és ezért ajánlatos ehhez csatlakoztatni a gyors perifériákat. Egyszerre mindig egy perifériával foglalkozik. A multiplex csatorna szolgálja ki a lassu perifériákat, és egyszerre több perifériával dolgozik. Ezek a csatlakoztatási módok nem kötelező előirások, de figyelembe vételük mellett maximális lesz a számitógéprendszer kihasználtsága.

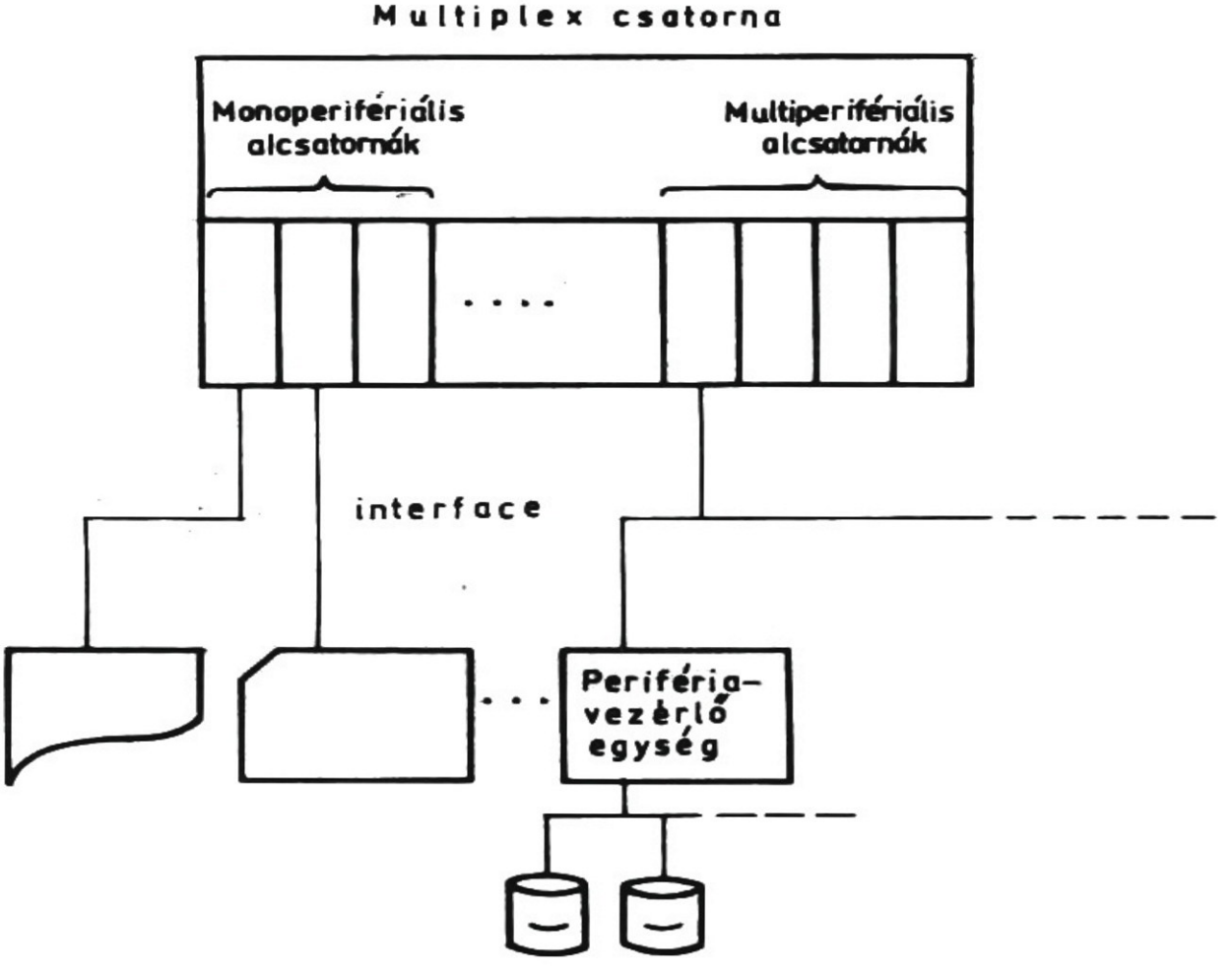
Az egyes csatornák vezérlési funkcióikat ugynevezett alcsatornáik segitségével valósitják meg. Alcsatornán mindig meghatározott nagyságu tárolóterületet értünk. Ez lehet magában a csatornában, vagy lehet az operativ tár erre kijelölt helyén. Az R20 számitógépnél az alcsatornák az operativ tárnak egy a programból nem cimezhető területén vannak. Az alcsatornák tartalma mindaz a vezérlési információ, amely alapján a csatorna végrehajtja az input/output müveleteket. Ezenkivül lehet adatokat is tárolni itt az átvitel során.

A szelektorcsatornáknak egyetlen alcsatornájuk van, a multiplex csatornának pedig a gép kiépitettségétől függően esetleg több száz is lehet. Egy számitógépen biztosan van egy multiplex csatorna és egy vagy több szelektorcsatorna. Utóbbiak maximális száma hat.

Kétféle alcsatornát különböztetünk meg: monoperifériális és multiperifériális alcsatornát. A monoperifériális alcsatornákra csatlakoznak azok a perifériális egységek, amelyeknél minden perifériának saját külön perifériavezérlő egysége van (például sornyomtatók, konzolirógép stb.). A multiperifériális



86. ábra. Perifériák csatlakoztatása a szelektorcsatornához

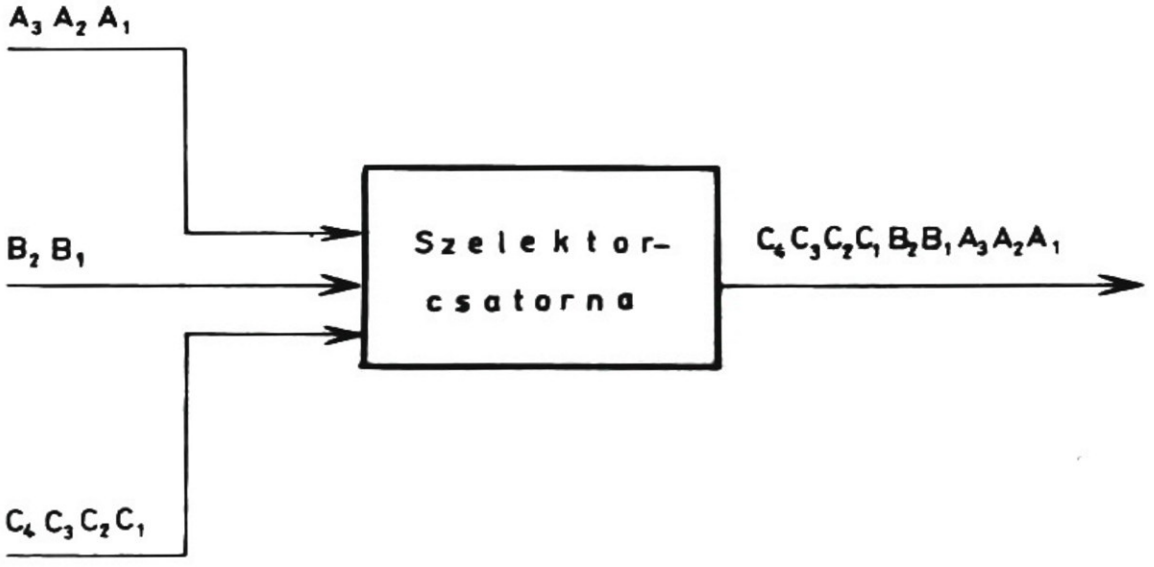


87. ábra. Perifériák csatlakoztatása a multiplex csatornához

alcsatornákról kapják a vezérlést azok a perifériák, amelyeknél több perifériának közös periféria-vezérlő egysége van: például mágnesszalag, mágneslemez egységek stb. A szelektorcsatornák alcsatornája multiperiférialis, a multiplex csatornának pedig vannak monoperiférialis és multiperiférialis alcsatornái. Mindezt a 86. és 87. ábrák szemléltetik.
A valóságban nem szoktak a multiplex csatornára mágneslemezegységeket csatlakoztatni; itt csak az elvi lehetőséget szemléltettük.

3.5.2 A SZELEKTORCSATORNA MŰKÖDÉSE

A szelektorcsatorna csak ugynevezett monopol üzemmódban tud dolgozni. Ez azt jelenti, hogy ha valamelyik perifériával adatátvitelt kezdeményezett, akkor az a periféria "kisajátitja", monopolizálja magának a csatornát, a többi periféria várni kényszerül. Ez az üzemmód alkalmas a mágneses perifériák müködtetésére, mert ezeknél a nagy adatsürüség és átviteli sebesség miatt a megkezdett input/output müveletet nem lehet adatvesztés nélkül megszakitani. A monopol üzemmódot mutatja a 88. ábra.

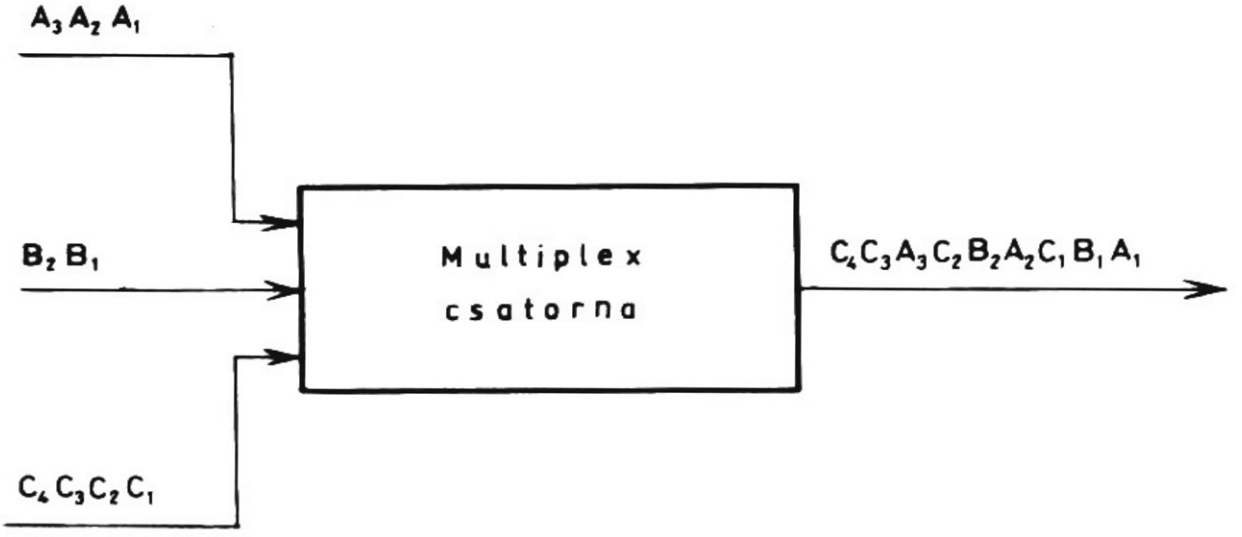


88. ábra. A szelektorcsatorna adatátvitele (monopol üzemmód)

Itt A, B és C egy-egy perifériát jelöl, A_1, B_1 stb. pedig az A, B stb. perifériáról érkező byte-okat jelenti. Látjuk, hogy a három periféria csak egymás után kapja meg a csatornát: az A perifériával megkezdett müvelet végén indítható csak a B, majd ezután a C periféria. A csatornában a különböző helyekről származó byte-ok nem keverednek egymással. Ezért lehetséges az, hogy egyetlen multiperiférialis alcsatornára sok perifériát kapcsoljunk: egy periféria adatátvitelt vezérlő információja (az alcsatorna tartalma) ugyanis a perifériamüvelet befejezése után érvényét veszti! Az input/output müvelet alatt pedig minden egyes byte átvitele után aktualizálja az átvitelt vezérlő információt; ennek végső állapota jellemző a müvelet lefolyására. E körülményeket azután a szelektorcsatorna input/output programmegszakitás révén a feldolgozó program tudomására hozza.

3.5.3 A MULTIPLEX CSATORNA MŰKÖDÉSE ÜZEMMÓDOK

A multiplex csatorna egyaránt tud monopol és ugynevezett multiplex üzemmódban dolgozni. A monopol üzemmódra ugyanazok a szabályok vonatkoznak, mint szelektorcsatorna esetében. A multiplex üzemmód azt jelenti, hogy a csatorna, miután valamelyik perifériával adatátviteli müveletet kezdeményezett, 1 byte átvitele után lekapcsolódik arról a perifériáról, és a következő perifériával folytatja az átvitelt. Igy a csatorna mintegy "végigjárja" azokat a perifériáit, amelyekkel már elinditott egy input/output müveletet, és sorban mindegyikkel csak 1 byte átviteléig foglalkozik. Ezt a multiplex üzemmódot mutatja a 89. ábra.



89. ábra. A multiplex csatorna adatátvitele multiplex üzemmódban

Látjuk, hogy az A, B és C perifériáról származó byte-ok keverednek a csatornán történő áthaladásuk során. Emiatt meg kell oldani e byte-ok szétválogatását és megfelelő helyre továbbitását. Mindezt az egyes alcsatornákban levő vezérlő információ végzi. Ezért nem lehetséges az, hogy monoperiférialis alcsatornára több perifériát kapcsoljunk: a csatorna ugyanis minden byte átvitele után aktualizálja az átvitelt vezérlő információt, igy az csak a legutolsó byte átvitele után veszti érvényét! Egy-egy periféria átviteli müvelete tehát időben szét van darabolva. Emiatt tünik ugy, mintha a multiplex csatorna egyidőben több perifériával dolgozna. Szorosabb értelemben azonban mindig csak egy perifériával foglalkozik. Ha valamelyikkel befejezte az átvitelt, input/output megszakitási feltételt generál, és értesiti a fennálló feltételekről, körülményekről a feldolgozó programot.

3.5.4 AZ INPUT/OUTPUT MŰVELETET VEZÉRLŐ INFORMÁCIÓ SZERKEZETE. A CSATORNACÍMSZÓ, CSATORNAPARANCSSZÓ, CSATORNAÁLLAPOTSZÓ FELÉPÍTÉSE

Vizsgáljuk meg most részletesen is, mit tartalmaz az input/output müveletet vezérlő információ! Erre a célra a számitógép 3 kötött formátumu szót használ. A csatornacimszó az operativ tár 72-es cimü rekeszében található; pontosabban: az elinditott csatorna mindig ennek a rekesznek a tartalmát tekinti csatornacimszónak. A csatornaparancsszavak duplaszó hosszuságuak, és az operativ tárban bár-

hol lehetnek. Ezek alkotják a csatornaprogramot, egy a gépi kódu programhoz hasonló olyan programot. amely speciálisan az input/output müveleteket irányitja. A csatornaállapotszó az operativ tár 64-es cimü rekeszében található, hossza 8 byte. Itt tárolja a csatorna azokat a körülményeket, amelyek az input/output művelet lefolyására jellemzőek, és amelyeket a feldolgozóprogramnak a tudomására akar hozni.

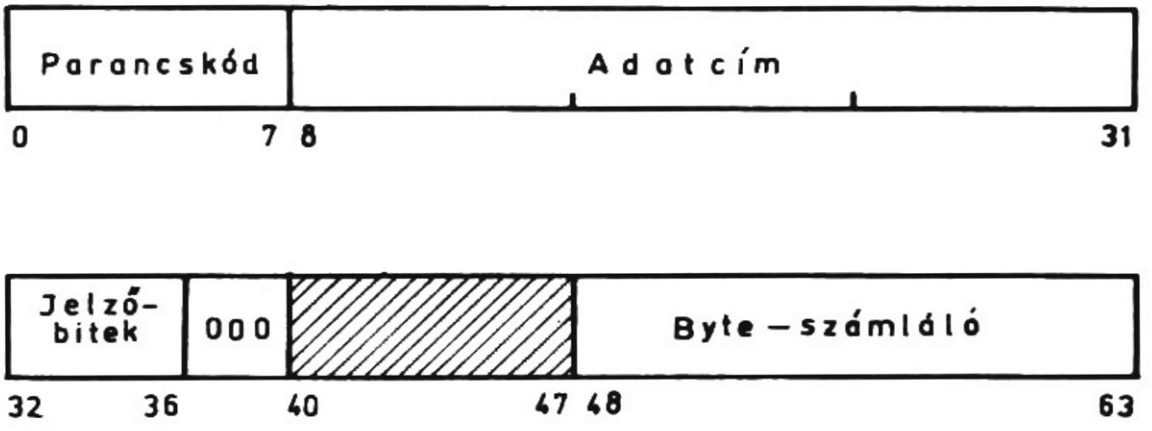
a) A csatornacimszó felépitését a 90. ábra mutatja.

90. ábra. A csatornacimszó felépitése

A 0-3. biteken található a csatornaprogram tárvédelmi kulcsának numerikus része. Ezt használja a vezérlő egység akkor, amikor a csatorna az operativ tárhoz fordul, és a tárhivatkozás jogosultságát ellenőrzi. A kulcs funkciója azonos a programállapotszóban található tárvédelmi kulccsal.

A 8-31. biteken az a tárcim található, amelyen a csatornaprogram kezdődik. Ez az általunk megirt első csatornaparancsszó cime.

b) A csatornaparancsszó felépitését a 91. ábra mutatja.



91. ábra. A csatornaparancsszó felépitése

A 0-7. bitek tartalmazzák a parancskódot. Ez hasonló - funkcióját tekintve - a gépi utasitások müveleti kódjához: megadja, hogy milyen perifériamüveletet kell végrehajtani, például irás, olvasás, stb. Ez a kód kerül ki az interface-en a perifériavezérlő egységhez, amely értelmezi azt.

Az adatcim annak az operativ tárbeli területnek a kezdőcime, amelyet az adatátvitelben szerepeltetni akarunk. Mágnesszalagos hátraolvasásnál ez a tárterület utolsó (legnagyobb) cime.

Az öt jelzőbit jelentése a következő:

32 - adatláncolás: a csatornaparancsszó végrehajtása után azonnal át kell.térni a következő csatornaparancsszóra, de az előző (mostani) parancskóddal;

33 - parancsláncolás; a csatornaparancsszó végrehajtása után azonnal át kell térni a következő csatornaparancsszóra, de annak saját parancskódja lesz érvényben;

34 - hosszhibaelnyomás; nem kell megszakitási feltételt generálni, ha a perifériával nem az adott számu byte átvitele történt meg;

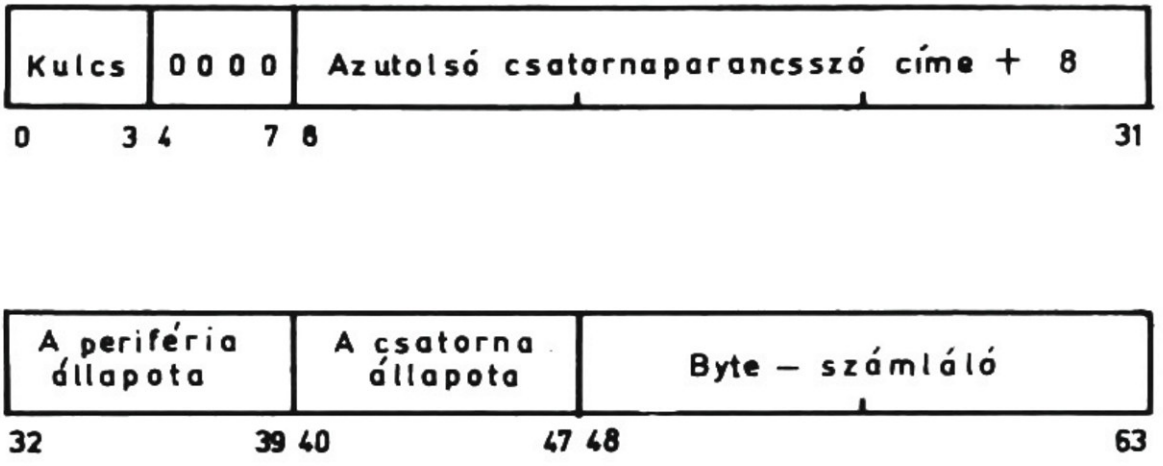
35 - átvitelelnyomás; input müveleteknél letiltja az adatátvitelt;

36 - programvezérelt megszakitás; ennek hatására a csatorna megszakitási feltételt generál.

A jelzőbitek akkor érvényesek, ha értékük "1", a megfelelő funkció egyébként kikapcsoltnak tekintendő.

A 48-63. bitek azt a bináris számot tárolják, amely megmondja, hogy hány byte-ot kell átvinni a perifériáról vagy a perifériára. Az átvitel során minden egyes byte után a csatorna 1-et levon a byte-számláló tartalmából, igy normális esetben a müvelet végén értéke 0.

c) A csatornaállapotszó felépitését a 92. ábra mutatja.



92. ábra. A csatornaállapotszó felépitése

A 0-3. bitek a csatornaprogram tárvédelmi-kulcsának numerikus részét tárolják.

A 8-31. biteken olyan tárcim áll, amely 8-cal nagyobb, mint az utoljára végrehajtott csatornaparancsszó cime. Normális esetben ez a csatornaprogram végét jelzi, hibás esetben pedig ebből lehet kiszámitani azt a cimet, ahol a csatornaprogram végrehajtása elakadt.

A következő 2 byte-ban tárolja a csatorna az input/output rendszer állapotát. A bitek jelentése a következő:

32 - "Figyelem" - a periféria valamilyen különleges eseményről akarja értesiteni a programot;

33 - Állapotmódositó - a perifériavezérlő egység foglalt, vagy a csatornaprogram végrehajtási sorrendjén változtatni kell;

34 - Perifériavezérlő egység kész - a perifériavezérlő egység befejezte az input/output müveletet;

35 - Periféria foglalt;

36 - Csatorna kész - nincs szükség tovább a csatorna részvételére az input/output müveletben;

37 - Periféria kész - a periféria befejezte az input/output müveletet, felszabadult;

38 - Perifériahiba;

39 - Különleges esemény a periférián, például a mágnesszalag olvasásakor szakaszjelet észleltünk stb.

40 - Programvezérelt megszakitás;

41 - Hosszhiba;

42 - Hiba a csatornaprogramban;

43 - Tárvédelem megsértése;

44 - Adathiba a csatornában;

45 - Vezérlési hiba a csatornában;

46 - Vezérlési hiba az interface-en;

47 - Láncolási hiba.

Ez az a 2 byte-nyi információ, amelyből nagy vonalakban következtetni lehet az input/output müvelet lefolyására. Ezt vizsgálják meg az input/output megszakitást kezelő alprogramok is, ennek alapján teszik meg a szükséges intézkedéseket.

A 48-63. bitek azt a byte-számláló értéket tartalmazzák, amely a csatornaállapotszó kitöltése pillanatában érvényben volt. Normális esetben ez 0, ekkor az összes byte átvitele megtörtént.

Felhivjuk a figyelmet arra a körülményre, hogy az operativ tárban található csatornaprogram - a csatornaparancsszavak láncolata - az input/output müvelet végrehajtása után is változatlan marad. A byte-számláló értékének eggyel csökkentése, az adatcim aktualizálása, általában az átvitelt vezérlő információ aktualizálása csak a csatorna megfelelő alcsatornájában, illetve regisztereiben történik meg. Hasonlóképpen, a csatorna nem tölti ki minden egyes byte után a csatornaállapotszót, hanem csak a teljes input/output müvelet lefolyása után. Ez vonatkozik a multiplex csatornára is multiplex üzemmódban! Ebből érezhetjük, miért olyan sürgős az input/output megszakitási feltétel kezelése: valamennyi csatorna ugyanazt a csatornaállapotszót és csatornacimszót használja, ezek gyors felszabaditása tehát igen lényeges.

3.5.5 AZ INPUT/OUTPUT MŰVELET LEFOLYÁSA GÉPI SZINTEN

Az előzőekben megismertük annak a vezérlő információnak a szerkezetét, amely az input/output müveleteket irányitja. Most nézzük meg, hogyan zajlik le időben az adatátvitel, hogyan használja fel a csatornarendszer az átvitelt vezérlő információt!

Minden input/output müveletet a központi számitógép indit el a gépi kódu program alapján. Erre a célra a számitógépnek négy gépi utasitása szolgál. Ezek:

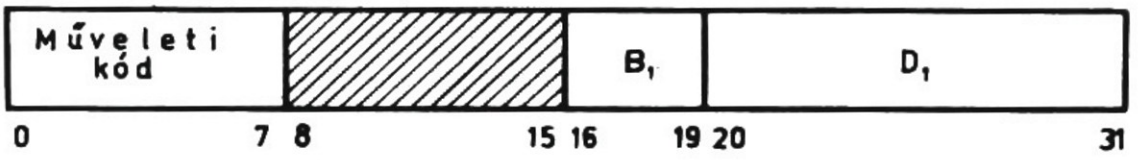
SIO - Start Input/Output, a müvelet inditására,

TIO - Test Input/Output, állapot lekérdezésre,

HIO - Halt Input/Output, a müvelet leállitására,

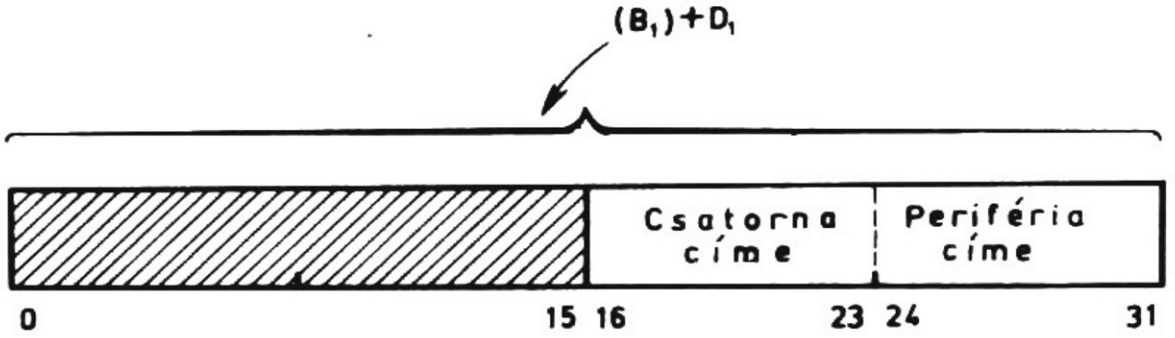
TCH - Test Channel, a csatornaállapot lekérdezésére.

Ezek valamennyien SI tipusuak, felépitésük a 93. ábrán látható.



93. ábra. Az input/output utasitások felépitése

Ahhoz hasonlóan, ahogy a közönséges gépi utasitások megadják, milyen müveletet kell végezni és melyik operandusokkal, az input/output utasitások is megadják, hogy a központi számitógép milyen jellegü müveletet végezzen (inditás, leállitás, állapotlekérdezés) és melyik perifériával. Természetesen, mivel a müveletek vezérlése már a csatornarendszer feladata, ezek az utasitások részletesebb információt nem tartalmaznak. A periféria kiválasztása annak megcimzése révén történik. A $(B_1) + D_1$ számot a gép nem operanduscimként, hanem input/output rendszer cimeként értelmezi, a 94. ábra szerint.



94. ábra. Az input/output rendszer cimzése

Az input/output utasitás végrehajtása tehát véget ér a megfelelő periféria kiválasztásával. A kiválasztás kimenetelét (szabad, foglalt stb. periféria) a feltételkód tartalmazza. A továbbiakban csak a SIO utasitással foglalkozunk.

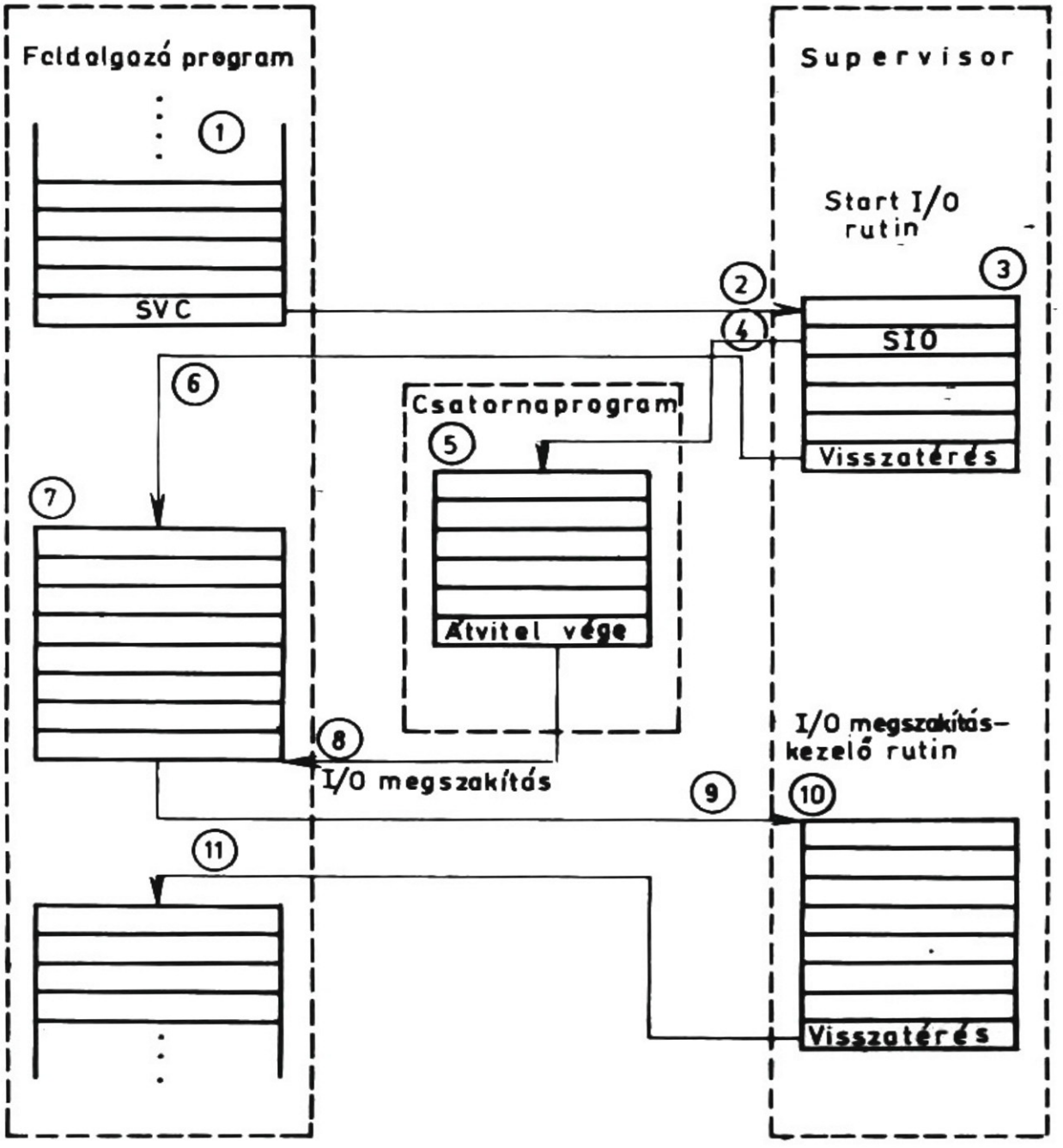
Az input/output müvelet lebonyolitása ezután már a csatornarendszer és a csatolt perifériák feladata. A csatorna először megkeresi a csatornaprogramot, azaz kiolvassa a csatornacimszót, majd kiolvassa az első csatornaállapotszót és hozzáfog annak végrehajtásához. Ezt követi a második stb. csatornaparancsszó kiolvasása és végrehajtása egészen addig, mig a csatornaprogram véget nem ér. A csatornaprogram vége lehet például egy olyan csatornaparancsszó, amelyben nincs láncolás stb. Végül a csatorna kitölti a csatornaállapotszót és megszakitási feltételt generál, hogy az input/output müvelet lefolyásának körülményeit a feldolgozó program tudomására hozza. A csatornaprogram és a feldolgozó program időben egyszerre fut; ezzel javitjuk a számitógéprendszer kihasználtságát. Az input/output müveletek lefolyását szemlélteti a 95. ábra.

Mivel az input/output utasitások privilegizáltak, ezeket a gép csak Supervisor állapotban (lásd 3.4.8 pont) hajtja végre. Az ábrán feltételeztük, hogy a feldolgozó program probléma-módban fut, ennélfogva az input/output müvelet inditása előtt át kell kerülnie Supervisor állapotba. Erre szolgál az SVC utasitás, amely programállapotszó cserét eredményez. Az SVC uj programállapotszóban már eleve Supervisor módot állitottunk be. Ha a számitógép operációs rendszer vezérlete alatt dolgozik, akkor az input/output müveleteket a Supervisor inditja, egyébként tetszőleges, általunk megirt programból is indithatók.

Kövessük most végig az ábrát az események sorrendjében!

(1) A feldolgozó program futása során olyan helyre értünk, ahol egy input/output müvelet végrehajtása válik esedékessé. Átvisszük tehát a központi számitógépet Supervisor állapotba az SVC utasitással.

(2) Az SVC eredetü programmegszakitás programállapotszót cserél, és az uj programállapotszóban előkészitett utasitáscim a gépet az input/output müveleteket inditó programra (rutinra) irányitja.



95. ábra. Az input/output müveletek lefolyása

(3) Elindul a Start Input/Output rutin; ez a csatorna cimszóba betölti a megfelelő csatornaprogram kezdőcimét, majd kiadja a Start Input/Output utasitást. Ebben az utasitásban megcimeztük a csatornát és a perifériát, amellyel a müveletet el akarjuk végezni.

(4) A SIO végrehajtása után - feltéve, hogy sikerült kapcsolatot teremteni a perifériával az interface-en - a csatorna megkeresi a csatornaprogramot. Ezt a felhasználó előzőleg megirta, bevitte az operativ tárba, cimét pedig a Start I/O rutin betöltötte a csatornacimszóba. Ennek alapján tudja a csatorna azt megtalálni.

(5) Elindul a csatornaprogram végrehajtása. Ez állhat egy vagy több csatornaparancsszóból.

(6) A csatornaprogram inditása után a Start Input/Output rutin visszatér a megszakitott feldolgozó programba (visszaveszi annak elmentett programállapotszavát).

(7) Most már időben egyszerre fut a gépi kódu feldolgozó program és a csatornaprogram.

(8) Mivel a csatornaművelet befejeződését előre tudni nem lehet, az átvitel végén a csatorna megszakitási feltételt generál, és kitölti a csatornaállapotszót.

(9) Az input/output megszakitás ismét programállapotszót cserél, de az uj programállapotszóban levő utasitáscim most az input/ output megszakitást kezelő rutin elejére mutat. Ez a rutin lehet a Supervisor része, de operációs rendszer nélküli üzemeltetésnél lehet tetszőleges, általunk megirt rutin is.

(10) Elindul a megszakitáskezelő rutin; ez "letapogatja", kiértékeli a csatornaállapotszó tartalmát, és megteszi a szükséges intézkedéseket (például szöveges üzenetet küld a gépkezelőnek stb.).

(11) Végül következik a visszatérés a másodszor megszakitott feldolgozó programba:, az input/output megszakitáskezelő rutin visszaveszi az elmentett programállapotszót, és ennek vezérlete alatt folytatódik a feldolgozó program végrehajtása.

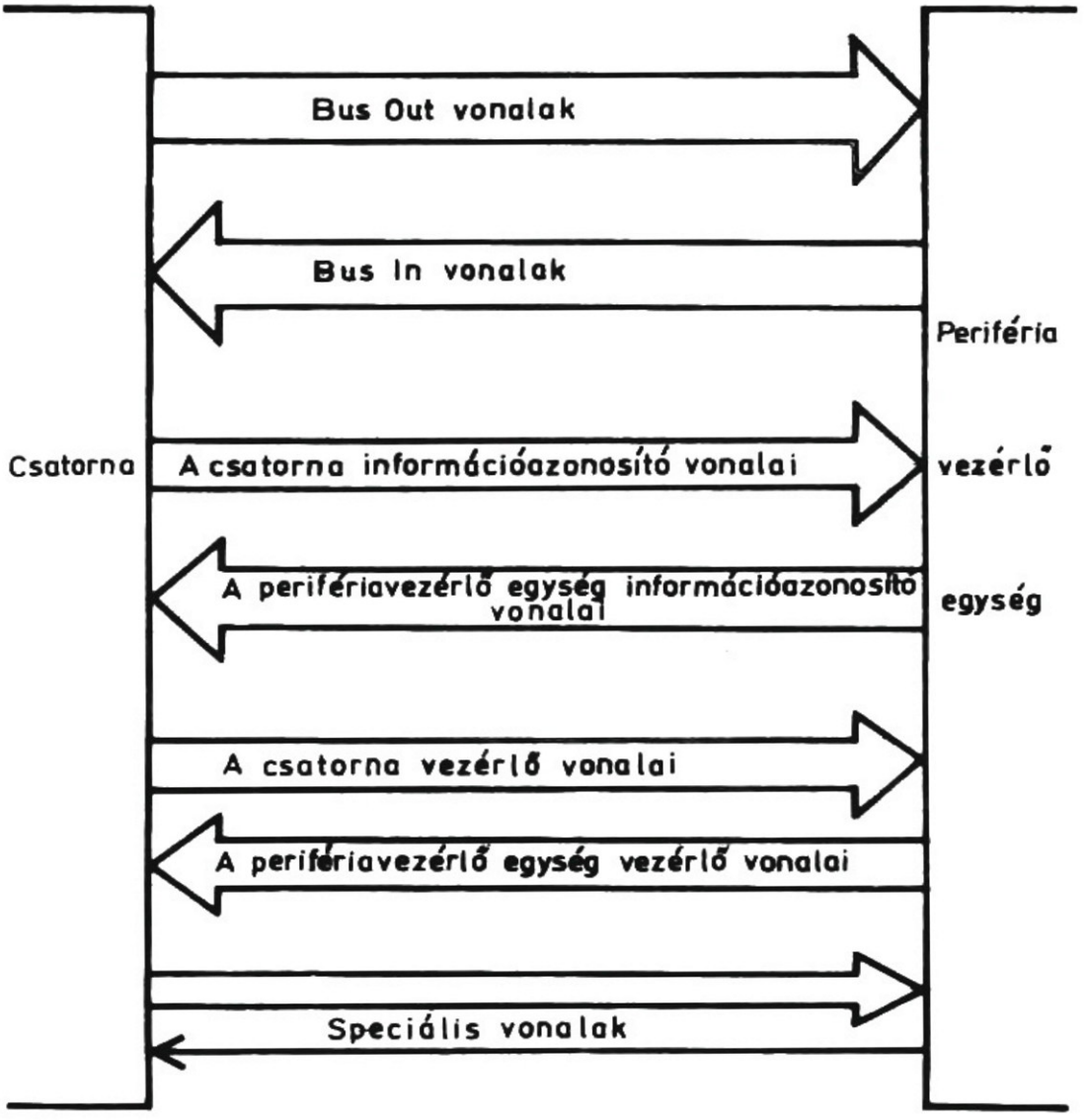
Összefoglalva: az operációs rendszer vezérlete alatt a feldolgozó programot kétszer szakitjuk meg az input/output művelet érdekében, operációs rendszer nélküli üzemeltetés esetén az első megszakitás elmaradhat, feltéve, hogy a feldolgozó program eleve Supervisor állapotban fut. Ilyenkor a SIO utasitás közvetlenül kiadható. Mivel azonban a feldolgozó programok döntő többsége operációs rendszer alatt fut, ezt az általánosabb esetet szemléltettük a példában is.

3.5.6 A CSATORNA-INTERFACE STRUKTÚRÁJA ÉS MŰKÖDÉSE. PÉLDA A PERIFÉRIÁVAL VALÓ KAPCSOLATTEREMTÉSRE SZABAD (ÜZEMKÉPES) ÉS FOGLALT PERIFÉRIA ESETÉBEN

Az input/output műveletek lefolyását már ismerjük; most ejtsünk néhány szót arról is, milyen utvonalon haladnak a számitógépbe beolvasott vagy a perifériákra kivitt adatok! Ez az utvonal a csatorna interface, tulajdonképpen az az egység, amely összekapcsolja a csatornákat az egyes perifériavezérlő egységekkel. Ugy képzelhetjük el, mint egy sok vezetékből álló kábelt, amelyen adatok, vezérlő jelek stb. haladnak mindkét irányban. Az interface felépitése egységes minden csatornára és perifériavezérlő egységre nézve, igy elvileg minden periféria csatlakoztatható minden csatornához. Az interface vonalait a 96. ábrán láthatjuk.

Az interface-vonalakat négy fő csoportba sorolhatjuk: információs vonalak, információazonositó vonalak, vezérlő vonalak, speciális vonalak. Érdekessége ezeknek, hogy minden vonalon csak egyirányban halad az információ; ez növeli a müködési biztonságot. Alább sorra vesszük a főbb vonalakat, és megállapitjuk azok funkcióját is.

a) Az információs vonalak arra szolgálnak, hogy a csatornából perifériacimet, parancskódot és adatokat, a perifériától pedig perifériacimet, állapotinformációt és adatokat továbbitsanak. Mind a csatornának, mind a perifériavezérlő egységnek 9 információs vonala van: 8 vonal a byte információs bitjei, 1 vonal pedig a byte paritásbitje számára. Egy byte tehát bitenként párhuzamosan, egy lépésben halad át az interface-en.



96. ábra. A csatorna-interface felépitése

b) Az információazonositó vonalak jelei arra szolgálnak, hogy egyértelmüvé tegyék, milyen jellegü információ halad az információs vonalakon, például cim, állapotbyte stb. A csatorna információazonositó vonalai a következők:

Address Out - a kiküldött információ perifériacim;

Command Out - a kiküldött információ parancskód;

Service Out - a kiküldött információ adatbyte.

A perifériavezérlő egység információazonositó vonalai:

Address In - a beküldött információ perifériacim;

Status In - a beküldött információ állapotbyte;

Service In - a beküldött információ adatbyte.

Ebből láthatjuk, hogy az információs vonalakon levő jeleknek csak akkor van értelmük, ha velük együtt valamelyik azonositó jel is érvényben van. Ugyanakkor két azonositó jel egyidőben nem állhat fenn egy irányban; például a kiküldött információ nem lehet cim is, meg parancskód is egyszerre.

c) A vezérlő vonalak funkciója az interface működésének időbeli irányitása. A csatorna vezérlő vonalai a következők:

Select Out - a perifériavezérlő egység kiválasztására;

Hold Out - a kiválasztást engedélyező jel;

Operational Out - a csatorna működő állapotát jelzi;

Suppress Out - a csatornaparancsszavak láncolására.

A perifériavezérlő egység vezérlő vonalai:

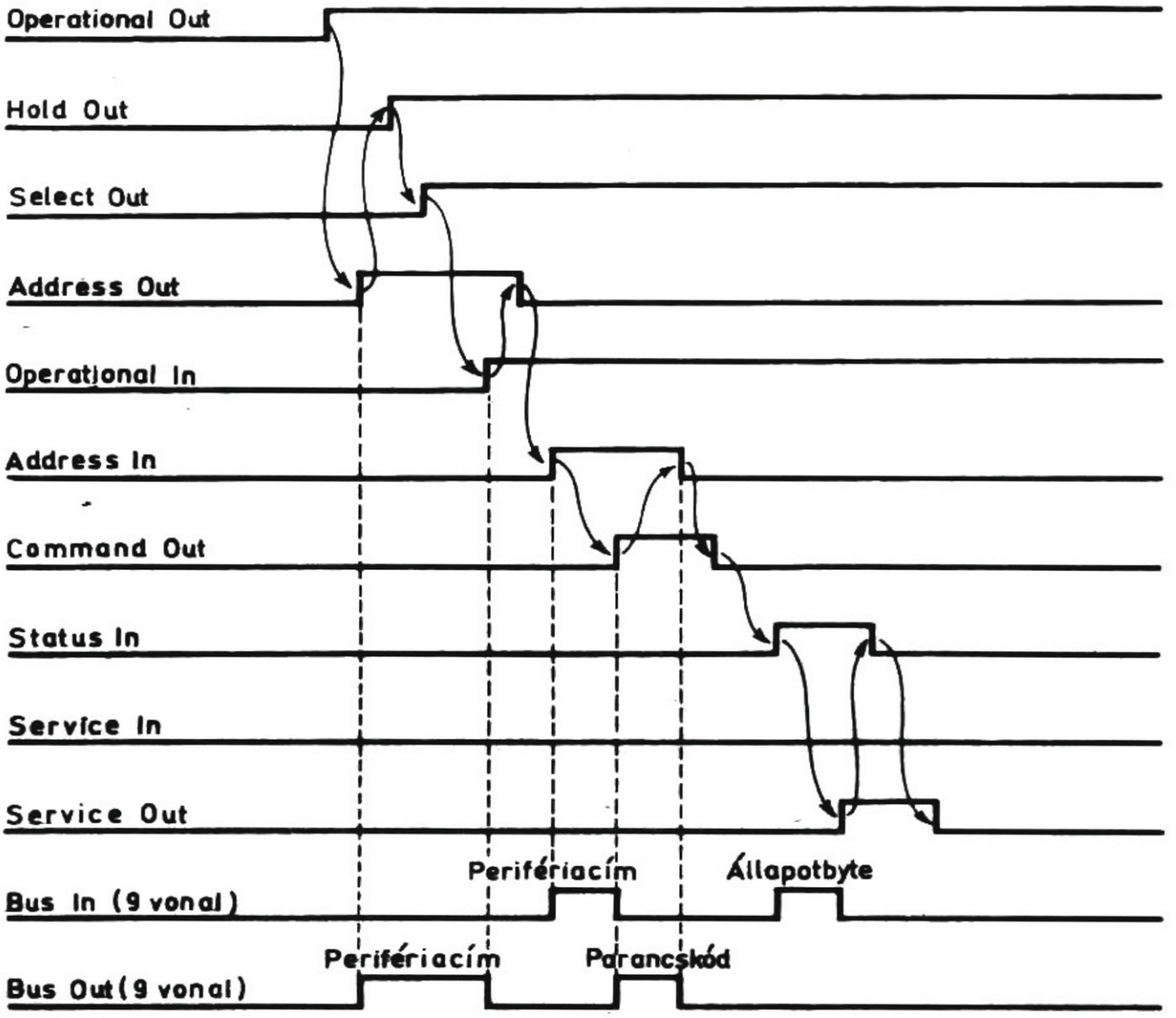
Select In - a Select Out vonal visszatérő ága;

Request In - a periféria szolgálatkérési vonala;

Operational In - a perifériavezérlő egység működő üzemmódját jelzi.

d) A speciális vonalak időmérésre és állapotváltásra szolgálnak; ezekkel itt nem foglalkozunk.

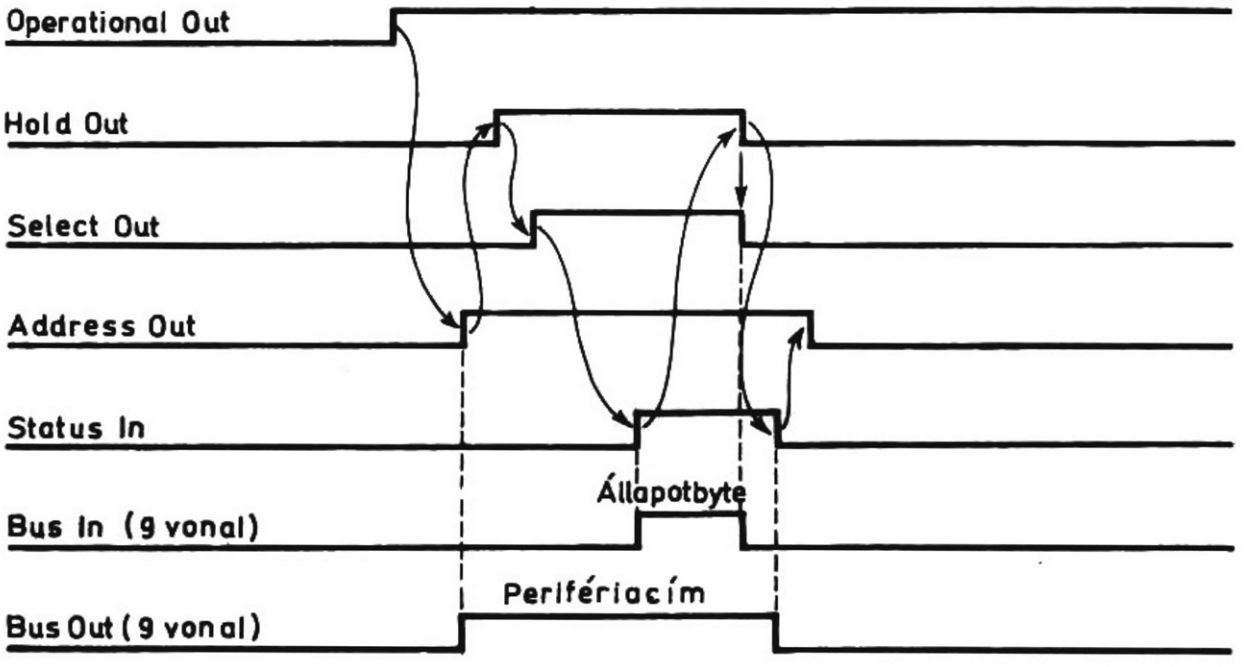
Az interface főbb vonalainak működését egy-egy példán mutatjuk be: az első a perifériával való sikeres kapcsolatteremtés, a második pedig a foglalt periféria kiválasztása. Mindkét eset az intput/output utasitás végrehajtása során fordulhat elő; ezek tehát az input/output művelet inditásakor érvényesek. Az ugynevezett kezdeti kiválasztás folyamatát a 97. ábra szemlélteti.



97. ábra. A kezdeti kiválasztás folyamata

A csatorna működő állapotát jelző Operational Out után a csatorna kiküldi a perifériacimet a Bus Out információs vezetékeken, és ennek azonositására felemeli Address Out jelét. A Hold Out kiválasztást engedélyező jel után megjelenik a Select Out, a perifériakiválasztó jel. Ha a csatornára kapcsolt perifériavezérlő egység és periféria felismeri a kiküldött cimet mint a saját cimét, akkor felemeli Operational In jelét; ezzel létrejött a logikai kapcsolat a csatorna és a megcimzett periféria közt. A perifériavezérlő egység visszaküldi a cimét a Bus In vezetékeken, az Address In azonositó jel kiséretében. Erre a csatorna kiküldi a Bus Out vezetékeken az éppen végrehajtandó csatornaparancs-szó parancskódját, és ezt a Command Out jellel azonositja. A parancskód értelmezése után a periféria-vezérlő egység beküldi állapotbyte-ját a Bus In vezetékeken, a Status In jel kiséretében. Ha ez az állapotbyte csupa zérust tartalmaz, akkor sikeres volt a kapcsolatteremtés; következhet az adatátvitel. A csatorna ezt a Service Out jel kiadásával inditja el. Ha az állapotbyte nemcsak zérus értékü bitekből áll, akkor az adatátvitel nem kezdődik el, mert a periféria "nincs olyan állapotban", hogy adatátvitelre alkalmas legyen (például foglalt).

Ha a perifériavezérlő egység foglalt, akkor a kiküldött cimre azonnal állapotbyte-tal válaszol, jelezve foglaltságát. Ezt a jelsorozatot a 98. ábra mutatja.



98. ábra. Foglalt perifériavezérlő egység kiválasztása

3.6 A KEZELŐPULT

3.6.1 A KEZELŐPULT FELÉPÍTÉSE

A számitógép kezelőpultja arra szolgál, hogy a rendszer működésébe való közvetlen beavatkozást a gépkezelő számára lehetővé tegye. Ebből következik, hogy a kezelőpultról a központi számitógépnek olyan részei (regiszterek, tárterületek, mikroprogramtár stb.) is elérhetők, amelyekhez a felhasználónak rendszerint semmilyen hozzáférési lehetősége nincs. A pultról történik a számitógéprendszer üzembe helyezése, a kezdeti programbetöltés is. A kezelőpultról egész gépi kódu programokat

futtathatunk le utasitásonként is, vagy akár mikroutasitásonként is, és minden egyes utasitás végrehajtása után megfigyelhetjük a gép állapotát, regisztereinek, tárterületeinek tartalmát. Olyan hibák megkeresésére, illetve kijavitására, amit hibakereső programok vagy tesztprogramok sem találnak, a pult igen hatásosan használható. Megjegyezzük, hogy az operációs rendszer vezérlete alatt üzemelő számitógép működtetése elsősorban nem a kezelőpultról, hanem a konzolirógépről történik; ez az operációs rendszer és a gépkezelő közötti kommunikáció eszköze. Sulyosabb hiba, gépleállás stb. esetén azonban végül mindig a pulthoz kell fordulnunk, hogy mélyebben beavatkozhassunk a hardware berendezések működésébe.

3.6.2 A KIJELZÉSEK ÉS A KEZELŐSZERVEK SZEREPE

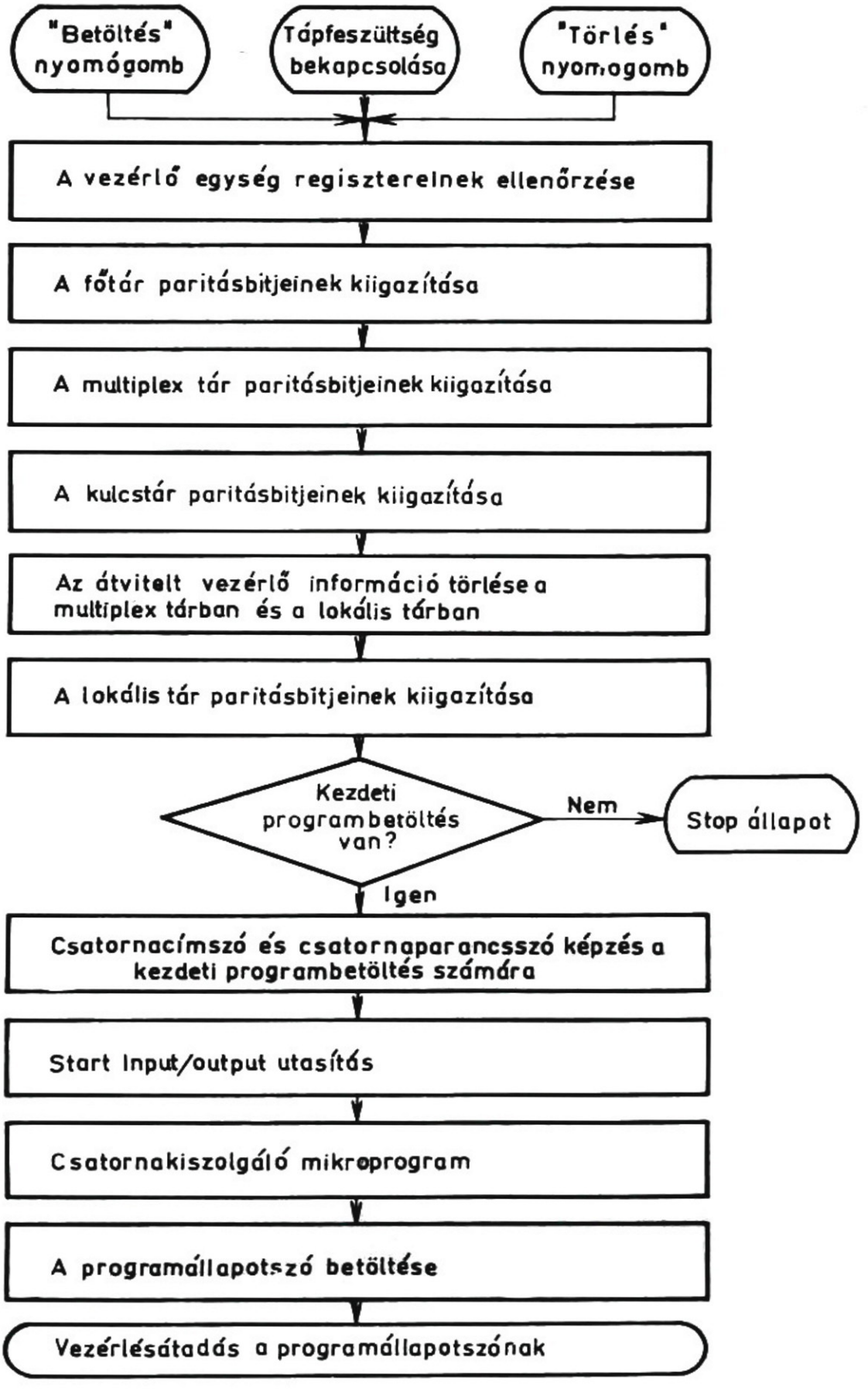
Funkcionális szempontból a kezelőpult két fő részre osztható: a vezérlő pultra és a kijelző pultra.

A vezérlő pulton helyezkednek el azok a nyomógombok, forgókapcsolók, üzemmódbeállitó kapcsolók, amelyek segitségével meghatározott tárcimeket, mikroprogramtárcimeket, perifériacimeket stb. állithatunk be. Az üzemmódkapcsolókkal a központi számitógép működésének módját szabályozhatjuk; például a programot folyamatosan, gépi utasitásonként vagy mikroutasitásonként akarjuk-e futtatni; mi történjen géphiba észlelésekor; a beállitott cim melyik memóriaterületre mutasson stb. A nyomógombok segitségével a központi számitógép működését indithatjuk, leállithatjuk, a beállitott gépi vagy mikroutasitáscimet a kapcsolókról átirhatjuk a tár vagy mikroprogramtár cimregiszterébe, eldönthetjük, hogy a pulton kapcsolókon beállitott bináris adatokat be akarjuk-e irni a tárba, vagy a tár adott területének tartalmát a pultra ki akarjuk-e vinni stb. Hasonló nyomógomb szolgál arra, hogy a pultról mesterségesen programmegszakitást hozzunk létre.

A kijelző pulton található a központi számitógép és esetleg egyes perifériák fontosabb regisztereinek kijelzése. A kijelzések mindig az aktuális regisztertartalmat mutatják: a világitó izzó "1"-et, a nem világitó izzó "0"-t jelent a megfelelő bitpozicióban. Igy például figyelemmel kisérhetjük az aktuális programállapotszó, csatornaállapotszó, csatorna interface, valamint a tár cimregisztereinek információtartalmát. Gépi utasitásonként végrehajtott program esetén megjelenithetjük a kijelző pulton a keletkezett eredményt, ellenőrizhetjük annak helyességét. Természetesen, a folyamatosan működő számitógéprendszernél a regiszterek, tárterületek tartalma olyan gyorsan változik, hogy a kijelzőpult lámpáinak nincs is elég idejük, hogy kialudjanak, majd ujból felvillanjanak. Az ilyen gép pultján a lámpák halványan, alig észrevehetően vibrálnak. A kijelző pult izzóinak ellenőrzésére külön nyomógomb szolgál: ezt megnyomva valamennyi izzó egyszerre világit.

3.6.3 A KEZELŐPULT MIKROPROGRAMJAINAK MŰKÖDÉSE

A kezelőpultról inditott müveletek is ugyanazt a mikroprogramozott vezérlő egységet használják, mint a rendszer többi eleme. Más szavakkal, a pultmüveleteknek is megvan a saját mikroprogramja. Különbség csupán az a gépi utasitások mikroprogramjaihoz viszonyitva, hogy e mikroprogramok inditásához nincs szükség a mikroprogramtár cimregiszterének feltöltésére, hanem a mikroprogramok egy-egy erre a célra kijelölt nyomógombbal indithatók. A 99. ábrán felrajzoltuk az R20 számitógép



99. ábra. A kezelőpult mikroprogramjai

pultmikroprogramjainak blokkdiagramját. Mint látjuk, a Törlés és a Kezdeti programbetöltés mikroprogramnak első része közös, majd elágazik aszerint, hogy melyik nyomógombot nyomtuk meg a pulton.

A Törlés, a Kezdeti programbetöltés mikroprogramjai ugyanazon a ponton indulnak, ahová a tápfeszültség (a hálózati feszültség) bekapcsolása is vezet. Ezzel egy fix cimről inditjuk el a közös mikroprogramot, amely ellenőrzi a vezérlő egység regisztereinek müködését, szükség esetén kiigazitja a különböző tárterületek paritásbitjeit, majd ha csak Törlés vagy Tápfeszültség bekapcsolása volt, a központi számitógépet leállitja.

Ha a kezdeti programbetöltés nyomógombját is aktiváltuk, akkor a mikroprogram előállitja mindazt az átvitelt vezérlő információt, amely az operációs rendszernek a főtárba történő beolvasásához kell. Láthatjuk, hogy a mikroprogram speciálisan erre szolgáló csatornacimszót és csatornaparancsszót hoz létre, majd egy Start Input/Output utasitással a vezérlést átadja a csatornát kiszolgáló mikroprogramnak. Ez elinditja a kezdeti programbetöltés folyamatát, majd betölti az operációs rendszert beolvasó program programállapotszavát, végül ehhez kerül a vezérlés.

3.6.4 A KEZDETI PROGRAMBETÖLTÉS FOLYAMATA. A SZÁMÍTÓGÉPRENDSZER ÜZEMBE HELYEZÉSE

Tekintsük most azt a folyamatot, amelynek az a célja, hogy a számitógép operációs rendszerét valamelyik háttértárról az operativ tárba juttassa! Az operációs rendszernek azt a részét, amely állandóan a tárban fog tartózkodni, egy speciális betöltő program: a kezdeti programbetöltő olvassa be a tárba. Ehhez azonban - hogy ti. a kezdeti programbetöltő futtatható legyen - magát a kezdeti programbetöltőt is be kell olvasni a tárba! Ennek a folyamatnak az elejét vázlatosan már láttuk a Kezdeti programbetöltés mikroprogramjánál. A kezdeti programbetöltés mikroprogramja generál egy csatornaparancsszót és egy ehhez tartozó csatornacimszót. Fontos, hogy ezeket megkülönböztessük a főtár 8-23. cimén található csatornaparancsszavaktól! A kezdeti programbetöltés legelső csatornaparancsszava csak "a mikroprogramok szintjén van meg". Ennek mindössze az a feladata, hogy az operációs rendszert tartalmazó háttértárról (legtöbbször mágneslemezről) beolvasson 24 byte-nyi információt a főtár 0-23. cimére. Ez tartalmazza azt a programállapotszót, amely vezérlése alatt fog majd lefutni maga a kezdeti programbetöltő, valamint tartalmaz 2 csatornaparancsszót. Ezek a csatornaparancsszavak parancsláncolással össze vannak kapcsolva, igy rövid csatornaprogramot alkotnak. Ezek feladata a kezdeti programbetöltő beolvasása. Ez szintén az operációs rendszert tartalmazó lemezcsomagról kerül be a tárba. Végül mikroprogramozott uton megtörténik a tár 0. cimén található programállapotszó aktualizálása, és ez vezérli a kezdeti programbetöltő futását.

A kezdeti programbetöltő beolvassa az operációs rendszernek azt a részét, amelynek állandóan a tárban kell lennie. Ezenkivül lehetőséget ad a gépkezelőnek, hogy speciális utasitásokkal az operációs rendszer egyes tulajdonságait, paramétereit meghatározza. Igy például megadható az éppen üzemképes perifériakészlet leirása, a naptári dátum, a pillanatnyi idő másodperc pontossággal. Ha mindezt a gépkezelő elvégezte, akkor a kezdeti programbetöltő még beolvassa a tárba az automatizált programfuttatást irányitó vezérlő programot, az ugynevezett Job Controlt. Ezzel a kezdeti programbetöltő szerepe véget ért: biztositotta az automatizált programfuttatás elkezdését, üzembe helyezte az egész rendszert.

Előfordulhat, hogy éppen a kezdeti programbetöltő beolvasása vagy futása alatt történik géphiba. Ekkor a kezdeti programbetöltést meg kell ismételni.

4. Perifériális egységek

Ezek az egységek zömmel különálló berendezéseket jelentenek, amelyek csak a csatornarendszeren (szelektor- és multiplexcsatornán) keresztül kapcsolódnak a központi egységhez. A központi processzor vezérlő egysége inditja el működésüket és ugyancsak ez szabályozza, koordinálja azt az operativ tárral. Önállóan nem,vagy csak elég korlátozottan használhatók,és rendszerint egymással is csak a központi egység operativ tárán keresztül kerülhetnek kapcsolatba. Ezért nemcsak elhelyezésük, hanem működésük alapján is indokolt a perifériális elnevezés. Ide tartoznak az input-output* (I/O) eszközök és a háttértárak, más néven külső tárak. Az I/O egységeken keresztül valósul meg az információcsere az ember és a gép között; a háttértárak feladata pedig a központi egység operativ tárának kapacitását kibővíteni lényegesen olcsóbb módon. Működésük során a perifériák tehát információt (adatokat, programokat) közvetitenek a központi egységbe,vagy vesznek át onnan. Adott esetben előfordul, hogy még egy járulékos feladatot el kell látnia a perifériáknak. Mégpedig a kódátalakitást; abban az esetben, ha az adathordozón levő adatok kódja nem azonos a gép belső adatábrázolási formájával (EBCDIC). Ilyen eltérő kód például a mágnesszalagnál használatos 6 bites BCD vagy a 7 illetve 8 bites ASCII kód, a lyukkártyán levő Hollerith kód, vagy a lyukszalagon alkalmazott kódok stb. A kódátalakitás egyik jelrendszerből a másikba általában automatikusan történik, de előfordulhat, hogy ezt is külön programozni kell.

A perifériális egységek működésüket, üzemmódjukat tekintve két csoportba sorolhatók. Ha a számitógép központi egysége vezérli őket és az operativ tárral közvetlen kapcsolatban is vannak, közvetlen, on-line üzemmódról beszélünk.

A közvetett, off-line üzemmód esetén a perifériális egységek nem állnak közvetlen kapcsolatban az operativ tárral, adatokat közvetlenül nem kapnak onnan, és nem továbbitanak oda (bár az összeköttetés lehetősége illeszthetőség szempontjából elvileg fennáll), és vezérlésüket nem a központi egység végzi. Igy működésük során nem használnak fel CPU időt. A közvetett berendezések vezérlését külön vezérlőegység látja el, és ez akár több berendezést is vezérelhet. Gyakran ugyanazt a berendezést, perifériát, közvetlen (on-line), vagy közvetett (off-line) üzemmódban is használhatjuk.

Megjegyzés: A közvetlen és közvetett jelző azonban nemcsak az üzemmódra használatos, hanem az adatátvitelre is! Az on-line, közvetlen adatátviteli rendszerek esetén a perifériáknál bevitt adatok az operativ tárba kerülnek, ott ellenőrzik, és további feldolgozásra előkészitik őket, majd ezek az adatok egy közbenső adathordozóra kerülnek. Igy egy későbbi pl. soros feldolgozás céljára készen állnak

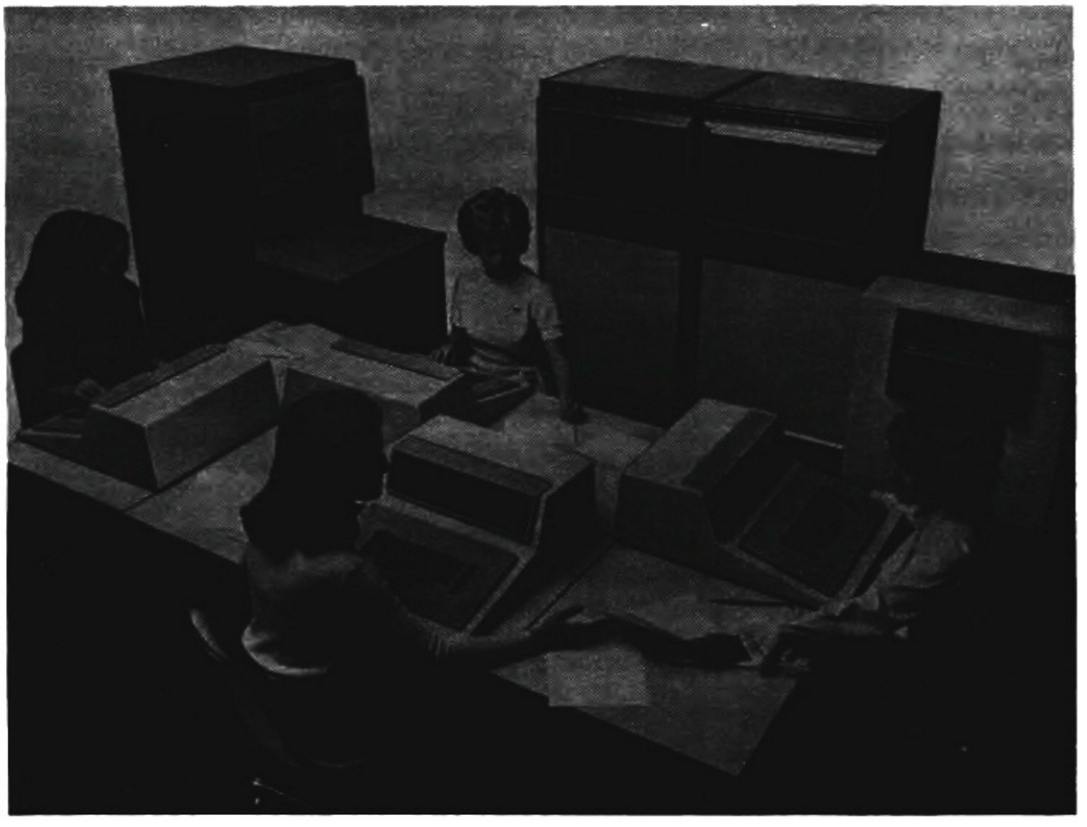
---

* Magyarul: bemeneti - kimeneti (B/K) egységek.

Az off-line közvetett adatátviteli rendszerekben a perifériákról jövő adatokat számitógép beiktatása nélkül gyüjtik egy központban, és ott adathordozóra (lyukkártya, lyukszalag) viszik. Az adathordozókat egy későbbi időpontban bocsátják az adatfeldolgozó berendezés rendelkezésére.

Az I/O egységek és a háttértárak funkciói müködésük során nem mindig választhatók el élesen egymástól. Gyakran előfordul, hogy pl. mágnesszalagot használnak fel input adathordozóként ugy, hogy másik számitógépen, vagy külön erre a célra alkalmas berendezésen viszik fel rá az információt. Ez utóbbi az un. közvetlen mágneses adatrögzitéshez tartozik. (Ilyen célra általában mágnesszalagot, ritkábban mégneslemezt használnak). A közvetlen mágneses adatrögzités előnye, hogy igy az információ beolvasási sebessége a rendszerbe kb. 20-30-szorosára nő a lyukkártyához vagy a lyukszalaghoz képest, kisebb a tárolási helyigénye, gazdaságosabb az adathordozó kihasználtsága, nagyobb az adatrögzités sebessége, könnyebb a hibajavitás, a szalag ismételten felhasználható stb. És az adatrögzités off-line üzemmódban, azaz a központi egység igénybevétele nélkül történik! Egyes esetekben előfordul, hogy nem a számitógép által beolvasható szalagra történik közvetlenül a rögzités, hanem először például kazettára. Ilyenkor természetesen egy külön konvertálással kell a kompatibilitást biztositani.

A közvetlen mágneses adatrögzités nagy hátrányát az jelenti, hogy egy személy csak viszonylag rövid szalagrészt (10-20 m) tud mondjuk naponta megirni (a nagy rögzitési jelsürüség miatt). Mivel arra nem lehet általában a feldolgozásnál időt hagyni, hogy ilyen tempó mellett egy egész szalagtekercs megteljen, egy-egy rögzitő napi "adagját" átmásolják, összegyüjtik egy szalagon (pooling), hogy a rendszerbe való beolvasás gazdaságos legyen. E mellett az un. egyedi rögzités mellett a csoportos rögzités is használatos. Itt minden egyes rögzitő, akár több helyről is egyetlen mágneses adathordozóra dolgozik. Ennek "lebonyolitásához", irányitásához azonban már általában minicomputereket alkalmaznak (100. ábra).



100. ábra. MDS 2400 csoportos, mágnesszalagos adatrögzitő rendszer

A háttértárak azonban nemcsak mint input, hanem mint output berendezések is felhasználhatók. Sok esetben a feldolgozás "eredményét", például a kimutatásokat nem sornyomtatón jelenitik meg, hanem mágnesszalagra viszik, és igy bocsátják a felhasználó rendelkezésére. Ilyenkor a mágnesszalag output adathordozóként szerepel. Ez a módszer részben a feldolgozási időt gyorsitja (sornyomtató kiiktatásával), részben a feldolgozás költségeit csökkenti (a kinyomtatást maga a felhasználó végzi el).

Ebben a fejezetben mi különválasztva tárgyaljuk a perifériákat: input-output egységek és háttértárak. Az előbbieket feldolgozási mód alapján csoportositjuk. Külön ismertetjük a kötegelt feldolgozás input, majd output eszközeit, és külön az interaktiv feldolgozás input-output eszközeit.

A perifériák tárgyalása során nem támaszkodunk konkrét berendezésekre, csak átlagos adatokat, müszaki jellemzőket ismertetünk. Viszont, ahol lehetséges, minden perifériatipusnál megadjuk a hazánkban nagy számban megtalálható R20 számitógép tipikus perifériáinak kivonatos müszaki adatait, jellemzőit.

4.1 KÖTEGELT FELDOLGOZÁS

A számitógépes feldolgozások klasszikus fajtája az ugynevezett kötegelt (vagy más néven adagolt) feldolgozás (batch processing). A felhasználó programjait kötegekbe, csomókba (batch) foglalva adja át feldolgozásra, és azok ennek megfelelően, kötegenként kerülnek be a rendszerbe. Az input és output müveletek közti időre a felhasználó el van zárva a programjától, nem tudja javitani, nem tud a futás menetébe beleszólni. A programját csak akkor kapja vissza, ha az hiba folytán vagy a feldolgozás eredményes befejezése után teljes egészében kikerül a rendszerből.

A következőkben először a kötegelt feldolgozás legfontosabb adathordozóit, a lyukkártyát és a lyukszalagot ismertetjük, majd az egyes input és output eszközöket.

4.1.1 LYUKKÁRTYA

Az adatfeldolgozó rendszerek bemenő adatainak legáltalánosabban használt és széleskörüen elfogadott adathordozója a lyukkártya. A számitógép által szolgáltatott információk (kimenő adatok) rögzitésére is használják, bár ezen a területen a sornyomtató erősen háttérbe szoritja. Az információ számok és szavak segitségével kerül megfogalmazásra, és ezek mindegyike karakterekből tevődik össze. A lyukkártyának ezeket a karaktereket kell rögziteni tudni.

A kártyán rögzitendő karakterek a következők:

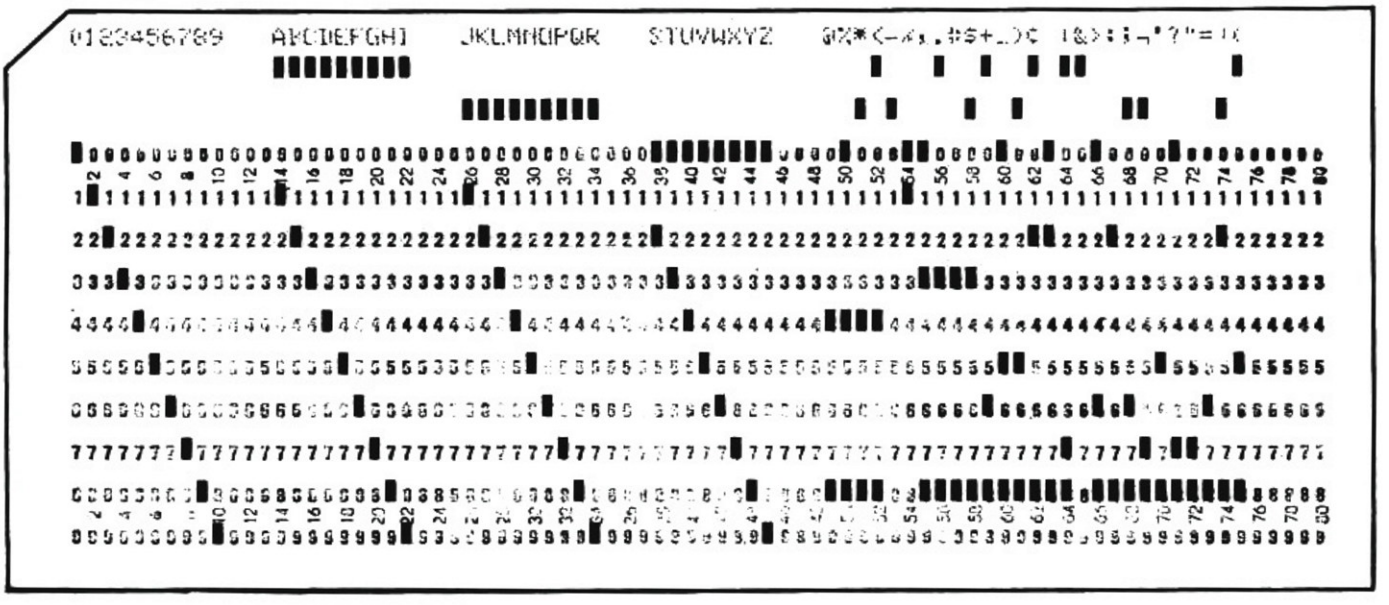
számjegyek	0,1,2,...9
betük	A,B,C...Z
különleges karakterek	+, -; /.* $ + "...

A kártya tulajdonképpen egy különleges minőségü papir lap, amelynek szabványos alakja és mérete van, hogy géppel is könnyen feldolgozható legyen. Hogy a kártyacsomagba a forditva bekerült kártyák felismerhetők legyenek, a kártyák bal felső sarkát lekerekitik vagy levágják. A kártya képzeletben 12 vizszintes

sorra van felosztva, ezeket "lyuksoroknak" is hivják. Ezek közül 10-et külön meg is jelölnek számjegyekkel (0-tól 9-ig terjedően). Minden sor a számjegyének megfelelő nevet viseli. Igy a 6-osokat tartalmazó sort 6-os sornak hivják. A két legfelső sor (amelyet nem jelöl külön nyomtatott számjegy) közül a felső az ugynevezett 12-es sor, az alatta levő pedig a 11-es. A 12 sort két csoportra osztják, ugynevezett zóna és numerikus sorokra. Három zónasor van, mégpedig a 12-es, 11-es és a 0-ás sor (az utóbbit, mint zónasort, ilyenkor szokták 10-es sornak is hivni). A többi sor (1-től 9-ig) az ugynevezett numerikus sor. A kártya ezen kivül oszlopokra is fel van osztva. Az oszlopokat több helyen (általában a 0-ás és 1-es, és a 8-as 9-es sor között, illetőleg a 9-es sor alatt) sorszámozással is jelölik. A legelterjedtebben használt lyukkártya-fajta 80 oszlopból áll. Ez az un. Hollerith-féle kártya. Hazánkban is ez terjedt el. (De ezen kivül létezik 45, 51, 60, 90, sőt 96 oszlopos kártya is.) Minden oszlopban egy karakter helyezhető el, igy a legáltalánosabb kártyafajta 80 karaktert képes rögziteni: az adatokat az oszlopon belül "téglalapalaku lyukak" kombinációja képviseli.

A bemenő adatok kártyán való rögzitéséhez használt legáltalánosabb kód az ugynevezett Hollerith-kód. (Az ESZ rendszerben ennek cirill betükkel kiegészitett változata a KPK-12.) Ennél valamilyen karakter ábrázolására az oszloponként lehetséges 12 lyukasztásból maximum 3 szükséges.

Minden számjegyet egy lyukasztás ábrázol, mégpedig abban a sorban, amelyik számjegyet jelenteni akarja. (101. ábra) Ha egy szám több számjegyből áll, a rögzités több egymás utáni oszlopot vesz igénybe.



101. ábra. Karakterek ábrázolása lyukkártyán

A betük rögzitése egy oszlopon belül két lyukasztással történik. A két lyukasztás egyike zónalyuk, a másik számjegylyuk. Az EBCDIC karakter ábrázolásához hasonlóan a zónalyukak itt is a betücsoportokat jelölik, a numerikus lyukak pedig a betü csoporton belüli sorszámát jelentik.

Betücsoportok	Zónalyukasztás	Numerikus lyukasztás
A-I	12-es	1...9-es
J-R	11-es	1...9-es
S-Z	10-es	2...9-es

A 0-ás sor lyukasztása a 0-ás számjegyet jelenti, ha az oszlopon belül máshol nincs lyuk. Ha ugyanebben az oszlopban másik számjegylyukasztás is van, akkor a 0-ás lyukasztás mint 10-es zónalyukasztás értelmezendő.

A különleges karaktereket egy, kettő vagy három lyukasztás ábrázolja.

Adott esetben a lyukkártyán pozitiv és negativ számokat is lehet ábrázolni. Ha egy szám előjele negativ, akkor a számot ábrázoló oszlopok közül az utolsóban a numerikus lyukasztáson kivül egy zónalyukasztás is szerepel, a 11-es sorban. Igy végeredményben a negativ számokat a második betücsoportnak megfelelő lyukkombináció jelenti. Erről a felhasználónak nem szabad megfeledkezni!

A lyukkártyán rögzitett információ értelmezését, elolvasását nagymértékben segiti az ugynevezett feliratozás. Ez a lyukak segitségével tárolt adatok szöveges formában való megadását, felirását jelenti a kártyán a 12. sor felett.

A lyukkártyán az egy fogalomhoz tartozó adatok (elemi adatok) ugynevezett mezőkben (adatmezőkben) helyezkednek el. A mező oszlopainak száma a mezőbe kerülő fogalom maximális karakterszámától függ. Igy egy mező általában több egymás melletti oszlopból áll (de előfordulhat egy oszlopból álló mező is!) Ha egy szám kevesebb számjegyből áll, mint amennyi ábrázolására a mező lehetőséget adna, a ki nem használt oszlopokat általában nullákkal vagy space-ekkel töltik ki. Az ugynevezett értékmezőkben ezért a számokat (az adatokat) "jobbra ütköztetve" adják meg. Szöveges adat esetén viszont, ha az információ kevesebb helyet (oszlopot) foglal el, mint amennyi a mezőben rendelkezésre áll, a szótól jobbra eső oszlopokat nem lyukasztják ki ("balra ütköztetés").

A kártyán felesleges adatokat (főleg irásjeleket) nem rögzitenek. Igy például a dátum megadásánál a pontokat elhagyják.

76.06.01. helyett 760601 kerül rögzitésre.

A lyukkártyáknak a felhasználási és feldolgozási módjuktól függően több fajtájuk van, és ennek megfelelően más és más elnevezésük is. Az ugynevezett duálkártyák részben output, részben input adathordozóként is szerepelnek. Az adatokat több fázisban viszik fel rájuk, az ismertek vagy kézi lyukasztással, vagy a rendszerből való kikerüléskor közvetlenül kerülnek a kártyán rögzitésre lyukasztással és esetleg feliratozással is. A további adatokat azután kézirással rávezetik az e célra kijelölt mezőkre. Ezeket az "irt jeleket" aztán vagy elektromosan érzékelve dolgozzák fel - Mark Sensing -, vagy egy későbbi továbbfeldolgozás során a kártyán lyukak formájában is rögzithetik. Ilyen felhasználásra mutat példát az alább bemutatott ugynevezett leolvasókártya (102. ábra). Ezt az utólagos lyukasztási fázist igyekeznek azonban kiküszöbölni. Sok kártyaolvasó képes a kézzel irt jelek beolvasására is (OMR-Optical Mark Reader). Ezek az adott pozicióba (általában bekeretezett kis helyekre) huzott, fényt vissza nem verő jeleket felismerik és ugyanugy beolvassák, mint a lyukakat. Egy kártyán belül vegyesen is szerepelhetnek "lyukasztott" és "irt" jelek. Ilyen tipusu közüzemi dijbeszedéshez használt kártyát mutat az alábbi, 103. ábra.

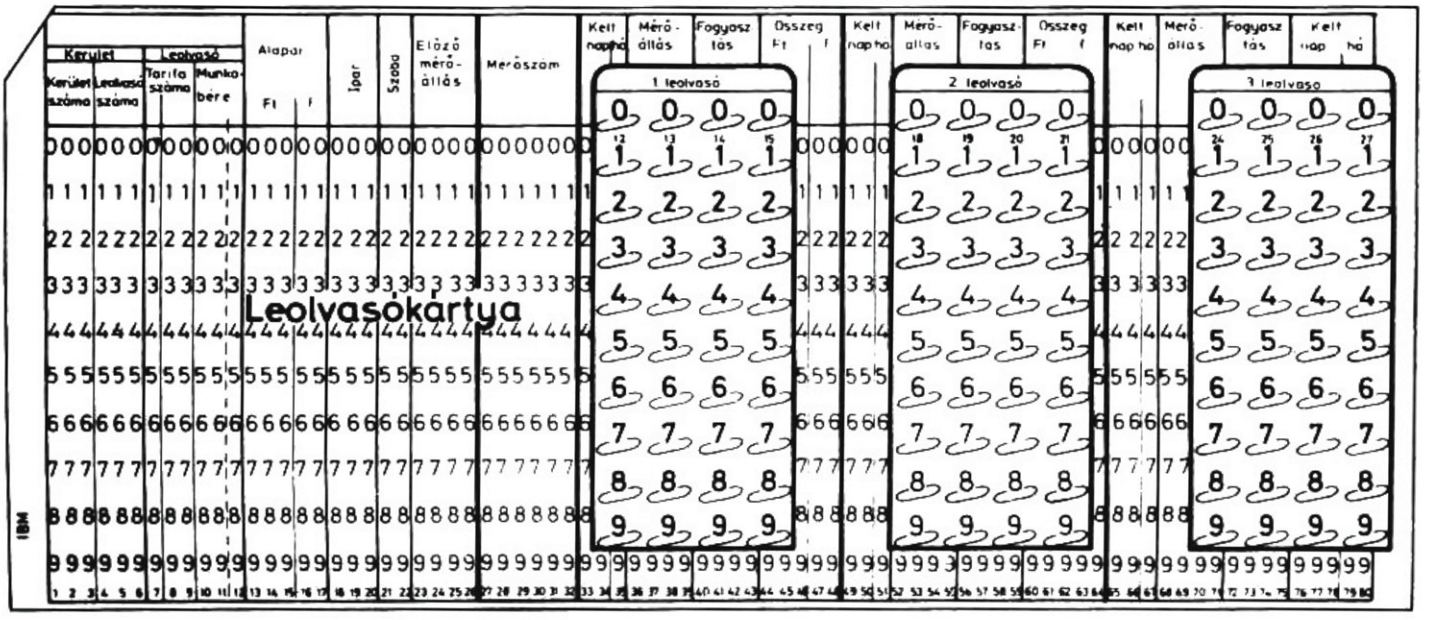
A kimenő adatok lyukkártyán való rögzitéséhez a már ismertetett Hollerith kódon kivül alkalmazzák az ugynevezett 6 bites BCD kódot, a bináris rögzitést, és ritkán az un. BULL-kódot.

A BCD kódban történő ábrázoláskor a kártya kapacitása tulajdonképpen megkétszereződik, mivel minden oszlopban két darab 6 bites BCD karakter helyezhető el. A kártya a 3-as és a 4-es sor között képzeletbeli vizszintes vonallal igy két részre van osztva.

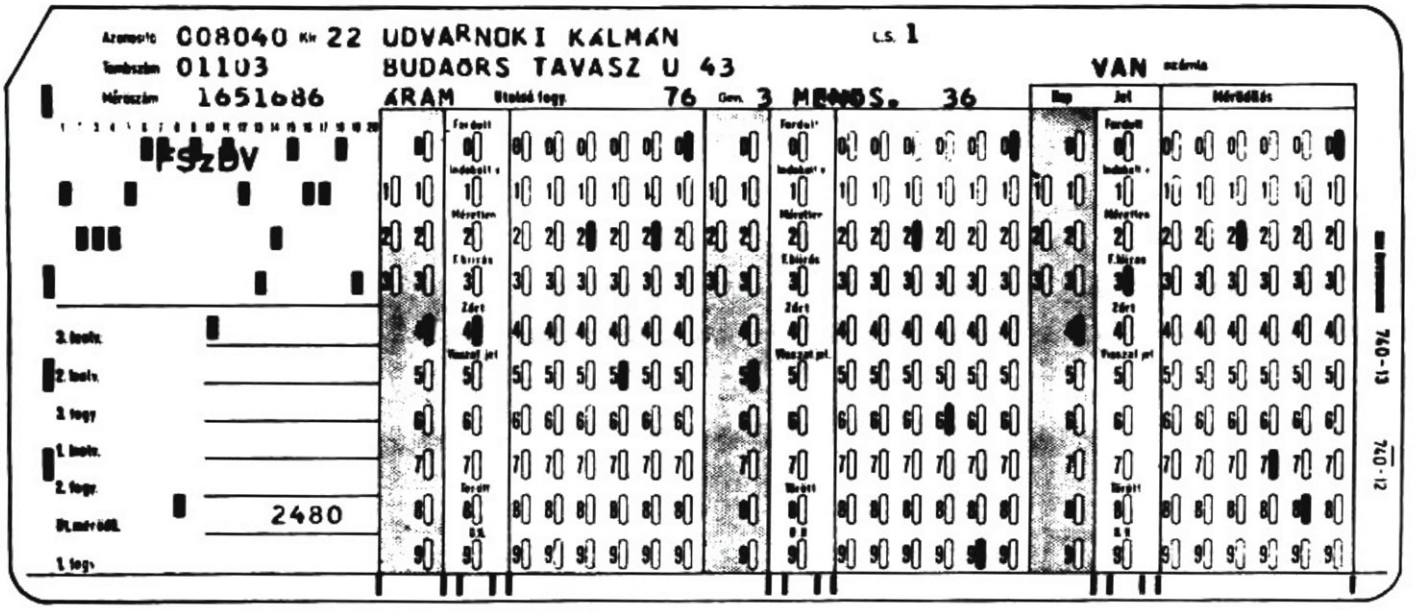
Megjegyzés: A 6 bites BCD (Binary Coded Decimal) kód nagyon hasonló az EBCDIC-hez. Ez utóbbi tulajdonképpen ennek továbbfejlesztett, kibővitett változata. Az eltérés az, hogy a BCD kódban a zónarész csak két bitből áll. Ezekkel az alábbi módon lehet a karaktereket megadni:

karaktercsoportok	zónabitek	numerikus bitek
A-I	11	0001-1001
J-R	10	0001-1001
S-Z	01	0010-1001
0-9	00	0000-1001

Az ábrázolható karakterkészlet igy természetesen kisebb, mivel a 6 pozición csak $2^6 = 64$ különféle állapot ábrázolható, szemben az EBCDIC 128 karakterével.



102. ábra. Jelölésolvasásra (Mark Sensing) alkalmas duálkártya



103. ábra. Optikai jelolvasásra (OMR) alkalmas duálkártya

A számitógép kimenő adatai a gépnek megfelelő bináris formában is rögzithetők a kártyán. Egy későbbi feldolgozásnál igy az információ konvertálás és kódátalakitás nélkül, felhasználásra készen olvasható be a gépbe.

A 80 oszlopos lyukkártyánál a ritkábban alkalmazott Bull-kód csak annyiban tér el a Hollerith-től, hogy az alsó három sor (7, 8, 9) alkotja a zónarészt, és a felette levő kilenc pedig a numerikus részt.

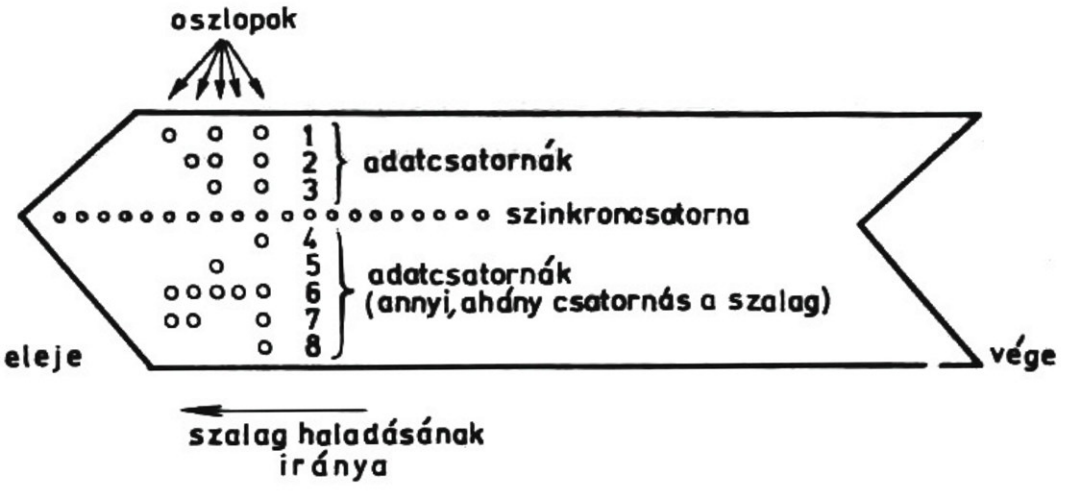
4.1.2 LYUKSZALAG

A lyukszalag egy hosszu, több száz méteres papircsik, amelyen a különféle karaktereket lyukkombinációk segitségével ábrázoljuk.

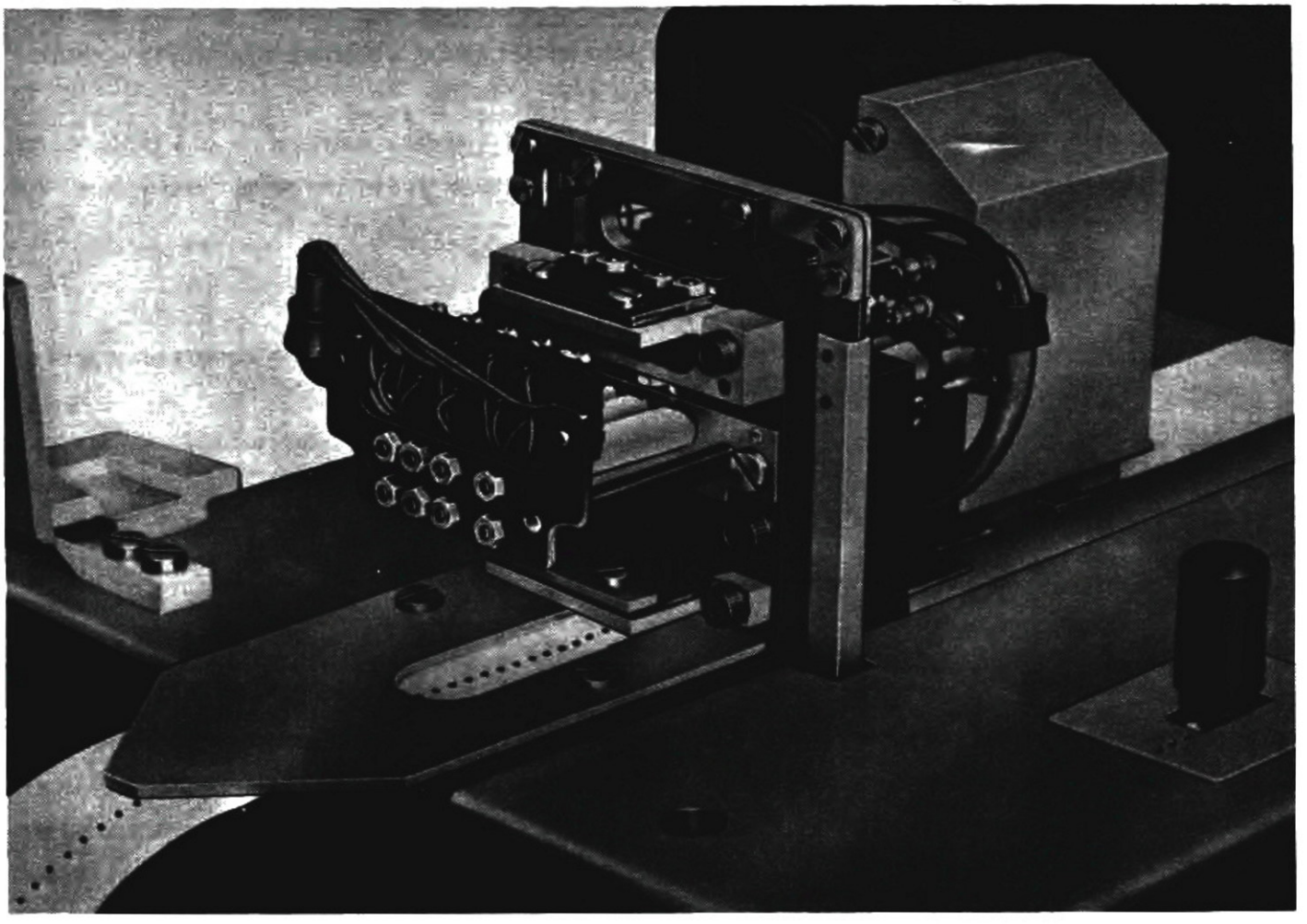
A lyukszalagon hossztengelyével párhuzamosan adatcsatornák (sorok, információsávok) és erre merőlegesen oszlopok (frame) vannak. Létezik 5, 6, 7 és 8 csatornás lyukszalag. A papircsik szélessége ebből függően változó:

5 csatornásnál 17,4 mm
6 csatornásnál 22,2 mm
7 és 8 csatornásnál 25,4 mm.

A karakterek rögzitése az oszlopokban történik. A felhasználatlan szalagon még semmiféle lyukasztás sincs. Az ugynevezett információs lyukak (átmérő ≈ 1,8 mm) lyukasztásával egyidőben ugynevezett szinkronlyukak (vezető-, ütemlyukak, átmérő ≈ 1,2 mm) lyukasztására is sor kerül egy külön ugynevezett szinkroncsatornában. Ezek az elektromechanikus olvasás és továbbitás esetén a szalag továbbitására szolgálnak, a gyorsabb olvasási és továbbitási módoknál viszont az olvasás szinkronizálását, a karakterek lehetséges helyét jelzik. Két szomszédos szinkronlyuk közepének és ilyen módon az esetleges információs lyukaknak (karaktereknek) a távolsága, az ugynevezett osztástávolság 2,54 mm. A csatornák közepének egymástól való távolsága szintén 2,54 mm. A szinkroncsatorna (szinkron lyuksor) mindig a 3. adatcsatorna után következik, vagyis a lyukszalagon a csatornák a 104. ábrának megfelelően helyezkednek el. A szalagon a továbbitás irányát is külön meg kell jelölni, mivel más módon az információ elhelyezkedése, a kiolvasás sorrendje nem állapitható meg. Erre szolgál a szalag elején és végén levő nyil alaku jel, amelyet egy "vágóberendezés" készit, amikor a felhasznált szalagrészt leszakitják (105. ábra)



104. ábra. Lyukszalag csatornái és oszlopai



105. ábra. Lyukszalaglyukasztó levett fedéllel és a "vágóberendezéssel"

Minden lyukszalagoszlopban egy karakter helyezhető el. Mivel minden információs "lyukhely" két állapotu lehet (ki van lyukasztva, vagy nincs), a csatornák számától függően különböző számu karakter ábrázolható az adott oszlopon.

A nemzetközi távíróforgalomban az 5 csatornás lyukszalag használatos, amelyen egy ugynevezett váltókarakterrel határozzák meg, hogy a következő oszlopban egy betüt vagy egy számot jelent-e az ott levő lyukkombináció. (9. táblázat). Erre azért van szükség, mert 5 csatornán csak $2^5 = 32$ különféle állapot fejezhető ki. Ez az un. Baudot-kód. Ennek nemzetközileg szabványositott változata az un. CCITT 2. számu távírókód. (A kettő csak néhány különleges karakter esetében tér el egymástól.)

6 csatornánál $2^6 = 64$; 7 csatornánál $2^2 = 128$; 8 csatornánál pedig már $2^8 = 256$ különféle állapot adható meg egy oszlopban. Ezeknél a karakterek rögzitéséhez a korábban már ismertetett BCD, vagy EBCDIC kódok valamelyik változata használatos. (Az EBCDIC-nek is van külön lyukszalagon alkalmazott 8 bites változata!)

Az adott hosszon való ábrázolás után a fennmaradó csatornákat, vagy a paritásbitek, vagy az átviteli vezérlőbitek, esetleg mindkettő rögzitésére használják. Ez utóbbi miatt alkalmazzák általában a 8 csatornás lyukszalagot. Ezen "járulékos bitek" kifejezésére azonban nincs kijelölve megadott csatorna, igy aztán a különféle kódokkal együtt szinte nemcsak gyártónként, hanem gyártmányonként is mást alkalmaznak. Nincs olyan általánosan elfogadott kód, mint a lyukkártya esetében a Hollerith volt. Még abban sem egységesek az egyes előállitók, hogy a berendezéseik alkalmaznak-e paritásbitet vagy sem,

9. táblázat

Az 5 csatornás lyukszalagon használatos két alapvető kódtipus

CCITT Sorszám	32	5	28	31	27	20	1	9	14	15	19	18	8	4	12	26	21	3	13	6	7	10	16	23	2	25	11	22	24	30	17	29
Bit-kombinációk		○					○				○			○		○	○			○	.	○		○	○	○	○		○	○	○	○
			○				○	○				○			○		○	○			○	○	○	○			○	○		○	○	○
	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
				○				○	○		○		○				○	○	○	○			○			○	○	○	○		○	○
					○				○	○		○		○				○	○	○	○	○			○		○	○	○	○		○
						○				○			○		○	○			○		○		○	○	○	○		○	○	○	○	○
CCIT Nr.2	⊖	E	≡	BK	<	T	A	I	N	O	S	R	H	D	L	Z	U	C	M	F	G	J	P	W	B	Y	K	V	X	⋮	Q	⋮
	⊖	3				5	–	8	,	9	'	4	↖	✢	)	+	7	:	.	↖	↖	♀	0	2	?	6	(	=	/	1⋮	1	A⋮
Teletype Baudot-kód	⊖	E	≡	BK	<	T	A	I	N	O	S	R	H	D	L	Z	U	C	M	F	G	J	P	W	B	Y	K	V	X	⋮	Q	⋮
	⊖	3				5	–	8	,	9	♀	4	#	$	)	"	7	:	.	!	&	'	0	2	?	6	(	;	/	1⋮	1	A⋮

□ Nincs lyukasztás

⊙ Van lyukasztás

< Kocsi vissza

≡ Soremelés

♀ Csengetés

1... Számváltó

A... Betűváltó

BK Betűköz

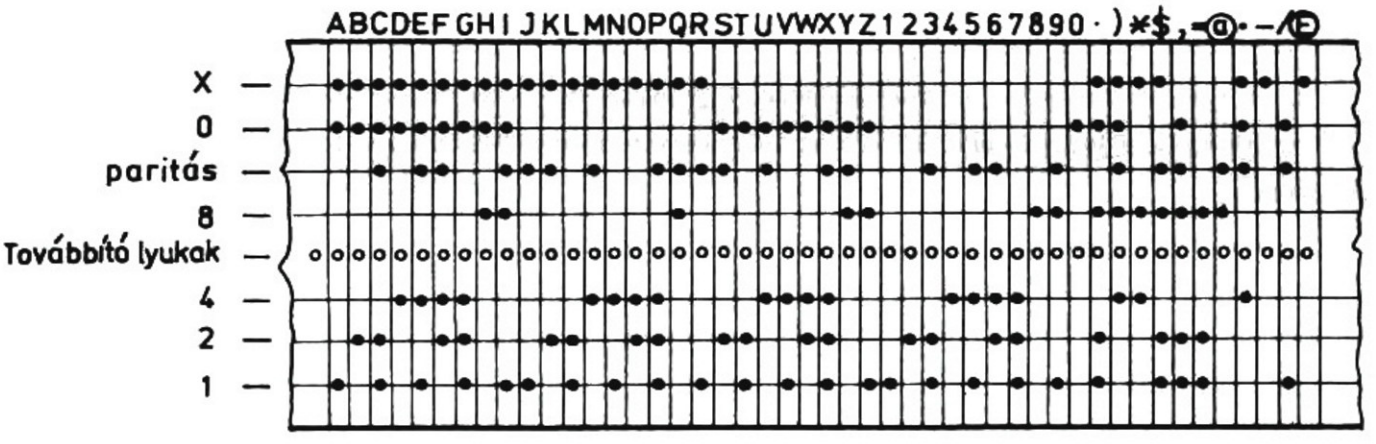
⊖ 32. Kombináció

↖ Különleges jeleknek fenntartva

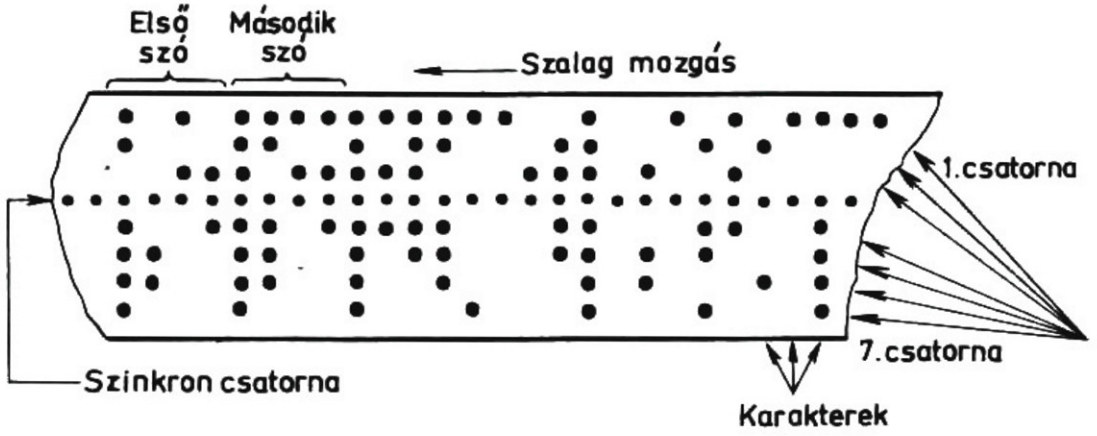
✢ Ki van ott ?

illetve ha igen, akkor párosat vagy páratlant. A nyolccsatornás lyukszalagon használatos sok kód közül mi egyet mutatunk be a 106. ábrán. (Az ESZ rendszerben használatos un . KOI-8 kódot már korábban, a 2. fejezet 3. táblázatában bemutattuk.)

A lyukszalagon is lehet a kártyához hasonlóan bináris információt rögziteni. Ilyenkor az egyes "szavak" kezdetét valamelyik csatornán külön kell jelölni. Erre mutat példát a 107. ábra. Itt a 7. csatornán levő lyukak vannak erre felhasználva. A szalagon levő első szó binárisan kiolvasva (balról-jobbra, alulról-felfelé) 73600514.



106. ábra. Nyolccsatornás lyukszalagkód



107. ábra. Lyukszalag bináris információval

4.1.3 LYUKSZALAG VAGY LYUKKÁRTYA?

Egyértelmü sorrendet a két adathordozó között felállitani nem lehet, mindegyik rendelkezik előnyökkel is és hátrányokkal is. Mindig az adott felhasználás, az alkalmazás körülményeinek ismeretében kell eldönteni, hogy melyiket is alkalmazzuk!

A lyukszalag különösen alkalmas adathordozó akkor, amikor az információforráshoz közeli adatrögzitésről van szó: olyan helyeken, ahol például a mérési eredményeket rögtön adathordozóra is lehet rögziteni vagy pedig a feldolgozás, a számitás adatairól közvetlen másolatot is lehet készitení, a lyukszalag vitathatatlan előnyökkel rendelkezik. Előállitása ilyenkor nem igényel külön munkát, szinte "melléktermékként" keletkezik (könyvelőgép, összeadógép stb.). Ezeknek a lyukszalagos adatrögzitést is elvégző gépeknek egy része azonban manapság csak numerikus feldolgozásra alkalmas (pl. az összeadógépek). Ezért ilyenkor a lyukszalag csak akkor "gazdaságos melléktermék", ha nincs szükség alfabetikus karakterek rögzitésére is. Az ilyen módon készült lyukszalag kis helyen elfér, egyszerüen szállitható, akár postán is továbbitható. Nagy tömegü bizonylatmozgatás takaritható meg ilyen formán.

A lyukszalag alkalmazásának további előnye, hogy a feldolgozásánál nincs előre megkötött fizikai rekordhossz. Minden információs mezőnek olyan hosszunak kell csak lenni, amilyen hosszu az adat, nem kell "kitöltő nullákat" alkalmazni. Az adathordozó fizikai kihasználtsága igy igen jó. Ugyanakkor a rekord hossza sem korlátozódik max. 80 byte-ra, mint a kártya esetében. A lyukszalag viszonylag nagy kapacitása ellenére kis helyen tárolható. Egy teljes 300 m-es lyukszalag tekercs kb. 100-120 ezer karakter tárolására képes. Ugyanennyi információ kedvező esetben is legalább 1200-1500 kártyát igényelne!

Előnye az is, hogy a hozzá tartozó olvasó és lyukasztó egységek olcsóbbak a kártyás I/O berendezéseknél és maga a lyukszalag is olcsóbb a kártyánál. Gyakran alkalmaznak a lyukszalag mellett ugynevezett konvertert is. Ez a tárolt információt mágnesszalagra rögziti és ez lényegesen gyorsabb beolvasást tesz lehetővé.

Ugyanakkor nem szabad megfeledkezni a lyukszalag hátrányairól sem! A rögzitett információ vizuális ellenőrzése nagyon nehéz, mivel nincs (nem szokásos) a szalagon a feliratozás és igy a tárolt

(kódolt) információ értelmezése meglehetősen bonyolult és lassu. Ezért a hiba helyét körülményes meghatározni, a hiba puszta felismerése is nehéz. A hibajavitás a lyukkártyától eltérően a lyukszalagon lényegesen bonyolultabb. A hibás részek kivágására, illetve uj részek közbeiktatására ugyan megvan az elvi lehetőség, de a hibajavitásnak ezt a formáját ritkán alkalmazzák, mivel a szalag elvágása és ujraragasztása a leggondosabb eljárás mellett is lényegesen rontja a szalag mechanikai tulajdonságait. Ezért amennyiben a hiba lyukasztás közben keletkezett, és még ekkor felismerték, a hibás adatot ugynevezett hibakóddal látják el, és utána a helyes adatot lyukasztják le. A beolvasáskor a hibakóddal ellátott részt a beolvasóprogram nem veszi figyelembe.

Gyakran alkalmazott módszer az is, hogy az adatok előkészitését, lyukszalagra vitelét un. verifier adatrögzitő berendezésen végzik, ahol ugyanezt az anyagot ketten lyukasztják le, egymás után. A második lyukasztás során a már meglevő szalag olvasása jelenti az ellenőrzést. Két személy ugyanazt a hibát, ugyanott igen kicsi valószinüséggel követi el, igy tehát a szalag várhatóan hibátlanul lyukasztott adatokat fog tartalmazni. A programozási hibák azonban továbbra is fennállnak, azok csak a fordítás, illetve a futás során derülnek ki.

A később felderitett hibákat (akár valamilyen adat megváltozását is) javitószalagon lehet korrigálni, a hiba helyének pontos megadásával. Ezt aztán külön kell beolvasni a számitógépbe, és ott helyettesiteni az igy kijavitott adatokkal a hibásakat. Ez általában a legjobb és leggyakrabban alkalmazott hibajavitási megoldás, annak ellenére, hogy tekintélyes CPU időt igényel.

10. táblázat

Lyukszalag bináris információval

Szempontok	80 oszlopos lyukkártya	8 csatornás lyukszalag
1. Tartósság	nagyobb	kisebb
2. Előállitási költség	nagyobb	kisebb
3. A szükséges I/O berendezések	drágábbak	olcsóbbak
4. Helyfoglalás tárolásnál	nagyobb	kisebb
5. Az adathordozó kihasználtsága	rosszabb	jobb (közel 3-szoros)
6. Eredeti bizonylatként felhasználható	igen	nem
7. Kódfelismerhetőség	egyszerü	nehéz
8. Adathordozón feliratozási hely	van	kevés
9. A lyukasztott adatok ellenőrizhetősége	egyszerü	nehéz
10. A lyukasztott adatok kiiratása	nehezebb (csak táblázógéppel)	könnyebb (lyukszalagvezérlésü irógéppel)
11. Rendezhetőség	jó	rossz
12. Hibajavitás	egyszerü	bonyolult
13. Fizikai rekord	van (80 karakter)	nincs
14. Programok rögzitésére	előnyös	hátrányos
15. Adatok rögzitésére	hátrányosabb	előnyösebb

A számitógépeknél a lyukkártya ma általánosabban használt adathordozó, mint a lyukszalag. Ennek legfőbb oka a feldolgozás előtt elvégezhető off-line rendezés, a könnyü hibafelismerés és javithatóság. A hibás utasitást, adatot tartalmazó kártya a feliratozás alapján könnyen kikereshető, eltávolitható, vagy ha kell,ujjal helyettesithető. Emiatt a programok kifejlesztésének, helyes összeállitásának optimális eszköze a kártya. Ugyancsak nagy jelentősége van a fentiek miatt a törzsadatok tárolásánál.

Valamely adat vagy programrészlet megváltozásakor is csak a változásokat tartalmazó kártyákat kell a régiek helyére (vagy közé) rakni. Igy korábbi programrészekből egy uj program kártyakötegét össze lehet állitani.

A kártya alkalmazásakor mindenképpen hátrányként jelentkezik a nagy helyfoglalása és a rossz kihasználtsága. Az összes felhasználható helynek csak kis részére kerül lyukasztás. (Gondoljunk csak a számok lyukasztása esetén az oszlopokban levő egy lyukra - a lehetséges 12 helyett!) A lyukkártya és a lyukszalag néhány jellemzőjének táblázatos formában való összehasonlitása látható. (10. táblázat.)

4.2 KÖTEGELT FELDOLGOZÁS INPUT ESZKÖZEI

A számitógép a környezetéből a feldolgozandó információt (programokat, adatokat) a bemeneti (input) egységein keresztül kapja meg. A kötegelt feldolgozásnál leggyakrabban használt bemeneti egységek:

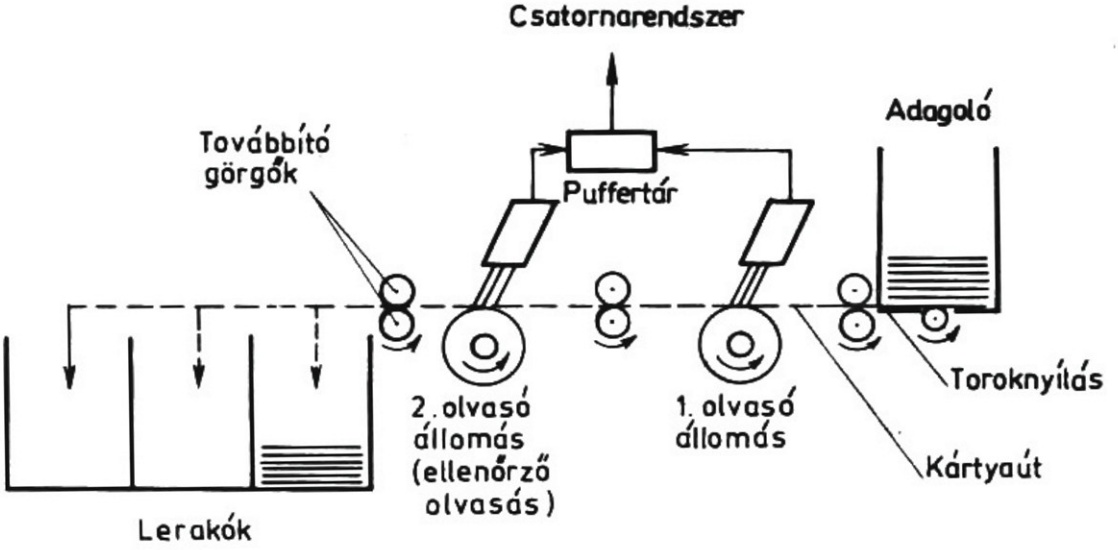
- lyukkártyaolvasó
- lyukszalagolvasó
- bizonylatolvasók

4.2.1 LYUKKÁRTYAOLVASÓ

A lyukkártyaolvasó (röviden kártyaolvasónak is hivják) az a perifériális egység, amely a lyukkártyán elhelyezett adatokat, programot beolvassa a rendszerbe, ennek segitségével azok a központi egység operativ tárába kerülhetnek. A kártyaolvasók külső felépitésre nagyon különbözőek lehetnek, de nagyjából mindegyik hasonló egységeket tartalmaz, és hasonló elven müködik.

A feldolgozásra, beolvasásra váró kártyacsomag (kártyaköteg) az adagolóba kerül. (108. ábra). Ebben 500-2500 kártya fér el egyszerre, tipustól függően. A kártyaköteg itt várakozik addig, amig a kártyák egyenként az olvasó állomáson át nem haladva egy vagy több lerakóba nem kerülnek. Ezek általában 800-3000 kártyát képesek befogadni. Több lerakó esetén a hibás kártyákat a gép automatikusan egy külön lerakóba rakja, de többnyire mód van ilyenkor a kártyák programmal való szelektált lerakására is (pl. off-line rendezés). A kártyaolvasók sebessége 300-2000 kártya/perc között változik, szokásos értékük a mai gépeknél 1200-1500 kártya/perc.

A kártya haladásának utját végig az adagolotól a lerakóig kártyautnak hivják. Az adagolóban levő kártyacsomag aljáról a toroknyiláson keresztül a kártyák egyenként kerülnek az adagoló mechanizmus



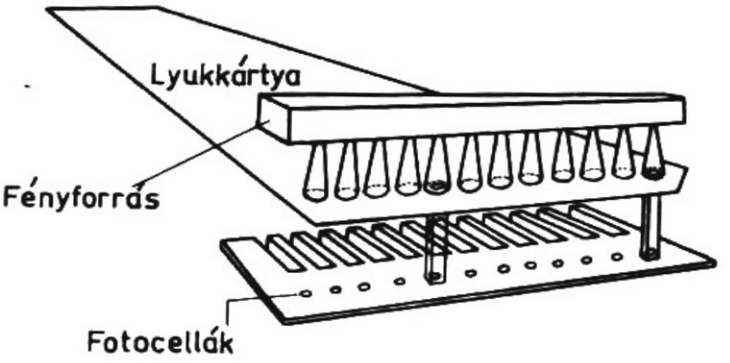
108. ábra. Lyukkártyaolvasó elvi felépítése

segitségével a kártyautba. A kártyák behuzására a kártyautba többféle módszer használatos. A legegyszerübb, de a kártyák rongálása miatt a legkedvezőtlenebb módszer a késes behuzás. Itt egy speciálisan kiképzett "kés" a kártyát hosszabbik oldala mentén áttolja a toroknyiláson. A kártya széle igy erős igénybevételnek van kitéve. Ennél kedvezőbb módszer a surlódó-, illetőleg a vákuumgörgős behuzás. Az előbbinél egy forgó gumi- vagy müanyaggörgő surlódás segitségével tolja a kártyát be a kártyautba. A vákuumgörgős megoldásnál még ez a kis mechanikus igénybevétel is csökken, mivel itt a görgő a vákuum segitségével (és nem surlódással) továbbitja a kártyát. A görgő palástján apró lyukak vannak, a belsejéből pedig folyamatosan kiszivattyuzzák a levegőt. Igy a kártyát a külső légnyomás nekinyomja a görgő felületének, és annak forgása közben a kártya továbbhalad. A ma használatos kártyaolvasóknál is általában a két utóbbi módszert alkalmazzák.

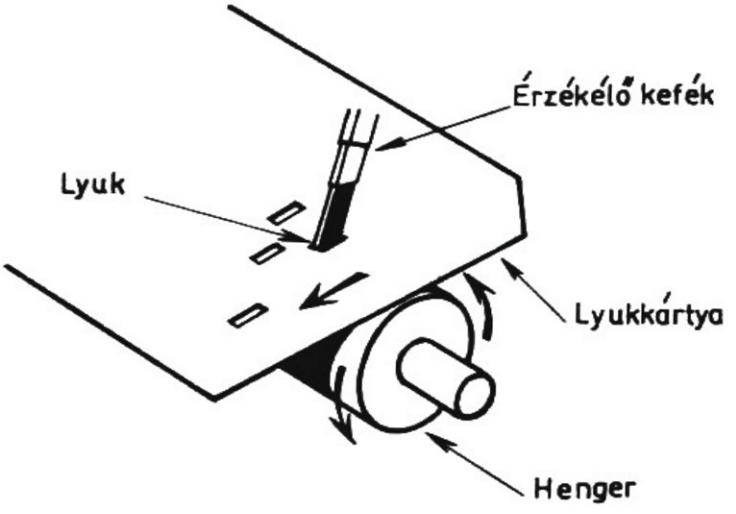
A kártyauton gumiszalag vagy továbbitó görgők (általában surlódógörgők) szállitják a kártyát, miközben mikrokapcsolók vagy fotodiódák figyelik a haladás szabályosságát. A tárolt információ beolvasását a kártyaut mentén elhelyezkedő egy vagy két leolvasó állomás végzi. A leolvasás vagy soronként (sorról-sorra haladva), vagy oszloponként (oszlopról-oszlopra haladva) történhet, géptipustól függően. Gyakoribb az oszloponkénti olvasás. A lyukak érzékelése fotoelektromos vagy elektromechanikus módon történik.

Fotoelektromos olvasásnál a kártyára párhuzamos fénynyalábot bocsátanak és a kártya tuloldalán minden egyes pozicióban egy-egy fényérzékelő elemmel (pl. fotodióda, fototranzisztor) külön figyelik, hol haladt át a kártyán a fény, vagyis hol volt lyukasztás. (109. ábra). Az OMR kártyák olvasása esetén a fényérzékelő elemek nem a fényforrással ellentétes oldalon, hanem ugyanazon oldalon vannak, igy a visszavert fényt érzékelik. Ahol megfelelő, fényt vissza nem verő jel került a kártyára (pl. puha ceruzavonás), ott a visszavert fény hiánya alapján adnak az érzékelők jelzést. Ez az olvasási módszer nagy sebességü olvasást tesz lehetővé, a beolvasás sebességét általában csak a kártyatovábbitással kapcsolatos problémák korlátozzák.

Az elektromechanikus (kefés) olvasás kisebb olvasási sebességet eredményez. Lényege, hogy a kártya egy érintkező (kefe) sor és egy forgó fém henger között halad el (110. ábra). A kefesorban annyi különálló fém kefe helyezkedik el, ahány pozicióban az olvasást végre kell hajtani. Például oszlo-



109. ábra. Fotoelektromos olvasás (oszloponként)



110. ábra. Elektromechanikus olvasás
(soronkénti olvasás - egy kefe a 80 közül)

ponkénti olvasásnál 12 kefe van. A kártya áthaladása közben csak azok az érintkezők (kefék) érhetnek a fém hengerhez, és igy csak ott záródhat az áramkör, ahol a kártyán lyukasztás van. Ez az olvasási mód kártyaolvasókban a kis sebesség miatt ritkán használatos.

Néhány kártyaolvasó, főleg a korábbi gyártmányok a kiolvasott adatokat azonnal továbbitják a processzorba.

A mai, korszerübb tipusuak egy helyi, belső tárral, puffertárral rendelkeznek. A kártyáról olvasott adatok először ide kerülnek. A kártya tartalmának helyes olvasása és a kiolvasott jelek ellenőrzése után jutnak a csatornarendszeren (multiplex csatornán) keresztül az operativ tárba. Igy egy kártya tartalmát sokkal rövidebb idő alatt tudja a csatorna továbbitani, vagyis rövidebb ideig lesz a csatorna foglalt, mintha sorról sorra, vagy oszlopról oszlopra olvasva külön történne az adatátvitel. Egyes olvasókban a sorról-sorra történő kiolvasás után a puffertár tartalma már oszlopról oszlopra kerül a csatornarendszerbe, tehát a puffertár a sor-oszlop konverziót is elvégzi.

A kártyaolvasók az olvasással együtt ellenőrzést is hajtanak végre, és a hibásan olvasott kártyát külön lerakóba helyezik vagy megállnak, esetleg más módon jelzik a hibát. A kártyán levő adatok olva-

sásának helyességét többféle módon is ellenőrizhetik. Az egyik általánosan használt mód a kettős olvasás. Ekkor a kártya két olvasófokozaton halad keresztül. A második olvasás eredményét a gép összehasonlitja a puffertár (első olvasás) tartalmával és ha a két olvasás eredménye azonos, az olvasott információt helyesnek fogadja el. Ha a két olvasás eredménye nem egyezik meg, az olvasás hibás.

Másik igen gyakran alkalmazott módszer az ugynevezett valós lyukkombináció ellenőrzésén alapszik. Ez ugy történik, hogy az egy-egy oszlopban levő lyukakat számolja meg a berendezés. Például a Hollerith-kódnak megfelelően lyukasztott kártyán egy oszlop (karakter) nem tartalmazhat 3-nál több lyukat. Ha az olvasó többet számolt, ez hibás olvasást vagy hibás adatot jelent. Igen kicsi a valószinüsége, hogy egy oszlopon belül egy lyukasztást ne érzékeljen és ugyanakkor egy lyukasztatlan helyen lyukat olvasson a gép. Ekkor a karakter hibás ugyan, de a lyukak száma nem változott, igy a gép nem jelez.

A lyukkártyán egy-egy oszlopban 12 lyukasztási pozició áll rendelkezésre, ami 4096 különböző lyukkombinációt tesz lehetővé. Mivel a különböző kártyakódok ennél jóval kevesebbet használnak fel (48, 64, 72 karaktert), a valós lyukkombináció ellenőrzése ugy is elvégezhető, hogy a leolvasott karaktereknél megvizsgáljuk, beletartozik-e a gép által használt karakterkészletbe. Ha nem, hibás az olvasás. Több ellenőrzési módot együttesen alkalmazva, gyakorlatilag nulla annak a valószinüsége, hogy hibás információt olvasson be a gép. Binárisan lyukasztott kártyánál a lyukszámlálás és a karakter-helyesség ellenőrzése természetesen nem alkalmazható, mivel itt bármely konkrét kombináció előfordulhat. Ekkor csak a kettős olvasás alkalmazható.

Magának az olvasó berendezésnek az ellenőrzését szolgálja az ugynevezett sötét-világos próba. Mivel két kártya áthaladása között valamennyi érzékelő áramkör (fotodióda) fényt kap, tehát valamennyinek "világost" kell jeleznie. Amint viszont a következő kártya széle a leolvasó egység alá kerül, az valamennyi diódát eltakarja, igy egy sem adhat jelzést (sötét). Ha valamelyik dióda teljes megvilágitáskor sötétet jelez, vagy takaráskor világosat, ez hibát jelent és a beolvasás megszakad. Ez a vizsgálat elegendő biztonságot nyujt, mivel nagyon kicsi a valószinüsége annak, hogy a beolvasás igen rövid ideje alatt hibásodjon meg a dióda. A mozgási sebesség ismeretében a gép azt is képes megállapitani, hogy a kártya hossza megfelelő volt-e. Igy a mérethiba miatti elcsuszásokat is el tudja kerülni.

A kártyák haladását valamennyi kártyaolvasó figyeli, és bármilyen rendellenességet tapasztal, megáll. A leggyakoribb ilyen rendellenesség a begyürés. Hibás, sérült vagy deformálódott kártya nem képes szabályosan haladni a kártyauton, valahol megakad, összegyürődik. A gép ilyenkor megáll, és rendszerint jelzőlámpával utal a hibára.

ESZ-6012 Kártyaolvasó

Bináris és alfanumerikus információ beolvasására képes, ha azok az ISO szabványos előirásainak megfelelően vannak rögzitve. Ha a lyukkártyán KPK-12 kódban lyukasztott információ található, a berendezés a kiolvasott információt DKOI kóddá alakitja át. Az olvasás üzemmódját az utasitáskód határozza meg.

A berendezés ellenőrzi a leolvasott információt a lyukkombináció szerint, valamint meggyőződik az olvasó és a szinkronizáló egység hibátlan müködéséről, mielőtt ujabb kártyát továbbitana. Ezenkivül jelzést ad, és leállitja a kártya olvasását, ha a kártyautban kártyát gyür össze, ha kettős kártyaadagolást végez, vagy ha kiürül az adagoló, illetőleg, ha a lerakó megtelik.

Olvasási sebessége (kártya/perc)	max. 500
Az információhordozó lyukkártya	45 vagy 80 oszlopos
Kártyaut	1
Az olvasás elve	oszloponkénti olvasás; fotodiódás
Adagoló és lerakó kapacitása (db)	max. 1000

4.2.2 LYUKSZALAGOLVASÓ

Ezt a perifériát az elsősorban müszaki, illetve tudományos számitásokra létrehozott rendszerekben alkalmazzák. Itt viszonylag kevés a be- és kimenő információ, igy ezek hordozására a lyukszalag előnyösen használható. Ezek az adatok igen gyakran egy korábbi feldolgozás eredményeképpen, vagy "melléktermékeként" is előállithatók. Adatfeldolgozásra szolgáló rendszerekben viszonylag ritka. Elsősorban ott alkalmazzák, ahol egy korábbi lyukszalagos gépparkhoz kapcsolódó számitógépes rendszert hoznak létre. Müködésük a kártyaolvasók müködésénél egyszerübb, hiszen az összefüggő szalag olvasását egyszerübben lehet megoldani.

Előfordul, hogy lyukasztóval egybeépitve, mint lyukszalag olvasó-lyukasztó egység kerül felhasználásra, vagy pedig mint átalakitó is müködik, és lyukszalagot olvas, lyukkártyát pedig lyukaszt.

Egyes tipusok csak adott szélességü, tehát rögzitett csatornaszámu szalagot képesek feldolgozni, a korszerübbek viszont egyszerü átkapcsolással bármilyen szélességü (5-8 csatornás) szalag olvasására is alkalmasak. A berendezések müködési (olvasási) sebessége a felépitéstől függően igen eltérő, azonban szinte valamennyi a 300-2000 karakter/sec teljesitményhatáron belül müködik.

A tárolt információ kiolvasásához leginkább vagy a fotoelektromos vagy a dielektromos módszert használják fel. Az előbbi müködési elvét a lyukkártyaolvasónál már ismertettük. A dielektromos módszer a fotoelektromoshoz hasonló gyors olvasást tesz lehetővé (1000-2000 karakter/sec). Itt a szalagon levő csatornákat egy-egy kondenzátor két fegyverzetét alkotó fémlemezpár veszi közre. Ahol a szalagon nincs lyukasztás, ott a papir jelenti a két fémlemez (fegyverzet) közti szigetelőanyagot, ahol lyuk van, ott pedig a levegő. A kondenzátorok kapacitásai annak megfelelően változnak, hogy mi van a lyukszalagon rögzitve és igy papir vagy levegő van-e a két fémlemez (fegyverzet) között.

A kártyaolvasóknál emlitett elektromechanikus (tüs, kefés) módszer elvileg a lyukszalag olvasásánál is felhasználható, azonban a kis olvasási sebesség miatt (10-50 karakter/sec) főleg csak a közönséges távirógépeknél alkalmazzák, adatfeldolgozásnál nem. A szalagtovábbitást itt a szinkronlyukakhoz kapcsolódó fogaskerék (ugynevezett tükerék) végzi.

A gyorsmüködésü olvasóknál a nagy szalagsebesség (1-4 m/sec) miatt a szinkronlyukak nem alkalmasak a szalagtovábbitásra. Ezért a továbbitást a szalaghoz nyomódó surlódógörgők végzik itt. Viszont ilyenkor a szinkronlyuksor "kiolvasásra" kerül, és a kiolvasott szinkronlyukak alapján várja (kapuzza) a berendezés a karaktereket jelentő lyukakat, és ezekkel szinkronban fogja azokat továbbitani is.

Müködésük alapján a lyukszalagolvasók két fajtája ismeretes:

a) start-stop üzemüek

b) szervó rendszerüek

A start-stop üzemü olvasók jellemző tulajdonsága, hogy bármely két karakter között is meg tudnak állni, még a maximális sebességü olvasás esetén is. Ehhez már speciális továbbitó és fékező mechanizmusra van szükség, mivel a karakterek közti távolság, amin belül meg kell tudni állitani a szalagot, vagy fel kell tudni gyorsitani, csak 2,54 mm. Sebességük max. 1000-1500 karakter/sec.
Az 1500 karakter/sec sebesség körüli olvasás esetén azonban nem minden esetben sikerül az adott osztástávolságon belül a megállás. Hogy az olvasófej előtt ilyenkor elhaladó, továbbfutó karakterek ne vesszenek kárba, egy néhány karakter befogadására alkalmas átmeneti tárat, puffertárat alkalmaznak. Az olvasás befejezését jelentő utasitás után az esetleg kiolvasásra kerülő karakterek ide kerülnek, itt kerülnek átmeneti tárolásra.

A szervó rendszerü olvasás esetén a szalag leállitása, illetve a kellő sebességre való felgyorsitása nem hirtelen, hanem fokozatosan történik. Az olvasás során a karakterek először egy néhány száz karakteres puffertárba kerülnek, ezt tölti fel folyamatosan az olvasó, és ennek tartalmát továbbitja a csatornarendszer felé. Az uj olvasási utasitás hatására először a puffertár tartalma kerül továbbitásra a rendszer felé és csak a szalag megfelelő sebességre való felgyorsulása után folytatódik az olvasás a szalagról. A szervó rendszerü hajtás a szalagot nem teszi ki nagy mechanikai igénybevételnek (a folyamatos lassitás, gyorsitás miatt), igy még ragasztott szalagok olvasását is lehetővé teszi. A néhány száz karakteres puffertár lehetővé teszi az olvasási sebesség megnövelését, akár 2000 karakter/sec fölé is. Jellemző sebességük 1500-2000 karakter/sec.

Az olvasás a számitógéptől kapott utasitás szerint történik. Az olvasás történhet egy adott jelig (ugynevezett határjel). Ilyenkor az operativ tár egy adott cimétől folyamatosan beirásra kerülnek a beolvasott adatok.

A határjel érzékelésekor az adatcsere befejeződik. Az uj olvasás uj utasitás hatására kezdődik és megint csak a következő határjelig tart. Az operativ tárba beolvasandó adattömb kezdő cimét az utasitásban kell megadni.

Más esetben a beolvasási utasitásban az operativ tárba kerülő adattömeg kezdő cimét és a hosszát adják meg. Például a 93-as cimtől kezdve 5 db rekeszt kell információval megtölteni. Az adatbevitel befejeződik, ha az 5 db rekesz megtelt. Egyes esetekben lehetőség van egy karakter beolvasására is egyetlen utasitással. A beolvasott információ a paritáscsatorna tartalma alapján ellenőrizhető. Az oszloponként olvasott lyukak páros vagy páratlan számát a paritáscsatornán található paritásbittel lehet összehasonlitani (parity check).

A berendezésnek az üzemképessége pedig az oszlopok közötti sötét részek "olvasásával" figyelhető. Ez az ugynevezett sötét-próba. Lényegét tekintve a lyukkártyaolvasóban alkalmazott sötét-világos ellenőrzéshez hasonlit.

ESZ-6022 Lyukszalagolvasó

Az információ olvasása elvégezhető másoló üzemmódban vagy kódátalakitó üzemmódban (az utóbbi esetben csak 8 csatornás lyukszalagról). A másoló üzemmódban az információ ellenőrizhető páros paritás, páratlan paritás szerint vagy az ellenőrzés kikapcsolható.

A berendezés mind multiplex, mind monopol üzemmódban dolgozhat. A szalagolvasó a központi egységhez a multiplex vagy a szelektor csatornákon keresztül illeszthető.

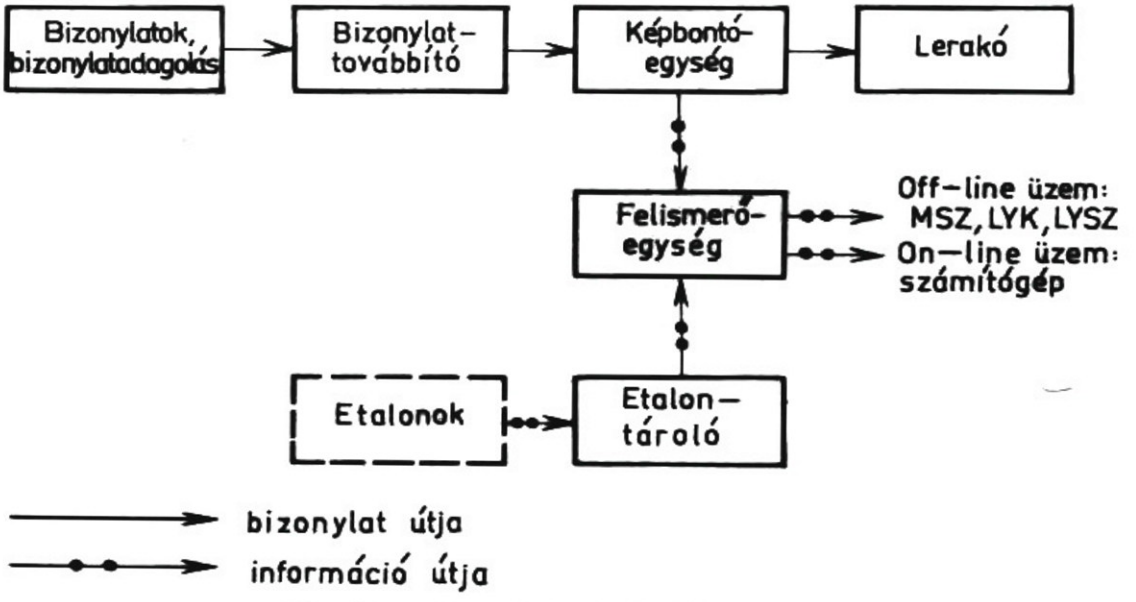
Max. olvasási sebesség (karakter/sec)	1500
Az információhordozó lyukszalag	5-8 csatornás
Olvasási elv	fotodiódás
A szalag tulfutása teljes sebességen végrehajtott leállás után (mm)	1,5

4.2.3 BIZONYLATOLVASÓK

Az eddig emlitett input perifériáknak volt egy közös hibájuk, mégpedig az, hogy külön adathordozóra volt szükségük. A különböző helyeken más-más céllal kitöltött elsődleges adathordozók (bizonylatok, számlák) tartalmát még a beolvasás előtt egy közbenső adathordozóra át kellett irni, mert csak ezt voltak képesek feldolgozni, erről tudtak csak beolvasni. Ennek kiküszöbölésére alkották meg az ugynevezett bizonylatolvasókat, amelyek szövegeket, adatokat képesek közvetlenül a számitógépbe olvasni. Egyes tipusok az olvasott jelek alapján még bizonylat rendezésre is alkalmasak. Ennek ellenére nagyobb arányu elterjedésüknek komoly akadálya, hogy maga a berendezés meglehetősen drága. A bizonylat anyaga csak jó minőségü papir lehet, és azok is drágák. Gyürött, elmosódott irásu, vagy pontatlanul kitöltött lapok olvasására a berendezések nem igen alkalmasak. Általában csak kötött formátumu bizonylatok feldolgozása lehetséges és speciális, erősen stilizált formáju karaktereket tudnak csak felismerni. Ez a speciális irás megfelelő irógéppel, összeadógéppel vagy valamilyen nyomtatási eljárással biztositható, ha az irószerkezete a megfelelő karaktereket tartalmazza, vagy arra kicserélhető (pl. gömbfej cseréjével, betühenger cseréjével - ld. később). Felhasználásuk elsősorban a nagy tömegü adatok feldolgozásánál akkor indokolt, ha a bizonylatokat a gépen kivül embereknek is fel kell dolgozni, vagyis, ha kihasználható, hogy az ilyen tipusu adathordozók (bizonylatok) ember és gép által egyaránt olvashatók.

A jelek "olvasása" két fő lépésben történik (111. ábra).

- A bizonylat adott, kijelölt helyén levő karaktereket valamilyen módon (általában részeire bontva, pontonként) érzékeli a berendezés és ennek megfelelően villamos jeleket állit elő (képfelbontó egység).
- A kapott villamos jeleket összehasonlitja a gép által felismerhető karakterek etalon villamos jeleivel. A megfeleltetés alapján dönti el a pillanatnyi karakter tipusát (felismerő egység).



111. ábra. Bizonylatolvasók elvi felépitése

Annak alapján, hogy a jeleket milyen módon érzékelik, a bizonylatolvasókat két csoportba sorolják:

- mágneses bizonylatolvasók és
- optikai bizonylatolvasók

4.2.3.1 Mágnesesen irt bizonylatok (MICR-Magnetic Ink Character Recognation)

A bizonylatra a nyomtatás vagy az irás mágnesezhető (vasoxidtartalmu) festékkel történik. Közvetlenül az olvasás előtt ezeket a bizonylatokat felmágnesezik, hogy az érzékelő egység a karaktereket alkotó mágneses vonalak nagyságából és helyzetéből meg tudja állapitani a karakter fajtáját. Az olvasófejekben a mágneses vonalak által keltett villamos impulzusok száma és alakja karakterenként más és más. Az ilyen módon történő olvasás a papir szennyeződésére, gyürődésére kevésbé érzékeny, viszont a papir anyaga nem tartalmazhat vasat. A speciális irásmód a hamisitásokkal szemben jó védelem, ugyanakkor a módszer nagy hátránya , hogy a "mágneses jeleket" a bizonylatnak csak előre meghatározott helyeire lehet elhelyezni. A berendezések egy lépésben a bizonylatnak csak egy-két sorát tudják "elolvasni". Ez, és az iróberendezések bonyolultsága, nagy ára, akadályozza széles körü elterjedésüket. Ezeknek a bizonylatoknak az alkalmazását és elterjedését a nagyforgalmu bankok kezdeményezték, és fő alkalmazási területük a csekkforgalom, számlák nyilvántartása, rendezése. Ennek megfelelően az ilyen módon való irás jelkészlete meglehetősen szegényes, általában csak a számjegyekre és néhány, még szükséges speciális karakterre korlátozódik. A használatos irásmódok közül az alábbiakban bemutatunk néhányat.

SIEMAG

Ennek érdekessége, hogy külön, más módon biztositja az embereknek az olvashatóságot és másképpen a gépi felismerhetőséget. Az előbbi célra szolgálnak tulajdonképpen a normál alaku számjegyek, mig az utóbbit a számjegyek alatt és felett külön elhelyezett ugynevezett vonáskódok biztositják (112. ábra). Az összesen 6 darab vonáskód eloszlása és sürüsége biztositja a gépnek a jelkészlet tiz darab számjegyének egyértelmü felismerhetőségét.

112. ábra. Siemag tipusu mágnesesen olvasható számjegyek

E13-B

A két célu olvasást ugyanaz a nagymértékben stilizált karakter biztositja. A tiz számjegy és az ábrán fel nem tüntetett négy különleges karakter felismerhetősége a pontosan rögzitett és eltérő vonalvastagság alkalmazásán alapszik (113.ábra). Ez a fajta mágneses irás az USA-ban használatos.

0123456789

113. ábra. E13-B tipusu mágnesesen olvasható számjegyek

CMC-7

A jelkészlet tiz számjegyet, öt különleges karaktert, és betüket is tartalmaz. Minden karakter a ferrittartalmu festékkel huzott hét darab függőleges vonalból áll (114. ábra). Ezek mindegyike maximum két helyen szakitható meg. A vonalak helyzete, a megszakitások száma és helye teszi lehetővé a gép számára a felismerést. A mágneses irásnak ez a fajtája terjedt el Európában is.

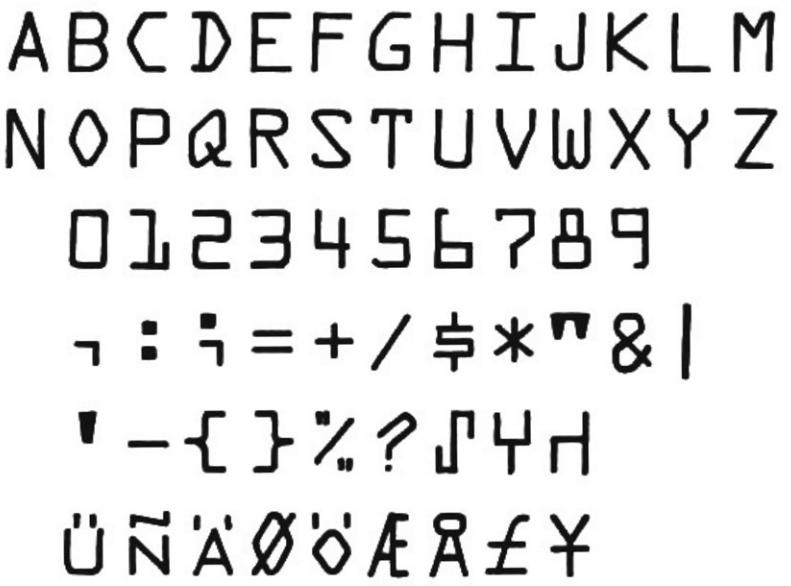
114. ábra. CMC-7 tipusu mágnesesen olvasható számjegyek

4.2.3.2 Optikai bizonylatolvasás (OCR-Optical Character Recognation)

A mágnestintás adatrögzités bár látszólag megoldotta az elsődleges bizonylatok és a gép által feldolgozható adathordozók kettősségét, néhány járulékos problémát okozott. Ezeket főleg a speciális iróberendezés, a mágnesezhető festék, a vasmentes papir, a viszonylag szük karakterkészlet okozta. Maguk az olvasóberendezések is általában csak két sornyi információt voltak képesek egyszerre feldolgozni. Az optikai karakterfelismerés (OCR) ezeket a hátrányokat igyekszik megszüntetni. Nincs szükség speciális iróberendezésre, s bár az ilyen jelkészlettel is rendelkező iró- vagy nyomtatógépek ára elég magas, cserélhető karakterkészlet segitségével a hagyományos adatfeldolgozó vagy adatrögzitő gépek is alkalmassá tehetők ilyen célra. Nem kell az iráshoz mágnesezhető tinta, nem kell vasmentes papir, viszont a papir gyürődésére, a piszkolódásra sokkal érzékenyebbek. Az ilyen módon olvasható bizonylatok mérete is szélesebb skálán mozog. Az egy-két sort olvasó berendezéstől kezdve vannak olyanok is, amelyek egész oldalakat (A4) képesek feldolgozni. A karakterkészletük kiterjed szinte a gyakorlatban használatos összes jelre, bár ezek is általában stilizáltak. Erre azért van szükség, hogy egyrészt a berendezés árát mérsékelni lehessen, másrészt a gép felismerőképességének biztonságát növelje. (Persze a stilizálás csak addig mehet, amig az olvashatóságot nem rontja!)

Nagyon sok ilyen karakterkészlet alakult ki, szinte gyártónként más és más, és ez részben még most is erősen bonyolitja a helyzetet. Az egységesitési, szabványositási törekvések eredményeképpen született meg az un. OCR-A és OCR-B jelkészlet.

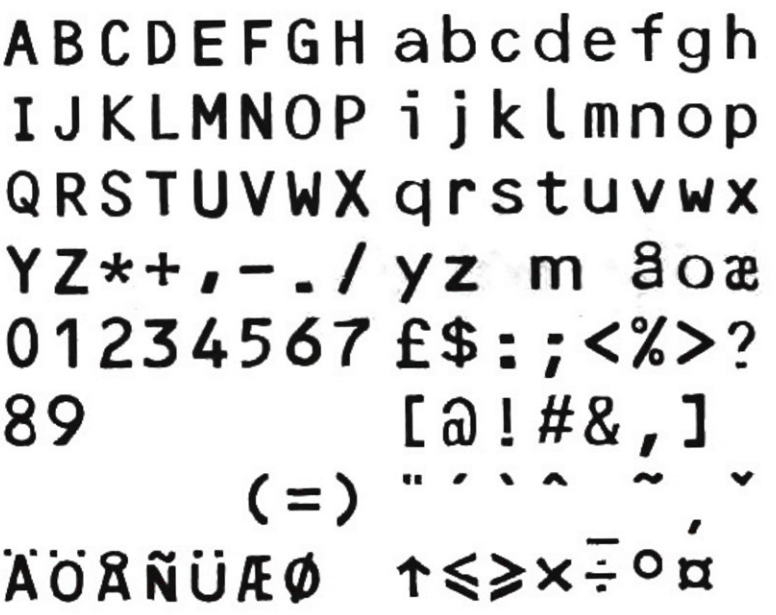
Az OCR-A esetében a karaktereket egy 5 x 9-es mátrixba kell képzelni, ahol a háló egyes pontjait összekötve jutunk el a megfelelő jelalakig. Láthatóan még erősen stilizált jeleket tartalmaz, de ezzel a könnyen összetéveszthető karakterek (O, Q, nulla) biztonságos felismerését is sikerült megoldani (115. ábra). Bár kidolgozását az Egyesült Államok Szabványügyi Hivatala végezte, nagyon sok irodagépgyártó cég átvette, és alkalmazza.



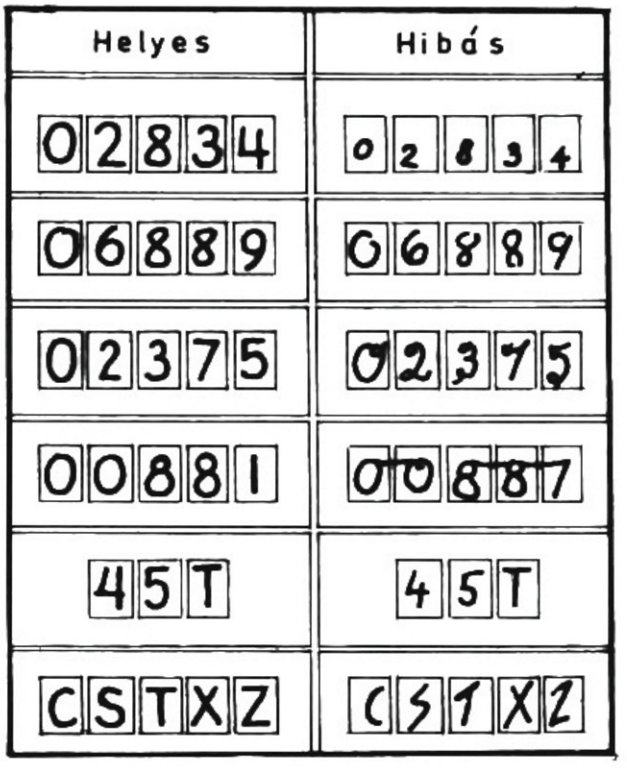
115. ábra. OCR-A tipusu
optikailag olvasható karakterkészlet

Az OCR-B irás jelkészlete már nem stilizált elemekből áll (116. ábra). A kis-, nagybetükön, számokon kivül a különleges karakterek nagyon bő választékát nyujtja. A Nemzetközi Szabványügyi Hivatal (ISO) ajánlásában ECMA-B néven ez a karakterkészlet szerepel. A legujabb eredmények már lehetővé teszik a kézirás felismerését is. Miután az optikai jelolvasók meglehetősen szigoru igényeket támasztanak a karakterek formájával és egymástól való távolságával szemben, ez a kézirás még csak előre meghatározott helyen, karakterenként kijelölt kockában, kötött formában történhet (117. ábra). A jeleket ugy kell irni, hogy minél inkább hasonlitsanak a nyomtatott jelhez (bár nem kell őket kimondottan rajzolni!)

Most röviden megemlitünk néhány képfelbontási módszert, amelyek segitségével a felismerendő jelet először elemeire, majd azok alapján villamos jellé lehet átalakitani.



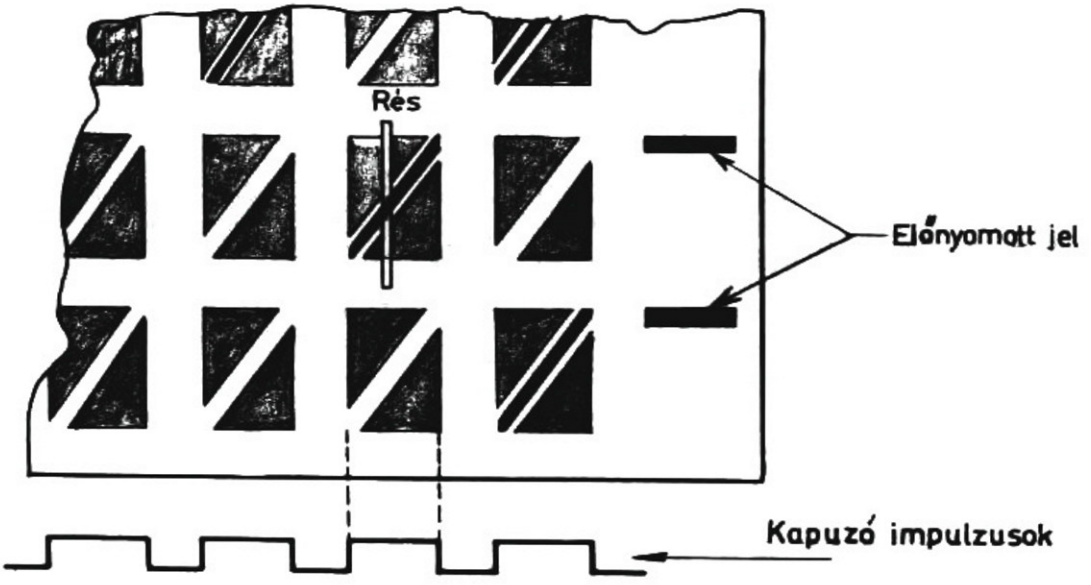
116. ábra. OCR-B (ECMA-B) tipusu
optikailag olvasható karakterkészlet



117. ábra.
Optikailag olvasható kötött kézirás szabályai

1. Vonalas képfelbontás

Ez a tulajdonképpen legegyszerübb módszer egy fényforrásból és egy nagyon keskeny résből áll, amelyen keresztül a fény a "képre", a bizonylat meghatározott helyeire huzható vonalakra jut és onnan visszaverődik egy fényérzékelő elemre. A bizonylat egy-egy sorában egymás után elhelyezkedő és felismerendő jelek a rés előtt haladnak el egyenletes sebességgel. Ahol a papir a "résen át nézve" valamilyen karakter egy pontját tartalmazza, ott a fény sokkal kevésbé verődik vissza, mint ahol a fehér papir van (118. ábra). Ily módon a képnek a rés előtt történő elhaladása közben a visszavert fény nagysága változik. Ez a müködési mód a soros képbontás tipikus esete, ahol a kép egyes részeinek megfelelő villamos jeleket sorban egymás után küldjük a felismerő egységhez.



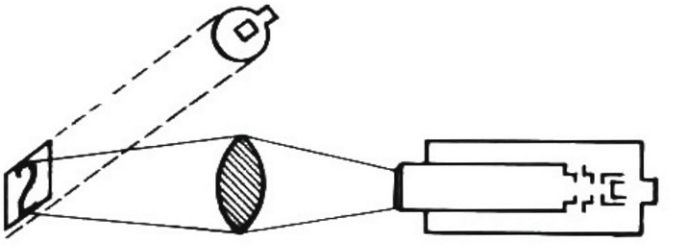
118. ábra. Optikailag vonalas képfelbontással olvasható bizonylat egy része, a megfelelő helyekre huzott ferde vonalakkal

2. Sugárnyalábos képfelbontás

A másképp katódsugarasnak is nevezett eljárás lényege, hogy egy "tühegy vékonyságu" fénynyaláb (katódsugár) vizszintes vagy függőleges irányban a leolvasandó jel vagy a bizonylat teljes területét végigpásztázza. A bizonylat tehát áll, a fénysugár pedig mozog. A bizonylatról visszaverődött és más-más fényérzékelő által villamos jellé alakitott fény alapján lehet a karaktereket érzékelni. (A fényérzékelő elemek egy raszterhálózat pontjaiban vannak.) Bár soros képbontás ez is, sokkal gyorsabb az előzőnél. A fénysugár kis felülete (~ Ø 0,05 mm) pedig lehetővé teszi az igen finom részletekre való felbontást is. Jelenleg ilyen módszerrel minden karaktert max. 1200 raszterpontra lehet felbontani.

3. Videografikus képfelbontás

A bizonylat egy részének vagy teljes egészének képe a TV technikában használatos képfelbontó csőre (vidikonra) kerül (119. ábra). Az itt megjelenő képet egy elektronsugár tapogatja le sorról sorra. A módszer előnye nemcsak a nagy gyorsaság és a nagy felbontóképesség, hanem az, hogy akár az egész bizonylatot egy lépésben "le lehet olvasni", nem kell karakterenként továbbmozgatni.

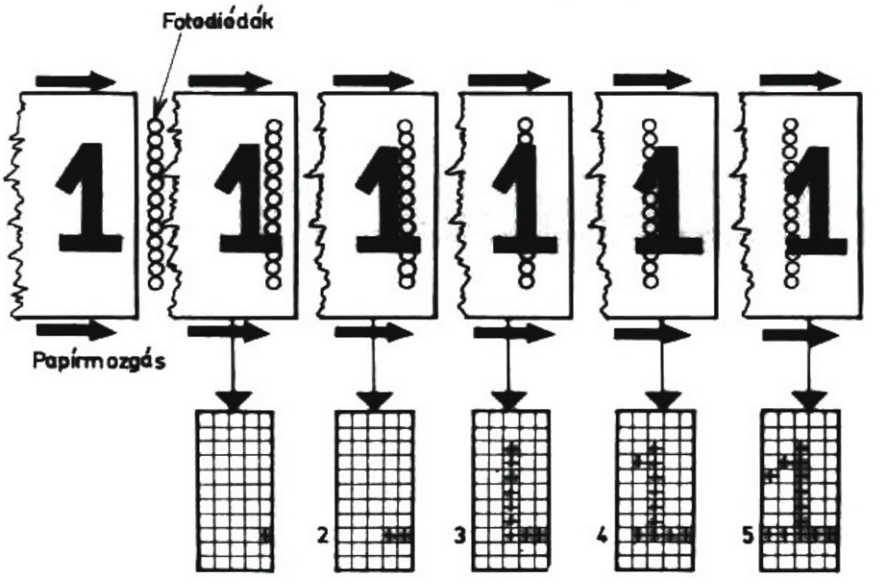


119. ábra. Videografikus képfelbontás elve

4. Osztott-vonalas képfelbontás

Az elve a vonalas felbontáshoz hasonlitható, de a visszaverődött fény nem egy, hanem például egymás alatt elhelyezkedő, több fényérzékelő elemre kerül (120. ábra). A jel tartalmától, "fekete pontjaitól" függően az egyes fényérzékelő elemek más és más helyen jeleznek "sötétet". Ezzel a módszerrel már nemcsak géppel készitett irást lehet felbontani, hanem egyes esetekben már kézirást is.

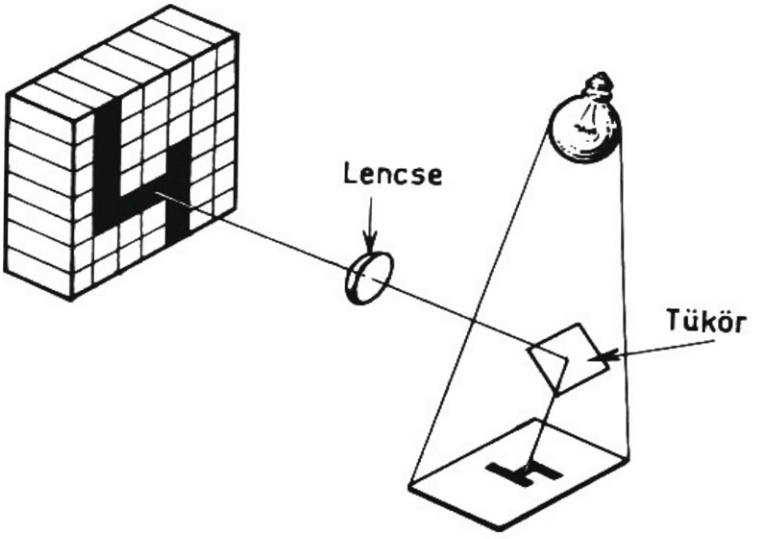
Ez a módszer már a párhuzamos képfelbontáshoz tartozik, mivel a jelet leiró információ egyidejüleg több csatornán (fényérzékelőnként) kerül a felismerő egység felé.



120. ábra. Osztott-vonalas képfelbontás müködési elve

5. Retinarendszerü képfelbontás

Ez a legteljesebben párhuzamos müködésü felbontás. Az olvasandó karakter képe egy optikai rendszeren keresztül egyszerre kerül az un. retinára, ami apró fényérzékelő elemek sokasága (121. ábra). Ennyiben a módszer hasonlit a sugárnyalábos felbontáshoz is. Az egyes fényérzékelő elemek a karakter képének megfelelően "sötétet" vagy "világosat" jeleznek. Az ennek megfelelő villamos jelek egy un. regisztermátrixba kerülnek, ami az összes fényérzékelő elem jelét egyidőben fogja tartalmazni.



121. ábra. Retinarendszerü (párhuzamos) képfelbontás elve

Ezeknek a képfelbontási eljárásoknak az "eredményét", a villamos jelsorozatot kell összehasonlitani a gép által felismerhető karakterek etalon jeleivel. Ha valamelyik etalonnal sikerül egyezést találni, akkor a karakter felismerhető volt. Ha nincs egyezés, akkor a készülék nem ismerte fel a karaktert, mert vagy nem megfelelően lett irva, vagy mert ilyen karaktert nem tárolt etalonként. Az összehasonlitásra alkalmas eljárások közül néhány:

Analóg felismerés: A karaktereknek megfelelő etalon jeleket (az idő függvényében változó feszültségek) a letapogatás során kapott jelekkel hasonlitják össze.

Vonalas felismerés: A felismerendő karakterek vizszintes és függőleges vonalainak száma és helyzete alapján történik a megfeleltetés. Csak egyenesekkel leirt jeleket lehet ilyen módon felismerni!

Optikai (maszkos) felismerés: A karakterképek pozitiv és negativ képét (maszkját) is tárolni kell. A letapogatott pozitiv képet a negativval, a negativ képet a pozitivval kell fedésbe hozni. Egyezés esetén a maszkok nem engednek fényt a mögöttük elhelyezkedő fényérzékelőkre. Ez a módszer csak elvileg egyszerü, mert a gyakorlatban a hasonló betük (O, Q) és az esetleg változó vonalvastagság nagyon sok problémát jelent.

Mátrixos felismerés: Az etalon karakter "képét" egy mátrix (ferrit v. ellenállás) őrzi. Ezt kell összehasonlitani az olvasott és felbontott karakterképpel. Ferritmátrix esetén a tárolt karakterkép könnyen változtatható, cserélhető.

Konturos felismerés: Az etalon karakterek helyett jellegzetes karakterelemek kerülnek tárolásra (csucsok, elágazások, találkozási pont-tipusok, különböző görbületi sugarak stb.). Az olvasásnál a karakter szintén ilyen elemekre lesz felbontva, és a felismerés ezek egyezése alapján történhet. Ez a legfejlettebb módszer, ez teszi lehetővé a kézirás felismerését is.

4.3 KÖTEGELT FELDOLGOZÁS OUTPUT ESZKÖZEI

A számitások eredményeit, az adatokat vagy a tárban levő bármilyen információt a kimeneti (output) egységeken keresztül továbbitja a rendszer a felhasználó felé. A kötegelt feldolgozásnál használatos kimeneti egységek:

- sornyomtató,
- lyukkártyalyukasztó,
- lyukszalaglyukasztó,
- rajzgép (plotter).

4.3.1 SORNYOMTATÓ

A számitógépek müködési sebességével összhangban a nagy tömegü adatokat, eredményeket rövid idő alatt ki kell hozni a rendszerből. A felhasználó számára különösen előnyös, ha a rögzitett adatok közvetlenül is olvashatók. Gyakran az eredményekről, táblázatokról nem is egy, hanem több példányos másolatra van szükség. Ezeket a feladatokat a sornyomtató látja el. Eltérően az irógépektől, egy sor betüit nem egymás után sorban, hanem az irásszélességnek megfelelően szinte egyidőben rögziti papirra.

A sornyomtatókhoz különleges papirt használnak. A folyamatos feldolgozást károsan befolyásolná, ha a gyors nyomtatási sebesség (100-2000 sor/perc) miatt gyakran kéne ujabb papirköteget rakni a gépbe, ezért "végtelenitett" papirt használnak. Ezt az olykor több száz méteres papircsikot összehajtható albumhoz hasonlóan lapokra hajtogatják és igy keletkezik az ugynevezett leporelló forma (100 darab ilyen lap még könnyen kezelhető). A cikkcakkban hajtogatott papir minden hajtásánál perforáció szolgál az alaktartásra és arra, hogy a kinyomtatás után ne csak géppel, hanem kézzel is könnyen szét lehessen lapokra választani. A lapok magassága (a hajtogatási távolság) 105 és 297 mm között változik. Egy lapon általában 60-72 sor szokott lenni, mivel az irássürüség szokásos értéke 6 vagy 8 sor/inch. A papir szélessége is többféle lehet, 80 és 420 mm között. A papir továbbitása a két szélén levő perforáció segitségével történik (körmöskerék). Az egy sorba kinyomtatható karakterek száma természetesen függ a berendezéstől is, a maximumon belül pedig a használt papir szélessége befolyásolja. Ennek megfelelően a szokásos iráspoziciók száma 96, 100, 120, 132, 160. Elterjedt érték a 120 vagy a 132 karakter/sor.

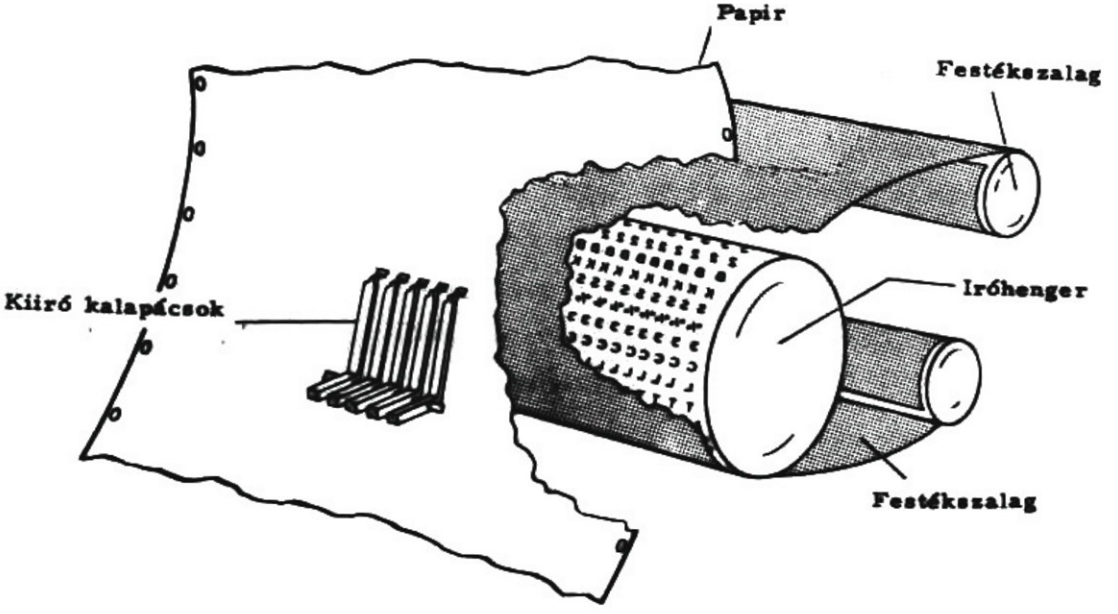
Általános célu felhasználásoknál a sornyomtató papirját "csikozzák". A csikok a sorok könnyü megkülönböztethetőségét szolgálják, az egy sorban levő karakterek összetartozását mutatják.

Speciális célra egyre több helyen használják az előrenyomtatott formanyomtatványt is.

A forgalomban levő sornyomtatók legjellemzőbb tulajdonsága a nyomtatási sebesség. A különféle tipusok értékelésének is ez a döntő szempontja. A ma általánosan alkalmazott tipusoknál az 1200-1500 sor/perc. Az 500 sor/perc érték feletti tipusokra a gyorsnyomtató elnevezést is használják.

A sornyomtatók egyik legelterjedtebb tipusánál az ugynevezett dobrendszerü vagy betühengeres nyomtatónál a karakterkészlet "betütestjei" egy tárcsákból álló forgó henger felületén helyezkednek el (122. ábra). A henger kerületén az egész karakterkészlet (pl. 64 karakter) megtalálható, és minden egyes karakter annyiszor ismétlődik a henger alkotója mentén, ahány pozicióban történik az irás (120, 128, 132 stb.). Minden pozicióhoz egy-egy kalapács (ugynevezett iró- vagy nyomtatókalapács) tartozik, ezek végzik a tényleges nyomtatást. A henger és a kalapácsok között van a festékszalag és a papir, amire a kiirás történik (123. ábra). Az egyes irópoziciókhoz tartozó kalapácsok csak akkor müködnek, ha a papir előtt a hengeren épp az a karakter van, amelyet arra a pozicióra nyomtatni akarunk. Ebből következik, hogy a sor azonos karaktereit egyszerre, egy időben irja ki a gép.

122. ábra.
Karakterkészlet elhelyezkedése
a dobrendszerü nyomtatónál

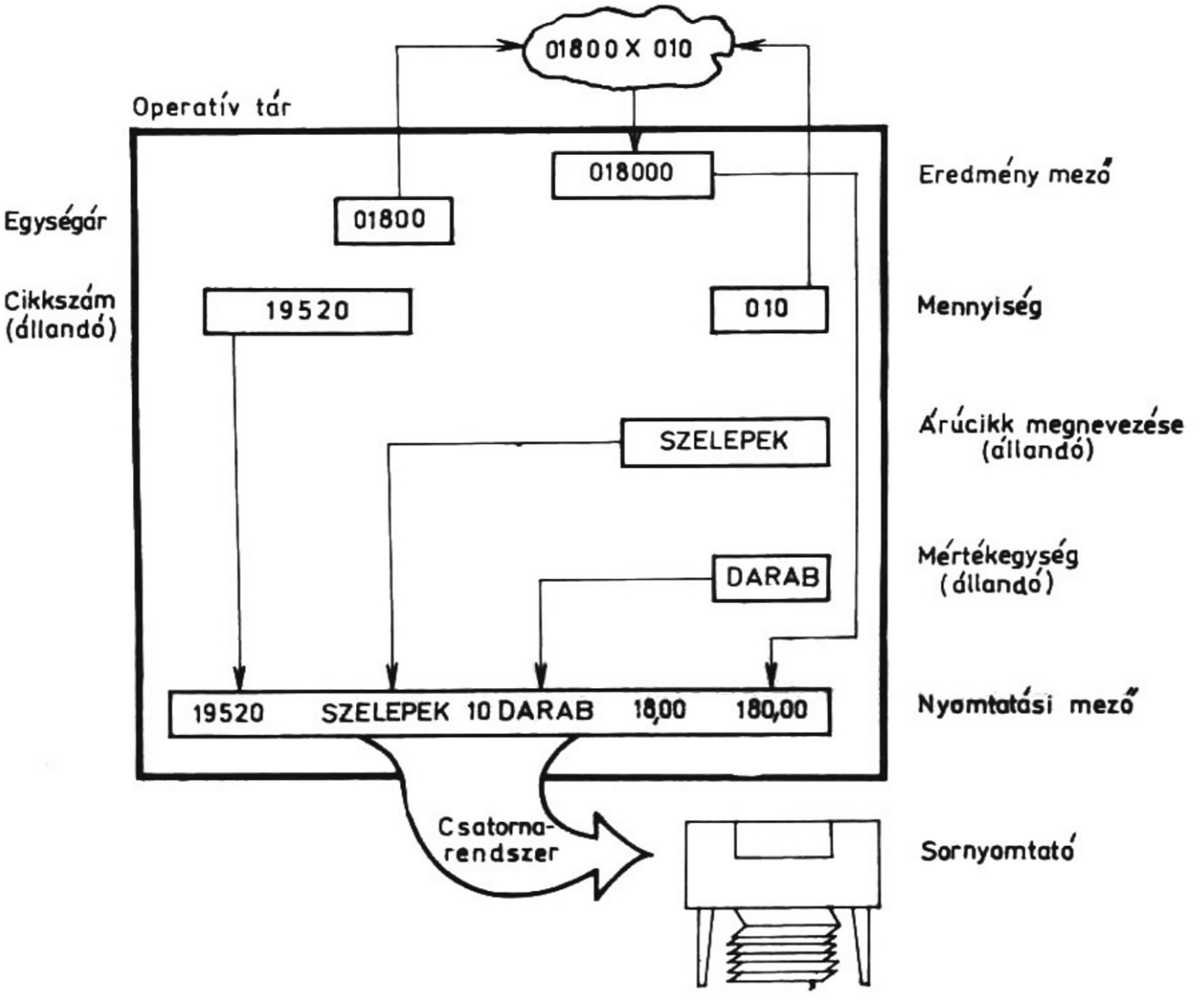


123. ábra. A dobrendszerü nyomtatás elve

Amig a nyomtató be van kapcsolva, a betühenger állandóan nagy sebességgel forog. Egy sor kinyomtatásához maximum egy hengerfordulatra van szükség, mivel a karakterkészlet ez után megint ismétlődni fog. A gyors forgás miatt ez az egy hengerfordulat olyan kis idő, hogy a sor összes karakterének kinyomtatása látszólag egyszerre történik. A valóságban viszont a különféle karaktereket egymás után irja ki a gép, és csak az azonos karaktereket egyszerre. Azokat a dobrendszerü nyomtatókat, amelyeknél minden uj sor nyomtatása a karakterkészlet adott jelétől (pl. A betűtől) kezdődhet csak, szinkron müködésüeknek nevezik. Amelyeknél viszont a soremelés után az uj sor nyomtatása bármelyik karakternél kezdődhet, azok az ugynevezett aszinkron müködésüek. Ha az output berendezés felhasználójának a lehetségesnél kisebb karakterkészletre van szüksége, akkor a henger kerülete mentén az akár többször is ismétlődhet, vagy csak egyes, gyakrabban használt karaktereket ismételnek meg több alkotó mentén is. Ezzel a maximális várakozási idő, amely addig tart, amig a forgó hengeren a kivánt karakter a nyomtatókalapácsok elé nem ér, jelentősen csökkenthető. Most nézzük meg, hogyan jön létre a következő kinyomtatott sor!

19 520 SZELEPEK 10 DARAB 18,00 180,00

A 124. ábrán az egyszerüség kedvéért csak az operativ tár tartalmát ábrázoltuk és nem tüntettük fel a müveletvégzésben részt vevő aritmetikai, vezérlő stb. egységeket.



124. ábra. Egy sor kinyomtatásának előkészitése

A kinyomtatásra szánt adatokat az operativ tárban olyan formátumra kell rendezni, ahogy azoknak a papiron meg kell jelenni. Azaz minden betü, minden szám, sőt az irásjelek (pl. tizedesvessző) EBCDIC kódját is a kinyomtatásnak megfelelő pozicióba kell helyezni. Azokon a helyeken pedig, amelyek a papiron majd üresen fognak maradni, az ugynevezett üres (blank) karakternek megfelelő $40_{(16)}$-et kell tárolni. Igy tehát e sor képét először elő kell állitani a tárban, vagyis az összes pozicióba a kinyomtatandó karaktereknek megfelelő hexadecimális kódot kell irni (pontosabban iratni) ! A sornyomtatón kiiratható karaktereknek megfelelő EBCDIC kódokat az alábbi, 11. táblázat tartalmazza.

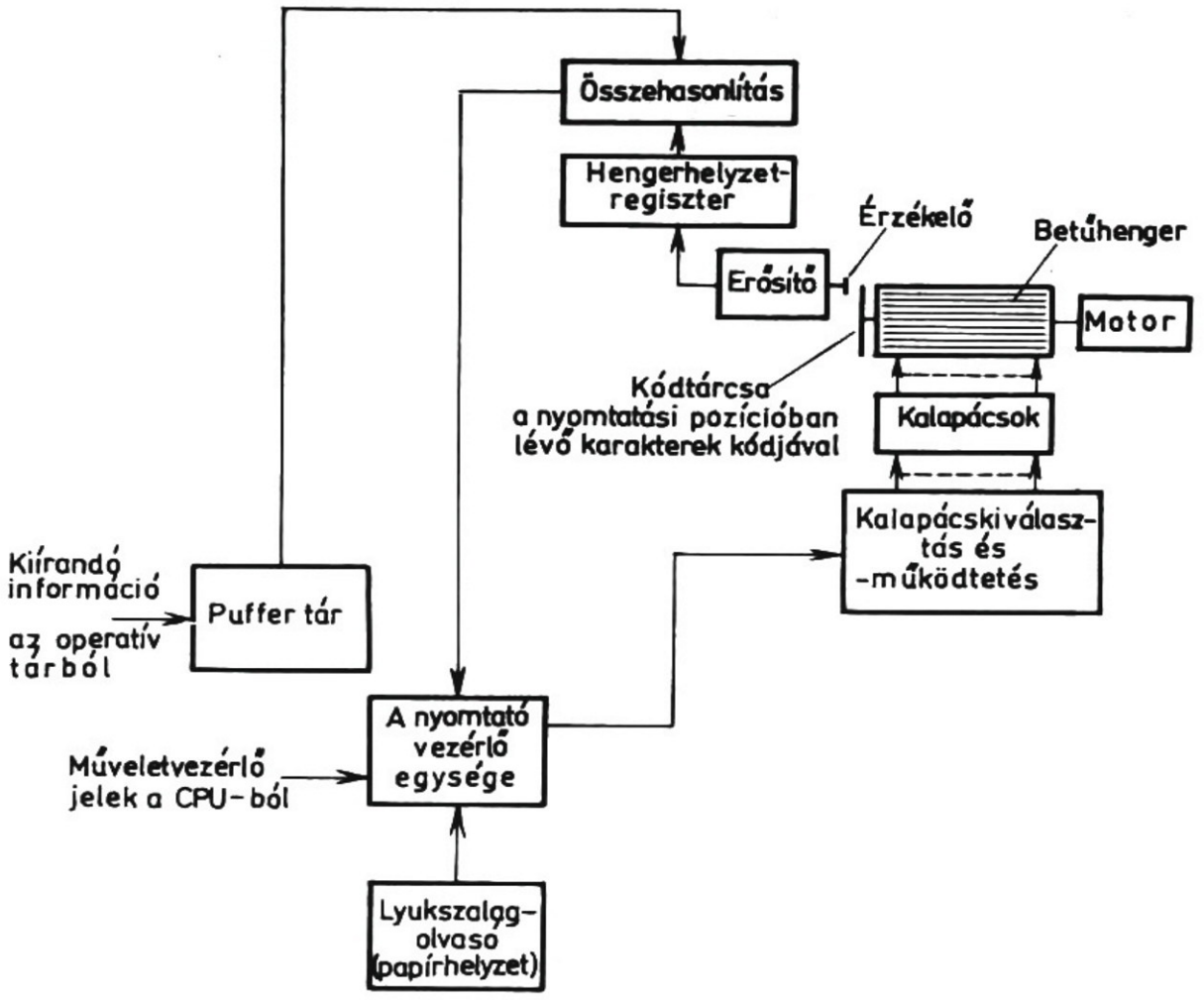
11. táblázat

Sornyomtatón kiiratható karakterek EBCDIC kódjai

Karakter	EBCDIC kód (hex.)	Karakter	EBCDIC kód (hex.)	Karakter	EBCDIC kód (hex.)
szóköz	4Ø	.	7A	Q	D8
¢	4A	#	7B	R	D9
.	4B	@	7C	S	E2
<	4C	,	7D	T	E3
(	4D	=	7E	U	E4
+	4E	"	7F	V	E5
|	4F	A	C1	W	E6
&	5Ø	B	C2	X	E7
|	5A	C	C3	Y	E8
!	5B	D	C4	Z	E9
*	5C	E	C5	Ø	FØ
)	5D	F	C6	1	F1
;	5E	G	C7	2	F2
¬	5F	H	C8	3	F3
minusz	6Ø	I	C9	4	F4
/	61	J	D1	5	F5
Λ	6A	K	D2	6	F6
,	6B	L	D3	7	F7
%	6C	M	D4	8	F8
_	6D	N	D5	9	F9
>	6E	O	D6		
?	6F	P	D7		

A program megfelelő utasitásának hatására ez az előkészitett karaktersorozat az operativ tárból a csatornarendszeren keresztül átkerül a nyomtató egység puffertárába, átmeneti tárába. Itt tárolódik addig, amig a nyomtatás be nem fejeződik.

A 125. ábrán egy dobrendszerü nyomtató blokkvázlata látható. A hengert a motor állandó fordulatszámmal forgatja. A kódtárcsa a betühengerrel együtt forog, az érzékelő (fotodiódák) által leolvasott kód megegyezik az éppen a kalapácsok előtt tartózkodó jel kódjával. (A kódtárcsán levő kód- és szinkronjeleket általában fotoelektromos uton vagy mágneses jeladó segitségével érzékeli a berendezés.)



125. ábra. A sornyomtató működésének elvi vázlata

Minden karakternél, amely a kalapácsok előtt elhalad, a henger -helyzetregiszter tartalma összehasonlításra kerül a puffertár tartalmával - minden egyes pozicióban. Ha a nyomtatás 132 pozicion történik, akkor a 132 db nyomtatandó karaktert kell a kalapácsok előtt levő jellel összehasonlitani. Amely pozicion egyezés van, ott a kiirókalapácsokat működtetni kell. Ezt a nyomtató vezérlő egysége végzi.

Most nézzük meg, hogyan is történt az előbbi sorban a DARAB szó kiirása! Tételezzük fel, hogy ezt a szót a sor 41, 42, 43, 44, 45-ös pozicióira szándékozunk elhelyezni, és hogy éppen az A betü van a kalapácsok előtt. Ez a szóban két helyen is szerepel. A 42. és 44. pozicióhoz tartozó kalapácsokat is működtetni kell!

```
                A     A
1..........41.42.43.44.45.46...........132
```

A henger forog tovább, a B betü kerül a kalapácsok elé. Ennek a 45. pozicion kell megjelenni, tehát az összes kalapács közül csak ez fog működni.

```
                A     A  B
1..........41.42.43.44.45.46...........132
```

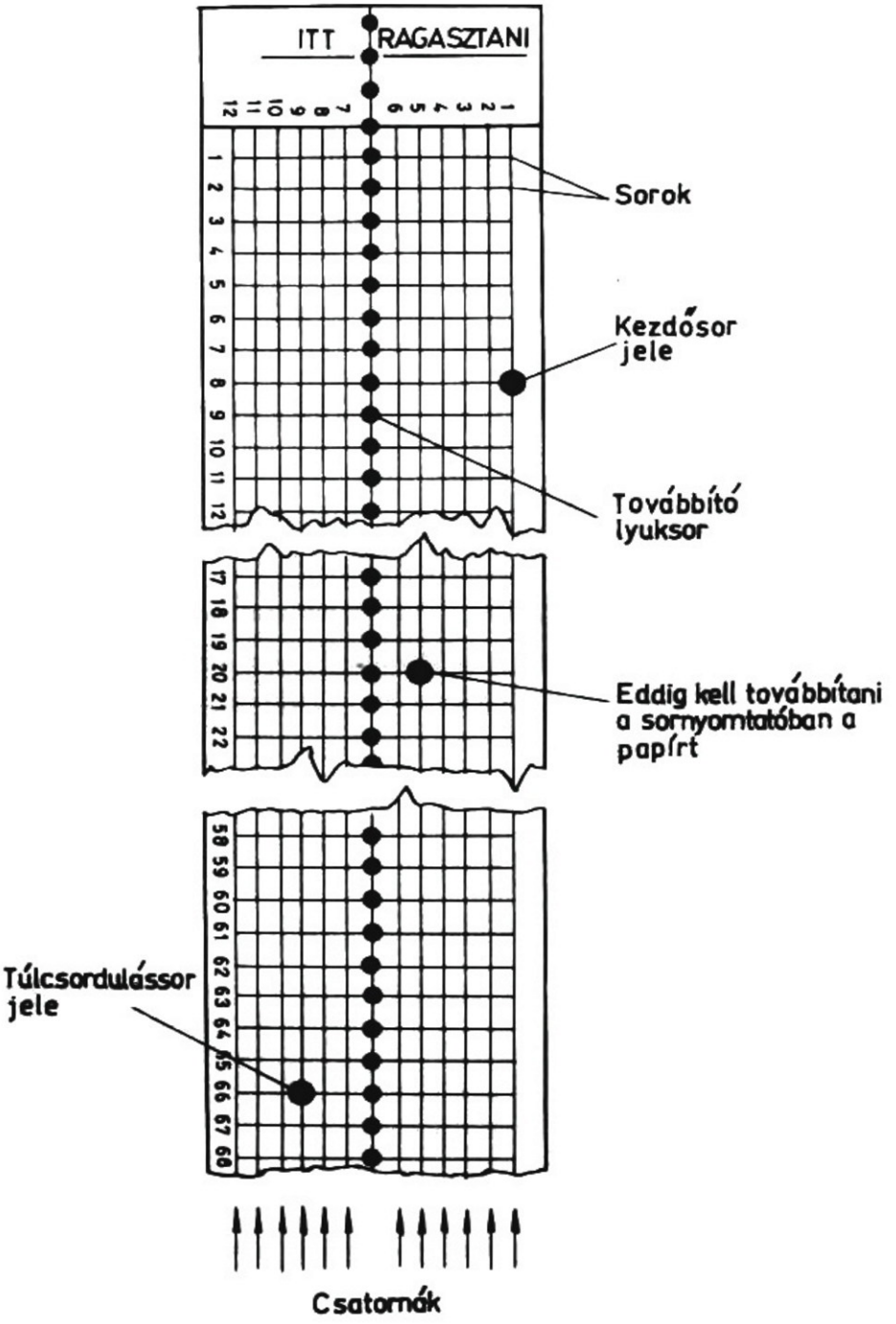
A következő karakter, a C betü sehol sem szerepel, tehát a kalapácsok sehol sem fognak működni. A D betüt a 41. pozicióra kell kiirni, tehát csak ez a kalapács fog dolgozni.

```
              D A     A  B
1..........41.42.43.44.45.46...........132
```

A henger következő karaktere, az E betű ebben a szóban nem szerepel, de a SZELEPEK szóban 3 helyen is előfordul. Tehát csak azok a kalapácsok fognak a papirra ütni. A DARAB szó kiirása csak akkor fog befejeződni, ha az R betü is a megfelelő helyzetbe kerül. De közben a SZELEPEK szóban az E-n kivül már a K, L, P betük is megjelentek.

A betük kinyomtatása után következnek a számjegyek, majd az irásjelek és igy a henger egy körülfordulása alatt elkészült a sor! A papirt a továbbitóberendezés mozgatja. Ezt vezérelheti a CPU a programban levő utasitás hatására, vagy ismétlődő táblázatok, forma-nyomtatványok készitésekor az ugynevezett format (vagy carriage) szalag. A program utasitása alapján a berendezés végezhet soremelést (1, 2, 3 sorral) vagy lapemelést (a következő lap elejére).

Ha a papirt a továbbitás után egy előre megadott helyzetbe akarjuk hozni, akkor ezt az ugynevezett format szalaggal végeztetjük el (126. ábra). Ez egy végtelenitett 8 vagy 12 csatornás lyukszalag fotoelektromos olvasással.



126. ábra. Format (carriage) szalag

A 126. ábrán feltüntetett helyre kell a lappal megegyező hosszuságu lyukszalag végét ragasztani és igy keletkezik a végtelenitett hurok. Ez a nyomtatóban levő papirral szinkron halad. A nyomtatás formáját megtervezve, a lyukszalag 12 csatornájának bármelyikében levő lyukkal megadhatjuk, hogy melyik sorig továbbitsa a berendezés a papirt, melyik sorba szabad már vagy még irni (pl. a lap elején, illetőleg a végén). A papir első soraiba rendszerint nem akarunk irni. Az első nyomtatott sort kezdősornak vagy első nyomtatási sornak nevezik. Ha tehát az ürlapra csak 7 üres sor után kell irni, akkor a 8.sor a kezdősor. Ezt a format szalag 1 csatornájába helyezett lyukkal jelöljük. Ha nyomtatás közben pl. a 20. sorhoz akarjuk továbbitani a papirt, akkor a szalag 12 csatornájának valamelyikében a 20. sornál lyukat kell tenni. Erre a célra az ábrán az 5. csatorna szolgál. Ahogy az ürlap elején is, ugy a végén is szokás néhány sort üresen hagyni. Meghatározzák tehát azt a sort, ahová még lehet irni. Ezt nevezik tulcsordulássornak. A 12. és 9. csatornák szolgálnak a sorkorlátozásra. Az ábrán a 9. csatornába helyezett lyuk jelezte az utolsó felhasználható sort. Ezeknek a soroknak a figyelése a program megfelelő utasitásával hajtható végre.

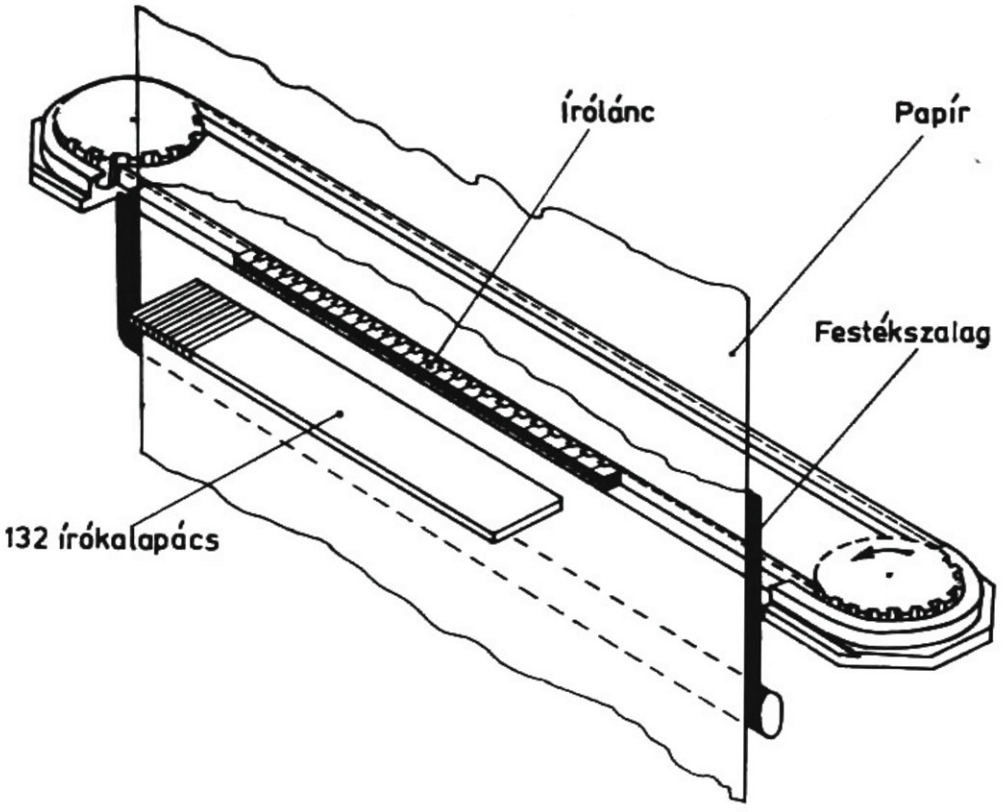
Vannak olyan sornyomtatók is (pl. ESZ rendszerben) ahol az illető csatornák bináris kódját kell a format szalagba lyukasztani. Igy kevesebb lyuksor is elegendő. Az előbbi példában szereplő 5. csatorna eszerint az első és harmadik sor egyidejü lyukasztásával adódik (a négy lyuksoros szalagon) - $0101_{(2)}$.

A dobrendszerü nyomtatóknál általában nehéz az egyenes sor biztositása, mivel a betühengeren levő jelek e sorra merőlegesen mozognak a henger forgómozgása miatt. Ha a kiiró kalapácsok "meghuzási ideje" nem azonos, akkor egy-egy pozicion előbb vagy utóbb történik meg a nyomtatás és ennek megfelelően lejjebb vagy feljebb csusznak a betük (127. ábra). Ezért a dobrendszerü nyomtatók ott használhatók előnyösen, ahol az egyes karaktereknek adott távolságra kell lenni, és a magassági eltolódás nem kritikus. Például a későbbiek során optikailag olvasandó (OCR) bizonylatoknál.
A dobrendszerü nyomtató másik hibája, hogy a karakterek általában csak a dob szétszerelésével együtt cserélhetők, csak igy lehet a karaktereket tartalmazó "tárcsákhoz" hozzáférni.

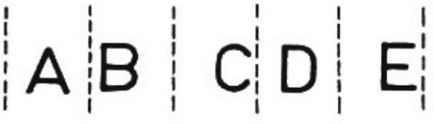
127. ábra.
Dobrendszerü nyomtatás tipikus iráshibája

Egy másik, nagyon gyakran használt nyomtatótipus, az ugynevezett láncrendszerü. Itt az egyes karakterek betütestjei egy láncra vannak felfüzve, és azok egy sor mentén, vizszintes sikban mozognak a nyomtatókalapácsok előtt (128. ábra). Ez a nyomtatási rendszer teljesen egyenes sort eredményez, viszont a kalapácsok eltérő repülési ideje a karakterek "osztástávolságát" változtatja meg (129. ábra). Az ugynevezett irólánc mentén a karakterkészlet általában többször ismétlődik, igy nincs szükség a lánc teljes körbefutására, hogy egy sor kinyomtatható legyen. Ha az egyes karakterkészletekben a betütestek a felhasználási gyakoriságnak megfelelően vannak sorba rakva, még tovább csökkenthető a sor kinyomtatásához szükséges idő (és igy a nyomtató sebessége tovább növekedhet).

A nyomtatási folyamat egyébként megegyezik a dobrendszerü nyomtatóéval: az adott pozicióra akkor történik nyomtatás, ha a pozicióhoz tartozó kalapács előtt épp a kinyomtatandó karakter betü-



128. ábra. Láncrendszerü nyomtatás elve



129. ábra. Lánc-, vonat-, irórudas rendszerü nyomtatás tipikus iráshibája

testje halad el. Ennek a rendszernek egy kissé módositott változatát használja az ugynevezett vonat-rendszerü nyomtató. A vizszintes sikban a kalapácsok előtt állandó sebességgel haladó betütestek itt nincsenek egy láncra felfüzve, nincsenek összekapcsolva, hanem mint egy vonat kocsijai, egy kényszerpálya mentén "tolják" egymást. A lánc- és vonatrendszerü nyomtatóknál a betütestek könnyen cserélhetők, igy a hibajavitás vagy a karaktercsere viszonylag olcsón, egyszerüen elvégezhető.

Az ugynevezett irórudas nyomtatókban a karakterkészlet a kalapácsok előtt oda-vissza mozgó rudon helyezkedik el. A rud mentén a karakterkészlet általában itt is többször ismétlődik. A rud mozgása lehet folyamatos vagy szakaszos (lépésenként). Az utóbbi az olcsóbb megoldás, de a nyomtatási sebessége is kisebb az előbbinél.

Speciális nyomtatási eljárásokkal igen nagy sebességü (még 25 000 sor/perc is) nyomtatás valósitható meg. A mechanikus rendszerü nyomtatók közül ide tartozik az ugynevezett mátrixnyomtató. Ennél nincsenek tárolva különálló karakterek, hanem minden egyes karakter egy 5 x 7-es (ritkábban 7 x 9-es) "raszter"-ben elhelyezkedő pontokból tevődik össze. A pontok kinyomtatása az ugynevezett tük segitségével történik, ezek ütnek a festékszalagon keresztül a papirra. A nyomtatási szélességnek megfelelő-

en egymás mellett, egy sorban sürün elhelyezkedő tük közül egyidőben csak a megfelelők ütnek a papirra. Ezáltal a kinyomtatandó sor karaktereinek egy "pontsora" készül el egy lépésben. A teljes sor karaktereinek pontjai a papir továbbhaladása közben igy tulajdonképpen 7(9) lépésben készülnek el. Mivel itt a tük kis tömegét kell csak mozgatni és nincsenek forgó alkatrészek, nagy sebesség érhető el.

A nem mechanikus müködésen alapuló nyomtatók közül megemlitjük az elektrosztatikus, valamint a katódsugárcsöves eljárásokat. Ezek alkalmazása azonban pillanatnyilag még nem általános.

ESZ-7030 Sornyomtató

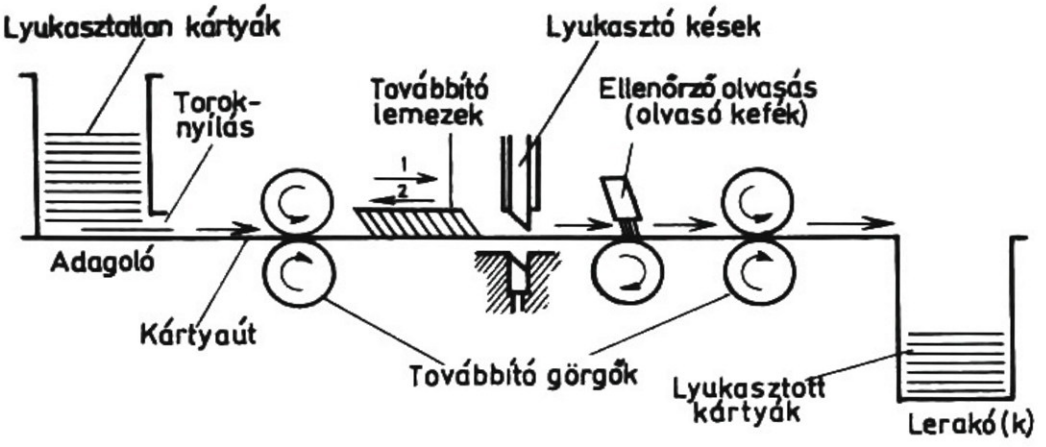
A központi egységből az ESZ-7030 berendezés 128 byte-os puffer tárolójába kerül egy sornyi információ. A puffertároló feltöltése után a berendezés kinyomtatja az információt a központi egység müködésétől függetlenül. A sornyomtató a központi egységből a puffertárba kerülő adatokon paritás ellenőrzést is végez.

Kiirási sebesség (sor/perc)	650-890
A kinyomtatott jelek száma egy sorban (db)	128
Jelkészlet (db)	82
Információátviteli módok (vagylagosan)	- monopol (burst multiplex)
	- multiplex
Müködési elv	aszinkron
Papirméret	
legnagyobb szélesség (mm)	420
legkisebb szélesség (mm)	80
Papir mozgásának vezérlése	négycsatornás lyukszalag
Másolatok száma (különleges papiron)	max. 2

4.3.2 LYUKKÁRTYALYUKASZTÓ

Miután a központi egység feldolgozta az információt, az eredmények rögzitésére, megjelenitésére nem minden esetben a sornyomtató az előnyös. Vannak adatok, amelyeket esetleg máshol akarunk feldolgozni (pl. kiértékelni, kinyomtatni stb.), vagy később, mint bemenő adatokat ujból felhasználni. Ilyen esetekben néha szükséges az output adatok kártyára rögzitése. Gondoljunk csak vissza a duál kártyára, amelyet a további feldolgozás előtt még ujabb, kiegészitő információkkal is el lehet látni! A lyukkártyalyukasztó ezekben az esetekben jól felhasználható output berendezés. Egyébként meglehetősen ritkán használják, mivel sebessége kicsi, és gyakran előnyösebb az adatok ismételt feldolgozása esetén a tárolást mágnesszalagon vagy lemezen elvégezni. A lassu müködési sebességet a lyukasztás mechanikai bonyolultsága okozza. Maga a kártya pedig, mint már emlitettük, sok helyet foglal el és ezért gazdaságtalan adathordozó. Például az az adatmennyiség, amely egy szokásos (730 m-es) mágnesszalagtekercsen tárolható, csak kb. 60 m magas kártyaoszlopon férne el.

A lyukasztóegység vázlatos rajza az alábbiakban (130. ábra) látható. A kártyák az adagolóból a toroknyiláson át itt is a kártyautba kerülnek. A lyukasztásnál általában késes adagolást használnak.



130. ábra. Lyukkártyalyukasztó elvi felépitése

(A kis adagolási sebesség és az uj, még fel nem használt kártyák ezt lehetővé teszik.) A kártyauton a kártya szakaszosan halad, mivel csak álló kártyába lehet lyukasztani (különben a kések elszakitanák a papirt). A szakaszos mozgást ugynevezett továbbitó lemezek biztositják. Ezek a lemezek az ábrán az 1. nyillal jelölt irányba elmozdulva a következő lyukasztási pozicióba továbbitják a kártyát. A lemezek kialakitása és elrendezése olyan, hogy a 2. irányu elmozduláskor a kártya mozdulatlan marad. Ekkor történik a lyukasztás.

Többféle lyukasztási módszer is ismeretes. A legáltalánosabban használt a soronkénti lyukasztás. Sebessége 100-300 kártya/perc. Ennél egy sor lyukasztása egyszerre történik, vagyis 80 lyukasztókés dolgozik egy időben és a kártyát 12-szer kell pozicionálni. Vannak lyukasztók, amelyek oszloponként lyukasztanak. Ez a leglassubb megoldás. Sebessége 50-100 kártya/perc. Itt csak 12 késre van ugyan szükség, de egy kártya elkészültéig ezt 80-szor kell továbbléptetni és pozicionálni. Létezik egy ugynevezett blokk lyukasztás (tömb lyukasztás) is. Szemben az oszloponkénti vagy soronkénti lyukasztással, amelynél a kártyát a lyukasztási folyamat közben pozicióról-pozicióra továbbitani kell, a blokklyukasztásnál a kártya nyugalomban van és az összes rögzitendő karakter egyetlen lépésben kerül lyukasztásra. Ez ugyan a leggyorsabb megoldás, de nem igen használatos, mert 960 db késre lenne szükség és ez nagyon bonyolulttá és drágává tenné a berendezést. A lyukasztás után a kártyákat azonnal visszaolvassák, ellenőrzik, hogy a helyes információ került-e rögzitésre. A lyukasztás kis sebessége miatt itt az elektromechanikus (kefés) olvasó is megfelelő, de azért gyakran használnak fotodiódás olvasást is. A kártyába lyukasztott információt vagy összehasonlitják az egység átmeneti (puffer) tárában - ha van ilyen - levő eredeti (lyukasztandó) információval, vagy a valós lyukkombináció alapján végeznek ellenőrzést. Ha az olvasóegység hibát észlelt, a hibás kártya egy külön lerakóba (ugynevezett másodlagos lerakóba) kerül és a hiba ellenőrizhető.

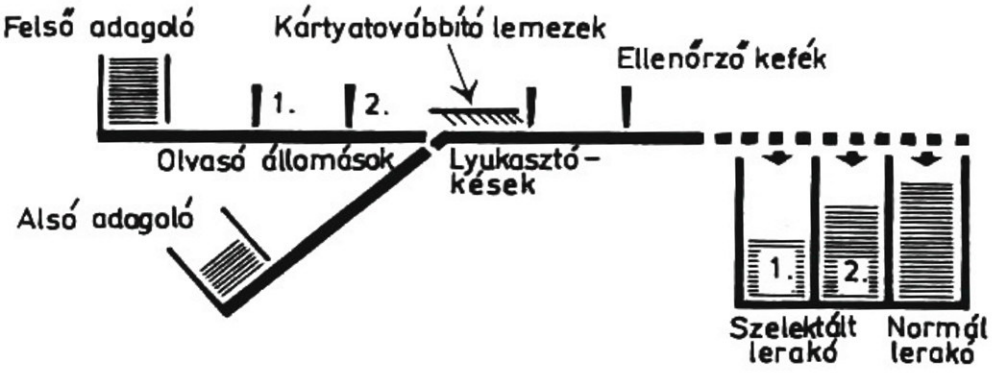
A lyukasztókések mozgását az ugynevezett visszhang-ellenőrzéssel (echo-check) vizsgálják. Ez ugy történik, hogy egy-egy érintkező zárása jelzi, hogy a megfelelő kés átjutott-e a kártyán, megtörtént-e a lyukasztás azon a pozicióon, ahol lyukasztani kellett. Valamely kés nem megfelelő működése hibajelzést eredményez.

A lyukkártyalyukasztók különleges fajtája az ugynevezett olvasólyukasztó. Ez esetben a kártyaut mentén a tulajdonképpeni lyukasztóállomás előtt is egy vagy két olvasóegység van. Az olvasólyukasztókat tehát nemcsak kiviteli, hanem beviteli (gyakran tartalék) egységként is lehet használni, bár beol-

vasási sebességük alacsony. Az olvasólyukasztók a beolvasott eredeti kártyákra további adatokat is tudnak lyukasztani. Ezt ugy hivják "lyukasztás ugyanarra a kártyára". Vannak olyan olvasólyukasztók is, amelyek egy második kártyauttal is el vannak látva (131. ábra). A második ut a lyukasztóállomás előtt csatlakozik a főkártyautba. Ennek a második utnak a segitségével adott időben, adott helyen - a program által vezérelve - a többi kártyához üres kártyákat lehet hozzákeverni, amelyekre uj információ lyukasztható, például az előző kártyák tartalmának összege.

A kártyalyukasztó egységek általában feliratozást is végeznek, vagyis a kártyába lyukasztott információt szöveges formában ráirják a kártya tetejére.

A kártyalyukasztók nagy része rendelkezik puffer tárral is. A központi egységből érkező egy "kártyányi" információ itt kerül átmeneti tárolásra, itt vár a lyukasztásra. A kártyalyukasztó vezérlőegysége a szükséges kódátalakitást (EBCDIC-ből Hollerithba) és a lyukasztásnak megfelelő rendezést (oszloponként vagy soronként) ennek segitségével tudja elvégezni.



131. ábra. Olvasólyukasztó két kártyauttal

ESZ-7010 Kártyalyukasztó

Az információt KPK-12 kódban vagy kódátalakitás nélkül tudja a lyukkártyán rögziteni. A DKOI kódot KPK-12 kóddá alakitja át. A lyukasztás módját az utasitás kódja határozza meg. Maximálisan 256 kódkombináció lyukasztása lehetséges.

A berendezés 256 byte kapacitásu, szinkron müködésü puffertárral rendelkezik, amely azonban maximálisan csak 160 byte befogadására képes. Miután a puffertár megtelt, a berendezés a központi egységtől függetlenül végzi el a betöltött információ egy kártyára történő kilyukasztását.

A központi egységből a puffertáron keresztül a lyukasztóba küldött információt byte-onként ellenőrzi páratlan paritás segitségével. A lelyukasztott információ helyességét ellenőrző olvasással állapitja meg a berendezés.

A kártyalyukasztó müködhet multiplex, illetőleg monopol (burst multiplex) üzemmódban, és a központi egység multiplex vagy szelektor csatornájához kapcsolódhat.

A lyukasztás sebessége (kártya/perc)	max. 100
Az információhordozó lyukkártya	80 oszlopos
A lyukasztás elve	elektromechanikus

A lyukkártya lyukasztásának módja	pozicionált
Az adagoló kapacitása (db)	max. 700
Kártyalerakók száma (db)	2
Az egyes kártyalerakók kapacitása (db)	max. 700

4.3.3 LYUKSZALAGLYUKASZTÓ

Kimenő adatok lyukszalagra történő lyukasztása a kisebb számitógépek környezetében szokásos. Az adatfeldolgozás céljára szolgáló számitógépeknél ma már nem igen használatos és gyakran előfordul, hogy uj gépbeállításoknál nem is kerül sor a felszerelésére. (Az alkalmazás lehetősége azonban továbbra is megvan!)

Ma általában csak ott célszerü alkalmazni, ahol müszaki, tudományos számitásoknál kisebb adatmennyiséget kell kiiratni. Az ilyen módon rögzitett eredményeket telexen továbbitani is, vagy akár kinyomtatni is lehet a felhasználás helyén. A lyukszalagos gépparkra épülő vállalatok adatfeldolgozásában pedig lehetőséget ad járulékos elemzésre, továbbfeldolgozásra vagy kiiratásra is.

A sornyomtató meghibásodása esetén adatok kiiratásában, továbbitásában tartalék szerepet is játszhat. A lyukszalaglyukasztók nagy része átkapcsolható, igy 5-8 csatornás lyukszalag a legtöbb berendezésen alkalmazható. Maga a müködés szakaszos, mivel csak álló szalagokba lehet lyukasztani (132. ábra). Egyszerre egyetlen karakter és a szinkronlyuk készül el. Igy 6-9 késre van szükség. A szinkronlyuksor lyukasztása mindig megtörténik, függetlenül attól, hogy lyukasztottunk-e információt vagy sem. (ld. 105. ábrán). Igy a lyukszalag haladásának sebessége állandó. A számitógépek környezetében használatos berendezések sebessége max. 100-150 karakter/sec.

132. ábra. Lyukszalaglyukasztó, a szalagorsó tehetetlenségét kiegyenlitő szalagpufferral

ESZ-7022 Lyukszalaglyukasztó

Az információ 8 csatornás lyukszalagra lyukasztható a KOI-8 kódból. Másoló üzemmódban az információ 5-8 csatornás lyukszalagra lyukasztható, paritásbit lyukasztásával vagy anélkül.

A szalagszélesség a lyukasztóban beállítható. Egy karakter kilyukasztása után a lyukszalag egy lépéssel tovább mozog. A berendezés ellenőrzi a csatornából származó kilyukasztandó információ helyességét, kódátalakitó üzemmódban a kódátalakitást, a papirszalag végét, a lyukasztó berendezés motorjának program szerinti bekapcsolását, illetőleg kikapcsolását.

A berendezés működhet multiplex és monopol üzemmódban, és igy csatlakoztatható a multiplex vagy szelektor csatornához.

Max. kiirási sebesség (karakter/perc)	150
Az információhordozó lyukszalag	5-8 csatornás
A lyukasztás elve	elektromechanikus

4.3.4 RAJZGÉP (PLOTTER)

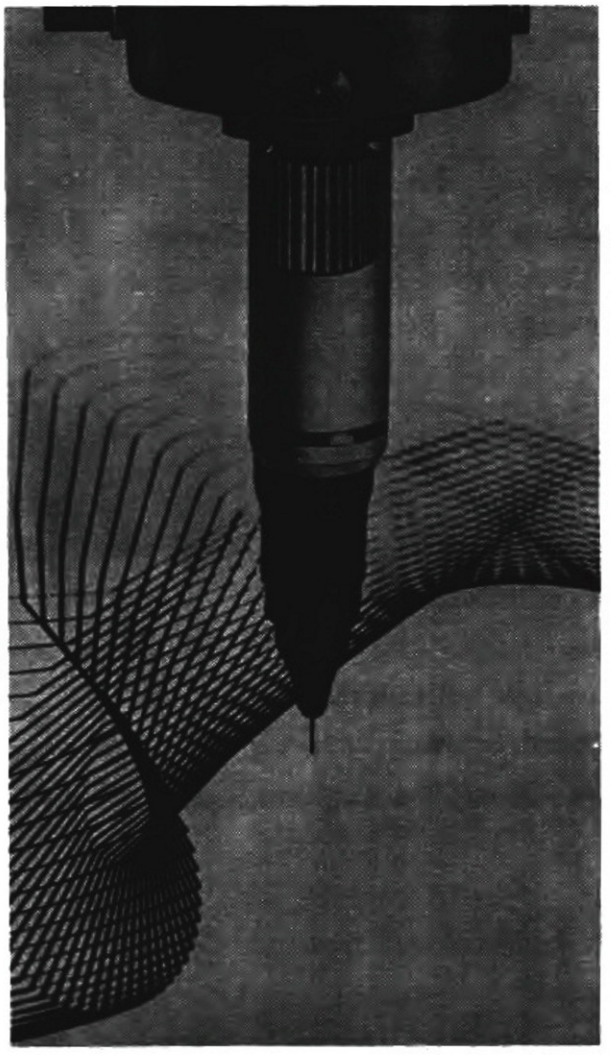
Igen sok olyan feladat van, amelynek megoldását, "eredményét" különféle diagramok, függvények, rajzok jelentik. Csak néhányat emlitsünk ezek közül!

- nyomtatott áramkörök topológiájának, félvezetők, integrált áramkörök maszkjának elkészitése (gyakran 1 μm-es pontosságra van szükség!)
- mérési eredmények, statisztikai adatgyüjtések gyors kiértékelése
- matematikailag, vagy tapasztalati uton megadott függvények, illetve diagramok, összefüggések felrajzolása
- térképészetben, földrajzi, meteorológiai, navigációs térképek rajzolása
- épitészetben szerkezeti, axonometrikus ábrák létrehozása, többszintes utkereszteződések vonalvezetésének meghatározása
- közmüvezetékek tervezése, rajzolása, vagy akár kapcsolási, szerelési rajzok elkészitése, dokumentálása
- szabásminták, kivágósablonok optimális kialakitása
- szerszámgépek numerikus vezérlésének ellenőrzése, kipróbálása.

Ezek a munkák gyakran sablonos, mechanikus tevékenységek sokaságából állnak össze (133. ábra). Ennek ellenére hosszu ideig tartó nagy figyelmet, egy bizonyos pontosságot és megfelelő teljesitőképességet igényelnek, ezt adott határon tul embertől már nem lehet elvárni. Ezt csak a "lelketlen" gép birja ki! Erre alkalmas a rajzgép, más néven plotter. Bár ezek speciális felhasználásuk miatt nem a leggyakrabban alkalmazott output perifériák, több fajtájuk is van:

- elektromechanikus plotterek
- elektrosztatikus plotterek
- elektronikus plotterek.

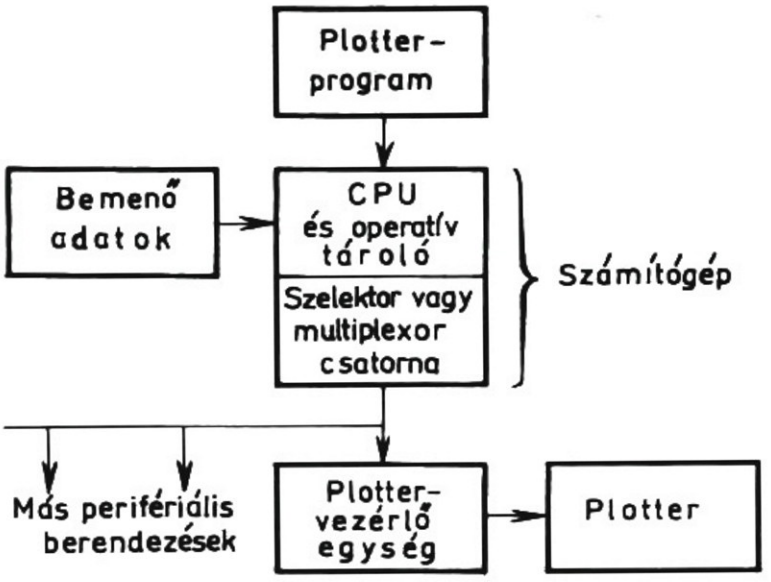
Mi csak az első csoportba tartozóval fogunk egy kicsit részletesebben, az adott lehetőségek között foglalkozni. A másik két csoportba tartozónak csak elvi lényegét ismertetjük. A rajzgépek a működésükhöz szükséges információt vagy on-line vagy off-line módon kaphatják a számitógéptől.



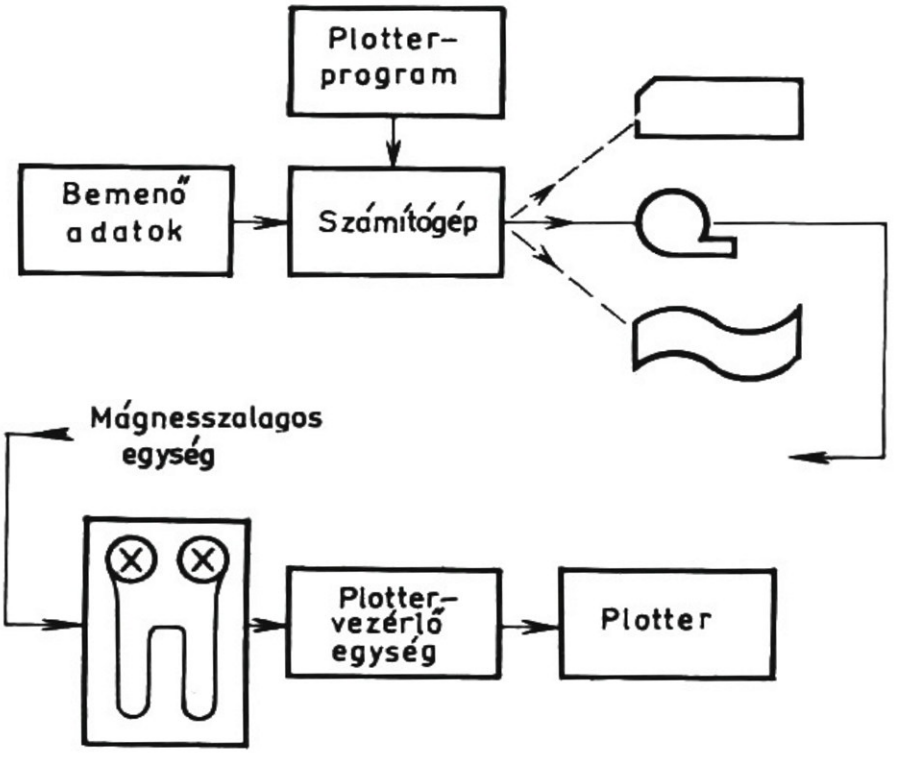
133. ábra. Geometriai (térbeli) felület szemléltetése plotter segitségével

On-line kapcsolat esetén a vezérléshez és a rajzoláshoz szükséges valamennyi információ közvetlenül jut el a számitógéptől a rajzgépig (134. ábra). Erre vagy egy külön kis "plottervezérlő" számitógépet, vagy egy nagy gép időosztásos üzemmódját lehet felhasználni. A plotter az átmeneti (puffer) tárának nagyságától függően az információt vagy byte-onként (a multiplex csatornán) vagy blokkonként (a szelektor csatornán) kaphatja meg.

Ennél a kapcsolatnál azonban gyakoribb az off-line. A számitógéptől jövő információ nem közvetlenül, hanem valamilyen adathordozó (lyukszalag, lyukkártya, mágnesszalag) segitségével jut el a plotterig (135. ábra). Igy a számitógép kapacitása nincs a rajzgép viszonylag lassu müködése folytán hosszu időre lekötve. A rajzgép tulajdonképpen a rendszertől függetlenül dolgozhat.



134. ábra. Számitógéppel on-line kapcsolatban levő plotter

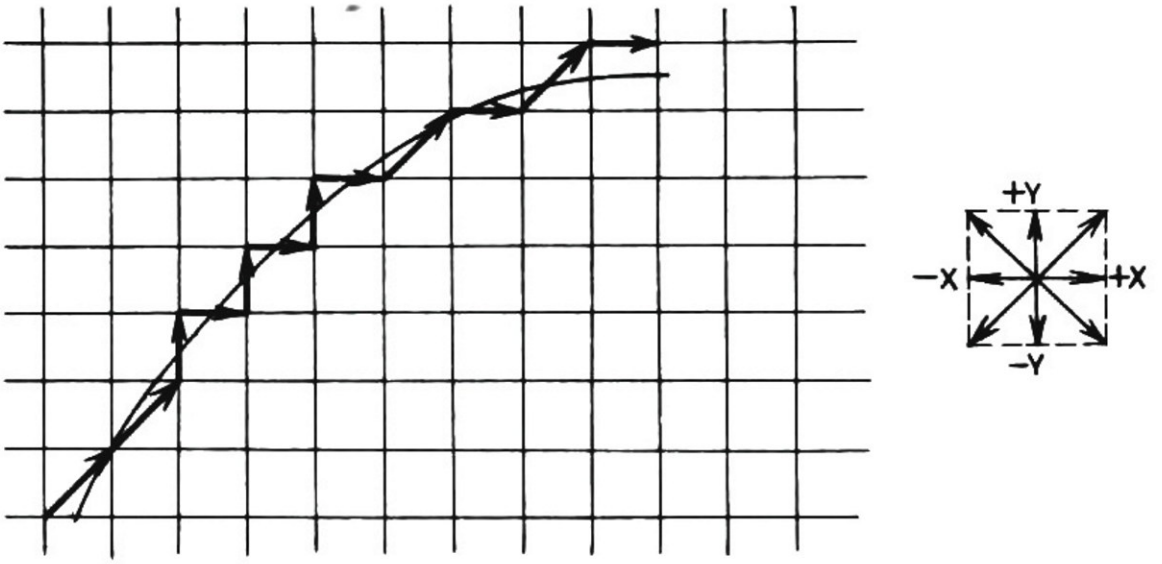


135. ábra. Számitógéppel off-line kapcsolatban levő plotter

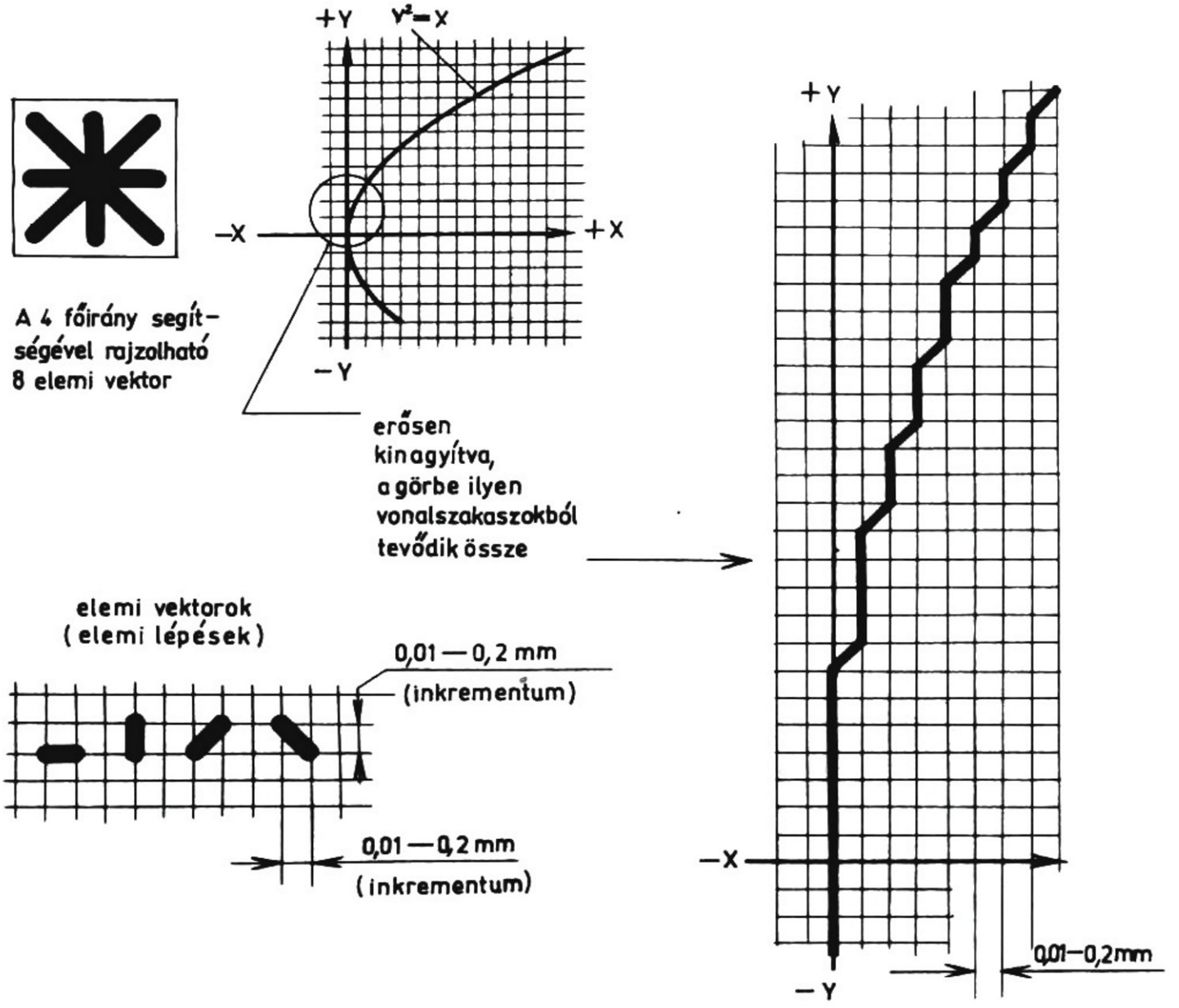
a) Az elektromechanikus plottereknek két fajtájuk van:

- digitális (növekményes) müködésü
- analóg müködésü

A növekményes vagy más néven inkrementális rajzgép müködése azon alapszik, hogy bármely görbe vonal elvileg tetszőleges pontossággal megközelithető elegendően rövid egyenes szakaszokkal (136. ábra). Ha egy irószerkezet a nyugalmi helyzetéhez képest egy megadott osztástávolságnyit (inkrementum-nyit) képes a sik $\pm X$ és $\pm Y$ irányában, akár egyidőben is elmozdulni, 8 un. elemi lépést tud végrehajtani. A legtöbb plotter az elemi lépések közben rajzolható 8 elemi vektor segitsé-



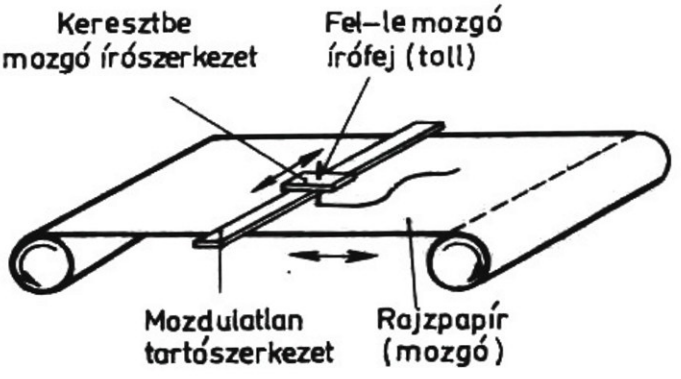
136. ábra. Görbe közelitése megadott irányu egyenes szakaszokkal

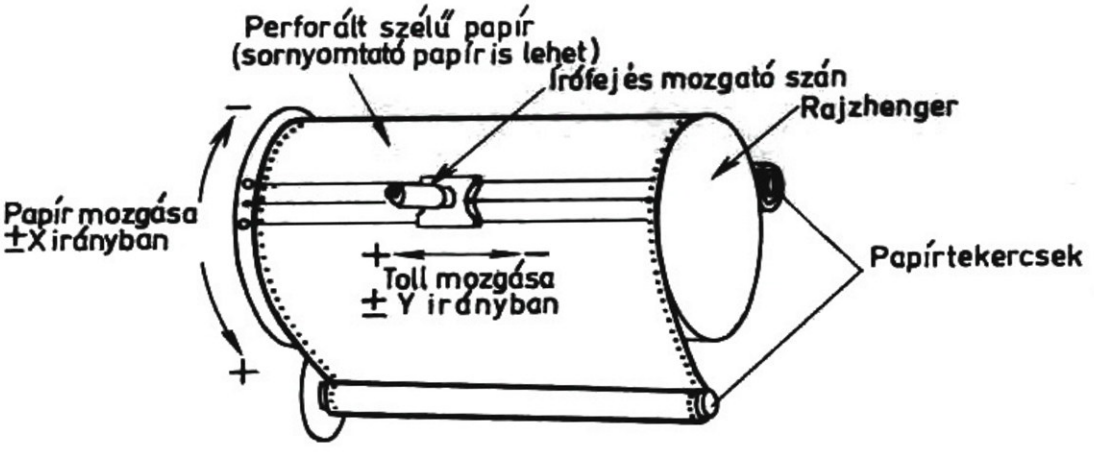


137. ábra. Függvény rajzolása adott elemi lépések segitségével

gével irja le a különféle görbéket. Meg kell azonban emliteni, hogy van un. 16, sőt nagy pontosságu növekményes müködésü rajzgépeknél un. 24 vektoros közelitési eljárás is. Azonban ezek is az előbbi elven rajzolják a görbéket, csak a 4 főirányban kétféle lépésnagyság (inkrementum) tehető meg. Az inkrementumok nagysága aitalában az olcsóbb és gyorsabb müködésü plottereknél 0,1-0,2 mm közötti. Az igen nagy pontosságu gépek 0,01-0,05 mm-es osztásközzel (felbontóképességgel) dolgoznak (137. ábra). A rajzolási sebességük általában 50-150 mm/sec, de van, amelyiké eléri a 300-500 mm/sec-ot is! Az "ábrák" rajzolása történhet jóminőségü papirra, müszaki vagy filmpauszra, müanyagfóliára, fényérzékeny papirra, fémfóliára. Igen gyakran ezek felületére vékony vonallal finom raszterhálót is készitenek, hogy a rajzolás és a vonalvastagság ellenőrzését megkönnyitsék. Felépitésüket tekintve az elektromechanikus rajzgépeknek két fajtájuk van. Az un. hengeres plotternél (drum plotter) a tekercs formában levő rajzpapir két orsón helyezkedik el, amelyek közül bármelyikre fel- és bármelyikről le-csavarodhat. A ± X irányu mozgást, ami megegyezés szerint a papir hossztengelyével megegyező, a papir előre-hátra haladása biztositja a hengerek között. Az alábbi ábrán ennek vázlatos elvi rajza látható, illetve a gyakorlatban elterjedt megvalósitási módja (138. ábra). A papir továbbitását a hengerek között általában a papir szélén levő lyuksor segitségével maga a fémből, gumiból vagy müanyagból levő un. rajzhenger végzi, amire a papir felfekszik. A ± Y irányu mozgást, a henger alkotójával párhuzamosan mozgatható szán segitségével az irófej biztositja. Ez az un. keresztszán a fixen beépitett csuszópályán mozoghat.

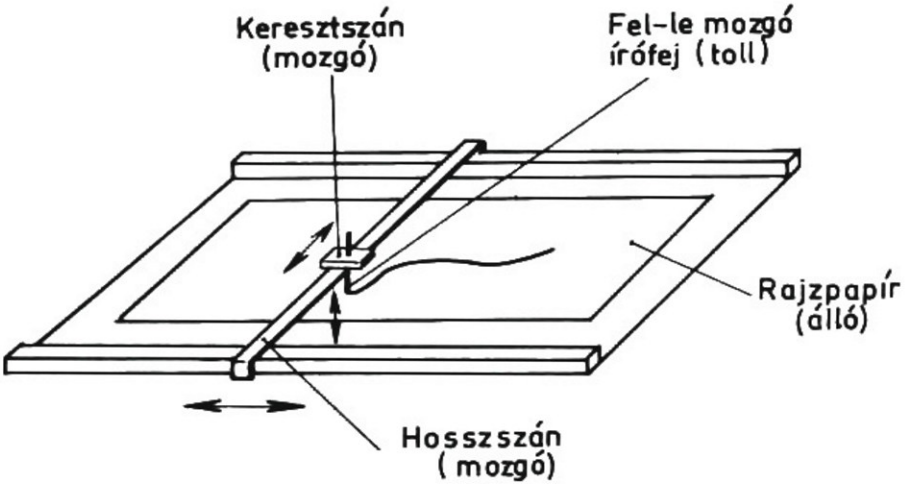
A legtöbb hengeres rajzgépben különböző szélességü papir használható fel részben a henger cseréjével, részben a huzókörmök állitásával. A mindkét irányu mozgatást (a papir ±X, az irófej ±Y) un. léptetőmotorok végzik. Ezek minden egyes impulzus hatására csak egy megadott szöggel (lépéssel)

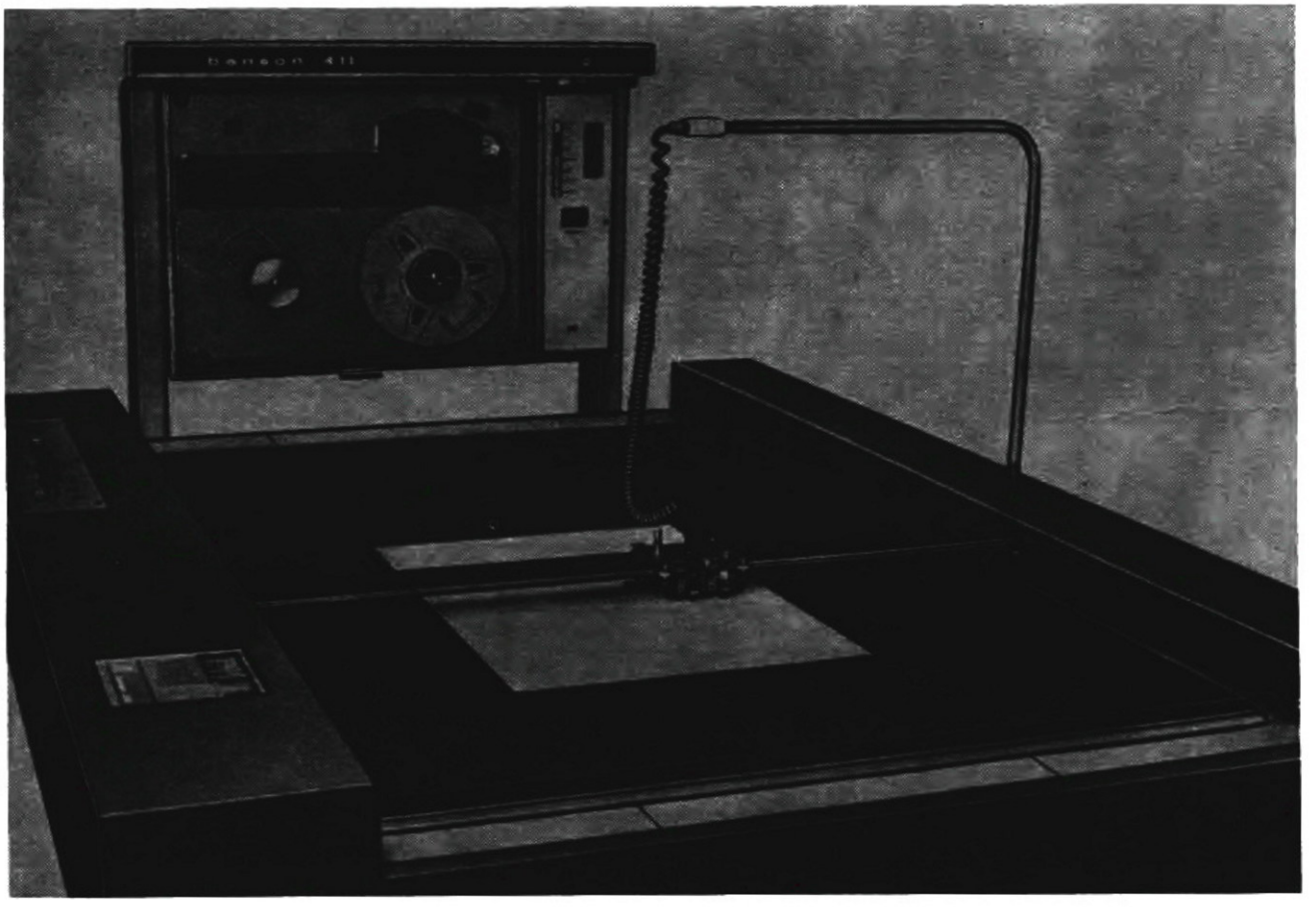




138. ábra. Hengeres plotter

fordulnak el. A mozgató mechanizmus, a megvezetések egyszerüsége miatt ez a rajzgép olcsóbb ugyan, de ugyanakkor pontatlanabb is, mint a másik fajta, az un. sik plotter (flatbed plotter). Itt a sik mindkét irányu mozgását egy-egy szánszerkezet végzi. Az előbbiekben már emlitett keresztszán, vagy kocsi az un. hosszszánon vagy hidon tud elmozdulni, ami a rajzasztal egyik oldalával megegyezően előre-hátra haladhat. A két szán együttes, illetve az irófej fel- és lemozgatásával lehet a kivánt görbe vonalat, ábrát létrehozni. Az alábbiakban (139. ábra) az elvi müködés, illetve egy gyakorlati megvalósitás látható (Benson 220 tipus, mögötte a Benson 411 mágnesszalagegység az off-line müködéshez).

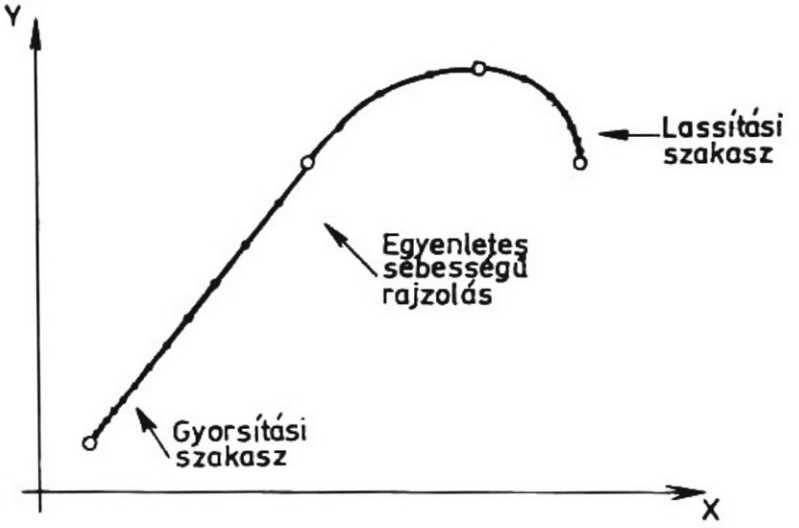




139. ábra. Sik plotter

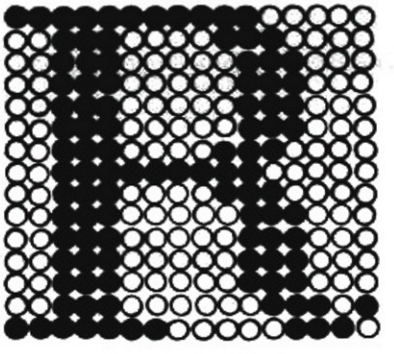
A sik rendszerü rajzgépeknél a papir és a rajzolás méretét maga a berendezés, az asztal nagysága korlátozza. Ez A3 mérettől akár egészen a szobanagyságig változhat. Általában ezek a berendezések mechanikailag nagy méretüek, stabilak, hogy a külső rezgéseket, behatásokat erősen csillapitani tudják. Az asztallapra a különféle méretü papirt lehet akár vákuumal is rögziteni, apró, néhány tized milliméteres furatok vagy rések segitségével, de használatos az elektrosztatikus rögzités is.

Az elektromechanikus plotterek másik nagy csoportjába tartoznak az analóg vezérlésü rajzgépek. Ezek a görbét nem egyenlő nagyságu elemi lépések (inkrementumok) segitségével közelitik meg, hanem a megteendő távolságot különböző hosszuságu lépésekben, fokozatmentesen teszik meg (140. ábra). Ennél a megoldásnál az irófej mindkét irányu mozgatásához un. szervomotorokat használnak.



140. ábra. Görberajzolás analóg müködésü plotterrel

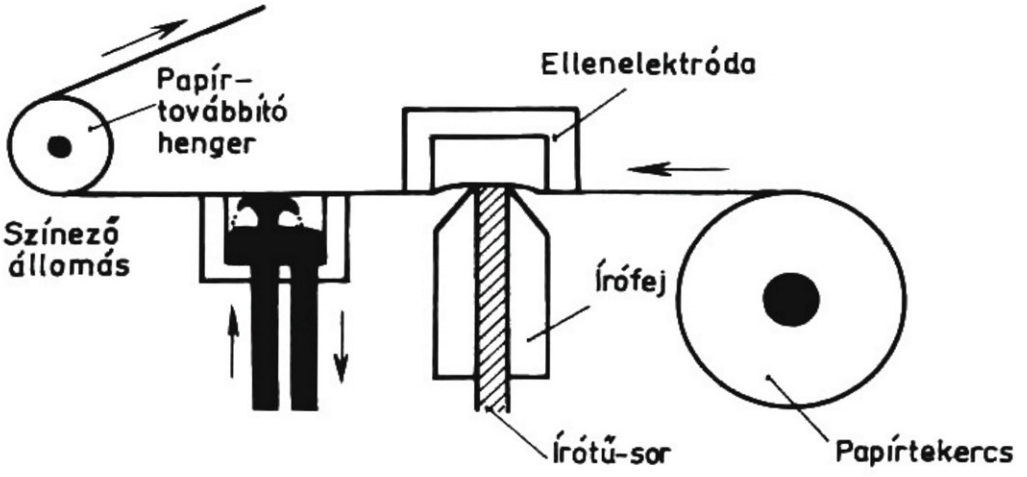
b) Az elektrosztatikus vagy más néven mátrixplotterek a zajtalan nem ütközéses (nonimact) elven müködő mátrixnyomtatókkal azonos elven dolgoznak. Ezért legtöbbször mint sornyomtatók és mint plotterek is alkalmazhatók (un. printer-plotter). Mozgó szerkezetet csak a papirtovábbitáshoz igényelnek. A rajzolás, vagy a karakterek "irása" egy sor mentén, pontonként történik (141. ábra). A papir (20-50 cm széles) először egy irófej előtt halad el, amelyen egymás mellett sürün (100-160-200 db/inch) tük helyezkednek el. Egy megfelelő logika segitségével kiválasztott tükre feszültséget adva, a tük hegyével szemben a dielektromos (szigetelő) tulajdonságu papiron egy kis helyi töltés jelenik meg. Az



141. ábra.
Karakter előállitás mátrixplotterrel

igy soronként más-más helyen (pontonként) elektromosan feltöltött papir egy un. szinező állomáson halad keresztül, ahol az elektrosztatikusan feltöltött helyekre festék csapódik. Igy a kivánt ábra vagy szöveg pontonként, folyamatosan készül el (142. ábra).

A berendezésből kikerülő papir semmilyen további "kikészitést" nem igényel, jól olvasható, másolható. Vannak olyan eljárások is, ahol a festék nem az elektrosztatikus feltöltődés hatására, hanem hőre és/vagy nyomásra kerül a papirra. A "rajzolt" karakterek felbontóképessége, finomsága az 5 x 7, 7 x 9 vagy 16 x 16-os elemi pontmátrixtól függően változik, amelyekből a szükséges jel képe kialakul.



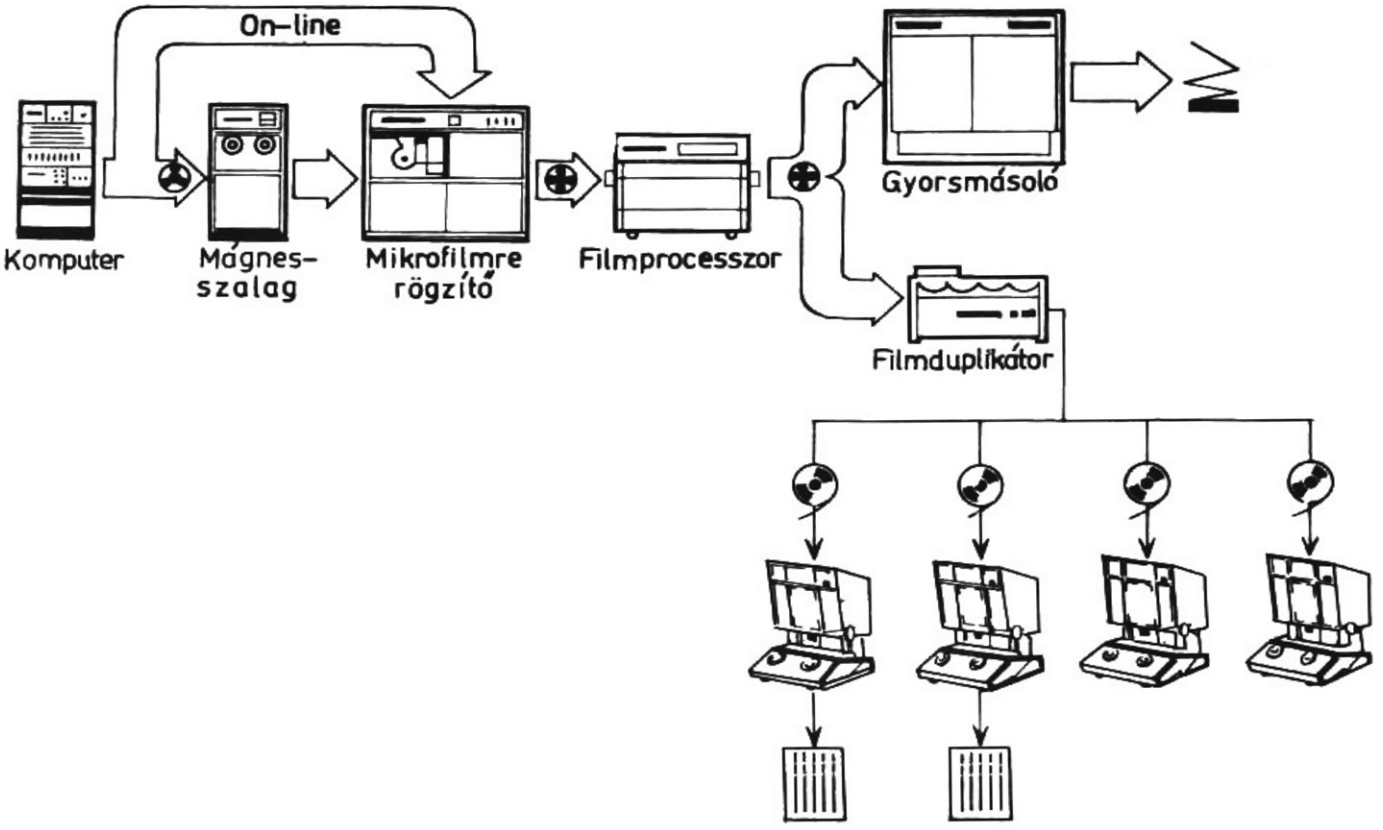
142. ábra. Mátrixplotter elvi működése

c) Elektronikus plotterek (COM - Computer Output to Microfilm)

Ezek tulajdonképpen már nem is kimondottan rajzgépek. Bár az ábrák, rajzok megjelenitésekor a programozás az inkrementumok segitségével történik itt is (ezért is emlitjük őket itt), nem csak ábrák megjelenitésére szolgálnak. Ahogy a printer-plotter is felhasználható információ közlésére, karakterek megjelenitésére, ugy ez is, sőt itt ez nagyon nagy szerepet kapott.

Egyes esetekben megjelenésük és katódsugárcsöves működésük miatt a display-ekhez hasonlithatók. Legfőbb jellemzőjük, hogy a számitógépből kikerülő, mindennemű információt mikrofilmre rögzitenek (143. ábra). Van olyan megoldás, amely ezt egy katódsugárcsöves megjelenitő képernyőjének automatikus, folyamatos lefényképezésével éri el. A katódsugárcső akár 100 ezer lépés/sec sebességű megjelenitését, rajzolását felhasználva, egy mágnesszalag tartalmát 8-10 perc alatt mikrofilmre lehet vinni. Más esetben az információ apró fénysugár segitségével közvetlenül kerül a filmre. A folyamatot követő filmhivás, filmmásolás már egyben az ellenőrzést vagy akár a sokszorositást is lehetővé teszi.

A számitógép kimenő adatainak vizuális formában mikrofilmen való rögzitése és ilyen formában való tárolása komoly, figyelemreméltó előnyöket jelent. Ismételt felhasználásra szánt adatoknál a kis tárolási helyigényt feltétlen meg kell emliteni. (A mikrofilmen tárolt információ helyigénye csak kb. 2 %-a a sornyomtatóénak.)



143. ábra. Mikrofilmes kiiró berendezes elvi felépitése (elektronikus plotter)

Teljesitménye messze tulszárnyalja a sornyomtatók teljesitményét. (Még akár 120 ezer jel/perc is lehet.) Müszaki dokumentumok, jogi, szabadalmi leirások mikrofilmmel történő tárolása esetén az igen nagy tömegü információból sokkal gyorsabb kikeresésre, elérésre van lehetőség. (Megfelelő indexekkel a visszakeresés sebessége az on-line rendszerek müködését is megközelitheti.)

Jelentős költségcsökkenés is elérhető! Ugyanazon információtömeg ilyen formában való tárolása kb. 20 % anyagköltségmegtakaritással jár a papirhoz képest.

Ez a számitógépes output jól automatizálható, igy kevesebb felügyeletet, ellenőrzést is igényel. A legkézenfekvőbb, legszembetünőbb megtakaritás a nagy tömegü információk tárolási helyének és szállitási költségeinek csökkentésében jelentkezik. A papirral szemben, ami közismerten nehéz anyag, szállitási költsége csak kb. 0,1 %-a!

Hogy ilyen sok és ilyen tekintélyes előnyök ellenére miért nem általános az alkalmazása, annak egyik fő oka a berendezés nagy ára. Másik lényeges dolog, hogy az emberek hozzászoktak már a papirnál lehetséges közvetlen olvasáshoz, javitáshoz, megjegyzések készitéséhez, és igy bizonyos foku idegenkedéssel fogadják.

4.4 HÁTTÉRTÁRAK

A számitógépek müködése közben a felhasználásra kerülő összes adatok, illetve a már "kész eredmények" tárolása az operativ tárban nem lehetséges. Ez részben a költségeket növelné meg nagyon, részben rendszertechnikai akadályai is vannak. (Az egyes gépekhez alkalmazható max. opera-

tiv tárkapacitás adott.) Ezért a gyakorlatban 512 Kbyte-1 Mbyte kapacitásu operativ tárnál nagyobbat ritkán alkalmaznak. A rendszer müködése szempontjából pillanatnyilag nem szükséges adatokat éppen ezért nem itt, hanem másodlagos tárakban, az un. háttértárakban tárolják. Ezek igen nagy mennyiségü információ befogadására alkalmasak, az operativ tárhoz viszonyitva lényegesen alacsonyabb költségtényező mellett. Van viszont egy hátrányuk is, mégpedig az, hogy az adatok kikeresése, elérése viszonylag hosszu időt vesz igénybe (ld. majd a 172. ábrán). A gyakorlatban a mágneses adattároláson alapuló eszközök terjedtek el, mint háttértárak. Valamilyen hordozó felületre (szalagra, lemezre, hengerre, kártyára, huzalra stb.) vékony mágnesezhető réteget visznek fel, és az ebben keltett mágneses állapotot használják fel az információ tárolásához. A mágneses jelrögzités azon a fizikai tényen alapszik, ami szerint minden villamos áram egy mágneses mezőt hoz létre maga körül, illetve megforditva, hogy minden mágneses mező adott esetben felhasználható elektromos áram létrehozásához. Ezt a jelenséget alkalmazzák a mindenki által ismert magnetofon, illetve magnószalag müködésében is. A kettő közti lényeges különbség az, hogy ott időben folyamatosan változó (analóg) jeleket kell tudni a szalagon rögziteni, a számitógépes adattárolásnál viszont csak az információnak megfelelő két diszkrét értéket (digitális jel) kell tudni biztosan, tévedésmentesen ábrázolni. A gyakorlatban ehhez több jelrögzitési módot is használnak. A közös jellemzőjük az, hogy az információt (abiteket) egy keskeny csikban irják rá a mágnesezhető felületre. Az igy felirtakat akárhányszor kiolvashatják, változatlanul felhasználhatják, vagy ujabb irás előtt letörölhetik. Az irás, olvasás folyamatában vagy ugyanazt a fejet használják fel, vagy külön mindegyikre kialakitanak egyet. (A törlést általában mindig külön fejjel végzik.) Ezeket a fejeket gyakran egy közös fejegységbe, un. fejblokkba foglalják össze. A fejek közül egy időben néha több is dolgozhat. Igy az adatok átviteli sebessége gyorsabb, hamarabb lehet ugyanazokat a jeleket felirni vagy kiolvasni.

A mágneses adattárolásra használt eszközök legismertebb tipusai (a hordozó alapján téve köztük különbséget):

a mágnesszalag
a mágneslemez
a mágnesdob
a mágneskártya.

A következő fejezetben a legnagyobb elterjedtségnek örvendő mágnesszalaggal és mágneslemezzel kissé részletesebben is foglalkozunk, mig a felhasználás szempontjából kisebb jelentőségü mágnesdobnak és mágneskártyának csak a müködési elvét ismertetjük.

A tárolásra szolgáló eszközöket szokták még az elérési mód alapján is elnevezni. E szerint van
- közvetlen elérésü tár
- közvetett, soros elérésü tár.

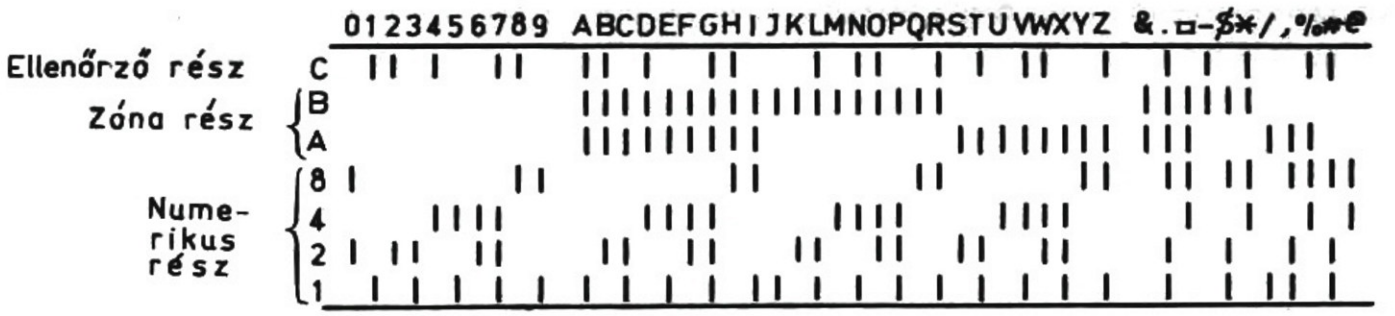
A közvetlen elérésü tárakhoz azokat sorolják, amelyeknél bármely tárolt adat elérése átlagosan ugyanannyi időt vesz igénybe, bárhol is helyezkedik el a tárban (elején, végén stb.)
Ez azt jelenti, hogy egy megadott un. átlagos elérési időtől az eltérés nagyon kis érték, akár a legkedvezőtlenebb esetben (max. kikeresési, elérési idő), akár a legkedvezőbb esetben (min. elérési idő) is vagyunk. Ide tartozik a háttértárak közül a mágneslemezes, a mágnesdobos, a mágneskártyás tár. (Bár nem háttértár, de közvetlen elérésü az operativ tár is!)

A közvetett, vagy soros elérésü tárnál valamely adat megkeresése csak ugy lehetséges, ha az összes többi, előtte levő adaton már átléptünk, azok cimeit végignéztük. Szokták ezért kereső vagy kiváró tipusu tárnak is hivni. Ezeknél átlagos elérési időt nincs értelme definiálni, mert attól az eltérés igen nagy lehet (a tényleges elérési idő). Igy az "előrejelzés" igen pontatlan volna. Ebbe a csoportba a mágnesszalagos tár tartozik.

4.4.1 MÁGNESSZALAGOS TÁR

Az egyik legáltalánosabban használt háttértár. Elterjedtségét részben nagy tároló kapacitása, részben a jeltárolás alacsony bitenkénti költsége okozza. Tárolás formai szempontból a lyukszalaghoz hasonlitható. Az információt ott egy vagy több lyuk jelentette egy oszlopon belül. A mágnesszalagon a lyukaknak mágneses jelek felelnek meg. A lyukszalaghoz hasonlóan a mágnesszalagon is hosszirányu csatornák, sávok vannak. Ezek persze nem kerülnek "kilyukasztásra", mint a lyukszalagnál, tehát vizuálisan nem láthatók, csak elképzelhetjük őket. A jelenleg használatos mágnesszalagok általában 9 vagy ritkábban 7 csatornásak. Közülük általában 8, illetőleg 6 sáv az információ rögzitésére szolgál, a 9., illetőleg 7. sávra pedig az ugynevezett keresztirányu paritásbitek kerülnek. Ezek a szalagon "keresztben" levő bitek számát egészitik ki páratlanra (páratlan paritás), vagy párosra (páros paritás). Többnyire kézi kapcsolóval állitható be, hogy a mágnesszalagos egység páros vagy páratlan paritással dolgozzon-e.

A rögzitendő karaktereket a szalag mozgására merőlegesen, keresztirányu oszlopokban, ugynevezett frame-okban helyezik el. Egy oszlopon belül az irás-olvasást az egyes sávokhoz tartozó fejek egyidőben végzik. A jelrögzitéshez használt mágnesezettséget, vagyis, hogy az oszlopok egyes csatornáiban milyen módon van tárolva az információ, a készülék gyártója irja elő. Az ábrázoláshoz használt kódok is többfélék lehetnek. A 7 csatornás szalagon gyakran alkalmazott rögzitési mód az ugynevezett 6 csatornás BCD kód vagy a 7 csatornás ASCII kód. A 9 csatornáson elterjedten használják az EBCDIC-et, valamint a 7 csatornás ASCII kódot, a legmagasabb helyiértéken egy 0-val kiegészitve 8 bitessé. Természetesen közvetlenül bináris formában is lehet tárolni a szalagon információt, ilyenkor a byteok egy-egy oszlopot foglalnak el. Az alábbi 144. ábrán egy 7 csatornás mágnesszalagkódot lehet látni, páros paritásellenőrzéssel.



144. ábra. Karakterkészlet 7 csatornás mágnesszalagon (BCD kód + páros paritás ellenőrzés)

A mágnesszalagon tárolható információ mennyiségét nagymértékben befolyásolja az alkalmazott irássürüség, ami a szalag (vagy egy csatornája) egységnyi hosszán elhelyezhető karakterek (vagy bitek) számával adható meg. Megadását általában a bit/inch-ben végzik, és ennek jelölése bpi. Tulajdonképpen felfoghatjuk inch-enkénti oszlopszámnak is. Általánosan használt a

200 bpi,

556 bpi,

800 bpi,

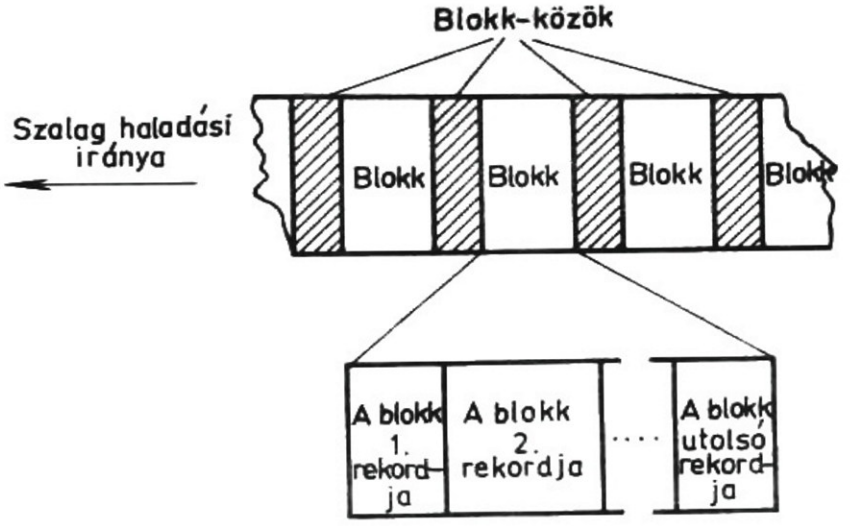
1600 bpi

érték. A 800 bpi körülbelül 320 bit/cm-nek vagy karakter/cm-nek felel meg, ebből számitva az 1600 bpi pedig durván 640 bit/cm. Nálunk ezek az értékek szabványositva is vannak. Egyes irássürüségeknél általában más és más rögzitési (irási) mód használatos. Ezért a berendezések nagy része csak egy vagy esetleg kétféle irássürüséggel tud dolgozni, a gyártótól függően. Ezeken az előbb felsorolt" szabványosan szokásos" értékeken kivül létezik még néhány, jóval nagyobb irássürüség is.

A mágnesszalag anyaga jó minőségü (nem nyulékony, alaktartó) vékony (16-50 μm) müanyag fólia, amelynek egyik oldalán helyezkedik el a mágnesezhető réteg. A szokásos mágnesszalag-szélességek 12,7 mm (1/2 inch) és 25,4 mm (1 inch) között változnak. A szalaghosszuság általában 730 m (2400 láb), de ez elvileg változtatható. Attól függően, hogy milyen (kis vagy nagy) szalagorsót használnak, a hossz 90 és 1100 m is lehet.

Korábban emlitettük már az adatok logikailag legmagasabb szintjét, a file-t (lásd: Adatszintek cimü rész.) Ez nem más, mint valamilyen szempontból összetartozó, tehát azonos fajtáju rekordok összessége. A mágnesszalag tulajdonképpen ezeknek a file-oknak a soros, egymás utáni tárolását végzi. A file-t alkotó logikai rekordok, amelyek hossza a feladattól függően fix vagy változó, a mágnesszalagon blokkokba, adattömbökbe kerülnek. Ezeknek a blokkoknak azonban saját cimük nincs! Egy blokkba kerül mindaz az adatmennyiség, amelyet a berendezes egy egységként kezel, vagyis amelyet a perifériális egység (szalag) és a központi egység között egyetlen utasitás (irási vagy olvasási) hatására átvisz. A mágnesszalagegység tehát nem a logikailag összetartozó elemi adatokat, az un. logikai rekordokat kezeli, nem azokat irja fel a szalagra, illetve olvassa le onnan, hanem egy fizikai adathalmazt, a szalag egy fizikailag meghatározott részét, egy ugynevezett blokkot. Ezért ezt fizikai rekordnak is szokták nevezni. (Nem a logikai, a tartalmi összetartozás, hanem a fizikai összetartozás az elsődleges jellemzője.) A blokk hossza lehet kötött (fix) vagy változó. Tartalmilag egy vagy több logikai rekordot foglal magában (145. ábra). Ha a blokk csak egy logikai rekordot tartalmaz, blokkolatlan adatállományról beszélünk. Egy logikai rekordnál kevesebbet viszont sohasem tartalmaz. Azt a számot, amely megadja, hogy a szalagon egy blokkban hány logikai rekord helyezkedik el, blokkolási tényezőnek nevezik. Értéke az előbbiek alapján nagyobb, vagy egyenlő 1-gyel. A blokkot alkotó logikai rekordok fizikai megcimzésére nincs külön lehetőség a szalagon, mivel az egyes fizikai I/O utasitások egy blokknyi információt fognak "megmozgatni"! Emiatt a blokkolás, az adatrekordok blokkokba való összefogása mindenképpen hátrányosnak tünik. Ennek ellenére főként gazdaságossági megfontolások miatt szükség van rá. Miért?

A szalagon a blokkokat közök, ugynevezett gap-ok határolják. Ezek általában 0,8-2 cm hosszu, adatot nem tartalmazó szalagrészek. Ez biztositja, hogy a mágnesszalagot a következő blokk elejéig az olvasásnak megfelelő sebességre fel lehessen gyorsitani, illetőleg a blokk után ezen a tartományon belül le lehessen fékezni. A gyorsitáshoz, illetőleg a lassitáshoz szükséges időt hivják start, illetőleg



145. ábra. Blokkok elhelyezkedése a mágnesszalagon

stop időnek. Egy 1,5 cm-es blokk-köznyi szalagdarab viszont jelentős mennyiségü információ tárolására lenne alkalmas (120-960 karakter, irássürüségtől függően)! Ha tehát a szalagon minden logikai rekord után következne egy ilyen blokk-köz (a blokkolási tényező ekkor 1 volna), nagyon sok "üres helyet" tárolnánk. Nézzünk erre egy példát!

Tételezzük fel, hogy a szalagra 1000 db lyukkártya tartalmát akarjuk felvinni, és egy-egy kártya tartalma egy logikai rekord. (A kártyának mind a 80 oszlopában van adat, tehát a rekord 80 byte-os). A rögzitéshez használt irássürüség 1600 bpi = 640 bit/cm. Igy egyetlen kártya tartalma, vagyis egy rekord 0,125 cm hosszu mágnesszalagon fér el ($\frac{\text{byte-ok száma rekordonként (byte)}}{\text{irássürüség (byte/cm)}}$). A blokk-köz mérete legyen 1,5 cm. Látható, hogy a blokk-köz több mint tizszer annyi helyet foglal el, mint a logikai rekord, a tényleges információ!

Először nézzük meg, milyen hosszu szalagra van akkor szükség, ha a blokkolási tényező = 1, vagyis minden logikai rekord (kártyatartalom) egy külön blokkot alkot. Ebben az esetben a blokkhosszusság egyenlő a rekordhosszusággal, igy a szalagszükséglet 1,625 cm.

blokkhosszusság = rekordhosszusság

szalagszükséglet = blokkhosszusság + blokk-köz = 0,125 cm + 1,5 cm = 1,625 cm

A kártyaköteg szalagigénye pedig

1000 x 1,625 cm = 1625 cm = 16,25 m

Most ugyanezt a kártyaköteget ugy vigyük fel szalagra, hogy a blokkolási tényező = 10, vagyis 10 kártya tartalma kerül egy blokkba.

A blokkhosszusság = 10 x rekordhosszusság = 10 x 0,125 = 1,25 cm

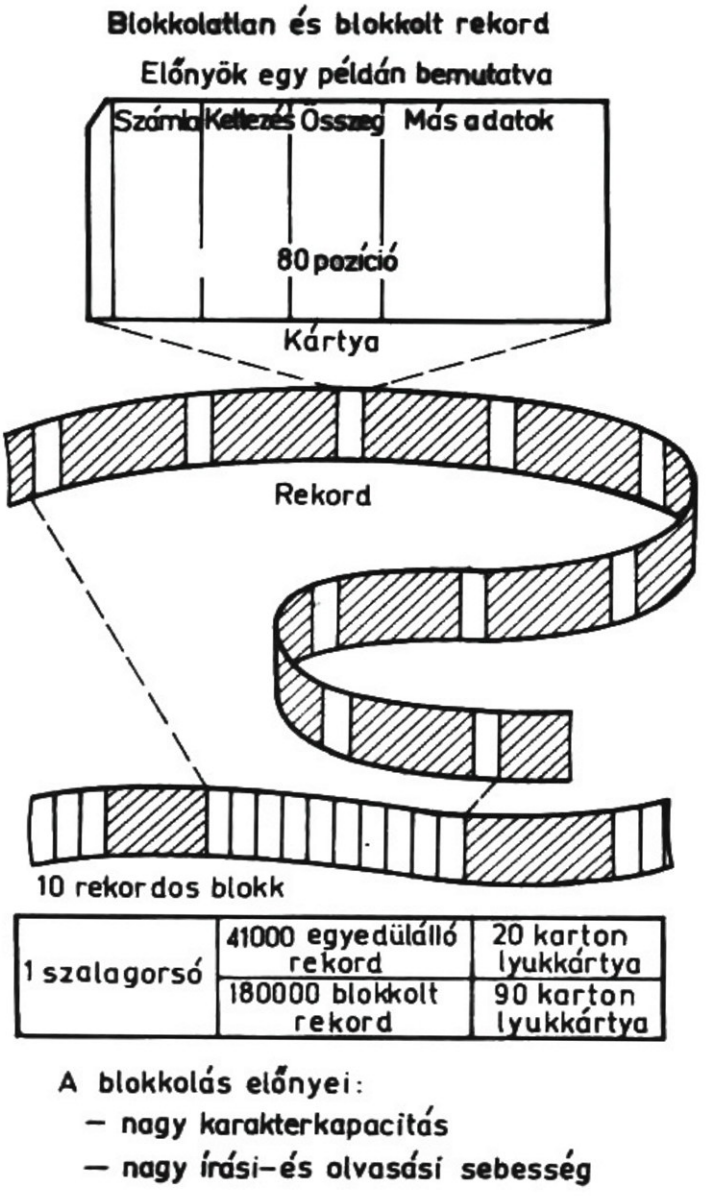
Egy blokk szalagigénye igy:

Szalagszükséglet = blokkhossz + blokk-köz = 1,25 cm + 1,5 cm = 2,75 cm

Az egész 1000 db-os kártyaköteg 100 ilyen blokkba fog elférni. Ehhez összesen

100 x 2,75 cm = 275 cm = 2,75 m

szalag kell. Szalagszükséglet tehát körülbelül egyhatodára csökkent le! (146. ábra). Igy egy szokásos szalag (730 m) kapacitása 5-30 Mbyte között változhat.



146. ábra. A blokkolás megnöveli a szalag kapacitását

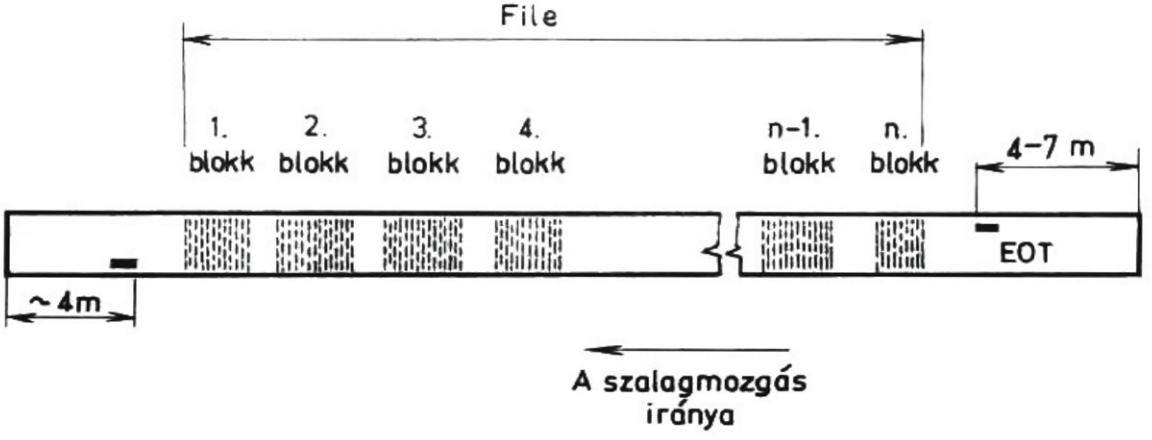
Ugyanekkor a szalagon levő logikai rekordokra vonatkoztatott átviteli sebesség is megnő, mert nem kell a berendezésnek állandóan megállni, elindulni. Azt hihetnénk tehát, hogy a nagyon nagy blokkolási tényező ideális szalagkihasználást jelent és ezért arra kell törekedni. Az adatblokk nagyságával kapcsolatban gondolnunk kell a következőkre is:

- Mivel minden ki- és bemeneti utasitás egy blokkot visz ki az operativ tárból a szalagra, illetőleg hoz be a szalagról az operativ tárba, a blokk tartalmának megfelelő nagyságu területet fenn kell tartani az operativ tárban. Minél nagyobb a blokk, az adott méretü op.tárban igy annál kevesebb hely marad az egyéb programok, adatok számára.
- Ha a szalag tartalmának rendezéséhez mágneslemezt kivánunk felhasználni, célszerü a blokk hosszának a mágneslemez sávkapacitásán belül maradni.
- Az átviendő blokk hosszát részben a software is meghatározza azzal, hogy mekkora adattömeget tud mozgatni egy utasitás hatására.

A számitógép I/O utasitásai a blokkok mozgatása során (a háttértárból a központi egységbe és viszont), automatikusan elvégzik azok logikai rekordokból való összeállitását, illetőleg az azokra történő szétbontását. A felhasználónak, illetve a programozónak kell viszont arról gondoskodni, hogy az irás előtt a rekord-, illetve a blokkhosszuság (byte-ban) meg legyen adva. Tehát nem a blokkolási tényezőt kell a rendszernek megadni, hanem annak ismeretében a blokk hosszát byte-ban!

A szalagon az információ kezdetének és a szalag végének a jelölésére egy külön jelzést alkalmaznak. Ez általában vékony aluminiumfólia, amelyet a szalag hátoldalára ragasztanak fel. A fényt visszaveri, igy segitségével a mágnesszalagegység felismeri a szalag elejét és végét.

A szalag kezdeténél van az ugynevezett szalagkezdet jel (BOT - Beginning of Tape) (148. ábra). A mágnesszalagos egység ajtajának becsukásakor vagy a szalag befüzése után automatikusan eddig tekeredik fel a szalag, erre a jelre áll rá a berendezés. A feldolgozás csak ettől a jeltől kezdődik, előtte a berendezés sem irni, sem olvasni nem próbál. Ez a jel a szalag fizikai elejétől kb. 3,5-4 m távolságra van. Ennyi szükséges körülbelül a szalag kényelmes és biztonságos befüzésére. A szalag végén van az ugynevezett szalagvég jel (EOT - End of Tape). Iráskor az jelzi a szalag felhasználhatóságának végét, ez után a jel után a berendezés uj blokk irásába nem kezd (amit még az EOT előtt elkezdett, azt természetesen befejezi). Ezt érzékelve, a szalagegység leállitja a szalagot, nem engedi lefutni az orsóról, vagyis nem rongálja, és ujabb befüzés nélkül ismét használható, visszatekerhető.



148. ábra. Szalagkezdet (BOT) és szalagvég (EOT) jel

A mágnesszalagon rendszerint vannak olyan blokkok is, amelyek feladata az adattekercs és az azon található adatok, file-ok azonositása, ellenőrzése. Ezek az ugynevezett belső cimkék vagy azonositók. A mágnesszalagtekercshez erősitett ugynevezett külső cimkékkel itt nem foglalkozunk, azok feladata a tekercsre vonatkozó vizuális információközlés.

A belső cimkéket vagy röviden csak cimkéket a gép ugyanugy beolvassa, mint az adatokat. Ennek segitségével az operációs rendszer, vagy maga a felhasználó is minden alkalommal megállapithatja, ellenőrizheti, hogy például kell-e még a szalag tartalmát őrizni, lehet-e már irni rá, a megfelelő szalag van-e az egységre feltéve, a megfelelő blokkszámot olvasta-e ki a berendezés a szalagról, helyes-e a file név stb. Abban az esetben, ha "helytelen hozzáfordulás" történt a szalaghoz vagy a szalagról olvasott cimketartalom nem felel meg a felhasználó által közölt adatoknak, a rendszer a konzolirógépen ezt közli és leállitja az olvasást vagy az irást, nehogy valamit "elrontson"!

Ezeknek a cimkéknek a használata azonban nem kötelező! A rendszerprogramok olyan file-okat is képesek feldolgozni, amelyek nem a szabványos cimkékkel vannak ellátva, vagy egyáltalán nincs is cimkéjük.

Abban az esetben, ha a file tartalmaz cimkéket, ugy azok lehetnek:

- rendszercimkék, amelyeket a rendszer (software) állit elő és dolgoz fel,
- felhasználói cimkék, amelyek a file azonositására szolgálnak és amelyeket a felhasználói programnak kell kezelni, feldolgozni, sőt előállitani is,
- nem szabványos cimkék, amelyek nem elégitik ki a szabványos cimkékkel szembeni követelményeket (hosszuság, forma tekintetében). Ezeket is a felhasználó programoknak kell feldolgozni.

A rendszercimkék lehetnek

- kötetcimkék (más néven tekercscimkék, adathordozó cimkék)
- file- cimkék.

Mindegyik cimke lehet kezdő- és végcimke. A kezdőcimke a kötet vagy a file kezdetét, a végcimke a kötet vagy a file végét határozza meg. Ennek megfelelően az alábbi szabványos cimkék vannak:

1. Szabványos rendszercimkék

VOL 1...8	Volume Label	- kötetcimke, adathordozó cimke
HDR 1...8	File Header Label	- file kezdőcimke
EOF 1...8	End of File Trailer Label	- file végcimke
EOV 1...8	End of Volume Label	- kötet végcimke

2. Szabványos felhasználói cimkék

UHL 1...8	User Header Label	- felhasználói kezdőcimke
UTL 1...8	User Trailer Label	- felhasználói végcimke
UVL 1...8	User Volume Label	- felhasználói kötet végcimke.

Abban az esetben, ha a mágnesszalagos kötet szabványcimkéket tartalmaz, a tekercs legelső rekordja (blokkja) a kötetcimke. Ezzel jelölhető meg a tekercs és a tekercs tulajdonosa. Ezt közvetlenül akkor kell a szalagra felirni, amikor a mágnesszalag tekercs beérkezik a számitóközpontba. Ezt a cimkét nem tekintjük egyetlen file-hoz tartozónak sem.

A tekercsen levő összes file legalább egy file kezdő- és egy file végcimkét tartalmaz. (A kezdőcimkék az adatok előtt, a végcimkék az adatok után állnak.) A kötet végcimkét az adatokat követően akkor kell a kötet (szalag) végére irni, hogy ha a file a következő kötetben (a következő szalagon) is folytatódik.

Minden cimke hossza fix, pontosan 80 byte. Az első három byte (karakter) a cimkék megnevezésére, a cimkejelzésre szolgál. A szalagon a felhasználás igényének megfelelően egymás után több (1...8) azonos fajtáju és igy azonos jelzésü cimke is állhat. Ezek ugynevezett cimkecsoportokat képeznek. A cimkecsoporton belül a legelső szabványos rendszercimke (VOL 1, HDR 1, EOF 1, illetőleg EOV 1) az ugynevezett kötelező (standard) cimke. Ezeknek mindig jelen kell lenni a szalagon. A rendszerprogramok csak ezeket dolgozzák fel. A csoport további tagjai (VOL 2...8, HDR 2...8, EOF 2...8, illetőleg EOV 2...8) az ugynevezett kiegészitő (járulékos) rendszercimkék. Ezek a kötelező cimkék után állhat-

nak csak és számukat a felhasználói program határozza meg. A cimkecsoporton belüli megkülönböztetést a cimke negyedik byte-ja szolgálja.

Ha vannak felhasználói cimkék is a szalagon, akkor ezek a file kezdő- vagy a file végcimke (a rendszercimkék) után következhetnek. Felirni is csak akkor lehet őket, ha a szalagon már vannak szabványos rendszercimkék.

A szalagon ezeken kivül alkalmaznak még ugynevezett szalagjelzőt, vagy másnéven szakaszhatároló jelet (TM) is. Ez egyetlen byte-ból álló blokk. A cimkéknek az adatblokktól történő elválasztását, illetőleg a szalagon levő információnak a végét jelzi. Ezért minden kezdőcimke-sorozat után, minden végcimke-sorozat előtt és után szalagjelző áll.

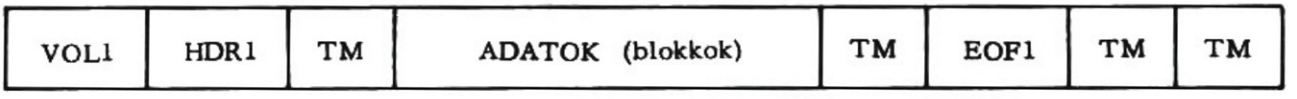
Az EOF cimkecsoport, illetve az esetleg meglevő UTL után egy szalagjelző jelenti a file végét. Az EOF után két szalagjelző a szalag utolsó file-ját (a felhasználható információ végét) mutatja. Ha a file egy másik szalagon is folytatódik, akkor ahogy ezt már mondtuk is, a szalag végére EOV kerül. Ez után két szalagjelző (ESZR/DOS -ban csak egy!) a szalag végét jelenti. Ha a szalagjelzők száma vagy pedig a helye nem az előirásos, a szalagegység nem tekinti a file-t szabványosnak.

Ezek után nézzük meg az adatok szervezését a mágnesszalagon a szabványos cimkékkel együtt.

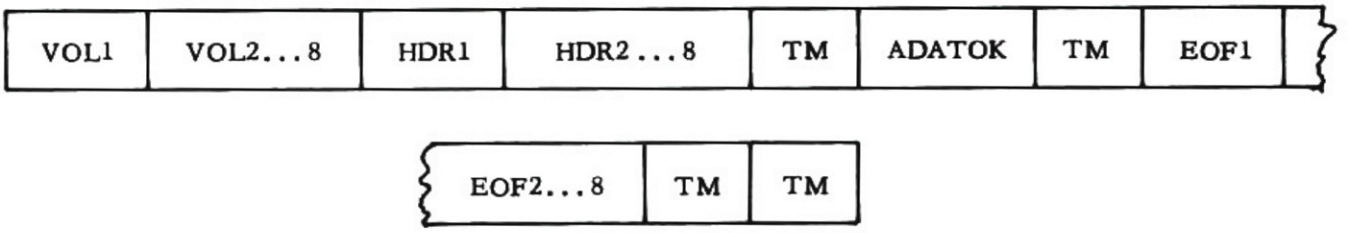
Négyféle eset fordulhat elő:

- egy file egy tekercs (egy szalagtekercs - "egykötetes file"),
- egy file több tekercs (egy file olyan hosszu, hogy egynél több mágnesszalagon fér csak el - "többkötetes file"),
- több file egy tekercs (a file-ok olyan rövidek, hogy több is elfér egy mágnesszalagtekercsen - "több file-os kötet"),
- több file több tekercs

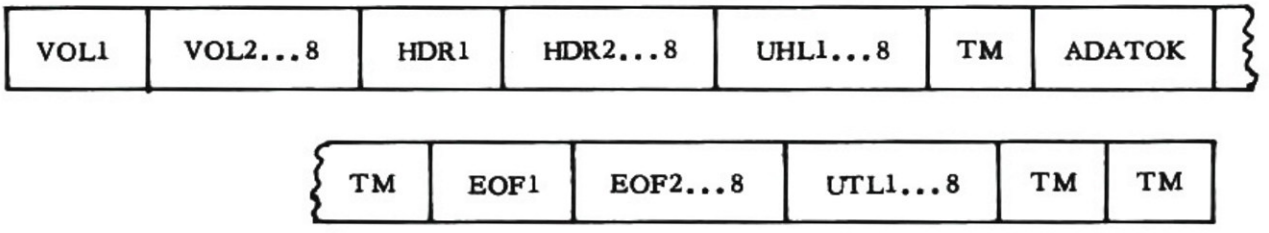
Egykötetes file kötelező cimkékkel:



Egykötetes file kötelező és kiegészitő (rendszer)cimkékkel:

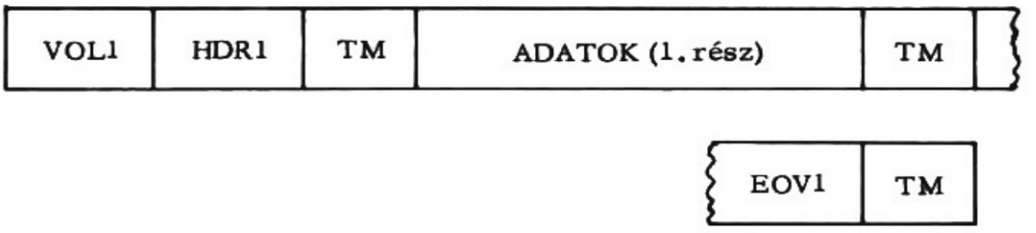


Egykötetes file kötelező, kiegészitő rendszercimkékkel és felhasználói cimkékkel:

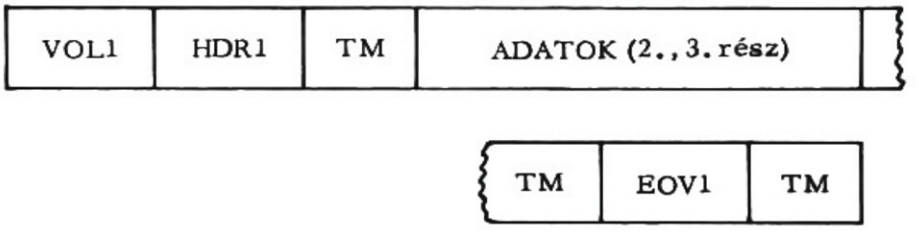


Többkötetes file kötelező cimkékkel:

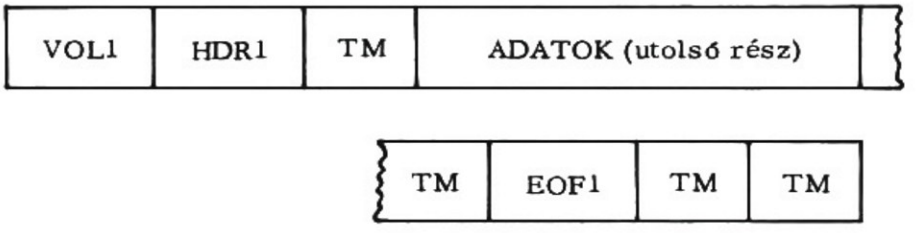
1. kötet (szalag)



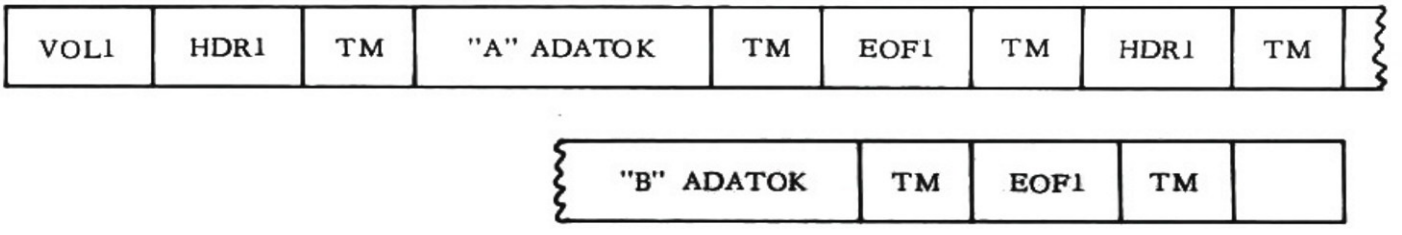
2, 3. kötet



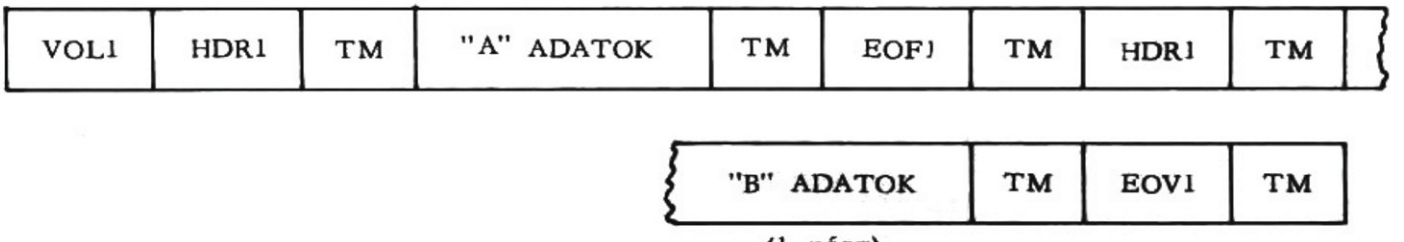
utolsó kötet



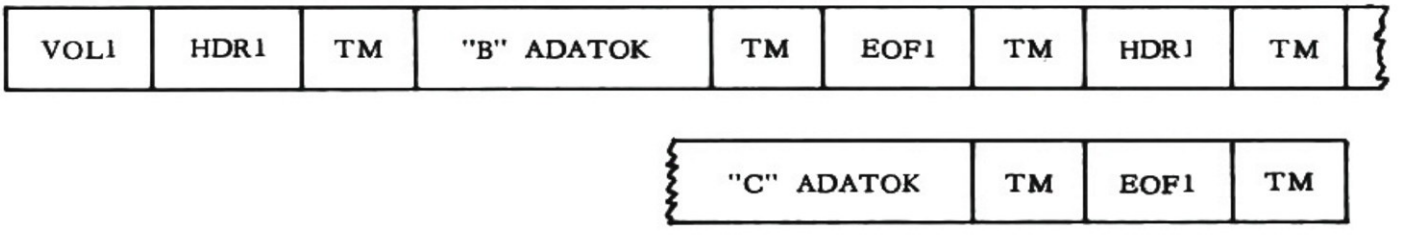
Több file-os kötet kötelező cimkékkel:



Többkötetes, több file-os szerkezet:



(1. rész)

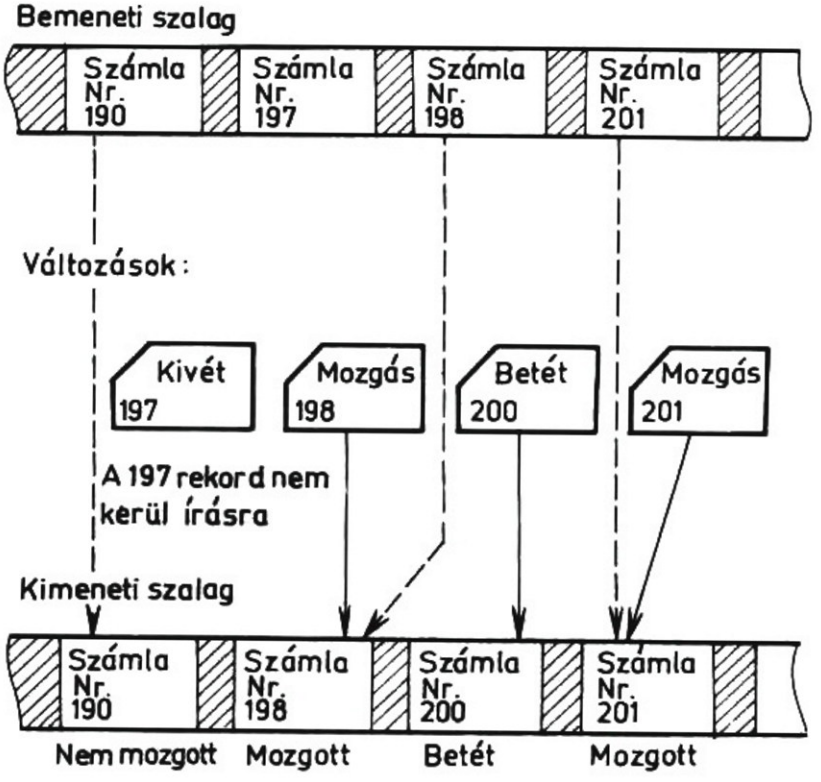


A file-ok megnyitását és lezárását (cimkékkel való ellátását), valamint a blokkok összeállitását, megirását és olvasását az operációs rendszer rutinjai végzik. Ezek egyéb feladatokat is ellátnak (hibalehetőségek, hibaüzenetek és az azokra kivülről érkező válaszok felismerése, a hiba elháritása stb.).

Meg kell még emliteni, hogy a mágnesszalagok tartalmazhatnak ugynevezett "ujrainditási pont" blokkot is. Az ujrainditási pont (check point) célja, hogy egy felhasználói programról olyan információkat őrizzen, amelyek segitségével a programot egy megszakitás után ismét el lehet inditani. Az ujrainditási pont felirását a felhasználói program vezérli. Például a feldolgozás meghatározott időpontjaiban az operativ tár, a regiszterek tartalma és a mágnesszalag poziciója feliródik a szalagra. Rendszerint ezt olyan programok esetén alkalmazzák, amelyekről feltételezhető, hogy a feldolgozásukhoz hosszabb időre van szükség. Ha ebben a programban megszakitás következik be, a programot nem kell utána elölről futtatni, hanem ujra lehet inditani az előző ilyen ponttól. Vissza kell tölteni a tárba, a regiszterekbe az előző ujrainditási pontnál kiirt tartalmat, a szalagot vissza kell tekerni az előző ilyen ponthoz, és a feldolgozást onnan lehet folytatni.

A mágnesszalagon a blokkolva vagy blokkolatlanul felvitt adatok egymás után sorban, a felvitelnek megfelelően, szekvenciálisan helyezkednek el. A feldolgozás akkor a leggyorsabb, ha az adatokat nem kell keresgélni, hanem azok a szükséges sorrendben vannak már. Ezért a felvitel előtt célszerü az adatok rendezését elvégezni. Ha a szalagra olyan ujabb adatot kell felvinni, amelyiket ebbe a rendezett sorba be kell illeszteni, az egész szalagtartalmat át kell másolni egy másik tekercsre. Ugyanez a helyzet akkor is, ha a szalagon tárolt rekordok közül kell valamelyiket törölni (149. ábra).

A mágnesszalag-egység müködése a magnetofonhoz hasonlitható. A szalag az irási, olvasási folyamat alatt egy leadó orsóról (lecsévélő orsó, adagoló orsó) a fejegység előtt 2-4 m/sec sebességgel elhaladva egy üres felvevő orsóra (felcsévélőorsó, olvasóorsó) kerül. A szalag mozgatását az un.

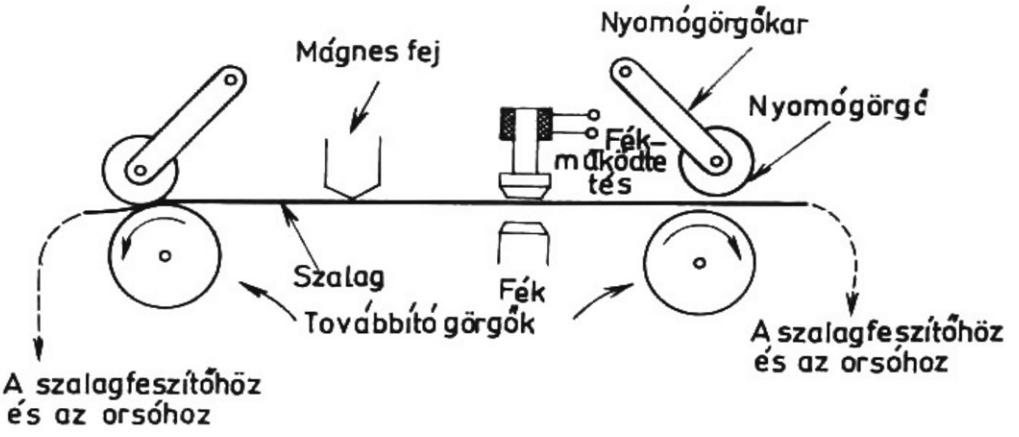


149. ábra. Mágnesszalag tartalmának módositása (update)

továbbitó görgők végzik. Attól függően, hogy ezek milyen módon, hogyan szállitják a szalagot, kétféle megoldásról lehet beszélni:

- nyomógörgős (surlódógörgős)
- vákuumgörgős

A nyomógörgős megoldásnál egy állandóan, egyenletes, nagy sebességgel forgó továbbitógörgőhöz az un. nyomógörgő rövid időre, hirtelen hozzányomja a szalagot (mint a magnetofonnál) és igy a szalag a surlódás folytán a továbbitógörgő által meghatározott irányban és sebességgel halad (150. ábra). A szalagegységben az előre-hátra mozgatáshoz általában két ilyen nyomógörgő-továbbitógörgő pár található, amelyek közül egyidőben csak az egyik érintkezhet a szalaggal, különben elszakitaná azt. A két továbbitógörgő forgásiránya természetesen ellentétes, az ellentétes irányu mozgatás biztositásához. Az alábbi ábrán azt az állapotot lehet látni, amikor a bal oldali nyomógörgő szoritja neki a szalagot a továbbitó görgőhöz, és igy az jobbról balra halad.

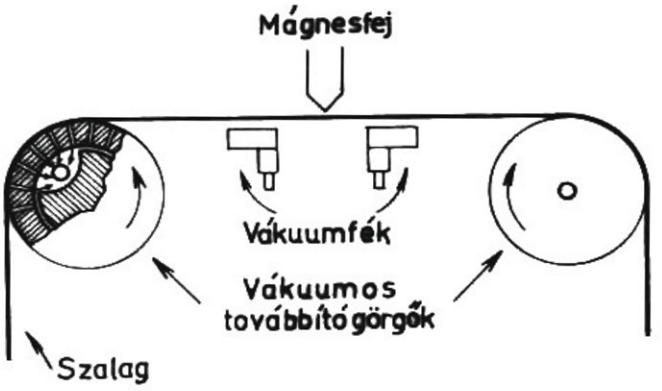


150. ábra. Nyomógörgős szalagtovábbitás

A szalag megállitását a beépitett mechanikus fék végzi. A nyomógörgős hajtás az egyszerüsége, az olcsó kivitelezhetősége mellett jelentős hátrányokkal is rendelkezik:

- a szalagot és főleg a mágneses réteget, a gyors továbbitásnál mechanikailag erősen igénybe veszi,
- a nagy nyomás miatt a szalag deformálódik, a mágneses réteg vastagsága is csökken, ami továbbitási és jelrögzitési problémákat okoz.

Ezeknek a hibáknak a kiküszöbölésére szolgál az un. vákuumgörgős szalagtovábbitó rendszer, ami lényegesen korszerübb és ezért egyre inkább azt használják. Ennél a megoldásnál a továbbitógörgő felületén apró lyukak, furatok vannak (151. ábra). Ezeken keresztül a görgő belsejében létesitett vákuum a szalagot a görgő felületére rászivja (illetve a külső levegő rányomja). A görgő belsejében létesitett tulnyomás viszont, a lyukakon kiáramló levegő segitségével a szalagot távol tartja a görgőtől. (Kis légpárna alakul ki a szalag és a görgő között.) A továbbitó görgőnek azonban csak 1/4 köriv mentén szabad a szalagra hatni, mivel különben becsipné, összegyürné. Ezt ugy oldják meg, hogy az állandóan, egyenletesen forgó továbbitó görgő belsejében egy álló betétet helyeznek el. Ennek kialakitása olyan, hogy a vákuumot vagy a tulnyomást csak kb. egy negyedköriv mentén engedi a furatokhoz jutni, máshol teljesen hozzásimul a továbbitó görgő belsejéhez, igy azon a sávon a furatokat elzárja.



151. ábra. Vákuumgörgős szalagtovábbitás

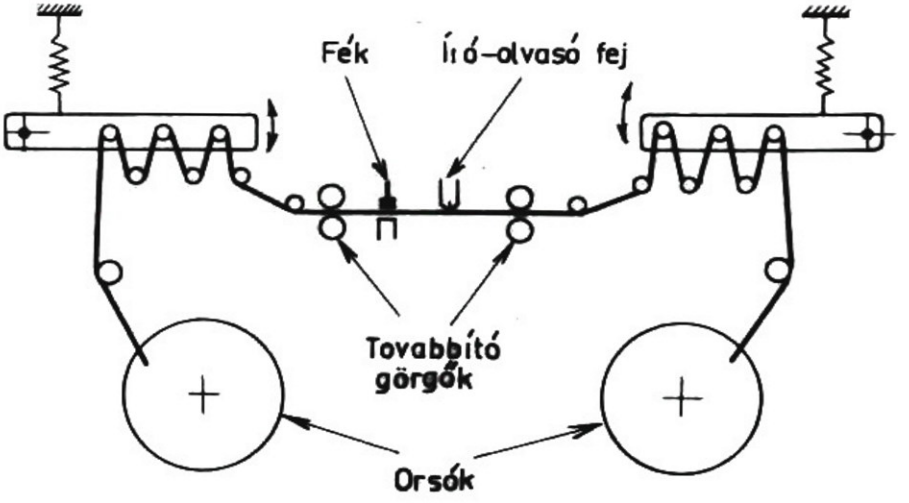
Ha a szalagegységben két továbbitógörgő van, akkor természetesen egyidőben csak az egyik kaphat vákuumot, a másiknak tulnyomást kell kapni, nehogy a szalagot elszakitsa. A szalag álló nyugalmi helyzetben való tartásakor mindkét görgő tulnyomást kap, igy a szalag nem ér hozzá a továbbra is sebesen forgó görgőkhöz, hanem csak a vákuumfékhez, mivel az biztositja a szivást.

Az orsók nagy tehetetlensége (tömege) folytán a blokkok olvasásával és irásával együtt járó, váltakozó, hirtelen inditások és megállások miatt a mágnesszalag nagy igénybevételnek volna kitéve. Ennek elkerülésére a fejegység alatti szalagrészt függetlenitik az orsók mozgásától. Ez a gyakorlatban azt jelenti, hogy az orsók és a fej között egy bizonyos szalagrészt átmenetileg tárolnak, pufferolnak. Ennek a szalagrésznek a mennyiségét érzékelőkkel figyelve, az orsók mozgását vezérelni (elinditani vagy leállitani) lehet. A továbbitó görgőnek és a féknek ilyen módon csak ezt a pufferolt szalagmennyiséget kell tudni mozgatni, megállitani. (Az orsókon levő szalagot viszont már nem!) A jelenlegi mágnesszalagegységek kis gyorsitási, lassitási ideje csak ilyen módon biztositható.

A szalag mechanikai pufferolására két megoldás létezik:

- lengőkaros (követőkaros)
- vákuumoszlopos

A lengőkaros megoldásnál az alábbi,152. ábrának megfelelően az egyik végükön rögzitett karok elmozdulhatnak a nyilakkal jelzett irányba. Igy a továbbitógörgőkhöz szalagot képesek biztositani, vagy az orsókról lejövő és felesleges szalagmennyiséget fel tudják venni, tárolni tudják. A szalag feszitő

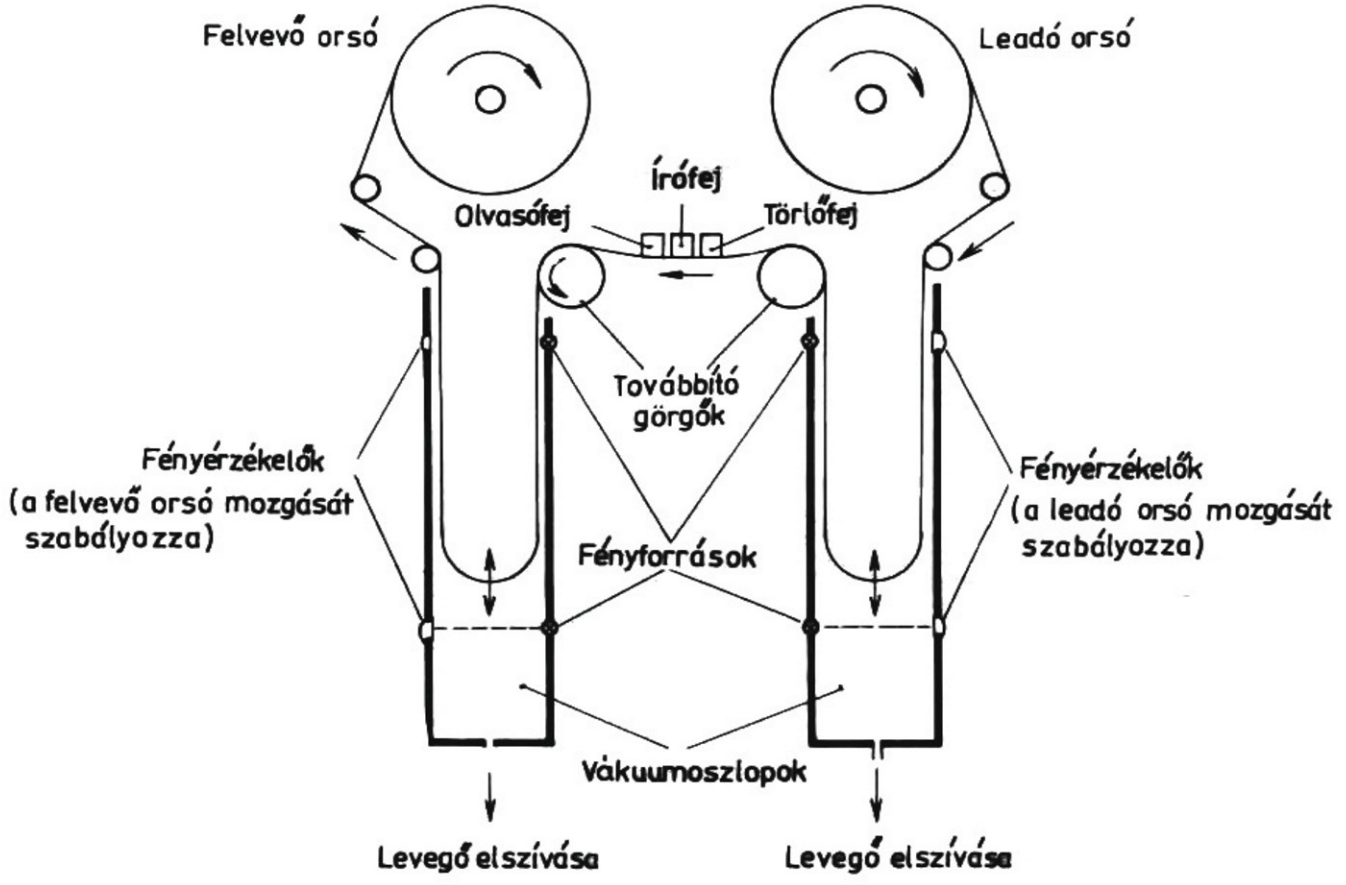


152. ábra. Lengőkaros szalagpufferolás

erejével a rugók tartanak egyensulyt. A karok elmozdulása által vezérelt mikrokapcsolók vagy elektronikus kapcsolók vezérlik az orsó meghajtó motorját (bekapcsolják, leállitják.) A lengőkaros megoldás hátránya, hogy a szalag mágneses felülete hozzáér a görgőkhöz, surlódik, kopik, valamint, hogy a kar mozgása, lengése is erősen igénybe veszi a szalag anyagát. Mágnesszalagos egységekben ezért ezt a megoldást már nem igen alkalmazzák. (Lyukszalagos egységekben a szalag pufferolására ez a megoldás viszont igen elterjedt.)

Kedvezőbb tulajdonságai folytán ma mágnesszalag pufferolására szinte kizárólag az un. vákuumoszlopos megoldás használatos. Itt a puffertár szerepét két vákuumoszlop (vákuumkamra) látja el. A vákuumoszlop egy felülről nyitott tartály, szélessége ugyanolyan, mint a szalagé. Müködés közben a szalag ezt az oszlopot felülről többé-kevésbé lezárja, igy a szalag alatti térben vákuum hozható létre. (A levegőt a vákuumoszlop alján szivják ki.) Ez a vákuum szivja (illetve a külső levegő nyomja) be a szalagot a vákuumoszlopba annyira, amennyire ezt a szalag pillanatnyi letekert, átmenetileg tárolandó hossza megengedi (153. ábra).

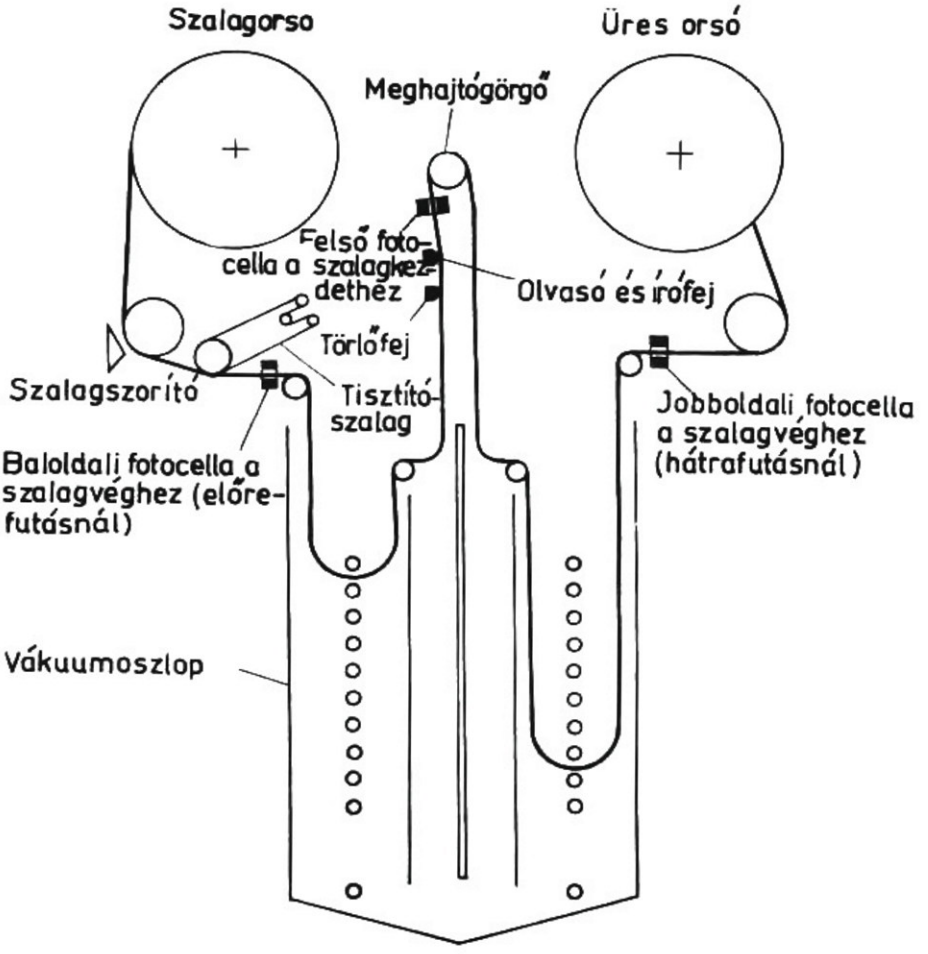
Ha az egyik hurokból a feldolgozás során szalagra van szükség, akkor a berendezés ezt a felette levő orsóról pótolja. Ha sok szalag került viszont a hurokba, a felesleget a hozzá tartozó orsó felcsévéli. A vákuumoszlopban levő szalag mennyiségének figyelésére és ennek alapján az orsók vezérlésére több megoldás is használatos. Van, amikor a szalag egy fénysugár utját zárja el, és igy a különböző magasságban levő érzékelő elemek jelzik a szalaghurok alját (lásd az előző ábrán!) Más megoldásnál a vákuumoszlopban levő lyukakhoz csatlakoztatott nyomásérzékelők segitségével állapitható meg a sza-



153. ábra. Vákuumoszlopos szalagpufferolás

lag alja. Ha az egyik érzékelő már külső légnyomást, a másik viszont még vákuumot jelez, a szalaghurok alja a két lyuk között helyezkedik el (154. ábra).

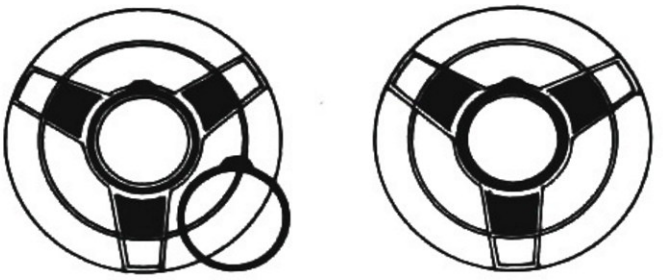
Ezen az ábrán csak egy meghajtógörgő látható. Ez a szalag továbbitási irányának megfelelően változtatja forgásirányát. Ez a megoldás az előbbiekben ismertetetthez képest lassubb, mivel nem csak a szalagot kell lefékezni, hanem a görgőt is, és utána ellenkező irányba kell gyorsitani.



Egy mágnesszalagos készülék felépítése

154. ábra.
Pufferolt szalag mennyiségének mérése nyomásérzékelők segitségével

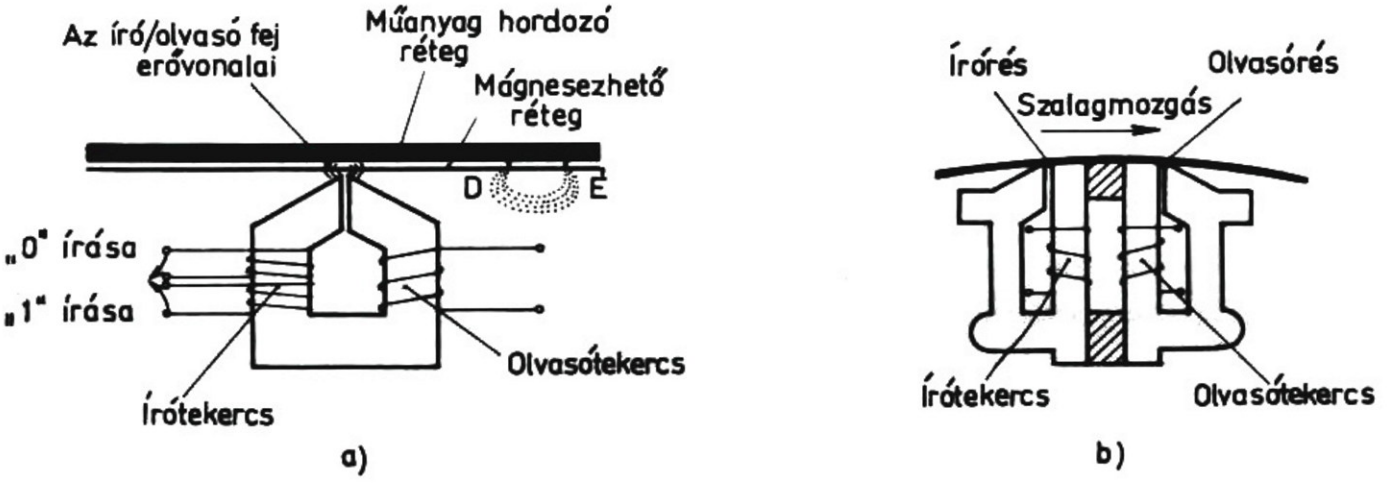
A vákuumoszlopok között helyezkedik el az olvasó, illetve az irófej. A korszerü berendezésekben a kombinált iró-olvasó fejek (ld. az alábbi 156. ábra "b" részét!) lehetővé teszik a szalagra felirt információ azonnali visszaolvasását és igy folyamatos ellenőrzését is, visszacsévélés nélkül. A szalagra történő iráskor egy automatikus törlési folyamat is végbemegy. Igy az előzőleg tárolt adatok megsemmisitésre kerülnek. Hogy tévedés, véletlen folytán ne lehessen a tárolt adatokat letörölni, az adatbiztositásra egy különleges biztonsági megoldást használnak, az un. irógyürüt (155. ábra). Iráskor ezt a müanyaggyürüt a szalagorsó egy rovátkájába kell helyezni, még a szalagnak a berendezésre való felrakása előtt. Irógyürü nélküli szalagra a berendezés nem tud irni! A szalagra felirt adatok ellenőrzéséhez nemcsak a már korábban emlitett keresztirányu paritást használják, hanem az un. hosszirányu paritást is. Itt nem karakterenként, hanem egy blokkon belül csatornánként, sávonként kerül ellenőrzés-



155. ábra. Irógyürü

re a bitek száma és ezután párosra vagy páratlanra történő kiegészitése. Ezek a csatornánként felirt bitek a blokk után egy külön ellenőrzőkaraktert alkotnak. Az olvasáskor a keresztirányu és a hosszirányu paritás alapján meg lehet állapitani az olvasás helyességét. (Gyakran a hosszirányu paritás ellenőrző karaktere előtt még ugynevezett ciklikus ellenőrző kódot is alkalmaznak a pontosság, a biztonság növelése érdekében.)

A szalagról történő olvasás nem jelent változtatást a tárolt adatokban, akárhányszor meg lehet azt ismételni. Egyes berendezések lehetőséget adnak a szalag visszafelé történő olvasására is. Ez például adatrendezés, keresés során igen meg tudja gyorsitani a feldolgozást.



156. ábra. Kombinált olvasó-iró fejek

ESZ-5010 (NML-67 tipusu) mágnesszalag egység

Az iró és olvasó mágnesfejegység kialakitása kombináltfejes, ez biztositja a felvétel (irás) után közvetlenül az ellenőrző olvasást. Az információ felirása a mágnesszalagra csak egy irányban történhet, akkor, amikor a jobb oldali (adagoló) orsóról a bal oldali (felcsévélő) orsóra megy a szalag.

Az információ olvasása a szalag mindkét irányu mozgása közben lehetséges.

Az információ törlését a szalag teljes szélességében a felirás során a törlőfej végzi. Ha a jobb oldali orsóról az irás engedélyező gyürü hiányzik, akkor a mágnesszalagegység csak olvasni képes. Ezzel lehet automatikusan megvédeni pl. a könyvtárszalagot a téves felirástól.

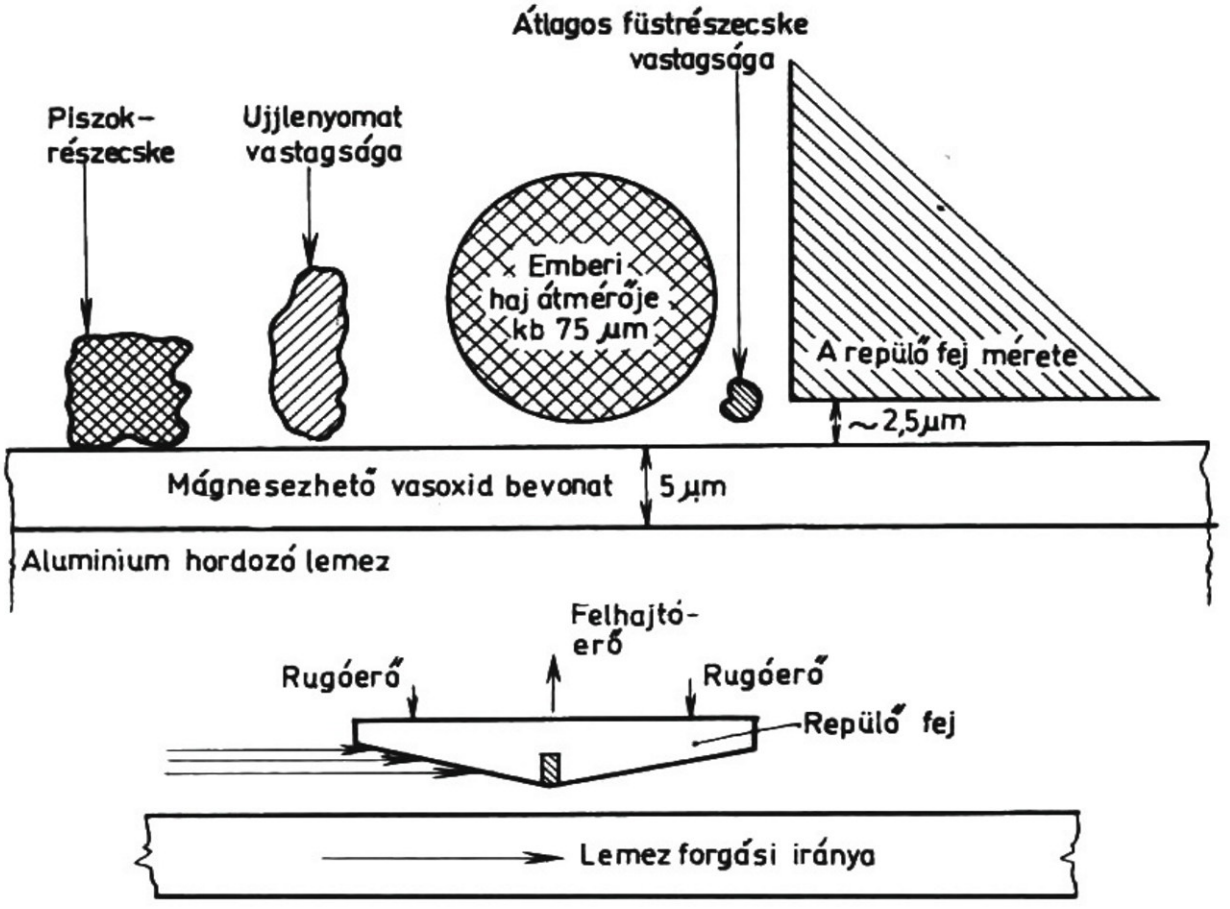
A mágnesszalag szélessége (mm)	12,7
A mágnesszalag vastagsága (μ)	45-55
A szalagtovábbitás sebessége irásnál és olvasásnál (m/sec)	2
Felirási sürüség (bit/mm)	8 és 32
Csatornák száma	9
Szalaghosszuság (m)	732
Start, stop idő (msec)	max. 6
Felirási mód	NRZ-I
Blokk-közök (mm)	25-30
Maximális áttekercselési idő üzemi sebességen (perc)	6
Szalagmozgató rendszer	pneumatikus továbbitás
Szalagpuffer	fotooptikai szalagérzékelő rendszer
Megbizhatóság	500 000 helyesen felirt bitre esik egy hibás bit
Automatika	jelfogós; tirisztoros vezérlésü egyenáramu motorok

4.4.2 MÁGNESLEMEZES TÁR

A mágneslemezes (mágnestárcsás) tárak közvetlen elérésü, viszonylag nagy kapacitásu (5-100 Mbyte), ugyanakkor kis hozzáférési idővel (20-150 msec) rendelkező tárak. Az adatok tárolására szolgáló mágnesezhető réteg (ferrit, Ni-Co) egy körlemez, egy tárcsa felületére van felvive. Üzem közben ez a tárcsa állandó, nagy sebességgel (1500-3000 ford/perc) forog az irő-olvasó-törlő fejek között. A mágneslemezegységben minden felülethez tartozik egy vagy több ilyen "fejblokk". Ezek a lemez sugarának irányában tudnak ki-be mozogni. A fejek azonban itt nem érnek hozzá a mágneses réteghez, mint a szalagnál, a nagy sebesség folytán képződő légpárna 2-5 μm távolságban tartja őket (un. repülő fejek). Az alábbi, 157. ábrán a könnyebb elképzelhetőség kedvéért ezt a távolságot néhány a gyakorlatból "ismert" tárgy méretéhez viszonyitjuk, és a repülő fej kialakitását is bemutatjuk.

A körlemez felületén az adatok koncentrikus körök, ugynevezett sávok (track) mentén helyezkednek el. Ezek meglehetősen sürün vannak (2-4 sáv/mm) és igy felületenként számuk 100-250 közötti. Igen elterjedt a 203 sáv használata, amiből a legbelső 3 tartaléksáv. Ezek a sávok a legkülsőtől kezdve és befelé haladva sorszámmal un. sávcimmel vannak ellátva. A 203 sávos lemeznél például az üzemi sávok cime 000-tól kezdődik és 202-ig tart. Vannak olyan lemezek is, ahol a sávokat még további részekre is felosztják. Ezek az un. szektorok. Ezek száma "gyárilag" előre meg van szabva, és a kezdetük az un. szektorjellel meg van jelölve. Általában 16, 20, 24, 32 körcikkre osztják fel képzeletben a lemezfelületet, és igy minden egyes sáv 16, 20, 24 vagy 32 szektorból áll. Ilyen esetben ezek a szektorok a lemezen a fizikailag legkisebb cimezhető egységek.

Egy lemez kapacitása viszonylag kicsi (0,5-2 Mbyte), ezért az ilyen un. egylemezes kazettát ritkán használják (158. ábra). Ennek előnye kis kapacitása ellenére az, hogy könnyen szállitható, könnyen kezelhető.

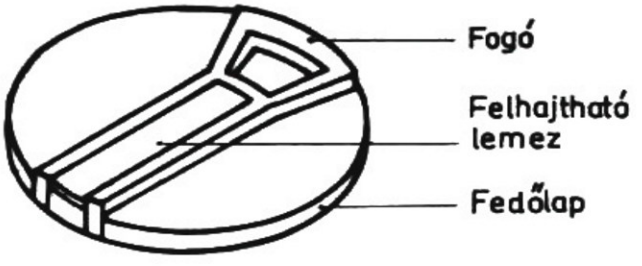


157. ábra. A mágneslemez és a fejek távolságának érzékeltetése és ennek megvalósitása

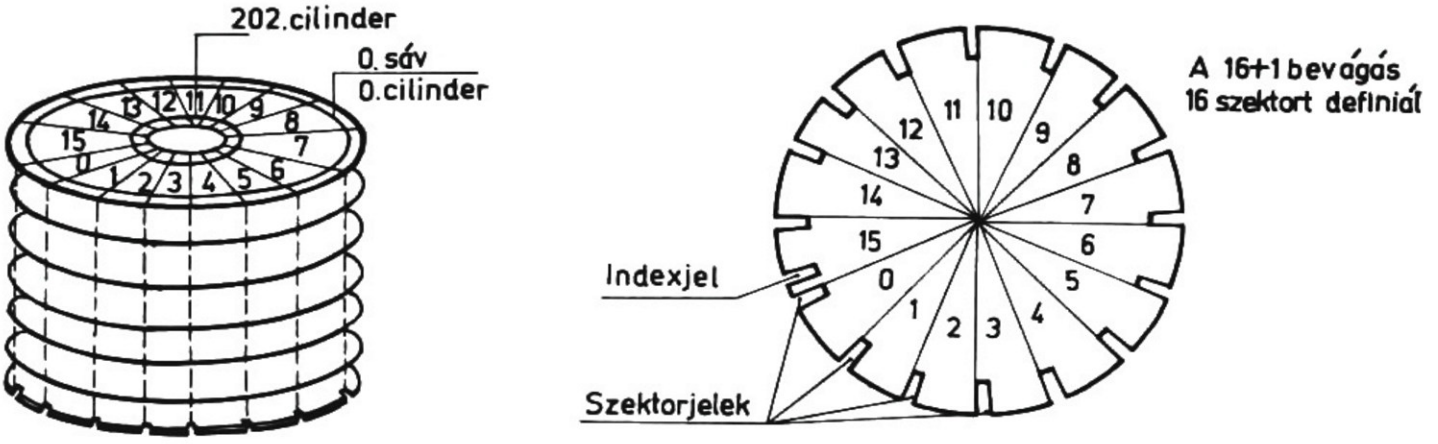
A lemezekből leggyakrabban un. lemezköteget, lemezcsomagot állitanak össze. Ezekben általában 4, 6, 11 lemez van egy központi tengelyre felerősitve. Ilyenkor a két legkülső lemezfelület a könnyü mechanikai sérülés miatt nem szolgál adatrögzitésre, csak "burkolja" a köteget. Igy a 6 lemezes köteg hasznos lemezfelületének száma 10, a 11 lemezesé pedig 20. A felületek is sorszámozva vannak. A legfelső hasznos felület a 0-adik, és attól kezdődik lefelé a számozás. A lemezköteg alá egy nagyobb átmérőjü, nem mágnesezhető, külön lemez is fel van erősitva (159. ábra). Ennek szélén jelölik meg fizikailag a koncentrikus sávok képzeletbeli "kezdetét" egy un. indexjellel. Ez általában egy bevágás, becsipés a lemez szélén. Ennek a jelnek az érzékelése alapján (fotodiódákkal) tudja meghatározni a lemezegység a sávok kezdetét. Ha a lemezen levő sávok még további szektorokra is fel vannak osztva, tehát a lemezköteg un. szektorszervezésü, a szektorok "kezdetét" jelentő szektorjel is ezen a legalsó lemezen található, és az indexjelhez hasonlóan ez is egy mechanikus jelezés. A szektorok sorszámozása az indexjeltől kezdődik, és ezeket a sorszámokat tekintjük a szektor cimének is.

Az alábbi, 160. ábrán egy korszerü 6-lemezes lemezköteg látható müanyag fogóval ellátott tokban, amelynek az alsó fedőlapját használat előtt le kell venni, s utána a lemezköteg a helyére illeszthető. A fogóval ellátott fedő csak az egységben való rögzités után távolitható el.

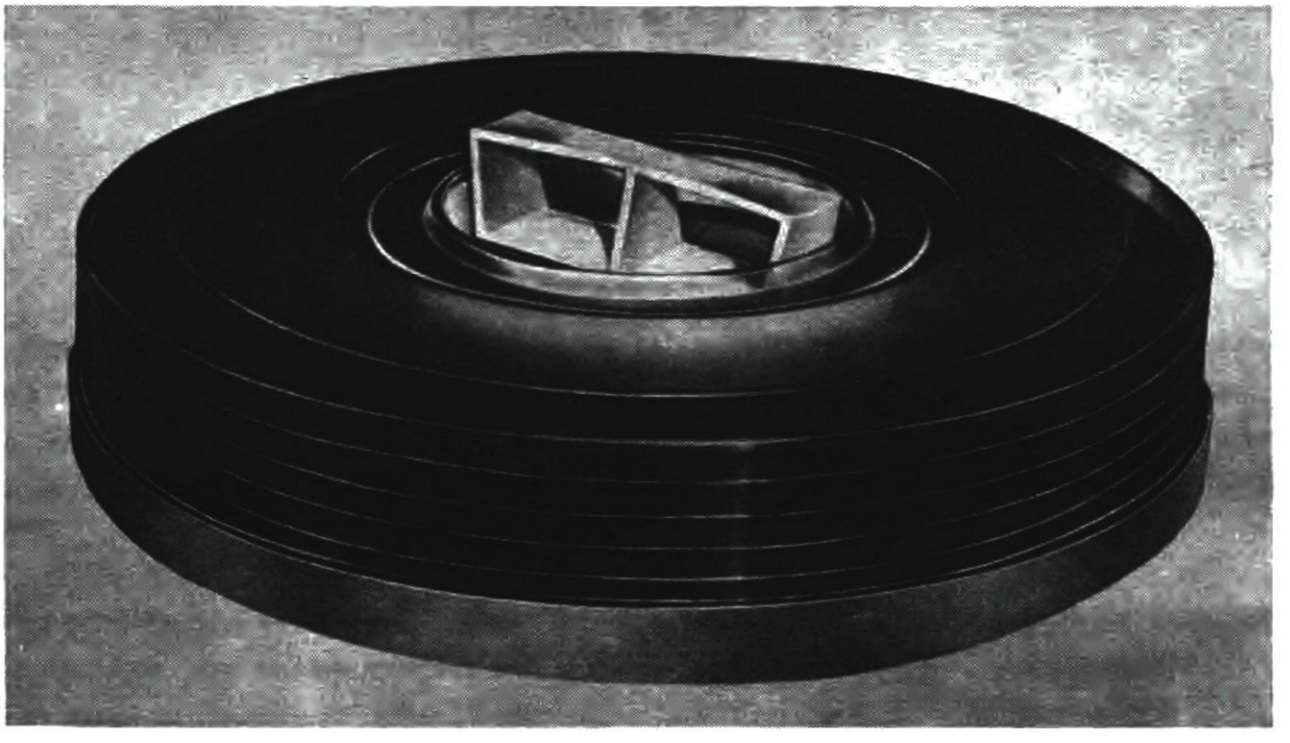
A lemezkötegek vagy fixen kerülnek a mágneslemezes egységbe beépitésre és ezek az un. fix-lemezes tárak, vagy cserélhetően és akkor ezek az un. cserélhetőlemezes tárak. A fixlemezes tárak-nál lemezfelületenként általában több fej is van, például 4, amelyek különböző irányokból nyulnak be a lemezfelületek fölé. Sőt van olyan megoldás is, ahol 200 fej van beépitve, fixen minden sáv fölé egy.



158. ábra. Egylemezes kazetta



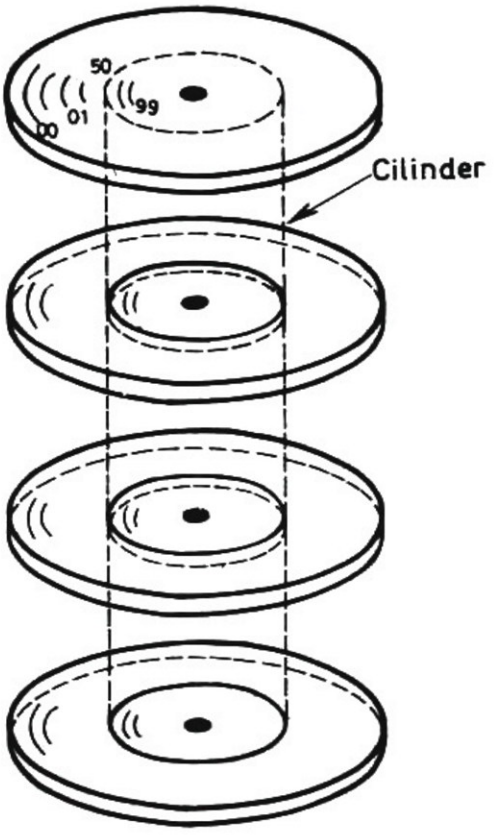
159. ábra. Szektorszervezésü lemezköteg felépitése



160. ábra. Hatlemezes lemezköteg lemezboritóval

Gazdasági megfontolások miatt azonban a legtöbb lemezköteg nem kerül fix beépitésre, és igy a tárkapacitás a cserélhetőség miatt szinte korlátlan. Itt minden felülethez általában csak egy fej tartozik. A lemezkötegen belül a fizikailag egymás alatt elhelyezkedő sávok képzeletben un. cilindert, vagy más néven palástot alkotnak (161. ábra). Ezeket a legkülső sávtól kezdve sorszámmal látják el, ez az un. cilindercim. A legkülső a 00-ás cilinder. Az ábrán egy 4 lemezből álló lemezköteget mutatnak be. A hasznos lemezfelület száma 6. Felületenként 100 db sáv van, 00-99-ig sorszámozva. Ennek megfelelően a cilinderek száma is 100. Mi csak az 50. cilinderhez tartozó sávokat rajzoltuk be.

A megfelelő sávra a fejeket a fejmozgató egység állitja rá. Ezek minden esetben együtt mozdulnak el vizszintes irányban. A fejek is sorszámozva vannak a hozzájuk tartozó felületnek megfelelően (felül-



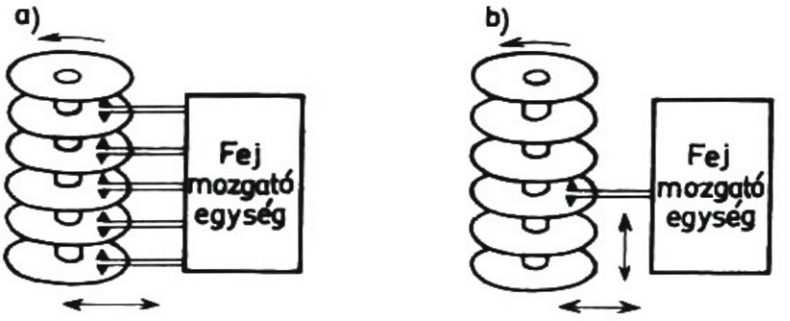
161. ábra. Az egymás alatti sávok cilindert alkotnak a mágneslemezkötegben

ről kezdve, 0-tól növekedve). Ez az un. fejcim. Az igy együtt mozgatott fejek a cilinder valamennyi adatsávját egyszerre érik el, de közülük csak egy lesz "aktiv", egyszerre csak egy dolgozik. Ha már a fejek a megfelelő cilindert elérték, az egy cilinderen levő adatokhoz a hozzáférési idő rendkivül rövid, mivel nem szükséges a fejek további mechanikus mozgatása, csak a megfelelő sávhoz tartozó fejet kell "bekapcsolni".

Az irás-olvasás elve hasonló, mint a mágnesszalagos tárnál, viszont itt az információ, a karakterek egyes bitjei a mozgási irányra nem merőlegesen, hanem a sávban egymás után helyezkednek el. A rögzités tehát sorosan történik. A koncentrikus körök hossza a lemez középpontja felé haladva ugyan egyre rövidebb lesz, de az irás-sürüség automatikus változtatásával minden egyes sávba ugyanannyi információ (pl. 3625 byte) irható.

A felirási idő illetve a fejmozgatások számának csökkentése érdekében az adatok elhelyezkedése és tárolása cilinderenként folyamatosan történik. Mit is jelent ez? Tételezzük fel, hogy az előző ábrán bemutatott 4 lemezből álló lemezkötegre akarunk irni. A tárolandó információ 15 sávot igényelne. Mondjuk a 4. cilindertől legyen üres helyünk, innen kezdve irjunk! A 4. és 5. cilindert (6-6 sáv) felülről lefelé haladva sorban mind teleirjuk, és a 6. cilinderen is felülről számolva 3 sávot lefoglalunk. Igy összesen csak 3 fejmozgatásra volt szükség. Ha egyetlen felületre irtuk volna fel mindezt, ötször ennyi, 15 fejmozgatás kellett volna. A 162. ábra "b" részén feltüntetett fejmozgató megoldást az adatok elérési idejének megnövekedése és a mozgató mechanizmus bonyolultsága miatt ezért nem is alkalmazzák.

A mágnesszalagos adatállománynál a kezelhető fizikai rekordok (blokkok) és a logikai rekordok kapcsolatáról elmondottak értelemszerüen alkalmazhatók a mágneslemezes tárak esetében is. A különbségek a kialakitható blokkok méreteiből következnek. A mágneslemez szervezésétől függően (szektor- vagy sávszervezés) a maximális blokkhosszuság eleve adott.

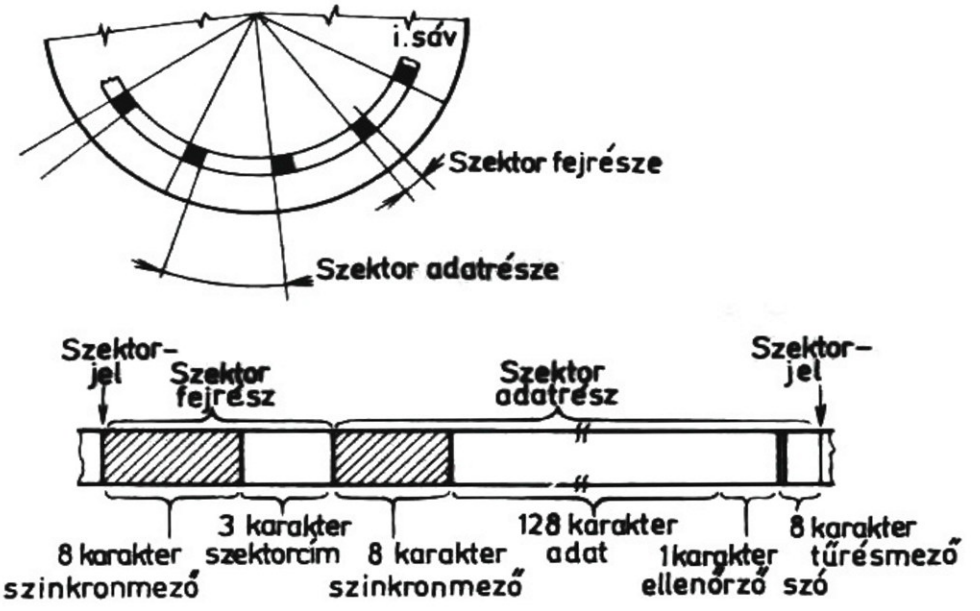


162. ábra. Fejmozgató megoldások

Szektorszervezés vagy más néven az adatok fix hosszuságu elrendezése esetén az információ eléréséhez meg kell adni a

- cilindercimet (melyik cilinderig kell a fejblokknak bemozdulni?)
- fejcimet (melyik sávhoz tartozó fejnek kell az összes közül aktivnak lenni?)
- szektorcimet (a sávon belül melyik szektorban van a keresett adat?)

Ezek az adatok a szektor un. fejrészében (cimrészében) helyezkednek el (163. ábra). Ezek segitségével ellenőrizhető, hogy a fejegység ténylegesen arra a cilinderre ment-e, hogy a megfelelő sávot választotta-e ki, és hogy a megfelelő szektorhoz jutott-e. Ugyanis nem ugy találjuk meg a cimet, hogy addig olvassuk a lemez tartalmát, amig a keresett cim meg nincs, hanem a szabályos elhelyezkedés folytán a fejmozgató mechanizmus egyből a kivánt helyre (cilinder) állitja be a fejeket és a megfelelő fejet az adott szektorjel után működteti.



163. ábra. A szektorok felépitése (szektorszervezés)

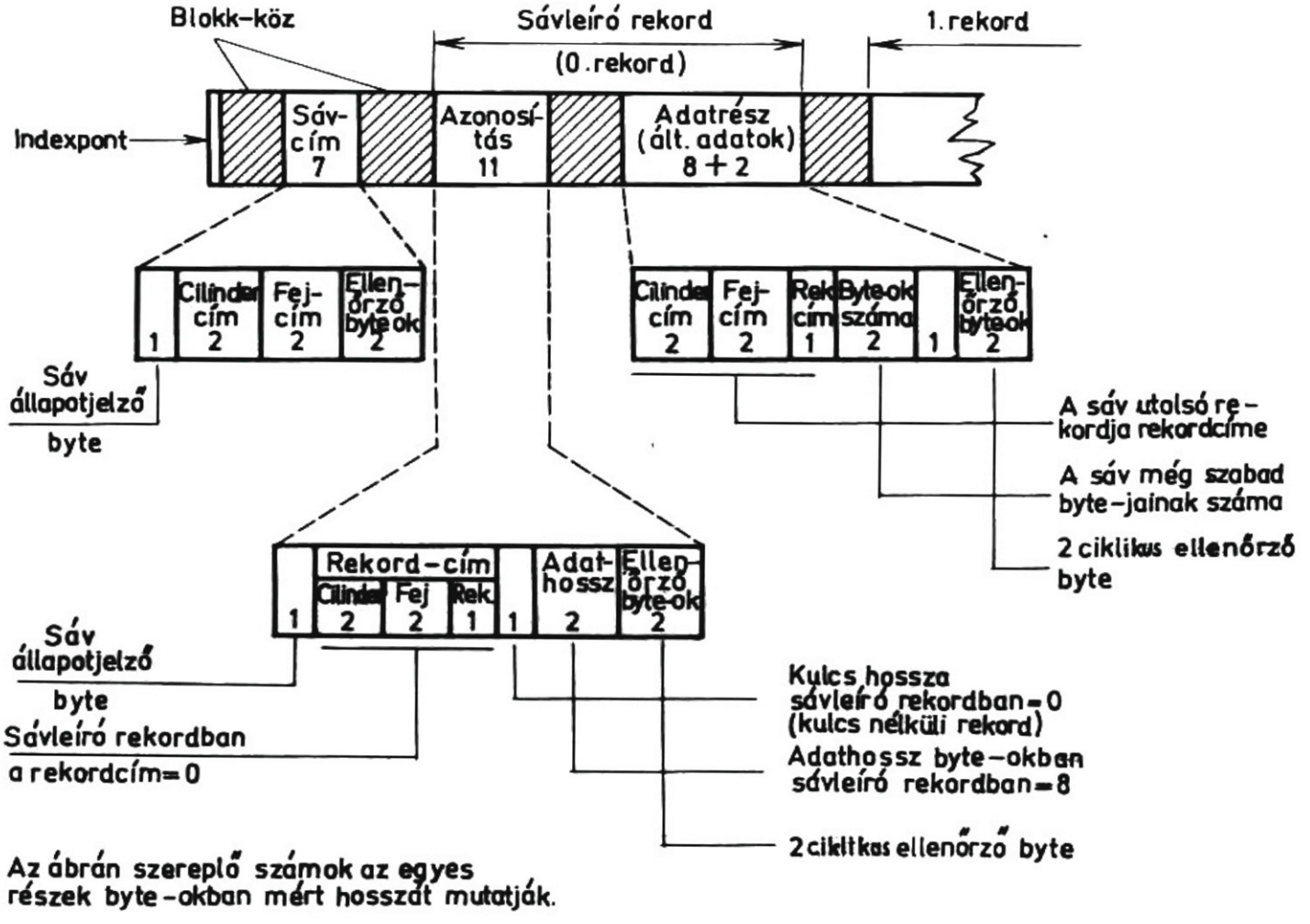
A tárolandó információ, az adatok a szektor un. adatrészében helyezkednek el. Ennek előre megszabott, kötött hosszából következően a file-t alkotó logikai rekordok elhelyezkedése az alábbi lehet:

- a logikai rekord hossza épp a szektor adatrészének hosszával megegyező. Ez a legritkább eset.
- a logikai rekord hossza vagy kisebb, vagy nagyobb a szektorban elhelyezhetőnél. Ez a gyakoribb eset. Megoldásként annyi szektort (egész számut) kapcsolnak össze, hogy a logikai rekord(ok) a leggazdaságosabban töltsék ki azt. Viszont itt, az adatrekordok után üresen maradó szektor - részek miatt, ahová már nem lehet ujabb adatokat irni, a lemez kihasználtsága rossz. Különösen nagyobb adatmennyiség esetén!

Ennek a problémának a kiküszöbölését célozza a lemezek ma leginkább használatos un. sávszervezése, vagyis az adatok változó hosszuságu elrendezése. Itt az információ hossza tág keretek között változhat, akár egy sávnyi is lehet. (Ez annak a módszernek a továbbfejlesztése, hogy a szektorszervezésü lemeznél a szektorbeosztást figyelmen kivül hagyva, a sávokat egységes egészként kezelte a berendezés.)

Foglalkozzunk ezzel egy kicsit részletesebben!

A közvetlen elérés érdekében a mágneslemezen minden rekordnak van egy un. kulcsa. Ezen kulcs (pl. termékszám, rajzszám) alapján van az adat tárolva, e szerint kereshető ki közvetlenül. Mielőtt azonban egy mágneslemezen adatokat lehetne tárolni, egy un. kiszolgáló programmal fel kell irni rá az indexpont után a sáv cimét (home adress) és az első, 0-ás rekordba az un. sávleiró rekordot (164. ábra). A lemez használatbavétele előtt meg kell vizsgálni valamennyi lemezfelületet is, mert



164. ábra. Sávcim és sávleiró rekord

ezek müszaki hibái az irás-olvasást lehetetlenné teszik. A nem megfelelő sávok helyett a berendezés tartaléksávot jelöl ki. A hibásnak itélt sáv sávleiró rekordjában, az adatrészben (1-4 byte) a tartaléksáv cime kerül megadásra. A tartaléksáv rekordjában pedig az azonositó a hibás sáv cimét adja meg. A lemezkötegben levő 3 tartaléksáv ugyanugy ellenőrzésre kerül, és a nem megfelelőt a berendezés megjelöli, az adatfelvételre már nem használható.

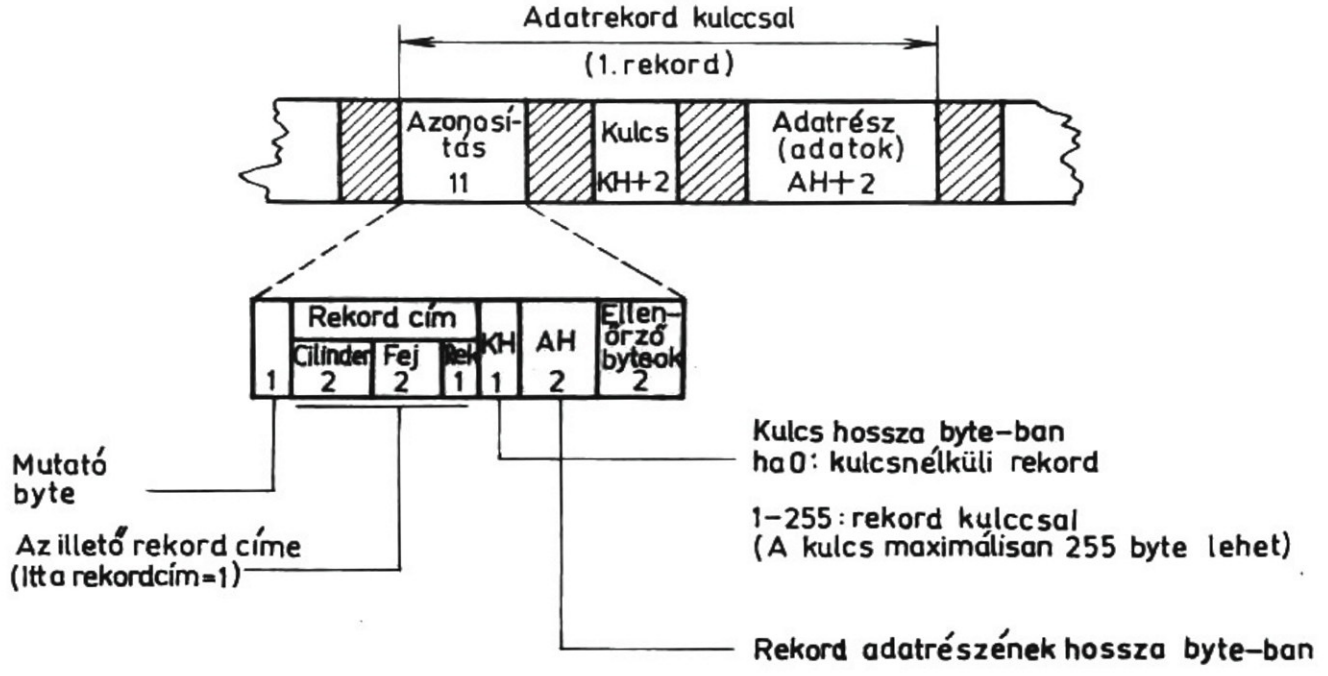
A sávcim első byte-ja ennek a vizsgálatnak az eredményeképpen a sáv pillanatnyi állapotát mutatja be, vagyis azt, hogy a sáv

- használható nem tartaléksáv
- használhatatlan nem tartaléksáv (sérült sáv)
- használható tartaléksáv
- használhatatlan tartaléksáv (sérült tartaléksáv)

A sávcim második és harmadik byte-ja a cilinder sorszámát, a negyedik és ötödik pedig a cilinderen belüli sáv (fej) sorszámát adja meg.

Az ez után következő 2 byte az ellenőrzéshez szolgál. A mágneslemezen minden mező végén a paritásbit(ek) helyett ciklikus ellenőrző kódot (Cyclic check) használnak, mivel ez itt előnyösebb. (Több hibafélesége ismerhető fel vele, kevesebb a helyigénye, gyorsitja az adatátvitelt stb.).

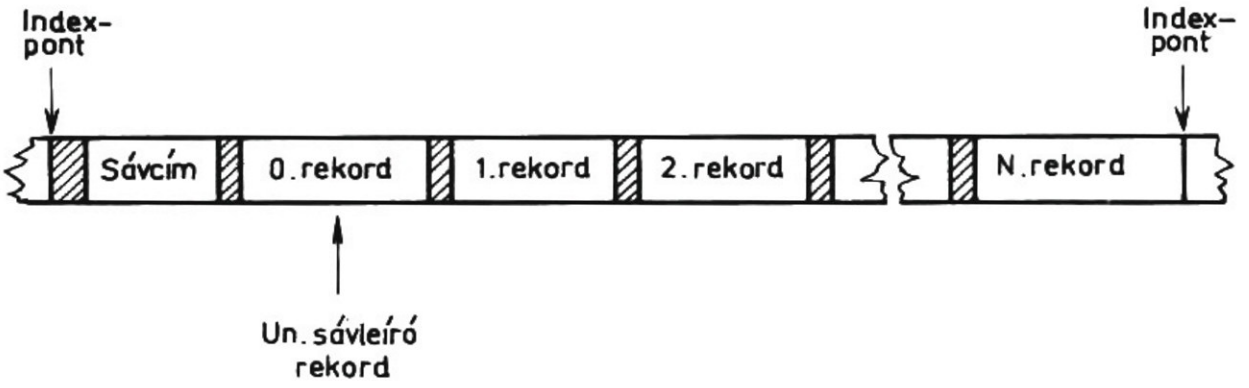
Ez után következnek a sávon az egyes rekordok. Mindegyik rekord egy un. azonositóval kezdődik, majd ezt követik az adatok (165. ábra). Az azonositó a rekordról illetve a sávról ad felvilágositást, megadja az illető rekord cimét, azonositókulcsának és az adatrészének a hosszát. A legelső, a 0-ás rekord e sávról közöl még információkat, ez az un. sávleiró rekord. Azonositó részének első 5 byte-ja megegyezik a sávcimben szereplőkkel. Ezt követi 1 byte-on az illető rekord sorszáma, cime, jelen esetben ez 0. A sávleiró rekordhoz nem használnak kulcsot, igy a következő 1 byte tartalma 0. Az un. általános adatok hossza az előirásnak megfelelően 8 byte (plusz két ellenőrző byte), tehát az adathosszt jelölő 2 byte-ban a 8 fog szerepelni.



165. ábra. Adatrekordok tartalma sávszervezés esetén

A sávleiró rekord adatrészében feltüntetett általános adatok a 8 byte-on a sávkapacitás nyilvántartását, ellenőrzését szolgálják. Ezért egyrészt a sávban levő utolsó rekord cime van itt megadva (az első 5 byte-on), másrészt a még felhasználható, a sávban levő még üres byte-ok száma (2 byte-on). Ezeket az adatokat a vezérlőprogram használja fel.

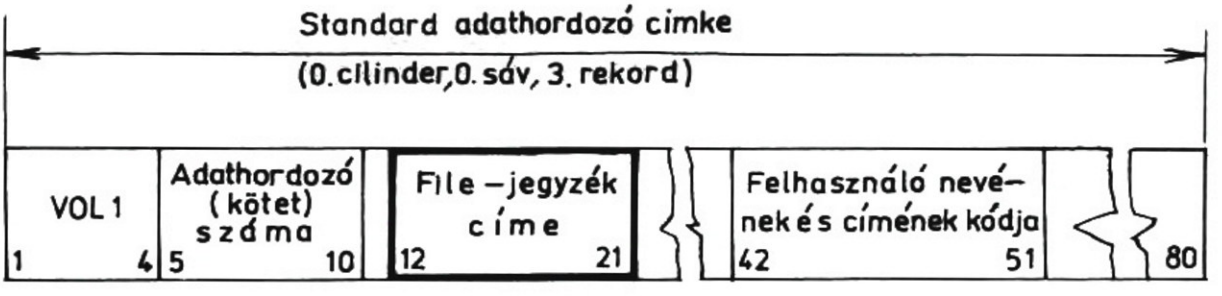
A sávleiró rekordot követően helyezkednek el az egyes adatrekordok (166. ábra). Számuk annyi lehet, amennyi a sávban elfér. A hosszuk tetszőlegesen változhat, de beosztásuk szabványositott. Az azonositó rész első byte-ja azt mutatja, hogy a rekord folytatása-e valamelyik másik rekordnak. Minthogy a tárolt adatokhoz alkalmazott kulcsok hossza és maga az adatrész is változó hosszuságu lehet, az azonositó 3 byte-ján ezek is megadásra kerülnek, még a felhasználás előtt. Az alkalmazott kulcsok hossza max. 255 byte lehet (mert 1 byte-on csak akkora hossz adható meg). Az adatok hossza (az adatrészé) $2^{16} - 1 = 65535$ byte, vagy annál kevesebb lehet, mert 2 byte-on több nem adható meg.



166. ábra. A lemez egy sávjának felépitése

Ahogyan a mágnesszalagnál, ugy a lemeznél és a többi közvetlen elérésü tárnál is vannak cimkék, amelyek segitségével a file-ok illetve a rekordok azonosithatók, ellenőrizhetők. A mágnesszalaggal ellentétben viszont megszünik a cimkék kettéosztottsága, nincs külön elő- és végcimke. Az adatokhoz képest az elhelyezkedésük sem annyira kötött, mint szalagnál, kivéve magát az adathordozó cimkét, amely mindig adott helyen, a 0. cilinder, 0. sáv, 3. rekordján helyezkedik el. Ez az un. VOL1 (Volume Label - kötet cimke). 80 byte hosszu, és tartalma hasonló a mágnesszalagnál levő hasonló cimkéhez. Tartalmazza többek között a lemezcsomag számát, a felhasználó (ügyfél) nevét, cimét és megadja a file-jegyzék pontos cimét a lemezkötegen belül (167. ábra).

A többi, lemeznél használatos kötelező (standard) cimke nem a file elején és végén van, mint a szalagnál, hanem a file-tól külön, a mágneslemez egy független tartományán (leggyakrabban egy külön



167. ábra. Mágneslemezen a kötelező adathordozó cimke (VOL1) tartalma

cilinderen). Ez az un. filejegyzék (VTOC - Volume Table of Contents). Tulajdonképpen a lemezen (a "kötetben") található file-ok "tartalomjegyzéke". A VTOC helyét a lemezkötegen belül, és a méretét a felhasználó határozhatja meg a lemezcsomag használatba vétele előtt. Az itt tárolt cimkék segitségével azonosithatók, ellenőrizhetők a lemezkötegben levő file-ok, megadják a file tényleges helyét (kezdő- és végcimét) az adathordozón, és leirják a file-ok tárolási, elérési módját (soros, index-szekvenciális, direkt). Ezek a cimkék 140 byte hosszuak és a felhasználástól függően 4 fajtájuk, 4 formátumuk van.

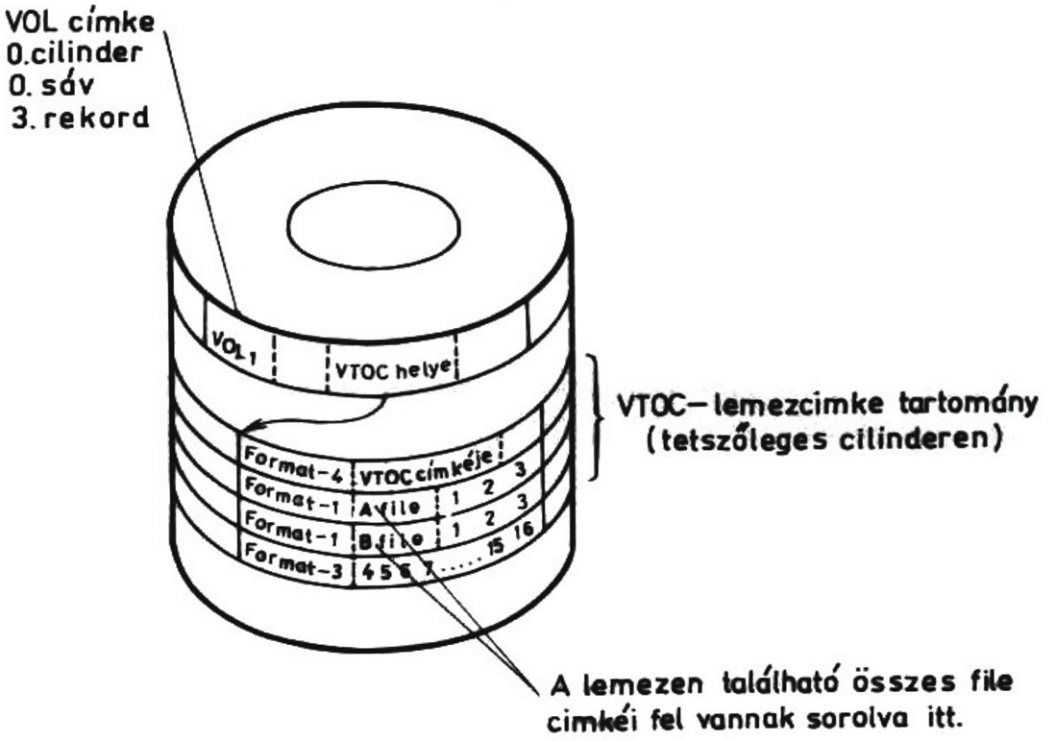
1. formátum (Format-1) Minden file-hoz szükséges, és a VTOC-ben ez áll az első helyen. Maximálisan 3 lemezterületet tud megadni.

2. formátum (Format-2) Ha a file index-szekvenciálisan van tárolva, az 1. formátum kiegészitőjeként, közvetlen utána következik. A cilinderek felosztásáról, az egyes indexek elhelyezkedéséről ad felvilágositást.

3. formátum (Format-3) Ha a file több, mint 3 lemezterületen helyezkedik el, tehát az 1. formátummal már nem adható meg, akkor az 1. formátum kiegészitőjeként kell ez. Az ott megkezdett területfelsoroláshoz további 13 terület adható még meg igy.

4. formátum (Format-4) A VTOC-t az operációs rendszer külön file-ként kezeli, igy neki is van cimkéje. Ez a 4. formátum, ami a filejegyzék (VTOC) első helyén áll, és a lemezen található file-ok cimtartományán kivül megadja a lemezkötegben még rendelkezésre álló tartaléksávok számát és az első ilyen cimét. .

Ha a mágneslemezen tárolt file-ra van szükség az adatfeldolgozásban, meg kell adni (pl. lyukkártyán) a file azonositó jellemzőit, amelyek alapján az operációs rendszer kikerestetni és ellenőriztetni tudja a cimkéket. (Először persze a VOL1 adathordozó cimke tartalma alapján megkeresi a lemezen a VTOC-t és utána abban keresi meg a megfelelő file 1. formátumu cimkéjét - 168. ábra.)



168. ábra. File-cimkék a lemez egy külön cilinderén (VTOC)

ESZ-5056 Mágneslemez-egység

Az információ a szabványos, cserélhető lemezcsomagokon helyezkedik el.

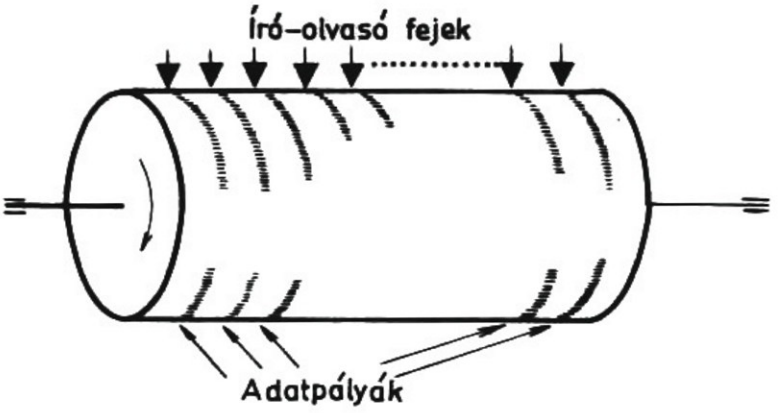
A lemezek bevonata Co-W tartalmu, a bevonat vastagsága 0,25-0,3 μm.

A cserélhető lemezcsomag kapacitása (bit)	$58 \cdot 10^6$.
Egy henger kiválasztásának maximális ideje (msec)	150
Egy henger kiválasztásának minimális ideje (msec) (szomszédos henger kiválasztása)	30
Egy henger kiválasztásának átlagos ideje (msec)	90
Az információ közepes hozzáférési ideje (msec)	105
Az információátviteli sebesség a vezérlőegységgel (bit/sec)	$1,25 \cdot 10^6$
Lemezek száma	6
Munkafelületek száma	10
Sávok száma lemezoldalanként	203
Kombinált iró-olvasó-törlő fejek száma	10
Fejmozgató szerkezet	közös mind a 10 fej számára
Felirási elv	elektromágneses
Felirás módja	frekvenciamoduláció
Felirási sürüség a külső sávokon (bit/mm)	30
Felirási sürüség a belső sávokon (bit/mm)	44
Az információ beiró sávjának szélessége (mm)	0,13
A lemezcsomag forgási sebessége (ford/perc)	2400
Sávosztás (mm)	$0,257 \pm 0,01$
Lemezátmérő (mm)	356

4.4.3 MÁGNESDOBOS TÁR

Az elektronikus számitógépekhez használt első mágneses elven müködő háttértár-berendezések a mágnesdobok voltak. Gyors müködésük miatt eleinte operativ tárként kisebb ferrittárak kiegészitésére is alkalmazták. Ez ma már háttérbe szorult, viszont néhány kisebb számitógépnél kis kapacitásu és gyors müködésü háttértárként alkalmazzák. A mágnesezhető réteg egy henger palástján helyezkedik el, és ez a henger fixen beépitve, állandó, nagy fordulatszámmal forog egy vizszintes vagy függőleges tengely körül. A dob átmérője általában 15-50 cm között változik.

Az információ ugynevezett sávokon vagy más néven adatpályákon helyezkedik el. Egy mágnesdobon a sávok száma 100-2000 között van. Minden sávhoz tartozik egy-egy iró-olvasó fej, amelyek a henger alkotójával párhuzamosan helyezkednek el (169. ábra). A fejeket a dob mentén általában csoportosan helyezik el. Egy fejcsoporton belül az egymás melletti fejek a sávok nagy sürüsége miatt nem egymás melletti adatpályákhoz tartoznak. A kihagyott sávok irása - olvasása egy másik fejcsoportba tartozó fejjel történik. Az összes sávot (a teljes dobfelületet) a fejcsoportok igy átlapolással érik el.



169. ábra. Mágnesdob felépitése

Az egy fejcsoporttal elérhető sávokat tulajdonképpen itt is tekinthetjük cilindernek. Egy kivánt adat eléréséhez maximálisan a dob egy körülfordulása szükséges. Az átlagos hozzáférési idő igy ennek a fele. Van olyan mágnesdobos berendezés is, ahol az elérési időt sávonként több fej alkalmazásával csökkentik. Például négy fej esetén az idő a negyedére csökken. Ez viszont külön áramkört tesz szükségessé, ami megállapitja, hogy a kivánt adat irásához - olvasásához melyik fej van az adott pillanatban a legközelebb. A dobok fordulatszáma a fix beépités miatt egész nagy érték is lehet, általában 2000-30 000 ford/perc között változik. Az átlagos hozzáférési idő ebből következően tehát meglehetősen kicsi, 2-30 msec. Egy egység tároló kapacitása 0,1-5 Mbyte közötti érték szokott lenni (20-400 bit/cm jelrögzitési sürüség esetén). A dobok fix beépitése miatt a kapacitás növelését csak ujabb egységek beállitásával lehet elérni, ez viszont tetemes költségnövekedést jelent. (A cserélhető mágneslemezek a felhasználónak előnyösebbek, igy kiszoritják általában a dobot.) Az adatátvitel sebessége 2-300 ezer byte/sec körüli érték. Ennek növelését és az átlagos elérési idő csökkentését (adott jelrögzitési sürüség esetén) a dobok nagyobb fordulatszámával lehetne biztositani. Ennek azonban már technikai-müszaki akadályai vannak (dobban ébredő feszültségek, csapágyazási problémák stb.).

A mágnesdobok nagy kerületi sebessége (40-100 m/sec) miatt itt sem engedhető meg, hogy a fejek érintkezzenek a dob felületével. A szokásos fej-réteg távolság fix rögzitésü fejeknél 5-20 μm. Ennél kisebb légrés (akár 2 μm-nél kisebb is) a már ismert "repülő fejek" alkalmazásával érhető el. Ebben az esetben a dob nagy kerületi sebességét használják ki, ami a dobfelület és a megfelelően kialakitott fejek között egy légpárnát hoz létre. Ez a fejeket a felülettől eltávolitani igyekszik. A fejek üzemi távolságát egy rugó biztositja. Mivel alacsonyabb fordulatszám esetén a légpárna nem jön létre, a fejek a megfelelő pozicióba csak a névleges fordulatszám elérése után kerülnek.

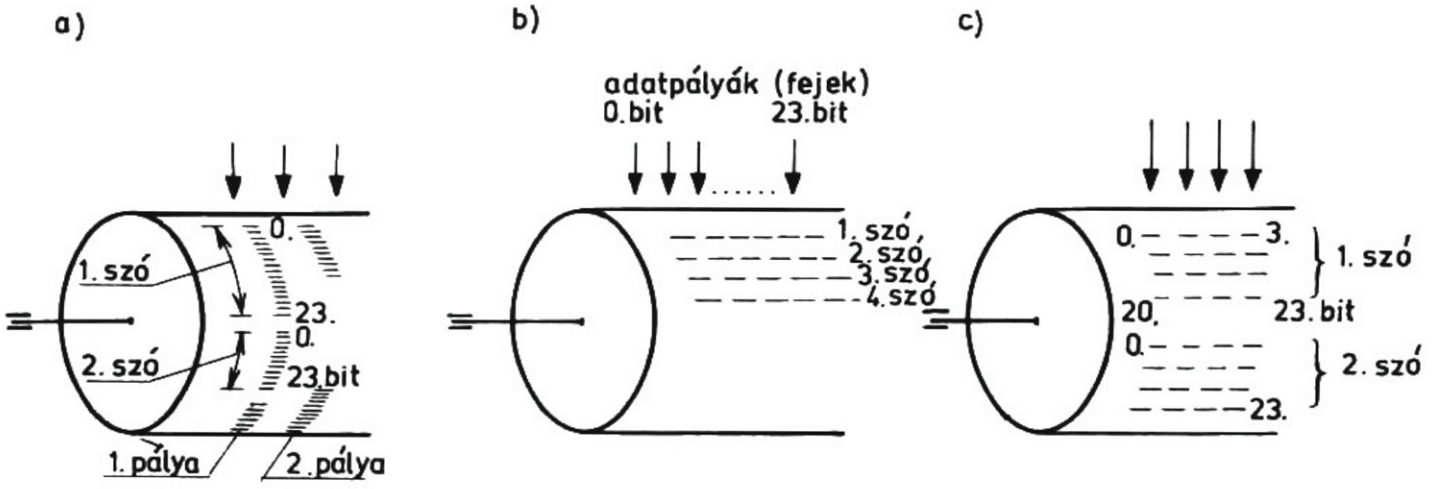
Az adott információ (byte vagy szó) a dob felületén a sávokon háromféleképpen helyezkedhet el: sorosan, párhuzamosan és soros-párhuzamosan (vegyesen).

Legelterjedtebb a soros elhelyezés, az ugynevezett soros üzemmód. Ekkor az adott információ egyes bitjei a kiválasztott sávon sorban egymás után helyezkednek el. Egy pályára általában néhány száz szó irható. Az információ irását, illetőleg olvasását egy fej végzi. Amint egy pálya megtelt, áttér a berendezés a következő sávra és ott folytatja az irást. Soros üzemmódnál nincs lehetőség egyetlen szó megcimzésére, ezért egy utasitás hatására, egyszerre egy egész adatpálya tartalmát kell irni vagy olvasni. Egy pálya tartalma ilyenkor tulajdonképpen egy blokknak felel meg.

A ritkábban alkalmazott párhuzamos üzemmódnál egy szó vagy byte összes bitje egy alkotó mentén helyezkedik el. Mégpedig annyi sávon, ahány bitet tartalmaz a szó vagy a byte. Az összes bit írása vagy olvasása azonos időpontban, egyszerre történik. Ennek megfelelően egy időben több fej dolgozik párhuzamosan. Az írás-olvasás sebessége igy igen nagy lehet, viszont a sok fej egyidejü müködésének vezérlése, szinkronizálási járulékos problémákat okoz.

A soros - párhuzamos üzemmódban a két előző megoldás hátrányait igyekeznek kiküszöbölni. Egyrészt gyorsabb, mint a soros, mivel az információ egyes részei (pl. 4-8 bitje) egyszerre, párhuzamosan kerülnek rögzitésre. Viszont nem olyan bonyolult a müködtetés, mint a párhuzamosnál, mivel egyidőben csak néhány (4-8) fej dolgozik. Az "információrészek" igy egymás után, sorosan kerülnek tárolásra.

Párhuzamos és soros - párhuzamos üzemmódnál legtöbbször lehetőség van már egyetlen szó megcimzésére, kikeresésére is. Az alábbi, 170. ábrán az információ elhelyezkedése látható a három üzemmódban, 24 bites szavakat feltételezve.



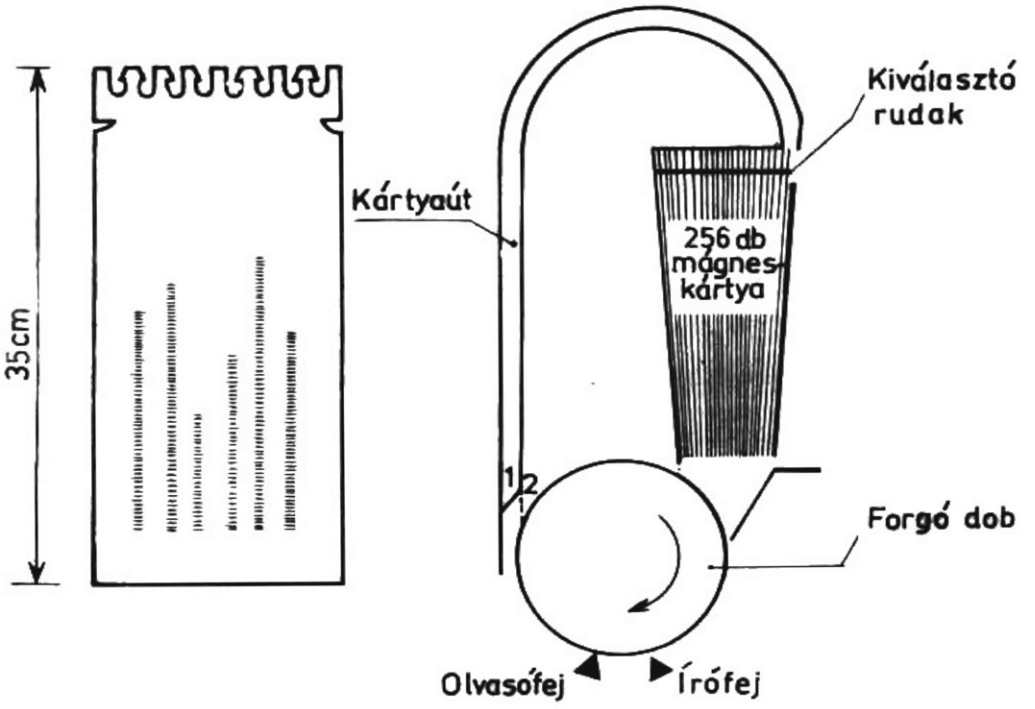
170. ábra. Adatok elhelyezkedése a mágnesdobon

4.4.4 MÁGNESKÁRTYÁS TÁR

A mágneskártyás tár a mágnesszalag nagy kapacitását és a mágnesdob kis elérési idejét igyekszik egyesiteni magában. Bár kapacitásához (100-1000 Mbyte) képest az átlagos elérési idő viszonylag kicsi (100-500 msec), beállitására nagy ára miatt ritkán kerül sor.

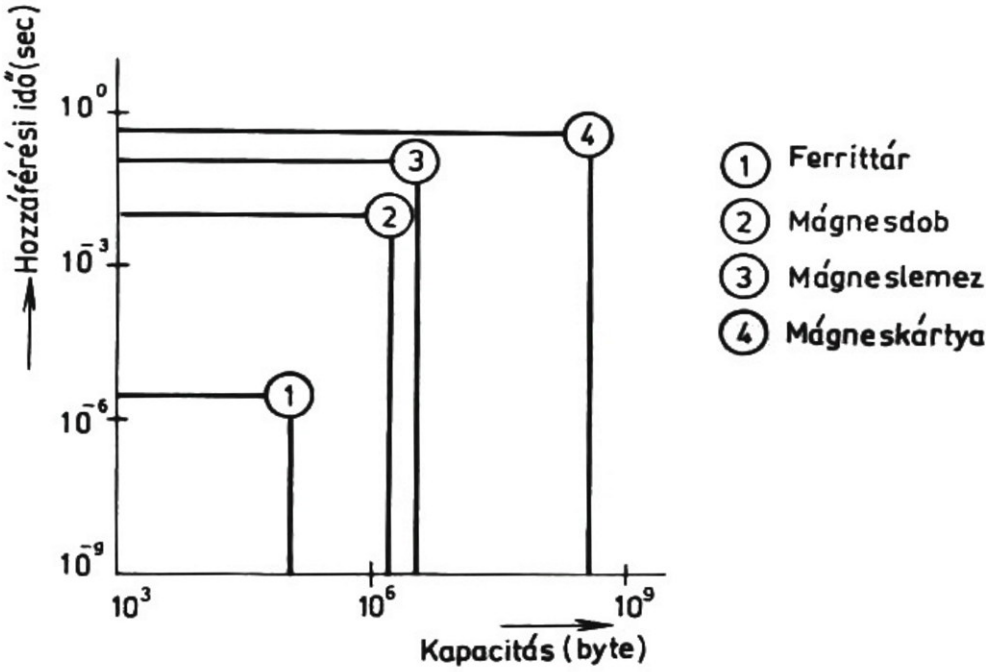
A mágneskártyás egység főbb részei a cserélhető kártyatárak (magazinok), ahol a kártyák tárolásra kerülnek, a gyorsan forgó dob, amely a kártyát hordozza az olvasás-irás müvelete alatt, és a kettőt összekötő kártyaut.

A mágneskártya hajlékony müanyag lap, egyik oldalán mágnesezhető vasoxid (ferrit) réteggel bevonva. A kártyákon az információ cimezhető sávokon, esetleg blokkokra tagolt sávokon, sorosan helyezkedik el. Egy-egy kártya kapacitása kb. 50-300 kbyte! A kártya szélén (hosszabb vagy rövidebb oldalán) a kikeresésre, kiválasztásra szolgáló bevágások vannak (171. ábra). A kivánt kártya kiválasztása ezeknek a bevágásoknak alapján valamilyen mechanikus szerkezet segitségével történik. A kártyá-



171. ábra. Mágneskártya és mágneskártyás tár

kat tartó rudak olyan állásba rendeződnek, hogy a kivánt kártya saját sulyánál fogva leesik és a gyorsan forgó dobra kerül. (A kártya a dobfelület csak mintegy kétharmad részét takarja le.) A forgó dobon levő kártya mágneses réteggel bevont oldala az iró-olvasó fejek előtt halad el, igy a párhuzamos sávokon végbemehet az adatcsere. A mágneskártyás tár tulajdonképpen "cserélhető dobfelületü" mágnesdobnak felel meg. A cserélhetőség miatt viszont a kapacitása lényegesen nagyobb annál. Egy kártyatárba általában 256 kártya fér el s egy egység több (8-16) kártyatárral is rendelkezik! Az igy elérhető több százmillió byte-os kapacitás a többi közvetlen elérésü tárral (lemez, dob) ma még nem biztositható (172. ábra). A mágneskártyás tárak rendszerint nem rendelkeznek annyi iró-olvasó fejjel, amenynyi a kártyán levő sávok száma. A sávok száma 50-200, a fejek száma pedig általában csak 10-20.

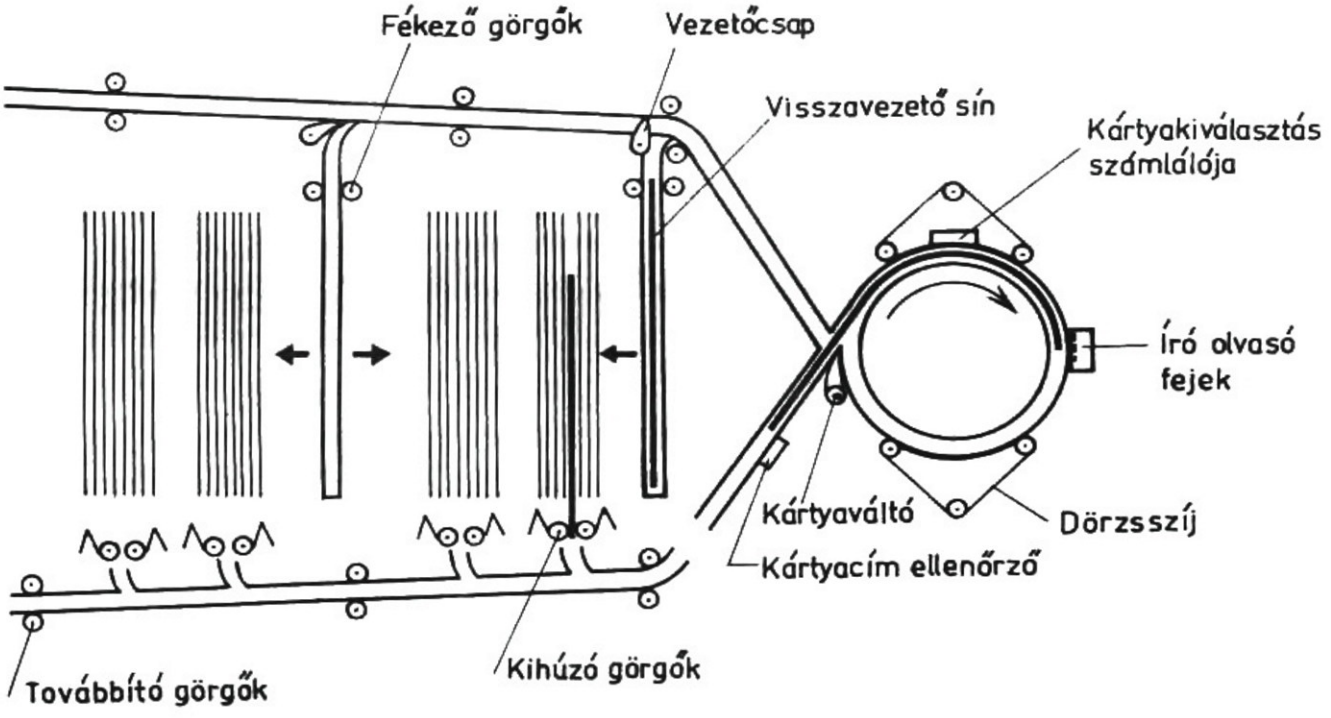


172. ábra. Kapacitás-hozzáférési idő összefüggése közvetlen elérésü tároknál

A fejek közül mindig csak egy olvas vagy ir, tehát a berendezés sorosan dolgozik. A cimzés szempontjából a fejegység egy pozíciójából elérhető sávokat igy itt is tekinthetjük cilindernek. A megfelelő sáv eléréséhez a mágneslemezes tárakhoz hasonlóan a fejegységet itt is mozgatni kell, megfelelő pozicióba kell vinni. Az adatátvitel sebessége 50-100 kbyte/sec.

A dob felületén a kártya helyzetét speciális mechanikus (pl. dörzsszijas) vagy pneumatikus rendszer biztositja. Ezek addig tartják ott a kártyát, amig szükség van rá, vagyis amig az adatcsere be nem fejeződött. A kártya pályáját a kártyaváltó állása határozza meg. Amig az zárva van, (1-es állás), a kártya a dobra feszül. A kártyaváltó nyitott állása esetén (az ábra jelölése alapján a 2-es állásban) a kártya visszakerül a kártyautba, majd a kártyatérbe. Ez után a folyamat kezdődhet elölről egy másik kártyával. Az átlagos hozzáférési idő a mechanikus kiválasztás és továbbitás következtében nagyobb, mint a többi közvetlen hozzáférésü tárnál.

A mágneskártyás tár elvi müködésének egy másik változata látható az alábbi ábrán (173. ábra). Amikor a dob felé halad, a kártya szélén levő bevágások optikai leolvasása alapján ellenőrzésre kerül a kártya száma. Igy a mechanikus kiválasztás hibái idejében felfedezhetők.



173. ábra. Mágneskártyás tár egy más fajta felépitése és müködése

A kártyák felhasználhatóságának, kopásának mérésére a kiválasztások számát magán a kártyán regisztrálják. Az előre tapasztalati uton megadott maximális kiválasztási számot (10-30 000) minden ujabb felhasználásnál 1-gyel csökkentik. A nulla elérésekor a kártya tartalmát egy uj kártyára át kell másolni.

4.5 INTERAKTÍV FELDOLGOZÁS INPUT/OUTPUT ESZKÖZEI

A számitástechnika fejlődése során előtérbe kerültek az ember és a számitógép kapcsolatának kérdései. Ennek megjavitására irányuló törekvések között elsőként szerepelt a rendszer és a környezet közötti párbeszédes üzemmódra való lehetőség megteremtése. A felhasználónak lehetőséget kellett adni, hogy munkájához (a programhoz) és annak feldolgozásához ne csak időszakosan, kártyakötegenként, "kötegelve" jusson hozzá, hanem tevékenyen részt vehessen benne, a feldolgozást figyelemmel kisérhesse, beleavatkozhasson, javithasson a programon, válaszolhasson a gép kérdéseire. Az ilyen "párbeszédes", ugynevezett interaktiv feldolgozás szükségessé tette a közbenső, másodlagos adathordozók (lyukszalag, lyukkártya, mágnesszalag) kiküszöbölését és a gépet használó ember számára a legkedvezőbb megjelenitési forma (irott szöveg, rajz) alkalmazását. Ezeket a lehetőségeket az input-output eszközök közül a

- karakternyomtató (konzolirógép) és a
- vizuális megjelenitő (display) biztositja pillanatnyilag.

4.5.1 KARAKTERNYOMTATÓ (KONZOLÍRÓGÉP)

Az ember-gép kapcsolat közvetlen párbeszédes (interaktiv) megvalósitásának egyik leggyakrabban alkalmazott eszköze a karakternyomtató.

A számitógéppel kapcsolatban alapvetően kétféle módon használható fel:

- konzolirógépként
- terminálként (adatvégállomásként)

A konzolirógép a számitógép (a rendszer) és az ember (környezet) közti közvetlen kapcsolatot biztositja. Egyrészt alkalmas kis mennyiségü adat, rövid információk bevitelére és kihozatalára, másrészt lehetővé teszi a gép operációs rendszerével a kapcsolatteremtést. Ennek segitségével a rendszer közölheti a feldolgozás állását, a müködéssel kapcsolatos üzeneteket, hibajelzéseket (pl. a program végét, a papircsere szükségességét stb.). Lehetőséget ad ugyanakkor a gépkezelőnek, hogy a szükséges utasitásokat, adatokat bevigye a rendszerbe, beleavatkozzék a feldolgozás menetébe (pl. programok behivása, inditása, hibajavitás, készülékkijelölés, üzemmódra való utalás stb.).

Komplett programok bevitelére lassusága miatt nem használják, bár az elvi lehetőség erre is megvan.

A terminálok tulajdonképpen olyan konzolirógépek, amelyeket a távadatfeldolgozásban adatvégállomásnak használnak. Vagyis olyan speciális perifériák, amelyek a rendszerbe nagyobb távolságról történő adatbevitelt, illetőleg a rendszerből származó adatok "szállitás utáni" kiirását, megjelenitését lehetővé teszik. Ilyen esetben előfordul, hogy a konzolirógépet lyukszalagolvasással is kombinálják. A lyukszalag itt a beviendő adatok tárolására, pufferelésére szolgál, és lehetővé teszi a karakternyomtató maximális hatásfokkal történő felhasználását. Ugyanakkor mint ellenőrző bizonylat is szerepel.

A karakternyomtatók előnyös tulajdonsága, hogy működtetésük az irógéphez hasonló billentyűzettel végezhető, működési sebességükben a felhasználóhoz alkalmazkodni tudnak és hogy a "párbeszédről" nyomtatvány is készül (esetleg több példányban is!).

A konzolirógép folytonos papirt használ (leporellót), amelyet a sornyomtatóhoz hasonlóan a két szélén levő lyuksor segitségével továbbit. A nyomtatható karakterek száma általában 52-120 között van, egy sorba a papir szélességétől függően 26-136 jelet tud irni. A klaviaturáján a kinyomtatandó karaktereken kivül még számos működtető, funkcionális billentyű is található. Ezek az adatforgalom lebonyolitását, illetőleg annak vezérlését szolgálják.

Működési sebessége változik a felhasználástól és a működési elvtől is függően, általában 10-150 karakter/sec közötti érték. Input esetében természetesen az operator "ügyességétől", irási sebességétől függ nagyrészt. Kiirásnál a géptől maximális sebességgel érkező információ egyrészt vezérli az irógépet (soremelés, sorkezdet, szóköz stb.), másrészt működteti azt, és megjelenik a papiron.

Pillanatnyilag a karakternyomtatók legelterjedtebb tipusai az ugynevezett ütközéses elv alapján működnek, azaz a karaktereket valamilyen mechanikus szerkezet papirra ütésével hozza létre (festékszalagon keresztül).

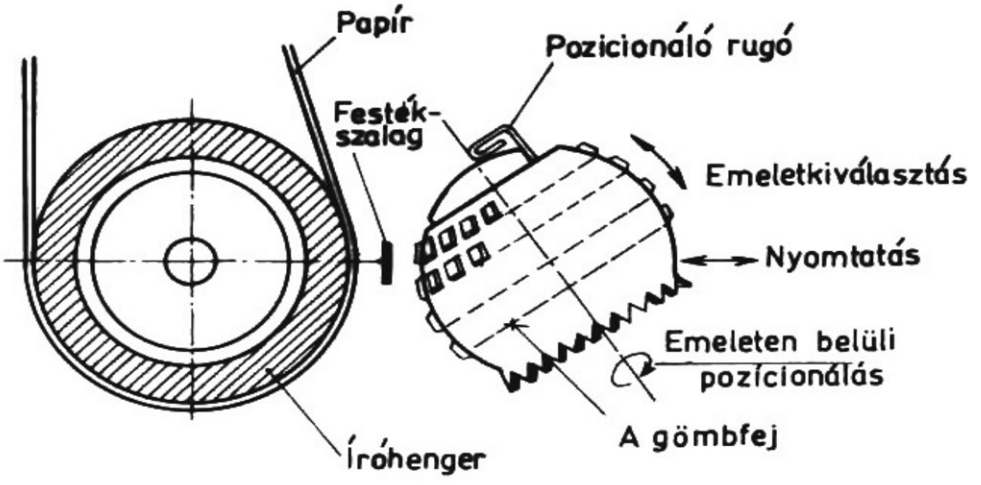
Ebbe a csoportba tartozik az elektromos irógép modernizált, átalakitott tipusán kivül a

- betükerekes (iróhengeres)
- gömbfejes,
- tűfejes (mátrixnyomtatós) konzolirógép.

1. A betükerekes karakternyomtatóban egy kerék kerületén egy vagy több sorban helyezkednek el a nyomtatandó jelek betütestjei. A kerék állandó fordulatszámmal forog és közben végighalad a kiirandó sor mentén. A papir mögött elhelyezkedő kalapácsok akkor ütnek a papirra, amikor az adott pozicióban nyomtatandó karakter jelenik meg a papir előtt. Ezzel a működési móddal körülbelül 40 karakter/sec sebesség érhető el. A betükerekes nyomtatók egy másik fajtájánál a kerék nem forog állandóan, hanem csak akkor, amikor a megfelelő betütest kiválasztása történik. A kiválasztás után pedig az egész kerék elmozdul, és ráüt a festékszalagon keresztül a papirra. Sebessége max. 35 karakter/sec. (Az iróhengeres karakternyomtatóban a karakterkészlet egy függőleges tengely körül forgó henger palástján helyezkedik el, egymás alatt több sorban. Az elnevezésbeli különbség is erre hivja fel a figyelmet.)

2. A gömbfejes karakternyomtatóban a jelkészlet egy gömbfelületen négy sávban, négy emeleten helyezkedik el (174. ábra).
A gömbfej cserélhető, igy a karakterkészlet igen könnyen cserélhető. A gömbfej billentésével biztositható, hogy a négy emelet valamelyike kerüljön a papirhoz közel. Az emeleten belüli karakter kiválasztása után a gömbfej a festékszalagon keresztül ráüt a papirra. Ez az egyik legszélesebb körben alkalmazott karakternyomtató, irási sebessége 14-15 karakter/sec.

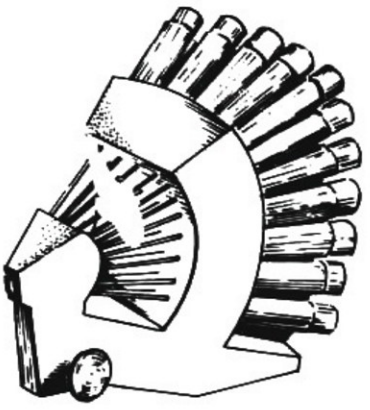
3. A tűfejes karakternyomtató működési elve a sornyomtatóknál emlitett mátrixnyomtatóhoz hasonló. Ennél nincsenek tárolva különálló karakterek, hanem minden egyes karaktert egy 5 x 7-es (vagy 7 x 9-es) "raszter"-ban elhelyezkedő pontok segitségével ir le (175. ábra).



174. ábra. Gömbfej nyomtatás közben

175. ábra. 5 x 7-es pontmátrix és karakterkészlet realizálása ennek segitségével

A pontok kinyomtatása "tük" segitségével történik. Ezek ütnek a festékszalagon keresztül a papirra. A tüfejes karakternyomtatók legtöbbjénél egyszerre nem az egész karakter kinyomtatása történik, hanem a raszternek csak egy oszlopa. Igy az ugynevezett nyomtatófejben (tüfejben) egymás felett csak 7 (esetleg 9) tü helyezkedik el. Ez a nyomtatófej nagy sebességgel halad végig a papir előtt és közben az egyes karaktereknek megfelelő oszlopokba "pontokat tesz". Egy karakter kinyomtatása tehát tulajdonképpen öt lépésben, az öt függőleges pontsornak megfelelően történik. Igy elérhető a 160 karakter/sec sebesség is! A tüfejes karakternyomtatóknak van olyan változata is, ahol a karakterek kinyomtatásához szükséges 5 x 7 = 35 tü egy fejegységbe van foglalva (176. ábra). Ez a karaktereket egy lépésben készíti el, és a papir előtt végighaladva egymás után jeleniti meg a szükséges betüket, számokat, irásjeleket. Ez a megoldás drágább, de ezt a nagyobb nyomtatási sebesség és az egyszerübb müködtetés, az egyszerübb szinkronizálás ellensulyozza.



176. ábra. Tüfej 5 x 7-es tümátrix-szal

Ezek az ugynevezett ütköztetéses elv alapján müködő karakternyomtatók zajosak, gyakori karbantartást igényelnek, előnyük viszont, hogy akár több példányos másolatot is képesek készíteni, és viszonylag nem is nagyon drágák.

A világpiacon vannak nem ütköztetéses elv alapján müködő karakternyomtatók is. Ezek - akár a sornyomtatónál emlitett néhány uj megoldás - elektrosztatikus-, hő- vagy fényhatás felhasználásán alapulnak. Szinte teljesen zajtalanok s igen megbizhatóak, viszont nagyon drágák, és csak egy példányos nyomtatványt képesek készíteni.

Ezek csoportjába tartozik a nagyobb gépeknél néhol már használatos ugynevezett konzoldisplay, vagyis a karakternyomtató helyett alkalmazott és a 4.5.2 fejezetben részletesen ismertetett vizuális megjelenitő. Ennek nagyon nagy előnye az, hogy akár a feldolgozás pillanatnyi helyzetét, akár a kiválasztott tárrész tartalmát képes állandóan a pillanatnyi helyzetnek megfelelően bemutatni (binárisan vagy hexadecimálisan). A rendszer müködése tehát állandóan figyelemmel kisérhető. Hátránya a nagyobb ár mellett az is, hogy nincs maradandó információ, nem készül a párbeszédről bizonylat. Hangtalan müködése is jelenthet hátrányt akkor, ha a gépkezelő nem ül épp előtte, igy a korábbi helyzet változását, az esetleges közbeavatkozás szükségességét nem veszi azonnal észre (ahogy az ütközéses elven müködő karakternyomtatós zajából erre következtethet).

ESZ-7070 konzol irógép

A "Consul-260"-as irógép billentyüzetéről lehet üzeneteket továbbitani a számitógéphez. Ezeket az üzeneteket, valamint a számitógép válaszait konzol papiron rögziti. Az átküldött információt byteonként páratlan paritásra vizsgálja. A berendezés mind multiplex, mind monopol (burst multiplex) üzemmódban működhet. A monopol üzemmódot a központi egység csatornájának kell kezdeményezni.

A nyomtatás sebessége (jel/sec)	max. 10
Egy sorba irható jelek maximális száma (db)	106
Jelkészlet (db)	93
A billentyük száma (db)	46
A készithető másolatok száma (db)	5
Az információ kódolása	átviteli kód (DKOI)

4.5.2 VIZUÁLIS MEGJELENÍTŐ (DISPLAY)

Az előző fejezetben tárgyalt karakternyomtató tulajdonképpen már lehetővé teszi a számitógép és az ember interaktiv kapcsolatát. Vele azonban a mechanikus működés folytán csak viszonylag lassu párbeszéd alakitható ki. A megjelenitett (input vagy output) adatok változtatása, javitása is meglehetősen körülményes. Ábrák, vonalak, diagramok megjelenitésére pedig egyáltalán nem alkalmas. Tehát nem minden tekintetben a legtökéletesebb eszköz! Hozzájuk képest a vizuális megjelenitők, a display-k már egy minőségileg jóval magasabb szinvonalu kapcsolatteremtést tesznek lehetővé! Az összes periféria közül ezek "alkalmazkodnak" a leginkább az ember igényéhez (leszámitva persze a beszéd megértését és a hangválasz adását!) Bár vannak olyanok is, amelyek csak output perifériaként használhatók, tehát szerepük a kapcsolattartásban passziv, a legtöbbjük beviteli üzemmódban, input perifériaként is dolgozik, vagyis aktiv ember-gép kapcsolatot tesz lehetővé. Több fajtájuk van. Közös jellemzőjük, hogy az információt egy katódsugárcső képernyőjén jelenitik meg.

A megjelenitett információ tartalma szerint három lényeges csoportjukat különböztetjük meg:

a) alfanumerikus display

b) grafikus display

c) speciális célu display

a) Az alfanumerikus display-k karakterek megjelenitésére szolgálnak, olvasható formában. Felhasználhatók a számitástechnika minden területén! Alkalmazhatók a számitógépnek "szóló" üzenetek, adatok összeállitásánál, rendezésénél, javitásánál és továbbitásánál, valamint a rendszertől "jövő" információ megjelenitésére, fogadására. Lehetővé teszik a gép tárában, rekeszeiben, regisztereiben levő adatok ellenőrzését, a "rendszerbe való betekintést". Tehát többek között a karakternyomtatók, a konzolirógépek feladatát is ellátják - zajtalanul. Ezért egyes számitógépeknél azokat teljes egészében ilyenekkel helyettesitik, a megjelenitést a papir helyett a katódsugárcső ernyőjén végezve. Adatgyüjtő rendszerek termináljaként, adatbeviteli alkalmazásoknál az inputot irányitó személy a képernyőn ellenőrizheti, hogy az adatok hibátlanok-e, és csak a minden tekintetben (formailag, tartalmilag) tökéletes információt továbbitja a rendszer felé. Vannak display-k, ahol a képernyő egyik fele a számitógép üze-

neteit, a másik fele a felhasználó parancsait tartalmazza, ezzel is megkönnyitve a tényleges párbeszéd létrejöttét. De megtalálhatók repülőterek, állomások, bankok információ rendszerében (pl. helyfoglalás, érkezési, indulási idők, valutaárfolyamok), laboratóriumokban, tervező irodákban nagy tömegü, gyors adatmegjelenitésre (pl. dokumentációk, szabványok), a vállalati folyamatok, a vállalatirányitás lekérdező, ellenőrző eszközeként, a gyógyászatban stb. A számitógépes oktatás (CAI), a programfejlesztés igen hatékony eszközt kapott benne, mivel a képernyőn megjelenő információkra a hallgatók közvetlen válaszadását teszi lehetővé és a helytelen adatok könnyüszerrel javithatók. Ilyen rendszerekben akár száznál több displayt is találhatunk.

b) A grafikus display-k az alfanumerikus karaktereken kivül ábrák, vonalak, függvények megjelenitésére is alkalmasak, mind output, mind input vonatkozásában. Felhasználásuk különösen előnyös az un. interaktiv tervezésnél (pl. épitészeti, közuti mütárgyak elhelyezkedésének, formájának kialakitásában), a térképkészitésnél (pl. meteorológiai), filmgyártásban (pl. reklám-, trükkfilmek) stb.
A grafikus információközlés előnye, az áttekinthetőség, a szemléletesség a fenti területeken azonban csak akkor használható ki, ha a felhasználó a számitógéppel on-line (közvetlen) kapcsolatot tud kiépiteni, módositani tudja az ábrát, törölni tudja egyes részeit, és mindezeket a rendszer közvetlenül érzékelni is tudja, a tárolt adatokon a változtatásokat végig tudja vezetni ("grafikus párbeszéd"). Ezért a grafikus display-k az alfanumerikusok billentyüzetén kivül még további eszközöket, kezelőszerveket is tartalmaznak (pl. fényceruza, pozicionáló gömb, grafikus tábla stb.) Ezekkel együtt a grafikus display-k olyan összetett rajzolási feladatokat is meg tudnak oldani, hogy vezérlésükhöz egy számitógép bonyolultságu berendezésre (display processor) van szükség. Ennek feladatai közé tartozik a központi számitógép igénybevétele nélkül az ábrák folyamatos vagy diszkrét lépcsőkben történő kicsinyitése, nagyitása, eltolása, elforgatása stb. Ilyen módon megoldható térhatásu ábrák rejtett vonalainak eltüntetése, a vonal intenzitások változtatása a dimenziók érzékeltetésére, szubrutinok által generált szabványos rajzelemek behivása bonyolult ábrák "építőkocka" szerinti összerakásánál stb. Ezekhez a display-khez gyakran mágneslemezes háttértár is csatlakozik, az "önálló munka" elvégzésének segitéséhez. A grafikus display-k még csak viszonylag szük körben használatosak, mivel áruk igen magas, az alfanumerikushoz képest gyakran 10-15-szörös. A minicomputerek elterjedésével viszont számuk várhatóan egyre nagyobb lesz.

c) A speciális célu display-k az alfanumerikus és grafikus információval egyidőben más információforrás adatainak megjelenitésére is alkalmasak. Ilyen "háttérinformáció" lehet pl. egy térkép, egy radarkép stb. ami a tervezéshez, a felhasználáshoz segitséget, támpontot ad. Ezt a tipust általában felügyelő rendszerekben használják.

A display-k az ábramegjelenitéshez adathordozót nem használnak. Ez egyrészt előnyös, mivel igy olcsóbb a "megjelenités", másrészt viszont hátrányos, mert a párbeszédről, az információról nem készül maradandó bizonylat, dokumentum. Ha ezekre mégis szükség van, akkor ellátják őket kiegészitő berendezésekkel, amelyek a megjelenitett információt rögziteni is tudják. Alfanumerikus display-k esetén a legegyszerübb megoldás, ha karakternyomtatót, konzolirógépet kapcsolnak hozzá, amelyen a végleges formáju és tartalmu információ ki is iratható. Gyakran azonban erre a célra speciális un. hard-copy berendezéseket használnak fel. Ezek felépülhetnek filmkamerával (pl. 16 vagy 35 mm-es), mikrofilmfelvevővel (COM), mátrixnyomtatóval, vagy akár rajzgéppel (pl. printer-plotter). Ezek működését ilyenkor a számitógép "koordinálja", vezérli.

A következőkben a display-kben alkalmazott karaktergenerálási módok közül mutatunk be röviden néhányat. Ezeknek a feladata, hogy a rendszerből jövő információk (az un. karakterkódok) alapján a felrajzolni kivánt karaktereknek megfelelően az elektronsugarat vezéreljék. (Az x és y irányu mozgást, valamint a fénysugár intenzitását - bekapcsolás, kikapcsolás - szabályozzák.) Minden karaktermegjelenitési utasitás esetén a karaktergenerátornak az illető karakterhez tartozó un. generálási rutint kell feldolgozni. Ezeket a rutinokat egy külön memóriában (karaktermemória - általában ROM) mikroprogramok formájában tárolják. A tárolt információ (mikroprogram) alapján történik az elektronsugár eltéritése, és a fényességének (intenzitásának) változtatása.

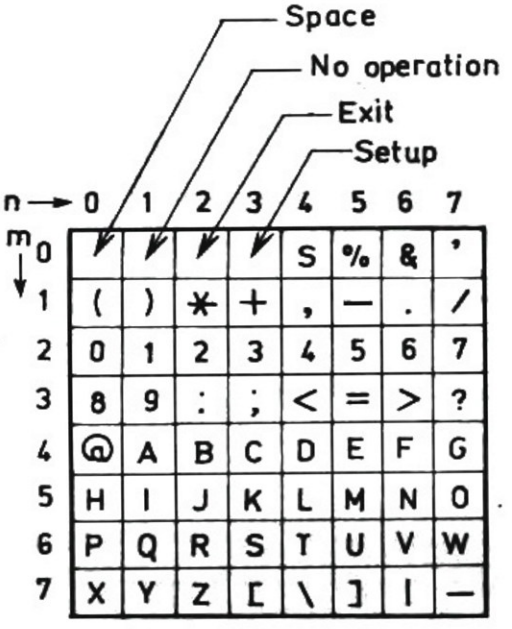
Attól függően, hogy a tárolt mikroprogram alapján hogyan történik a karakterek megjelenitése a képernyőn, három fő tipusról szoktak beszélni.

1. monoszkóp tipusu generátorok
2. pontfelvillanásos tipusu generátorok
3. vonalas tipusu generátorok

A három közül az első tipus speciális képcsövet igényel, a másik kettő viszont az általános tv-képernyőt használja fel a megjelenitésre.

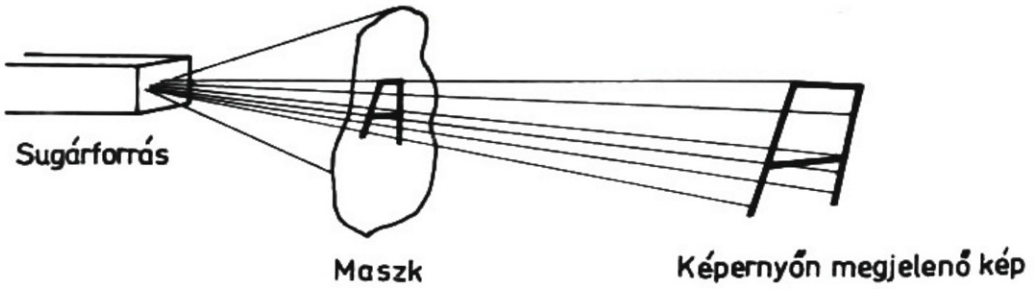
1. Monoszkóp tipusu generálás

Ennél a megjelenitendő karakterek képeit egy monoszkópcsőben elhelyezett un. képmátrix tartalmazza (177.a. ábra). A 8 x 8-as mátrixnak azt a karakterét, amit a karakterkód alapján a berendezés kiválaszt, egy elektronsugár letapogatja. Az erről visszaverődött sugarak (szekunder emisszió) alapján a megjelenitő képcsőben az elektronsugár intenzitását lehet változtatni (a karakter adott helyén bekapcsolni illetve kikapcsolni).



177.a. ábra. Karakterkép-mátrix

A monoszkóp tipusu generálás másik megoldási módjánál ez a képmátrix közvetlen a megjelenitő képcsőbe van beépitve, és a karakterek negativ képét tartalmazza (177.b. ábra). (A karakterek konturjai ki vannak egy lemezből marva.) Az elektronsugár egy négyszögletes katódból lép ki, és ennek a lemeznek (képmátrix) "ütközik". Ezen csak ott tud áthatolni, ahol a karakterek vonalai "hiányoznak". Az ilyen



177.b. ábra. Maszk segitségével történő karaktermegjelenités

módon "formált" sugár a képernyő belső felületére ütközve, a megfelelő karaktert megjeleniti. Az elektronsugarat a képcsőben két eltéritőrendszer vezérli. Az első a megfelelő karakter kiválasztását biztositja, a második a képernyő adott pontjára tériti el a "formált" sugarat. E módon már nemcsak karaktereket lehet megjeleniteni, hanem ábrát, vonalat is. A képmátrix egyik poziciójában egy furat van. Az ezen áthaladó fénysugár a képernyőn mint pont jelenik meg. Ha ezt a pontot a második eltéritő rendszerrel megfelelően mozgatják, akkor tetszőleges ábra, vonal hozható vele létre.

A monoszkóp tipusu generálás által keltett karakterképek nagyon konturosak, szépek. A módszer hátránya, hogy a karaktercsere csak a képcső cseréjével oldható meg.

2. Pontfelvillanásos tipusu generálás

Alapvetően két fajtájuk van:

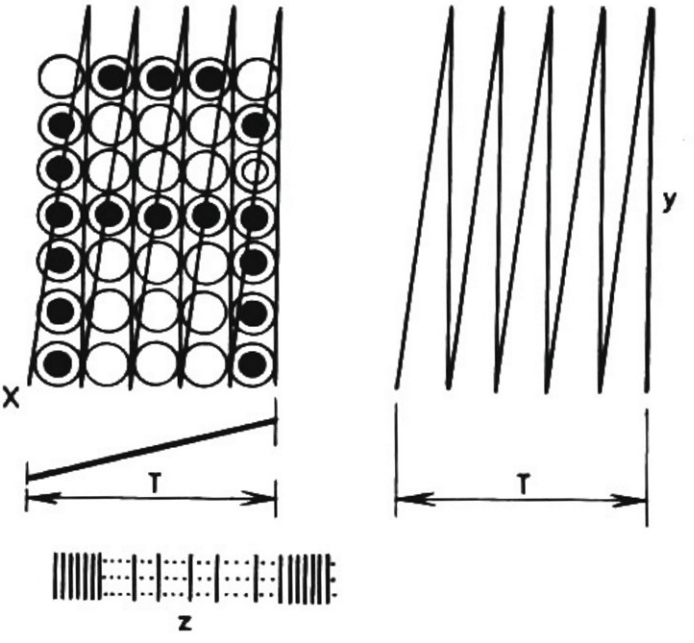
- - pontokkal rajzoló karaktergenerátorok
- mátrix rendszerü (fixpont elrendezésü) generátorok

Az első módszernél, a pontokkal rajzoló generátornál a képernyő minden egyes pontja megcimezhető. Az ábrák, karakterek elemi részekből, pontokból tevődnek össze. Minden egyes ilyen képpont helyzetét x, y koordinátákkal definiálni kell. Ezért az ábrát leiró program elkészitése nagyon hosszadalmas, fárasztó és sok tárterületet is igényel. Előnye viszont, hogy tetszőleges ábra, karakter jelenithető igy meg. Még a karakterek mérete is változhat.

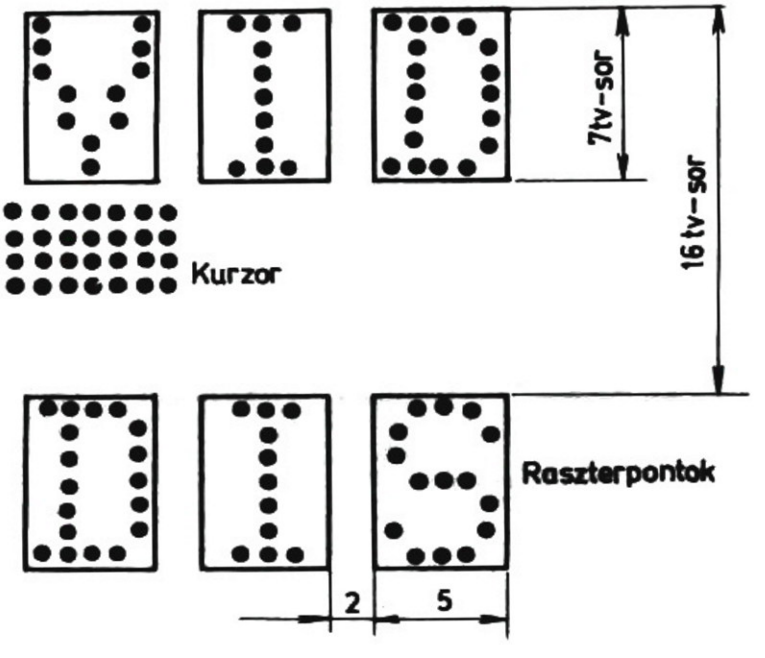
A mátrix rendszerü generálásnál viszont a karaktereket nem tetszőleges pontokból lehet összerakni, hanem azok egy mátrix elemei. Általában 5 x 7, vagy szebb, pontosabb "irás" esetén 19 x 19 pontból áll a mátrix (178. ábra). Azoknál a mátrixpontoknál, amelyek a karakter konturját alkotják, az elektronsugarat be kell gyujtani, ellenkező esetben viszont ki kell oltani, hogy ott ne rajzolódjon a képernyőre pont (a "pontfelvillanásos karaktergenerálás" elnevezés is ezt tükrözi). Az elektronsugár pályája bármely karakter generálásakor azonos, amig a sugár x irányban egy karakternyit elmozdul, y irányban ötször kell a karakter magasságának megfelelő utat megtenni. A karaktermemóriában az kerül tárolásra, hogy minden egyes karakternél mikor kell bekapcsolni, illetve kikapcsolni az elektronsugarat, vagyis az intenzitás szabályozása. Ez a módszer olcsósága, és a karakterek kielégitő megjelenitése miatt általánosan elterjedt.

Ennek a módszernek egyszerüsitett változata az un. tv-raszter módszer. Itt az elektronsugár nem külön-külön rajzol fel minden egyes karaktert, hanem soronként az összeset. A karakterek itt is pontmátrixban helyezkednek el (pl. 5 x 7-es), de az elektronsugár eltéritése a tv-hez hasonlóan törté-

nik, azaz a képernyő soronként kerül "letapogatásra". Tehát először a kiirt sorhoz tartozó karakterek felső 5 pontja lesz látható, aztán a második 5-5 pont, legvégül a hetedik sorhoz tartozó pontok. Így csak a 7 sor letapogatása után jelenik meg a képernyőn egy teljes sornyi (olvasható) információ. A VIDEOTON 340 alfanumerikus display-e is (sok más tipus mellett) ezt a módszert alkalmazza (179. ábra).



178. ábra. Mátrixrendszerü karaktergenerálás



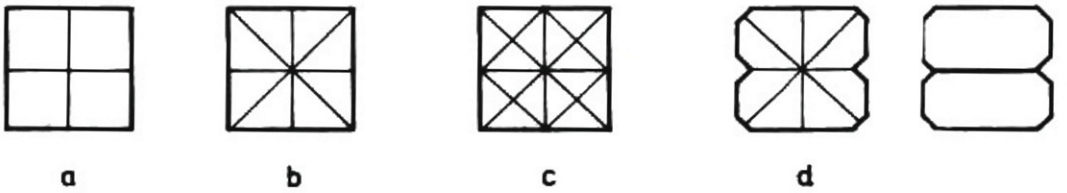
179. ábra. A karakterek és a kurzor kialakitása a VIDEOTON 340-es alfanumerikus display-nél

3. Vonalas tipusu karaktergenerálás

Ennél a tipusu generálásnál a karaktereket nem pontokból, hanem vonalakból teszik össze. E szerint két fajtájuk van:

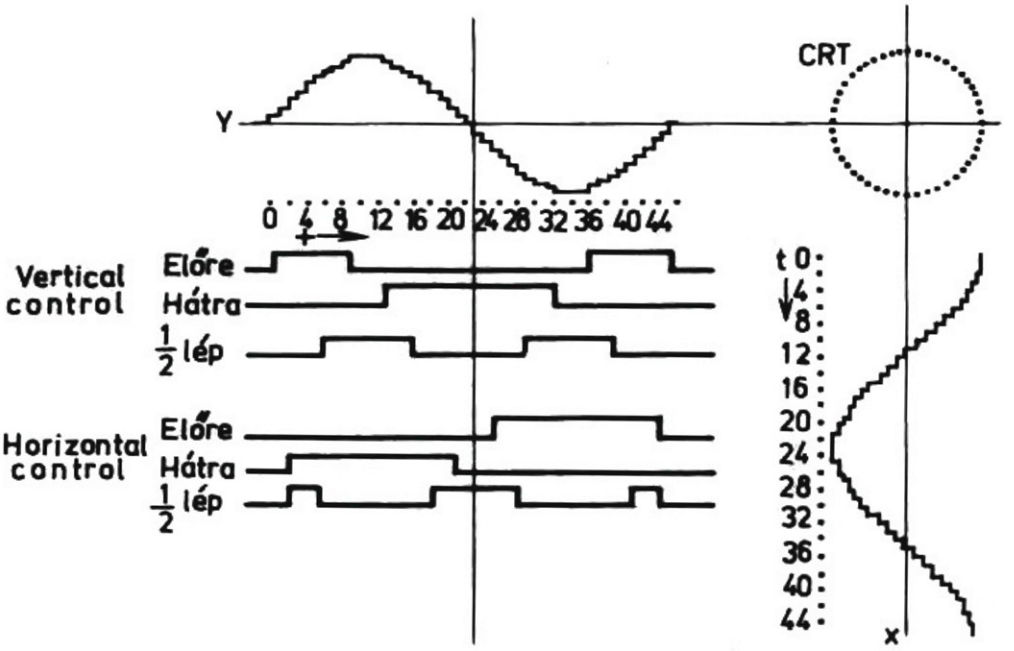
- univerzális minta szerinti ábrázolás
- karakterenként külön forma szerinti ábrázolás

Az univerzális minta szerinti rajzolásnál megvan szabva az elektronsugár lehetséges mozgáspályája, és a kivánt karakter csak az elektronsugár be- illetve kikapcsolásával tehető össze ezekből a lehetséges vonalakból (180. ábra). Igy viszonylag egyszerü, olcsó eljárás adódik, de a vonalak kötöttsége miatt a karakterforma nem szép (tul szögletes). A karakterek "szépsége" a lehetséges pályáktól függ. Az a) tipus csak számok, a b) már betük megjelenitését is lehetővé teszi. A c) d) pedig még tovább finomitja az ábrázolást.



180. ábra. Univerzális minta szerinti karaktergenerálás

A karakterenként külön forma szerinti ábrázolás már mentes ettől a "szépséghibától". Folytonos vonalak vagy pontokból összerakott vonalak ("programozható pont rendszer" - független pontok egymás utáni) sorozatával képezi a megfelelő karaktert. A vonaldarabok, pontok helyzete programozható (karaktermemóriában) igy tetszőleges vonalszakasz, ábra, vagy karakter megjelenithető ezek összességével (181. ábra).



181. ábra. Kör generálása

A számitógéptől érkező, megjelenitendő információk a display átmeneti, puffer tárába kerülnek. Ez a képernyőn egyidőben megjelenithető összes karaktert tárolja, kapacitása tehát a képernyő ábrázolási képességével vagy megegyezik, vagy annál nagyobb. Az előbbiekben röviden leirt módszerek valamelyikével aztán ennek a tárnak a tartalma alapján történik a megjelenités. Az egyes karakterek a képernyő azon pontjára kerülnek, ahol a bevitelnél is szerepeltek, vagy ami egy programban elő van irva.

A képernyőn megjelenithető információ mennyisége kötött. Általában 8-32 sorban, soronként 30-120 karakter jelenithető meg, a szokásos iráskép formájában. Ez természetesen függ a képernyő méretétől, a gyártmány tipusától, a karakterek nagyságától, a felhasználói igényektől, ártól stb.

A grafikus display-k vonal- és ábramegjelenitő képessége tulajdonképpen többféle módon is biztositható. Az előbbiekben ismertetettek közül a monoszkóp tipusu karaktergenerálás (egy pont képének megfelelő eltéritésével a képernyőn), a pontokkal rajzoló tipus (a képernyőn megjelenitendő ábra pontonkénti cimezésével), vagy a vonalas tipusu generálás (programozható pont rendszerrel) is mind alkalmas ilyen feladat ellátására. Ezek azonban meglehetősen hosszu, bonyolult előkészitést igényelnek, igy a "rajzolás" velük elég fáradságos.

Ezek helyett a grafikus megjelenités vonalakkal vagy vektorokkal történik. Mindkét esetben tulajdonképpen a képernyő valamelyik pontjából kiindulva (a kezdőpont koordinátái ismertek kell hogy legyenek) adott nagyságu és adott irányu egyes szakaszt hoznak létre, "rajzolnak". Csak a megadási mód más a két esetben.

A vonal, pontosabban szakasz két pontot összekötő egyenes. Igy tehát az ábra vonalainak, egyenes szakaszainak kezdő- és végpontját kell megadni, programozni.

A vektor pedig egy adott pontból (az előbb már emlitett kezdőpontból) adott irányban kiinduló, megadott nagyságu félegyenes. Az irány és a nagyság a különböző vektorkomponensekkel (Δx, Δy) adható meg. Bármelyik módszer felhasználható grafikus display-hez. Igy ezek legfontosabb része az un. vonal- vagy vektorgenerátor. (A kettő tulajdonképpen müködését tekintve alig különbözik egymástól, hiszen a vonalról vektorra való áttérés igen egyszerü.)

Ezek után nézzük meg, hogy milyen módon, milyen eszközökkel biztositják a display-k az információ bevitelét a rendszerbe, az inputot. Mivel az alábbi, ismertetésre kerülő módok a már bevitt, vagy még csak a képernyőn megjelenitett információ módositását is lehetővé teszik, biztositják a rendszer és a környezet közvetlen, interaktiv kapcsolatát. Ez a display-k növekvő elterjedésének fő oka.
Ezeknek a feladatoknak az elvégzéséhez azonban a display-ket különböző kiegészitő berendezésekkel is el kell látni. Ilyenek:

- alfanumerikus billentyüzet és funkcionális (vezérlő) billentyük
- fényceruza
- pozicionáló gömb vagy botkormány
- koordinátatábla.

1. Alfanumerikus billentyüzet és funkcionális billentyük

A billentyüzet felépitését és feladatát tekintve közel azonos az irógépeknél találhatókkal, a billentyük leütését, a beviendő információt elektromos jellé, a megfelelő karakterkóddá alakitja át (például 7 bites ASCII kóddá) - 182. ábra. Az irógépeknél a következő karakter helye a papiron közvetlenül adódik, ezt a kezelő látja. A display-nél kissé más a helyzet, más módon kell a karakterek helyét előre jelezni.

b_7	0	0	0	0	1	1	1	1
b_6	0	0	1	1	0	0	1	1
b_5	0	1	0	1	0	1	0	1

b_7	b_6	b_5	b_4	b_3	b_2	b_1		0	1	2	3	4	5	6	7
			0	0	0	0	0			SP	∅	@	P	Ю	П
			0	0	0	1	1			!	1	A	Q	A	Я
			0	0	1	0	2			”	2	B	R	Б	P
			0	0	1	1	3	ETX		#	3	C	S	Ц	C
			0	1	0	0	4			$	4	D	T	Д	T
			0	1	0	1	5			%	5	E	U	E	Y
			0	1	1	0	6			&	6	F	V	Ф	Ж
			0	1	1	1	7		PRINT	’	7	G	W	Г	B
			1	0	0	0	8	CURSOR LEFT	CURSOR RIGHT	(	8	H	X	Х	b
			1	0	0	1	9	HT	CURSOR UP	)	9	I	Y	И	bI
			1	0	1	0	10	LF	CURSOR DOWN	*	:	J	Z	Й	З
			1	0	1	1	11	INSERT LINE		+	;	K	[	К	Ш
			1	1	0	0	12	HOME	INSERT CHAR.	,	<	L	\	Л	Э
			1	1	0	1	13		DELETE CHAR.	–	=	M	]	M	Щ
			1	1	1	0	14		DELETE LINE	.	>	N	^	H	4
			1	1	1	1	15		ERASE	/	?	O	_	O	DEL

182. ábra. VT 340-es alfanumerikus display és kódrendszere

Erre szolgál az un. kurzor (cursor). Ez egy karakteraláhuzás, általában kis téglalap vagy csucs alaku jel, amely a sorban a következő karakter helyét határozza meg. A bebillentyüzött karakterek mindig e fölé kerülnek, a kurzor pedig eggyel továbblép és igy automatikusan jelzi a következő, bevitelre váró karakter helyét a képernyőn. A készülék bekapcsolásakor sok esetben a képernyő bal felső sarkában, az irás kezdőpontjában helyezkedik el és gyakran állandó villódzással is felhivja magára a figyelmet. A szövegtől függetlenül is mozgatható a funkcionális billentyük segitségével. Soronként feljebb, lejjebb, karakterenként balra vagy jobbra léptethető, igy a képernyő tetszőleges helyére elmozditható, és ott különféle javitás, irás, beszurás végezhető el segitségével. Természetesen eközben nemcsak a képernyőn megjelenő, hanem a puffertárban is megtalálható szövegen a változtatás végrehajtódik. A funkcionális billentyük segitségével a központi számitógéptől függetlenül különféle szövegszerkesztési eljárások is lehetővé válnak (mind a képernyő, mind a puffertár tartalmán). Például a VT 340 display-nél:

- a képernyő tartalmának teljes törlése
- a kurzor kezdőpontba állitása
- a kurzor következő sor elejére állitása
- a kurzor mozgatása a képernyőn négy irányban
- tabulálás
- karakter-felülirás
- karakterbeszurás
- karakterkiejtés (törlés)
- sorbeszurás
- sorkiejtés (sor törlés)
- aláhuzás
- memóriavédelem
- ROLL (Bekapcsolt állapotban, ha a képernyő tartalma megtelt, a teljes szöveg egy sort emelkedik, a legfelső sor elvész, és egy uj üres sor keletkezik - legalul - az irás számára. Kikapcsolt állapotban, ha a képernyő megtelt, a kurzor a jobb alsó sarokból a bal felsőbe ugrik, és a képernyő előző tartalma felüliródik.)

A képernyő tartalma (a puffertáré persze) csak akkor kerül a csatornarendszeren átvitelre, ha erre külön utasitás van (megfelelő funkciógillentyüvel). Igy általában lehetőség van karakterenként, soronként, vagy az egész képernyő tartalmának egyben történő átvitelére. Általában ez utóbbi a használatos, mert a rendszer erőforrásait a leggazdaságosabban használja ki, és csak a helyesen megszerkesztett képernyőtartalmat továbbitja az operativ tár felé. A grafikus display-k általában rendelkeznek alfanumerikus billentyüzettel. A hangsuly azonban az információ grafikus megadásán, módositási lehetőségén van. Ezért tehát rá kell tudni mutatni kivülről a képernyő egy pontjára (vagy pontjaira), meg kell tudni azt (azokat) adni, hogy aztán ott valamilyen müveletet el lehessen végezni, amit aztán már például funkcionális billentyükkel is meg lehet csinálni. Az alfanumerikus display-knél a hasonló célra alkalmazott megoldás (és jel) a kurzor itt nem használható, mivel csak diszkrét helyeket tud kijelölni. (Megszabott helyzetü sorokat, és a sorokon belül adott karakterhelyeket.) A grafikus display-knél a képernyő valamely pontjának megadására vagy a fényceruza, vagy a pozicionáló gömb (botkormány), vagy az x, y koordinátatábla szolgál.

2. Fényceruza

Ez az egyik legelterjedtebb hely-(képpont-) azonositó eszköz. A fényceruza egy fotodiódából áll, mely a képernyő fényceruza alatti pontjában az elektronsugarat érzékelni tudja, amikor az éppen ott halad el. A display egy áramköre az elektronsugár pillanatnyi eltéritésének függvényében ezt a fényceruza által szolgáltatott jelet aztán érzékelni és megfelelő módon tárolni tudja. Igy tehát a képernyőnek az a pontja (x, y koordinátáival), ahová a fényceruzával "rámutattunk", tárolásra kerül. Az igy kijelölt pont aztán jelölheti például egy uj vonalnak (szakasznak) a végpontját, lehet vele szövegrészt azonositani, lehet a megjelölt információt törölni, változtatni stb. A különféle felhasználásra speciális programok szolgálnak. Tulajdonképpen még "rajzolni" is lehet vele a képernyőre. Ilyenkor mindazoknak a pontoknak a koordinátái tárolásra kerülnek, amelyek felett a fényceruza elhaladt (ahol tehát az elektronsugarat a dióda érzékelte.) Az igy tárolásra került pontok alapján az elektronsugár a képernyő ismételt végigpásztázása közben már a megrajzolt vonalat jeleniti meg. (Az elektronsugár olyan gyorsan futkos végig a képernyőn, például másodpercenként 50-szer, hogy a két futás közti időt nem lehet észrevenni.) Az ilyen módon a képernyőre rajzolt információ, mint ahogy már mondtuk, a tárban is megjelenik, és ez alapján az ábra, a vonaldarab mint uj, feldolgozandó információ a számitógép felé is továbbitható. Az ilyen "rajzolási" lehetőséget különösen az interaktiv tervezésben lehet jól felhasználni. Ujabban már az alfanumerikus (karakteres) display-knél is alkalmaznak fényceruzát, ott persze csak a szövegrész azonositására (gyorsabb azonositást, kikeresést tesz lehetővé, mint a kurzor).

3. Pozicionálógömb

A pozicionálógömb vagy egyes esetekben botkormány a kurzor mozgatására, pozicionálására szolgál. Igy folyamatosan lehet azt akár vizszintesen, akár függőlegesen elmozditani, és a képernyő tetszőleges pontjára ráállitani. A gömb illetve a kar mozgatása két potenciométer tolóérintkezőjét állitja be valamilyen helyzetbe. A potenciométerekre kapcsolt feszültség igy a tolóérintkező helyzetétől függően "leoszthatő".

4. Koordinátatábla

Az x, y koordinátatábla szintén a képernyő valamely pontjának azonositására szolgál. A berendezés a tábla egy pontjára helyezett "toll" koordinátáit érzékeli és ez alapján jelöli ki a képernyő megfelelő pontját. Többféle módon is kialakitható. A legáltalánosabbak:

- az ellenállásréteggel bevont felület
- szendvics szerkezetü tábla

Az ellenállásréteggel bevont tábla esetén annak szélein vezetőréteg van kialakitva. A toll egy feszültségforrásra kapcsolódik. Ha ezt a tollat a táblához érintjük, akkor az x és y irányban az ellenállásréteg miatt más-más feszültséget lehet a táblák szélén érzékelni (kivéve, ha a tábla középpontjára mutatunk, mert akkor mind a 4 vezetőréteg ugyanakkora feszültséget jelez.)

A szendvics szerkezetü táblánál két vezető fémlap közé van egy un. feszültségháló helyezve. Ennek a hálónak a vezetékei különböző nagyságu feszültségre vannak kapcsolva. (A feszültség értéke függ a vezetőhálón belüli helytől). Ha a tollat a táblához érintjük, az összeköttetést hoz létre a háló és a két vezetőlap között. A vezetőlapokon megjelenő feszültség az érintési ponttól függően más és más.

5. Az operációs rendszer

5.1 AZ OPERÁCIÓS RENDSZER KIALAKULÁSA

5.1.1 A GÉPI KÓDÚ PROGRAMOZÁSRÓL

A számitástechnika szakemberei valamennyien jól ismerik a mottót: mi az elefánt? Egy egér operációs rendszerrel. Az analógia tréfásan hangzik, de fontos igazságot rejt magában; a számitógép hardware- és software erőforrásainak arányára utal. Mi lehet ez az operációs rendszer, amely a megismert számitógéprendszert oly hatalmassá növeli, alkalmazási lehetőségeit ugy megsokszorozza? Erre a kérdésre keresünk választ most, amikor betekintünk az operációs rendszer működésébe.

A számitástechnika és számitógépgyártás hőskorában, az 50-es évek táján operációs rendszerről még nemigen beszéltek az emberek. Az akkori számitógépek tulnyomórészt elektroncsöves felépitésűek voltak, az I. generációhoz tartoztak. Egy-egy gép olyan hatalmas méretű volt, hogy tágas termet foglalt el a számitóközpontokban. A gép központi része, a processzor és az operativ tár nem vált el élesen a használt perifériakészlettől: a központi számitógép és a hozzátartozó input/output eszközök egységes egészet alkottak. Szó sem volt csatornáról, csatorna-interface-ről, vagy a perifériakészlet rugalmas változtatásáról. Ezeket a gépeket ma már alighanem csak a muzeumokban láthatjuk, hiszen a fejlődés oly hatalmasat lépett előre mintegy negyedszázad alatt, hogy az I. generációs gépek teljesen kiszorultak a gyakorlati alkalmazásból.

Eltekintve bizonyos sajátosságoktól, e gépek üzemeltetése, programozása is a megismerthez hasonló gépi kódban történt. Minden gépnek megvolt a maga gépi nyelve, tehát az az utasitáskészlet, amelyet a processzor (a vezérlő egység) értelmezni tudott és képes volt végrehajtani. A felhasználó dolga itt az volt, hogy a gépi nyelven működőképes programot irjon, azt bejuttassa az operativ tárba, és átadja ennek a programnak a vezérlést. Hangsulyozzuk, hogy mindezt a mai számitógépekkel is meg lehet tenni; például tetszőleges programokat vihetünk be és futtathatunk a központi gép kezelőpultjáról.

A programozó tehát saját maga irta meg a programot, és ő felügyelt a program futására, vagyis végig a gépteremben tartózkodott. Ő volt az egyedüli, aki ismerte a programját, és tudta azt is, hogy adott esetben (például programhiba észlelésekor) milyen intézkedésre lehet szükség. Ebből is érezhetjük, hogy a korai számitógépek kihasználása, működésük hatékonysága nem lehetett tul nagy! Ezen tulmenően, a programokat tisztán bináris formában kellett megirni, ez pedig tudvalevően igen fáradságos dolog, a hibakeresésről nem is beszélve. Az a programozó, aki egy gép nyelvét már elsajátitotta, ha más gépen akart dolgozni, annak a gépi nyelvét is meg kellett tanulnia. Ugyanez vonatkozik a programokra is: egy adott gépre irt program csakis azon a gépen volt futtatható. A számitóközpontok közötti kommunikáció tehát fejletlen volt.

Mindemellett nem szabad lebecsülnünk a korai számitógépeket: a konstruktőrök és programozók ezeken tanulták meg a számitástechnikát, másküllönben ez a technika sohasem tudott volna eljutni mai fejlettségi fokára.

A gépi kódu programozásnak tehát előnye volt az, hogy a gép "minden bitjét" el lehetett érni, amely a programozó számára hozzáférhető volt. Hátrányos tulajdonsága viszont, hogy még kisebb programok megirása/tesztelése is rendkivül nagy programozói munkaráforditást igényelt, s ez akadályozta a fejlődést, a számitógépek bevetését bonyolultabb feladatok megoldásába. Ki kellett tehát valamit találni, ami a gépi nyelvet közelebb hozza az ember gondolkodásmódjához, és egyszersmind lehetőséget teremt az algoritmusok áttekinthetőbb, kényelmesebb kódolására. Ezt a problémát a szimbolikus programozás oldotta meg.

5.1.2 A SZIMBOLIKUS PROGRAMOZÁSRÓL

Szimbolikus programozáson azt értjük, hogy a gépi programot nem gépi kódban, hanem szimbolikus, elsősorban szöveges program-elemek használatával irjuk le. Ebből következik, hogy a szimbolikus programot a gép közvetlenül "nem érti", a szimbolikus programból valamilyen módon gépi kódot kell csinálnunk.

Világos, hogy milyen előnyökkel jár a szimbolikus programozás bevezetése: a program kódolása jóval gyorsabb, a program áttekinthetőbb, bináris számok helyett a müveleti kódokat és tárcimeket szövegesen adhatjuk meg. Ezzel együtt azonban áldozatot is kell hoznia a gyártó cégnek: meg kell teremtenie azokat az eszközöket, amelyek segitségével a szimbolikus kódból gépi kódot készithetünk. Mindezt azonban csak egyszer kell megtennie, ezután már minden program szimbolikusan kódolható.

Lássunk egy példát a szimbolikus és gépi kód kapcsolatára! Tekintsünk egy karakterlánc-átvivő utasitást, legyen ez SS1 tipusu. Ha az OP2 operanduscimtől kezdődően 8 byte-ot át akarunk vinni az OP1 operanduscimtől kezdődő tárterületre, akkor ezt szövegesen az alábbi módon irhatjuk le:

```
MVC   OP1(8), OP2
```

Az utasitás három részből áll: müveleti kód, első operandus cime, második operandus cime. Az MVC rövidités angol eredetü, a Move Characters (=Karakterek átvitele) szöveg helyett áll. Az utasitásnak ez a része tehát azt mondja meg, milyen müveletet kell végezni. Igy ez megfelel a müveleti kódnak, természetesen szimbolikus formában. Mivel a szimbolikus müveleti kódok kialakitásánál arra törekszünk, hogy a rövidités utaljon a müveletre, annak jellegére, egyszóval könnyen megjegyezhető legyen, az ilyen müveleti kódokat mnemotechnikus müveleti kódnak nevezzük. A szimbolikus programozás során tehát a müveleteket mnemotechnikus kódokkal adjuk meg.

Az OP1 program-elem az első operandus szimbolikus cime a tárban. Ugy képzelhetjük el a szimbolikus cimzést, mintha a tár bizonyos rekeszeit - amelyekre utasitásokban hivatkozunk - elláttuk volna sajátos nevekkel. Természetesen két különböző tárrekeszt nem lehet ugyanazzal a névvel illetni! Most már igen kényelmes dolog lesz ennek a rekesznek a tartalmára hivatkozni: a programban ugyanis bárhol leirjuk az OP1 nevet, ez mindig az ilyen "nevü" rekeszt fogja jelenteni. A rekeszek elnevezésének csak addig van értelme, amig a szimbolikus kódról át nem térünk gépi kódra. A gép ugyanis

már nem ismeri a szimbolikus cimet! OP1-hez teljesen hasonlóan értelmezhető OP2 szerepe, de itt az utasitás második operandusát irtuk fel szimbolikus alakban.

OP1 mellett megadtuk még azt is, hogy hány byte szerepeljen a müveletben, esetünkben ez 8. Az operandusok tehát az alábbiak:

1. operandus - az a 8 byte, amelyek közül az elsőnek a cime OP1,
2. operandus - az az ugyancsak 8 byte, amelyek közül az elsőnek a cime OP2.

Mivel tárterületek tartalmának mozgatásáról van szó, az operandushosszt elegendő egyszer megadni! Ha most elgondoljuk, hogy ennek az utasitásnak gépi megfelelője egy 48 bites bináris formátumu utasitás, akkor a szimbolikus irásmód előnyei, könnyebbségei nem igényelnek további bizonyitást.

A példában egy olyan szimbolikus irásmódot mutattunk be, amely nagyon közel áll a gépi nyelvhez: a szimbolikus utasitás felépitése pontosan követte a gépi utasitás felépitését. Vannak olyan szimbolikus kódolási rendszerek is, amelyek a gépi nyelvtől jobban eltávolodó programirást is lehetővé teszik. Ilyen irásmódot használtunk az alábbi utasitásban:

```
ALFA = 8.47*(SIN(ALFA - BETA) + GAMMA - 6.Ø)
```

Itt egy összetett aritmetikai kifejezést süritettünk bele egyetlen szimbolikus utasitásba. Az elnevezett tárcimek itt ALFA, BETA és GAMMA. Ezek az utasitás végrehajtásakor egy-egy számot tartalmaznak. Az utasitás hatására az ALFA nevü tárrekesz tartalma megváltozik, éspedig felveszi azt az értéket, amely az egyenlőség jobb oldalán álló kifejezés kiértékelése során adódik. Azt mondjuk, hogy ALFA, BETA és GAMMA cimek tartalmának aktuális értékéből kiszámitjuk ALFA tartalmának uj, következő értékét.

Ilyen és ehhez hasonló, zárójelekkel és müveleti jelekkel ellátott utasitások tetszőleges bonyolultsági fokban irhatók; az ilyen utasitások gépi megfelelője azonban nem egyetlen utasitás lesz, hanem egy olyan utasitássorozat, amely hatásában egyenértékü az itt felirt egyetlen utasitással. Ez a lehetőség is a szimbolikus programozás létjogosultságát támasztja alá.

Napjainkban csaknem mindenütt kizárólag szimbolikusan programozunk, és a gépi kódhoz csak akkor térünk vissza, ha ezt valamilyen feltétel különösen indokolttá teszi (például a hardware tesztelése, bizonyos funkciók kipróbálása stb.).

A szimbolikusan megirt programot azonban át kell alakitani gépi kódu programmá, hogy alkalmassá váljon a hardware vezérlésére.

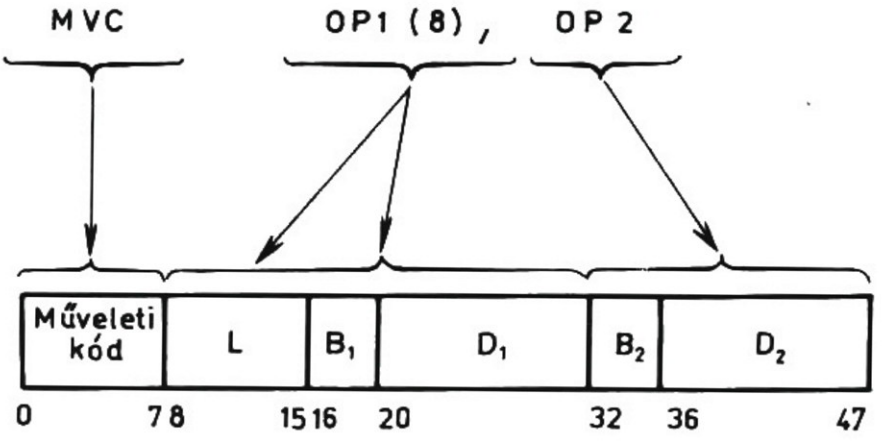
5.1.3 A PROGRAMFORDÍTÁS FOGALMA ÉS SZÜKSÉGESSÉGE

Programforditáson azt értjük, amikor a szimbolikusan megirt program alapján létrehozzuk annak gépi kódu megfelelőjét. A programot szimbolikus nyelvről leforditjuk a gép nyelvére. Ezt a müveletet szintén egy program, a forditóprogram végzi el. Ennek megirása volt az a bizonyos áldozat, amelyet a gyártó cégnek kellett hoznia a szimbolikus programozás megteremtésekor.

Idézzük most fel a nemrég bemutatott szimbolikus karaktermozgató utasitást:

```
MVC   OP1(8),OP2
```

Ha ebből gépi kódu SS1 tipusu utasitást akarunk csinálni, vagyis ezt le akarjuk forditani gépi nyelvüre, akkor a szimbolikus program-elemekből gépi-utasitás elemeket kell kialakitanunk. A forditás alapelvét a 183. ábrán láthatjuk.



183. ábra. A programforditás alapgondolata

A forditóprogram feladata tehát az, hogy a müveleti kódot, operandushosszt, az 1. és 2. operandus cimét előállitsa. A müveleti kódot a mnemotechnikus müveleti kódból nyeri ugy, hogy minden egyes müveleti kódnak egy-egy mnemotechnikus kód felel meg. Ha most a forditóprogramnak a szimbolikus programban megadjuk a mnemotechnikus kódot, ő könnyüszerrel megkeresi a hozzá tartozó gépi müveleti kódot, és azt "beforditja" az utasitásba. Az operandushosszt is megadtuk: ez 8 byte. Itt mindössze annyit kell tennie, hogy bináris alakra konvertálja, és a kapott értéket elhelyezi az utasitásban. (Mivel az L mező a tényleges hossznál eggyel kisebb számot tárol, még levon 1-et a tényleges hosszból. Ez azonban csak a gépi nyelv sajátosságából következik, a forditás elvét nem befolyásolja.) Az első és második operandus szimbolikus cimének meg kell keresnie ezen operandusok tényleges cimét, vagyis azt a tárcimet, amelyet OP1, illetve OP2 névvel illettünk. Ezt a cimet azután bináris alakra hozva szintén "beforditja". Az megint a gépi nyelv sajátossága, hogy a cimeket a bázisregiszter relativ cim alakjában tárolja, ettől azonban a cim még cim marad!

Egyszerü példát mutattunk a programforditásra. Világos, hogy az összetettebb szimbolikus utasitásokkal egyenértékü gépi-utasitás sorozat megkeresése már nem ilyen egyszerü feladat, de lényegében ott is ehhez hasonló forditási eljárást kell végrehajtani.

Van tehát forditóprogramunk, amely "érti" a szimbolikus nyelvet, és szorgalmasan készíti a gépi programokat. Ezzel a számitógép sokkal hatékonyabb segitőtársa lett az embernek: a nehézkes, fárasztó rutinfeladatokat, a bináris kód létrehozását forditóprogramja révén magára vállalta! Ezután már kezdtek a gépeken bonyolultabb problémákat is megoldani. A szimbolikus programozás lehetővé tette, hogy a programozó az erőforrásait az algoritmusra forditsa, ne a kódolással bajlódjon. A gép lassan kezdett univerzálisan felhasználható eszközzé válni. Ennek most már csak egyetlen akadálya volt: a gépi nyelvhez tulságosan közel álló szimbolikus nyelv kötöttsége. A feladatok differenciálódása azonban ezt is forradalmasitotta.

5.1.4 ASSEMBLY SZINTŰ ÉS A MAGASSZINTŰ PROGRAMOZÁSI NYELVEK

Tulajdonképpen géporientált nyelv az ASSEMBLER, bár egyes szolgáltatásai a magasszintü nyelvekhez hasonlóak. Ez a legrugalmasabban használható nyelv. Elemei olyan szimbolikus utasitások, amelyek formátuma analóg a gépi utasitásokéval, igy egy ASSEMBLER-utasitásból egy gépi utasitás keletkezik a fordités (assemblálás) során. Emellett a nyelv lehetővé teszi az ugynevezett makróutasitások és makródefiniciók használatát is. Ezek olyan utasitások, illetve utasitássorozatok, amelyek ASSEMBLER nyelvi utasitásokból állnak, és egy-egy sajátos funkcióra szolgálnak. A makródefiniciók utasitássorozataiból a makróutasitás paraméterezésével kiválogatjuk azokat az ASSEMBLER nyelvi utasitásokat, amelyekre szükségünk van, és ezeket bemásoljuk az ASSEMBLER nyelvü szimbolikus programba. Igy tehát egy utasitás, a makróutasitás helyére általában több, esetleg igen sok ASSEMBLER utasitás kerül.

A számitógép hatékonyságának további növeléséhez arra volt szükség, hogy a forditóprogramokat jobban a feladatokhoz igazitsuk, a szimbolikus programozást pedig jobban feladatorientálttá tegyük. Ez azt jelenti, hogy például a bonyolult strukturáju input/output-ot igénylő statisztikai, adatfeldolgozási feladatokat olyan szimbolikus kódban lehessen irni, amely egyszerüsiti a struktura leirását, vagyis a forditóprogram többet vállal magára, nagyobb lehet a távolság a felhasználó és a gép között. Ugyanakkor a számitásigényes tudományos-müszaki feladatok programjait "matematikaibb" alakban, képletek és függvények felhasználásával lehessen kódolni. Ez a feladatorientáltsági igény hozta létre az ugynevezett magasszintü programozási nyelveket, illetve ezek forditóprogramjait. Mig a géporientált, géphez közeli szimbolikus kód forditását assemblálásnak (összeállitásnak) nevezzük, az ilyen forditóprogramokat pedig assemblereknek, addig a magasszintü programozási nyelvek forditását kompilálásnak, forditóprogramjaikat kompilátoroknak nevezzük. Kompilálás során egyetlen "süritett" információtartalmu szimbolikus utasitásból több gépi utasitás keletkezik, hiszen a szimbolikussal egyenértékü gépi program is jóval bonyolultabb lehet.

Alább röviden vázoljuk a legelterjedtebb magasszintü programozási nyelvek jellegzetességeit.

(1) FORTRAN (FORmula TRANslation = Képletforditás)

Különösen tudományos-müszaki feladatok megoldására használható. Több változata létezik, az elsőt 1954-ben fejlesztették ki. Jelenleg a FORTRAN IV változat az egyik legkorszerübb. A FORTRAN forditóprogram a FORTRAN nyelven irt forrásprogramokat elemzi nyelvhelyesség szempontjából, és ezekből gépi kódu programot készit. Egy sor könnyitést vezet be a programozó számára, ezek közül néhányat felsorolunk:

- Konstansok, függvények, tárterületek kényelmes használata,
- Egy- és többdimenziós adattömbök kezelése,
- Kényelmes input/output kezelés,
- Alprogramok kényelmes használata, programszegmentálás, eljárásfüggvények használata,
- Több szegmens által használható közös adatterület fenntartása,
- Összekapcsolhatóság egyéb nyelven irt programokkal,
- Előre megirt függvények (például sinus, cosinus stb.).

Mindezek a szolgáltatások lehetővé teszik, hogy a FORTRAN nyelven programozók bonyolult, számitásigényes programokat viszonylag könnyen megirjanak.

(2) PL/I (Programming Language I = I. programozási nyelv)

Olyan univerzális magasszintű nyelv, amelyben mind a tudományos-műszaki, mind az adatfeldolgozó/gazdasági feladatok kényelmesen kezelhetők. Előnyösen alkalmazható például olyan esetekben, amikor nagymennyiségű adaton számitásigényes feldolgozást kell végezni. A PL/I-et azzal a céllal fejlesztették ki, hogy az összes addigi magasszintű és géporientált programozási nyelv előnyeit egyesitse magában. A PL/I kedvező tulajdonságai az alábbiak: egyszerű kifejezések, áttekinthető programok, tetszőleges matematikai kifejezések használhatósága, adattömegek kényelmes kezelése stb. A PL/I-nek van egy "kis" változata is (Subset), ezt használva a programozónak nem kell ismernie a teljes nyelvet.

(3) COBOL (COmmon Business Oriented Language = Kereskedelemorintált nyelv)

Ez jellegzetesen adatfeldolgozásra készült. Kényelmessé teszi az input/output műveletek irását, különböző formátumok kezelését. Ugyanakkor nem támogatja a bonyolult, számitásigényes feladatokat, hiszen ezek az adatfeldolgozásban nem gyakoriak. Megjegyezzük, hogy a számitógéppark döntő többsége üzemszerűen adatfeldolgozást végez, igy a COBOL nyelv rendkivül elterjedt.

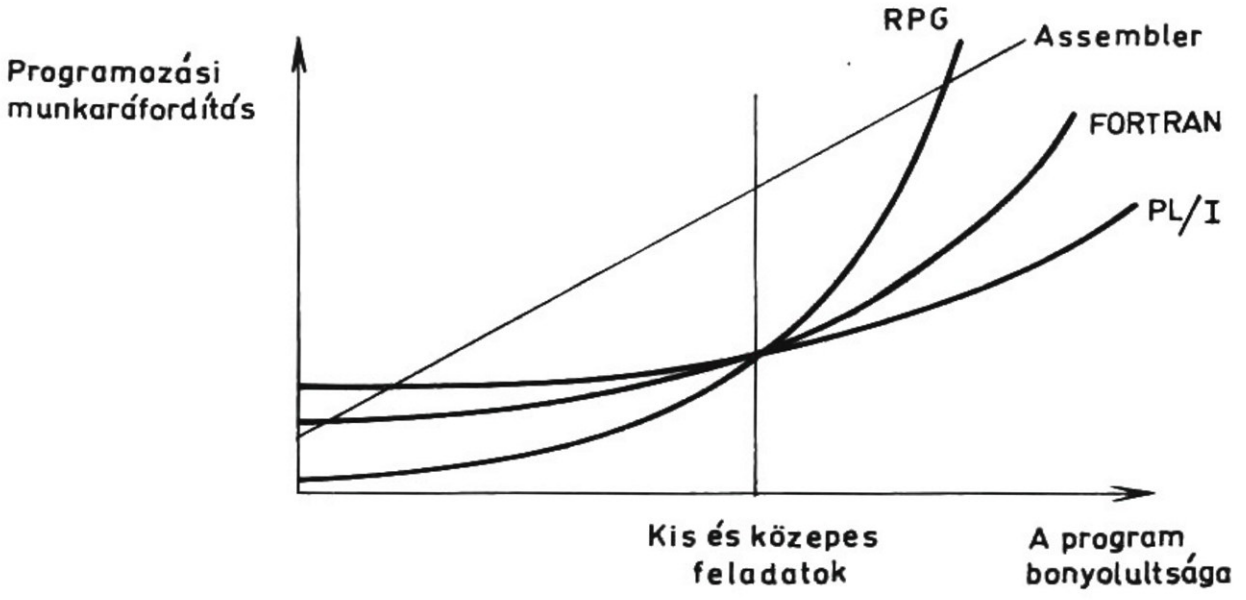
(4) RPG (Report Program Generator = Jelentéskészitő)

Olyan speciális nyelv, amely az input/output igényes, de nem számitásigényes feladatokat támogatja. Jellegzetességei:

- Egyszerű input/outputkezelés,
- Listakészités adott rekordokból,
- Egyszerű aritmetikai műveletek programozhatósága,
- Listák és táblázatok nyomtatási képének szabályozása,
- Adatállományok felépitése,
- Táblázatok végigvizsgálása stb.

Ezek mind olyan szolgáltatások, amelyek könnyedén, kevés munkával programozhatók. Ennek oka az, hogy a kompilátor sokat vállal magára e feladatokból.

A legelterjedtebb programozási nyelvek minőségi összehasonlitását a 184. ábrán láthatjuk.



184. ábra. Programozási nyelvek összehasonlitása

Az ábrán a megirandó program bonyolultságának függvényében láthatjuk a programozási munkaráforditást. Látható, hogy ASSEMBLER-ben mindig "többet kell dolgoznunk", mint a magasszintü nyelveken, de sokszor szükség lehet az alacsonyabb szintü programozásra.

A magasszintü programozási nyelvek megjelenésével a felhasználó már elszakad a számitógéptől. E nyelvek nyelvtani szabályait, az utasitások irásmódját ugyanis nemzetközi szabványositási tárgyalásokon rögzitették, tehát bárki - ismerve a nyelvet - müködőképes programot irhat. Ezt a programot bármely számitógépen lefuttathatja, feltéve, hogy a gépnek van megfelelő forditóprogramja. Az ilyen egységesitett, nemzetközileg definiált programozási nyelveket hivatkozási nyelveknek nevezzük.

5.1.5 AZ AUTOMATIZÁLT PROGRAMFUTTATÁS MEGJELENÉSE. MONITOROK, EXECUTIV PROGRAMOK, FELÜGYELŐ PROGRAMOK

A magasszintü programozási nyelvek és forditóprogramjaik bevezetésével a számitógépen megoldható feladatok köre rendkivüli módon kiszélesedett. A gyártók egyre inkább arra törekedtek, hogy a gép üzemeltetését automatizálják, vagyis csökkentsék az emberi beavatkozás gyakoriságát, és optimalizálják a számitógéprendszer elemeinek leterhelését, vagyis lehetőleg minél többet "foglalkoztassák" a gépet. Igy hamarabb megtérül a gépvásárlási beruházás is, és a drága berendezés nem áll kihasználatlanul. Ezek a törekvések vezettek oda, hogy a számitógépnek legyen egy kitüntetett szerepü, vezérlő programja, amely a többi program (forditóprogramok stb.) futására felügyel, s egyszersmind a számitógép erőforrásait közel optimálisan szétosztja a futó programok között. Ezt a felügyelő/vezérlő programot a különböző géptipusokon másként nevezték; igy szokás beszélni monitor, executiv programokról. Valamennyien hasonló funkciójuak: feladatuk az automatizált vezérlés megvalósitása. Az Egységes Számitógéprendszerben (ESZR) ennek a vezérlő programnak már több komponense van; például a rendszer épségét veszélyeztető funkciók mind a Supervisor (Felügyelőprogram) hatáskörébe tartoznak, az áttérést egyik programról a másikra a Job Control (Munkavezérlő) program biztositja stb. Ha most az eddig megismert programokat a vezérlő program felügyelete alatt futónak, egymással kapcsolatban állónak, együttmüködőnek képzeljük el, akkor tulajdonképpen már operációs rendszerről beszélünk. Tehát az operációs rendszer olyan programrendszer, amely a számitógép automatizált üzemeltetését biztositja lehetőleg ugy, hogy a gép kihasználása maximális legyen. Természetesen a mai operációs rendszerek, főleg a III. generációs számitógépeken, nemcsak forditó- és vezérlő programokból állnak, hanem a fejlődés során sok egyéb funkciót is egyesitettek. A korszerü operációs rendszer felépitését, funkcióit később mutatjuk be.

5.1.6 A MULTIPROGRAMOZÁS GYAKORLATA

Sokat beszéltünk arról, milyen lényeges a számitógéprendszer optimális vagy legalábbis közel optimális kihasználása. Mivel van már felügyelő/vezérlő programunk, jogos az igény, hogy mindent tegyen meg annak érdekében, hogy a rendszer minél kevesebbet álljon. Ez azonban csak akkor valósitható meg, ha a központi számitógép (tulajdonképpen a processzor) szünet nélkül dolgozik, és emellett a perifériakészlet viszonylag nagy része is müködésben van. Ha most elgondoljuk, hogy egy futó prog-

ram input/output müveletbe kezdett, és egy adatot vár a perifériájáról, akkor világos, hogy az adat beérkezéséig a program tovább nem futtatható, hiszen "nincs mit feldolgoznia!" A perifériamüvelet alatt tehát értékes processzoridő megy veszendőbe: a központi gép esetleg több ezer utasitást is végre tudna eközben hajtani!

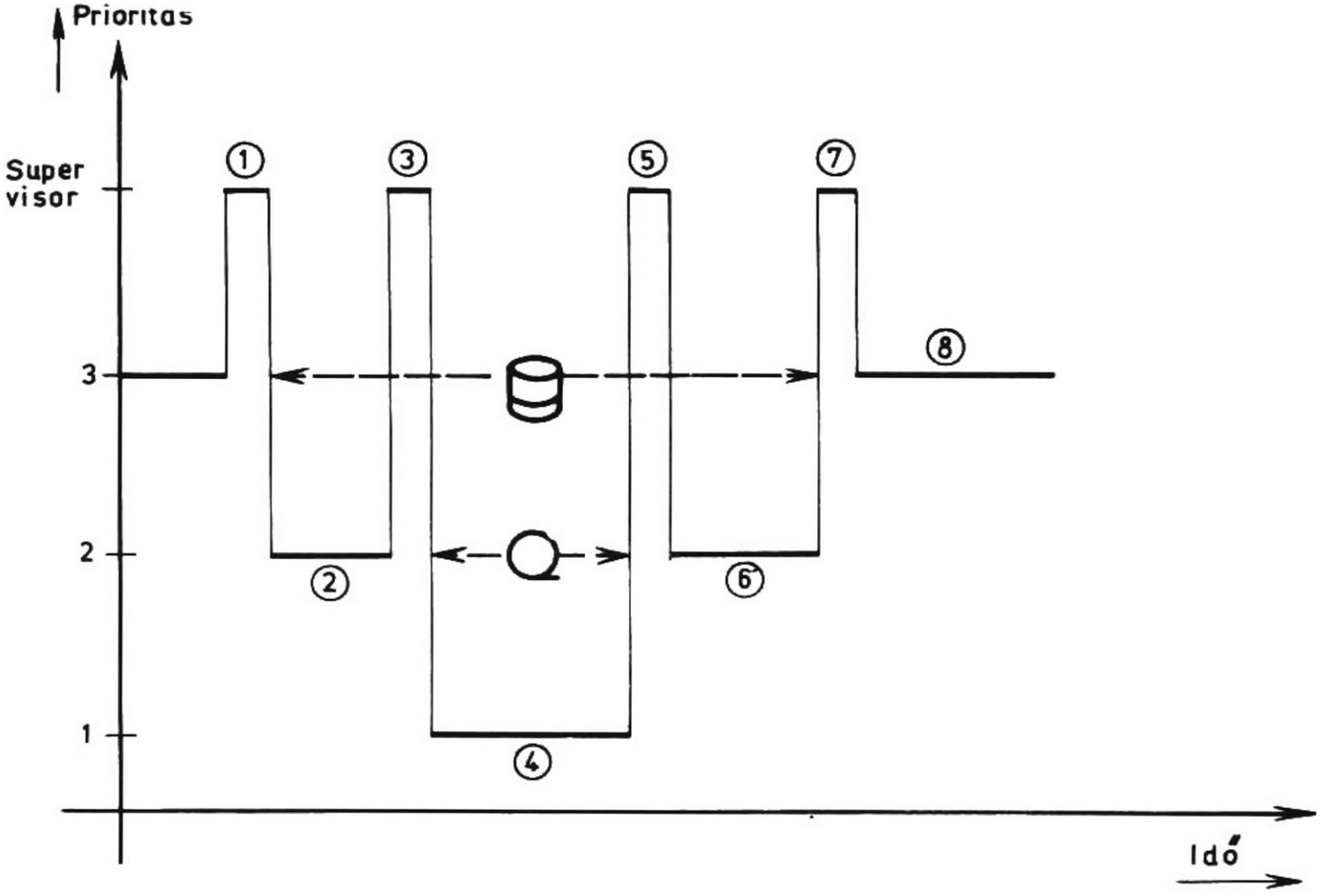
Magától kinálkozik az a lehetőség, hogy az input/output müvelet idejére vegyük el a vezérlést az álló programtól, és egy másik programnak bocsássuk rendelkezésére a processzort. Igy közel állandóan üzemelni fog a processzor, csupán a programok között szét kell osztani üzemidejét. Ez nem tul nehéz feladat, és a korszerü vezérlőprogramok valamennyien elvégzik! A most leirt üzemmódot nevezzük multiprogramozott üzemmódnak. A multiprogramozott rendszer vezérlő programja természetesen bonyolultabb, hiszen a multiprogramozás szervezését is el kell látnia.

Multiprogramozott üzemmódban tehát az operativ tárban nem egy, hanem több program tartózkodik. Ezek közül mindig csak egy kapja meg a központi gép erőforrásait, más szavakkal: egy időben csak egy program utasitásainak végrehajtása folyik. Abban a pillanatban azonban, amikor ez a program nélkülözni tudja vagy nélkülözni kényszerül a központi gépet, azonnal átadjuk a vezérlést egy másik, várakozó programnak. Igy minimális idő megy veszendőbe!

A multiprogramozás szervezésére több módszer terjedt el. Van olyan, ahol a vezérlő program azonos időszeleteket "juttat" minden programnak, tehát periodikusan elveszi a vezérlést az egyes programoktól. Az ilyen rendszert időosztásosnak nevezik, arra utalva, hogy a processzor-időt szétosztja a programok között. Ennek van olyan változata is, amelyben az egyes programok nem egyenlő hosszu időszeletet kapnak futásra; bizonyos programok (a sürgősek) többet futnak, vagy relative többször kapnak processzor-időt. A programokat fontosság szerint sulyoztuk. Az ilyen rendszerben a programok nagyjából egyenletesen terhelik a processzort. Egy harmadik változat az ugynevezett prioritásos multiprogramozás (ezt használja az ESZR/DOS operációs rendszer is). Itt az egyes programok között fontossági sorrendet állapitunk meg, azaz az egyes programokat különböző prioritási szintre tesszük. Mindez jelképesen értendő: a programok között a Supervisor állapit meg bizonyos fontossági/elsőbbségi sorrendet. Ezután mindig az a program kapja a vezérlést, amely a legmagasabb prioritásu, és nem éppen perifériamüvelettel foglalt. A 185. ábrán egy háromszorosan multiprogramozott rendszer müködését láthatjuk.

Legmagasabb a Supervisor prioritása, legalacsonyabb az 1. programé. Tegyük fel, hogy kezdetben a 3. program fut. Ez input/output müveletbe kezd, a processzor tehát felszabadul. Rövid ideig fut a Supervisor, amely elinditja az input/output müveletet, és továbbadja a vezérlést a 2. programnak (1). Ezután a 2. program fut addig, mig ő is input/outputba nem kezd (2). Ezt ismét a Supervisor inditja, és a processzort az 1. programnak "adja" (4). Amikor a 2. program mágnesszalagmüvelete befejeződött, az input/output megszakitást a Supervisor kezeli, és a legmagasabb prioritásu várakozó programnak adja a vezérlést, ez most a 2. program (5). Ez fut mindaddig, mig a 3. program lemezmüvelete véget nem ér (6). Ekkor a Supervisor kezeli a lemez miatti megszakitást és a vezérlést a 3. programnak adja (7). Ez fut most tovább (8).

Szeretnénk felhivni a figyelmet arra, hogy a prioritásos multiprogramozást tönkre lehet tenni akkor, ha minimális input/output igényü programot magas prioritási szintre teszünk! Ekkor ugyanis a többi program "nem jut szóhoz", mert ez a másik nem adja át a vezérlést, csak hiba esetén. Ezért a prioritások hozzárendelésénél nagy körültekintéssel kell eljárni.



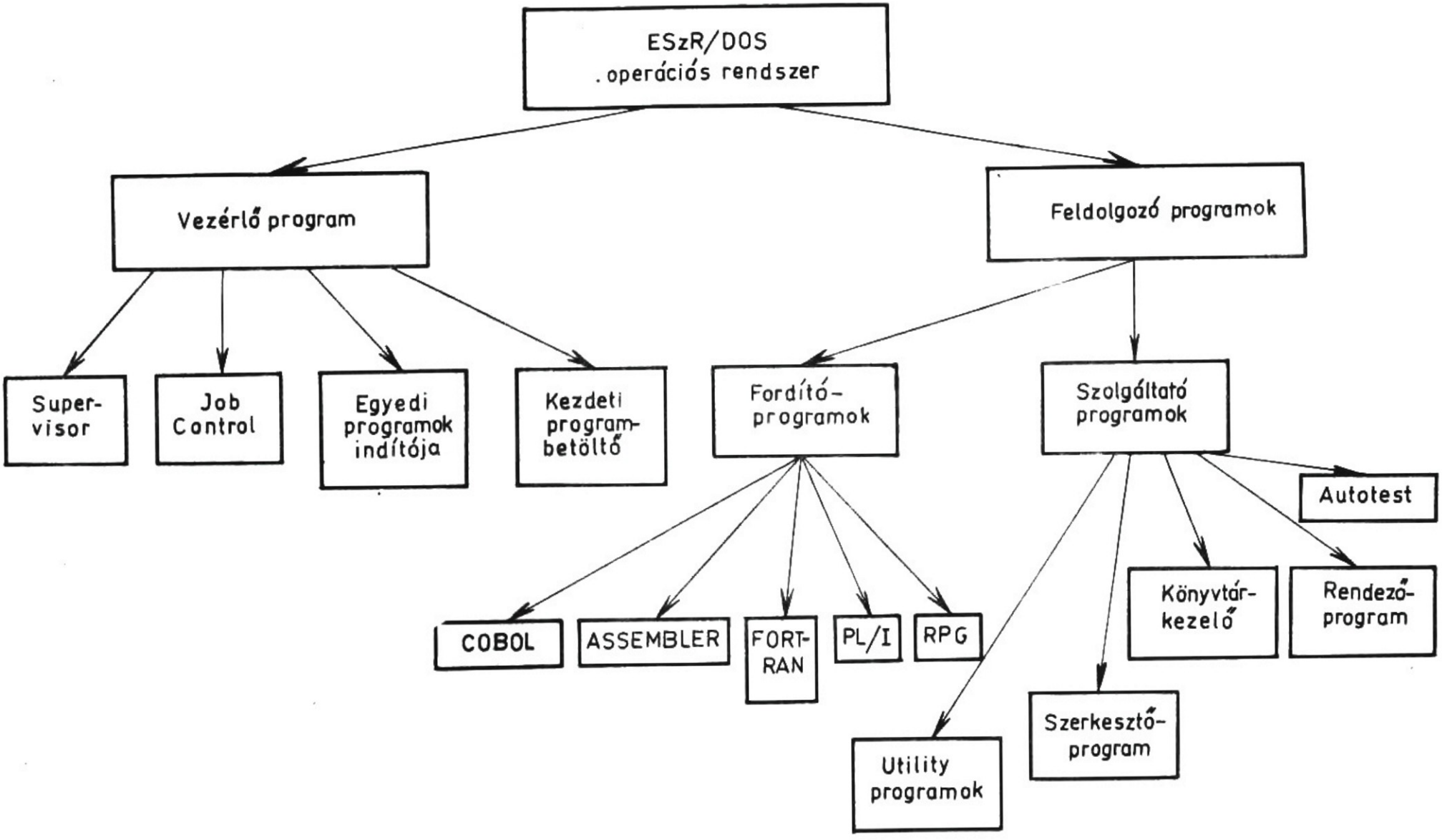
185. ábra. Példa prioritásos multiprogramozásra

Fizikailag a prioritás meghatározása azt jelenti, hogy az egyes programok melyik tárrészben futnak. A tár több önálló részre van osztva (csak software szempontból!) és a Supervisor annak alapján itél egy programnak alacsony vagy magas prioritást, hogy melyik tárrészben van az illető program. E tárrészeket particióknak nevezzük, számuk 1-16 lehet.

Végezetül álljon itt egy tömör ESZR definició, amely az operációs rendszert jellemzi: az operációs rendszer olyan, az automatikus üzemeltetést végző programrendszer, amely a számitógép optimális kihasználására törekszik multiprogramozott üzemmódban.

5.2 AZ OPERÁCIÓS RENDSZER FELÉPÍTÉSE

A III. generációs számitógépeknél az operációs rendszer jelenléte már követelmény. Itt ugyanis olyan fejlett és nagysebességü hardware berendezések üzemeltetésére van szükség, amely csak egy automatizált üzemmódban dolgozó, hatékony software támogatás mellett válik lehetségessé. Az operációs rendszer feladata tehát nem csupán az, hogy a számitógéprendszer egyes elemeit hatékonyan üzemeltesse, hanem az is, hogy az egész rendszer kihasználtságát maximálisra növelje. Más szavakkal: az a célja az operációs rendszernek, hogy lehetőleg sohase álljon a gép egyik eleme sem. Ez természetesen csak közelitőleg érhető el, de az állásidőt nagymértékben lehet csökkenteni ügyes munkaszervezéssel. Az operációs rendszer vezérlése alatt a multiprogramozott üzemmód kivánatos, hiszen csak



186. ábra. Az ESZR/DOS operációs rendszer felépítése

igy lehet a rendszer összes elemét egyidejüleg foglalkoztatni. Ezen tulmenően az operációs rendszer sok biztonsági, ellenőrzési funkciót magára vállal, ezzel egy üzembiztosabb gépet ad a felhasználó kezébe.

Tekintsünk meg most egy rendkivül elterjedt operációs rendszert, milyen elemekből áll, és mi ezek feladata! Az operációs rendszer komponensei a 186. ábrán láthatók.

5.2.1 AZ OPERÁCIÓS RENDSZER VEZÉRLŐ PROGRAMJAI

Az operációs rendszer programjai két fő csoportba sorolhatók: vezérlő programok és feldolgozó programok. A vezérlő programok feladata az üzembiztos, automatizált müködés megvalósitása. A feldolgozó programok különféle szolgáltatásokat nyujtanak a felhasználónak, és általában a vezérlő programok felügyelete alatt futtathatók.

A számitástechnika fejlődésével együtt jár az, hogy a meglevő rendszerprogramokat állandóan továbbfejlesztik, javitják tulajdonságaikat. A javitott változatokat azután a gyártók a felhasználó rendelkezésére bocsátják. Ebből adódik az, hogy némelyik számitóközpontban ugyanannak a rendszernek esetleg igen sok változata is megvan.

Az ESZR/DOS operációs rendszer mágneslemez-orientált. Ez azt jelenti, hogy a rendszerprogramok mind mágneslemezen találhatók, és többnyire mágneslemez-területeket használnak munkaterületként. A mágneslemez a leggyorsabb periféria, ez tehát előnyösen befolyásolja a rendszer müködési sebességét.

A vezérlő program maga is több részből áll. A Supervisor az a program, amely a rendszer épségét érintő összes funkciót magának tartja fenn. Ilyenek például az input/output müveletek, a megszakitáskezelés, memóriavédelem stb. A Supervisor legfőbb eszköze itt a központi számitógép Supervisor állapota, amely, mint tudjuk, felhasználói programok számára a rendszer vezérlése alatt nem hozzáférhető. A Supervisor ily módon mintegy "felügyel" a rendszer helyes müködésére.

A Job Control feladata az automatizált programfuttatás megszervezése. Az egyes programok az operációs rendszer számára munkaegységeket, job-okat jelentenek; Job Control tartja fenn a job-ok közötti folyamatosságot. Minden job belépése előtt előkészíti a rendszert a job fogadására, adminisztrálja a felhasznált gépidőt, kiszolgálja a job-ok igényeit. A Job Control-hoz a programból Job Control utasitásokkal tudunk hozzáfordulni.

Az egyedi programok inditója lényegében hasonló feladatokat lát el, mint a Job Control, azzal a különbséggel, hogy csak egyetlen program futtatását végzi.

A vezérlő programhoz sorolják még az IPL programot is (Inital Program Loader = Kezdeti programbetöltő). Ennek az a feladata, hogy az operációs rendszer egy részét - lényegében a Supervisort - a rendszer inditásához betöltse a tárba. Sulyos hiba esetén többször is szükség lehet kezdeti programbetöltésre, de normális esetben ez a program csak müszakkezdéskor használatos.

5.2.2 AZ OPERÁCIÓS RENDSZER FELDOLGOZÓ PROGRAMJAI

A feldolgozó programok egy sereg lehetőséget nyujtanak a hatékonyabb üzemeltetésre. Igy például a fordítóprogramok hidalják át a programozó algoritmikus gondolkodásmódja és a hardware kötött, merev gépi nyelve közötti problémákat.

A szolgáltató programok célja is a hatékonyabb üzemeltetés elősegitése. A utility programok a sokszor ismétlődő, rutinfeladatok megoldására szolgálnak. A szerkesztőprogram teszi lehetővé azt, hogy már leforditott, "félig kész" programokat számos kombinációban összekapcsoljunk, ily módon esetenként más-más programot készitsünk ugyanazon elemekből. A könyvtárkezelő programok az operációs rendszer könyvtárait tartják karban. A könyvtárak olyan mágneslemezterületek, ahol már megirt programokat őrzünk későbbi feldolgozás céljából. Ezek már benne vannak a rendszerben, igy igen könnyen aktivizálhatók. A felhasználónak tehát nem kell külső adathordozón, például lyukkártyán tartania programjait, sokkal kényelmesebb azokat a könyvtárakba katalogizálnia.

A rendezőprogramok adatállományok rendezését végzik megadott szempontok alapján. Lehetőséget nyujtanak arra is, hogy több input adatállományból egyetlen rendezett állományt hozzunk létre. Ezzel igen sok programozási munkaráforditást takaritanak meg a felhasználónak.

Az AUTOTEST program a megirt programok vizsgálatát, tesztelését segiti. Erre különösen akkor van nagy szükség, ha a programokat ki akarjuk próbálni, hogy helyesen működnek-e.

Az operációs rendszer helyfoglalása igen nagy. Ezért ugy van szétosztva funkciók szempontjából, hogy egy adott része állandóan a főtárban van, ez a Supervisor nucleus. A többi eleme eseténként kerül be a tárba, a futás időtartamára. A továbbiakban részletesen is megvizsgáljuk az egyes komponensek működését.

5.3 AZ OPERÁCIÓS RENDSZER VEZÉRLŐ PROGRAMJA

5.3.1 A SUPERVISOR FELÉPÍTÉSE ÉS FUNKCIÓI

Mint láttuk, a Supervisor az operációs rendszernek az a vezérlő és felügyelő programja, amely a többi, feldolgozó és vezérlő programmal állandó kapcsolatban áll. A rendszer legvédettebb funkcióit a Supervisor magának tartja fenn, ezzel egyszersmind vigyáz is a rendszer épségére (például megakadá lyozza azt, hogy egyik program a másik számára kijelölt területet a tárban felülirja stb.). A feldolgozó programok igényelhetik a Supervisor ezen szolgáltatásait; a Supervisorhoz fordulás eszköze a már ismert SVC gépi utasitás, amely a program szándékos megszakitását eredményezi, és a vezérlést a Supervisornak adja át. Egyébként a Supervisor is ugyanazokat a hardware eszközöket használja, mint a többi program, igy kiváltságos helyzetéről csak software szempontból van értelme beszélni!

Alább felsoroljuk azokat a főbb szolgáltatásokat, amelyeket a Supervisor nyujt a feldolgozó programoknak. Ezek előnye nem csupán az, hogy a Supervisor "vigyáz a rendszer épségére", hanem e sok szor előforduló müveleteket nem kell minden esetben ujra programoznunk! Azt mondjuk, hogy az operá ciós rendszer bizonyos támogatást nyújt a feldolgozó programnak.

a) Memóriavédelem

A memóriavédelem működésének tárgyalásakor láttuk azt, hogy a főtár minden egyes 2 Kbyte-os területéhez 1-1 tárvédelmi kulcs rendelhető hozzá. A kulcsok értékének a beállitása azonban privilegizált művelet, és csak akkor hajtható végre, ha a központi számitógép Supervisor állapotban van. Ne tévesszük össze a Supervisor állapotot a Supervisor programmal; az előbbi azt jelenti, hogy az aktuális programállapotszó 15. bitje a Supervisor állapotnak megfelelő értékű! (lásd 3.4.8 pont) A Supervisor feladata lesz tehát az, hogy a kivánt tárvédelmi kulcsokat beállitsa. Ezzel a Supervisor a főtárat négy részre osztja, ezeket particióknak nevezzük. Minden partícióban egy program futhat egy adott időben. A partíciók elhelyezkedését és tárvédelmi kulcsuk értékét a 187. ábrán láthatjuk.

Super-visor nucleus Kulcs: 0000	Állandóan felosztott tárterület Kommunikációs terület Supervisor nucleus funkciók
Super-visor tranziens terület Kulcs: 0000	Supervisor tranziens funkciók
Háttér-terület (BG) Kulcs: 0001	Feldolgozó programok és Job Control
2. előtér-terület (F2) Kulcs: 0010	Feldolgozó programok SPI vagy Job Control
1. előtér-terület (F1) Kulcs: 0011	Feldolgozó programok SPI vagy Job Control

187. ábra. A főtár felosztása partíciókra

A négy partició mérete 2 Kbyte egész számu többszöröse lehet, bár bizonyos megkötések érvényesek a partíciók minimális méretére nézve. Multiprogramozásnál a Supervisor az alábbi, csökkenő prioritási sorrendet állapitja meg:

1. (legmagasabb) prioritás: Supervisor
2. Foreground-1 (1. előtér) partició
3. Foreground-2 (2. előtér) partició
4. Background (háttér) partició.

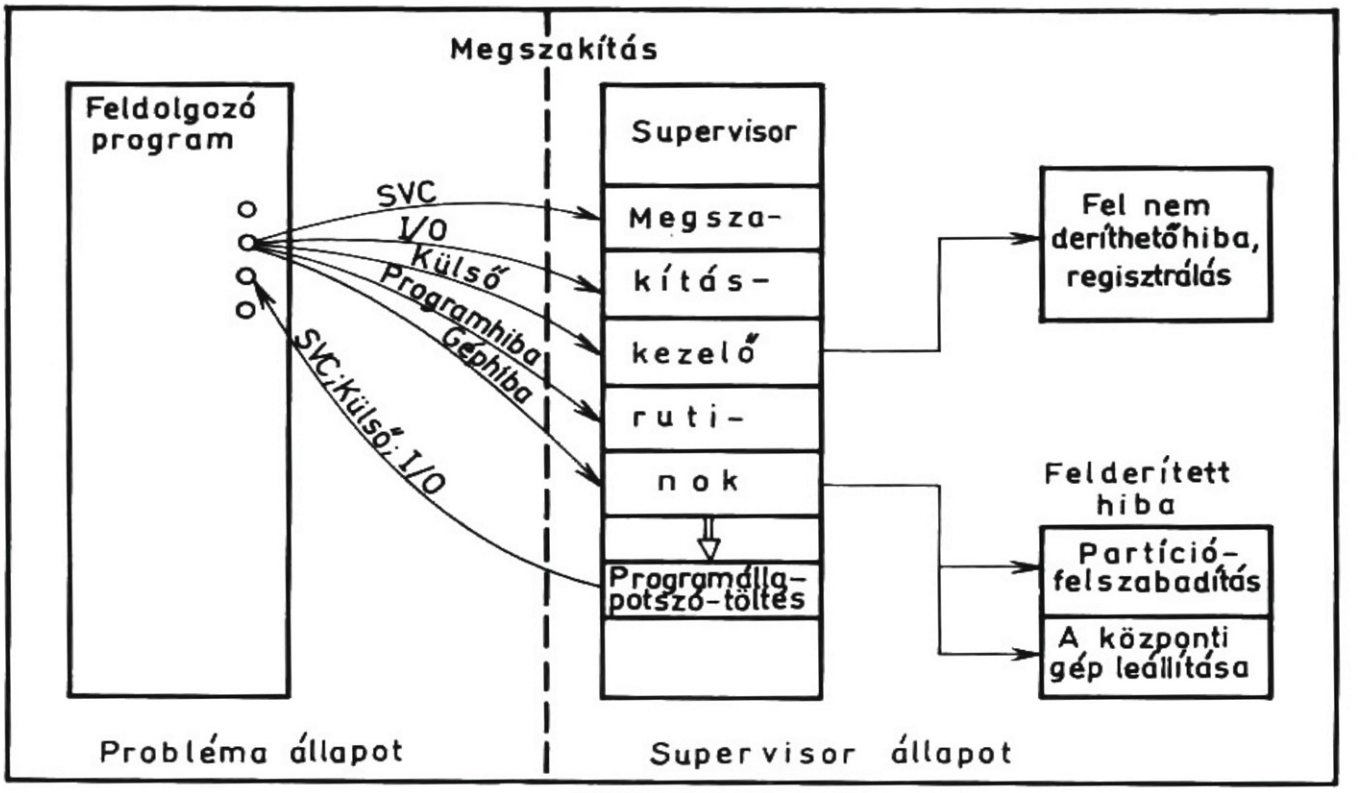
A Supervisor partíciója két részre osztható: a főtár elején található a Supervisor rezidens része (nucleus), ez után pedig a tranziens terület áll. A felosztást az indokolja, hogy a Supervisor-szolgáltatások egyik részére "állandóan és azonnal" szükség lehet, igy ezeket állandóan a tárban kell tartanunk. A többi szolgáltatás "megvárhatja", mig a rendszerlemezről beolvassuk az illető funkciót megvalósitó kis programot. Ezek számára tranziens területet foglalunk, ahol a tranziens rutinok időben egymásután futhatnak, tulajdonképpen "egymás helyét" foglalják el a tárban. Ha minden tranziens rutint állandóan a tárban tartanánk, igen nagy lenne a Supervisor mérete, és ez rossz tárkihasználással járna. A többi partícióban tetszőleges feldolgozó programok futhatnak; ezek lehetnek az operációs rendszer saját elemei is, vagy valamelyik felhasználói program. E tekintetben nincs különbség közöttük.

b) Megszakitáskezelés

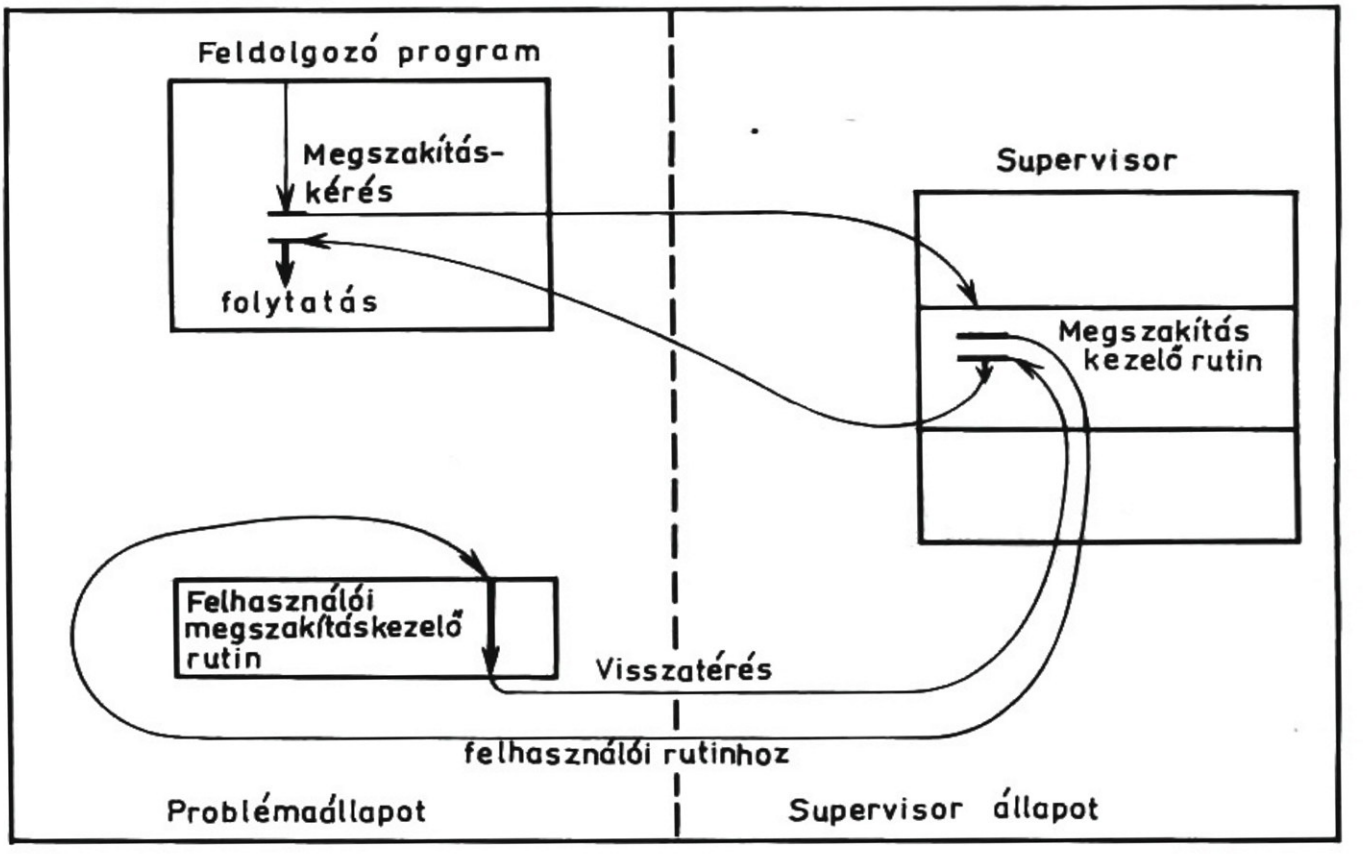
Minden programmegszakitás programállapotszó-cserével kezdődik. A betöltött uj programállapotszóban levő utasitáscim a Supervisor valamelyik megszakitáskezelő rutinjának kezdetére mutat, ez egyszersmind átviszi a központi számitógépet Supervisor állapotba, hogy a Supervisor bármelyik utasitást végrehajthassa. Az öt megszakitási forrás első közelitésben öt különböző kezelést igényel (például más intézkedésekre van szükség a géphiba felderitéséhez, mint az input/output vagy külső megszakitás kezeléséhez stb.). Ennek megfelelően ötféle megszakitáskezelő rutin található a Supervisorban; ezeket mutatja vázlatosan a 188. ábra.

Valamennyi megszakitáskezelő rutin ugy végeződik, hogy visszatölti az elmentett programállapotszót, s igy visszaadja a vezérlést a megszakitott programnak. Ha ez a program például sulyos hiba miatt nem folytatható, akkor természetesen más a helyzet. Ilyenkor a Supervisor "kilövi" a hibás programot, más szavakkal: felszabaditja az illető partíciót a következő program számára.

Bizonyos fajta megszakitási feltételek kezelését a feldolgozó program is elvégezheti. Ilyenek: a gépkezelő bejelentkezése, programhibák, programmegszakitás meghatározott idő elteltével stb. A megszakitáskezelés itt is programállapotszó-cserével és a Supervisor-rutin futásával kezdődik, ám ez a rutin visszaadja a vezérlést egy előre megirt, adott cimre betöltött megszakitáskezelő rutinnak. Ez fogja elvégezni az érdemi megszakitáskezelést, és ennek kell gondoskodnia a Supervisorba való visszatérésről is! A leirt folyamatot a 189. ábrán láthatjuk arra az esetre, amikor meghatározott idő elteltével fellépő megszakitási feltételt kezel egy felhasználói rutin. Megjegyezzük, hogy a rendszer épsége szempontjából fontos megszakitási feltételeket (például input/output, géphiba) a Supervisor mindig fenntartja magának!



188. ábra. A Supervisor megszakitáskezelése

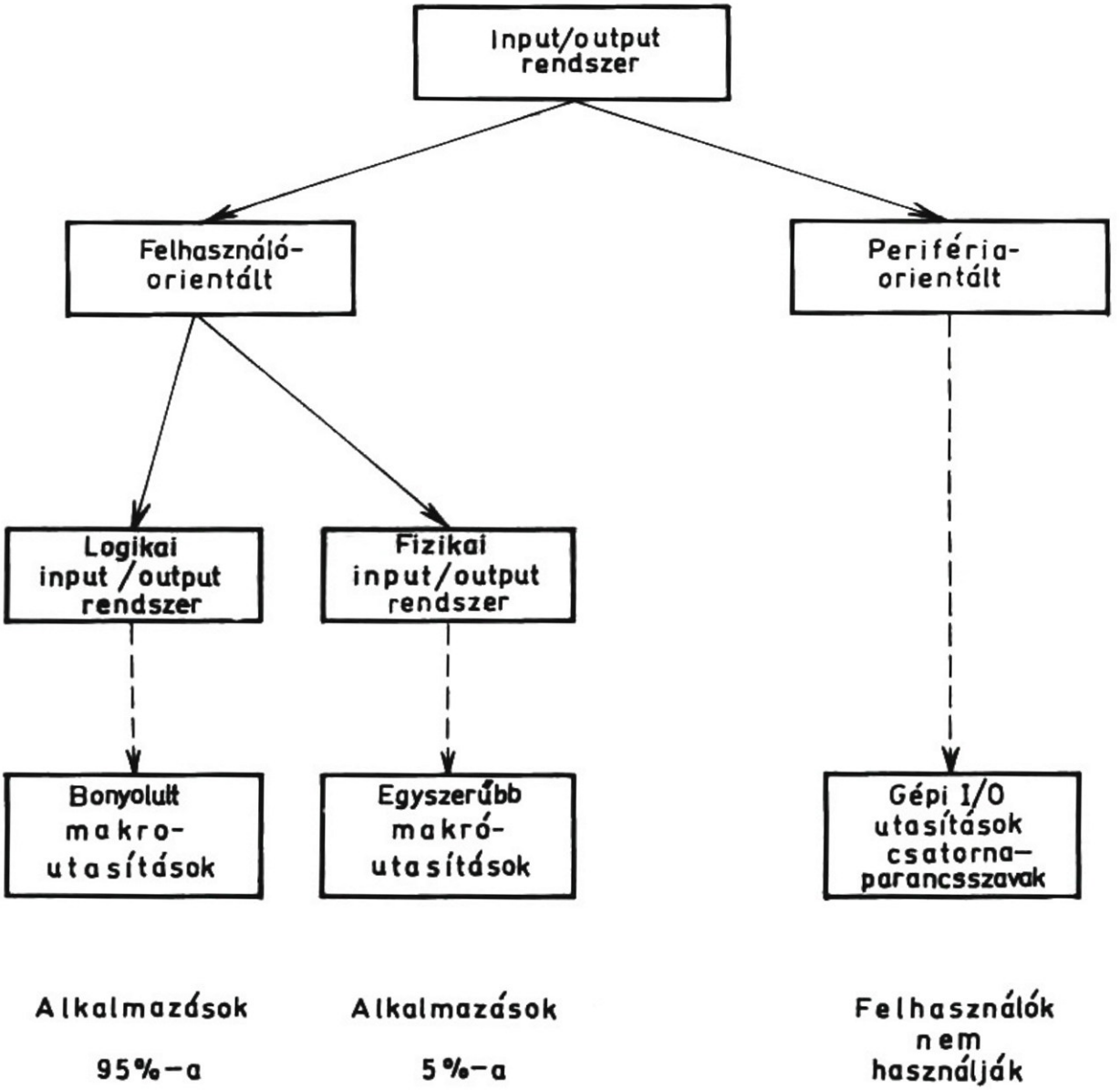


189. ábra. Megszakitáskezelés feldolgozó programmal

c) Input/output müveletek lebonyolitása

Az input/output rendszer hibátlan működése az operációs rendszer életképességének egyik legfontosabb feltétele. Gondoljunk csak arra, hogy például mágneslemezen van maga a teljes operációs rendszer, a Supervisor-tranziensek, cimkék, adatbázisok stb. Ezért valamennyi input/output művelet csak a Supervisor által indítható; az input/output gépi utasitások is privilegizáltak.

Input/output müveleteket három különböző szinten végezhetünk: gépi szinten rendszer nélkül, fizikai szinten és logikai szinten. A különböző szintek programozástechnikai eszközeit a 190. ábra mutatja. A három közül a Supervisor a fizikai szinthez szükséges rendszert biztositja a programok számára, de a logikai szintü programok is a fizikai szintü programrendszert használják, hiszen a Supervisort nem tudják megkerülni! A logikai szintü kezeléshez szükséges szolgáltatásokat az operációs rendszer program-moduljai (logikai I/O rendszer) nyujtják. A fizikai és logikai elnevezés onnan származik, hogy fizikai szinten az átvitt adatok strukturájáról jóval többet kell tudnunk, mig logikai szinten adatállományok "logikai" neveivel, rekordhosszakkal stb. dolgozunk, a müveletek részletes programozását a logikai input/output vezérlő rendszer magára vállalja.



190. ábra. Az input/output vezérlő rendszer felépitése

A fizikai szintü input/output programozás eszköze három makróutasitás. Ezek:

CCB (Command Control Block - I/O vezérlő blokk)

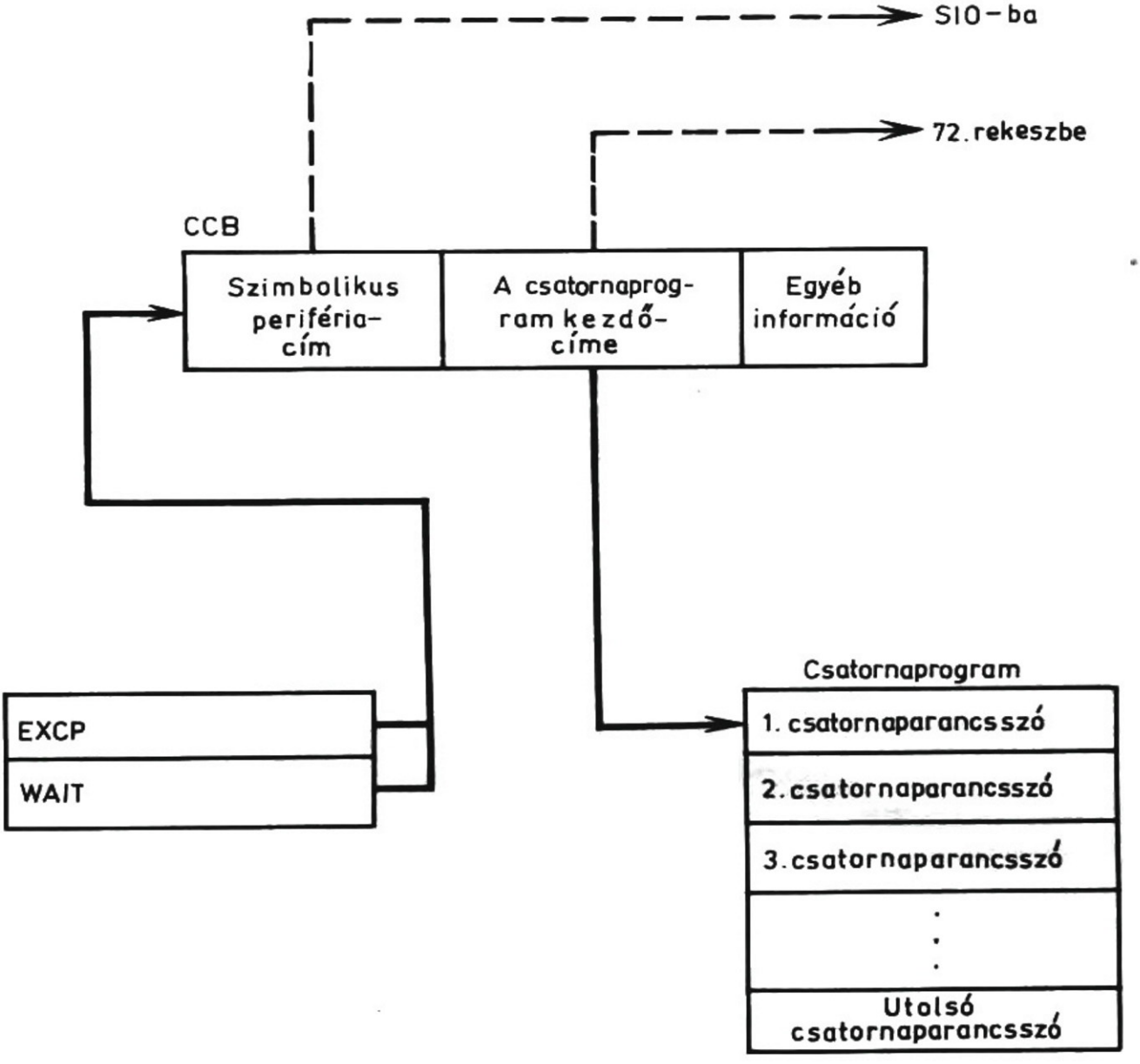
EXCP (Execute Channel Program - csatornaprogram végrehajtása)

WAIT (Várakozás input/output megszakitásra).

A CCB makróutasitás meghatározott nagyságu tárterületet foglal a programban; ebben irhatja le a felhasználó az inditani kivánt input/output müveletet. Megadja a használni kivánt periféria cimét és a csatornaprogram kezdőcimét (vagyis az első csatornaparancsszó cimét).

Az EXCP makróutasitás mindössze azt a Supervisor-szolgáltatást kéri, hogy a Supervisor inditsa el a CCB-ben leirt müveletet.

A WAIT makróutasitás biztositja azt, hogy ez a feldolgozó program ne folytatódjon addig, mig a teljes input/output müvelet végrehajtása meg nem történt. A WAIT elhagyható, ha a program az I/O müvelet befejezése nélkül is tud tovább dolgozni (pl. másik periférián is végezhet I/O müveletet párhuzamosan). Ilyenkor azonban ügyelni kell arra, hogy a WAIT elhagyása miatt ugyanarra a perifériára ne "torlódjanak össze" a várakozó I/O kérelmek, mert ez a rendszer funkcionális zavarát okozza. A fizikai szintü input/output programozást a 191. ábrán láthatjuk.



191. ábra. A fizikai szintü input/output programozás eszközei

Az EXCP makróutasitás hatására a Supervisor elinditja a müveletet. Ehhez kiad egy Start Input/ /Output utasitást, amelyben a perifériacimet a CCB-ből veszi. Ugyancsak a CCB-ből tudja meg azt, hogy hol kezdődik a tárban a csatornaprogram; ezt a cimet betölti a csatornacimszóba (72. rekesz). A többi már a csatorna hardware dolga: egymás után végrehajtja az összeláncolt csatornaparancsszavakat, vagy ha csak egy van, azt az egyet. Ezután programmegszakitási feltételt generál, és kitölti a csatornaállapotszót (64 rekesz). Ha WAIT-et is használunk, akkor mindeddig áll a programunk. Ezután a Supervisor input/output megszakitáskezelést végez, és a csatornaállapotszó összes lényeges információtartalmát beirja a CCB-be. Igy visszakapjuk a byte-számláló, perifériaállapot, csatornaállapot stb. információt, amelyet ezután továbbinditott programunkkal dolgozunk fel. Látjuk, hogy tulajdonképpen az egész müveletet mi programoztuk, a Supervisor csak a privilegizált SIO kiadásánál és a védett 72. és 64. rekesz eléréséné "segitett". Megjegyezzük, hogy a logikai szintü input/output programozásnál nem kapjuk vissza a csatornaállapotszó tartalmát, emiatt nem áll módunkban kiértékelni az input/output müvelet lefolyását. Ezt a rendszer másképpen teszi lehetővé, hibaüzenetek, egyéb értesitések segitségével. A fizikai szintü input/output programozás tehát nagyon "közel áll" a gépi szinthez.

d) Periféria-hibakezelés és regisztrálás

Az input/output müveletek inditásánál, lefolyásánál gyakran előfordul, hogy adatátviteli hiba keletkezik. Nyilvánvaló, hogy hibás, megbizhatatlan adatokkal értelmetlen dolog lenne foglalkoznunk, ezért meg kell kisérelni az ujbóli adatátvitelt, ha ez lehetséges. Azért, hogy ezt ne kelljen külön programoznunk, a Supervisor bizonyos tipusu hibáknál a felderitett hibát megkisérli kijavitani. Itt nem olyan hibákról van szó, mint például vezetékszakadás, hiszen ezekkel szemben a Supervisor tehetetlen. Ha azonban mágnesszalagról olvasunk, és a beolvasott adatok paritáshibásak, érdemes ujból megkisérelni az olvasást stb. Az ilyen tipusu hibákat már javithatja a Supervisor, sőt ezek előfordulásáról statisztikát is készit. Igy időnként kiértékelhetjük a hardware egységek megbizhatóságát. A gyüjtött adatokat a Supervisor egy speciális, hibastatisztika file-ba teszi.

e) Kapcsolattartás a gépkezelővel

Müködés közben szükséges, hogy a gépet kezelő személy értesüljön az üzem közben előforduló eseményekről, illetve bizonyos beavatkozási lehetősége legyen a programfutásba. Erre már láttunk egy lehetőséget, mégpedig a kezdeti programbetöltésnél. Ott a gépkezelő a rendszer egyes jellemzőit határozhatta meg. A Supervisor ennél sokkal sokrétübb, kétirányu kommunikációt tesz lehetővé. A kommunikáció eszköze a konzolirógép (nem pedig a kezelőpult!), a konzolirógéppel akármelyik program irathat üzenetet. Mivel azonban a konzolirógép közönséges input/output perifériális egység, a vele folytatott adatforgalom input/output müveletnek minősül. Emiatt minden program a Supervisor-on keresztül tud a konzolirógéphez férni, és a gépkezelő üzeneteit is a Supervisor fogadja.

Az üzenetek előtt megjelenik az, hogy melyik partícióban futó program üzent (BG, F1 stb.). Ezután egy alfanumerikus kód következik, ez az üzenet azonositója. Ebből már kiderül, hogy az illető program gépkezelői beavatkozást kér, döntést kér valamiben, vagy egyszerüen csak értesit valamilyen eseményről. Végül szöveges formában megjelenik az üzenet tartalma is. Az alábbi üzenetben például a rendszer egy periféria kijelölését kéri a gépkezelőtől:

BG 1 C10A PLEASE ASSIGN SYSRDR

Itt BG jelenti azt, hogy a háttérparticióból érkezett az üzenet, 1 mutatja, hogy a Job Control vezérlő program üzent, C10 az üzenet kódja, A pedig gépkezelői beavatkozást, aktivitást jelent.

Hasonló a kommunikáció a gépkezelőtől a rendszer felé: a gépkezelő parancsokat adhat a rendszer egyes elemeinek. Például felszabadithat egy particiót, leállithatja egy program futását, közvetlenül beirhat az operativ tárba stb. A gépkezelőnek tehát számos lehetősége van az üzemeltetés módositására, ezért a gépkezelő képzettségén, szaktudásán igen sok mulhat egy számitóközpontban. A gépkezelői parancsok általában rövidebbek, mint a rendszer üzenetei. Szabványos formájuak, tehát a gépkezelő csak rögzitett szöveget irhat le. Ha hibás karaktert üt, a rendszer nem tudja értelmezni a parancsot, ugyanis a parancsot fogadó rutinok éppen a szöveg karaktereinek végigvizsgálásából döntik el, hogy mit akart a gépkezelő. Téves gépelés esetén lehetőség van az egész parancs törlésére és ujrairására.

A gépkezelővel való kapcsolattartás eredménye a gépnapló; ezen minden üzenet és parancs rajta van. Ennek alapján lehet ellenőrizni visszamenőleg a gép működését.

f) Programbetöltés

Programbetöltésen azt értjük, hogy egy futtatható programot valamilyen háttértárolóról az operativ tárba töltünk. A betölthető programok mind gépi kódban vannak, és közvetlenül futtathatók. Tulajdonképpen a programbetöltés is input művelet, ezért a Supervisor felügyelete alá tartozik. A betöltő program (System Loader) a megadott név alapján előkeresi a futtatandó programot a háttértárolón, és az operativ tár meghatározott címétől kezdődően betölti. A futtatható programok mindig a háttértároló erre kijelölt helyén találhatók; a DOS operációs rendszerben ez egy mágneslemezterület, amelyet fáziskönyvtárnak nevezünk.

Kétféle betöltési mód létezik: az egyikkel azonnal futtatandó programokat töltünk be, a másikkal pedig olyan információt, amelyet nem akarunk azonnal "végrehajtani" (például kódolt táblázatokat stb.).

g) Ellenőrző/ujrainditási pontok elhelyezése

Olyan programoknál, amelyek futási ideje igen hosszu, esetleg több óra, előre felkészülhetünk arra, hogy hiba esetén ne kelljen az egész programot elölről inditani. A programban ugynevezett ellenőrzési pontokat helyezhetünk el a Supervisor segitségével; ezeknél a pontoknál a Supervisor egyik tranziens rutinja kimenti a futó programot és a rendszer állapotát a központi számitógépből valamelyik háttértárolóra. Ez lehet mágnesszalag vagy mágneslemez. (Lásd 4.4.1 pont.) A kimentett információban benne van az aktuális programállapotszó, a 16 általános regiszter tartalma és maga a program. A regiszterek tartalmát feltétlenül ki kell menteni, mert ezek báziscimeket, indexeket stb. is tartalmaznak, és nélkülük a program nem lenne tovább futtatható! A programfutás alatt fellépő hiba esetén most már csak a legutolsó ellenőrzési/ujrainditási pontig kell visszamennünk, és onnan folytathatjuk a program végrehajtását. Ezzel sok gépidőt és felesleges géphasználatot takarithatunk meg.

h) End-of-Job kezelés

A job vége feltétel észlelésekor a Supervisor feladata az, hogy betöltse a megfelelő particióba a Job Control vezérlőprogramot, amely előkésziti a particiót a következő job fogadására. Maga a Job Control is közönséges, futtatható program, betöltése tehát a Supervisor-ra hárul. Ha a job a normálistól eltérően fejeződik be (például programhiba miatt nem lehetett végig feldolgozni), akkor a Super-

visor egyik tranziense kinyomtatja a program által elfoglalt tárterület tartalmát byte-onként 2-2 hexadecimális jegy formájában, az összes általános és lebegőpontos regiszter tartalmát hexadecimálisan, és az aktuális programállapotszó tartalmát. Ezenkívül szöveges hibaüzenetet is ad a sornyomtatón; mindebből már a felhasználó dolga felderiteni a hiba okát, keletkezési helyét a programban. Ezt a folyamatot dump-olásnak nevezzük, és a Supervisor DUMP tranziense végzi el. Itt jól látható, hogy miért nem lenne érdemes a dump funkciót rezidensként megvalósitani: dump-ra ugyanis általában csak akkor van szükségünk, ha a program futása hiba miatt megszakadt. A dump olvasásához igen fontos, hogy jártasak legyünk a hexadecimális számrendszerben.

Összefoglalva: a Supervisor az operációs rendszernek az a vezérlő programja, amely az input/output müveleteket, megszakitásokat kezeli, és általában a rendszer épségét közvetlenül érintő beavatkozásokat magának tartja fenn. Ehhez eszközként a központi számitógép Supervisor-állapotát használja fel; ez az állapot feldolgozó programok számára közvetlenül nem hozzáférhető. A Supervisor egyik része - a nucleus - állandóan a főtárban van, többi része - a tranziensek - csak addig, amig szükség van rájuk. A számitógép hardware erőforrásain a Supervisor a többi programmal osztozik, éspedig a megismert prioritás elve alapján.

5.3.2 JOB CONTROL. AZ AUTOMATIZÁLT ÜZEMELTETÉS MEGVALÓSÍTÁSA

Az operációs rendszer másik nagy vezérlő programja a Job Control, a "munkavezérlő" program. Ez a program végzi el az egyik job-ról a másikra való automatikus áttérést, és ezzel biztositja a megfelelő partició folyamatos leterhelését. A Job Control helyigénye 10 Kbyte, ennélfogva csak azokban a partíciókban folytathatunk automatikus feldolgozást, amelyek mérete 10 Kbyte vagy ennél nagyobb.

A Job Control programhoz ugynevezett Job Control utasításokon és Job Control parancsokon keresztül fordulhatunk. A Job Control utasításokat a programban helyezzük el, általában kártyára lyukasztva. A Job Control utasitások irásának szabályait a Job Control nyelv rögziti. Az utasitások kötött formájuak, és az első két karakterpoziciójukon törtvonalat tartalmaznak (//). Ezzel tudjuk megkülönböztetni őket más programoknak szóló utasitásoktól. Job Control utasitást csak akkor adhatunk ki, ha a Job Control program a tárban van, vagyis ha előzőleg a Supervisor betöltötte! A Job Control esetenkénti betöltéséről a Supervisor gondoskodik, ez nem a programozó feladata.

A Job Control parancsokat a gépkezelő adja ki, általában a konzolirógépen. Mivel ezek is Job Controlnak szólnak, csak akkor adhatók ki, ha Job Control a tárban van. Job Control betöltésére a gépkezelőnek külön lehetőségei vannak.

Emlékezzünk arra, hogy Job Control a számitógéprendszer üzembe helyezésekor a kezdeti programbetöltő hatására is a tárba került! Erre azért van szükség, mert Job Control-nak kell elinditania az automatizált programfuttatást, tehát minden feldolgozó program előtt a Job Control intézkedik az illető program inditásáról.

Ha egy adott partícióban futó programokat időben egymásután szemügyre vesszük, azt tapasztaljuk, hogy az egyes programok a Job Control-lal felváltva futnak; amikor egy program befejeződött, Job Control kerül a helyére, és ez ismét beolvastat egy programot stb. A job-nak azt a részét, amely két, egymást követő Job Control futás között helyezkedik el, job step-nek (job lépés) nevezzük. Pontosan meg-

fogalmazva a következőt mondhatjuk: minden egyes job step alatt egy-egy programot futtat a központi számitógép. A Job step-eket éppen a Job Control választja szét: két egymást követő job step között mindig Job Control fut rövid ideig. Ő készíti elő a következő job step alatt futtatandó programot a betöltésre. Egy job természetesen akárhány job step-ből állhat.

Alább felsoroljuk Job Control szolgáltatásait, amelyeket az összes többi program igénybe vehet. Több step-ből álló job-ra az 5.4 fejezetben láthatunk példákat.

a) Programok előkészitése futtatásra

Ezt a funkciót tulajdonképpen a job step inicializálásnak is nevezhetnénk. Arról van itt szó, hogy az adott job step számára elő kell keresnünk a futtatandó programot, és azt be kell töltenünk a megfelelő tárcimre. Mindezt Job Control és System Loader közösen végzi; System Loader a Supervisor egyik része (lásd 5.3.1 pont). Az összes futtatható program nevét és lemezcimét tartalmazó bejegyzésekből Job Control egy ugynevezett fázis-tartalomjegyzéket állit össze. Ez a rendszerlemez 00 09 1 cimén kezdődik, vagyis a 0-dik cilinder 9-ik sávjának első rekordjával. A betöltendő program nevét mi adjuk meg Job Control-nak, mire ő System Loader-t erre a tartalomjegyzékre irányitja. A kikeresett nevü programot ezután System Loader a rendszerlemezről az operativ tárba tölti, és elinditja a futtatását. Ha a futtatandó programot nem vettük fel a fáziskönyvtárba, vagyis nem katalogizáltuk, akkor a Job Control máshonnan keresi meg az illető program lemezcimét. Egyébként a betöltés ugyanugy történik, mint katalogizált programok esetében.

b) Perifériák hozzárendelése szimbolikus cimekhez

Job Control egyik leglényegesebb szolgáltatása a szimbolikus és fizikai perifériacimek kapcsolatának nyilvántartása. Szimbolikus perifériacimekre azért van szükség, hogy a programokat a konkrét perifériacimektől függetlenül lehessen irni. Képzeljük el, hogy egy adott számitógép-konfigurációra megirtunk egy programot, és ez esetleg több száz órányi munkaráforditással járt. Ha most ezt a programot egy másik számitógépen akarjuk futtatni, amelynek gépi nyelve azonos, csak a perifériák vannak másképp elosztva az interface-en, akkor programunkban esetleg a perifériacimek nagy részét át kell irnunk az uj konfigurációnak megfelelően! A program tehát az adott géphez kötött, merev software-termék, más gépeken csak módositásokkal használható. Ennek kiküszöbölésére vezették be a szimbolikus vagy logikai perifériacimeket. Az összes lehetséges perifériákat elnevezték ugy, hogy nevük utaljon a funkciójukra. A programokat most már ugy irhatjuk, hogy konkrét perifériacimek helyett e szimbolikus cimeket (neveket) használjuk, természetesen betartva az előirt korlátozásokat. Amikor a programot egy adott konfiguráción futtatni kivánjuk, nem kell átirnunk egyetlen perifériacimet sem, hanem csupán a programban elnevezett perifériához (a szimbolikus cimhez) hozzá kell rendelnünk az adott konfiguráció egy konkrét perifériájának cimét. A szimbolikus és konkrét perifériacimek kapcsolatát Job Control tartja nyilván egy kettős táblázatban. A táblázat egyik része tartalmazza a használatos szimbolikus perifériacimeket, másik része pedig az adott konfigurációban létező konkrét, fizikai perifériacimeket. A hozzárendelés eszköze egy szám, amely megmutatja, hogy a szimbolikus perifériacimek valamelyikéhez melyik fizikai perifériacim tartozik. A logikai cimek tehát fizikai cimekre "mutatnak". E "mutatók" átirásával lehet megváltoztatni a szimbolikus és fizikai perifériacimek kapcsolatát.

Az operációs rendszerben használható szimbolikus perifériacimek rögzitettek. Egyuttal az is megkötés, hogy melyik cimhez milyen tipusu periféria tartozhat. A szimbolikus cimek kétféle perifériára vonatkoznak: ugynevezett rendszerlogikai egységekre és programozói logikai egységekre. A rendszerlogikai egységeket az operációs rendszer programjai használják; ezeket eleve ugy irták meg, hogy egy-egy adott szimbolikus cimü perifériával dolgozhatnak. A programozói logikai egységek felhasználási köre szabadabb; a programozó "elnevezheti" perifériáit egy adott névválasztékot felhasználva. A szimbolikus cimek és perifériák az alábbi felsorolásban láthatók. Azt, hogy egy szimbolikus perifériacimhez milyen konkrét perifériális egység tartozik, az adott konfigurációnál döntik el.

SYSRDR - System Reader, Job Control input perifériája. Általában kártyaolvasó.

SYSIPT - System Input Unit, a többi program input perifériája. Általában kártyaolvasó ez is.

SYSPCH - System Punch, a rendszer "lyukasztó" output perifériája. Általában kártyalyukasztó vagy mágnesszalag egység.

SYSLST - System List, az összes program listázó perifériája. Általában sornyomtató.

SYSLOG - System Log, a gépnaplót vezető periféria. Általában konzolirógép.

SYSLNK - System Link, a szerkesztőprogram input perifériája. Ez mindig mágneslemez-terület.

SYSRES - System Residence, magát az operációs rendszert tartalmazó periféria. Mindig mágneslemez-terület.

SYSCLB - Private Core Image Library, magán fáziskönyvtár. Mindig lemezterület.

SYSRLB - Private Relocatable Library, magán modulkönyvtár. Mindig lemezterület.

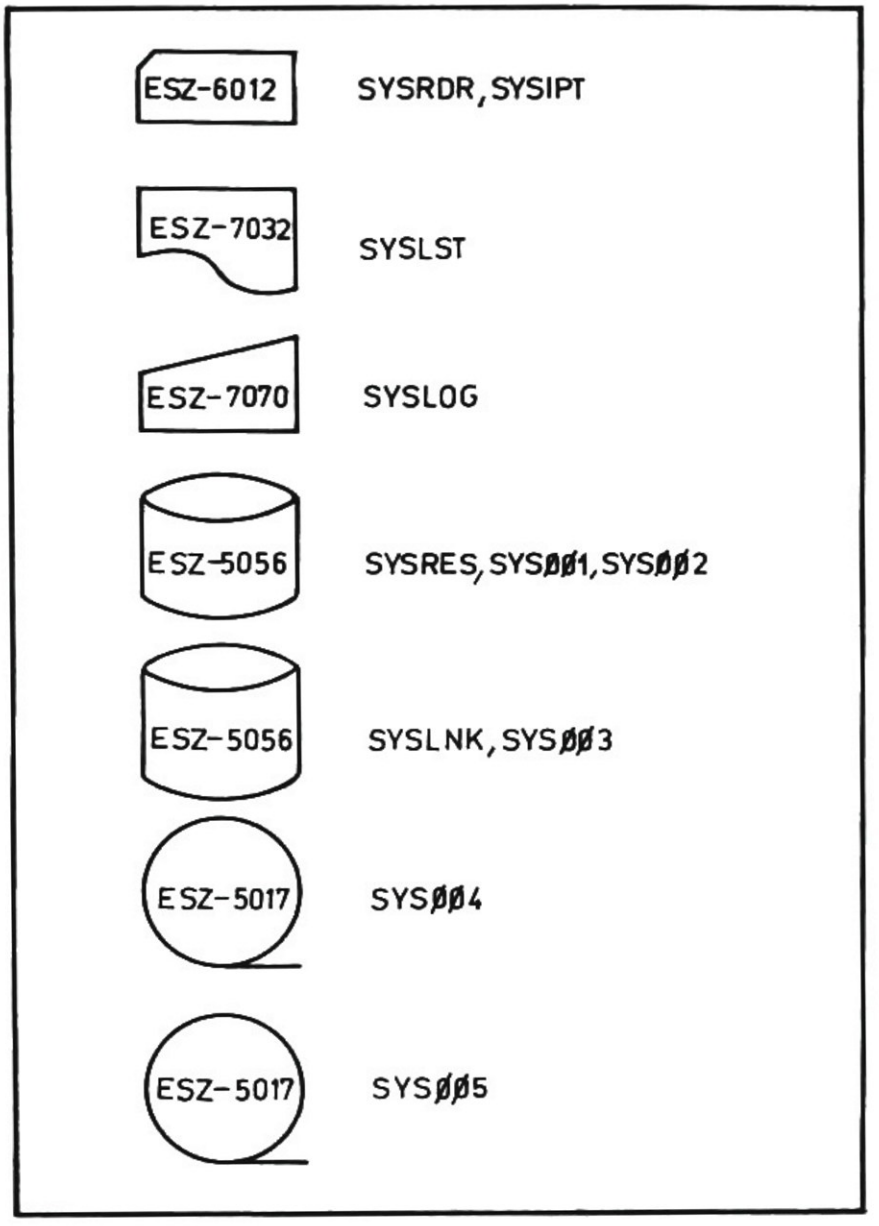
SYSSLB - Private Source Statement Library, magán forráskönyvtár. Mindig lemezterület.

SYSREC - System Recorder File, hibastatisztikákat tartalmazó periféria. Mindig lemezterület.

A felsorolt tizenegy név a rendszerlogikai egységekhez tartozik. Mint látjuk, minden egységhez csak bizonyos tipusu perifériák rendelhetők. Itt a leggyakoribb hozzárendeléseket tüntettük fel.

A programozói logikai egységek neve az alábbi alaku: SYSØØØ-SYSnnn, vagyis a név utolsó karakterei sorszámot tartalmaznak. Maximálisan 244 programozói logikai egység használható, de erre általában nincs szükség. Gyakorlatilag 10-20 programozói logikai egységet használunk, de ezt mindig az adott számitóközpontban működő operációs rendszer határozza meg. Ezekhez tetszőleges fizikai perifériacim hozzárendelhető. Az is megengedett, hogy egy perifériának két vagy több szimbolikus cime legyen. Megforditva viszont nem igaz: egy adott szimbolikus cim csak egy perifériára vonatkozhat (ugyanis a szimbolikus cim csak egy perifériacimre tud "mutatni").

A fizikai és logikai perifériacimek kapcsolatát rendszergeneráláskor határozzák meg. Ezeket a hozzárendeléseket maradandónak, permanensnek nevezzük. Ezzel szemben, a programmal módositott hozzárendelés csak addig van érvényben, mig az adott program feldolgozása tart, ezért az ilyen hozzárendelést időlegesnek nevezzük. A permanens hozzárendelés mindig "erősebb", tehát a hozzárendelést módositó program (job) befejeztével Job Control automatikusan visszaállitja a permanens kapcsolatot. Egy lehetséges perifériahozzárendelés látható a 192. ábrán. Itt némelyik perifériát több logikai névvel is elláttuk; igy mindezen logikai nevek ugyanerre a fizikai perifériára mutatnak a Job Control perifériatáblázatában.



192. ábra. Példa szimbolikus perifériacimek használatára

A perifériahozzárendelések az ASSGN utasitással végezhetők el. Ennek alakja az alábbi:

// ASSGN SYSxxx, fizikai perifériacim.

Itt SYSxxx tetszőleges, érvényes rendszerlogikai vagy programozói logikai egység neve, a fizikai perifériacim pedig olyan alaku, mint az input/output gépi utasitásokban. Ez tehát kifejezi a csatorna és a rákapcsolt periféria cimét. Hexadecimális alakban irjuk, igy például X'281' azt jelenti, hogy a 2. szelektorcsatornán levő 81 cimü perifériáról van szó. A

// ASSGN SYSØØ7, X'281'

Job Control utasitás hatására tehát Job Control ugy módositja saját perifériatáblázatát, hogy a SYSØØ7-nek megfelelő táblázatelem a 281-nek megfelelő táblázatelemre mutasson. Maga a táblázat a Supervisorban van.

c) A kommunikációs terület kitöltése

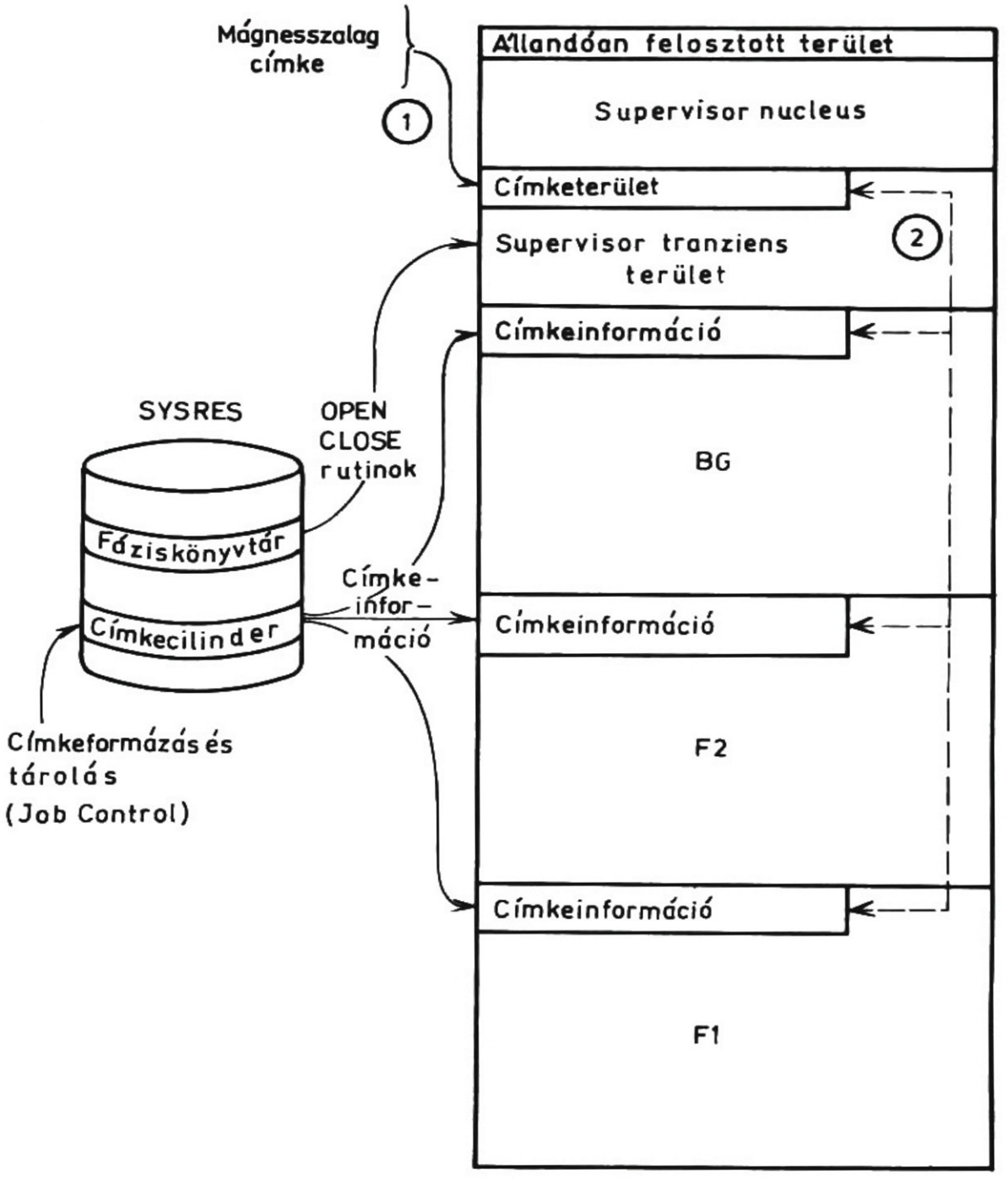
A kommunikációs terület a Supervisor része, és arra szolgál, hogy egyik program (job step!) információt adjon át benne a másik, utána következő programnak. Ha a kommunikációs területet az egyes particiókban helyeztük volna el, akkor a job step végén Job Control rátöltődött volna erre a területre, igy nem tudtunk volna átmenteni információt a következő job step számára. A Supervisorban minden particiónak saját kommunikációs területe van. Egy-egy job inditásakor Job Control beirja ide a job-ra vonatkozó lényeges információkat: a job nevét, a job inditásának időpontját stb. Ezeket részben a gépnapló készitésére, részben gépidőelszámolásra használja. A következő job belépésekor ez az információ felüliródik az aktuális job-ra vonatkozó információval. Megjegyezzük, hogy a kommunikációs területre - annak kijelölt részére - a feldolgozó programok is irhatnak tetszőleges információt, természetesen a Supervisor-on keresztül.

d) Cimkék formázása és tárolása

Mielőtt egy adatállományhoz hozzáférhetnénk, meg kell nyitnunk az illető állományt. Ez a megnyitás abban áll, hogy az állomány cimkéit össze kell hasonlitanunk a programban levő cimkeinformációval. Ha a kettő megegyezik, ez azt jelenti, hogy a program jogosult az állományhoz való hozzáfordulásra, tehát onnan rekordokat olvashat, illetve abba rekordokat irhat. A file-megnyitás természetesen a program futási idejében történik, de a file-cimkéket és a programbeli cimkeinformációt már előzőleg meg kell adnunk. A file-cimkéket a file létrehozásakor rögzitjük, a cimkeinformációt pedig megnyitás előtt a programba beirt Job Control utasitásokkal adjuk meg. Ezek alapján Job Control szabványos cimkéket generál, és azokat a feldolgozás idejére SYSRES egyik cilinderén, az ugynevezett cimkecilinderen tárolja. Nagyon sokszor megnyitandó állományok számára a cimkeinformációt standardizálni lehet. Ez azzal az előnnyel jár, hogy Job Control nem igényli többé a cimkeinformáció megadását a programban. Az egyes particiók cimkeinformációi a cimkecilinder különböző sávjain vannak. A nem standardizált cimkeinformációt a következő file-megnyitás felülirja. Hasonló cimkevizsgálat történik az állomány lezárásakor is; az esetleg módositott, aktualizált cimke visszairásra kerül. A 193. ábra a Supervisor állománynyitó/állomány-lezáró (OPEN/CLOSE) tranzienseinek és Job Control-nak az együttmüködését mutatja a cimkevizsgálatban.

(1) A Supervisor OPEN tranziensei beolvassák a mágnesszalag cimkét a Supervisor tranziens területére.

(2) A beolvasott cimke tartalmát a programban megadott cimkeinformációhoz hasonlitják. Ezt a cimkeinformációt Job Control helyezte el a cimkecilinderen, és ugyancsak az OPEN tranziensek olvassák be a megfelelő particióba. A cimke és cimkeinformáció egyezése esetén az állomány hozzáférhető a program számára. Ellenkező esetben a rendszer megakadályozza a hozzáfordulást. Lényegében ugyanez történik mágneslemez-állományok megnyitásakor is, de ott a cimkéket nem mágnesszalagról, hanem mágneslemezről kapjuk. Figyeljük meg, hogy a cimkecilinder szerepe csupán az, hogy a programból megadott (vagy standardizált) cimkeinformációt tárolja.



193. ábra. Mágnesszalag-állományok megnyitása

e) Programok ujrainditása

Az ellenőrző/ujrainditási pontokkal ellátott programot Job Control inditja ujra a kivánt ellenőrzési ponttól. Betölti a tárba az illető ellenőrzési pont felvételekor elmentett programot, visszaállitja az általános regiszterek és az aktuális programállapotszó tartalmát, valamint ujra kitölti a kommunikációs területet. A program ujrainditását alkalmas Job Control utasitással lehet elvégezni. Minden program csak abban a particióban inditható ujra, amelyben az ellenőrzési/ujrainditási pontokat felvettük.

Összefoglalva:

Job Control az operációs rendszernek az a vezérlő programja, amely biztositja az automatikus programfuttatást az egyes partíciókban. A feldolgozó programok és Job Control partíciónként időben felváltva futnak; Job Control-t a Supervisor tölti be az egyes job step-ek végén, majd Job Control előkészíti a partíciót a következő job step inditására. Az input/output stb. müveleteket Job Control is a Supervisor-on keresztül hajtja végre. Mivel egy müszak alatt Job Control-t igen sokszor be kell olvasni az operativ tárba a rendszerlemezről, nem közömbös az input/output rendszer adatátviteli sebessége. A számitógéprendszer automatizált üzemeltetése addig folytatható, mig el nem fogy Job Control inputja, vagy a gép valamilyen hiba miatt le nem áll. A program-utánpótlásról általában a gépkezelő gondoskodik. Ezt az üzemmódot - mivel a feldolgozás job-onként, kártyakötegenként történik - kötegelt (idegen szóval batch) feldolgozásnak nevezzük. Ez a feldolgozási mód ott terjedt el, ahol a számitógép adatfeldolgozást végez, tehát a számitógéppark döntő többségénél.

A következőkben összefoglaljuk a Job Control-nak szóló főbb utasitásokat és irásmódjukat. A nagybetüs részek kulcsszavak, változtatás nélkül irjuk őket. A kisbetüs szövegek helyére felhasználói információ kerül.

// JOB jobnév	- a job kezdetét jelöli;
/&	- a job végét jelöli;
/*	- lyukkártyafile végét jelöli;

// ASSGN SYSxxx,X'cuu' - perifériahozzárendelésre. Az utasitás első operandusa egy logikai perifériacim, például SYSPCH stb. Ehhez rendeli hozzá az utasitás a második operandusban megadott fizikai cimü perifériát. A cimet aposztrófok közé kell zárni, X pedig azt jelenti, hogy a cimet hexadecimálisan fejeztük ki. Igy például az X'281' cim a 2. szelektorcsatorna 81 cimü perifériáját jelenti. A hozzárendelés végső soron a Supervisorban levő perifériatáblázatok tartalmát módositja.

// OPTION opciólista vesszővel elválasztva; az opciólista elemei kulcsszavak, ezekkel adhatjuk meg azt, hogy Job Control milyen szolgáltatását kérjük vagy nyomjuk el. Ilyen opció például az, hogy a szerkesztendő fázisprogramot katalogizáljuk, ha igen, akkor a CATAL kulcsszót használjuk. Ha a szerkesztés eredményét nem katalogizáljuk, akkor a LINK kulcsszót használjuk. Ha nem is szerkesztünk a job-ban, akkor NOLINK-kel adhatjuk ezt tudtára a rendszernek. Ezeken kivül még sok kulcsszó van, ezekre itt nem térünk ki.

// EXEC fázisnév - a megadott nevü fázisprogramot betölti a tárba és lefuttatja. Operandus hiányában az utoljára szerkesztett fázist tölti be.

5.3.3 SPI. EGYEDI PROGRAMINDÍTÁS

Az operációs rendszer lehetőséget nyujt arra is, hogy egy-egy partícióban ne automatikus programfuttatást végezzük Job Control segitségével, hanem a programokat egyenként inditsuk. Ezt az SPI - Single Program Initiator, az egyedi programok inditója segitségével érhetjük el. Nagy vonalakban az SPI programot Job Control-hoz hasonlónak tekinthetjük azzal a kikötéssel, hogy csak egy program automatikus futtatását végzi el. A programvégrehajtás befejeztével az illető partició leáll.

Az SPI programot elsősorban ott használhatjuk, ahol a partíció mérete kisebb mint 10 Kbyte, és igy Job Control nem fér el benne. Az egyedi programinditás eszköze az a néhány, SPI-nek szóló parancs, amelyet a gépkezelő adhat ki általában a konzolirógépről. A program betöltését itt is a Supervisor végzi. Az SPI-nek szóló parancsok hasonlóak a Job Control parancsokhoz; a legtöbb Job Control-funkció az SPI-nél is megtalálható, természetesen csak egy programra.

Az SPI-vel való folyamatos üzemeltetés meglehetősen nagy terhet ró a gépkezelőre, hiszen állandóan figyelnie kell az illető partíció állapotát. Emiatt SPI-t gyakorlatilag nem használják.

5.4 AZ OPERÁCIÓS RENDSZER FELDOLGOZÓ PROGRAMJAI

5.4.1 FORDÍTÓPROGRAMOK. A PROGRAMFORDÍTÁS MENETE. A FORRÁSPROGRAM FOGALMA, SZERKEZETE

A számitógép-programokat - mint tudjuk - irhatjuk közvetlenül gépi kódban, de irhatjuk valamilyen szimbolikus (szöveges) kódban is. A gépi kódu programozás főleg régen volt divatos, amikor a számitógép még nagyon "uj" találmány volt, és nem álltak rendelkezésre a szimbolikus kódokat feldolgozó eszközök. Ahogyan azonban a számitógépen megoldani kivánt feladatok bonyolultsága nőtt, a gyártó cégek rákényszerültek olyan szolgáltatásokra is, amelyek könnyebbé és egyszerűbbé tették a bonyolult feladatok kódolását. Igy jöttek létre a különféle programozási nyelvek és forditóprogramjaik. Minden forditóprogram feladata végső soron az, hogy egy adott szimbolikus kódból gépi kódot állitson elő.

A sokféle géptipus és gépi nyelv azt eredményezte, hogy a számitógépgyártók egy-egy géptipusra fejlesztették ki a szükséges forditóprogramokat. Mint tudjuk, ez a dolog általában nem érinti a nemzetközileg definiált hivatkozási nyelveket; az egységes felépitésü programozási nyelvek utasitásait ugyanis bármilyen gépi nyelvre lefordithatjuk! Emiatt az egyes programozási nyelveknek sok, egymástól teljesen különböző forditóprogramja van forgalomban, mindegyik egy adott gépre vagy gépcsaládra alkalmazható. Igy például az ESZR-ben valamennyi forditóprogram ugyanarra a gépi nyelvre forditja a FORTRAN nyelven irt programokat, de egy másik gép (például CDC) forditóprogramja ezekből egy teljesen más felépitésü gépi kódot állit elő stb. Ha mindehhez még hozzávesszük azt is, hogy egy adott forditóprogramnak is különböző fejlettségi szintü változatai léteznek, akkor nyilvánvalóvá válik, hogy feladatunk megoldásához igen sok és sokféle forditóprogram közül válogathatunk.

Egy adott forditóprogramot a hozzá tartozó hivatkozási nyelv gépi reprezentációjának nevezzük. Ezek a gépi reprezentációk általában mindig eltérnek kismértékben a hivatkozási nyelvektől; ennek oka az, hogy nincs szükség minden gépi reprezentációban a teljes hivatkozási nyelv lehetőségeinek megvalósitására. A gépi reprezentációk tehát általában szükitik a hivatkozási nyelv adta lehetőségeket (például nem minden, a hivatkozási nyelvben definiált utasitást "ismernek"). A feladatok számitógépre vitelénél tehát ismernünk kell magát a hivatkozási nyelvet és a használt gépi reprezentációját is! A feladat megoldását ugyanis nem egyedül a programozó végzi, hanem ő meghatározott módon "együttmüködik" a forditóprogrammal.

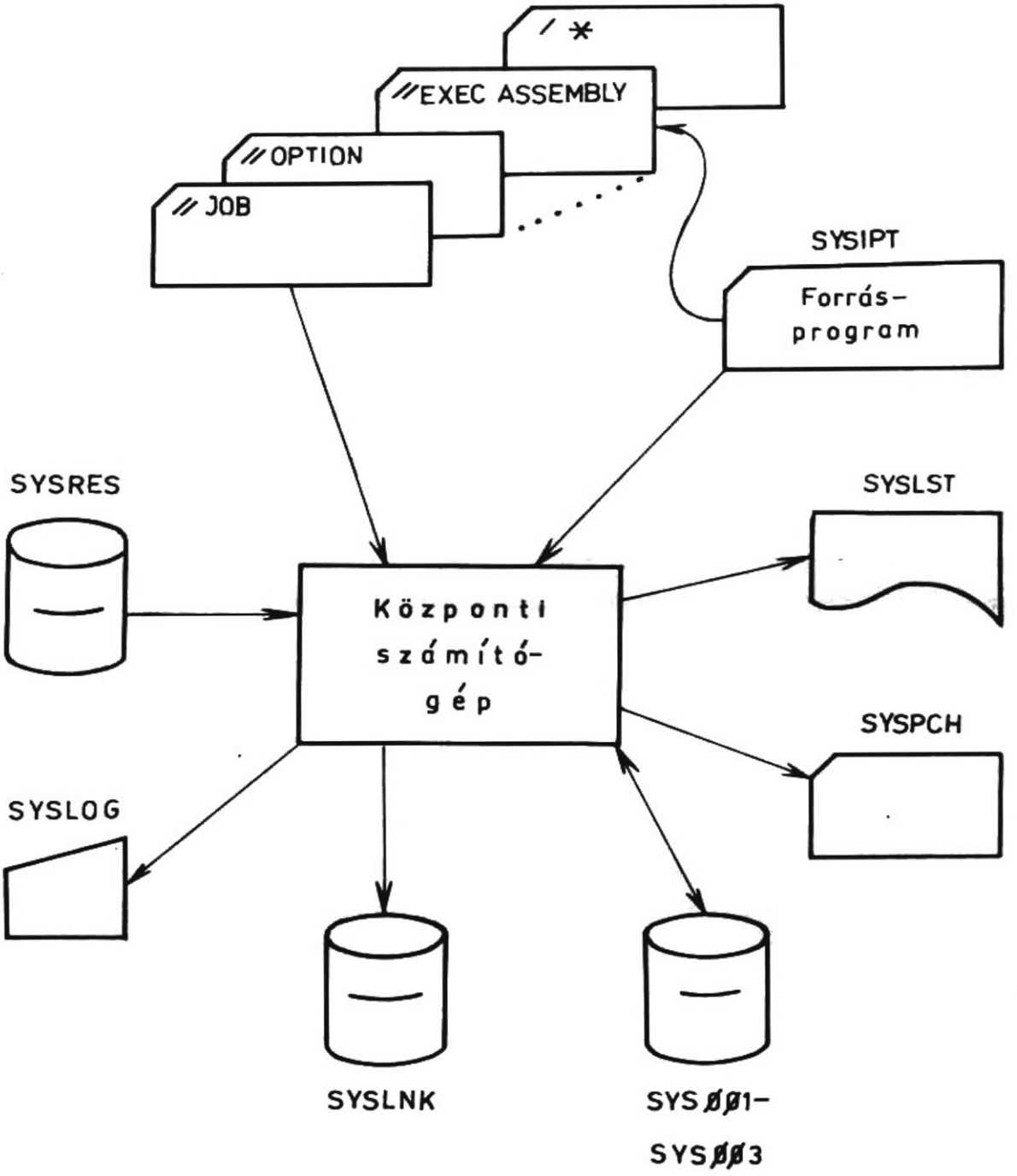
A forditóprogramokat müködésük és a hardware-hez való viszonyuk alapján két csoportra szokták osztani: a magasszintü programozási nyelvek forditóprogramjai a kompilátorok, a gépi kódhoz közelebb álló nyelvek forditóprogramjait pedig assemblereknek nevezik. A megkülönböztetés annyiban indokolt, hogy a kompilátorok rendszerint egy magasszintü nyelvi utasitást több gépi kódu utasitással valósitanak meg, mig az assemblerek egy assembler nyelvü utasitásból egy gépi utasitást generálnak. Mindez azzal függ össze, hogy a magasszintü nyelvek utasitásaiban sokkal több elemi lépést foglaltunk össze szimbolikusan, mint az assembler nyelvek utasitásaiban (például egészen hosszu, bonyolult aritmetikai kifejezések is "belesürithetők" egy magasszintü utasitásba). Világos, hogy az ilyen utasitások lebontása gépi szintre bonyolultabb gépi kódhoz vezet. Meg kell jegyeznünk, hogy az assembler nyelveknek is vannak olyan utasitásai (az ugynevezett makróutasitások), amelyek a fordités során több gépi utasitást generálnak. Ezek azonban lényegileg eltérnek a magasszintü utasitásoktól.

A korszerü általános célu számitógép operációs rendszerénél ma már követelmény, hogy legyen benne valamilyen assembler és egy vagy több kompilátor. A feladatok sokrétüsége ugyanis sokszor indokolttá teszi azt, hogy bizonyos programrészeket más-más programozási nyelven irjunk meg. Az általánosan elterjedt és használatos nyelveknek megfelelően például az ESZR/DOS operációs rendszerben FORTRAN, COBOL, PL/I és RPG nyelvek forditóprogramjai találhatók meg. Ezekkel a magasszintü nyelvekkel ugyszólván minden feladat megoldható. Ezenkivül támogatja az operációs rendszer az ASSEMBLER nyelvet is, tehát assembler forditóprogramja is van. Egyes operációs rendszerekben ALGOL forditóprogramot is találunk; itt csak a legismertebb nyelveket emlitve. A számitógépes folyamatirányitástól kezdve rendszerszimulációig számos speciális célu nyelv használatos. Nem lenne érdemes felsorolni a forditóprogramokat, ezek száma igen nagy! Ha szem előtt tartjuk a forditóprogram fő funkcióját, a szimbolikus és gépi kódu programozás közötti "szakadék" áthidalását, akkor minden forditóprogram müködését könnyen megérthetjük. Az egyes gépi reprezentációk sokfélesége az alapelveket nem változtatja meg.

Ha egy forditóprogram müködésének filozófiájába be akarunk tekinteni, mindenekelőtt a gépi nyelv szerkezetét kell felidéznünk emlékezetünkben. Vegyük példaként a kétcimes számitógép gépi nyelvét (lásd 3.4.7 pont). Itt minden egyes gépi utasitás egy müveleti kódot és két operanduscimet tartalmaz. Ezek lehetnek főtárcimek, de lehetnek regiszterek cimei is. A forditóprogram feladata tehát az, hogy a szimbolikus kódból, annak alapján "kitöltse" a generálandó gépi utasitások müveleti kód részét és operanduscimeit. Mivel bázisregiszteres cimzést használunk, a generált gépi utasitások olyan programot alkotnak, amely a tárban bárhol futtatható lenne, a báziscim értékétől függően. A forditóprogramok azonban csak olyan szintü gépi kódot állitanak elő, amely még nem futtatható közvetlenül. Annak oka, hogy az operációs rendszer egy ilyen közbenső szintü, "félkész" gépi kódot is használ, az, hogy igy nagyon rugalmassá teszi a programozást. A "félkész" gépi kódu programokból ugyanis többet tetszés szerint összeválogathatunk, összeszerkeszthetünk, mindig az adott feladat igénye szerint. Azt a programot, amelyből a forditóprogram ilyen "félkész" gépi kódot készit, forrásprogramnak nevezzük. Az elnevezés utal arra, hogy ez a program közvetlenül a felhasználótól származik, ez a feladatmegoldás kiindulási szintje. A forrásprogram tehát tisztán szimbolikus kód, magának a hivatkozási nyelvnek az utasitásaiból áll. Beszélhetünk tehát FORTRAN, COBOL, ASSEMBLER stb. forrásprogramokról. Ezek egy-egy feladat megoldását szolgáltatják szimbolikus, előre elképzelt módon. A forrásprogramok irásának szabályait az adott programozási nyelv szabályai, az ugynevezett szintaktikai szabályok rögzitik. Ezek jellegüket tekintve hasonlóak a közönséges nyelvtani szabályokhoz, és ha eltérünk tőlük,

akkor a forrásprogramunk hibás lesz. A fordítóprogramok inputja tehát egy-egy forrasprogram, outputja pedig "félkész" gépi kódu program, ugynevezett modulprogram. Valamennyi fordítóprogram azonos szerkezetü outputot szolgáltat, jóllehet inputjuk különböző!

Mivel a programfordítás általában összetett feladat, a fordítóprogramok is bonyolult müködésü, komplex programok. Gondoljuk csak meg: nem könnyü dolog egy teljesen szimbolikus, szöveges kódból egy vele egyenértékü, szigoruan kötött formáju gépi kódot készíteni! A fordítóprogramok ezért többször haladnak végig a forrásprogramon, mig a gépi kódot elő tudják állitani. Ezekre azt mondjuk, hogy többmenetes forditók. Egyszerübb programozási nyelveknek lehetnek egymenetes forditóik is, ezek csak egyszer "nézik át" a forrásprogramot. A többmenetes fordítóprogramok több önálló részre vannak osztva, és minden programrész, amikor végzett a saját fordítási feladatával, hivja az utána következőt. A legutolsó rész szolgáltatja azután a modulprogramot.



194. ábra. A programfordításhoz szükséges perifériák és munkaterületek

Tekintsük például az Assembler fordítóprogramot! Tudjuk, hogy az ASSEMBLER nyelv viszonylag közel áll a gépi kódhoz, de szimbolikusan fejezi ki a gépi kódu utasításokat. A fordítónak tehát az lesz a dolga, hogy a szimbolikus utasításokból bináris kódot készitsen. Ezt több menetben valósitja meg. Maga a fordítóprogram is több részből áll: az egyik a makrógenerátor, a másik a tényleges fordító. Először mindig a makrógenerátor fut le; ennek az a feladata, hogy a makróutasitasok helyére bemásolja a megfelelő makródefiniciókat. Itt persze nem egyszerűen másolásról van szó; a makródefiniciók beillesztésekor még sok szempontot figyelembe kell venni. A makrógenerátor futása után tehát olyan forrásprogram keletkezik, amelyben már nincs makróutasitás, csak szimbolikusan kifejezett gépi utasitás. Itt lép működésbe a fordítóprogram második része, amely a neki szóló, fordításvezérlő utasitások alapján "egy az egyben" átalakitja a szimbolikus kódot bináris gépi kóddá. Az utasitás mnemotechnikus kódjából müveleti kódot készit, a szimbolikus cimekből pedig bázisregiszterből és relativ cimből álló főtárcimeket vagy regisztercimeket generál. Nagyon vázlatosan leirva ez tehát a programfordítás menete. Hangsulyozzuk, hogy a valóságban ez sokkal bonyolultabb, mint ahogyan itt ismertettük. A fordítóprogramok viszonylag sok munkaterülettel dolgoznak, hiszen a fordítás közbenső eredményeit valahol tárolni kell. A munkaterületek lehetnek mágneslemezen vagy mágnesszalagon. A 194. ábra az ESZR/DOS Assembler fordítóprogramja által használt perifériákat és munkaterületeket mutatja.

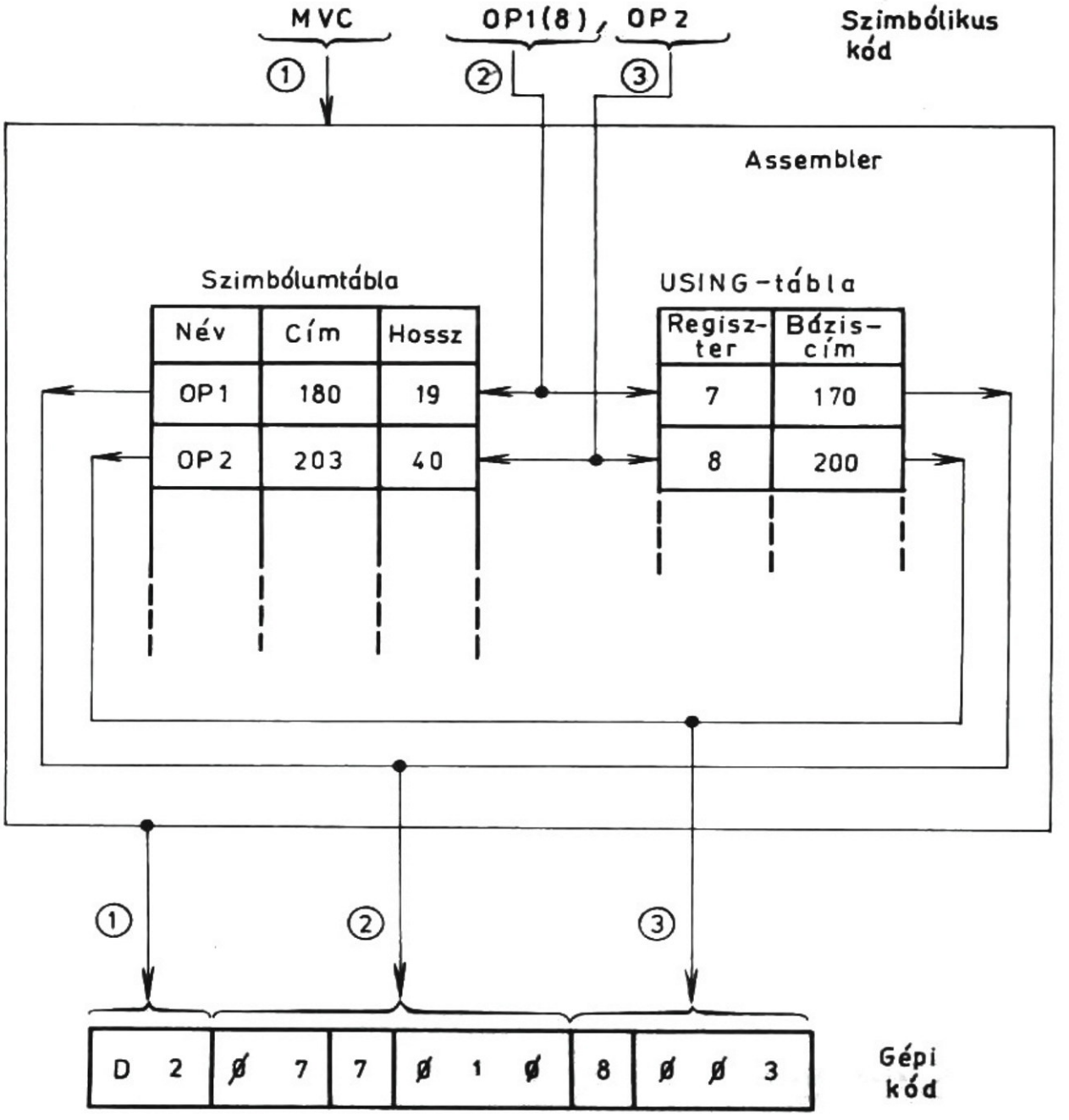
A fordítóprogram SYSIPT-ről olvassa be a forrásprogramot. SYSRES-ről kerül a tárba maga a fordítóprogram is, de innen származhatnak makródefiniciók, egyéb programrészek is. SYSLOG szolgál a fordítási események naplózására, egyben a gépkezelői beavatkozást is lehetővé teszi. SYSØØ1-SYSØØ3 lemezterületek a fordító munkaterületei. SYSLNK-ra kerül a leforditott program, ha szerkeszteni kivánjuk, SYSPCH-ra pedig akkor kerül, ha meg akarjuk őrizni például lyukkártyán. SYSLST-re adja ki a fordítóprogram a fordítási listát. Ezen egymás mellett szerepel a forrásprogram és a leforditott program. Egyben szintaktikai ellenőrzést is végez, és a forrásprogram hibás utasitásait a listán megjelöli. Ennek alapján javithatjuk a forrásprogramunkat.

Végül tekintsük át egyetlen utasitás leforditásának menetét! Legyen ez egy SS1 tipusu utasitás, amelynek feladata a főtár egyik területéről adott számu byte-ot átvinni egy másik területre. Az operandusok hossza azonos. Az utasitás leforditásának mozzanatait a 195. ábrán láthatjuk.

Az ábrán a leforditandó szimbolikus utasitást láthatjuk. Ennek mnemotechnikus kódja MVC, első operandusa (ahova a byte-okat vinni kell) OP1, a müveletben 8 byte-ot kell mozgatni. A második operandus (ahonnan a byte-okat vinni kell) OP2. OP1 és OP2 szimbolikus cimek a programban, a nekik megfelelő helyet külön utasitásokkal lefoglaltuk.

A fordításhoz az Assembler két táblázatot használ. A szimbólumtáblába gyűjti össze az összes szimbolikus cimet, amely a programban előfordul, ide került tehát OP1 és OP2 is. A szimbolikus cimek mellé irja a tényleges tárcimet; a példában tehát OP1 névvel neveztük el a $180_{(16)}$ értékü tárcimet stb. Ugyancsak ide kerül a szimbolikusan cimzett terület hossza, igy az OP1 névhez a $180_{(16)}$ cimtől kezdődően 19 byte tartozik stb. A USING-táblába gyűjti a fordítóprogram a használható bázisregisztereket és báziscimeket. Ezeket a felhasználó adja meg, a fordító pedig csak ezeket használhatja. A két táblázat felépitése után fog hozzá a tényleges fordításhoz.

(1) Az MVC karaktersorozat alapján a $D2_{(16)}$ müveleti kódot generálja, és beirja az utasitás első byte-jába.



195. ábra. Az Assembler forditóprogram működése

(2) Az első operandus leforditásához először a szimbólumtáblához fordul, és kikeresi az OP1 nevet. Feladata az, hogy a $180_{(16)}$ tárcimet felbontsa bázisregiszter/relativ cim alakra. E célból bázisregisztert keres a USING-táblában. Mivel a relativ cim (D_1) csak pozitiv lehet, $180_{(16)}$-nál kisebb báziscimet kell használnia. Megfelel tehát a 7-es regiszter a $170_{(16)}$ báziscimmel. B_1 és D_1 helyére tehát 7-et, illetve $Ø1Ø_{(16)}$-t fordit a gépi kódba. (A relativ cim értéke 16, ennek 12 bites hexadecimális megfelelője a $Ø1Ø_{(16)}$.) Az operandushossz 8 byte; mint tudjuk, a hosszmező ennél eggyel kisebb számot tartalmaz, innen a $Ø7_{(16)}$ hossz! Figyeljük meg, hogy a szimbólumtáblában szereplő hosszak egyike sem 8 byte; ezt a hosszt "rákényszeritettük" az operandusokra, teljesen szabályosan.

(3) A második operandus leforditása hasonlóan történik, de itt a 8-as regisztert használja bázisregiszterként, hogy kisebb relativ cim adódjon! Ha ugyanis ezt is a 7-es regiszterrel és $170_{(16)}$ báziscimmel forditotta volna, akkor a relativ cim $203_{(16)}$-$170_{(16)}$ lett volna: $93_{(16)}$.

Ez a folyamat ismétlődik tehát ciklikusan egészen addig, mig a forrásprogram végét el nem érjük. Az utolsó utasitás hatására, amely a forrásprogram végét jelzi, a forditás befejeződik. Ezután a fordító még elvégzi azokat a járulékos feladatokat, amelyeket "rábiztunk", igy például kinyomtatja a forditási listát SYSLST-re, szükség esetén egyéb információt is ad.

A példában az Assembler forditóprogram működését vázoltuk, mert ez a legkevésbé bonyolult. Ennek analógiájára most már képet formálhatunk arról, hogy a magasszintü nyelvek forditóprogramjai is végső soron igy dolgoznak, csupán a forrásprogrammal "lényegesen több dolguk lehet". A gépi nyelv azonban őket is köti a megismert formátumhoz.

5.4.2 PÉLDA PROGRAMFORDÍTÁSRA. A FORDÍTÁSI LISTA ÉRTELMEZÉSE

A 196. ábrán egy konkrét feladat megoldásának részletét láthatjuk forrásprogram szinten és a megfelelő gépi kódban.

A programot ASSEMBLER nyelven irtuk, és egész számok válogatását végzi különféle szempontok alapján. Az ábrán a teljes programnak csak egy kis részletét mutatjuk be, de ez érzékelteti a szimbolikus és gépi kódu programozás kapcsolatát.

A forditási lista 1-3 oszlopai gépi kódban vannak, a 4-5 oszlopok pedig szimbolikusak.

(1) Ebben az oszlopban azokat a tényleges tárcimeket láthatjuk hexadecimálisan, amelyekre a programforditás történt.

(2) Ez az oszlop tartalmazza a leforditott utasitásokat hexadecimálisan. Ez a keletkezett gépi kód.

① LOC	② OBJECT CODE	③ ADDR1	ADDR2	④ STMT		⑤ SOURCE STATEMENT	
000030	0530			24	KEZD	BALR	3,0
			00032	25		USING	*,3
000032	4180 3122		00154	26		LA	8,CIM+81
000036	4170 30D1		00103	27	E3	LA	7,CIM
00003A	D202 30C6 30C9	000F8	000FB	28	E0	MVC	N,N1
000040	FA20 30C6 3146	000F8	00178	29		AP	N,=P'1'
000046	FA20 30C9 3146	000FB	00178	30		AP	N1,=P'1'
00004C	F922 30C6 3147	000F8	00179	31		CP	N,=P'1999'
000052	4720 308E	000C0		32		BP	E1
000056	FD20 30C6 314A	000F8	0017C	33		DP	N,=P'3'
00005C	F900 30C8 314B	000FA	0017D	34		CP	N+2(1),=P'0'
000062	4780 304E	00080		35		BZ	E2
000066	D202 30C6 30C9	000F8	000FB	36		MVC	N,N1
00006C	FD20 30C6 314C	000F8	0017E	37		DP	N,=P'5'
000072	F900 30C8 314B	000FA	0017D	38		CP	N+2(1),=P'0'
000078	4780 304E	00080		39		BZ	E2
00007C	47F0 3008	0003A		40		B	E0
000080	D202 30C6 30C9	000F8	000FB	41	E2	MVC	N,N1
000086	FD20 30C6 314D	000F8	0017F	42		DP	N,=P'7'
00008C	F900 30C8 314B	000FA	0017D	43		CP	N+2(1),=P'0'
000092	4780 3008	0003A		44		BZ	E0
000096	D208 7000 312B	00000	0015D	45		MVC	0(9,7),MASK
00009C	DE08 7000 30C9	00000	000FB	46		ED	0(9,7),N1
0000A2	1578			47		CLR	7,8

196. ábra. A forditási lista értelmezése

(3) Ez az oszlop tulajdonképpen csak a lista olvasását segiti azzal, hogy az első és második operandus tényleges cimét mutatja. Könnyitést jelent annyiban, hogy nem kell minden tárcim ellenőrzésekor ujból kiszámitanunk a bázis/relativ cim alakból az operandus cimét.

(4) Ez az oszlop a forrásprogram utasitásait sorszámozza decimálisan. Katalogizált forrásprogramok javitásakor ennek alapján lehet hivatkozni az utasitásokra.

(5) Itt látható maga a forrásprogram olyan alakban, ahogyan a fordítóprogram beolvasta. Ha lyukkártyás input volt, akkor itt tulajdonképpen a kártyák láthatók kinyomtatva, minden változtatás nélkül. Elöl áll az utasitások neve (KEZD, E3 stb.), ebből készíti a fordítóprogram a szimbólumtáblát. Középen áll az utasitások mnemotechnikus kódja, jobbra pedig az operandusok cimei. Látjuk, hogy itt minden információ szöveges alaku.

Példaként elemezzük a forrásprogram 28. utasitását! Ez a már megismert karakterátviteli utasitás, és az N szimbolikus cimre viszi a byte-okat N1 cimről. A leforditott kódban ezt látjuk: D202 30C6 30C9. Az utasitás tipusa SS1, ennek alapján könnyü elemezni a gépi formátumot. D2 a müveleti kód, 02 az operandushossz, tehát 3 byte mozgatásáról van szó; 30C6 az első operandus cime, ebből 3 a bázisregiszter száma, $0C6_{(16)}$ a relativ cim; 30C9 a második operandus cime, szintén 3 a bázisregisztere, és $0C9_{(16)}$ a relativ cime. Ha most eláruljuk azt, hogy a báziscimet $32_{(16)}$-re állitottuk be, akkor az operandusok cimei az alábbiak:

1. operandus cime

$$\begin{array}{r} 32_{(16)} \\ +\ 0C6_{(16)} \\ \hline 0F8_{(16)} \end{array}$$

2. operandus cime

$$\begin{array}{r} 32_{(16)} \\ +\ 0C9_{(16)} \\ \hline 0FB_{(16)} \end{array}$$

Ha most megnézzük a (3) oszlopot a listán, ott látjuk ezeket az értékeket az ADDR1 és ADDR2 alatt. A tényleges cimeket tehát nem kell kiszámitanunk, mert a listán szerepelnek. Itt csak ellenőriztük azt, hogy helyesen számoltunk. Hasonlóan olvassuk egyéb fordítóprogramok listáit is, a sajátosságoknak megfelelően.

5.4.3 A SZERKESZTŐPROGRAM. A PROGRAMSZERKESZTÉS MENETE. A MODULPROGRAM FOGALMA, SZERKEZETE

Láttuk, hogy az operációs rendszer három különböző szintü programfajtát ismer, ezek: a forrásprogram, a modulprogram (más néven: áthelyezhető program) és a fázisprogram. A gépi nyelvtől legtávolabb álló forrásprogramokból az egyes fordítóprogramok készitenek azonos felépitésü modulprogramokat. Ezen modulprogramok összekapcsolása, összeállitása kész gépi nyelvü, tehát futtatható programmá - a szerkesztőprogram feladata.

Felmerül a kérdés, hogy miért kell a kész, gépi nyelvü programot "több helyről" összeszedni. Képzeljük el, hogy egy nagyobb méretü programot több programozó egymástól függetlenül ir meg. A feladatnak lehetnek különlegesen számitásigényes részei, mig más részek esetleg bonyolult adatren-

dezést, vagy bonyolult, sokszor ismétlődő (például input/output) műveleteket tartalmaznak. Ilyenkor nyilvánvaló, hogy a programozók szempontjából az a kedvező, ha a feladat egyes részeit azon a programozási nyelven irhatják meg, amelytől a legtöbb támogatást kapják. Támogatáson azt értjük itt, hogy az illető programozási nyelv szerkezete és a fordítóprogram szolgáltatásai "elébe mennek" a feladatnak, és ezzel biztositják a feladat minél egyszerübb kódolását. Igy például a számitásigényes programrészeket FORTRAN vagy PL/I nyelven célszerü irni, mert ezek a nyelvek a matematikai kifejezések kódolását nagyon egyszerüvé teszik, és ugyanakkor sok beépitett, előre megirt függvénykiszámitó algoritmust is rendelkezésre bocsátanak. A kevésbé számitásigényes, a hardware-hez közelebb álló programrészeket ASSEMBLER-ben célszerü irni, mert sokkal több lehetőségünk van itt a hardware kihasználására (magunk válogathatjuk meg a regisztereket, amelyeket használunk, mi döntjük el az operandusok méretét, helyét a programban stb.). Mivel minden programozási nyelv bizonyos mértékig feladatorientált, az összetett feladatok programozását igazán hatékonyan csak több, különböző nyelv használatával oldhatjuk meg.

Annak érdekében, hogy a különböző forrásnyelvről forditott modulprogramokat egymással össze tudjuk kapcsolni, az operációs rendszer valamennyi forditóprogramja azonos felépitésü modulprogramot szolgáltat. Ezeket dolgozza fel, kapcsolja össze a szerkesztőprogram egységes egésszé. A következőkben vázlatosan megismerjük a rendszer egységes modulprogramstrukturáját, majd a programszerkesztés lehetőségeit.

A forditóprogramok outputja, a modulprogram bináris formáju. Tartalmazza mindazt az információt, amire a szerkesztőprogramnak szüksége van a modulok összekapcsolásához, valamint tartalmazza a forrásnyelvi program bináris megfelelőjét, gépi nyelvü képét. A modulprogram önmagában általában nem futtatható, jóllehet a gépi utasitásokat a gép értelmezni tudná. Ennek az az oka, hogy egy adott modulprogramban majdnem mindig hivatkozások találhatók olyan program-elemekre (például szimbolikus cimekre), amelyek egy másik modulprogramban vannak. Ezek a cimhivatkozások akkor nyernek értelmet, ha a két modult összekapcsoljuk, egymás mellé tesszük a tárban, és az egymás közötti cimhivatkozásokat feloldjuk. A feloldás ugy történik, hogy a cimhivatkozás helyére az elérendő cim értékét irjuk be. Igy a modulok mintegy "összeolvadnak"; ha például 10 modult kapcsolunk össze egyetlen futtatható programmá, akkor mind a 10 modul egymásra hivatkozásainak el kell tünnie a szerkesztés során, minden egyes cimhivatkozásnak meg kell találnia a hozzátartozó cimet.

Mindezek alapján a modulprogramot ugy képzelhetjük el, mintha lyukkártyákon egymás után először fel lennének sorolva a külső cimhivatkozások, és utána következne a leforditott, gépi nyelvü program. Végül még egyéb információ következne arra nézve, hogy a gépi nyelvü program milyen kezelést igényel a szerkesztéskor. Valójában igy is van: a modulprogram elején találjuk az ugynevezett Külső szimbólumok szótárát (External Symbol Dictionary), ezt követi a gépi nyelvü rész, a program "szövege" (Text), majd végül a program apró módositásaira vonatkozó információ áll, ez az áthelyezendő szimbólumok listája (Relocation List Dictionary). A modulprogram végét egy végjelző kártya jelenti (END). Az egyes kártyák azonositása ugy történik, hogy a 2-4. oszlopba karakterformáju kártyatipust tesz a forditóprogram. Az alábbi kártyatipusok keletkeznek forditáskor:

ESD - External Symbol Dictionary

TXT - Text

RLD - Relocation List Dictionary

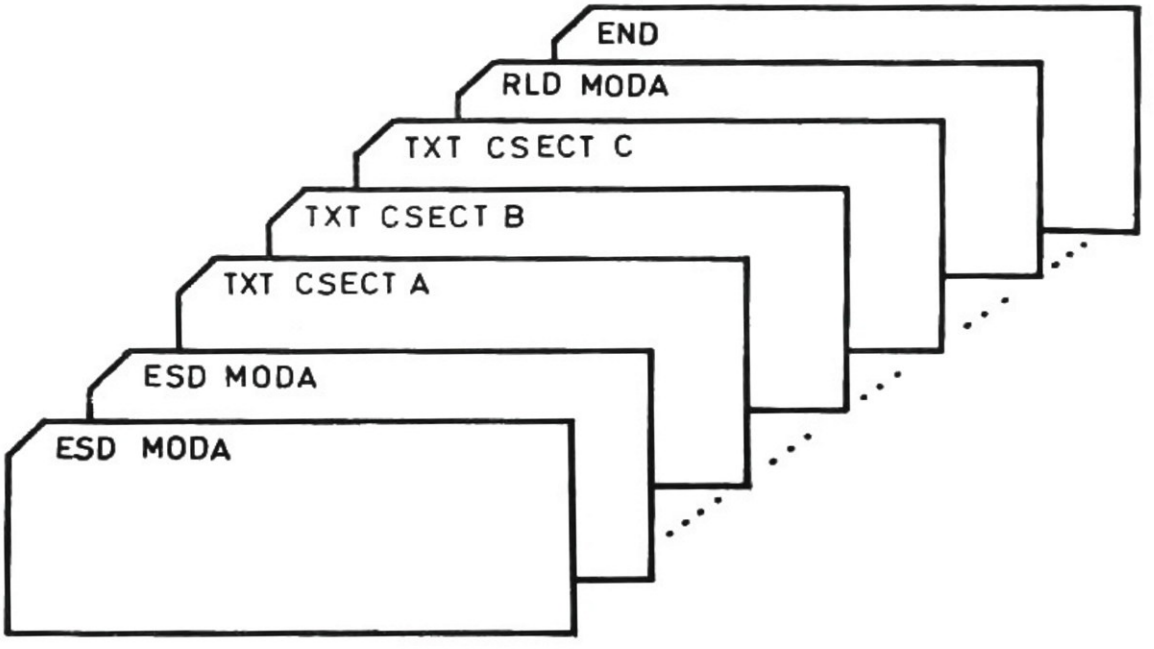
END - End.

Ha a modulprogramot módositani akarjuk, akkor a gépi nyelvü részbe javitókártákat helyezhetünk el. A módositás nem jelent szükségképpen hibás modulprogramot! A javitókártya tipusa:

REP - Replacement to Text (szöveghelyettesités).

A modulprogramok módositása általában körülményes dolog, hiszen közvetlenül gépi kódban kell dolgoznunk! Akkor lehet rá szükség, ha a teljes forrásprogram javitott ujraforditása nagyon gépidőigényes lenne.

Minden modulprogram szövege több, független gépiutasitás sort tartalmazhat. Ezek a gépi utasitások általában egy-egy kisebb programrészt valósitanak meg, afféle programszegmenseknek tekinthetők. Például, képzeljük el azt, hogy - gépi kódban gondolkodva - az első szegmens beolvassa az adatot, a második átkonvertálja decimálisból binárisba, a harmadik elvégzi a számitási müveleteket stb. Ezen szegmensek futtatásának csak együtt van értelme, bár mindannyian egy-egy sajátos feladatot oldanak meg. Ezeket a szegmenseket, minthogy a hardware vezérlését végző gépi utasitásokból állnak, vezérlő szekciónak (Control section, CSECT) nevezzük. Egy modulprogram tehát egy vagy több CSECT-ből áll. A 197. ábrán egy olyan modulprogram kártyáit láthatjuk, amely három CSECT-ből áll; ezek alkotják a modulprogram szövegét.



197. ábra. A modulprogram szerkezete

A szerkesztőprogram feladata az, hogy a modulprogramok CSECT-jeiből összeállitsa a kész gépi programot. A szerkesztésnél nem kötelező tehát az, hogy a modulprogram teljes szövege benne legyen a kész programban! A szerkesztőprogram "kiválogatja" azokat a CSECT-eket, amelyeket fel akarunk venni a futtatható gépi programba. A kiválogatás négy különböző helyről történhet a szerkesztőprogramnak szóló utasitások alapján. A szerkesztendő programrészek mágneslemezterületeken vagy egyéb perifériális egységeken lehetnek. Ezek:

1. SYSLNK - ide helyezik el a forditóprogramok a forrásprogramból készitett modulprogramot akkor, ha Job Control-nak a LINK opciót adtuk meg.

2. SYSIPT - ide teszi a felhasználó a modulprogramot, ha azt előzőleg kilyukasztatta vagy mágnesszalagra iratta a fordítóprogrammal.
3. SYSIPT és a modulkönyvtár.
4. A modulkönyvtár.

Mindössze négy utasitásfajtára van szükség ahhoz, hogy a modulprogramokból összeállittassuk a kész programot.

A fázisprogram kezdetét a PHASE utasitás jelöli. Ebben adjuk meg a szerkesztendő fázisprogram nevét és azt a főtárcimet, amelytől kezdődően a fázisprogramot majd futtatni akarjuk. Ez lesz a program betöltési cime, és System Loader ezt a fázisprogramot mindig ugyanerre a cimre fogja betölteni, valahányszor csak le akarjuk futtatni. Ez a cim nem feltétlenül a program első végrehajtható utasitásának a cime, hanem ez a program "eleje" (gondoljunk arra az esetre, amikor a program elején konstansok vannak; ezeket nem lehet végrehajtani!). Ha ez a cim zérus, akkor a program akármelyik cimre betölthető a tárba. Az ilyen programokat önáthelyező programoknak nevezzük. Nyilvánvalóan előnyös az, ha egy program önáthelyező, mert igy bármelyik tárterületen futtatható (kivéve természetesen a Supervisor particióját). Az önáthelyező programok irása viszont körülményesebb, és több programozói munkaráforditást igényel.

A modulprogramok, illetve CSECT-ek kiválogatását INCLUDE utasitásokkal adhatjuk meg (include = belevesz). Ez tehát a fázisprogramba belevenni kivánt CSECT-ek nevét tartalmazza. Ugyancsak ezekkel az utasitásokkal adhatjuk meg a szerkesztőprogramnak, hogy honnan származik az illető modulprogram. Az utasitás az alábbi alaku:

INCLUDE [modulnév] [, (CSECT-lista)]

Mind az első, mind a második operandus el is maradhat. Az operandusok elhagyásának hatását a szerkesztésre az alábbi táblázatban mutatjuk be.

1. operandus	2. operandus	modulprogram helye	CSECT-válogatás
hiányzik	hiányzik	SYSIPT	nincs
hiányzik	van	SYSIPT	van
van	hiányzik	modulkönyvtár	nincs
van	van	modulkönyvtár	van

Igy például az INCLUDE , (CSECT1, CSECT3) utasitás azt jelenti, hogy a SYSIPT-n található modulprogramból ki kell választani a CSECT1 és CSECT3 nevü szövegrészt, és csak ezeket kell belevenni a fázisprogramba. A többi operanduskombináció hasonlóan értelmezhető. Az utasitásban szögletes zárójellel jelöltük azt, ami elhagyható. Magában az utasitásban szögletes zárójelet sohasem irunk, ez csak a magyarázat célját szolgálja.

A később majd betöltendő fázisprogram első végrehajtandó utasitásának cimét (az ugynevezett belépési pontot a programba) az ENTRY utasitással adjuk meg. Ennek alakja:

ENTRY [az első végrehajtandó utasitás cime]

Ha az operandus elmarad, a szerkesztőprogram keresi meg - bonyolult módon - a program kezdőutasitását.

A szerkesztőprogram szolgáltatásait az ACTION utasitáson keresztül vehetjük igénybe. Igy például rendelkezhetünk afelől, hogy mi történjen a sikertelen szerkesztés esetén, kérünk-e szerkesztési listát a szerkesztőprogramtól stb.

Összefoglalva: a szerkesztőprogramnak szóló utasitások és funkciójuk a következők:

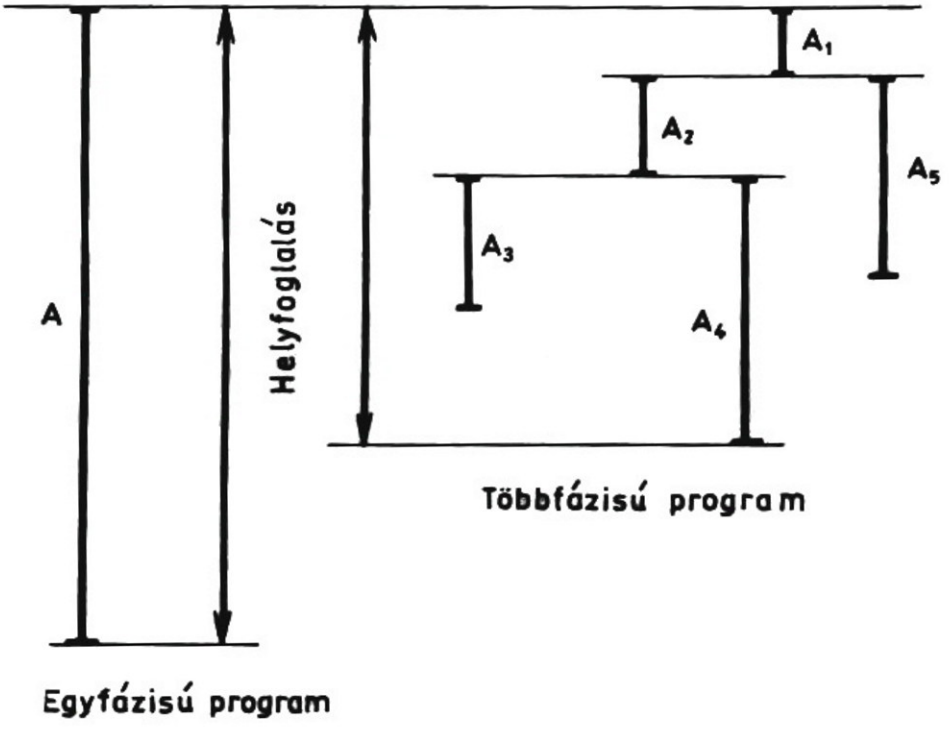
PHASE - fázisprogram kezdetét jelöli

INCLUDE - megadja, mi tartozzon bele a fázisba

ENTRY - megadja a program inditási cimét

ACTION - a szerkesztőprogram szolgáltatásait igényli.

A szerkesztőprogram lehetőséget ad arra is, hogy a hosszabb fázisprogramokat több fázisra tördelve szerkesszük. Ezt például ugy érhetjük el, hogy a PHASE utasitásban mindkét fázisnak azonos betöltési cimet adunk. Igy mindkét fázis mindig csak erre a tárcimre tölthető be, következésképpen egyszerre csak egyikük lehet bent a tárban. Az ilyen fázisok egymást "átfedik", részben vagy teljesen ugyanazt a főtárterületet foglalják el. Az ilyen fázisokat overlay-fázisoknak nevezzük. Programozási szempontból az overlay-fázisok használata az overlay-technika. Ennek előnye nyilvánvaló: a feladatot egyetlen óriási méretü program helyett több kisebb programban megoldva a feladat tárigénye jelentősen csökken. A több fázisu programokat multifázisu programoknak nevezzük. Ezeknél mindig van egy első (fő-) fázis, amely további fázisokat hivhat. Az első fázis lehet ugynevezett gyökérfázis (ROOT) vagy közönséges fázis. A gyökérfázisra jellemző, hogy az általa elfoglalt tárterületre egyik további fázis sem tölthető be. Általában egyik fázis a másikat hivja, azaz betölteti a tárba. Igy a feladatot megoldó programnak mindig csak az a része van a tárban, amely éppen fut. A 198. ábrán az A nevü egyfázisu program és a vele egyenértékü A1-A4 multifázisu program tárigényét szemléltetjük. Overlay-fázisként



198. ábra. A multifázisu program szerkezete

csak olyan funkciók programozhatók, amelyekre soha sincs "egyszerre" szükség. Az overlay-technika jellegét tekintve hasonló a Supervisor-tranziensekhez: egy adott főtárterületen időben egymásután egymást hivó, egymásra rátöltődő programfázisok futnak.

Annak eldöntésére, hogy a sok fázis közül melyek tartoznak egy-egy multifázisu programhoz, a fázisnév első 4 karakterét használják. Azok a fázisnevek, amelyeknek első 4 karaktere megegyezik, a multifázisu programhoz tartoznak. Például PROGØØØ1, PROGØØØ2, PROGØØØ3 egy három fázisból álló program fázisainak neve. Itt ugyanis mindhárom fázisnév a PROG karakterekkel kezdődik.

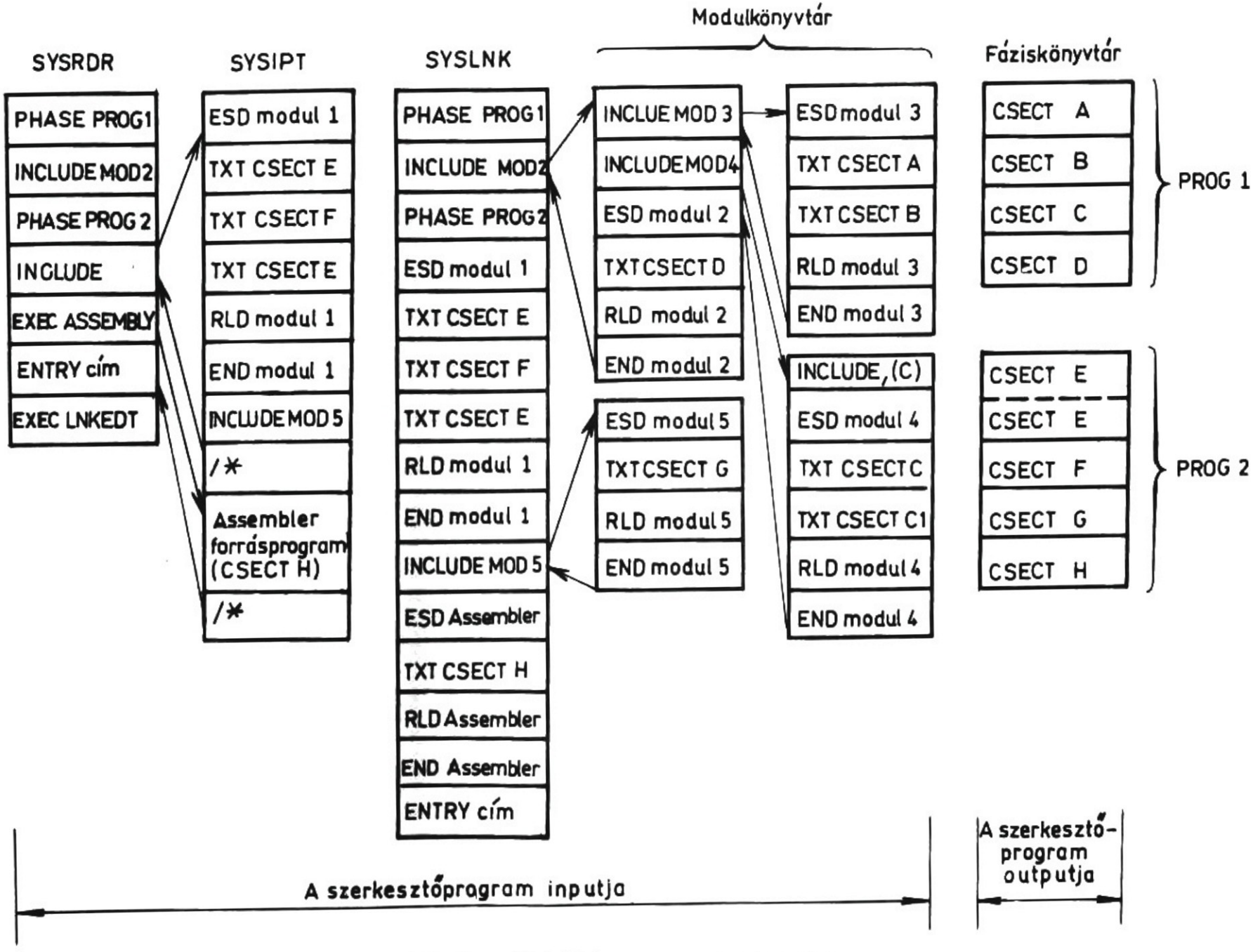
5.4.4 PÉLDA PROGRAMSZERKESZTÉSRE. A FÁZISPROGRAM FOGALMA, SZERKEZETE

A programszerkesztés menetét egy olyan példán mutatjuk be, amelyben több modulprogramból CSECT-válogatással és anélkül, több fázisprogramot állitunk elő. A szerkesztett két fázisprogram PROG1 és PROG2 nevü lesz, tehát egy multifázisu program két eleme. Kövessük végig a 199. ábrán, hogyan dolgozik a szerkesztőprogram, hogyan értelmezi a neki szóló utasitásokat!

A felhasználó számára közvetlenül hozzáférhető perifériák: SYSRDR és SYSIPT. Legyenek ezek kártyaolvasók, az egyszerüség kedvéért. SYSRDR perifériánkon összeállitottuk azt a kártyaköteget, amelyet Job Control-nak kell beolvasnia és végrehajtania. SYSIPT-re került a szerkesztendő program egyik modulja, valamint egy ASSEMBLER nyelven irt forrásprogram. Első lépésben Job Control "előkészíti" SYSLNK lemezterület tartalmát a szerkesztőprogramnak, vagyis sorban oda másolja a neki szóló kártyákat. Ez látható a PHASE PROG1, INCLUDE MOD2 és PHASE PROG2 kártyán: ezek tartalma bekerült SYSLNK-ra. A következő kártya SYSRDR-en olyan INCLUDE, amelynek hiányoznak az operandusai. Ez azt jelenti, hogy a következő modulprogram SYSIPT-n van, és teljes egészében be kell venni a készülő fázisprogramba. Az INCLUDE hatására tehát Job Control SYSIPT-n folytatja az olvasást, és a teljes 1. modult átmásolja SYSLNK-ra. A /* végjel visszamutat SYSRDR-re. Ezután az ASSEMBLY nevü forditóprogramot hivjuk, amely szintén SYSIPT-ről olvassa az ASSEMBLER nyelvü forrásprogramot, leforditja, és a keletkezett modulprogramot SYSLNK-ra teszi. Végül az ENTRY utasitás is SYSLNK-ra kerül, ez a szerkesztőnek szóló utolsó utasitás.

Figyeljük meg, hogy eddig a szerkesztőprogram még egyáltalán nem futott! Csupán Job Control és az Assembler fordító dolgozott azon, hogy előkészitse a modulprogramokat a szerkesztőprogram számára!

Utolsóként olvassa Job Control az EXEC LNKEDT utasitást. Ezzel hivjuk a szerkesztőprogramot, amelynek neve Linkage Editor (összekapcsoló-szerkesztő) és a rendszerben LNKEDT néven lehet elérni. Erre betöltődik a tárba a szerkesztő, és hozzáfog a fázisprogramok készitéséhez SYSLNK információtartalma alapján. Először azt látja, hogy PROG1 néven fázisprogramot kell készitenie, amelyben legyen benne a MOD2 nevő modulprogram. Fel kell tehát kutatnia a modulkönyvtárban a MOD2 nevü modulprogramot. Ezután PROG2 nevü második fázisprogramot kell készitenie; ennek szövege részben már SYSLNK-n van, részben a MOD5 nevü modulprogramban. Utóbbiért ismét a modulkönyvtárba kell mennie. A MOD2 és MOD5 programok előkeresésével a szerkesztés még nem ért véget, ugyanis a MOD2 nevü modul ujabb két modult hiv: MOD3-at és MOD4-et! Ezekért ismét a modulkönyvtárba kell mennie. Figyeljük meg, hogy az utolsó előtti oszlopban már nincs olyan INCLUDE, amelynek van első operandusa! Ez azt mutatja, hogy a szerkesztőprogram már valamennyi hivott modulprogramot "előkereste"



199. ábra. Multifázisu program szerkesztése

a modulkönyvtárból. Ezután hozzáfog a modulok összekapcsolásához: feloldja a külső hivatkozásokat a modulok között. A PROG1 nevü fázisba tartozik tehát a MOD3, részben a MOD4, és teljesen a MOD2 modul. MOD4-ből kiválasztottuk a C nevü CSECT-et, és csak ez került bele PROG1-be! A PROG2 fázisba került az 1. modul, a MOD5 nevü modul, és az Assemblertől származó modul szövege. A SYSLNK-n még két részletben levő, E CSECT két részét a szerkesztőprogram közvetlenül összekapcsolta, "összeigazitotta".

Látjuk, hogy a programszerkesztés nagyon rugalmas dolog; a fenti modulokból vagy azok részeiből később bármikor, bármilyen más összeállitásban ujabb fázisokat szerkesztethetünk. Ennek egyetlen feltétele az, hogy a modulok összekapcsolása "értelmes" legyen, vagyis a keletkező fázisprogram azt az algoritmust valósitsa meg, amelyet megkivánunk tőle. Ezen ismeretek birtokában most már akármilyen bonyolult szerkesztési feladatot is meg tudunk oldani, vagy meg tudunk fejteni.

Ha a fáziskönyvtár tartalmát szemügyre vesszük, szembetünő, hogy valamennyi ESD, RLD stb. rekord eltünt a szerkesztés során! A szerkesztőprogram tehát feloldotta a CSECT-ek közötti összes kölcsönös hivatkozást, és elvégezte a kért apró módositásokat a szövegben. Ha most feltételezzük, hogy az ENTRY-beli cim az A nevü CSECT-be mutat, akkor a kétfázisu program itt kezd futni, és PROG1 hivhatja PROG2-t maga után.

5.4.5 PROGRAMKÖNYVTÁRAK. A KÖNYVTÁRKEZELŐ PROGRAMOK FUNKCIÓI

Az operációs rendszer programjai, valamint egyes felhasználói programok kijelölt lemezterületekre gyüjthetők össze, amelyeket programkönyvtáraknak nevezünk. A programkönyvtárak elemei programok; minden itt tárolt programnak 8 karakterből álló neve van, amely alapján ki lehet keresni a könyvtárból. A keresés a könyvtár tartalomjegyzékének átvizsgálása utján történik. A tartalomjegyzék a könyvtár elején található, és meghatározott számu sávot foglal el a lemezen. Minden a könyvtárban tárolt programnak van egy bejegyzése a tartalomjegyzékben; ez tartalmazza az illető program nevét és kezdőcimét a könyvtárban. Ha tehát a keresett programnévhez megtaláltuk a tartalomjegyzék-bejegyzést, egyszersmind tudjuk a program helyét is a lemezen, le tudjuk onnan olvasni.

Az operációs rendszerben - a programok készültségi fokához igazodóan - három különböző szintü könyvtárat lehet használni. Ezek az alábbiak:

1. Fáziskönyvtár (Core image library)
2. Modulkönyvtár (Relocatable library)
3. Forráskönyvtár (Source statement library)

A könyvtárak közül csak a fáziskönyvtár használata kötelező, a többi könyvtár nélkül az operációs rendszer még müködik. A programozási nyelvek és a bonyolult input/output rendszer szervezése azonban gyakorlatilag indokolttá teszi mindhárom könyvtár kijelölését.

A fáziskönyvtár elemei azok az egyfázisu vagy multifázisu programok, amelyeket a szerkesztőprogram készitett a már megismert módon. Ezek a programok mind a Supervisor vezérlete alatt futhatnak. Minden fázisnak rögzitett betöltési cime van, ez még szerkesztési időben eldől (a PHASE utasitás alapján). A fáziskönyvtárban vannak mindazok a programok, amelyek közvetlenül futtathatók, igy például az operációs rendszer egy része, a forditóprogramok, egyes felhasználói programok stb.

A megőrizni kivánt fázisprogramokat a könyvtárba katalogizálhatjuk. Ez azt jelenti, hogy a program a könyvtár első szabad helyére kerül, és egyuttal egy megfelelő tartalomjegyék-bejegyzést is kap. Ezután a program név alapján bármikor betölthető és futtatható. A katalogizálást mindig a szerkesztőprogram végzi el, ha Job Control-nak CATAL opciót adtunk meg. Felhivjuk a figyelmet arra, hogy itt már közvetlenül futtatható programok vannak, ezért nem engedhető meg az, hogy más helyről is lehessen ide katalogizálni. A szerkesztőprogram ugyanis csak futóképes, a rendszer épségét nem veszélyeztető programokat katalogizál! A szerkesztőprogram "vigyáz" a fáziskönyvtár tartalmára.

Elképzelhető az is, hogy a fázisprogramot nem kivánjuk megőrizni, hanem csak egyszer futtatjuk, ez a helyzet például programok kipróbálásakor stb. Ilyenkor a szerkesztés után a fázisprogramot azonnal futtatjuk, nem katalogizáljuk a fáziskönyvtárba. A szerkesztőprogram ilyenkor is a fáziskönyvtárba teszi a szerkesztés eredményét, de annak egy "egyezményes" helyére. Itt az illető program addig tartózkodik, mig a következő - hasonló sorsu - program felül nem irja. Ez a terület afféle munkaterület jellegü, programok ideiglenes tárolására szolgál. Az ide töltött program név alapján nem érhető el, hiszen nincs is tartalomjegyzék-bejegyzése. Ez történik akkor, ha a Job Control-nak LINK opciót adtunk meg.

A modulkönyvtár elemei azok a modulprogramok, amelyeket különböző forditóprogramok állitottak elő, és őket ide katalogizáltuk. Mivel ezek közvetlenül nem futtathatók, nincs szükség olyan jellegü védelemre, mint a fáziskönyvtár esetében, ezért ide a könyvtárkarbantartó programmal tudunk katalogizálni. A modulprogramok formátumát már ismerjük: ESD, TXT, RLD stb. rekordokból állnak. A modulprogram itt természetesen nem lyukkártyán van, hanem a könyvtár számára kijelölt lemezterületen. Annak azonban nincs akadálya, hogy a megfelelő könyvtári szolgáltató program segitségével átmásoljuk lyukkártyára. Az információtartalom semmit sem változik.

A modulkönyvtár használatát az indokolja, hogy az egyszer már leforditott, félig kész, nagyon rugalmasan szerkeszthető programok benne legyenek a rendszerben. Ha ezeket minden szerkesztés előtt kártyakötegek formájában kellene összeállitanunk, esetleg ujraforditattnunk, akkor a feldolgozás nagyon nehézkes lenne. A modulkönyvtárban vannak többek között a logikai input/output rendszer általános és konkrét perifériákra vonatkozó file-kezelő moduljai. Futtatás előtt ezeket hozzá kell szerkeszteni a végső program egyéb moduljaihoz. A modulkönyvtár elemeit tehát a szerkesztőprogramnak futás előtt még fel kell dolgoznia.

A modulkönyvtárnak is hasonló tartalomjegyzéke van, mint a fáziskönyvtárnak. A tartalomjegyzék minden egyes katalogizált modul részére 1-1 bejegyzést tartalmaz. A bejegyzésben szerepel a modul neve és lemezcime. A modulok megkeresése tehát a tartalomjegyzék átvizsgálásával történhet. Ezt a módszert használja a szerkesztőprogram is akkor, amikor katalogizált modulprogramot kell felvennie a leendő fázisprogramba.

A forráskönyvtár elemei azok a programok, amelyeket valamilyen programozási nyelven irtunk és oda katalogizáltunk. A forrásprogramokra "még inkább igaz" az, hogy közvetlenül nem futtathatók, hiszen le sincsenek forditva gépi nyelvre! Ezek tehát szimbolikus nyelvü programok, és közvetlenül azokból az utasitásokból állnak, amelyeket a programozó az illető programozási nyelv szabályai szerint leirt. Katalogizáláskor a forrásnyelvi utasitások sürithetők; ez azt jelenti, hogy a szóköz-karaktereket kihagyjuk belőlük. Ezzel sok helyet takarithatunk meg a könyvtárban, és nem tárolunk ott fölöslegesen sok szóközt.

A forráskönyvtár használatát leginkább a makrókönyvtár kiterjesztésének lehetősége indokolja. Itt található az a sok makródefinició, amelyet a forrásprogramba megfelelő makróutasitással "behivhatunk". Ezeket tehát nem kell minden esetben ujra kódolni. A makródefiniciók használatának és az áthelyezhető modulprogramok használatának végső soron ugyanaz az alapgondolata, bár a konkrét feldolgozásuk lényegesen eltér egymástól! Tudjuk, hogy a modulprogramokat nem kell ujrafordittatni, ezzel szemben a hivott makródefiniciókat minden esetben fordittatni kell, csak kódolni nem! Mégis nagy előnye a makróknak az, hogy forrásnyelvi, tehát szimbolikus programozási szinten rendkivül rugalmasan alkalmazhatók.

A forráskönyvtárban természetesen nemcsak makródefiniciók találhatók, hanem bármely más nyelven irt programok is. Ennek megfelelően a forráskönyvtárt több alkönyvtárra osztják ugy, hogy minden programozási nyelvnek saját alkönyvtára van. Igy például az összes COBOL nyelvü programot a COBOL alkönyvtárba, PL/I programokat a PL/I alkönyvtárba stb. katalogizálják. A különböző programozási nyelveken irt programok tehát nem keverednek egymással.

A forráskönyvtárnak is megvan a maga tartalomjegyzéke. Ebben minden egyes katalogizált programhoz 1-1 bejegyzés tartozik, amely a program nevét és lemezcimét tartalmazza. A programok előkeresése ugy történik, hogy a keresett név ismeretében előbb a tartalomjegyzéket vizsgáljuk át, majd a megfelelő bejegyzésből megtudjuk, hol található a keresett program a lemezen.

A forráskönyvtár használata nagyban megkönnyitheti az egyes programok javitását is. Ha ugyanis forrásprogramunkat ide katalogizáltuk, többé nem kell kártyacsomagokkal közlekednünk a gépteremben: elég azt a néhány kártyát összeállitanunk, amely a könyvtárkarbantartó programot aktivizálja programjavitásra. A törölni kivánt, vagy beszurandó stb. forrásnyelvi utasitásokat könnyen elérhetjük, illetve bevihetjük a forrásprogramba, és egy-egy javitás eredményéről listát is kapunk. A listán szerepel egy olyan nyilvántartási szám is, amely a javitásokat sorszámozza. Igy mindig tudjuk, hogy az előző javitásnak mi lett az eredménye, és az aktuális javitást ehhez igazodóan végezhetjük el. Felhivjuk a figyelmet arra, hogy a forrásprogram javitását nem csak a programhiba teszi indokolttá; gondolunk arra, hogy például módositani akarjuk a forrásprogramot valamilyen aktuális szempont szerint stb.

Az eddig leirt könyvtárak voltak az ugynevezett rendszerkönyvtárak; ezek a SYSRES mágneslemezen találhatók. Az operációs rendszerben ezenkivül használhatók még ugynevezett privátkönyvtárak is. Ezek használatát az indokolhatja, hogy az illető könyvtárt nem SYSRES-en, hanem egy másik mágneslemezen kivánjuk elhelyezni. A könyvtárak létrehozását, illetve másolását a könyvtárkezelő programok végzik el. Elvileg tetszőlegesen sok privátkönyvtárt hozhatunk létre, de particiónként csak 1-1 privát fáziskönyvtárt, modulkönyvtárt és forráskönyvtárt tarthatunk fenn. Ha privátkönyvtárakat is használunk, a programok keresése a privátkönyvtárban kezdődik. Ha itt nem található a keresett program, akkor a keresés a rendszerkönyvtárakban folytatódik.

A privátkönyvtárak létrehozásakor a felhasználó dönti el, hogy hány cilindert akar a könyvtár számára fenntartani, és ebből hány sávot foglaljon el a könyvtár tartalomjegyzéke. A tartalomjegyzék a rendszerkönyvtárakhoz hasonlóan itt is a könyvtár elején van.

A könyvtárak telitettségét a könyvtárkarbantartó programok állandóan figyelik. Ha a könyvtár megtelt, vagy közel áll ahhoz, akkor a gépkezelő üzenetet kap a konzolirógépen a könyvtár állapotáról. Ennek alapján dönthet, hogy megnöveli-e a könyvtár területét, vagy más módon (törléssel, süritéssel) szabadit fel helyet a katalogizálni kivánt program számára.

A könyvtárkezelő programok funkciói, szolgáltatásai három fő csoportba sorolhatók:

1. Könyvtárkarbantartás
2. Szerviz-funkciók
3. Másolás.

Ezeket a főbb szolgáltatásokat külön-külön könyvtárkezelő programok látják el. A könyvtárakhoz forduláskor tehát valamilyen könyvtárkezelő programnak a tárban kell lennie, mert csak ennek futása során lehet hozzáférni a könyvtárakhoz. Az egyes szolgáltatásokat ugy lehet igénybe venni, hogy a megfelelő könyvtárkezelő programot EXEC utasitással betöltjük az operativ tárba, majd ennek a programnak adunk utasitásokat. A könyvtárkezelő programok mind SYSIPT-ről olvasnak. Tekintsük át most vázlatosan, hogyan dolgoznak a könyvtárkezelők, és milyen szolgáltatásokat nyujtanak a felhasználónak!

1. Könyvtárkarbantartás

A könyvtárkarbantartást a MAINT nevü program végzi (neve a maintenance = fenntartás szóból származik). Ezt a programot használhatjuk katalogizálásra, programok törlésére, programok nevének megváltoztatására, könyvtársüritésre, könyvtárak határainak megváltoztatására és forrásprogram javitására. Igen lényeges dolog az, hogy a MAINT programmal nem lehet a fáziskönyvtárba katalogizálni! Ez ugyanis a szerkesztőprogram kizárólagos joga és feladata. A többi könyvtárba MAINT-tel katalogizálunk.

a) Katalogizálás

Katalogizálás során a megőrzendő forrás- vagy modulprogramot MAINT felviszi a megfelelő könyvtár első szabad területére, majd összeállitja a hozzá tartozó tartalomjegyzék-bejegyzést. Ha a MAINT programot beolvastuk a tárba, akkor tetszőlegesen sok programot katalogizáltathatunk vele.

b) Programok törlése

Programok törlésén azt értjük, hogy az illető programhoz tartozó tartalomjegyzék-bejegyzést töröljük. Maga a program változatlanul megmarad a könyvtárban! Mivel azonban a könyvtárbeli keresés a tartalomjegyzék átvizsgálásával történik, az ilyen program a rendszer számára többé "nem létezik". A program törlése - vagyis az általa elfoglalt hely felszabaditása - könyvtársüritéssel történik.

c) Programok nevének megváltoztatása

Ez a funkció is a könyvtár tartalomjegyzékét érinti. Az ujra elnevezendő programhoz tartozó bejegyzésbe az uj név kerül; a lemezcim változatlan marad. A program a régi nevén tehát többé nem érhető el.

d) Könyvtársürités

Segitségével fizikailag is törölhetjük azokat a programokat, amelyek bejegyzéseit már korábban töröltük a tartalomjegyzékből. Sürités kor MAINT kiemeli a "nem létező" programokat, és helyükre a többit összetolja ugy, hogy összefüggő lemezterületet foglaljanak el. Igy a könyvtár végén szabad hely

keletkezik ujabb katalogizálások számára. Természetesen a megmaradó programok bejegyzésében a lemezcimet MAINT átirja, hiszen e programok más helyre kerülhettek a lemezen. A sűrítés sok input/output-tal jár, bonyolult művelet.

e) Könyvtárak határainak megváltoztatása

Ezt a szolgáltatást akkor igényeljük, ha a könyvtár területét meg akarjuk változtatni. Ide tartozik a könyvtár növelése, csökkentése, és a tartalomjegyzék méretének megváltoztatása. Az uj határok kijelölésével egyidejűleg automatikus könyvtársűrítés is történik.

f) Forrásprogramok javitása

A MAINT programot használjuk a forrásprogramok módositására, javitására is. Segitségével egy vagy több forrásnyelvi utasitást módosithatunk, törölhetünk, beszurhatunk bármelyik itt katalogizált programba. A módositás lehetővé teszi a forrásutasitások sorszámának megváltoztatását is. Emellett MAINT nyilvántartja az egyes módositott programváltozatokat. Ha egy forrásprogramot módositunk, akkor az uj változat kerül a régi helyére a könyvtárban. A módositás eredményéről listát kapunk; ez tartalmazza a régi programrészt és a módositott részt egyaránt.

2. Szerviz-funkciók

A szerviz-funkciók arra szolgálnak, hogy a különböző könyvtárak tartalmát megjelenitsük, vagyis láthatóvá tegyük. Lehetőség van a programok kinyomtatására, kilyukasztására, illetve e kettő összevont alkalmazására. A programnyomtatás SYSLST-re, a programok kilyukasztása pedig SYSPCH-ra történik. Természetesen ha például SYSPCH-hoz mágnesszalag egységet rendeltünk hozzá, akkor lyukasztás helyett mágnesszalagos tárolás történik. A fáziskönyvtár szervizfunkcióit a CSERV, a modulkönyvtárét az RSERV, a forráskönyvtárét pedig az SSERV nevű program látja el. Mielőtt tehát egy szervizfunkciót igénybe vehetnénk, a megfelelő szervizprogramot be kell töltenünk a tárba.

Külön program szolgál a tartalomjegyzékek megjelenitésére, ez a DSERV. A tartalomjegyzékben szereplő bejegyzéseket vagy előfordulásuk sorrendjében, vagy betűrendben nyomtatja. Mivel egyes könyvtárakban adott esetben igen sok program lehet, a betűrendes nyomtatásnak nagy jelentősége van, a tartalomjegyzék átvizsgálást könnyiti.

3. Másolás

A másolási funkciót használhatjuk arra, hogy a rendszerlemez tartalmát részben vagy egészben másik lemezcsomagra másoljuk át. Ilyen "háttér" lemezcsomagra azért lehet szükség, mert igy az eredeti károsodása esetén a rendszer tovább működhet. A szelektiv másolást olyankor használjuk, ha a kész operációs rendszerből egy "kevesebbet tudó", esetleg speciális célu operációs rendszert akarunk készitení. Ugyancsak a másoláshoz tartozik a privátkönyvtárak létrehozása. A másolóprogram egyaránt alkalmas a forráskönyvtár, modulkönyvtár és fáziskönyvtár másolására. Az egyes könyvtárak tartalmát részben vagy egészen "összefésülhetjük", a könyvtári programokat összeválogathatjuk anélkül, hogy kártyakötegekkel kellene dolgoznunk, illetve ujból katalogizálnánk. A másoló program tehát sok olyan feladatot automatikusan elvégez, amelyek különben jelentős gépkezelői munkaráfordítást

igényelnének. Hasznos a másolás az operációs rendszer generálásakor is; a rendszer egyik verziójából az átvenni kivánt könyvtári rutinokat változatlanul átmásolhatjuk az uj rendszerbe.

A könyvtárkezelő másolóprogramja a CORGZ. Csak a BG particióban futtatható. A másolási funkciók igénybe vétele előtt tehát a CORGZ nevü programot be kell töltenünk a tárba, majd tetszőlegesen sok utasitást adhatunk neki. Ezek között lehet privátkönyvtárak létrehozása, rendszerlemez másolása, összefésülés stb.

Összefoglalva: az operációs rendszer könyvtárai arra szolgálnak, hogy adott készültségi foku programokat a rendszeren "belül" tároljanak. A könyvtárak karbantartását, megjelenitését, másolását különálló programok végzik, ezek összessége a "könyvtáros".

5.4.6 PÉLDA KATALOGIZÁLÁSRA A FÁZISKÖNYVTÁRBA

Egy példát mutatunk arra, hogy milyen kártyákra van szükség egy fázisprogram katalogizálásához. Itt egyetlen fázisprogramot készitünk, ez azonban több modulprogram összekapcsolásával jön létre. A katalogizálást a szerkesztőprogram végzi.

```
// JOB KATALOG
// ASSGN SYSLNK,X'190'
// OPTION CATAL
   PHASE PROGØØØ1,S
   INCLUDE
```

Ide kerülnek a szerkesztendő modulprogramot tartalmazó kártyák

```
/*
 INCLUDE SUBR1
 INCLUDE SUBR2
 INCLUDE
```

Ide kerülnek a soron következő modulprogramot tartalmazó kártyák

```
/*
// EXEC LNKEDT
/&
```

Nézzük meg, mi a szerepe az egyes kártyáknak ebben a feladatban!

Az első kártya a job kezdetét jelzi. Ez Job Control-nak szól, és azt jelenti, hogy az automatizált feldolgozás számára egy uj feladat következik. Ezt több szempontból, például gépidőelszámolás miatt is egységesen kell kezelni. A job neve KATALOG, ezen tartja nyilván a rendszer egészen addig, mig véget nem ér.

Az ASSGN kártyával kijelöltük a SYSLNK logikai perifériát. Mint tudjuk, SYSLNK mágneslemezterület, és innen kezdi a szerkesztőprogram a fázisprogramkészitést. Esetünkben a 190 cimü mágneslemez egységen kell lennie, erről a feladat elinditása előtt a gépkezelő gondoskodik. Mivel a gyakorlatban SYSLNK mindig ki van jelölve, ez a kártya többnyire el is maradhat.

A harmadik kártyán közöljük Job Control-lal, hogy milyen szolgáltatását választjuk. CATAL itt azt jelenti, hogy a szerkesztendő fázisprogramot katalogizáltatni fogjuk a szerkesztőprogrammal.

Ezután következik a szerkesztendő fázisprogram szövege, illetve a hivott modulprogramok felsorolása. Esetünkben a fázisprogram neve PROGØØØ1 lesz; S pedig a betöltési cim kiszámitási módjára utal. A fázisprogramban négy modulprogram szerepel; az első és negyedik közvetlenül a kártyaolvasón található, mig a középsőkért a szerkesztőprogramnak a modulkönyvtárba kell "mennie". A modulprogramok a felsorolás sorrendjében kerülnek be a fázisprogramba.

A /* kártyák 1-1 kártyafile végét jelzik. Természetesen ezek, mint speciális végjelek, nem kerülnek be a fázisprogramba!

Az egyetlen job step kezdetét jelzi a // EXEC kártya. Ez szintén Job Control-nak szól, és arra utasitja, hogy töltesse be System Loader-rel a tárba a szerkesztőprogramot. Tudjuk, hogy ennek neve Linkage Editor és a rendszerben LNKEDT néven lehet elérni. Most már a szerkesztőprogram kerül be a tárba felülirva Job Control-t, és elvégzi az előirt szerkesztést. Végül a kész fázisprogramot elhelyezi a fáziskönyvtár első szabad helyére, és összeállitja a hozzá tartozó tartalomjegyzék-bejegyzést. Ha a későbbiekben ezt a programot le akarjuk futtatni, akkor // EXEC PROGØØØ1 utasitással tölthetjük be végrehajtás céljából.

A /& kártya szintén speciális: a job végét jelöli. Ezzel ért véget egy feladat a rendszer számára. Figyeljük meg, hogy Job Control-t leszámitva egyetlenegy program futott a job-ban, ez pedig a szerkesztőprogram volt. Ez a job tehát egyetlen step-ből áll. Ha a szerkesztett programot nem akartuk volna katalogizáltatni, akkor CATAL helyett LINK opciót adtunk volna meg, és a program futtatását inditó EXEC utasitásnak nem lett volna operandusa.

5.5 AZ OPERÁCIÓS RENDSZER EGYÉB SZOLGÁLTATÁSAI

5.5.1 AUTOTEST. PROGRAMOK BELÖVÉSE

Az elkészült programok kipróbálása, tesztelése általában bonyolult, nehézkes, de ugyanakkor rendkivül fontos feladat. Ilyenkor derül fény az esetleges algoritmushibákra, a program szemantikai hibáira. A program akkor hibás szemantikusan, ha "nem azt csinálja, amire irták". Képzeljük el, hogy rengeteg számot kell összeadnunk. Ezt a müveletet beépitjük egy programciklusba. Programozási hiba folytán ez a ciklus eggyel kevesebbszer fut le, mint kellene. Az ilyen tipusu, rejtett programhibák felderitése a tesztelés feladata. Ennek a tesztelésnek megkönnyitésére szolgál az AUTOTEST program. Ez az alábbi szolgáltatásokat nyujtja:

1. Tesztelési funkciók kiváltása

Alkalmas vezérlő utasitásokkal megjelölhetjük a vizsgált programnak azokat az utasitásait, amelyek végrehajtásakor tesztelési müveletet akarunk kezdeni. A tesztelési pontokat leszámitva a vizsgált program az AUTOTEST-től teljesen függetlenül fut, egymást nem zavarják. A tesztelési funkciók számát korlátozni is lehet. Az igényelhető tesztelési funkciók: főtár tartalmának kinyomtatása, regiszterek és állandóan felosztott főtárterület tartalmának kinyomtatása.

2. Automatikus programjavitás

Lényege az, hogy a program ujraforditása nélkül lehetőségünk van a program egyes utasitásait módositani, ujabb utasitásokat beszurni, illetve a program számára uj konstansokat megadni.

3. Dump

Akkor kerül sorra, ha a vizsgált program normálisan vagy hiba miatt véget ér. Ilyenkor a teljes programterület képe megjelenik nyomtatásban. Ha a program normálisan ért véget, akkor a dump-ot le lehet tiltani.

4. Fázisprogram-lista nyomtatása

AUTOTEST automatikusan kinyomtatja a tárba betöltött programfázisokat abban a sorrendben, ahogy egymás után futottak.

5. Végtelen ciklus elleni védelem

Ha a vizsgált program valamilyen hiba folytán végtelen ciklusba kerül, akkor lehetőségünk van a tesztelés befejezésére.

Mint látjuk, ezek a funkciók igen hasznosak főleg nagyobb, bonyolultabb programok belövésénél; igen nagy előnyt jelent, hogy ezeket nem kell minden esetben ujra programoznunk. Elég csupán az AUTOTEST vezérlő utasitásaival megismerkednünk.

5.5.2 RENDEZŐPROGRAMOK

A rendező/válogató program segitségével a nem rendezett adatállományból rendezett adatállományt hozhatunk létre, illetve több rendezett állományt egy rendezett állományba vonhatunk össze. Ez a rendező/válogató program a Supervisor felügyelete alatt futtatható. A program feltételezi azt, hogy az adatállomány nem rendezett. Amennyiben egyes részei mégis rendezettek lennének, ezt is elfogadja. Maximálisan 12 szempont szerint lehet rendezni az egyes rekordokat. A rendező/válogató program négy fő részből áll:

1. Bevezető rész
2. Belső rendezés
3. Külső rendezés
4. Válogatás.

Rendezéskor az 1-3. részek dolgoznak, válogatáskor pedig csak az 1. és 4. programrész.

A bevezető rész olvassa be a programnak szóló vezérlő utasitásokat. Megvizsgálja, hogy van-e hiányzó vagy kettős vezérlő utasitás, adott esetben üzenetet küld a gépkezelőnek. Végül átalakitja a megadott paramétereket bináris formába. Ha kihagytunk egyes paramétereket, akkor az alapértelmezés szerinti paramétert használja. Ez tehát afféle előkészitő rész a tényleges rendezés előtt.

A belső rendezésnél az adatállomány előrendezésen esik át. A rekordokat a program beolvassa a tárba, majd sorba rakja őket, és munkaszalagokra irja. Erre a müveletre azért van szükség, mert jelentősen csökkenti a további rendezés gépidőigényét.

A külső rendezésnél az előrendezett rekordok egymás után kerülnek. Ekkor a hatékonyság érdekében a mágnesszalagok előre és hátra történő leolvasását egyaránt hasznositjuk. Lehetőség van arra is, hogy felhasználói rutinokat kapcsoljunk a rendezőprogramhoz.

A válogatás során a már előrendezett adatállományokból egyetlen rendezett állományt kreálunk. Legfeljebb 7 input állomány vehet részt a müveletben. A rendező/válogató program csak a BG partícióban futtatható.

5.5.3 UTILITY PROGRAMOK

Minden számitóközpontban gyakoriak az olyan rutinfeladatok, amelyeket nagyon sokszor el kell végezni. Célszerü tehát ezeket a feladatokat rugalmasan előre programozni, és e programokat felvenni az operációs rendszerbe. Mivel az alapvető funkciók ugyanazok, csak a konkrét feladatok térnek el, a programok megfelelő paraméterezéssel rendkivül sok dolgot "ügyesen" elvégeznek. Az ilyen programokat nevezzük utility programoknak. A utility programokat a számitógépgyártó cég az operációs rendszerrel együtt szállitja. Feladatuk a felhasználó mindennapi, hatékony támogatása. Elsősorban adatállományok másolására szolgálnak, egyik adathordozóról a másikra. Ha meggondoljuk, hogy például mágneslemezen levő rekordokat kell lyukkártyára másolni, akkor nyilvánvaló, hogy programozástechnikailag ez nem is olyan egyszerü feladat! A utility programokat három fő csoportba soroljuk:

1. Makróutasitásokkal generált programok
2. Adatállomány-másoló programok
3. Speciális programok.

1. A makróutasitásokkal generált programok igen nagy előnye a többivel szemben, hogy a felhasználónak rendelkezésére áll a makróutasitások rugalmas alkalmazási lehetősége. Ennélfogva saját igényeihez rendkivül jól alkalmazkodó programokat generálhat. Pusztán paraméterek megváltoztatásával könnyedén juthat módositott programhoz. Több funkciót összevonhat programjában, például a beolvasott lyukkártyaállományt kinyomtathatja és egyidejüleg ki is lyukasztathatja.

2. Az adatállomány-másoló programok adatállományok másolását végzik minden lehetséges periféria-kombinációban. Módositó utasitásokkal "bele is nyulhatunk" a rekordok belsejébe, mezőket irhatunk át, blokkolási tulajdonságokat változtathatunk meg stb.

3. A speciális programok olyan feladatokat látnak el, amelyek elsődlegesen nem adatátvitellel függnek össze. Ilyen például a mágneslemez tartaléksávjának kijelölése, mágneslemez törlése, mágneslemez inicializálása (sávcimek felirása), mágnesszalagok tartalmának összehasonlitása, mágneslemezes adatállományok visszaállitása lyukkártyáról és mágnesszalagról. A röviden felsorolt funkciók mögött terjedelmes, komplikált csatornaprogramok és feldolgozó rutinok állnak, bár a utility programok használata viszonylag egyszerü. A speciális utility programok mind előtér-, mind háttér tárterületen futtathatók.

5.6 AZ OPERÁCIÓS RENDSZER GENERÁLÁSA

A gyártó cégek az operációs rendszert különálló programok formájában bocsátják a felhasználó rendelkezésére. A teljes operációs rendszer megtalálható az ugynevezett disztributiv szalagon; ez egy közönséges mágnesszalag. Az ilyen állapotu operációs rendszer azonban nem alkalmas közvetlenül a számitógép üzemeltetésére, mert nincs meg az egyes programkomponensek között a szükséges kapcsolat. Igy például a Supervisor valamennyi funkciója különálló programrész alakjában áll rendelkezésre; sok program modulprogram állapotban, azaz "félkészen" szerepel stb. Az operációs rendszer ilyen felépitését nevezzük moduláris strukturának. A modularitás itt arra utal, hogy az operációs rendszer funkciói különálló egységenként, mintegy épitőkockák alakjában adottak.

A moduláris struktura hallatlanul nagy előnyt jelent a felhasználónak. Egy átlagos operációs rendszer is általában sokkal "többet tud", több funkcióval rendelkezik, mint amennyire egy-egy számitóközpontban ténylegesen szükség van. Ez azzal magyarázható, hogy az operációs rendszer univerzális software termék, ennélfogva lehetőséget nyujt eltérő jellegü feladatok megoldására is. Igy például tartalmazza az összes nyelv forditóprogramját, szerkesztőprogramokat, utility programok egész sorát, távadatfeldolgozást vezérlő programrendszereket, hibafelderitő rendszereket stb. Világos, hogy az olyan számitóközpontban, ahol a gépet tulnyomórészt adatfeldolgozásra használják, nincs értelme a rendszerben tartani például a távadatátviteli software-t. Ezzel jelentős helyet takarithatunk meg mind az operativ tárban, mind az operációs rendszert hordozó háttértárolón!

A teljes operációs rendszerből tehát a felhasználó válogatja ki azokat a funkciókat, amelyekre közvetlenül szüksége van. Emellett természetesen továbbra is megmarad neki a disztributiv szalag, amelyről szükség esetén minden nehézség nélkül ujabb, más funkciókat tartalmazó rendszert készithet. A müködőképes, effektiv operációs rendszer előállitását a disztributiv szalagon található operációs rendszer komponensekből - rendszergenerálásnak nevezzük. A rendszergenerálás tehát afféle alkotó-átszervező munka; ennek során egy maximális programkészletből válogatunk, de ugyanakkor e programok belső strukturájába is beavatkozunk. Mindezek alapján látható, hogy a rendszergenerálás igen komoly, nagy körültekintést igénylő feladat! Meg kell ugyanis becsülni a számitógép várható üzemeltetési profilját, a majdan megoldandó feladatok jellegét, és ezekhez kell alkalmazni a rendszergenerálás alapelveit. Egy számitógéprendszer hatékonyságát az operációs rendszere döntően befolyásolja.

Az operációs rendszer generálása lényegében azonos a SYSRES létrehozásával. Generálás során két fő müveletet kell elvégeznünk:

a) A Supervisor-funkciók meghatározása és a programkönyvtárak tartalmának definiálása,

b) A tervezett operációs rendszer létrehozása.

A forráskönyvtár a gyártó cég által szállitott makródefiniciókat tartalmazza. Ugyanitt mintaprogramokat is találhatunk, amelyek segitenek a rendszergenerálás egyes lépéseinél.

A modulkönyvtárban "félkész" modulprogramok találhatók. Itt vannak az input/output rendszer leforditott programjai is, az ugynevezett logikai input/output modulok.

A fáziskönyvtárban kész, futtatható programok vannak. Ezek nagyrészt a rendszer vezérlő-program komponensei, valamint szolgáltató, feldolgozó programok. Itt található a könyvtárkezelő programrendszer is.

A rendszergenerálás folyamatának megtervezése, ha helyesen járunk el, csökkentheti a teljes generálási időt. A rendszergenerálás megtervezésénél az alábbi feladatokat kell elvégeznünk:

a) Opciók tervezése, a Supervisor méretének becslése. Ennek során a rendelkezésre álló opciók közül kiválogatjuk azokat, amelyekre szükségünk lesz a Supervisorban, majd hozzávetőlegesen megbecsüljük ennek a Supervisornak a tárigényét.

b) Rendszerkönyvtárak és privát könyvtárak szervezése, méretük becslése. Ennek során a rendelkezésre álló mágneslemezterületet f elosztjuk az egyes könyvtárak között, ügyelve a helyes arányokra. Itt figyelemmel kell lennünk az egyes programok (például a szerkesztőprogram) munkaterületeinek meghatározására is.

A Supervisor-opciók kiválasztása a Supervisort generáló makróutasitások paraméterezésével történik.

A programkönyvtárak tartalmának meghatározásakor el kell döntenünk, mely programokat kivánjuk használni szerkesztett, futtatható formában, és mely program-elemek, modulok tárolása célszerübb "félkész" állapotban. A nem kivánatos modulokat természetesen kihagyjuk a leendő könyvtárakból.

A rendszergenerálási eljárás során keletkezett programlistákat az összes későbbi módositással együtt gondosan meg kell őrizni. Ugyanez vonatkozik a Supervisor-generálás listáira is. Ezekből lehet ugyanis később kideriteni, hogy mely funkciókat vettük be a rendszerbe, milyen összeállitásban. E listákat többnyire a rendszerprogramozók és a rendszerkarbantartó mérnökök használják. A rendszergenerálási folyamat vázlatát a 200. ábrán láthatjuk. Ha a generált rendszer nem fér el a rendelkezésre álló mágneslemez-területen, akkor a generálási folyamatot ujból kell kezdenünk.

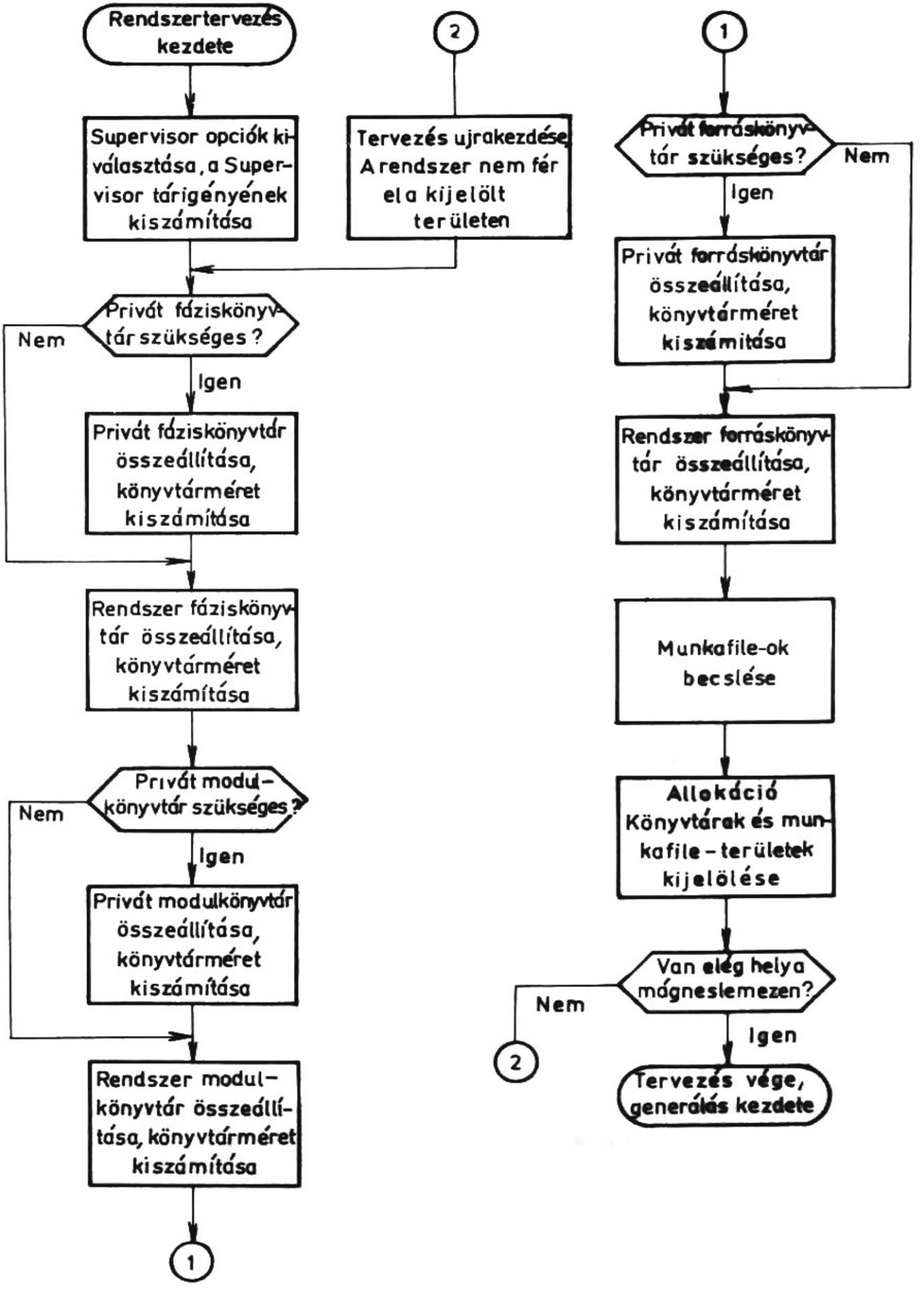
Tekintsük most át, milyen főbb opciókat lehet eldönteni a Supervisor-ra nézve a rendszergenerálás során! Mint tudjuk, maga a Supervisor-program is ilyen makróutasitások hatására jön létre; a kért Supervisor-funkciót a makróutasitás megfelelő paraméterezésével lehet igényelni, illetve kihagyni.
A főbb makróutasitások az alábbiak:

SUPVR - ezzel definiáljuk a rendszer olyan paramétereit, mint a multiprogramozás jellege, ASCII-kód alkalmazása, távadatfeldolgozás. Világos, hogy például a multiprogramozást vezérlő Supervisor-nak sokkal "többet kell tudnia", mint az egyszerübb változatnak;

CONFG (Konfiguráció) - ez a makróutasitás határozza meg a rendszer konfigurációját, valamint egyes opcionális Supervisor-szolgáltatásokat. Igy például itt dől el, hogy beépitették-e a processzorba a decimális lebegőpontos aritmetikai utasitáskészletet;

STDJC (Standard Job Control) - a Job Control opciók kiválogatására szolgál. Az itt meghatározott opciók maradandók, standard-ok lesznek; az egyes job-okban kiadott OPTION utasitások ezeket csak átmenetileg, a job futásának időtartamára irhatják felül;

FOPT (Optional Features = opcionális tulajdonságok) - egyéb Supervisor-opciókat dönt el. Igy például felkészithetjük a Supervisort arra, hogy bizonyos tipusu programmegszakitások kezelését átadhassa a felhasználói rutinoknak. Ha ezeket elhagyjuk, a Supervisor "magától" nem lesz alkalmas ilyen járulékos szolgáltatásokra. Itt dőlnek el az olyan dolgok is, mint például a file-védelem, a csatornaprogram ujrainditása az utolsó végrehajtott csatornaparancsszótól stb. kivánatos-e vagy sem;



200. ábra. Az operációs rendszer generálása

ALLOC - a Supervisor tárfelosztását határozza meg. Ebben az utasitásban rendelkezhetünk afelől, mekkora területet tartson fenn a Supervisor a BG, F1 és F2 particiók számára;

IOTAB - a Supervisor input/output perifériatáblázatait generálja. Ezek tartalmazzák a logikai és fizikai perifériacimeket és azok kapcsolatát, például azt, hogy a SYSIPT logikai perifériacim melyik fizikai egységhez van rendelve;

ASSGN - ezzel tölthetjük ki az előzőekben generált input/output perifériatáblázatokat;

SEND (Supervisor End = Supervisor vége) - a Supervisor végét mutatja. Ez mindig duplaszócim a tárban.

A vázolt makróutasitásoknak igen sok paramétere van. Ezek általában olyanok, hogy több lehetőség közül választhatunk bennük; például eldönthetjük azt, hogy akarunk-e multiprogramozott üzemmódot használni, és ha igen, milyen jellegűt stb. Ezeket a lehetőségeket az illető makróutasitás paraméterezésével adjuk meg. Ha valamelyik paramétert kihagyjuk a makróból, akkor a rendszergenerálás során ugynevezett alapértelmezés lesz érvényben erre a paraméterre nézve. Az alapértelmezésre a makrók eleve "felkészültek", ugy irták meg őket, hogy a kihagyott paraméter helyett ezt tekintsék érvényesnek. Ezzel ugyancsak egyszerűbbé válik a rendszergenerálás ott, ahol az alapértelmezések elfogadhatók.

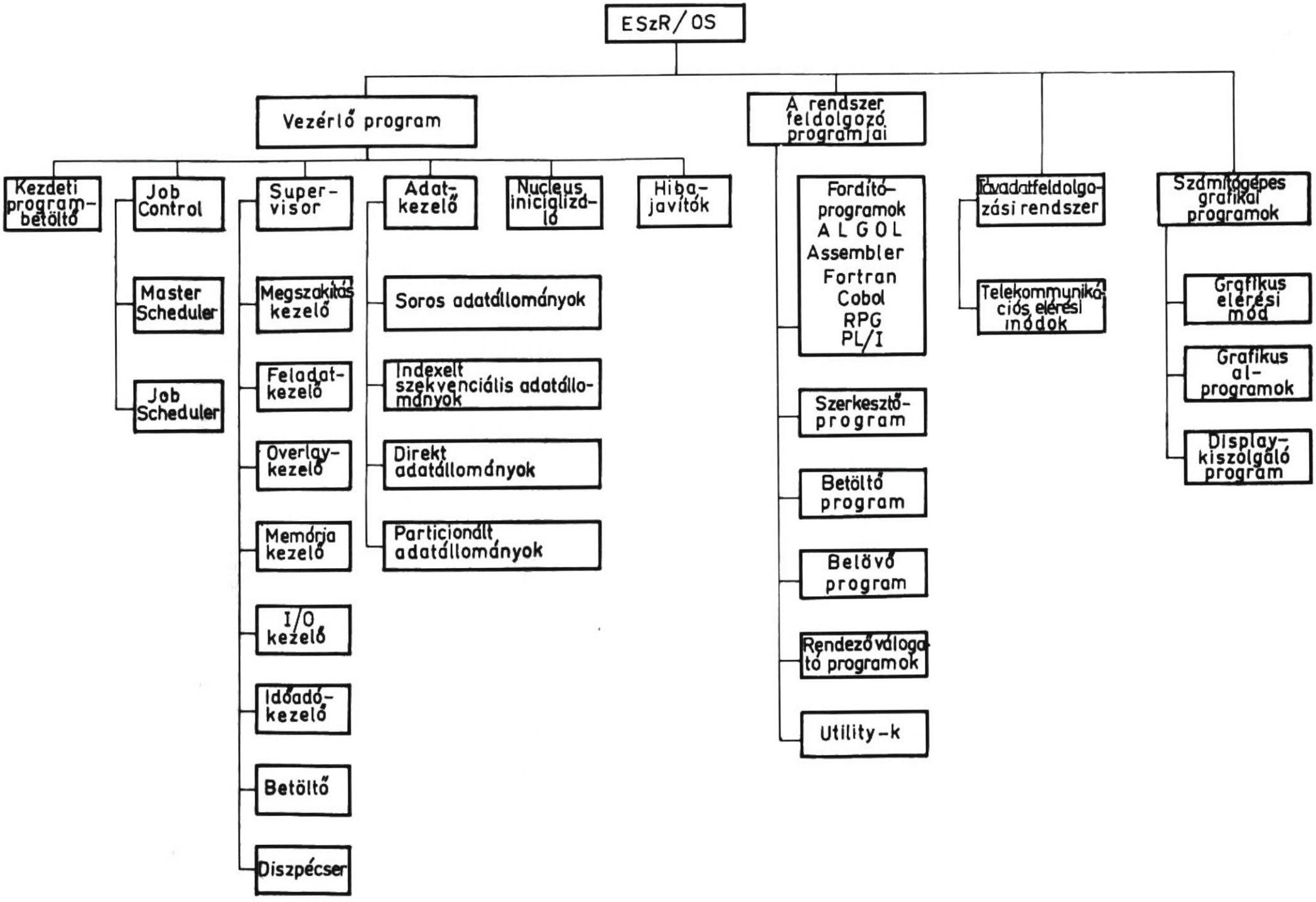
5.7 AZ ESZR/OS OPERÁCIÓS RENDSZERRŐL

Az operációs rendszer - mint a software-ellátás egyik alkotóeleme - felépitését és működését egy sajátos operációs rendszeren, az ESZR/DOS-on mutattuk be. Ez viszonylag eléggé feladatorientált rendszer volt, és főleg a kisebb ESZR számitógépek üzemeltetésére alkalmas. A nagyobb ESZR modelleknél az ESZR/DOS hatékonysága jóval kisebb lenne, mivel ez nem alkalmazkodna a nagy számitógép erőforrásaihoz, műveleti sebességéhez, tágabb lehetőségeihez. Az ilyen modellek üzemeltetését ezért legtöbbnyire nem az ESZR/DOS, hanem az ESZR/OS operációs rendszerrel végzik. Ez sokkal kevésbé feladatorientált, ugyszólván teljesen univerzális rendszer. Bizonyos feltételek fennállása esetén (operativ tár-kapacitás, perifériakészlet stb.), a kisebb ESZR számitógépeken is üzemeltethető, azonban az ESZR/OS itt "nem találna magára" kellőképpen; többnyire szűknek találná a kisebb számitógép adta lehetőségeket a hatékony üzemeltetés szempontjából. A következőkben szeretnénk nagyon vázlatos betekintést nyujtani az ESZR/OS operációs rendszer felépitésébe, főbb működési elveibe. Ebből bárki számára nyilvánvalóvá válik a különbség a kétféle rendszer között. Hangsulyozzuk azonban, hogy az ESZR/OS igen bonyolult rendszer, itt csak érintőlegesen szólhatunk róla.

5.7.1 AZ ESZR/OS FELÉPÍTÉSE

Az ESZR/OS operációs rendszer felépitését a 201. ábrán láthatjuk.

Ez a rendszer is - mint minden operációs rendszer, amelyet III. generációs számitógépekre terveztek - vezérlő és feldolgozó programokból áll. A feldolgozó programok valamennyien a vezérlő program irányitása, felügyelete alatt futnak, ugyanakkor szükség szerint igényelhetik a vezérlő prog-



201. ábra. Az ESZR/OS operációs rendszer felépítése

ram sajátos szolgáltatásait. Az ESZR/OS univerzális rendszer lévén, jóval "többet tud", mint társa, az ESZR/DOS, és ennek megfelelően a vezérlő program funkciói is lényegesen kibővültek.

Itt is megtaláljuk a kezdeti programbetöltőt, a Job Controlt, a Supervisort, Job Control azonban már két különálló részre osztható: a Job Scheduler a job-ok beolvasását, együttes végrehajtását, a job control utasitások olvasását stb. végzi, a Master Scheduler pedig a rendszer és a gépkezelő közötti kapcsolattartásért felelős. A Supervisor is több önálló komponensből épül fel. Ugyancsak a vezérlő programhoz tartozik az adatkezelés, amely az ESZR/OS-ben sokkal bonyolultabb adatstrukturákkal dolgozik, mint az ESZR/DOS-ban. Végül, a vezérlő program részét képezik a hibajavitó, a rendszer állapotát visszaállitó programok is. Maga a vezérlő program rezidens része - a nucleus - mérete is ennek megfelelően alakul.

A feldolgozó programok között itt is megtaláljuk a már megismert forditóprogramokat, a szerkesztőprogramot, rendező/válogató programokat és a utility programokat. Különleges adatfeldolgozást tesznek lehetővé a távadatfeldolgozás programjai: külön telekommunikációs adatelérési programok, programcsomagok feladata a távolból érkező adatok, adatállományok fogadása, előkészitése feldolgozásra. A számitógépes tervező munkát segitik a számitógépes grafika programjai alfanumerikus és grafikus display-ek alkalmazásával. Ennek révén lehetőség van arra, hogy a grafikus tervezési eljárásokhoz előre megirt és belőtt könyvtári rutinokat használjunk, ami a rendszer hatékonyságát jelentős mértékben növeli. Adott esetben a gépkezelő is dolgozhat display-berendezéssel a konzolirógép helyett, ha a feladatok jellege ezt indokolttá teszi. Mint látjuk, a rendszer bonyolultsága elég nagy, és ez nemcsak az egyes rendszerprogramok mennyiségében, hanem ezek együttmüködésének szervezésében is megmutatkozik. A teljes rendszer minimális operativ tár-igénye 128 Kbyte, ezzel az ESZR/OS "még éppen" müködőképes. Hatékony üzemeltetésére ennél jóval nagyobb tárméretre és sok-sok perifériára van szükség.

5.7.2 A PROGRAMOK FELDOLGOZÁSA

A programok feldolgozása lényegében ugyanugy történik, ahogyan ezt az előzőekben megismertük. A programozó a feladat megoldását egy programozási nyelven leirja, kódolja, majd ezt a szimbolikus programot a megfelelő forditóprogrammal gépi kóduvá alakittatja át. A folyamat tulajdonképpen a tárgyprogram (object module) létrehozása a forrásprogramból. A szerkesztőprogram ezután összeállitja a betöltendő modult (load module), és a végrehajtás megkezdődhet. Itt azonban nem kap rögzitett kezdőcimet a betöltendő modul: ennek következtében adott esetben - hiváskor - a tár bármely területén futtatható! Ez a rugalmas programszerkesztés igen hajlékonnyá teszi a rendszert, mert nem köti szorosan az egyes particiókhoz a programvégrehajtást.

A rendszer programjait és a felhasználói programokat háromféle könyvtárban tárolhatjuk. Ezek: az általános, az ideiglenes és a privát könyvtárak.

Az általános könyvtárban vannak a gyakran végrehajtásra kerülő programok, például a forditóprogramok stb. Ez a könyvtár mindig nyitott a felhasználó felé, és rendszergeneráláskor definiálják.

Az ideiglenes könyvtárakban azok a programok találhatók, amelyekre a job feldolgozásának egy későbbi időpontjában lesz majd szükség. Ezt használják például a forditóprogramok is output területként. A job befejeztével az ideiglenes könyvtárak tartalma törlődik.

A privát könyvtárakban azok a programok vannak, amelyek viszonylag ritkán kerülnek feldolgozásra, és igy nem érdemes őket az általános könyvtárba felvenni. A privát könyvtárak tartalmához alkalmas job control utasitásokkal lehet hozzáférni.

A betöltött program akkor alkalmas a futtatásra, ha benne nem szerepelnek feloldatlan külső hivatkozások, tehát a szerkesztőprogram helyesen "igazitotta össze" az egyes modulokat. Itt azonban, mivel a betöltési cim a szerkesztés alatt nem dől el, a betöltési modul is tartalmazhat külső hivatkozásokat, de a programozó felelőssége az, hogy a futás során ezeket ne kivánja felhasználni! Ez esetben a szerkesztés sikeresnek minősült, és a betöltött program futtatható.

A tárban végrehajtás alatt álló program bármikor hivhat egyéb programokat, hiszen a szerkesztett modulok bármely tárterületen futtathatók.

5.7.3 A PROGRAMOK SZERKEZETE DINAMIKUS PROGRAMBETÖLTÉS

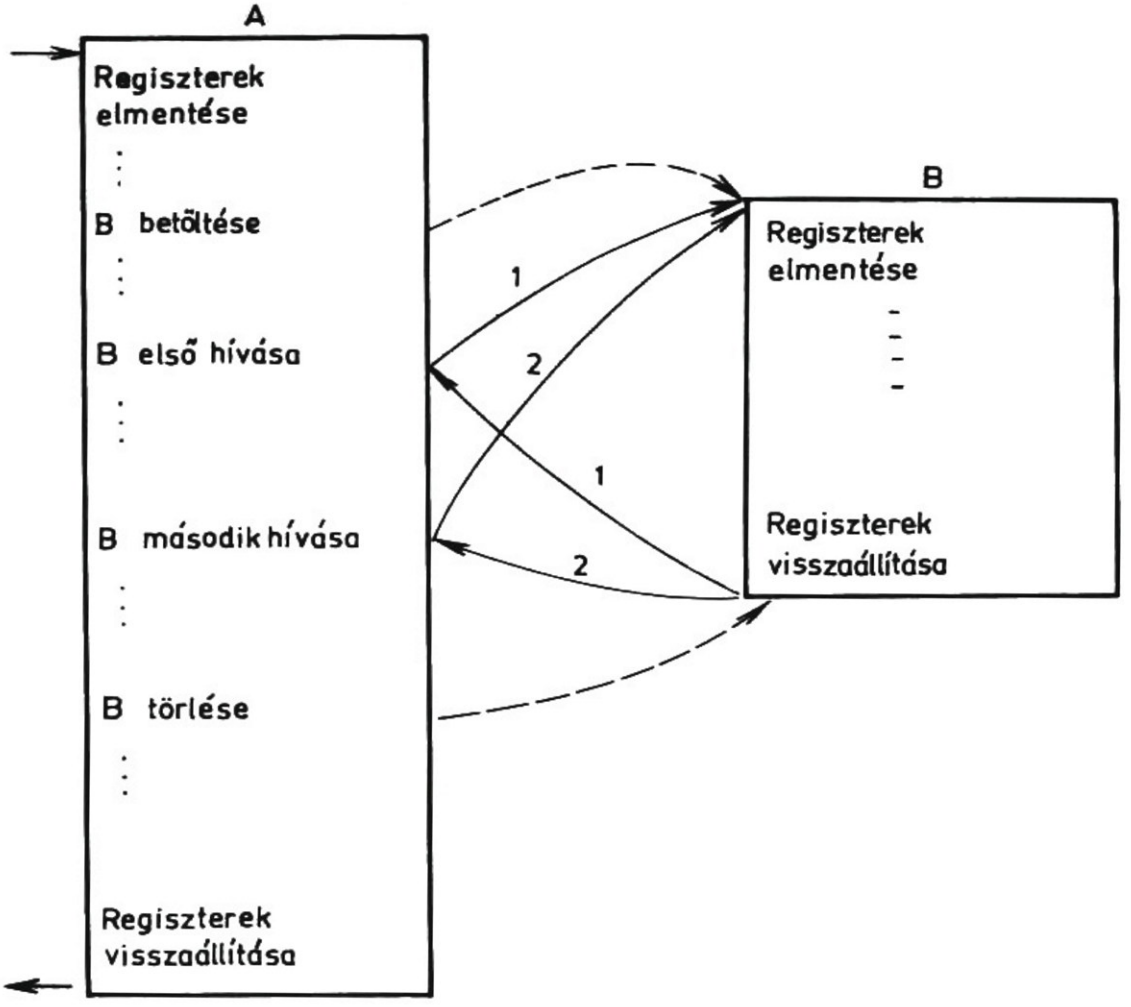
A betöltési programok - szerkezetük alapján - négy fő csoportba sorolhatók. A csoportositás alapján ezek más-más módon hasznositják a rendszer adta lehetőségeket.

1. Az egyszerü modulszerkezet a futtatáskor felhasznált összes gépi kódot magába foglalja. Ez a modul futása alatt semmilyen másik modulnak nem adja át a vezérlést, mindent maga valósit meg. Betöltés szempontjából egységes egészet képez. Hasonló ez az ESZR/DOS-ban megismert egyfázisu programhoz.

2. Az overlay-szerkezetü modul is betöltési egységet képez a programkönyvtárban, de itt kijelölhetünk olyan szegmenseket a modulon belül, amelyek nem tartózkodhatnak egyidejüleg az operativ tárban. Az ugynevezett gyökérszegmens mindig a tárban van a végrehajtás alatt, mig a többi szegmens az aktuális hivás során kerül be oda. Hasonló ez a szerkezet a már ismerős multifázisu programhoz.

3. A dinamikus soros modulszerkezet azt jelenti, hogy az egyes modulokat megfelelő makróutasitásokkal hivjuk be a tárba. A behivás többek között a modulnév alapján történik. A hivott modul végrehajtása után a vezérlés mindig a hivó modulhoz kerül vissza. A modulok végrehajtása tehát időben egymásután, sorosan történik. Ha egy modulra már nincs szükség a tárban, akkor a megfelelő makróutasitással eltávolithatjuk. A dinamikus soros modulhivás folyamatát a 202. ábrán láthatjuk. A "B" nevü modul mindaddig a tárban marad, mig a törlő utasitást ki nem adja a B-t hivó A modul.

4. A dinamikus párhuzamos modulszerkezet használatával elérhető az, hogy a hivott és a hivó modul végrehajtása egyidejüleg történjen. Az "egyidejüséget" természetesen ugy kell értelmezni, hogy a két modul közül mindig csak az egyik használhatja a processzort. Az ilyen modulok végrehajtása mindig azon időszeletek alatt történik, amelyekben a másik modul input/output-ot végez, vagy más feltétel teljesülésére vár.



202. ábra. A dinamikus soros modulhivás folyamata

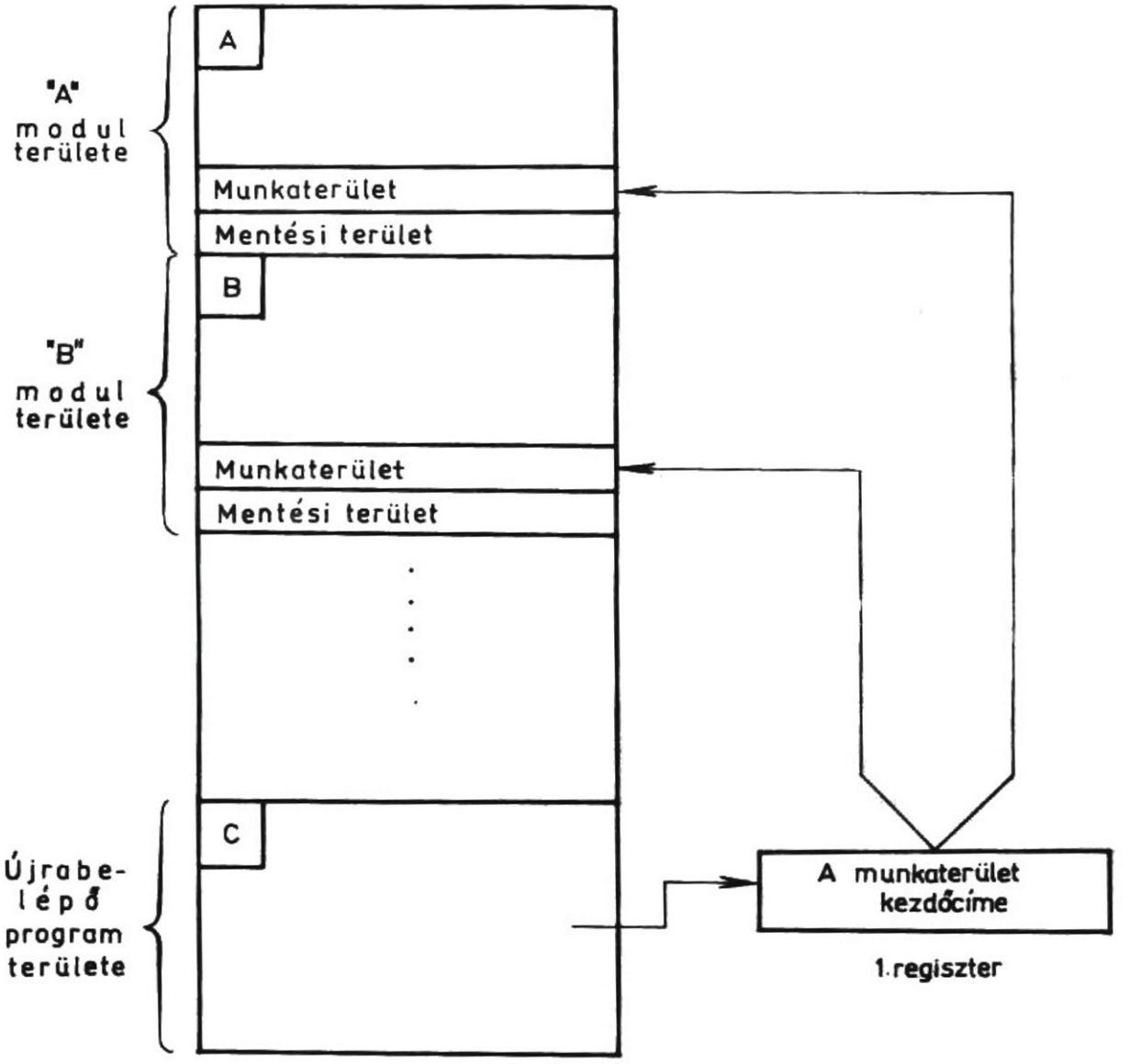
Valamennyi modul végrehajtása azzal kezdődik, hogy a modul elmenti az általános regiszterek tartalmát a tár egy meghatározott területére. Erre azért van szükség, mert a modulok önállóan vagy együttesen is végrehajthatók, és minden modul ugyanazokat a regisztereket tudja csak használni. A modulvégrehajtás után az utolsó utasitással minden modul visszaállitja a regiszterek tartalmát, hogy a vezérlés visszakerülhessen a hivó modulhoz. Most már eléggé indokoltnak érezhetjük azt, hogy a szerkesztés során a modulok nem kapnak rögzitett betöltési cimet: ha kapnának, a dinamikus modulhivás nem lenne megvalósitható, a rendszer merevebb lenne.

5.7.4 AZ UJRABELÉPŐ PROGRAMOK FOGALMA ÉS HASZNÁLATA

Az eddig megismert programfajták a végrehajtás alatt általában nem őrzik meg azt az alakjukat, amely a végrehajtás kezdetekor fennállt, hanem a futás során kisebb-nagyobb változáson mennek keresztül. Az ilyen programot hiába tartjuk tovább a tárban, ismételten nem hajtható végre, mert ez már egy "másik" program végrehajtását jelentené. Vannak olyan programok, amelyek ismételt végrehajtás előtt visszaállitják eredeti alakjukat. Végül vannak olyanok, amelyek a futás során nem változnak

meg, ezeket ujrabelépő programoknak nevezzük. Ezek a vezérlő program területére tölthetők be, és ennek megfelelően memóriakulccsal védettek minden felülirás ellen. Igen nagy előnye az ilyen programoknak, hogy egyidejüleg több feladat megoldására is használhatók, függetlenül attól, hogy az egyes feladatok "hol tartanak" az ujrabelépő program használatában.

Multiprogramozott üzemmódban az ujrabelépő programok ilyen felhasználása közönséges esetnek minősül. Megtakaritjuk az ismételt programbetöltéshez szükséges input/output müveleti időt, és az operativ tárat is jobban kihasználjuk (a modulnak csak egy példányára van szükség). A vezérlő programnak több komponense ujrabelépő. Az ujrabelépő programok számára az egyes job-ok memóriaterületén munkaterületet és mentési területet kell fenntartani. A munkaterületet az ujrabelépő program használhatja, a mentési területre pedig akkor kerül információ (programállapotszó, regiszterek tartalma stb.), ha az ujrabelépő program futását valamilyen feltétel miatt meg kellett szakitani. Mivel a munkaterület kezdőcime és nagysága változó lehet, az ujrabelépő program sajátos makróutasitással keresi meg a szükséges kezdőcimet. Ezt egy általános regiszterben kapja meg a Supervisor-tól.



203. ábra. Az ujrabelépő program használata

Az A és B program hivja a C ujrabelépő programot, amely a Supervisor tárterületén található. Miután az A program kiadta a hivást, az ujrabelépő C program a munkaterület cimét és magát a munkaterületet kéri a rendszertől. A kezdőcim az 1. regiszterben található. Ha most a B program hivja C-t, akkor a kijelölt mentési területre kerül a hardware aktuális állapotát leiró információ, és B használhatja C-t. Az 1. regiszterben a B mentési területének/munkaterületének cime lesz. B végrehajtása után az előbbi mentési területről visszaállitható a hardware előző állapota, és az A program folytatódhat. Ennek alapján nyilvánvaló, miért nem szabad a C ujrabelépő programnak a futás alatt megváltoznia: minden hozzáfordulásnak, bármely időpillanatban is történjen az, "ugyanazt" a C programot kell használnia.

5.7.5 A VEZÉRLŐ PROGRAM ÜZEMMÓDJAI

Az ESZR/OS vezérlő programja három lehetséges üzemmódban dolgozhat: egyetlen program futtatása, multiprogramozás rögzitett számu feladattal és multiprogramozás változó számu feladattal.

Az egyetlen program futtatása azt jelenti, hogy az illető program a rendszer összes erőforrását használhatja. Egyébként bármilyen szerkezetü lehet. Ha ez a program egy dinamikus soros modulszerkezet első eleme, és egy további modult hiv, akkor a hivott modul - ha ujrabelépő - nem töltődik be még egyszer a tárba, hanem a már ott levő példányát kapja meg a hivó program.

Multiprogramozás során a rendszerben párhuzamosan több job step fut, és a job-ok egymástól teljesen független egységeket képeznek. Ugyanazon job két step-je azonban már nem futtatható időben egyszerre: a step-ek egymásnak adatállományokat adhatnak át, például a forditóprogram tárgymodult szolgáltat a szerkesztő számára stb. Emiatt a step-ek sorrendje job-on belül kötött.

A rögzitett számu feladattal történő multiprogramozásnál a tár két fő területre oszlik: a rendszer területére és a dinamikus területre. A rendszer területén van a vezérlő program rezidens része és táblázatai. A dinamikus terület több particiót tartalmaz; ezek számát rendszergeneráláskor döntik el. Maximálisan 15 partició használható job-ok feldolgozására. A rendszernek legfeljebb 3 adatbeviteli és 36 adatkiviteli programja dolgozhat egy adott időben. Az egyszerre elvégezhető feladatok száma is korlátozott, legfeljebb 255 lehet. A partíciók különböző prioritási értéket kapnak, ez kihatással van az illető partícióban futó program végrehajtási fontosságára. A felhasználó döntheti el, hogy melyik partícióban kivánja futtatni programját, melyik "osztályba" sorolja azt. Igy a sok processzoridőt "fogyasztó" programokat alacsony prioritási szintre, mig a telekommunikációs programokat magas prioritási szintre célszerü tenni, hogy a válaszidő rövid legyen.

A változó számu feladattal történő multiprogramozásnál a dinamikus tárterület partícióhatárai változóak, s ez lényeges különbség az előbbi multiprogramozáshoz képest. Ha a partícióméretet a JOB-utasitásban adjuk meg, akkor a job valamennyi step-je ilyen méretü partícióban fog futni. Ha a partició-

méretet az EXEC utasitásban adjuk meg, akkor minden job step más-más méretü particióban futhat, ami jelentősen javithatja a tárkihasználást. A változó számu feladattal történő multiprogramozásnál lehetőség van arra is, hogy az adott job step járulékos tárigényt jelentsen be, és a kért területet meg is kapja, az esetlegesen fel nem osztott tárrészből. Ha ilyen tárterület nem állna rendelkezésre, akkor a rendszer hajlandó egy másik job-hoz rendelt területet ideiglenesen felszabaditani ugy, hogy ezt a job-ot háttértárra teszi. A kért terület felszabadulása után a kimentett job visszakerül a helyére, és folytatódik végrehajtása. Háttértárra csak az a job menthető ki, amelynek JOB vagy EXEC utasitásában erre a felhasználó engedélyt adott (lásd a ROLL paramétert).

Az előzőekben megkiséreltük vázlatosan összefoglalni az ESZR/OS operációs rendszer olyan főbb sajátosságait, amelyekben különbözik az ESZR/DOS-tól. Kitünt, hogy ez a rendszer a lehetőségek sokkal bővebb halmazát bocsátja a felhasználó rendelkezésére, de ennek "ára is van": nagyobb konfigurációt, több hardware erőforrást igényel. Ezen tulmenően igyekszik a lehető legjobban alkalmazkodni az adott feladat sajátosságaihoz, igényeihez, és ily módon optimalizálni próbálja a gép kihasználását. Ez lenne végső soron minden operációs rendszer célja: ebből az egyik többet, a másik kevesebbet valósit meg a fejlettségtől függően.

6. Számítógépes hálózatok

Az előző fejezetekben az olvasó megismerkedett a számitógépek müködési elvével. Ha ezen a ponton befejeznénk a rendszertechnikai szemléletü tárgyalást, az eddig kialakitott kép nem lenne átfogó jellegü, hiszen még nem beszéltünk a napjainkban egyre nagyobb jelentőségü számitógépes hálózatokról.

Azokat a rendszereket, amelyekben a felhasználók saját termináljukon, számitógépükön keresztül más, - az esetek döntő többségében távoli - számitógépek erőforrásait is igénybe vehetik, összefoglaló jelleggel és a teljesség igénye nélkül számitógépes hálózatnak nevezzük. A számitógépes hálózatok teremtik meg azt az infrastrukturát, amely lehetővé teszi, hogy a számitógépes szolgáltatások közkinccsé váljanak, az információfeldolgozás és -továbbitás térbeli és lehetőleg időbeli korlátok nélkül valósulhasson meg. Figyelemre méltó az alábbi idézet:

"Mig a tizenkilencedik század végén az emberi hang átvitele jelentett forradalmat, addig a huszadik század végén a számitógépes kommunikáció kérdéseinek megoldása lesz a fő problémánk".

A fejezet célja, hogy összefoglaló jelleggel, - matematikai eszközök felhasználása nélkül - átfogó képet adjon a mintegy másfél évtizede intenziv fejlesztés alatt álló uj alkalmazási területről.

6.1 A TÁVOLRÓL TÖRTÉNŐ FELDOLGOZÁSOK FEJLŐDÉSE, FELHASZNÁLÁSI TERÜLETEK

A távoli feldolgozások lehetőségét a számitástechnika és a távközléstechnika egymásra kölcsönösen visszaható integrálódása alapozta meg. Ez a folyamat a hatvanas években indult el intenziven. A két szakterület eltérő müszaki - gazdasági paraméterekkel rendelkezik. A számitástechnikára jellemző a rendkivül gyors fejlődés, ebből következően a berendezések viszonylag gyors elavulása, a nagy megbizhatósági követelmények, a néhány százezer vagy néhány millió elemi müvelet elvégzése másodpercenként, stb. Ezzel szemben a távközléstechnika viszonylag alacsony sebességgel dolgozik, a kapcsolatok felépitése a néhány másodperc és néhány perc nagyságrendjébe esik átlagosan, a terület fejlődése nem ugrásszerü, rendkivül lassu az elavulás mértéke, időt álló rendszerek alakultak ki, stb.

A számitástechnika fejlődése a hatvanas évek során rendkivüli mértékben felgyorsult. A hagyományos módok mellett addig még ismeretlen uj szolgáltatásokat kellett kifejleszteni. Megoldásra várt a nagytömegü információ tárolása, lekérdezése, módositása. Előtérbe került az ipari vállalatok termelésirányitása, a kereskedelmi tevékenységekkel kapcsolatos adminisztráció gépre vitele, a biztositó társaságok, bankok, pénzintézetek ügyvitelének automatizálása. Létrejöttek a földrészekre kiterjedő repülő,

vasuti és egyéb helyfoglaló rendszerek. Általánossá vált a bérmunka jellegü feldolgozás, kifejlesztették az individualizált oktatási rendszereket, stb.

A fenti felhasználási területek nagyrészén létrejött az ember-gép kapcsolat interaktiv, vagy másnéven párbeszédes (dialógus) formája. Ezekben az alkalmazásokban közös jellemvonás, hogy a feldolgozás igénye a számitóközpontoktól távol, esetenként többszáz, vagy többezer kilométer távolságból jelentkezik. Ezért a távközlés-technikának magas szinvonalon meg kellett oldani a digitális információ továbbitását. Kézenfekvő volt a gondolat, hogy az országokra, földrészekre kiterjedő telefonhálózatot vegyék igénybe a digitális információ továbbitására.

A hatvanas évek második feléig a telefonvonalakat a telefonbeszélgetésekhez hasonlóan alkalmazták a digitális átvitel céljára. Ennek két formája volt: vagy telefonközpontokon keresztül kapcsolással, vagy a felhasználó által bérelt vonalon. Az emlitett rendszereket összefoglalóan vonalkapcsolt hálózatoknak nevezzük, hiszen minden esetben közvetlen áramköri összeköttetést kellett létrehozni a felhasználók berendezései között, és ez a kapcsolat mereven fennmaradt a kapcsolat szétbontásáig, illetve bérelt vonal esetében állandóan fennállt. Vonalkapcsolt számitógépes hálózatokban a hálózat erőforrásait igy az egyes felhasználókhoz rövidebb - hosszabb ideig hozzárendelik.

A hatvanas évek végére fejlesztették ki az üzenetcsomag kapcsoló rendszereket, tömörebben a csomagkapcsolt rendszereket. A csomagkapcsolt hálózatok legfontosabb jellemzője, hogy nincs közvetlen áramköri kapcsolat a hálózat egyik végpontja között sem, hanem a továbbitandó üzenet egy, vagy több állandó hosszuságu üzenetcsomagot alkot, és ezeket a csomagokat egy kisszámitógépeket is tartalmazó adatátviteli hálózat továbbitja rendeltetési helyükre. A csomagkapcsolt rendszerek másik jellemzője, hogy a rendszer erőforrásait az üzenetcsomagokhoz rendelik. A csomagokat az adatátviteli hálózat csomópontjaiban tárolják, ellenőrzik és a hálózat pillanatnyi állapotának megfelelően optimális irányba eső szomszédos csomópont felé továbbitják.

A hetvenes évek elején előtérbe került az üzenetcsomag rádiózás. Itt állandó, vagy változó méretü csomagokat küldenek szét az URH rádiózáshoz hasonló körülmények között. Jelentősége elsősorban az, hogy nem igényel előre kiépitett - és főleg nem a digitális átvitelre szánt - telefonhálózatot, alkalmazza a csomagkapcsolt rendszerekben jól bevált üzenetcsomagokat, ugyanakkor lényegesen olcsóbb és egyszerübben bővithető, mint az előbb emlitett két hálózat, két tipus.

6.2 A FELHASZNÁLÓK KARAKTERISZTIKÁJA

Bármilyen tipusu kommunikáció megvalósitásához minimálisan három elem szükséges, sorrendben információ forrás (adó), a továbbitáshoz szükséges közeg és információ nyelő (vevő). Bármelyik elem hiánya lehetetlenné teszi a kapcsolatteremtést. Számitógépes hálózatok környezetében a felhasználókat az általuk üzemeltetett berendezések alapján az alábbi három csoportba soroljuk:

- kötegelt feldolgozás terminál állomásai
- számitógépek

Az interaktiv terminálok közé soroljuk az alfanumerikus display-ket, az irógéptipusu berendezéseket, a teletype tipusu telexgépeket, valamint a tárolócsöves grafikus display-ket. A nem tárolócsöves gra-

fikus display-ket a számitógépek csoportjában tartjuk nyilván, hiszen távoli terminálként - tekintettel a különböző speciális feladatok ellátására is - minden esetben egy-egy kisszámitógép mögött találjuk őket.

A kötegelt távoli feldolgozás terminál állomásai általában kártyaolvasót, sornyomtatót, ritkábban még mágnesszalagegységet és kártyalyukasztót is tartalmaznak. Helyenként a lyukszalagos perifériákkal is találkozhatunk.

A három jellegzetes csoportot alkotó berendezések mind információ forrásként, mind pedig információ nyelőként szerepelhetnek. A közeget alkotó eszközök aktiv, vagy passziv jelleget mutatnak. Aktiv a közeg akkor, ha az információforgalom lebonyolitásában számitógépek is részt vesznek, ezek a mini, illetve microcomputerek nagyságrendjébe esnek. A passziv közeg nem tartalmaz számitógépet, ebben az esetben leggyakrabban telefonközpontokon keresztül kapcsolt vonalak (áramkörök) vagy bérelt vonalak (áramkörök) alkotják az adatátviteli csatornákat. A felhasználókat elsősorban az általuk generalt, vagy igényelt átlag és csucssebességekkel jellemezhetjük legjobban. Ezek az alábbiak:

Tipus	Átlagos sebesség (bit/sec)	Maximális sebesség (bit/sec)
Interaktiv terminálok (input)	1	10^2
Interaktiv terminálok (output)	10	10^3
Kötegelt távoli feldolgozás terminál állomásai	10^2	10^4
Számitógépek	10^4	10^6

A táblázat adatai a számitógépes hálózatok szempontjából fejlett országokban mért adatokat reprezentálják. Könnyü utánagondolni, hogy az interaktiv terminálok kategóriában például az irógép előtt ülő felhasználó átlagosan nem képes több információt begépelni, mint 1-2 bit másodpercenként. Egy elmélyült munkát végző software-fejlesztő, vagy kutató esetén viszonylag hosszu a vizsgált időintervallumba eső gondolkodási idő, igy csupán lassu vagy közepes billentési sebesség érhető el. A maximális sebesség egyszerüen kiszámitható az interaktiv eszköz paramétereiből. Az általánosságban használt irógépek 10-11 kar/sec-os maximális sebességgel képesek müködni. A gömbfejes irógépnek 15 kar/sec a maximális irássebessége. Az utóbbi adatot figyelembe véve és például 8 bites EBCDIC karaktereket feltételezve 120 bit/sec-ot kapunk eredményül. Input irányban ennél még valamivel nagyobb sebességet lehet elérni ha az alfanumerikus display billentyüzetéről egy-egy rövid időintervallumban néhány jól begyakorolt parancsszót vagy utasitást viszünk be. (Pl.: ready, start...)

Output irányban átlagosan egy nagyságrenddel nagyobb sebességekkel kell számolnunk a rövidebb várakozási idők, valamint az alfanumerikus display-k aszimmetrikus tulajdonságai miatt. (A képernyőn output irányban megjelenő karakterek sebességét nem korlátozza a lassu billentyüzet).

A kötegelt távoli feldolgozás terminál állomásain üzemelő perifériális berendezések maximális sebessége (pl. sornyomtató, mágnesszalag) meghatározza a 10^4 bit/sec nagyságrendet. A számitógép-számitógép közötti kapcsolatra a millió bit/sec nagyságrend a jellemző. A táblázat jól mutatja, hogy a csucssebességek és az átlagos sebességek között nagyságrendi különbségek vannak. Ez a tény jelentős, amint azt a későbbiek folyamán látni fogjuk. Ennek kapcsán válik lehetővé a különböző módszerek felhasználásával az adatátviteli csatornák terminál ⟶ számitógép irányu jobb kihasználása.

A felhasználók három csoportjához az évek során létrejött szabványok három sebességkategóriát ajánlanak.

a) kissebességü adatcsatornák (45-300 bit/sec)

b) középsebességü adatcsatornák (600-9600 bit/sec)

c) nagysebességü adatcsatornák (9600-10^6 bit/sec)

6.3 SZÁMÍTÓGÉPES ADATÁTVITEL ALAPJAI

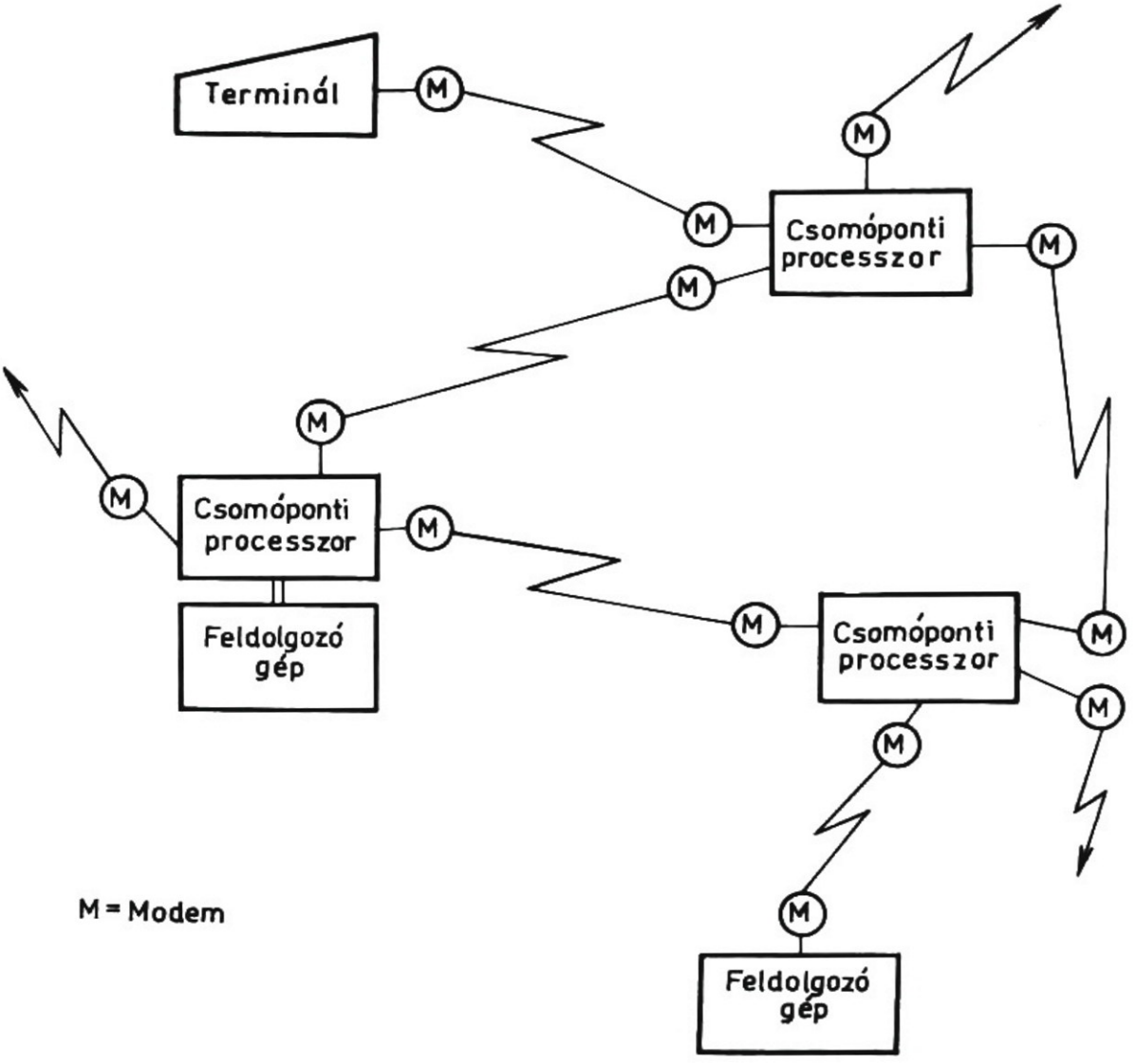
A számitógépes hálózatoknál a digitális információ átvitele adatcsatornákon keresztül történik. Az adatcsatornák fizikai megvalósitása megoldható mind vezetékes, mind pedig vezeték nélküli rendszerben. A kettő minőségileg különbözik egymástól, mert a vezetékes adatátviteli rendszerek minden esetben két vagy több pont között létesitenek kapcsolatot, ezzel szemben a rádióhullámok egy adott geográfiai területet fednek le, amelyen belül bárhol elhelyezkedhetnek a felhasználók. A napjainkig kifejlesztett hálózatokban az esetek döntő többségében a digitális információ továbbitására vezetékes adatátviteli rendszereket használtak. Az alacsonyabb sebességkategóriákban (néhányszor tiz karakter/sec) megoldható a közvetlen digitális átvitel is (pl.: a telex hálózatokban), az ennél nagyobb sebességkategóriákban azonban szükségessé válik a digitális információ átalakitása a vezetékek, vonalak paraméterei miatt analóg formára.

Nézzük meg, hogy a távoli felhasználó és a feldolgozást végző számitógép, vagy számitógépek közé milyen funkcionális elemek kerülnek. A 204. ábrán látható egy lehetséges adatut a felhasználó és a feldolgozó gépek között.

A továbbitásra váró információ digitális formában indul utjára a termináltól és digitális formában is érkezik a hálózat csomópontjaiba, vagy a feldolgozó gépekbe egyaránt. Azokon a szakaszokon, ahol az adat utja pl.: telefonvonalakon keresztül halad, a bitsorozatok, karakterek, vagy karaktersorozatok analóg formában kerülnek átvitelre. A digitálisról analógra és az analógról digitálisra történő átalakitást a MODEM-ek végzik. A MODEM elnevezés a berendezés funkciójából ered a modulátor és demodulátor szavak néhány első betüjének összevonásából. A csomóponti processzor vezérlési, utvonalfelépitési és kapcsolási funkciókat lát el, mig a feldolgozó gépek a számitási erőforrásokat és a megfelelő nagy file rendszereket szolgáltatják.

A valóságban az adatátviteli csatornák zajosak, ezért elkerülhetetlen a hibakorlátozás alkalmazása. A csatornakapacitás elméleti maximumánál alacsonyabb átviteli sebesség mellett megvalósitható a tetszőlegesen kicsi hibavalószinüség, azaz a megbizható információátvitel a hibakorlátozás alkalmazásával. A 204. ábrán válasszunk ki egy tetszőleges vonalszakaszt és kövessük végig az adat utját az információ forrástól a vevőig.

Az adatcsatorna tulajdonságai miatt a különböző forrás kódokat (igy nevezzük a terminálokon, processzorokon és számitógépeken belül alkalmazott kódokat) nem célszerü változatlan formában a csatornára adni. Általában szükség van valamilyen hibakorlátozási módszer alkalmazására.



204. ábra. Az adatut egy lehetséges modellje

6.3.1 HIBAKORLÁTOZÁS

Alapvetően két tipikus csatorna hibáról beszélhetünk. A véletlen hibára az jellemző, hogy egy-egy bit hibás lesz az átvitel során és ez a meghibásodás egyenlő valószinüséggel bármikor bekövetkezhet. A hibacsomó (burst) olyan tipikus hibát jelent, amikor néhány egymás mellett levő bit csoportosan hibás lesz. Ebben az esetben a meghibásodás valószinüsége nem azonos az üzenetet alkotó összes bitre nézve. A véletlen hibák elsősorban az atmoszférikus zavarok következtében jelentkeznek, mig a hibacsomókat nagy százalékban a vezetékes rendszerek (pl.: telefonhálózat) zavarai okozzák.

A hibakorlátozás a gyakorlatban a hibakorlátozó kódok és hibakorlátozási eljárások alkalmazásával történik.

A hibakorlátozó kódolás a forráskódhoz meghatározott módon ellenőrző elemeket ad, igy a kapott kód redundánssá válik. A hibakorlátozó kódolás eredményeként az átvitel során keletkezett hibák felismerhetők lesznek, sőt bizonyos esetekben a hibafelismerésen tul a hibajavitás is lehetővé válik. Széles körben alkalmazzák a paritás ellenőrzést, az aránykódokat és a ciklikus ellenőrző kódokat. A paritás ellenőrzésnek több fajtája létezik, általában a két dimenziós paritás ellenőrzés terjedt el.

Az aránykódok olyan redundáns kódgenerálást jelentenek, amelyben pl. minden karakterben az egyesek száma állandó. A ciklikus ellenőrző kódok előállitása viszonylag egyszerü feladat. Megoldható visszacsatolt léptetőregiszter segitségével a hardware szintjén, vagy software-el egyetlen szubrutin segitségével is.

A hibakorlátozási eljárások közül az automatikus ismétléskérés terjedt el széles körben. Ez azt jelenti, hogy amennyiben a vevő hibát észlel az adatátvitel során, akkor automatikusan a hibás információblokk megismétlését kéri az adótól. Emlitést érdemel a közvetlen hibajavitás módszere is, amikor a vevő fél saját maga képes az átvitel során keletkezett hibák javitására, természetesen a forráskód megfelelő mértékü redundáns felhigitásából következően.

6.3.2 ANALÓG ÉS DIGITÁLIS ÁTVITELI ELJÁRÁSOK

A telefonhálózat tipikusan analóg csatornákat reprezentál, hiszen az emberi hang átvitelére tervezték. A mi esetünkben akkor beszélünk analóg átvitelről, ha a digitális információt annak tartalmi változása nélkül transzformáljuk időben folyamatos (analóg) jelekké. Itt az információt egy állandó frekvenciáju jel különböző amplitudói, vagy egy állandó amplitudóju jel különböző frekvenciái, vagy fázisai hordozzák. Igy három féle modulációs eljárásról beszélhetünk. Az amplitudómoduláció során az átvitt bitsorozat jelentésének megfelelően a telefonvonalon egy állandó frekvenciáju jel amplitudója változik, amit a vevőoldalon demodulálunk és visszakapjuk az eredeti bitsorozatot. Frekvenciamoduláció esetén a bitsorozat képe egy állandó amplitudóju, de a bitsorozat értelmének megfelelően változó frekvenciáju jel formájában él a vonalon. A vevőoldali demoduláció után ismét visszanyerjük az eredeti bitsorozatot. A fázismoduláció a legbonyolultabb talán a három módszer között. Itt a bináris információ egy állandó amplitudóju és frekvenciáju jel fázisváltozásaihoz van hozzárendelve. Pl. legegyszerübb esetben a nulla és 180°-os fázisok reprezentálhatják bitsorozat egyeseit, illetve nulláit.

Itt célszerü megjegyezni, hogy az információátvitel sebességét nemcsak bit/sec-ben mérik, hanem létezik egy olyan egység is (Baud), amely az analóg vonalakon másodpercenként lezajló állapotváltozásokat reprezentálja. A bit/sec és a Baud akkor nem egyenlő egymással, ha például fázismodulációval négy, nyolc, vagy esetleg több állapotot realizálunk egy periódus alatt, s igy azok leirásához kettő, három, esetleg több bitre van szükség. Amennyiben például 45°-os fázisváltozás jelent egy uj állapotot, akkor egy periódus alatt nyolc állapotváltozás történik, amelynek leirásához ($2^3 = 8$) három bit szükséges. Igy itt a bit/sec-ban kifejezett sebesség háromszorosa a Baud-ban mért értéknek. ($C_{bit/sec} = \frac{1}{3} C_{Baud}$, ahol C a csatorna átviteli sebessége.)

A modulációs és demodulációs eljárásokat megvalósitó MODEM-eket két nagy csoportra oszthatjuk. Léteznek aszinkron és szinkron MODEM-ek. Az aszinkron MODEM-ek karaktereket véletlenszerüen továbbitanak, pl. olyan ritmusban, ahogy a felhasználó billentyüz a terminálon, ezért a vevő oldalon is véletlenszerüen érkeznek meg a karakterek. Az átvitel idejére a karaktereket felruházzák start és stop bitekkel. Ezek a járulékos bitek oldják meg az adó fél modulátora és a vevő fél modulátora közötti együttfutást a véletlenszerüen bekövetkező átvitel alatt. A szinkron MODEM-ek bitsorozatokat, karaktersorozatokat (blokkokat) továbbitanak állandó sebességgel. Ennek megfelelően szinronizálásra van szükség, amit ugy oldanak meg, hogy a rögzitett átviteli sebességet belső órajelekkel vezérlik. Ezen tulmenően előre meghatározott (ismert) karaktereket, vagy bitsorozatokat helyeznek el az átvitelre

szánt szövegben. Természetesen itt nincs szükség a start és stop bitek alkalmazására, viszont elengedhetetlenül szükséges a már szabványositott adatátviteli vezérlési eljárások alkalmazása.

Tipikus aszinkron MODEM átviteli sebességek: 110, 134,5, 150, 300, és 1200 bit/sec. Tipikus szinkron tipusu átviteli sebességek: 2000, 2400, 4800, 7200, 9600 bit/sec.

A digitális átvitel során a digitális információ közvetlenül rákerül a vonalakra MODEM-ek alkalmazása nélkül. Ilyen vonalakon, tehát egyenáramu átvitellel dolgoznak a hagyományos távíró- és telexgépek is. Ezt a közvetlen, bináris formában történő átvitelt alapsávi digitális átvitelnek nevezzük. Nagyobb sebességkategóriákban is létezik digitális átvitel. Ezek a rendszerek azonban már pulzus kód modulációval dolgoznak mintavételezés alapján.

6.3.3 AZ ADATÁTVITEL VEZÉRLÉSI ELJÁRÁSAI

A hagyományos számitógépes hálózatokban lényegében a karakter- és bitorientált adatátviteli vezérlési módok terjedtek el. A karakter-orientált vezérlési eljárás lehet aszinkron és szinkron, mig a bitorientált vezérlési mód csak szinkron. A különböző tipusu adatátviteli vezérlési eljárásokra azért van szükség, mert az esetek legnagyobb részében csupán két vezeték áll rendelkezésre a vezérlés és a hasznos információ továbbitására. Tehát elkerülhetetlenné vált bizonyos konvenciók bevezetése. Ma már széles körben alkalmazott, sőt szabványositott adatátviteli eljárásokkal rendelkezünk. Az átvitel lebonyolitására négy szakasz jellemző. A kapcsolat felépitése az adó és vevő fél, vagy felek között, a továbbitásra szánt üzenet átvitele, a hasznos információ átvitelének befejezése és végül a kapcsolat bontása a felek között. A karakter orientált vezérlési eljárások közül a szinkron vezérlési mód a kiforrottabb. Az adatátvitel vezérlése a felek között a vezérlőkarakterekkel lefolytatott beszélgetés alapján történik. A leggyakrabban használt vezérlő karakterek az alábbiak:

SOH (start of heading) = fejléc kezdete
STX (start of text) = szöveg kezdete
ETX (end of text) = szöveg vége
ETB (end of text block) = blokk vége
EOT (end of transmission) = adás vége
ENQ (enquire) = lekérdezés
ACK (acknowledge) = pozitiv nyugtázás
NAK (negative acknowledge)= nemleges válasz
SYN (synchronous idle) = szinkronizálás
DLE (data link escape) = vezérlés átkapcsolás

A vezérlőkarakterek segitségével lefolytatott párbeszédet a software vezérli. Az átvitel üzenetekre és üzenetblokkokra tagozódik. Egy-egy blokk átvitele a "fejléc kezdete" vezérlő karakterrel indul. A fejléc tartalmazza az adó és vevő állomás azonositóját, a cimzett terminál számát és prioritását, a felhasználásra vonatkozó utasitásokat, az üzenet, vagy üzenetblokk sorszámát, stb. A fejlécet követi a "szöveg kezdete" vezérlő karakter, majd ezután az átvitelre szánt hasznos információ (szöveg, adat) következik. Ezt zárja le a "blokk vége", vagy a "szöveg vége" vezérlő karakter. Ezek után az adó fél várja a vevő részéről nyugtázást, hogy igy meggyőződhessen az előzőleg átküldött üzenet helyes vagy hibás vételéről. Hiba esetén automatikus ismétlésre kerül sor.

A karakter-orientált adatátviteli vezérlési eljárások az alábbi hátrányos tulajdonságokkal rendelkeznek:

- az összes eljárás a vezérlőkarakterekre épitett párbeszéden alapul,
- viszonylag rövid üzenetek esetén a hasznos információ átvitelére és a párbeszédekre forditott időarányok kedvezőtlenek,
- az egyes eljárások azonos kódstrukturát követelnek a teljes üzenetre nézve,
- a hibakorlátozás nem egyenletes a vezérlő karakterekre és a hasznos információra vonatkozóan.

A fenti hátrányok feloldására fejlesztették ki a bitorientált szinkron adatátviteli eljárásokat. Ezek közül a legismertebb az IBM által bevezetett SDLC (Synchronous Data Link Control) és az ISO által szabványositott HDLC (High Level Data Link Control). Közös jellemzőjük, hogy a kódfüggetlen (tehát bitsorozatnak tekintett) változó hosszuságu hasznos információt egy szabványos formátumu keretbe foglalják. A keretet két azonos nyolcbites kódszó képezi. Közöttük foglal helyet sorrendben a nyolcbites cimmező, a nyolcbites parancsmező, a továbbitásra szánt változó hosszuságu bitsorozat és a tizenhat bit hosszuságu ellenőrző szó. Az eljárás nagy előnye a kódfüggetlenség, valamint az, hogy a keretbe foglalt vezérlő és hasznos információ azonos szintü hibakorlátozás alá került. Előnyösen használható mind a kötegelt, mind az interaktiv tipusu feldolgozásoknál.

6.3.4 AZ ADATCSATORNÁK KIHASZNÁLÁSA

Az adatcsatornák fizikai megvalósitására vezetékes, vagy vezeték nélküli rendszereket használunk. Gyakran találkozunk közvetlen megfeleltetéssel, azonban nagyszámu felhasználó esetében célszerü az adatátviteli közeg jobb kihasználását biztositó módszerek valamelyikét alkalmazni. Az adatátviteli közeg lehető legjobb kihasználására kell törekedni elsősorban azért, mert a hálózati feldolgozások esetében a teljes költség jelentős hányadát a kommunikációs költségkomponens teszi ki. Leegyszerüsitve a kérdést azt mondhatjuk, hogy a felhasználó fizet a gépidőért, amit a számitógépen futó programjai használtak fel és az adatcsatorna használatáért. Az utóbbi komponens rendkivül gyengén kihasznált adatcsatorna esetén akár nagyobb is lehet, mint a számitási költséghányad. A két terület intenziv fejlesztésének eredményeként mind a két költséghányad csökkenő tendenciát mutat, azonban a fizikai gépek rohamos fejlődése következtében a számitási költségek csökkenése gyorsabb mint a kommunikációs költségkomponenseké.

Az adatcsatorna kihasználásának módszerei lehetővé teszik, hogy egyetlen adatcsatornát több felhasználópár vehessen igénybe valamilyen előre meghatározott stratégia alapján. Különböző stratégiák alakithatók ki a térosztás, időosztás, frekvenciaosztás, lekérdezés és a véletlen hozzáférésü adatcsatorna alkalmazásával. Az első négy módszer vezetékes rendszerekben terjedt el és megvalósitásuk költséges berendezéseket igényel. A térosztáshoz vonalkapcsoló központ, az időosztáshoz idő multiplexer, a frekvenciaosztáshoz frekvenciamultiplexer, a lekérdezés vezérlésének megvalósitásához pedig számitógép szükséges. A gyakorlatban a különböző adatcsatorna kihasználási módszerek több szinten egymásra épülnek, például térosztás - frekvenciaosztás, vagy térosztás - lekérdezés, stb.

Uj fogalmakkal találkoztunk, amelyek rövid magyarázatára szükség van. A térosztásos stratégia legegyszerübb realizálását például a nyilvános telefonhálózatot vezérlő telefonközpontok jelenthetik.

Tehát a felhasználók tárcsázással a telefonközpontokon keresztül hozzáférhetnek egy-egy számitógéphez és forditva.

Az időosztásos módszer megvalósitásához időmultiplexer szükséges. Ez olyan hardware berendezés, amelynek működése hasonlit a központi gép multiplex csatornája szervezésénél megismert módszerhez. Pontosan arról van szó, hogy egy nagyobb sebességü adatcsatornát több alacsony sebességü adatcsatornára oszt fel időszeletek segitségével. A nagyobb sebességü csatorna kapacitása egyenlő az alacsonyabb sebességü csatornák kapacitásának összegével. Két fajta időmultiplexelést alkalmaznak a gyakorlatban, elterjedtebb a szinkron időosztásos multiplexelés az aszinkronhoz képest. Ez utóbbi jobb vonalkihasználást eredményez, azonban lényegesen bonyolultabb és drágább a szinkron változatnál.

A frekvenciaosztás megvalósitásához frekvenciamultiplexerre van szükség. A frekvenciamultiplexer ugy osztja fel az adatcsatornát több alacsonyabb sebességü elemi adatcsatornára, hogy a fizikailag egy vezetékpárt használó felhasználókhoz külön-külön egy-egy frekvenciapárt jelöl ki. Ezzel a módszerrel egy telefonvezetékpáron akár tizenkét, esetleg huszonnégy alacsony sebességü felhasználó is kiszolgálható teljesen egyidőben. Gyakran találkozunk vivőfrekvenciás rendszerekkel a közepes és nagysebességü adatcsatornák frekvenciaosztásos használatánál. Itt egy az egyes felhasználókhoz hozzárendelt alapfrekvenciát (vivőfrekvenciát) modulálnak a digitális információt reprezentáló frekvenciapárral. A frekvenciaosztásos rendszereket széles körben alkalmazzák. Mind az időosztásos, mind pedig a frekvenciaosztásos rendszerekre jellemző, hogy a terminálvezérlő egységek viszonylag bonyolultak és drágák, mindkét esetben hardware multiplexerre van szükség, a felhasználók azonositása azonban egyszerü, valamint mindkét esetben a nem aktiv felhasználók is lefoglalják a számukra kiosztott elemi adatcsatornát.

A lekérdezésen alapuló adatcsatorna kihasználási módszer olyan (általában közepes, vagy nagyobb sebességü) adatcsatorna forgalmának vezérlésére szolgál, amely mentén a felhasználók termináljai felfüzéses rendszerben működnek. Ez azt jelenti, hogy fizikailag ugyanarra a vonalra több terminál csatlakozik. Minden felhasználóhoz egy cimet (azonositót) rendelnek hozzá. A vezérállomás valamilyen algoritmus alapján sorra lekérdezi az egyes felhasználókat, hogy van-e valamilyen továbbitásra váró üzenetük. Amennyiben nincs, ugy lekérdezi a következő felhasználót, ha viszont egy-egy üzenet átvitelére igény van, akkor azt le is bonyolitja. A vezérlés software szinten történik és ez nagy előny az előző vonalkihasználási eljárásokhoz képest, azonban a lekérdezés némi veszteséget jelent, hiszen azokra a terminálokra is vonatkozik, amelyek nem kivánnak átvitelt kezdeményezni.

Az utóbbi években előszeretettel alkalmazzák a véletlen hozzáférésü csatornákat, szinte kizárólag nem vezetékes hálózatokban. Ez a módszer elsősorban rövid üzenetek által generált forgalom hatékony lebonyolitását oldja meg. Előnye, többek között, hogy nem igényel hardware multiplexert, a terminálvezérlő egységek egyszerüek és nem költségesek, a felhasználók száma több száz is lehet, a hálózat bővithető stb. Az ilyen tipusu hálózatokban a felhasználót semmiféle szinkronizmus nem korlátozza időben, az adott földrajzi környezeten belül pedig még a felhasználók helye sincs rögzitve. A fenti adatcsatorna kihasználási módszereken tul számitógépes vezérlést igénylő további két eljárást érdemes megemliteni. Az adatkoncentrálás és a vonalkoncentrálás általában kisszámitógépekkel (csomóponti gépekkel) megvalósitott eljárás, ahol kihasználják a forgalom statisztikus jellegét, igy ezek forgalomkiegyenlitő hatásuak, ellentétben a multiplexerekkel.

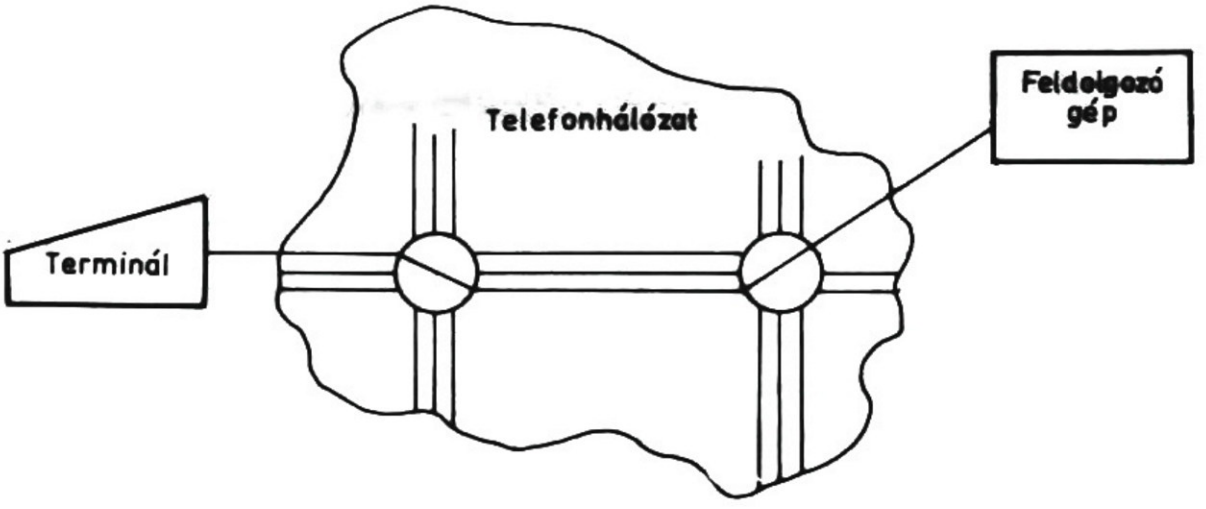
6.4 SZÁMÍTÓGÉPES HÁLÓZATOK TÍPUSAI

Átfogó kép alkotáshoz elegendő három tipikus hálózatépitési módot bemutatni, röviden jellemezni. A hagyományos hálózatok, vagy másnéven közvetlen kapcsolaton alapuló rendszerek fejlesztése még a hatvanas évek elején indult meg. A csomagkapcsolás alapján müködő hálózatok a hatvanas évek végére születtek meg, mig a csomagrádiózás elvén müködő hálózatok a hetvenes évek fejlesztéseként jöttek létre. Mindhárom épitési mód tipikus jellemzőkkel rendelkezik, rövid bemutatásuk és összehasonlitásuk segitséget nyujt az esetleges további tájékozódáshoz.

6.4.1 HAGYOMÁNYOS SZÁMÍTÓGÉPES HÁLÓZATOK

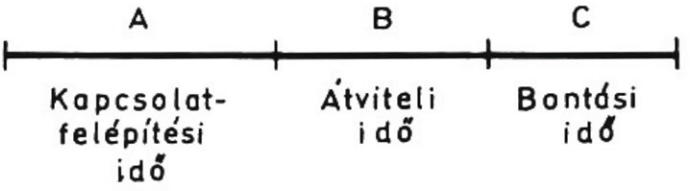
A hagyományos épitésü számitógépes hálózatokra jellemző, hogy ugy létesitenek kapcsolatot a felhasználók között, hogy közvetlen áramköri kapcsolat jön létre az adó és vevő állomás között. A közvetlen áramköri kapcsolat kiépitése során megtaláljuk a vezetékes rendszereknél alkalmazott adatcsatorna kihasználási módszereket is. A közvetlen kapcsolaton alapuló rendszerek esetében a legtipikusabb a térosztásos vonalkihasználási séma. Általában ezt még kombinálják a többi szokásos vonalkihasználási módszerrel. A térosztás lényegében vonalkapcsolást jelent, ugyanolyan módon, mint ahogy azt a telefonok használata esetében megszoktuk. Igy a hagyományos hálózatok esetében a jelen keretek között elegendő a vonalkapcsolt rendszerek ismérveit tisztázni. A 205. ábrán látható a vonalkapcsolás elve.

A vonalkapcsolt hálózatokban a hivott fél cimét (azonositóját) a kapcsolatfelépités során közölni kell a hálózattal. Ezt általában a tárcsázással végzik el. A számitógép software-rel utánozza a tárcsázás müveletét. Látható, hogy a kapcsolatfelépitést bármelyik felhasználó kezdeményeheti. A közvetlen kapcsolat felépitésétől kezdve az információáramlás ezen a rögzitett utvonalon folyik. Az egyes központok által összekapcsolt vonalszakaszok az átvitel teljes időtartamára foglaltakká válnak más felhasználók számára. A teljes utvonalat alkotó vonalszakaszok egyidejüleg vesznek részt az átvitel lebonyolitásában. Az ilyen közvetlen áramköri szintü kapcsolat minden esetben átlátszó, tehát sem tárolási,



205. ábra. A vonalkapcsolás elve

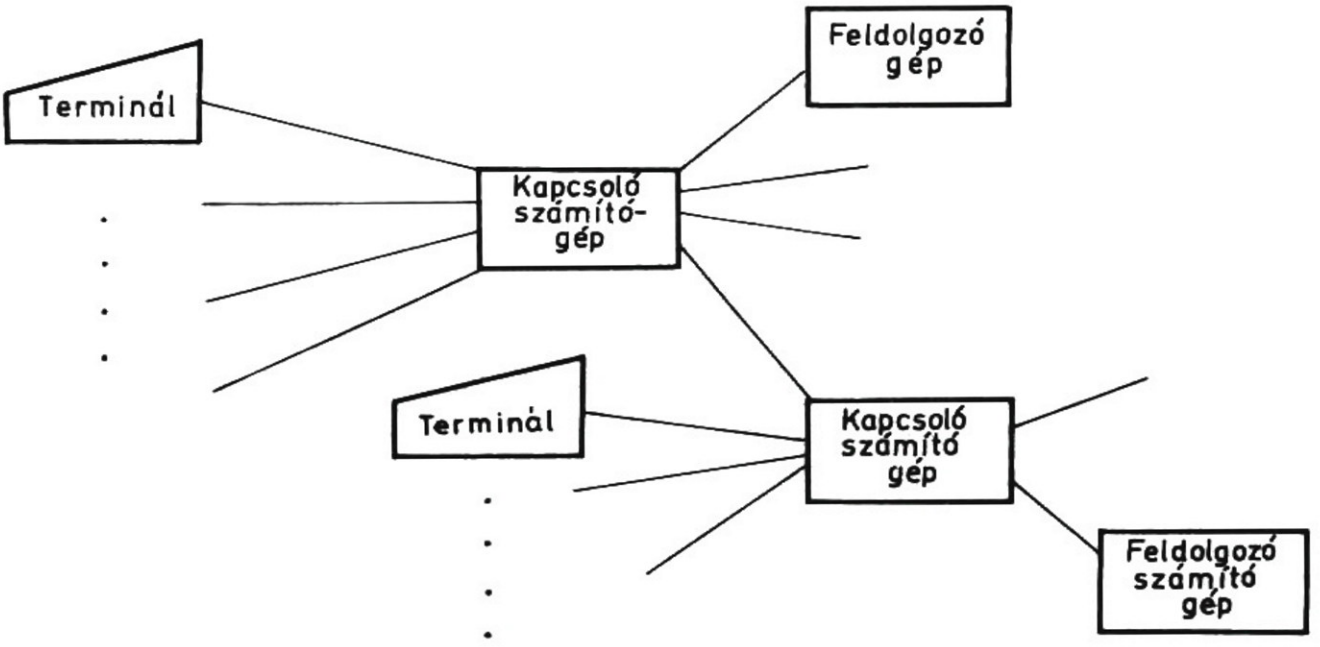
sem pedig kódkonvertálási funkciókat nem tud megoldani. A vonalkapcsolás klasszikus változata nem adaptiv, ami azt jelenti, hogy nem képes alkalmazkodni a hálózat időben változó állapotához, tehát pl. nem tud kikerülni meghibásodott vonalszakaszokat. Az alkalmazott hibakorlátozási stratégiákat a két végállomás (az adó és vevő fél) valósitja meg a vonal két végpontja között. Érdemes szakaszokra bontani és külön-külön vizsgálni a teljes átvitelt (206. ábra).



206. ábra. A teljes átvitel összetevői

Az első szakasz a kapcsolásfelépitési időt reprezentálja. Ez néhány másodperctől kezdve akár fél óráig is terjedhet a hálózat állapotától függően. Tapasztalhatjuk a fővároson belül is, hogy a különböző irányokban és napszakokban a telefonbeszélgetések kapcsolási ideje sem egyforma. A középső szakasz azt az adásidőt jelenti, amelynek csupán egy töredék része szolgál a továbbitásra szánt hasznos információ átvitelére, az adatátvitel vezérléséhez általában nélkülözhetetlen párbeszédes funkciók miatt. A fix utvonal mentén, ha az már egyszer felépült, nincs közbülső várakozás, mindössze a terjedési idő növeli az adásidőt, azonban a terjedési idő itt még elhanyagolható. A kapcsolat bontása általában néhány másodperc alatt megtörténik.

A vonalkapcsolásra épülő számitógépes hálózatok jellegzetes alkalmazási területe a hosszu üzenetekből álló és lehetőleg szimmetrikus forgalom lebonyolitása. A telefonvonalak paramétereiből adódóan alacsony és közepes sebességü adatcsatornák megvalósitására nyujt lehetőséget. A vonalkapcsolt hálózatok fejlettebb változatát jelenti a virtuális vonalkapcsolás. Itt a kapcsolási funkciót csomóponti számitógépek látják el (207. ábra).



207. ábra. A virtuális vonalkapcsolás elve

A kapcsolóközpont funkcióinak számitógépes megvalósitása lehetővé teszi az adaptivitást, miután az egyes utvonalak logikai döntések eredményeként épülnek fel. A hivott fél azonositóját a hivó fél tárcsázással, vagy más módon pl. billentyüzéssel közli a kapcsolószámitógépekkel. Az utvonal felépitése után lényegében a klasszikus vonalkapcsolásra vonatkozó időviszonyok érvényesek itt is. Tehát egy-egy utvonal az adatátvitel lebonyolitása után itt is szétbomlik. A virtuális vonalkapcsolás további előnyét jelenti, hogy a csomóponti kisszámitógépek egyuttal a lekérdezés vezérlési funkciókat is el tudják látni, igy a két vonalkihasználási módszer igen könnyen hasznositható olyan vonalak esetében is, ahol több felhasználó felfüzéses részhálózatot képez egyetlen vonalon.

Azokat a rendszereket, ahol általában egy-két feldolgozó számitógéphez egy kisszámitógépen keresztül lehet hozzáférni, front-end rendszereknek nevezzük. Ezek a kisszámitógépek ujabban átveszik a feldolgozó gépektől az adatátvitel összes vezérlő funkcióját. Az ilyen hálózatokban gyakran több szintü multiplexelést is alkalmaznak.

A front-end rendszerek előnye, hogy tehermentesitik a feldolgozó gépeket, hátrányuk, hogy általában a feldolgozó gép már meglevő operációs rendszerét meg kell változtatni.

6.4.2 A CSOMAGKAPCSOLÁS ELVÉN MŰKÖDŐ HÁLÓZATOK

A virtuális vonalkapcsolás után ujabb előrelépést jelentettek az üzenetkapcsolás elvét felhasználó hálózattipusok. Ezek annyiban különböznek a virtuális vonalkapcsolástól, hogy a kapcsoló számitógépek számára nem tárcsázás segitségével juttatják el a hivott fél azonositóját, hanem az üzenetet megelőző fejléc tartalmazza ezt, többek között a hivó fél azonositójával együtt. Egy-egy üzenet lényegében tetszőleges hosszuságu lehet, tehát nincs semmiféle megkötés a továbbitott üzenet felépitésére vonatkozóan. A feldolgozógépek és a felhasználók termináljai különböző hosszuságu üzeneteket küldenek egy kisszámitógépekből álló aktiv adatátviteli hálózatba, amely az üzeneteken semmiféle módositást nem végezve adaptiv jelleggel optimális utvonalon továbbitja azokat rendeltetési helyükre. A különböző hosszuságu üzenetek optimális célbajuttatása általában más-más stratégiát igényel, ezért az üzenetkapcsolás racionalizáltabb válfaja terjedt el, a csomagkapcsolás (208. ábra).

Ennek a technikának az az alapötlete, hogy feldolgozó gépek és az aktiv adatátviteli hálózat között az üzenetkapcsolás elve érvényesül, az adatátviteli hálózaton belül azonban a változó hosszuságu üzenetek az adott rendszeren belüli szabványos méretü csomagokra vannak szabdalva. Igy az adatátviteli hálózat az átvitel során kizárólag jól kezelhető, általában azonos méretü csomagokkal foglalkozik. A 209. ábrán látható a csomagok felépitése.

Az adatátviteli hálózat csomóponti processzorai mindig csak a szomszédos csomópontok egyikéhez kötelesek eljuttatni a továbbitásra váró csomagokat. A szomszédos csomópontok közül a célállomás figyelembevételével kiforrott stratégiák segitségével történik az optimális csomópont kiválasztása. Az adaptivitás és az optimális döntéselőkészités alapja az, hogy meghatározott időközönként, pl. néhány másodpercenként az összes csomópont tájékoztatja az összes szomszédját pillanatnyi állapotáról. Ez a kölcsönös tájékoztatás többek között tartalmazza az egy-egy adott csomópontból az összes lehetséges kimenet irányában várakozó csomagok számát, várakozási idejét, stb. Ennek alapján minden csomópont felfrissiti saját késleltetési tábláját, ebből kikeresi a minimál késleltetési vektort és ennek felhaszná-

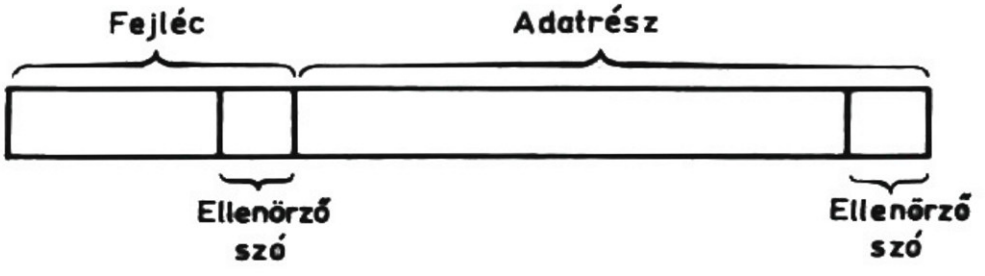
CS.P. = Csomóponti processzor

Feldolgozó gép

CS. P.

Terminál

208. ábra. A csomagkapcsolás elve



209. ábra. A csomagok felépítése

lásával kapcsoló vektort generál a továbbitásra váró csomaghoz, amelynek alapján a csomag a számára az adott pillanatban legkedvezőbb kimenő irány várakozási sorába kerül.

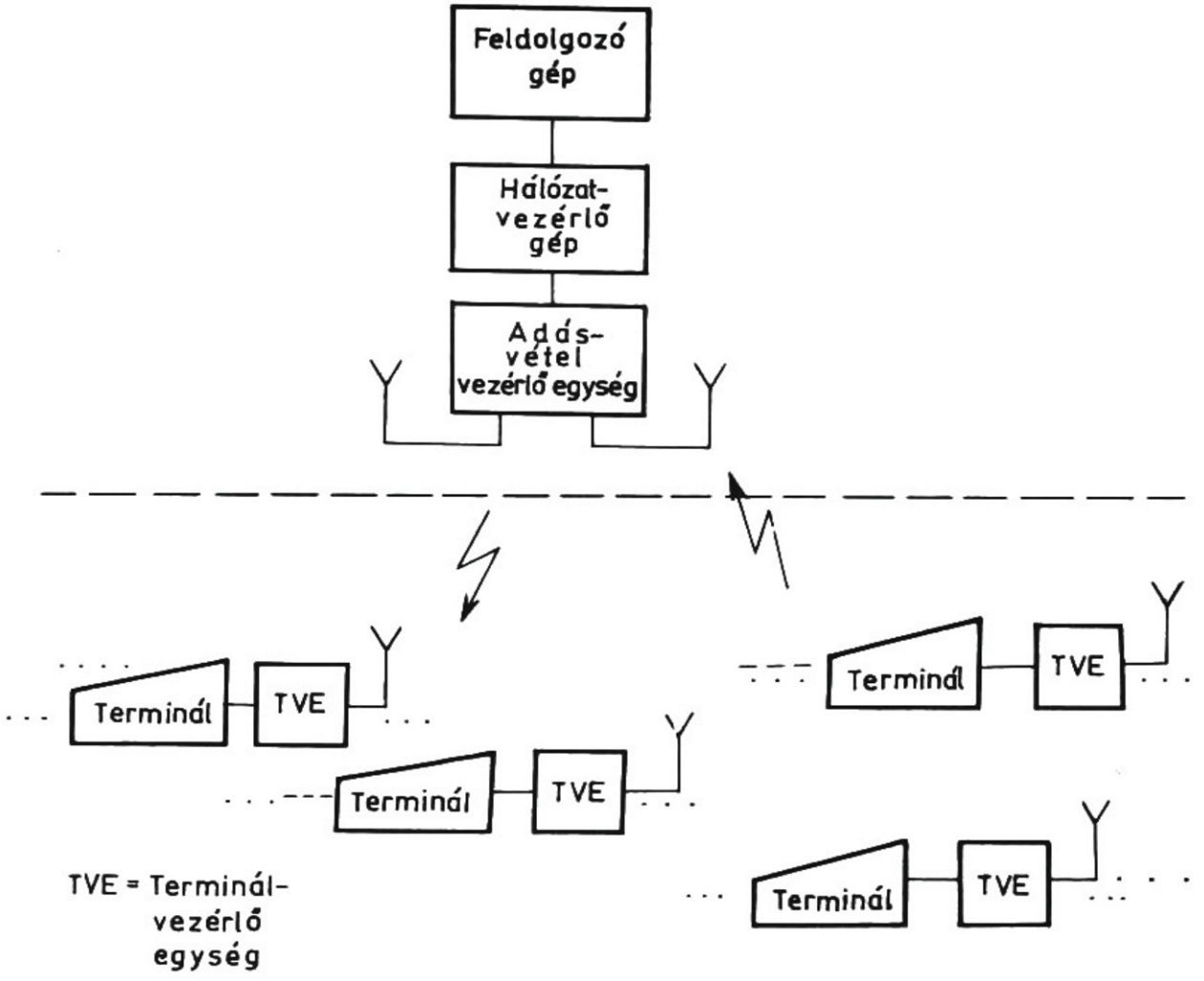
Összefoglalva ezt a néhány gondolatot, a csomagkapcsolt rendszerekben a csomagok csomópontról csomópontra haladnak a célállomás felé, az egyes csomóponti processzorok addig tárolják a csomagokat, amig azok hibátlan megérkezéséről megkapják a visszajelzést a szomszédos csomóponti processzortól. A teljes adatátvitel lebonyolitása, a csomagok képzése, hibavédelme, valamint szinkronizálása és ujra összerakása az aktiv adatátviteli hálózat feladata. Az üzenetcsomagot küldő és fogadó felhasználó közvetett kapcsolatban van egymással, de sokkal közvetlenebbnek érzi a kapcsolatot, mint a vonalkapcsolt rendszerekben. A válaszidő általában fél másodperc alatt marad többezer kilométeres távolságok esetén is. Az egyes csomópontok összekötésére nagysebességü bérelt vonalakat használnak.

A csomagkapcsolt hálózatok már valóban alkalmasak igen nagy és akár burst jellegü forgalom lebonyolitására is. A számitástechnikai szempontból legfejlettebb országokban müködik néhány csomagkapcsolt hálózat. Ezek általában több országot, sőt világrészeket fognak át, leterhelésük ma még nem megoldott.

6.4.3 A CSOMAGRÁDIÓZÁSON ALAPULÓ SZÁMÍTÓGÉPES HÁLÓZATOK

Az utóbbi években előtérbe került a nem vezetékes közeg felhasználása digitális átvitel céljára, ennek több oka is van. A vezetékes rendszerekben, elsősorban a hagyományos hálózatok üzemeltetésével szerzett tapasztalatok bebizonyitották, hogy az adatátviteli költséghányad csökkentése igen nehéz feladat a számitási költséghányad gyors csökkenésével szemben. Ez azt jelenti, hogy egy távoli felhasználó feldolgozási költségeiben évről évre növekszik az adatátviteli költségek hányada. A vezetékes rendszerek bővitése nehézkes. Egy ujabb terminál beállitásához ujabb vonalra, esetleg még további multiplexerre is szükség lehet. A vonalak minősége nem minden esetben megfelelő. A közepes sebességkategóriákra (600-2400 bit/sec) nálunk például a kapcsolt távbeszélő hálózatra, illetve a bérelt közvetlen távbeszélő tipusu összeköttetésekre 10^{-4} várható bithibaarányt jelöl meg a Posta Központi Távíró Hivatal az "Adatátviteli lehetőségek a Magyar Posta Távközlési hálózatain" c. 1974-es kiadványában. A nem vezetékes digitális rendszerek egyik nagy előnye, hogy nem igényelnek előre kiépitett adatátviteli hálózatot, azaz olyan helyen is alkalmazhatók, ahol nincs kiépitett telefonhálózat, vagy a meglevő hálózat már tulterhelt, vagy esetleg nem megfelelő minőségü. A már beszédátvitellel tulterhelt telefonhálózatot nem célszerü terminálok üzemeltetésével tovább terhelni, hiszen egy-egy terminál több tiz, esetleg száz telefonbeszélgetésnek megfelelő forgalmával csak tovább növeli a tulterhelés mértékét. Azt sem lehet persze mondani, hogy az iparilag fejlett országokban a már különböző, elsősorban beszédátviteli és televiziós célokra lefoglalt néhányszáz MHz-es frekvenciatartományban tul sok szabad hely maradt volna a digitális rádiózás céljára. Ennek ellenére a számitástechnikai szempontból legfejlettebb országokban a számitástechnikai szolgáltatások valóban közmüszerü, sőt helyfüggetlen elosztását a digitális rádiózástól várják. Igy jelentős kutatás-fejlesztés projekteket tüztek ki a terület felfuttatására, természetesen a távközlési müholdak ilyen célu alkalmazását is belevonva (210. ábra).

A csomagrádiózás elnevezés onnan adódik, hogy ezek a hálózatok is a csomagkapcsolt rendszerek esetében jól bevált csomagokat kezelik, azonban nem vezetékes adatátviteli közegben. Jól szétválaszt-



210. ábra. A csomagrádiózáson alapuló hálózat felépitése

ják a feldolgozó gép–terminálok és a terminálok–feldolgozó gép irányu forgalmat. A két forgalmi irány teljesen eltérő paraméterekkel rendelkezik. A feldolgozó gép–terminálok irányu forgalom lebonyolitása rendkivül egyszerü, hiszen az üzenetek sorbaállithatók. A terminálok–feldolgozó gép irány már sokkal körültekintőbb megoldást igényel. A véletlen hozzáférésü csatornán keresztül a terminálok minden szinkronizmus nélkül bármikor küldhetnek csomagokat. A minden szinkronizmus nélkül inditott csomagok esetleges ütközését a hálózatvezérlő kisszámitógép felismeri. Ilyen esetben nem küld nyugtázó csomagot az ütköző felhasználók felé. A terminálok egy időszelet lejárta után véletlenszerüen előállitott késleltetésselujrainditják csomagjaikat. A sikeres átvitelt a nyugtázás megérkezése jelenti. Ilyen stratégiával az adatcsatorna kapacitása az elméleti maximum mintegy egyhatodára lecsökken, azonban igy is pl. egy 9,6 Kbit/sec-os csatorna megoldja többszáz alfanumerikus display terminál kezelését egyidőben (szimultán). A terminálvezérlő egységek mikroprocesszorokra épitve olcsón előállithatók, egy elosztott intelligenciáju hálózat alakitható ki, a terminálok nem helyhez kötöttek, az adatátvitelt vezérlő software egyszerü, igy viszonylag olcsó és igen megbizható hálózat kiépitése válik lehetővé. További előny, hogy egy-egy ujabb terminál beállitása a rendszerbe a terminál és a hozzátartozó (az alfanumerikus display árának megfelelő) terminálvezérlőegység megvásárlását jelenti mindössze. Ez azt jelenti, hogy a hálózat rendkivül rugalmasan bővithető. A véletlen hozzáférésü csatorna a távközlési müholdak felhasználása esetén - tehát nemcsak egy-egy szük földrajzi környezetben - is alkalmazható.

Irodalom

Abramson-Kuo: Computer-Communication Networks (Prentice-Hall, 1973)

Antoni A. - Brückner H. - Dobrovolni T. - Vásony S.: Számitógépismeretek. SZÁMOK, Bp. 1972.

Antoni A. - Frigyes L. - Vásony S.: Mágnesszalagos, mágneslemezes, mágnesdobos perifériák. SZÁMOK, Bp. 1973.

Balogh I. - Hermán J. - Kenesi B. - Tuba P.: Számitógépek digitális áramkörei. SZÁMOK, Bp. 1976.

Brückner H. - Dobrovolni T. - Lohonyai M.: Display, rajzgép, bizonylatolvasó, konzolirógép, lyukszalagos perifériák. SZÁMOK, Bp. 1975.

Csomor Gy. - Limperger I. - Lohonyai M.: Távadatfeldolgozás alapjai. SZÁMOK, Bp. 1975.

Electrostatic Printers and Plotters. Versatic, 1974.

ESZR/DOS Mágnesszalag cimkék. Programozói kézikönyv. E10.132.018D1 NOTO-OSZV, Bp. 1975.

Esztó Z. - Dr. Halassy B. - Krupa P. - Krupa Pné- Dr. Mészáros Tamásné - Tollár I.: Számitógépes információ rendszerek tervezési és módszertani eszközei. SZÁMOK, Bp. 1975.

Frank, H. - Kahn, R.E. and Kleinrock, L.: Computer Communication Network Design-Experience with Theory and Practice, Spring Joint Computer Conf. AFIPS Conf. Proc. (May 1972,) 255-270.

Gergely Cs. - Gyarmati P. - Hujber E. - Bakos L.: A számitástechnika legujabb eredményei 2. Statisztikai Kiadó Vállalat, Bp. 1976.

Hujber E. - Tomcsányi Gy. - Zentai T.: Adatrögzités, távadatátvitel, adatfeldolgozási módok. Közgazdasági és Jogi Könyvkiadó, Bp. 1974.

Introduction to IBM Data Processing Systems. Student Text, GC20-1684-3.

Krajcsovics M.: Elektronikus adatfeldolgozási rendszerek. Müszaki Könyvkiadó, Bp. 1975.

Dr. Mészáros Tné - Morvay J. - Weidl L. - Zentai T.: Információrendszerek számitógépes adattárainak tervezése és kezelése. SZÁMOK, Bp. 1975.

Dr. Nagykálnai E.: A vállalati rendszerszervezés gyakorlata. Statisztikai Kiadó Vállalat, Bp. 1975.

R20 - Az ESZ-1020 tipusu elektronikus számitógép általános müszaki leirása. NOTO - OSZV, Bp. 1973.

Schleuder, G.: Periphere Geräte in der Datenverarbeitung. Carl Hanser Verlag, München, 1972.

Shannon, C.E.: The Mathematical Theory of Communication, Bell. Syst. Tech. J., (July and Oct. 1948.).

Trede, W. - Herkelmann, A. - Schwarzer, E.: Számitástechnikai alapismeretek. Müszaki Könyvkiadó, Bp. 1975.

T. Dobrovolni - M. Herboly - P. Tuba

COMPUTER SYSTEMS ARCHITECTURE

The main purpose of this publication is to support computer education courses in the CSO International Computer Education and Information Centre. It provides an overall view of the computers by introducing the principles of UCS, utilizing our training experience of several years. The computer is treated as a complex automaton. The concepts of physical and abstract machines are introduced by the textbook. Because of presenting a technical knowledge, related to computer systems, the material does not include the creation of abstract machines (the programming). In addition to the demonstration of the physical machine, the interface between physical and abstract machines (operating system) is discussed. Also computer network problems are treated in a comprehensive, user-oriented way.

KÖZPONTI STATISZTIKAI HIVATAL

NEMZETKÖZI SZÁMÍTÁSTECHNIKAI OKTATÓ ÉS TÁJÉKOZTATÓ KÖZPONT

kiadásában megjelent rendszertechnikai jegyzetek

Pétervári L. B.:
A MIKROMÁSOLÁS ALKALMAZÁSA AZ INFORMÁCIÓFELDOLGOZÁSBAN 80,–Ft

Hunyadyné–Ilcsik–Klatsmányi:
AZ IRODASZERVEZÉS SEGÉDESZKÖZEI 50,–Ft

műszaki jegyzetek

Billingné–Botka–Gáspár–Hermán–Kenesi:
A KÖZPONTI EGYSÉG (PROCESSZOR) 80,–Ft

Csomor–Limperger–Lohonyai:
KÁRTYAOLVASÓ, KÁRTYALYUKASZTÓ, SORNYOMTATÓ PERIFÉRIÁK 60,–Ft

Antoni–Frigyes–Vásony:
MÁGNESSZALAGOS, MÁGNESDOBOS, MÁGNESLEMEZES PERIFÉRIÁK 60,–Ft

Brückner–Dobrovolni–Lohonyai:
DISPLAY, RAJZGÉP, BIZONYLATOLVASÓ, KONZOLÍRÓGÉP, LYUKSZALAGOLVASÓ PERIFÉRIÁK 70,–Ft

Balogh–Hermán–Kenesi–Tuba:
SZÁMÍTÓGÉPEK DIGITÁLIS ÁRAMKÖREI 60,–Ft

Csomor–Limperger–Lohonyai:
A TÁVADATFELDOLGOZÁS ALAPJAI 60,–Ft

szótárak

A VÁLLALATIRÁNYÍTÁSI SZÁMÍTÓGÉP-ALKALMAZÁS FOGALMAINAK ÉRTELMEZŐ SZÓTÁRA 80,–Ft

A VÁLLALATIRÁNYÍTÁSI SZÁMÍTÓGÉP-ALKALMAZÁS FOGALMAINAK TÖBB NYELVŰ SZÓTÁRA 80,–Ft

A jegyzetek megvásárolhatók

BUDAPEST: SZÁMOK Könyvesbolt, 1115 Budapest XI., Szakasits Á. út 68. • SKV Statisztikai és Számítástechnikai Könyvesbolt, 1024 Budapest II., Keleti K. u. 10. • ÁKV Műszaki Könyváruház, 1061 Budapest VI., Liszt F. tér 9. • Közgazdasági és Jogi Könyvesbolt, 1051 Budapest V., Münnich F. u. 8. • Technika Könyvesbolt, 1114 Budapest XI., Bartók B. út 15. • **A Művelt Nép Könyvterjesztő Vállalat vidéki boltjaiban: PÉCS:** Közgazdasági és Jogi Könyvesbolt, 7622 Szalai A. u. 6. • **SZEGED:** Móra Ferenc Könyvesbolt, 6701 Kárász u. 5. • **GYŐR:** Műszaki és Idegennyelvű Könyvesbolt, 9021 Lenin út 18. • **MISKOLC:** Műszaki és Idegennyelvű Könyvesbolt, 3530 Széchenyi út 54. • **DEBRECEN:** Ady Endre Könyvesbolt, 4024 Vörös Hadsereg útja 26.

Ára: 80,–Ft