MATIJEVICS ISTVÁN

MIKROSZÁMÍTÓGÉPEK

TANKÖNYV

KÉSZÜLT AZ APÁCZAI CSERE JÁNOS ALAPÍTVÁNY TÁMOGATÁSÁVAL

SZABADKA, 2000

ELŐSZÓ

A SZABADKAI MŰSZAKI FŐISKOLÁN 1976-ban kezdődött el a magyar nyelvű oktatás a villamosmérnöki szakon. A tanári kar összetétele, valamint a hallgatók száma lehetővé tette az első és második éven az oktatás teljes megszervezését magyar nyelven, az Informatika szakon pedig a képzés teljes ideje alatt. A **Mikroszámítógépek** tantárgy a Villamosmérnöki szak mindkét szakirányán, Irányítás- és robottechnika, Elektronika és híradástechnika valamint a Műszaki informatika szakon is kötelező tantárgy. A Főiskoláról kikerülő mérnökök megismerkednek a szakmájukban ma már nélkülözhetetlen mikroszámítógép-technikával, képessé válnak ezen berendezések tervezésére, bővítésére, üzemeltetésére és karbantartására.

A **Mikroszámítógépek** nevű tantárgy évről évre változik, a témakörben igen dinamikus a fejlődés. Nyomtatott anyaggal igen nehéz ellátni a hallgatókat, de a tantárgy jobb elsajátítása ezt szükségessé teszi.

Az itt nyomtatásban, az Interneten és CD-n megjelenő tankönyv a tantárgy a Műszaki Főiskola tantervében előírt anyagát tartalmazza.

A tankönyv követi az órákon előadott, valamint a gyakorlatokon begyakorolt, illetve a laboratóriumban kipróbált anyagot.

A tantárgy feldolgozza a két szakon tanított más szaktantárgyakhoz kötődő mikroszámítógépes ismereteket, természetesen külön hangsúlyt fektetve a mikrovezérlőkre is. Mivel az ilyen tipusú mérnökképzés erősen kötődik az alapok jó elsajátításához, ezért a tantárgy, illetve a tankönyv csak alpjaiban foglalkozik a nagyon elterjedt személyi számítógépek (**PC**-k) felépítésével, csak a legszükségesebb kapcsolódási pontokat ismertetve.

Minden oktatási egységet kérdések követnek, amelyek egyuttal vizsgakérdések is.

Itt mondok köszönetet kollégáimnak, munkatársaimnak az anyag összeállításánál, illetve a tankönyv megírásánál nyújtott segítségért.

Külön köszönöm az Apáczai Csere János Alapítvány támogatását.

Szabadka, 2000.06.07.

A szerző:

Prof. Dr. Matijevics István, egyetemi docens
(mistvan@vts.su.ac.yu)

TARTALOMJEGYZÉK

ELŐSZÓ…….. 2
TARTALOMJEGYZÉK……. 3
1. BEVEZETÉS…… 9
1.1 A MIKROSZÁMÍTÓGÉPEK ALKALMAZÁSI TERÜLETEI……. 9
1.1.1 Számítástechnika……... 9
1.1.2. Távközlés……... 10
1.1.3. Műszertechnika…… 11
1.1.4. Folyamatirányítás…… 12
1.1.5. Orvostudomány…… 12
1.1.6. Robottechnika…….. 12
1.2. HOL, MIKOR ÉS MIT HASZNÁLJUNK ?…….. 13
Ellenőrző kérdések…… 14
2. A SZÁMÍTÁSTECHNIKA FELOSZTÁSA…….. 15
2.1. DIGITÁLIS ÉS ANALÓG SZÁMÍTÁSTECHNIKA…….……... 15
Ellenőrző kérdések…… 16
3. INFORMATIKAI BEVEZETŐ……. 17
3.1. AZ ADAT, MINT INFORMÁCIÓHORDOZÓ……. 17
Ellenőrző kérdések…… 18
4. DIGITÁLIS TECHNIKA…… 19
4.1. BEVEZETÉS…….. 19
4.2. DIGITÁLIS-TECHNIKAI ÁRAMKÖRÖK, ELEMEK…… 20
4.2.1. Alapáramkörök……... 20
4.2.1.1. ÉS (AND) kapu…… 20
4.2.1.2. VAGY (OR) kapu…… 21
4.2.1.3. NEM (NOT, inverter) kapu……. 21
4.2.1.4. ÉS-NEM (NAND) kapu…….. 21
4.2.1.5. VAGY-NEM (NOR) kapu…….. 22
4.2.1.6. KIZÁRÓ-VAGY (EXCLUSIVE OR, XOR) KAPU…….. 22
4.3. NÉHÁNY ISMERT LOGIKAI ÖSSZEFÜGGÉS, JELÖLÉS…….. 23
Ellenőrző kérdések…….. 24
5. SZÁMRENDSZEREK…… 25
5.1 BEVEZETÉS……... 25
5.2. A TIZES SZÁMRENDSZER……. 25
5.3. A KETTES (BINÁRIS) SZÁMRENDSZER……. 25
5.4. SZÁMÁBRÁZOLÁS A KETTES (BINÁRIS) SZÁMRENDSZERBEN 26
5.4.1. Előjeles, abszolútértékes számábrázolás……... 26
5.4.2. Kettes komplemensú számábrázolás……. 27
5.5. A TIZENHATOS (HEXADECIMÁLIS) SZÁMRENDSZER…….. 29
Ellenőrző kérdések…… 29
6. SZÁMÍTÓGÉPEK……... 30
6.1. SZÁMÖTÓGÉPHARDVER FEJLŐDÉSÉNEK ÁTTEKINTÉSE…… 30
6.2. SZÁMÍTÓGÉPEK CSOPORTOSÍTÁSA…….. 31
Ellenőrző kérdések…… 33
7. JELLEGZETES DIGITÁLIS SZÁMÍTÓGÉP ARCHITEKTÚRÁK…… 34
7.1. BEVEZETÉS…….. 34

7.2. NEUMANN-FÉLE SZÁMÍTÓGÉPEK……..34
7.3. HARVARD STRUKTÚRÁJÚ SZÁMÍTÓGÉPEK…… 35
7.4. VEKTORSZÁMÍTÓGÉPEK…….. 36
7.5. TÖMBPROCESSZOROS SZÁMÍTÓGÉPEK……..37
7.6. ÜZENETÁTADÁSOS SZÁMÍTÓGÉPEK…… 38
7.7. ADATVEZÉRELT SZÁMÍTÓGÉPEK……. 39
Ellenőrző kérdések…… 40
8. SZÁMÍTÓGÉPEK UATSÍTÁSKÉSZLETE…… 42
8.1. BEVEZETÉS…….. 42
8.2. MIKROPROCESSZOROK UTASÍTÁSKÉSZLETE…… 42
Ellenőrző kérdések…… 43
9. MIKROPROCESSZOR TECHNOLÓGIÁK……... 44
9.1. BEVEZETÉS…….. 44
9.2. ALKALMAZOTT FÉLVEZETŐ TECHNOLÓGIÁK…….. 44
9.2.1. MOS (Matal Oxide Semiconductor) technológia……. 45
9.2.1.1. PMOS technológia…….…….. 45
9.2.1.2. NMOS technológia…….. 45
9.2.1.3. CMOS technológia…….……. 45
9.2.1.4. MOS integrált áramkör…… 46
9.2.2. Bipoláris félvezető technológia…….. 46
9.2.2.1. Schottky TTL (Transistor Transistor Logic) technológia 46
9.2.2.2. LPS TTL (kisfogyasztású Schottky TTL) technológia… 46
9.2.2.3. ECL (Emitter Coupled Logic) technológia…….. 46
9.2.2.4. I2L (Integrated Injection Logic) technológia……... 46
9.3. KÜLÖNBÖZŐ TECHNOLÓGIÁK ÖSSZEHASONLÍTÁSA…….. 47
Ellenőrző kérdések…… 48
10. MIKROPROCESSZOR, MIKROSZÁMÍTÓGÉP, MIKROVEZÉRLŐ. 49
10.1. BEVEZETÉS…….. 49
10.2. MIKROPROCESSZOR…….. 52
10.3. MIKROSZÁMÍTÓGÉP…….. 53
10.4. A MIKROSZÁMÍTÓGÉPES RENDSZER…… 54
10.5. A MIKROSZÁMÍTÓGÉP ÉS A MIKROVEZÉRLŐ ÖSSZE-HASONLÍTÁSA…… 55
Ellenőrző kérdések…… 56
11. A MIKROSZÁMÍTÓGÉP MŰKÖDÉSÉNEK ISMERTETÉSE…….. 57
11.1. BEVEZETÉS……... 57
11.2. A MIKROPROCESSZOR MODELL……. 57
11.2.1. Az utsítások…… 57
11.2.2. Egyszerű mikroprocesszor modell…… 60
Ellenőrző kérdések……. 76
12. A MIKROPROCESSZOR ELEMEI……... 77
12.1. BEVEZETÉS……... 77
12.2. A VEZÉRLŐ EGYSÉG……... 77
12.3. AZ ARITMETIKAI-LOGIKAI EGYSÉG……. 81
12.4. AZ AKKUMULÁTOR ÉS AZ ÁLTALÁNOS CÉLÚ REGISZTEREK 84
12.5. CÍMREGISZTEREK……... 84
Ellenőrző kérdések…… 86
13. MEMÓRIAKEZELÉS…….. 87

13.1. A MEMÓRIA KIVÁLASZTÓ LOGIKA (TELJES CÍMDEKÓDOLÁS)…….. 87
13.2. A MEMÓRIA KIVÁLASZTÓ LOGIKA (NEM TELJES CÍMDEKÓDOLÁS)…….. 97
13.3. VEREMTÁR ÉS VEREMTÁR-MUTATÓ (STACK ÉS STACK POINTER)……. 98
Ellenőrző kérdések…… 99
14. PERIFÉRIAKEZELÉS……. 100
14.1. KIMENETI ÉS BEMENETI INTERFÉSZEK, PERIFÉRIÁLIS EGYSÉGEK…….. 100
14.2. PÁRHUZAMOS I/O VEZÉRLŐ-ILLESZTŐ EGYSÉGEK…… 102
14.2.1. A CENTRONIX párhuzamos adatátviteli csatorna……. 104
14.3. SOROS I/O VEZÉRLŐ-ILLESZTŐ ESZKÖZÖK…….. 107
14.3.1. Soros szinkron és aszinkron adattovábbítás…….. 108
14.3.1.1. Szinkron soros adatátvitel……. 108
14.3.1.2. Aszinkron soros adatátvitel……... 109
14.3.1.3. RS-232 C aszinkron, feszültségszintű soros átviteli csatorna…….. 110
Ellenőrző kérdések…… 111
15. I2C SOROS BUSZ……. 112
15.1. BEVEZETÉS…….. 112
15.2. EGY BIT ÁTVITRLE AZ I2C BUSZON…….. 113
15.3. EGY BÁJT ÁTVITELE AZ I2C BUSZON…….. 114
15.4. ADATFORGALOM A BUSZON TÖBB ESZKÖZ KÖZÖTT……. 115
15.4.1. MASTER WRITE (mester írás)-állapot……. 116
15.4.2. MASTER READ (mester olvasás)-állapot…….. 117
15.4.3. Az ismételt START állapot használata folyamatos adatátvitelnél…… 117
Ellenőrző kérdések……. 118
16. ADATÁTVITEL A MIKROSZÁMÍTÓGÉP, MIKROVEZÉRLŐ ÉS A PERIFÉRIÁK KÖZÖTT…… 119
16.1 BEVEZETÉS……... 119
16.2. PROGRAMOZOTT I/O ADATÁTVITEL…… 120
16.2.1. Feltétel nélüli programozott I/O adatátvitel…… 120
16.2.2. Feltételes programozott I/O adatátvitel……... 121
16.2.3. Feltételes, kiválasztásos (Polling) I/O adatátvitel…… 123
16.3. MEGSZAKÍTÁSSAL KEZDEMÉNYEZETT ADATÁTVITEL….. 123
16.3.1. A megszakításforrás azonosítása…….. 127
16.3.2 A megszakításforrás azonosítása – az eszközök lekérdezése.. 127
16.3.3. A megszakításforrás azonosítása – vektoros megszakítás….. 127
16.4. ADATÁTVITEL KÖZVETLEN MEMÓRIAHOZZÁFÉRÉSSEL – DMA…….. 128
16.4.1. Bevezetés……. 128
16.4.2. A közvetlen memóriahozzáférés szervezése……. 128
16.4.2.1. DMA adatátvitel a processzor leállításával…… 131
16.4.2.2. DMA adatátvitel cikluslopással……. 131
16.4.2.3. DMA adatátvitel cpu/dma multiplexálással……... 131
16.4.3. A DMA műszaki megvalósítása…… 132
Ellenőrző kérdések……. 134

17. A/D ÉS D/A EGYSÉGEK ILLESZTÉSE MIKROSZÁMÍTÓGÉPEKHEZ…….. 135
17.1. BEVEZETÉS…….. 135
17.2. AZ ANALÓG DIGITÁLIS ÁTALAKÍTÁS…….. 137
17.2.1. Az A/D átalakítás sebessége…….. 137
17.2.2. Az analóg jel átalakítása, szóhosszúság…… 138
17.2.3. Az analóg feszültség tartománya, normalizálás…….. 140
17.2.4. Az érzékelők karakterisztikájának hatása az A/D átalakításra……... 140
17.3. A DIGITÁLIS ANALÓG ÁTALAKÍTÁS……. 140
Ellenőrző kérdések……. 142
18. A MIKROSZÁMÍTÓGÉP, MIKROVEZÉRLŐ PROGRAMJÁNAK ÍRÁSA……... 143
18.1. BEVEZETÉS……. 143
18.2. A PROGRAMOZÁSHOZ SZÜKSÉGES HARDVER ISMERETEK143
18.2.1. A mikroprocesszor…… 144
18.3. A MIKROPROCESSZOR, MIKROVEZÉRLŐ PROGRAMOZÁSÁNAK SZOFTVER SEGÉDESZKÖZEI……146
18.3.1. Az asszembler…… 146
18.3.2. A C-32 asszembler…… 147
18.3.2.1. Néhány gyakaran használt direktíva……. 148
Ellenőrző kérdések……. 148
19. KÜLÖNBÖZŐ JELEK ILLESZTÉSE A MIKROSZÁMÍTÓGÉPHEZ, MIKROVEZÉRLŐHÖZ……. 150
19.1. BEVEZETÉS…….. 150
19.2. BEMENETI ELEMEK……... 150
19.2.1. Mikrokapcsolók és végálláskapcsolók…… 151
19.2.2. Fotoelektromos kapcsolók…… 151
19.2.2.1. Egyutas fénysorompó……... 151
19.2.2.2. Visszaverődéses fénysorompó……. 152
19.2.2.3. Diffúz reflexiós típusú fenysorompó……... 152
19.2.3. Közelítéskapcsolók…… 153
19.2.4. Szintérzékelők…… 154
19.2.4.1. Kapacitív szintérzékelők……... 154
19.2.4.2. Ellenállásos szintmérők……. 154
19.2.4.3. Ultrahangos szintmérők…… 154
19.2.5. Hőmérséklet-mérők…….……. 155
19.2.5.1. Hőelem…….……. 155
19.2.5.2. Ellenállás hőmérő…….. 155
19.2.5.3. Termisztor……. 155
19.3. KIMENETI ELEMEK…….……... 156
19.3.1. Érintkezős kimenetek……... 156
19.3.1.1. Jelfogók…….…… 156
19.3.1.2. Szilárdtest-jelfogók……... 157
19.3.1.3. Triak-kimenetek…….……... 158
19.4. BEMENETI ÉS KIMENETI EGYSÉGEK ILLESZTÉSE A MIKROSZÁMÍTÓGÉPHEZ……... 158
19.4.1. Bevezetés…… 158

19.4.2. Jelformálás a mikroszámítógép, mikrovezérlő bemenetein…… 158
19.4.3. Jelformálás a mikroszámítógép, mikrovezérlő kimenetein…… 162
19.4.4. Bemeneti és kimeneti egységek illesztése…… 163
Ellenőrző kérdések…… 163
20. AZ INTEL 8051 MIKROVEZÉRLŐ CSALÁD…… 165
20.1. BEVEZETÉS…….. 165
20.2. AZ MCS-51 CSALÁD ÁLTALÁNOS JELLEMZÉSE……. 165
20.3. AZ MCS-51 CSALÁD LEGFONTOSABB JELLEMZŐI…… 166
20.4. A 8051/8031 MIKROVEZÉRLŐ HARDVER FELÉPÍTÉSE……... 170
20.4.1. A 8051/8031 memória szervezése és belső regiszterei…… 170
20.4.2. A 8051/8031 órajelének előállítása……... 172
20.4.3. A 8051 párhuzamos portjai…….. 173
20.4.4. A 8031 összekapcsolása külső memóriával (RAM és EPROM)……... 173
20.4.5. Beépített időzítő/számláló áramkörök…… 174
20.4.6. Soros adatátvitel és soros port vezérlő és állapotregiszter… 175
20.4.7. Megszakításkezelés (Interrupt)…… 177
20.4.8. A rendszer alaphelyzete…….. 178
20.4.9. A mikrovezérlő kisebb fogyasztású változata……. 179
20.4.10. A T0 és T1 időzítő/számláló és üzemmódjaik (0-3)…….. 181
20.4.11. A portok……... 183
20.4.12. A portok terhelhetősége (Fan-Out)……... 186
20.4.13. READ-MODIFY-WRITE utasítások…… 186
20.4.14. A megszakítások és használatuk…… 186
Ellenőrző kérdések…… 187
21. A 8051 MIKROVEZÉRLŐ CSALÁD UTASÍTÁSKÉSZLETE……... 188
21.1. BEVEZETÉS…….. 188
21.2. BIT-, ÉS BÁJTSZERVEZÉSŰ MŰKÖDÉS…….. 188
21.3. AZ UTASÍTÁSOK HOSSZA……. 188
21.4. UTASÍTÁSVÉGREHAJTÁSI ÉS FUTÁSI IDŐ……... 188
21.5. AZ UTASÍTÁSOK CÍMZÉSI MÓDJAI…… 189
21.5.1. Regiszter címzés…… 189
21.5.2. Direkt címzés……. 190
21.5.3. Indirekt címzés…….. 190
21.5.4. Közvetlen címzés……... 190
21.5.5. Indirekt regiszter-címzés…….. 191
21.6. A KÜLÖNBÖZŐ UTASÍTÁS FAJTÁK…… 191
21.6.1. Az adatátviteli utasítások……. 191
21.6.2. Az aritmetikai utasítások…….. 192
21.6.3. Logikai és Boole utasítások…….. 192
21.6.4. Vezérlő utasítások……. 193
21.7. AZ UTASÍTÁSOK HOSSZA ÉS VÉGRAHAJTÁSI IDEJÜK……. 196
21.8 A FLAG-EKET BEFOLYÁSOLÓ UTASÍTÁSOK…… 200
21.9. READ-MODIFY-WRITE UTASÍTÁSOK……. 200
Ellenőrző kérdések…… 201
22. A PIC 16Cxx MIKROVEZÉRLŐ CSALÁD……... 202
22.1. BEVEZETÉS…….. 202

22.2. A PIC 16c71-ES MIKROVEZÉRLŐ……. 204
22.2.1. A PIC 16c71 belső hardver felépítése, regiszterei…….. 205
22.2.2. A mikrovezlrlő regiszterei…… 208
22.2.3. A PIC 16c71 megszakítás-kezelése, az INTCON regiszter… 211
22.2.4. A PIC 16c71 A/D konverziós egysége……. 213
22.3. A PIC 16c84 MIKROVEZÉRLŐ……... 215
Ellenőrző kérdések…… 217
23. A PIC MIKROKONTROLLEREK UTASÍTÁSKÉSZLETE……... 218
Ellenőrző kérdések…… 220
24. IBM PC ÉS KOMPATIBILIS SZEMÉLYI SZÁMÍTÓGÉPEK…….. 221
24.1. BEVEZETÉS…….. 221
24.2. A 8086, 8088 MIKROPROCESSZOR BELSŐ FELÉPÍTÉSE…….. 221
24.2.1. Az INTEL 8086 processzor felépítése…….. 222
24.2.2. A 8086 mikroprocesszor működése……. 223
24.2.3. A 8086 címzési technikája…… 224
24.2.4. A programozási címzési módok……... 225
24.2.5. Adatmemória címzési módok……... 226
24.3. 80286, 80386, 80486, PENTIUM, PENTIUM II ÉS PENTIUM III PROCESSZOROK……. 226
24.4. AZ IBM PC SINRENDSZERE…….. 226
24.4.1. Bevezetés…… 226
24.4.2. IBM PC belső sinrendszer (busz)…… 227
24.4.3. IBM PC külső sinrendszere…….. 227
24.4.3. AZ IBM PC, vagy XT sin……. 227
24.4.3.1. AT, ISA sin…….. 229
24.4.3.2. A VESA sin…….. 231
24.4.3.3. A PCI sin…….. 231
24.4.3.4. A PCMCIA sin……. 231
Ellenőrző kérdések…… 232
IRODALOM……. 233

1. BEVEZETÉS

Az utóbbi néhány évben a számítástechnika fejlődése rendkívüli méreteket öltött, betört életünk minden területére. Ma már nincs olyan háztartás, üzem, berendezés stb. ahol ne lenne jelen valamilyen módon a mikroszámítástechnika. Óriási terhet vesz le a vállunkról alkalmazása, felhasználása, teljesen új, sokszor előre nem látott szolgáltatásokat nyújtva, gondoljunk csak a most robbanásszerű fejlődést mutató Internet hálózatra, vagy a globálissá, világot átfogóvá fejlődő mobil telefonhálózatra.

A tervezésben, gyártásban alkalmazva új és jobb berendezáseket állíthatunk elő, ez pedig visszahat a számítástechnika fejlődésére. A mikroszámítógép bizonyos feladatokat gyorsabban, pontosabban, olcsóbban és hatásosabban végez el mint az ember, vagy más műszaki megoldást felhasználó berendezés.

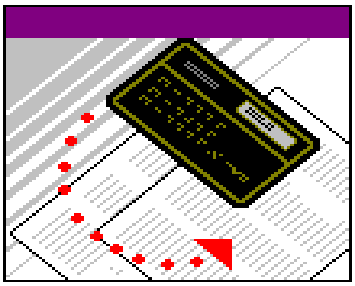
1.1. A MIKROSZÁMÍTÓGÉPEK ALKALMAZÁSI TERÜLETEI

Hol találkozhatunk mikroprocesszorokkal, mikrovezérlőkkel? A válasz meglepő, mindenhol, de ma már ez valahogy fel sem tűnik. A digitális karórában az óra lelke egy kis processzor, de a TV hangolóegységét is mikroprocesszor vezérli, ez a kisméretű integrált áramkör (IC) gondoskodik a mikrohullámú sütő programjának végrehajtásáról is.

Mi az oka a mikroszámítógépek (lelke a mikroprocesszor ami köré különböző elemeket építünk) ilyen méretű elterjedésének? A magyarázat egyszerű, ezek az elemek ma már hihetetlenül olcsók, kicsi a fogyasztásuk, nem elhanyagolható az elterjedésükből eredő sok alkalmazási tapasztalat, megbízhatóak, de ami rendkívül fontos, a hagyományos elemekkel szemben teljesen új szolgáltatásokat is nyújtanak, sokkal összetettebb feladatok megoldására is alkalmasak.

A teljesség igénye nélkül sorolunk fel néhány jellegzetes alkalmazási területet.

1.1.1. **Számítástechnika**



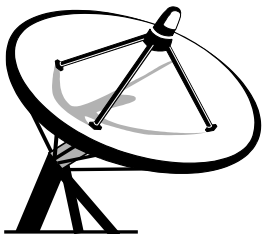
A mikroprocesszorok, mikroszámítógépek és mikrovezérlők megjelenésük után azonnal a különböző számítógépperiférián nélkülözhetetlen elemeivé váltak. Így a monitorok, teleprinterek, párhuzamos és soros nyomtatók, mágnesszalag egységek, Winchesterek, rajzolók stb. mind mikroprocesszoros, mikrovezérlős megoldásokat tartalmaznak. A perifériák összetett feladatait, mint adatátvitel szinkronizálás, adattárolás, a mechanikai elemek, meghajtómotorok vezérlését képesek elvégezni.

A különböző házi és személyi számítógépek processzorai is a mikroszámítógépek csoportjába tartoznak. Annyira hatásosak, hogy ezen gépek még komoly ipari vezérlések nélkülözhetetlen részeivé is váltak.

A mikroprocesszorok azon tulajdonsága, hogy párhuzamos üzemmódban is használhatók biztosította a multiprocesszoros (többprocesszoros) rendszerek kialakítását. Ez igen fontos a gyors, sok adat egyidejű feldolgozását igénylő rendszerek esetében, mint amilyenek a tudományos számítások (alapkutatásoknál szimulálás, képfeldolgozás, keresés nagy adatbázisokban stb,).

A mikroszámítástechnika komoly teljesítménye, valamint viszonylag alacsony ára segítette elő a disztribuált (térben szétosztott) adatfeldolgozó rendszerek kialakulását, ahol egy vagy több központi gép köré telepített alállomás képes elvégezni helyben összetett feladatokat. Egy ilyen rendszer ma már térben igen nagy távolságokra levő viszonylag sok gépet tartalmazhat.

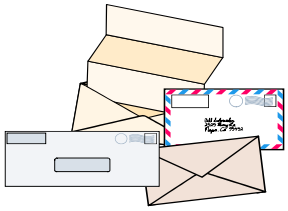
1.1.2. **Távközlés**



A különböző távközlési berendezések kis és nagy számítógépei, a terminálok és hálózatok a digitális formában történő adatátvitelt biztosítják. Ezen berendezésekben a jelek átalakítását, a berendezések közötti kapcsolatfelvételt és adatátvitelt (protokoll) bonyolítják le. Itt a legkülönbözőbb információ átvitele lehetséges, kép, hang, számadat stb. Mikrogépekkel oldják meg a kapcsolatfelvételt, adattömörítést, és ami nagyon fontos a hibamentes adatátvitelt is.

Igen fontos és nélkülözhetetlen szerepet kaptak a mikroprocesszoros berendezések a postai rendszerekben, a telefonközpontokban, az egészen kis kapacitású házi központoktól a nagy kapacitású, városokat, országokat összekötő rendszerekig.

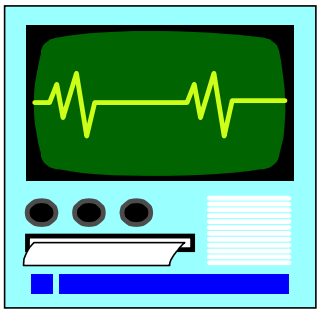
A távközlési berendezések fontos feladata a mindenkori hibafelderítés és javítás, ami ma már teljes mértékben számítógépesített eljárás.



Külön figyelmet érdemel a rendkívüli ütemben fejlődő Internet, egész világra kiterjedő számítógépes hálózat. A rendszer óriási előnye az, hogy nem igényel új berendezéseket, hanem a meglevő különböző számítógépeket köti egy egész világot átfogó olcsó hálózatba. Az elektronikus posta forradalmasítja a levelezést. A vállalatok, könyvtárak, szórakoztatóipar, egyszóval bárki és bármilyen szervezet ezen a rendszeren keresztül tarthat akár még on-line (azonos idejű) kapcsolatot fenn egymással, de még pénzügyi forgalmat is lebonyolíthat. Egyes szakemberek az Internet jelentőségét a kerék feltalálásával hasonlítják össze.

Az Internet mellett ma a legdinamikusabban fejlődő másik távközlési rendszer a mobil telefónia, amely szintén alkalmazza a számítástechnika eredményeit, új távlatokat nyit az emberek közötti kommunikációban. Összekapcsolódik az informatikával, átalakítja az élet minden területét. Ma már elektronikus levelezés is lebonyolítható egy mobil készülékkel, de akár az autóriasztóval is összeköthető.

1.1.3. **Műszertechnika**



A legkülönfélébb műszerek ma már nélkülözhetetlen részét alkotják a mikroszámítógépek, mikrovezérlők. A digitális többfunkciójú mérőműszerektől kezdve a számlálókon, oszcilloszkópokon, logikai analizátorokon keresztül a nagybonyolultságú hibaellenőrző berendezésekig mindenütt találkozhatunk velük.

A mikroprocesszorok egyszerű és gyors átprogramozhatósága teszi lehetővé a legösszetettebb feladatok megvalósítását, a gyors és hatásos adatbevitelt, a mért értékek legkülönbfélébb kielemzését, a felhasználó igényeit a legmesszemenőbben figyelembe vevő adatkijelzést, az állandó önellenőrzést, autómatikus kalibrációt, a számítógépekkel való kapcsolatteremtést.

A különféle szabványos sinrendszerek (busz) alkalmazása a műszerek összekapcsolásával a legkülönfélébb autómatikus mérőrendszerek kialakítását teszik lehetővé.

1.1.4. **Folyamatirányítás**

Ipari folyamatok autómatizásása ma már elképzelhetetlen ezen berendezések alkalmazása nélkül. Fokozatosan szorították ki a hagyományos, analóg technikákon alapuló szabályozókat, adatgyűjtőket nemcsak átvéve azok feladatait, hanem minőségileg új funkciók beépítésével forradalmasítva azok működését, felhasználási területeit. Helyettesítik a diszkrét logikai elemeket is, a jelfogókat, tranzisztoros- és félvezetős logikai kapukat, mégpedig jórészt szoftverrel, ami ezen berendezések gyors átprogramozását is lehetővé teszik.

A mikroszámítástechnika alkalmazása az automatizálásban nemcsak pontosabb, gyorsabb mérést és távmérést tesz lehetővé, de megnyitotta az utat az ipar területén az energiatakarékosabb és ami szintén nem elhanyagolható, a környezetbarát ipari berendezések létrehozására.

A folyamatirányító rendszerek kiépítési, üzemeltetési költségei lényegesen csökkennek a mikroszámítógépek alkalmazásával. Még a térben viszonylag távol álló berendezéseknél is egy hálózat kialakítása egyszerű, a helyi feladatok helyszínen való megoldása miatt pedig a kábelezési költségek óriási mértékben csökkennek, ezzel csökkentve az összköltséget, de a hibák lehetőségének a számát is, növelve a rendszer megbízhatóságát.

1.1.5. **Orvostudomány**

A mikroszámítógépek alakalmazása igen elterjedt a különféle orvosi műszerekben, kezdve a bonyolult véranalizátoroktól egészen a vérnyomásmérőkig. Igen sokfajta segédeszköz készül a valamilyen formában károsult egyének számára, mint például az írott szöveget hangos beszéddé átalakító készülék is, kükönféle művégtagok vezérlése.

A bonyolult kórházi betegőrző és ellenőrző, felügyelő rendszerek költségei a mikroszámítógépek alkalmazásával csökkentek, a szolgáltatások minősége és mennyisége viszont nőtt. Egy-egy ágynál elhelyezett rendszer a beteg számára fontos megfigyelést végzi, ezt helyileg kijelzi, de a központi számítógépbe is átküldi az adatokat további feldolgozásra. Különleges szoftver, a szakértői rendszer alkalmas diagnózis felállítására és terápia meghatározására anélkül, hogy állandóan orvosi konzíliumot kellene tartani sok szakember segítségével.

1.1.6. **Robottechnika**

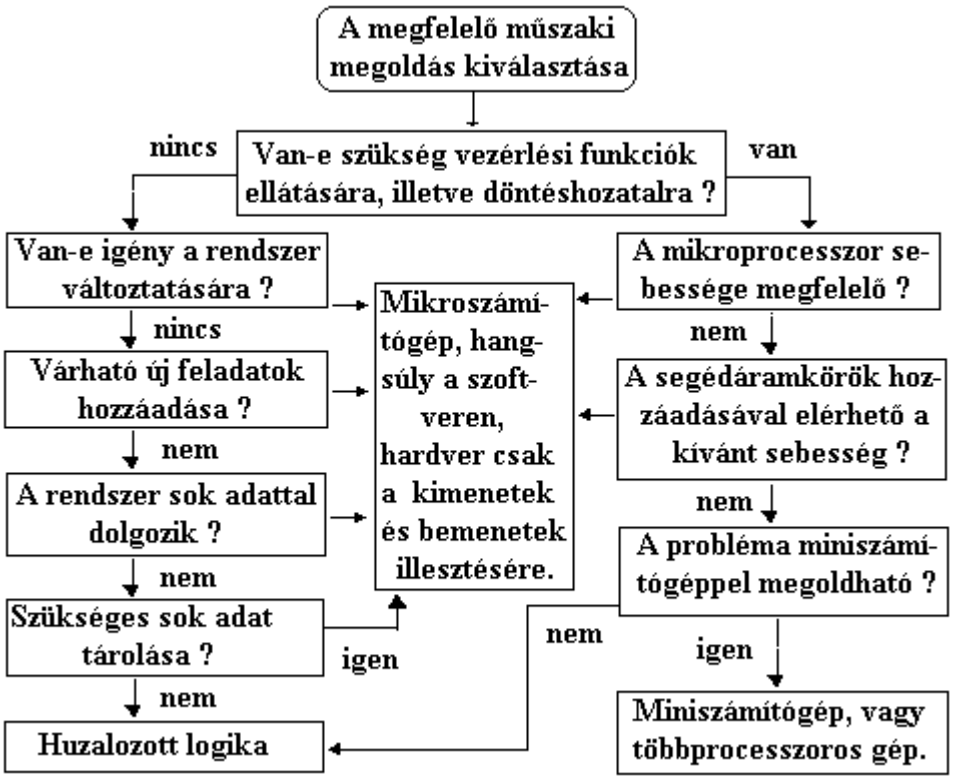
A gyáriparban a legkülönbözőbb berendezések összeszerelését végzik a robotok. Nemcsak állandó pontosságuk, fáradhatatlanságuk miatt terjedtek el ilyen mértékben az

autóipartól kezdve a kőolajiparig, de igen fontos hogy az ember számára veszélyes helyeken is kitünően alakalmazhatók. A személyi számítógépek alcsony ára jórészt az összeszerelésükkor nélkülözhetetlen robotok alkalmazása miatt alakult igy.

1.2. HOL, MIKOR ÉS MIT HASZNÁLJUNK

A mérnökök feledataik megoldása közben állandóan szembesülnek a problémával, mikor milyen megoldást a legcélszerűbb alkalmazni. A kérdés igen bonyolult, mert a nem megfelelően megválasztott műszaki megoldás amellett hogy nagyon drága is lehet, lehet hogy nem látja el megfelelően a kitűzött feladatokat (ami megint csak költség).

Az 1.1. ábrán található folyamatábra segíthet annak megválaszolásában, hogy elemezve bizonyos feltételeket, mikor milyen megoldás használata a legcélszerűbb.



1.1. ábra: Folyamatábra a legcélszerűbb műszaki megoldás megválasztására

Ellenőrző kérdések

1. Sorolja fel hol alkalmaznak mikroszámítógépet, mikrovezérlőt.
2. A számítástechnika mely eszközeiben található mikroszámítógép ?
3. A távközlési berendezések mely részei tartalmaznak mikroszámítógépet ?
4. Milyen műszerekben találhatók mikroszámítógépek és milyen előnnyel jár beépítésük ezekbe a berendezésekbe ?
5. A mikrogépek alkalmazása a folyamatirányításban milyen előnnyel jár ?
6. Az orvostudomány hogyan alkalmazza a mikroszámítógépeket ?
7. Mi a szerepe a mikroszámítógépeknek a robottechnikában ?
8. Ismertesse a mikroszámítógép alkalmazásának kritériumait, mikor érdemes alkalmazni ezeket az eszközöket ?
9. Amennyiben a mikrogép működési sebessége nem elégíti ki a feldolgozási sebességgel szemben támasztott igényeket, mit alkalmazunk az ilyen berendezésekben?

2. A SZÁMÍTÁSTECHNIKA FELOSZTÁSA

2.1 DIGITÁLIS ÉS ANALÓG SZÁMÍTÁSTECHNIKA

A technológia mai fejlettségi fokán a legelterjedtebb az elektronikus digitális (ezen belül a kettes (bináris) számrendszert használó) számítógép. Itt is a legismertebb a képernyőt, nyomtatót és billentyűzetet használó rendszer. Az ilyen konfiguráció kialakulásának oka egyrészt anyagi jellegű (az alacsony ár), kihatnak kialakulására a különböző fizikai paraméterek (méret, adatfeldolgozási sebesség, megbízhatóság), de meghatározó a kitűzött feladat jellege is (számok, szöveg bevitele, egyszerű színes képes megjelenítés, nyomtatás).

Természetesen a számítástechnika fejlődése magával hozta az újabbnál újabb megoldások alkalmazását, ezek közül néhány tartósan megmaradt, sok pedig elveszítette jelentőségét.

Bizonyos különleges alkalmazási területen megmaradt ezen megoldások közül néhány, itt ma is indokolt alkalmazásuk (például nagy sebesség, zavarmentesség, környezeti károsodás).

Megszoktuk a hagyományos képernyő-billentyűzet-nyomtató összetételű rendszert, amihez ma már a legkülönbözőbb eszközöket kapcsolhatjuk, így kialakítva a multimédiás berendezést. Természetesen egy mikroszámítógépes rendszer egészen más elemeket is tartalmazhat, ugyanakkor lehet, hogy sem képernyő, sem billentyűzet sincs a rendszerben.

A 2.1. táblázat tartalmazza az analóg- és digitális számítástechnika néhány jellegzetességét. A bennünket körülvevő fizikai mennyiségek jó része a folytonos jelek közé tartozik (hőmérséklet, nyomás, feszültség stb.), de sok csak két, vagy esetleg több jól meghatározott értéket vehet fel (motor ki- be kapcsolása, jelzőlámpa világit, nem világit stb). Ezen jelek feldolgozására ugyan alkalmas mindkét módszer, de a műszaki megoldások egyszerűbbé, pontosabbá teszik a digitális feldolgozást. Bizonyos esetekben nagyon gyors, folytonos jelek feldolgozását csak analóg elemeket tartalmazó rendszerrel oldhatunk meg.

2.1. táblázat: A számítástechnika felosztása	
analóg	digitális
folytonos jelek folytonos feldolgozása	folytonos jelek diszkrét feldolgozása
	diszkrét jelek feldolgozása
Összeadó, integrátor, szorzó	hardver,szoftver
analóg-digitális és digitális-analóg átalakítók	

A számítástechnikai elemek különböző fizikai hordozókkal valósíthatók meg, ezek összefoglalása található meg az 2.2. táblázatban.

2.2. táblázat: A számítástechnika fizikai hordozói		
Elektronikus	mechanikus	pneumatikus, hidraulikus
Elektromechanikus elemek		

Ellenőrző kérdések

1. Melyek a legelterjedtebb számítógépek ?
2. Mely eszközök alkotják a mai modern személyi számítógépeket ?
3. A környezetünkben levő jelek milyen két csoportba sorolhatók ?
4. Soroljon fel néhány folytonos jelet .
5. Soroljon fel néhány diszkrét jelet .
6. Milyen fizikai elemek (hordozók) használatosak a számítástechnikában ?

3. INFORMATIKAI BEVEZETŐ

3.1. AZ ADAT, MINT INFORMÁCIÓHORDOZÓ

Minden számítástecnikai elem információfeldolgozást végez, a vele adat formájában közölt információt dolgozza fel- és át a felhasználó számára értelmes formájú eredménnyé. Az információ adat formájában kerül a gépbe, majd vissza a felhasználóhoz. Felhasználó alatt nem feltétlenül embert értünk, pl. egy hőmérsékletszabályozásnál a szabályozó egy mérőátalakítótól kap adatot a pillanatnyi hőmérsékletről, egy beavatkozó szerven keresztül pedig adatot küld a hőmérséklet megváltoztatására.

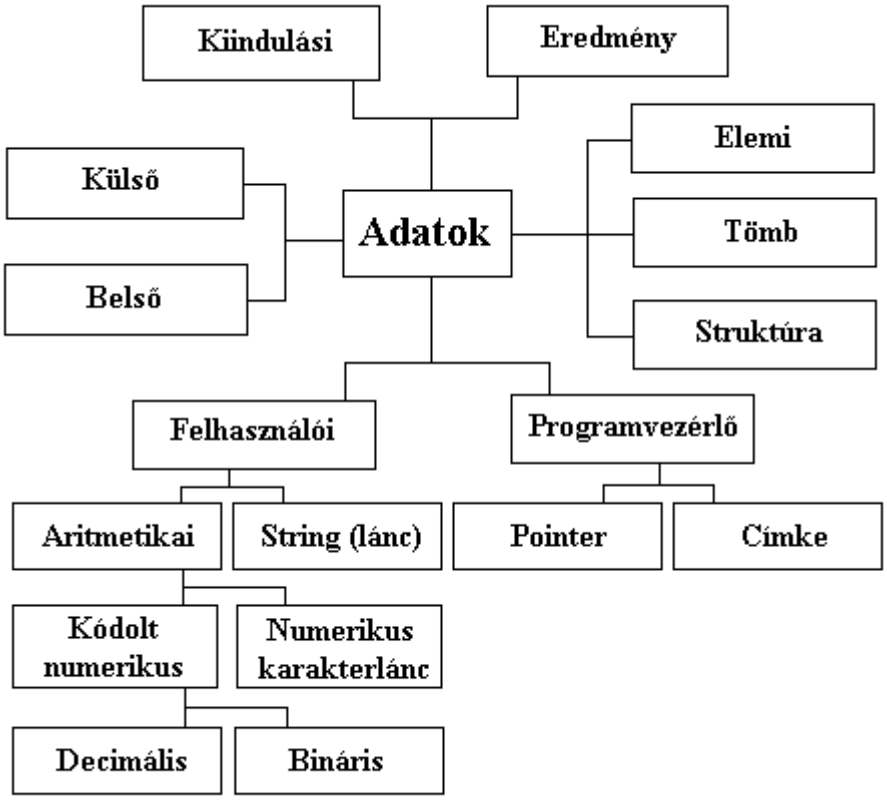
Az **adat** olyan tapasztalati tény (vagy tények összessége), mérési eredmény vagy számított érték, amely egy adott feltételrendszerben egy állapotot jellemez. Igen sokféle adatról beszélhetünk, ezeket többféle szempont szerint osztályozhatjuk. A 3.1. - 3.4. táblázatok, illetve 3.1 ábra szerinti felosztás leginkább a programozási gyakorlatra jellemző, de jól mutatja az adatok sokféleségét.

3.1. táblázat: Az adatok felosztása	
adat	
Kiindulási	eredmény

3.2. táblázat: Az adatok felosztása	
adat	
Külső	belső

3.3. táblázat: Az adatok felosztása		
adat		
Elemi	tömb	struktúra

3.4. táblázat: Az adatok felosztása					
adat					
Felhasználói				programvezérlő	
Aritmetikai			lánc	pointer	cimke
kódolt numerikus		numerikus karakterlánc			
decimális	Bináris				



3.1. ábra: Az adatok felosztása

A kiindulási adatok a probléma megoldásának kezdőértékei; ezek segítségével állítja elő a rendszer az eredményadatokat. Az eredményadatok meghatározására szolgál tulajdonképpen a program. A külső adatokat a rendszer a külvilágtól várja valamilyen bemenő műveleten keresztül, a belső adatokat maga a program állítja elő. A belső adatok a kiindulási és eredményadatok közti összekötő láncszemet is jelentik. Ezeket a program csak saját céljaira használja. Ha az adat egyedi azonosítóval rendelkezik, akkor ez elemi adat. Ha több adatot egyetlen azonosítóval jellemezhetünk, akkor ez

adatcsoport. Ha egy adatot csak a program futásának befolyásolására használunk, akkor ez programvezérlő adat. Számunkra a felhasználói adatok a legfontosabbak.

A számítógépek adatábrázolása igen egyszerű. Amikor egy adat bekerül a mikroprocesszorba, akkor a mikroprocesszor automatikusan nem vizsgálja meg, hogy az milyen adat. Az egyes adattípusokat programmal kell megkülönböztetnünk, vagyis maga a művelet határozza meg az adat típusát.

Ellenőrző kérdések

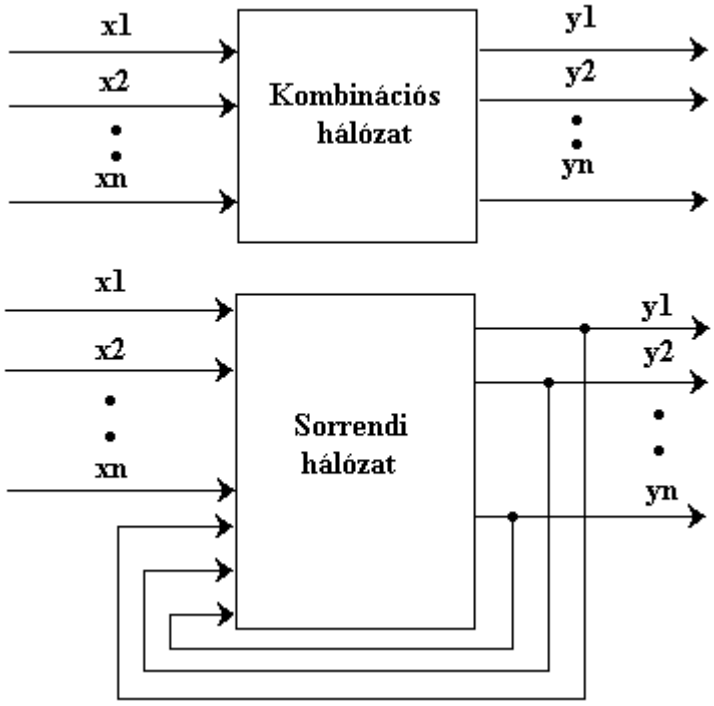
1. Mi az információ ?
2. Mi az adat ?
3. Milyen szempontok szerint és hogyan oszthatjuk fel az adatokat ?

4. DIGITÁLIS TECHNIKA

4.1 BEVEZETÉS

A hagyományos logikai tervezés eltér a mikroprocesszoros tervezéstől, számunkra azonban mégis fontos a logikai tervezés mivel a mikrogép interfészeket (illesztő elemeket) általában így kell létrehoznunk. Valamilyen logikai áramkör létrehozása a megvalósítandó logikai függvény meghatározásából és ennek alapján az ezt megvalósító áramkör összeállításából áll. A logikai függvény meghatározása azonban nem azt jelenti, hogy pontosan ezt a függvényt kell realizálnunk, hiszen a logikai függvények bizonyos szabályok alapján lényegesen egyszerűsíthetők. Az áramkörtervezésnél a megvalósítandó logikai függvény egyszerűsítésének nemcsak az a célja, hogy az áramkör kevesebb elemet tartalmazzon és a lehető legegyszerűbb legyen, hanem az is, hogy a véletlenszerűen fellépő, a tervezésből adódó hibajelenségeket a minimálisra csökkentsük, az esetleges hibajavítást és áramkör-módosítást minél könnyebbé tegyük.

Kétféle – alapvetően különböző – logikai hálózattípus ismert, a kombinációs és a sorrendi (szekvenciális) hálózat (4.1. ábra):



- a **kombinációs hálózatok** kimeneteinek (függő változók) értéke (állapota) csak az aktuális bemeneti kombinációtól, azaz a független változók pillanatnyi értékétől függ,
- a **sorrendi** (szekvenciális) **hálózatok** aktuális kimeneti kombinációja az aktuális bemeneti kombinációktól és az előző időpillantban, és/vagy időpillanatokban fennálló kimeneti kombinációtól gügg.

4.1. ábra: Kombinációs és sorrendi hálózat

A mikroszámítógépekben használatos elemek a digitális technika elemkészletéből állnak, ezért szükséges az alapfogalmak ismertetése.

4.2. DIGITÁLIS-TECHNIKAI ÁRAMKÖRÖK, ELEMEK

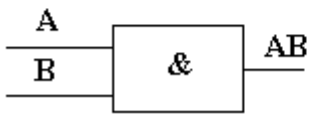
Mint már a 2. fejezetben látható volt, a mikroszámítógépeknél a digitális technikában alkalmazott eljárásokhoz hasonlóan történik a műveletvégzés, ezért célszerű néhány alapfogalom ismertetése. A két elektromos értékkel leírt digitális bináris jel az áramkörök be- ill. kimenő értéke. Az áramkörök működése jól leírható a kapcsolók működésével is.

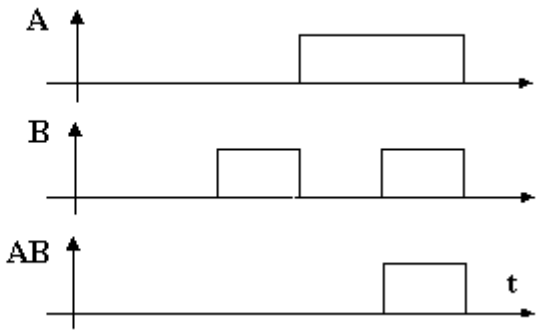
A két érték jelölése sokféle lehet, legelterjedtebb a **0** és **1**, **LOW** és **HIGH**, **HAMIS** vagy **IGAZ** stb. A további tárgyalásban feltételezzük, hogy a két állapot mellett csak az átmentben lehet más értékű a jel, elhanyagolhatóan rövid ideig.

4.2.1. **Alapáramkörök**

4.2.1.1. **ÉS** (AND) kapu

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1





A 2, vagy több bemenetet és egy kimenetet tartalmazó logikai szorzást végző áramkör kimenetén logikai egyes csak akkor jelenik meg, ha az összes bemenet logikai egyes értékű.

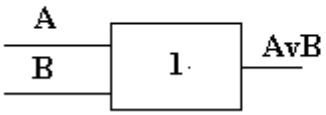
Az áramkör igazságtáblája, jelképi jelölése és idődiagramja a 4.2. ábrán látható.

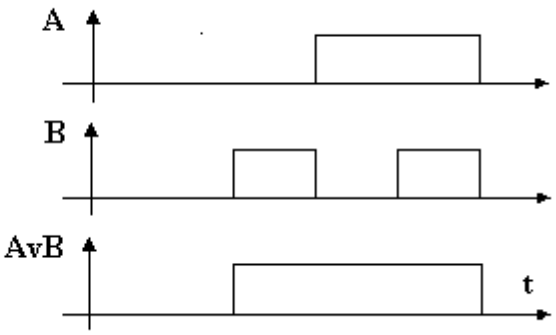
4.2. ábra: Az **ÉS** kapu igazságtáblázata, jelölése és idődiagramja

4.2.1.2 **VAGY** (OR) kapu

A 2, vagy több bemenetet és egy kimenetet tartalmazó logikai összeadást végző áramkör kimenetén logikai egyes akkor jelenik meg, ha legalább egy bemenet logikai egyes értékű.

A	B	AvB
0	0	0
0	1	1
1	0	1
1	1	1





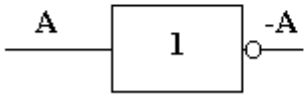
Az áramkör igazságtáblája, jelképi jelölése és idődiagramja a 4.3. ábrán látható.

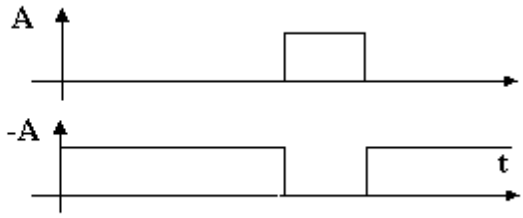
4.3. ábra: A **VAGY** kapu igazságtáblázata, jelölése és idődiagramja

4.2.1.3. **NEM** (NOT, inverter) kapu

A **NEM** logikai kapu olyan egybementű és egykimenetű áramkör, amely kimenetén mindig a bemenettel ellentétes szint jelenik meg.

A	-A
0	1
1	0





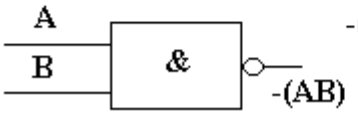
Az áramkör igazságtáblája, jelképi jelölése és idődiagramja a 4.4. ábrán látható.
4.4. ábra: A **NEM** kapu igazságtáblázata, jelölése és idődiagramja

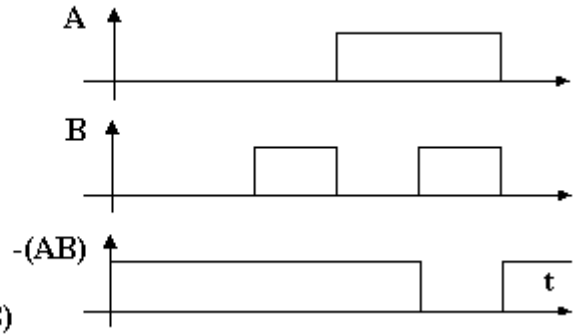
4.2.1.4. **ÉS-NEM** (NAND) kapu

A 2, vagy több bemenetet és egy kimenetet tartalmazó áramkör kimenetén logikai nulla csak akkor jelenik meg, ha az összes bemenet logikai egyes értékű.

Az áramkör igazságtáblája, jelképi jelölése és idődiagramja a 4.5. ábrán látható. Az áramkör kimenetén levő értékek pontosan az **ÉS** kapu kimenő értékeinek invertált értékei.

A	B	-(AB)
0	0	1
0	1	1
1	0	1
1	1	0



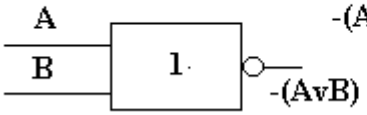


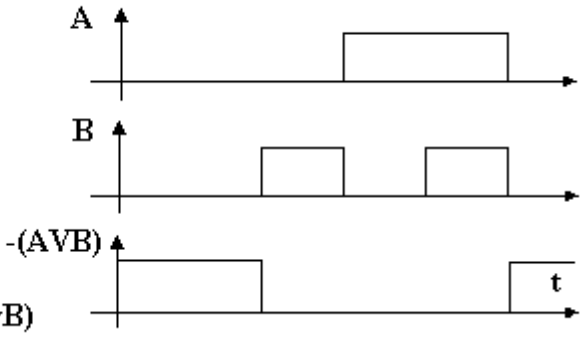
4.5. ábra: Az **ÉS-NEM** kapu igazságtáblázata, jelölése és idődiagramja

4.2.1.5. **VAGY-NEM** (NOR) kapu

A 2, vagy több bemenetet és egy kimenetet tartalmazó logikai **VAGY-NEM** áramkör kimenetén logikai nulla akkor jelenik meg, ha legalább egy bemenet logikai egyes értékű.

A	B	-(AvB)
0	0	1
0	1	0
1	0	0
1	1	0





Az áramkör igazságtáblája, jelképi jelölése és idődiagramja a 4.6. ábrán látható. Az áramkör kimenetén levő értékek pontosan a **VAGY** kapu kimenő értékeinek invertált értékei.

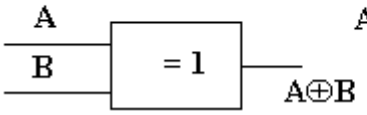
4.6. ábra: Az **ÉS-VAGY** kapu igazságtáblázata, jelölése és idődiagramja

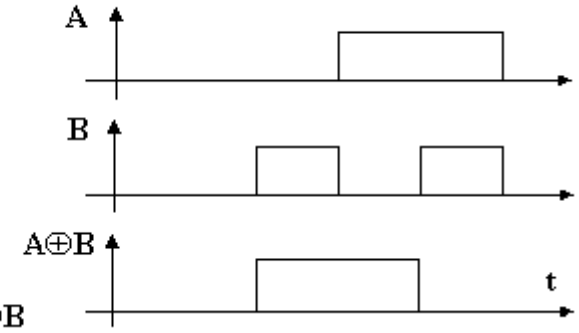
4.2.1.6. **KIZÁRÓ-VAGY** (EXCLUSIVE OR - XOR) kapu

A 2 bemenetet és egy kimenetet tartalmazó logikai **KIZÁRÓ-VAGY** áramkör kimenetén logikai egyes akkor jelenik meg, ha csak az egyik bemenet logikai egyes értékű. A jelölése **=1** jellel történik.

Az áramkör igazságtáblája, jelképi jelölése és idődiagramja a 4.7. ábrán látható.

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0





4.7. ábra: Az **KIZÁRÓ-VAGY** kapu igazságtáblázata, jelölése és idődiagramja

4.3. NÉHÁNY ISMERT LOGIKAI ÖSSZEFÜGGÉS, JELÖLÉS

A mikroszámítógép tervezésekor a logikai elemeket tartalmazó részek egyszerűsíthetőek néhány ismert összefüggés felhasználásával.

De Morgan tétele szerint:

$$-(AB) = (-A) + (-B) \qquad (4.1)$$
$$-(A + B) = (-A)(-B) \qquad (4.2)$$

Ezekből következik az **ÉS**, **VAGY** és **NEM** közötti összefüggés:

$$AB = -((-A) + (-B)) \qquad (4.3)$$

A **KIZÁRÓ VAGY** létrehozható:

$$A \oplus B = -((-A) + B) + (-(A + (-B))) \qquad (4.4)$$

A használt jelölések:

ÉS ----------• (pont), * (csillag) vagy üres hely,
VAGY ---------- + (összeadásjel),
NEM ---------- - (minuszjel) és
KIZÁRÓ VAGY -- $\oplus$, = 1.

Ellenőrző kérdések

1. Milyen két logikai hálózattipus ismeretes ?
2. Mi jellemzi a kombinációs hálózatot ?
3. Mi jellemzi a szekvencális hálózatot ?
4. Mely logikai alapáramköröket ismeri ?
5. Ismertesse a logikai ÉS kapu jelölését és időDiagramját .
6. Ismertesse a logikai VAGY kapu jelölését és időDiagramját .
7. Ismertesse a logikai TAGADÁS (NEM) kapu jelölését és időDiagramját .
8. Ismertesse a logikai ÉS-NEM kapu jelölését és időDiagramját .
9. Ismertesse a logikai VAGY-NEM kapu jelölését és időDiagramját .
10. Ismertesse a logikai KIZÁRÓ-VAGY kapu jelölését és időDiagramját .
11. Ismertesse a leggyakrabban használt logikai összefüggéseket .
12. Ismertesse a De-Morgan azonosságot .

5. SZÁMRENDSZEREK

5.1. BEVEZETÉS

A számítástechnika alapfeladata a számokkal történő műveletvégzés. Mint már a 3. fejezetben is látható volt az adat az információhordozó. Egy számítógép egyszerre sokféle adattal dolgozik, ezek csoportosítása a a 3.1. - 3.4. táblázatokban látható. A bináris digitális számítógép nemcsak a számok ábrázolását végzi kettes számrendszerben, de az utasítások ábrázolása is ugyanilyen módon történik.

5.2. A TIZES SZÁMRENDSZER

A mindennapi életben használatos számábrázolási mód a tizes számrendszer, ahol a számrendszer alapja a **10**. A tíznél nagyobb számok ábrázolására a helyértékes módszert használjuk, vagyis pl. a 67 szám olvasható úgy is mint 10 x 6 + 1 x 7. A számrendszer jelkészlete a **0,1,2,….8,9**. Egynél kisebb számok ábrázolásakor a tizedespontot (**.**) használjuk. Mint minden számrendszernél, itt is lehetséges tetszőleges nagyságú, illetve pontosságú szám megadása.

5.3. A KETTES (BINÁRIS) SZÁMRENDSZER

A tizes számrendszerhez képest a jelkészlete jóval szegényesebb, csak a **0** és **1**. Használata a mindennapi életben értelmetlen, nehézkes, az igy leírt szám is semmitmondó az ember számára. A számítástecnikában alkalmazása nélkülözhetetlen, egyrészt azért mert sok jelenség eleve a természetben kétértékű (kapcsoló **KI-BE**, állítás **IGAZ-HAMIS** stb.), de műszaki korlátok miatt az elektronikus számítógépekben a legkönnyebben, legolcsóbban megvalósítható megoldás.

A matematikai alapok megegyeznek a tizes számrendszerben használatosakkal, itt is helyértékes számábrázolást használunk. Alkalmas a tizedes számok ábrázolására is. Igaz hogy a kettes számrendszerben is tetszőleges nagyságú és pontosságú szám írható le, de a számrendszer tárgyalásánál figyelembe kell venni a számábrázolásra használt műszaki korlátokat is. Ez az ún. regiszter, vagy a memória-rekesz, ami korlátozott nagyságú (szóhosszúság). Több regiszter összekapcsolásával ez a probléma is megoldható.

Az információ legkisebb alapegysége a **bit** (angol szó, jelentése apró, kicsi), **0** vagy **1** értékű lehet. Több bit egymás mellé írva adja az információ nagyobb alapegységét, a **bájt**ot (byte), amely **8** bitet tartalmaz (5.1 ábra). A nagyobb egység a **kb,** a kilobit, a **kB,** a kilobájt, a **MB**, a megabájt stb., vigyázzunk a **b** és **B** közötti különbségre, a **B** nyolcszor nagyobb érték mint a **b**.

	MSB							**LSB**
hely	**D7**	**D6**	**D5**	**D4**	**D3**	**D3**	**D1**	**D0**
szám	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**

5.1. ábra: A **bájt**

Az ábra jelölései:

- **MSB** - Most Significant Bit - a legnagyobb helyértékű bit,
- **LSB** - Least Significant Bit - a legkisebb helyértékű bit,
- **Di** - A bit jelölése (Data),
- **i** - index a bit jelölésére.

Az egyes bitek mellé rendelt számok nem véletlenül kerültek az adott helyre, ezek a számok egyuttal a helyértéket is jelölik, mint 2 hatványait. Az 5.1. táblázatból jól látható a bitek, illetve a kettes-, valamint a tizes számrendszer közötti kapcsolat.

5.1. táblázat: Kapcsolat a kettes, a tizes és a 16-os számrendszer között					
A bitek jelölése				tizes számrend-szerben	hexadecimális számrend-szerben
D3	**D2**	**D1**	**D0**		
2^3	**2^2**	**2^1**	**2^0**		
8	**4**	**2**	**1**		
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**1**	**1**	**1**
0	**0**	**1**	**0**	**2**	**2**
0	**0**	**1**	**1**	**3**	**3**
0	**1**	**0**	**0**	**4**	**4**
0	**1**	**0**	**1**	**5**	**5**
0	**1**	**1**	**0**	**6**	**6**
0	**1**	**1**	**1**	**7**	**7**
1	**0**	**0**	**0**	**8**	**8**
1	**0**	**0**	**1**	**9**	**9**
1	**0**	**1**	**0**	**10**	**A**
1	**1**	**1**	**1**	**11**	**B**
1	**1**	**0**	**0**	**12**	**C**
1	**1**	**0**	**1**	**13**	**D**
1	**1**	**1**	**0**	**14**	**E**
1	**1**	**1**	**1**	**15**	**F**

Az 5.1. táblázat csak 4 bit összes lehtséges variációjára adja meg a tizes számrendszerben az értékeket. Amennyiben még egy bittel növelnénk a számábrázolási tartományt, vagyis **D4** kerülne **D3** elé, akkor már **0** és **31** közötti számokat tudnánk ábrázolni. Ebbõl következik, hogy egy bit hozzáadása megkétszerezi a számábrázolási tartományt, 2 bit hozzáadása megnégyszerezi stb.

5.4. SZÁMÁBRÁZOLÁS A KETTES (BINÁRIS) SZÁMRENDSZERBEN

Gyakran elegendõ csak pozitiv egész számokat kettes számrendszerben leírni, például ha sorszámot akarunk így kifejezni, vagy osztályzatot stb. Ilyenkor egyszerűen az ábrázolandó szám nagysága határozza meg a szükséges bitek számát, vagyis a regiszter nagyságát.

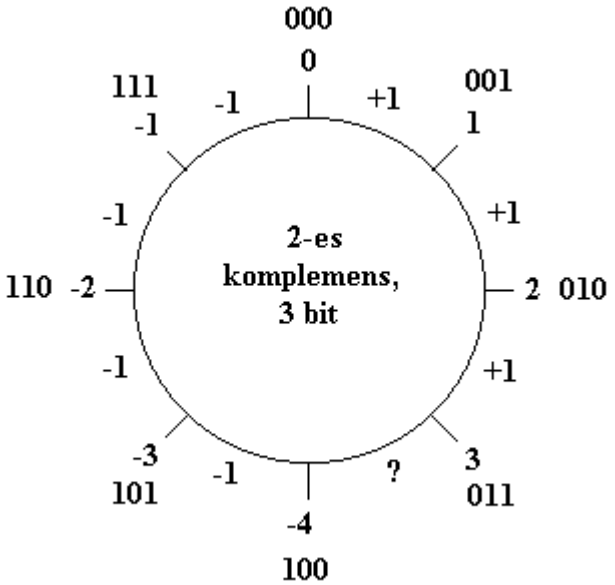
Sokszor a pozitiv számok mellett szükségesek a negatív értékek is, valamint a nullánál kisebb számokkal való műveletvégzés is. A számok elõjeles ábrázolása több módon

oldható meg, leggyakrabban az ún. **előjel és abszolút érték,** illetve a **kettes komplemens** módszer használatos.

5.4.1. **Előjeles, abszolútértékes számábrázolás**

Mivel a digitális bináris számítógépek csak **0** és **1** jeleket különböztetnek meg, ezért az előjel ábrázolására is csak ez a két jel szolgálhat. Megegyezés szerint a **0** a pozitiv (+), mig az **1** a negativ (-) előjelet jelöli. A csak számábrázolásra használt **8** bites regiszter **0** és **255** között összesen **256** különböző számot tud ábrázolni. Az előjel bevezetésével a **8** bitből csak **7** maradt a számokra, ez nem jelenti azonban azt, hogy csökkent volna a rendelkezésre álló egymástól különböző számkombinációk száma, ugyanis most **127** pozitiv, **127** negativ előjeles szám írható az ugyancsak **8** bites regiszterbe (a **0** és **-0** értékekkel együtt).

Vegyük észre, hogy a 2^8 **256**-ot ad, tehát **1** bájton mindig **256** különféle jel ábrázolható. Az előjel nélküli **8** bites számábrázolás **0 - 127** tartománya megegyezik az előjeles, abszolútértékes szánábrázolás pozitiv részével, de az **MSB** helyen levő egyes már *teljesen más jelentést ad a másik 128 számnak.*



5.4.2. **Kettes komplemensű számábrázolás**

Az 5.2. ábra segítségével lehet megérteni a **2-es komplemens** meghatározását és alkalmazását.

5.2. ábra: 3 bit lehetséges összes számkombinációja (2-es komplemens esetén)

A 3 bit-es számot a szokásos számegyenes helyett 'számkörön' ábrázolva látjuk, hogy 8 különböző értéket kapunk. Ez azt jelenti, hogy míg egy számegyenes a minusz

végtelen és plusz végtelen között bármely számot képes bemutatni, addig egy regiszter a korlátozott kapacitása miatt 'körbefut'.

Induljunk el a **0** számból (tizes számrendszer), ami binárisan **000**. Ehhez egyet hozzáadva megkapjuk az **1** értéke, ami binárisan **001**. Folytatva az eljárást. eljutunk 3-ig (**011**), amiből látható, hogy a szám pozitív, és a lehető legnagyobb szám ami még beírható a regiszterbe.

Most induljunk el újból a **0**-ból, de vegyünk el 1-et belőle. Tizes számrendszerben ez egyszerű feladat, a kettes számrendszerben egy kis kitérőt kell tenni. Bevezetve az **egyes** komplemens fogalmát megkapjuk az 5.3. ábra szerinti értéket, mégpedig úgy, hogy minden bitet az ellenkezőjére, vagyis a komplemensére változtatunk.

	MSB						**LSB**
hely	**D7**	**D6**	**D5**	**D4**	**D3**	**D3**	**D1**	**D0**
szám	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**
1-s komplemens	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**1**
+1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
2-es komplemens	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**0**

5.3. ábra: A bájt , 1-es komplemens és 2-es komplemens

Ekkor az igy kapott **1-es komplemens**ből **1** hozzáadásával (ami kettes számrendszerben 00000001) megkapható az eredeti szám **2-es komplemense**, ami egyuttal az adott szám negativ értéke is. Az 5.4. ábrán látható hogy hogyan határozható meg **5**-ből **-5** értéke.

	MSB							**LSB**
hely	**D7**	**D6**	**D5**	**D4**	**D3**	**D3**	**D1**	**D0**
5	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**
1-s komplemens	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**0**
+1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
-5	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**1**

5.4. ábra: 5 , 1-es komplemense és -5

Hogyan bizonyítható be az, hogy a **11111011** valóban **-5?** Könnyű belátni ezt, hiszen ha a **2-es komplemens**nek megkeressük ismét a kettes komplemensét, akkor visszakapjuk az eredeti számot, 5.5. ábra.

	MSB							**LSB**
hely	**D7**	**D6**	**D5**	**D4**	**D3**	**D3**	**D1**	**D0**
-5	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**1**
1-s komplemens	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
+1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
5	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**

5.5. ábra: -5 , 1-es komplemense és 5 (-5 2-es komplemense)

Ugyanígy bebizonyítható **11111011**-ről, hogy **-5** a **5 + (-5)** művelet elvégzésével, ugyanis **0**-át kell kapni, ez látható az 5.6. ábrán.

	MSB							**LSB**
hely	**D7**	**D6**	**D5**	**D4**	**D3**	**D3**	**D1**	**D0**
-5	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**1**
+5	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

5.6. ábra: -5 + 5 = 0

A **D7** helyen történő összeadás egy átvitelt okoz, ami tulajdonképpen elveszik, hisz nincs megjegyzésére fenntartott bit.

5.5. A TIZENHATOS (HEXADECIMÁLIS) SZÁMRENDSZER

A számrendszer alpja a **16**, igy 16 különféle jelet használ az értékek megkülönböztetésére. A **10**-es és **16**-os számrendszerek közötti kapcsolatot az 5.1 táblázatban találhatjuk meg. Mivel nincs külön jelkészletünk a 9 feletti számokra, ezért megegyezés szerint az **ABC** első 6 betűjét használjuk fel. Használata a számítástechnikában igen előnyös, ugyanis ha 4 bitet összekapcsolunk, akkor éppen egy 16-os értéket kapunk (ugyancsak 5.1 táblázat). A kettes számrendszeren alapul a ma használatos bináris digitális számítógépek működése, de a kettes számrendszerrel való programozás igen nehézkes (vagy majdnem teljesen lehetetlen) lenne, ezért a **16**-os számrendszert csak mint segédjelölést használjuk. Például a **1100|1001|0101|1011** memóriacím egyszerűen **C95B** hexadecimálisan.

Ellenőrző kérdések

1. Milyen számrendszereket használunk a leggyakrabban a mindennapi életben és a számítástecnikában ?
2. Ismertesse a 10-es számrendszer tulajdonságait, a műveletvégzések szabályait .
3. Ismertesse a 2-es számrendszer tulajdonságait, a műveletvégzések szabályait .
4. Ismertesse a 16-os számrendszer tulajdonságait, a műveletvégzések szabályait .
5. Mi a kapcsolat a kettes és a tizenhatos számrendszer között ?
6. Ismertesse a BCD számábrázolás tulajdonságait, a műveletvégzések szabályait .
7. Mi a bit, mi a bájt (byte) és mekkora a számábrázolási tartománya ?
8. Mi az LSB és MSB, a bájt bitjeit milyen szabály szerint indexeljük ?
9. Az előjeles számábrázolás mely technikáit ismeri ?
5. Előjeles, abszolútértékes számábrázolás .
6. Kettes komplemensű számábrázolás .
7. Ismertesse a tizes számrendszerből kettesbe való átalakítás szabályait .
8. Ismertesse a kettes számrendszerből tizesbe való átalakítás szabályait .
9. Ismertesse a tizenhatos számrendszerből kettesbe való átalakítás szabályait .
10. Ismertesse a kettes számrendszerből tizenhatosba való átalakítás szabályait .
11. Ismertesse a tizes számrendszerből tizenhatosba való átalakítás szabályait .
12. Ismertesse a tizenhatos számrendszerből tizesbe való átalakítás szabályait .
13. Ismertesse az egynél kisebb számok ábrázolását kettes számrendszerben.

14. Ismertesse az egynél kisebb negativ számok ábrázolását .
15. Mi határozza meg kettes számrendszerben az ábrázolható szám nagyságát ?
16. Mi határozza meg kettes számrendszerben az ábrázolható szám pontosságát ?

6. SZÁMÍTÓGÉPEK

6.1. SZÁMÍTÓGÉPHARDVER FEJLŐDÉSÉNEK ÁTTEKINTÉSE

A számítástechnika, így a számítógépek fejlődése során több ún. számítógépgeneráció alakult ki. Az egyes generációk meghatározása az alapáramkörök típusa, valamint a műveleti sebesség szerint történik. Teljesen pontos határt soha sem húzhatunk a számítógépgenerációk közé, előfordulhat átfedés.

A 6.1. számú táblázatban láthatók az egyes számítógépgenerációk.

6.1. táblázat: számítógépgenerációk					
tulajdonságok	1. generáció 1950-1955	2. generáció 1956-1964	3. generáció 1965-1977	4. generáció 1978-1990	5. generáció 1991-2005
művelet/sec	10 ezer	1 millió	10 millió	100 millió	1000 millió
operatív memória [bájt]	50 ezer	500 ezer	10 millió	100 millió	1000 millió
hardver elemek	elektroncső, mágnesdob, szalag, relé	tranzisztor, mágneslemez, ferritmátrix	integrált áramkör, félvezetőtár	LSI áramkör, mikroprocesszor, merevlemez	VLSI áramkör, programozható áramkörök
háttértárkapacitás [bájt]	10 MB	**1000 MB**	10 GB	100 GB	1000 GB
szoftver elemek	gépi kód, programkönyvtár	operációs rendszer, programnyelvek	többfelhasználós operációs rendszer, fejlett programnyelvek, adatbázisok	egységes, hordozható operációs rendszer, hálózati szoftver, 4GL programnyelvek	világhálózati szoftver, multimédia, mesterséges intelligencia
Jelölések: M- mega, G-giga					

Az 1. táblázatban megadott adatok maximális értékek, egy adott gép szerényebb kapacitásokkal is tartozhat az adott csoportba.

Új áramköri elemek kifejlesztése minőségi ugrást eredményez a számítástechnika fejlődésében, igaz ezen eszközök kifejlesztése nagy összegeket emészt fel. Egy számítógép teljesítményének növelése nemcsak új elemekkel érhető el, hanem rendszertechnikai megoldásokkal is.

A 6.2. táblázatban néhány példát találhatunk a fejlődés lépéseire.

6.2. táblázat: néhány példa a fejlődésre	
generáció	példa a fejlődésre
1.	Neumann elvű számítógép
2.	I/O processzorok, komplex utasításkészlet, verem,
3.	mikroprogram vezérlés,virtuális tároló
4.	mikroprocesszorok, RISC processzorok, párhuzamosítás az utasítás szintjén
5.	számítógéphálózatok, nagymértékű párhuzamosítás

6.2. SZÁMÍTÓGÉPEK CSOPORTOSÍTÁSA

A számítástechnika fejlődése során az elmélet fejlődése, a technológiák kialakulása, az igények változása mind megszabta bizonyos megoldások fejlődését, más számítástechnikai eszközök eltünését a piacról.

Ahhoz hogy áttekintésünk legyen arról, hogy bizonyos számítógépek mely fejlődési irányt követik, bizonyos tulajdonságokat kell meghatározni a gépeknél, amely tulajdonságok alapján aztán besorolhatjuk a gépet egy-egy csoportba. Sokszor nem egyértelmű ez a csoportosítás, de az is gyakran előfordul, hogy más szempontok előtérbe helyezése más csoportba sorolja a gépeket.

Néhány csoportosítási szempont:

- a műveleti sebesség: időegység alatt feldogozott műveletszám, ahol használatos a MIPS (Millions of Instructions Per Second – hány millió utasítást hajt végre a gép egy másodpercp alatt), a MOPS (Millions of Operations Per Second – hány millió műveletet hajt végre a gép egy másodpercp alatt) és az MFLOPS (Millions of Floating Point Operations Per Second - hány millió lebegőpontos aritmetikai műveletet hajt végre a gép egy másodpercp alatt) mértékegység.
- órajelfrekvencia: a gép órajele szinkronizációs feladatok elvégzése mellett biztosítja a párhuzamosan végrehajtható műveletek egymásmellettiségét, de nem utolsó sorban korlátozza a számítógép legnagyobb működési sebességét. Jelenleg ez az adat néhány MHz frekvenciától az 1 GHz-ig terjed (adat 2000-ben).
- az áramköri egységek, elemek technológiája: olyan tényező, amely az egyes egységek, de a teljes gép működési sebességét, megbízhatóságát, fogyasztását, méretét, tervezési, gyártási folyamatát, a használhatóság hőmérsékleti tartományát stb. megszabja.
- a külső és a belső sinrendszer szélessége, felépítése: vagyis a párhuzamos, egyidejűleg átvihető adtszélesség.
- utasítások, műveletek időbeli átlapolhatósága: párhuzamosan a műveletek, vagy azok részei hogyan hajthatók végre.
- szóhosszúság: utasításvégrehajtáskor mekkora az a szóhosszúság, amit a gép egyszerre, párhuzamosan kezel.
- memória-sebesség: a memória ciklusideje és a sinrendszer szélessége.
- perifériák-sebessége: a memóriák adatátviteli sebessége, amely a perifériák működési sebességétől, valamint a perifériavezérlők kapacitásától függ.

Ha a számítógépeket sebességük és teljesítőképességük alapján szeretnénk csoportosítani, akkor azt is figyelembe kell venni, hogy a gyors fejlődés miatt az egyes paraméterek olyan mértékben változnak, hogy a még nemrégen nagygépnek számító számítógép paramétereit messze túlszárnyalja egy közepes gép, amely új termékként jelentkezik. Három csoportot különböztethetünk meg:

- nagygépek (mainframe vagy super computer): ezek igen nagy műveleti sebességgel, nagy kapacitású tárolóval és nagy teljesítményű perifériákkal rendelkező gépek. Fő alakalmazási területük a sok adat feldolgozása, vagy a nagybonyolultságú, pl. tudományos számítások. Mivel teljesítményük igen nagy, ezért a rajtuk futó operációs rendszer is nagy teljesítményű, sok felhasználó számára teszi lehetővé az egyidejű géphezférést. Igen gyakran teljesítményük miatt számítógépes halózatok központi gépeként szerepelnek.
- közepes (mini) gépek: a nagygépek teljesítménye, memóriakapacitása és periféria száma, bonyolultsága alatt helyezkednek el. Kevesebb adattal, kevesebb felhasználóval dolgoznak. Különösen előnyös használatuk az iparban, ahol folyamatvezérlési, adatgyűjtési és termelésirányítási feladatok végzésére kiválóan alkalmasak. Ha munkaállomások kapcsolódnak a mini számítógéphez, akkor ezen különböző CAD programok futhatnak, térinformatikai problémákat lehet velük hatásosan megoldani.
- kisgépek (mikroszámítógépek): sokszor önállóan alkalmazott igen egyszerű gépektől kezdve hálózatba kötött munkaállomásokig terjed ezen gépek kategóriája. Amikor egyszerű feladatok elvégzése a cél, akkor igen hatásosan alkalmazhatóak önálló vezérlőként, adatgyűjtő, szabályozó, irányító berendezésekben helyi adatfeldolgozó, átalakító egységenként. Komolyabb változataik személyi számítógépként munkahelyen, otthon igen széles feladatkört látnak el. Kapcsolhatók hálózatba, gyakorlatilag az Interneten keresztül a világra kiterjedő igen bonyolult rendszer részei.

Abban az esetben, ha a számítógépnél az ún. utasításfolyamot és adatfolyamot vizsgaljuk, akkor más csoportosítást kapunk. A kezelt folyamok száma alapján a következő négy csoportot kapjuk, ahova a bonyolultabb számítógép architektúrák is besorolhatók:

- **SISD** (Single Instruction Stream Single Data Stream): egy utasításfolyamhoz egy adatfolyam párosul. Ezekben a processzorokban (tehát számítógépekben) csak egy vezérlő egység és egyetlen egy aritmetikai egység található. Ez azt jelenti, hogy egyszerre csak egy utasítás végrehajtására képesek. Tulajdonképpen a Neumann elvű gépek tartoznak ide.
- **SIMD** (Single Instruction Stream Multiple Data Stream): egy utasításfolyam mellett több adatfolyam végrehajtása. Itt egy vezérlő egységhez több aritmetikai egység párosul. Egy utasítás végrehajtására kerül sor több adaton, természetesen egyidőben. Ide sorolhatók a vektor- és tömbprocesszoros gépek.
- **MISD** (Multiple Instruction Stream Single Data Stream): tehét több utasításfolyam mellett egy adatfolyam van, ezek a gépek nem léteznek, néha ebbe a csoportba soroljuk a pipeline feldolgozást alkalmazó gépeket.
- **MIMD** (Multiple Instruction Stream Multiple Data Stream): több utasításfolyam és több adatfolyam jellemze a csoportba tartozó gépeket, amik a különböző multiprocesszoros megoldások, vagyis több processzort tartalmazó számítógépek.

A nagygépek és a mikrogépek között a következő különbségek fedezhetők fel:

- a nagygépek teljesítménye, perifériáinak száma nagyobb, de a perifériák teljesítménye is lényegesen meghaladja a mikrogépek perifériának teljesítményét,
- a nagygépek processzora több chipből áll, a mikrogépeknél a vezérlőegység és aritmetikai egység egy tokban helyezkedik el,
- a nagygépekhez alkalmazott perifériák megbízhatósága, teljesítménye lényegesen nagyobb, mint a mikroszámítógépeknél alkalmazottaké és
- a mikrogépek egy házban elhelyezett berendezések, a nagygépek ellenben szekrény, több szekrény méretűek is lehetnek.

Ami közös a nagygépeknél és a kisgépeknél is az az, hogy mindkettő nagy integráltsagi fokú elemekből (VLSI) épül fel.

Ellenőrző kérdések

1. Mi a számítógépgeneráció ?
2. Sorolja fel a számítógépgenerációk jellemző paramétereit .
3. Mi szabja meg egy új számítógépgeneráció fejlődését ?
4. Sorolja fel a számítógépek csoportosításának szempontjait .
5. Hogyan oszthatjuk fel a számítógépeket, ha sebességük és teljesítményük alapján készítjük a csoportosítást .
6. Mi jellemzi a nagygépeket ?
7. Mi jellemzi a Minigépeket ?
8. Mi jellemzi a mikroszámítógépeket ?
9. Hogyan csoportosíthatjuk a gépeket, ha az utasításfolyamot és az adatfolyamot vesszük figyelembe ?
10. Mi jellemzi a SISD számítógépeket ?
11. Mi jellemzi a SIMD számítógépeket ?
12. Mi jellemzi a MISD számítógépeket ?
13. Mi jellemzi a MIMD számítógépeket ?
14. Milyen lényeges különbségek fedezhetők fel a kisszámítógépek és nagyszámítógépek között ?
15. Mi a hasonlóság a kisgépek és a nagygépek között ?

7. JELLEGZETES DIGITÁLIS SZÁMÍTÓGÉPARCHITEKTÚRÁK

7.1. BEVEZETÉS

Számítógépeket különböző elvek alapján építenek, sokszor igen nagy elvi különbség van az egyes megoldások között, de vannak közös megoldások is. Az alkalmazási terület, különféle igény, ár, megbízhatóság stb. szabja meg, hol mit alkalmazunk.

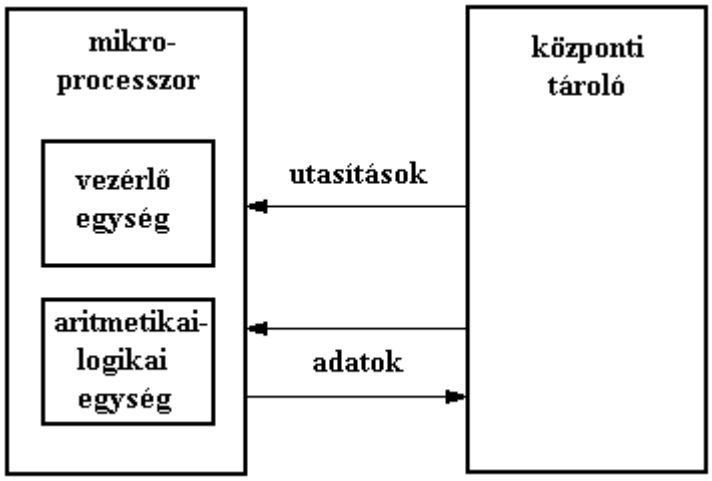
7.2. NEUMANN-FÉLE SZÁMÍTÓGÉPEK

Legelterjedtebb architektúra, valójában ez a struktúra alakult ki a legkorábban, legtöbb alkalmazásban ezt használják.

Legfontosabb tulajdonságok:

- tárolt program alapján történő gépvezérlés.
- a vezérlő egység egy regiszterben tárolja a következő utasítás memóriabeli (tároló) címét, egyszerre egy utasítást hajt végre a gép, az utasítások egymás után követik egymást, ezt az elvet nevezzük vezérlésáramlásos irányításnak (control-flow),
- a gép szerves része az operativ memória, amely bináris alakban tárolja a végrehajtandó programot és azokat az adatokat, amelyeken műveleteket kell végrehajtani,
- az önálló aritmetikai-logikai egység hajtja végre az aritmetikai és a logikai műveleteket is,
- adatbevitel, kivitel önálló egységeken keresztül történik.

A 7.1. ábrán a gép elvi felépítése látható. A gép problémája az operativ tár és processzor közötti adatáramlás, ugyanis a memória sebessége a soros utasítás- és adatáramlás miatt lassítólag hat a rendszer működésére. Ez a szűk keresztmetszet azért alakul ki, mert egy sinrendszere létezik a rendszeren belül, ahol az utasítások és az adatok ugyanazon vezetékeket használják.



7.1. ábra: A Neumann-féle számítógép tárolókapcsolata

Ezek a gépek a **SISD** (egy uatsításfolyam egy adatfolyam) csoportba tartoznak, úgyhogy elég kevés lehetőség van a párhuzamosítási folyamatokra. Párhuzamosítani a következő módon lehet:

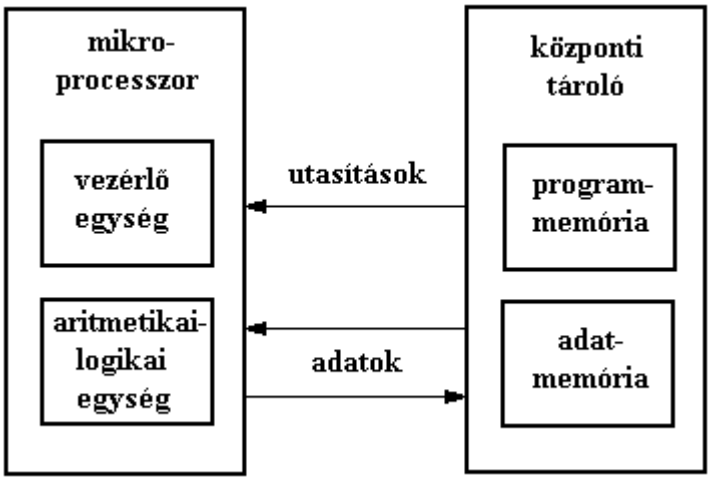
- multiprogramozott üzemmód, vagyis több program egyidejű futtatása, ami az egyes egységek egyenletesebb terhelését biztosítja,
- multifunkciós processzorok tervezése, ahol, ha mód van rá bizonyos egységeket többszöröznek,
- átlapolási (spooling) technika, ahol a processzor- és I/O művelet átlapolt végrehajtását hozzák létre és
- pipelining, utasításvégrehajtás és aritmetikai műveletek átlapolása.

A Neumann-típusú gépeknél meg kell említeni azt a lehetőséget, hogy az egységes operatív, vagy munkamemória lehetővé teszi a program működés közbeni felülírását. Ez előnyös az önmódosító (re-entrante kód) programok írásánál, ugyanakkor veszélyes a véletlen felülírásoknál, hiszen ez hibás működést okoz.

7.3. HARVARD STRUKTÚRÁJÚ SZAMÍTÓGÉPEK

A Harvard struktúrájú gépek felépítése megegyezik a Neumann gépekével, egyetlen helyen van eltérés. A programtárolási és adattárolási funkciók szétválasztása külön sinrendszert igényel, melyek párhuzamosan, egyidőben működhetnek, ez gépteljesítmény-növekedést okoz.

Folyamatkezelési szempontból a Harvard struktúra is SISD típusú gép. A Harvard struktúra a 7.2. ábrán látható.



7.2.ábra: Harvard típusú gép

A program- és adatmemória szétválasztása lehetővé teszi optimális utasítás- és adatszóhosszúság kialakítását, ami tovább növeli a hatékonyságot. A szétválasztás nem engedélyezi a programok működés közbeni módosítását, mint ahogy ez lehetséges a Neumann gépeknél.

7.4. VEKTORSZÁMÍTÓGÉPEK

A matematikai-tudományos számításoknál gyakran szerepelnek számsorozatok, vektorok. A vektorműveletekre jellemző, hogy ugyanazt a műveletet kell egymás után sok adaton elvégezni. Az adatsoron végzett műveleteket át lehet lapolni, vagyis ún. pipelining technikát alaklmazni. Ez a gép a **SIMD** csoportba sorolható, ami egy utasításfolyam, több adatfolyam technika.

A gép teljesítménynövelésének két oka van:

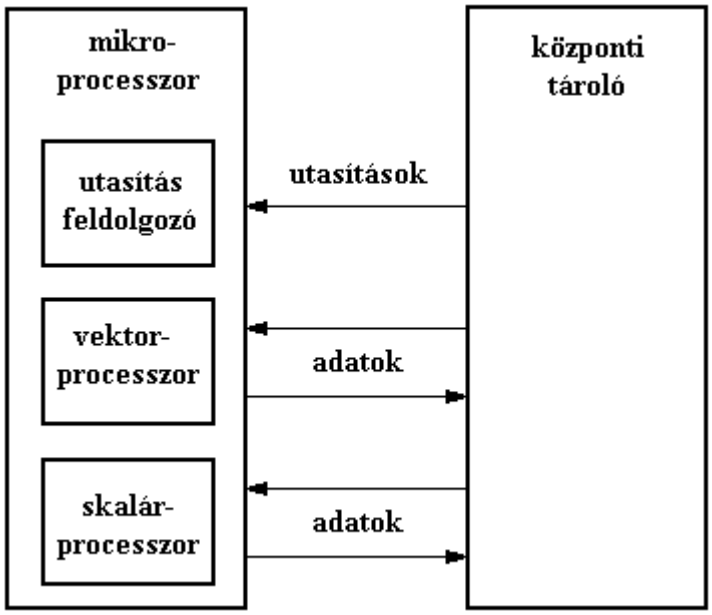
- ugyanazt a műveletet kell elvégezni egymás után sok adaton, tehát a memóriából csak egyszer kell lehívni az utasítást,
- valamint a vektorelemek közötti műveletvégzés átlapolható (függetlenek egymástól feldolgozás szempontjából ezek az elemek), külön vektorutasítások biztosítják ezt a datapipeline (adatpipeline) technikát.

Belátható, hogy a gép teljesítménye a feldolgozandó vektor hosszának függvénye, ugyanakkor itt is jelentkezik az a probléma, hogy az adatok memóriából való lehívása és visszaírása csökkenti a feldolgozási sebességet.

Ennek a problémának a megoldása két módon történhet:

- egy ideiglenes memória, a memórialánc alkalmazása és
- a memória több részre való felosztása (memory interleaving), a vektorelemek lehívása sorra az egymást követő memóriablokkokból történik.

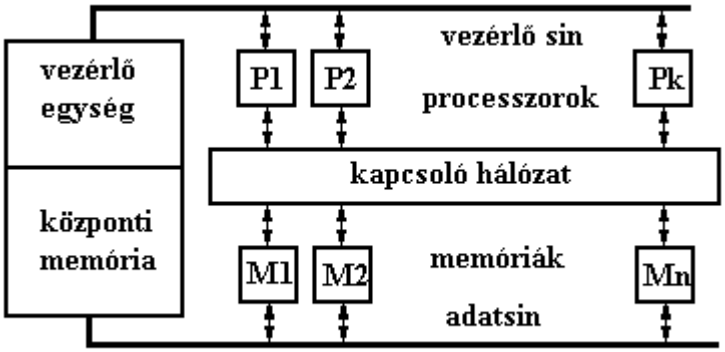
Sokszor a vektorok mellett skaláris adatokkal is kell dolgozni, ami nagyon rontja a vektorprocesszorok hatékonyságát. Ezért a vektorprocesszorok mellett a processzor skalárprocesszort is tartalmaz. A vektorprocesszor tárolókapcsolata a 7.3. ábrán látható.



7.3. ábra: A vektorprocesszor tárolókapcsolata

7.5. TÖMBPROCESSZOROS SZÁMÍTÓGÉPEK

Tulajdonképpen a tömbprocesszoros gépek a vektorszámítógépek továbbfejlesztett változatai. Ezek a gépek több processzort, vagyis műveletvégző egységet tartalmaznak, valamint több memóriamodult is. A gép tárolókezelése a 7.4. ábrán látható.



7.4. ábra: Tömbprocesszoros számítógép tárolókezelése

A vektorszámítógépekhez képest további sebességnövekedést lehet elérni, ha minden processzoron ugyanazt a műveletet hajtja végre a gép a vektorok, illetve mátrixok különböző elemeivel. Így ezek a gépek a **SIMD** csoportba sorolhatók, vagyis egy utasításfolyam, több adatfolyam létezik.

A processzorok és memóriák összekapcsolását egy kapcsolóhálózat biztosítja, ez azt biztosítja, hogy bármelyik processzor bármelyik memóriához kapcsolódhassson.

A tömbprocesszoroknak három fő csoportja létezik:

- array computer (tömbprocesszor) – a processzzorok ugyanazon utasítás végrehajtásán egymástól függetlenül dolgoznak, az adatokat a hozzájuk rendelt memóriából veszik és teszik vissza, ha szükséges más egységhez az adatokat a kapcsoló hálózaton keresztül továbbítják.
- associative processor (asszociatív processzor) – az adatok visszakeresése nem cím, hanem tartalom szerint történik, az ilyen megoldás előnyös az összehasonlítási, hasonlósági feladatok megoldásánál, alakfelismerési problémáknál.
- systolic array (szisztolikus tömbszámítógép) – a processzorok csak a szomszédos egységekkel vannak kapcsolatban, a kapott eredményeket csak a szomszédos egységeknek tudják további feldolgozási céllal továbbadni. Ebből következik, hogy a külvilággal a kapcsolattartást csak a szélső elemek tudják ellátni.

7.6. ÜZENETÁTADÁSOS SZÁMÍTÓGÉPEK

Ha olyan architektúrát lehet kialakítani, ahol több, önálló feladatvégzésre alkalmas processzor kerül alkalmazásra, akkor tovább nő a teljesítőképesség.

Multiprocesszoros rendszerekben két fő kérdés merül fel:

- a processzor hogyan használja a memóriákat, vagyis egy adott memóriát használhat-e több processzor (shared memory), vagy csak egy processzor használhatja (disjoint, distributed memory).
- processzorok közötti kapcsolattartás módja, van-e megosztott buszhasználat (shared bus system), illetve van-e közvetlen kapcsolat kiépítve (interconnection network).

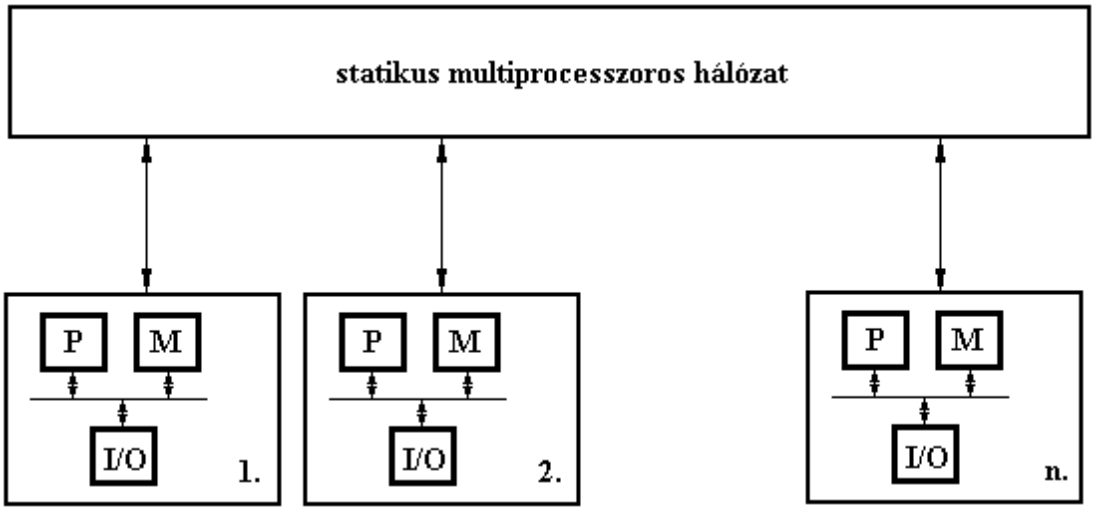
A multiprocesszoros számítógépek **MIMD** gépek, vagyis több utasításfolyam, több adatfolyam típusúak.

A processzorok, illetve a processzorok és megosztott használatú memóriák közötti kapcsolat lehet:

- statikus – csomóponti egységek (processzorok) közötti állandó struktúrájú kiépítás és
- dinamikus – csomópontok közötti kapcsolat igény szerinti kiépítése.

Célszerű a csomóponti egységek között egységes üzenettovábbítási rendszert kialakítani. Ez megvalósítható az egységes **üzenetek** létrehozásával. Minden üzenet tartalmazza a fogadó proceszor (cél) címét, úgyszintén a küldőét (forrás), az adatokkal és kiegészítő információkkal együtt.

Üzenetádasos elven megvalósított számítógép (multiprocesszoros gép) statikus kapcsolati rendszerrel látható az 7.5. ábrán.



7.5. ábra: Multiprocesszoros számítógép statikus kapcsolattal

A 7.5. ábra szerinti megoldásnál létre lehet hozni egy-, két-, három-, vagy többdimenziós (hiperkocka) csomópont elrendezéseket. Látható, hogy a csomópontok

egyformák, mindegyik rendelkezik I/O kapcsolattal, ami hatásos I/O adatátvitelt biztosít.
Az egychipes transzpúter a processzor, memória és I/O egység mellett még tartalmazhat DMA (közvetlen memóriahozzáférés) egységet, Tetszőleges számú transzpúter összeépítésével igen hatásos, nagyteljesítményű multiprocesszoros számítógépet lehet kiépíteni.

7.7. ADATVEZÉRELT SZÁMÍTÓGÉPEK

Az adatvezérelt (data-flow, data driven) elvű gépeknél a műveletek végrehajtása az adatok rendelkezésre-állásától függ. Ez a működési elv ellentétes a Neumann-gépek vezérlésáramlásos elvétől.

Az adatáramlási gráf határozza meg a gép logikai struktúráját és az elvégzendő műveletek egymáshoz kapcsolódását. Az adatáramlási gráf csomópontjaihoz (nodes) rendelik hozzá az elvégzendő műveleteket.

A hozzárendelés lehet:

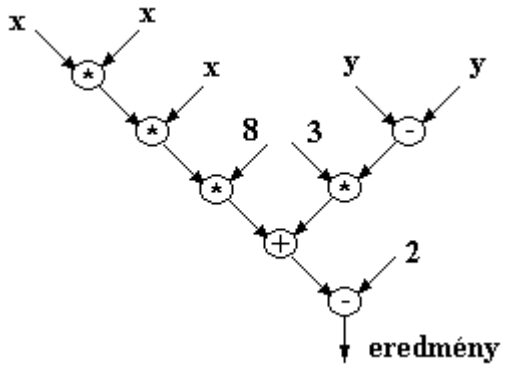
- elemi aritmetikai művelet,
- elemi logikai művelet,
- választás több adat közül és
- adat valamilyen irányba való továbbítása.

A gráf élei az adatok (token-ek):

- **az igényelt adat,**
- **az eredményként kapott adat.**

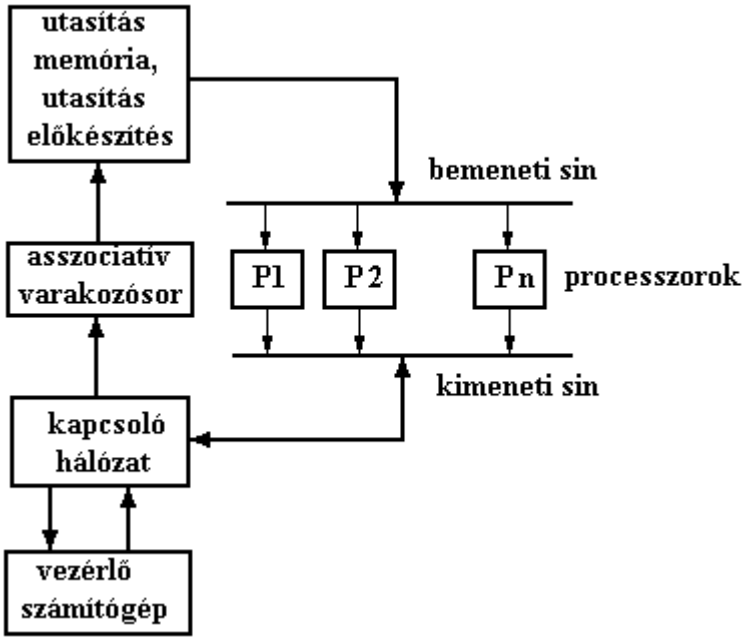
A fordítóprogram az adatáramlási gráf alapján állít elő programgráfot. Maga a számítógép egy multiprocesszoros gép, ahol a processzorokat kapcsolóhálózat köti össze, minden lépésben az adatok egy-egy processzorpár között mozognak az adatáramlás szerint előírt módon.

A 7.6. ábrán a $z = 8x^3 + 3y^2 - 2$ kifejezés adatáramlási gráfja látható.



7.6. ábra: A $z = 8x^3 + 3y^2 - 2$ kifejezés adatáramlási gráfja

Az adatvezérelt számítógép blokkvázlata a 7.7. ábrán látható.



7.7. ábra: Adatvezérelt szémítógép blokkvázlata

Ellenőrző kérdések

8. Sorolja fel milyen számítógép-architektúrákat ismer .
9. Ismertesse a Neumann típusú számítógépet .
10. Mi a Neumann számítógépek legfőbb tulajdonsága ?
11. Rajzolja le a Neumann számítógép blokkvázlatát .
12. Mi a szűk keresztmetszet a Neumann-féle számítógépnél ?
13. Melyik számítógépcsoportba atrtozik a Neumann gép ?
14. Hol nyílik alkalom a folyamatok párhuzamosítására a Neumann számítógépeknél?
15. Mi a programfelülírás, lehetséges-e a Neumann gépeknél, ha igen, mi az előnye, mi a hátránya a módszernek ?
16. Ismertesse a Harvard típusú számítógépet .
17. Mi a Harvard számítógépek legfőbb tulajdonsága ?
18. Rajzolja le a Harvard számítógép blokkvázlatát .
19. Hogyan, mi miatt dolgozik gyorsabban a Harvard gép, mint a Neumann ?
20. Melyik számítógépcsoportba atrtozik a Harvard gép ?
21. Hol nyílik alkalom a folyamatok párhuzamosítására a Harvard számítógépeknél ?
22. Mi a programfelülírás, lehetséges-e a Harvard gépeknél, ha igen, mi az előnye, mi a hátránya a módszernek ?
23. Ismertesse a vektorszámítógépet .
24. Milyen alkalmazásokban célszerű a gép használata ?
25. Mi a vektorműveletek jellegzetessége ?
26. Mi a műveletátlapolás (pipelining)?
27. Mi az oka a számítógép teljesítménynövekedésének ?
28. Hogyan függ össze a gép teljesítménye a vektor hosszával ?
29. Mi a vektorszámítógépeknél az alapvető probléma ?
30. **Hogyan küszöbölhető ki az a probléma, hogy az adatok memóriából olvasása, valamint visszaírása lassítja a vektorszámítógép működési sebességét ?**

31. Rajzolja le a vektorszámítógép blokkvázlatát .
32. Mi a tömbprocesszoros számítógép ?
33. Rajzolja le a tömbprocesszoros számítógép blokkvázlatát .
34. Hogyan lehet tömbprocesszoros számítógépeknél sebességnövekedést elérni a vektorszámítógépekhez képest ?
35. Melyik számítógépcsoportba tartozik a tömbprocesszoros számítógép ?
36. Hányféle tömbprocesszoros számítógépet ismer ?
37. Mi az array computer (tömbprocesszor) ?
38. Ma az associative processor (asszociativ processzor) ?
39. Mi a systolic array (szisztolikus tömbszámítógép) ?
40. Mi az üzenetátadásos számítógép működési elve ?
41. **Milyen két fő kérdés merül fel a multiprocesszoros gépek memóriahasználatánál ?**
42. A multiprocesszoros számítógépek melyik számítógépcsoportba sorolhatók ?
43. Milyen a kapcsolat a processzorok és a megosztott használatú memóriák között a multiprocesszoros gépeknél ?
44. Hogyan történik a csomóponti egységek közötti datkapcsolat ?
45. Rajzolja le az üzenetádasos elven megvalósított számítógép (multiprocesszoros gép) statikus kapcsolati rendszerrel ellátott megoldását .
46. Mi az egy-, két-, három-, vagy többdimenziós (hiperkocka) csomóponti elrendezés ?
47. A csomópontok I/O kapcsolata milyen előnnyel jár a multiprocesszoros gépeknél ?
48. Mi az adatvezérelt számítógép ?
49. Mi határozza meg a gép logikai struktúráját és az elvégzendő műveletek egymáshoz kapcsolódását ?
50. Az adatáramlási gráf csomópontjaihoz milyen műveleteket lehet hozzárendelni ?
51. Mik a gráf élei ?
52. A fordítóprogram mit állít élő az adatáramlási gráfból ?
53. Mutasson be egy példán egy adatáramlási gráfot .
54. Rajzolja le az adatvezérelt számítógép blokkvázlatát .

8. SZÁMÍTÓGÉPEK UTASÍTÁSKÉSZLETE

8.1. BEVEZETÉS

A processzor hardverja végrehajtja a gépi kódú utasításokat. Egy adott processzorhoz a gyártó meghatározza az utasításokat, az utasításkészletet.

A felhasználó függetlenül attól, hogy milyen szinten megírt szoftverrel használja a gépet, mindig csak az adott utasításkészlet elemeit, a gépi szintű utasításokat használja. Ez a legalsó szint a felhasználó számára, az ezen szint alatti lépések végrehajtása már a processzorban levő vezérlőegység feladatát képezi. Ez lehet, mint ahogyan azt később látni fogjuk huzalozott logika, illetve mikroprogram segítségével megvalósított lépés.

8.2. MIKROPROCESSZOROK UTASÍTÁSKÉSZLETE

A mikroprocesszorok összehasonlításánál fontos elemezni a géphez tartozó utasításkészletet, ennek lehetőségei hozzájárulnak egy gép hatékonyságához.

Az utasításkészlet jellemzésére használhatók a következő kritériumok:

- az elemi utasítások száma, valamint az utasítások tartalma, vagyis milyen feladatot oldanak meg. Ha egy utasításkészlet több utasítással rendelkezik, akkor az adott processzor is sokoldalúbban használható fel. Igaz, ilyenkor a programozó a sok utasítás közül egyeseket gyakrabban, másokat ritkábban, de vannak olyanok is, amelyeket egyáltalán nem használ. Ez azt jelenti, hogy a chip gyakorlatilag felesleges információt tartalmaz. Ha egy utasítás összetett feladatokat képes elvégezni, csökken a program hossza, de a feldolgozási idő nő az összetett utasítások miatt.
- az utasításokkal kezelhető feltételek száma hatékonyabbá teszi a lefordított programot, igaz a fordítóprogramot megbonyolítja.
- mennyire egységes a különböző utasításoknál a feltételek kezelése, címzési módok alkalmazása, vagyis minél kevesebb kivétel legyen.
- mennyire támogatja az utasításkészlet a programozást, a programok fordítását és a programok ellenőrzési lehetőségét. Tulajdonképpen ez azt jelenti, hogy a gyakran előforduló feladatnak, lépésnek legyen utasítása. Fontos, hogy a forrásprogramot a fordító egyszerűen tudja gépi kódra lefordítani. Elvárható, hogy az utasításkészlet tarttalmazzon olyan utasításokat, melyek alkalmasak a program futási állapotának, a gép egyes részeinek működés közbeni ellenőrzését.

Nincs szabály arra, hogy milyen utasításkészletet kell kialakítani egy-egy hardver megoldáshoz, de az évek során felhalmozott tapasztalati tények többé-kevésbé behatárolják azon feladatokat, amiket egy adott processzornak meg kell oldani. Így kialakult két típusú utasításkészlet:

- az összetett utasításkészlet (**CISC** Complex Instruction Set Computer) és
- a csökkentett utasításkészlet (**RISC** – Reduced Instruction Set Computer).

Ha elemezzük az utasításkészletek kialakulását, észre kell venni, hogy a számítástechnika indulásakor egyszerű és kevésszámú utasítás jellemzett egy-egy processzort, tehát RISC típusú utasításkészlettel rendelkeztek a gépek. A hardver

fejlődése, a programozási technikák, az igények a fejlődést a CISC **típusú utasításkészlet megalkotására kényszerítették. Ugyanakkor a hatékony szoftvertechnikák, fordítóprogramok fejlődése visszahatott a hardver fejlesztésre is és igen hatásos** RISC **architektúrák kialakulását tették lehetővé.**
A 8.1. táblázat tartalmazza a kétféle utasításkészlet összehasonlítását.

8.1. táblázat: A CISC és RISC processzorok utasításkészletének összehasonlítása		
processzor	**CISC**	**RISC**
utasítás bonyolultsága	**bonyolult műveletsor, több gépi ciklus**	**egyszerű műveletsor, egy gépi ciklus**
utasítások száma	**sok, akár 100-300**	**kevés, kevesebb mint 128**
címzési mód	**sok, akár 8-20**	**kevés, 2-4**
memóriautasítás	**sok**	**csak 2**
vezérlés	**mikroprogram, bonyolult, nagy tárolóigény**	**huzalozott**
utasításhossz	**változó, gyakran használt utasítások rövidek**	**rögzített**

Ellenőrző kérdések

1. Melyik a programozás legalacsonyabb szintje ?
2. Sorolja fel az utasításkészletek összehasonlításának ismérveit .
3. Melyik két utasításkészlet alakult ki a számítástechnika fejlődése során ?
4. Ismertesse a CISC processzorok utasításkészletének jellegzetességeit .
5. Ismertesse a RISC processzorok utasításkészletének jellegzetességeit .
6. Hasonlítsa össze a CISC és RISC utasításkészleteket .

9. MIKROPROCESSZOR TECHNOLÓGIÁK

9.1. BEVEZETÉS

Századunk második felének nagy vívmánya a félvezető, amely lehetővé tette nagyon sok tudományág gyökeres átalakulását, de egyúttal az elektronika, valamint az informatika robbanásszerű fejlődését.

A fejlődés ütemére jellemző, hogy az egységnyi felületen létrehozott áramkörökben található félvezetők számának növekedése az évek függvényében csak exponenciálisan fejezhető ki. Ma már léteznek olyan elemek, melyekben ez a szám meghaladja a 10 millót (példa rá a Pentium III-as processzor, ez az adat 2000-re vonatkozik).

A digitális technikában, illetve a mikroprocesszoros technikában kifejlesztett elemek száma rendkívül nagy. A segítségükkel felépített berendezések gyakorlatilag az élet minden területét uralják.

Időrendi sorrendben és bonyolultságuk szerint az egyse technológiák a következőképpen sorolhatók csoportokba:

- SSI (Small Scale Integration) – kis integráltsági fokú elemek, melyek 1964-ben jelentek meg. Gyakorlatilag egy chipen egy, esetleg néhány logikai kapu foglal helyet, jellemző, máig is fennmaradt és használatos képviselőjük a TTL kapuáramkör-sorozat.
- MSI (Medium Scale Integration) – közepes integráltsági fokú elemek, 1968-ban jelentek meg. Itt már megjelennek a különféle regiszterek, számlálók.
- LSI (Large Scale Integration) – nagy integráltsági fokú elemek, melyek a katonai és világűrkutatási programok célkitűzései alapján születtek meg. Megjelenésük 1971-re tehető. Ezen év első felében megjelenik az 1 kbit kapacitású dinamikus RAM memória, az év második fele pedig áttörést hoz a mikroprocesszorok gyártásában, kifejlesztik az első mikroprocesszort, mely 4 bites. A fejlesztés az Intel cég sikere, mely a mai napig megalapozza a cég vezető szerepét a miroszámítógép-gyártás területén. Valójában mai értelemben vett mikroprocesszornak nem nevezhetjük ezt a terméket, nemcsak azért mert főleg kalkulátorokban használták, hanem bebizonyosodott, hogy eredményesen a legalább 8 bites rendszerek használhatók mikroszámítógépekben. 1973 az az év, amikor megszületik az első valódi mikroprocesszor, az Intel 8080 A.
- VLSI (Very Large Scale Integration) – nagyon nagy fokú integrálás. Az év 1979, sorra jelennek meg a 16 és ennél nagyobb szóhosszúságú rendszerek, nagy kapacitású memóriák, bonyolult mikroszámítőgép elemek.

9.2. ALKALMAZOTT FÉLVEZETŐ TECHNOLÓGIÁK

A bennünket érdeklő mikroszámítógép-elemek feloszthatók több ismérv szerint, például:

- az alkalmazott félvezető technológia,
- szóhosszúság stb.

Maga a félvezető technológia alapjában két csoportba sorolható:

- a MOS technológia és

- a bipoláris technológia.

9.2.1. **MOS (Metal Oxide Semiconductor) technológia**

A MOS technológiában készített elemek gazdaságosabbak mint a bipoláris elemek, egyrészt olcsóbb a gyártásuk, másrészt fogyasztásuk is kisebb. Ezért az elemek nagy része ebben a technológiában készül. Hátrányuk a bipoláris technológiához képest a kisebb működési sebesség.

A MOS technológiában a töltéshordozók csatornabeli viselkedése alapján megkülönböztetünk:

- kiürítéses (depletion) és
- növekményes (enhancement) MOS-t.

Az adalékanyag, illetve a töltéshordozók típusától függően lehet:

- P csatornás eszköz (P channel MOS),
- N csatornás eszköz. (N channel) és
- komplementáris MOS (CMOS Complementary MOS).

9.2.1.1 **PMOS** technológia

A P-csatornás MOS technológiájú tranzisztor nagy küszöbfeszültségű, lassú eszköz, melynek táplálása negativ feszültségről történik. Mivel gyártása egyszerű, jó kihozatalt lehet vele élérni alacsony gyártási költségek mellett. 3-tól 15000 tranzisztor helyezhető el ezzel a gyártási technológiával egy lapkán. Ez ma már egy elavult technológia.

9.2.1.2. **NMOS** technológia

Az NMOS technológiában gyártott elemek kedvezőbb paramétereket mutatnak mint az NMOS, kisebb a nyitófeszültség, ami pozitiv, így TTL kompatibilis eszközök gyárthatók, rövidebb csatorna kell az elemnél, nő az elemsűrűség emiatt, valamint nagyobb lesz a működési sebesség. Bonyolultabb a gyártási technológia, kisebb a kihozatal, így nagyobbak a gyártási költségek.

9.2.1.3. CMOS technológia

Ez a technológia egyesíti magában a PMOS és NMOS elveket, amiból mint alapelem invertor készül. Mivel függetlenül logikai állapotától egyik tranzisztor nyitva van, a másik zárva, kis áram folyik az eszközön, így fogyasztása rendkívül kicsi. A kis fogyasztás mellett kitűnő zajérzéketlenséggel rendelkezik az elem, tápfeszültsége pedig igen tág határok között mozog (3 – tól 18 V-ig). Igen elterjedt a katonai, űrkutatási alkalmazásuk.

9.2.1.4. MOS integrált áramkör

A MOS IC-ben az aktív elem a MOS tranzisztor, míg az ellenállásokat állandó feszültséggel lezárt MOS tranzisztorokból lehet kialakítani. A MOS IC sem kondenzátorokat, sem diódákat nem tartalmaz.

Mivel a MOS feszültségvezérelt elem, a növekményes típusa közvetlen csatolása egyszerű, így egyszerű áramkört kapunk. Bemenő ellenállás 10^{14} **Ω.**

9.2.2. Bipoláris félvezető technológia

A bipoláris LSI félvezető eszközök ugyan gyorsabbak mint a MOS eszközök, de nagyobb teljesítményt igényelnek és előállítási költségük is magasabb a MOS elemeknél.

Legjelentősebb bipoláris technológiák a következők:

- **Schottky TTL (Tarnsistor Transistor Logic),**
- **LPS TTL (kisfogyasztású Schottky TTL),**
- **ECL (Emitter Coupled Logic)**
- I^2L **(Integrated Injection Logic).**

9.2.2.1. Schottky TTL (Tarnsistor Transistor Logic) technológia

Félvezető felületen létrehozott fémelektróda és a félvezető elektromos tulajdonsága, azok anyagától függően lehet kontaktus vagy egyenirányötó jellegű.

Olyan Schottky dióda alakötható ki, amelyben műkídést lassötó tíltéstárolási effektus nem lép fel. Egyszerűen előállöható, kis felületű a dióda, ezért a réteg kapacitása kicsi, nincs tíltéstárolás, tehát gyors. A kis nyitóirányú feszültség miatt a diódán disszipált teljesötmény kicsi.

9.2.2.2. LPS TTL (kisfogyasztású Schottky TTL) technológia

Ugyanaz mint előző, csak a fogyasztását tovább csökkentették.

9.2.2.3. ECL (Emitter Coupled Logic) technológia

Rendkövül nagy sebesség, 1-2 ns érhető el vele. Mivel nem telötésben, hanem az aktöv tartományban műkídik, nagy a fogyasztása. Különösen a bit-szeletelt processzoroknál alkalmazzák.

9.2.2.4. I^2L **(Integrated Injection Logic) technológia**

Ez egy módosötott bipoláris technológia, ahol a hagyományos NPN és laterális PNP-tranzisztor feladatot funkcionálisan egyetlen szigetben valósötják meg. Így egy egyszerű komplementer tranzisztorpár alakul ki.

9.3. KÜLÖNBÖZŐ TECHNOLÓGIÁK ÖSSZEHASONLÍTÁSA

A 9.1. táblázatban a különböző technológiák összehasonlítására kerül sor, ahol fontos paraméterek a sebesség, elemsűrűség, a fogyasztás, valamint a gyártási folyamat bonyolultsága.

9.1. táblázat: Az egyes technológiák összehasonlítása							
	PMOS	NMOS	CMOS	CMOS/SOS	TTL	I^2L	ECL
sebesség 1– legnagyobb	6	5	3	2	2	3	1
sűrűség 1- legnagyobb	2	1	3	2	4	2	4
fogyasztás 1–legkisebb	3	2	1	1	4	1	5
tapasztalat 1- legtöbb	2	3	4	5	1	6	3
bonyolultság 1-legkisebb	1	2	3	4	3	3	3

A táblázatban található CMOS/SOS jelölésnél az SOS rövidítés a Silicon on Sapphire szóból ered.

A 9.2. táblázat az egyes technológiákban előállított kapuk kimeneti terhelhetőségét (Fan-out, vagyis egy kimenetre hány bemenet kapcsolható), a kapunkénti fogyasztást, a kapcsolási időt és a zavarmentességet mutatja

9.2. táblázat: A különféle technológiák paraméterei				
technológia	kimeneti terhelhetőség FAN-OUT	fogyasztás mW/kapu	késés ns/kapu	zavarérzéketlenség V
TTL	10	10	10	0.4
Schottky TTL	10	22	3	0.4
LPS TTL	20	2	10	0.4
ECL	25	25	2	0.2
CMOS	50	0.1	25	3

Ellenőrző kérdések

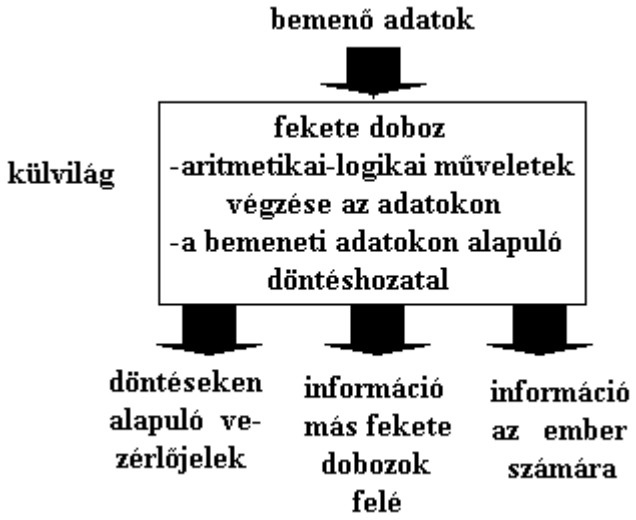
1. Sorolja fel a különbözö integráltságú technológiák elnevezését és megjelenésük időpontját.
2. Milyen alapelvek szerint csoportos1thatók a mikroprocesszoroknál alkalmazot félvezető elemek ?
3. A félvezétőket mely csoportokba sorolhatjuk ?
4. Ismertesse a MOS technológiákat, hány csoportba sorolhatók ?
5. Ismertesse a PMOS technológiát.
6. Ismertesse az NMOS technológiát.
7. Ismertesse a CMOS technológiát.
8. Ismertesse a MOS integrált áramkört.
9. Ismertesse a bipoláris technológiát.
10. Sorolja fel a legjelebtősebb bipoláris technológiákat.
11. Ismertesse a Schottky TTL technológiát.
12. Ismertesse a LPS TTL technológiát.
13. Ismertesse az ECL technológiát.
14. Ismertesse a I^2L technológiát.
15. Hasonlötsa össze a különféle technológiákat jellegzetes paramétereik szerint (sebesség, sűrűség, fogyasztás, gyártási tapasztalat és bonyolultság szerint).
16. Hasonlötsa össze a különféle technológiákat jellegzetes paramétereik szerint (kimeneti terhelhetőség, kapunkénti fogyasztás, kapunkénti késleltetés és zavarérzéketlenség).

10. MIKROPROCESSZOR, MIKROSZÁMÍTÓGÉP, MIKROVEZÉRLŐ

10.1. BEVEZETÉS

A címben szereplő három fogalom hasonlóan hangzik, sokszor össze is keverik ezeket a fogalmakat, valójában azonban lényeges közöttük a különbség.

A **mikro** (μ) görög betű jelentése kicsi, apró, ami itt az elemek méretére utal és nem a teljesítményére. Először nézzük meg mi is egy mikroszámítógépes rendszer. A 10.1. ábra fekete doboza a mikroszámítógépes rendszer, ami egy 'intelligens' ('okos') berendezés a feldolgozandó, 'nyers' adatok és a kimenetekre kerülő adatok, vezérlések



között. Mivel a fekete doboz az elektronikus digitális elemek mellett még mechanikus, pneumatikus, analóg elemeket is tartalmazhat, szükséges megvilágítani a tartalmát.

10.1. ábra: A bemenetek és kimenetek között levõ fekete doboz, a mikroszámítógépes rendszer

A fenti fekete doboz feltétlenül kell hogy tartalmazzon:

- egy aritmetikai és logikai számításokat elvégző egységet, valamint
- egy vezérlõegységet, amelyik döntéseket képes hozni.

Ez a két egység teszi ezt a berendezést (a mikroszámítógépes rendszert) 'intelligenssé'. A gyorsabb és könnyebb feladatvégzés érdekében a fekete doboznak tartalmaznia kell adattárolásra és utasítástárolásra memóriát (tárat). Az adatok felosztásával a 3. fejezetben találkozhattunk.

Ha az adatok és utasítások számára ugyanazt a tárat (memóriát) használja a gép akkor **Neumann-féle** számítógéprõl beszélünk, ha viszont a program tárolására is és az adatok tárolására is elkülönített tárakat használ a gép, akkor **Harvard-típusú** számítógéprõl van szó.

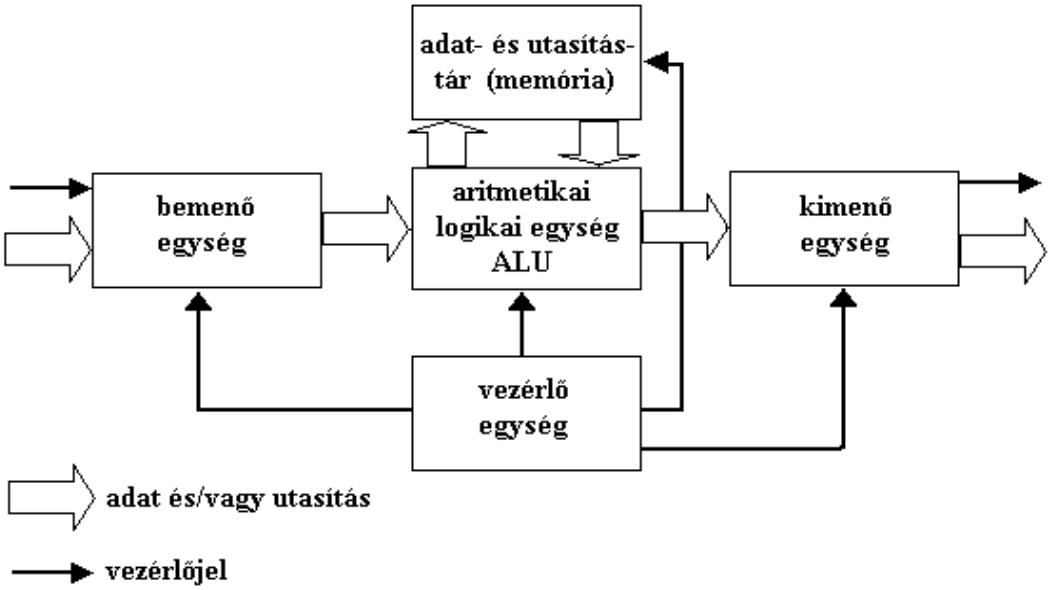
Mind az adatok beviteléhez, mind az adatok kiírásához fizikai hordozókra, egységekre van szükség. A 10.1. táblázatban felsoroltunk néhány ismert bevitelt biztosító egységet, a 10.2. táblázatban pedig a kimenõ adatok fizikai megjelenítését szolgáló eszközt.

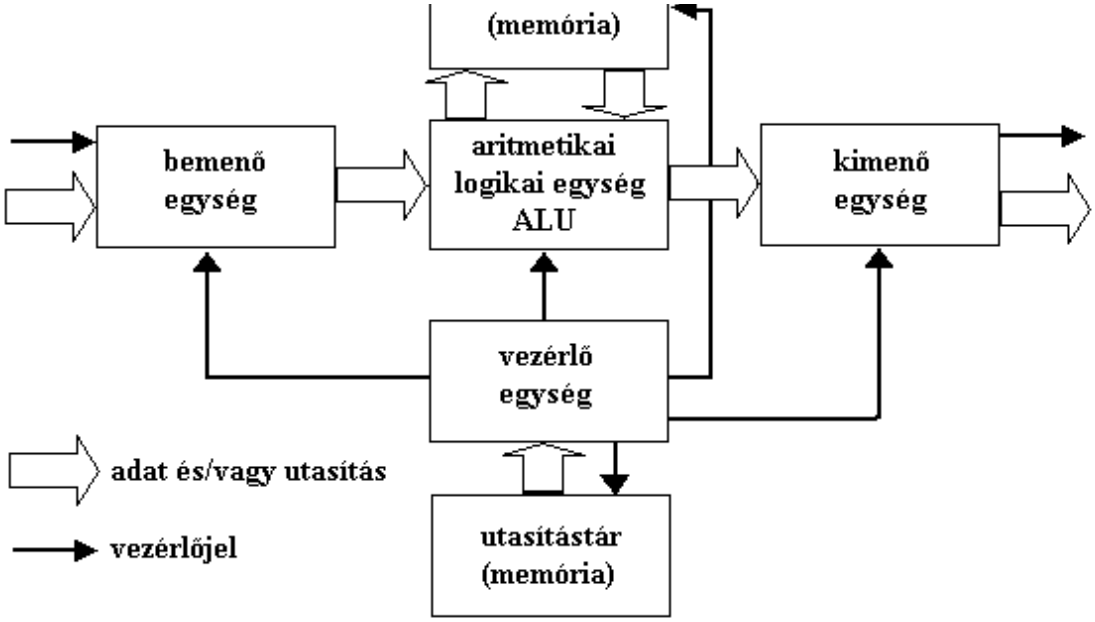
Midkét táblázatban látható, hogy három tipusú külvilági kapcsolatfajta létezik, egyrészt az ember, másrészt a másik számítógépek, de nem utolsó sorban a környezet felé. A felsorolás nem teljes, sok egységet, mely sokáig igen fontos volt a számítástechnikában túlhaladott az idõ, a számítástechnika fejlõdése pedig újabbnál újabb egységek alkalmazását hozza magával.

10.1. táblázat: Bemeneti adthordozók (információhordozók)	
A bemeneti adat	**Bemeneti adthordozó**
az embertől	Kapcsolók Klaviatúra (billentyűzet) Teleprinter Fényceruza Egér
más számítógépektől	Hajlékony lemez (floppy) CD ROM Modem soros átvitel (RS 232, RS 485) párhuzamos átvitel (CENTRONIX)
a környezetből	Érintkezők Analóg-digitális átalakítók

10.2. táblázat: Kimeneti adthordozók (információhordozók)	
A kimeneti adat	**Kimeneti adthordozó**
az ember felé	Kijelző tábla Numerikus kijelzõegység Nyomtató Képernyõ
más számítógépek felé	Hajlékony lemez (floppy) Modem soros átvitel (RS 232, RS 485) párhuzamos átvitel (CENTRONIX)
a környezetbe	Digitális-analóg átalakító Jelfogó Léptetõ motor

A 10.2. ábrán a **Neumann-**, míg a 10.3. ábrán a **Harvard-**típusú számítógépek modelljei láthatók. Jól látszik közöttük a különbség, míg a **Neumann** gép adat- és programtárolásra ugyanazt a memóriát használja, addig a **Harvard** gép különbséget





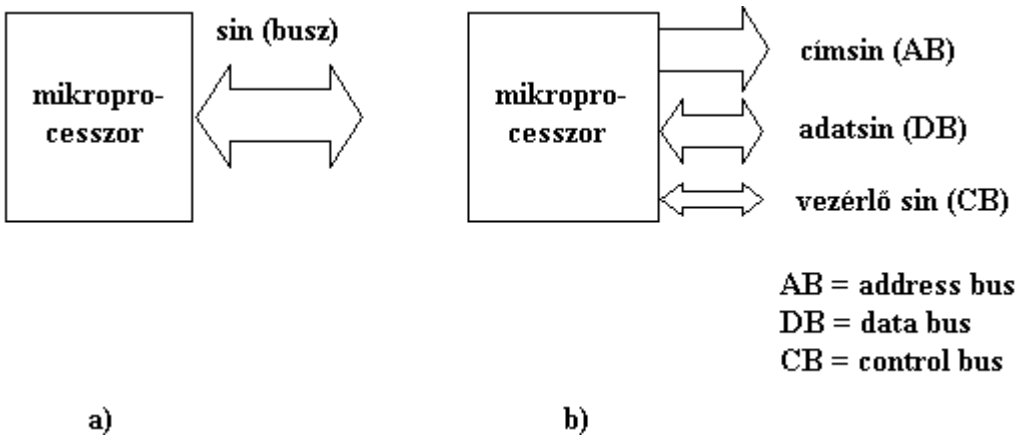
tesz az adat és utasítás között, így más tipusú tárat igényel ezek tárolására.

10.2. ábra: Neumann-féle számítógépmodell

10.3. ábra: Harvard-féle számítógépmodell

10.2. MIKROPROCESSZOR

A mikroszámítógépes rendszer megértéséhez induljunk el a 10.4. ábrából, ahol a



mikroprocesszor sematikus jelölése látható.

10.4. ábra: A mikroprocesszor sematikus ábrázolása (**a** - egyszerű, **b** - részletes)

Az ábra **a**) részén csak egy a külvilággal kapcsolatot tartó sinrendszer (angolul **bus,** innen ered a magyarul is használatos **busz** elnevezés) látható, ami a **mikroprocesszor** és más elemek (memóriák, illesztõ körök) közötti kapcsolatot biztosítja. Ez nem más mint lábak, illetve vezetékek összessége. Az ábrán a **sin** kétirányú, ami kétirányú adatáramlást tesz lehetõvé, de nem azonos idõben, hanem a feladattól függõen egyszerre csak egyirányút.

A 10.4. ábra **b**) részén ez a **sin** már három részbõl áll, itt:

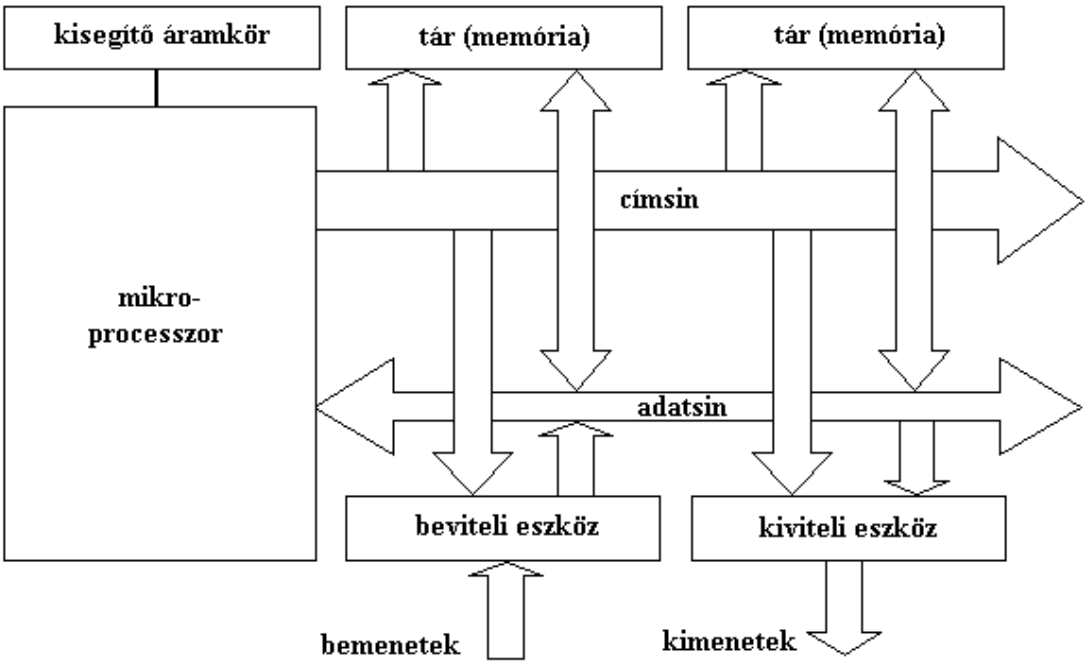
- **AB** jelöli az ún. **címsin**t, angolul **Address Bus**,
- **DB** jelöli az ún. **adatsint**, angolul **Data Bus,**
- **CB** jelöli a **vezérlõ sin**t, angolul **Controll Bus**.

Látható, hogy a **címsin** csak egyirányú adatforgalmat engedélyez, mégpedig a **mikroprocesszor**tól más elemek felé, az **adatsin**en már kétirányú adatáramlás lehetséges, de egyidõben csak egyik irányban.. A **vezérlő sin** kétféle irányban dolgozhat, de egy vezeték mindig csak egyik irányban, mig a másik tipus a másik irányban.

Mi a mikroprocesszor? **A mikroprocesszor olyan integrált áramkör (IC), amely egy számítógép központi egységének megfelelő funkciókat lát el, vagyis a memóriából (tárból) kapott adatokon a programnak megfelelően logikai és/vagy számítási műveleteket végez, valamint az eredmények alapján kiválasztja a következő lépést**.

10.3. MIKROSZÁMÍTÓGÉP

A 10.5. ábra egy mikroszámítógép sematikus rajzát tartalmazza, ahol is a központi egység, a **mikroprocesszor** köré néhány elem került, mégpedig a 10.2. pontban



ismertetett **cím-, adat-** és **vezérlő sin** felhasználásával.

10.5. ábra: A **mikroszámítógép** sematikus ábrázolása

A mikroszámítógép nyitott struktúrát képez, így kiépítése, bővítése a tervezõmérnöktől függ. Mindig a **sin** köré épül fel egy gép. Korlátot a **gép** kiépítettségének a rendelkezésre álló címezhetõ terület szab.

Az ábrán két memória található, amelyek logikailag egy egységet képeznek, tehát a szoftver, a program egy egységnek 'látja' a két memóriát.

Az ábrán egy bemeneti-, valamint egy kimeneti illesztőegység van, a kiviteli és beviteli eszközök számát szintén a rendelkezésre álló címzési terület szabja meg. Minden elemnek, egységnek külön, saját címe van a **mikroszámítógépen** belül, az esetleg egy címhez rendelt két vagy több egység adatütközést okoz, ami lehetetlenné teszi a rendszer működését.

A címet mindig a **processzor** állítja elő, ami aktív elem, ellentétben a rendszer többi elemével, ami mind passzív elem (egy kivétel van, erre visszatérünk).

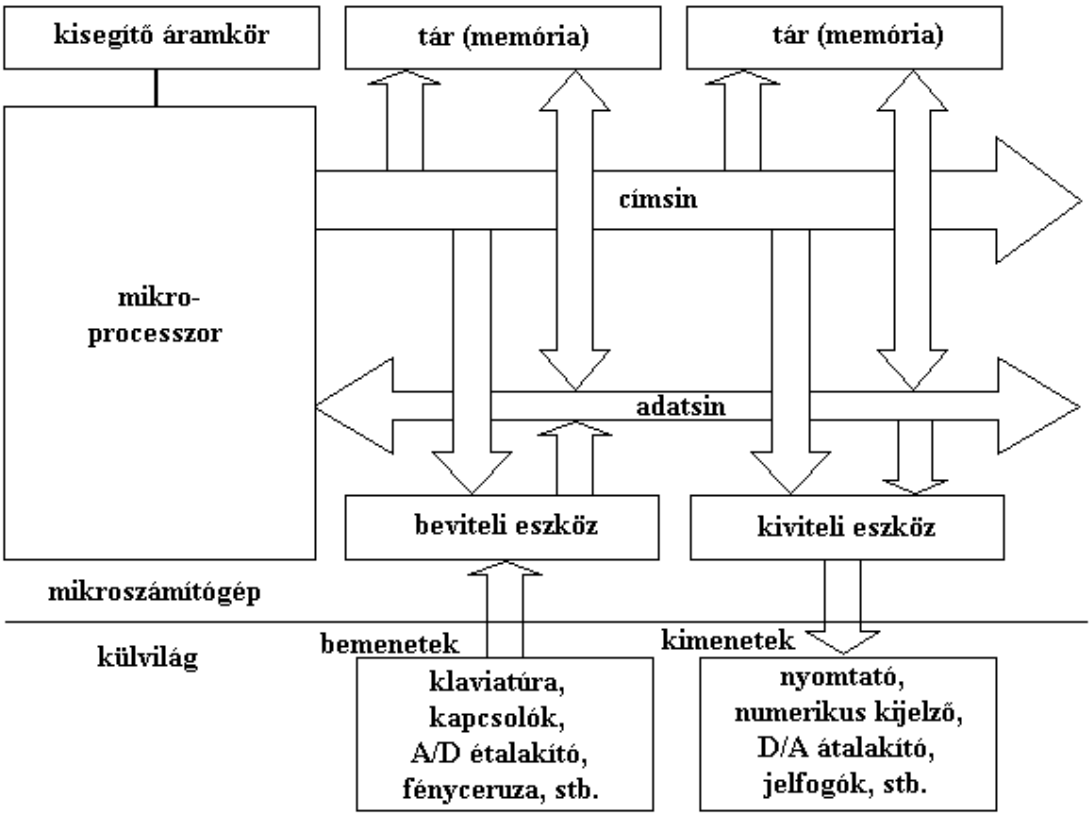
Az egységek az adatközléskor a kétirányú **adatsint** használják információcserére, ami egyes esetekben egyirányú is lehet, mint pl. az ábrán a **bemenõ illesztő áramkör** és **adatsin** között csak a **mikroprocesszor** felé történik adatáramlás.

10.4. A MIKROSZÁMÍTÓGÉPES RENDSZER

Egy **mikroszámítógép** feladata az, hogy a külvilágból vett adatokat a beírt program segítségével átalakítsa a felhasználó által megkívánt alakú adatokká és vezérlőjelekké.

A **mikroszámítógép**et megfelelő elemekkel kegészítve kapjuk meg a **mikroszámítógépes rendszert**. A külvilágból származó adatok, jelek különböznek a gépen belüli jelektől, pl. analóg jel a hőmérséklet. Diszkrét bináris jel esetében is lehet eltérés a jelek között, jelszintben (feszültség) is. Ezért szükséges a jeleket illeszteni.

A 10.6. ábrán látható egy **mikroszámítógépes rendszer** sematikus rajza.



Mind bemenő- mind kimenő áramkörök széles skálájával találkozhatunk egy **mikroszámítógépes** rendszernél.

10.6. ábra: A **mikroszámítógépes rendszer** sematikus ábrázolása

10.5. A MIKROSZÁMÍTÓGÉP ÉS A MIKROVEZÉRLŐ ÖSSZEHASONLÍTÁSA

A 10.2. fejezetben definiált **mikroprocesszor** általában egy integrált áramköri tokban (**IC**) helyezkedik el, ehhez hozzáépítve különböző elemeket a 6.3. fejezet szerinti mikroszámítógép építhetől fel. Általában ezek az elemek is, mint a **memória, kimenő- és bemenő illesztőegységek** szintén egy-egy **IC**-ben helyezkednek el. Az elemek közötti kapcsolat a **sinrendszeren** keresztül valósul meg. Ez a **sinrendszer**, mindamellett hogy biztosítja a rendszeren belüli adatáramlást, illetve információáramlást, a további bővítéshez is lehetőséget biztosít. Ezt a bővítést egyrészt a felhasználó igénye, másrészt az elemek (**mikroprocesszor, memória, ki- bemenő egységek**) műszaki korlátai szabják meg. Természetesen egy-egy feladat megoldásakor más és más igények merülnek fel sebesség, kapacitás szempontjából is, ilyenkor ezt is figyelembe kell venni.

A műszaki haladás magával hozta a technológia fejlődését, egy egy **IC** több és több elemet tartalmazhat. Így a mikroprocesszor mellé ugyanarra a lapkára már más elemek is felkerülnek. Ekkor is érvényesek azonban a fentiekben ismertetett architektúrák, valamint működési elvek.

A viszonylag egyszerű, tehát néhány benenetet, kimenetet és kis kapacitású memóriát tartalmazó rendszer kiépítése egyedi elemekből bonyolult hardvert igényel a hagyományos **mikroprocesszor, memória, ki-,** és **bemenő** egységek felhasználásával. Ezért hozták létre az ún. **mikrovezérlőket**, amelyek egy **IC** tokon belül még memóriát és ki- bemenetet tartalmaznak. Ekkor azonban a kevés kivezetés miatt már a külvilágba nem is vezetik ki a **sineket**, igy a rendszer nem is bővíthető. Ugyanakkor a belső felépítés, működés továbbra is megegyezik a **mikroszámítógépnél** leírtakkal. Van egy köztes megoldás is, amikor a **mikrovezérlő** alapkiépítésben (tehát általában egy tok) ellátja a teljes feladatkört, kis bövítéssel pedig létrehazható, ha szükséges a **sinrendszer**, amihez ismét a **mikroszámítógépnél** használatos módón hozzákapcsolhatók a rendszert bővítő elemek.

A jegyzetben egy zárt, külső sin létrehozását nem támogató mikrovezérlő található, valamint egy másik tipusú, külső sin létrehozására alkalmas **mikrovezérlő** ismertetése is megtalálható.

Ellenőrző kérdések

1. Rajzolja fel a mikroszámítógép-rendszert, mint fekete dobozt .
2. Milyen feladatok elvégzésére szolgál a számítógép ?
3. Milyen egységeket kell feltétlenül tartalmaznia egy számítógép rendszernek ?
4. Mi az aritmetikai és logikai egység (ALU) feladata ?
5. Mi a vezérlő egység feladata ?
6. Hol tárolja a számítógép a programot ?
7. Hol tárolja a számítógép az adatokat ?
8. Ismertesse a Neumann-féle számítógépmodellt és adja meg a gép blokkvázlatát .
9. Ismertesse a Harvad-féle számítógépmodellt és adja meg a gép blokkvázlatát .
10. Hasonlítsa össze a Neumann- és Harvard tipusú mikrogépeket .
11. Soroljon fel néhány adatbeviteli eszközt .
12. Soroljon fel néhány adatkiviteli eszközt .
13. Mely eszközök szolgálnak az ember-gép kapcsolat létrehozására ?
14. Mely eszközök szolgálnak a számítógép-számítógép kapcsolat létrehozására ?
15. Mely eszközök szolgálnak a számítógép-folyamat kapcsolat létrehozására ?
16. Mi a mikroprocesszor meghatározása ?
17. Mi a mikroprocesszor szerepe a számítógépben ?
18. Mi a sinrendszer, mi a szerepe és fizikailag mely része a processzornak ?
19. Mi a címsin és mi a szerepe ?
20. Mi az adatsin és mi a szerepe ?
21. Mi a vezérsin és mi a szerepe ?
22. Rajzolja le a mikroszámítógép blokkvázlatát .
23. Sorolja fel a számítógép aktív és passzív elemeit, részeit .
24. Rajzolja le a mikroszámítógép-rendszer blokkvázlatát .
25. Milyen feladatot látnak el az illesztőelemek a számítógép-rendszerben ?
26. Miért szükséges a külvilágból jövő és a külvilágba küldött jeleket illeszteni ?
27. Mi a mikroszámítógép és mi a mikrovezérlő ?
28. Miért nevezzük nyitott hardvernek a mikroszámítógépet és miért nevezzük zárt hardvernek a mikrovezérlőt ?
29. Van-e működésbeli különbség a mikroszámítógép és a mikrovezérlő között, ha van ismertesse azt .
30. Miért előnyös egy integrált áramköri tokban létrehozni a mikrovezérlőt ?
31. Mi a hátránya annak, hogy a mikrovezérlő egy tokban tartalmazza a processzort, memóriát és ki- bemenő elemeket ?

32. Lehetséges-e mikrovezérlőnél létrehozni külső sinrendszert, ha igen hogyan, mi az értelme ?

11. A MIKROSZÁMÍTÓGÉP MŰKÖDÉSÉNEK ISMERTETÉSE

11.1. BEVEZETÉS

A **Neumann** és a **Harvard** tipusú számítógép is ugyanazon az elven működik, vagyis a bemenő adatokon a gép memórájában levő program műveleteket végez, létrehozva részeredményeket, kimenő adatokat és vezérlőjeleket. Az utasítás végrehajtása a következő lépésekből áll:

1. a program kezdőcímének beállítása,
2. az itt található utasítás végrehajtása,
3. ugrás a következő címre,
4. visszalépés a 2. pontra.

A 3. pontban szereplő címmeghatározás lehet feltétel nélküli, ekkor gyakorlatilag a következő címről van szó, vagy valamilyen feltételhez kötött, ami a programban egy ugrást jelent.

A **mikroszámítógép** programozás szempontjából legfontosabb elemei az aritmetikai és logikai műveletek elvégzésére szolgáló **aritmetikai-logikai egység**, az **akkumulátor** (**A**), az utasításokat értelmező és végrehajtó **vezérlőegység**, a programot tartalmazó **programmemória**, valamint az adatokat tartalmazó **adatmemória.**

11.2. A MIKROPROCESSZOR MODELL

Ez az egység a mikroszámítógép, illetve a mikrovezérlő alapegysége. Feladata a programtárból beolvasott utasítások dekódolása, amikből vezérlőjeleket hoz létre a mikrogép egységes működtetéséhez.

Tekintettel arra, hogy gyakorlatilag a ma használatos processzorok legkisebb szóhosszúsága 8 bit, illetve a nagyobb szóhosszúságú gépek is valamilyen módon kötődnek a 8 bithez (például a 32 bites 4 x 8 bites szóhossz), az itt bemutatott modell is 8 bites. Ez a struktúra bővíthető bármilyen módon, így megérthető ennek a 8 bites modellnek a segítségével bármely más gép működése is.

11.2.1 **Az utasítások**

Minden számítógógép más és más utasításokkal oldja meg az elemi feladatokat. Az utasításkészlet kialakításánál a tervezéskor a következő tényezőket kell figyelembe venni:
a programtárolás gazdaságossága,

- az utasítás átmásolása az operatív memóriából a központi egységbe,
- az utasításon belül a műveleti kódra fenntartott mező nagysága akkora kell hogy legyen, hogy a kívánt számú utasítás kódolható legyen és
- a címezhető memória nagysága.

Egy feladat lépéseit utasításokkal adjuk meg, vagyis a **mikroszámítógépen** belül létrehozható elemi lépések egymásután való elhelyezésével programot írunk, beírjuk a

memóriába. A bináris alakban tárolt programot különböző segédprogramokkal hozhatjuk létere. Egy utasítás általános alakja látható a 11.1. ábrán.

műveleti kód	**címzési mód**	**cím / adat**

11.1. ábra: Az utasítás általános alakja

A **műveleti kód,** mint az utasítás többi része is, kettes számrendszerben megadott információ, ami meghatározza a végrehajtandó feladatot. Az utasítás harmadik mezõjében levõ **cím/adat** információ határozza meg a műveletben szereplõ adatot,

a)

műveleti kód

b)

műveleti kód	**adat**

c)

műveleti kód	**cím**

vagy annak helyét. A 11.2. ábra az utasítások különbözõ lehetõségeit sorolja fel. Egyenlőre hanyagoljuk el a 'címzési mód' nevű mezőt.
11.2. ábra: Az utasítás lehetséges formái

d)

műveleti kód	**az eredmény címe**	**az 1. operandus címe**	**a 2. operandus címe**

e)

műveleti kód	**az 1. operandus címe / az eredmény címe**	**a 2. operandus címe**

f)

műveleti kód	**az operandus címe**

Az **a** esetben processzoron belüli műveletrõl van szó, nincs szükség külsõ információra. A **b** esetben maga az utasítás tartalmazza a műveletben résztvevõ adattot, a **c** esetben pedig az utasítás az operandus helyét adja meg, cím alakjában. A 11..2. ábra utasításformái már egyszerűsített alakúak, ugyanis általános formában pl. a **c** pont alatti utasítás a 11.3. ábra szerint kellene hogy kinézzen.
11.3. ábra: Utasításformátum általános esetben, d) 3 címes, e) 2 címes és f) 1 címes utasítás

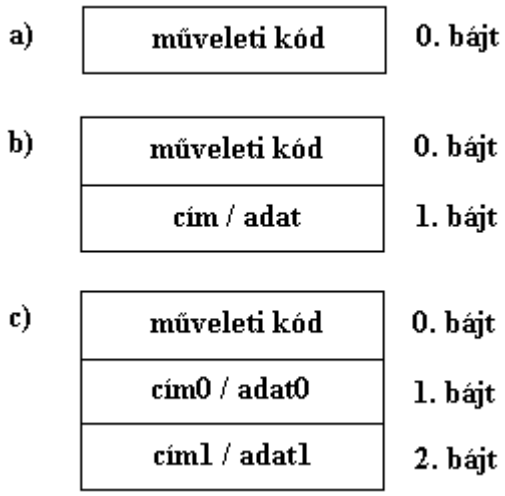
A **d** pont alatti utasítás 3 címes, valójában ezt ki kellene bővíteni általános esetben 4 címes utasításra, ahol a 4. mező a következő utasítás címét adná meg. Ezt most azért nem tettük, mert a **PC** programszámláló bevezetésével egy utasítás végrehajtása után a **PC** mutat a következő végrehajtandó utasításra.

A **d** alakú utasítás láthatóan igen hosszú, ezért használata nem célszerű. Ez egy 3 címes utasítás, feltételezzük hogy két adat között végzünk műveletet, mindkettőnek címe van, ugyanakkor az eredmény címe is szerepel az utasításban.

Csökkenthető az utasítás tárolására felhasznált bitek száma ha 2 címes utasítást használ a gép (11.3. ábra, e) rész), ilyenkor azonban a művelet előtti, az 1. operandus címe nevű rekeszbe kerül az új eredmény, egyuttal a régi 1. operandus felülíródik (vagyis elveszik).

A 11.3. ábra f) része egy olyan megoldás, ahol csak egy operandus címére van szükség. Hogyan végez akkor mégis el ez a típusú utasítás műveletet két érték között ? Ilyenkor az egyik operandus már megtalálható a mikroprocesszor valamelyik belső regiszterében, a másik operandus az adattárban, az eredmény pedig újból a processzor belső regiszterébe kerül, felülírva az ott található értéket. Ez a megoldás ugyanaz, mint a 11.2. ábra c) pontja. Látható, hogy az utasítások tárolására sokkal kevesebb bitre van szükség ilyen megoldás alkalmazásakor, ugyanakkor szükséges a processzor belső regiszterében előkészíteni az egyik operandust.

Műszaki, technológiai és megbízhatósági korlátok miatt egyszerűbb és olcsóbb kisebb szóhosszúságú processzorokat gyártani. Ma az alsó határ a 8 bit, illetve ennek általában egészszámú többszöröse. Ha a nyolc bites gépnél elemezzük azt, hogy hogyan férhet el 1 bájtban a 11.2. ábra szerint értelmezett utasítás, akkor a válasz az, hogy gyakorlatilag sehogy, ugyanis a 256 különböző állapot kevés utasítás létrehozását, illetve igen kis



címmező elérését teszi lehetővé. Ezért célszerű több bájt felhasználása egy utasítás tárolására. Itt is többféle megoldás lehetséges, ezek közül modellünknél a 11.4. ábra szerinti megoldást alkalmazzuk.

11.4. ábra: Az utasítások alakjai 8 bites mikrogépnél

Egy ilyen megoldásnál a 11.4. a) ábra utasítása processzoron belüli művelet, a b) rész egy nyolcbites közvetlen adattal végzett művelet, vagy egy nyolcbites címmel címezhető elem művelete. A 11.4. c) ábra 2 bájtos (16 bit) címet ad, ami 0 és $2^{16} - 1 = 65535$ címek között tesz lehetővé adatelérést (0000h – FFFFh). Ez ugyan nem túl nagy tárterület, de sokszor kisebb alkalmazásoknál még ennél is lényegesen

kevesebb adatra van szükség. Ha nem címről, hanem közvetlen adatról van szó, akkor értelemszerűen ez 0 és 65535 közötti érték.

Mivel műveleti kódra 1 bájt áll rendelkezésre, egy ilyen processzor értelemszerűen legfeljebb 256 különféle művelet végrehajtását teszi lehetővé.

11.2.2. **Egyszerű mikroprocesszor modell**

A 7.1.1 pontban tárgyalt utasítás meghatározza a modell (illetve a valódi mikroprocesszor) több fontos alkotóelemét, annak méretét. Az 1 bájtos szervezés nemcsak az operatív tár szóhosszúságát korlátozza 1 bájtra, de a regiszterek, belső sinek méretét is megadja, ami 1 bájt, illetve ennek többszöröse, például 2 bájt.
A 11.5. ábra mutatja az egyszerűsített modellt. Itt a legfontosabb, nélkülözhetetlen részek láthatók. Felépíthető más modell is, ettől eltérő struktúrával, de az is hasonlítani fog az itt tárgyalt modellhez.

Vizsgáljuk meg először a modell **A** jelzésű **akkumulátor**át. Látható, hogy a belső sin szóhosszúságával megegyező méretű, feladata pedig egy műveletben szereplő operandus ideiglenes tárolása. A mikroprocesszorok általában egy akkumulátort tartalmaznak, de vannak kettő, vagy több akkumulátort tartalmazó processzorok is. Szerepe főleg az aritmetikai és logikai műveleteknél van, de a perifériákkal való adatcserében is főszerepet játszik.

A **PC** jelzésű **programszámláló** (program counter) a következő ciklusban végrehajtásra kerülő utasítás címét tartalmazza. Itt vegyük figyelembe azt, hogy bizonyos mikroprocesszorok a különböző utasításokat 1, vagy több bájton tárolják, így a **PC** tulajdonképpen mivel bájtokat címez, van amikor műveleti kódra mutat, van amikor a cím, vagy adat egy-egy bájtjára. Látható, hogy szóhosszúsága kétszerese a belső sin, tehát a rendszer szóhosszúságának. Ennek oka az, hogy egy bájt mindössze 256 tárhely címzését teszi lehetővé. Logikus PC szóhosszúságának növelése a bájtok hozzáadásával, adatátvitelnél továbbra is bájtok átvitele történik. 2 bájtot használva a PC-hez 65536 tárhely címezhető az operatív memóriában, ami egy elfogadható program és adattárolást tesz lehetővé.

Az operatív tárban elhelyezkedő adatok címzését a **DC adatszámláló** (data counter) végzi, amely szintén 2 bájtos.

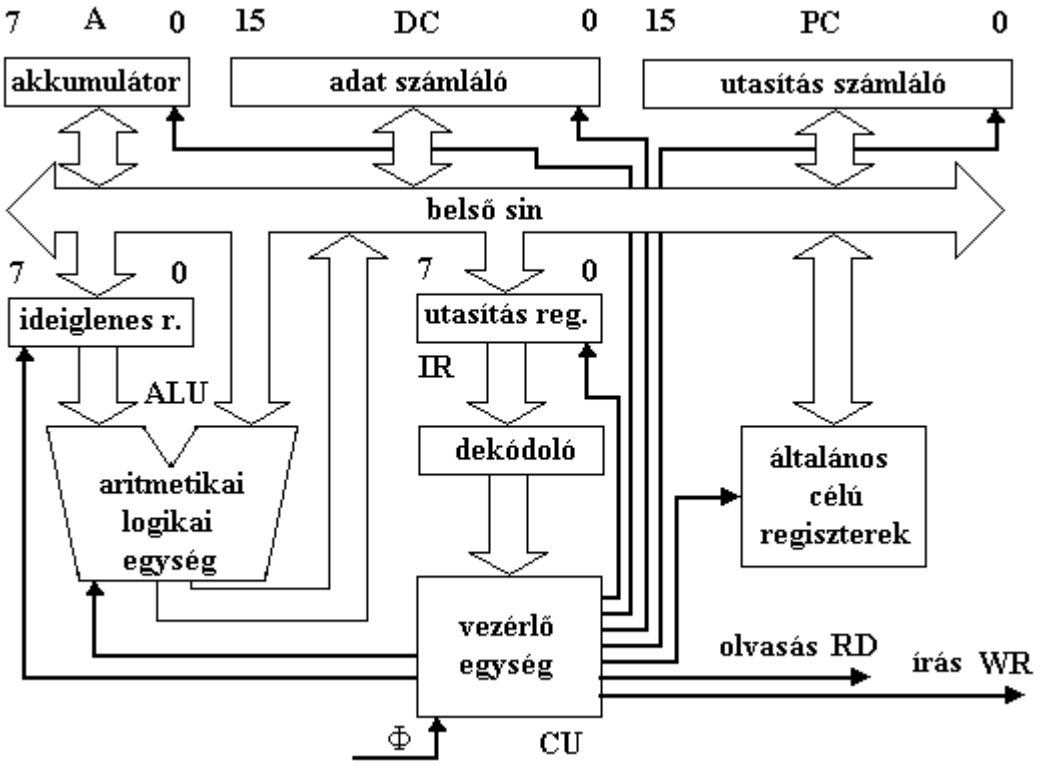
Az **IR utasítás regiszter** (instruction register) a műveleti kód tárolását végzi, az itt tárolt kód kerül egy **dekódoló áramkör** segítségével a **vezérlő egységbe**.

Az utasítás végrehajtása két fázisban történik:

- **előkészítés** (fetch) – másképpen utasítás kihozás és
- **végrehajtás** (execute).

Minden előkészítés-fázis úgy történik (ez érvényes a gép bekapcsolása után is), hogy a vezérlő egység a benne tárolt mikroprogram segítségével, szinkronban a Φ **külső órajel**lel önállóan végrehajt néhány lépést. Ennek a célja az, hogy megszerezze az aktuális utasítás gépi kódját, megtudja hogy mi a feladata.

A mikrogép bekapcsolásakor egy áramkör kezdő helyzetbe hoza a mikroprocesszor több egységét, regiszterét. Ilyen regiszter többek között a **PC programszámláló** is. Ennek kezdőértéke **0000h** lesz, ami egyuttal azt is jelenti, hogy az első utasítás az operatív tárban ezen a címen kell hogy elhelyezkedjen. A vezérlő egység a **PC** tartalmát a belső sinen keresztül kiküldi a processzor címsinjére (6.4. ábra). Az egyszerűsített modellen nem látszik minden részlet, mivel **PC** 16 bites, a **belső sin** pedig 8 bites, ezért értelemszerűen ez csak két részletben, bájtonként történhet meg. A parancs a művelet végrehajtására a vezérlő egység és a **PC** közötti úgynevezett **belső vezérlő vezeték**en keresztül kerül **PC**-be. A mikrogép mikroprocesszor és operatív memória között elhelyezkedő **dekódoló** áramköre felismeri a cím alapján a címzett memóriarekeszt, majd a vezérlőegység egy **külső vezérlő vezeték**en keresztül **RD olvasási parancs**ot küld a memória felé. Ennek eredményeként a memória a **külső adatsin**re (11.4. ábra) helyezi a **PC** által címzett memórirekeszben található adatot, ami műveleti kód. Ez a műveleti kód ezek után bekerül a mikroprocesszor belső sinjére,

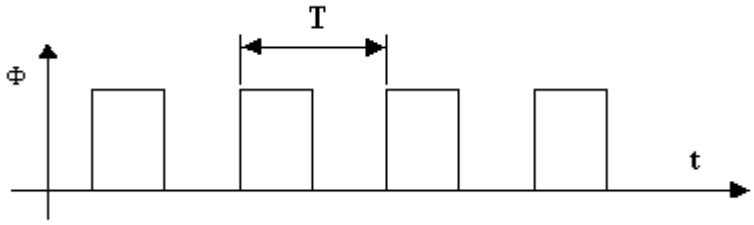


ahonnan a vezérlő egység egy következő lépésben egy belső vezérlő jel segítségével az IR utasítás regiszterbe irányítja. Ekkor befejeződött az előkészítés, a vezérlő egység ettől a pillanattól kezdve már nem önállóan dolgozik, hanem értelmezi a műveleti kódót és mikroutasítások sorozátával belső és külső vezérlőjeleket állít elő.Ezek a vezérlő jelek irányítják az adatáramlást a mikroprocesszoron belül, illetve a mikrogépen belül. Attól függően, hogy az utasítás milyen feladatot hajt végre a mikroprogram különbözik, például csak processzoron belüli utasítás gyorsabb mint az operatív memóriába való adatküldés, ahol szükséges a vézérlőegységnek az utasításban szereplő címet is bekérnie az adatszámlálóba.

11.5. ábra: Egyszerűsített mikroprocesszor modell

A mikroműveletek nem tetszőleges időben kezdődnek, illetve nem tetszőleges hosszan tartanak, szinkronizációs jel szabja meg pontos létrejöttüket, ezt egy óragenerátor állítja elő.

Gyakorlatikag kétféle, úgynevezett egyfázisú (11.6. ábra) és kétfázisú (11.7. ábra) órejelet használnak a mikroprocesszoroknál. Ennek az órajelnek a frekvenciája szabja meg a mikrogép működési sebességét. Meghatározásakor figyelembe kell venni az egyes elemek működési sebességét.

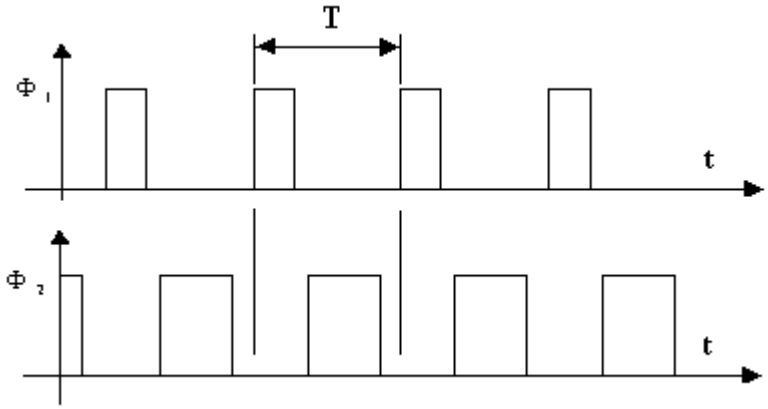


11.6. ábra: Egyfázisú órajel

11.7. ábra: Kétfázisú órajel

A modellnél egyfázisú órajelet használunk.

Ahhoz hogy megértsük a mikroprocesszor működését egy egyszerű program lépésenkénti végrehajtását elemezzük. Legyen az **operatív memóriában** a 11.8. ábra



szerinti program, aminek kezdőcíme **0100h**.

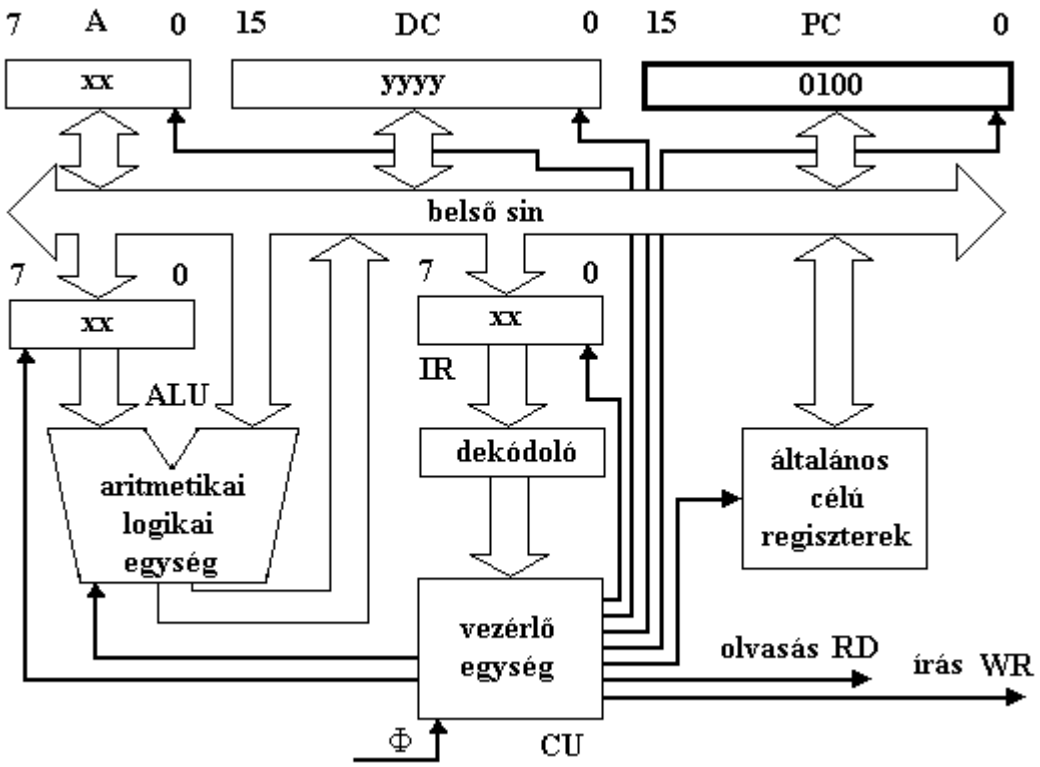
cím	tartalom	jelentés		
0100	B6	betölti a memória 0201	program	memória
0101	02	címéről az adatot az A		
0102	01	akkumulátorba		
0103	9B	összeadja az A akkumulátor		
0104	FF	tartalmát a 00FF címen levő adattal		
0105	97	betölti az A akkumulátor		
0106	01	tartalmát a 010F címre		
0107	0F			
0108				
	⋮			
00FF	1A	2. operandus	adat	
	⋮			
010F	xx	eredmény		
0201	23	1. operandus		

11.8. ábra: Mintaprogram

A 11.9. ábrán ismét a mikroprocesszor modellje látható, a vastaggal jelzett részek vesznek részt a műveletben.

Induláskor feltételezzük, hogy a **PC programszámláló** programunk kezdőcímét a 0100-át tartalmazza. A 8 és 16 bites regiszterek tartalmaznak valamilyen értéket, amely értékek közömbösek, ezért **xx** illetve **yyyy** értékkel jelöltük azokat.

A **CU vezérlő egység** elindít egy folyamatot, **mikroutasítások**at hajt végre annak érdekében, hogy az utasítás kihozási fázisban (előkészítés) az operatív memóriából az **utasításregiszter**be másolja át az első utasítás műveleti kódját. A kiolvasás után változatlanul ez az érték megmarad a memóriában.



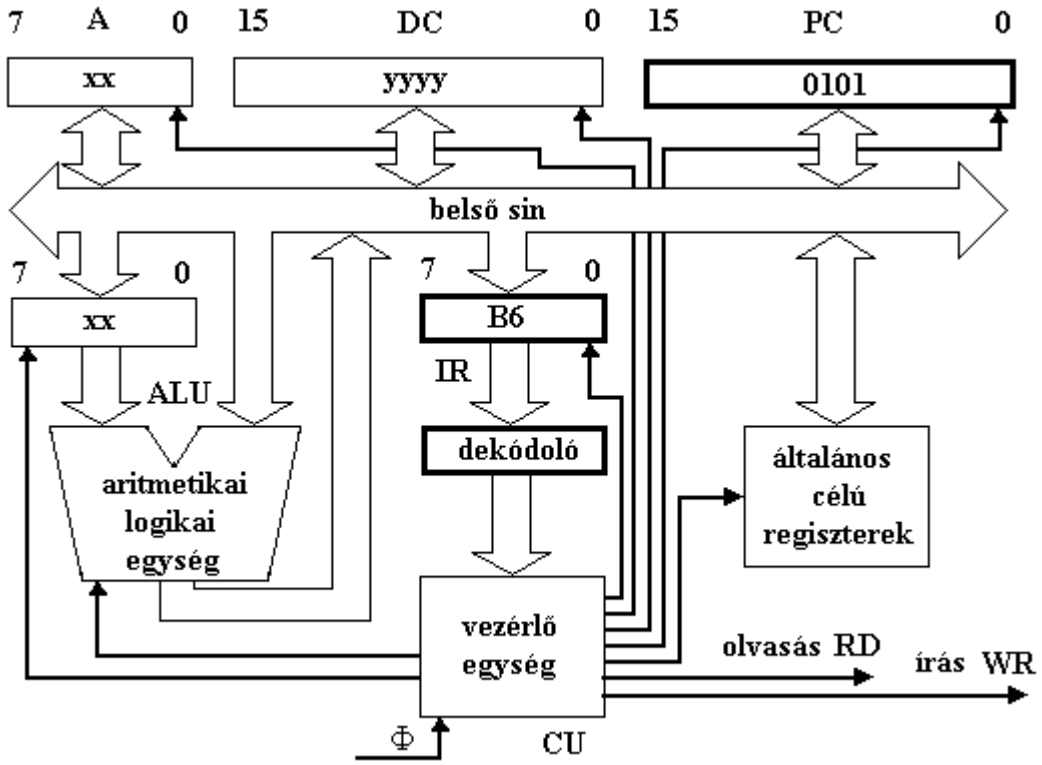
11.9. ábra: Kezdőállapot

A **PC** programszámláló 16 bites, így a **CU** vezérlő egység csak két lépésben képes a **PC** tartalmát kiküldeni a mikrogép **AB címsin**jére, először PC felső bájtját, majd utána **PC** alsó bájtját. Miután kikerült a mikroprocesszorból a memóriacím, azonnal autómatikus **PC** tartalomnövelést hajt végre a **CU** vezérlő egység, függetlenül attól, hogy mi a következő lépés.

PC tartalomnövelésre szükség van, hiszen egy utasítás lehet 1, 2 vagy 3 bájtos, a következő bájtot mindenféleképpen be kell olvasni.. A 0100 címen található utasítás 3 bájtos, a műveleti kód mellett még 16 bites címet is tartalmaz, ami az adat memóriabeli címét adja meg. A mikrogép **AB címsin**jén megjelenő cím a **címdekódoló** segítségével a kívánt elemet címezi, ez esetünkben a memória 0100 címen levő rekesze. A CU vezérlő egység egy külső vezérlő jelet, **RD olvasási parancs**ot küld a mikrogépbe. A cím és RD együttes hatására a memória az adott címről az adatot kiküldi a rendszer **DB adatsin**jére, amit a CU vezérlőegység a mikroprocesszor belső sinjére irányit, onnan pedig egy újabb mikroutasítással az **IR utasításregiszter**be enged, ahol azt tárolja.

Ez az új állapot, vagyis az előkészítés vége látható a 11.10. ábrán, tehát ekkor a műveleti kódot a CU vezérlőegység az IR utasításregiszterbe tárolta.

A B6 kód, ami binárisan 10110110, tartalmazza a processzor számára azokat az információkat, amelyek szükségesek a feladat végrehajtásához, vagyis hogy a memóriából egy bájtot át kell másolni az **A akkumulátor**ba. Ez a műveleti kód azt az információt is hordozza, hogy a memóriacím az utasítás után helyezkedik el, hossza két bájt. Ettől a pillanattól kezdve a CU vezérlőegység a műveleti kód szerint hozza létre a vezérlőjelek sorozatát a feladat végrehajtásához.



11.10. ábra: Az első utasítás előkészítési fázis utáni állapota

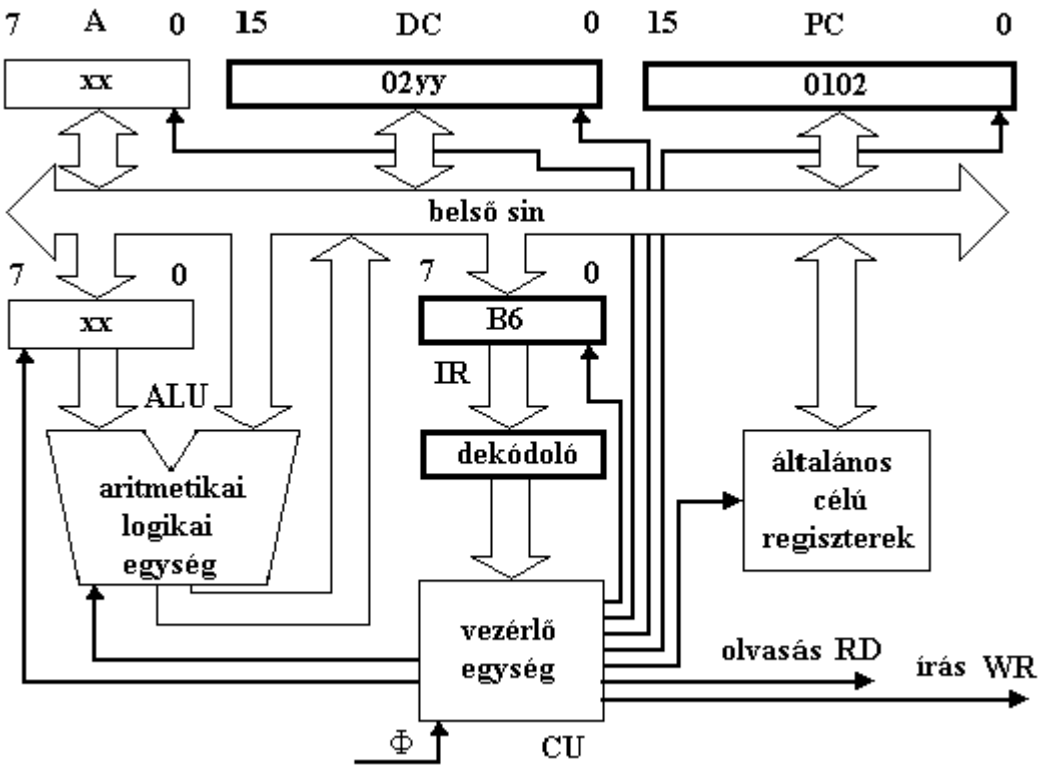
Valójában az eddigiekhez nagyon hasonló módon másolja át a memóriacímet a DC adatszámlálóba.

Az aktuális PC utasításregiszter tartalmat két fázisban kiküldi az AB adatsinre, a memóriadekódoló felismeri a címet, megcímzi a 0101 címen levő adatot, ami most az operandus memóriacímének felső bájtja, vagyis 02, majd RD olvasási parancsot ad.

A memória kihelyezi az adott címen levő adatot a DB adatsinre, amit a mikroprocesszor először a belső adatsinre vesz át, majd innen tárolja a CU vezérlőegység vezérlőjelének hatására a DC adatszámláló felső bájtjában. Ez az állapot, tehát a második ciklus utáni állapot látható a 11.11. ábrán.

Természetesen most is PC programszámláló értéknövelés történik, hiszen a következő bájt, ami az operandus memóriacímének az alsó helyértékű bájtja a memória következő rekeszében található (0102 cím).

A memóriacím alsó, kisebb helyértékű bájtját nagyon hasonló módon olvassa be a processzor, vagyis az aktuális PC utasításregiszter tartalmat két fázisban kiküldi az AB adatsinre, a memóriadekódoló felismeri a címet, megcímzi a 0102 címen levő adatot, ami most az operandus memóriacímének alsó bájtja, vagyis 01, majd RD olvasási parancsot ad.



11.11. ábra: Az első utasítás második ciklusa, az operandus címének magasabb helyértékű bájtjának beolvasása

A memória kihelyezi az adott címen levő adatot a DB adatsinre, amit a mikroprocesszor először a belső adatsinre vesz át, majd innen tárolja a CU vezérlőegység vezérlőjelének hatására a DC adatszámláló alsó bájtjában. Ez az állapot látható a 11.12. ábrán.

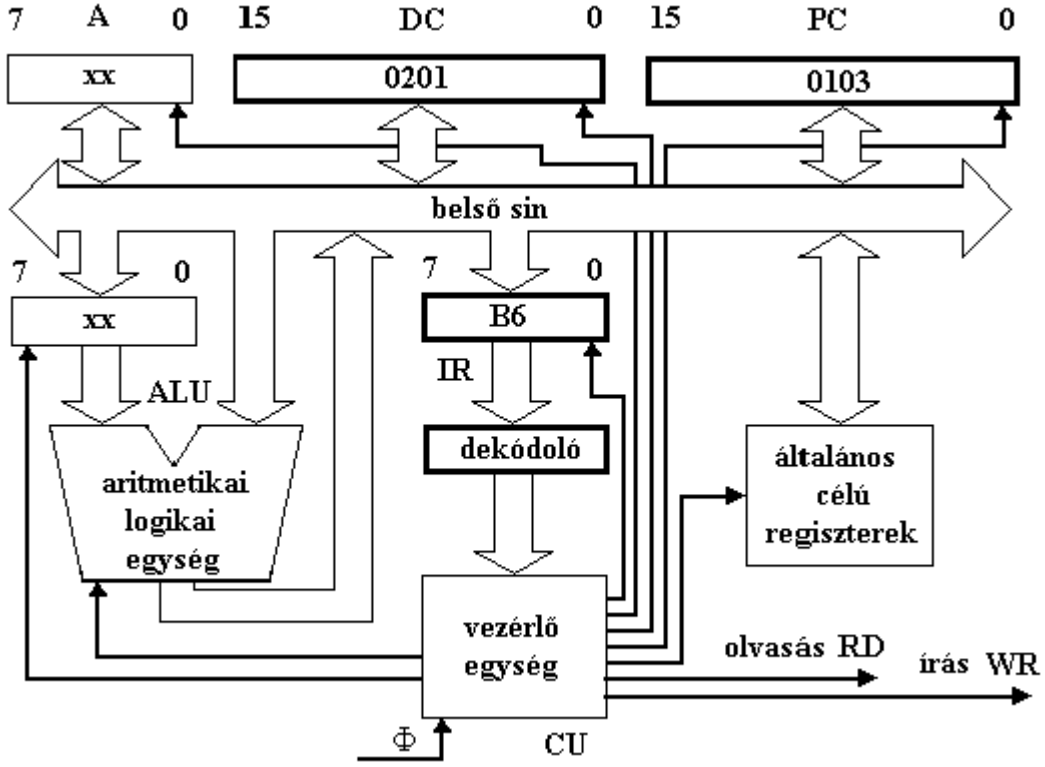
Természetesen most is PC programszámláló értéknövelés történik, hiszen a következő bájt, ami a következő utasítás, a memória következő rekeszében található (0103 cím). Ekkor már három memóriához fordulás volt (ez három ciklus), de így is csak még a műveleti kód mellett az operandus memóriabeli címe található meg a mikroprocesszorban, annak is DC adatszámlálójában.

A PC programszámláló már a következő utasítás műveleti kódjára mutat, de elötte még szükséges az operandus beolvasása a memóriából az A akkumulátorba. Ehhez a CU vezérlőegység a DC adatszámláló kétbájtos címét helyezi ki lépésenként a mikrogép AB címsinjére, hasonlóan az előző lépésekhez a mikroprocesszor és operatív memória közötti címdekódoló hozza létre a kapcsolatot az operatív memória megfelelő rekeszével, ahol az operandus található, ez a 0201 cím.

Ezek után ismét a már ismert lépéseket figyelhetjük meg, vagyis a 0201 cím kiküldése után a CU vezérlőegység külső vezérlőjelet küld, vagyis RD olvasási parancsot, a memória kihelyezi a 0201 címen található operandust, ami 23 a mikrogép DB

adatsinjére, ahonnan a belső sinre kerül az át. A belső sinről a CU vezérlőegység belső vezérlőjel segítségével az A akkumulátorba másoltatja az értéket.

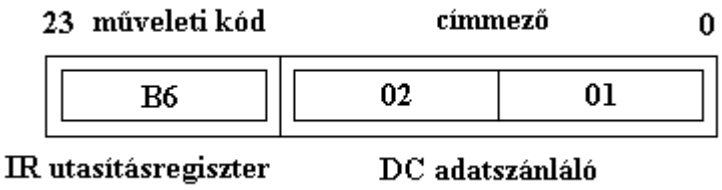
Ezzel az utolsó lépéssel be is fejeződött az első utasítás végrehajtása, PC utasításszámláló már az új utasításra mutat (0103 cím). Az új állapotot a 11.14. ábrán



láthatjuk.

11.12. ábra: Az első utasítás harmadik ciklusa, az operandus címének alacsonyabb helyértékű bájtjának beolvasása

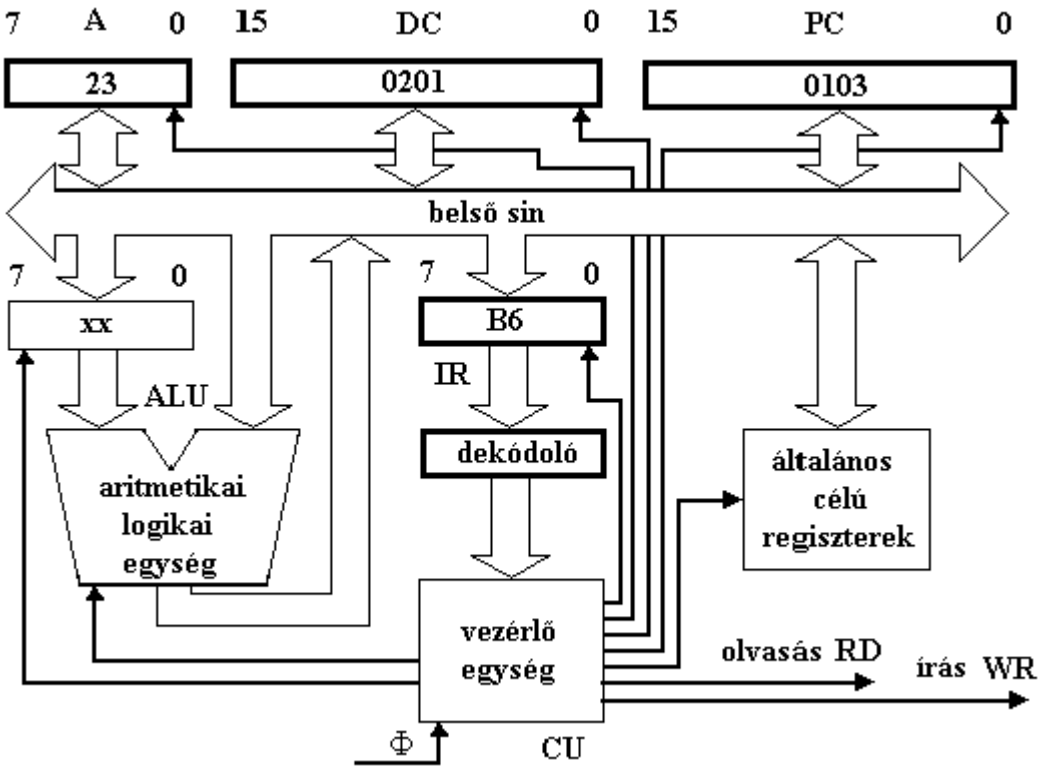
Ha most képzeletben összekötjük IR utasításregisztert, ami 8 bites DC adatszámlálóval,



ami viszont 16 bites, akkor egy 24 bites regisztert kapunk, aminek 23-16 bitjei a műveleti kódot adják, míg 15-0 bitek a címmezőt. Ez a képzeletbeli 24 bites regiszter látható a 11.13. ábrán.

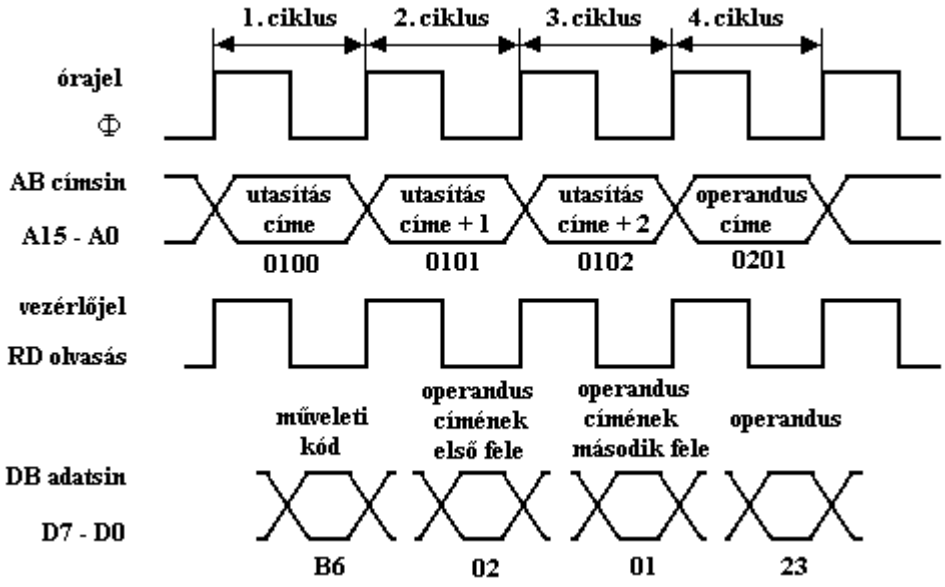
Ez csak képzeletbeli regiszter, a rendszerben ez nem létezik, hiszen a rendszer 8 bites, úgyhogy a mikroprocesszorba való utasításbeolvasás 3 ciklusideig tart, ha a rendszer 24 bites lenne, akkor egy ciklus elég lenne az utasítás beolvasásához. Ehhez még hozzá kell adni 1 ciklust, ami az operandus beolvasásához szükséges.

11.13. ábra: A három részes utasításszó



11.14. ábra: Az első utasítás negyedik ciklusa, az operandus beolvasása az A akkumulátorba

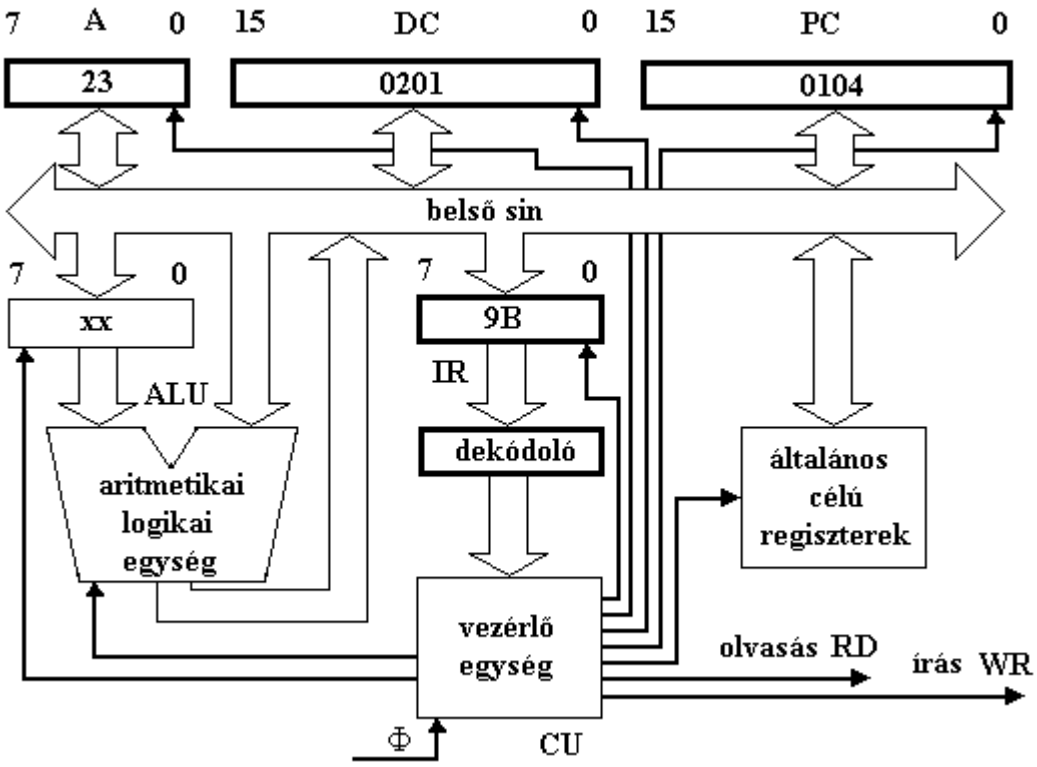
A fenti működési magyarázat végigkövethető egy egyszerűsített idődiagramon is. Ez az idődiagram látható a 11.15. ábrán. Az AB címsin példánkban 16 bites, nehézkes lenne minden egyes vezeték állapotát egyenként ábrázolni, ezért szimbolikusan jelöljük az AB címsinen történő változást két, címváltozásnál egymást metsző vonallal. Ugyanez a megjegyzés vonatkozik a DB adatsinen történő változásokra, ami modellünknél 8 bites. A DB adatsinen nem folytonos az egyes változások közötti átmenet, ez azért van, mert itt lehetséges a 0 és 1 értékek mellett egy harmadik, végtelen impedanciájú állapot is (ez szakadás). A modell az órajellel egyformán futó RD olvasási jelet használ. Az idődiagramon is látszik, hogy négy ciklus szükséges a mintapélda (11.8. ábra) első utasításának végrehajtásához. Az ábráról az is leolvasható, hogy melyik ciklusban mi történik, tehát például az első ciklusban az AB címsin a memória azon rekeszét címezi, ahol az utasítás műveleti kódja található, a második ciklusban az operandus tárcímének felső bájtja, és így tovább. Az is látszik, hogy az első három ciklusban az AB címsinen a program 3 egymásutáni bájtjának címe jelenik meg, míg a negyedik ciklusban a tár azon helyét címezi, ahol az operandus van.



11.15. ábra: Az első utasítás egyszerűsített idődiagramja

Az első utasítás befejezése után a PC programszámláló a következő utasítás műveleti kódját címezi a tárban, ami a 0103 címen található. Az utasítás végrehajtásakor a fentiekben leírt eljárás ismétlődik, ezt részletesen nem elemezzük, csak az érdekesebb részekre hívjuk fel a figyelmet. A második utasítás az elsőhöz hasonlóan műveleti kódrészt tartalmaz (első bájt), viszont van egy lényeges különbség, mindössze egy bájt hosszúságú címrész tartozik az utasításhoz. Mi ennek a magyarázata ? Tekintettel arra, hogy az AB címbusz 16 bites, ezért a modell csak 16 bites címek kezelését teszi lehetővé. Valójában a második utasítás FF címe 00FF, tehát csak a 0. lapon összesen 256 hely címzését teszi lehetővé (lapcímzés). Mintapéldánk utasítása a műveleti kódban tartalmazza ezt a tényt és FF betöltésekor a felső bájt értékét 00 értékre állítja. Itt jegyezzük meg, léteznek olyan processzorok, ahol lapcímzésnél külön be kell állítani a lapcímet, amin belül hat az utasítás.

Röviden összefoglalva, a 0103 cím a PC programszámlálóból kijut az AB címsinre, az RD olvasási paranccsal együtt a cím kikéri a tárból az utasítás műveleti kódját, ami a DB adatsinen és a mikroprocesszor belső sinjén keresztül bejut az IR utasításregiszterbe, PC programszámláló pedig a következő értéket veszi fel, ami 0104 (11.16. ábra).



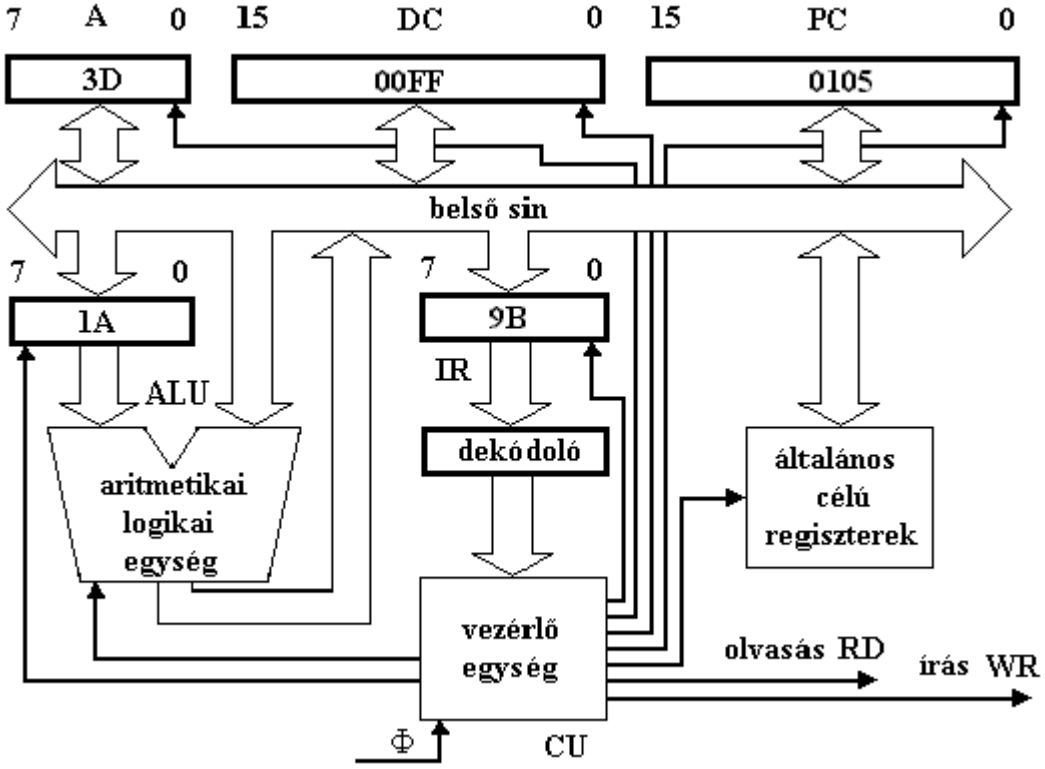
11.16. ábra: A második utasítás előkészítési fázis utáni állapot

A műveleti kód után helyezkedik el a 8 bites cím. A PC utasításszámlálóból a 0104 cím kiküldésével és az RD olvasás paranccsal a memóriából beolvas a mikroprocesszor egy 8 bites adatot a DC adatszámláló alsó helyértékű bájtjába, ugyanakkor a magasabb helyértékű bájt értékét nulláza, ami a 00FF címet adja esetünkben.

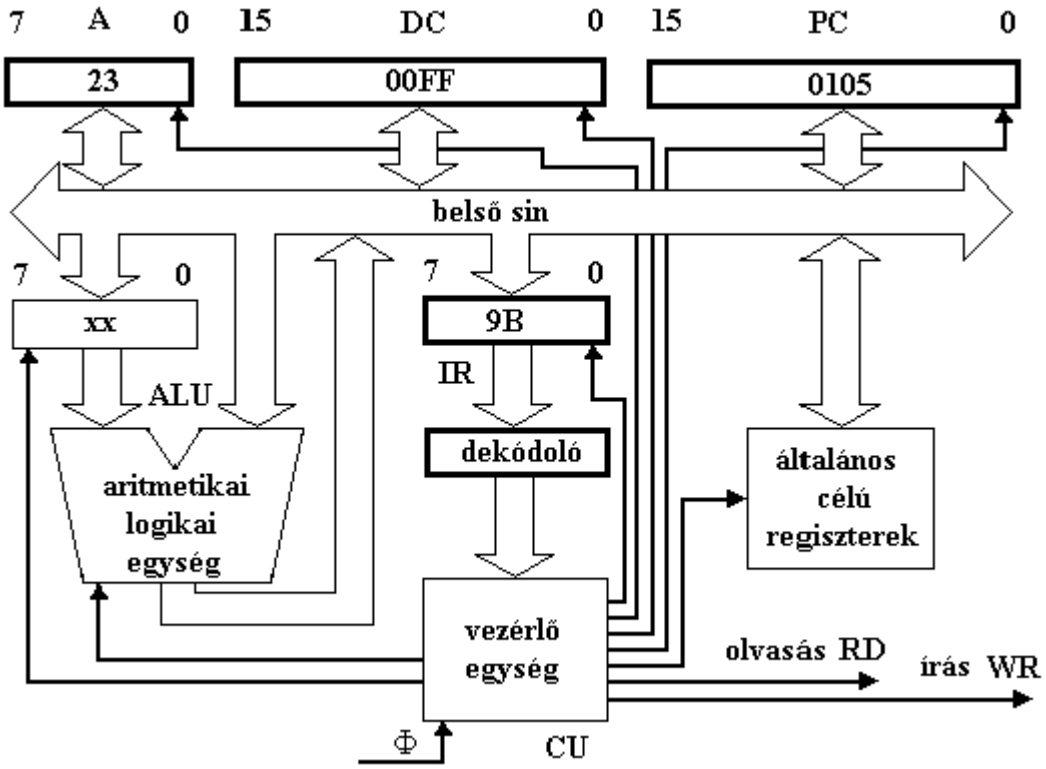
A PC programszámláló értékének mégnövelésével az új érték, a 0105 már a következő utasítás műveleti kódjára mutat. Az előző folyamat a második ciklus eseménye volt, ez az új állapot látható a 7.17. ábrán. Ekkor a processzor DC adatszámlálója azt a címet tartalmazza, ahol a memóriában megtalálható a második operandus (a 00FF címen az 1A érték). Most a belső sinen keresztül két lépésben a DC adatszámláló tartalma kikerül az AB címsinre, a CU vezérlőegység RD olvasásparancsot állít elő, a memória a DB adatsinre másolja a címzett memóriatartalmat. Ezt az értéket, ami a második operandus, átveszi a mikroprocesszor belső sinjén keresztül az ideiglenes regiszter, majd még ugyanebben a ciklusban a vezérlőegység vézérlő jelei összeadják az ALU aritmetikai-logikai egység segítségével az A akkumulátor tartalmát és az ideiglenes regiszter tartalmát, az eredményt visszaírva az A akkumulátorba.

Ez a végeredmény látható a 11.18. ábrán. A 11.19. ábra idődiagramja az események időbeli lefolyását mutatják. Az első utasításhoz képest a második utasítás egy ciklussal rövidebb ideig tart, ami logikus, hiszen 16 bites cím helyett 8 bites cím címezi az adatot a memóriában.

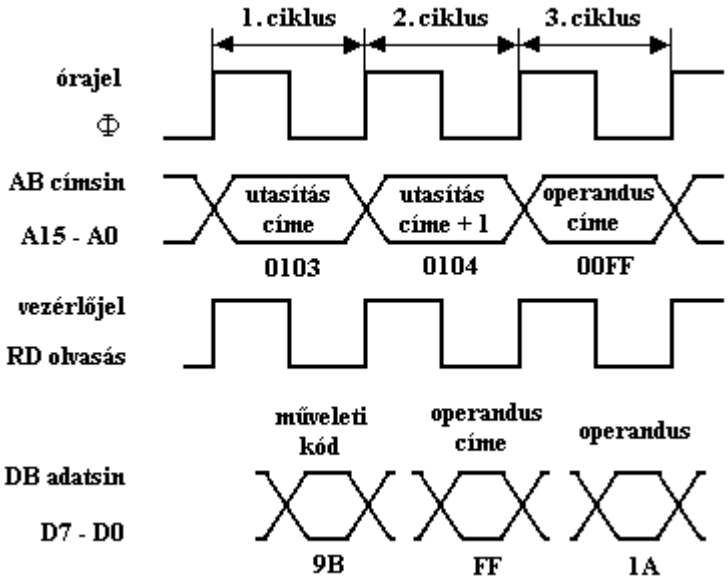
Még egy tényre fel kell hívni a figyelmet, szépen látszik, hogy a memóriához fordulás lényegesen hosszabb ideig tart, mint a processzoron belüli regiszter-regiszter művelet.



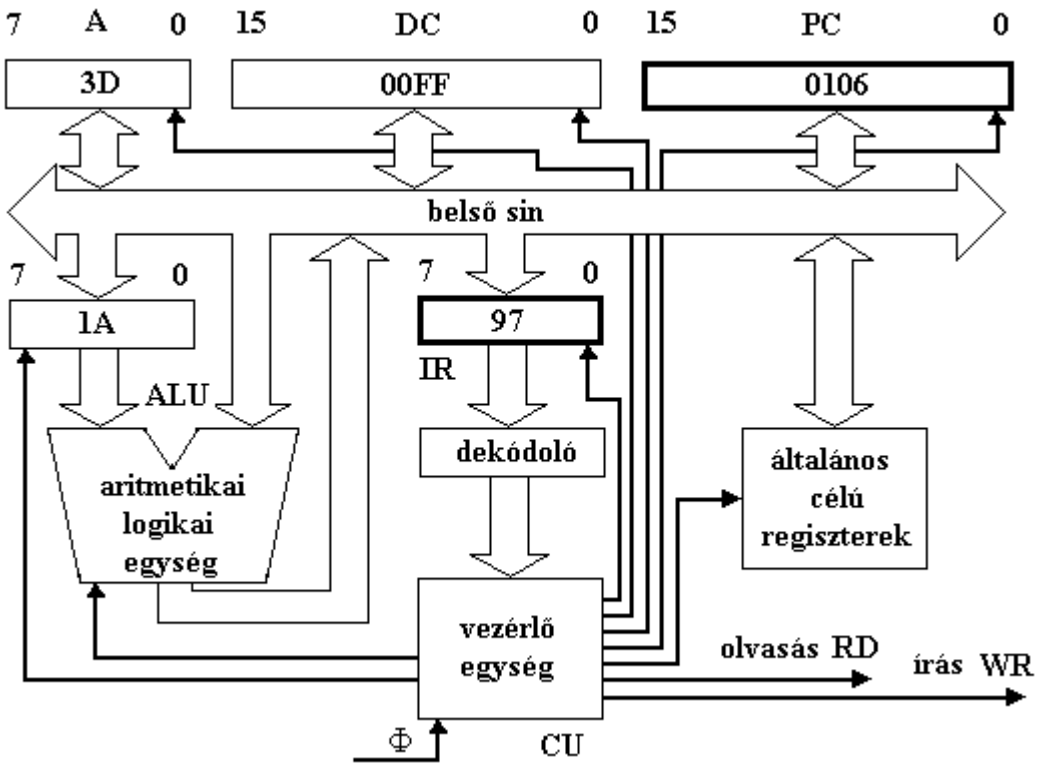
11.17. ábra: A második utasítás második ciklusa, az operandus címének alacsonyabb helyértékű bájtjának beolvasása (magasabb helyérték 00)



11.18. ábra: A második utasítás harmadik ciklusa, az operandus beolvasása az ideiglenes regiszterbe, valamint összeadása az A akkumulátorban levő operandussal

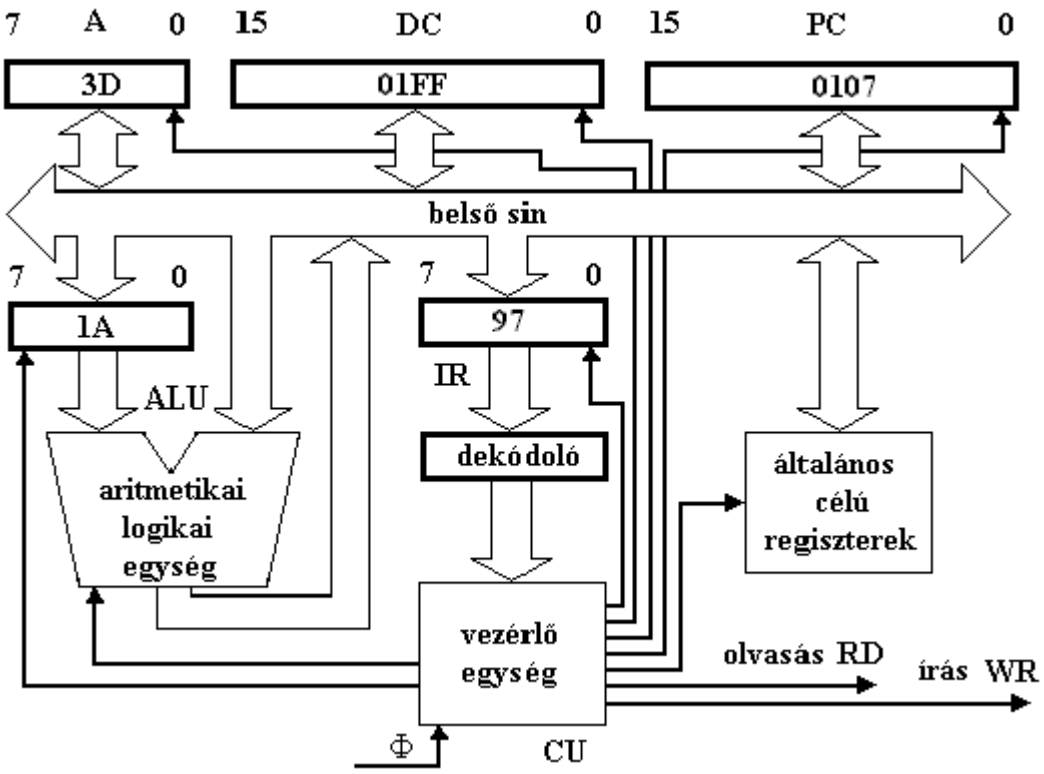


11.19. ábra: A második utasítás egyszerűsített idődiagramja



A harmadik utasítás feladata az A akkumulátor tartalmának kiírása a memóriába. Ez a folyamat az első utasításhoz hasonlóan négy ciklus ideig tart, az első három ciklus lépései hasonlóak az első utasítás 3 első ciklusának lépéseivel, a 11.20., 11.21. és 11.22. ábra szemlélteti ezeket a lépéseket.

11.20. ábra: A harmadik utasítás előkészítési fázis utáni állapota

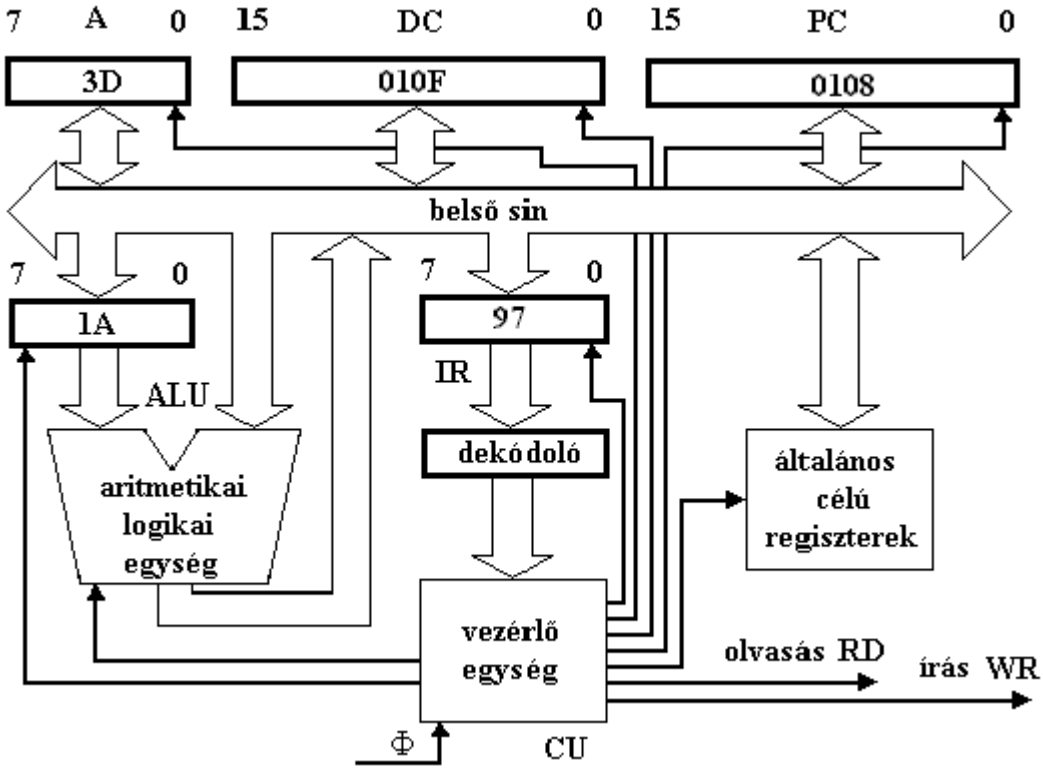


11.21. ábra: A harmadik utasítás második ciklusa, az operandus címének magasabb helyértékű bájtjának beolvasása

11.22. ábra: A harmadik utasítás harmadik ciklusa, az operandus címének alacsonyabb helyértékű bájtjának beolvasása

A harmadik utasítás negyedik ciklusában eddig még nem alkalmazott műveletekre kerül sor. A 11.22. ábrán látszik az, hogy a PC utasításszámláló a program következő utasítására mutat, a DC adatszámláló a memória azon helyét címezi, ahova az A akkumulátorból a processzornak ki kell másolnia az adatot, a 3D értéket.

Eddig mindig csak a memóriából olvasott a processzor adatokat, amik műveleti kódok,



címek részei és operandusok. Ezek az adatatok a DB adatsint használták, ugyanez a helyzet most is az adatkiírással, egyedül az átvitel iránya fordított. A DC adatszámláló kerül címként ki az AB címsinre, majd WR írási külső vezérlőjel jelzi a memória felé az adatbírás igényét. Az A akkumulátorból az érték a processzor belső sinjére kerül, ahonnan átkerül a DB adatsínre, innen pedig a tárba, a megfelelő címre. A harmadik utasítás idődiagramja látható a 11.23. ábrán.

11.23. ábra: A harmadik utasítás egyszerűsített idődiagramja

A 11.1. táblázatban a fent ismertetett mikroprocesszormodell működését lehet nyomon követni. A táblázatból, de az ábrákból is kiderül, hogy egy utasítás előkészítése és végrehajtása több alapművelet ismétlését igényli. Minden utasításban szereplő bájt bekérése egyszerű vezérlőjelekkel történik, amiket a vezérlőegységben levő mikroprogram állít elő. A mikroszámítógép, illetve a mikroprocesszor elemei több műveletben is szerepelnek, igaz van olyan rész amely csak egyfajta feladat elvégzését teszi lehetővé, de az elemek nagy része univerzális. Ilyen többek között a mikroprocesszor belső sinje, de elmondható ez a mikrogép DB adatsinjéről is. Akár az utasítás műveleti kódjának bekérése, akár egy külső (operatív) mamóriacím beolvasása, vagy kiírása történik, minden művelet használja az adatsint. A vezérlőegység az utasítás alapján dönti el, mely egységbe kell beírni az adott értéket.

A fenti elemzésből látható az is, hogy a szóhosszúság növelése gyorsítja a rendszer működését, ugyanakkor bonyolítja a hardvert, hiszen például a 32 bites rendszerben a belső sin, de a többi elem is 32 bites, ami gyártási problémákkal jár. Nem állítható egyetlen mikroprocesszorról sem az, hogy jobb mint a másik, hiszen mindig a feladatnak megfelelő processzort, memóriakapacitást, külvilághoz való illesztést kell megválasztani egy adott feladat megoldásához.

11.1. táblázat: Állapotok AB cím- és DB adatsinen az utasítások végrehajtásakor

utasítás	ciklusok száma	ciklus	RD olvas	WR ír	AB címsin	DB adatsin
B6	4	1.	1	0	Utasítás címe = 0100	Műveleti kód B6
		2.	1	0	Utasítás címe +1 = 0101	Operandus címének felső bájtja = 02
		3.	1	0	Utasítás címe +2 = 0102	Operandus címének alsó bájtja = 01
		4.	1	0	Operandus címe = 0201	Operandus = 23
9B	3	1.	1	0	Utasítás címe = 0103	Műveleti kód 9B
		2.	1	0	Utasítás címe +1 = 0104	Operandus címe = FF

		3.	1	0	Operandus címe = 00FF	Operandus = 1A
97	4	1.	1	0	Utasítás címe = 0105	Műveleti kód 97
		2.	1	0	Utasítás címe +1 = 0106	Operandus címének felső bájtja = 01
		3.	1	0	Utasítás címe +2 = 0107	Operandus címének alsó bájtja = 0F
		4.	0	1	Operandus címe = 010F	Operandus = 23

Ellenőrző kérdések

1. Mi az utasítás ?
2. Mi a program ?
3. Hogyan működik a számítógép, mi a program szerepe működésében ?
4. Melyik lépésekből áll az utasítás végrehajtása ?
5. Programozás szempontjából melyek a számítógép legfontosabb elemei ?
6. Mi a mikroprocesszor feladata a számítógépben ?
7. Mekkora az a szóhosszúság amely ma a legelterjedtebb ?
8. A mikroprocesszor tervezőmérnökei milyen szempontokat vesznek figyelembe a számítógép utasításkészletének meghatározásakor ?
9. Milyen alakban tárolja a számítógép a programot ?
10. Hogyan néz ki egy utasítás általános alakja ?
11. Mi a műveleti kód, mit határoz meg és hol helyezkedik el az utasításban?

12. Az utasítás a műveleti kód mellett milyen mezőket tartalmaz még ?
13. Ismertesse a 3 címes utasítást .
14. Ismertesse a 2 címes utasítást .
15. Istmertesse az í címes utasítást .
16. Foglalja össze a 3, 2 és 1 címes utasítások előnyeit és hátrányait .
17. Hogyan lehet 8 bites géppel megoldani a műveleti kód és cím/adat ábrázolását ?
18. 2 x 8 bites cím mekkora memória megcímzését teszi lehetővé ?
19. Ha 8 bit szóhosszúságú a műveleti kódrész, hány különféle utasítás hozható létre ?
20. Rajzolja fel a mikroprocesszor modellt és sorolja fel a modell legfontosabb alkotóelemeit .
21. Nyolc bites belső sin esetében milyen szóhosszúságot alkalmaznak az egyes elemeknél ?
22. Mi az **A** akkumulátor és mire szolgál ?
23. Mi a **PC** programszámláló és mire szolgál ?
24. A PC programszámláló műveleti kódot címez az utasításon belül, de mi történik akkor, ha tévesen a címrészt címezi ?
25. Mi a **DC** adatszámláló és mire szolgál ?
26. Mi az **IR** utasításregiszter és mire szolgál ?
27. Mi a CU vezérlő egység és mi a feladata processzoro, illetve a mikrogépen belül ?
28. Sorolja fel és ismertesse az utasításvégrehajtás fázisait .
29. Mi a Φ szinkronizációs órajel és mi a szerepe ?
30. Miért kell a processzornak bekapcsolás után alaphelyzetbe hozni egyes elemeit ?
31. Mi a belső és mi a külső vezérlő jel ?
32. Mi a mikroprogram és hol helyezkedik el ?
33. Ismertesse egy utasításvégrehajtás menetét a sineken és regiszterekben történt adatváltozások követésével .
34. Ismertesse az egyfázisú és kétfázisú órajelet .
35. Adjon meg egy operatív tárban elhelyezkedő mintaprogramot és magyarázza el segítségével a mikrogép működését .
36. Rajzolja fel egy utasítás idődiagramját és ismertesse az idődiagram segítségével az utasítás végrehajtását .
37. Táblázatosan ábrázolja az utasításvégrehajtást .
38. Mi az RD és WR külső vezérlőjel szerepe és mely egység hozza létre ?

11.3 AZ **ILP** (Instruction Level Parallel) – UTASÍTÁSSZINTEN PÁRHUZAMOS MŰKÖDÉSŰ PROCESSZOROK

Nagyobb sebbességgel dolgozó processzorok állíthatók elő az órajel-frekvencia növelésével, de ennek a megoldásnak technológiai korlátai vannak. Célszerű megtalálni az utasításvégrehajtás során fellelhető esetleg páruzamosan, egyidőben végrehajtható részfeladatokat. Ezt a módszert átlapolt feldolgozásnak, pipelining-nak nevezzük. A pipe angol szó, csövet, a pipelining pedig csővezetéket jelent. Talán az adatcsatornás, vagy a futószalagos elnevezés közelíti meg legjobban az eredeti elnevezést.

11.3.1 AZ UTASÍTÁSOK GYORSÍTÁSA PÁRHUZAMOSÍTÁSSAL, A PIPELINING

Tekintettel arra, hogy a gépi utasítások elemi lépései gyakorlatilag mind külön hardvert igényelnek, ezért egy adott utasításhoz tartozó elemi lépés végrehajtása után a következő utasítás elemi lépése azonnal végrehajtható. Ugyanez érvényes a többi elemi lépésre, illetve az elemi lépéshez kötődő megfelelő hardver erőforrásra.

Bővítsük ki az imént tárgyalt processzormodell két (utasítás lehívás és végrehajtás) elemi lépését háromra:

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

A 11.23. ábra szemlélteti négy utasítás esetén az átlapolási (pipelining) techikát.

utasítás-lépés	**i-2**	**i-1**	**i**	**i+1**	**i+2**	**i+3**
1.	**F1**	**D1**	**E1**			
2.		**F2**	**D2**	**E2**		
3.			**F3**	**D3**	**E3**	
4.				**F4**	**D4**	**E4**

11.23. ábra: 4 utasítás esetén az átlapolási technika

A fent ismertetett módszer természetesen idealizált megoldás, a valóságban az átlapolási technika sokkal bonyolultabb, több feladat megoldását is meg kell oldani.A legsúlyosabb problémák, amelyeket kezelni kell és amelyeknek meg kell találni a leghatékonyabb megoldását a következők:

- eltérő időtartamú elemi lépések, fázisok, gondoljunk például arra, hogy az adatlekérés ideje regiszterből (a processzor része) és külső operatív memóriából 1:10 arányban áll egymással,
- a vezérlésátadó utasítások (feltételes és feltétel nélküli ugrások) megzavarják az utasítások soros végrehajtását,
- a megszakítások kérése, kiszolgálása az előző pontban ismertetett problémát okozza, gátolja a futószalag folyamatos feltöltését,

- egymás után következő utasítások,ahol a következő utasítás az előző utasítás eredményére hivatkozik, amely még nem jöhetett létre,
- hardver ütközések, igen gyakori a buszhasználat során fellépő ütközés, ez azért okoz komoly problémát, mert a műveleti kód mellett az operandusok is a főtárban vannak, lehívásuk csak ugyanazon a buszrendszeren keresztül történhet meg.

A felsorolt probémák megoldására több módszer van, ilyen a program fordítása során töténhet, ez statikus megoldási módszer. Dinamikus a problémamegoldás, ha a hardverrel futás közben történik.

Néhány megoldás szoftver eszközzel (fordítóprogram):

- üres utasítások beiktatása, vagyis NOP-ok alkalmazása, annyi ideig kell várakoztatni a pipelinig-et, ameddig befejeződik a memóriautasítások végrehajtása,
- utasítások átrendezése, amennyiben lehetséges az utasítások sorrendjét úgy kell megváltoztatni, hogy a lépések logikai értelmét ne befolyásoljuk, így hasznos utasítások kerülnek oda, ahol a pipelining várakozik.

Néhány megoldás hardver eszközzel:

- scoreboarding, ahol minden használt regiszter egy listába kerül, azért, hogy ha egy adott utasítás hivatkozik rá, akkor azt a rendszer várakoztassa,
- data forwarding esetén a hardver adatátadást biztosít a második utasítás számára, amennyiben azok számára ugyanaz az adat szükséges (ez csak akkor valósul meg, ha az első utasítás befejeződött).

Más architektúra alkalmazása:

- a harvard architektúránál az adat- és programmemóra szétválasztása külön adat- és programsint követel, így itt nincs sinütközés, ez a megoldás használatos a Pentium processzoroknál is, L1 cashe esetében.

Egy program utasításainak általában 10-15 %-át a vezérlésátadó utasítások teszik ki. Amikor egymás utáni utasításokról van szó, a csővezetéket egymást követő címekről kell tölteni, ezt a szabályosságot bontja meg a vezérlésátadó utasítás (ugróutasítások). Ilyenkor más memóriacímről kell a következő bájtot lehívni.

A fenti probléma legegyszerűbben a pipeline leálításával oldható meg. Vezérlésátadó utasításnál a processzor két módon oldhatja meg a problémát:

- megállítja a pipeline töltését addig, amig bizonytalan, hogy a programnak új címre kell ugrania, vagy a következő utasítástól (címtől) kell folytatni,
- előre feltételezett módon kezeli a problémát, vagyis feltételezi, ogy nem lesz ugrás, ha ez így történik, normálisan folytatódik az utasítások végrehajtása, ellenkező esetben törli a pipeline-ban tárolt utasítássort.

A ma korszerű processzorok már az ún. spekulatív elágazás feldolgozása módszerrel megkísérlik kitalálni, megjósolni az utasítás várható kimenetelét. Ez két módon történhet meg:

- **statikus módszernél a fordítóprogram az ugrási feltételekből kiszámítja a legnagyobb valószínűséggel bekövetkező ugrási címeket ami alapján szervezi meg a pipeline működését,**
- dinamikus esetben programfutás közben a processzor egy táblázatot hoz létre, ahol az ugróutasítások címét vezeti, illetve az ugrások kimenetelét, ami alapján megjósolja a pilpeline felépítését.

A mai Pentium processzorok a dinamikus előrejelzésre beépített két gyors cache tárat használnak:

- a BTB (Branch Target Buffer) tárolót, amely az eddig megvalósított ugróutasítások átlaga alapján elágazási valószínűséget tárol,
- az RSB (Return Stack Buffer) tárolót, amely tárolja a szubrutinból való visszatérési címeket.

A statikus, illetve dinamikus előrejelzés létrehozása fontos lépés a processzorok korszerűsítése terén, ugyanis ha pl. egy ciklust n-szer kell végrehajtani, akkor n-1 alkalommal visszaugrik a program a ciklus elejére, csupán egy alkalommal, a ciklus végén hagyja el azt.

11.4. A SZUPERSKALÁR PROCESSZOROK

A Neumann elven megépített gépek gyakorlatilag a 80-as évek kezdetéig számítottak korszerű gépeknek, ahol a soros (szekvenciális) utasításvégrehajtás volt a jellemző. Úgyszintén skalárprocesszor a RISC processzor, ami utasítás szinten párhuzamos működésű (ILP, Instruction Level Parallel) gép.

A 90-es gépek elején megjelenő megoldásokra jellemző a több pipelining, ami alkalmas a párhuzamos feldolgozásra, több párhuzamos működésű vegrehajtó egységet tartalmaz. Ezt csak úgy lehetett elérni, hogy a szekvenciális utasításkibocsájtást párhuzamossal helettesítették, ami lehet:

- **VLIW (Very Large Instruction Word) felépítésű, ahol az utasítások több műveletet tartalmaznak,**
- **dinamikus ütemezéssel létrehozott szuperskalár processzorok, ezek ciklusonként több utasítást hajtanak végre, a 90-es évek processzorai a 4x-es kibocsájtást használták.**

Természetesen ez az új megoldás új problémákat hozott magával, amiket meg kellett oldani. Néhány jelentkező probléma és annak megoldása:

- **párhuzamos dekódolás, előrekódolással,**
- **szuperskalár utasításkibocsátás, amit utasítás várakoztatással, regiszterátnevezéssel és spekulatív elágazás-kezeléssel,**
- **párhuzamos utasításvégrehajtás, ahol előfordulhat, hogy az utasítások befejezési sorrendje esetleg elérheti a soros utasításokét.**

A Pentium processzorok mivel CISC típusúak, ezért tervezésükkor két komoly problémát kellett megoldani:

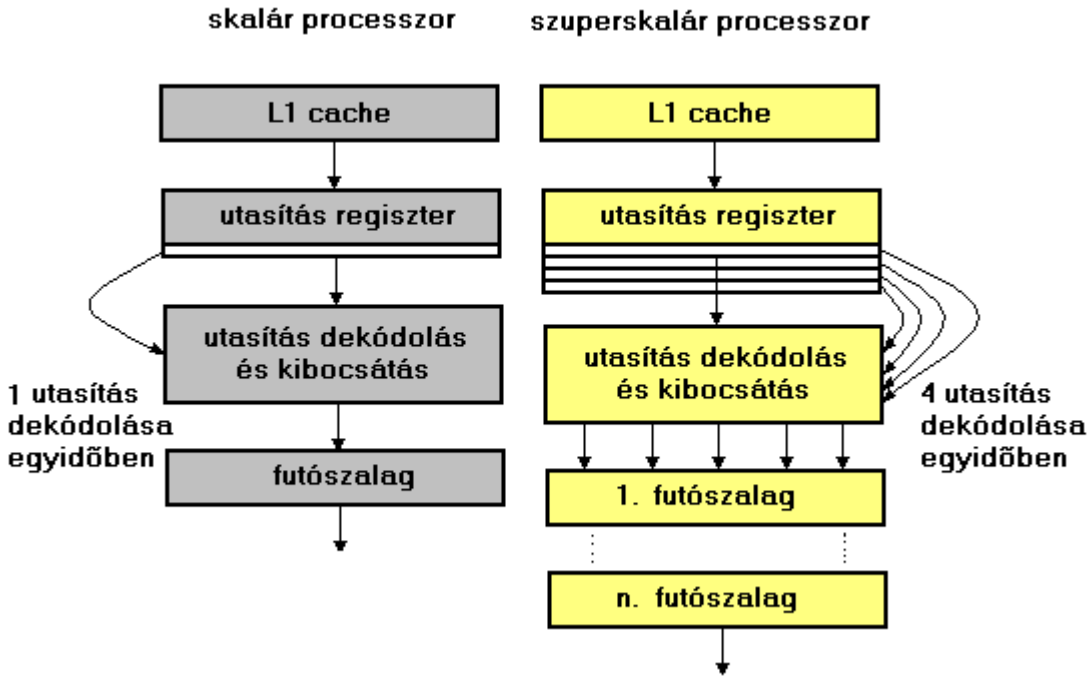
- **az utasítások eltérő (változó) szóhosszúsága és**
- **a különféle memóriautasítások.**

A fenti problémát csak a gép RISC emulációjával tudták eredményesen megoldani, vagyis a processzorokban egy RISC mag oldja meg a feladatot, bontja fel mikroutasításokra az x86-os utasításokat.

11.4.1. A PÁRHUZAMOS DEKÓDOLÁS ELVE

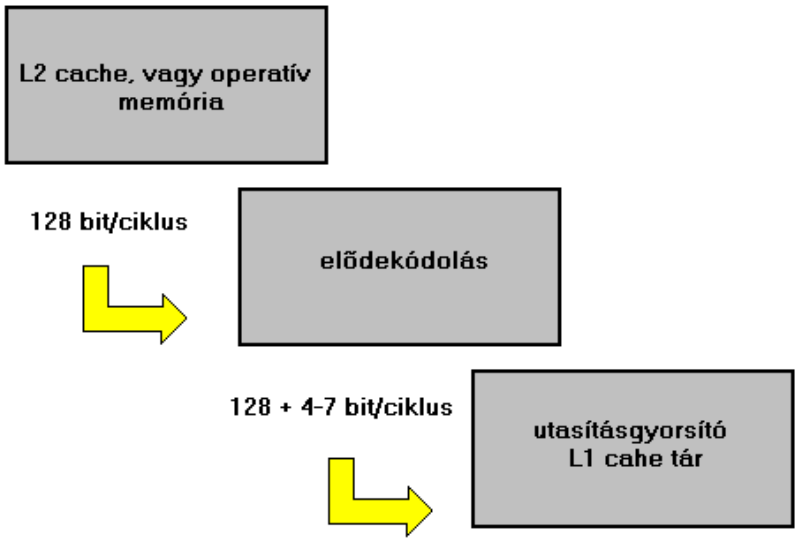
Mivel a végrehajtó egységek párhuzamosan dolgoznak, ezért a dekódolt utasításokat is párhuzamosan kell kapniok, vagyis az utasításdekódolás párhuzamos.

A 11.24. ábrán látható a hagyományos (skalár) és egy 4x-es kibocsátású processzor összehasonlítása.



11.24. ábra: dekódolásás és utasításkibocsátás a hagyományos (skalár) és a szuperskalár processzornál

A két elv között nagy eltérés van. Addig, amíg a skalár processzor egy ciklus alatt egy utasítást dekódol, figyelnie kell hogy a kibocsátandó utasítás függ-e a végrehajtás alatt álló utasítástól. Ettől eltérően a szuperskalár processzor ciklusonként 4 utasítást dekódol, hasonlóan a skalár processzorhoz figyelnie kell arra, hogy a kibocsátandó utasítások függenek-e az éppen végrehajtott tasításoktól, valamint arra is kell figyelnie, hogy a négy kibocsátandó utasítás egymással függőségi viszonyban van-e. Mivel ez a függőségvizsgálat időigényes, ezért bevezették az elődekódolást (11.25. ábra).



11.25. ábra: az elődekódolás menete

Az ábrából látható,hogy az elődekódolásmegtörténik már akkor, amikor az utasítás a főtárból (illetve L2 cahe tárból) L1 utasításgyorsító cache tárba kerül. Ilyenkor elődekódoló bitek aódnak az utasításhoz, melyek információt jelentenek a további feldolgpzásnak.

11.4.2. SZUPERSKALÁR UTASÍTÁSKIBOCSÁTÁS ÉS VÁRAKOZTATÁS

Ez az eljárás két részből áll:

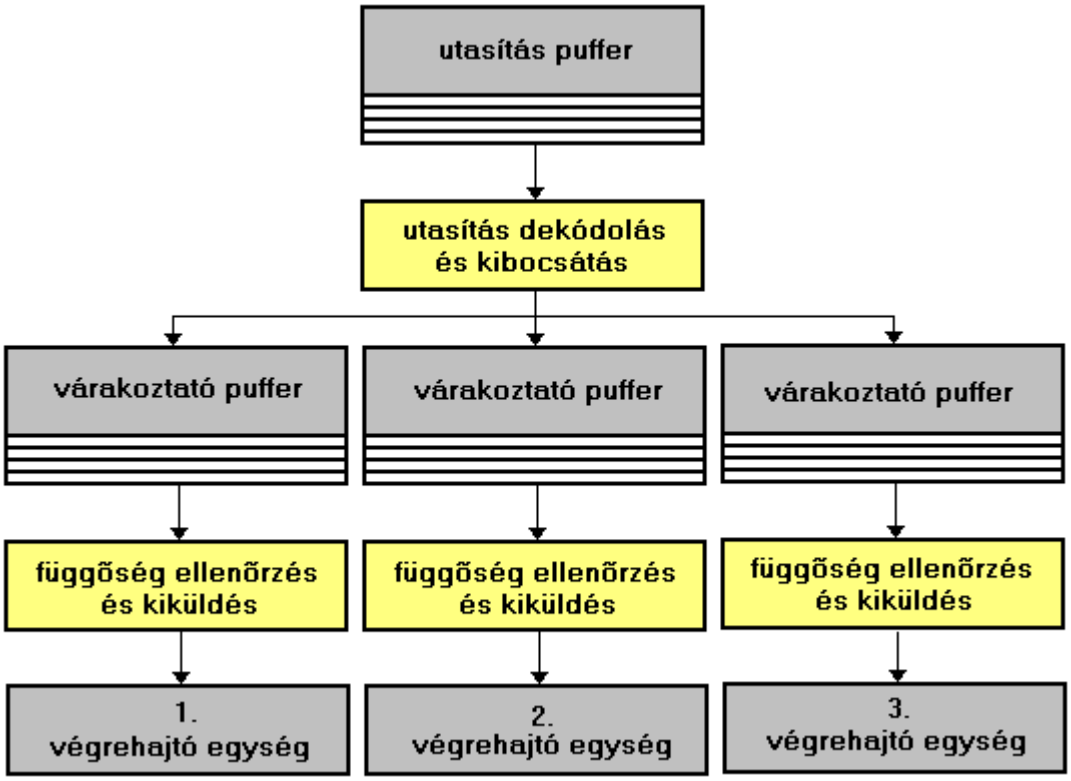
- **a végrehajtó egységhez továbbítható utasítások száma, amiket a processzor ciklusonként kibocsáthat,**
- **a kibocsáthatóság stratégiájától függ a függőségek kezelése, ami lehet:**
  - **a hivatkozott regiszterek közötti adatfüggőség,**
  - **olyan vezérlésátadó utasítások, ahol a feltétel kiértékelése nem befejezett.**

A processzor regiszterátnevezéssel az adatfüggőséget feloldhatja.

A vezérlésfüggőségeket a processzor két módon oldhatja fel:

- **a processzor a feltétel kiértékeléséig felfüggeszti az utastáskibocsátást,**
- **a processzor megbecsüli a feltételes utasítás kimenetelét.**

Összefoglalva, az adat- és vezérlésfüggőségek kezelése a processzor részéről történhet:
utasításkibocsátás blokkolásával,
illetve a függő utasítások várakoztatásával a végrehajtó egység elött (11.26.ábra).



11.26. ábra: függő utasítások pufferelése

A megelőző utasítás végrehajtása oldja fela következő utasítását a várakozási pufferben.

11.4.3.REGISZTER ÁTNEVEZÉS

Már találkoztunk azzal a ténnyel, hogy egy processzor összes regiszterét nem használhatja közvetlenül a programozó, vagyis a gépi szintű programozás csak bizonyos regiszterek elérését teszi lehetővé. Ilyenkor nem lehetséges minden adatot processzoron belül kezelni, szükséges az operatív memóriához fordulás. Ez a helyzet ún. ál-adatfüggőséget okoz.

Az ál-adatfüggőség megszüntetése a regiszter átnevezés technikájával oldható meg.

Ezt a problémát, illetve a probléma megoldását szemléltetni lehet az I8086, illetve a későbbi, PENTIUM II processzorokon. Mivel az INTEL a fejlesztéseknél a kompatibilitás megtartása érdekében az előző modellekre építette az újakat, az I8086 négy 16 bites regisztere, az AX, BX, CX és DX a PENTIUM II processszornál 32 bites EAX, EBX, ECX és EDX regiszterekként került megvalósításra. Ugyanakkor, a fejlettebb technológia már 40 általános célra felhasználható regiszter létrehozását engedélyezte, amiket a programozó nem tud elérni programból. A programozónak gyakorlatilag csak a memóriában való adattárolás áll a rendelkezésére, amit közvetlen memóriacímzéssel, vagy verembe való mentéssel old meg. Ennek a technikának az a baja, hogy így a kb. 1-10

sebességviszony miatt (a processzor regiszterműveletei kb. tízszer gyorsabban hajtódnak végre, mint a memória-műveletek) rendkívüli mértékben lelassul a programfutás.

Elemezzük a következő problémát: a memória osszeg1 **rekeszének tartalmát növeljük meg kettővel, majd az** osszeg2 **rekesz tartalmát csökkentsük néggyel. A következő programrészlet tartalmazza a probléma megoldását:**

```
mov     eax,2
add     [osszeg1],eax
mov     eax,4
sub     [osszeg2],eax
```

Adatfüggőséget állapíthatunk meg, mivel ezek az utasítások eax regisztert használják, így nem párhuzamosítható az utasításvégrehajtás. Amennyiben a processzor legalább két regisztert tartalmaz, a probléma a következőképpen oldható meg:

```
mov     reg1,2
add     [osszeg1],reg1
mov     eax,4
sub     [osszeg2],reg2
```

A regiszterátnevezésnek két módja lehetséges:

- **a fordítóprogram segítségével, e a statikus átnevezés,**
- **a hardver segítségével, ami futás közben történik, ez a dinamikus technika.**

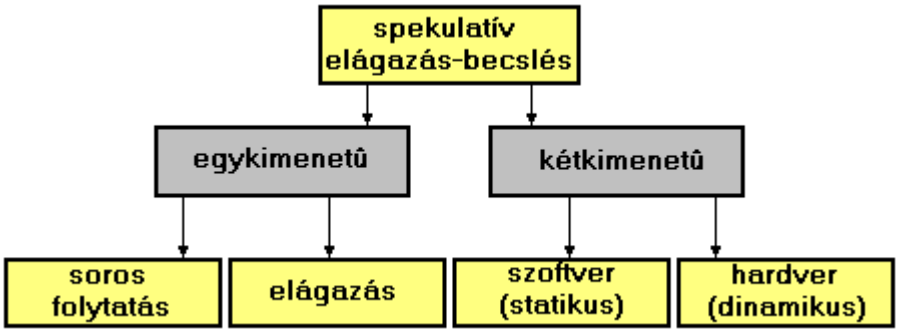
A mai korszerű processzorok az utóbbi, dinamikus technikát alkalmazzák.

Maga a folyamat a úgy zajlik le, hogy a rendszer tartalmaz egy táblázatot a meglevő regiszterek neveivel, egy másikat pedig az átnevezés adataival, amely kettőt a leképzőtáblázat köt össze.

11.4.4 SPEKULATÍV ELÁGAZÁS-FELDOLGOZÁS

A pipelining zavartalan működését az ugróutasítások megzavarják, hiszen amiatt, hogy az ugrási cím a memóriában van, nem párhuzamisíthatók az egyes műveletek. Ennek a problémak egy lehetséges megoldása az ún. spekulatív elágazásfeldolgozás, amikor a processzor megkísérli megjósolni az utasítások kimenetelét.

A 11.27. ábra foglalja össze a lehetséges spekulatív elágazás-becslési módszereket.



11.27. ábra: a lehetséges spekulatív elágazás-becslés módszerei

Látható, hogy az egykimenetű, vagy rögzített módszernél az eredmény mindig ugyanaz, vagy elágazik, vagy nem. A kétkimenetű módszernél már statisztikai módszerek szerint történik abecslés, amit régebben szoftver űton, a fordításkor, illetve a programösszefűzéskor (LINK) alkalmaztak, míg a mai modern processzoroknál már a hardver futás közben, dinamikusan végzi.

11.4.4 PÁRHUZAMOS VÉGREHAJTÁSI TECHNIKA

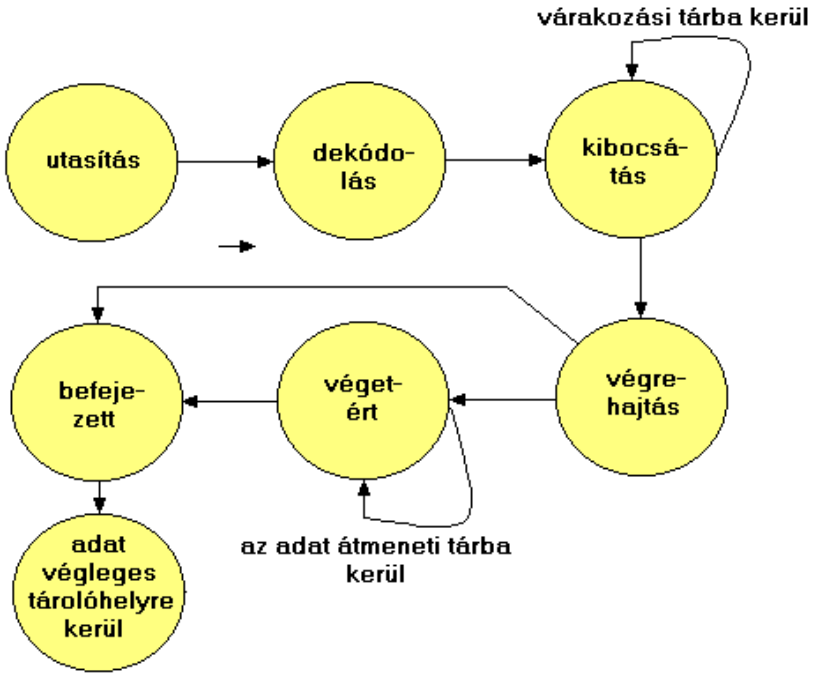
Amennyiben a processzorban több végrajtó egység található, akkor az egyes utasítások eredményei párhuzamosan keletkeznek, ami eltér az eredeti utasítássorrenddtől. Az így keletkezett eredményeket ilyenkor addig tárolni kell, amíg véglegesen nem kerülnek a helyükre.

A fenti probléma megoldása csak akkor eredményes:

- **ha megtörténik az eredeti utasítások sorrenben való végrehajtása, amit másképpen processzor konzisztenciának nevezünk,**
- **és akkor, ha a memória-hozzáférés is az eredeti utasítássorrend szerint megy végbe, ezt pedig memória-konzisztenciának hívjuk.**

A processzor- és memória konzisztencia képezi a mikroszámítógép soros konzisztenciáját. Ennek megvalósítására a szuperskalár processzoroknál egy ún. átrendező regisztertár szolgál (ROB, ReOrder Buffer). A 11.27. ábra szemlélteti a soros konzisztencia elvét egy állapotdiagram segítségével. Itt látható, hogy egy utasítás lehet:

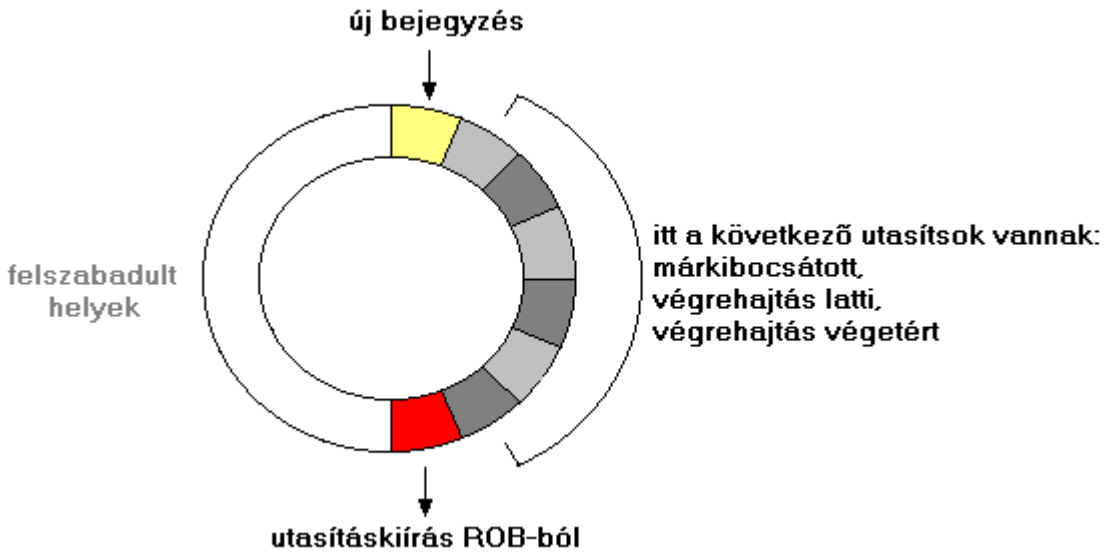
- **végetért állapotú, azaz a hozzárendelt végrehajtó egység már befejezte az utasítást, ugyanakkor az eredményadat egy átmeneti tárolóba íródik,**
- **befejezett állapotú, amikor a processzor a keletkezett eredményt már a feltételek létrejötte miatt beírta a végleges helyére, a memóriába.**



11.27. ábra: utasítások adott időpillanatbeli állapotai

Láthattuk, hogy a párhuzamosan végrehajtott utasítások különböző állapotokban lehetnek. Ezeket az állapotokat nyilván kell tartani, azért, hogy meg lehessen állapítani, mely utasítás adata mikor kerülhet végleges helyére. Ezt lehet a ROB-ba való beírással és törléssel lehet megoldani.

Világos, hogy egy adott utasítás eredményadatai csak akkor kerülhetnek az átmeneti tárolóból az előírt végleges tárolói helyre, ha az adott utasítást megelőző összes utasítás, a kívánt végrehajtási sorrendben befejeződött. Ezt a folyamatot lehet a ROB használatával szabályozni. Amikor az utasítás a ROB-ból tehát törlődik, az utasítás befejeződött. A ROB működési elvét a 11.28. ábrán követhetjük nyomon.



11.28: A ROB működési elvének követése

11.4.5 EGY PROCESSZOR BEMUTATÁSA, A PENTIUM II

Néhány fontos műszaki adat:

elemek száma:	**több mint 7,5 millió trnzisztor,**
processzor órajel:	**233, 266, 300, 350 és 400 MHz,**
gyorsítás:	**utasítás L1 cache, 16 KB,**
	adat L1 cache, 16 KB,
	256 (512) KB L2 cache,
buszsebesség:	**66 és 100 MHz,**
adatbusz:	**64 bit,**
címbusz:	**32 it.**

A gépi utasításokat elemi utasításokra bontja le a processzor (RISC megoldás), amiket egymással párhuzamosan hajt végre (szuperskalár megodás). A gyorsítás érdekében külön busz az adat- és utasításcache-hez.

Feltételes ugrásoknál spekulatív elágazásvégrehajtás, egy 512 beírás kapacitású regiszterrel (BTB – Brancs Target Buffer).

12. A MIKROPROCESSZOR ELEMEI

12.1. BEVEZETÉS

A mikroprocesszor modelljét tanulmányozva levonható a következtetés, hogy tulajdonképpen regiszterek és a regiszterek közötti kapcsolatok (sin, busz) alkot egy mikroprocesszort. Az adatáramlást a vezérlő egység belső, illetve külső vezérlő jelei szabályozzák.Van amikor a vezérlő egység önállóan, van amikor kényszer hatására (az utasítást végrehajtva) végzi a vezérlő jelek előállítását.

Az, hogy a tervezőmérnökök milyen struktúrát hoznak létre függ a számítástechnika fejlettségi fokától, a technológiai korlátoktól és nem utolsó sorban attól is, hogy milyen feladat megoldására terveznek meg egy-egy mikroprocesszort, mikrovezérlőt.

A hagyományos Neumann és Harvard struktúrájú mikrogépek nagyon hasonló belső architektúrával rendelkeznek, csak részletekben térnek el egymástól (10. fejezet).

A következőkben a komplex, összetet utasításkészlettel rendelkező **CISC** (Complex Instruction Set Computer) számítógépekről lesz szó. Ezeknél a gépeknél a felhasználó által magasszintű programozási nyelven megírt programot valamilyen fordító szoftver segítségével gépi kódra fordítjuk. Ez a gépi kódú programnyelv összetett gépi utasításokkal rendelkezik, amit mikroprogramok segítségével az adott hardveren végrehajt a vezérlő egység, vagyis értelmezi a programot. Ez tehát egy értelmező (interpreter) rendszer.

Egy CISC gép főbb elemei a következők:

- CU vezérlőegység,
- ALU aritmetikai-logikai egység,
- A akkumulátor (esetleg több),
- általános célú regiszterek,
- címregiszterek és
- belső sinek.

12.2. A VEZÉRLŐ EGYSÉG

Feladata a mikroszámítógép összes elemének működését összehangolni, szinkronizálni. Egyrészt az egyes elemi lépéseket egymás után végre kell hajtani ahhoz hogy elvégezze a mikrogép a feladatát, másrészt az elemek műveletvégzési sebességei különböznek, az egymás közötti adatátvételt kell megfelelő időpillanatokban biztositania, vagyis időzítési problémákat kell megoldani.

A gépi kódú utasítás (a műveleti kód rész) tartalmazza a feladatot, de ahhoz hogy ezt a mikroprocesszor végrehajtsa először az:

- **utasításelőkészítési** (fetch) **fázisban** a **CU** vezérlő egység beolvassa a műveleti kódot az IR utasítás regiszterbe, ha szükséges egyéb információkat is megszerez, az így megszerezett információk után következik a
- **végrehajtási fázis** (execute), ahol kivitelezi, végrehajtja az utasítást.

Gyakorlatilag két megoldás, két elv létezik a **CU** vezérlő egység felépítésére:

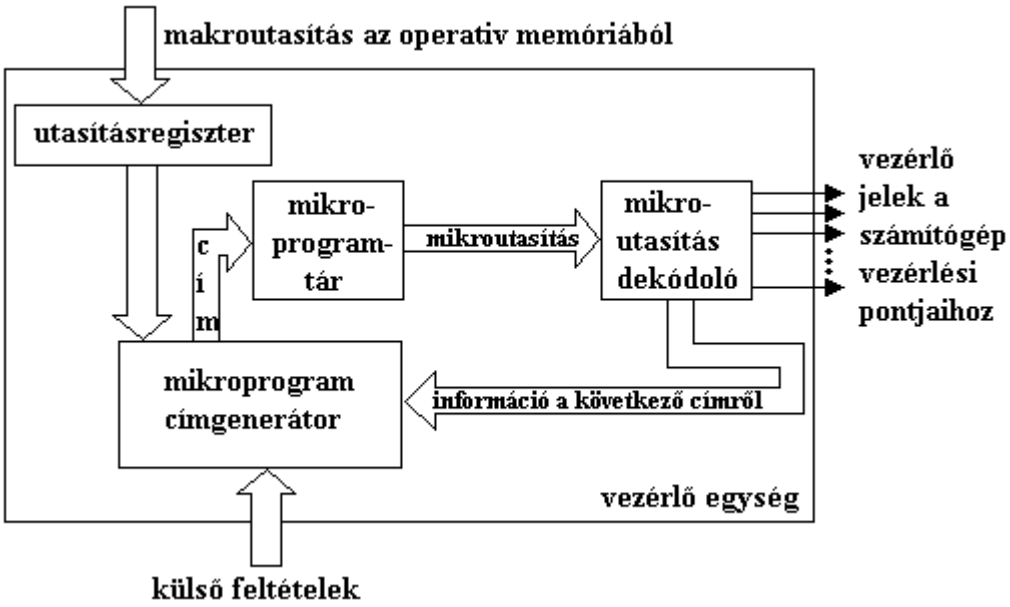
- **huzalozott**, vagy hardver kiépítés, amelynél az utasításvégrehajtás elemi lépéseit kombinációs és sorrendi áramkörök oldják meg, ahol fontos szerepet játszik a vezérlő jelek sorrendje mellett azok időzítése is. Ez a megoldás gyors vezérlőjel generálást jelent, de a rendszer merev, nem könnyű a módosítása, azon kívül költséges is. Főleg célgépekben érdemes az ilyen technika alkalmazása az elérhető nagy működási sebesség miatt.
- **mikroprogramozott**, vagy szoftver megvalósítás, amikor is a vezérlő egység tartalmaz egy **ROM** tipusú memóriát egy kis program tárolására. A gépi kódú utasítás gyakorlatilag a mikroprogramtárban egy címet jelöl ki, amely címtől kezdve egy kis program állítja elő a vezérlő jeleket. Ezt úgy kell elképzelni, mintha a mikroprocesszor egy kis beépített számítógépet tartalmazna. Ez a megoldás olcsóbb az előzőnél, a módosítása is könnyebb mint a másik megoldásnál.

Meg kell említeni, hogy a mikroprogramozott logikával működő processzorok főleg fix, tehát nem változtatható mikroprogramot tartalmaznak. Valójában a felhasználó nem is akar saját maga létrehozni utasításokat, a legalacsonyabb szint, ahol programozni kezdi a számítógépet, az a gépi kódú szint. Olyan esetben, ha különleges utasítások létrehozása a cél, léteznek olyan processzorok is, ahol a felhasználó a számára megfelelő gépi utasításokat hozhatja létre a rendelkezésre álló mikroutasítások felhasználásával. Ez természetesen komoly számítástechnikai szaktudást igénylő feladat.

A vezérlő egység feladata a vezérlő jelek előállítása, amelyek lehetnek:

- **belső vezérlő jelek**, ezek a mikroprocesszoron belül elhelyezkedő elemek munkáját hangolják össze, így az aritmetikai-logikai egység, regiszterek valamint belső sinek közötti adatutakat engedélyezik (nyitják), illetve tiltják (zárják), és
- **külső vezérlő jelek**, amik a processzor és operatív memória, illetve ki- és beviteli eszközök közötti adatátvitelt szabályozzák, ellátják a megszakítással kapcsolatos feladatokat és a sinvezérlést szabályozzák.

A 12.1. ábrán egy olyan vezérlő egység látható, amely mikroprogram segítségével állítja elő a vezérlő jelek sorozatát az utasítás előkészítési fázisban **IR** utasításregiszterbe beolvasott műveleti kód alapján.

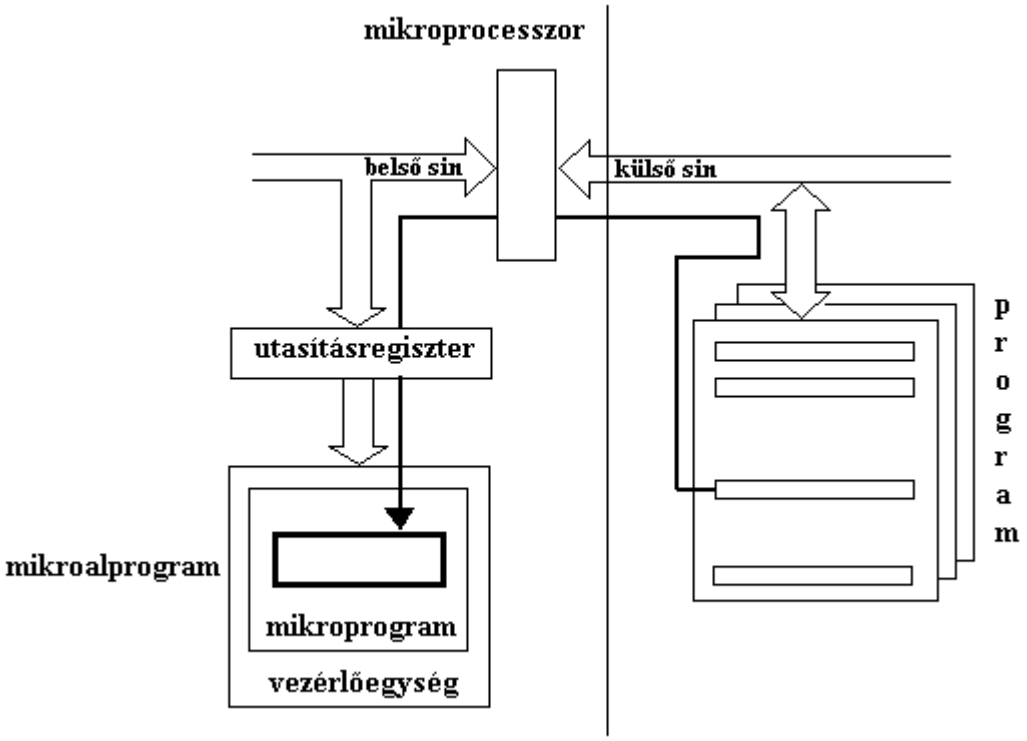


12.1. ábra: Mikroprogramozott vezérlőegység

Az **IR** utasításregiszterbe került **műveleti kód**, a '**külső feltételek'** és az '**információ a következő címről'** együttesen a **mikroprogram címgenerátorban** határozzák meg a mikroprogramtárban a következő címet, ami az adott utasítás következő mikroutasítására mutat. A 'külső feltételek' a külvilágból érkező jelek, ilyen például a **RESET**, a megszakítás stb. Egy makroutasítás mikroprogramja lehet egymás után elhelyezkedő mikroutasítások sorozata, de a 'külső feltételek', illetve az 'információ a következő címről' ettől eltérő címet is előállíthat a mikroprogramtárban.

Egy mikroprocesszor, mikrovezérlő adat-, vagy utasítás szóhosszúsága független a mikroprogramtár szóhosszúságától.

A 12.2. ábrán látható a makroprogram (gépi kódú program) és a mikroprogram közötti kapcsolat. Látható az ábrán, hogy egy gépi kódú utasításhoz, végrehajtásához több alacsonyabb szintű mikroutasítás tartozik. A mikroutasítások száma függvénye a makroutasítás bonyolultságának. Ezeknél a mikroprogramoknál fellelhetők ugyanazok az elemek, mint a gépi kódú programoknál, alprogram, feltételes és feltétel nélküli ugrás, stb.



12.2. ábra: A makroprogram (gépi kódú progarm) kapcsolata a mikroprogrammal

Ahhoz, hogy megértsük a mikroprogram működését az **A** akkumulátor tartalmának komplementálására szolgáló **COM A** utasítás mikroalprogram működését vizsgáljuk.

Az utasítás az **A** akkumulátor minden bitjét ellenkező értékűre állítja, ami egyidejűleg történik.

Az utasítás végrehajtását felbonthatjuk három lépésre, ezek a következők:

- az **A** akkumulátor tartalmának átmásolása a belső buszon keresztül a komplementáló egységbe (a mikroprogramtár 0010 címjén elhelyezkedő mikroutasítás),
- a komplementáló egység indítása (0011-es cím mikroutasítása),
- a komplementált érték visszaírása a belső sinen keresztül az **A** akkumulátorba (0012 címen levő mikroutasítás).

Természetesen azok az utasítások, amelyek memóriához fordulnak több mikroutasítást tartalmaznak. Egy ilyen mikroalproram több hasonló, csak kis részletében eltérő részt tartalmaz, gondoljunk csak a mikroproceszor modellnél tárgyalt adatbeolvasásra, a műveleti kód, a címrészek és operandus beolvasása a processzorba megegyezett, különbség csak a rendeltetési hely meghatározásánál volt.

Ezek a megegyező részek ugyanazt az alprogramot használják.

A **COM A** utasításhoz tartozó mikroalprogram a következő alakú:

Mikroprogramtár						
Cím	**Tartalom**					
	a	**b**	**c**	**d**	**e**	**f**
0010	**00**	**111**	**0010**	**0100**	**000**	**00**
0011	**00**	**100**	**0000**	**0000**	**000**	**00**
0012	**00**	**111**	**0100**	**0010**	**000**	**00**

12.3. ábra: A **COM A** utasítás mikroalprogramja

Az egyes mezők jelentése:

- **a** – mező a vezérlő egységen belül ható mikroutasítást jelöli,
- **b** – az **ALU** aritmetikai-logikai egység belső elemeit címezi, például 100 a komplementáló egység címe, 111 azt jelenti, hogy az **ALU** nem aktiv, stb.),
- **c** – mező a célregiszter címét adja (0100 az **A** akkumulátor, 0010 a komplementáló egység, stb. címe),
- **d** – mező a forrásregiszter címét adja,
- **e** – mező információt tartalmaz a következő utasítás címéről (pl. 000, következő cím) és
- **f** – megadja azt, hogy melyik külső feltétel hat ki a következő cím meghatározására (például 01 az átvitelbit, stb.).

Látható, hogy a COM A utasítás 3 darab18 bites mikroutasításból áll.

12.3. AZ ARITMETIKAI-LOGIKAI EGYSÉG

Az **ALU** aritmetikai-logikai egység egy többfunkciós digitális kombinációs hálózat. Alkalmas alap aritmetikai és logikai műveletek elvégzésére. Általában a következő részek találhatók meg az egységben:

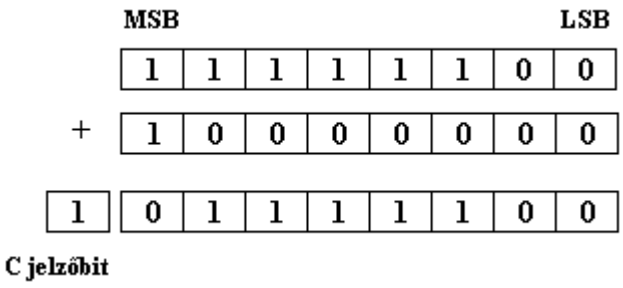
- komplementálás (minden bit ellenkező értéket vesz fel),
- léptetés (általában léptetés és körbeforgatás),
- összeadás (2 operandus, egyszerűbb processzoroknál bináris, fixpontos),
- inkrementálás (az érték növelése eggyel),
- dekrementálás (az érték csökkentése eggyel) és
- átvitel.

Előző 12.2. fejezetben adott példában a **b** mező határozza meg ezen egységek címét.

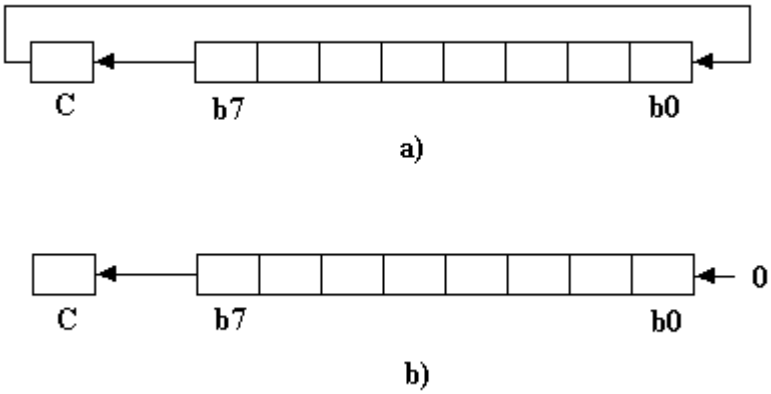
Az aritmetikai-logikai egységhez kapcsolódik több bistabil (flip-flop) amelyek a műveletvégzés eredményétől függően tárolják a kapott eredmény jellemzőit. Ezeket nevezik jelzőbiteknek is, vagy angolul a zászló, flag szóval. Összefogva a bistabilokat egy állapotjelző regiszter keletkezik. Az állapotregiszter minden egyes bitje önálló, értékük mindig a műveletvégzés befejezésekor az eredménytől függő értéket vesz fel.

A mikroprocesszoroknál, mikrovezérlőknél leggyakrabban használt jelzőbitek a következők:

- **C – átvitel (carry) jelzőbit**, kéttős szerepe van:
  - Ha az eredmény legmagasabb helyértékén átvitel keletkezik **C** értéke 1-re állítódik, erre példa a 7.27.ábrán látható. A **D7** helyen történő átvitel **D8** helyre kerülne, de ilyen nem létezik egy 8 bites regiszternél, e helyett az átvitel a **C** átvitelbitbe kerül. Ezt bizonyos utasításokkal vizsgálhatjuk, **C** értéke alapján elágazást lehet létrehozni a programban.
  - A **C** jelzőbitet használhatjuk léptetésnél és forgatásnál is, amire példát a 12.4. ábrán láthatunk.



12.4. ábra: Példa olyan összeadásra, ahol átvitel keletkezik

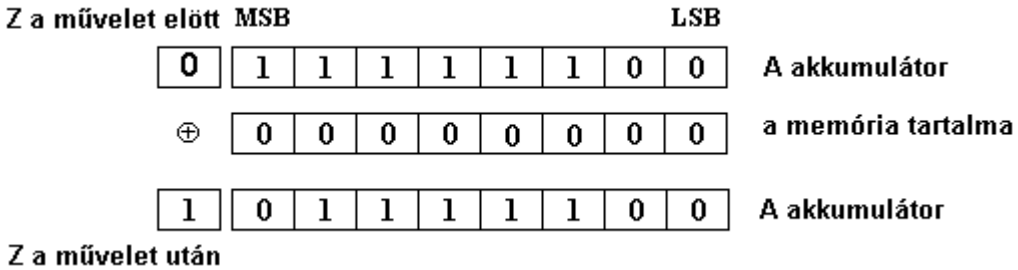


12.5. ábra: Körbeforgatás **a**) és léptetés balra **b**)

- **V** – túlcsordulás (overflow) jelzőbit, a V = 1 azt jelzi, hogy egy matematikai művelet eredménye nem értelmezhető, hibás eredmény keletkezett. Ez matematikailag a kettes komplemens előjelbit hibája: $V = C_s \oplus C_p$, ahol C_p az előjelbitbe az átvitel, vagyis b7, C_s pedig az előjelbitből való átvitel. Tulajdonképpen egy nyolcbites regiszternél a művelet eredménye kilépett a -128 , $+127$ tartományból.
- **N**- a negativ számot jelöli, csak előjeles számábrázolásnál, kettes komplemensnél van értelme, megegyezik a jelzőbit értéke megegyezik b7 értékével.
- **Z** – jelzi, hogy a regiszter tartalma 0 (Z = 1), illetve nem nulla (Z = 0). Értéke aritmetikai műveletek, valamint összehasonlító utasítások végrehajtása után állítódik, adatmozgatás után nem változik értéke a

tartalomtól függően. A 7.29. ábrán a példában, a kizáró vagy művelet elvégzése utáni állapot látható.

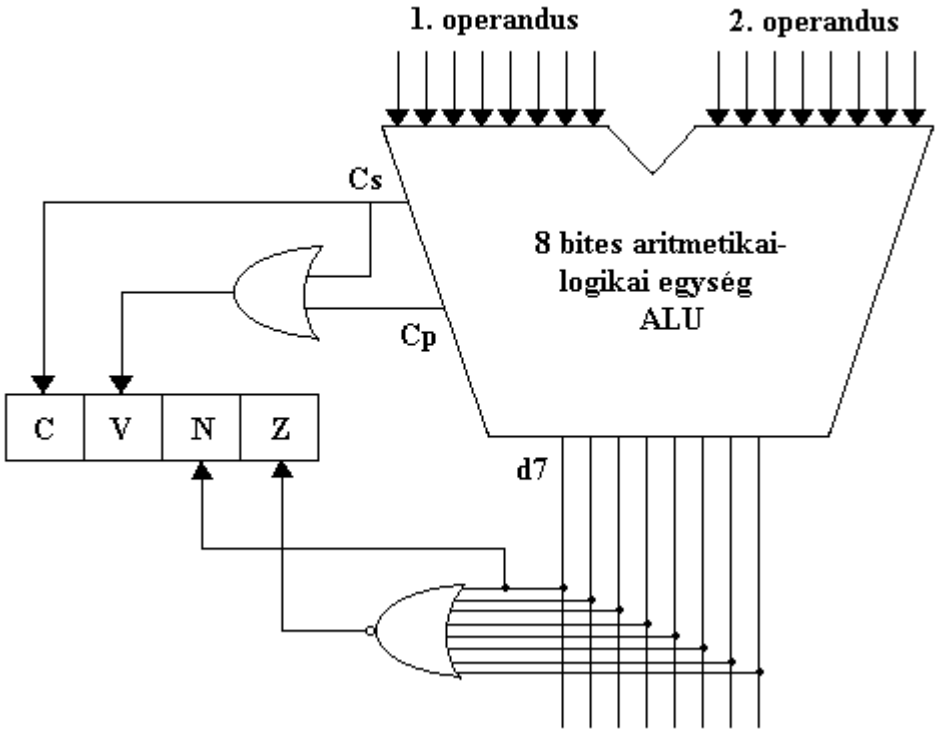
- **H** – ’félátvitel’, tulajdonképpen ugyanaz, mint a C átvitel, de 4 bit után, vagyis b3-ről b4-re. Mivel a **BCD** (Binary-Coded Decimal) számokat 4 biten lehet ábrázolni, ezért szükséges figyelni az átvitel jelentkezését. A BCD számoknál **9** után újból **0** következik (és egy átvitel), ugyanakkor a 4 bit 16 különböző számot ábrázolhat, de ebből csak a 0 és 9 közé eső bináris kombinációknak van értelme.



12.6. ábra: Mintapélda a **Z** jelzőbit beállítására (kizáró vagy)

A fent felsorolt jelzőbiteken kívül processzortipustól függően találhatunk még egyéb indikátorbiteket, ez mindig az adott géptől függ.

A 12.7. ábrán az ALU aritmetika-logikai egység és a különböző jelzőbítek közötti kapcsolat látható.



12.7. ábra: A jelzőbitek és az aritmetikai logikai egység kapcsolata

12.4. AZ A AKKUMULÁTOR ÉS AZ ÁLTALÁNOS CÉLÚ REGISZTEREK

Mint már a mikroprocesszor modellnél is látható volt, a processzor egyik regisztere, az A akkumulátor kitüntetett szerepet játszik a műveletek végrehajtásának nagy részénél. Az akkumulátor mellett a processzor tartalmaz még néhány regisztert, amelyek száma igen eltérő lehet tipustól függően, néhánytól néhány százig terjedhet számuk. A regiszterek adatelérési ideje a 10 ns nagyságrendben van, míg az operatív memóriához való hozzáférés kb. egy nagyságrenddel lassúbb. Rendszerint a több regiszter egy processzoron belüli memóriát alkot. A műveletek egy része hasonló módon használja ezeket a regisztereket, mint az akkumulátort, de korlátozottak a lehetőségek. Például a C jelzőbitet nemcsak az A akkumulátorral végzett műveletek állíthatják, hanem bizonyos általános célú regiszteren végzett művelet is módosíthatja az értékét. Az adott processzor utasításkészlete tartalmazza ezeknek a lehetőségeknek a leírását.

12.5. CÍMREGISZTEREK

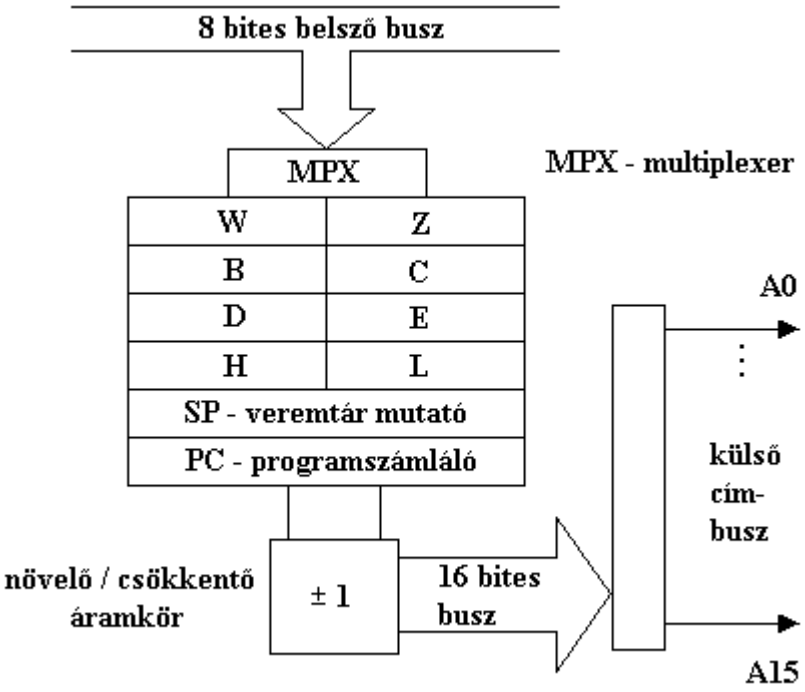
A címregiszterek mindig az operatív memória valamelyik rekeszét címezik, ez már a a mikroprocesszor modellnél is látható volt.

A következő címregiszterek találhatók meg egy processzorban:

- PC **– utasítás, vagy programszámláló,**
- DC **– adatszámláló,**
- SP **– veremtár mutató (stack pointer) és**
- I **– indexregiszter.**

Sok processzornál az a lehetőség is fennáll, hogy valamilyen általános célú regiszter címzésre is használható.

A 12.8. ábrán egy lehetséges regiszterelrendezés látható (processzorrészlet).



12.8. ábra: Címregiszterek kapcsolata a külső és belső sinekkel

A 12.8. ábrán levő részletből látszik, hogy a címregiszterekbe egyedül a mikroprocesszor belső buszán keresztül lehet címet beírni. A belső busz, mivel 8 bites, ezért a 16 bites adatok beírása két lépsben megy végbe. A W **és** Z **regiszterek csak segédregiszterek, ezek programból közvetlenül nem használhatók fel (nincs utasítás amelyben** W **és** Z **közvetlenül használható). Ennél a processzornál nincs külön** DC **adatszámláló, ennek szerepét a** HL **regiszterpár, illetve** WZ **regiszterpár veszi át.** H**, illetve** L **külön is címezhető (írható – olvasható) az utasítások segítségével, de** HL **együttesen is kezelhető mint 16 bites regiszter.**

Ellenőrző kérdések

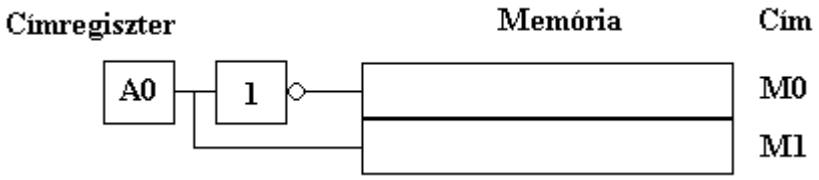
1. Milyen két fő alkotóelemből áll a processzor?
2. A processzorban és a számítógépben mely egység irányítja az adat- és információáramlást?
3. Mi a vezérlő jel?
4. Mitől függ a vezérlő jelek előállításának a sorrendje?
5. Mi határozza meg azt, hogy a tervezőmérnökök milyen processzorarchitektúrát hoznak létre?
6. Milyen formájú programokat hajt végre közvetlenül a processzor?
7. Egy gépi kódú utasítást (makroutasítás) hogyan hajt végre a processzor?
8. Sorolja fel a CISC gép fő elemeit .
9. Mi a vezérlő egység és mi a feladata?
10. Az utasításvégrehajtásnak melyek a fázisai?
11. Mi az előkészítési, vagy utasításkihozatali fázis?
12. Mi az utasításvégrehajtási fázis?
13. Milyen megoldást alkalmaznak a vezérlő egység megvalósításánál?
14. Mi a huzalozott logika a vezlrlőegység megvalósításánál?
15. Mi a mikroprogram a vezérlőegység létrehozásánál?
16. Hol helyezkedik el a mikroprogram?
17. A felhasználó megváltoztathatja-e a mikroprogramot, ha igen hogyan?
18. Milyen vezérlőjeleket ismer?
19. Mi a belső vezérlőjelek feladata?
20. Mi a külső vezérlőjelek feladata?
21. Rajzolja fel a mikroprogramozott vezérlő egység blokkvázlatát és ismertesse működését.
22. Mi az IR utasításregiszter?
23. Mi az összefüggés a mikroprogram és a gép szóhosszúsága között?
24. Hány mikroutasítás tartozik egy gépi kódú utasításhoz?
25. Rajzolja le a gépi kódú utasítás és mikroprogram közötti kapcsolatot a gép hardver elemeinek segítségével és magyarázza el egy utasítás végrehajtását .
26. Példán keresztül magyarázza el a gépi kódú utasítás végrehajtását a mikroprogram segítségével .
27. Mi az aritmetikai-logikai egység (ALU) és mi a feladata?
28. Mely állapotjelző bitek a leggyakrabban használatosak az aritmetikai-logikai egység műveletvégzésekor?
29. Mi az átvitelbit és adjon példát működésére .
30. Mi a zérobit és adjon példát működésére .
31. Mi a túlcsordulásbit és adjon példát működésére .
32. Mi a félátvitel-bit és adjon példát működésére .
33. Rajzolja le az aritmetikai logikai egységet a jelzőbitekkel együtt .
34. Mi az akkumulátor és mi a feladata?
35. Mik az általános célú regiszterek és mi a feladatuk?
36. Melyek a címregiszterek és mi a feladatuk?
37. Mi a programszámláló és mi a feladata?
38. Mi az adatszámláló és mi a feladata?
39. Mi a veremtár-mutató (stack pointer) és mi a feladata?
40. Mi az inexregiszter és mi a feladata?

41. Adjon példát az általános célú regiszterek elrendezésére .

13. MEMÓRIAKEZELÉS

13.1. A MEMÓRIA KIVÁLASZTÓ LOGIKA (TELJES CÍMDEKÓDOLÁS)

A **Neumann** tipusú mikroszámítógépnél egy közös memória tartalmazza a programot, az adatokat, változókat és ugyanennek a közös memóriának lehet a része a stack memória is. A **Harvard** tipusú számítógépnél már külön memóriában található a program, más memóriában az adatok. A memórában levő rekeszeket, amelyek több bit szóhosszúságúak (pl. 8, 16 vagy 32) címezni kell. Ha 1 bit szolgál a címzésre, akkor azzal 2 különböző rekesz (vagy más elem) különböztethető meg (13.1. ábra). Az így létrehozott címkiválasztó-logika igazságtáblázata a 13.1. táblázatban található.

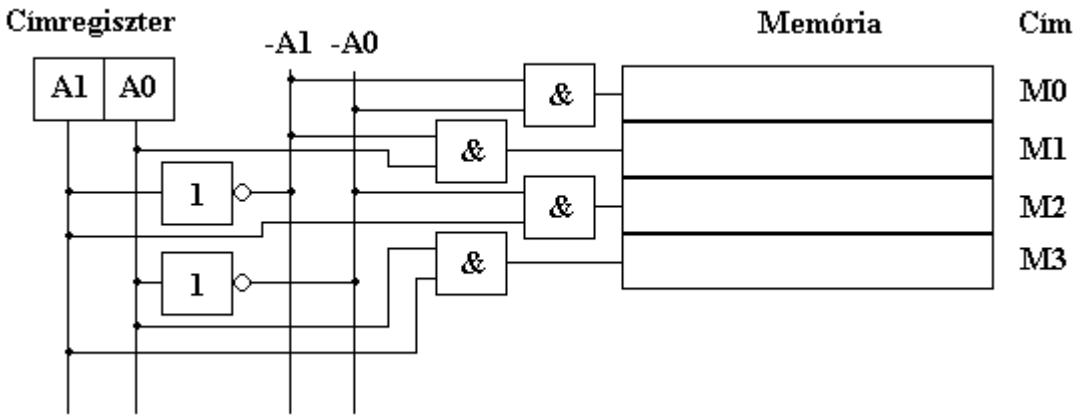


13.1 ábra: Egy bit szóhosszúságú címregiszter és cimzési tartománya

A0	M1	M0
0	0	1
1	1	0

13.1. táblázat: Egy bites cím igazságtáblázata

2 bit szóhosszúságú címregiszter már 4 különbözö rekeszcímet állithat elő (2^2), ennek logikai rajza a 13.2 ábrán látható és igazságtáblázata a 13.2 táblázatban látható.



9.2 ábra: Két bit szóhosszúságú címregiszter és cimzési tartománya

13.2 Táblázat: 2 bites cím igazságtáblázata

A1	A0	M3	M2	M1	M0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Ezek után látható, hogy bármilyen szóhosszúságú címregiszterhez megtervezhető a **címkiválasztó logika**, a címzett terület nagysága pedig a következő képlettel számítható ki:

$$\textbf{összes cím} = 2^{n} \qquad (13.1)$$

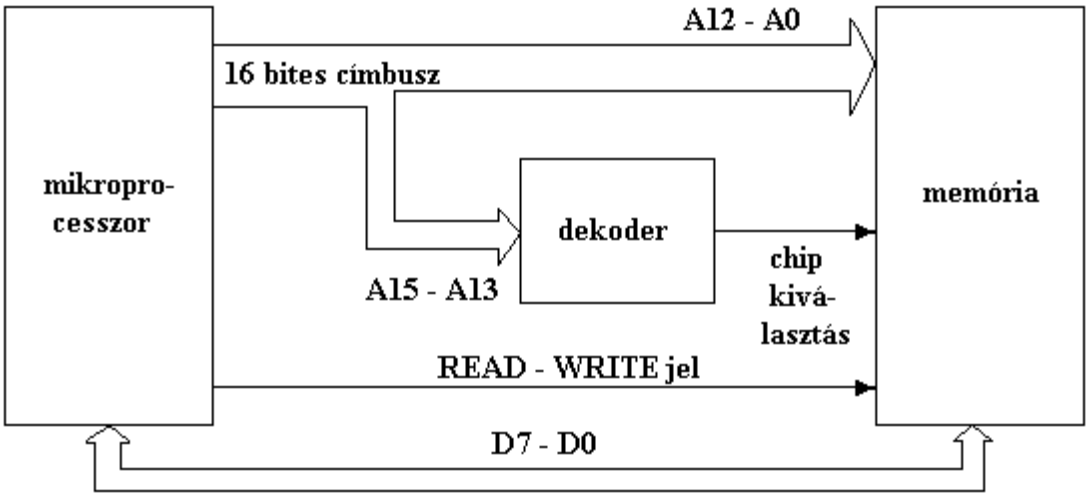
ahol:

- **összes cím** -- a címezhető memóriarekeszek száma,
- **n** -- a címregiszter szóhosszúsága bitekben.

Ha a mikroprocesszor 16 bites cimzéssel rendelkezik, akkor a (13.1) képlet szerint 65536 különböző elem (rekesz) különböztethető meg a rendszeren belül, ami a 16-os (hexadecimális) számrendszerben **FFFFh.** A szám utáni **h** betű utalás a hexadecimális számrendszerre. Ez az érték még **64 kB** (64 kilobájt) alakban is megadható. A 65536 és a 64 kB közötti eltérés oka abban van, hogy a 2^{10} értéke 1024 a tizes számrendszerben és nem 1000.

Rövidebb program, vagy kevesebb adat esetén nem szükséges a mikrogép ill. a mikrovezérlő teljes címezhető területét kiépiteni. Ugyanakkor egy memóriaterület felépítésekor különböző kapacitású memóriaelemek között válogathatunk. Ezek a kapacitások különböző értékűek, de mindnél közös az, hogy 2 hatványaként fejezhetőek ki, igy pl. lehet **1 kB, 2 kB, 4 kB, 8 kB, 16 kB, 32 kB** és **64 kB** nagyságú. Az 1 kB-nál kisebb kapacitásnak ma már nincs gyakorlati jelentősége. Ezek a memóriák egymásnak kétszeresei. Kialakításuk olyan, hogy a címdekódolást elvégzik, így adott számú címvezetékkel rendelkeznek. Ez pl. a 8 kB-os memória esetén 2^{13} = 8096, vigyis 13 címvezetékkel (A12-től A0-ig) rendelkezik a memória. A 64 kB nagyságú memóriaterületre így 64 kB / 8 kB = 8 db. ilyen IC köthető be.

Ha maradunk a 8 kB-os memóriánál, akkor a 13.3. ábrán megérthetjük a memóriacímzés és memóriakiválasztás lényegét, a 13.4. ábrán pedig a címek szerepét találhatjuk.



13.3. ábra: Példa a memóriakiválasztásra, 8 kB memória esetén

A memóriakapacitás növelésének érdekében több 8 kB-os memóriát is rákötheténk a címsinre. A mikroprocesszor adatsinjére párhuzamosan rácsatlakozik az összes memória adatvezetéke, ugyanígy a címsin A12 - A0 vezetékeire minden 8 kB-os memória 13 címvezetéke. A két, vagy több memória megkülönböztetése ekkor a 13.4. ábra 16 bites címregiszterének 'címdekódolóra' nevű mezője szolgál. Ez a példában **3 bit**, ami 2^3 = 8 elem megkülönböztetését teszi lehetővé (8 x 8 kB = 64 kB). A 8 jel közül kell kiválasztani azt a vezetéket, amelyik a memória **CS** (chip select) vezetékén keresztül engedélyezi, vagy tiltja az adott memóriát. Egy ilyen csipkiválasztó (dekódoló) logika igazságtáblázata látható a 13.3. táblázatban.

MSB LSB

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
címdekódolóra			**8 kB közvetlen címzése**												

13.4. ábra: 8 kB címzése és kiválasztása 16 bites címbusz esetén

13.3. táblázat: 8 darab 8 kB-os memóriacsip dekódolása

csip	cím			memória							
	A15	A14	A13	M7	M6	M5	M4	M3	M2	M1	M0
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	1	0	0
3	0	1	1	0	0	0	0	1	0	0	0
4	1	0	0	0	0	0	1	0	0	0	0
5	1	0	1	0	0	1	0	0	0	0	0
6	1	1	0	0	1	0	0	0	0	0	0
7	1	1	1	1	0	0	0	0	0	0	0

Ekkor a 13.5. ábra szerinti memóriaképet kapjuk. Itt a baloldali oszlop az egyes memóriák kezdő- és utolsó címét jelöli, mig a jobb oldali oszlop az egyes memóriák sorszámát.

cím	memória sorszám
0000	0.
1FFF	
2000	1.
3FFF	
4000	2.
5FFF	
6000	3.
7FFF	
8000	4.
9FFF	
A000	5.
BFFF	
C000	6.
DFFF	
E000	7.
FFFF	

13.5. ábra: 64 kB memóriaterület kitöltése 8 db. 8 kB kapacitású memóriával

A 13.6. ábra a 0. 8 kB, a 13.7. ábra az 1. 8 kB, mig a 13.8. ábra a 6. 8 kB memória címeit mutatja, ahol **x** értéke **0** vagy **1** lehet, így pl. a 0. memória **0000** kezdõ- és **1FFF** utolsó címe között helyezkedik el pl. **0B5A**, de a **2B5A** az 1. 8 kB memória címe, míg a **CB5A** már a 6. 8 kB nemóriában van.

MSB LSB

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x
címdekódolóra			közvetlenül 8 kB.-os memóriára (0. memória)												

13.6. ábra: A 0. 8 kB címei

MSB LSB

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	1	x	x	x	x	x	x	x	x	x	x	x	x	x
címdekódolóra			közvetlenül 8 kB.-os memóriára (1. memória)												

13.7. ábra: Az 1. 8 kB címei

MSB LSB

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x
címdekódolóra			**közvetlenül 8 kB.-os memóriára (6. memória)**												

13.8. ábra: A 6. 8 kB címei

A memóriák kiválasztásához szükséges hardver megtervezése a 13.3. táblázat alapján történik. **M0** vezetéket a 13.6. ábrán látható memória **CS** bemenetéhez kell csatlakoztatni. Az itt megjelenõ **1** jel engedélyezi a memória működését, a **0** pedig teljesen leválasztja azt a mikroprocesszorról. Matematikailag ez a következő:

$$\mathbf{M0 = -A15 \ \& \ -A14 \ \& \ -A13} \qquad (13.2)$$

Hasonlóan határozható meg az összes memóriakiválasztó jel, pl. **M1** és **M6** is :

$$\mathbf{M1 = -A15 \ \& \ -A14 \ \& \ A13} \qquad (13.3)$$

$$\mathbf{M6 = A15 \ \& \ A14 \ \& \ -A13} \qquad (13.4)$$

Az **A12 - A0** cimek eljutnak ugyan mindegyik memóriára, de csak a **CS** vezetéken keresztül engedélyezett memóriában címeznek egy rekeszt.

Különböző kapacitású memóriák is használhatók ugyanazon a memóriatartományon belül, de figyelembe kell venni néhány szabályt a memóriák elhelyezésére.

Legyen a feledat az, hogy a **8000** címen elhelyezkedő **4 kB** memória után közvetlenül 2 db. **2 kB** kapacitású memória kerüljön. A memóriakiosztás a 13.9. ábrán látható.

Természetesen a szoftver ezt a területet egybefüggően 'látja', vagyis címezi.

cím	memória kapacitás
0000	üres terület
7FFF	
8000	4 kB
8FFF	
9000	2 kB - 1. memória
97FF	
9800	2 kB - 2. memória
9FFF	
A000	üres terület
FFFF	

13.9. ábra: A 8000 címen elhelyezkedõ 8 kB memória megvalósítása 1 db **4 kB** és 2 db **2 kB**-os memóriával

A 13.10. ábrán a 4 kB, a 13.11. és 13.12. ábrákon pedig a 2-2 kB címzése látható, amely címek meghatározása a 13.5 képlettel számítható ki:

$$2^m = \frac{K}{k} \qquad (13.5)$$

ahol:

- **m** - a dekódoló bitjeinek száma,
- **K** - a címezhetõ terület teljes nagysága és
- **k** - a bekötendő memória kapacitása.

A 4 kB memória esetén 64 kB / 4 kB = 16 adódik, amibõl m = 4 lesz, illetve a 2 kB esetén 64 kB / 2 kB = 32, amibõl m = 5.

MSB LSB

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x
címdekódolóra				közvetlenül a 4 kB-os memóriára											

13.10. ábra: A címregiszter felosztása közvetlen memóriacímekre és címdekódoló vezetékekre, 4 kB

MSB LSB

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	0	0	1	0	x	x	x	x	x	x	x	x	x	x	x
címdekódolóra					**közvetlenül a 2 kB-os memóriára**										

13.11. ábra: A címregiszter felosztása közvetlen memóriacímekre és címdekódoló vezetékekre, 2 kB-os memória, első blokk

MSB LSB

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	0	0	1	1	x	x	x	x	x	x	x	x	x	x	x
címdekódolóra					**közvetlenül a 2 kB-os memóriára**										

13.12. ábra: A címregiszter felosztása közvetlen memóriacímekre és címdekódoló vezetékekre, **2 kB** második blokk

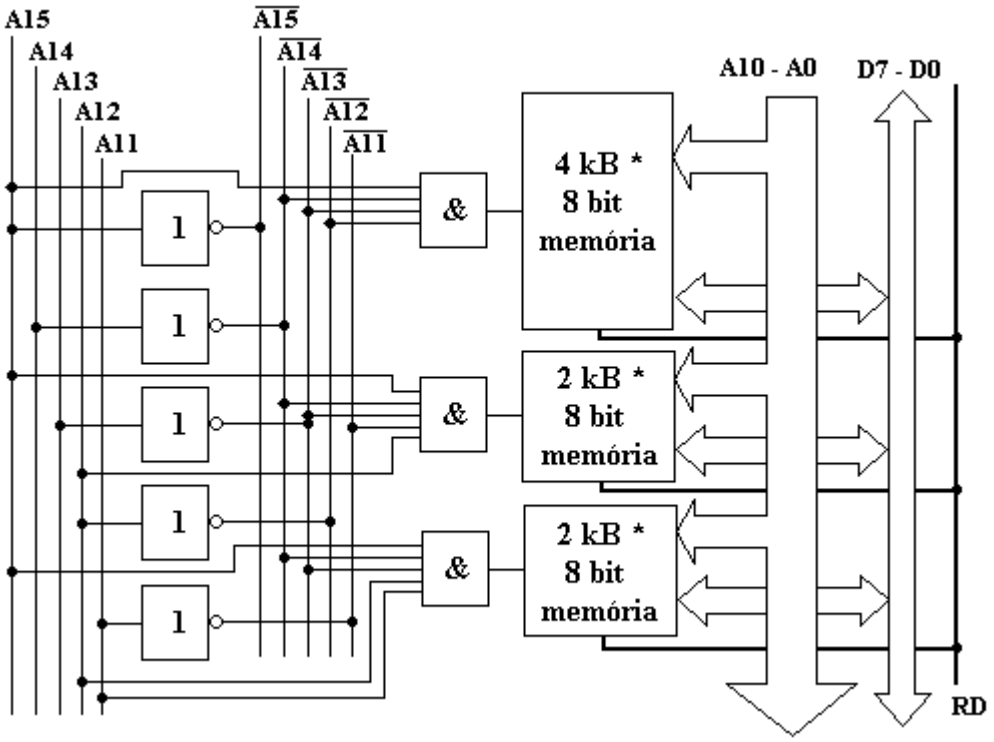
Látható, hogy a 2 kB-os memóriáknak 11, a 4 kB-osnak 12 közvetlen címvezetéke van.

Ha a 2 db. 2 kB-os memóriát egybefogjuk, akkor 4kB címzése válik lehetõvé, tehát a feladat most a 2 memória szelektálása, ami az **A11** bit segítségével valósítható meg.

Ezek után a címdekódolók matematikai egyenletei a következõk:

M-4 kB = A15 & -A14 & -A13 & -A12 (13.6)
M-2 kB-1 = A15 & -A14 & -A13 & A12 & -A11 (13.7)
M-2 kB-2 = A15 & -A14 & -A13 & A12 & A11 (13.8)

A kapcsolás a 13.13. ábrán látható.



13.13. ábra: 4 kB + 2 kB + 2 kB memória bekötése 8000 címtõl

Megoldható a feladat máshogyan is, itt csak egy megoldás látható.

Legyen az elõzõhöz hasonló feladat, vagyis most is tartalmazzon a rendszer 8000 címtől egy 4 kB-os és 2 2 kB-os memóriát, de most más sorrendben, 2 kB után következzen a 4 kB és ezután ismét 2 kB. A 2 kB után elhelyezett 4 kB memóriánál átfedés keletkezik a címzésnél, ugyanis egyrészt az **A11** címvezeték szerepel a memória közvetlen címzésénl is, és a dekódoló bemeneteként is.

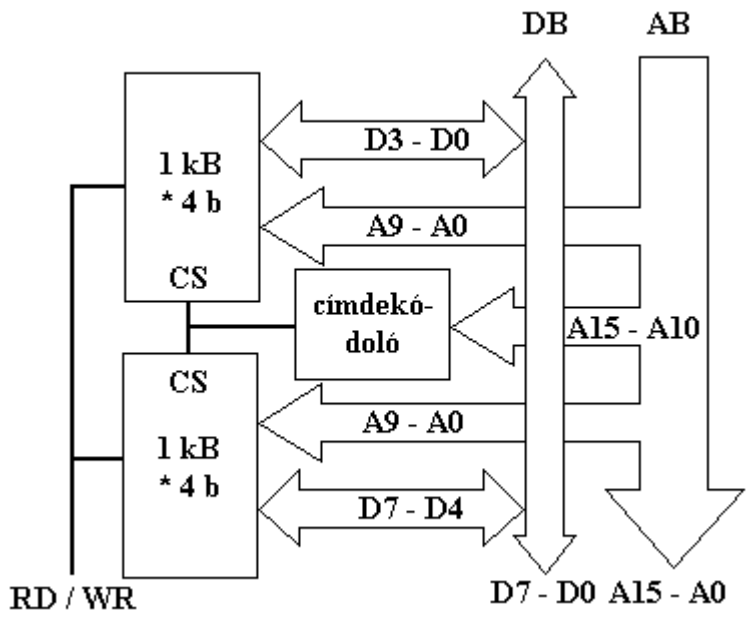
A szabály a különbözõ kapacitású memóriák egy egységes memóriaterületen való elhelyezére az, hogy először a legnagyobb kapacitású memóriával kell meghatározni a lehetséges kezdőcímeket. Az ettől kétszer kisebb kapacitású memóriák vagy ugyanezen a címeken kezdődnek, vagy ezen címek felénél. Értelemszerűen ezzel a logikával továbbmenve bármilyen kapacitás elhelyezhető a címterületen.

A fenti megfontolásokból következik, hogy 2 kB után csak 2 kB kihagyása után lehet beépíteni a rendszerbe a 4 kB, majd ezután folyamatosan a 2 kB. memóriát. Ezek után elvégezve a tervezést megkapjuk a 13.14. ábra szerinti memóriaképet, ahol látható, hogy az első 2 kB után 2 kB üres terület következik, amit a 4 kB memória, majd ezt 2kB folytat.

cím	memória kapacitás
0000	üres terület
7FFF	
8000	2 kB - 1. memória
87FF	
8800	2 kB üres terület
97FF	
9800	4 kB
9FFF	
A000	2 kB - 2. memória
A7FF	
B000	üres terület
FFFF	

13.14. ábra: A **8000** címen elhelyezkedõ **8 kB** memória megvalósítása 1 db **4 kB** és 2 db **2 kB**-os memóriával

A fenti példákban mindig **1 Byte** szóhosszúságú memória szerepelt. Az ettõl eltérõ szóhosszúságú memóriák bekötésére a 13.25. ábrán látható egy példa. Maga a dekódolás megegyezik az elõzõekben ismertetett megoldásokkal, csak most a megfelelõ szóhosszúság létrehozása a feladat. A példában **4 bit** szóhosszúságú memóriákból alakítjuk a szükséges 1 B szélességet azok párhuzamos kötésével. Itt mindkét memória ugyanazon címekre válaszol, de az adatsin más és más vezetéke kapcsolódik az egyes bitekhez.



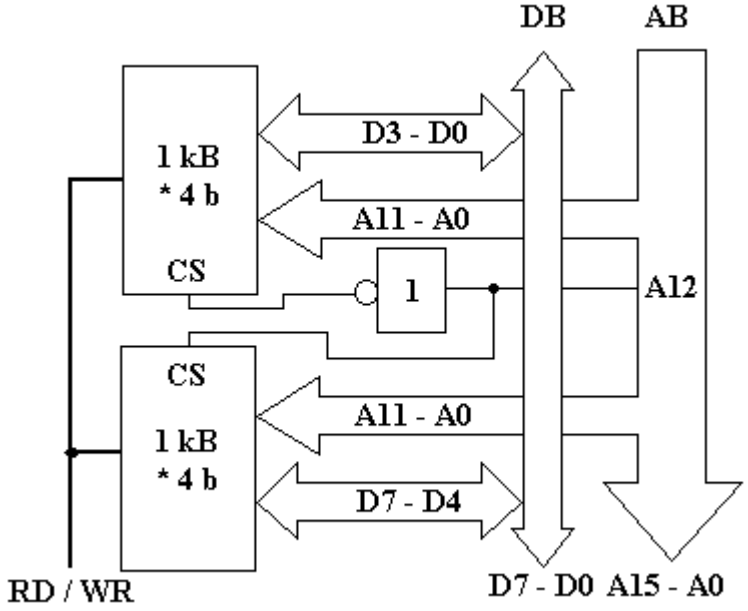
13.25. ábra: 2 db. 1 kB x 4 bit memória felhasználása 1 kB x 8 bit memóriaterület kiépítéséhez

13.2. A MEMÓRIA KIVÁLASZTÓ LOGIKA (NEM TELJES CÍMDEKÓDOLÁS)

A 13.1. fejezetben az ún. teljes címválasztó-kód meghatározása volt látható. Ilyenkor minden cím egy adott fizikai elem (memóriarekesz) működését teszi lehetõvé, ami a teljes címzés felhasználását jelenti .

Egyes esetekben, ha eleve a rendszer nem igényli a teljes címezhetõ terület felhasználását, akkor használható az ún. nem teljes címválasztó-kód létrehozása. Ez csökkenti a hardver elemek számát.

Tartalmazzon egy rendszer 2 db. 4 kB nagyságú memóriát. Mint látható volt, a 4 kB címzése 12 vezetéket, vagyis **A11 - A0** sineket használja. Ugyanígy a másik 4 kB is ugyanezen vezetékeken keresztül címezi az egyes rekeszeket. Az **A12** bit segitségével szétválaszthatjuk a két területet (13.26. ábra).



13.26. ábra: A nem teljes dekódolás 2 db. **4 kB**-os memória esetében.

A **0C2D** és **1C2D** címek között ez a kötés különbséget tesz, az elsõ cím a felsõ, míg a második az alsó memóriában választ ki rekeszt, de pl. a **2F59** és **CF59** címeket ez a címválasztás már nem különbözteti meg. Mivel itt a címválasztás nem használja fel az **A15, A14** és **A13** címvezetékeket az ezeken fellépõ bármilyen érték hatástalan. Például a memória **100111010010 (9D2)** címen levő cellája aktiv lesz a következő címeknél is: **09D2, 29D2, 49D2, 69D2, 89D2, A9D2, C9D2** és **E9D2.** A 3 nem használt legnagyobb helyértékû címvezeték 2^3 = 8 címet ad.

13.3. A VEREMTÁR ÉS VEREMTÁR-MUTATÓ (STACK ÉS STACK POINTER)

A magasszintű programozási nyelveknél megismert alprogramok, eljárások visszatérési címének tárolását biztosító memóriát nevezik **stack memóriának**. A gépi szintű programozás is használ alprogramokat, ezért szükséges ennek a memóriatipusnak a tárgyalása. Fontos még az alprogramoknál, megszakításoknál a megszakított program adatainak, paramétereinek ideiglenes megőrzése is, ez lehetséges a stack memóriában is.

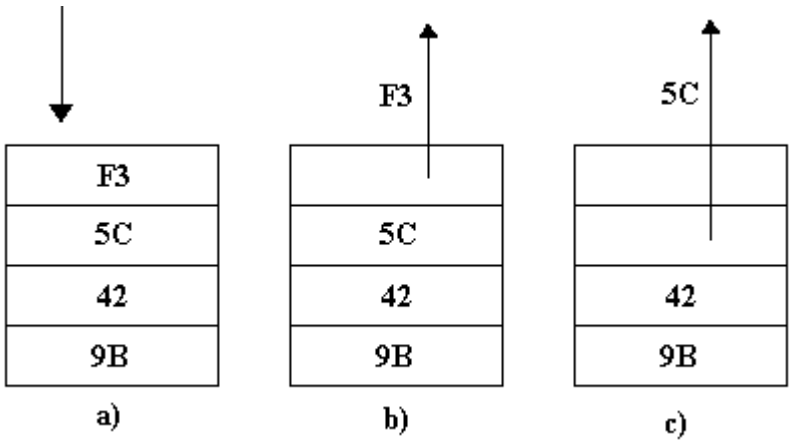
Két féle megoldás terjedt el a gyakorlatban:

- programozott (szoftver) stack memória és a
- hardver stack memória.

A programozott stack memória gyakorlatilag a rendszer **RAM** memóriájának egy részét foglalja el, míg a hardver megoldásnál a mikroprocesszoron belül egy bizonyos számú regiszter alkotja ezt a memóriát.

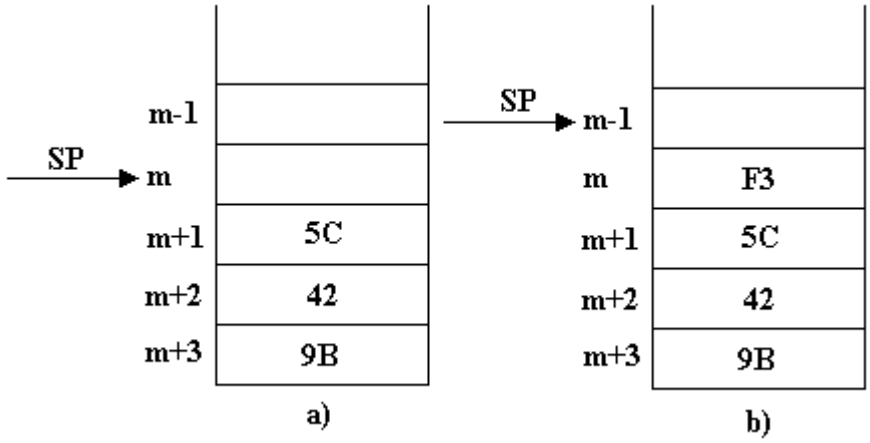
A programozott stack viszonylag nagy kapacitású memória, míg a hardver stack nagy sebességű hozzáférést biztosít. A kisteljesítményű mikrovezérlőknél a hardver megoldást alkalmazzák, míg például adatfeldolgozásnál a szoftver megoldást (a PC gépeknél is).

Általában a mikroprocesszorok rendelkeznek a stack memóriába adatot beíró utasítással, amit rendszerint **PUSH** utasításnak hívnak, valamint adatot a memóriából kiolvasó utasítással, ami **POP** (néha **PULL**). Ez a memória **LIFO** (**Last In First Out**) struktúrájú, ami azt jelenti, hogy a legutoljára beírt adat kiolvasása történik meg legelőször, a következő kiolvasott adat még korábban került a memóriába. A 13.27. ábrán látható, hogy a **PUSH** utasítással a veremtár csúcsára beírt (**a**) ábra) adat a **POP** utasítással (**b**) ábra) olvasható ki, szintén a veremtár csúcsáról. A következő **POP** a régebben beírt adatot távolítja el (**c**) ábra).



13.27. ábra: Példa a **PUSH** (a ábra) és a **POP** (b és c ábra) használatára

A 13.28. ábra szemlélteti a stack memória és a stack pointer (veremtár mutató) közötti kapcsolatot. Ha programozott veremtár használatáról van szó, akkor a veremtár mutató szélessége rendszerint a 8 bites gépeknél 16 bit, hardver megoldásnál értelemszerűen kisebb szélességű. A 13.28. ábrán programozott megoldás látható. A veremtár mutató 'sétál' a memória rekeszein, minden stack memóriába való íráskor értéke csökken, adatkivételnél értéke növekszik. Létezik fordított megoldás is, vagyis adatbeírás esetén a veremtár mutató tartalama inkrementálódik, kiolvasáskor dekrementálódik.



13.28. ábra: A veremtár mutató (**SP**) és a veremtár viszonya a **PUSH** utasítás elött (a ábra) és után (b ábra)

Ellenőrző kérdések

1. Ismertesse a Neumann tipusú számítógép felépítését és működési elvét .
2. Ismertesse a Harvard tipusú számítógép felépítését és működési elvét .
3. Miért szükséges a memória minden egyes elemének külön címzése ?
4. Egy adott szóhosszúságú címregiszter hány memóriarekesz (illetve más elem) címzését teszi lehetővé ?
5. Milyen kapacitású memóriák építhetők be egy mikroszámítógép operativ tárjába ?
6. Mi a címdekódoló logika és mi a szerepe ?
7. Mi a teljes címdekódolás ?
8. Különböző kapacitású memóriák felhasználásakor milyen szabály szerint lehet a címdekódolót felépíteni ?
9. Adjon példát egy címdekódoló meghatározására .
10. Mi a nem teljes címdekódolás ?
11. Hasonlítsa össze a teljes és nem teljes címdekódolást, mik az előnyök és mik a hátrányok a kétféle megoldásnál ?
12. Adjon példát egy nem teljes címdekódolóra .
13. Milyen stack memória (veremtár memória) megoldásokat ismer ?
14. Mire szolgál a stack memória ?
15. Mi a programozott stack memória jellegzetessége, előnye és hátránya ?
16. Mi a hardver stack memória jellegzetessége, előnye és hátránya ?

14. PERIFÉRIAKEZELÉS

14.1. KIMENETI ÉS BEMENETI INTERFÉSZEK, PERIFÉRIÁLIS EGYSÉGEK

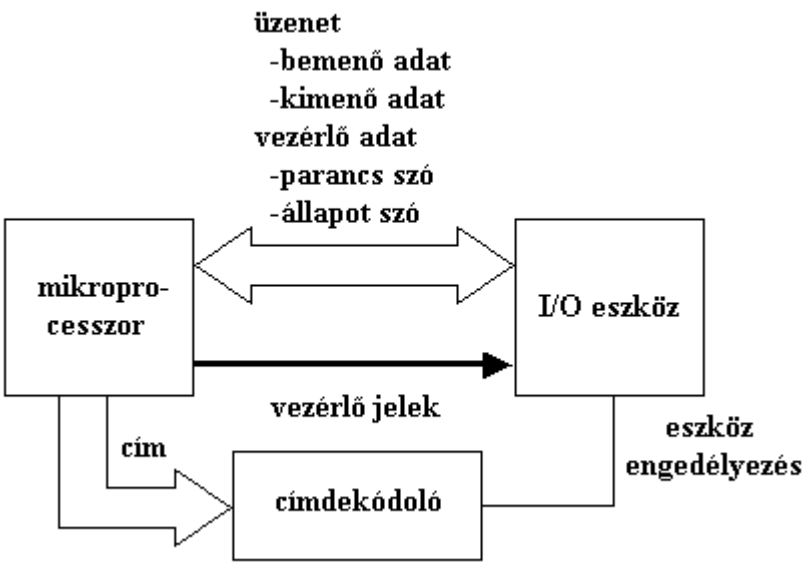
A mikrogép alapfeladata a bemeneti jelek beolvasása, feldolgozása és a kapott eredmények, vezérlőjelek visszajuttatása a folyamatba, emberhez, illetve másik géphez, gépekhez. Ebben a folyamatban fontos szerepet játszik a mikrogép és perifériális egységek közötti adatáramlás. Ilyen peifériális berendezés a billentyűzet, képernyő, egér, jelfogó, léptetőmotor, vagy az analóg jelek beolvasását lehetővé tevő A/D átalakító, stb.

Az I/O perifériális eszközök LSI technológiában megvalósított eszközök, amelyek rendszerint programozhatóak. Függetlenül attól, hogy mint önálló chipek képezik egy mikrogép építőelemét, vagy szerves részét alkotják egy mikrovezérlőnek, működési elvük, felépítésük ugyanaz. Esetleg különbség csak ott van közöttük, hogy addig míg a mikrogépnél a tervezőmérnök hozzárendeli egy címhez az eszközt, addig ez a mikrovezérlőn belül már adott, ezen változtatni nem lehet.

A mikrogéphez csatolt I/O perifériális eszköz szerves részét képezi a rendszernek, feltétlenül szükséges használatakor is biztosítani a mikrogép, rendszer összehangolt működését. Ezt teszi lehetővé az **illesztő áramkör**, az **interfész** (interface). A mikrogépekhez az I/O eszközt ugyanolyan logika szerint kapcsolhatjuk hozzá, mint a memóriáknál látott teljes, illetve nem teljes címdekódolás (címkiválasztás) segítségével.

Ezen eszközöknél a programozhatóság azt jelenti, hogy a rendszer indulásakor, vagy működése közben a programozó az állapotszó meghatározásával egy adott üzemmódot állít be, ami szerint működik az eszköz a következő állapotszó változtatásig. Ilyen működési mód például az, hogy az eszköz bemenet vagy kimenet, kérhet-e megszakítást a mikrogéptől stb.

A 14.1. ábrán látható blokkvázlat bemutatja, hogy elvileg milyen jel, illetve adatáramlást kell biztosítani a mikroprocesszor és az I/O eszköz között.



14.1. ábra: Vezérlőjel és adatkapcsolat a mikroprocesszor és az I/O eszköz között

Az ábrán látható, hogy a mikroprocesszor és az I/O eszköz közötti kétirányú buszon (DB adatsin) keresztül többféle adat áramolhat, így a **kimenő** és **bemenő** adatok mellett az I/O eszköz üzemmódját meghatározó **parancsszó**, illetve az eszköz állapotára utaló **állapotszó** is.

A számítógép-elemeket gyártó cégek sokfajta chipet gyártanak az I/O feladatok elvégzésére. Ezeket nemcsak a mikrogépekhez köthetjük, hanem mikrovezérlőkhöz is, ha nincs elegendő kimenet, vagy bemenet, illetve analóg jeleket szeretnénk feldolgozni.

Az I/O eszközöket három fő csoportba sorolhatjuk, azzal a kikötéssel, hogy ezek működési elve között átfedés lehet:

- párhuzamos I/O eszközök,
- soros I/O eszközök és
- különleges feladatot ellátó I/O eszközök.

Mint már az előző fejezetekben láttuk, a CISC struktúrájú gépekben az adatok és programok ábrázolása párhuzamosan történik 8 biten, illetve ennek egészszámú többszörösén. Amikor két eszköz között egy így ábrázolt adat átvitele válik szükségessé, kézenfekvő a szóhosszúságnak megfelelő bitszámú párhuzamos adatátvitelt, csatornát alkalmazni. Ekkor valóban az adatátvitel sebessége nagy, de például egy 32 bit szóhosszúságú információcsere 32 párhuzamos vezetéket igényel, és még egy közös földvezetéket, nem beszélve arról, hogy ilyenkor a zavarszűrés érdekében minden egyes vezetékkel párhuzamosan összesodorva kell vezetni a közös, földvezetéket, ami így már legalább 64 eret jelent. Ezt a módszert rövid távolságok áthidalására lehet alkalmazni, általában 8 bit szélességben.

Nagyobb távolságokra való adatátvitelnél rendszerint soros átviteli csatornát használunk. Ennek egyik oka az, hogy ilyenkor mindössze egy érpárra van szükség az adatátvitelhez, ami lényegesen egyszerűsíti a kábel kivitelezését, de a soros adatátvitel a már meglevő rendszereknél is használható. Ilyen például a telefonvonal. Ekkor már

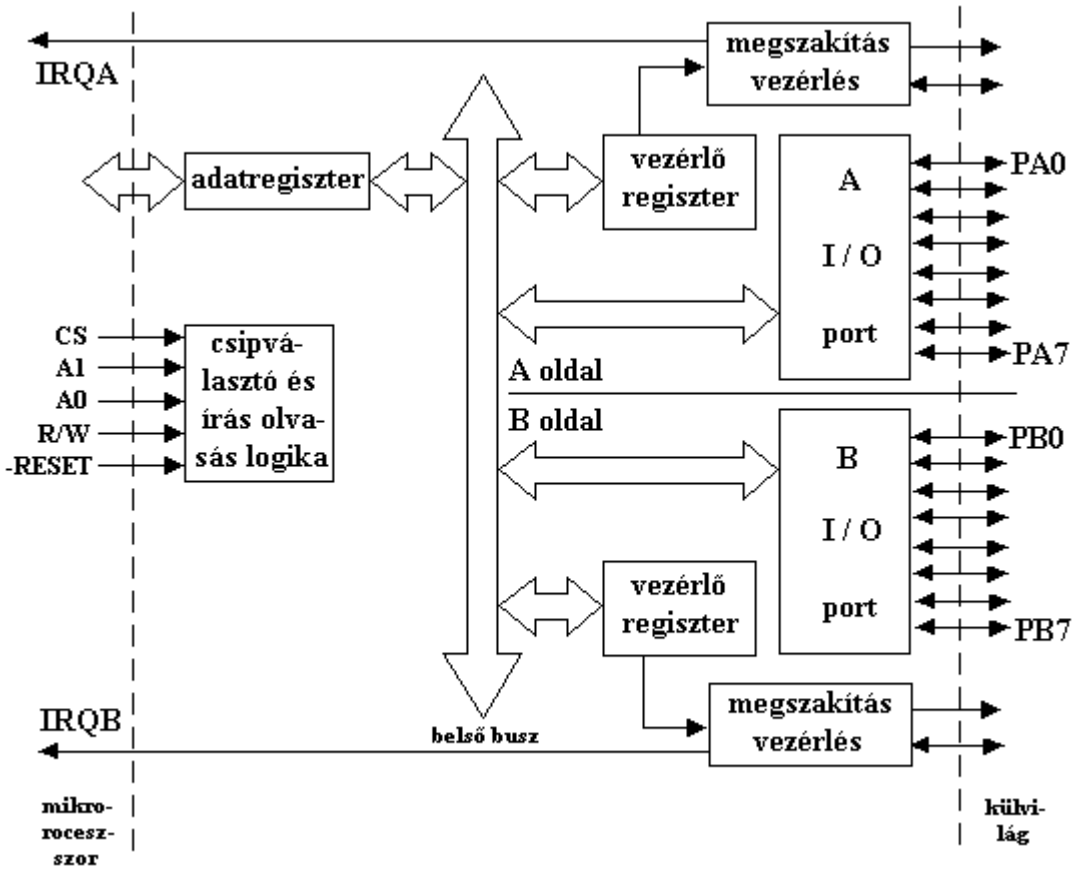
lényegesen lecsökken az adatátviteli sebesség, hiszen minden egyes bitet külön-külön kell továbbítani, valamint az adó és vevő között szinkronizációt, együttfutást kell biztosítani.

A soros és párhuzamos adatátviteli elv kombinálható is, például 16 bites adat átvitele megoldható négy lépésben egy négy bit szóhosszúságú adtátviteli csatornával.
A különleges I/O eszközök használata célszerű gyakran használt, bonyolult adatátvitelnél, ahol nagy mennyiségű adat mozgatását kell biztosítani gyorsan, például háttértárak (hajlékonylemez-meghajtó, Winchester stb.).

14.2. PÁRHUZAMOS I/O VEZÉRLŐ-ILLESZTŐ ESZKÖZÖK

Ezek a párhuzamos I/O vezérlő-illesztő eszközök rendszerint bájt szervezésű adatátvitel lebonyolítását teszik lehetővé a mikrogép és a külső perifériális eszközök között. Két, vagy több 8 bites **port**tal (kapuval) rendelkeznek, tartalmaznak vezérlő regisztereket és vezérlő logikát. Programozhatóságuk, üzemmódbeállíthatóságuk és felépítésük miatt aránylag kevés hardver hozzáadásával építhető hatásos mikrogép.

A 14.2 ábrán egy programozható I/O párhuzamos interfészt mutat.



14.2. ábra: Párhuzamos I/O interfész blokk-sémája

A két vezérlőregiszter (A port és B port) biztosítja az áramkör programozhatóságát. Az ide beírt vezér-szó határozza meg az adatátvitel paramétereit, megszakítás-

engedélyezést, stb. Ugyanezen információ határozza meg az I/O vonalak adatáramlási irányát, tehát azt, hogy bemenetkét, vagy kimenetként használja a program ezeket a csatlakozópontokat. Egyes I/O eszközöknél a kimenetek és bemenetek csak csoportonként határozhatók meg, de léteznek olyan párhuzamos elemek is, ahol minden egyes csatlakozási pont tetszőlegesen használható kimenetnek, vagy bemenetnek.

A mikroprocesszor és az eszköz belső sinje között található 'adatregiszter' úgynevezett háromállású regiszter, ami azt jelenti, hogy a CS csipválasztó jel megléte engedélyezi a mikroprocesszor adatsinjének és az eszköz belső adatsinjének összekapcsolódását, illetve a jel hiánya (0) teljesen szétkapcsolja a két sint. A CS jel akkor 1-es értékű, ha az eszköz címe (dekódoló áramkör) megegyezik a szoftver által kiküldött címmel.

A mikroprocesszor és az eszköz között tehát egyrészt az adatok áramlásánál az adatregiszter, másrészt a címeknél és a vezérlőjeleknél a 'csipválasztó és írás-olvasás logika' tartja a kapcsolatot. A logikánál használt jelek a következők:

- **CS** a csipkiválasztás jele,
- **R/W** (olvasás/írás) jel, aminek megléte is szükséges az adatáramláshoz, ez a jel szinkronizálja az adatátvitelt a processzor és az eszköz között, ha például R/W = 1, akkor az eszköz felől a mikroprocesszor felé van adatátvitel, R/W = 0 esetén a processzor küld adatot az I/O eszközbe,
- **-RESET** jel biztosítja bekapcsolásnál az eszköz alapállapot-beállítását, ez a jel tulajdonképpen hardver jel, a táplálás bekapcsolásakor jelentkezik,
- **A0** és **A1** címvezetékek, amik meghatározzák a vezérlőregisztereket, itt a két vezeték 4 belső vezérlő regiszter meglétét teszi lehetővé,
- **IRQA** és **IRQB** vezetékek megszakítást kérhetnek a mikroprocesszortól akkor, ha azt a parancs-szó lehetővé teszi.

A bekapcsoláskor fellépő –RESET hardver jel csak bizonyos belső regisztereket, illetve a kimeneteket állítja alaphelyzetbe. Az alaphelyzet nem mindig 0, illetve nem minden eszköznél veszik fel a kimementi pontok a 0 értéket, van ahol ez az érték 1-es.
A programozó feladata, hogy megírjon egy inicializáló programrészt, amely a főprogram indulásakor a hardver-szoftver-igényeknek megfelelő kezdőhelyzetet hoz létre. Ez főleg a kimenetek, illetve bemenetek meghatározása, de ide tartozik a megszakítás engedélyezése, vagy tiltása is. Ez a kezdőhelyzet beállítás két részből áll:

- az eszköz alapállapotba hozása –RESET jellel, vagy programutasításokkal és
- az új parancsszó beírása az I/O eszköz parancsregiszterébe.

Ettől a pillanattól kezdve a program megfelelő, adatátvitelt biztosító része igény szerint lebonyolítja a processzor ←→ külvilág adatcserét. Ez az adatcsere három módon mehet végbe:

- közvetlen adatcsere vezérlőjelek használata nélkül,
- adatátvitel vezérlőjelekkel (hand shake) és
- vezérlőjeles adatátvitel, megszakítással.

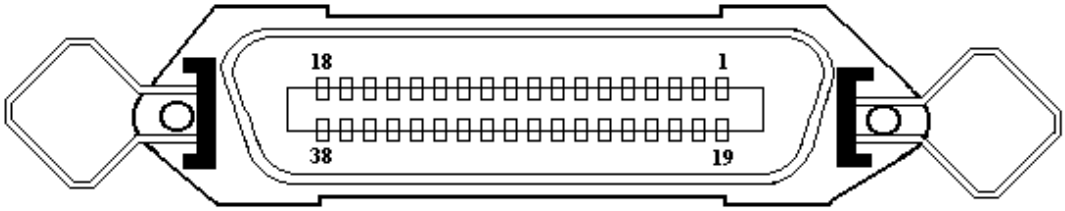
Az első adtátvitel ugyan a legegyszerűbb, de ilyenkor minden feltétel nélkül a kimeneti eszközre adat kerül, annak vizsgálata nélkül, hogy az működőképes, illetve foglalt, vagy nem. Ugyanez vonatkozik adatbeolvasásra is, feltétel nélküli adatbeolvasásra kerül sor, annak ellenőrzése nélkül, hogy az adott bemeneteken

valóban megtalálható-e a stabil, helyes adat. Látható, hogy az ilyen adatforgalom hardvere egyszerű, ám a megbízhatósága is kicsi.

Az előző adatátviteli mód hibáinak kiküszöbölésére szolgál a második tipusú, ahol megjelenik egy, vagy néhány vezérlő jel az adatátvitel szinkronizálására. Ilyen vezérlőjelek például az adat stabil, eszköz foglalt, vagy egy konkrét példa, a nyomtatóban nincs papir stb. A bonyolultabb hardver általában képes megoldani a szinkronizációs feladatokat, amennyiben nem, ezt szoftver eszközökkel kell pótolni.
A harmadik adtátviteli tipus hasonlít a másodikra, a különbség csak ott van, hogy nincs állandó szoftveres vezérlőjellekérdezés, hanem a hardver eszközön létrejövő változás hatására programmegszakítási igény lép fel, amit az I/O eszköz továbbít a processzor felé.

14.2.1 **A CENTRONIX párhuzamos adatátviteli csatorna**

Nagyon sokszor szükséges vézérlő-, mérő-, adatgyűjtő, vagy más rendszereket az igen elterjedt IBM PC személyi számítógéphez kötni adatcsere céljából. A berendezés és PC között párhuzamos adatcserelehetőség valósítható meg az úgynevezett CENTRONIX párhuzamos porton keresztül, ami gyakorlatilag minden PC tartozéka. Ez a port szolgál a párhuzamos nyomtatók személyi számítógéphez való kapcsolására. A 14.3. ábrán a PC személyi számítógép hátoldalán levő csatlakozó képe látható. A kivezetéseket, a lábakat az ábra szerint számozzák 1 és 38 között.



14.3. ábra: A PC személyi számítógép hátoldalán található CENTRONIX párhuzamos port

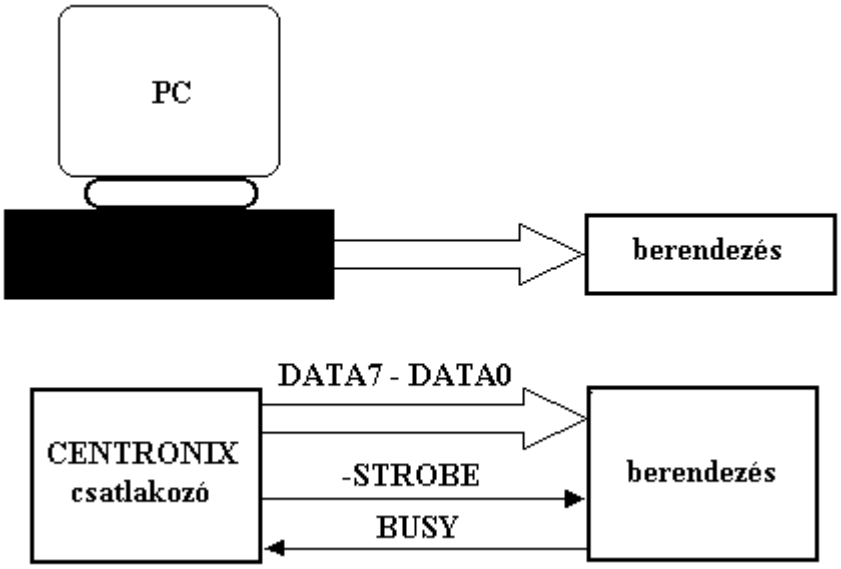
A 38 csatlakozási pont megnevezése és feladata a 14.1. táblázatból olvasható le. A táblázat **irány** oszlopa a csatlakozóhoz kapcsolt berendezés (például nyomtató) szempontjából értendő.

14.1. táblázat: A CENTRONIX párhuzamos csatlakozó lábkiosztása

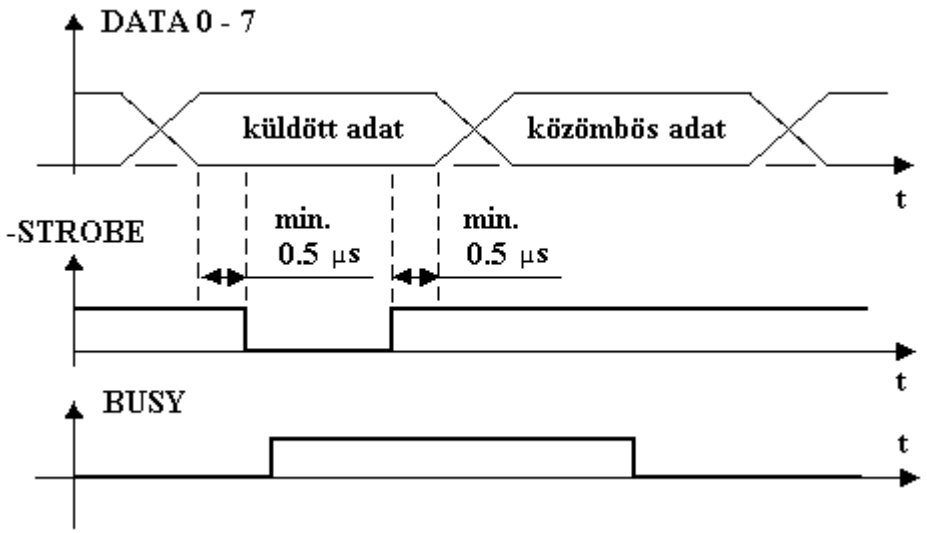
Láb	Jelölés	Irány	Leírás
1	-STROBE	bemenet	0 esetén beolvassa a DATA adatot (DATA7 – DATA0). Az impulzus legalább 0.5 µs kell hogy legyen

2	DATA 0 (LSB)		
3	DATA 1		
4	DATA 2		
5	DATA 3	bemenet	A nyolc párhuzamos adtbit
6	DATA 4		
7	DATA 5		
8	DATA 6		
9	DATA 7 (MSB)		
10	-ACKNOWLEDGE	kimenet	A jel 0 aktiv lesz, ha az adatátvitel befejeződött. Ekkor újabb adatátvitel kezdődhet.
11	BUSY	kimenet	Jelzi a PC felé, hogy foglalt a berendezés.
12	PE	kimenet	
13	SELECT	kimenet	Ha értéke 0, nem vehető a 8 adatbit
14	-AFD	bemenet	
15	NC		Nincs kihasználva.
16	OV		
17	CHASSIS GND		A berendezés házának földje.
18	+ 5 V	kimenet	Legfeljebb 50 mA-os áramforrás.
19-30	GND		Jel földelés.
31	-INPUT PRIME	bemenet	
32	-FAULT	kimenet	
33	GND		
34	NC		Nincs kihasználva.
35	+ 5 V	kimenet	
36	SLCT IN	bemenet	

A PC személyi számítógép és a mi általunk tervezett berendezés között kialakítható egy párhuzamos adatátvitel (14.4. ábra). Az eljárást megérthetjük a 14.5. ábrán látható idődiagramból.



14.4. ábra: Párhuzamos adatátvitel PC-ből berendezésbe CENTRONIX párhuzamos csatlakozóval

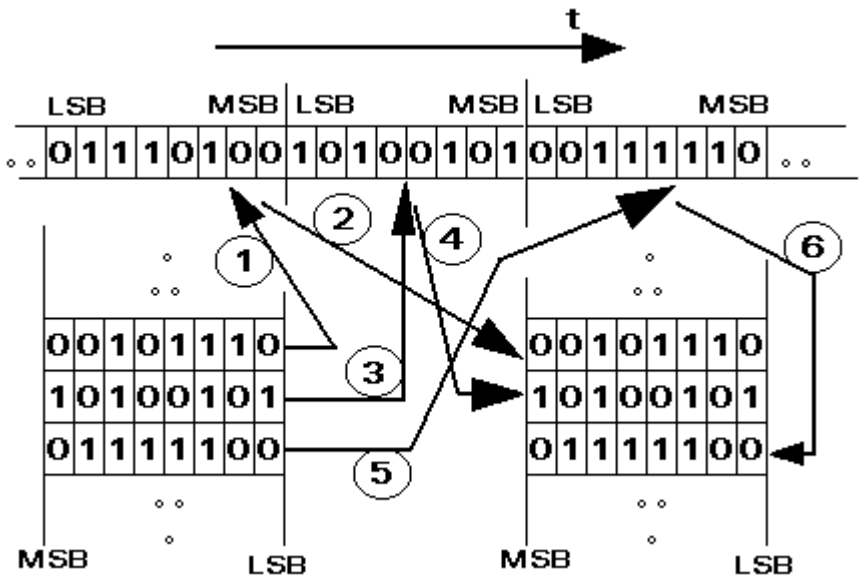


14.5. ábra: A párhuzamos adatátvitel idődiagramja PC-ből berendezésbe CENTRONIX párhuzamos csatlakozóval

Az idődiagramon látható, hogy nem szükséges az összes jel felhasználása a vezérelt adatátvitelhez. A PC program, ami megírható bármilyen nyelven, először leellenőrzi, hogy a berendezés fogadóállapotban vane-e. Ezt az állapotot a BUSY jel 0 értéke jelzi. Ezek után ki kell küldeni párhuzamosan, egyszerre a nyolc adatot (DATA7 – DATA0). Az adónak, a PC-nek az eddig 1-es szinten levő –STROBE jel 0-ra húzásával kell jeleznie a berendezés felé azt a tényt, hogy az adatok immár stabilak (nincsenek átmeneti állapotban), tehát a berendezés átveheti azokat. Az adatok beállítása és a –STROBE megjelenése között legalább 0,5 µs-nak kell eltelnie. A vevő érzékeli a –STROBE aktivvá válását, átveszi az 1 bájtos adatot, de egyuttal a BUSY jel felemelésével jelzi, most foglalt, fel kel dolgozni az adatot. Ez idő alatt az adó (PC) ellenőrzi a berendezés állapotát, de nem küld új adatot, mert foglalt a vevő. Miután a vevő (a berendezés) feldolgozta a kapott adatot, szabaddá válik, amit a BUZY jel 0-ra ejtésével jelez az adó felé.

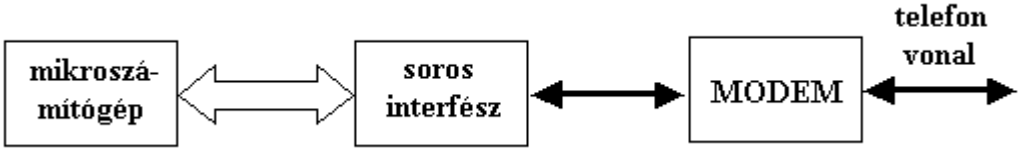
14.3. SOROS I/O VEZÉRLŐ-ILLESZTŐ ESZKÖZÖK

A soros adatátvitelnél az adóoldali I/O eszköz feladata az, hogy a párhuzamos (rendszerint 8 bites) adatot soros impulzusok sorozatává alakítsa át, azért, mert a soros sadatátvitel egy vonalon keresztül történik. A vevőben található soros I/O eszköz a sorosan vett impulzusok sorozatát alakítja vissza párhuzamos adattá (14.6. ábra).



14.6. ábra: Párhuzamos adatok sorossá való átalakítása és visszaalakítása párhuzamos adatokká

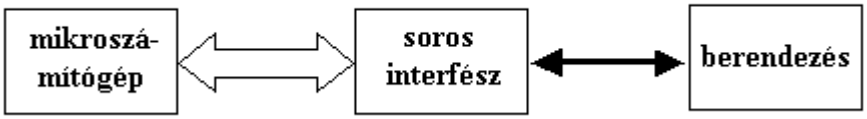
A soros adatátvitel igen elterjedt, mindössze egy vonal (érpár) szükséges megvalósításához. Főleg térben távoli helyek között történik ily módon az átvitel. Igen gyakran a már meglevő telefonvonal használható adatátvitelre (MODEM közbeiktatásával), aminek segítségével Interneten keresztül bármely távoli helyen levő számítógép elérhető. A 14.7. ábrán látható a számítógép és telefonvonal összekapcsolása.



14.7. ábra: Adatátvitel mikroszámítógép és távbeszélővonal között

A hagyományos telefonvonal analóg jelek továbbítására képes egy bizonyos frekvenciatartományon belül, így szükséges a bináris digitális jeleket átalakítani folytonos jelekké, amelyek szinuszos jelek. Ezt végzi a MODEM, ami MODulátor-DEModulátor rövidítése.

Egyéb berendezések, például adatgyűjtő eszközök, szabályozók stb. is kapcsolhatók soros interfész alkalmazásával a mikroszámítógépen, ahogy az a 14.8. ábrán látható.



14.8. ábra: Eszköz csatlakoztatása soros interfésszel a mikroszámítógéphez

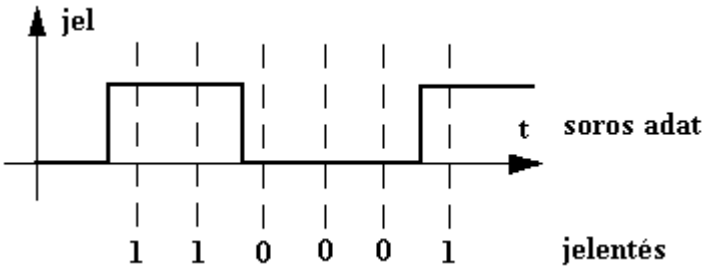
14.3.1 **Soros szinkron és aszinkron adattovábbítás**

Soros adatátvitel esetén megkülönböztetünk:

- szinkron és
- aszinkron adatátvitelt.

14.3.1.1 Szinkron soros adatátvitel

Ennél az adatátvitélnél egy órajelnek van kiemelt szerepe, az órajel megadása után minden egyes óraimpulzus alatt egy bitet kell átvinni. Ekkor az átvitt információ pontosan értelmezhető. A 14.9. ábrán látható a soros adatátvitel bináris alakban, illetve fészültség (áram) értékekben kifejezve.



14.9. ábra: Szinkron soros adatátvitel

Mivel általában mindig bájtokat visz át a soros egység, szükséges a bájtok közötti határ megtalálása. Amennyiben ez nem történik meg, teljesen használhatattlan adatok kerülnek feldolgozásra. A bájthatár megtalálásának módja az, hogy az értékes adatok átvitele elütt az adó két szinkronizációs jelet küld ki. Mi van akkor, ha nem biztosítható folyamatosan adatátvitel, nincs mindig adat, például kézi adatbevitel esetén az operátor hol gyorsabba, hol lassabban viszi be az adatokat ? A megoldás az, hogy ilyenkor az adó szinkronizációs jelekkel tölti ki az üres helyeket. Vegyük a következő példát:

Nehéz megtanulni a tantárgyat.

Az egyszerűség kedvéért használjuk csak az ASCII karaktereket, ott is a nagybetűket, ekkor a küldendő adat:

NEHEZ MEGTANULNI A TANTARGYAT.

A betűközt helyettesítsük a # jellel, ekkor a küldendő adat:

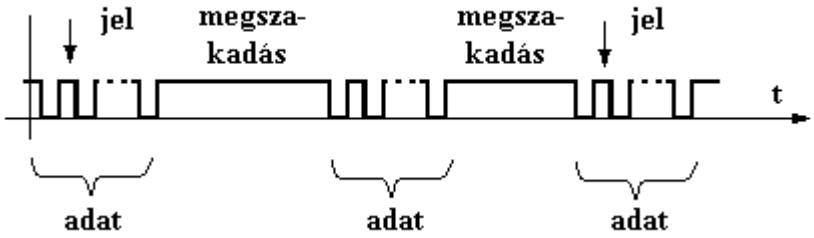
NEHEZ#MEGTANULNI#A#TANTARGYAT.

Amikor nincs értékes adat, az adó a $ szinkronizációs jelet iktatja közbe, így kapjuk az átvitelnél a következő adatfolyamot:

NE$$H$E$$$Z#$ME$$GTANUL$$NI#A$#TAN$$TARGYAT.

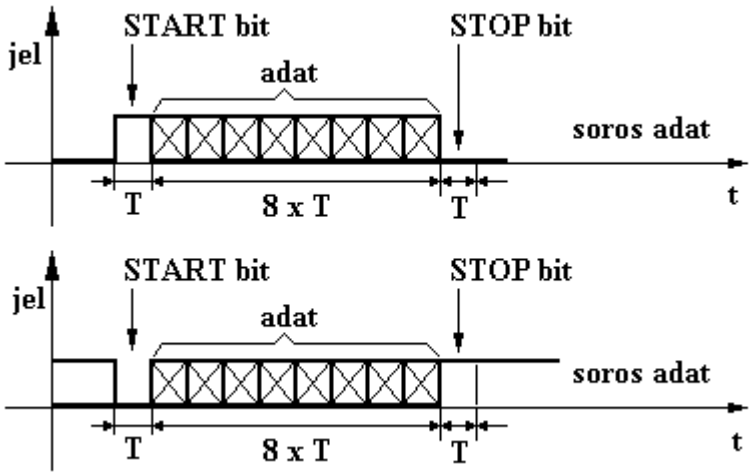
14.3.1.2 Aszinkron soros adatátvitel

Az aszinkron adatátvitel abban különbözik a szinkron átviteltől, hogy akkor, amikor nincs adat, nem történik semmilyen jelátvitel, a karakterek (bájtok) között egy szakadás van (BREAK). Amennyiben a logikai 0-hoz 0 V feszültséget, a logikai 1-hez például 5 V feszültséget rendelünk, akkor nem lehetne különbséget tenni egy valódi fizikai vezetékszakadás és egy jelmegszakadás között. Ezért rendszerint fordított logikát használunk, vagyis a logikai 0 lesz 5 V érték, míg a logikai 1-es 0 V. Így akkor amikor nincs adatátvitel állandó magas szint van a vezetéken. Egy soros aszinkron adatátvitel látható a 13.10. ábrán.



14.10. ábra: Aszinkron soros adatátvitel

A két adat közötti idő tetszőleges hosszúságú lehet, úgyhogy valami módon meg kell oldani a szinkronizálást az adó és a vevő között. Ehhez szükséges egy adadtátvitelt indító jelzőbitet bevezetni, illetve az adatok végét jelző bitet. A valódi 8 bites adatot megelőzi egy START bit, valamint lezárja (legalább) egy (vagy több) STOP bit. Így valójában 1 bájt, vagyis 8 bit átviteléhez legalább 10 bitre van szükség. Az aszinkron adatátvitel látható a 14.11. ábrán, az **a**) részen a logikai 0 0 V feszültségnek, míg a logikai 1-es 5 V feszültségnek felel meg, az ábra **b**) részén pedig fordítva. Értelemszerűen a START és STOP bitek is megfordulnak, ellenkező értékűek. A 8 adat logikai , illetve 1 értékű lehet, ezt az ábrán a jelben levő áthúzással jelöltük.



14.11. ábra: Aszinkron adatátvitel szervezése

14.3.1.3. RS-232C aszinkron feszültségszintű soros átviteli csatorna

A **CCITT** nemzetközi távközlési bizottság a **V.24**-es és **V.28**-as ajánlásaiban meghatározta a soros adatátviteli interfész feladatait és elektromos jellemzőit. Az adatátvitel a **DTE** (Data Terminal Equipment) berendezés, ami a számítógép, illetve számítógépes terminál és a **DCE** (Data Communications Equipment) távközlési, vagy

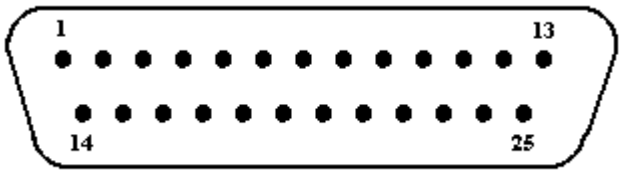
egyéb berendezés között történik. Az amerikai **EIA** szervezet egy szabványt fogalmazott meg az interfész létrehozására, aminek neve **RS-232C**. Ezen szabvány szerint soros összeköttetés valósítható meg bármilyen két berendezés között. A berendezés lehet számítógép, mérőfej, megjelenítő eszköz stb. Aszinkron üzemmód mellet az RS-232C interfésszel szinkron kapcsolat is létrehozható.

Az interfész csatlakozóból, kábelből és elektromos elemekből áll. Az interfész feszültségértékei a 14.2. táblázatban láthatók. A szabványban használt TTL jelszintnél magasabb feszültségértékek csökkentik az átvitelnél zavarólag ható zavarjelek hatását.

14.2. táblázat: Az RS-232C **feszültségszintjei**

TTL jelszintek		**RS 232 jelszintek**	
Feszültség értékek	**Jelentés**	**Feszültség értékek**	**Jelentés**
0 V…..0,8 V	alacsony (0)	3 V……15 V	alacsony (0)
2,4 V……5 V	magas (1)	-15 V…..-3 V	magas (1)

A csatlakozó lábkiosztása a 14.12. ábrán látható, ennek neve **D25**.



14.12. ábra: **D25**-ös **RS-232C** csatlakozó lábkiosztása

Ellenőrző kérdések

1. Milyen technológia jellemzi a perifériális eszközöket ?
2. Mi az I/O eszköz feladata ?

3. Mit jelent az, hogy a periféria interfész programozható ?
4. Rajzolja le az adat- és vezérlőjel kapcsolatot az I/O eszköz és a mikroprocesszor között .
5. Mi a parancsszó az I/O eszközöknél ?
6. Mi az állapotszó az I/O eszközöknél ?
7. Milyen típusú I/O eszközök léteznek ?
8. Jellemezze a párhuzamos I/O eszközöket .
9. Rajzolja le egy párhuzamos I/O eszköz blokksémáját .
10. Ismertesse a párhuzamos I/O eszköz egyes blokkjainak, egységeinek szerepét .
11. Ismertesse a CENTRONIX párhuzamos adatátviteli csatornát .
12. Jellemezze a CENTRONIX csatlakozó egyes jeleit .
13. Ismertesse a PC és berendezés összekötését CENTRONIX párhuzamos csatornával .
14. Rajzolja le a CENTRONIX párhuzamos adatátviteli csatorna idődiagramját .
15. Ismertesse a soros I/O vezérlő-llesztő eszközöket .
16. Hogyan történik a párhuzamos adatok átvitele soros adatátviteli csatornával ?
17. Mi a MODEM ?
18. Ismertesse a soros szinkron adatátvitelt .
19. Ismertesse a soros aszinkron adatátvitelt .
20. Mi a START bit szerepe ?
21. Mi a STOP bit szerepe ?
22. Ismertesse a szabványos RS 232C aszinkron feszültségszintű adatátvitelt .
23. **Hasonlítsa össze az RS 232C csatorna jelszintjeit az TTL jelszintekkel .**

15. I^2C SOROS BUSZ

15.1. BEVEZETÉS

Berendezések közötti soros adatátvitel megvalósítható az ún. **RS 232C** szabvány szerinti soros átviteli megoldással. Ennek a soros átvitelnek hátránya a viszonylag lassú adatátvitel. Amennyiben nagybonyolultságú LSI technológiában készült elemek, integrált áramkörök között szeretnénk gyors adatátvitelt létrehozni, más megoldást kell választani. Ennek egy lehetséges formája az úgynevezett I^2C soros busz, amely rövidítés az Inter IC rövidítése, jelentése IC-k közötti busz. Szokásos jelölés még az **I2C** rövidítés.

Az I^2C busz nagybonyolultságú integrált áramkörök közötti soros információcserét biztosít félduplex módon fizikailag három vezetékkel:

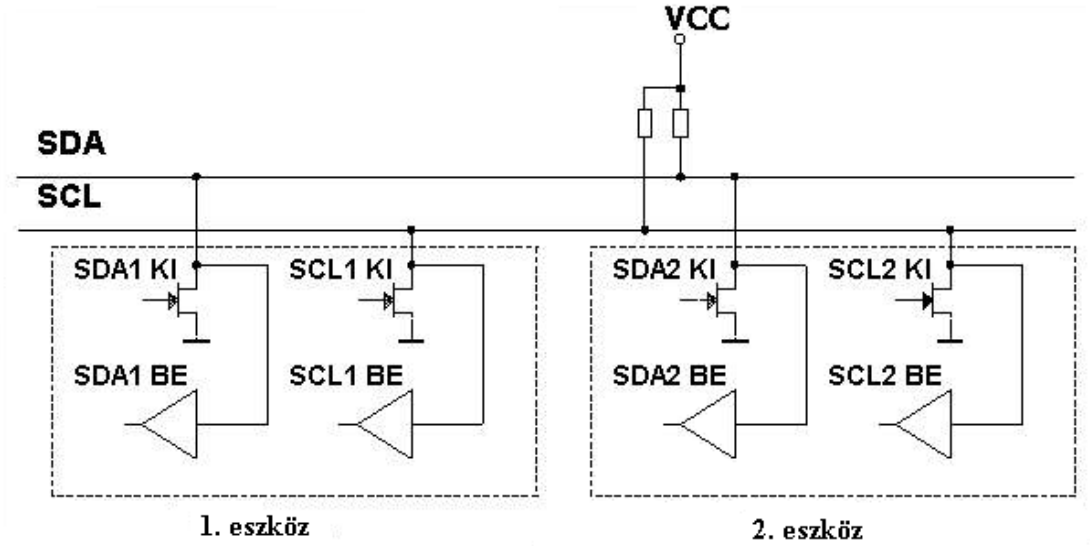
- **SDA**, azaz Serial Data kétirányú adatvonal és
- **SCL**, azaz Serial Clock órajel.

Az adatátvitelnek két egység között három lehetséges megoldása van:

- **simplex** adatátvitel, egyik eszköz az adó, a másik a vevő, egyirányú adatátvitel,
- **halfduplex**, mindkét eszköz lehet adó, illetve vevő, de egyszerre csak egy irányban van adatátvite,
- **fullduplex** adatátvitel, mindkét eszköz egyidőben ad, illetve vesz.

Az átviteli sebesség elérheti akár a 100-400 kbit/s sebességet is. Sok ilyen nagybonyolultságú integrált áramkör már tartalmaz ilyen buszt, egymással könnyen, kevés vezetékkel képesek sorosan kommunikálni.

Az SDA **vezeték kétirányú, félduplex adatátvitelt biztosít a** SCL **szinkronizáló jel segítségével. Az átvitel az 15.1. ábra kapcsolási rajza segítségével magyarázható meg.**



15.1. ábra: I2C soros adtátvitel elve

Alaphelyzetben a tranzisztorok kikapcsolt állapotában a felhúzó ellenállás miatt, a vonalak magas állapotban vannak (**Vcc** feszültség, ami **TTL 5 V**). Ha bármelyik tranzisztort bekapcsoljuk, az a vezetéket a földre húzza le, vagyis a vezeték '**0**' állapotban lesz. Ez a "huzalozott vagy kapcsolat"-s.

A vezérlési elv alapján mindig csak egy egység vezérelheti az adott vezetéket, a többi egység a tranzisztorát nem kapcsolhatja be, hiszen akkor zavarnák egymást a jelek. Ez a működési mód az ún. adás. Ugyanerre a pontra, vagyis az **SDA** vezetékre minden eszköznél egy erősítő van kapcsolva, amely erősítők veszik a küldött jelet.

Egy adott eszköz adatküldéskor könnyedén megállapíthatja azt, hogy a vonal szabad-e, azaz küldheti az eszköz az adatot, hiszen ugyanarról a vezetékről olvas, mint amire adatot küld, ha az olvasott adat nem egyezik a küldöttel, más eszköz is adatot küld tehát foglalt a busz.

A buszon az információáramlás iránya alapján megkülönböztethetünk **adó** és **vevõ** egységeket. Az átvitel vezérlését a **Master** (mester) eszköz végzi, irányítva a **Slave** (szolga) eszköz működését. Így két funkció és két szerep különböztethetõ meg.

A két funkció:

- **TRX** = Transmitter (adó), az az egység, amelyik adatot küld a buszra, illetve a buszon keresztül a másik eszközre,
- **RCV** = Recevier (vevő), a buszon keresztül adatot fogadó eszköz.

A két szerep:

- **MST** = Master (mester), az az egység, amelyik kezdeményezi az átvitelt, egyúttal az átvitelhez az órajelet generálja, és be is fejezi az átvitelt.
- **SLV** = Slave (szolga), a mester által megcímzett egység, amely figyeli vételkor az órajelet is.

Egy mikrokontroller, amennyiben rendelkezik I^2C egységgel mindegyik szerepre és funkcióra képes. Az az egység, amely a perifériákat vezérli lesz a Master (mester).

A busz ún. multi-master kialakítású, vagyis a buszra kapcsolódó eszközök közül nem csak egy, hanem több is lehet Master, így átveheti az átvitel vezérlését. Ilyenkor fontos tulajdonság a **busz arbitráció** (arbitration), vagy döntés. Ez biztosítja azt, ha egynél több mester akarja a buszt vezérelni, akkor ezt csak egyetlen egy tudja megtenni, így adatvesztés nincs.

Adatátvitelkor biteket visz át a rendszer, amely átvitel bájtszervezésben történik.

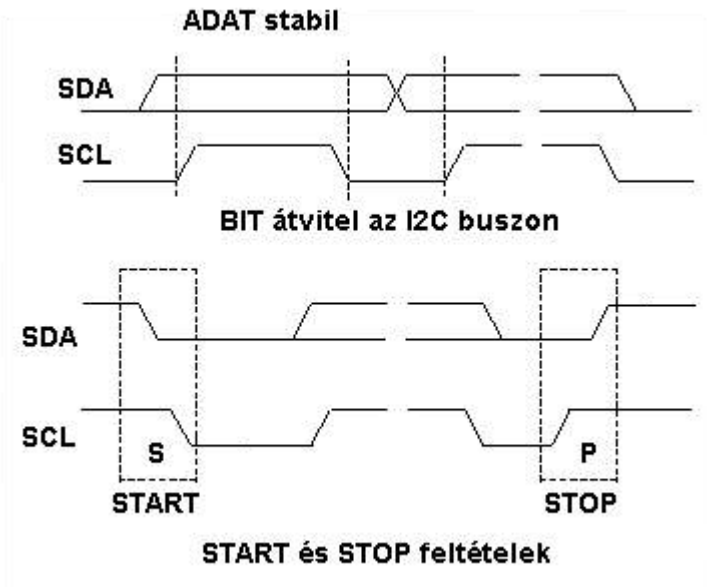
15.2. EGY BIT ÁTVITELE AZ I^2C BUSZON

Az **SDA** és **SCL** vonal alapállapotban magas szintű.

Egy bit átvitele a következõ módon történik:

- a kiinduláskor magas szinten lévõ SDA adatvonalra kerül az átviteli érték, a ’**0**’ vagy ’**1**’ értéknek megfelelõ **0 V** vagy **5 V** –os feszültség.
- Az **SCL** vonal magas szintje alatt érvényes az adat, vagyis a vevő ekkor olvashatja, az adat csak az SCL vonal alacsony szintje alatt változhat, ilyenkor a vevő nem ilvas adatot.

A folyamat a 15.2. ábrán követhető.



15.2. ábra: Egy bit átvitele, **START** és **STOP** feltétel az I^2C buszon

A busz aktív a **START** és inaktív a **STOP** állapot után, ami másképpen a következő:

- START feltétel akkor lép fel és a busz aktív lesz amikor SCL magas állapotában az SDA vonalon egy magas állapotbólból alacsony állapotba történő átmenet van (SCL=1, SDA = 1 $\Rightarrow$ 0).
- STOP feltétel akkor lép fel, amikor SCL magas állapotában az SDA vonalon egy alacsonyból magasba való átmenet van (SCL=1, SDA = 0 $\Rightarrow$ 1).

A **START** és **STOP** állapotokat (és az órajelet) csak a mester eszköz hozhatja létre. A busz aktív a START és STOP állapot között, a STOP jel után újból szabaddá válik a busz (inaktöv állapot).

15.3. EGY BÁJT ÁTVITELE AZ I^2C BUSZON

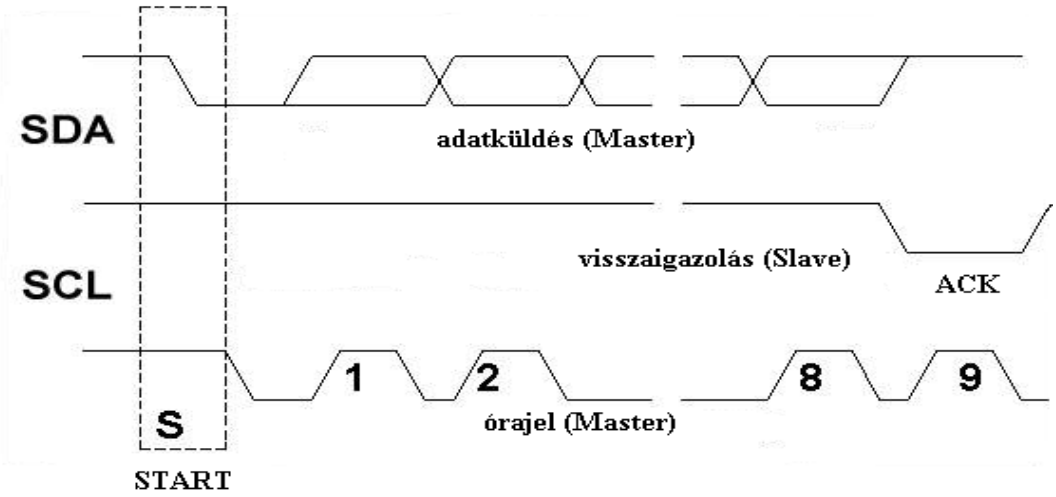
A számítástechnikában elfogadott tény, hogy adatok átvitelénél a bájtszervezést használjuk. Az I^2C busz adatátvitele is bájt szervezésű. Az átvitt bájtok száma tetszőleges.

A 15.3. ábrán követhető nyomon egy bájt átvitele a Master eszközről a Slave eszközre.

Az adónak információt kell szereznie arról, hogy a vevő sikeresen vette-e a küldött jelet, vagyis a bájtot. Az adó a vevőnek minden bájt vétele után egy **L** szintű nyugtázó (**ACK**, azaz acknowledge) bitet küld, igazolva a sikeres vételt. Természetesen az

órajelet most is a Master generálja, ilyenkor az adó az SDA vonalat felszabadítja. A vevő ekkor az ACK generáláshoz lehúzza az SDA vonalat. Az átvitel a legmagasabb helyiértékű (MSB) bittel kezdõdik.

Az adatbiteket az adó, az ACK bitet a vevő küldi. Ha a vevõ nem képes egy bájtot, adatot venni akkor az ACK bit küldése elmarad, és egyúttal az SCL vonalat 0 szinten tartja. Ez egy várakozó állapot. Látható, hogy az adat- és az órajel vonalat az adó és a vevõ felváltva, igényeik szerint használja.



15.3. ábra: Bájtátvitel az **I2C** buszon

Adatátvitel során nincs mindig nyugtázás. Ez két esetben fordul elő:

- ha a mester a vevõ (MST/RCV), az adónak valahogy jelezni kell az adatbájt sorozat végét, de úgy, hogy a küldőnek nem adva ACK-ot. Az ACK jelhez kapcsolódó órajelet a Master természetesen generálja, de az SDA vonalat nem húzza le L szintre. Ez az ún. negatív nyugtázásnak (NACK).
- a szolga akkor nem küld NACK jelet, ha nem képes újabb adatbájtokat elfogadni. Ez akkor történik meg, ha olyan adatátvitelt kezdeményez, amit nem képes fogadni.

15.4. ADATFORGALOM A BUSZON TÖBB ESZKÖZ KÖZÖTT

A buszon lévõ minden eszköznek saját címe van, csak így tehető egyértelművé minden egység szerepe. Az adatátvitelt a Master kezdeményezi azzal, hogy a buszt START állapotba hozza, majd kiküldi a buszra a szolga címét, amelyikkel adatot akar cserélni. Az a szolga, amelyik felismeri a saját címét, ACK jellel jelzi azt, hogy felismerte a címet és képes adatot fogadni A címzést a mester végzi közvetlenül START állapot létrehozása után, ez az elsõ küldött bájt.

A cím hossza hét bit, így legfeljebb 128 eszköz címezhető. A nyolcadik, a legkisebb bit (LSB) határozza meg el a szolgával történõ adatcsere irányát. 0. bit írást jelent, ilyenkor a Master küld adatokat (**W**), 1 pedig az olvasást (**R**) jelenti.

A buszra kapcsolt eszközök címei két csoportba sorolhatók:

- programozható címmel rendelkező eszközök, ezek általában a mikrokontrollerek,
- a különféle funkciókat megvalósító periféria áramkörök címei.

Az I^2C eszközök címe két részbõl áll:

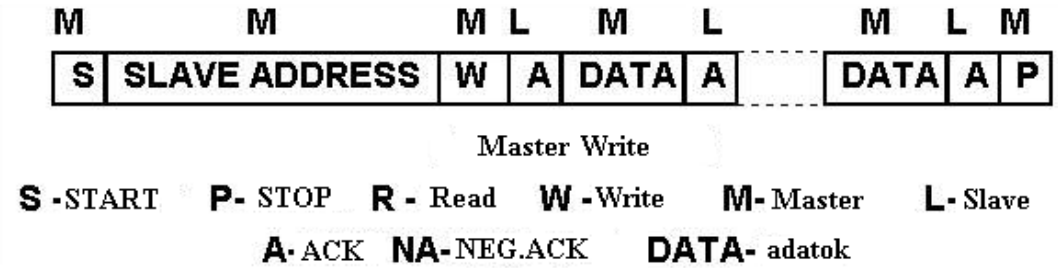
- típus címbõl (4 bit), amit a gyártók rendelnek az eszközökhöz és
- egy hardver címbõl (3 bit) amely a címző lábainak ’0’-ba ill. ’1’-be kötésével állítható.

A típus cím az azonos (típusú) tokokra jellemző cím és mindig megegyezik. Ezzel a címmel jelentkezik be a slave eszköz ill. ezzel a címmel szólítja meg a Master eszköz a Slave-et adatcsere elõtt.

Elemezzük azt az esetet, amikor egy I^2C buszon egy Master mellett több szolga eszköz található. M a mester által küldött adatbiteket jelöli, míg L a szolga által küldött adatbiteket.

15.4.1 **MASTER WRITE (mester írás) – állapot**

A 15.4. ábrán követhető idődiagramon a folyamat.

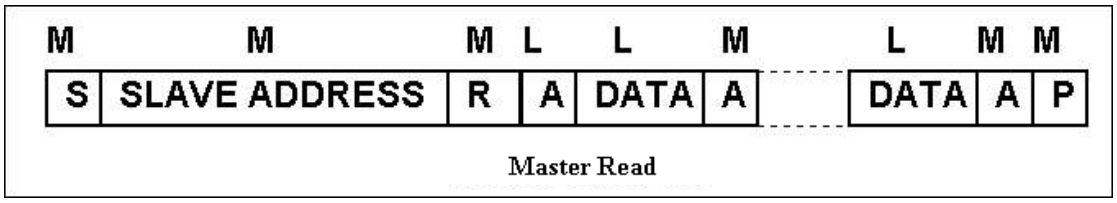


15.4. ábra: Master write (mester írás)

A Master START állapotba hozza a buszt (**S**) és kiküldi a szolga címét. A cím legkisebb helyiértékű bitje **W=0**. A címet felismerve a szolga az **ACK** jellel jelez a Master felé (**A**). Ezután a Master elküldi az adatokat a szolgának (**DATA**), és a Slave minden bájt vételét (**A**) küldésével visszaigazolja. Az utolsó adat küldése után a mester **STOP** állapotba hozza a buszt (**P**).

15.4.2. **MASTER READ (mester olvasás) - állapot**

A 15.5. ábrán követhető idődiagramon a folyamat.



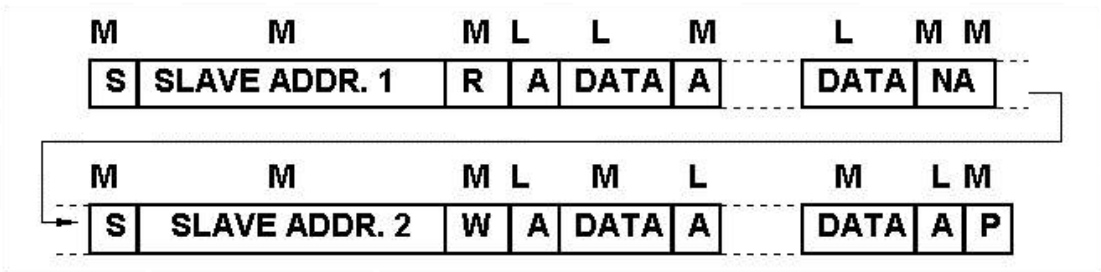
15.5. ábra: Master read (mester olvasás)

A Master miután **START** állapotba hozza a buszt (**S**), kiküldi a szolga címét. A cím legkisebb helyiértékű bitje (LSB) **R=1**. Ezt a Slave az **ACK** jel visszaküldésével igazolja (**A**). Ezek után a Master fogadja az adatokat a szolgától (**DATA**), és minden bájt vételét (**A**) kiküldésével igazolja vissza. Az utolsó adat küldését a Master negatív nyugtázással jelzi (**NA**) a Slave felé. Ezek után a Master a buszt a **STOP** állapotba hozzással (**P**) fejezi be az adatátvitelt.

A fent ismertetett adatátviteli eljárásnak hátránya, hogy a Master minden átvitelnél a buszt újból nyitja és zárja. Ilyenkor, ha a buszon több szolgával akar a Master adatot cserélni, a minden átvitelt lezáró **STOP**, majd az indító újabb **START** sokat lassít az átvitelen. Ennek a hiányosságnak a kiküszöbölése oldható meg az ismételt **START** állapot előállításával. Ez az jelenti, hogy az átviteleket nem **STOP** (**P**) hanem a következõt indító **START** (**S**) állapottal fejezzük be, azaz a Master a buszt folyamatosan használja.

15.4.3 **Az ismételt START állapot használata folyamatos adatátvitelnél**

Az ismételt **START** állapot módszere idődiagramon a 15.6. ábrán látható.



15.6. ábra: Ismételt START állapot adatátvitelnél

Az ábra egy olyan esetet mutat be, ahol elõször a Master adatokat kapott egy szolgától, majd utána adatokat küld egy másik szolgának.

A fent tárgyalt adatátviteli protokollokat, módszereket két módon lehet megvalósítani:

- beépített hardver segítségével (ilyeneket már tartalmaznak az erre kifejlesztett mikrokontrollerek és az I^2C buszra tervezett memóriák és periféria áramkörök) és
- szoftver programot írunk a folyamat lejátszására.

Az I^2C busz igen eredményesen használható olyan esetben, ha egy mikrovezérlő képtelen ellátni a tervezett feladatokat. Ekkor szétbontható a feladat több részre, mindegyik magvalósítása egy-egy mikrovezérlőhöz rendelhető, amiket I^2C buszon keresztül köthetünk össze. Mivel itt csatlakozó nem létezik (a nyomtatott lapon a megfelelő lábakat csak össze kell kötni), nincs semmilyen csatlakozóval, kábellel jelentkező szabványosítási, megbízhatósági, stb probléma.

Ellenőrző kérdések

1. Milyen eszközök összeköttetésére szolgál az I^2C busz ?
2. Hány vezeték szükséges az adatátvitel lebonyolításához és hogyan nevezik ezeket a jeleket ?
3. Milyen átviteli sebesség érhető el a módszerrel ?
4. Vázolja fel az I^2C busz hardver megoldását, magyarázza el a busz alaphelyzetét és az adatátvitelt.
5. Hogyan ellenőrzi a rendszer azt, hogy csak egy adó működik egyidőben ?
6. Információáramlás alapján milyen eszközöket különböztetünk meg a rendszerben és hogyan nevezzük ezeket ?
7. Mi az adó és a vevő két funkciója ?
8. Mi az adó és a vevő két szerepe ?
9. Mi a busz arbitráció ?
10. Hogyan történik egy bit átvitele az I^2C buszon keresztül, rajzolja le az átvitel idődiagramját .
11. Mia START és mi a STOP bit ?
12. Hogyan történik egy bájt átvitele az I^2C buszon keresztül, rajzolja le az átvitel idődiagramját .
13. Mit jelent az ACK és a NACK rövidítés ?
14. Magyarázza el az adatátvitelt I^2C buszon keresztül, ha a rendszerben kettőnél több elem van .
15. Az I^2C busznál mi az eszközcím, hány részből áll és mi a bitek jelentése ?
16. Mi a Master Write állapot, idődiagramon ábrázolja a folyamatot .
17. Mi a Master Read állapot, idődiagramon ábrázolja a folyamatot .
18. Mi az 'ismételt START' módszer, ábrázolja a folyamatot idődiagramon és miért van rá szükség ?
19. Hogyan lehet az I^2C busz szerinti adatátvitelt olyan eszzközöknél létrehozni, amelyek nem rendelkeznek beépített hardver I^2C busz megoldással ?

16. ADATÁTVITEL A MIKROSZÁMÍTÓGÉP, MIKROVEZÉRLŐ ÉS A PERIFÉRIÁK KÖZÖTT

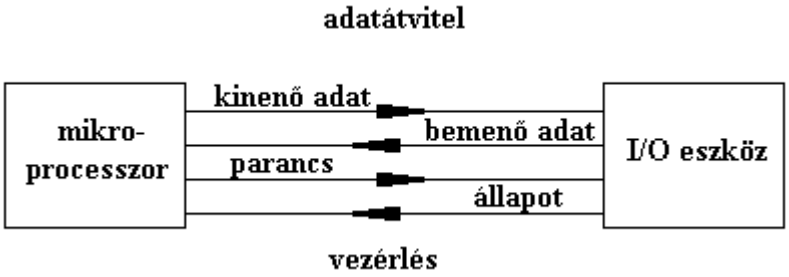
16.1. BEVEZETÉS

A számítógépek fő feladata adatokon műveleteket végezni annak érdekében, hogy egy, vagy több faladatot végrehajtsanak. Ennek során állandó adatcserére kerül sor a külvilág és a mikroszámítógép, mikrovezérlő között. A külvilág (ember, másik számítógép, illetve folyamat) a legkülönbözőbb tipusú jeleket állítja elő (analóg és digitális, feszültség, nyomás, hőmérséklet, elmozdulás, szög stb.), ugyanúgy a legkülönfélébb jeleket kell a számítógépnek visszaküldenie az adott rendszerbe.

Mint már láttuk, a számítógépen belül, tehát a processzor és memóriák között egy aránylag szoros, bináris, illetve logikai kapcsolat van, ellentétben a számítógép és külvilág közötti kapcsolatot biztosító I/O elemekkel szemben, ahol meg kell oldani:

- a különféle analóg és digitális fizikai mennyiségek bináris alakba való átalakítását,
- szint és teljesítményillesztést és
- az időzítési problémákat.

Általánosan az adatátviteli folyamatot a 16.1. ábra szerinti blokksémával szemléltethetjük. Előfordul, hogy egy I/O eszköz nem végez mindkét irányú adatátvitelt, ilyenkor vagy csak adatbevitelről, vagy csak adatkivitelről beszélünk.



16.1. ábra: Mikroprocesszor és I/O eszköz közötti adatáramlás és vezérlés

Az I/O eszköznek negy feladatot kell megoldania:

- az adatok ideiglenes tárolása,
- címdekódolás és eszközkiválasztás,
- parancsértelmezés és
- időzítés és vezérlés.

Az adatok ideiglenes tárolására azért van szükség, mert a bemeneten megjelenő adat, valamint a feldolgozás ideje nem mindig esik egybe, illetve adatkivitelnél a következő, új adat kiírásáig a régi adatnak stabilnak kell lennie.

A címdekódolás és eszközkiválasztás a megfelelő, megcímzett periféria aktivizálását jelenti, azon időperiódusra, míg lezajlik az adatátvitel. Ez idő alatt a rendszer összes más perifériája az adatsinről lekapcsolt állapotban van.

Mivel az I/O eszközök több, összetettebb feladat elvégzését is lehetővé teszik, szükséges a parancsok dekódolása, értelmezése majd azok végrehajtása.

A fent felsorolt feladatok végrehajtása folyamán mind a mikroprocesszornak, mind az I/O eszköznek különböző hosszúságú idő alatt lezajló folyamatait kell szinkronizálni.

A mikroprocesszor, mikrovezérlő és I/O eszközök közötti adatátvitel három módon oldható meg. Ezeket a megoldásokat önállóan is lehet használni, de sokszor előfordul a módszerek egyidejű használata is. A három módszer a következő:

- programozott I/O adatátvitel,
- megszakítással kezdeményezett I/O adatátvitel és
- közvetlen memóriahozzáférés (DMA).

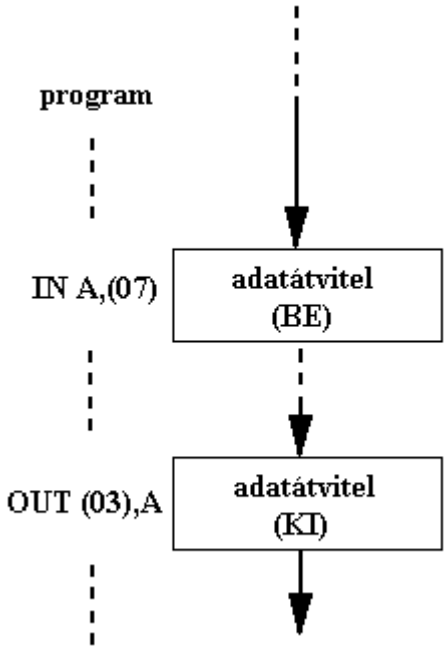
16.2. PROGRAMOZOTT I/O ADATÁTVITEL

A mikroprocesszor és bemeneti/kimeneti eszközök közötti programozott adatátvitelnek három lehetséges formája van:

- feltétel nélküli adatátvitel,
- feltételes adatátvitel és
- feltételes kiválasztásos adatátvitel.

16.2.1. Feltétel nélküli programozott I/O adatátvitel

Csak olyan esetben használatos, amikor a külső periféria válaszideje ismert. Ennél a megoldásnál a külső eszköznek mindig késznek kell lenni az adatátvitelre (akár küldés, akár fogadás) a programban szereplő megfelelő utasítás végrehajtásakor. Egyszerűségéből következik, hogy a program, valamint a hardver felépítése is igen egyszerű. A 16.2. ábrán látható a feltétel nélküli adatátvitel vázlata.



16.2. ábra: Feltétel nélküli I/O adatátvitel

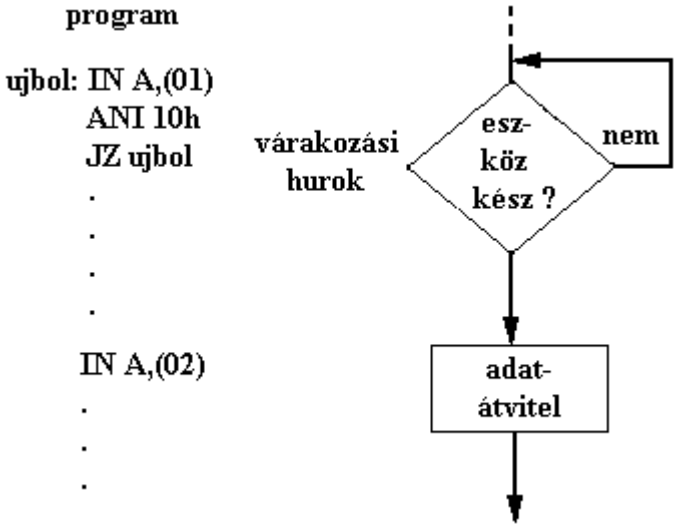
A programban szereplő **IN A,(07)** utasítás hatására a bemenő eszköz portján levő 8 bit bekerül a processzor akkumulátorába. Az **IN** utasítás (07) része a beviteli eszköz címét adja meg, itt ennek értéke 7. Ezt követően egy programrész végrehajtására kerül sor, amely rész az ábrán pontokkal van jelölve. A másik adatátviteli utasítás, az **OUT (03),A** hatására az akkumulátorban levő 8 bit átkerül a (03) címmel ellátott kiviteli eszközre, ahol az eszközben levő regiszter tárolja az új adatot egy újabb adatkiírásig, adatváltoztatásig.

Mindkét adatátviteli utasításban közös az, hogy nem történik semmilyen vizsgálat annak megállapítására, hogy a beviteli, vagy a kiviteli eszköz képes-e az adott feladat elvégzésére. Tehát megtörténhet az is, hogy a kiviteli, illetve beviteli eszköz hibás, vagy kikapcsolt állapotban van, esetleg nics csatlakoztatva a rendszerre. Ilyenkor adatbeolvasáskor véletlenszerű érték kerül az akkumulátorra, vagy a rendszer ír ugyan a kimeneti eszközre, de a kiírandó adat nem jelenik meg az eszközön.

16.2.2. Feltételes programozott I/O adatátvitel

A feltétel nélküli adatátvitel hibái miatt egy olyan adatátviteli eljárást kell alkalmazni, ahol a beviteli/kiviteli eszköz állapotát a rendszer vizsgálja az adtátvitel elött és annak állapotához köti az adatátvitel végrehajtását, vagy elhalasztását. Ennek a módszernek a neve 'feltételes I/O adatátvite', és gyakorlatilag a programozott adatátvitelek közül ezt alkalmazzuk minden esetben mikroprocesszor és I/O eszköz között.

A 16.3. ábrán látható a feltételes programozott adatátvitel. A program az I/O eszköz állapotát vizsgálja, amennyiben nincsenek meg a feltételek az adatátvitel lebonyolítására egy ciklus alakul ki. Itt látszik a módszer előnye mellett az is, hogy ebben az esetben gyakorlatilag a program csak az eszköz állapotát teszteli, ami azonban a még nagyobb baj, mindaddig nem engedi a programot tovább, míg nem alakul ki az adatátviteli feltétel, tehát más I/O eszköz lekezelése nem jöhet létre.

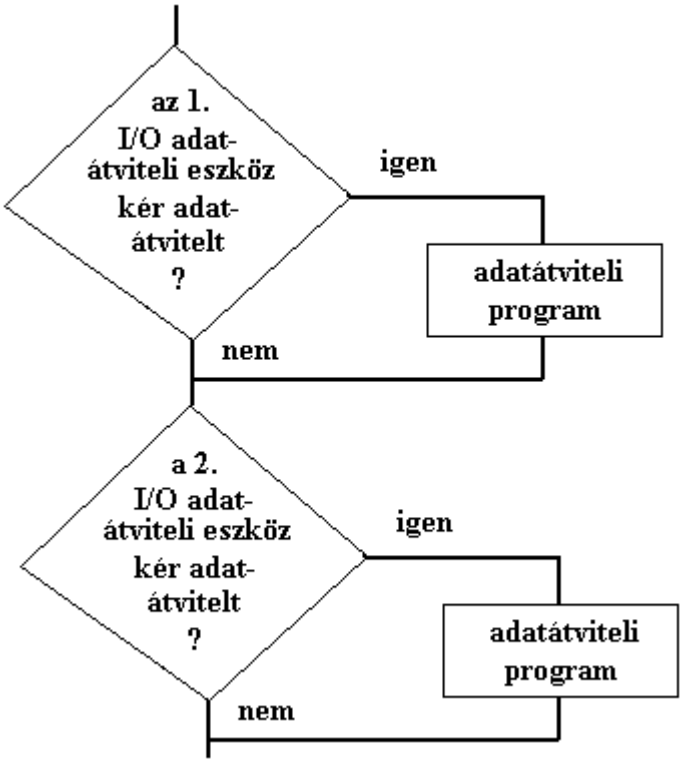


16.3. ábra: Feltételes I/O adatátvitel

A programban az **IN A,(01)** utasítással kerül az akkumulátorba az eszköz(ök) állapota, amit az **ANI 10h** utasítással maszkolunk (logikai ÉS művelet), vagyis csak az aktuális I/O eszközhöz tartozó állapotbitet választjuk ki az összes állapotbit közül. Ha ennek értéke 0 lesz, vagyis az eszköz nem képes az adatátvitel lebonyolítására a **JZ ujbol** utasítással újból elindítjuk a programot. Adatátvitelre kész I/O eszköz esetében kilép a program a ciklusból és elvégzi a szükséges adattranszfert, vagyis a példánkban az akkumulátorba olvassa a kettes címre kötött perifériából az adatot.

16.2.3. **Feltételes, kiválasztásos (polling) I/O adatátvitel**

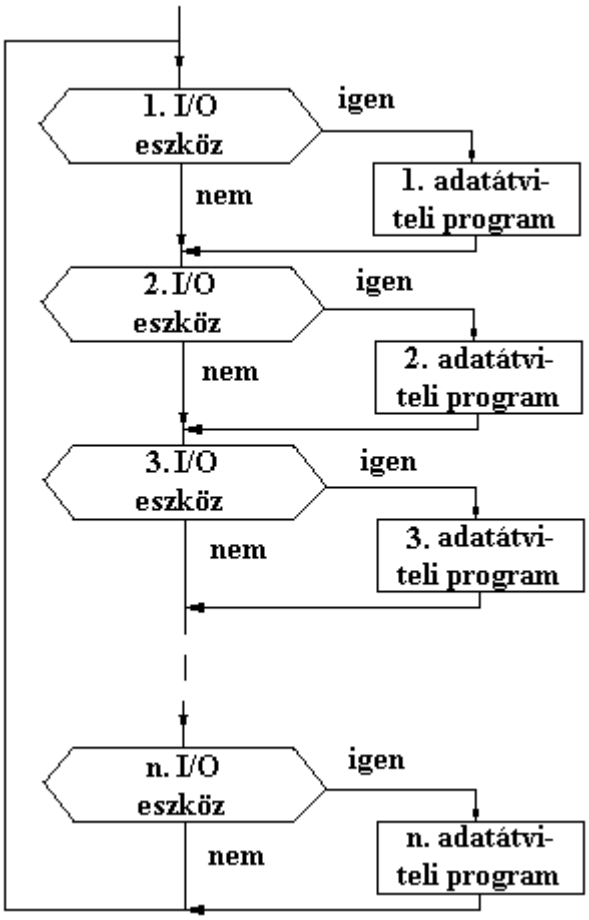
Több I/O eszköz esetében a feltételes adatátvitel nem működik, hiszen egy eszköz már leállíthatja a többi periféria adatátvitelét. Ennek kiküszöbölésére használjuk a kiválasztásos, feltételes adaátviteli müdszert, amely módszer a 16.4. ábrán látható.



16.4. ábra: Feltételes, kiválasztásos I/O adatátvitel

Az ábrán látható, hogy nincs várakozás egyik I/O eszköz lekérdezésénél, illetve kiszolgálásánál sem, hiszen abban az esetben, ha a periféria nem képes adatátvitelre, a programban nem történik hurok kialakítása, tovább lép a következő eszköz kiszolgálására. Akkor, ha a periféria igényelt adatátvitelt, illetve képes az adatátvitel lebonyolítására, ez megtörténik, ami után ugyancsak a következő I/O eszköz vizsgálatára, illetve kiszolgálására kerül sor.

A 16.5. ábrán látható egy teljes ciklus több I/O eszköz kiszolgálására a kiválasztásos (polling), feltételes I/O adatátviteli módszerrel. Ennek a módszernek előnye az, hogy minden eszközzel csak addig foglalkozik a program, míg meg nem állapítja, hogy szabad-e a periféria, viszont ez a módszer hátránya is, hiszen minden ciklusban minden eszközről meg kell állapítani ezt az adatot, ami sok időt emészt fel. Ez különösen akkor zavaró, ha a perifériák ritkán képesek adatátvitelre.



16.5. ábra: A kiválasztásos, feltételes adatátvitel ciklusa

16.3. MEGSZAKÍTÁSSAL KEZDEMÉNYEZETT ADATÁTVITEL

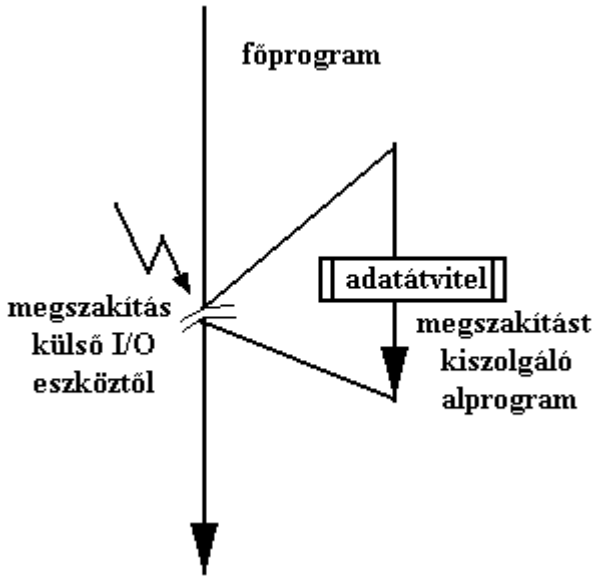
Az előző, programozott I/O adatátvitelnél viszonylag sok idő szükséges annak megállapítására, hogy az I/O eszköz igényele, illetve képes-e az adatátvitelre. Ez lassúbb programvégrehajtást idéz elő, ugyanakkor az események pontos egymás utániságát is befolyásolja.

Ezért célszerűbb az adatátvitelnél a megszakítással kezdeményezett adatátvitelt alkalmazni, itt ugyanis csak akkor kerül sor adatátvitelre, ha azt az I/O eszköz megszakítás-igénnyel jelezte. A főprogram akadálytalanul végrehajthatja az ott előírt feladatait, anélkül, hogy időt pazarolna a perifériák lekérdezésére.

Megszakítással kezdeményezett adatátvitelnél a főprogram végrehajtása, amennyiben ez engedélyezett megszakad, a vezérlés átkerül egy alprogramra, amely ellátja az igényelt adatátvitelt. Ez a folyamat a 16.6. ábrán látható.

A mikroszámítógép megszakításos rendszere aszinkron eseményekre is tud válaszolni, vagyis a külső I/O eszközök igényeire, úgy, hogy nem várakozik ciklusban az eszköz

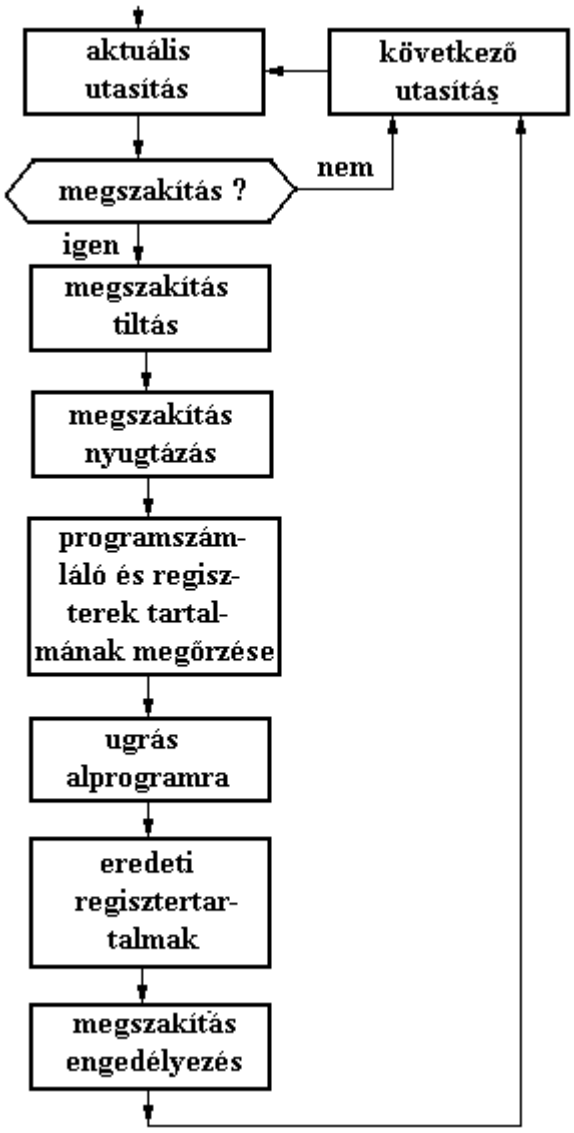
szabaddá válására. Mivel a mikroprocesszoroknak több megszakítást fogadó bemenetük van, illetve ha kevés a bemenetek száma az külső logikával bővíthető, több periféria kiszolgálása is lehetővé válik.



16.6. ábra: Megszakítással kezdeményezett adatátvite

A 16.7. ábrán követhető a megszakítással kezdeményezett adatátvitel folyamata, ahol feltételezzük, hogy előzőleg már programból engedélyeztük a megszakítás elfogadását:

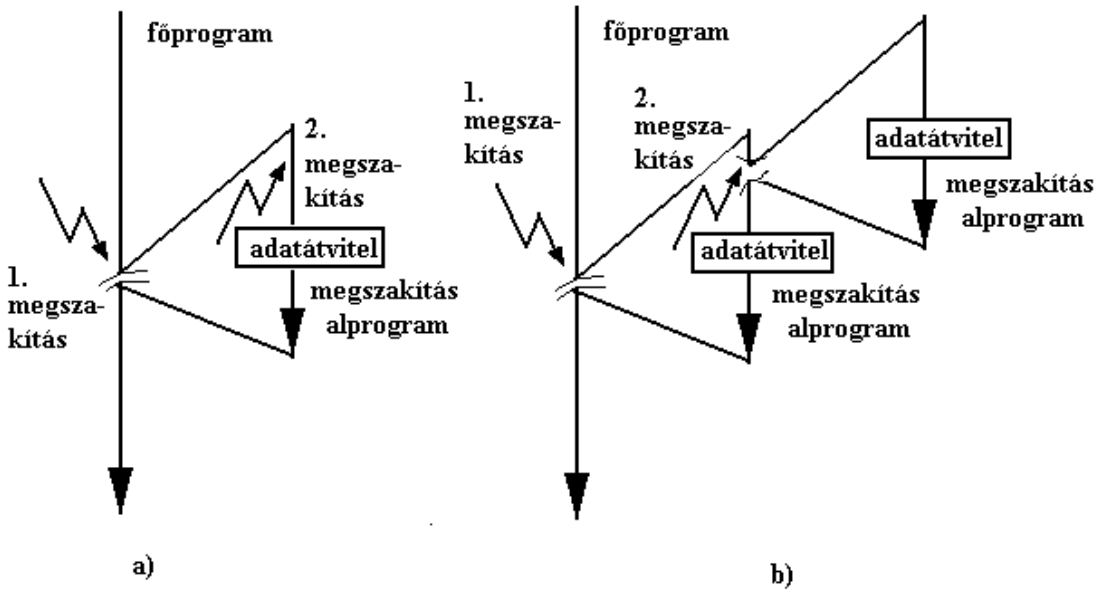
- valamilyen soron következő utasítás végrehajtása van folyamatban,
- az I/O eszköz logikája megszakítási igényt küld a mikroprocesszor, mikrovezérlő megfelelő interrupt lábára,
- a mikroprocesszor, mikrovezérlő befejezi az elkezdett utasítás végrehajtását,
- elfogadja a megszakítást és megszakítást nyugtázó jelet küld vissza az I/O eszköz felé,
- letiltja ugyanennek a megszakításnak az elfogadását,
- megtörténik a programszámláló, valamint a regiszterek tartalmának mentése a veremtárba,
- a vezérlésvégrehajtás átkerül a megszakítást kiszolgáló alprogramra, vagyis az adatátviteli alprogram elvégzi az adatcserét a mikroprocesszor és az I/O eszköz között,
- a vezérlés visszakerül a főprogramra,
- megtörténik az elmentett regiszterek tartalmának visszamentése,
- újból engedélyezett megszakítás elfogadása,
- a főprogramban a következő utasítás végrehajtása következik.



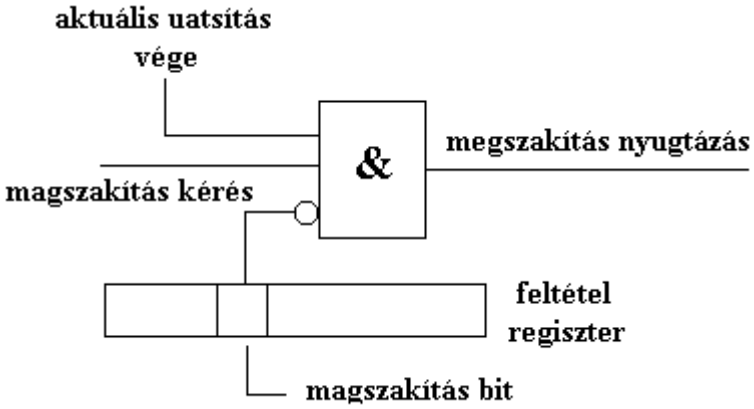
16.7. ábra: megszakítással kezdeményezett adatátvitel folyamatábrája.

A mikroszámítógépek, mikrovezérlők lehetővé teszik a megszakítások maszkolását, valamint a megszakítások közötti prioritás, fontossági sorrend meghatározását. A 16.8. **a**) ábrán egy olyan eset látható, amikor két megszakításból csak az 1. engedélyezett (maszkolással), ilyenkor hiába keletkezik hardver jel formájában igény a megszakításra, ezt a mikrogép nem fogadja el. A 16.8. **b**) ábrán két megszakítás engedélyezett, ez maszkolással biztosítható, valamint az első megszakítást a második megszakíthatja, vagyis a 2. megszakításnak nagyobb a prioritása.

Az I/O eszköz megszakítási igénnyel fordul a mikroprocesszorhoz, hogy az szakítsa meg az éppen futó programot és egy megfelelő adatátviteli alprogram segítségével végezze el az adatátvitelt. Amennyiben programból engedélyezett a megszakítás elfogadása, valamint az aktuális utasítás végrehajtása befejeződott, a mikroprocesszor áttér a megszakítás feldolgozására. Erről az eseményről értesíti a perifériát egy ún. megszakítás-nyugtázó jellel. A nyugtázó jel létrehozása a 16.9. ábra szerinti feltételek teljesülésével jön létre.



16.8. ábra: Megszakítások egymásbaágyazása



16.9. ábra: A megszakítást nyugtázó jel feltételei

A megszakítással kezdeményezett adatátviteli eljárás előnye a nagy gyorsaság, gyors válaszadás, ez azért lehetséges, mert a megszakításkezelés a mikroprocesszoros, mikrovezérlős rendszerekben azonnali végrehajtást eredményez. Különösen a valós idejű (Real Time) rendszerekben alkalmazható a módszer hatékonyan.

A módszer hátránya abban van, hogy a többforrásos és több prioritási szinttel rendelkező megoldások hardver kiegészítő elemeket igényelnek. Lassítja a megszakításos kiszolgálást az is, hogy minden megszakításkiszolgálás esetén programszámláló és regisztermentési műveletet kell végrehajtani, illetve a főprogramba való visszatéréskor ezen adatok visszamentése is kötelező. A PC és regisztertartalmak mentése memóriareferens művelet, ami időigényes folyamat.

Ez az eljárás is aszinkron folyamat, ami programszervezési feladatok megoldását igényli az esetleges konfliktushelyzetek feloldására, ez szoftver probléma, az adott szoftver futtatása pedig lassító jellegű.

Az adatátvitel a megszakításkéréssel elindított adatcserénél egy programvezérelt folyamat, ami időt vesz el a rendszertől.

16.3.1. **A megszakításforrás azonosítása**

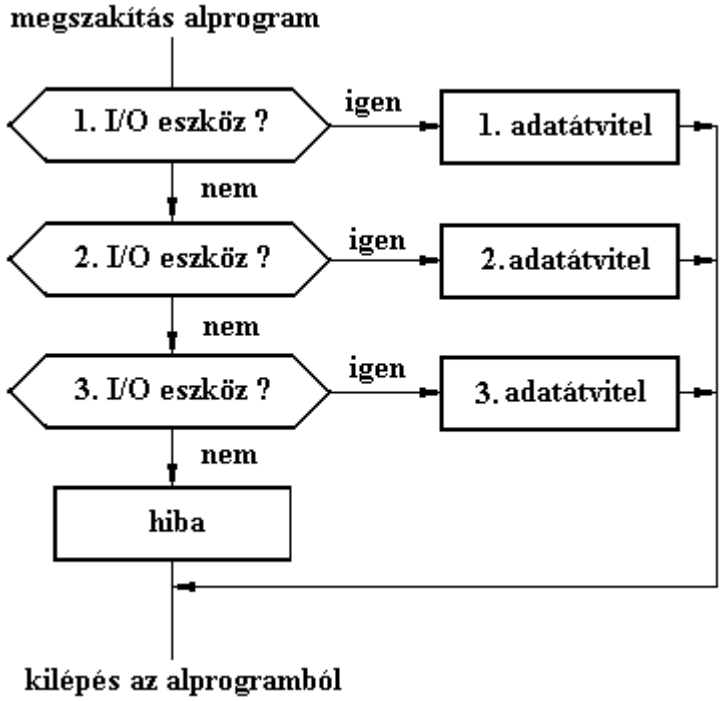
Mint már volt róla szó, a mikroprocesszoroknak, mikrovezérlőknek általában több megszakítás-vonaluk van. Ha minden egyes megszakításhoz csak egy I/O eszközt rendelünk hozzá, a rendszer könnyen kezeli a kéréseket, hiszen minden megszakításnak címe van, könnyen azonosítható.

Abban az esetben, ha egy megszakítás-vonalhoz több periféria csatlakozik a rendszernek fel kell ismernie a megszakítás-forrést. Ez két módon oldható meg:

- az eszközök lekérdezése és
- vektoros megszakítás módszere.

16.3.2. A megszakításforrás azonosítása - az eszközök lekérdezése

Ennél a módszernél az adott megszakításhoz egy alprogram tartozik, amely a megszakítás hatására elindul, lekérdezéssel kideríti a megszakítás forrását, majd kiszolgálja az adott eszközt, vagyis végrehajtja az adatátvitelt. A folyamat a 16.10. ábrán követhető nyomon.



16.10. ábra: A megszakítás-forrás felismerés az eszközök lekérdezése módszerrel

16.3.3. **A megszakításforrás azonosítása – vektoros megszakítás**

A vektoros megszakítás olyan processzort feltételez, amiben egy logika ismeri fel a megszakításforrás címét. Ekkor minden I/O eszközhöz egy cím van hozzárendelve. Erre a címre ugrik az alprogram (ágazik el), amely címen végrehajtja az adott

eszközhöz tartozó adatátviteli eljárást. A címek egymás után, szekvenciálisan állnak rendelkezésre.

16.4. ADATÁTVITEL KÖZVETLEN MEMÓRIAHOZZÁFÉRÉSSEL – **DMA**

16.4.1. **Bevezetés**

Mint ahogy a programozott és megszakítással elindított adatátvitelnél láttuk, a processzor a megírt program segítségével bonyolítja le az adatátvitelt, ami tulajdonképpen azt jelenti, hogy a minden egyes műveletehez tartozó program az operatív memóriából bekerül a processzorba, ott értelmezi a vezérlő egység, majd annak alapján a memória és a processzor regisztere között (rendszerint akkumulátor) lebonyolítja az adatátvitelt.

A mikroszámítógépeknél nagymennyiségű adat átvitele lehet pl. a memrevlemez egység és memória, egy A/D és memória stb. között.

Felmerül a kérdés, ha több adat mozgatása válik időszerűvé, nem lehetne-e közvetlenül átvinni az adatokat a memória és az I/O eszköz között, kikerülve a mikroprocesszort. Ekkor ugyanis:

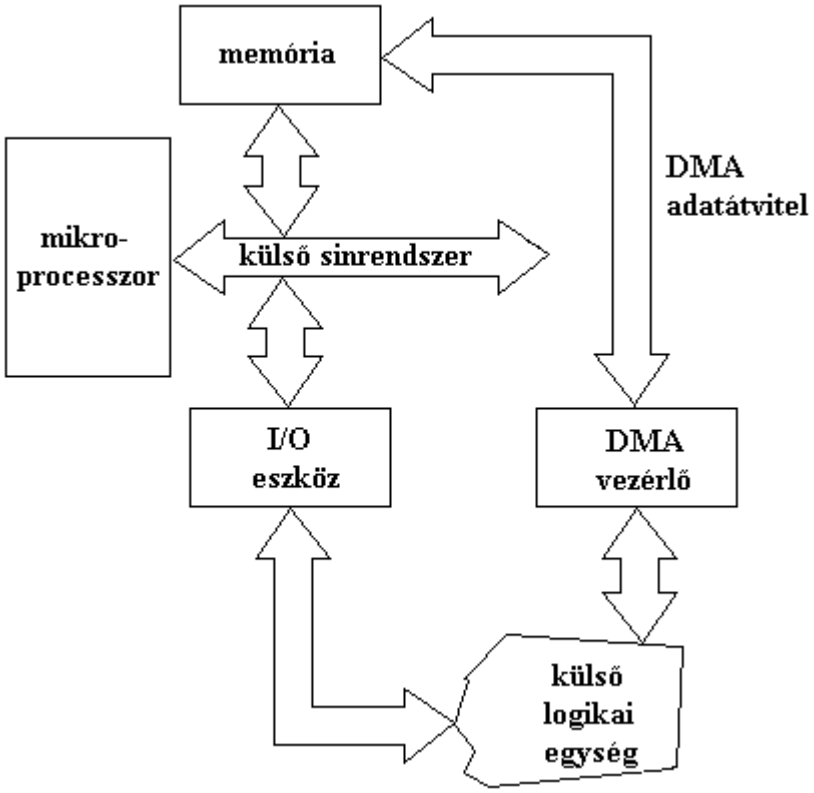
- nincs programvégrehajtás,
- nem kell a regisztertartalmakat elmenteni, visszamenteni,
- beállítható a legnagyobb adatátviteli sebesség a memória és periféria közott és
- feleslegesen az adat nem kerül a processzorba, valamint kétrszer is az adatbuszra.

16.4.2. **A közvetlen memóriahozzáférés szervezése**

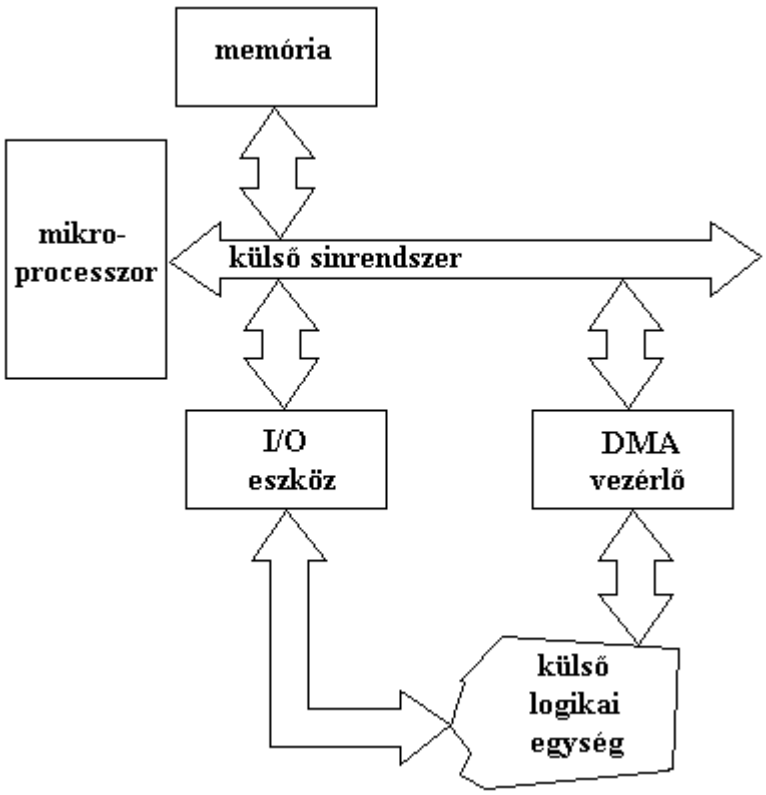
Maga a **DMA** (Direct Memory Access) mint periféria kapcsolódik a mikroszámítógép rendszhez. Felépítése elvileg nagyon hasonlít a processzor felépítéshez, azzal a különbséggel, hogy itt nem egy univerzális, hanem célfeladatot ellátó egységről van szó. Mivel a DMA hasonlóan működik egy CPU-hoz, ez az egyetlen elem egy mikroszámítógépes rendszerben, ami a processzor mellett aktív egység lehet. Eddig csak mindig azzal az esettel találkoztunk, hogy a CPU volt az aktív elem (minden feladatot a CPU irányított), míg a többi egység (memóri és I/O eszközök) passzívak voltak.

A 16.11. ábrán látható egy DMA egységet tartalmazó mikroszámítógép. Látható, hogy a periféria és az operatív memória közvetlenül van összekapcsolva, nincs kötődés a processzorhoz, illetve a buszrendszerhez.

Gyakorlati okok miatt, például amiatt, hogy a memóriáknak nincs, csak egy adatsinjük, ha lenne, akkor pedig amiatt, hogy a sinrendszert mégegyszer meg kellene valósíta a nyomtatott lapon, valamint amiatt, hogy már eleve létezik a mikroszémítógépben a címsint, adatsint és vezérlősint tartalmazó sinrendszer, kihasználják a feladat megoldására ezeket a megoldásokat. A 16.12. ábrán ezen feltételek figyelembe vétele mellett kialakított DMA struktúrát láthatunk.

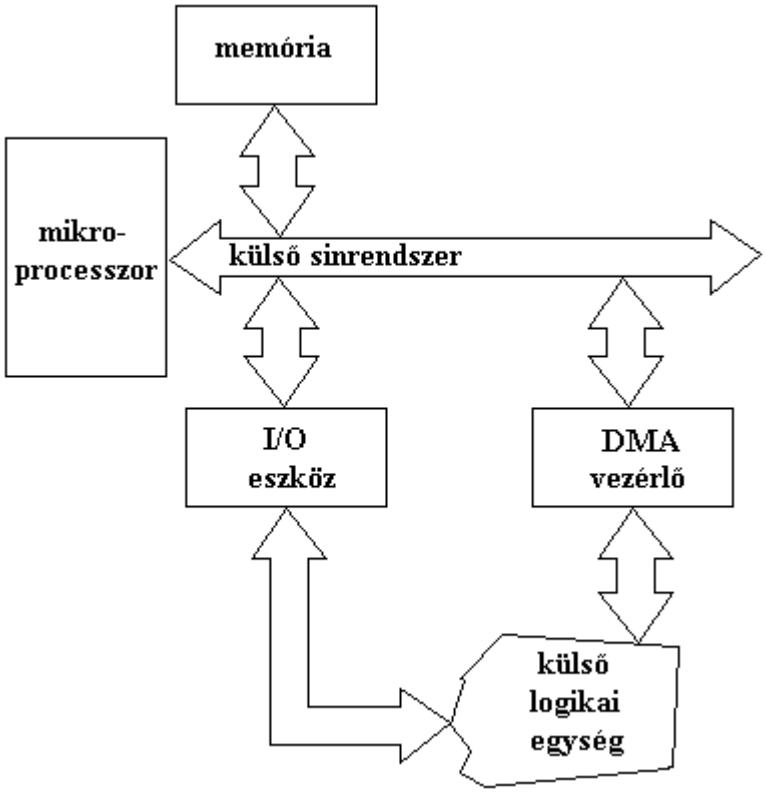


16.11. ábra: A DMA sematikus ábrázolása



16.12. Egy valós, a meglevő sinrendszerre épülő DMA

Amikor a mikroprocesszor végzi a rendszer vezérlését, vagyis közönséges programvégrehajtásról van szó, akkor a DMA vezérlő elektromosan a haromállású logika segítségével lekapcsolódik a mikroszámítógépről. Ez az állapot a 16.13. ábrán látható.

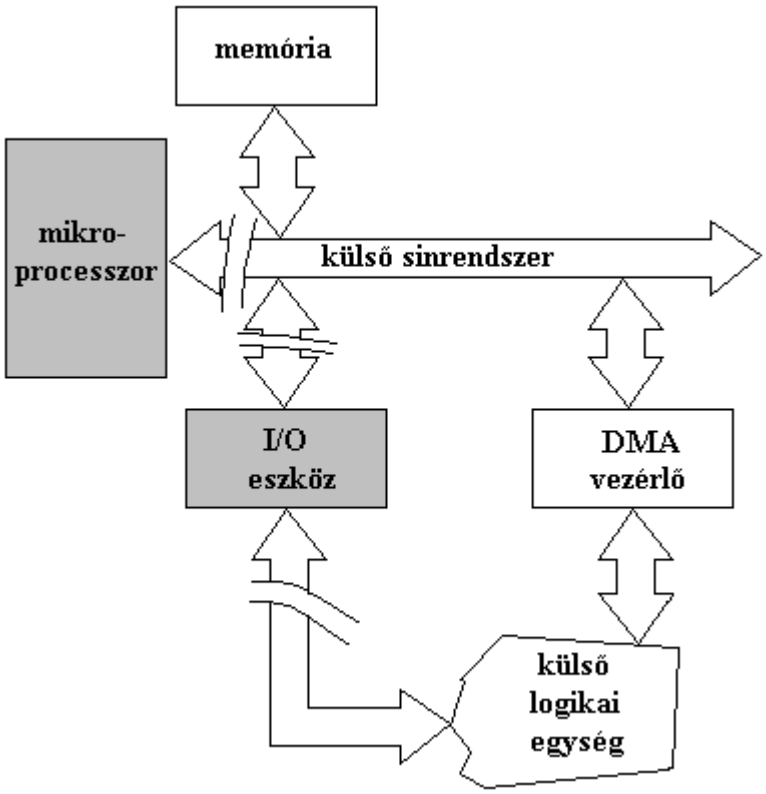


16.13. ábra: A mikroszémítógép 'normális' üzemmódban, DMA lekapcsolva

Akkor, amikor DMA tipusú adatátvitel igénye meül fel, a különböző paraméterek, címek, adatok beállítását a CPU kezdeményezi és hajtja végre programok és adatok alapján. Megállapítja a DMA állapotát az állapotszó alapján, majd átküldi a DMA egységbe a memória kezdőcímét, az átvitelre kijelőlt adatblokk hosszát, valamint a működéshez szükséges parancsszó tartalmát. Ezen információk átvitele után a CPU elindítja a DMA segítségével a közvetlen adatátvitelt. Ekkor a 16.14. ábra szerinti állapotba kerül a mikroszámítógép, látható, hogy a DMA az adatátvitel befejezéséig aktív egységként viselkedik. Az adatátvitel után újból I/O eszközként viselkedik (16.13. ábra).

Többféle módon lehet végrehajtani a DMA adatátvitelt:

- DMA adatátvitel a processzor teljes leállítása mellett,
- DMA adatátvitel cikluslopásos elven,
- DMA az előző két módszer egyesítésével és
- DMA adatátvitel a CPU/DMA műveletek multiplexálásával.



16.14. ábra: DMA adatátvitel

16.4.2.1. DMA adatátvitel a processzor leállításával

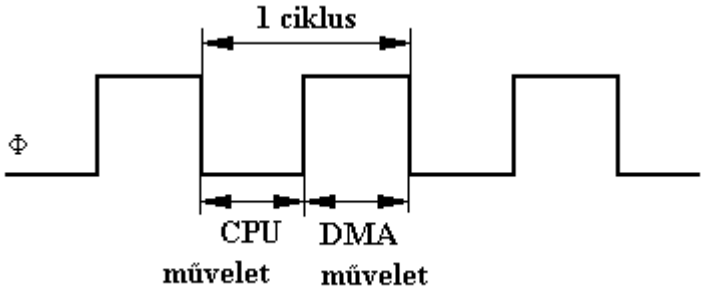
Ez a legegyszerűbb módszer a DMA adatátvitel lebonyolítására I/O eszköz és memória között, de a legkevésbé használatos módszer. Gyakorlatilag a processzorműveletek leállnak, ami nem teszi a rendszert hatékonnyá,

16.4.2.2. DMA adatátvitel cikluslopással

Ennél a módszernél a processzor tovább működik, maga a DMA viszonylag rövid időt vesz el a processzoridőből. Gyakorlatilag lassítja a rendszer frekvenciáját az ilyen módon megvalósított adatátvitel, de a módszer előnye, hogy a processzor tovább végzi a feladatát.

16.4.2.3. DMA adatátvitel CPU/DMA multiplexálással

Ez az eljárás biztosítja a legnagyobb működési sebességet, ugyanis a processzor nincs leállítva, sem lelassítva a DMA adatátvitel alatt. A mikroprocesszor a memóriával dolgozhat a ciklusidő egyik felében (16.15. ábra), míg a DMA a ciklusidő másik részében. Természetesen ilyenkor a folyamat sebességét a memória sebességéhez kell hangolni. Rendszerint a periódusidő a memóriaciklus kétszerese.



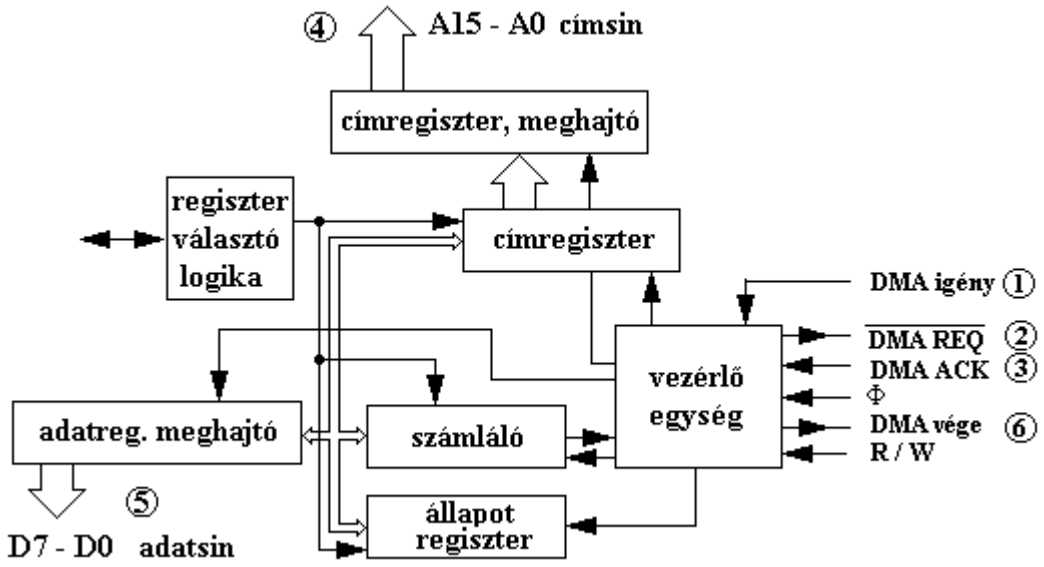
16.15. ábra: CPU/DMA adatátvitel multiplexálása

16.4.3. **A DMA műszaki megvalósítása**

Ahhoz, hogy létrejöjjön a DMA adatátvitel a következő feladatokat kell megoldani:

- a címbusz vezérlése, a címbusz mikroprocesszorhoz, illetve DMA vezérlőhöz való hozzárendelése,
- az adatbusz vezérlése,
- a memória címzése, a CPU a programszámláló segítségével címezi a megfelelő memóriarekeszt, mivel ugyanezt a feladatot a DMA-nak is el kell látnia, ezért a DMA is tartalmaz egy címzésre alkalmas regisztert, illetve a regisztert inkrementáló logikát,
- az átvitt adatok mennyiségének ellenőrzése, ez az átvitt adatblokk hosszának ellenőrzése, a folyamat befejezése,
- magának a DMA vezérlési módnak a meghatározása, amire egy parancsregiszter szolgál, itt található információ az átvitel irányára, arra, hogy aktív-e a DMA átvitel stb.

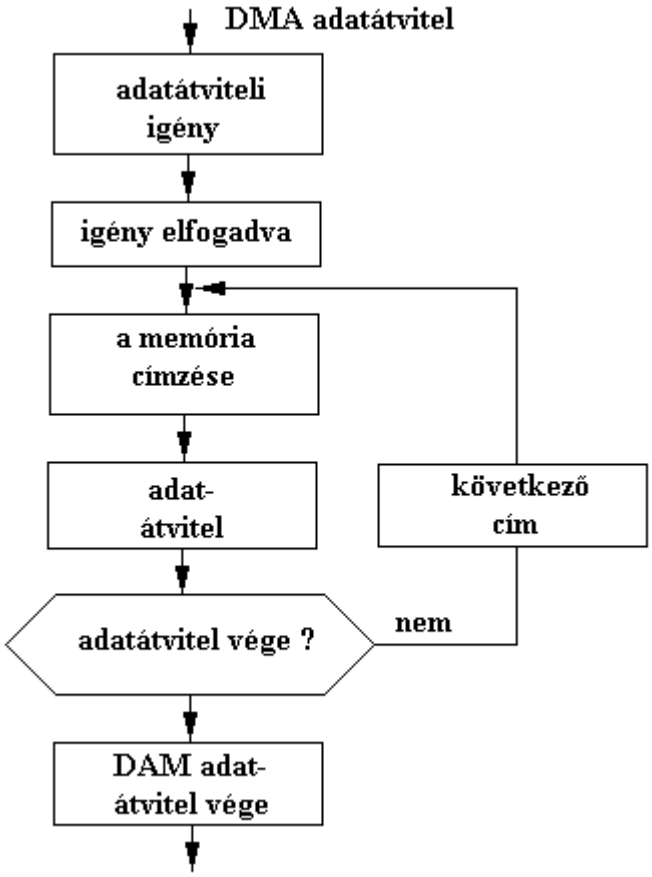
A 16.16. ábrán látható egy DMA vezérlő belső felépítése a fent felsorolt feladatok megvalósítására.



16.16. ábra: DMA vezérlő belső felépítése

A DMA adatátvitel megérthető a 16.16. ábráról, ahol az egyes lépések számokkal vannak ellátva, a folyamat a 16.17. ábrán látható folyamatábrán követhető nyomon. Az egyes lépések a következők:

1. az I/O eszköz logikai elemei létrehozzák a közvetlen memóriahozzáférés igényét,
2. az igényt továbbítja a logika a mikroprocesszor felé,
3. nyugtázójelet küld a DMA az adatátviteli igény elfogadásáról,
4. memóriacímzés,
5. adatátvitel a memória és a periféria között,
6. a DMA adatátvitel vége.



16.17. ábra: DMA adatátvitel folyamata

Ellenőrző kérdések

1. Milyen feladatokat kell megoldani az I/O eszközök és mikroprocesszorok közötti adatátvitel lebonyolításakor ?
2. Rajzolja le az adatátvitelt a különböző jelekkel együtt.
3. Az I/O eszköznek (periféria) milyen feladatokat kell megoldani adatátvitelkor ?
4. Miért szükséges az, hogy az I/O eszköz ideiglenes adattárolást végezzen adatátvitelkor ?
5. Miért kell címmel kiválasztani adatátvitelkor az I/O eszközt ?
6. Miért szükséges szinkronizálni a processzor és periféria közötti adatátvitelt ?
7. Milyen módon oldható meg adatátvitel processzor és I/O eszköz között ?
8. Ismertesse a programozott adatátvitelt.
9. Milyen három tipusa létezik a programozott adatátvitelnek ?
10. Ismertesse a feltétel nélküli programozott adatátvitelt, annak előnyeit és hátrányait.
11. Ismertesse a feltételes programozott adatátvitelt, annak előnyeit és hátrányait.
12. Ismertesse a feltételes, kiválasztásos programozott adatátvitelt, annak előnyeit és hátrányait.
13. Ismertesse a megszakítással kezdeményezett I/O adatátvitelt.
14. Ismertesse a megszakítással kezdeményezett I/O adatátvitel előnyeit és hátrányait.
15. Mi a maszkolás a megszakításoknál, mire használható a megszakítással kezdeményezett adatátvitelnél ?
16. Mi a prioritás a megszakításoknál, mire használható a megszakítással kezdeményezett adatátvitelnél ?
17. Hogyan történik a megszakításforrás azonosítása I/O adatátvitelnél ?
18. Hogyan történik a megszakításforrás azonosítása eszközök lekérdezése módszerrel ?
19. Hogyan történik a megszakításforrás azonosítása a vektoros megszakítás módszerrel ?
20. Mi a közvetlen memóriahozzáféréssel lebonyolított adatátvitel lényege ?
21. Ismertesse a mikroszámítógép-DMA egység összekapcsolását.
22. Mely módokon lehet megszervezni a DMA adatátvitelt a mikroszámítógépeknél ?
23. Ismertesse a DMA adatátvitel módszerét a processzor teljes leállítása módszerrel.
24. Ismertesse a DMA adatátvitel módszerét a cikluslopás módszerrel.
25. Ismertesse a DMA adatátvitel módszerét a CPU/DMA multiplexálás módszerrel.
26. Rajzoljon le egy DMA egységet és magyarázza el a DMA adatátvitel folyamatát a modellen.
27. Hasonlítsa össze a programozott, megszakítással kezdeményezett és DMA egységgel végrehajtott adtátviteli elveket, adja meg az előnyöket és a hátrányokat.

17. A/D ÉS D/A EGYSÉGEK ILLESZTÉSE MIKROSZÁMÍTÓGÉPEKHEZ

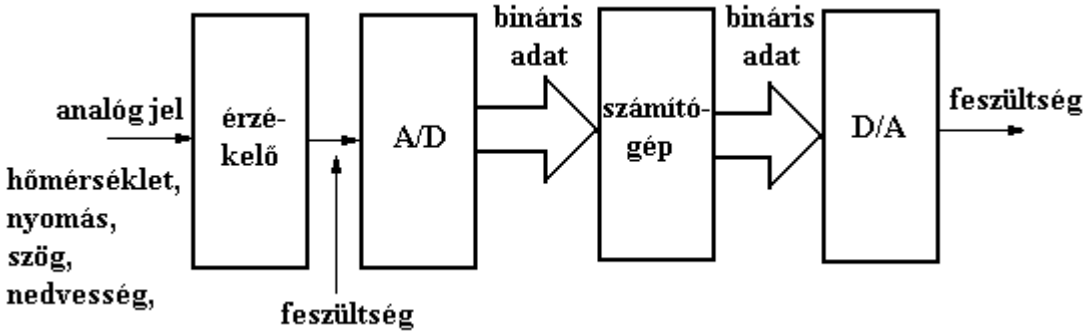
17.1. BEVEZETÉS

A feldolgozandó jelek egy része folytonos, analóg jellegű. A jelenleg legelterjedtebb számítógépek bináris formában ábrázolt adatokkal képesek dolgozni, így szükséges az adatbevitelnél az analóg-digitális, illetve az adatkihozatalnál a digitális-analóg jelátalakítás.

Az analóg jelek a legkülönfélébb fizikai mennyiségek lehetnek:

- hőmérséklet,
- nyomás,
- szőgelfordulás,
- nedvességtartalom,
- fényerő,
- stb

Ezeket a különböző fizikai mennyiségeket először is át kell alakítani feszültséggé, az A/D átalakítók ugyanis feszültség bemenettel rendelkeznek. A kapott feszültséget az A/D konverter bináris információvá alakítja át. A számítógép a számítógépes program és a bemeneti értékek alapján a külvilág felé binárisan kódolt értékeket küld, amit vissza kell alakítani a megfelelő fizikai jellé. Az átalakítás menete a 17.1. ábrán látható.



17.1. ábra: Fizikai mennyiségek átalakítása feszültséggé

Ebben a fejezetben az A/D és D/A átalakítókkal kapcsolatban csak a mikroszámítógépes rendszerbe való bekötéssel foglalkozunk, nem térünk ki az átalakítókkal kapcsolatos pontossági, átalakítási stb. problémákra.

A számítógépeknél használatos átalakítókat többféle ismérv alapján csoportosíthatjuk.

Az analóg-digitális jelátalakítás az alkalmazott technika szerint lehet:

- A/D átalakítóval és
- feszültség-frekvencia elven megépítve.

Az A/D, illetve D/A lehet az integráltsági fok szerint:

- beépített, a mikrovezérlő szerves része, vagy
- külön egység, melyet a rendszerbeépítés szabályai szerint kell illeszteni a mikrogéphez.

Az **A/D** lehet az alkalmazott átalakítási módszer szerint:

- a fokozatos közelítés elve szerinti (szukcesszív approximáció), vagy
- integráló típusú.

A **D/A** lehet az alkalmazott átalakítási módszer szerint:

- súlyozott ellenállásokkal és
- ellenállásos létrahálózattal megépítve.

Az A/D, illetve a D/A a mikrogéppel adatcserét végezhet:

- sorosan,
- párhuzamosan és
- kombináltan (sorosan néhány párhuzamos vezetéken keresztül).

Az átalakított, átalakítandó szóhosszúság szerint beszélhetünk:

- 8 bites,
- 10 bites,
- 12 bites és
- 16 bites átalakítókról.

A bemeneti, kimeneti analóg feszültség előjele szerint:

- csak pozitiv feszültségű és
- pozitiv és negativ feszültségű átalakító.

Az átalakított kód, az átalakítandó kód szerint lehetnek az átalakítók:

- binárisak,
- egyes komplemensűek,
- kettes komplemensűek és
- BCD átalakítók.

Az A/D elektromos szempontból tartalmazhat:

- 3 állapotú kimenetet, ami lehetővé teszi közvetlen illesztését a mikroszámítógép adatbuszára,
- nem tartalmaz 3 állapotú logikát, így szükséges közé és a mikrogép közé beépíteni egy háromállapotú logikát.

Az A/D átalakító referenciafesültségelehet:

- beépített,
- kívülről kell biztosítani és
- választható (beépített és külső).

Az A/D átalakító csatornaszáma:

- **egycsatornás és**
- **többcsatornás, ahol egy bemeneti multiplexer bináris vezérlés alapján valamelyik kijelölt csatornát rákapcsolja az átalakítás idejére az A/D-re.**

A fenti felsorolásból látható, hogy igen nagyszámú, a legkülönbözőbb paraméterekkel rendelkező átalakító található a piacon. Azt az esetet kivéve, mikor az átalakító a mikrovezérlő szerves része, a tervezőmérnöknek alaposan

elemezni kell a felhasználandó elem paramétereit. Az elemzésnél a következőket kell figyelembe venni:

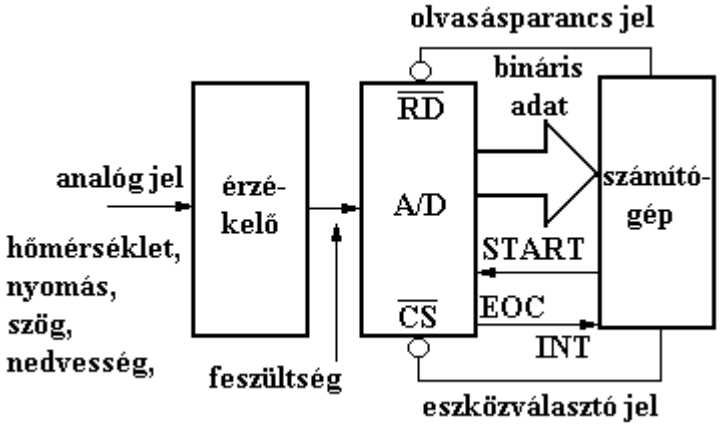
- az átalakítás sebessége,
- a bináris érték szóhosszúsága, felbontás,
- az átalakítás pontossága (ez összefügg az előző ponttal),
- az analóg feszültség tartománya,
- a zajok, zavarok kiszürése.

17.2. AZ ANALÓG-DIGITÁLIS ÁTALAKÍTÁS

17.2.1. Az A/D átalakítás sebessége

A mintavételezés sebessége gyakorlatilag megegyezik az átalakítási sebességgel, mértékegysége pedig az átalakítások száma másodpercenként.

Egy A/D eszköz általában mindig lassabban dolgozik, mint maga a processzor. Ezért van szükség a feladatok szinkronizálására. A 17.2. ábrán egy olyan A/D átalakító bekötését tárgyaljuk, amelynél az átalakítási odő hosszabb mint a processzor feldolgozási ideje. Ez nem konkrét típushoz kötött elemzés, hanem általános, így esetleg lehetséges, hogy bizonyos típusú A/D átalakítók esetleg más elnevezésű vezetékekkel rendelkeznek.



17.2. ábra: A/D eszköz bekötése a mikroszámítógépbe

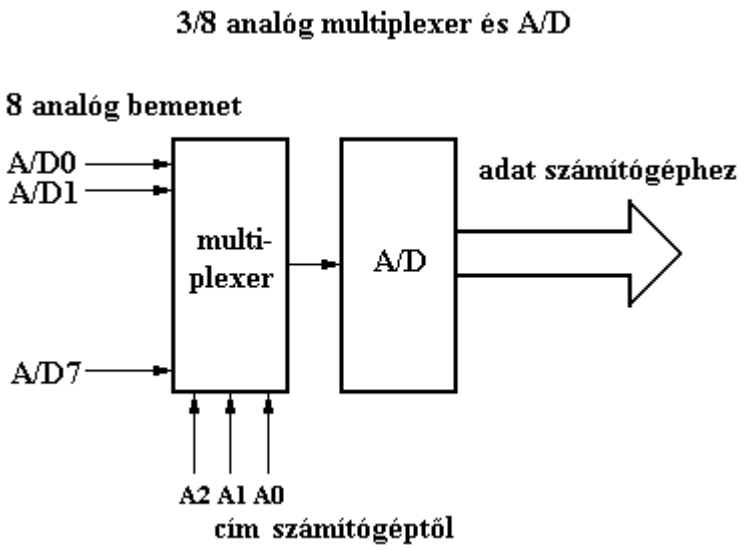
Az ábrán vázolt kötésnél a következő lépéseket kell a programmal létrehozni egy analóg jel beolvasásához:

1. **a program a** START **jel kiküldésével elindítja az** A/D **átalakítónál az átalakítási folyamatot,**
2. **utána pedig két módon folytathatja a program működését:**
   - **ciklusban várja az átalakítás befejezését jelző** EOC **jelet, vagy**
   - **engedélyezi a megszakítás elfogadását és folytatja a program végrehajtását,**
3. **az** A/D **átalakító a** START **jel után egy beépített mintavételező/tartó (Sample Hold) áramkörrel mintát vesz a jelből és ezt az értéket tartja az átalakítás alatt, az átalakító végzi az átalakítást,**
4. **az átalakító az átalakítás befejezésekor az** EOC **(End of Conversation) jel kiküldésével jelzi az átalakítás végét,**

5. **ami jel vételének alapján (ez ciklusban várakozáskor leolvasás, illetve megszakítás lehet, mint az ábrán látható) a processzor az A/D átalakítót mint perifériát a –CS jellel megcímezi, kiválasztja,**
6. **a –RD jellel pedig beolvassa a mikrogép megfelelő regiszterébe az átalakított értéket.**

Sokszor gyorsan változó jelből kell sok mintát venni. Vigyázni kell a többcsatornás A/D-k átalakítási sebességének meghatározásánál, ugyanis a bemeneti multiplexer mindig a címzett csatorna adatát továbbítja az A/D bemenetére, n számú bemenet $n * t$ idővel később címezi újból az adott bemenetet (17.3. ábra).

Nem érdemes az igényeket messze meghaladó átalakítási sebességgel rendelkező egységet választani a feladat megoldására, felesleges többletkiadást okoz a gyorsabb átalakító.



17.3. ábra: több bemenetes, multiplexeres A/D átalakító

A kétféle, alapjában különböző átalakítási elven működő A/D konverter lényegesen különböző értékekkel rendelkezik. A fokozatos közelítés elve szerint megépített átalakító a µs, 10 µs nagysegrendű sebességet éri el, míg az integráló típusú 20 ms-os átalakítási sebességgel dolgozik, ez azért van, mert szinkronizálja az átalakítást a hálózati frekvenciával a hálózati zavarok kiszűrésére.

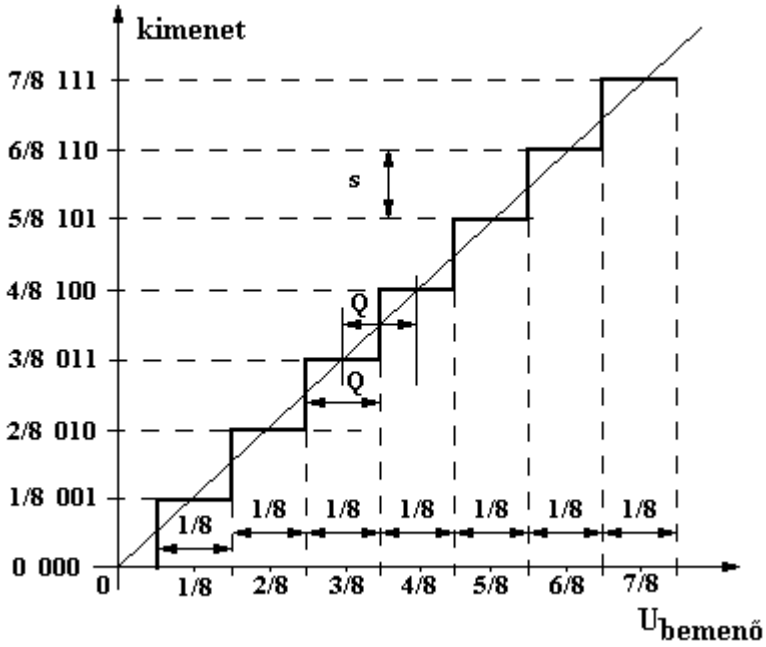
17.2.2. **Az analóg jel átalakítása, szóhosszúság**

Az átalakítás lényegét a 17.4. ábrán látható analóg-digitális átalakítási karakterisztikából érthetjük meg. A példában, az áttekinthetőség miatt három bites átalakítást láthatunk. ez azt jelenti, hogy a mikrogépbe három biten kódolva, tehát nyolc különböző szinten jut be a normalizált bemenő feszültség, bináris alakban (0 – 7 közötti értékek). A **Q** lépés a kívetkező képlettel számolható ki:

$$Q = \frac{U_{bemenet}}{2^{n-1}} \qquad (17.1)$$

ahol:

- Q – lépésköz,
- U – feszültség és
- n – az áatalakító bitszáma.



17.4. ábra: Analóg-digitális átalakötási karakterisztika

Az átalakító működését úgy érthetjük meg, hogy elképzeljük a feszültség folyamatos növekedését a bementen, ha **0 V** feszültségből indultunk ki. A 0 V-tól növekedő feszültség az A/D kimenetén **000** bitkombinációt ad ki mindaddig, míg a feszültség nem éri el a tartomány **1/8**-ának felét, amikoris **001** értéket ad, mindaddig, míg a bemenő feszültség nem éri el ataromány 1/8-ának másfélszeresét. A legnagyobb érték, **111** akkor kerül ki az A/D-ből, amikor a bemeneti feszültség nagyobb mint a legnagyobb érték –1/8 értékének a fele.

A fentiekből következik, hogy a **Q** felbontás egy bit hozzáadásával **Q/2** értékű lesz, vagyis minden bit megkétszerezi a felbontást.

Felmerül a kérdés, hány bites A/D átalakítót érdemes használni ? A számítógépes rendszerek 8 bitesek, illetve ennek egésszámú többszörösei, így a párhuzamos feldolgozás esetén nincs értelme a 8 bitnél kisebb szóhosszúság alkalmazásának. Ekkor 1/256, vagyis kb. 0.4 % pontosság érhető el. Amennyiben ez nem elég, kiválaszthatjuk a több-bites átalakítót, meghatározhatjuk az elérhető felbontást. A túl ngay felbontású A/D átalakító használata a zaj, illetve más zavarok miatt már elektronikai problémákat okoz, amiket meg kell oldani ahhoz, hogy a nagy felbontás használható legyen.

17.2.3. **Az analóg feszültség tartománya, normalizálás**

Az A/D átalakítók csak bizonyos fészültségek között tudnak átalakítani, ami lehet:

- 2.5 V,
- 5.0 V és
- 10.0 V.

Ezek az értékek rendszerint beállíthatóak. A bemenet analóg jelét az érzékelő átalakítja feszültséggé, ez a feszültség egy erősítővel kerül az átalakító bemenetére. Úgy kell az erősítést beállítani, hogy a bejövő jel az átalakító bemeneti feszültségének legnagyobb értéke közelébe kerüljön, ekkor lesz a legkisebb az átalakítás hibája.

Az A/D átalakítók a pontos átalakítás érdekében rendelkeznek egy ún. refereciafeszültség értékkel. Ez lehet az eszköz része, de sok típusnál kívülről is rákapcsolható az átalakítóra.

17.2.4. A**z érzékelők karakterisztikájának hatása az A/D átalakításra**

A fizikai mennyiségeket elektromos feszültséggé átalakító érzékelők karakterisztikája lehet:

- lineáris, vagy
- nemlineáris.

A nemlineáris karakterisztika lineárissá tétele lehet:

- elektronikával, vagy
- szoftverrel.

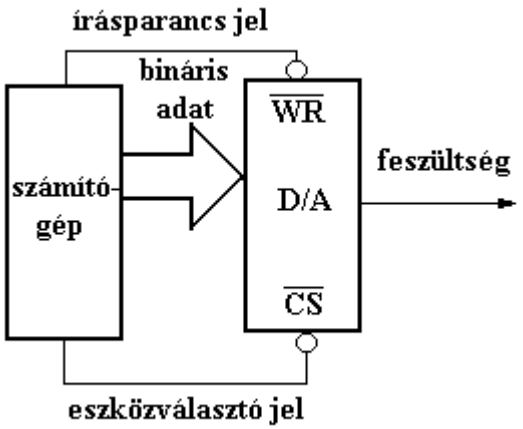
A szoftveres linearizálás két módon oldható meg:

- képletek alaklmazásával, ez egyszerübb utasításkészlettel rendelkező mikrokontrollerekkel (ahol nincs pl. szorzás-osztás), eléggé problémás,
- táblázattal, ahol a táblázat címe a bemenő érték, a táblázat adott helyén levő adat pedig a linearizált adat.

17.3. A DIGITÁLIS ANALÓG ÁTALAKÍTÁS

Az A/D egységnél ismertetett és D/A átalakítónál is meglevő problémákat ebben a fejezetben nem ismételjük meg, csak a különleges D/A igényekre térünk ki.

A D/A átalalító a rendszerbe szintén mint periféria kerül beépítésre, tehát címe van. Nem kell hogy tartalmazzon 3 állapotú logikát. A 17.5. ábrán látható a blokksémája a D/A átalakítónak.

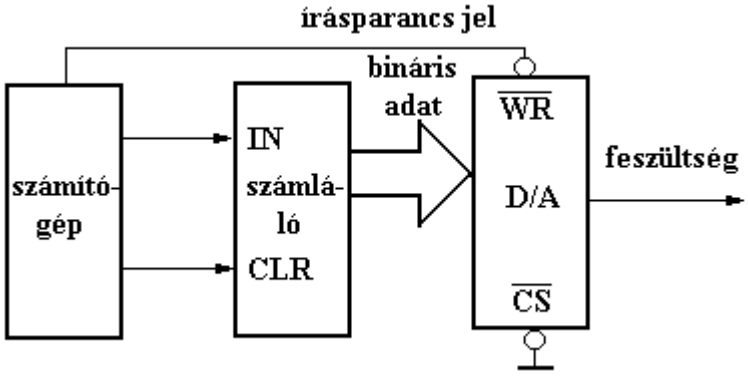


17.5. ábra: D/A átalakító blokksémája

A **D/A** a következőképpen működik:
a számítógép kiírja a 8 bites párhuzamos adatot az adatsinre, ezután az eszköz címével kiválasztja a **D/A** átalakítót a **–CS** jellel, majd a **–WR** írásparanccsal beírja az adatot a **D/A-**ba, aminek hatására megjelenik az analóg feszültség a kimeneten.

Sokszor a mikrovezérlőknek nincs elegendő kivezetésük, ekkor célszerű a soros átalakítókat használni.

Ha csak párhuzamos D/A átalakítónk van, ugyanakkor korlátozott a kivezetések száma, akkor valamilyen TTL, vagy CMOS áramkör közbeiktatásával is megoldhatjuk a problémát. Ez lehet egy soros beírású és párhuzamos kimenetű regiszter, de akár egy bináris számláló is. Egy bináris számlálóval megvalósított kapcsolás látható a 17.6. ábrán. Természetesen ilyenkor a feladatból a szoftverre hárítottunk többletfeladatot.



17.6. ábra: Párhuzamos D/A bekötése számlálóval, három vezetékkel

A kapcsolás a következő módon működik:

1. a vezérlő a **CLR** (Clear) lábon át törli a számláló tartalmát,
2. az **IN** soros bemeneten keresztül annyi impulzust küld a számlálóba, amennyit annak a kimeneten tartalmazni kell,
3. végül elfogadtatja a D/A-val az értéket.

Ellenőrző kérdések

1. Ismertesse a fizikai analóg jelek átalakítását bináris jellé, valamint a bináris jelek visszaalakítását analóg jellé blokksémán .
2. Milyen eljárásokat ismer analóg jelek digitálissá való átalakítására ?
3. Ismertesse a lehetséges D/A átalakítási technikákat .
4. Rendszerint milyen szóhosszúságú A/D és D/A átalakítókat hsználunk ?
5. Milyen digitális kódokat használnak a digitális átalakítók ?
6. Mi a referenciafeszültség és hogyan lehet előállítani ?
7. Hány csatornés lehet az A/D átalakító ?
8. Az A/D átalakító kiválasztásánál milyen szempontokat kell elemezni ?
9. Hogyan határozzuk meg az A/D átalakítás sebességét ?
10. Ismertesse a számítógéphez kapcsolt A/D működését .
11. Ismertesse a többcsatornás A/D hardver megoldását .
12. Mi az A/D és D/A felbontása és higyan határozzuk meg ?
13. Mi az A/D és a D/A szóhosszúsága és mit határoz meg ?
14. Rajzolja le és magyarázza meg az analóg-digitális átalakítás karakterisztikáját .
15. Határozza meg a szóhosszúság és pontosság közötti összefüggést .
16. Mi az A/D feszültségtartomány és mi a normalizálás ?
17. Az érzékelőknek milyen karakterisztikája van ?
18. Hogyan lehet korrigálni a nemlineáris érzékelőkarakterisztikát ?
19. Rajzolja le a D/A átalakító és számítógép kapcsolatát .
20. Ismertesse a D/A működését .
21. Hogyan lehet kevesebb mikrovezérlő csatlakozóponttal párhuzamos eszközt csatlakoztatni a vezérlőhöz ?

18. A MIKROSZÁMÍTÓGÉP, MIKROVEZÉRLŐ PROGRAMJÁNAK ÍRÁSA

18.1. BEVEZETÉS

Ahhoz, hogy hatásos, gyorsan és hibátlanul működõ programot írjunk egy mikroszámítógépre, el kell sajátítani bizonyos, ezekre a tipusú gépekre vonatkozó szabályokat. A különféle magasszintű programozási nyelveknél megismert technikák ismerete segíthet a gépi programozásnál. A programfejlesztés lépései a következők:

1. a feladat pontos, részletes megfogalmazása, megértése,
2. a különféle megoldások megtalálása, tehát a megoldás **algoritmusának** meghatározása,
3. a műszaki- és egyéb feltételeket legjobban kielégítõ megoldás kiválasztása,
4. az algoritmus valamilyen grafikus ábrázolása, pl a **folyamatábra**,
5. egyszerű adatokkal a folyamatábra **kézi tesztelése**,
6. a **részletes folyamatábra** felrajzolása,
7. annak **kézi tesztelése**,
8. a kódolás (**programírás**),
9. a program tesztelése **szimulátoron**,
10. a valós rendszeren való tesztelés,
11. a programdokumentáció teljessé tétele.

Látszólag a fent vázolt hozzáállás bonyolult, a gyakorlat azonban azt mutatja, hogy hatásos gépi programok csak a fenti módszerrel írhatók. Az azonnal programot író programozók igen sokszor a nem teljesen elemzett feltételek miatt rosszabb, hosszabb ideig íródó programokat írnak, kihagyva még a rendszer dokumentálását is.
A magasszintű programozási nyelvekkel szemben (**Pascal, C++, Prolog** stb.) a programíráskor rendszerspecifikus ismeretekre is szükség van. Ellentétben a magasszintű programozási nyelvekkel, ahol nem kell tudni az adatok fizikai helyét a memóriában, itt szükséges pontosan ezt ismerni egyéb hardverinformációk mellett. A **mikroszámítógép, mikrovezérlő** hardverjének pontos és részletes ismerete teszi lehetõvé azt, hogy a **mikrogép** tulajdonságait a lehetõ legjobban használjuk ki, pl. a lehetõ legnagyobb működési sebességet érjük vele el. Ott, ahol viszont a magasszintű programozási nyelv használata jár elõnnyel, feltétlenül élnünk kell vele, gondoljunk csak egy mátrix-szorzásra, ami megvalósítása Pascal nyelven egyszerű feladat, gépi programozással rendkívül nehézkes.

18.2. A PROGRAMOZÁSHOZ SZÜKSÉGES HARDVER ISMERETEK

Egy számítógép, legyen az nagy gép, vagy akár mikrovezérlõ mindig a **hardver** és **szoftver** tökéletes együttműködésekor válik hatásos eszközzé. Ebben a fejezetben ugyan programozással foglalkozunk, de ehhez bizonyos minimális hardverismeret is szükséges. Az előző fejezetekben már megismerkedtünk a legfontosabb hardverelemekkel, most itt azoknak egy a szoftver írásához szükséges tárgyalását találjuk.

18.2.1 **A mikroprocesszor**

A 18.1. ábra tartalmazza a legfontosabb mikroproceszorelemeket, amiket a programozónak ismernie kell.

A	jelzőbitek
B	C
D	E
H	L
SP - veremtár mutató	
PC - programszámláló	

A - akkumulátor

18.1. ábra: A mikroprocesszor felépítése a programozó számára

Hangsúlyozzuk, hogy a különféle mikroprocesszorok, mikrovezérlők felépítése különböző, a működési elvek azonban ugyanazok, habár más és más azok belső felépítése. Az ábrán szereplő megoldás egy lehetséges belső felépítést mutat.

- **A - akkumulátor** (18.2. ábra), központi szerepe van, gyakorlatilag az összes aritmetikai, logikai müvelet elvégzésekor használjuk, az eredmény is a leggyakrabban ide kerül vissza, sokszor a ki-/bemenő műveletek is ezt a regisztert használják. Szóhosszúsága egyben a gépre jellemző adat, pl. lehet 8 bites, 16 bites stb., ami egyuttal a memóriára jellemző adat is. A benne, illetve ezen keresztül végrehajtott műveletek sokszor használják a jelzőbíteket (indikátorokat), akár bemenő-, akár kimenő eredmények jelzésére.

MSB LSB

D7	D6	D5	D4	D3	D2	D1	D0

18.2 ábra: Az **A** akkumulátor

- **Jelzõbitek (indikátorok)**, főleg az aritmetikai műveletek végzésénél használatosak, de más, pl. logikai műveletekben is kaphatnak szerepet. A különböző mikroprocesszorok más és más indikátorokat használnak, de néhány közülük minden tipusnál megtalálható. Főleg az **akkumulátor** műveletek állítják ezek értékeit, de más regiszterekhez is kapcsolódhatnak:

- **C - átvitel** (Carry) értéke 1 lesz olyankor, amikor pl. összeadásnál az eredmény már nem tárolható az adott regiszterben.
- **V - tulcsordulás** (Overflow) bit jelzi a kettes komplemensű összeadásnál azt, hogy a kapott eredmény nagyobb mint a számábrázolási tartomány (hibás, nem értelmezhető eredmény).
- **N - negativ eredmény**, az elvégzett aritmetikai művelet után negativ szám keletkezett. Ez egyuttal az **MSB** (D7) bit.

- **Z - nulla eredmény**, ha egy elvégzett aritmetikai művelet eredménye 0, akkor **Z = 1**, bármely más számnál **Z = 0**.
- **H - BCD átvitel** (Half Carry), hasonló a **C** indikátorhoz, de nem 8 bit után, hanem 4 (**D3**) után keletkezik. Az ún. **BCD** aritmetikánál van szerepe.
- **P - páros szám,** jelzi, hogy a kapott eredmény páros (**P = 1**), illetve páratlan (**P = 0**). Ez nemcsak az előjel nélküli és az előjeles számokra érvényes, hanem a kettes komplemensre is. Valójában az **LSB** (D0) értékével egyezik meg.
- **I megszakítás (interrupt) indikátor**, jelzi a hardver vagy szoftver úton fellépő megszakításkérést.

Gyakorlati megfontolásokból az indikátor biteket egy regiszterben fogják össze, de ezeknek a biteknek mind egyenként van csak értelmük.

- **Általános célú regiszterek**, számuk gépenként különbözik, sokban hasonlítanak az akkumulátorra, de nem annyira általános a szerepük, vagyis nem minden akkumulátoron is végrehajtható utasítás végezhető el rajtuk. A példában ezek a **B, C, D, E, H** és **L** regiszterek. Sokszor, mindamellett hogy ezek 8 bites regiszterek, kettő összekapcsolásakor 16 bites regiszterként is használhatók, ami lényegesen gyorsítja a nagyobb számokon történő műveletvégzést. Itt a **BC**, **DE** és **HL** regiszterek lehetnek 16 bitesek, más összekapcsolásuk nem engedélyezett (például **B** és **E** összekapcsolása nem lehetséges).

- **PC - programszámláló** (program counter), 18.3. ábra. Minden mikroszámítógépen belüli elem (legyen az memóriarekesz vagy ki-/bemenő egység) saját, egyedi cimmel rendelkezik. A memóriában levő program (utasítások sorozata) is fizikailag pontosan meghatározott helyen helyezkedik el. A programszámláló (**PC**) mindig az aktuális, vagyis végrahajtandó utasítás címét adja meg. Nagysága **8 bites** gépeknél általában **16 bit**, ez lehetõvé tesz egy **64 kB** nagyságú memórahasználatot.

MSB LSB

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

18.3. ábra: A **PC** programszámláló (utasításszámláló)

- **SP - stack pointer**. Magyarul fordítják veremtár-mutatónak, de zsákmemória mutatónak is. Nem keverendő össze a **stack memóriával**, a **SP** ebben a memóriában meghatároz egy helyet. Szerepe főleg az alprogramok visszatérési címeinek tárolásában van, de használható még adatok ideiglenes tárolására is. Különböző gépeknél különböző stack memória megoldások vannak, ettől függ a **SP** kialakítása is. Ha a stack memória a mikroprocesszor része, akkor nagysága viszonylag kicsi, néhány tíz-száz bájt, ekkor a **SP** is rövid szóhosszúságú regiszter, pl. 8 bites. Ha a stack memória nem a mikroprocesszor része, hanem a memória, illetve az adatmemória része, akkor jóval nagyobb lehet a kapacitása, ilyenkor a **SP** is értelemszerűen nagyobb méretű. A fenti modellen (18.3. ábra) ez egy 16 bites regiszter.

18.3. A MIKROPROCESSZOR, MIKROVEZÉRLŐ PROGRAMOZÁSÁNAK SZOFTVER SEGÉDESZKÖZEI

A mikroprocesszor, mikrovezérlő programját gépi kódban kell a memóriába betölteni, ennek előállítása közvetlenül gépi kódban nagy nehézségekbe ütközik. Gépi kódú programírás csak nagyon rövid, néhány soros programoknál indokolt.

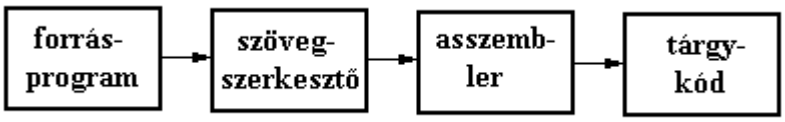
Sokféle segédeszköz (szoftver támogatás) áll a programozók rendelkezésére egy program kifejlesztésekor. Különböző mikroprocesszor gyártók eltérő rendszereket kínálnak, bár az alapelveket tekintve hasonló ezen segédprogramok felépítése.

Az ún. asszembler (assembly) szintű programozás biztosítja a hardver képességek jobb kihasználtsági fokát, de igen elterjedt magasszintű programozási nyelvek használata is, mint pl a **C** nyelvé. Mivel a különféle szoftverek egyforma, szabványosított gépi programot eredményeznek, sokszor célszerű többféle programozási nyelv használata. Ilyenkor a bonyolultabb programok (pl. mátrix-számítás) könnyebben oldható meg C-ben, a megszakításkezelés asszemblerben, ebben az esetben a mátrix-számítási részt C-ben, a megszakításkezelést pedig asszemblerben írjuk meg, a kapott gépi kódokat pedig összefüzzük.

Nagyon hasznos, sokszor nélkülözhetetlen segédeszköz a szimulátor, ami a megirt programot lépésenként, blokkonként engedi végigfuttatni, kijelezve minden regiszter- és indikátor bit tartalmat. Ezek a segédprogramok, asszemblerek, szimulátorok a mindenütt megtalálható PC személyi számítógépeken futtatható szofverek.

18.3.1. **Az asszembler**

A mnemonikokkal (szimbolikus utasításokkal) megírt program fordítási eljárása (gépi kód előállítása) a 18.4. ábrán szerint történik.



18.4. ábra: Az asszembler fordítás menete

Az assembly programnyelv a processzor utasításaira épül. A műveleti kód helyett **mnemonikok,** memória abszolút címek helyett szimbólumok, **címkék** használhatók.

Néhány, nem végrehajtható utasítás - **direktíva** - segíti a programozót a programnyelv írása közben. Az assembly nyelvű programsort a következő alakban kell megadni (18.5. ábra):

sorszám cimke utasítás operandus megjegyzés

18.5. ábra: Az assembly nyelvű programok alakja

Írás közben a következő szabályokat kell tiszteletben tartani:

- egy műveletet egy sorban kell megadni,
- a sor alakja kötött, az egyes mezők jelentése a következő:
  - sorszám - egyes asszemblereknél megadja a sor helyét a programban, általában nem használatos,
  - cimke - szimbolikus hely a programban, általában 6-8 karakter, az első karakter mindig betű,
  - utasítás - a végrehajtható utasítások és direktivák mnemonikus kódja,
  - operandus - értelmezése a műveleti kódtól függ, van amikor nincs értéke és
  - megjegyzés - tetszőleges szöveget tartalmazhat, általában valamilyen jellel kezdődik, pl. ";".

18.3.2. **A C-32 asszembler**

Az itt ismertetésre kerülõ **C-32** (Cross-32, keresztasszembler) valamivel egyszerűbb mint a kifejezetten csak egy mikroprocesszorhoz, mikrovezérlőhöz kifejlesztett asszemblerek, de jó tulajdonságai közé tartozik az, hogy egy egyszerű táblázatra való hivatkozással különböző 8-, 16- és 32 bites processzorokhoz használható. A legfontosabb tulajdonságai a következők:

- 2 menetes fordítás (two-pass cross-assembler), szükség esétén harmadik menet,
- mind a bemenő-, mind a kimenő adattárak (fájlok) ASCII kódban íródnak, sor végén CR és LF,
- az ASCII kód 0..127 értékű,
- soronként egy processzorutasítás vagy direktiva lehet,
- minden cimke után ':' (kettőspont) kell tenni,
- minden cimkét, kifejezést és operandust 32 bites előjeles számként tárol, ill. végez rajtuk műveletet,
- nem különbözteti meg a nagy- és kis betűket,
- a cimkék, utasítások tetszőleges hosszúságúak, egy sor legfeljebb 255 karaktert tartalmazhat.

Bármelyik gépre egyszerűen felmásolható, a C-32.exe asszembler, a táblázat pedig 8051.tbl nevű adattárban található.

Indítása a következő módon történik:

- C32 forrásprogram [-lista] [-hexkód], ahol a [] zárójelben levő kimenő fájl opcionális.

Legyen a forrásprogram neve MINTA.ASM, kérjük a LISTA.LST és HEXA.HEX fordítást, ekkor a C32 MINTA.ASM -LISTA.LST -HEXA.HEX alakban kell indítani az asszemblert. A *.LST fájl kibővíti a *.ASM fájlokat a hexadecimális címekkel és műveleti kódokkal, míg a *.HEX fájl tartalmazni fogja a 8051 (ill. az EPROM) számára a hexadecimális kódokat, amit az EPROM programozó felismer és beír az EPROM-ba.

A számrendszerek használatának szabálya a következő:

- a szám után semmi, vagy D betű, akkor az tizes számrendszer,
- a B betű a kettes számrendszerre utal, míg
- a H a hexadecimális számokra utal.

A hexadecimális beírás megköveteli az A,B,C,D,E és F számok elött a 0-át, pl. 0D3F9, a D3F9 azért hibás, mert ezt címkének értelmezi a fordító.

18.3.2.1. Néhány gyakran használt direktiva

CPU - A mikroprocesszor tipusának megadása,

Cimke: CPU "8051.TBL" ;a cimke nem szükséges, a pocesszor 8051 lesz.

DFB - Define Byte, egy bájthoz értéket rendel, alakja:

Cimke: DFB kifejezés1,kifejezés2,...,kifejezés(n) ;a kifejezes1 az 1. bájtba, a ;kifejezés2 a 2. bájtba stb. kerül.

END - End of Source Program, a forrásprogram vége, a prografájl fizikai végére kell írni.

EQU - Equate Label, cimkéhez rendel egy értéket, alakja:

Cimke: EQU kifejezés ;a kifejezés lehet szám, másik cimke, vagy egy +,-,* és / művelettel megadott egész szám.

HOF - Hexadecimal Output Format, hexadecimális kimenő fájl formátum, itt INTEL 8, aminek alakja:

Cimke: HOF "INT8" ;8 bites INTEL alak.

ORG - Program Counter Origin, a program címének beállítására szolgál, alakja:

Cimke: ORG kifejezés ;hol kezdődik a program, egy programon belül lehet több helyen is, de a ;programrészek nem fedhetik át egymást, ez hibajelzést eredményez.

Hibajelzést, vagy üzenetet küld az asszembler minden esetben amikor nem tudja értelmezni a forrásprogramot.

A C-32 nem állít elõ összefűzhető programmodulokat, csak végrehajtható gépi kódot. Olyan esetben amikor már megírt programot v. programrészt szeretnénk használni, akkor a szövegszerkesztőbe kell bemásolni ezt a részt, egy egységes programot képezve.

Ellenőrző kérdések

1. Az assembly szintű (gépi kódhoz közelálló szimbolikus nyelv) programok írásakor használható-e a magasszintű programozási nyelveknél elsajátított technika ?
2. Sorolja fel a programfejlesztés lépéseit .
3. Miért fontos a programfejlesztés során a feladat pontos megfogalmazása, megértése?
4. Mi az algoritmus ?
5. Mi a folyamatábra és mi a kapcsolata az algoritmussal ?
6. Miért célszerű a folyamatábra tesztelése ?
7. Mi a különbség a globális és részletes folyamatábra között, és mi köti össze őket ?
8. Mi a szimulátor, milyen előnyöket nyújt használata a programfejlesztés során ?
9. Mi a programdokumentáció, hogyan kell elkészíteni és milyen előnnyel jár használata ?
10. Az assembly szintű programok írásakor ellentétben a magasszintű programozási nyelvekkel milyen különleges rendszerismeretekre van szükség ?
11. Miért szükséges és célszerű részletesen ismerni a gép hardver paramétereit assembly programok írásánál ?
12. Mikor előnyös a gépi programok használata ?
13. Mikor előnyös a magasszintű nyelveken írt programok írása ?
14. Ismertesse egy mikroprocesszor, mikrovezérlő legfontosabb hardverelemeit programozás szempontjából .
15. Mi az A akkumulátor, mi a szerepe a mikrogépben ?
16. Mik a jelzőbitek, milyen feladatuk van ?
17. Mi a C átvitelbit ?
18. Mi a V túlcsordulásbit ?
19. Mi az N negativ eredmény bit ?
20. Mi a Z zerobit ?
21. Mi a H félátvitelbit ?
22. Mi a P páros szám bit ?
23. Mi az I megszakításbit ?
24. Mik az általános célú regiszterek ?
25. Mi a PC programszámláló regiszter és mi a feladata ?
26. Hogyan számítható ki, hogy egy adott szóhosszúságú PC programszámláló mekkora memóriát címez ?
27. Mi az SP veremtár-mutató ?
28. Mi a stack memória (veremtár) ?
29. Ismertesse mely két megoldást alkalmazzák a veremtár kialakítására ?
30. Ismertesse a hardver megoldású veremtárat, annak előnyeit és hátránayit .
31. Ismertesse a szoftver megoldású veremtárat, annak előnyeit és hátrányait .
32. Ismertesse, hogy hogyan lesz a forrásnyelvű programból gépi kódú program.
33. Ismertesse egy assembly nyelvű program egy sorának felépítését.
34. Hogyan kell használni a C32 asszemblert, adja meg a parancssor leírását.
35. Hogyan kell megadni asszemblerben a különböző számrendszerekben az adatokat.
36. Mi a direktiva, miért szükséges használata az asszemblerben ?
37. Ismertesse a CPU direktívát és szerepét.
38. Ismertesse a DBF direktívát és szerepét.
39. Ismertesse az END direktívát és szerepét.
40. Ismertesse az EQU direktívát és szerepét.
41. Ismertesse a HOF direktívát és szerepét.
42. **Ismertesse az ORG direktívát és szerepét.**

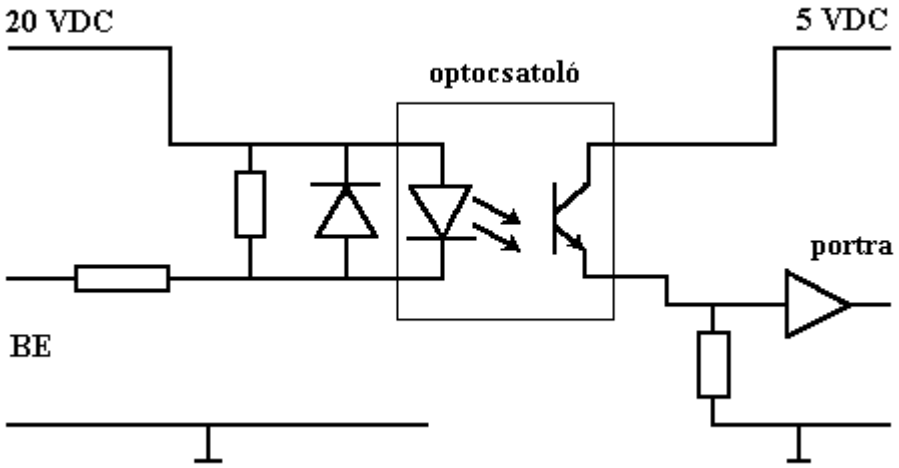
19. KÜLÖNBÖZŐ JELEK ILLESZTÉSE A MIKROSZÁMÍTÓGÉPHEZ, MIKROVEZÉRLŐHÖZ

19.1. BEVEZETÉS

Minden számítógépes rendszer feladata az, hogy bizonyos bemenő jeleket beolvasson, azokat feldogozza majd kimenő jelek formájában visszaküldje az eredményt az adott rendszerbe (emberhez, számítógépbe és folyamatba).

19.2. BEMENETI ELEMEK

A különböző készülékek gyártói többnyire ún. galvanikusan leválasztott bemeneti kialakitásokat fejlesztettek ki. Ennek az az előnye, hogy bemeneti rövidzár, vagy túlfeszültség esetén csak a bemeneti fokozat károsodik, a nagyfeszültségű jel nem tud bejutni a mikroszámítógépbe, mikrovezérlőbe. Ezeknek a megoldásoknak másik előnye, hogy a zavarjeleket tartalmazó bemenő jelek sem tudnak bejutni a rendszerbe. A 19.1. ábrán egy optocsatolós bemeneti fokozat kapcsolási rajza látható. A bemenetet



a **BE** felirat jelöli, itt egy rövidrezárt érintkező (kapcsoló, nyomógomb) áramot enged át az optocsatoló fénydiódáján, aminek hatására a tranzisztor vezetni kezd, 5 V feszültség kerül a processzor, vezérlő megfelelő bemeneti pontjára. Bemeneti jeladóként potenciálmentes érintkezők használhatók.

19.1.ábra: Optocsatolós bemeneti fokozat

A mikroprocesszoros, mikrovezérlős rendszerekben a következő tipikus jeladókat használhatjuk:

- mikrokapcsoló,
- végálláskapcsoló,
- fotoelektromos kapcsoló,
- közelitéskapcsoló,
- szintmérő,
- hőmérséklet-mérő,
- jelfogó-vagy mágneskapcsoló,
- peremkerekes kapcsoló (BCD-kódban),

- analóg kimenetü jeladók.

Különböző jeladókat különböző feladatok megoldására használunk. Egy projekt optimális megtervezéséhez ajánlatos a gép (technológia) vezérlendő berendezésén (készülékein) elhelyezendő jeladók megfelelő sajátosságaival megismerkedni.

19.2.1 **Mikrokapcsolók és végálláskapcsolók**

A mikrokapcsoló egy rugós kapcsolóelemből áll, amely egy műanyag házban foglal helyet. Ez a konstrukcó rendkivül preciz, kis erőt igénylő kapcsolásokat tesz lehetővé. A mikrokapcsoló tartalmazhat záró, vagy bontó érintkezőt, esetleg mindkettőt, sokszor ezekből többet is.

Konstrukciós felépitésüket tekintve a mikrokapcsolók a következők:
- báziskapcsolók,
- kiskapcsolók,
- miniatűr kapcsolók,
- Reed-jelfogós kapcsolók és
- szilárdtest-mikrokapcsolók.

A végáláskapcsollókat gyakran tokozott mikrokapcsolónak is nevezik. Ez az elnevezés abból ered, hogy nagyon sok végálláskapcsoló kapcsolóelemként valamilyen mikrokapcsolót tartalmaz.

19.2.2. Fotoelektromos kapcsolók

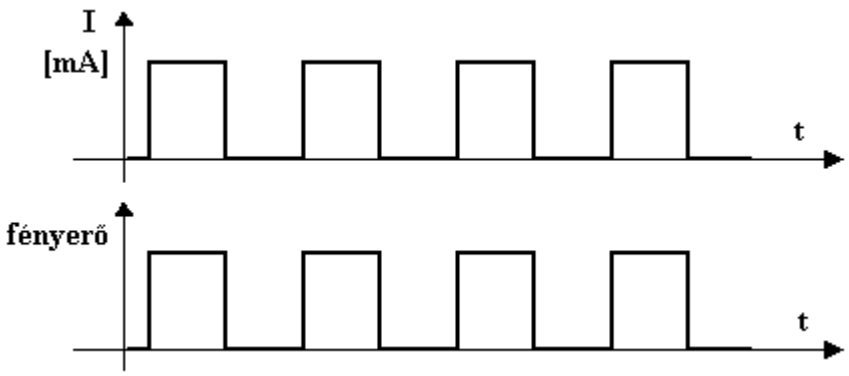
A fotoelektromos kapcsolók érintésmentesen képesek információt adni egy objektum állapotáról. Ez az érintésmentes letapogatás gyorsabb mint az érintkezővel megvalósított, valamint amiatt, hogy nincs mechanikus, mozgó alkatrész, megbízhatóbb is. Nem számít az objektum anyaga sem.

A fotoelektromos kapcsolókat a következő csoportokba sorolhatjuk:
- egyutas fénysorompó,
- viszaverődéses tipusú és
- diffúz reflexiós tipusú.

19.2.2.1 Egyutas fénysorompó

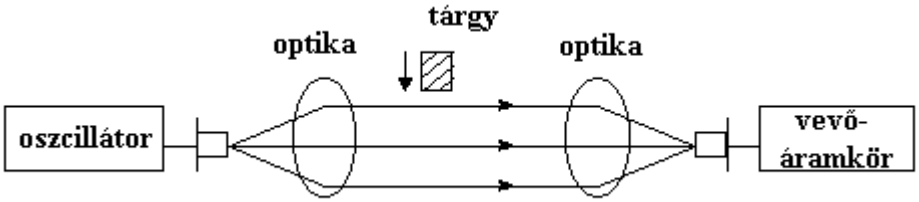
Az egyutas fénysorompó egy fényadóbol és -vevőből áll. Mivel a környezetben sok zavaró fény található, amely esetleg befolyásolná a kapcsoló működését főleg a kevésbé zavarérzékeny infravörös jeladókat használjuk.

Az adóként általánosan használt integrált áramkör egy oszcillátor (astabil multivibrátort) tartalmaz. A fényforás gallium-arzenid (GaAs) fénykibocsájtó dióda (LED). Ezt az oszcillátor-áramkör váltakozó áramú jellel modulálja (19.2. ábra).



19.2. ábrán: Modulált fénysugár

Az adóoldali lencsén a homogén fényáram nagy úthosszt biztosít. A **LED** a lencse gyújtópontjában helyezkedik el. A vevőoldli lencse gyújtópontjába kerül a fotodióda, vagy fototranzisztort. Ez az érzékelő a fény villamos jellé alakitja át, amit aztán erősíteni kell, majd továbbadható a kapcsolóáramkör felé. A kapcsolóáramkörbe célszerű **Schmitt-triggert** beépíteni, ami a bejövő jel egy adott szintjénél kapcsol be,



illetve csak akkor kapcsol ki, ha a bejövő jel a bekapcsolási szint alatt van egy bizonyos értékkel. Az infravörös fénysorompón belül egy szűrő is van, amely a látható fényt elválasztja az infravöröstől. A fénysorompó felépítése a 19.3. ábrán látható.

19.3. ábra: Egyutas fénysorompó

19.2.2.2 Visszaverődéses fénysorompó

Visszaverődéses (retroreflex) tipuson egy olyan optikai kapcsolót értünk, amelynél a fényadó és -vevő egy tokozatban foglal helyet. A kibocsátott fény egy reflektorról viszaverődik,és az igy visszaverődött fényt a fényérzékeny áramkör felfogja.

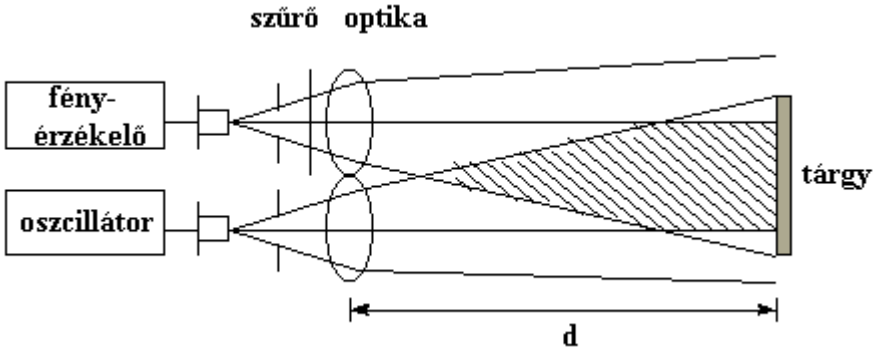
Kétféle változatot különböztetünk meg :

- az adó és a vevő közös optikát használ, ebben az esetben a fényemittáló diódát a vevőlencse közepére szerelik
- az adó és a vevő elválasztott optikával működik. Mindkét lencse elhelyezése párhuzamos. A LED és a fototranzisztor a megfeleő lencsék gyujtópontjába vanak helyezve.

19.2.2.3 Diffúz reflexiós típusú fénysorompó

Ez a tipusú fénysorompó úgy működik, hogy a kibocsátott fénysugarat a letapogatandó tárgy visszaveri. Nagy előnye a diffúz reflexiós fénysorompóknak, hogy sem a vevő, sem a reflektor nem igényel helyet. A berendezés hátrányos tulajdonsága, hogy aránylag kicsi az átfogási távolság (0,5-1 m).
Ennek a tipusnak is kétféle változata van:

- az adó és a vevő egyetlen közös optikával rendelkezik. A fénykibocsátó diódát a vevőlencse közepébe épitik be, és igy mindkét optikai tengely egybesik.
- az adó és a vevő egymástól elválasztott optikával rendelkezik. A 19.4. ábrán egy ún. kétoptikás diffúz-visszaverődéses tipusú fénysorompó látható. Az optikai tengelyek egymással párhuzamosak, és az ábrán bevonalkázott terület az átfogható tartomány. Azok a tárgyak, amelyek ezen kivül esnek -(d)-, nemérzékelhetők a fotokapcsolóval.

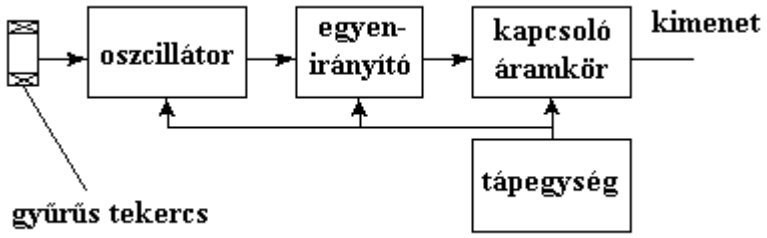


19.4. ábra: Kétoptikás diffúz-visszaverődéses tipusú fénysorompó

Mindkét változat működési elve hasonló. Egy oszcillátor egy meghajtóáramkörön keresztül modulálja a GaAs fénykibocsájtó diódát. Az elektronikának része egy érzékenységbeállító potenciométer. A vevőrész csak a meghatározott frekvenciájú jeleket veszi, így teljesen kizárt a zavaró jelek hatása. A mérési tartomány (letapogatási tartomány) függ a letapogatandó tárgy, közeg reflexiós tényezőjétől. A mattfekete tárgyak és a fényes tárgyak letapogatási távolsága 1 : 3 arányban állnak.

19.2.3. Közelitéskapcsolók

A közelítéskapcsolók is az érintésnélküli kapcsolóelemek csoportjába tartoznak



hasonlóan az optikai elven működő kapcsolókhoz. Az érintésnélküli kapcsoló vezérlőegysége egy oszcillátort tartalmaz, amelynek nagyfrekvenciás mágneses mezeje az egyik oldalon szabadon kiléphet a környezetbe. Ha ehhez az aktiv felülethez

valamilyen fémes tárgy közelit, az a tárgy elvonja (csökkenti) az oszcillátor energiájának egy részét. Ez csillapítást okoz a rezgések amplitúdójánál. Ezt az amplitúdó- változást értékeli ki a kapcsoló, és ez indit el egy jelet a kapcsokóerősitő felé (19.5. ábra).

19.5.ábra: Indukciós közelitéskapcsoló tömbvázlata

A közelitéskapcsoló zavarmentes használatakor a letapogatandó tárgynak egy minimális nagyságot el kell érnie.

19.2.4. Szintérzékelők

A szintérzékelőket három csoportra osztjuk:

- kapacitiv szintérzékelők,
- ellenállásos szintérzékelők és
- utrahangos szintérzékelők.

19.2.4.1 Kapacitív szintérzékelők

A kapacitív szintérzékelő mérőkör müködése a rezonanciamódszeren alapul. Az osszcillátor frekvenciáját az elektródán létrejövő kapacitásváltozás hangolja el. A szükséges jelformálás, erősités és demodulálás után a mérendő mennyiség villamos jel formában dolgozható fel tovább, tehát vezethető a mikrovezérlő, vagy bemenő port megfelelő lábára.

19.2.4.2 Ellenállásos szintmérők

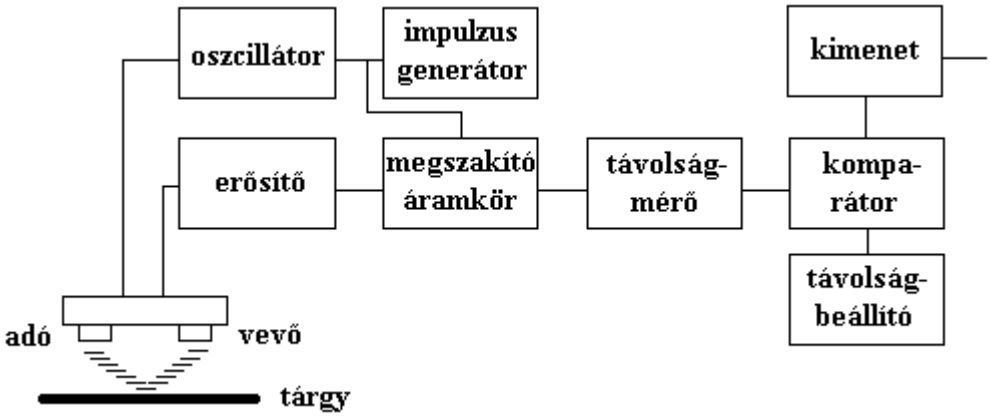
Az ellenállásos szintmérőket elektromosan vezető folyadékok és szilárd anyagok (pl. granulátumok) szintmagasságának a mérésére fejlesztették ki. Ezen berendezések működési elve a villamos ellenálásmérésen alapul.

19.2.4.3 Ultrahangos szintmérők

Az ultrahangos tartályszintmérők nagy előnye, hogy működésükre a mérendő közeg állapotának semmilyen hatása nincs. Bármilyen folyékony, vagy szilárd halmazállapotú vezető vagy szigetelő tipusú közeg tartálybeli magassága mérhető. Ezenkivül a mérendő közegbe semmilyen szonda nem nyúlik be, ami a mérendő közegre esetleg kihatna.

Az ultrahangos szintmérők hátránya viszonylag magas áruk, valamint az, hogy por, gőz és hőmérsékletváltozás kialakulása esetén csökken az érzékelők működőképessége.

Az ultrahangos érzékelő adóból és vevőből áll, tartalmaz egy távolságmérőkört, amely komparátoron keresztül hat a kimenetre. A kibocsájtott ultrahangrezgések a mérendő közeg és a felette levő levegő határfelületéről visszaverődnek. A visszaverődési időből meghatározható a távolság. Az ultrahangos szintmérő tömbvázlata a 19.6. ábrán látható.



19.6. ábra: Ultrahangos szintmérő tömbvázlata

19.2.5. Hőmérséklet - mérők

A villamos hőmérsékletmérés igen nagy jelentőségű, mivel sok fizikai és kémiai folyamat lefolyása hőmérsékletfüggő. Egy hőmérsékletmérő rendszer hőmérséklet-érzékelőből és erősítő, jelformáló egységből áll.
Háromféle hőérzékelőt különböztetünk meg:

- hőelem,
- ellenállás-hőmérő és
- termisztorok.

19.2.5.1. Hőelem

A hőelem egy aktiv, termofeszültséget előállitó mérőátalakitó. A hőlem nem más, mint két-egymással öszeforrasztott, vagy összehegesztett-vékony huzal. Ha a két csatlakozási pontot egymástol eltérően melegitjük, akkor a két vezetőből álló áramkörben elektromos feszüétség ébred, amely arányos a két hely között fennáló hőmérsklet-különbséggel.

19.2.5.2. Ellenállás hőmérő

Az ellenállás-hőmérő egy passzív átalakitó, amelyben az ellenállás hőmérsékletfüggőségét használjuk fel a méréseknél. Az érzékelőt legtöbbször

kerámiaszigetelőre tekercselt, vagy üvegcsőbe beolvasztott vékony platina ellenálláshuzal alkotja.

19.2.5.3. Termisztor

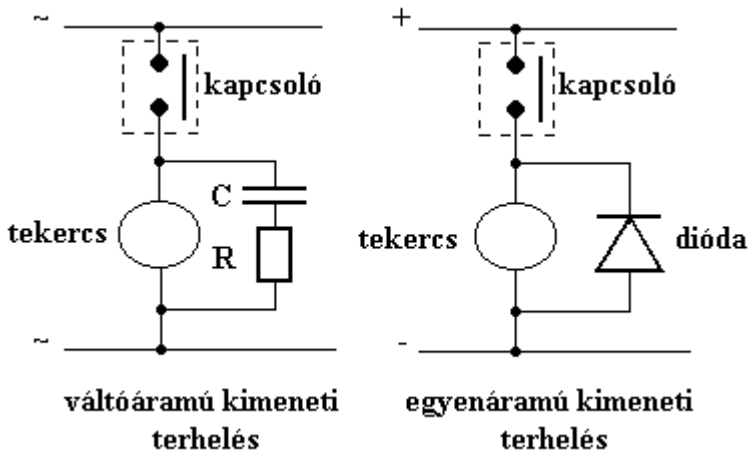
A termisztorok kerámia anyagú hőmérsékletfüggő félvezető ellenállások, amelyeknek erősen negativ hőmérsékleti tényezőjük van. Ezen tulajdonságaik miatt ezeket az elemeket mind kompenzációs kapcsolásokban, mind hőmérséklet-érzékelőként alkalmazzák.

19.3. KIMENETI ELEMEK

A mikroszámítógép, mikrovezérlő 5 V-os és néhány 10 mA-rel terhelhető kimeneteihez kell illeszteni a folyamatvezérlésben használt végrehajtó szerveket, amik rendszerint villanymotorok, pneumatikus, hidraulikus elemek, de lehetnek kijelzők stb. A feladattól függően többféle megoldás is alkalmazható. Hasonlóan a bemeneti elemekhez, célszerű itt is galvanikus leválasztást létrehozni, ugyanis ilyenkor a kimeneteken esetleg létrejövő nagyfeszültség, rövidzár nem tud a mikroszámítógép, mikrovezérlő felé hatni.

19.3.1. Érintkezős kimenetek

Érintkezős kimeneteknél legtöbször a 2 A / 220 VAC váltóáramot, ill. 24 VDC, vagy 12 VDC egyenáramot kapcsoló jelfogókat alkalmazunk. A kimeneti jelfogók érintkezőinek megkimélése érdekében ajánlatos, hogy az utánuk álló alkatrészeknél zavarelnyomó megoldást hozzunk létre. Igy pl. az elektromágneses jelfogók, mágnesszelepek stb. esetében egészen 1300V-ig terjedő zavarfeszültségvédelem hozható létre.
A váltakozó feszültséggel üzemelő készülékek minden tekercsénél párhuzamosan zavarszűrőt iktatunk be.



Az egyenfeszültséggel működő készülékeknél a tekercsel párhuzamosan egy diódát kapcsolunk, ennek a diódának a feladata a kikapcsolásnál megjelenő elektromágnesben felhalmozott mágneses energia felemésztése. (19.7. ábra).

19.7. ábra: Váltóáramú és egyenáramú kimeneti terhelés

19.3.1.1. Jelfogók

A jelfogó, más nevén relé tekercsére adott feszültség (és áram) hatására összezáródik a jelfogó **záróérintkezője**, illetve, ha van, a **bontóérintkező** nyit. Így a vezérlés és a kimenet között nincs galvanikus kapcsolat, tehát a mikrovezérlő felé nem tud visszajutni zavaró jel.

A jelfogókat általánosan a következő csoportokba sorolhatjuk be:

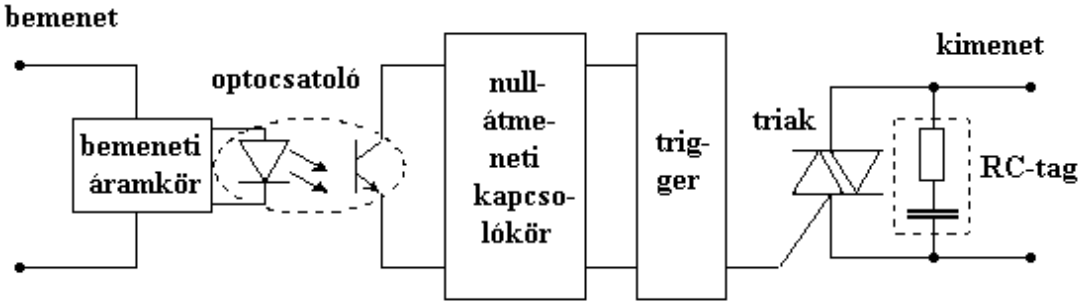
- ipari jelfogók,
- miniatűr jelfogók,
- nyomtatott áramkörbe szerelhető (forrasztható) jelfogók,
- Reed- jelfogók és
- szilárdtest- jelfogók.

A jelfogó-konstrukciók egyik alapkérdése az érintkezési pontok alakja, a másik az érintkezők anyaga, ill. bevonata. A leggyakrabban használt érintkezőváltozatok a szimpla érintkezőpár, az iker-érintkezőpár és a crossbar- érintkezőpár.

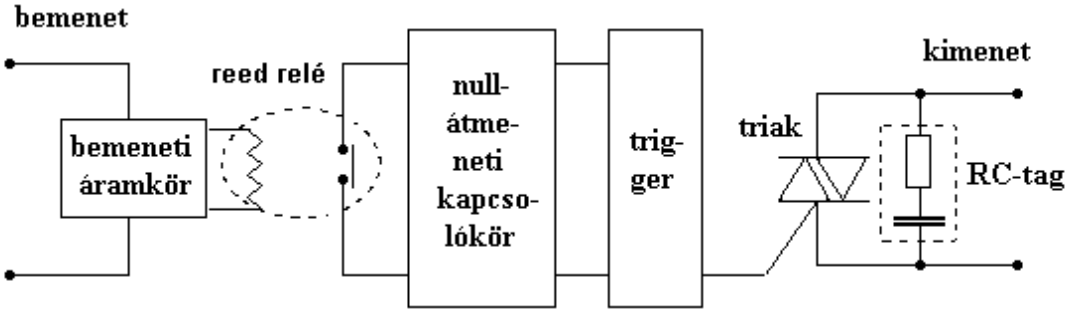
19.3.1.2 Szilárdtest-jelfogók

A szilárdtest-jelfogók bemeneti áramköreinek kétféle változata ismert:

- optocsatolók és
- Reed-jelfogók.



A 19.8. ábrán látható fényzáró házban egy fényemittáló diódát és egy fototranzisztort látunk (optocsatoló). Ha a LED kigyullad, a tranzisztor bekapcsol. Az ilyen bemenettel ellátot jelfogó egy nullátmeneti kapcsoló jellegével rendelkezik.
19.8.ábra: Szilárdtest-jelfogó optocsatolóval



A bemeneti áramkörben lévő Reed-jelfogós szilárdtest-jelfogók ugyanazt a szerepet töltik be, mint azok elektronikus megfelelői (19.9. ábra). A bemeneten mechanikus átalakító van, ezért a kapcsolási frekvencia, valamint az érintkező életartama (Reed-jelfogóé) korlátozott. Ezeknél a jelfogóknál a Reed-jelfogók tekercsét a bemeneten vezérlik.

19.9.ábra: Szilárdtest-jelfogó Reed-érintkezővel

Abból a célbol, hogy a névleges adatokat meghaladó feszültségcsúcsokat megakadályozzuk, a szilárdtest-jelfogókba ún. abszobert kell beépiteni. Ez valamilyen RC-kombinációból áll és a csúcsfeszültségeket csillapítja, amelyek főleg a kikapcsolási folyamatokban keletkezhetnek.

19.3.1.3. Triak-kimenetek

A triaknak a szilárdtest-jelfogóhoz hasonlóan az az előnye, hogy nagy áramerősségek kopásmentes átkapcsolására alkalmas.

19.4. BEMENETI ÉS KIMENETI EGYSÉGEK ILLESZTÉSE A MIKRO-SZÁMITÓ-GÉPHEZ

19.4.1. **Bevezetés**

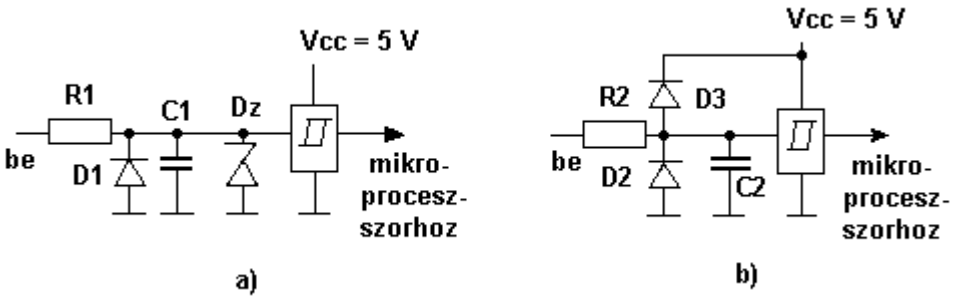
A külvilágból érkező jelek sokszor zavarokkal együtt kerülnek a mikrogépbe, és általában nem TTL szintűek ezért kell a bemeneteken sajátos áramköri megoldásokat

alkalmazni. A kimenetek jeleit pedig gyakran nem TTL áramkör, hanem egyéb eszköz használja fel, igy itt is jelformálási többletfeladatok jelentkeznek.

19.4.2. Jelformálás a mikroszámitógép, mikrovezérlő bemenetein

A **TTL** jelszint eléréséhez a bemeneti berendezés és a mikroszámitógép közé illesztőegységet kell helyeznünk.

A 19.10. ábrán látható jelfogadóáramkörök olyan esetekben használhatók, amikor a bejövő jel magas szintje 2,5 - 15 V, alacsony szintje –15 - 0,8 V közötti érték. Az R1 és R2 áramkorlátozó ellenállások akkor jutnak szerephez, ha valamelyik védődióda



kinyitott, vezet. Az a) ábrán a D1, ill. a b) ábrán a D2 dióda a negativ bemenő jeleket vagy a bejövő jelen lévő negativ zavarokat, tüskéket vágja le, a Dz Z-dióda, ill. a D3 dióda a pozitiv túlfeszültség ellen véd. A C1, ill. C2 kondenzátor a tűimpulzus jelegű nagyfrekvenciás öszetevőkből álló zavarjeleket szűri ki.

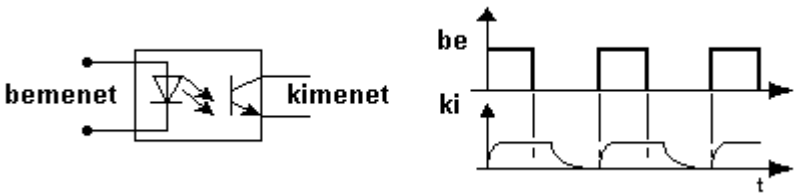
19.10. ábra: Áram- és feszültségkorlátozás a bemeneten

A jelfogadó áramkörben lényeges a **Schmitt-trigger**. Ha a bemenő jel szintváltozása lassan történik, vagy hoszabb ideig tartózkodik a tiltott sávban (az **L** és **H** szintek közötti definiálatlan, nem-értelmezett tartományban), ill. ha a zavarójel miatt a közönséges **TTL** kapuk rendszertelenül váltogatják a kimeneti szintjüket, akkor a hiszterézises bemenetű Schmitt-trigger biztositja a korrekt jelfogadást.

A logikai áramköröknél, így a mikroszámitógépeknél is a kialakitott bemeneti pontokat akkor sem szabad lezáratlanul hagyni, ha azokra nemcsatlakozik külső elem.

A mikroszámitógépek és mikrovezérlők be- és kimenetein egyaránt gyakori csatolóelem az optocsatoló. Az optocsatoló egy zárt egységben egy LED-et és egy optikai érzékelőt, általában fototranzisztort tartalmaz, ezek egymással optikai csatolásban vannak (19.11. ábra). A LED árama szokásosan 1…100 mA, és a fototranzisztor a LED áramától függő mértékben nyit ki. Az optocsatolónak két olyan tulajdonsága van, amely miatt jelkezelő elemként használhatjuk:

- nagy átütési feszültségű (1…10kV) galvanikus leválasztást biztosit a bemenete és a kimenete között és

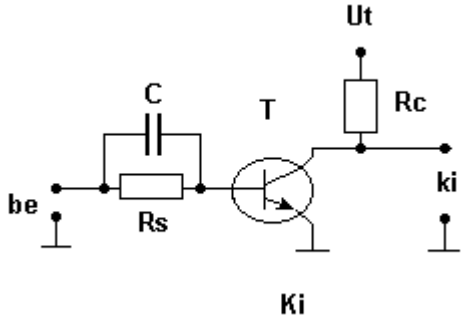


- bemenő jele áramjel is lehet.

19.11. ábra: Optocsatoló és átviteli görbéje

A mikroszámitógépek épitésekor, felhasználásakor alapszabályként kezelhető az az előirás, hogy a mikrogép galvanikusan nem kapcsolódhat a hálózathoz. Ez az esetleges zavarok és átütések miatt követelmény. Az optocsatoló kimeneti eleme, a fototranzisztor a digitális áramkörökhöz képest lassú áramköri elem, a LED fényintenzitásváltozásai is viszonylag kis frekvenciával válthatják egymást igy az optocsatolót is célszerű Schmitt.triggeresen csatolni. Természetesen léteznek nagyfrekvenciás optocsatolók is, amelyek segítségével jóval nagyobb frekvenciájú változások is átvihetők.

A digitális áramkörök és a Schmitt-triggerek esetében a küszöbfeszültségek a katalógusokban meghatározott értékek. Tetszőleges bemeneti feszültségszintekhez rendelhetjük a H és az L tartományt, ha a jelek fogadására analóg komparátort vagy

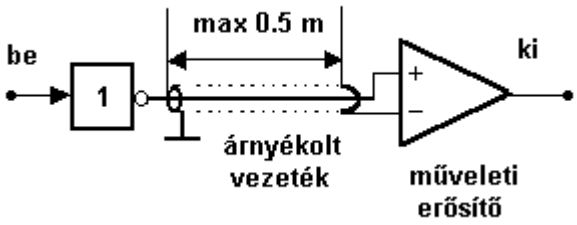


műveleti erősitőt használunk.

A mikroszámitógépek kimenetei segitségével kisebb-nagyobb rezisztiv terheléseket lehet müködtetni. Jelentősebb áramot igénylő terheléseket tranzisztoros meghajtófokozatokon keresztül kapcsolhatunk a digitális áramkörökhöz. A 19.12. ábrán egy igen egyszerű megoldás látható. Ez a kimeneti fokozat azonban nagyon érzékeny a terhelés kapacitására.

19.12. ábra: Tranzisztoros kimenőfokozat nagyobb áramokhoz

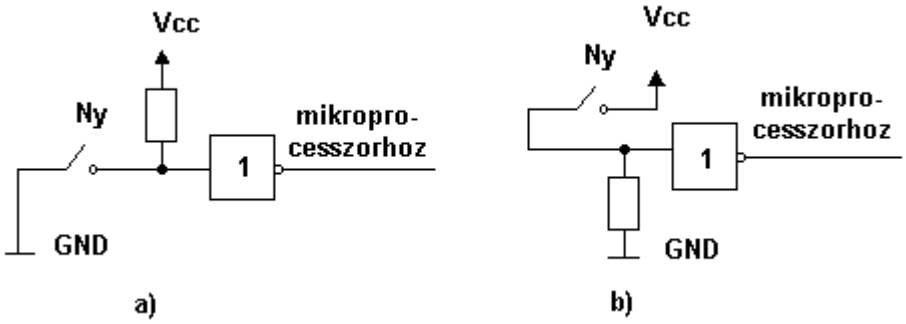
Ha a kimnő jelet nagyobb távolságra kell vezetni, akkor a kimeneten vonalmeghajtó áramköröket kell használnunk. Egyes esetekben azonban ilyen speciális áramkör nélkül is megoldható a jelátvitel. Az egyszerű TTL áramkörök kimenetéről kb. 30…50 cm t Már ilyen távolságok esetén is növeli a vel, komparátorral vagy vonalvevő á kszor a zavarérzéketlenség fokozása é sználása is.



19.13. ábra: TTL kimenet árnyékolt vezetékkel

Nagyobb távolságra ellenütemű, kétvezetékes jelátvitelt célszerű kiépiteni, a két vezetéket a szórt mágneses tér hatásainak minimalizálása érdekében összecsavarhatjuk. Az ellenütemű jelet optocsatolóval is fogadhatjuk, de a LED áramát korlátozó ellenállásról ilyen estben nem szabad megfeledkezni.

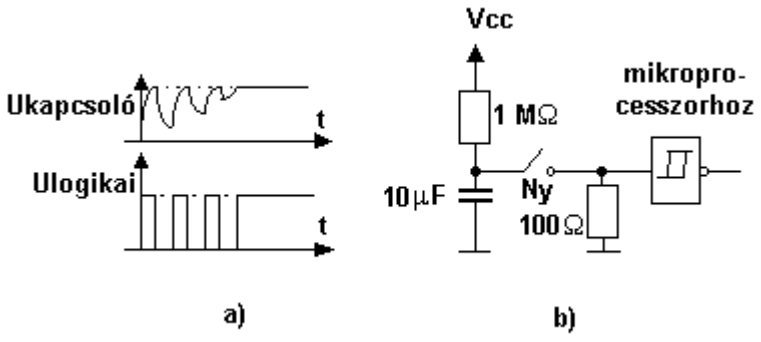
A mikroszámitógépnek gyakran egy kapcsoló, nyomógomb zárt vagy nyitott helyzetét kell érzékelnie. Ha a kapcsoló, nyomógomb egyik érzékelőjét a mikroszámitógép GND pontjával össze lehet kötni, a másik pontot vezetjük a digitális áramkörök bemenetére, és ott egy felhúzó ellenálást alkalmazunk(19.14. **a**) ábra). Ha viszont a Vcc ponttal kötjük össze, akor lehúzó ellenállást alkalmazunk (19.14. **b**) ábra). Az érintkezőket



fogadó áramköröket minden esetben úgy kell kialakitani, hogy az érintkező nyitott vagy zárt helyzetében ne álljon elő a logikai áramkör bemenetén szakadás (bekötetlen állapot). A meghajtóerősítő használata sokszor szükségtelen, ilyenkor elegendő egy érintkező és egy felhúzóellenállás bekötése is a bemenet lekezeléséhez.

19.14. ábra: Nyomógomb (kapcsoló) bekötése mikroprocesszorhoz

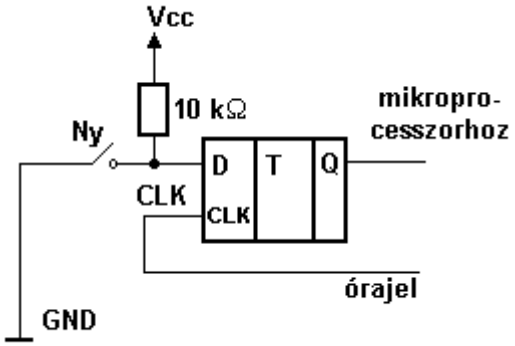
Az érintkezők, rugalmas jellegük miatt, záródáskor az első érintkezés után nem maradnak végérvényesen zárt állapotban, hanem néhányszor rugalmasan elválnak, majd ismét összeérnek. Ha a kapcsoló jó minőségű, néhány ms alatt lezajlik a pergési jelenség, de sok jelfogónál 20…40 ms időtartalmű lehet a jelenség. A **TTL**, **NMOS** és **CMOS** áramkörök működési sebessége lehetővé teszi, hogy e kapcsolók pergésével együtt járó jelszintváltásokat hűen kövessék (19.15. **a**) ábra). A pergésmentesités egyik lehetséges megoldása, hogy az első jelszintváltás után **RC** taggal késleltetjük a kimeneti jelszint változást (19.15. **b**) ábra).



19.15. ábra: A kapcsolók pergése, a) és pergésmentesítő áramkör, b)

A kapcsolásban az RC tag időálandóját a kapcsoló, érintkező egyedi tulajdonságai alapján kell méretezni.

Különleges megoldású pergésmentesitési módot mutat be a 19.16. ábra. Akkor tudjuk alkalmazni, ha a mikroszámitógépben rendelkezésünkre áll egy órajel, kb. 50…60 Hz frekvenciával. Mivel a pergés kb. 20 ms alatt zajlik le, ebbe az időtartamba legfeljebb



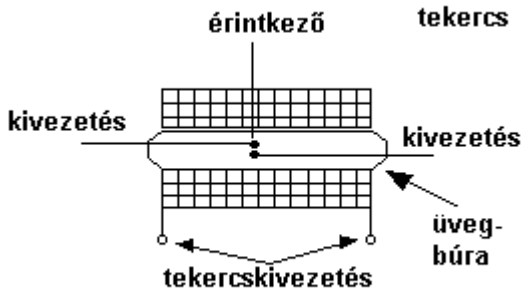
egy órajel esik bele. Az estleges pergés miatt ekkor véletlenszerűen az előző állapotot erősiti meg a kapcsolás, de az órajel már az új állapotot állitja elő, és a második órajel már mindenképpen az új állapotnak megfelelő bemenő jel idején érkezik. Látható, hogy az átment egyértelmüen alakul ki, a kimeneti jelben nincs pergés.
19.16. ábra: Pergésmentesítés D tárolóval

Meg kell jegyezni, hogy sok mikroprocesszor, mikrovezérlő már eleve pergésmentesített bemenettel rendelkezik. Gyakran alkalmaznak a bemeneteken Schmitt-triggert, sok berendezésnek beépített eleme a felhúzóellenállás, amit programozással lehet letiltani, vagy engedélyezni.

A kézzel be- és kikapcsolható események egyik korszerű kezelési módja az érintéses működtetés. A CMOS digitális áramkörök GΩ nagyságrendű bemeneti ellenállása lehetővé teszi, hogy az emberi bőr ellenállása rövidzárként hasson. Ugyanigy rosszul vezető folyadékot (pl. vizet) is lehet érzékelni.

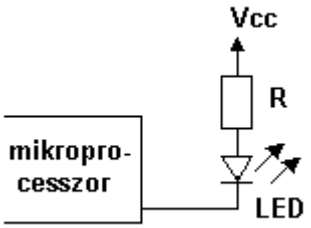
19.4.3. Jelformálás a mikroszámitógép, mikrovezérlő kimene-tein

Gyakran müködtetünk a mikroszámitógép kimenetéről jelfogókat. A digitális áramkörökkel, igy a mikroszámitógépekkel is a korszerű, miniatűr kivitelű reed (üvegcsöves) jelfogókat lehet a legegyszerűbben kapcsolni. A reed jelfogó egy



üvegcsőbe forasztott érintkezőpár (19.17. ábra), amelyben az érintkezők lágymágneses anyagból készülnek.
19.17. ábra: Reed jelfogó

Ha az üvegcsőhöz állandómágnessel közelitünk, az érintkezők összetapadnak, ha a mágnes távolabb kerül, az érintkezők rugalmasan szétválnak. A miniatűr reed jelfogókban az üvegcsövet több száz vagy töb ezer mentes tekrcs veszi körül de így is elfér egy DIL integrált áramköri tokban. A jelentős menetszámú tekercs nagy induktivitást jelent, és a kikapcsoláskor akkora induktiv feszültséglökés keletkezik, ami

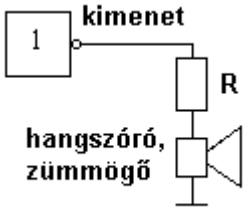


az IC-t kárositaná. Ez ellen véd a dióda, amelyet a jelfogóval párhuzamosan, a feltüntetett polaritással kell bekötni. Az 5V-nál nagyobb feszültségű jelfogókat nyitott kollektoros tranzisztor kimenettel lehet meghajtani, és ha a jelfogó áramigényét az IC kimenet nem képes kielégiteni, akkor tranzisztoros előfokozatot épitünk be.

Ha a mikroszámitógépben jelzőfényt használunk, ezt leggyakrabban LED beépitésével oldjuk meg (19.18. ábra).

19.18. ábra: LED használata jelzőfény előállítására

Gyakran állitunk elő hangjelzéseket is mikroszámitógépekkel. A hanghatásokat egy alaposzcillátorral és vezérelt frekvenciaosztó áramkörrel, vagy szoftver úton beállított frekvenciájú impulzussorozattal lehet megvalósitani. Mozgó alkatrész nélküli, korszerű, kis méretű hangkeltő egységek a kerámiazümmögők (piezoelektromos hangkeltők). Ezek csak kapacitiv terhelést jelentenek, igy illesztőelem nélkül is a kapuk kimeneteihez kapcsolhatók (19.19. ábra).



19.19. ábra: Hangjelzés álőállítása mikroprocesszor kimeneten

Hálózati fogyasztókat is működtethetünk a mikroszámitógéppel. Ilyen esetben azonban a szereléskor az érintésvédelmi előirásokat maradéktalanul be kell tartanunk, és a korábban már említett általános elvnek megfelelően a hálózat és a mikroszámitógép között nem lehet galvanikus kapcsolat. Ha ilyen megoldást válsztunk, a galvanikus leválasztást több ponton is megoldhatjuk. Beszerezhetőek olyan tirisztorok és triakok, amelyek a gyújtófeszültséget beépitett optocsatolóval fogadják, igy a gyújtóáramkör és a hálózati oldal galvanikusan független. Külső optocsatolót is lehet alkalmazni, ill. gyakran gyújtótranszformátort használunk.

A mikroszámitógép tápegységét úgy is ki tudjuk alakitani, hogy ha munka közben a hálózat kimarad, a mikroprocesszor a futó program későbi folytatásához szükséges fontosabb információkat védett helyre küldhesse. Ha ezen információk és a RAM tárak tartalma a hálózatkimaradás idején pl. háttérelemekkel, akkumulátorral megőrizhető, akkor a hálózati feszültség visszatérésekor a program tovább futhat. Ha a mikroszámitógép egészében is kis fogyasztású, akkor a normális üzem is folytatható akkumulátoros, vagy telepes táplálással. Az átkapcsolás megvalósitásához olyan áramkört kell kialakitanunk, amely a lehető leghamarabb jelzi, ha a hálózati feszültség kimarad.

19.4.4. Bemeneti és kimeneti egységek illesztése

A kis- és közepes integráltságú elemekkel változatosan oldhatjuk meg a mikroszámitógépek bemeneti és kimeneti illesztési feladatait. A kimeneteket minden esetben úgy kell kiképezni, hogy ne csak addig a rövid ideig jelenjen meg a kimenő adat, amig azt a mikroprocesszor kiküldi, mert ez mindösze 5…20 μs lenne. A kiküldött adatnak mindaddig jelen kell lennie a kimeneti pontokon, míg egy későbbi kimenő adat át nem irja a kimenti vonalak jeleit. Ez azt jelenti, hogy a kimenő jeleket mindenképpen tárolni, pufferelni kell. A bejövő adatokat is célszerű tárakkal fogadni, így ugyanis megoldható, hogy a mikroprocesszor az adott bemenet olvasása esetén ne a tényleges bemeneti vezetékeken esetleg éppen változó jelszinteket érzékelje, hanem a bemenő jelet fogadó tároló stabil jelszintjeit. Az ilyen jellegű problémákat a fejletebb mikrovezélőknél már belsőleg megoldották.

Ellenőrző Kérdések

1. Mi a számítógép feladata ?
2. Milyen tipusú bemeneti elemeket használunk a leggyakrabban ?

3. Mi a galvanikus leválasztású bemenetek előnye ?
4. Milyen elemekkel valósítható meg a galvanikus bemenet ?
5. Rajzoljo le egy optocsatolós bemeneti áramkört .
6. Milyen tipikus jeladókat használunk a mikroprocesszoros rendszerekben ?
7. Mi a mikrokapcsoló és milyen tipusait ismeri?
8. Mi a fotoelektromos kapcsoló és mi az előnye ?
9. A fotoelektromos kapcsolóknak milyen tipusait ismeri ?
10. Ismertesse az egyutas fénysorompó működési elvét .
11. Ismertesse a visszaverődéses fénysorompó működési elvét .
12. Ismertesse a diffúz reflexiós fénysorompó működési elvét .
13. Ismertesse a közelítéskapcsolók működési elvét .
14. Milyen szintérzékelőt ismer ?
15. Hogyan működik a kapacitív szintérzékelő ?
16. Hogyan működik az ellenállásos szintmérő ?
17. Hogyan működik az ultrahangos szintmérő ?
18. Milyen hőmérsékletmérési elvet ismer ?
19. Ismertesse a hőelemmel történő hőmérsékletmérést .
20. Ismertesse az ellenállással történő hőmérsékletmérést .
21. Ismertesse a termisztorral történő hőmérsékletmérést .
22. Miért kell illeszteni a mikrogép, mikrovezérlő kimeneteit a végrehajtó elemekhez?
23. Az érintkezővel megoldott kimeneteknél leggyakrabban melyekre van igény ?
24. Egyen- illetve váltó kimeneteknél hogyan oldjuk meg a tekercsben felhalmozott mágneses energia felemésztését ?
25. Mi a jelfogó (relé) és milyen tipusú jelfogókat ismer ?
26. Mi jellemzi az ipari jelfogókat ?
27. Mi jellemzi a miniatűr jelfogókat ?
28. Mi jellemzi a nyomtatott lapra szerelhető jelfogókat ?
29. Mi jellemzi a Reed jelfogókat ?
30. Mi jellemzi a szilárdtest jelfogókat ?
31. Mi jellemzi a triak kimeneteket ?
32. Milyen jelek érkeznek a külvilágból a mikrogépbe és milyen jeleket kell a külvilág felé küldeni ?
33. Milyen jelszinttel dolgoznak a mikrogépek, mikrovezérlők ?
34. Ismertesse a bemenő illesztőket, amik más feszültségű jeleket illesztenek a mikrogép bemenetére ?
35. Mi a Schmitt-trigger és miért kell használni a mikrogépeknél a jelfogadásra ?
36. Mi az optocsatoló, hogyan működik és miért előnyös használata a digitális áramkörökben ?
37. Milyen az optocsatoló átviteli görbéje ?
38. Milyen tranzisztoros kimeneti fokozatokat alkalmazunk mikrogépeknél ohmos terhelések meghajtására ?
39. Milyen problémák jelentkeznek digitális jelek nagyobb távolságra való vezetésénél ?
40. Milyen megoldásokkal lehet kiküszöbölni digitális jelek nagyobb távolságra való vezetésénél jelentkező zavarok káros hatását ?
41. Hogyan lehet érintkezős bemenetek állapotváltozását digitális bemenetekre vezetni ?
42. Mi az érintkezők pergése, milyen problémát okoz digitális bemeneteknél és hogyan, mivel lehet kiküszöbölni ?

43. Mi a Reed-relé és hogyan működik ?
44. Hogyan lehet mikrogéppel világító kimeneteket működtetni ?
45. Hogyan lehet hangjelzést létrehozni mikrogéppel ?

20. AZ *INTEL 8051* MIKROVEZÉRLŐ CSALÁD (MCS-51)

20.1. BEVEZETÉS

Kisebb készülékek, berendezések viszonylag egyszerű feladatait nem célszerű nagyteljesítményű mikroprocesszorokkal és bonyolult, drága kiegészítő elemekkel megoldani. Az ilyen processzorok utasításainak képességei is messze meghaladják a feladat igényeit. Ezeknek, a viszonylag egyszerű feladatok megoldásainál nagyon igéretesek az **egy chip-es mikroszámítógépek (one chip microcomputers)**, amiket még **mikrovezérlőknek** is neveznek. A tantárgyon belül a az **Intel 8051**-es mikrovezérlő-családjára azért esett a választás, mert ez egy valódi egy chip-es mikroszámítógép a beépített memóriákkal, számlálókkal és ki-/bemenő portokkal (csatlakozási pontokkal), ugyanakkor a nagyobb teljesítmény eléréséhez külső kiegészítő elemek hozzáadásával juthatunk, a mikroszámítógép struktúrájához hasonló berendezést építhetük ezzel a mikrovezérlővel.

Az **MCS-51 mikrovezérlő család** alaptipusai a 20.1. táblázatban láthatók.

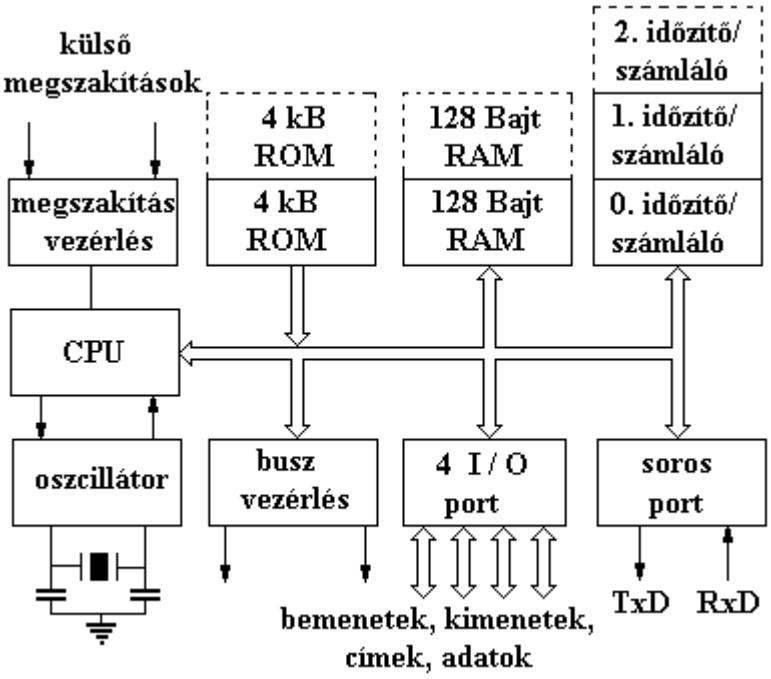
20.1. táblázat: Az Intel 8051 mirovezérlő család

mikrovezélő	belső RAM	belső ROM
8051	128 x 8	4 kB x 8 b ROM
8031	128 x 8	-
8751	128 x 8	4 kB x 8 b EPROM
8052	256 x 8	8 kB x 8 b ROM
8032	256 x 8	-
8732	256 x 8	8 kB x 8 b EPROM

Egy ilyen mikrovezérlõ megvalósítása kb. 65.000 tranzisztor felhasználásával lehetséges. Csak érdekességképpen jegyezzük meg, hogy az Intel Pentium III processzor kb. 10 millió tranzisztort tartalmaz (2000-es adat).

20.2. AZ MCS-51 CSALÁD ÁLTALÁNOS JELLEMZÉSE

A típusszám végén levõ 1, ill. 2 a kétféle tipust jelöl, mindkettõ tipusnál van maszkprogramozott ROM memóriát tartalmazó chip (8051 és 8052), de ROM nélküli változat is (8031 és 8032), valamint a belsõ EPROM-os (8751 és 8752). Az alaptipus ismertetésekor, ha felmerül eltérés a különböző változatoknál, azt külön ismertetjük. A vezérlő belső felépítésének egy egyszerűsített változata látható az 1. ábrán. A 2. ábrán látható a mikrovezérlő részletesebb vázlata, természetesen ez az ábra a programozás szempontjai szerint adja meg a belső achitektúrát, a fizikai megvalósítás ettől eltér.



20.1. ábra: Az Intel 8051 mikrovezérlő egyszerűsített felépítése

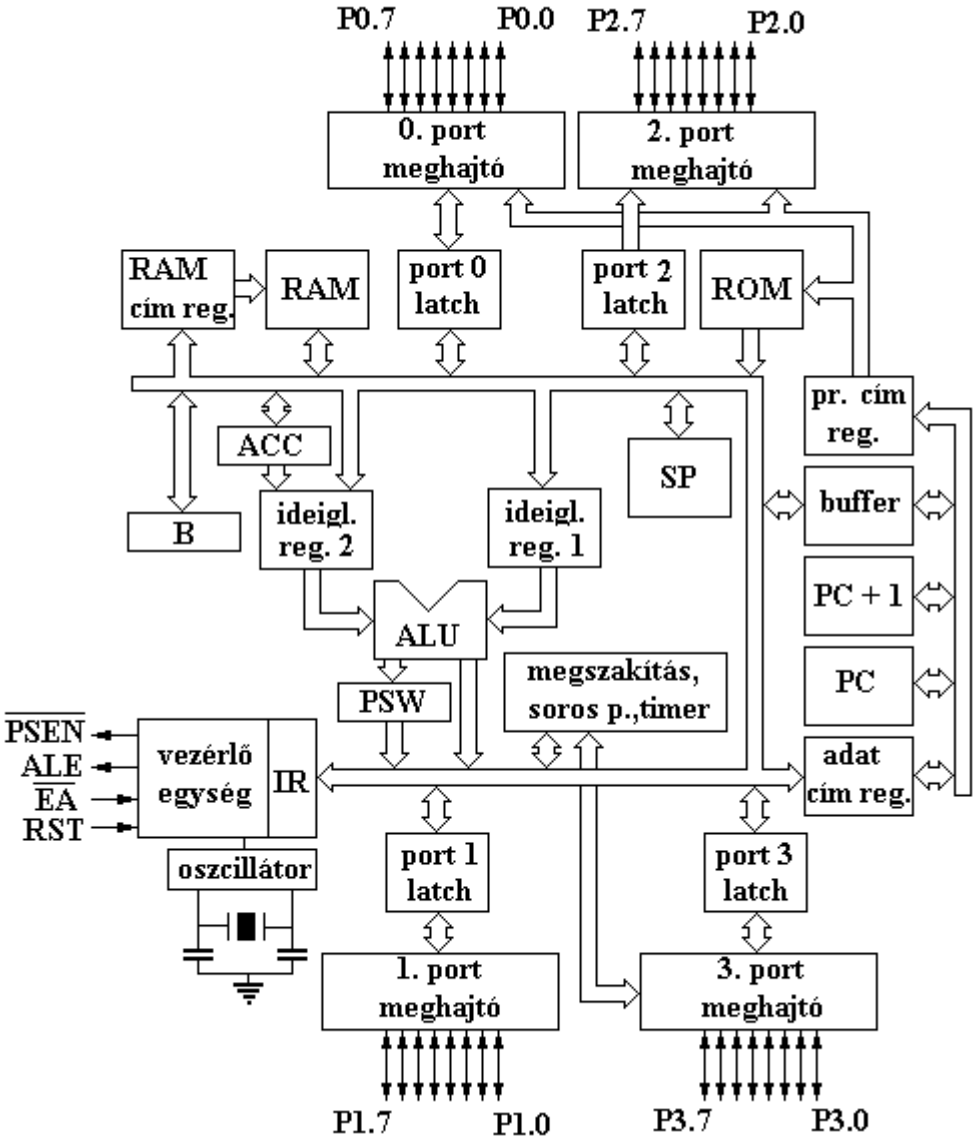
20.3. AZ MCS-51 CSALÁD LEGFONTOSABB JELLEMZŐI

A mikrovezérlőcsalád legfontosabb jellemzői a következők:

- 8 bites CPU,
- 1,2 - 12 MHz oszcillátor frekvencia,
- IC-n belüli oszcillátor és órajel áramkör,
- 1 μs-os tipikus utasításciklus,
- 8 bites, egésszámú, előjel nélküli szorzás, 4 μs alatt,
- 32 be-/kimenő vonal, ez a szám bizonyos bővítések esetén csökken,
- 64 kB külső címterület adatmemória kiépítéshez (RAM),
- 64 kB külső címterület programmemória kiépítéshez (ROM),
- két 16 bites időzítő/számláló (---2 tipusoknál három),
- ötforrásos megszakítás rendszer (----2 esetén hatforrásos), két priortási szinttel,
- teljes duplex port soros adatátvitelhez (RS 232),
- Boole processzor, Boole műveletek elvégzési lehetőségével.

A 20.3. ábrán látható a 40 kivezetéses IC lábkiosztása. Az IC lábait mindig felülről nézve határozzuk meg, kiindulva az első láb jelölésétől, ami lehet egy pont, egy félkör alakú bevágás, vagy mindkettő, sorban számozva az utolsó lábig, onnan áttérve a szemközti sorban visszafelé haladunk az utolsó kivezetésig. Így az 1. lábbal szemben mindig a legnagyobb sorszámú lábat találjuk, ami esetünkben 40. Az egyes lábak, csatlakozópontok elnevezései és szerepük a 20.2. táblázatban találhatók. A 20.4. ábrán látható a mikrovezérlő logikai blokksémája, rendszertervezéskor, rajzoláskor ezt a jelölési rendszert használjuk.

A 20.1. táblázat ’**nincs’, ’ROM’** vagy ’**EPROM’** feliratú blokkja azt jelzi, hogy a mikrovezérlő nem tartalmaz belső programmemóriát, ezt a nincs felirat jelenti (a 8031 és 8032 tipusok), illetve a programmemória belső **ROM** vagy **EPROM**.



20.2. ábra: Az **MCS-51** mikrovezérlő család tagjainak belső felépítése

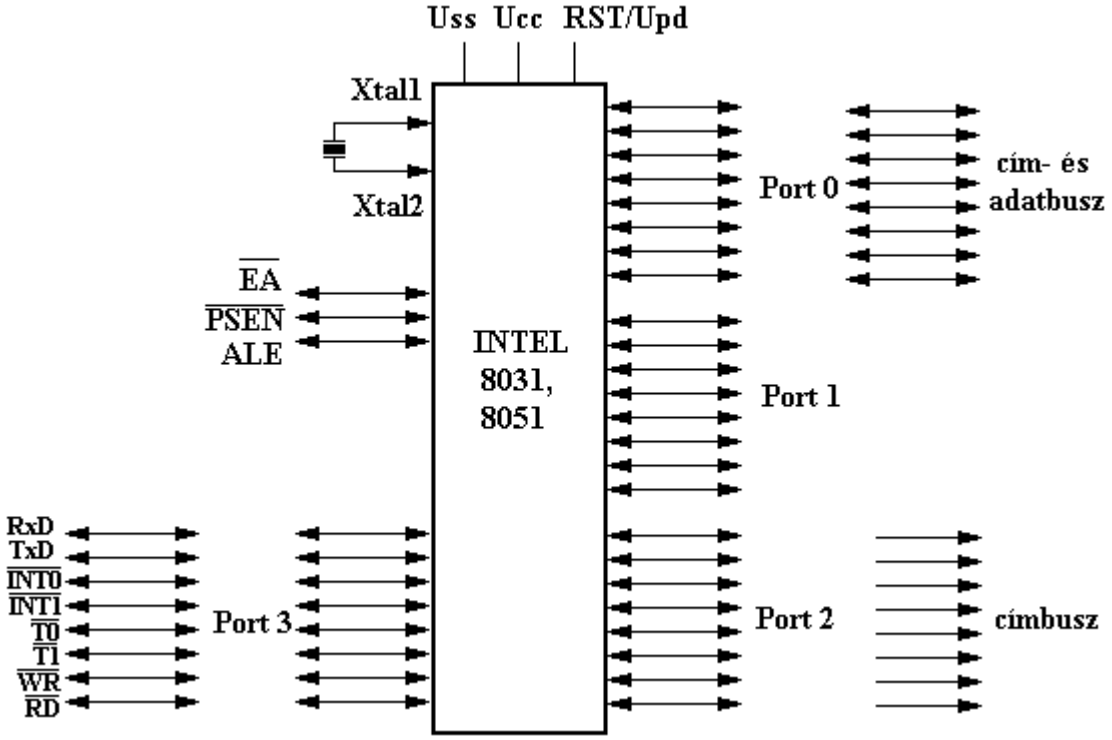
csak 8052	elnevezés	sorszám		sorszám	elnevezés
T2	P1.0	1.		40.	Ucc
T2EX	P1.1	2.		39.	P0.0/AD0
	P1.2	3.		38.	P0.1/AD1
	P1.3	4.		37.	P0.2/AD2
	P1.4	5.		36.	P0.3/AD3
	P1.5	6.		35.	P0.4/AD4
	P1.6	7.		34.	P0.5/AD5
	P1.7	8.		33.	P0.6/AD6
	RST/UPD	9.		32.	P0.7/AD7
	RxD/P3.0	10.		31.	-EA
	TxD/P3.1	11.		30.	ALE
	-INT0/P3.2	12.		29.	-PSEN
	-INT1/P3.3	13.		28.	P2.7/A15
	T0/P3.4	14.		27.	P2.6/A14
	T1/P3.5	15.		26.	P2.5/A13
	-WR/P3.6	16.		25.	P2.4/A12
	-RD/P3.7	17.		24.	P2.3/A11
	Xtal2	18.		23.	P2.2/A10
	Xtal1	19.		22.	P2.1/A9
	USS	20		21.	P2.0/A8

20.3. ábra: Az **Intel 8051/8031** mikrovezérlõ lábkiosztása

A 20.4. ábrán a logikai pontok szerint csoportosítva láthatók a 8051 csatlakozási pontjai.

20.2. táblázat: A csatlakozópontok sorszáma, jelölése és rövid leírása		
csatlakozás sorszáma	**csatlakozás jelölése**	**a csatlakozás feladatának ismertetése**
1….8	**P1.0….P1.7**	**PORT 1, bitenként címezhetõ, 8 bit széles, 4 LSTTL meghajtó képességû pontok.**
9	**RST/UPD**	**Bemenet, H→L lefutó él hatására a 8051 alap-állapotba kerül. Ha UCC 4.5…4.0 V-ra csökken és itt H van (Power Down) funkció, a belső RAM innen táplálható.**
10….17	**P3.0….P3.7**	**PORT 3, bitcímezhetõ 8 bites port, 4 LSTTL meg-hajtó képességgel. Az alternatív funkciók teljesítése-kor a megfelelõ kimenet latch-ében H kell hogy legyen.**
10	**RxD**	**Soros port, aszinkron üzemben bemenet, szinkron üzemben beme-net/ kimenet.**
11	**TxD**	**Soros port, aszinkron üzemben soros kimenet, szinkron üzemben kimenõ órajel.**
12	**-INT0**	**Interrupt bemenet, Timer 0 Gate bemenet.**
13	**-INT1**	**Interrupt (megszakítás) bemenet, Timer 1 Gate bemenet.**

14	T0	A 0. számláló bemenete.
15	T1	Az 1. számláló bemenete.
16	-WR	Kimenõ vezérlõ jel, külsõ adattárhoz írásparancs.
17	-RD	Kimenõ vezérlõ jel, külsõ adattárhoz olvasásparancs.
18	Xtal2	Az oszcillátor erõsítõ kimenete ill. a hangoló rezgõkvarc, kerámia rezo-nátor vagy külsõ órajel csatlakozó pontja, külsõ órajel esetén GND-re kell kötni.
20	GND (USS)	0 V, a tápfeszültség testpontja.
21….28	P2.0….P2.7	PORT 2, Bitcímezhetõ, 8 bit széles port, 4 LSTTL meghajtó képesség-gel, a külsõ program- és adatmemória címek is itt lépnek ki (felsõ címbájt).
29	PSEN	Kimenõ vezérlõjel, külsõ programtár olvasását jelzi.
30	ALE	(Address Latch Enable), kimenõ vezérlõjel , cím tároló engedélyezés, az alsó címbájt a külsõ memóriák számára tárolható.
31	EA	Bemenõ vezérlõ jel, H szintje esetén az alsó 4 kB programtár cím a belsõ tárra vonatkozik, L szintje esetén az utasítások csak külsõ tárból olvasódnak.



32….39	P0.0….P0.7	PORT 0, Bitcímezhetõ, 8 bit széles port, 8 LSTTL meghajtó képességû. Külsõ tár esetén idõmultiplexen itt lép ki a cím alsó bájt, majd itt halad át az adat ill. utasítás is.
40	Ucc	Tápfeszültség (+ 5 V).

20.4. ábra: A 8031/8051 csatlakozópontjai

20.4. A 8051/8031 MIKROVEZÉRLŐ HARDVER FELÉPÍTÉSE

20.4.1 **A 8051/8031 memória szervezése és belső regiszterei**

Viszonylag nagy belső adatmemória-területet tartalmaz a mikrovezérlő, aminek kezelése független a programmemória kezelésétől. A programmemóri-terület legfeljebb 64 kB lehet, amibe beletartozik a belső 4 kB vagy 8 kB terület is. Ha a belső adatmemória nem elegendő, akkor legfeljebb még 64 kB külső adatmemória is alaklmazható, aminek nincs átfedése a belső adatmemóriával.

A 128 v. 256 bájt belső adatmemóriához tartozik még egy **speciális funkcióregiszter memória** is (**SFR**). A belső adatmemória és SFR direkt v. indirekt címezhető, a külső adatmemória 8 v. 16 bites címekkel. A belső adatmemória (RAM) szervezése a 20.5. ábrán látható.

Decimális cím	**A memóriaterület neve**	**Hexadecimális cím**	
255 **128**	**SFR terület**	**FF** **80**	**128 bájt RAM terület, csak ---2 estében**
127 **48**	**Szabad terület**	**7F** **30**	
47 **32**	**Bit-címezhető terület**	**2F** **20**	
31 **24**	**Reg. Bank 3**	**1F** **18**	
23 **16**	**Reg. Bank 2**	**17** **10**	
15 **8**	**Reg. Bank 1**	**0F** **08**	
7 **0**	**Reg. Bank 0**	**07** **00**	

20.5. ábra: A 8051 belső RAM szervezése

A 20.5. ábra 80 - FF területén levő adatmemória csak a ---2 tipusoknál létezik, címzése kizárólag indirekt lehet, értelemszerűen az SFR címzése pedig direkt..

A 20.3. táblázat tartalmazza az SFR regiszterek elnevezéseit, címeit és magyarázatát.

Az egyes jelölések jelentése a következő:

+ jel, csak a ---2 jelű tipusnál létező regisztert jelzi,
- jel mind bit- mind bájtcímzést jelöl.

20.3. Táblázat: Az SFR (speciális funkcó regiszter) készlet					
Szim-bólum	**cím**	**---2-nél**	**megnevezés**	**cím**	**feladat**
ACC	*		**accumulator**	**E0**	**a müveleteknél a leggyakrabban használt reg.**
B	*		**B register**	**F0**	**szorzásnál, osztásnál egyik operandus helye,ált.célú reg.**
PSW	*		**Program Status Word**	**D0**	**programállapotszó**
SP			**stack pointer**	**81**	**veremtár-mutató, alaphelyzet 07 belső adatmem., átírható**
DPTR			**Data Pointer**	**82**	**DPH felső-, DPL alső bájt, külső adatmemória címreg.**
P0	*		**PORT0**	**80**	**kétirányú külső adatprot**
P1	*		**PORT1**	**90**	**kétirányú külső adatprot**
P2	*		**PORT2**	**A0**	**kétirányú külső adatprot**
P3	*		**PORT3**	**B0**	**kétirányú külső adatprot**
IP	*		**Interrupt Priority Control**	**B8**	**a megszakítás prioritásának vezérlése**
IE	*		**Interrupt Enable Control**	**A8**	**a megszakítás engedélyezése ill. tiltása**
TMOD			**Timer/Counter Mode Control**	**89**	**16 bites időzítő/számláló üzemmód beállítás**
TCON	*		**Timer/Counter Control**	**88**	**16 bites időzítő/számláló vezérlése**
T2CON	*	+	**Timer/Counter 2 Control**	**C8**	**16 bites időzítő/számláló vezérlése**
TH0			**Timer/Counter 0 (high byte)**	**8C**	**0. időzítő/számláló regiszter, nagyobb helyértékű bájt**
TL0			**Timer/Counter 0 (low byte)**	**8A**	**0. időzítő/számláló regiszter, kisebb helyértékű bájt**
TH1			**Timer/Counter 1 (high byte)**	**8D**	**1. időzítő/számláló regiszter, nagyobb helyértékű bájt**
TL1			**Timer/Counter 1 (low byte)**	**8B**	**1. időzítő/számláló regiszter, kisebb helyértékű bájt**
TH2		+	**Timer/Counter 2 (high byte)**	**CD**	**2. időzítő/számláló regiszter, nagyobb helyértékű bájt**
TL2		+	**Timer/Counter 2**	**CC**	**2. időzítő/számláló regiszter,**

			(low byte)		**kisebb helyértékű bájt**
RCAP 2H		+	**Timer/Counter 2 Capture Register**	**CB**	**RCAP2H és RCAP2L regiszterpár különleges üzemmódjához**
RCAP 2L		+	**Timer/Counter 2 Capture Register**	**CA**	**RCAP2H és RCAP2L regiszterpár különleges üzemmódjához**
SCON	*		**Serial Control**	**98**	**a soros adatátvitel vezérlése**
SBUF			**Serial Data Buffer**	**99**	**két regiszter, egyik az adatbeolvasásra, a másik az adatküldésre**
PCON			**Power Control**	**87**	**a párhuzamos portok vezérl.**

Az SFR-ben szereplõ **PSW** (programállapotszó) regiszter egyes bitjei a 6. ábrán láthatók.

PSW							
MSB							**LSB**
CY	**AC**	**F0**	**RS1**	**RS0**	**OV**	**-**	**P**
PSW7	**PSW6**	**PSW5**	**PSW4**	**PSW3**			**PSW0**
átvitel (Carry) flag	**félbájtos átvitel (Auxiliary Carry), BCD számoknál használat-tos**	**általános célra használa-tos jelzõbit, flag**	**regiszter csoport kiválasztó (Register Bank Select) bitek, szoftver állitás 11 - BANK3 10 - BANK2 01 - BANK1 00 - BANK0**		**túlcsordu- lás (Over-flow) flag**		**paritás (Parity) flag, minden művelet után 0, ha az ACC-ben az egyesek száma páratlan, 1 ha páros**

20.6. ábra: A PSW regiszter bitjei

A belsõ adatmemória csoportjainak kiválsztása (címzése) a 4. táblázat szerint történik.

20.4. táblázat: A csoportok kiválasztása			
RS1	**RS0**	**elnevezés**	**cím**
0	**0**	**BANK0**	**00 - 07**
0	**1**	**BANK1**	**08 - 0F**
1	**0**	**BANK2**	**10 - 17**
1	**1**	**BANK3**	**18 - 1F**

20.4.2. **a 8051/8031 órajelének elõállítása**

A 20.2. ábrán látható beépített belsõ óragenerátor hangolása rezgõkvarccal történik, ahol a két kondenzátor azonos értékű, C = 30 pF +/- 10 pF. Kerámia rezonátornál C = 40 pF +/- 10 pF. A 8051 családnál az alaposzcillátor frekvenciája f^q = 1,2..12 MHz. Külső órajel esetében a jelet **Xtal2** bemenetre kell kötni, ezen a ponton 4,7 kΩ felhúzóellenállást kell alkalmazni, **Xtal1** pont pedig **GND**-re kerül.

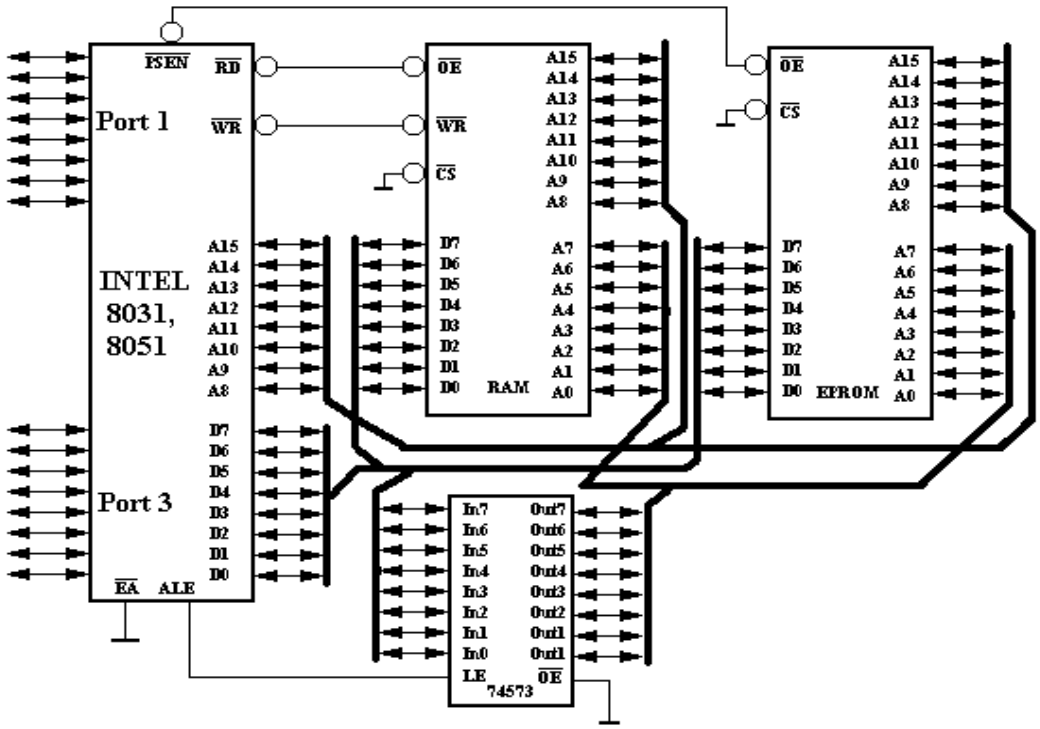
A mikrovezérlő egy-egy utasításának végrehajtási időtartama az **utasításciklus**, amit gépi ciklusokra osztunk fel. Minden gépi ciklus 6 állapotciklust tartalmaz. A 8051 utasításainak nagy része egy gépi ciklus alatt hajtódik végre, egy része két gépi ciklus alatt és két utasítás 4 gépi ciklus ideig tart (szorzás és osztás).

20.4.3. **A 8051 párhuzamos portjai**

Mind a 4 párhuzamos port kétirányú és tartalmazzák a latch-et (adattároló), kimeneti meghajtó fokozatot és a bemeneti puffert. Külső adat- és/vagy programmemória elérést biztosít a **P0** és **P2** port kimeneti meghajtó fokozata és a **P0** bementei puffere. Ilyenkor P0 kimenetein a a cím alsó bájtja jelenik meg, ezután időmultiplexelten az adat (vagy írás, vagy olvasás). A 16 bites cím felső bájtja a P2 kimeneten jelenik meg. A **P3** port alternativ funkciókkal is rendelkezik, ezek leírása a 2 táblázatban található meg.
Programból a portok (vagy azok egyes jelei) könnyen kezelhetők, minden SFR regiszter írása (P0 - P3) egyúttal a külső kivezetés írása is. Olvasás elött az egyes pontokat 1-es logikai szintre kell írni, ez bekapcsoláskor (tehát RESET esetében) is megtörténik.

20.4.4. **A 8031 összekapcsolása külső memóriával (RAM és EPROM)**

A 8051 mikrovezérlő család kialakításánál igyekeztek a tervezők egy olyan rendszert kialakítani, aminek a segítségével a felhasználó a legegyszerűbb hardver kiépítéssel, sokszor egy, vagy néhány elem segítségével már mikroszámítógépet hozhat létre. Ugyanakkor meghagyták a lehetőségét annak is, hogy egy hagyományos 8 bites mikroszámítógép struktúrát is létrehozhasson a felhasználó. Így egy fejlesztőrendszer használásával építse ki a legkülönfélébb kapcsolásokat, a teljes szoftverkompatibilitás megtartása mellett. A 20.7. ábrán látható kapcsolás egy 64 kB külső adat- és egy 64 kB külső programmemóriát tartalmaz. Látható, hogy a két memórián és mikroprocesszoron kívül mindössze még egy elemre van szükség a P0 PORT cím-adat szétválasztására.



20.7. ábra: Mikroszámítógép Intel 8031 mikrovezérlővel, 64 kB ROM és 64 kB RAM memóriával

20.4.5. **A beépített időzítő/számláló áramkörök**

Sokszor szükséges egy rendszeren belül időzítők ill. számlálók használata (Timer-Counter). A 8051 család minden tagja tartalmaz két-, illetve a ---2 tipusok három belső egységet. A TIMER0, TIMER1 és TIMER2 16 bites számláló. Mindegyik tetszés szerint használható időmérésre, vagy eseményszámlálásra. Időméréskor minden gépi ciklusban inkrementálódik a regiszter belső tartalma, 1 gépi ciklus 12 óraciklus, tehát pl. 12 MHz esetén ez 1 μs. Számláló üzemmódban a külső **T0**, **T1** vagy **T2** pontok **1→ 0** átmenete inkrementálja a számlálót. Az üzemmódok meghatározása a **TMOD** regiszter megfelelő beállításával történik (20.8. ábra).

TMOD							
MSB							**LSB**
GATE	**C/-T**	**M1**	**M0**	**GATE**	**C/-T**	**M1**	**M0**
TIMER1				**TIMER0**			
ha 1, a TCON TR1 =1 és INT1= 1, ha 0, INT1 nem hat a	**ha 1, számlálja a T1 impulzu- sait, ha 0, akkor időzítõ**	**00 TL 5-bites elõosztó 01 TH/TL 16 bites időzítõ/számláló 10 8 bites automa-tikusan újratöltõdõ időzítõ/számláló, TH töltõdik TL-**		**ha 1, a TCON TR0 =1 és INT0=1, ha 0, INT0 nem hat a működés-re**	**ha 1, számlálja a T0 impulzu-sait, ha 0, akkor időzítõ**	**00 TL 5-bites elõosztó 01 TH/TL 16 bites időzítõ/számláló 10 8 bites automatikusan újratöltõdõ időzítõ/számláló, TH töltõdik TL-**	

működésre		**be 11 A Timer/counter leállítása**			**be 11 8 bites időzítő v. számláló, normál TIMER0 vez. bitek kezel- nek, TH0 8 bites időzítő, TIMER1 vez. bitek kezelnek.**

20.8. ábra: A TMOD regiszter bitjei

A mikrovezérlőnél a következő üzemmódok állíthatók be:

- **MODE 0**: 13 bites számláló, 8 bites alapszámláló és egy 5 bites elõosztó. Ha idõzítõ, akkor a legnagyobb értékrõl (csupa 1) 0-ra való átmenetkor 1 értékre állítja a **TCON** regiszterben levõ megszkításfleget (9. ábra). Ha számláló, akkor **TRx = 1** és/vagy **GATEx = 0**, vagy **-INTx = 1** (**GATE = 1**-nél).
- **MODE 1**: megegyezik a **MODE 0**-val, csak a számlánc 16 bites.
- **MODE 2**: 8 bites újratöldődő, autómatikusan kezdőértékre álló számláló. **TL** regiszterben történik a számlálás, ha ez túlcsordul bebillenti a megszakításbitet (**TFx**) a **TCON** regiszterben, áttölti **TH**-ból az értéket **TL**-be.
- **MODE 3**: míg az előző 3 üzemmód megegyezik a **TIMER0** és **TIMER1** esetében, itt eltérés van a két számlálónál. A **TIMER1**-nél a pillanatnyi értéket változatlanul őrzi, mintha **TR = 0** beállítás lenne. A **TIMER0** esetében 2 db 8 bites számláló működik, egymástól függetlenül.

TCON							
MSB							**LSB**
TF1	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**
Timer1 túlcsordulás jelzõ bit 1-re íródik, a megszakítás rutinra való ugrás törli	**Timer1 futásen gedélyező bit, szoft-ver állítás, 0 leáll, 1 engedélyezett**	**Timer0 túlcsordulás jelzõ bit 1-re íródik, a megszakítás rutinra való ugrás törli**	**Timer0 futásen gedélyező bit, szoft-ver állítás, 0 leáll, 1 engedélyezett**	**INT1 él-jelzõ flag. 1 lesz, ha a INT1 bemene-ten lefutó- él keletke-zik**	**INT1 bemenete él-aktiv ha 1, LOW aktiv szint ha 0**	**INT0 él-jelzõ flag. 1 lesz, ha a INT0 bemene-ten lefutó- él keletke-zik**	**INT0 bemenete él-aktiv ha 1, LOW aktiv szint ha 0**

20.9. ábra: A **TCON** regiszter bitje

20.4.6. **Soros adatátvitel és soros port vezérlő és állapot regiszter**

A soros port egyidejű adást és vételt biztosít az **SBUF** regiszteren keresztül. Egy cím létezik a regiszter elérésére, de ez fizikailag egy bemenő-, ill. egy ettől teljesen független kimenő regiszter. Adásnál az **SBUF**-ba írt adat autómatikusan elindítja a soros átvitelt. A soros port négyféle üzemmódjának leírása az 20.5. táblázatban található.

20.5. táblázat: A soros port (SBUF) üzemmódjai			
MODE 0	**MODE 1**	**MODE 2**	**MODE 3**
A soros adat az RxD ponton lép be, a TxD ponton keresztül lép ki. Az átviteli sebesség rögzített, 1/12 oszcillátor frekvencia. 8 bit lép be, ill. 8-at küld ki, 1. bit az LSB.	**8 bit előtt startbit=0, utána stopbit=1, először LSB, adás RxD, vétel TxD. Vételkor SCON RB8-ba kerül a stopbit. Az atviteli sebesség (Baud rate) változtatható.**	**Adás RxD, vétel TxD, startbit=0, 8 adatbit után 9. bit 0 v. 1, SCON TB8-ból, végül stopbit=1. Vételnél a 9. bit kerül SCON RB8 bitjebe. Bauud rate az oszcillátor frekvencia 1/32 v. 1/64 része.**	**Teljesen megegyezik a MODE 2 üzemmóddal, de itt az átviteli sebesség állítható.**
Baud rate= oszcillátorfrekv./ 12	**Timer1 (---2-nél Timer1 vagy Timer2).**	**PCON SMOD=0, Baud rate= oszcillátorfrekv./64, PCON SMOD=1, Baud rate= oszcillátorfrekv./32**	**Timer1 (---2-nél Timer1 vagy Timer2).**

Mind a 4 üzemmódban adás megindul **SBUF**-ba való íráskor, a vétel megindítható:

- **MODE 0**-ban **RI=0** és **REN=1** feltételekkel,
- más üzemmódban **REN=1** esetén a beérkező **startbit=0** értékkel.

A soros port vezérlő- és állapotregiszter (**SCON -** Serial Port Control Regiszter) bitjeinek elrendezése a 20.10. ábrán látható.

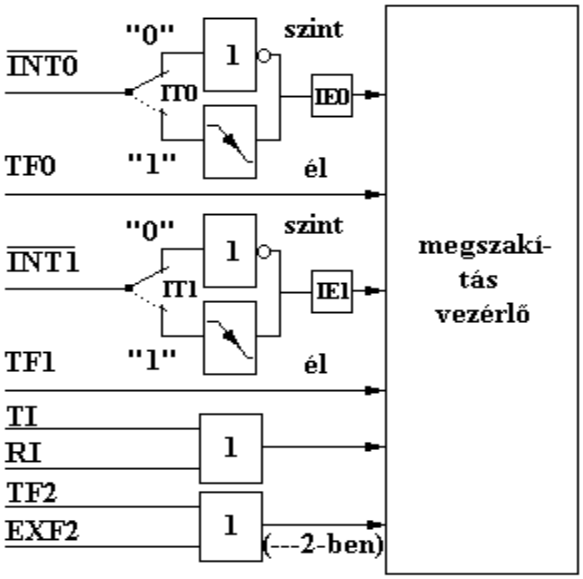
SCON							
MSB							**LSB**
SM0	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**
üzemmód választó bitek		**a 2. és 3. Üzem-módban lehe-tõvé teszi a multi-procesz sszoros**	**a soros vételt engedélye zõ bit, szoftver úton állítható, 1 enged, 0 tilt**	**a 2. és 3. üzemmód-ban a küldendõ 9. bit tárolása, szoftver úton állítható**	**a 2. és 3. üzemmód ban a vett 9. Bit., 1. Üzemmód ban ha SM2=0, stopbit. 0. Üzemmód**	**adás meg-szakítás bit, 0. Üzem-módban a 8. Bitidõ után, a**	**vétel meg-szakítás bit, 0. Üzem-módban a 8. Bit kezde-tén, a**

	kommu - nikációt			**ban nem használa-tos**	**többi üzem-módban a stopbit kezdetén vált 1-re.. Szoftver úton törolhetõ**	**többi üzem-módban stopbit félide-jén vált 1-re. Szoftver úton tö-rölhetõ**

20.10. ábra: Az SCON regiszter bitjei

20.4.7. **Megszakításkezelés (Interrupt)**

A ---1 mikrovezérlõ változatok 5, a ---2 változatok pedig 6 megszakítási lehetõséggel



(foorással) rendelkeznek (20.11. ábra és 20.6. táblázat).

20.11.ábra: A **8051** megszakítás forrásai

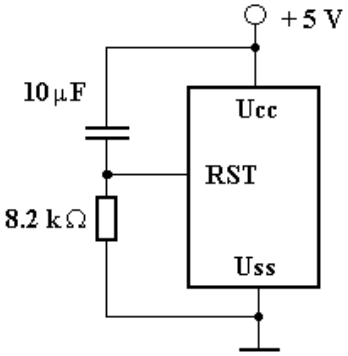
A megszakításforrások leírása a 20.6. táblázatban találhatók meg.

20.6. táblázat: A 8051 megszakítás források					
-INT0	**-INT1**	**TF0**	**TF1**	**TI/RI**	**TF2/EXF2**
(IT0)	**(IT1)**				**(---2-nél)**
Él-aktív vagy szint aktív külsõ megszakítás		**Timer0 túlcsordulás jelzõ bit**	**Timer1 túlcsordulás jelzõ bit**	**Soros átvitel megszakítás**	
Állítás TCON regiszterben, 20.12. ábra					

A megszakításengedélyező bitek és azok jelentése a 20.12. ábrán talákható meg.

IE							
MSB							**LSB**
EA	**-**	**ET2**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**
Globális engedélyezés/tiltás, ha 0, semmilyen megszakítást nem fogad el, ha 1 akkor minden bit önállóan		**A Timer2 túlcsordulás vagy kiolvasás megszakítás (TF2/EXF2). Ha ET2=0, akkor a megszakítás tiltott.**	**A soros port megszakítás (TI/RI) engedélyezõ bitje. Ha ES=0, akkor nem fogadja el a megszakítást.**	**A Timer1 túlcsordulás megszakítás (TF1) engedélyezõ bitje. Ha ET1=0, ez a megszakítás tiltott.**	**Az -INT1 külsõ megszakítás kérõ bemenettel generált megszakítás kérést engedélyélyezõ bit. Ha EX1=0, a megszakí-**	**A Timer0 túlcsordulás megszakítás (TF0) engedélyezõ bitje. Ha ET0=0, ez a megsza**	**Az -INT0 külsõ megszakítás kérõ bemenettel generált megszakítás kérést engedé**

tilt vagy engedé-lyez					**tás tiltott.**	**kítás tiltott.**	**lyélyezõ bit. Ha EX0=0, a megsza kítás tiltott.**



20.12. ábra: Az **IE** (megszakítás engedélyezés) regiszter bitjei

20.4.8. A rendszer alaphelyzte

A 20.13. ábra szerinti kapcsolás biztosítja a rendszer alapállapotát induláskor, de ugyanezt az alaphelyzetet **RST** bemenet aktivizálása bármikor létrehozza.

20.13. ábra: Automatikus bekapcsolási RESET

A regiszterek bekapcsolás utáni alaphelyzete a 20.8. táblázatban található.

20.8. táblázat: Regisztertartalmak RESET folyamat után					
regiszter	**tartalom**	**regiszter**	**tartalom**	**regiszter**	**tartalom**
PC	**0000**	**IP(8052)**	**xx000000**	**TH2**	**00**
ACC	**00**	**TMOD**	**00**	**TL2**	**00**
B	**00**	**TCON**	**00**	**RCAP2H(---2)**	**00**
PSW	**00**	**T2CON(---2)**	**00**	**RCAP2L(---2)**	**00**
SP	**07**	**TH0**	**00**	**SCON**	**00**
DPTR	**0000**	**TL0**	**00**	**SBUF**	**határozatlan**
P3-P0	**FF**	**TH1**	**00**	**PCON(HMOS)**	**xxxxxxxx**
IP(8051)	**xxx00000**	**TL1**	**00**	**PCON(CMOS)**	**0xxxxxxx**

20.4.9. **A mikrovezérő kisebb fogyasztású változata**

A kezdetben gyártott nagyobb fogyasztású (**NMOS**) változat mellett megjelent a kisebb fogyasztású **CMOS** változat is. A **PCON** regiszter (20.14. ábra) bitjeinek

programozásával a vezérlő inaktívvá tehető, valamint csökkentett teljesítményű üzemmódba is átkapcsolható.

PCON							
MSB							**LSB**
SMOD	**-**	**-**	**-**	**GF1**	**GF0**	**PD**	**IDL**
Kétszeres Baud rate bit. Ha Timer1=1 a Baude rate generátor, a soros port 1, 2 v. 3 üzemmód -ban dolgozik.				**Általános célra használhat ó flag.**	**Általános célra használhat ó flag.**	**Power down bit. Ha 1, csökkentett teljesítmény felvétel.**	**Idle üzemmód. Ha 1, inaktív állapot.**

20.14. ábra: A **PCON** regiszter bitjei

A mikrovezérlő engedélyezi a szoftver úton tírténő megszakítás létrehozását, egyszerűen a megfelelő bitet 1-re kell állítani, ami a hardver megszakítással egyenértékű megszakítást engedélyez. A megszakítás rutin hardver úton törli a megszakítás bit értékét, de csak él-vezérelt beállítás esetében.

Az előbbiekben ismetetett megszakítások egyenként, vagy globálisan engedélyezhetők ill. tilthatók az **IE** regiszter bitjeinek segítségével (20.12. ábra).

A megszakítások közötti prioritást, vagyis fontossági sorrendet az **IP** regiszter bitjeivel lehet meghatározni, két szint mélységig (20.15. ábra).

IP							
MSB							**LSB**
-	**-**	**PT2**	**PS**	**PT1**	**PX1**	**PT0**	**PX0**
		A Timer2 megszakí-tás (TF2-EXF2) prioritási szintje. PT2=1, magasabb prioritás	**A soros megszakí-tás (TI/RI) prioritási szintje. PS=1, magasabb prioritás.**	**A Timer1 megszakí-tás (TF1) prioritási szintje. PT1=1, magasabb prioritás**	**A -INT1 külső magszakí-tás prioritási szintje, PX1=1, magasabb prioritás.**	**A Timer0 megszakí-tás (TF0) prioritási szintje. PT0=1, magasabb prioritás**	**A -INT0 külső magszakí - tás prioritási szintje, PX0=1, magasab b prioritás.**

20.15. ábra: Az IP regiszter bitjei

Alacsonyabb prioritású megszakítást kiszolgáló rutint csak magasabb szinttel rendelkező megszakítás szakíthat meg, magasabb prioritást kiszolgáló rutint már semmi sem. Azonos időben fellépő megszakítások között egy belső logika dönti el az elsőbbséget a 20.16. ábra szerint.

	forrás	**szint a másodlagos prioritási sorban**
1.	**IE0**	**legnagyobb, legmagasabb**
2.	**TF0**	**lefutó él**
3.	**1E1**	**lefutó él**
4.	**TF1**	**lefutó él**
5.	**RI+TI**	**lefutó él**
6.	**TF2+EXF2**	**legkisebb**

20.16. ábra: Azonos időben érkezett, azonos prioritású megszakítások közötti döntési logika

Megszakításkor (ha az kiszolgálható) egy **LCALL** szubrutinhívó utasítás generálódik, amihez a hardver állít elő egy címet (20.7. táblázat), vagyis a megfelelő megszakítást kiszolgáló program ezen címek valamelyikén kell hogy kezdődjék.

20.7. táblázat: A megszakításokat kiszolgáló rutinok kezdõcímei	
forrás	**kezdõcím**
IE0 (-INT0)	**0003**
TF0	**000B**
IE1 (-INT1)	**0013**
TF1	**001B**
RI/TI	**0023**
TF2-EXF2	**002B**

A megszakítást kiszolgáló program indulásakor (**LCALL**) a programszámláló+1 (**PC+1**) cím a veremtárba mentõdik, de **PSW** nem. A megszakítási rutin végén levõ **RETI** visszatölti a címet a **PC**-be. A **RET** is elvégzi ezt a feladatot, de nem jelzi a megszakítás végének lekezelését.

20.4.10. **A T0 és T1 idõzitõ/számláló és üzemmódjaik (0 - 3)**

A **T0** és a **T1** áramkörök idõzítõként, vagy eseményszámlálóként használhatók. A működési módot a **TMOD**-regiszter 2. és 6. **C/T** jelü bitjeivel (TMOD.2 ill. TMOD.6) kell beállítani. A logikai 1 - hez a számláló üzemmód tartozik.

Az idõzítõ üzemmódban a számtartalom gépi ciklusonként dekrementálódik. Ebbõl következik, hogy az idõzítés alapegysége az oszcillátor periódusidejének a tizenkettede.

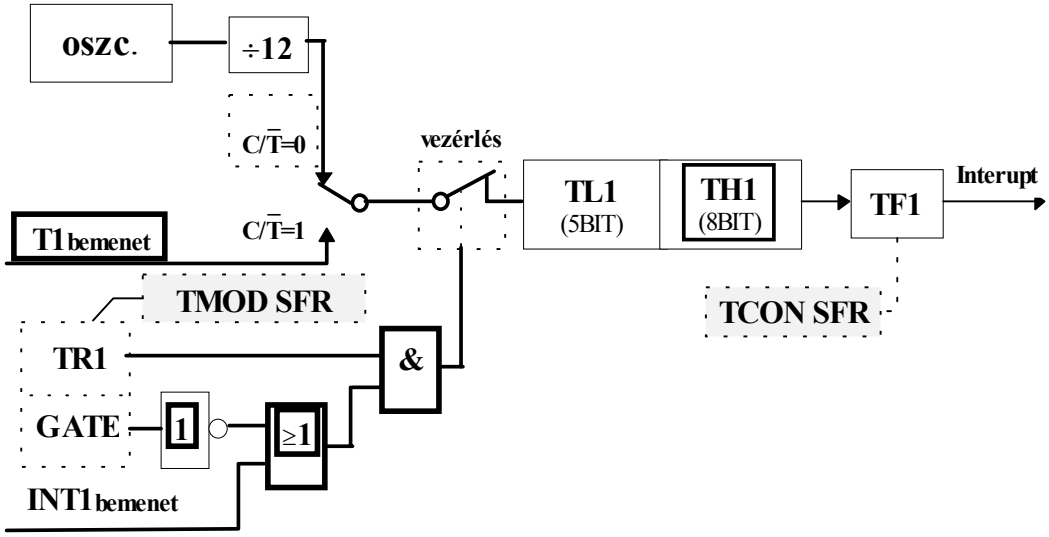
A számláló üzemmódban mindegyik számláló a hozzátartozó bemenetre - **T0**, **T1**, - kapcsolt jel lefutó élénél inkrementálódik. A maximális számlálási frekvencia tehát az órajel frekvenciájának huszonnegyede. A 12 MHz-es oszcillátor használatánál a számlálási frekvencia maximálisan 500 kHz.

Az áramkört kiegészítõ - szoftverbõl vezérelhetõ kapu - elõnyösen használható a pontos eseményszámlálásnál. A kapuzás a **TR0** és **TR1** bitekkel (TMOD.7 és a TMOD.3) történik.

A T0 ill. T1-es jelű áramkörök négy különbözõ üzemmódban működtethetõk. Az üzem-módokat - a **T1**-nél - a **TMOD.4** ($M1_0$) és a **TMOD.5** ($M1_1$), illetve a - a **T0**-nál - pedig a **TMOD.0** ($M0_0$) és **TMOD.1** ($M1_0$) bitekkel kell kiválasztani. Az M0-M1 bitekbõl alkotott bináris szám értéke az üzemmód index-e. Az egyes üzemmódok a következõk:

- **0 - ás üzemmód**

A 0-ás üzemmód a 8048-as mikrokontroller idõzítõ/számlálójának a működésével egyezik meg. A TH0 (TH1) regiszter 8 bites, míg a TL0 (TL1) csak 5 bites. Az áramkörök tehát egy 32-es elosztóval rendelkezõ 8-bites számlálóként használhatók (20.17. ábra).



20.17. ábra: A T0 és T1 számláló 0-ás üzemmódja

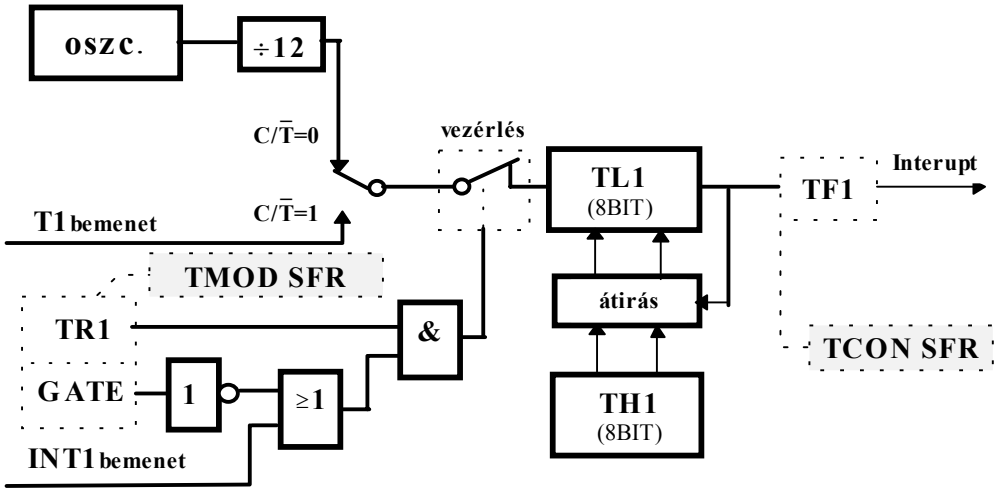
A számláló túlcsordulása 1-be írja a megfelelõ megszakításkérõ bitet (Timer-Interrupt-Flag TF0/TF1). A számlálandó impulzusokat külsõ és belsõ vezérlés együttesen kapcsolja a számlálóra. A feltételek, - a 17. ábrának megfelelõen - hogy a **TR0/TR1** bit **1**- szintű és ugyanakkor a **TMOD.3/TMOD.7** bit alacsony vagy az **INT0/INT1** csatlakozási ponton magas szint legyen. A számlálók külsõ-, (hardver) és belsõ-, (szoftver) vezérelhetõsége nagyon széleskörű, rugalmas felhasználást biztosít.

- **1-es üzemmód**

Az 1-es üzemmód hasonló a 0-áshoz, azzal az eltéréssel, hogy a számláló 16-bites.

• **2-es üzemmód**

A számláló a 2-es üzemmódban 8 bites, de automatikus visszatöltés is történik (20.18.ábra).



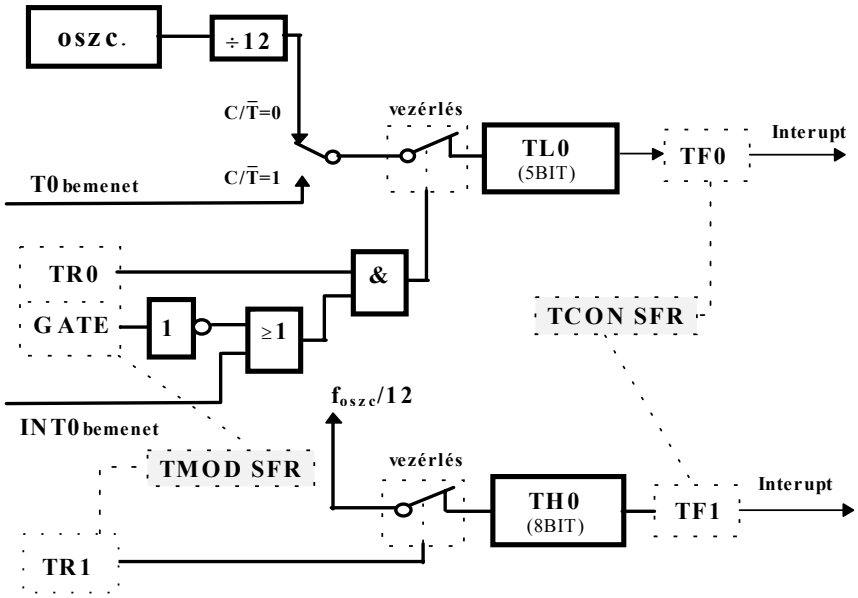
20.17. ábra: A T0 és T1 számláló 2-es üzemmódja

A **TL0/TL1** regiszter tulcsordulása a megfelelõ megszakításkérõ bitet (TF0/TF1) beállítja és ugyanakkor a **TH0/TH1** regiszter tartalmát beírja a **TL0/TL1** regiszterbe. A TH0/TH1 regiszter tartalma nem változik. Ezek felülírása szoftver úton végezhetõ. A megoldással programból változtatható késleltetés valósítható meg.

• **3-as üzemmód**

A 3-as üzemmódba csak a **T0**-ás idõzítõ/számláló áramkör üzemeltethetõ. Ha a T1-et állítjuk 3-as módba, akkor az leáll. Ebben a módban **TL0** és a **TH0** regiszterek önálló 8 bites számlálóként használhatóak és egymástól teljesen függetlenül dolgoznak (20.19. ábra). Az üzemmódot akkor használják, ha több idõzítõ funkció is szükséges. (Megjegyzés: a TH0 csak késleltetõként mûködtethetõ!)

Ha a **T0** a 3-as módban üzemel, akkor **T1**-es áramkör a többi üzemmódban azzal a kikötéssel dolgozik, hogy a belsõ leállítás nem mûködik, és nem kezdeményezhet megszakítást. (A T1 a soros kommunikációs csatorna Baud-rate generátoraként is használható).

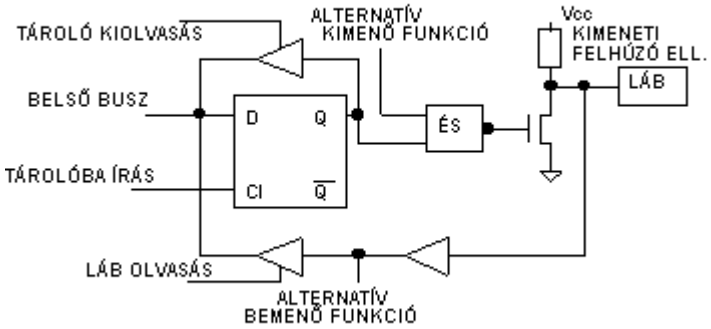


20.19. ábra: A T0 és T1 számláló 3-as üzemmódja

20.4.11. **A portok**

A portokon keresztül történik a mikrokontroller és a külvilág közötti adatcsere. A 8051-es család tagjainál egyre több periféria-funkciót integráltak egy tokba. 8051 portjai eltérnek a hagyományos számítástechnikai I/O áramköröktõl. A mikrokontrollernél egy 8 bites port több funkciót is elláthat. Így pl. busz-meghajtást, vagy soros adatátviteli interfész feladatát.

A négy port tulajdonképpen 32 db. önálló egy-bites port, amelyek bitenként és bájtonként címezhetõek. Ez azt jelenti, hogy bármelyik port-bit egyedileg írható, olvasható is. A portok mindegyik bitjéhez egy-egy tároló (Latch) tartozik, melyek a megfelelõ SFR bitjei. Ezek mindegyikéhez kimeneti meghajtó, és bemeneti illesztõ áramkör kapcsolódik. A legtöbb porthoz - az alap funkció mellett egy-egy alternatív funkció is tartozik, de egy adott alkalmazásban csak az egyik funkció használható. A 20.20. ábrán egy ilyen láb általános kialakítását mutatjuk be, egyes portok áramkörei ettől kisebb mértékben eltérnek, de jól illusztrálja a portok tényleges működését.



20.20. ábra: Egy I/O láb felépítése

Az egyes portokhoz kapcsolódó másodlagos funkciók a következõk:

P0 a külsõ memória írás/olvasás ciklusokban idõmultiplex módon a **cím**-, és **adatbusz** funkciót teljesít az alábbi sorrendben:

- egy gépi ciklus kezdetétõl az **ALE** jel végéig a cím alsó bájtját (A0..A7) adja ki,
- ezt követõen a gépi ciklus végéig adat írás/olvasás feladatát látja el.

P1 csak **I/O** funkciót lát el.

P2 a külsõ memória írás/olvasás ciklusokban **címbusz** felsõ bájtját (A8..A15) adja.

P3 minden egyes bitje más-más funkciójú az alábbiak szerint:

P3 PORT EGYES LÁBAINAK A FUNKCIÓJA

Láb	**Szimbólum**	**Funkció**
P3.0	RXD	Soros vonal bemenete
P3.1	TXD	Soros vonal kimenete
P3.2	INT0	0-ás külsõ megszakítás
P3.3	INT1	1-es külsõ megszakítás
P3.4	T0	0-ás idõzítõ/számláló bemenete
P3.5	T1	1-es idõzítõ/számláló bemenete
P3.6	WR	Író-jel
P3.7	RD	Olvasó-jel

Amikor az egyes portokat az alternatív funkciójukban kívánjuk használni, akkor az alábbiakra kell figyelni:

A portoknak van belsõ felhúzó (Pull-up) ellenállásuk. Az alternatív funkció csak akkor működik, ha a megfelelõ port tárolóban **HIGH** szint van.

Ha a port 1-be írása elmarad, akkor a megfelelõ lábat egy belsõ tranzisztor lehúzza 0-ra. Ilyen esetben egy külsõ Ucc (+5V) szintű jel tönkre teheti az áramkört.

Az egyes portokhoz kapcsolt alternatív funkciók miatt ezek már nem használhatók klasszikus értelemben vett I/O-portnak. A többfunkciós kialakítás alapvetõ oka, hogy a mikrokontroller lábszáma korlátozott.

Az egyes portok bitjeinek áramköri felépítése kismértékben különbözik. A különbözõ felépítésű portok között azonos, hogy mindegyik bithez egy-egy tároló tartozik. Ezek D-flip-flopok, amelyek mindegyike valamelyik SFR egyik bitje. A belsõ buszról az adatot a WRITE LATCH (a D-flip-flop clock-ja) jel írja be a flip-flopba. A tárolóba írt tartalom - a P1 portot kivéve - a belső vezérlõ jeltől függően jelenik meg a hozz tartozó lábon, vagyis csak az alapfunkcióban. Az alternatív funkció esetében a kontroller egy másik SFR tartalmát kapcsolja a lábakra.

Fordított irányban, vagyis amikor egy jelet olvasunk be, akkor két lehetõség van: vagy a kiválasztott lábon lévõ jelet (READ PIN), vagy a tároló (READ LATCH) jelét olvassuk be. Egyes utasítások a tároló olvasásra, mások pedig a lábon lévõ jel beolvasására alkalmasak.

A tárolók csak a kimenő jeleket tárolják. A bemeneti jeleket a beolvasás után külön tárolni kell mert azok elvesznek. A felhasználáskor a programban kell gondoskodni a jelek változásának figyeléséről. Erre az ismételt lekérdezés (polling) alkalmas. Amikor ez nem alkalmazható, akkor külső hardverrel kell megszakítást kérni a figyelt szintváltásnál.

P0 PORT

A P0 alternatív funkciója a külső tárolók és a kontroller közötti adatforgalomhoz kapcsolódik. E port adja egyrészt a címbusz alsó 8 bitjét, másrészt pedig a klasszikus értelemben vett adatbuszt. A belsõ CONTROL jel kapcsolja a port-funkciót (CONTROL=0) ill. a külső memóriakezelői funkciót. Mielött a porton keresztül a külső memóriából olvasás történik, a tárolóba 1 íródik. Ezzel a port eredeti tartalma elvész.

P1 PORT

A 8051mikrokontroller egyetlen portja, amelyikhez nincs alternatív funkció is rendelve. A felépítése is ezért egyszerű. Egy D-tárolóból, két ÉS kapuból, valamint egy végtranzisztorból és felhúzó ellenállásból áll. Az ÉS kapuk a lábról (READ PIN), illetve a tárolóból (READ LATCH) történő olvasást vezérlő belső jeleket kapuzzák.

P2 PORT

A P2 port alternatív funkciója - a P0-áshoz hasonlóan - a külső memória és mikrokontroller közötti adatíráshoz, illetve adatolvasáshoz kapcsolódik. Ilyenkor ez a port adja a címbuszra a cím felsõ 8 bitjét. Ha nem használjuk ezt a portot bemenetként, akkor nem is kell a tárolóba 1-t írni.

A port tárolóiba írt információ a címzési funkció végrehajtása alatt változatlan. Ezért a belsõ CONTROL jel inaktiválódásakor ismét a lábakra kerül az eredetileg beírt információ.

P3 PORT

A P3 port alternatív funkciója bitenként más és más. Egyesek be-, mások pedig kimeneti jeleket szolgáltatnak. A P0 és a P2-es portokkal ellentétben, itt bitenként külön használhatók az alternatív, vagy az alap port funkciók. Egy lábról történő olvasáskor nincs szétválasztva áramkörileg az alternatív-, és a portfunkció. Amikor egy bemeneti jelet (pl.az INT0-át) kívánunk használni, akkor azt a programban inicializálni kell, vagyis a lábhoz tartozó D-tárolóba 1-t kell írni. A tároló kimenete és az alternatív jel között ÉS kapcsolat van. Ezért alapfunkció - normál port - használatakor előbb az alternatív jelet kell 1-be írni. Az alternatív funkció érvényesülése viszont csak a tároló 1-be írásával biztosítható.

20.4.12. **A portok terhelhetõsége (Fan-Out)**

Miután az egyes portok áramköri kialakítása különböző, ezért különbözik az általuk vezérelhetõ TTL bemenetek száma is. A P0 maximálisan 8 LS-TTL bemenetet tud meghajtani. A P1 - P3 portok átlagosan 4 egységgel terhelhetők. Figyelni kell arra, hogy a P0 port kimenetéhez - kivéve, ha csak külső tárolók eléréséhez használjuk - felhúzó-ellenállást kell csatlakoztatni.

20.4.13. READ-MODIFY-WRITE utasítások

A 8051-es családnál mód van arra, hogy csak egy lábon lévő jelet (READ PIN), vagy csak a hozzátartozó tároló tartalmát olvassuk be (READ LATCH). Az olvasási módok között szoftverben választhatunk. Azokat az utasításokat, amelyek a tárolók tartalmát olvassák ki nevezik READ-MODIFY-WRITE (olvas - változtat - ír) utasításoknak. A READ-MODIFY-WRITE típusú utasítások a címzett tároló tartalmát kiolvassák, az adott értékre változtatva írják vissza azt.

20.4.14. A megszakítások és használatuk

A mikrokontroller megszakításainak segítségével oldhatók meg az események valós idejű lekezelése. A megszakítás (Interrupt) egy hardver esemény kiszolgálására szolgál. A kiszolgálást kérő jel a hozzá tartozó megszakításkérõ (Interrupt -Request) flag-et 1-be állítja.

A megszakítást külső, pl. a megfelelő bemenetre érkező feszültségszint, vagy szintváltás, illetve belső esemény, pl. az egyik számláló túlcsordulása kérhet. Mindegyik megszakításhoz meghatározott prioritás-érték rendelhető. Amikor egy megszakítást kérő jel érkezik, akkor az éppen futó program végrehajtása felfüggesztődik és a megszakítást kiszolgáló (Interrupt-Service) rutin fog futni. Ha a mikrokontroller éppen egy azonos, vagy magasabb prioritású megszakítást szolgál ki, akkor csak a futó rutin befejezése után hívja az újabb megszakítás-alprogramot.

Ellenőrző kérdések

1. Mi az egy-chipes mikroszámítógép, vagy más nevén a mikrokontroller, mikrovezérlő ?
2. Jellemezze az Intel 8051 mikrovezérlő család egyes tagjait.
3. Rajzolja le az Intel 8051 mikrovezérlő egyszerűsített vázlatát és ismerttesse az egyes blokkok szerepét.
4. Rajzolja le az Intel 8051 mikrovezérlő részletes vázlatát és ismerttesse az egyes blokkok szerepét.
5. Adja meg az Intel 8051 mikrovezérő család legfontosabb jellemzőit.
6. Mi a jelentése az egyes lábaknak, kivezetéseknek az Inetel 8051 mikrovezérlőnél ?
7. Rajzolja le a 8051 mikrovezérlő csatlakozópontjait.
8. Adja meg az Intel 8051 mikrokontroller belső RAM memóriájának felosztását.
9. Adja meg az SFR (speciális funkció regiszter) felosztását és a regiszterek elnevezését, jelentését
10. Ismertesse a PSW (Program Status Word) regiszter egyes bitjeit.
11. Mi a belső RAM memória bank része ?
12. Ismertesse a 8051 órajelét és annak előállítását.
13. Ismertesse a párhuzamos portokat.
14. Ismertesse a külső adat és programmemóriák 8051-eshez való kapcsolásának alapelveit, adjon példát egy ilyen megoldásra.
15. Ismertesse a beépített időzítő/számláló áramköröket.
16. Mi a TMOD regiszter, mit jelentenek egyes bitjei ?
17. Mi a TCON regiszter, mit jelentenek egyes bitjei ?
18. Ismertesse a soros adatátviteli mechanizmust, az SBUF regisztert és annak üzemmódjait.
19. Mi a SCON regiszter, mit jelentenek egyes bitjei ?
20. Ismertesse a 8051 megszakításkezelési rendszerét.
21. Ismertesse a 8051 megszakításforrrásait.
22. Mi az IE regiszter, mit jelentenek egyes bitjei ?
23. Mia rendszer alaphelyzete és hogyan állítható ez elő ?
24. Ismertesse a rendszer regisztereinek alaphelyzetét az indítás/újraindítás után.
25. Ismertesse a 8051 kisfogyasztású változatait.
26. Mi a PCON regiszter, mit jelentenek egyes bitjei ?
27. Mi a IP regiszter, mit jelentenek egyes bitjei ?
28. Ismertesse a megszakítási rendszer egyes forrásokhoz rendelt hardver címeit.
29. Ismertesse a T0 és T1 időzítő/számláló áramkörök üzemmódjait.
30. Ismertesse a 0-ás üzemmód hardver kapcsolatait.
31. Ismertesse az 1-es üzemmód hardver kapcsolatatit.
32. Ismertesse a 2-es üzemmód hardver kapcsolatait.
33. Ismertesse a 3-as üzemmód hardver kapcsolatait.
34. Ismertesse a kimenő/bemenő lábak hardver kapcsolását.
35. Ismertesse P0 portot és az alternativ funkciókat.

36. Ismertesse P1 portot.
37. Ismertesse P2 portot és az alternativ funkciókat.
38. Ismertesse P3 portot és az alternativ funkciókat.
39. Ismertesse az egyes portok terhelhetőségét.
40. Mik a READ-MODIFY-WRITE utasítások ?

21. A 8051 MIKROVEZÉRLŐ CSALÁD UTASÍTÁSKÉSZLETE

21.1. BEVEZETÉS

Minden mikroprocesszorra, mikrovezérlőre jellemző az autasításkészlete. Az Intel 8051 mikrokontroller család utasításkészlete az Intel 8080 A mikroprocesszor utasításkészletében található utasításokhoz hasonló utasításokkal rendelkezik, természetesen átalakítva a vezérlőben megvalósított hardver–specifikus megoldásokra.

Az utasításkészlet ismertetése elött bizonyos fogalmakkal kell megismerkedni,

21.2. BIT-, ÉS BÁJTSZERVEZÉSÛ MÛKÖDÉS

A 8051-t elsõsorban különbözõ vezérlési feladatok ellátására fejlesztették ki. A mikrokontrollert olyan utasításokkal látták el, amelyekkel nagyon egyszerűen lehet bitműveleteket végezni. A 8080 A mikroprocesszor még nem rendelkezett bitműveletekkel, ami megnehezítette a programozását.
A 8051-nél a bitműveletek végrehajtását szolgálja a bitcímzés és a bitekre is értelmezett logikai műveletek (Boole utasítások). A hagyományos mikroprocesszorokhoz hasonlítva még van egy lényeges eltérés. A mikroprocesszorok fõleg bájt szervezésû számításokat végeznek. Ennek oka abban van, hogy inkább számítási műveleteket hajtatnak velük végre. A vezérlési feladatoknál ritkábban van erre szükség. Többségében egyes kapcsolók állapotát kell lekérdezni, és relék ki-, bekapcsolását vezérelni, LED kijelzőket ki/be kapcsolgatni. Az ilyen jellegű működés bitszervezést igényel.

21.3. AZ UTASÍTÁSOK HOSSZA

A 8 bites szervezés miatt egyes utasítások nem kódolhatók egyetlen bájtban. Ezért szükséges a műveleti kód, adat/cím több bájtban való ábrázolása. Az 1, 2 és 3 bájt szóhosszúságú utasításokat tartalmazó utasításkészlet felöleli a leggyakrabban használatos utasításokat.
A 8051 mikrokontrollernek összesen 111 utasítása van. Az utasítások között egy-, két- és három bájt hosszúakat találunk. Az utasítás bájtszámban a mûveleti kód és - rendszerint közvetetten - az operandusok is benne vannak. Az utasítások között:

- 49 egy-bájt
- 45 két-bájt és
- 17 három-bájt

hosszúságú van.

21.4. UTASÍTÁSVÉGREHAJTÁSI ÉS FUTÁSI IDÕ

Egy program futási idejének pontos meghatározásához ismerni kell azt, hogy az egyes utasítások feldolgozása hány gépi ciklus alatt történik. A 8051mikrokontroller utasításainak a végrehajtása - a **MUL** és **DIV** aritmetikai utasítások kivételével - egy-, vagy két gépi ciklus alatt befejezõdik. Az említett **MUL** szorzás és **DIV** osztás utasítás végrehajtásához négy gépi ciklus szükséges. Egy műveleti ciklus 12

oszcillátor-periódus hosszú. Az órajel frekvenciájának ismeretében a program végrehajtásához szükséges idõ - a futási idõ - a következõk szerint számolható ki:

- **Az program összes utasításaihoz tartozó ciklusok összegét szorozni kell a quartz periódusidejének 12-szeresével .**

A 8051 utasításai - a végrehajtási ciklusszám alapján - az alábbiak szerint oszlanak meg:

- 63 egy-ciklusú,
- 46 két-ciklusú és
- 2 négy-ciklusú.

21.5. AZ UTASÍTÁSOK CÍMZÉSI MÓDJAI

A memóriában levő adat különféle módokon juthat a processzor valamelyik regiszterébe. Egyszerűbb processzoroknál kevesebb, nagyobb teljesítményű processzoroknál több és összetetteb címzési móddal találkozunk.
A 8051-nél az alábbi öt különbözõ címzési mód alkalmazható:

- regisztercímzés,
- direkt címzés,
- indirekt címzés,
- közvetlen címzés,
- indirekt regisztercímzés.

A mikrokontroller különbözõ memóriaterületekkel kommunikál. Így létezik:

- belső adatmemória (belső RAM),
- külső adatmemóri (külső RAM) és
- programmemória (ROM).

A fizikailag, vagy logikailag elkülönített területek más-más címzési móddal érhetõek el.
A 8051 öt féle címzési módja alapvetõen két nagy csoportot alkot:

- direkt és
- indirekt címzésűeket.

A direkt címzésnél az utasítás része a megfelelõ tároló-cella címe (hexadecimális szám). Az indirekt címzésnél a tároló-cella címét egy regiszterben vagy egy másik tároló-cellában van. Az utasítás ez utóbbiak címét tartalmazza.
Az indirekt címzés kissé nehézkesebb, mivel a programozónak elõször arról kell gondoskodnia, hogy a szükséges cím a megfelelõ tároló-egységbe kerüljön. Ugyanakkor rugalmasabb programozást tesz lehetõvé, például egy összetartozó adatblokk egy regiszteren keresztül érhető el. Az indirekt címzés egy másik lehetõsége az un. adatmutató, vagyis a Data-Pointer használata. Az adatmutatóba egy 16 bites szám írható, amely 64 Kbájt kapacitású (külső) adatmemória címzését teszi lehetõvé. Az adott érték elérése relatív címzéssel is megoldható.
A relatív címzés a strukturált programozást teszi könnyebbé. Sok esetben elõnyös ez a módszer.

21.5.1. Regiszter-címzés

Az aktuális bank R0, R1, R2, R3, R4, R5, R6 és R7 regiszterei címzésre is használhatóak. Összesen 4 bank van, mindegyik 8 regisztert tartalmaz, jelölésük R0 és R7 közés esik és a belső RAM memória 00F címtől kezdődő 32 helyét foglalják el. Segítségükkel a belsõ RAM egyes memória elemei - így az A,B a CY és DPTR is - közötti kommunikáció megvalósítható. Az utasításkód három legalacsonyabb helyiértékű bitje határozza meg az aktuális regisztert.

21.5.2. Direkt címzés

A direkt címzésnél az utasítás része az elérendõ memória címe. A cím az utasítás műveleti kódja utáni hexadecimális szám.
Például: MOV A,32H alakú utasítás a 32H című belsõ memória tartalmát az Akkumulátor-ba (A) viszi.

A speciális funkció-regiszterek (SFR-k) csak a direkt címzéssel érhetõk el. Az SFR regisztereken kívül a belsõ RAM alsó 128 bájtja is címezhetõ direkt módon.

21.5.3. Indirekt címzés

Az indirekt címzésnél nem az utasítás, hanem - az aktuális regiszterbank - R0, vagy R1 (és másik nem) regisztere tartalmazza az elérendõ memóriacella címét. A 8 bites tartalommal 256 bájt széles RAM terület címezhetõ. E terület lehet a belsõ, vagy a külsõ RAM-ban is. Természetesen a 8051 típusnál a belsõ RAM csak 128 bájtos. A 8052 típusú mikrovezérlő felső 128 béjtja (80H – FFH) is így címezhető.

Az indirekt címzésre példa a MOVX A,@Ri utasítás, amely az akkumulátorba viszi a külsõ RAM egy cellájának tartalmát. A cella címe az Ri regiszterben van. A magasabb helyiértékû 8 címbit ezalatt nem változik meg. Az utasítás segítségével relatív címzést is meg tudunk valósítani. Elõször a P2 SFR-be kell beírni a cím magasabb bájtját, s az Ri-be pedig az alacsonyabb bájtot.

Az indirekt címzésű utasítások egy bájtosak. Segítségükkel a teljes 64 Kbájtnyi külsõ RAM egy kisebb területe érhetõ el. A teljes memóriaterület címzése a DPTR adat-mutató regiszterrel történhet (@DPTR).
Mivel a verem-terület kisebb 256 bájtnál, akkor a PUSH és POP utasításoknál is indirekt a címzés. A verem-terület címe itt a verem-mutatóban (SP) van. Ezen az alapon a verem-terület is a RAM tetszõleges területére helyezhetõ. A programozónak kell biztosítani a magasabb helyiértékû 8 címbit megfelelõ beállítását.

21.5.4. Közvetlen címzés

A közvetlen címzésnél az adat, amellyel műveletet akarunk végezni nem valamelyik RAM egyik cellájában, hanem az utasításban van. Ez azt jelenti, hogy az értéket - az utasítás részeként - a programmemóriába (ROM, vagy EPROM) kell beírni.
Például: a MOV A,#23H utasítás 23 hexadecimális értéket ír az akkumulátorba. A közvetlen címzést leggyakrabban a matematikai, vagy a fizikai képleteket megoldó programoknál használjuk.

21.5.5. Indirekt regiszter-címzés

Az indirekt-regisztercímzésnél a tényleges fizikai címet két regiszter tartalmának az összege adja. A cím tehát egy bázis-, és egy eltolási (offset) címrészbõl áll. A bázis cím vagy az adatmutatóban (DPTR) (külső RAM címzése), vagy a programszámlálóban (PC) (programmemória címzése) van. E címhez adja hozzá egy belsõ regiszter (pl. @A) tartalmát.

21.6. A KÜLÖNBÖZÕ UTASÍTÁS FAJTÁK

A 8051-nek 111 különbözõ utasítása van. Ezek a következõ négy csoportba sorolhatók:

- adatátviteli utasítások,
- aritmetikai utasítások,
- logikai, ill. bit mûveleti utasítások és
- vezérlõ utasítások.

21.6.1. **Az adatátviteli utasítások: általános-, akummulátor-, és a data-pointeres adatmozgató utasítások**

Az adatátviteli - az un. MOV - utasítások regiszter vagy RAM cella tartalmát viszi át másikba. A PUSH és POP utasítások is a MOV csoportba tartoznak.

Ez az utasítástípus három csoportra bontható.

- *Az általános adatátviteli utasítások*

A csoportot az alábbi utasítások alkotják:

MOV egy bitet, vagy bájtot visz át regiszterbõl regiszterbe, vagy RAM-ba, illetve fordítva.

PUSH inkrementálja a verem-mutató (SP) tartalmát majd a vonatkozó adatot az SP által címzett memóriába írja.

POP az SP által címzett memória tartalmát átírja a vonatkozó helyre, majd dekrementálja az SP tartalmát.

A MOV utasítások a címzett helyrõl az adatot átmásolják a cél helyre, de közben a forrás tartalmát nem változtatják meg.

- *Akkumulátor-utasítások*

Ezeknél az utasításoknál az ACC-regiszter a célja, vagy a forrása az adatátvitelnek. Az akkumulátor önmaga lehet a cél is és a forrás is.

XCH (exchange) az akkumulátor és a címzett memória tartalmát cseréli fel.

XCHD hasonló az XCH utasításhoz, de csak az akkumulátor és a címzett memória tartalmának az alsó négy bitjét cseréli meg.

MOVX a külsõ adatmemória és az akkumulátor között végez adatátvitelt.

MOVC programtárolóból visz egy bájtot az akkumulátorba. A címzésnél a DPTR-ben, vagy PC-ben van a báziscím.

- *Data-Pointer utasítás*

MOV DPTR, # áll a megadott 16 bites állandóval tölti fel a Data-Pointer DPH és DPL regisztereít.

21.6.2. Az arítmetikai utasítások: összeadás, kivonás, szorzás és osztás

A 8051 mikrokontroller matematikai műveletei korlátozottak. Csupán 8 bites elõjeles számokkal lehet műveleteket végezni. Az Overflow-flag (OV) segíti a felhasználót az elõjeles számok összeadásánál és kivonásánál. A 8051 aritmetikai utasításai hasonlítanak a 8080 és a 8085 mikroprocesszorok azonos utasításaihoz.

- *Összeadó utasítások*

ADD a címzett memória és az akkumulátor tartalmát adja össze. Az eredmény az akkumulátorba kerül.
ADDC mint az ADD utasítás, de még a CY flag (átvitelbit) értékét is az eredményhez adja.
DA a BCD számok összeadásakor végez korrekciót.
INC az adott memória tartalmát inkrementálja (1-el növeli).

- **Kivonó utasítások**

DEC az INC utasítás ellentettje. Az adott memória tartalmát dekrementálja (1-el csökkenti).
SUBB az akkumulátor tartalmából levonja az adott bájtot. Ha a CY=1, akkor az eredményböl még 1-t levon. A művelet eredménye az akkumulátorba kerül.

- **Szorzó és osztó utasítások**

A B regisztert (az SFR-ben) kizárólag csak ezeknél az utasításoknál használja a kontroller.

MUL két elõjel nélküli 8 bites számot szoroz össze. A szorzandókat az ACC és a B regiszterekbe kell vinni. A szorzat két bájtos lesz. Az eredmény alacsonyabb helyiértékû bájtja az ACC-be, míg a magasabb helyiértékû pedig a B-be kerül. A 256-nál nagyobb eredménynél az OV 1-be íródik.
DIV az ACC tartalmát osztja a B tartalmával. Elõjelet nem vesz figyelembe! Az osztás egészrészét az ACC fogja tartalmazni. A maradék kerül a B regiszterbe. Ha a hányados 0, akkor az OV flag 1-be íródik.

21.6.3. Logikai és Boole utasítások**.**

A logikai utasítások formailag nagyon hasonlóak a mikroprocesszorok azonos jellegű utasításaihoz. E csoport viszont - az aritmetikai műveletekkel ellentétben - sokkal bõvebb alkalmazási lehetõséget nyújt. A 8051-es mikrokontrollert elsõdlegesen

vezérlésekhez fejlesztették és ezért mind bitekkel, mind pedig bájtokkal tud logikai műveleteket végrehajtani.

ANL logikai ÉS művelet az operendusok azonos helyiértékű bitjei között. Az eredmény az elsõ operandus helyére íródik, míg a második nem változik meg.
CLR törli a direkt címzett bitet, vagy az akkumulátort.
ORL az ANL-hez hasonlóan végez logikai VAGY műveletet.
RL az akkumulátor tartalmát egy hellyel balra forgatja.
RLC az akkumulátor tartalmát a CY közbeiktatásával forgatja egy hellyel balra.
RR az akkumulátor tartalmát egy hellyel jobbra forgatja.
RRC az akkumulátor tartalmát a CY közbeiktatásával forgatja egy hellyel jobbra.

Az RLC és RRC utasításoknál a CY az akkumulátor kilencedik bitjének tekinthetõ.

SETB a direkt címzett bitet 1-be írja.
SWAP felcseréli az akkumulátor felsõ- és alsó négy bitjét.
XRL az EXKLUSIV-VAGY (kizáró VAGY) művelet két operandus azonos helyiértékű bitjei között. Az eredmény - az ANL és ORL műveletekhez hasonlóan - az elsõ operandus helyére kerül.

Az ANL, ORL műveletek egyes bitek között is alkalmazhatóak. A hagyományos diszkrét logikai hálózatok kiválthatók a 8051 bázisú rendszerrel (például a különbözõ tárolt programú vezérlések).

21.6.4. Vezérlõ utasítások: feltétel nélküli call és jump, feltételes jump

A vezérlõ utasítások alkalmazhatók a programokon belüli különbözõ ugrások végrehajtására. Ilyen lehet egy szubrutin hívása, vagy egy feltételtõl függõ programelágazás. A vezérlõ utasítások az alábbi három osztályba sorolhatók:

- feltétel nélküli CALL és JUMP,
- feltételes JUMP,
- megszakítások.

Ezen utasítások közös jellemzõje, hogy a programszámláló tartalmát változtatják meg. A PC határozza meg, hogy a kontroller mely címrõl hív be utasítást. Ennek megváltoztatásával vezérelhetõ egy programelágazás.

A JUMP és a CALL utasítások megtörik a program tiszta sorrendi (lineáris) lefutását.
A **JUMP**-ok (esetleg egy meghatározott feltételtõl függõen) a program meghatározott helyére történõ ugrást vezérlik. A program végrehajtása e helyrõl fog folytatódni.
A **CALL** utasítás egy szubrutint (alprogramot) hív. A rutin feldolgozása egy **RET** utasításig tart. A RET hatására a fõprogram - a CALL utasítást követõ helyrõl - fut tovább. Ugyanaz a szubrutin a fõprogram tetszõleges helyérõl és többször is hívható. Elsõdlegesen a különbözõ összetettebb matematikai műveletekhez használjuk a szubrutinokat.

A megszakítási elágazás, hasonlóan a CALL-hoz egy szubrutin hívását jelenti. Lényeges eltérés az hogy ezt az ugrást egy hardveresemény váltja ki. A szubrutin hívása a megszakítás után azonnal megtörténik.

- *Feltétel nélküli CALL és JUMP*

A feltétel nélküli ugrás, mint ahogyan a nevében is benne van, nem függvénye valamilyen eredménynek. Az ugrás minden esetben bekövetkezik, amikor a program egy ilyen utasításhoz ér. Szigorúan véve a RETURN utasítás is e csoportba tartozik. A CALL végrehajtása elõtt a fõprogram következõ utasításának címe a verembe íródik. Ezután a Verem-pointer (SP) értéke kétszer inkrementálódik (16 bites cím kerül be a verembe). A RET utasítás hatására ez a cím visszaíródik a PC-be. Végül az SP tartalma kettõvel csökken.

ACALL két bájtos szubrutin hívó utasítás. 2 K-bájtos szegmensen belüli programugrást hajt végre. Az ACALL-hoz 11 bites cím tartozik. A PC-ben lévõ legnagyobb helyiértékű öt bit érvényes marad (együtt adják a 16 bites címet). Az ACALL hívásakor a PC tartalma inkrementálódik. Ha az ACALL egy 256-bájtos szegmens utolsó két bájtja, akkor a PC inkrementálása miatt az a következõ szegmensbe kerül.

LCALL három bájtos szubrutinhívó utasítás. Hasonló az ACALL-hoz, de alkalmas a teljes 64-Kbájton belüli tetszõleges című rutin hívására.

AJMP/LJMP lényegében az ACALL és LCALL utasításokhoz hasonlóan használhatók.

SJMP egy rendkívüli ugrás utasítás. A SHORT JUMP használatával csak 256 bájtos területen belüli ugrás oldható meg (-128 és + 127 között)..

JMP @A+DPTR az ugrás címét a DPTR és az akkumulátor tartalmának összege adja. A 8 bites akkumulátor-tartalom egy lapot fog át. A DPTR a teljes 64 Kbájtos programmemória tetszõleges helyére mutathat.

RET az ACALL vagy az LCALL utasításokkal meghívott szubrutinból való visszatérést vezérli. Hatására a PC-be íródik a szubrutin hívást követõ utasítás címe.

- *Feltételes JUMP*

Az elõzõekkel ellentétben az ugrás csak akkor következik, ha meghatározott feltétel teljesül. A feltételes ugrás mindig relatív. Az éppen aktuális helytõl számítottan 8 bites címtávolságon belül lehet a célhely. Ez azt jelenti, hogy a feltételes ugrásoknál az utasítás helyétõl számítva -128, vagy +127 területen belül lehet a cél cím.

A 8051-nek a következõ feltételes ugróutasításai vannak:

JZ akkor következik az ugrás, ha az akkumulátor tartalma 0.

JNZ akkor következik az ugrás, ha az akkumulátor tartalma nem 0.

JC akkor következik az ugrás, ha a Carry-Flag 1.

JNC akkor következik az ugrás, ha a Carry-Flag 0.

JB akkor következik az ugrás, ha egy tetszõleges, direkt címzett bit 1.

JNB akkor következik az ugrás, ha egy tetszõleges, direkt címzett bit 0.

JBC akkor következik az ugrás, ha egy tetszõleges, direkt címzett bit 1.Utána törli a bitet.

CJNE összehasonlítja két regiszter tartalmát. Akkor következik az ugrás, ha a két tartalom nem egyforma. Amikor az elsõnek adott regiszter tartalma a kisebb, akkor a CY is 1-be íródik. Ellenkezõ esetben törlõdik.

DJNZ elõször dekrementálja az adott címen lévõ értéket. Majd ellenõrzi, hogy a csökkentett érték 0 vagy nem és az utóbbi esetben hajtja végre az elõírt ugrást.

A továbbiakban megadjuk a pontos Assembly jelölésüket (mnemonic) is. Az INTEL cég processzoraira és kontrollerjeire érvényes a következõ írási sorrend: elsõ művelet, második az adat-cél és végül következik a forrás. A felhasználói könyvek, katalógusok az alábbi rövidítéseket használják:

Rn az R0 - R7 munkaregiszterek valamelyikét jelöli. Az utasítás e regiszter tartalmára vonatkozik.

direct a belsõ RAM-ban egy cím, I/O port vagy státusregiszter. Az utasításban hexadecimálisan kell megadni.

@Ri az R0,vagy R1 regiszterek tartalma, amely az elérendõ bájt címe.

#data az utasításban megadott 8 bites adat.

#data 16 az utasításban megadott 16 bites adat (a 2. és a 3. bájt).

rel egy relatív cím. A következõ utasítás címéhez viszonyítottan -128 és +127 területen belülre mutathat.

bit jelentheti a 128 "softver-flag" valamelyikét, egy I/O bitet illetve vezérlõ vagy státuszbitet.

21.7. AZ UTASÍTÁSOK HOSSZA ÉS VÉGREHAJTÁSI IDEJÜK

A 21.1. – 21.4 táblázatokban az egyes utasítások hosszát és az oszcillátor periódusában megadott végrehajtási idejét adtuk meg. A felhasználó ezek alapján kiszámíthatja a szükséges memóriaterületet és a futás idõt. Már említettük, hogy egy gépi ciklus hossza 12 oszcillátor periódus.

21.1. táblázat: Adatátvíteli utasítások

Utasítás		Hossz bájtban	Oszcillátor periódus
MOV	A,Rn	1	12
MOV	A,direct	2	12
MOV	A,@Ri	1	12
MOV	A,#data	2	12
MOV	Rn,A	1	12
MOV	Rn,direct	1	24
MOV	Rn,#data	2	12
MOV	direct,A	2	12
MOV	direct,Rn	2	24
MOV	direct,direct	3	24
MOV	direct,@Ri	2	24
MOV	direct,#data	3	24
MOV	@Ri,A	1	12
MOV	@Ri,direct	2	24
MOV	@Ri,#data	2	12
MOV DPTR,#data16		3	24
MOVC A,@A+DPTR		1	24
MOVC	A,@A+PC	1	24
MOVX	A,@Ri	1	24
MOVX	A,@DPTR	1	24
MOVX	@Ri,A	1	24
MOVX	@DPTR,A	1	24
PUSH	direct	2	24
POP	direct	2	24
XCH	A,Rn	1	12
XCH	A,direct	2	12
XCH	A,@Ri	1	12
XCHD	A,@Ri	1	12

21.2. táblázat: Aritmetikai utasítások

Utasítás		Hossz bájtban	Oszcillátor periódus
ADD	A,Rn	1	12
ADD	A,direct	2	12
ADD	A,@Ri	1	12
ADD	A,#data	2	12
ADDC	A,Rn	1	12
ADDC	A,direct	2	12
ADDC	A,@Ri	1	12
ADDC	A,#data	2	12
SUBB	A,Rn	1	12
SUBB	A,direct	2	12
SUBB	A,@Ri	1	12
SUBB	A,#data	2	12
INC	A	1	12
INC	Rn	1	12
INC	direct	2	12
INC	@Ri	1	12
INC	DPTR	1	24
DEC	A	1	12
DEC	Rn	1	12
DEC	direct	2	12
DEC	@Ri	1	12
MUL	AB	1	48
DIV	AB	1	48
DA		1	12

21.3. táblázat: Logikai és boole- utasítások

Utasítás		Hossz bájtban	Oszcillátor periódus
ANL	A,Rn	1	12
ANL	A,direct	2	12
ANL	A,@Ri	1	12
ANL	A,#data	2	12
ANL	direct,A	2	12
ANL	direct,#data	3	24
ORL	A,Rn	1	12
ORL	A,direct	2	12
ORL	A,@Ri	1	12
ORL	A,#data	2	12
ORL	direct,A	2	12
ORL	direct,#data	3	24
XRL	A,Rn	1	12
XRL	A,direct	2	12
XRL	A,@Ri	1	12
XRL	A,#data	2	12
XRL	direct,A	2	12
XRL	direct,#data	3	24
CLR	A	1	12
CPL	A	1	12
RL	A	1	12
RLC	A	1	12
RR	A	1	12
RRC	A	1	12
SWAP	A	1	12
CLR	C	1	12
CLR	bit	2	12
SETB	C	1	12
SETB	bit	2	12
CPL	C	1	12
CPL	bit	2	12
ANL	C,bit	2	24
ANL	C,/bit	2	24
ORL	C,bit	2	24
ORL	C,/bit	2	24
MOV	C,bit	2	12
MOV	bit,C	2	24
JC	rel	2	24
JNC	rel	2	24
JB	bit,rel	3	24

JNB	bit,rel	3	24
JBC	bit,rel	3	24

21.4. táblázat: Vezérlõ utasítások

Utasítás		Hossz bájtban	Oszcillátor periódus
ACALL	addr11	2	24
LCALL	addr16	3	24
RET		1	24
AJMP	addr11	2	24
LJMP	addr16	3	24
SJMP	rel	2	24
JMP		1	24
JZ	rel	2	24
JNZ	rel	2	24
CJNE	A,direct,rel	3	24
CJNE	A,#data,rel	3	24
CJNE	Rn,#data,rel	3	24
CJNE	@Ri,#data,rel	3	24
DJNZ	Rn,rel	3	24
DJNZ	direct,rel	3	24
NOP		1	12
RETI		1	12

21.8. A FLAG-EKET BEFOLYÁSOLÓ UTASÍTÁSOK

A 8051 utasításai közül csak nagyon kevés változtatja a mikrokontroller flagjeit, jelzőbitjeit. A 21.5. táblázatban foglaltuk össze, hogy az egyes flag-ekre melyik utasítás hogyan hat. Az **X** azt jelenti, hogy az adott flaget az utasítás változtatja. A **0** ill. az **1** jelõli azt a konkrét értéket, amelyre a flag mindig beáll a művelet hatására.

21.5.táblázat. A flag-ket állító utasí-tások

Utasítás	**CY**	**OV**	**AC**
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	0	X	
DIV	0	X	
DA	X		
RRC	X		
RLC	X		
SETB C	X		
CLR C	X		
CPL C	X		
ANL C,bit	X		
ANL C,/bit	X		
ORL C,bit	X		
ORL C,/bit	X		
MOV C,bit	X		
CJNE	X		

21.9. READ-MODIFY-WRITE UTASÍTÁSOK

A **READ-MODIFY-WRITE** utasítások egy port tartalmát kiolvassák, a kívánt értékre változtatják és azonnal visszaírják a port-latchbe. A READ-MODIFY-WRITE utasításokhoz mindig egy port címe tartozik. A 21.6. táblázat foglalja össze a csoportba tartozó utasításokat.

21.6. táblázat: READ-MODIFY-WRITE utasítások

Utasítás	**Példa**
ANL	ANL P2,A
ORL	ORL P1,A
XRL	XRL P1,A
JBC	JBC P2.2,re
CPL	CPL P1.1

INC	INC P1
DEC	DEC P1
DJNZ	DJNZ P1,rel
MOV	MOV P2.1,C
CLR	CLR P1.0
SETB	SETB P1.0

A lista elsõ utasításainál azonnal belátható, hogy azok READ-MODIFY-WRITE utasítások. Az utolsó háromnál elsõ látásra kétséges. Ezekre is érvényes, hogy elõször a port kiolvasása, az érték módosítása és a visszaírás követik egymást.

Ellenőrző kérdések

1. **Elsősorban milyen feladatok ellátására fejlesztették ki a 8051-es mikrokontroller családot ?**
2. Mi a bit-, illetve a bájtszervezésű működés ?
3. Mi az utasítás hossza és milyen hosszúságú utasítások vannak a 8051-es mikrovezérlőnél ?
4. Mi az utasításvégrehajtási és futási idő, valamint hogyan számoljuk ki a 8051-esnél ?
5. Mit jelent az utasítás címzési módja és milyen címzési módok vannak a 8051 mikrovezérlőnél ?
6. Mely memóriaterületekkel kommunikál a 8051-es ?
7. Jellemezze a direkt címzési módot .
8. Jellemezze az indirekt címzési módot .
9. Mi a regiszter címzés ?
10. Mi a direkt címzés ?
11. Mi az indirekt címzés ?
12. Mi a közvetlen címzés ?
13. Mi az indirekt-regiszter címzés ?
14. Milyen csoportokba sorolhatók a 8051-es mikrovezérlő utasításai ?
15. Ismertesse az adatátviteli utasításokat .
16. Ismertesse az aritmetikai utasításokat .
17. Ismertesse a logikai és Boole utasításokat .
18. Ismertesse a vezérlő utasításokat .
19. Ismertesse a flageket befolyásoló utasításokat .
20. Mi a READ-MODIFY-WRITE utasítás ?

22. A PIC16cXX MIKROVEZÉRLÕ CSALÁD

22.1. BEVEZETÉS

A PIC mikrovezérlő család **CMOS** technológiával készül. A mikrovezérlő család tagjai azonos **CPU** maggal készülnek, különféle belső programtár-típus hozzáadásával.

Található olyan változat, amelyben a program módosítható, ez a belsõ **EPROM**-os, kvarcablakos kivitel, drágább ugyan, de programfejlesztésre kiválóan használható, a végleges program maszkprogramozott **ROM**-ba kerül (**OTP** – One Touch Programming). Ha kis példányszámról van szó, akkor a **ROM**-os változat maszkjának elkészítése nem kifizetõdõ, ezért erre a célra is az **EPROM**-os változatot használják, azzal hogy nem drága kvarcablakos kivitelben, hanem olcsó műanyag tokozással, amiből a betöltött program később már nem törölhető. A Michrochip PIC vezérlők csak belső programtárukból hajtanak végre programot, külsõ programtár nem is csatlakoztatható hozzájuk (teljesen zárt Harvard struktúra).

A **PIC** mikrovezérlők utasításkészlete alig több mint harminc utasításból áll, amelyeknek többsége egyetlen gépi ciklus alatt végrehajtódik. A mikrovezérlők nyolc bites adatokat kezelnek, de utasításaik hosszabbak, általában 12-14 bitesek.

A PIC mikrovezérlők négyféle órajelgenerátorral működhetnek:

- RC oszcillátor (DC...4MHz),
- kvarc, illetve rezonátoros oszcillátor (0,1...4MHz),
- nagysebességű kvarc-, illetve rezonátoros oszcillátor (4...20Mhz) és
- kisfogyasztású kvarc-, illetve rezonátoros oszcillátor (DC...40kHz).

A kvarcablakos tokozású kivitelnél az óragenerátor típusa programozható, megadható, a zárt tokozásúaknál pedig gyárilag beállított.

A bekapcsolási **RESET** folyamat után egy belső időzítő egység addig nem engedélyezi a mikrovezérlő működését, míg az oszcillátor jelei nem állnak be végleges, stabil impulzusra. A mikrovezérlő hibátlan működését **WATCH-DOG TIMER** felügyeli, ez biztosítja azt, hogy az esetleges mikrovezérlő hiba ne okozzon vezérlési problémát. Ha nem szükséges a folyamatos programvégrehajtás, lehetséges a kis fogyasztású **SLEEP** állapot használata is. Ebből az állapotból kihúzható a vezérlő, így folytatni tudja a normális működését.

A mikrovezérlők a Harvard tipusu számítógépek csoportjába tartoznak, így belső, operativ memóriájuk külön adat- ill., programmmemóriából áll.

A PIC mikrovezérlő változatok a chip-en belül hagyományos mikroszámítógép elemeket tartalmaznak:

- CPU
- programmemória,
- adatmemória,
- párhuzamos, bitenként kimenetnek, bemenetnek használható portokat,
- időzítő-számláló egységet.

A PIC mikrovezérlők belsõ egységei, a mikroszámítógépnél használatos busz-hierarchiával kapcsolódnak egymáshoz, azzal a különbséggel, hogy külön-külön kezeli a rendszer az adat-, illetve programrészt.

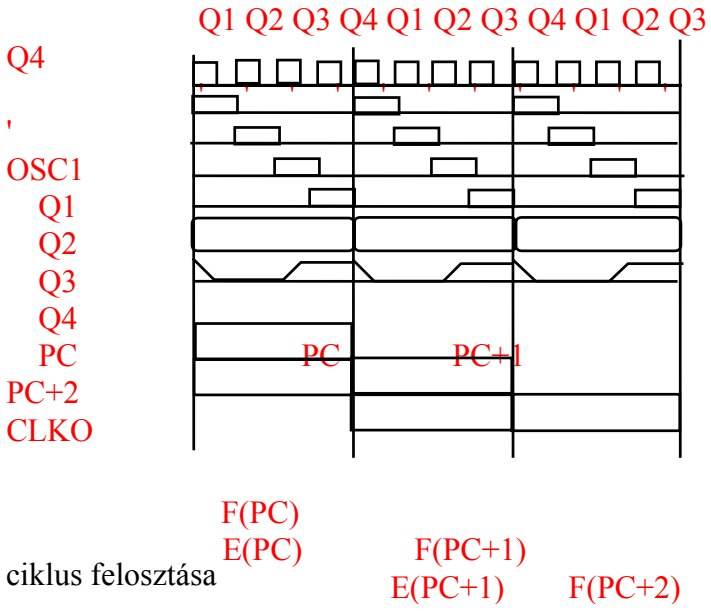
A következő sinek találhatók meg a mikrovezérlőben:

- címbusz, a programelem 13 bites busza és
- nyolcbites adatbusz.

A különválasztott buszok lehetővé teszik, hogy amíg a CPU egy utasítást végrehajt, azalatt már a következő utasítás beolvasásra kerüljön. Az így megvalósuló, átlapolt utasításbeolvasás és -végrehajtás eredménye az, hogy az utasítások zöme egyetlen gépi ciklus alatt végrehajtódik.

A mikrovezérlő oszcillátorában kialakuló impulzussorozatot, órajelet, a mikrovezérlõ idõzítõ egysége, néggyel leosztva, négy egymást át nem lapoló (**Q1**, **Q2**, **Q3** és **Q4**) órajellé alakítja. A négy, egymást követõen fellépõ óraimpulzus alkot egy gépi ciklust. Az így létrehozott órajel a 22.1. ábrán látható. A Q1...Q4 impulzusok pontosan meghatározott működési fázisokat jelentenek a gépi cikluson belül, ahol az egyes feladatok a következők:

- Q1 fázis alatt történik a PC inkrementálása,
- Q2 és Q3 idején a megcímzett rekeszből bekerül az utasítás a CPU-ba,
- Q4 alatt kerül az utasítás az utasításregiszterbe,
- a következõ gépi ciklus alatt fog az utasítás dekódolása és végrehajtása megtörténni, miközben egy következő utasításszó már bekerült a processzorba (lásd Q1, Q2, Q3 és Q4).



22.1. ábra: A gépi ciklus felosztása

A PIC mikrovezérlők belső adatbusza két fő egységet kapcsol össze, a nyolcbites ALU-t, és a REGISTER FILE nevű statikus RAM területet. Az alap SRAM (statikus RAM) méret 32 regiszter, ha nagyobb az adattár, akkor 32 bájtos ún. BANK területekre osztható. A bankok kijelölése az FSR (File Select Register) szolgál.

Az adattárban a következő általános célú, valamint speciális funkciójú regiszterek helyezkednek el:

- RTCC, a valós idejű óra/számláló regiszter,
- PC, programszámláló,
- STATUS, állapotregiszter,
- PORTi, porttartalom regiszter,
- FSR, fájlválasztó regiszter.

Több funkciót a RAM-tól elkülönített regiszterekkel kezelnek a mikrovezérlők, ezek a TRISi és OPTION regiszterek.

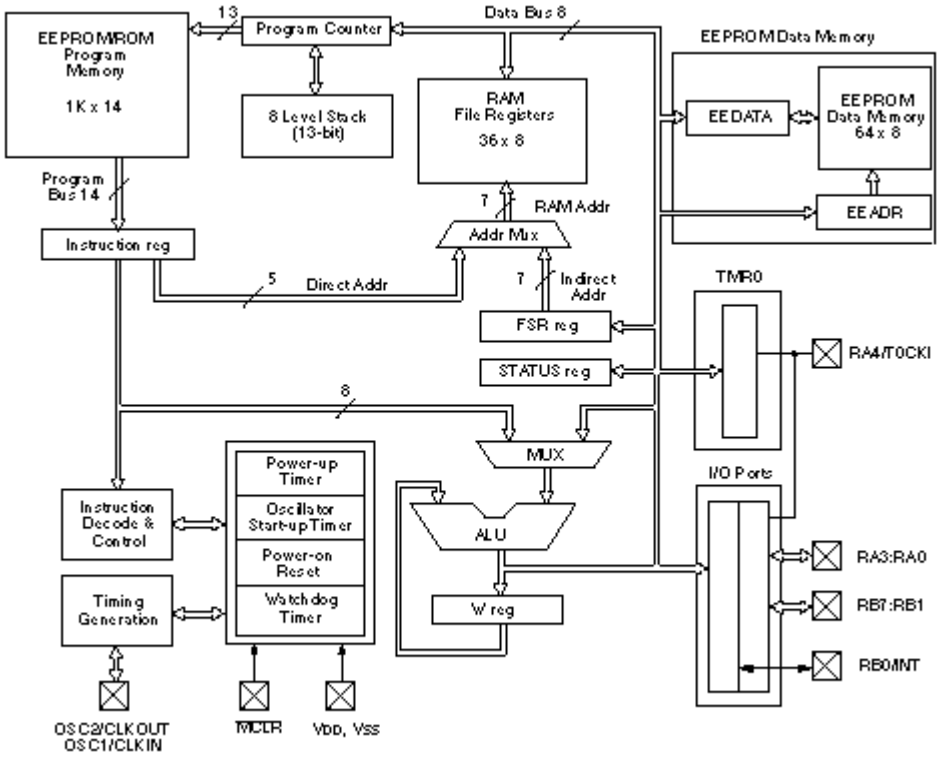
22.2. A PIC 16c71-ES MIKROVEZÉRLÕ

A Microchip kissé átrendezte a mikrovezérlõ belső felépítését az addig használatos struktúrákhoz képest. A belsõ programozható egységeket kizárólag csak fájlregiszteren keresztül lehet kezelni. A **PIC 16c71** mikrokontroller utasításkészlete is bővült néhány utasítással a régebbi típusokhoz képest, az utasításszó hossza pedig 14 bit lett.

A legfontosabb hardveres újdonság, hogy a PIC 16c71 mikrokontroller **A/D** átalakítót is tartalmaz, ami elött négycsatornás multiplexer alakítja át a chipet 4 A/D bemenetes eszközzé. A nyolcbites felbontású A/D egység konverziós ideje 20μs.

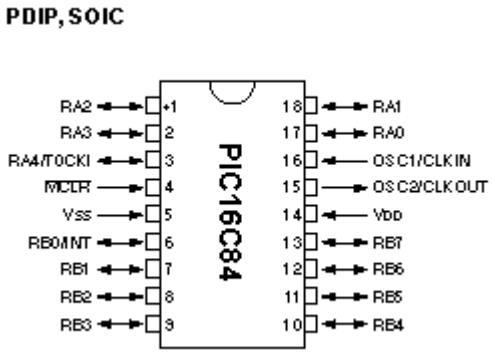
22.2.1. **A PIC 16c71 belső hardver felépítése, regiszterei**

A PIC 16c84 belsõ felépítése a 22.2. ábrán látható, amely teljesen megegyezik a 16c71 mikrovezérlőével, csak az EEPROM adatmemória helyén A/D egység található.



22.2. ábra: A PIC 16c71 mikrovezérlő belső felépítése

A mikrovezérlő csatlakozópontjai a 22.3. ábrán láthatók.



22.3. ábra: A PIC 16c71 csatlakozópontjainak leírása

Az egyes lábak, csatlakozópontok (pin) leírása, jelentése a következő:

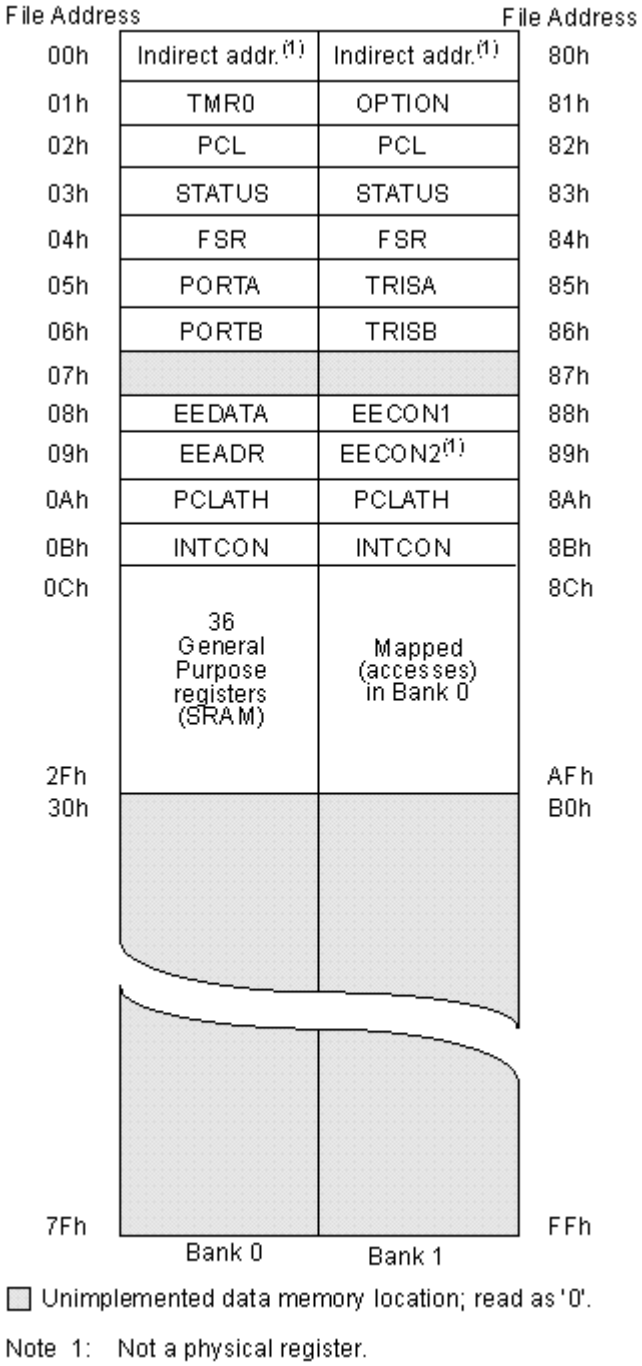
- RA1: analóg bemenet / digitális ki-/ bemenet,
- RA0: analóg bemenet / digitális ki-/ bemenet,
- CLOCKIN: bemenő oszcillátor jel,
- CLOCKOUT: kimenő oszcillátor jel,
- U_{dd} : tápfeszültség,
- RB7: digitális ki-/ bemenet,
- RB6: digitális ki-/ bemenet,

- RB5: digitális ki-/ bemenet,
- RB4: digitális ki-/ bemenet,
- RB3: digitális ki-/ bemenet,
- RB2: digitális ki-/ bemenet,
- RB1: digitális ki-/ bemenet,
- RB0/INT: digitális ki-/ bemenet / külső hardver megszakítás,
- U_{ss}: test,
- -MCLR: RESET vonal,
- RA4/RTCC: analóg / digitális bemenet / számláló léptetőjel,
- RA3: analóg bemenet / digitális ki-/ bemenet / referenciafeszültség,
- RA2: analóg bemenet / digitális ki-/ bemenet.

RESET után a programvégrehajtás a **000h** címen kezdõdik, a megszakítás kezelő szubrutin belépési pontja a **004h** cím.

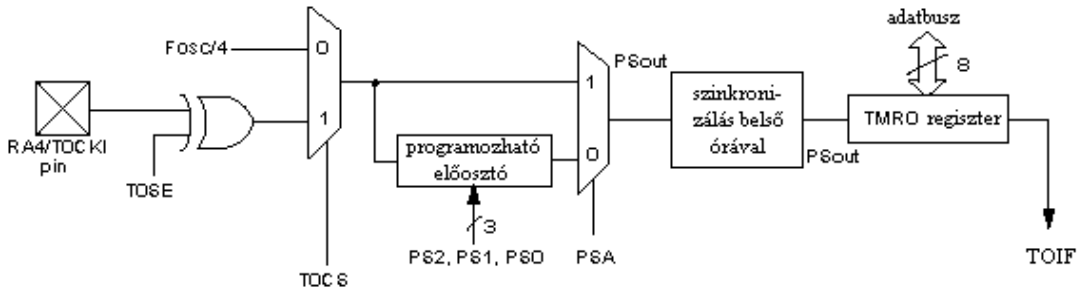
A 22.2. ábrán, a vezérlő belső rajzán látható, hogy a veremtár mélysége nyolc szintre bővült az eredeti PIC 16cXX típusokhoz képest. A 13 bites **programszámláló** (PC) sajátságos felépítésű. Az alsó része (PCL) írható és olvasható regiszter, míg a felső rész (PCH) közvetlenül, a programozó számára nem is írható és nem is olvasható, hanem automatikusan töltődik fel a PCLATCH regiszterből. Ezt a regisztert a CALL és GOTO vezérlésátadó utasítások használják.

A PIC 16c71 adatmemóriája SRAM (statikus RAM) jellegű, 256x8 bit szervezésű belső tár. Az adatmemória minden rekesze elérhető akár direkt, akár indirekt módon, felosztása a 22.4. ábrán látható.



22.4. ábra: A 16c71 mikrovezérlő adatregiszterei

Az egyes regiszterek jelentése a következő:
RTCC regiszter: óra és számláló regiszter, amely írható és olvasható, általános célra használható. A tartalma különböző forrásokból érkező jelek hatására inkrementálódik. Léptethető az RTCC lábon érkezõ impulzusokkal, vagy a belsõ $f_{osc}/4$ frekvenciájú ún. utasításciklus órajellel (ez léphet ki a CLOCKOUT ponton is). Az RTCC belső felépítését a 22.5. ábra szemlélteti. Látható, hogy az RTCC működését vezérlő bitekkel lehet szabályozni. Ezek a vezérlõ bitek (RTE, RTS, PSA, PS0, PS1, PS2), az OPTION regiszterben találhatók.



22.5. ábra: Az RTCC regiszter vezérlése

A PSA bit a nyolcbites elõosztót is az RTCC-hez rendeli, így ezen keresztül jutnak el a léptetõimpulzusok az RTCC számlálójába. Az osztás arányát a PS0, PS1, és PS2 bitek határozzák meg a kõvetkezõ (22.1.) táblázat szerint:

22.1. táblázat: Az osztás arányának beállítása a számlálóban

PS2	PS1	PS0	0-RTCC	1-WDT
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

Ha az elõosztót a számlálóhoz rendeltük, minden RTCC-t író utasítás hatására az elõosztó nullázódik. Az RTS bit határozza meg a léptetõ jel forrását:

- „0” esetén a léptetõ jel az oszcillátor frekvenciájának néggyel leosztott értéke lesz,
- „1” esetén pedig az RTCC lábon érkezõ jel lesz a léptetőjel.

Az utóbbi esetben az RTE=0 a lefutó élet, az RET=1 pedig a felfutó élet jelöli ki aktívnak. Ha a léptetések során az RTCC eléri az FFh értéket a következõ léptetõ jel hatására tartalma nullázódik, amit a TOIF biten keresztül jelez.

22.2.2. **A mikrovezérlő regiszterei**

A **PC** (Program Counter) programszámláló regiszter a következõ, végrehajtandó utasítás címét tartalmazza.

A **STATUS** regiszter az **ALU** állapotáról, és a **CPU** állapotáról informáló biteket tartalmazza. aregiszter egyes bitjeinek jelentése a 22.6. ábrán látható.

D7	D6	D5	D4	D3	D2	D1	D0
IRP	RPI	RP0	TO	PD	Z	DC	C

22.6. ábra: A **STATUS** regiszter tartalma

A STATUS regiszter egyes bitjeinek jelentése a következő:

- **C**: Carry bit, a rotáló és léptetõ utasítások is használhatják, de elsõsorban az összeadáskor keletkezõ túlcsordulás, illetve a kivonás eredményének elõjelét jelzõ bit. Itt fontos megemlíteni, hogy a Carry bit akkor lesz nulla értékű, ha a kivonás eredménye negatív.
- **DC**: Digit Carry bit, az alsó négybites részen fellépõ túlcsordulást, illetve kölcsönt jelzi összeadáskor, illetve kivonáskor.
- **Z**: Zero bit flag, „1”-es értéke jelzi, ha aritmetikai-logikai művelet, illetve adatmozgatás eredménye nulla.
- **PD**: Power Down bit, a tápfeszültség bekapcsolása után egyes értéket vesz fel, ezenkívül a CLRWDT (Clear Watch Dog Timer) utasítás is egyesre állíthatja. A SLEEP utasítás törli.
- **TO**: Time Out bit, értéke „1”-es lesz a tápfeszültség bekapcsolásakor, valamint a CLRWDT és a SLEEP utasítások hatására. A WDT timeout állapota nullázza.
- **RP0,1**: Lapválasztó bitek, bitpáros direkt címzéshez
- **IRP**: Regiszterlap választó bit indirekt címzéshez.

A **TO** és **PD** bitek a CPU-ban végbemenő, különleges eseményekre utalnak a 2. táblázat szerint. Ha a mikrovezérlõ RESET folyamat után van, a RESET kiváltó okára is utal ez a két bit.

2. táblázat: TO és PD bitek jelentése

TO	PD	a RESET folyamat oka
0	0	WDT ébresztett SLEEP-bõl
0	1	WDT timeout, normál
1	0	MCLR ébresztés SLEEP-bõl
1	1	Power up
u	u	MCLR normál mûködéskor

u-változatlan

A **fájlkiválasztó regiszter** (FSR) alsó 5 bitje, a 0...4 bitek a 32 létezõ fájlregiszter címzésére használhatók, az 5. és 6. bit a bankkiválasztó 2 bit. Az összes bank alsó 16 bájtja - fizikailag ugyanaz a 16 bájt- mindig kijelölődnek, ha a címben a bit4 = 0. Ha a cím ötödik bitje „1”-es akkor érvényesülnek a bankkiválasztó bitek (5.,6.). A fájlreferenciás utasításokban a 7. bit szolgál a fájlregiszterek címzésére, tehát csak a lapon belüli cím szerepelhet itt.

Az **OPTION** regiszter egyes bitjei a 22.7. ábrán láthatók.

-RBPU	INTEDG	RTS	PSA	PS2	PS1	PS0	PS0
7.bit	6.bit	5.bit	4.bit	3.bit	2.bit	1.bit	0.bit

22.7. ábra: az **OPTION** regiszter bitjei

Az OPTION regiszter egyes bitjeinek a jelentés e akövetkező:

- -RBPU: A B port felhúzóellenállásainak vezérlése. „0” esetén a felhúzóellenállások engedélyezve vannak, „1”-es esetén pedig tiltva.
- INTEDG: Az aktív él kijelölése az INT külsõ megszakításhoz. Ha értéke „0” akkor a megszakítás lefutó élre aktív, ha „1”-es akkor a megszakítás felfutó élre aktív.
- RTS: Az RTCC jelforrás kijelölése, leírása megtalálható az RTCC regiszter leírásánál .
- RTE: Az RTCC-t léptetõ külsõ jel aktív élét kijelölõ bit. „0” esetén az aktív él a felfutó él, „1”-es esetén pedig a lefutó él.
- PSA: Az előosztót az RTCC-hez rendelő bit.
- PSA0..2: Az előosztás értékét meghatározó bitek, az előosztás értéke megtalálható a táblázatban.

A PORTA és **TRISA** regiszterek az A port működésének meghatározására szolgálnak. Az A port a 16c71-ben ötbites, mivel az **RTCC** léptető bemeneti pont, ha nem használjuk az **RTCC** számoltatására, portbitként használható. Az A port alsó négy pontja (RA0...3) vagy digitális I/O pontként, vagy analóg bemenetként használható. Az A port csatlakozópontjainak alkalmazási lehetõségeit a 22.3. táblázat foglalja össze. A **PORTA** adatregiszter, a **TRISA** pedig iránykijelölő regiszter, melynek csak az alsó öt bitje működik. Az analóg működés programozására az **ADCON1** regiszter szolgál.

22.3. táblázat: Az A port csaatlakozási pontjai

port csatl pont	bit n	funkció	alternatív funkció
RA0/AIN0	bit0	I/O Port,TTL	Analóg bemenet 0.csatorna
RA1/AIN1	bit1	I/O Port,TTL	Analóg bemenet 1.csatorna
RA2/AIN2	bit2	I/O Port,TTL	Analóg bemenet 2.csatorna
RA3/AIN3	bit3	I/O Port,TTL	Analóg bemenet 3.csatorna
RA4/RT	bit4	I/O Port Kimenet:nyitott kollektoros	Külsõ lépteeõ jel az RTCC-hez

A **B** port fontos tulajdonsága, hogy a port felső négy bitjén kialakuló értékváltozások megszakítást eredményezhetnek. A **B** port csatlakozópontjaira, ha kimenetként állítottuk be a portot, egy-egy felhúzóellenállás csatlakozik kb. 250 µA áramerõsséggel. Ha ezekre nincs szükség akkor kikapcsolhatók az **OPTION** regiszter legfelsõ bitjének segítségével. Ha valamelyik csatlakozópont kimenetként konfigurálódik, akkor ezzel automatikusan kikapcsolódik a felhúzóellenállások sorából. A **PORTB** itt is adatregiszter, míg a **TRISB** iránykijelölő regiszter. A **B** portot kezelő regiszterek a 22.4. táblázatban vannak összefoglalva.

22.4. táblázat: a B port csatlakozási pontjai

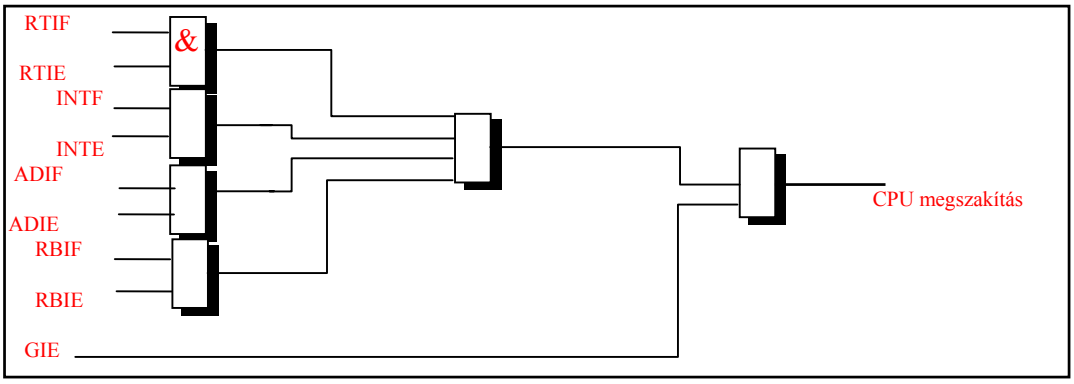
Port csatl. pont	Bit n	Funkció	Alternatív funkció
RB0/INT	bit0	I/O Port,bemenet,TTL programozható felhúzó ellenállás	Külsõ megszakítás
RB1	bit1	I/O Port,bemenet,TTL programozható felhúzó ellenállás	
RB2	bit2	I/O Port,bemenet,TTL programozható felhúzó ellenállás	
RB3	bit3	I/O Port,bemenet,TTL programozható felhúzó ellenállás	
RB4	bit4	I/O Port,bemenet,TTL programozható felhúzó ellenállás	Megszkítás PORT bemenet változáskor
RB5	bit5	I/O Port,bemenet,TTL programozható felhúzó ellenállás	Megszkítás PORT bemenet változáskor
RB6	bit6	I/O Port,bemenet,TTL programozható felhúzó ellenállás	Megszkítás PORT bemenet változáskor
RB7	bit7	I/O Port,bemenet,TTL	Megszkítás PORT

22.2.3. **A PIC 16C71 megszakítás-kezelése, az INTCON regiszter**

A PIC 16c71 mikrovezérlő a korábbi PIC mikrovezérlők egy nagy hiányosságát már nem örökölte, az alkalmazók ugyanis gyakran kifogásolták, hogy az 16c5x elemcsaládnak nincs megszakítás-kezelése. A 16c71-es már négy megszakításforrást tud kezelni:

- Az INT lábon érkezõ külsõ megszakítást,
- Az RTCC túlcsordulását jelzõ megszakítást,
- Az A/D konverzió végét jelzõ megszakítást,
- A B port felsõ négy bitjén jelentkezõ értékváltozás miatt jelentkezõ megszakítást.

A globális megszakítás engedélyezővel letilthatjuk a megszakításokat, és külön-külön mindegyik forrásból jelentkező megszakítás is maszkolható a saját maszkbitjével. A megszakítási rendszer felépítése a 22.8. ábrán látható.



22.8. ábra: A PIC 16c71 mikrovezérlő interrupt logikája

Az ábrán látható megszakítás-kezelő bitek az INTCON, és az ADCON0 regiszterben találhatók meg. Az INTCON regiszter tartalma a 22.9. ábrán látható.

GIE	ADIE	RTIE	INTE	RBIE	RTIF	INTF	RBIF
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

22.9. ábra: AZ INTCON regiszter bitjei

Az INTCON regiszter egyes bitjeinek a jelentése:

- GIE: Általános megszakítást engedélyezõ bit, „1”-es esetén engedélyezi a megszakításokat, ez érvényes a többi maszkbitre is.
- ADIE: Az A/D konverzió végét jelzõ megszakítás engedélyezése.
- RBIE: Az RTCC túlcsordulását jelző megszakítás engedélyezése.
- INTE: Az RB0 lábon érkező külső megszakítás engedélyezése illetve tiltása.
- RBIE: A B port felső négy bitjén fellépő értékváltozás miatt jelentkező megszakítás engedélyezése illetve tiltása.
- RTIF: Az RTCC túlcsordulása. Megszakításjelző bit.
- INTF: Külső megszakítást jelző bit.
- RBIF: A B port értékváltozása miatti megszakítást jelző bit.

A GIE a RESET folyamán „0”-ra vált, így a megszakítások indítás után tiltva vannak. Hasonlóképpen tiltott állapotúra vált a GIE, ha a CPU egy megszakításkérést elfogadott, így minden további megszakítás-elfogadás tiltott. A megszakítás-elfogadás ciklus következő eseménye az, hogy a PC-ből a következő programlépés címe a verembe kerül, majd a PC-be töltõdik a 004h érték. A megszakítás-kezelő rutinnak kell megvizsgálnia a megszakításkérõ flag-eket, és megállapítani a megszakítás forrását. A megszakítást kérő flag-et törölni kell, mielött újra engedélyezzük a megszakítást. A RETFIE utasítás, amely a megszakítás-kezelő rutinból való visszatérés utasítása, visszatölti a megszakítás elötti legutolsó címet a veremből a PC-be, és egyben engedélyezi a globális megszakításkérést (GIE=1).

A külső megszakításkérés az RB0/INT kétfunkciós lábon beérkező kérést jelenti. Ez a kérő jel lehet lefutó, vagy felfutó él - az OPTION regiszterben található INTEDG bittől függően. Ha egy aktív él lép fel az INT ponton, az INTF bit bebillen az INTCON regiszterben. Ha nincs letiltva a megszakítás elfogadása, a CPU elfogadja.

Ha a SLEEP állapotba úgy léptetjük be a mikrovezérlőt, hogy a külső megszakítást engedélyeztük, egy késõbbi megszakításkérő hardver jel „fel tudja ébreszteni” a mikrovezérlőt, a GIE-től függetlenül. Az, hogy az ébredés után mi történik a továbbiakban, már a GIE értékétől függ.

RTCC megszakítás akkor keletkezik, ha az RTCC számláló tartalma FFh értékrõl 00h értékre változik. Ez az esemény bebillenti az INTCON regiszterben az RTIF bitet.

Mivel a SLEEP állapotban a CPU letiltja az RTCC működését, ez a megszakítás nem képes felébreszteni a mikrovezérlőt.

A B port felsõ négy bitjén lévő bitelrendezés változása is okozhat megszakítást. Ha ezek a portvonalak bemenetként szerepelnek, minden utasításciklusban összehasonlítja a CPU a korábbi betárolt állapotot a pillanatnyival. Ha eltérés tapasztalható, ezt az RBIF bit bebillenése jelzi az INTCON regiszterben. Ha a négy felsõ B portbit valamelyikét, vagy mindet kimenetként konfiguráljuk, akkor azzal kirekesztjük az összehasonlító vizsgálatokból. Ez a megszakítástípus is képes arra, hogy felébressze a mikrovezérlőt a SLEEP állapotból. Ha a tárolt legutolsó bemeneti érték, és a pillanatnyi bemenet esetleges különbségét meg akarjuk szüntetni, akkor B portot olvasó utasítást kell kiadnunk.

A negyedik megszakítási forrás a chipre telepített A/D konverter egység. Az átalakítás befejezésekor az INTCON regiszterben egyesre vált az ADIF flag.

22.2.4. **A PIC 16c71 A/D konverziós egysége**

A megszakítási lehetőségek mellett a PIC 16c71 mikrovezérlő legfontosabb tulajdonsága az A/D átalakítás. A chipre integrált A/D modul négy analóg bemenetet tud kezelni, ezekre analóg multiplexeren keresztül egy mintavevő-tartó, majd egy A/D konverter kapcsolódik. A nyolcbites konverter referenciafeszültsége kívülről bevezethető, vagy belül is képezhető a tápfeszültségből. Az A/D átalakító szukcesszív aprokszimációs típusú. A konverzió eredménye az ADRES regiszterben érhető el. Az analóg jelkezelés regisztereit a következő, 22.5. táblázat írja le.

22.5. táblázat: Az analóg jelkezelés regiszterei

Regiszternév	Funkció	Cím
ADRES	A/D eredmény	09h
ADCON0	A/D vezérlés és állapot	08h
ADCON1	A/D vezérlés	88h
INTCON ADIF bit	Megszakítás vezérlés	0Bh

A konverzió elött a megfelelő csatornát ki kell választani, majd a mintavételezéshez elegendő időt kell hagyni. Az A/D egység működési ideje az oszcillátorfrekvencia függvénye. A minimálisan szükséges átalakítás időtartam 20µs. A konverzió indításához „1”-re kell állítani a vezérlő bitet (GO/DONE). Amikor a konverzió befejezõdik, ez a bit automatikusan „0”-ra vált, és egyben aktivizálódik az A/D megszakításkérés is. Az A/D konvertert az ADCON0 és az ADCON1 regiszter bitjeivel lehet vezérelni. A 22.10. ábrán az ADCON0, a 22.11. ábrán az ADCON1 regiszter bitjei láthatók.

ADCS1	ADCS0	ált.	CHS1	CHS0	GO/ -DONE	ADIF	ADON
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

22.10. ábra: az ADCON0 regiszter bitjei

Az ADCON0 és ADCON1 regiszter egyes bitjeinek jelentése:

- ADON: Az A/D modul be- illetve kikapcsolása. „0” állapotban az A/D modul ki van kapcsolva és nem fogyaszt tápáramot. „1”-es állapotban az A/D modul működik.
- ADIF: Az A/D konverzió végét jelző, megszakítást kérő flag, a konverzió végén egyesre vált. Szoftverből kell törölni.
- GO/DONE: Az A/D konverziót indító vezérlő bit. Egyesre állítása indítja az A/D konverziót. Az átalakítás végén hardverileg nullázódik.
- CHS1,0: Ezek a bitek az analóg csatorna sorszámának beállítására szolgálnak.
- bit6: Általánosan használható, írható-olvasható bit.
- ADCS1,0: Az A/D konverzió órajelének kijelölésére szolgáló bitek. Az órajel a 22.6. táblázat szerint határozható meg.

22.6. táblázat: az A/D átalakítás órajelének meghatározása

ADCS1	ADCS2	t_{AD}
0	0	2 t_{osc}
0	1	8 t_{osc}
1	0	32 t_{osc}
1	1	t_{RC}

						PCFG1	PCFG0
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

22.11. ábra: az ADCON1 regiszter bitjei

Az ADCON1 működő bitjeinek leírását a 22.11. ábrán láthatjuk. Az A/D konverzió időegysége a T_{ad}. Ezt az időtartamot az ADCS1,0 bitektől függően, vagy egy független belső oszcillátorból, vagy pedig a PIC órajeléből állítja elő a mikrovezérlő. A T_{ad} idő az egyes bitek konverziós időigénye. A teljes átalakítás időtartama 10 T_{ad} . A T_{ad} legkisebb lehetséges értéke 2µs. Ha a megkezdett A/D konverzió közben a mikrovezérlő SLEEP állapotba kerül, és az időegység az OSC1 jelből volt előállítva, akkor a konverzió félbeszakad. Ha ilyen esetnél a konverziós szinkronizációt az A/D saját RC oszcillátora biztosította, akkor az átalakítás folytatódik, s a konverzió befejezésekor a kialakult megszakításkérés felébreszti a mikrovezérlőt a SLEEP állapotból.

Az analóg bejövő jeleket vágódiódák fogadják, megakadályozva ezzel azt, hogy a mikrovezérlő belső egységeire túlfeszültség jusson. Az analóg jelforrás belső ellenállása 500Ω és 10kΩ között lehet. Amikor egy csatorna ki van jelölve, a bemenet a mintavevő kondenzátorra kapcsolódik (ennek tipikus értéke 51pF). Amikor az ADCON0-ban az ADON vezérlõ bit „1" értékű, s a beállított csatornaszám az érvényesített analóg bemenetre mutat; a bemenet mintavétele folyamatos. A mintavétel akkor zárul le, amikor a konverzió megkezdődik. Mivel nincs külön lehetőség a mintavétel indítására, talán mintavételezés/tartás helyett célszerűbb a PIC 16c71-es esetében követés/tartásról beszélni. A konverzió befejeződése után kb. 1 T_{ad} idő múlva a mintavételezés ismét beindul.

22.3. A PIC 16C84 MIKROVEZÉRLŐ

A Pic 16c71 mikrovezérlőnél ismertetett felépítés teljesen megegyezik a 16c84 mikrovezérlő felépítésével, három részlet kivételével:

- belső programtárolója EEPROM,
- belső EEPROM adattárolója van és
- nincs beépített A/D átalakítója.

Az a tény, hogy belső adattárolója EEPROM (elektromosan programozható és törölhető ROM) típusú biztosítja a gyors, teljesen elektomos úton történő programozást és programtörlést, így ennek a típusnak nincs is ablakoz változata. Ugyanaz a chip szolgálhat programfejlesztésre is és végleges alkalmazásra is, hiszen ára is igen alacsony.

A belső EEPROM típusú adattár lehetővé teszi a működés közbeni adatírást, ami kikapcsoláskor is memgmarad.

Az EEPROM adattár kezelése az EECON1 regiszter segítségével végezhető el. Az EECON1 regiszter felépítése a 22.12. ábrán látható.

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	EEIF	WRERR	WREN	WR	RD

22.12. ábra: az EECON1 regiszter

Az EECON1 regiszter egyes bitjeinek a szerepe a következő:

- EEIF: EEPROM írás befejezést jelző megszakítás flag,
- WRERR: írási folyamatot szakított félbe a RESET,
- WREN: EEPROM írás engedélyezés, 1=engedélyezett, 0=tiltott,
- WR: írás vezérlés, ha 1, írható az EEPROM, az írás végén a hardver nullázza,
- RD: olvasás vezérlés, ha 1, olvasható az EEPROM, időtartama 1 ciklus, olvasás végén a hardver nullázza.

A vezérlőben levő EEPROM adatmemória kapacitása 64 x 8 bit, ami a 00h és 3Fh címeken helyezkedik el. Ez a memória programból írható és olvasható, írni lehet még bele a programozókészülékkel is.

Elérése két regiszteren keresztül lehetséges:

- EDATA adatregiszter, melynek címe 08h és
- EEADR címregiszter, melynek címe 09h.

A vezérlés megszervezése a következő két regiszteren keresztül történik:

- **EECON1 regiszter, melynek címe 88h és**
- EECON2 regiszter, melynek címe 89h, de ez fizikailag egy nem létező regiszter.

A bekapcsolás, illetve RESET után EEDATA és EEADR véletlen értéket kap, míg EECON1 0000x000b értékre áll be.

Programból EEPROM rekeszbe lehet írni, mely írás az előző tartalmat felülírja. Az írási folyamat meglehetősen lassan megy végbe, 10 ms-ig tart. Íráskor a rekesz címét az EEADR regiszterbe kell töltenünk, az adatot pedig a EEDATA regiszterbe. WREN bit értékét 1-re kell állítani, az írás elindításakor WR-t 1-re kell állítani. Az írás megvalósítható a következő programmal:

```
iras:   MOVLW   55h
        MOVWF   EECON2
        MOVLW   Aah
        MOVWF   EECON2
        BSF     EECON1,WR
```

Aban a pillanatban, hogy WR-t a BSF EECON1,WR utasítással 1-be billentettük megindul az írási folyamat. Ezután az írás 10 ms-ig tart, befejezése után WR 0 lesz, valamint a EEIF megszakítás flag 1–re vált. Szoftverrel kell ellenőrizni az írási folyamat végét, ami alatt tiltani kell a megszakítást.

Olvasási folyamatkor a rekesz címét EEADR regiszterbe kell tölteni, RD bitet 1-re kell állítani, amit a hardver töröl, következő ciklusban EEDATA regiszter már tartalmazza a kiolvasott adatot.

Ellenőrző kérdések

1. Milyen technológiával készül a PIC vezérlő család ?
2. Hány utasításból áll a PIC 16c84/16c71 mikrovezérlő utasításkészlete és mennyi idő szükséges végrehajtásukhoz ?
3. Hányféle órajelgenerátorral működhet a vezérlő ?
4. Mely típusú architektúra jellemzi a PIC 16c84/16c71 mikrovezérlőt ?
5. Mely egységekből épül fel a PIC 16c84/16c71 mikrokontroller ?
6. Mely buszok találhatók meg a mikrovezérlőben, mi a szerepük és hány bitesek ?
7. Mi az utasításátlapolás, hogyan játszódik le és mi az előnye ?
8. Rajzolja le a mikrovezérlő egy ciklusát, annak fázisait és mi játsződik le az egyes fázisok alatt ?
9. Sorolja fel az adattárban levő regisztereket és ismertesse szerepüket .
10. Rajzolja fel és ismertesse a PIC 16c84 mikrovezérlő belső felépítését .
11. Rajzolja fel és ismertesse a PIC 16c71 mikrovezérlő belső felépítését .
12. Ismertesse a PIC 16c84 mikrovezérlő csatlakozó pontjait (pin).
13. Ismertesse a PIC 16c84 mikrovezérlő csatlakozó pontjait (pin).
14. Hol kezdődik aprogramvégrehajtás és hol van a megszakításprogram belépési pontja ?
15. Ismertesse a PC programszámláló felépítését .
16. Ismertesse a PIC SRAM memóriájának felosztását .
17. Rajzolja le az RTCC számláló/időzítő regisztert és ismertesse a vezérléséhez szükséges bitek szerepét az OPTION regiszterben .
18. Ismertesse a számláló osztási arányának beállítását .
19. Ismertesse a STATUS regisztert, annak bitjeit .
20. Ismertesse az A port csatlakozópontjait, szerepét a 16c84 és a 16c71 mikrovezérlőknél .
21. Ismertesse a mikrovezérlő megszakításkezelését, az INTCON regiszter bitjeinek szerepét .
22. Rajzolja le a mikrovezérlő megszakításkezelő részét .
23. Ismertesse a 16c71 mikrovezérlő A/D mechanizmusát, az ADCON0 és ADCON1 regiszter bitjeit .
24. Hasonlítsa össze a PIC 16c84 és 16c71 mikrovezérlőket, adja meg a két vezérlő közötti különbségeket .
25. Magyarázza el a PIC 16c84 mikrovezérlő EEPROM adtmemória írását és olvasásást .
26. Ismertesse a 16c84 EECON regiszterének bitjeit .
27. Mennyi ideig tart az írás, mennyi ideig tart az olvasás az EEPROM memóriánál ?

23. A PIC MIKROKONTROLLEREK UTASÍTÁSKÉSZLETE

A különféle PIC mikrovezérlők utasításai nagyon hasonlítanak egymáshoz, sok ponton pedig teljesen azonosak. Ezért célszerű általánosan tárgyalni az utasításkészletet, az eltéréseket megemlíteni.

A PIC család 16c5x vezérlőinek utasítás-szóhosszúsága 12 bit, a 16cxx, vagyis a 16c6x, a 16c7x és a 16c8x típusok utasításai pedig 14 bitesek.

A PIC 16cxx típusoknál minden fájlregiszter 7 bites (fff ffff), a programmemória címek pedig 11 bitesek.

A PIC 16cxx 35 utasításából 33 a PIC 16c5x 33 utasításával megegyezik. Az 23.1. táblázat tartalmazza az utasításokat **abc** sorrendben.

23.1. táblázat: a PIC 16cxx utasításkészlet						
mnemonik, operandus		leírás	ciklus	opkód	flag	megj.
bájt orientált, fájl regiszter utasítások						
ADDWF	f,d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f,d	AND W and f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 1fff ffff	Z	2
COMF	f,d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f,d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f,d	Decrement f, Skip if 0	1, 2	00 1011 dfff ffff		1,2,3
INCF	f,d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f,d	Increment f, Skip if 0	1, 2	00 1111 dfff ffff		1,2,3
IORWF	f,d	Inclusive OR W and f	1	00 0100 dfff ffff	Z	1,2
MOVF	f,d	Move f	1	00 1000 dfff ffff	Z	1.2
MOVWF	f	Move W to f	1	00 0000 1fff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f,d	Rotate left f through carry	1	00 1101 dfff ffff	C	1,2
RRF	f,d	Rotate right f through carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f,d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f,d	Swap halves f	1	00 1110 dfff ffff		1,2
XORWF	f,d	Exclusive OR W and f	1	00 0110 dfff ffff	Z	1,2
Bit orientált, fájl regiszteres utasítások						
BCF	f,b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f,b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f,b	Bit Test f, Skip if Clear	1, 2	01 10bb bfff ffff		3
BTFSS	f,b	Bit Test f, Skip if Set	1, 2	01 11bb bfff ffff		3
Literal (konstansos) és vezérlő utasítások						
ADDLW	k	Add literal to W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal to W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear watchdog timer	1	00 0000 0110 0100	-TO, -PD	
GOTO	k	Go to address	2	10 1kk kkkk kkkk		

IORLW	k	Inclusive OR literal to W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from subroutine	2	00 0000 0000 1000		
SLEEP	-	Go to stanby mode	1	00 0000 0110 0011	-TO, -PD	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR to W	1	11 1010 kkkk kkkk	Z	

A táblázatban található megjegyzések a következők:

1. Ha az utasítás I/O regiszterrel kapcsolatos és annak tartalma maga az operandus, akkor az aktuális érték a lábon levő hardver jel és nem a kimeneti regiszter tartalma.
2. TMR0 regiszterhez kapcsolódó utasításnál (d=1) ilyenkor az előosztó törlódik.
3. PC módosulásnál, illetve a vizsgált feltétel teljesülésekor 2 ciklus végrehajtási időre van szükség, ahol a második ciklus NOP.

A PIC mikrovezérlők Harvard architektúrával rendelkeznek, ami azt jelenti, hogy az adattár és a programtár külön-külön, egymástól függetlenül kialakított egységet alkot. Az adattár 8 bites, hagyományos kialakítású, míg a programtár, így az utasítások is optimálisabb felépítésűek lehetnek. A 16cxx utasítások hossza 14 bit, az utasítás típusától függően optimálisan lehet a műveleti kód és adat/cím viszonyát összehangolni. Ezért tömörebbek, több információt tartalmaznak az utasítások a Neumann struktúrájú gépekhez képest. A PIC 16cxx uatsítások felépítése az 23.1. ábrán látható.

Bájt orientált, fájl regiszteres művelet

13	12	11	10	9	8	7	6	5	4	3	2	1	0
műv. kód						f(fájl#)							

d = 0, ha W a cél
d = 1, ha f a cél
f = 7 bites fájl regiszter cím

Bit orientált, fájl regiszteres művelet

13	12	11	10	9	8	7	6	5	4	3	2	1	0
műv. kód				b(bit#)		f(fájl#)							

b = 3 bites cím
f = 7 bites fájl regiszter cím

Literál (konstans) és vezérlő művelet

13	12	11	10	9	8	7	6	5	4	3	2	1	0
műv. kód						k(konstans,literál)							

k = 8 bites közvetlen érték

23.1. ábra: PIC 16cxx utasítások felépítése

Ellenőrző kérdések

1. Hány bit hosszúságúak az egyes PIC mikrokontrollerek utasításai ?
2. Mi a féjl regiszter, hány bites ?
3. Hány bites a 16cxx programmemóriája ?
4. Hány utasítása van a PIC 16cxx mikrovezérlőnek ?
5. Mely csoportba lehet beosztani a mikrovezérlő utasításait ?
6. Ismertesse a bájt orientált, fájl regiszter uatsításokat .
7. Ismertesse a bit orientált, fájl regiszter uatsításokat .
8. Ismertesse a literal (konstansos) és vezérlő utasításokat .
9. Mely utasítások állítják a C, DC és Z flag-eket ?
10. Hasonlítsa össze a Neumann és Harvard architektúrákat az utasítások kialakítása szempontjából .
11. Adja meg a bájt orientált, fájl regiszter uatsítások felépítését .
12. Adja meg a bit orientált, fájl regiszter uatsítások felépítését .
13. Adja meg a literal (konstansos) és vezérlő utasítások felépítését .

24. IBM PC ÉS KOMPATIBILIS SZEMÉLYI SZÁMÍTÓGÉPEK

24.1. BEVEZETÉS

A mikroprocesszor 1977-es megjelenése lehetővé tette a számítástechnika igen gyors, tömeges elterjedését. Igen rövid idő alatt az ipari vezérlő-, irányító-, adatgyűjtő berendzések nélkülözhetetlen építőelemeivé váltak ezek az eszközök. Nagyon gyorsan igény jelentkezett személyi számítógépek kifejlesztésére, melyek ára alacsony, így akár irodai, akár otthoni használatra is a széles rétegek számára elérhető a berendezés. A kezdeti kísérleteknél 8 bites processzorokat építettek be a számítógépekbe, a hardver aránylag egyszerű volt, de mindig specifikus. Jelentkezett egy másik nagy probléma, ami az olcsó háttértárak hiánya volt. A ZX Spectrum, Atari vagy Commodore a maga idejében forradalmian új berendezés volt alcsony áron, de mindegyik fejlesztés hibája a nem továbbfejleszthető hardver és szoftver.

Az 1981-ben megjelent IBM XT gyártmányú személyi számítógép (PC – Personal Computer) döntően meghatározta és ma is meghatározza a számítástechnika fejlődését. Elérhető áron egy nyitott hardverrel és szoftverrel rendelkező rendszer született és fejlődik állandóan. Az Intel 8088 (illetve a vele teljesen kompatibilis 8086) processzorral megépített számítógép az otthoni, irodai és ipari alkalmazások nélkülözhetetlen eszköze. Döntően megszabta a mikroprocesszorok, merevlemezes háttértárak, az Internet és még sok minden hihetetlen méretű továbbfejlődését.

Az IBM PC gépekre jellemző tulajdonságok:

- nyitott hardver és szoftver felépítés, vagyis az igényeknek megfelelő egységek, programok rendszerbeépítése, ami a gép legkülönbözőbb területeken való hasssználatát teszi lehetővé egy viszonylag egysége magra támaszkodva,
- szabványos, piacon kapható elemekre épülő hardver, ami egyszerű tervezést, kivitelezést tesz lehetővé,
- az újabb gépek lefelé szoftverkompatibilisak, vagyis e régebbi típusokra megírt programok futnak az újabb gépeken,
- az IBM nem szabadalmaztatta a gépet, ami lehetővé teszi a bárki által való gyártását, ezzel versenyhelyzetet hozva létre a piacon.

Meg kell említeni a PC-k azon tulajdonságát is, hogy ipari berendezésekben is előszeretettel használják alacsony ára és gyors működése, szabadon kialakítható hardver-szoftver konfigurációja miatt.

Számunkra azért érdekes használata, mert a mikrovezérlőkkel felépített berendezések hozzáköthetők a PC-hez, akár a soros RS 232 C, vagy a párhuzamos CENTRONIX csatlakozón keresztül, de még a gép buszát felhasználva is létrehozható a kapcsolat.

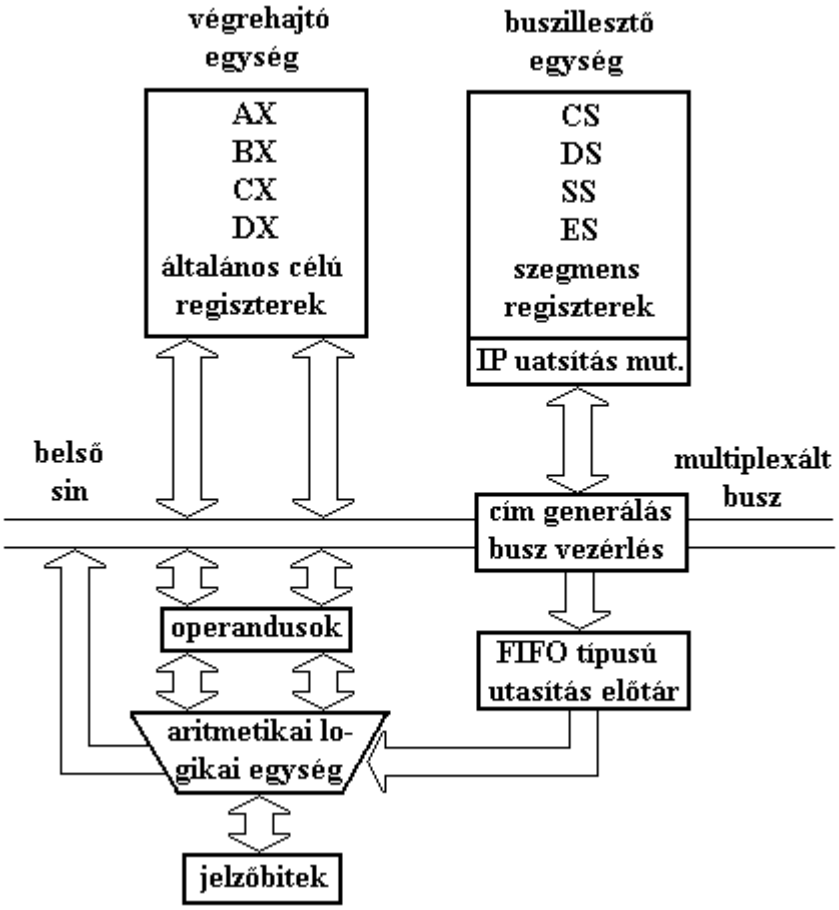
24.2. A 8086, 8088 MIKROPROCESSZOR BELSŐ FELÉPÍTÉSE

Több más komoly céggel ellentétben az Intel fejlesztési staratégia része volt mindig a kompatibilitás, vagyis a hardver és szoftver azon tulajdonsága, hogy a technika, technológia fejlődése ne olyan új megoldások kifejlesztését okozza, melyeknél a régebbi berendezések, eszközök teljesen használhatatlanná válnának. Ennek az elvnek

köszönhetően az Intel gyártmányok ama is nagyban uralják a piacot, ezzel a cég döntően meghatározza a fejlődés mai és jövőbeli irányait.

24.2.1. **Az Intel 8086 processzor felépítése**

Ennek a processzornak 2 változata létezett, a 8 bites külső sinnel rendelkező 8088-as és a 16 bites 8086-os, melyeknél teljes szoftverkompatibilitás létezett. Az 24.1. ábrán látható a processzorarchitektúra.



24.1. ábra: Az Intel 8086 processzor architektúrája

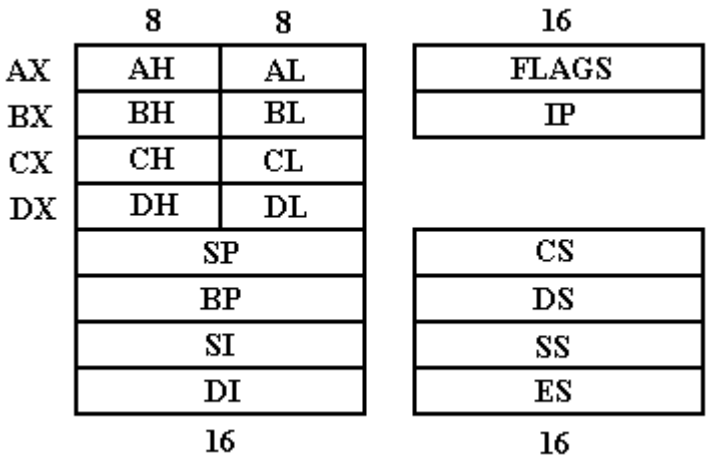
Az ábrán szereplő végrehajtó egység (EU – Execution Unit) és busz interfész egység (BIU – Bus Interface Unit) adatcserét végez egymással, de feladatuk teljesen más. A BIU feladata a külvilággal való kapcsolattartás.

A 24. 2. ábrán az Intel 8086 processzor legfontosabb regiszterei találahtók.

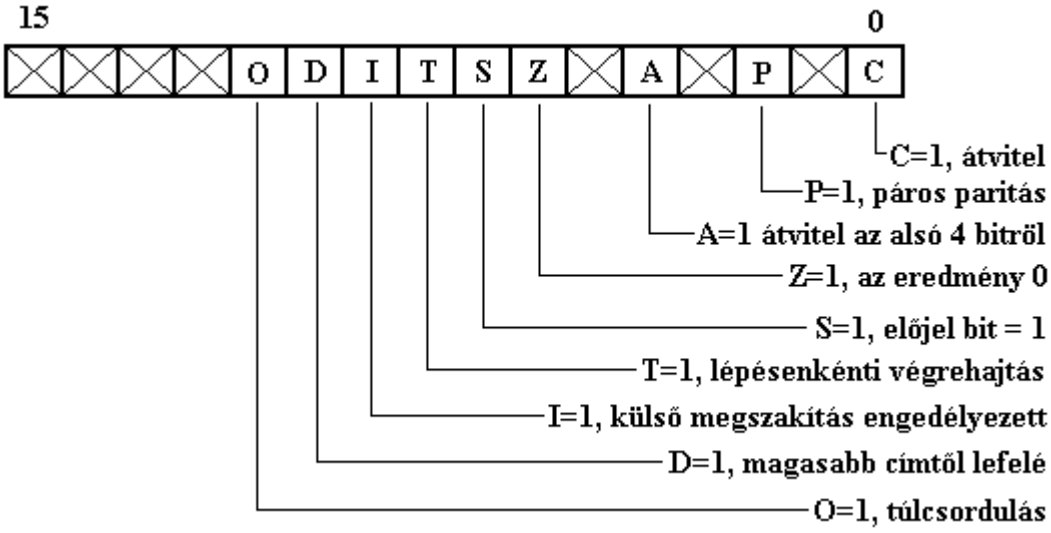
Ez a 16 bites architektúra megörökölt néhány 8 bites gépnél is használt megoldást, pl. a 16 bites akkumulátor 2 8 bitesként is használható, vagy a memória címzésére itt is 16 bites regiszterek szolgálnak, de érdekes megoldásokkal kilépett a 8 bites struktúra korlátai közül.

A processzor regiszterei a következők:

- **AX – általános célú 16 bites akkumulátor, használható mint 2 8 bites akkumulátor is (AH és AL),**
- **BX – 16 bites bázis (base) regiszter, lehet mint 2 8 bites regisztert is használni (BH és BL),**
- **CX – 16 bites számláló regiszter (count), mint 2 8 bites regiszter is használható (CH és CL),**
- **DX – 16 bites adatregiszter (data), mely két 8 bites regiszterként is használható (DH és DL),**
- **SP – 16 bites veremtár mutató regiszter (stack pointer),**
- **BP – 16 bites bázismutató regiszter (base pointer),**
- **SI – 16 bites forrásindex regiszter (source index),**
- **DI – 16 bites célindex regiszter (destination register),**
- **FLAGS – 16 bites regiszter, mely részletesen a 24.3. ábrán látható,**
- **IP – 16 bites utasítás mutató regiszter (instruction pointer),**
- DS – 16 bites adatszegmens-regiszter (data segment), az adatók érhetők el ezzel a regiszterrel,
- CS – 16 bites kódszegmens-regiszter (code segment), programutasítások érhetők el ezen a regiszteren keresztül,
- SS – 16 bites veremtár-szegmens regiszter (stack segment), a veremtár címzésére szolgál és
- **ES – 16 bites különleges memória regiszter (extra segment), egyéb memóriaterületek területek érhetők el ezen a regiszteren keresztül.**



24.2. ábra: Az Intel 8086 legfontosabb regiszterei



24.3. ábra: A jelzőbitek (FLAGS) regiszter bitjei

24.2.2. A 8086 mikroprocesszor működése

Mint a 24.1. ábrán látható, a processzorban két egység található, az EU, vagy végrehajtó egység, illetve a BIU, vagy buszillesztő egység.

A BIU **a** FIFO **(First In First Out – az először beírt adatot veszi ki először) regiszterblokkba előre beolvassa az utasításokat az operativ memóriából, amit** utasításlehívásnak **nevezünk. Az EU végrehajtó egység ebből a FIFO memóriából veszi ki ezek után az utasításokat és az operandusokat, így nincs szükség kivárni a memóriahozzáférés idejét. A BIU másik feladata az operandusok, események továbbítása, átvitele az EU, a memória és az I/O eszközök között. Ezt a feladatot úgy tudja végrehajtani, hogy létrehozza a fizikai címeket egy beépített összeadómű segítségével. Ez a mechanizmus azért lényeges, mert így lehet nagy sebességű processzorok mellett viszonylag lassú memóriákat használni.**

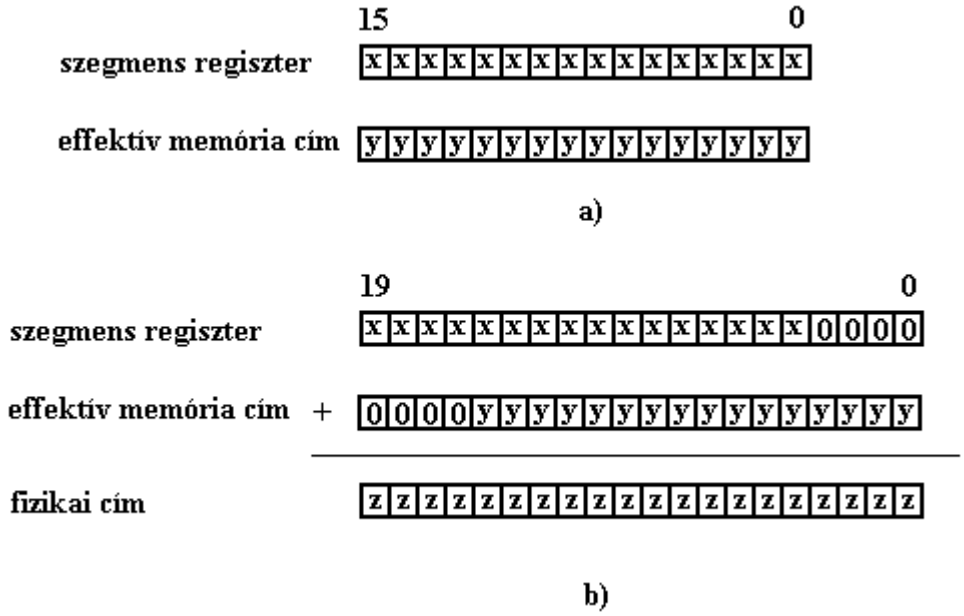
Az EU, a végrehajtó egység az általános célú regeszterekből, az ALU aritmetikai-logikai egységből, az operandusregiszterből és flag, vagy jelzőbitregiszterből áll. Ez az egység értelmezi és hajtja végre a BIU által lehívott utasításokat, valamint adja át az adatokat a BIU-nak.

Összehasonlítva az első 8 bites gépekkel az először nagyobb méretekben elterjedt Intel 8086 16 bites gépet megállapíthatjuk, hogy kb. négyszeres órajelfrekvecia a hatékonyságban már mint tízszeres érték jelentkezik. Ennek oka nemcsak abban keresendő, hogy 8 helyett 16 bites adatátvitellel, ALU-val stb. dolgozik a rendszer, hanem a fent ismertetett átlapolásos **üzemmóddal. Ez a** pipeline **technológia, mely megalapozta a** RISC **struktúrák kialakulását.**

24.2.3. A 8086 címzési technikája

A gépnél alkalmazott 16 bites címregiszterek nem teszik lehetővé 64 kB-nál nagyobb memória közvetlen címzését. Ezt a problémát egy egyszerű technikával

lehet áthidalni, mely a szegmens regiszter tartalmából és egy effektív memóriacím összegéből határozza meg az effektív, fizikai memóriacímet. Ez a megoldás látható a 24.4. ábrán.



24.4. ábra: A fizikai cím meghatározása szegmensregiszter és cím összegeként

A fizikai cím kiszámítása úgy történik, hogy az adott szegmensregiszter tartalmat balra ellépteti a rendszer 4 bittel, tehát szorozza 16-tal (16 x 64 kB = 1 MB), majd az effektív memória címet hozzáadja, a legnagyobb helyérték helyer 4 biten 0 kerül.

A szegmensregiszterek tartalmára nincs semmilyen megkötés, így tetszés szerinti szgmensregisztertartalom átlapolódást lehet használni.

A 8086 címzései a kívetkezők:

- **programmemória címzés és**
- **adatmemória címzés.**

24.2.4. A programozási címzési módok

A kiszámított fizikai címről a processzor bekéri az utasítást, az operandusokkal, címekkel együtt, ez a fetch eljárás, a PC **programszámláló értéke ezután annyival nő, amennyi az utasítás hossza volt (**CISC **struktúra miatt az utasítások hossza változó), így az újabb utasítás kezdetére, a műveleti kódra mutat. Minden esetben, amikor nem következő címről kell az utasítást bekérni, vagyis ugrásra, alprogramhívásra került sor, a következő módon számolja ki a processzor az új cím helyét:**

- **program-relatív, szegmensen belüli címzés, ahol a szegmensen belül egy az utasításban közvetlenül szereplő 8 vagy 16 bites előjeles adat hozzáadódik PC tartalmához,**

- **direkt címzéssel, szegmensek közötti címzéssel, amikor az utasításban mint adat foglal helyet a cím, ez PC-be, illetve a szegmensregiszterbe töltődik,**
- **indirekt címzéssel, amikor egy adatot a JUMP (ugrás), vagy CALL (szubrutinhívás) memóriacímként értelmez.**

24.2.5. Adatmemória címzési módok

Eltérően a 8 bites mikroprocesszoroktól, de akár a RISC struktúrájú gépektől a 8086 igen sok, változatos címzési móddal rendelkezik, amik a következők:

- **közvetlen címzési mód,**
- **direkt címzési mód,**
- **direkt, indexelt címzési mód,**
- **implicit címzési mód,**
- **bázis relatív címzési mód és**
- **stack címzési mód.**

24.3. 80286, 80386, 80486, PENTIUM, PENTIUM II ÉS PENTIUM III PROCESZ-SZOROK

A technológiai fejlődés mindig nagyobb működési sebességet biztosít a mikroszámítógépeknek, amit nemcsak órajelfrekvencia-növeléssel érnek el, hanem párhuzamos feldolgozással, gyorsítómemóriákkal. Az Intel processzorok fejlesztésénél az alapelv az, hogy a későbbi típusok képesek a régebbi, egyszerűbb típusokra fejlesztett szoftver futtatására. Így csak érdekességképpen említjük meg, hogy az újabb 32 bites processzorok természetesen 32 bites általános célú regiszterekkel rendelkeznek, amikor 16 bites szoftverrel dolgoznak, akkor csak az alsó 16 bitet használja.

24.4. AZ IBM PC SINRENDSZERE

24.4.1. Bevezetés

Az IBM PC alaplap a szükséges mikroelektronikai elemek (mikroprocesszor, memória, I/O eszközök stb.) mellett tartalmaz csatlakozókat (slot), melyek az IBM PC mikroszámítógép külső buszának jeleit tartalmazzák. Több fajta csatlakozó használatos, szerepük akkor van, ha a számítógép hardverét szeretnénk új egységekkel bővíteni, pl. EPROM programozóval, MODEM-mel, A/D, D/A kártyával stb.Természetesen általunk fejlesztett eszközök is csatlakoztathatók a rendszerhez ezen a sinen keresztül, ilyenkor a laptervezéskor, illetve megépítésekor tiszteletben kell tartani a különféle elektromos, mechanikai, illesztési előírásokat.

A mikroszámítógép buszrendszere több tucat vezetékből áll, melyeken címek, adatok, parancsok jelenhetnek meg.

A kiegészítő lap tervezésekor a következő feladatokat kell megoldani:

- **eszközök kijelölése, ez a címvezetéken érkező cím azonosítása, felismerése,**

- **adatátviteli irány meghatározása (rákapcsolódás az adatvezetékre és lekapcsolódás az adatvezetékről) és**
- **az adatátvitelben szereplő eszközök szinkronizálása.**

IBM PC mikroszámítógépek esetén két sinrendszer létezik:
- **belső és**
- **külső.**

24.4.2. IBM PC belső sinredszer (busz)

Ez a sinrendszer az alaplap tulajdonságát határozza meg, melyet az elérni kívánt teljesítmény szab meg. Három féle kialakítása lehetséges:
- **nagyobb sebességeknél 3-sines rendszer használatos, ahol a címsin mellett külön adatsin van írásra és külön adatsin olvasásra, így, ha lehet azonos idejű adatolvasás és adatírás alakítható ki,**
- **2-sines rendszer, vagyis külön címsin és külön adatsin található a legtöbb gépnél és**
- **közös adat- és címsin olyan rendszereknél, ahol egyszerű célfeladatok létrehozása szükséges.**

A belső sinrendszerhez hardver eszközökkel nem férhetünk hozzá, ez adott az alaplapoknál.

24.4.3. IBM PC külső sinrendszere

A külső sinrendszerhez kapcsolhatunk lapokat különféle I/O feladatok ellátására. A külső sinrendszer az összekapcsolt területek alapján a következő lehet:
- **helyi sin (local bus), közvetlenül a processzorhoz kapcsolódó eszközök illesztésére szolgál, ilyen eszköz a társprocesszor, ahol igen lényeges a nagy adatátviteli sebesség,**
- **rendszersin (system bus), buszmeghajtó közbeiktatásával a processzort köti ossze a gép más részeivel, főleg az I/O eszközökkel és**
- **memóriasin (memory bus), amely általában a rendszersin része, de nagyobb rendszereknél önálló sin.**

Az IBM PC gépek fejlődése során több fajta szabványos sin alakult ki, melyek közül némely még ma is használatos, de előfordult olyan megoldás is, amely a gyakorlatban csak rövid ideig volt életképes.

A sinek vezérlési módjuk szerint lehetnek:
szinkron ütemezésűek, vagyis saját órajellel vezérlik a sinműveleteket és aszinkron ütemezésűek, nem rendelkeznek saját órajellel, az egymást követő elemi lépések hozzák létre a sinműveletet.

A legismertebb PC sinek a következők:
- **IBM PC, vagy XT sin,**
- **AT sin,**
- **VESA sin,**
- **PCI sin,**
- **PCMCIA sin.**

24.4.3.1. Az IBM PC, vagy XT sin

Ez az első kifejlesztett IBM PC sin, mely 8 bites. Időzítését tekintve szinkron megoldású, paritásvédelemmel ellátva. Mivel Intel 8086-os processzorhoz fejlesztették ki, ezért a 8 bites adatsin mellett 20 bites címsinnel rendelkeztek (1 MB memória). A sin 10 bites perifériacímzést engedélyez (1024 eszköz).
A csatlakozó 62 érintkezős, kétoldalas lap helyezhető bele. Összesen 8 csatlakozó található egy XT **alaplapon, az adatvonalak közvetlenül a processzorra kapcsolódnak, laphiba leállítja a működést. A csatlakozó felépítése a 24.5. ábrán látható, míg az egyes jelek jelentése a 24.1. táblázatban.**

24.5. ábra: az **XT** sin csatlakozója

24.1. táblázat: Az XT csatlakozó felépítése és jelei

jel neve	csat-la-kozó	jel neve	csat-la-kozó	jel neve	csat-la-kozó	jel neve	csat-la-kozó
GND	B1	-DACK1	B17	-IOCHCK	A1	A14	A17
RESDRV	B2	DRQ1	B18	D7	A2	A13	A18
+ 5 V	B3	- REFRESH	B19	D6	A3	A12	A19
IRQ2	B4	CLOCK	B20	D5	A4	A11	A20
- 5 V	B5	IRQ7	B21	D4	A5	A10	A21
DRQ2	B6	IRQ6	B22	D3	A6	A9	A22
- 12 V	B7	IRQ5	B23	D2	A7	A8	A23
-CRSEL	B8	IRQ4	B24	D1	A8	A7	A24
+ 12 V	B9	IRQ3	B25	D0	A9	A6	A25
GND	B10	-DACK2	B26	IOCHRDY	A10	A5	A26
-MEMW	B11	T/C	B27	AEN	A11	A4	A27
-MEMR	B12	ALE	B28	A19	A12	A3	A28
-IOW	B13	+ 5 V	B29	A18	A13	A2	A29
-IOR	B14	OSC	B30	A17	A14	A1	A30
-DACK3	B15	GND	B31	A16	A15	A0	A31
DRQ3	B16			A15	A16		

Az egyes jelek jelentése:

- **-IOCHCK – Input/Output Channel Check, be/kimeneti csatorna ellenőrzés, bemenet, 0 jel, a bővítőkártya hibája,**
- **D0-D7 – 8 bites adatsin, D0 LSB, D7 MSB bit, kétirányú,**
- **IOCHRDY – Input/Output Channel Ready, be/kimenet csatorna kész, bemenet, 0 jel, I/O ciklusban várakozás processzor részéről,**
- **AEN – Address Enable, cím engedélyezve, 0 processzor, 1 DMA vezérli a sint,**
- **A0-A19 – 20 címvezeték,**

- **RESDRV – Reset Drive, meghajtó reset, 1 esetén minden alapállapotba kerül,**
- **IRQ2-IRQ7 – Interrup Request, megszakítás kérés, bemenetek, 1 esetén megszakítás,**
- **DRQ1-DRQ3 – DMA Request, DMA kérés, 1 esetén DMA kérés,**
- **-MEMW – Memory write, memóriaírás, 0 esetén írás memóriába,**
- **-MEMR – Memory Write, memóriaolvasás, 0 esetén processzor, vagy DMA olvas memóriából ,**
- **-IOW – I/O Write, I/O írás, 0 esetén periféria írás,**
- **-IOR – I/O Read, I/O olvasás, 0 esetén periféri olvasás,**
- **-DACK1-DACK3 – DMA Acknowledge, DMA elfogadás, 0-ra vált elfogadás esetén,**
- **-REFRESH – memóriafrissítés, 0 esetén RAM frissítési igény,**
- **CLOCK- sinórajel,**
- **T/C – Termonal Count, átvitelvége jel, 1 jelenik meg DMA átvitel befejezése után,**
- **ALE – Address Latch Enable, címregiszter engedélyezés, 1 esetén a cím a sinen van,**
- **OSC – Oszcillátor frekvencia, ami 14,31618 MHz.**

24.4.3.2. AT, ISA sin

1984-ben megjelent Intel 80286 processzor már más típusú alaplapra került, ahol a sin is bővült. Ennek több oka volt, egyik az, hogy a memória 16 MB nagyságú lett, de a 16 bites adatsin is hozzájárult ennek az új busznak a kifejlesztéséhez. Itt jelenik meg az az XT sinnél nem létező lehetőség, hogy a sint külső egység vezérelje.

Annak érdekében, hogy a az XT sinhez kifejlesztett lapok itt is használhatók legyenek az AT g;pekben, a sint úgy alakították ki, hogy a 62 pólusú első rész megegyezik az XT sinnel, ehhez csatoltak hozzá még egy 36 pólusú csatlakozót, ennek az ISA (Industry Standard Application) busznak a képe a 24.6. ábrán

látható.

23.1. ábra: Az ISA **sin csatlakozójának rajza**

Ennél a csatlakozónál az XT –CRSEL jelének helyére egy –OWS jel került, míg a többi csatlakozópont a 36 pólusú csatlakozőra. A jelek és csatlakozópontok kapcsolata 24.2. táblázatban található.

24.2. táblázat: Az ISA csatlakozó felépítése és jelei

jel neve	csat-la-kozó	jel neve	csat-la-kozó	jel neve	csat-la-kozó	jel neve	csat-la-kozó

- MEMCS16	D1	-DACK5	D10	-SBHE	C1	-MEMW	C10
-IOCS16	D2	DRQ5	D11	LA23	C2	SD8	C11
IRQ10	D3	-DACK6	D12	LA22	C3	SD9	C12
IRQ11	D4	DRQ6	D13	LA21	C4	SD10	C13
IRQ12	D5	-DACK7	D14	LA20	C5	SD11	C14
IRQ15	D6	DRQ7	D15	LA19	C6	SD12	C15
IRQ14	D7	+ 5 V	D16	LA18	C7	SD13	C16
-DACK0	D8	-MASTER	D17	LA17	C8	SD14	C17
DRQ0	D9	GND	D18	-MEMR	C9	SD15	C18

Az egyes jelek jelentése:

- **-SBHE – System Bus High Enable, ha 0, akkor a felső adatvezetéken is vannak jelek,**
- **LA17-LA23 – címvezetékek, az A17-A19 és LA17-LA19 jelek egyméshoz fél órajellel vannak eltolva,**
- **-MEMR – Memory Read, memóriaolvasás, 0 ha a processzor, vagy a DMA olvas a memóriából,**
- **-MEMW – Memory Write, memóriaírás, 0 írás memóriába,**
- **SD8-SD15 – adatvezetéke, a 16 bites adatátvitel felső bájtja,**
- **-MEMCS16 – Memory Chip Select 16, 16 bites memória áramkör kiválasztás, a jelet a kártya küldi ha 16 bites adatátvitelt kezdeményez,**
- **-IOCS16 – Input/Output Chip Select 16, 16 bites periféria kiválasztás,**
- **IRQ10-12,14,15 – Interrupt Request, megszakításkérés, kártya igényli, kisebb szám, nagyobb prioritás,**
- **-DACK0 – Dma Acknowledge, DMA nyugtázás,**
- **-DACK5 – Dma Acknowledge, DMA nyugtázás,**
- **-DACK7 – Dma Acknowledge, DMA nyugtázás,**
- **DRQ0 – DMA Request, DMA átvitel kérés, periféria kezdeményezi,**
- **DRQ5 – DMA Request, DMA átvitel kérés, periféria kezdeményezi,**
- **DRQ6 – DMA Request, DMA átvitel kérés, periféria kezdeményezi,**
- **-MASTER – másik processzor 0 jellel jelzi, hogy átvette a sinvezérlés jogát.**

Az Intel processzorok jellegzetessége, hogy a perifériacímzés elválasztható a memóriacímzéstől. Erre a célra a címvezetékből 10 bitet használnak fel, ami 1024 I/O cím előállítását teszi lehetővé. A 24.3. táblázatban a már előre lefoglat és kiosztott perifériacímeket találjuk.

24.3. táblázat: A PC perifériák címkiosztása

I/O cím [hexa]	eszköz
000-01F	DMA vezérlő
020-03F	megszakítás vezérlő
040-05F	számláló
070-07F	8255 PIO, AT billentyűvezérlő
080-09F	DMA lapregiszter
0A0-0BF	2. megszakítás vezérlő
0C0-0DF	2. DMA vezérlő
0F0-0FF	matematikai társprocesszor

100-1EF	**szabadon felhasználható**
1F0-1F8	**merevlemez vezérlő**
1F9-1FF	**szabadon felhasználható**
200-20F	**game (játék) port**
210-217	**bővítő egység**
220-24F	**fenntartott**
250-277	**szabadon felhasználható**
278-27F	**2. nyomtatóport**
280-2EF	**szabadon felhasználható**
2F8-2FF	**2. soros port**
300-31F	**prototipus kártya**
320-32F	**merevlemez vezérlő**
330-377	**szabadon felhasználható**
378-37F	**1. nyomtatóport**
380-38F	**SDLC port**
390-39F	**szabadon felhasználható**
3A0-3AF	**első BISYNC port**
1F9-1FF	**monokróm gr. kártya, párhuzamos port, vagy Hercules gr. kártya**
3C0-3CF	**EGA grafikus kártya**
3D0-3DF	**VGA grafikus kártya**
3E0-3EF	**szabadon felhasználható**
3F0-3F7	**hajlékony lemez vezérlő**
3F8-3FF	**első soros port**
400-FFF	**nincs meghatározva**

24.4.3.3. A VESA sin

A 32 bites processzorok megjelenése új követelményeket támasztott, így ezek az igények már nem kielégíthetők sem az XT, sem az AT buszokkal. A 33 MHz-es frekvenciával működő processzort feltételező VESA sinből néhány használható, ugyanis a felhasznált kártyák számával a sinsebesség csökken.

A tervezők igyekeztek a régi ISA kártyákat is felhasználni, így egyszerűen az ISA cstlakozóval sorban, attól 5 mm távolságban egy 2 x 58 érintkezővel ellátott, sűrűbb csatlakozót helyeztek.

A Pentium processzorok megjelenésével ennek a busznak megszűnt a jelentősége.

24.4.3.4. A PCI sin

Tervezésekor már figyelembe vették a 64 bites adatátvitel lehetőségét. Kezdetben 33 MHz-es órajellel volt képes dolgozni, de ezt a frekvenciát növelik az újabb alaplapoknál. A + 5 V feszültség mellett már a + 3.3 V is megtalálható a buszon.

Az összesen 10 eszköz csatlakoztatási helyből 5 az alaplapon levő I/O eszközhöz van rendelve, az 5 szabad pedig kártyák illesztésére szolgál.

A PCI sin nem kompatibilis az előző sinekkel, de azokkal kombinálható az alaplapon, így többfajta kártya is beépíthető a számítógépbe.

Az ún. Plug and Play technológiát támogatja a sin, vagyis ha az operációs rendszer erre alkalmas, akkor az új eszközöket autómatikusan felismeri a rendszer és már használhatók is.

24.4.3.5. A PCMCIA sin

A lap top, vagy hordozható PC számítógépekhez kifejlesztett PCMCIA (Personal Computer Memory Card International Association) sin kisméretű, alkalmas statikus és dinamikus RAM , Flash Eprom és periféri kártyák kezelésére. A perifériakártyák háromféle méretben készülnek.

Ellenőrző kérdések

1. Sorolja fel az IBM PC személyi számítógépek alaklmazási területeit .
2. Hogyan nevezték az első IBM gyártmányú személyi számítógépet és milyen processzort építettek az alaplapra ?
3. Mik az IBM PC személyi számítógépek legfontosabb jellemzői ?
4. Rajzolja le az Intel 8086-os processzor belső felépítését .
5. Rajzolja le az Intel 8086 processzor belső regisztereit és ismertesse szerepüket .
6. Ismertesse az Intel 8086-os processzor jelzőbitjeit (flag) .
7. Mi a BIU (busz illesztő egység), mi a szerepe ?
8. Mi az EU (végrehajtó egység és mi a szerepe ?
9. Mi az átlapolás, vagy pipeline technológia ?
10. Ismertesse az Intel 8086 címzési technikáját .
11. Ismertesse a fizikai cím kiszámítását a szegmens- és effektív cím segítségével .
12. Mekkora memóriát tud címezni az Intel 8086 ?
13. Ismertesse a programcímzési módokat Intel 8086-nál .
14. Ismertesse az adatcímzési módokat Intel 8086-nál .
15. Sorolja fel és ismertesse az Intel processzorcsalád tagjait és legfontosabb tulajdonságait.
16. Ismertesse az IBM PC sinrendszerét .
17. Mi a sinrendszer feladata ?
18. Hányféle sint különböztetünk meg IBM PC számítógépeknél ?
19. Ismertesse hányféle belső sinrendszert használnak IBM PC számítógépeknél .
20. Az IBM PC külső sinrendszere hány részből áll ?
21. A sin vezérlési módja szerint hány féle lehet ?
22. Ismertesse az XT sin legfontosabb tulajdonságait, felépítését .
23. Ismertesse az AT (ISA) sin legfontosabb tulajdonságait, felépítését .
24. Ismertesse a VESA sin legfontosabb tulajdonságait, felépítését .
25. Ismertesse a PCI sin legfontosabb tulajdonságait, felépítését .
26. Ismertesse a PCMCIA sin legfontosabb tulajdonságait, felépítését .
27. Ismertesse az IBM PC perifériáihoz rendelt címek kiosztásást .

IRODALOM

1. Ajtonyi István: Digitális rendszerek. Miskolci Egyetemi Kiadó, 1998. ISBN 963 661 399 5.

2. Budai Attila: Mikroszámítógép-rendszerek. LSI Oktatóközpont, Budapest, 1999. ISBN 963 577 278 5.
3. Cserny László: Mikroszámítógépek. LSI Oktatóközpont, Budapest, 2. javított kiadás, 1996. ISBN 963 577 188 6.
4. Kovács Magda: Egyszerűen a mikroszámítógépről. LSI Oktatóközpont, Budapest. ISBN 963 577 072 3.
5. Kovács Magda: Első lépés a mikroszámítógépek világába. LSI Oktatóközpont, Budapest. ISBN 963 577 179 7.
6. Madarász László: A PIC 16C mikrovezérlők. Kecskemét 1996.
7. Madarász László: Az Intel 8051 mikrovezérlő család. Rádiótechnika évkönyve, Budapest, 1992.
8. Markó Imre: PC-k konfigurálása és installálása. . LSI Oktatóközpont, Budapest. ISBN 963 577 245 9.
9. Marschik Iván: Mikroprocesszorok, mikrogépek. Számok, Budapest, 1979.
10. Rector, Russel – Alexy, George: Das 8086 / 8088 Buch. Programmieren in Assembler und Systemachitektur. te-wi.
11. Ribarić, Slobodan: Arhitektura Mikroprocesora. Treće Izdanje. Tehnička Knjiga, Zagreb, 1988.
12. Tooley, Mike: Practical Digital Electronics Handbook. PC publishing, Kent, 1988. ISBN 963 10 9ö69 0.
13. Wolfe, W. Gordon: Computer Peripherials That You Can Build. Tab Books, 1982.
14. : Cross-32 Meta-Assembler, User's Guide, Version 1.5. Canada, Saint John, 1990.
15. --: Mikroprozessor-Lehrgang. Regelungstechnische Praxis, 1978.
16. www.kando.hu/~konya