Dr. Benesóczky Zoltán

INFORMATIKA ALAPJAI I.

(HARDVER)

összefoglaló jegyzet

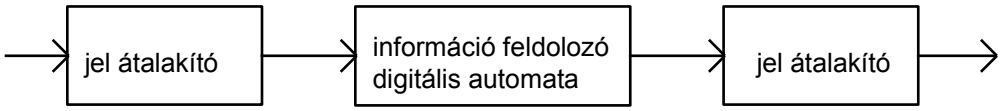
2001

Bevezetés

Ez az jegyzet elsősorban a környezetvédelmi mérnökök számára tartott informatika alapjai tárgy során megtartott, a hardeverről szóló előadásokat tartalmazza. Célja a számítógép működésének és az ott használt elveknek és fogalmaknak a megismertetése a számítógépet használók számára. Azonban azok számára is hasznos lehet, akik később részletesen, tervezői szinten is meg fognak ismerkedni a hardverrel. A jegyzetben lehetnek hibák A jegyzet olvasása során felmerült problémákat, javaslatokat a benes@mit.bme.hu címre küldött leveleben lehet felvetni.

1. Kódoláselméleti alapfogalmak

Információ feldolgozó gép:



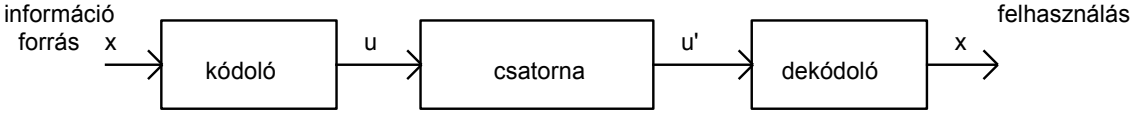
Az információ feldolgozó gép kódokkal dolgozik

analóg kód: amplitúdóban és időben folytonos (véges idő alatt végtelen információt hordoz, de a zajok miatt csak véges használható fel)

digitális kód: amplitúdóban és időben diszkréten értelmezett (véges idő alatt véges információt hordoz)

-analóg jelből digitális kódot annak diszkrét időpontokban való mintavételezésével, s a mintavett jel amplitúdójának kvantálásával kapunk

A kódokat és kódolást az információ átvitel oldaláról közelítjük meg.



információ forrás: pl: magyar szöveg
kódolás: pl: a magyar szöveg betűnkénti kódolása bináris számokká
csatorna: pl: két állapotú, fizikailag két feszültségszint reprezentálja
dekódolás: pl: a bináris számok magyar karakterekké alakítása

A csatorna lehet több állapotú is de itt csak a két állapotúval foglalkozunk

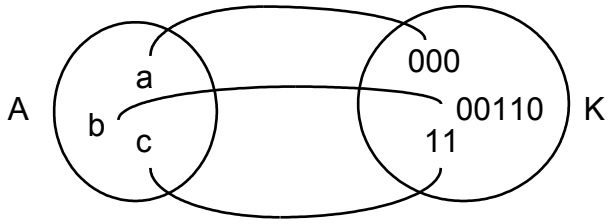
$A = \{a_1, a_2, \ldots a_n\}$ forrás ABC

$K = \{\alpha_1, \alpha_2, \ldots \alpha_n\}$ a kód ABC betűiből képzett véges hosszúságú sorozatok halmaza

Bináris kód esetén a kód ABC 2 elemű: {0, 1}

<u>Betű szerinti kódolás:</u>

Az *A* halmaz minden karakterének megfeleltetjük K egy-egy elemét ($a_1 \to \alpha_1,\ a_2 \to \alpha_2, \ldots$)



A K halmazt *kódnak* vagy nevezzük.

A K halmaz elemei a *kódszavak*.

Kódok osztályozása:

karakterkészlet alapján:	- bináris,
	- nem bináris
a kódszavak hosszúsága alapján:	- fix hosszóságú,
	- változó hosszúságú
alkalmazási cél alapján:	- aritmetikai,
	- pozíció,
	- hibajavító stb...

Mekkora fix hosszúságú kód szükséges, ha a kódolandó halmaz számossága N=9, a kód ABC betűinek száma k=3? $k^x \geq N,\ x = \lceil \log_k N \rceil$
x=2

1.1. Változó hosszúságú kódolás

A változó hosszúságú kódolás esetén a kódszavak hossza különböző lehet.
Megfejthetőség
Egy kód megfejhető, ha a kódszavaiból előállított tetszőleges üzenet egyértelműen felbontható a kód kódszavaira.
A következő kód nem megfejthető {a: 00, b:01, c:11, d:0001}, ugyanis az abd kódolásával adódó 00010001 üzenet abd,dab, abab és dd üzenetként is értelmezhető.

Ha a kódszavak hossza egyforma, akkor a kódolt üzenet biztosan megfejthető.

Prefix tulajdonság*:* egyik kódszó sem folytatása egy másiknak.
pl: a {01, 001, 100, 1100} kód prefix

A prefix tulajdonság a megfejthetőség elégséges, de nem szükséges feltétele.

A {10, 100, 1000, 10000} kód nem prefix, de megfejhető.

Ha *az információs csatorna zajmentes*, akkor a minél gyorsabb átvitel vagyis rövid üzenet a cél.

Kód költsége

Egy információ átvitelének (tárolásának) költsége annál nagyobb, minél hosszabb. Ezért zajmentes esetben a kódolás elsődleges célja, az átlagos kódhossz csökkentése.

Ha adott egy-egy karakter (a_i) előfordulási valószínűsége, p_i, az a_i hez tartozó kódszó hossza l_i és $\sum p_i = 1$ (teljes eseményrendszer), akkor az M számú karakterből álló üzenet kódjának átlagos hossza a következő gondolatmenettel számítható:

- az M karaktert kódoló üzenetben átlagosan $n_{a_i} = Mp_i$ szer fordul elő az a_i karakter, mert $p_i \cong \frac{n_{a_i}}{M}$, $M \to \infty$ *eseté n*

- a üzenetben előforduló a_i karakterek kódjának együttes hossza: $Mp_i l_i$

- a teljes kódolt üzenet hossza: $M\sum_i p_i l_i$

- a *kód költsége:* $\lim_{M\to\infty} \frac{\textit{Kódolt üzenet hossza}}{M} = \sum_i p_i l_i$ létezik alsó korlátja

Általános elv, hogy a ritkábban előforduló eseményt (karaktert) kódoljuk hosszabb kóddal.

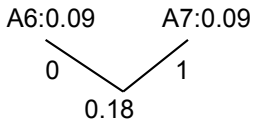
<u>Huffmann kód</u>

A Huffmann kód változó hosszúságú optimális költségű prefix kód. A kódolás algoritmusát a következő példán mutatjuk be:

Adott az alábbi kód és valószínűségek:

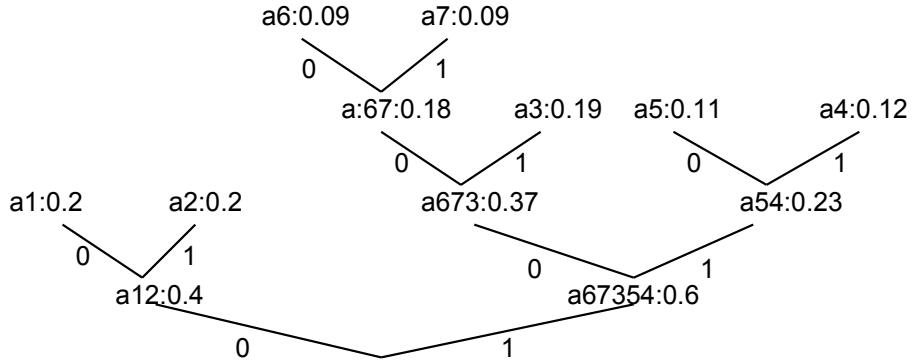
{a1:0.2 a2:0.2, a3:0.19, a4:0.12, a5:0.11,a6:0.09,a7:0.09} $\sum_i p_i = 1$ teljesül

a. Vegyük a két legkisebb valószínűségű eseményt és különböztessük meg őket egy bittel, s utána vonjuk őket össze egyetlen olyan eseménnyé, melyek valószínűsége a két esemény valószínűségének összege.



Ezután az összevont eseménnyel helyettesítve azokat, amelyek összevonásából keletkezett, folytassuk az előző pont szerint, amíg lehetséges.

Az összes lépés elvégzése után a következő adódik:



Az egyes események kódolása a kiadódó fa gyökerétől kiindulva egy-egy levélig (kódolandó karakterek) található 0-kat ill. 1-eket egymásután írva adódik:

a1:00, a2:01, a3:101, a4:111, a5:110, a6:1000, a7:1001

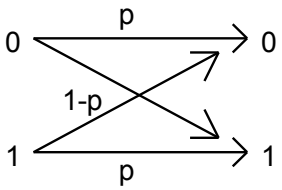
Felhasználási példák

A változó hosszúságú kódolás felhasználására gyakorlati példa, a file tömörítés. A fileban levő karakterekről statisztikát készítve megállapítható a karakterek előfordulási valószínűsége. Ez alapján pedig elvégezhető a tömörítés. Természetesen a tömörített fileban el kell tárolni az egyes karakterek kódját is.

Fekete fehér képek tömörítésére (FAX) futamhossz kódolást alkalmaznak. Mivel a fekete fehér képen sok egymást követő 1-es ill. 0-van, egy-egy képsor kódjában azt adják meg, hogy éppen 1-es vagy 0 következik, s hogy hány darab van belőle.

1.2. Fix hosszúságú kódolás

Zajos csatorna esetén a cél, a mennél kisebb hibával történő átvitel.
A bináris szimmetrikus zajos csatorna modellje:



- a helyes átvitel valószínűsége p (p>0.5, ha ez nem teljesül akkor a csatorna invertál...)
- a hibás átvitel valószínűsége 1-p

Ebben a hibamodellben un. *átállítódásos hibák* szerepelnek, vagyis hiba esetén az információs bit negáltját érzékeli a vevő logika.

A lehetőleg hibátlan átvitel érdekében
- fix hosszúságú kódolást alkalmazunk
- nem használjuk ki a kódtér összes elemét (az adott bit hosszúsággal lehetséges összes kódot)

Kódszó súlya W(a): az 1-es bitek száma
W(10110)=3

Két kódszó Hamming távolsága H(a,b): az eltérő bitek száma
Pl: H(10110,01011)=4
H(a,b)=W(a ⊕b)

```
  10110
⊕01011
------
  11101        W(11101)=4
```

Nem csak bináris kód esetén van értelme:
k1= BABUCI
k2= PAPUCI
eltérés: 101000 W(101000)=2, H(k1,k2)=2

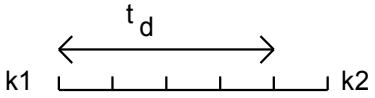
Kód Hamming távolsága: a minimális Hamming távolság a kódszavak között
Hiba detektálása: a hibás kódszó nem eleme a kódtérnek
Hiba javítása: a kódtér legközelebbi elemére javítunk.
Példák:
1 hibát detektál a kétszer ismétlő kód (pl: 0000, 0101, 1010, 1111). Hiba van, ha a kód két fele nem megegyező.

1 hibát javít a 3-szor ismétlő kód (pl: 000000,010101,101010,111111). Amelyik két kódrész egyező, az a helyes.

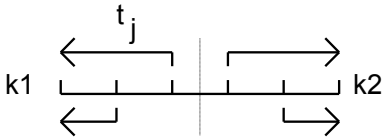
1 ill. páratlan hibát detektál a paritás kód. (Az utolsó bit megegyezéstől függően párosra vagy páratlanra egészíti ki a többi bitet.)
(pl: páratlan paritású paritás kód: (pl: 001, 010, 100, 111)

A Hamming távolság és a *hiba detektálás* ill. *hiba javítás* közötti összefüggés

a. t_d hiba detektálható, ha $H \geq t_d + 1$

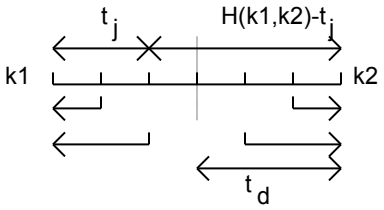


b. t_j hiba javítható, ha $H \geq 2t_j + 1$



c. t_j hiba javítható és t_d hiba detektálható ha $H \geq t_d + t_j + 1$

A $t_d < H - t_j$ egyenlőtlenségnek teljesülni kell, különben a detektálandó hiba az egyik kódtól t_j, vagy kisebb távolságra esik, tehát nem lesz eldönthető, hogy javítani, vagy detektálni kell. Tehát $t_d < H - t_j \Rightarrow H \geq t_d + t_j + 1$

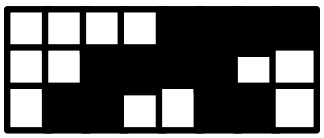


Az átállítódásos hibák mellet léteznek az eltörlődéses hibák, ahol a vevő azt érzékeli, hogy az átvitt bit se nem 0, se nem 1 szintű, hanem hibás. Ebben az esetben tehát a hiba helyét ismerjük, vagyis ideális esetben minden eltörlődéses hiba érzékelhető, jelezhető. Mivel n eltörlődésés hibát feltételezve egy m hosszúságú kódszóban a konkrét esetben tudjuk, hogy mely helyeken keletkezett a hiba, vagyis ismerjük a kódszó m-n helyes bitjeinek helyét és értékét is. Az ettől n+1 Hamming távolságra levő kódszavak biztosan különböznek az aktuálisan helyes bitek helyén 1 bitben, így n+1 Hamming távoságú kód esetén n eltörlődéses hiba javítható, arra a kódszóra kell javítani, amelynek megfelelő bitjei megegyeznek a hibás kódszó hibátlan bitjeivel.

Néhány fontos kód

<u>Gray kód</u>

Pozició (helyzet) kódolására használják (pl: a forgó szinpad éppen hogy áll). Az egymásután következő pozíciók kódja egy Hamming távolságú. Igy a pozíció érzékelők (pl. foto érzékelők) a pozíció határ átmenetnél nem adnak hibásan "távoli" pozíciót jelentő kódot, ahogy az több Haming távolságú kód estén előfordulhatna.Az alábbi ábra egy vízszintes szakaszt 8 részre osztva kódol, Gray kóddal.



A kombinációs hálózatok grafikus egyszerűsítésénél használt Karnaugh tábla peremezése is Gray kódú.
Tükrözéses módszerrel lehet kisebb bitszámú Gray kódból nagyobbat készíteni.
Induljunk ki a 2 bites Gray kódból (most a kisebb helyfoglalás miatt az egymás alatti számok adják a kódot, 00,01,11,10):
0011
0110
Irjunk hozzá egy újabb bitet úgy, hogy az a régi kód előtt legyen 1, s írjunk hozzá még ugyanennyi 0-át. Ezután a hiányzó biteket a régi kód tükrözésével állítjuk elő:
1111 0000
0011 1100
0110 0110

Aritmetikai kódok

BCD kód: a decimális 0...9 számokhoz 4 bites bináris számokat (0000...1001) rendel.
Pl: 1997 BCD kódban: 0001 1001 1001 0111
Számábrázolások

	előjeles abszolut értékes	egyes komplemens	kettes komplemens	offszet
3	011	011	011	111
2	010	010	010	010
1	001	001	001	110
0	000, 100	000, 111	000	100
-1	101	110	111	011
-2	110	101	110	010
-3	111	100	101	010
-4			100	000

Áttérés az egyes számábrázolások között
Egyes komplemens-> kettes komplemens: Ha MSB=1, akkor X=X+1.
Kettes komplemens -> egyes komplemens: H MSB=1, akkor X=X-1

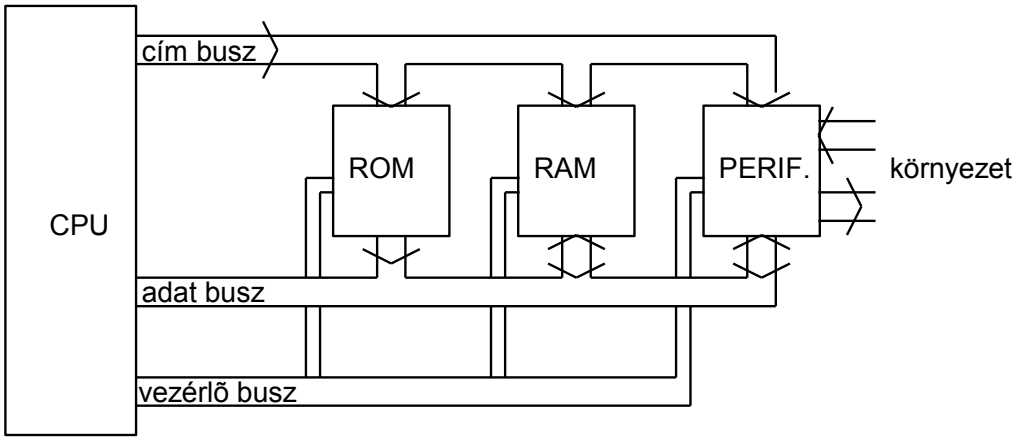
Előjeles absz. értékes -> egyes komplemens: Ha MSB=1, akkor MSB kivételével invertálni.
Egyes komplemens -> előjeles absz. értékes: Ha MSB=1, akkor MSB kivételével invertálni.
Előjeles absz. értékes -> kettes komplemens: Ha MSB=1, akkor MSB kivételével invertálni, majd 1-et hozzáadni.
Kettes komplemens -> előjeles absz. értékes: Ha MSB=1, 1-et kivonni, majd MSB kivételével invertálni.
Offszet -> kettes komplemens: MSB-t invertálni.

2. Boole algebra, kombinációs és sorrendi hálózatok

- Boole algebra axiómái és fontos tulajdonságai (kommutativitás, kétféle disztributivitás, konstansokal való műveletek, inverz elem definíciója, elnyelési tulajdonság, De'Morgan azonosságok.)
- Logikai függvények megadása (normál alakok, általános algebrai alak, igazságtábla)
- Logikai kapuk (AND, NAND, OR, NOR, EXOR, EKVIVALENCIA) és viselkedésük, kombinációs hálózat fogalma (a bemeneti logikai változók minden kombinációijához egy logikai értéket rendel, a kimenete csak az aktuális bementi kombinációtól függ)
- Sorrendi hálózat fogalma (Emlékezettel rendelkezik, melyeket állapotokkal reprezentálunk. Logikai értékű kimenetet állít elő, mely a hálózatot ért változások hatására előállt állapottól és a bemenettől függ. Működését állapotgráffal, állapottáblával írjuk le.)

3. Mikroprocesszor, mikroprocesszoros rendszer

Mikroprocesszoros rendszer blokkvázlata



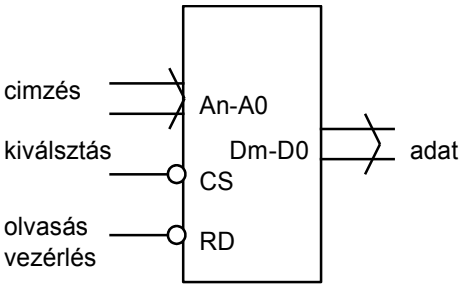
A részegységek és feladataik

ROM (Read Only Memory)
Csak olvasható memória, kikpcsolás (tápfeszültség megszüntetése) után sem felejt. A RAM-nál (általában) lassabb. Jellemző hozzáférési idő: 70-300nsec

(1 nanosecundum = 10^{-9} sec).

Jellemzők

- Szószélesség: egyszerre kilvasható adatbitek száma (8, 16)
- Tároló kapacitás: hány bit, byte (8 bit), szó (általában 16 bit)
  Információt képes tárolni. 1kbyte=2^10byte=1024byte, 1Mbyet=2^20byte=1024kByte



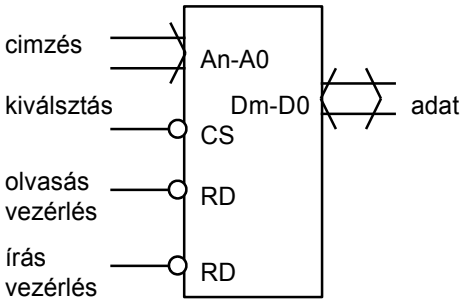
Feladata a mikroproceszoros rendszerben a bekapcsoláskor elinduló program tárolása.

Legfontosabb tipusai:
PROM: egyszer írható ROM
EPROM: UV fénnyel törölhető, újra programozható ROM
EEPROM, FLASH: elektromosan törölhető és újraírható ROM

Bizonyos tipusaikat a mikroprocesszoros rendszer működése közben is lehet írni. A futás közben keletkező és kikapcsolás után is megőrzendő információk tárolására is használhatók.

Nagy méretű FLASH memóriákból szilárd test háttértárakat is készítenek, melyek a kisebb méretű winchesterekett (mozgó alkatrészeket tartalmazó háttértárolókat) képesek helyettesíteni.

RAM (Random Access Memory)
Írható és olvasható memória. Kisebb tárkapacitású méretekben nagyon gyorsak is léteznek. Jelemző hozzáférési idő: 10-100ns



Feladata a futás közben változó adatok (változók) tárolása, ill. nagyobb rendszerekben a háttértárolóból betöltött program tárolása.

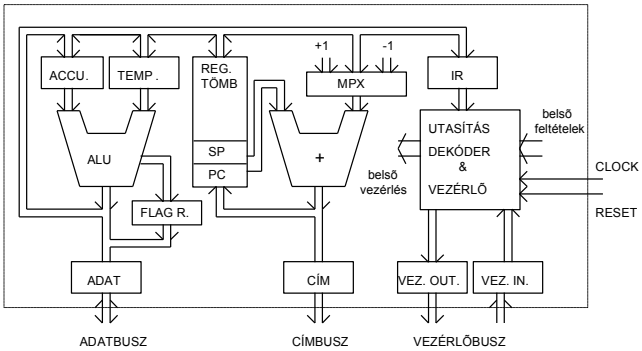
Gyorsabb mint a ROM, ezért bizonyos esetekben a ROM-ban tárolt programot is RAM-ba másolják és onnan futtatják.
Tárolócella kialakítása szempontjából két csoportra oszthatók:

- Statikus RAM (SRAM): gyors, de a felületegységen kialakítható tárolók száma jelentősen kisebb, mint a dinamikus RAM-nál, s az egy tárolócellára jutó ár magas.
- A dinamikus RAM (DRAM): lassabb mint a SRAM, viszont jelentősen nagyobb tárolók építhetők fel, sokkal olcsóbban, mint az SRAM-okból. Járulékos, ún. frissítő áramkört is igényelnek.

Perifériák
A perifériák általában a mikroprocesszoros rendszer és a környezet közötti kapcsolatot biztosítják (pld. soros port, párhuzamos port, kijelző, billentyűzet). De pl. a timer (időmérő) egységet is a perifériákhoz soroljuk. A perifériák és a mikroprocesszoros busz közötti kapcsolatot (illesztést) a periféria interfészek biztosítják.

Mikroprocesszor (**CPU**, Central Processing Unit)
A mikroprocesszor főfeldata a memóriában tárolt utasítások (program végrehajtása).

Egyszerű mikroprocesszor belső felépítése



A mikroprocesszor működését a hozzá kapcsolódó program memóriában tárolt *utasítások* vezérelik.

Egy általános mikroprocesszor legfontosabb képességei:

- Utasítások és adatok megcímzése,
- A beolvasott utasítás értelmezése (utasítás dekódolás) és végrehajtása,
- Aritmetikai, logikai műveletek elvégzése (ALU, Arithmetical Logical Unit),
- Adatok mozgatása regiszterek között, ill. a regiszterek és a memória között,
- A RAM kijelölt részét *verem memória*ként (*stack*) tudja kezelni. Itt az utolsónak betett adathoz lehet először hozzáférni. A verem kezeléséhez az ún. *verem mutató*t (*stack pointer*) használja.
- Feltételes elágazások a program végrehajtás során (conditional jump and call),
- Külső események hatására az aktuális program félbehagyása és az eseményt kiszolgáló program végrehajtása, majd a félbeszakított program folytatása (interrupt),
- A busz vezérlésének felfüggesztése és a vezérlési jogának átadása (arbitráció),
- Rugalmas alkalmazkodás a perifériák és memóriák sebességéhez.

A mikroprocesszor fenti képességei alkalmassá teszik a mikroprocesszoros rendszert bonyolult számítási ill. vezérlési feladatok megoldására.
A mikroprocesszor fontos jellemzője az adatbusz szélessége (8-16-32-64 bit teljesítménytől függően), a fizikai címezhető tartomány (64 kbyte-64 Gbyte...), és az órajel frekvenciája. Az egyszerűbb mikroprocesszorok 1-20MHz-es (1MHz=10^6 Hz) órajellel működnek, de pl. a Pentium 4 processzor 1.5 GHz-el (1 GHz= 10^9 Hz)!

3.1 Mikroprocesszoros busz, kommunikáció az egységek között

A mikroprocesszor a hozzá kapcsolódó egységekkel az ún. buszon keresztül kommunikál. A **mikroprocesszoros busz** tulajdonképpen egy digitális információ továbbítására alkalmas kommunikációs csatorna, mely több, a funkciójukat tekintve egy halmazba sorolható jelcsoportból áll.

A mikroprocesszoros buszra kapcsolódó egységeket két csoportra osztjuk.

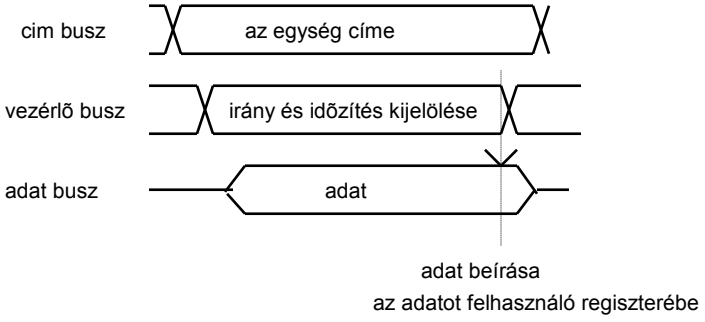
- **Master**: képes a buszon lezajlódó adatátvitel vezérlésére. Ezt a szerepet többnyire a CPU tölti be, ezért a továbbiakban master helyett CPU-t írunk..
- **Slave**: ha a master kijelöli, képes résztvenni az adatátvitelben (mint az adat forrása, vagy mint az adatátvitel célja). Slave egység a memória és a perifériák (de bizonyos perifériák időnként masterek is lehetnek).

A mikroprocesszoros busz részei:

- **Címbusz:** memória, vagy periféria kijelölésére. Jellemzője a címbitek száma. 10 bittel 1 kbyte, 20 bittel 1Mbyte, 30 bittel 1Gbyte címezhető. Egyszerű mikroprocesszoroknál 16 címbit szokásos, amivel 64kbyte címezhető, de pl. a 80386-nak 32 címbitje van, amellyel 4 Gbyte címezhető, a Pentium III-nak 36, amivel 64 Gbyte címezhető..
- **Adatbusz:** az adat továbbítására. Jellemzője az adatbitek száma. Egyszerű mikroproceszoroknál 8 bit, nagyobb teljesítményűeknél 16, 32, 64 bit a szokásos. A 80386-os processzornak 32 bites az adatbusza., a Pentium III-nak 64 bites.
- **Vezérlőbusz:** (az adatforgalom típusának, irányának, időzítésének vezérlésére és egyéb funkciókra).

A kommunikáció folyamata:

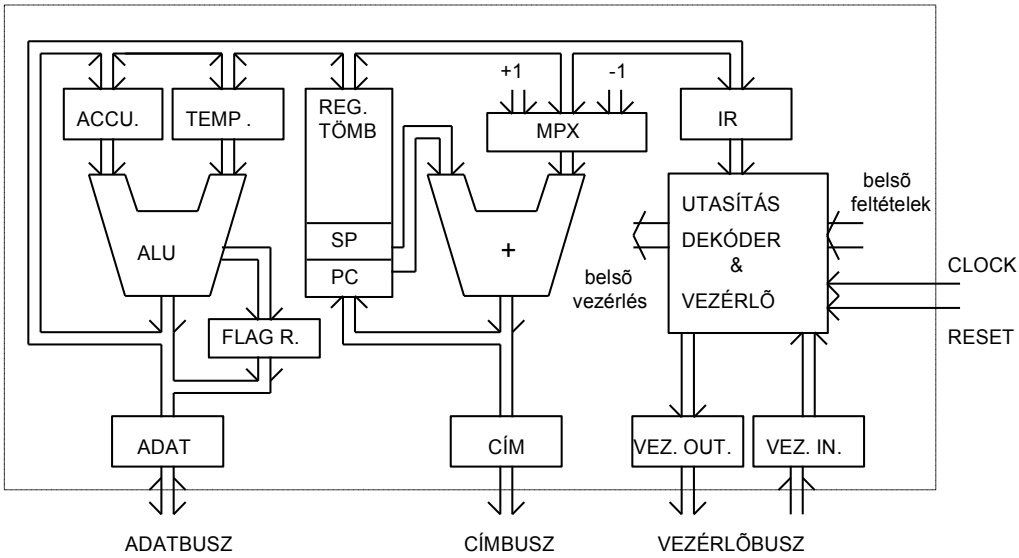
- A CPU kijelöli az egységet, amellyel kommunikálni fog (címzés, a címbuszon),
- A CPU kijelöli a kommunikáció irányát (olvasás vagy írás, a vezérlőbusz segítségével),
- Az adat forrása ráteszi az adatbuszra az adatot (a vezérlőbusz irányításával),
- Az adat felhasználója bemintavételezi az adatot a saját belső regiszterébe (a vezérlőbusz irányításával).



Egy-egy busz műveletetet busz ciklusnak nevezünk. Legfontosabb busz ciklusok:

- Utasítás olvasási ciklus (más néven FETCH ciklus)
- Adat olvasási ciklus (memória vagy periféria olvasás)
- Írási ciklus (memória vagy periféria írás)
- Interrupt (megszakítás) elfogadási ciklus

3.2. A mikroprocesszor belső felépítése



Egyszerű CPU belső blokkvázlata

A CPU architektúrájának legfontosabb elemei a következők:

Utasítás dekóder
Az utasítás regiszterben levő utasítás értelmezését végzi.

Vezérlő és időzítő egység
Ez az egység vezérli a CPU részegységeinek működését (az adatutakat, a belső regiszterek beírását, ALU-t stb.) a felismert utasításnak megfelelően, ill. reagál a külső jelekre

Aritmetikai logikai egység (ALU)
Feladata az aritmetikai (összeadás, kivonás stb.), logikai (bitenként ÉS, VAGY stb.) és egyéb (shiftelés, rotálás stb.) műveletek elvégzése a bemeneteire kerülő operandusokon. Az ALU a belső buszon keresztül a CPU belső regisztereivel és az adatbusszal van kapcsolatban. Egy művelet operandusai ill. az eredmény helye a belső regiszterek valamelyike, vagy a külső RAM lehet. Azt a belső regisztert, amelybe az eredmény kerül, akkumlátornak nevezik. Egyes CPU-k több akkumlátorral is rendelkeznek.

Regiszterek
A CPU belső regisztereit funkciójuk alapján két csoportra osztják, általános célú regiszterekre és speciális funkciójú regiszterekre.
Az **általános célú regiszterek**et belső adattárolásra használják. Ezek sokkal gyorsabban hozzáférhetők, mint a külső memória. Az általános célú regiszterek mérete többnyire a mikroprocesszor szószélességével vagy annak néhányszorosával egyezik meg.

Speciális funkciójú regiszterek a következők:

- **PC** (Program Counter) program számláló. Ez a regiszter tartalmazza a végrehajtandó utasítás címét. Normál utasítás végrehajtási sorrend esetén minden utasítás végrehajtása után inkrementálódik (1-el nő). A normál szekvenciát megváltoztató (ugró, szubrutin hívó stb.) utasítások esetén a megfelelő címmel feltöltődik. A PC mérete többnyire a CPU címbuszának szélességével megegyező.
- **IR** utasítás regiszter. Az utasítás elővételi (fetch) ciklusban beolvasott utasítást tárolja. Ennek tartalmát dolgozza fel az utasításdekóder.
- **SP** verem mutató (stack pointer). Ez a többnyire külső RAM memóriában kialakítható verem memória (szinonímái: LIFO, Last in First Out, stack, zsák memória) jellegű memóriakezelés megvalósításához szükséges *mutató* értékét tárolja (ez egy memória cím). Ez az ún. szubrutinok (egy utasítás (CALL) segítségével hívható program részek) megvalósításához és az interrupt működéséhez szükséges. Az SP mérete többnyire megegyezik a PC méretével.

- **Akkumlátor.** A műveletek egyik operandusa és eredménye többnyire ebben a regiszterben helyezkedik el. Mérete a mikroprocesszor belső adatbuszának szószélességével egyezik meg.

- **Jelzőbit (flag) regiszter.** Többnyire a műveletek eredményétől függő jelzőbitek (flagek) és egyéb, a CPU állapotára vonatkozó információk helyezkednek el ebben a regiszterben. A műveletek eredményeitől függő flageket az ALU állítja elő (összeadás és kivonás esetén az átvitel bit, az eredmény előjele stb.). A flagek értéke alapján feltételes elágazásokat és egyéb összetett műveleteket végez a processzor, ill. aritmetikai műveleteknél használja fel azokat.

  A műveletek által állított flagek processzor függők, de a következőkben felsoroltak szinte minden CPU-nál megvannak:

  Signum*:* Az eredmény előjele pozitív vagy negatív.

  Zero: Az eredmény 0.

  Carry*:* Aritmetikai *átvitel* és rotáló/shiftelő műveleteknél ez a bit is bekapcsolódhat az operandus adatbitjei közé.

  Auxiliary carry vagy **Half carry:**

  A 3. és 4. bit közötti átvitel (BCD műveletek esetén)

Parity: A művelet eredményében az 1-esek száma páros.

N: Az utolsó művelet kivonás volt.

Az alábbi regiszter funkciókat nem implementálják minden CPU-ban:

- **Bázis regiszterek.** Az adatok bázisrelatív címzésénél az adatterület bázis címét tartalmazza.
- **Index regiszterek.** Az adatok indexelt címzésű hozzáférését teszik lehetővé.

3.3. A program végrehajtása

A legegyszerűbb mikroszámítógépes rendszereken (célrendszerek) nem fut operációs rendszer. A programjuk általában fix (nem módosítják) és a végrehajtandó program kódját általában ROM tartalmazza.

Mi van a ROM-ban és hogyan kerül oda?
A programozó valamilyen nyelven (pl. assembly, C, Pascal), megírja a programot, egy másik számítógépen futó ***szövegszerkesztő*** program segítségével.
Az ASCII kódolású szövegfilet egy ún. ***fordító*** *program* feldolgozza és az assembly vagy magas szintű nyelven megadott utasításokat a fejlesztendő rendszer CPU-ja által értelmezhető utasításokká (egy utasítás CPU által értelmezhető kódja 1 vagy több byte-ból áll) ill. utasítássorozatokká alakítja. (A CPU viszonylag egyszerű utsításokat képes csak végrehajtani. Az assembly programozási nyelv áll ehhez legközelebb. Itt általában egy assembly utasítás egy CPU utasítássá alakul. Magasabb szintű nyelvek esetén általában egy utasítás több CPU utasítással valósítható meg.)
A fordító által generált kódhoz a ***linker*** program rendel konkrét memóriacímeket. Így előáll egy olyan byte sorozat amelyben mostmár minden byte-nak ismert a memória beli elhelyezkedése, s amelyet a CPU képes értelmezni.
Ezt a byte sorozatot egy speciális ***programozó készülékkel*** beprogramozzák a ROM-ba (EPROM, EEPROM, FLASH). A ROM-ot aztán belehellyezik a fejlesztett processzoros rendszerbe (target rendszerbe), általában egy IC foglalatba. Mikrokontrollerek (perifériákat, memóriát és mikroprocesszort egy chipben tartalmazó esközök) esetében sokszor a ROM is a chipbe van integrálva, így a programozás során magát a mikrokontrollert kell a programozóba helyezni a ROM (sokszor FLASH jellegű memória) programozásához, ill. sok esetben megoldott a NYÁK-ba (nyomtatott áramkör) ültetés utáni programozás is, mely utóbbi esetben szinkron vagy aszinkron soros módon tölthető fel a ROM tartalma.

Mit történk, amikor bekapcsoljuk a mikroprocesszoros rendszert?
A bekapcsoláskor a CPU alaphelyzetbe hozó jelet kap (RESET). Ekkor a CPU PC regisztere (program counter, utasítás számláló), amely mindig az aktuálisan végrehajtandó utasítás címét tartalmazza, az első utasítás címére mutat (ez általában a legkisebb című (0) memóriahely, de ez CPU tipustól függ).

Egy komplett utasítás végrehajtási (execute) ciklus a már említett gépi ciklusokból áll össze. Konkrét mintapéldaként megmutatjuk, hogy miképp történik a Z80 CPU (ez egy 8 bites CPU),

LD DE,(A150H)

utasításának a végrehajtása. Az utasítást assembly nyelven adtuk meg. A fordító ebből az utasításból az

ED 5B 50 A1

kódot generálja, amit a Z80 CPU képes dekódolni.

Az előbbi utasítás az A150 hexadecimálisan megadott címtől kezdve beolvas két byte-ot, az elsőt a Z80 CPU E regiszterébe, a másodikat pedig a D regiszterébe teszi.

Tegyük fel, hogy az utasítás kódja a hexadecimális 0000-es címen kezdődik. Az alábbiakban látható, hogy az utasítás kód byte-jai hogyan helyezkedik el a memóriában, és hogy azokhoz milyen gépi ciklusok kapcsolódnak.

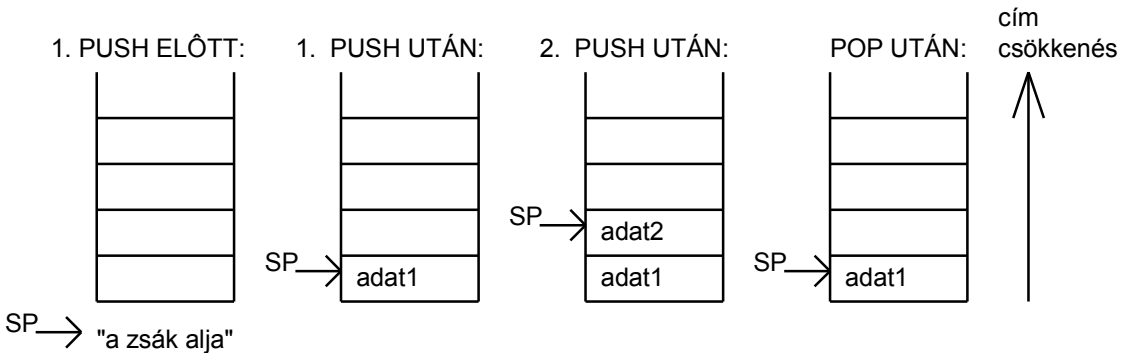
cím	adat		gépi ciklus típusa
0000	ED	az utasítás kód 1. byte-ja,	1. FETCH ciklus
0001	5B	az utasítás kód 2. byte-ja	2. FETCH ciklus
0002	50	a cím alsó fele	3. memória olvasási ciklus
0003	A1	a cím felső fele	4. memória olvasási ciklus,
			5. memória olvasási ciklus, E=(A150H),
			6. memória olvasási ciklus D=(A151H).
0004	A következő utasítás 1. byte-ja.		

Amikor a processzor ehhez az utasításhoz ér, a PC-ben 0000H van. Erről a címről beolvassa az utasítás kód első byte-ját az utasítás regiszterbe, ez FETCH ciklus, melynek a végén a PC-t inkrementálja (PC=PC+1). Az utasítást dekódolva megállapítja, hogy az utasításnak van még egy bye-ja, s ezt is beolvassa a következő FETCH ciklus alatt (PC=PC+1). A teljes utasítást dekódolva megállapítja, hogy be kell olvasnia még két byte-ot, melyeket majd memória címként kell értelmezni. Ez két memória olvasási ciklus, melynek végén a PC a következő utasítás címére mutat. Ezután az A150H címről beolvassa az adatot az E regiszterbe, majd az A151H címről a D regiszterbe, ez két memória olvasási ciklus. (A memória címen levő adatot zárójelbe tett memória címmel jelöltük.) Ezután jön a következő utasítás elővétele.
A processzor a vezérlés átadó (ugró, szubrutin hívó) utasítások és program megszakítás kivételével mindig szekvenciálisan (és növekvő memória címek felé) hajtja végre az utasításokat, vagyis az utasítás végén a PC a következő utasítás címére mutat.

3.4. A verem tár (stack) kezelése

A stack (LIFO, verem tár, zsák memória) a RAM memóriának egy speciálisan kezelt része, melynek a program végrehajtása során fontos szerepe van. A kezelésének lényege, hogy az adatokhoz a beírással ellentétes sorrendben lehet hozzáférni, vagyis ***mindig a***

legutoljára beírt adatot lehet legelőször kiolvasni. A beírás ill. kiolvasás címét a stack pointer (SP) tartja nyilván.
Két olyan utasítás van, amely csak a stackbe való írást (PUSH forrás) ill. az onnan történő olvasást (POP cél) szolgálja. Azonban egyéb utasítások is kezelik a stacket, ilyenek a szubrutin hívó (CALL) ill. abból visszatérő (RET) utasítások.
A stack pointer értéke a legtöbb CPU-nál egy-egy adat betétele során csökken (a tartomány a csökkenő memóriacímek felé hízik), de van ellentétes példa is. A POP és PUSH utasítások által egyszerre betett ill. kivett adat byte-ok száma CPU és utasítás függő. A következő ábra mutatja, hogy mi történik, ha az eredetileg üres stackbe két egymást követő PUSH utasítással adatokat teszünk, majd egy POP utasítással kiveszünk egy adatot.



A stack kezelése

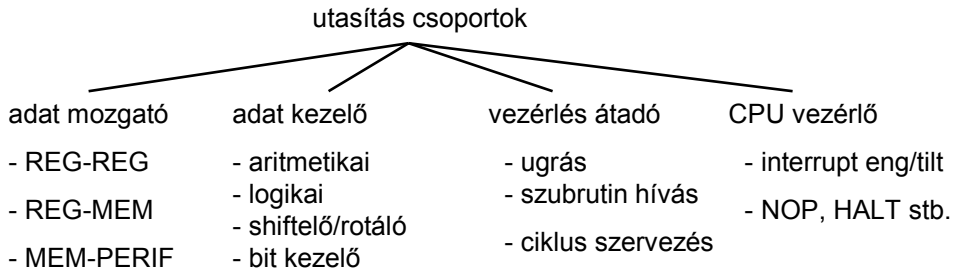
3.5. Utasításrendszer

Az utasításkészlet szempontjából két alapvető változatot szokás megkülönböztetni. A **CISC** (Complex Instruction Set Computer) architektúrájú CPU jellemzői a sok és bonyolult utasítás és a sokféle címzési mód. A **RISC** (Reduced Instruction Set Computer) CPU-t a kevés és egyszerű utasítás és kevés címzési mód jellemzi. Az alábbiakban szereplő utasítások jó része mindkét architectúrájú CPU-ban előfordulhat, kivétel a szorzás és osztás mely a RISC-ben nem szokásos.
A mikroprocesszorok működését az utasítások vezérlik. Az utasítások processzor specifikusak, minden processzorhoz tarozik egy utasítás halmaz, amelyet az végrehajtani képes. Ezt az utasítás halmazt nevezik a processzor *utasításkészlet*ének. Az utasításkészlet definíciójában leírják az utasítás kódját (egy utasítás kód több szóból állhat), az utasítás hatását (mit csinál), az utasítás mnemonikját (nevét). Az utasítás a műveleti kódból (op. code, ez kódolja, hogy mit kell csinálni) és az operandus elhelyezkedésére vonatkozó adatokból áll.

Az utasításkészletben szereplő utasításokat az elvégzett művelet típusa alapján szokás csoportosítani, az egyes processzorok programozási kézi könyvében is ily módon szerepelnek. Egy szokásos csoportosítás: **adat mozgató, adat kezelő, vezérlés átadó, CPU vezérlő**. Itt is ezt a csoportosítást követjük.

Az μP-k gyártói többnyire ún. mikroprocesszor családokat hoztak létre. A család újabb elemeinél (amíg az a fejlesztést nem gátolja túlzottan) arra törekednek, hogy az

újabb típusok szoftver (esetleg hardver) kompatibilisek legyenek a régebbiekkel. A szoftver kompatibilitás azt jelenti, hogy a régebbi CPU-k utasítás kódját felismeri és hibátlanul végrehajtja az újabb processzor. (Természetesen többnyire újabb utasításokkal is bővül az utasítás készlet.) Ilyenkor a régi készülékek szoftvere változtatás nélkül (vagy minimális változtatással) futtatható az újabb processzorokon, amivel sok fejlesztési munkát lehet megspórolni.



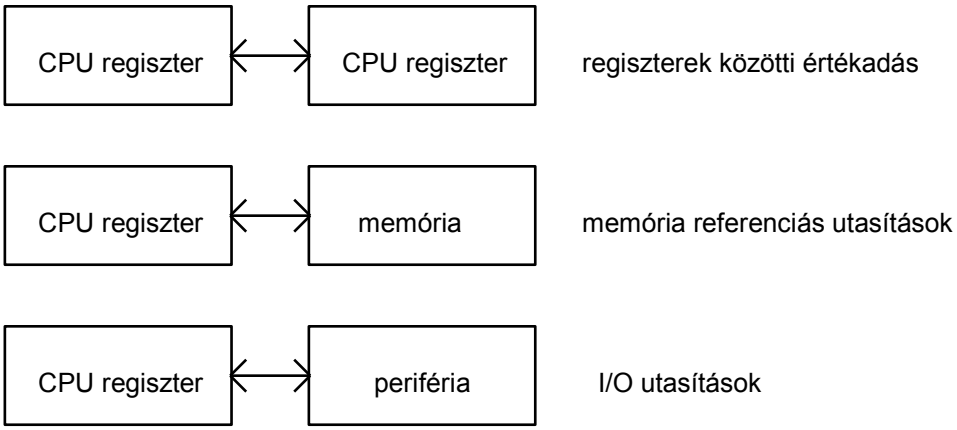
Az utasítás csoportok

A különféle mikroprocesszorok utasításkészlete többé-kevésbé eltérő, de a konkrét utasítások többnyire besorolhatók a következőkben megadott kategóriákba. Az utasításokra azok ún. mnemonikjával (nevével) szokás hivatkozni assembly nyelven. Ez az a string, amelyet feldolgozva egy assembler (az assembly nyelv fordítója) előállítja az utasítás kódját (azt a bináris adatot, amelyet a processzor értelmez).

Az alább felsorolásra kerülő utasítások közül vannak olyanok amelyeket egyik másik processzor nem ismer, ill. a ***mnemonikjuk*** *(nevük) nem egészen az itt megadottnak megfelelő! Az óriási választék miatt nem soroljuk fel az utasítások változatait, és olyanok is vannak, amelyeket meg sem említünk. A konkrét utasításokat és azok mnemonikját minden esetben megtalálhatjuk az adott CPU kézikönyvében.*

Adatmozgató utasítások

Az adatmozgató utasítások adatokat visznek át:
- a CPU regiszterei között
- a CPU valamely regisztere és a memória között (memória referenciás utasítások)
- a CPU valamely regisztere és a periféria között (I/O utasítások)



Adatmozgató utasítások forrás-cél szerinti csoportosítása

Az adatmozgatás vonatkozhat egy adatra, vagy egy adat blokkra, az utóbbiakat *blokkmozgató, tömb kezelő ill. string kezelő utasításoknak* nevezzük.

Az egyszerre átmozgatott adat szószélessége lehet byte, szó (2 byte), hosszú szó (4 byte) stb. processzortól és utasítástól függően.

Sok processzor rendelkezik olyan utasítással, amely két regiszter tartalmát felcseréli (exchange).

Az memória referenciás adatmozgató utasítások mnemonikja többnyire:

MOV cél, forrás

vagy LOAD cél, forrás

A stack kezelő utasítások is ide sorolhatók:

PUSH forrás a stack tetejére ír
POP cél a stack tetejéről olvas

Az I/O utasítások mnemonikja:

IN cél, forrás beolvas a perifériából
OUT cél, forrás kiír a perifériába

Adatkezelő utasítások (aritmetikai, logikai, shiftelő, rotáló, bit kezelő)

Az adatkezelő utasítások valamilyen manipulációt hajtanak végre a forrás operanduson. A művelet eredménye (op1-el jelölve) többnyire a CPU akkumlátorában keletkezik, így az akkumlátor az operandus tárolására is szolgál és célregiszter is.
A művelet típusa szerint a következő csoportok állíthatók fel:

Aritmetikai utasítások

Összeadás: ADD op1, op2
Kivonás: SUB op1,op2
A fenti utasítások carry flag-et figyelembe vevő változata is mindig létezik.

Szorzás: MUL op1,op2
Osztás: DIV op1,op2

Logikai utasítások

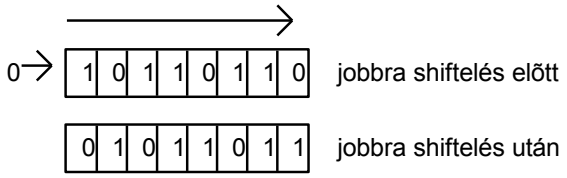
Bitenkénti ÉS: AND op1, op2
Bitenkénti VAGY: OR op1,op2
Bitenkénti XOR: XOR op1,op2

A bitenkénti logikai utasítások az operandusok azonos sorszámú bitjei között hajtódnak végre, párhuzamosan.

Shiftelő, rotáló utasítások

A shiftelő utasításoknál az adat i-edik bitje jobbra shiftelésnél az i-1-edik bit pozícióba másolódik s a legnagyobb helyiértékű bit törlődik.

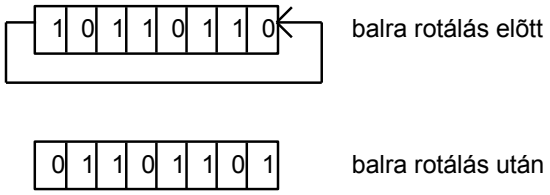
SHR op



Jobbra shiftelés

A rotálás olyan shiftelés. ahol a legnagyobb és legkisebb helyiértékű bitet egymás folytatásának kell tekinteni.

RR op



Rotálás

A shiftelés és rotálás jobb és bal irányba is történhet.

Bit kezelő utasítások

A bit kezelő utasítások valamely regiszter, memória esetleg periféria bitet képesek 1-be ill. 0-ba állítani, invertálni, esetleg valamely flagbe átmásolni.

SETB bit 1-be állítja a bitet
CLR bit törli a bitet
TEST bit a bitet bemásolja a Z flagbe

Vezérlés átadó utasítások

Ezen utasítások hatására a valamely feltétel flagtől függően vagy attól függetlenül a következő végrehajtandó utasítás nem a következő címen levő utasítás lesz.

Feltétel nélküli ugrás:

JUMP cím

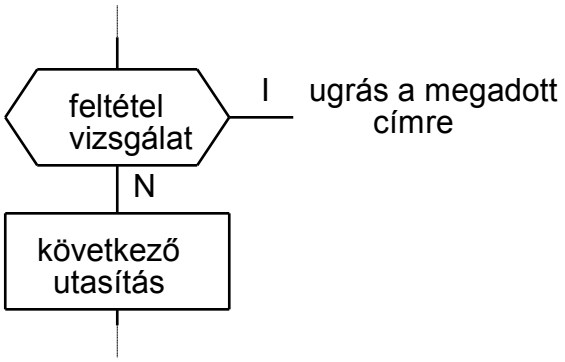
vagy JP cím stb.

Hatására a megadott címen folytatódik a program végrehajtás (PC=cím).

Feltételes ugrás:

JUMP feltétel, cím

Itt az ugrás csak akkor hajtódik végre, ha a feltétel teljesül, egyébként a következő utasításra adódik a vezérlés (4.27. ábra). Feltételként többnyire egy flag 0, vagy 1 értékét lehet előírni.



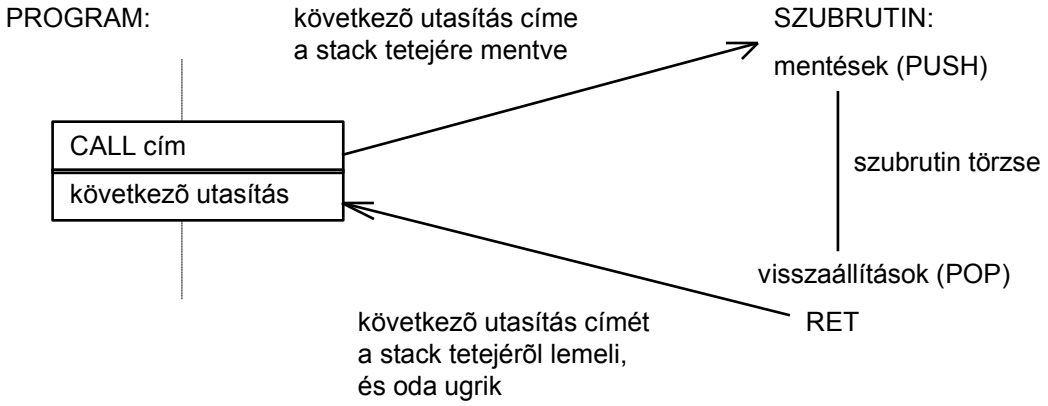
Feltételes ugrás

(Pl: JUMP NZ, cím ugorj, ha a zero flag értéke 0.)

Szubrutin hívások

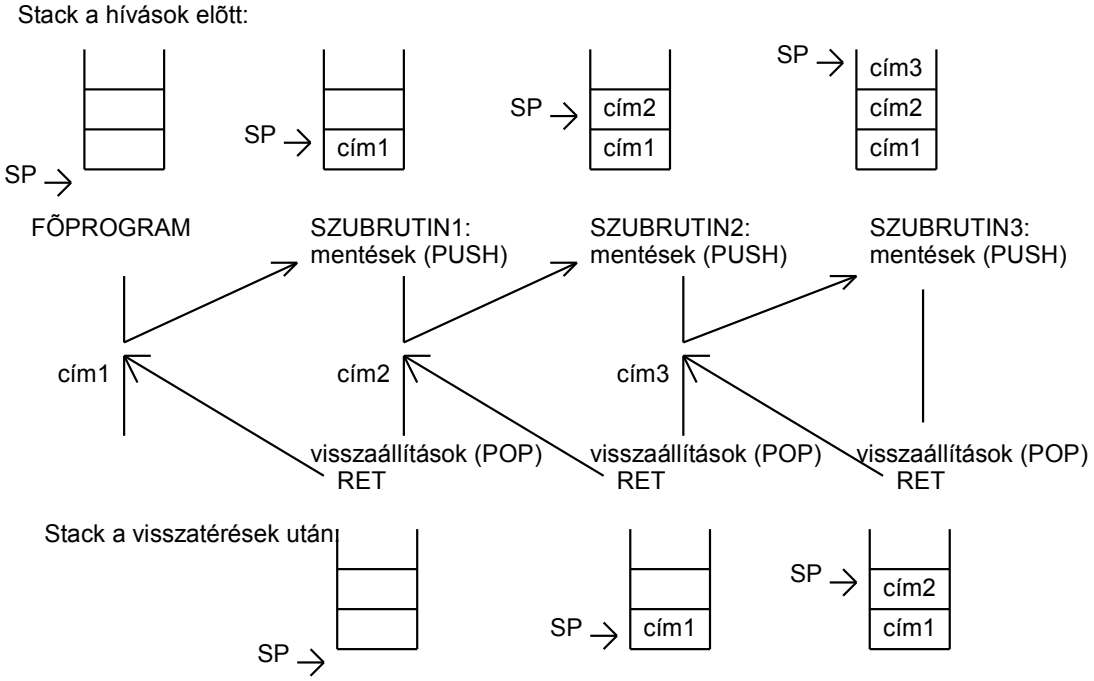
A programozás során sokszor előfordul, hogy bizonyos programrészeket azonos formában a program több különböző helyén kell végrehajtani. Ilyenkor, ha a programrészlet mérete nem túl kicsi, azt szubrutinként célszerű megírni. A szubrutinra a program tetszőleges helyéről lehet hivatkozni az ún. szubrutin hívással (CALL). Ez az utasítás a stack tetejére menti a következő utasítás címét, majd a szubrutin elejére ugrik.

A szubrutinból visszatérő utasítás (RET) leveszi a stack tetejéről a visszatérési címet, majd oda ugrik, vagyis visszatér a félbeszakított program folytatásához.



A szubrutin működése

A szubrutin hívásokat a stack terület által korlátozott mértékben egymásba lehet ágyazni, vagyis egy-egy szubrutinból újabbakat lehet hívni. Ezt a mechanizmust a stack teszi lehetővé. Három egymásba ágyazott szubrutin esetét mutatja az alábbi ábra.



Szubrutin hívás-visszatérés folyamata

Feltétel nélküli szubrutin hívás:

CALL cím

Az utasítás először a stack tetejére menti a következő utasítás címét, majd a megadott címre ugrik.

Feltételes szubrutin hívás:

CALL feltétel, cím

A szubrutin hívás csak a feltétel teljesülése esetén következik be.

Feltétel nélküli visszatérés szubrutinból:

RET

Az utasítás hatására a STACK tetejéről a PC-be kerül a visszatérési cím, és a következő utasítást erről a címről veszi a CPU. Feltételes változata is létezik. Külön meg kell említeni az interruptból való visszatérést (RETI) és a nem maszkolható interruptból való visszatérést (RETN), de ezekről részletesebben majd az interrupt tárgyalásánál szólunk.

Feltételes visszatérés szubrutinból:

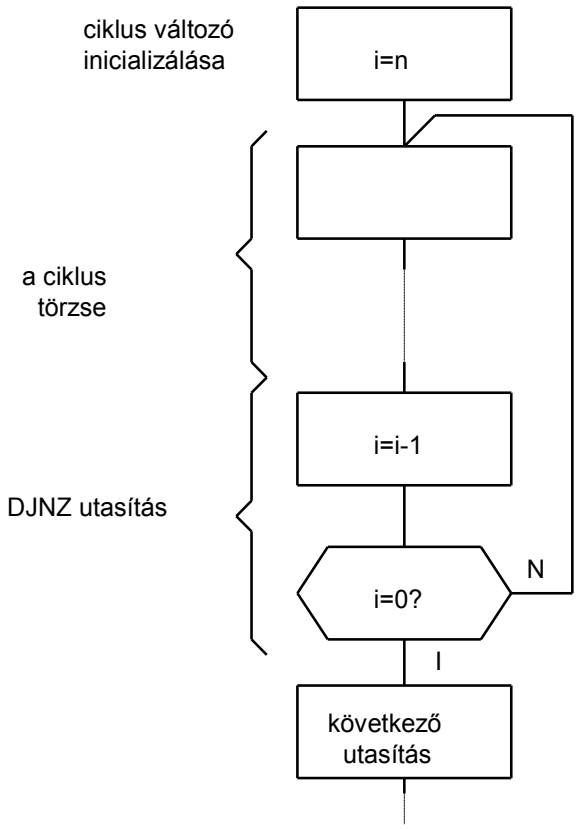
A feltétel teljesülése esetén a CPU visszatér a szubrutinból, egyébként a következő utasításnál folytatja a program végrehajtását.

RET feltétel

Ciklus szervező utasítások

Ezekkel az utasításokkal egy program részlet előre megadott számú ismétlése oldható meg. Az utasításhoz tartozik egy ciklus változó, melynek helye előre definiált (pl. egy adott regiszter). A ciklikusan ismételendő program részlet előtt a ciklus változót be kell állítani. Az utasítás a ciklus végén helyezkedik el. Dekrementálja a ciklus változó értékét, s ha az még nem 0, akkor a megadott címre (a ciklus elejére) ugrik.

DJNZ relatív cím



Ciklus szervező utasítás működése

CPU vezérlő utasítások

NOP: Időhúzó utasítás, nem csinál semmit (No operation). Ciklusban, vagy többször egymás után alkalmazva programból lehet időzítést előállítani..

HALT: Hatására interruptra várakozó állapotba kerül a processzor, NOP-okat hajt végre, amíg egy interrupt be nem következik, vagy RESET jelet nem kap a CPU.

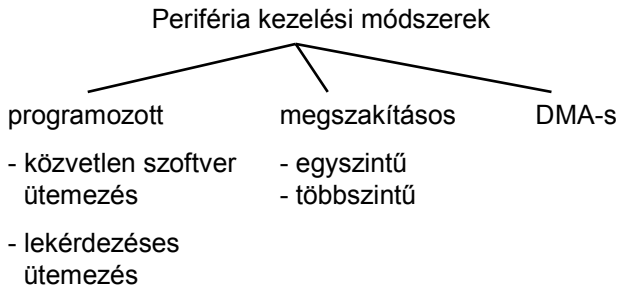
EI: Globális interrupt engedélyező utasítás.

DI: Globális interrupt tiltó utasítás. Hatására a CPU az NMI-n (nem maszkolható megszakításon) kívül nem fogad el megszakítást.

CPU-tól függően egyéb vezérlő utasítások is léteznek.

3.6. Periféria kezelési módszerek

Egy periféria kezelő program a perifériához vagy bármikor fordulhat, vagy a periféria jelzi, hogy mikor ez lehetséges. Utóbbi esetben a CPU kénytelen a perifériához alkalmazkodni (szinkronizálódni). Ezen célból a perifáriák rendelkeznek ún. státus regiszterrel. A státus regiszter olvasásával lehet információkhoz jutni a periféria állapotáról (foglalt vagy kész, kér-e megszakítást vagy sem, stb.) A CPU perifériához szinkronizálódásának célszerű módja a periféria és CPU relatív sebességétől függ. A periféria megkerülésével is lehetséges a periféria és a memória között adatátvitel. Ezeknek megfelelően az alábbi periféria kezelési módszereket különböztetjük meg:



Periféria kezelési módszerek csoportosítása

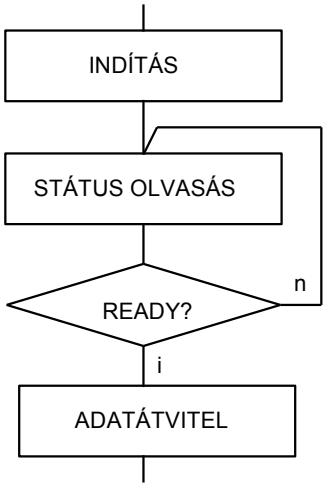
3.6.1. Programozott periféria kezelés

Közvetlen szoftver ütemezés

Egyes perifériáknál nincs szükség szinkronizálásra mert vagy nincs mihez szinkronizálódni, vagy a periféria egy művelet elindítása után a következő utasítás végrehajtására már biztosan elkészül. Az ilyen perifériákhoz a processzor tetszőleges időpontban fordulhat.

Lekérdezéses ütemezés

Ha a periféria sebessége olyan, hogy egy perifériás művelet néhány utasítás végrehajtási időn belül fejeződik be, akkor a szinkronizálást státus figyeléssel célszerű megoldani. Ilyenkor a program, a periféria adatregiszteréhez fordulás előtt ciklikusan addig olvassa a státusz regisztert, amíg az azt nem jelzi, hogy kész a következő műveletre (olvasásra vagy írásra). Csak ezután történik meg az adatátvitel. Ennek folyamatábráját mutatja a következő ábra.



4.56. ábra. Lekérdezéses ütemezés folyamatábrája

Ez a fajta kezelés lassú periféria esetében sok várakozással jár, ami alatt a CPU nem végez érdemleges működést. Ezért csak akkor engedhető meg, ha CPU egyéb feladatait így is képes elvégezni. Az igényelt hardver tekintetében általában a lekérdezéses ütemezés a legolcsóbb megoldás.

3.6.2. A program megszakításos (interruptos) periféria kezelés

Gyakran a periféria csak viszonylag hosszú idő vagy előre kiszámíthatatlan idő múlva lesz kész az újabb adatátvitelre. Ha ilyen esetben lekérdezéses ütemezést alkalmaznánk, akkor nagyon leromlana a processzor hatásfoka, hiszen az a státus figyelés időtartama alatt esetleg több száz egyéb utasítást is végrehajthatott volna.

Ebben az esetekben a periféria kiszolgálása hatékonyabb kényelmesebb, ha a periféria aktívan kezdeményezi a kiszolgálási igényét. Ekkor a főprogramban nem kell a perifériával törődni (időnként lekérdezni), hanem a jelzés (megszakítás) hatására időlegesen abbamarad a főprogram futása és automatikusan a perifériát kezelő program rész (periféria handler) aktivizálódik.

A perifériák a processzor IT kérő bemenetén keresztül jelzik, ha valamilyen kiszolgálást igénylő esemény történt (pl. befejeződött valamely működés, kiolvasandó adat keletkezett, újabb adat küldhető ki, hiba történt stb.).

Ha teljesülnek az interrupt elfogadás feltételei, s a CPU észlel egy megszakítás kérést, akkor *az aktuális utasítás végrehajtását befejezi*, s a következő utasítás végrehajtása helyett, *az interrupt kiszolgáló szubrutinra adja a vezérlést*.

A perifériát kiszolgáló szubrutin címének meghatározása processzor ill. hardver függő. A periféria azonosításához bizonyos esetekben (ha több perifériához ugyanaz az IT rutin kezdőcím van rendelve) a státus regisztert is ki kell olvasni. Maga az interrupt rutin némileg hasonlít egy közönséges szubrutinhoz, a különbség az aktivizálás módjában és az induláskor ill. visszatéréskor elvégzendő feladatoknál van.

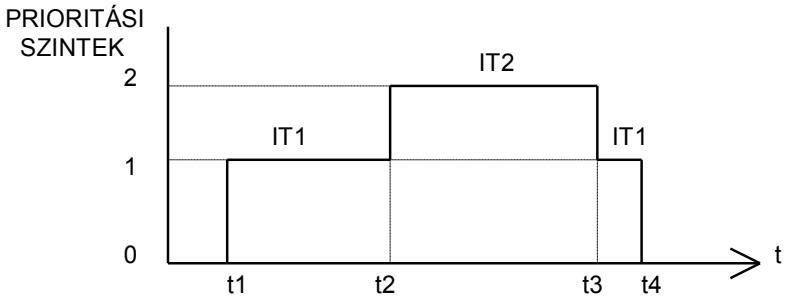
Ezek a feladatok a következők:

- Az interrupt rutin első utasításának végrehajtása előtt a CPU automatikusan a stack tetejére menti a visszatérési címet és esetleg a CPU interrupt előtti állapotára vonatkozó információkat tartalmazó állapotszót, továbbá többnyire letiltja a további interrupt elfogadást a CPU-ban.
- A periféria kezelését az IT rutin elvégzi, engedélyezi a további IT-ket, majd egy speciális (return from interrupt, RETI) utasítás végrehajtása után visszatér a megszakítás észlelését követő utasításhoz. A RETI utasítás egyrészt a CPU automatikusan elmentett állapotszavát állítja vissza (ha volt ilyen), másrészt a visszatérési címet veszi le a stack tetejéről, s erre a címre adja a vezérlést.

Többszintű interrupt rendszer

Ha az aktuálisan futó interrupt rutint megszakíthatja egy fontosabb, gyorsabban kiszolgálandó (magasabb prioritású) interrupt, akkor többszintű interrupt rendszerről beszélünk. Ebben az esetben az egyes interruptos perifériák ún. *prioritási szintek*hez vannak rendelve, a legfontosabb prioritása a legmagasabb.

A következő. ábrán 2 IT forrás szakíthatja meg a főprogramot, ill. a magasabb prioritású a kisebbet (a főprogram a legkisebb prioritású, azt mindenki megszakíthatja). A prioritás növekvő sorrendben IT1, IT2. Itt t1 időpontban IT1 megszakítja a főprogramot, t2-ben IT1-et megszakítja IT2, t3-ban befejeződik IT2 és folytatódik IT1, t4-ben befejeződik IT1 és folytatódik a főprogram.



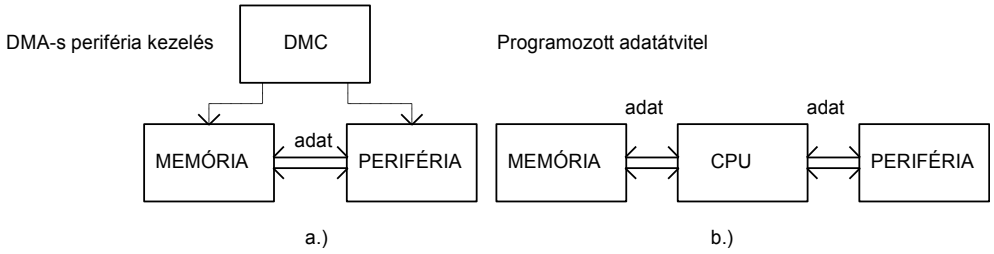
Többszintű IT rendszer működése

Szoftver megszakítások

Némely processzorban a megszakítások szoftverből is kiválthatók, speciális utasítások segítségével (pl. I8x86, INT i utasítás). Az ilyen megszakítások azonban inkább speciális szubrutin hívásnak tekinthetők, hiszen nem szakítják meg egy program futását. Az elnevezés inkább arra utal, hogy a meghívott rutin címének előállítása hasonló módon történik, mint a szokásos interrupt esetén.

3.6.3. A közvetlen memória hozzáférés (DMA, Direct Memory Access)

Az ún. DMA vezérlő segítségével a processzort kikerülve, közvetlen adatátvitel lehetséges a memória és a periféria között. Ezt nevezik közvetlen memória hozzáférésnek Ennek egyrészt az az előnye, hogy mivel az adatátvitelt egy speciálisan erre a célra kialakított hardver végzi, az többnyire gyorsabb, mintha a CPU végezné (nem kell közben a memóriából utasításokat olvasni és nem kell az adatot a CPU-n keresztül áramoltatni. Másrészt, a DMA-s kezelés tehermentesíti a processzort (egyszerűbb lesz program) A DMA-s kezelést leginkább nagy sebességű, blokkos adatátvitelt igénylő perifériáknál alkalmazzák (floppy disk, winchester).



Az adat útja DMA-s és programozott kezelés esetén

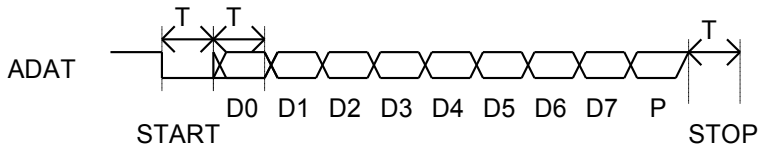
Az átvitel lebonyolítását az ún. DMA vezérlő (DMC, DMA Controller) végzi. Ez egy speciális periféria, amely felprogramozása után busz master funkciót képes ellátni (átvenni a CPU-tól). A DMA vezérlő főként a periféria és a memória közötti átvitel CPU-nál gyorsabb elvégzésére készített speciális hardver elem.

3.7. Tipikus számítógépes perifériák és interfészek

A számítógépek a környezettel a perifériákon keresztül tartják a kapcsolatot. A perifériák és a mikroprocesszoros rendszer közötti illesztést végzik a periféria illesztők, másnéven interfészek Az alábbiakban néhány általánosan használt interfészről és perifériáról lesz szó. A periféria elnevezést a szakirodalom sokszor magára az interfészre ill. az interfésszel egybeépített perifériára is alkalmazza. Az alábbiakban főként a személyi számítógépekben (PC) is alkalmazott interfészekre koncentrálunk.

3.7.1. Aszinkron soros port, RS232 interfész

Az adatok mozgatásához két egység (pl. mikroprocesszoros rendszer és periféria) között, a jelvezetékekkel való spórolás miatt gyakran alkalmazzák a bitsoros átvitelt. Ennél az átviteli formánál az adatok az adó egységből egyetlen jelvezetéken, bitenként időben egymásután jutnak el a vevő egységbe. Ennek egy speciális esete a PC soros portjánál is alkalmazott ***aszinkron soros átvitel***. Jellemző idődiagrammja a következő ábrán látható:



Aszinkron soros átvitel idődiagramja

Az átviendő minden egyes bit-re T idő jut, ezt az időtartamot ismeri az adó és a vevő egység, ennek reciproka a BIT rate, amit sokszor helyetlenül BAUD rate-nek neveznek. Alapvetően (ha nincs átvitel) az adatvezeték magas szintű. Az átvitel kezdetét az alacsony szintű START bit jelzi. Ezután jönnek az adatbitek (általában 8). Az adatbitek számát is ismeri az adó és a vevő egység. Az átvitel végén kötelezően egy magas szintű STOP bit van. A hardver többnyire lehetővé teszi, hogy a legutolsó adat bit után egy paritás bit jöjjön, mellyel az adat pártlan bitszámú sérülése jelezhető. Az RS232 interfész esetén az adatvonalra egy szintkonverter kerül, mely a szokásos logikai szinteket az RS232 szabványban rögzített feszültség szintekre konvertálja. Az RS232 esetén két adatvonal van, egy adás és egy vétel oldali, hogy egyszerre lehessen adni és venni. Ezen kívül egyéb ún. modem vezérlő jelek is vannak, melyek használata azonban nem kötelező. RS232 interfész esetén az egységek közötti távolság néhány méternél nem lehet több. Más jellegű meghajtó és vevő áramköröket tartalmazó interfész esetén, pl. RS485, a 100 métert is meghaladhatja a távolság (de az RS485 nem kompatibilis az RS232-vel). RS232 esetén a legnagyobb átviteli sebesség 100 kBaud nagyságrendű.

Példák soros porton keresztül kezelt perifériákra, soros adatkapcsolatra:

- **Egér:** Az egér grafikus mutató eszköz. RS232 interfészre, mozgatásakor az x és y elmozdulás értékét, ill. az egéren levő nyomógombok állapotváltozását küldi el a soros portnak. További információk: [2] 92. oldal.
- **Tablet:** A tablet, az egérhez hasonlóan grafkus beviteli eszköz. Egy speciális táblára lehet vele, egy speciális toll segítségével írni. A tablet hardvere érzékeli, hogy milyen koordinátájú pont felett van a toll és ezt ill. a tollon levő nyomógomb állapot változását küldi el a soros vonalon.
- **Modem:** A modem segítségével a számítógép telefonvonalon keresztül képes információ cserére. A számítógép felé egy külső modem aszinkron soros porton keresztül csatlakozik. A modemek általában beépített adattömörítési, és hiba jelzési/javítási lehetőséggel rendelkeznek
- Két számítógép is összeköthető RS232 interfészen keresztül. A PC-n futó bizonyos programok lehetővé teszik adatok átvitelét a PC-k között. Gyakran a célszámítógépeket tartalmazó mérőműszerek is ilyen a porton keresztül kapcsolhatók PC-hez, a mérési adatok továbbítása, archiválása, további feldolgozása céljából.

3.7.2. Párhuzamos port

A párhuzamos portokon keresztül egyszere több bit információt továbbítanak. Így az információ továbbítás sebessége nagyobb lehet, mint bitsoros esetben. A PC-ben eredetileg kialakított a párhuzamos port a teljesen programozott kezelés miatt viszonylag lassú (150kbyte/sec) és elsősorban a mátrixnyomtatókhoz való csatlakozást volt a

feladata. Az eredeti párhuzamos port egyirányú adatátvitelre képes (PC-> printer. A mai PC-k párhuzamos portjának különféle üzemmódjai (EPP mód, ECP mód) már viszonylag nagy sebességű kétirányú átvitelt is lehetővé tesznek (500kbyte/sec-2Mbyte/sec). Bizonyos portok ECP módban valós idejű adattömörítésre is képesek (RLE, Run Length Encoding), mely csak akkor használható, ha a kapcsolódó periféria is képes erre. Ez a kódolás a raszter képek továbbításánál (nyomtatók, kép scannerek) jelentős sebességnövekedést tesz lehetővé.

A printer portra egyéb perifériák is rákapcsolhatók:

- bizonyos CD ROM-ok,
- hordozható winchesterek
- a kép scannerek egy része,
- laser nyomtatók

3.7.3. Nyomtatók

A nyomtatók segítségével lehet papírra nyomtatni a számítógépben tárolt információkat. Ezek karakteres, vagy grafikus jellegűek lehetnek. A nyomtatók képesek külön kezelni a karakteres és grafikus információkat. Erre azért van szükség, mert így a karakteres információkat gyorsabban feldolgozhaók, hiszen a nyomtatható karakterek grafikus képe előre el van tárolva a nyomtatóban (vagy letölthető). Így ebben az esetben sokkal kevesebb információt kell küldeni a nyomtatónak. Grafikus esetben minden egyes képpont elhelyezkedéséről el kell küldeni az információt a nyomtatónak A következő nyomtató tipusok a legelterjedtebbek:

Mátrix nyomtató
Itt egy 9, 12, 18 vagy 24 tűből álló fej nyomtatja egy festékszalag megütésével a nyomtatási kép pontjait. Az egyes tűket pici elektromágnesek mozgatják. Az egész nyomtató fejet is folyamatosan mozgatják.

Laser nyomtató
Itt először egy elektrosztatikusan feltöltött félvezető hengeren alakítják ki a nyomtatási képet, LASER vagy LED fényének segítségével. A fényt egy tükör rendszer pozicionálja a megfelelő helyre. A megvilágított pontokból eltávoznak a negatív töltések, ezért a por alakú, elektrosztatikusan feltöltött festéket magukhoz vonzzák. A hengerre tapadt festéket ellentétes térrel viszik át a papírra, mely hőkezelés hatására ott is marad (ráolvad).

Tintasugaras nyomtató
A tintasugaras nyomtató esetén sok vékony csövecskéből lövell a tintasugár a papírra. A tintasugarat piezoelektromos kristállyal vagy hőhatásra kialakuló piciny gőzbuborék segítségével lövik ki és vagy elektrosztatikus térrel pozicionálják a megfelelő helyre, vagy egy fejen sok (pl. 128 db) csövecske van, melyeket a kialakítandó raszterpontoknak megfelelően vezérlenek

A nyomtatókról jó összefoglaló található a [2]-ben, 81-87 oldalakon.

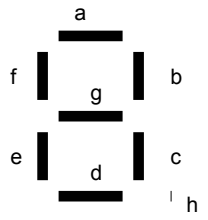
3.7.4. Billentyűzet

A legtöbb számítógép rendelkezik valamilyen közvetlen kezelői bemenettel, nyomógombokkal, billentyűzettel. A különféle számítógépek/mikroszámítógépek különféle módon kezeli a billentyűzet nyomógombjait. A célszámítógépekben (pl. műszerekben levő számítógépek) általában speciális hardvert alakítanak ki e célra.
A legelterjedtebb standard billentyűzet a *PC billentyűzete*. Ez *szinkron soros interfészen* keresztül csatlakozik a számítógéphez. Az adatátvitel két jel segítségével történik, egy szinkron órajel (SCK), melyet a billentyűzet ad ki (de a PC képes letiltani) és egy kétirányú adat vezeték (SDAT), melyet hol a billentyűzet, hol a számítógép hajt meg. A PC billentyűzete minden billentyű lenyomása és felengedése esetén átad egy-egy billentyű kódot, mely több byte-ból állhat (ennek semmi köze az adott billentyűhöz esetleg hozzáendelt karakter ASCII kódjához). A PC is küld a billentyűzetnek információt, pl. gyújtsa ki a CAPS LOCK LED-et. További információk: [2] 90-91 oldal.

3.7.5. A kijelzők

A célszámítógépek és általános célú számítógépek rendelkelznek valamilyen kijelzővel, melyen a számítógép információkat jelenít meg. Célszámítógépek esetén a kijelzők lehetnek egyszerű LED-ek (Light Emitting Diode, fényemittáló diódák), amelyek a készülék állapotáról adnak információt, lehetnek számjegyes kijelzők, melyen egy műszer a mért mennyiséget is képes kijelezni.

A 7 szegmenses kijelző
Ha csak számokat kell kijelezni, azt 7 szegmenses kijelzőkön szokás megjeleníteni. A *7 szegmenses kijelző* az alábbiak szerint elhelyezett 7 db szegmens segítségével jeleníti meg a számokat:



A kijelző egy számot megjelenítő kijelzőjét digitnek (számjegy) nevezik. Különféle elveken működő kijelzők vannak:

LED kijelző: LED-ek alkotják a szegmenseket, ezek sötétben jól láthatóan világítanak, erős fényben kevésbé, bár ma már nagy fényerejűek is léteznek. Viszonylag nagy a teljesítmény felvétele.
LCD (Licquid Cristal Display) kijelző: a folyadékkristály elektromos tér hatására megváltoztatja a fény polarizációs síkját, ezért a polárszűrő mögé tett folyadékkristállyal

a visszavert ill. áteresztett fény mennyisége megváltoztatható. A kijelző erős külső fényben is jól látható, viszont korlátozott a szög, amelyből még jól láthatók. Léteznek háttérvilágítással rendelkezők is. Nagyon kicsi a teljesítmény felvétele.
Vákum fluoreszcens kijelző (VFD, Vacuum Fluorescent Display): Ez tulajdonképpen egy vákkuum cső, melyeben a szegmenseket a foszforral bevont anódok alkotják, ebbe csapódnak be a fűtőszálon keletkező elektronok, az anód és katód közötti 12-40V nagyságrendű feszültség által létrehozott elektromos tér hatására. Viszonylag erős fényt ad, kis teljesítményfelvétel mellett.

Karakteres kijelzők
A kijelzők egy másik csoportja karaktereket is képes megjeleníteni. A karaktereket vagy a 7 szegmenses kijelzőkhöz hasonló, de több szegmensből álló részekből, vagy pontmátrixból állítják elő. Ez utóbbi az elterjedtebb. LED-es LCD-s és VFD

Grafikus kijelzők
Ha betűk és számok mellett grafikus jellegű kijelzésre is szükség van, akkor alkalmazzák a grafikus kijelzőket. Ezek pontokból állítják elő a grafikus képet.

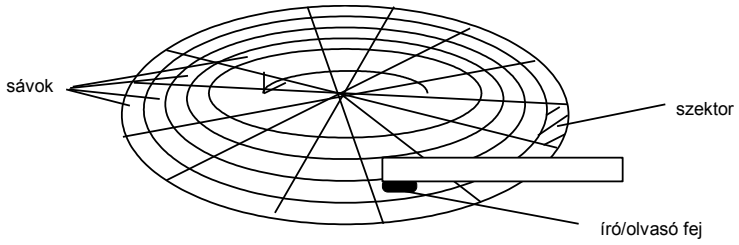
Kis felbontású grafikus LCD kijelzők
Az egyszerűbb kijelzőkben (egyszerűbb műszerekhez, zsebszámológépekhez) a raszter pontok sötétségi szintjét (gray scale) nem, vagy csak kis felbontással (4) lehet szabályozni. Ezek jól használhatók menük és grafikonok megjelenítésére. Ilyen célra főként LCD kijelzők terjedtek el. Felbontásuk 128x128 pont nagyságrendű.

Közepes és nagy felbontású grafikus kijelzők
A nagy felbontásúak már a PC-nél elterjedt VGA (640x480) felbontást is tudják, s a raszter pontok sötétségi szintjét is sokkal nagyobb felbontással kezelik, így fekete-fehér képek megjelenítésére is alkalmasak. Régebben katód sugár csöves (CRT, Chatode Ray Tube) változatokban is gyártották.

Nagy felbontású szines grafikus kijelzők
Ezek a kijelzők nagy felbontású VGA, SVGA szines képek megjelenítésére alkalmasak. Léteznek LCD, plasma display és CRT változatokban.
A *CRT display*-ben egy elektron sugárat irányítanak egy foszforeszkáló bevonattal ellátott felületre. Az elektron által bombázott bevonat világítani kezd, az elektronsugárral szabályozható fényerővel. Így vizszintesen, sorról sorra rajzolja ki a képet az eredeti képnek megfelelően modulált elektronsugár. Szines esetben egy pont 3 különböző szinű vörös (red) , zöld (green), és kék (blue) foszforeszkáló pontból áll elő (innen az RGB rövidítés). 3 elektronsugár rajzolja a képet, mindegyik csak az egyik színt állítja elő, s a 3 színből keveredik ki a kívánt szín. A sorokat közvetlenül egymás alá rajzolja ki az elektronsugár (non interlased mode) vagy pedig először a páratlan sorszámú sorokat, majd a páros sorszámú sorokat (interlaced mode). Utóbbi esetben kevésbé vibrál a kép. A teljes képek 50-100Hz-el követheik egymást, a monitor tipustól és vezérlőtől függően. Gyorsabb képváltás kevésbé vibráló képet eredményez.

3.7.6. Mágnes lemezes háttértárak

Winchester (hard disc, kemény lemezes tároló)
A winchester egy közvetlen elérésű háttértár, mely egy vagy több kemény mágneslemezen tárolja az adatokat. Az adatokat tartalmazó blokkot közvetlenül lehet címezni, írni, olvasni. (Szemben a mágnesszalagos háttértárolóval, amelynél csak szekvenciálisan lehet elérni az adatokat, vagyis oda kell tekerni a szalagot az adatot tartalmazó részhez). A winchester egy-egy lemezén egy vagy több olvasó fej olvassa/írja az adatokat. A lemez nagy fordulatszámmál forog. A fejet sugár irányban képes mozgatni a mechanika.
A mágneslemez az adatok tárolása szempontjából koncentrikus körök mentén elhelyekedő sávokra , egy-egy sáv sáv pedig további részekre, ún. szektorokra van osztva. A szektor lemezről leolvasható legkisebb egység, melyen 256-1024byte nagyságrendű adat található. Az egymás alatt elhelyezkedő sávokat cilindernek nevezik. Az első használat előtt szükséges hogy a lemezekre a további írást segítő szinkronizáló információkat is felírjanak, ezt nevezik alacsony szintű formázásnak. A mai winchesterek ilyen szinten gyárilag előre formázottak. A lemez további szervezését, a particionálást és magas szintű formázást az operációs rendszerek segédprogramjaival lehet elvégezni.



- Egy mai átlagos winchester 5-20Gbyte adat tárolására képes. A nagyszámítógépekhez ennél jóval nagyobb méretűek is léteznek.
- Az adatok elérése viszonylag lassú, több msec (10^-3sec) nagyságrendű. Összehasonlítva a félvezetős memóriákkal, pl. DRAM-mal amely kb. 6*10^-9sec hozzáférési idejű, ez *egymilliószor lassabb* adattároló eszköz!

A winchestert valamely szabványos párhuzamos porton keresztül illesztik a processzoros rendszerekhez. Az adatátvitel blokkosan történik, vagyis egyszerre egy de inkább több szektor adatait viszik át. Az adatátvitel többnyire DMA-val történik, vagyis közvetlenül a memóriába.
A winchester a adatátvitelének gyorsítására gyorsító tárat alkalmaznak. A tapasztalatok szerint, ha egy adatblokkra szükség van a winchesterről, akkor nagy valószínűséggel az azt követő blokkra is szükség lesz. Pl. az operációs rendszert is tartalmazó processzoros rendszereknél a programokat és bizonyos adatokat a winchesteren tárolják, összefüggő, szerkezettel rendelkező adathalmazként, amit file-nak neveznek és onnan töltik be. Egy-egy file általában összefüggö szektor területen tárolódik. Így egy program betöltése során

sok egymás utáni szektort kell betölteni. Ezért a winchester és a proceszoros rendszer közé RAM memóriából kialakított gyorsító tárat tesznek (ezt ma már a winchesterekbe is beépítik) és ha pl. adatot kell olvasni a winchesterről, akkor az igényelt adatblokkot tartalmazó szektorokat követő szektorokat is kiolvassák és a gyorsító tárban tárolják. Mivel legközelebb nagy valószínűséggel ezekre az adatokra lesz szükség, ezért azokat ekkor a sokkal gyorsabb gyorsító tárból lehet elővenni. Írásnál hasonlóan a gyorsító tárba írnak, s nem kell megvárni a gyorsító tárbó a winchesterbe történő a tényleges átírást, mert ezt automatikusan elvégzi a gyorsító tár vezérlője.
A winchesterek általában a PC-vel, számítógéppel egybeépítve helyezkednek el, de léteznek hordozható változatok is, sőt a normál winchester is hordozhatóvá tehető, ha hordozó keretet építünk a PC-be. Az ilyen keret két részből áll, az egyik a számítógépbe van építve, s a másik, szállítható részébe van beszerelve a winchester, ez könnyen bedugható ill. kivehető a számítógépből. Azonban a szállításnál óvatosnak kell lenni, mert ütődésre megsérülhet a winchester. Egyébb információk: [1] 146-148 old. és [2] 62-65 old.

Floppy disk (hajlékony lemez)
A floppy disk szintén mágnes lemez, amit adatok tárolására és hordozására használunk (két gép közötti adat átvitel). Itt magát a floppy lemezt lehet hordozni. Az adatárolás módja és szervezése (sávok, szektorok) hasonló mint a winchesternél. A floppy disk viszonylag kevés adatot képes tárolni (1.2, 1.44, 2.88 Mbyte), és jelentősen lassabb mint a winchester.

3.7.7. CD ROM és DVD

A CD ROM és a DVD optikai elven tárolja az információkat. Egy szokásos CD lemezen tipustól függően 650 Mbyte vagy 700 Mbyte információ tárolható. Egy DVD lemezen tipustól függően 4.7, 8.5, 9.4 vagy 17 Gbyte. A felületükön levő többrétegű anyagon alakítanak ki a tárolandó információs bitek alapján fény visszaverő ill. elnyelő részeket. A kiolvasás a forgó lemezt megvilágító LASER fény segítségével történik. A visszavert fényt modulálja a felületen kialakított információs réteg, amely alapján visszaállítható az eredeti adattartalom. Írás esetén a LASER hő hatása váltja ki a fényvisszaverő képesség megváltozását. Az adatok tárolása nagyon jó hibajavító tulajdonságú kódolással történik, hogy a felületi karcok lehetőleg ne okozzanak adatvesztést. Ennek ellenére ezeket a lemezeket is óvatosan kell kezelni, csak a szélükön illik megfogni, és a tokjukban kell tárolni. A CD-ről és DVD-ről jó öszefoglaló található a [2] 65-73 oldalakon.

3.8. A processzorokban és számítógépekben alkalmazott sebességnövelő módszerek

A program végrehajtás sebességét részben az alkalmazott technológiával gyártható áramkörök sebessége határozza meg, azonban a gyorsabb áramköröket eredményező technológia jelentősen drágább. Megfelelő struktúrális kialakítással ugyanolyan

sebességű áramkörök esetén is jelentő sebességnövekedés érhető el. A következőkben néhány ilyen, a mai számítógépekben alkalmazott sebességnövelő elveről lesz szó.

Nagy méretű regiszter tömbök a CPU-ban és a RISC architectúra

A CPU a leggyorsabban a belső regisztereihez képes hozzáférni. Ezért ha CPU-ban a regiszterek számát jelentősen megnövelik, akkor ritkábban kell a külső tárolóhoz fordulnia operandusért, hiszen hosszú ideig dolgozhat csak a regiszterekben levő adatokkal. Ehhez a kialakításhoz szorosan kapcsoldik a **RISC** (Reduced Instruction Set Computer) kialakítású processzor. Itt a processzor utasításkészlete kevés számú, csak a belső regisztereken manipuláló és egy memóriába író ill. egy onnan olvasó, egyforma hoszúságú utasításból áll, melyeket egyórajel alatt képes végrehajtani a CPU. Így minden bonyolultabb műveletet ezek segítségével kell megoldani. Ez az architetctúra több utasításból álló programot eredményez, de a tapasztalat az, hogy a bonyolult utasításokra viszonylag ritkán van szükség. Az adott algoritmus végrehajtásához tartozó több utasítás miatt megnőtt utasításbeolvasási igény ellenhat a nagy regisztertömbbel elért gyorsításnak, ezért gyors külső memória hozzáférés is szükséges a kellő hatékonyság eléréséhez.

Pipe line

A pipe line (cső) elve tulajdonképen a futószalag feldolgozás elvének digitális változata. Az elvégzendő feladatokat egymás után önállóan elvégezhető műveletek sorára osztjuk. Az egyes műveleteket önálló művelet végző egységek hajtják végre. Ha egy feladatot, amelynek komplett elvégzése Tf időbe telik M1, M2, M3 egymás után elvégzendő műveletekre osztjuk, és az egyes feladatok kb. azonos idő alatt (Tm=Tf/3) végezhetők el, akkor, ha folyamatosan kell elvégezni ugyanazt a feladatot, a futószalagon egyszerre (átlapoltan) dolgozik mindhárom műveletvégző egység. Így egy feladat átlagosan Tm idő alatt készül el. Ezzel szemben, ha nem futószalagon készülne, hanem egyetlen műveletvégző végezné el az egész feladatot, akkor egy feladat elvégzése Tf=3Tm időt igényelne.

Ezt a megoldást használják egyes CPU-kban, pl. az utasítás pipe line-ban a műveletek lehetnek: utasítás olvasás (UO), utasítás dekódolás (UD), operandus beolvasás (OO), utasítás végrehajtás (UV). Amíg a CPU végrehajtja az i-edik utasítást, beolvassa a az i+1-edik operandusát, dekódolja az i+2-edik utasítást és beolvassa az i+3-adik utasítást.

i

	T	t+1	T+2	t+3	t+4	t+5	t+6
Utasítás olv.	U1O	U2O	U3O	U4O	U5O	U6O	U7O
Utasítás dek.		U1D	U2D	U3D	U4D	U5D	U6D
Operandus olv			O1O	O2O	O3O	O4O	O5O
Utasítás végreh.				U1V	U2V	U3V	U4V

Az utasítás lehívás-dekódolás pipe line szerű szervezésénél akkor van probléma, ha elágazó utasítást kell végrehajtani. Amikor ezt dekódolja a CPU, kiderül, hogy nem az lesz a következő utasítás, ami a pipe line-ban van. Ezért ennek végrehajtása során a teljes

pipe line-t ki kell üríteni, ami lassulást eredményez a feldolgozásban. Ennek megoldására is léteznek módszerek.
Egyes processzorokban magát az utasítás végrehajtást is pipe line-szerűen szervezik.

Párhuzamos működés a processzoron belül
A processzoron belüli egységek többszörözésével megoldható, hogy egy CPU egyszerre több különböző utasítást is végrehajtson. Ezzel szintén megnő az egységnyi idő alatti utasítás végrehajtás. Így működnek a szuperskalár processzorok.

Többprocesszoros (multiprocesszoros) rendszer
A program végrehajtási ideje azzal is csökkentehető, ha több processzor áll rendelkezésre a feladatok megoldására. Ekkor egyrészt szét kell osztani a feladatot a processzorok között, párhuzamosan elvégezhető részekre, másrészt meg kell oldani a processzorok közötti kommunikációt és szinkronizációt, hogy az eredményeket egymásnak átadhassák. Megoldandó feladat a közös erőforrások kezelése is.

A főtárhoz való hozzáférés gyorsítása
A CPU-nak a főtárban levő utasításokhoz, vagy adatokhoz való hozzáférését a főtár (memória) hozzáférési ideje korlátozza. Nagy méretekben olcsón dinamikus memóriákat tudnak előállítani, ezért ilyen memóriákat alkalmaznak főtárként. A dinamikus memóriáknak viszont hátránya, hogy kb. 10-szer hosszabb a hozzáférési idejük, mint a gyors statikus RAM-oké, s ez nagyon lelassítaná a CPU működését. Ezen a problémán a *főtár és a CPU közé* tett gyorsító tárral (cache, ejtsd: kes) lehet segíteni.

Cache memória (asszociatív gyorsító tár)
A cash a *a program lokalitás elvét* használja ki. Egyrészt, ha egy utasítást elővesz a CPU, akkor nagy valószínűséggel a következő címen levő utasításra lesz szüksége legközelebb, a szekvenciális utasításvégrehajtás miatt (térbeli lokalitás). Másrészt, ha egy utasításra szüksége van a CPU-nak, akkor rövid időn belül újra szüksége lehet rá (időbeli lokalitás, a ciklikus programrészek miatt).

A cache működési elve
A cache memória egy – a főtárhoz képest – gyors és kis méretű, asszociatív szervezésű statikus memória. A cache memóriában (ha még nincs benne) eltárolják az aktuálisan végrehajtott utasítást és néhány utána következő utasítást (egy utasításokból álló blokkot). Így a következő utasítást a szekvenciális utasítás végrehajtás miatt a lassú főmemória helyett nagy valószínűséggel a sokkal gyorsabb cache-ből veheti elő a CPU. Ha pl. éppen egy ciklikus program részt hajt végre a CPU, és a ciklikus törzse teljesen belefér a cache-be, akkor az első végrehajtás után benne is lesz. Ezért, amíg a ciklus be nem fejeződik, vagy meg nem szakítja egy interrupt, a cache-ből dolgozhat a CPU, ami jelentősen megnöveli a program végrehajtás sebességét.

Asszociatív keresés (cím alapján)
A cache-ben az adat blokkal együtt eltárolják a blokk főmemória beli helyére vonatkozó információt is. A cache a cím alapján megmondja, hogy az adott című adat benne

található-e, s ha igen, akkor az adatot is kiadja. Ezt nevezik *cache találatnak* (cache hit). Ha nem, (cache tévesztés, cache miss) akkor a cache vezérlő kiolvassa az adatot tartalmazó blokkot a főtárból, az aktuálisan igényelt adattal kezdve, amit megkap a CPU és a blokk beíródik a cache-be. Ez jelentősen hosszabb ideig tart, mint a cache-ből történő elővétel. A tapasztalatok szerint a cache találati arány (hit rate) a cache méretétől és kialakításától függően jobb min 90%.

Melyik blokkot írja felül, ha betelt a cache?
Ha a cache-be újabb blokkot kell írni, de már betelt, akkor el kell dönteni, hogy melyik blokkot írja felül. Erre különféle stratégiák léteznek. Egy lehetséges algoritmus, hogy a legkevesebbszer használtat írják felül. Ehhez plusz információkat kell tárolni a cache-ben, vagyis számolni kell, hogy hányszor fordult a CPU egy-egy adat blokkhoz.

A cache-beli és főmemóriabeli levő adatok egyezőségének biztosítása (adat konzisztencia)
A cache-be írás esetén, biztosítani kell az *adat konzisztenciát*, vagyis, hogy a főtárbeli adat is módosuljon (egyezzen meg a főtár és a cache tartalma). Erre különféle algoritmusok léteznek. A *közvetlen átírási* (write through) stratégiánál az adatblokkot elkezdik átírni, amint a cache-beli adat módosult. A viszaírási stratégiánál csak akkor írják vissza, ha a módosított cache-beli blokkot felül kell írni egy újabb blokkal. Ez utóbbi esetben nyilván kell tartani a cache-ben, hogy egy adatblokk módosult-e a beolvasása óta.

3.9. Egységesített hardver kezelés, egységes felület a futtatandó programok és a kezelő felé

A különféle általános célú számítógépekben más és más a hardver kialakítása, ugyanakkor ugyanazokat a rendszer és felhasználói programokat szeretnénk futtani rajtuk. Ez akkor lehetésges, ha egyrészt a különféle gépekben alkalmazott processzor utasításkészlete kompatibilis (bár ez nem szükségszerű, mert megfelelő interpreter segítségével megoldható, csak lassabb lesz a program futása). Másrészt a különféle képpen kezelendő hardver elemek kezelését egységes hívási/paraméter átadási felülettel rendelkező periféria kezelő programokkal (handlerek) oldják meg. Az ilyen programok egy részét a számítógép ROM-ja tartalmazza. Ezt nevezik *BIOS*-nak (Basic Input Output System).
A PC BIOS-a többek között például a következő feladatokat képes elvégezni:

- Billentyű lekérdezése
- Soros port felprogramozása
- Karakter beolvasás a soros portról
- Karakter kiírás a soros portra
- Párhuzamos (printer) port felprogramozása
- Karakter küldése a printer portra
- Idő lekérdezése
- Karakter pozíciójának beállítsa
- Karakter kiíratása

- Szektor formázása
- Szektor kiírása
- Szektor beolvasása
- Operációs rendszer betöltőjének betöltése

Az általános célú számítógépek (de ma már sok célszámítógép is) rendelkezik operációs rendszerrel. A számítógép bekapcsolása után a BIOS loader programja betölti az operációs rendszer betöltő programját, amely aztán betölti, majd elindítja az operációs rendszert. Először az operációs rendszer fut tehát a számítógépen, és különféle szolgáltatásokat nyújt a felhasználó(k) és a programok felé. Ezek a szolgáltatások többek között:

- Kezelői felületet biztosít a felhasználóknak és a rendszer karbantartójának (system administrator) a programok elindításához, file-ok, ill. a rendszer karbantartásához.
- Egységes felületet biztosít a programok által használandó erőforrások (perifériák és memória) kezelésére. Ezen szolgáltatások egy részét a BIOS segítségével végzi. Mivel a számítógépeket lehetőség van különféle perifériákkal bővíteni, az ezekhez tartozó handlereket (periféria kezelő programok) is az operációs rendszer aktivizálja.
- Biztosítja a számítógép erőforrásainak hatékony kihasználását. Ez a szolgáltatás főként a több program egyidejű futtatására alkalmas, ún. multitasking operációs rendszereknél fontos.
- Több szinten is ellát védelmi funkciókat. Pl: Védi saját magát, vagyis egy jobb operációs rendzer megakadályozza, hogy a felhasználó hibás programja az operációs rendszerhez rendelt memória területhez hozzáférjen és esetleg működésképtelenné tegye. Megakadályozza, hogy ha egyszerre több program fut, egyik program a másikhoz tartozó memória területehez illetéktelenül hozzáférjen. Megakadályozza, hogy többfelhasználós rendszerben az arra jogosulatlan hozzáférjen adatokhoz.

Ma már a bonyolultabb feladatokat ellátó kisméretű célszámítógépek (pl. adatgyűjtők műszerek), egy része is tartalmaz operációs rendszert (mikro operációs rendszer), mely tk. valamely normál operációs rendszer kissé lebutított változata, vagy kimondottan ilyen célra (beágyazott rendszerekhez) létrehozott operációs rendszer. Bonyolultabb műszerekbe komplett PC-t (operációs rendszerrel) is beépítenek (ipari PC-k).

Javasolt szakirodalom:

[1] Budai Attila: Mikroszámítógép rendszerek (LSI Oktatóközpont, 1999)
[2] Csala-Csetényi-Tarlós: Informatika alapjai (Computer Books 2001)
[3] Knapp Gábor: Operációs rendszerek (LSI Oktatóközpont, 1998)
[4] Benesóczky Zoltán: Digitális tervezés funkcionális elemekkel és mikroprocesszorral (Műegyetemi kiadó, 1998)