Bevezetés a számítástechnikába (BSc,FSZ, on-line tananyag)

Tartalomjegyzék

1 Bevezetés az internetes távoktatással tanulók részére
2 Informatikai alapismeretek
2.1 Alapfogalmak
2.1.1 A Neumann-elvű számítógép
2.1.2 A rendszer fogalma
2.1.3 Az algoritmus és a program fogalma
2.1.4 Információ és adat
2.1.5 Információtechnológia, információs és adatfeldolgozó rendszer
2.1.6 Kódolás, kódrendszer, kódkészlet
2.1.7 A hardver, szoftver és a förmver fogalma
2.2 Az analóg és digitális technika
2.2.1 Analóg technika
2.2.2 Digitális technika
2.2.3 Jelátalakítás
2.2.3.1 Analóg–digitális átalakítás
2.2.3.2 Digitális–analóg átalakítás
2.2.4 Logikai áramkörök
2.2.4.1 Bináris logika
2.2.4.2 Boole-algebra
2.2.4.2.1 Boole-művelet
2.2.4.2.2 Igazságtáblázat
2.2.4.3 Kapuáramkörök, logikai áramkörök
2.3 Digitális adatábrázolás a számítógépekben
2.3.1 Számrendszerek
2.3.1.1 A tízes (decimális) számrendszer
2.3.1.2 A kettes (bináris) számrendszer
2.3.1.3 A tizenhatos (hexadecimális) számrendszer
2.3.1.4 Számrendszerek átalakítása
2.3.1.4.1 Decimális-bináris átalakítás
2.3.1.4.2 Bináris-decimális átalakítás
2.3.1.5 Műveletvégzés bináris számokkal
2.3.1.5.1 Bináris számok összeadása
2.3.1.5.2 Bináris számok kivonása
2.3.2 A számítógépes számábrázolás
2.3.2.1 Fixpontos számok
2.3.2.2 Binárisan kódolt decimális számok (BCD kódok)
2.3.2.3 Lebegőpontos számok
2.3.3 Karakterek kódolása
2.4 Ellenőrző kérdések
3 A mikroszámítógép
3.1 Bevezetés
3.2 A mikroszámítógép felépítése
3.2.1 A központi egység (CPU = Central Processor Unit)
3.2.2 Perifériák
3.3 A mikroszámítógép működése
3.3.1 Órajel és gépi ciklus
3.3.2 Tárak, tárolók (memória)
3.3.3 Memóriacímzés
3.3.4 A gépi utasítások végrehajtásának lépései
3.4 A mikroszámítógép szoftvere
3.5 Ellenőrző kérdések
4 Hardver alapismeretek
4.1 Mikroprocesszor
4.1.1 Bevezetés
4.1.2 A mikroprocesszor feladata
4.1.3 A mikroprocesszor funkcionális egységei
4.1.3.1 Regiszterek
4.1.3.1.1 Tipikus regiszterek
4.1.3.1.2 A regiszterek használata, az utasításvégrehajtás elemi lépései
4.1.3.2 Aritmetikai logikai egység (ALU)
4.1.3.3 Vezérlőegység (CU)
4.1.3.4 Állapotinformáció
4.1.3.4.1 A jelzőbit

4.1.3.4.2 Állapotregiszter
4.1.4 A processzor utasításkészlete, címzési eljárások
4.1.4.1 A gépi utasítás szerkezete
4.1.4.2 Címzési módszerek
4.1.4.2.1 Az abszolút címzés
4.1.4.2.2 Relatív címzés
4.1.4.2.3 Közvetlen adatcímzés
4.1.4.2.4 Veremcímzés (STACK-címzése)
4.1.4.2.4.1 Memóriaverem
4.1.4.2.4.2 Veremtár műveletek
4.1.4.2.4.3 A kaszkád verem
4.1.4.2.4.4 Mire használható a veremtár?
4.1.4.2.4.5 Egymásba illesztett – egymásba ágyazott – szubrutinok és a veremtár
4.1.4.2.5 Közvetett címzés
4.1.4.2.6 Indexelt címzés
4.2 A mikroprocesszor alapú rendszer
4.2.1 Memóriák csatlakoztatása a mikroprocesszorhoz
4.2.1.1 ROM (Read Only Memory) csak olvasható tár
4.2.1.2 RAM (Random Access Memory) közvetlen elérésű
4.2.2 Adatátvitel a mikroszámítógép és a hozzá csatlakoztatott perifériális
4.2.2.1 Interfész, I/O PORT
4.2.2.2 Az adatátvitel típusai
4.2.2.3 Programozott adatátvitel (PIO mód)
4.2.2.4 Programmegszakítással (interrupt) történő adatátvitel
4.2.2.4.1 A programmegszakítás okai
4.2.2.4.2 A megszakítás kiszolgálása
4.2.2.4.3 A megszakítások típusai
4.2.2.5 Adatátvitel közvetlen memóriahozzáféréssel (DMA)
4.2.2.5.1 DMA adatátviteli eljárások típusai
4.2.2.5.2 A DMA regiszterei
4.2.2.5.3 A DMA művelet végrehajtása
4.2.2.5.4 DMA vezérlőfunkciók
4.2.2.6 Külső buszrendszer (system bus)
4.2.2.7 Soros adatátvitel
4.2.2.7.1 A soros és párhuzamos adatátvitel
4.2.2.7.2 A soros adatátvitel bitjeinek felismerése
4.2.2.7.3 Az adó és vevő szinkronizálása
4.2.2.7.4 Kommunikációs protokoll
4.2.2.7.5 Soros adatátvitel távbeszélő vonalakon
4.2.2.7.6 Szinkron és aszinkron soros adatátvitel
4.3 Ellenőrző kérdések
5 Szoftver alapismeretek
5.1 Bevezető
5.2 Az operációs rendszer
5.2.1 Az operációs rendszer felépítése
5.2.1.1 A rendszermag alsó (hardver) felülete
5.2.1.2 A rendszermag felső (szoftver) felülete
5.2.1.3 A rendszermag „magja” (röviden)
5.2.2 Állománykezelés
5.2.2.1 Elnevezések, hivatkozások
5.2.2.2 Az állományok egyéb jellemzői
5.2.2.3 Katalógusok (directory)
5.2.2.4 Kötetek (volume)
5.2.2.5 Hivatkozások
5.2.2.6 Közvetett elérés, keresési útvonal
5.2.2.7 Fájlok fizikai elhelyezése
5.2.3 A felhasználói felület
5.2.3.1 Karakteres felhasználói felület, a parancsértelmező
5.2.3.2 Grafikus felhasználói felületek
5.2.3.3 Segédprogramok, alrendszerek
5.3 Programkészítés
5.3.1 A forráskód elkészítése
5.3.2 Fordítás
5.3.3 Szerkesztés
5.3.4 Betöltés, dinamikus könyvtárak
5.3.4.1 A program összeállítás teljes folyamata
5.4 Röviden a hálózatokról
5.4.1 Mire jó a számítógép-hálózat?
5.4.2 A számítógépek közötti kommunikáció
5.4.2.1 Az összeköttetés
5.4.2.2 Közös nyelv (protokoll)

5.4.2.3 Címek
5.4.2.3.1 Fizikai címek
5.4.2.3.2 IP címek
5.4.2.3.3 Levelezési címek
5.4.3 Hálózattípusok
5.4.4 Munkamegosztás a hálózaton
5.5 Ellenőrző kérdések
Melléklet A: Fogalomtár

Szerző: Dr. Kovács Magda, Ágoston György, Budai Attila, Knapp Gábor

1 Bevezetés az internetes távoktatással tanulók részére

Az on-line tananyag Kovács Magda - Knapp Gábor - Ágoston György - Budai Attila: Bevezetés a számítástechnikába, LSI Oktatóközpont, Budapest, 2001 tankönyv alapján készült.

Javasolt képernyőbeállítás: 1024 x 768, teljes képernyő (F11), közepes betűméret (Nézet/Szövegméret/Közepes).

Ha számítástechnikával kezdünk foglalkozni, az első kérdés amit fel kell tennünk magunknak az, hogy mi az oka a számítógépek rohamos elterjedésének az élet csaknem minden területén. Mit tudnak a számítógépek, amivel az elmúlt évtizedekben nélkülözhetetlenné váltak a számunkra? Valóban képesek-e a számítógépek az emberhez hasonlóan gondolkodni, vagy csak nagyon gyors működésű számológépek?

A tananyagot a multimédiás oktatóprogram tartalomjegyzékében található sorrendben, fejezetenként, ezen belül tananyagrészenként kell feldolgozni kb. 15-25 perces egységekben.

Javasoljuk, hogy minden hallgató készítsen egyéni tanulási tervet, figyelembe véve a tantárgyra rendelkezésre álló időt az időrendi táblában.

Egy tananyagrész feldolgozása:

- Elolvassuk a tananyag megfelelő részét, megnézzük a képeket, animációkat.
- Az esetleg elfelejtett fogalmak definícióit visszakeressük a fogalomtárban.
- A tanultakat átgondoljuk, meghatározzuk a lényegi részeket, megkeressük az összefüggéseket, így pl.:
  - A tárgyaltak hogyan valósulnak meg az otthoni számítógépemben?
  - Mit tanultunk az adott témakörről a "Bevezetés a számítástechnikába" tantárgyban?
  - A tanult rész a már megismert tananyag mely részeivel hozható kapcsolatba?
- Átismételjük a tananyagrész lényegi (dőlt, barna színnel jelölt) elemeit.
- A legfontosabb törvényszerűségeket, fogalmak definícióit megfogalmazzuk önállóan, majd az ILIAS fogalomtárban ellenőrizzük, hogy
  - nem hagytunk-e ki lényeges elemet;
  - pontos-e a szakfogalmak használata?
- Megkeressük a tanult részre vonatkozó ellenőrző (vizsga) kérdéseket, megválaszoljuk, és a válaszunkat ellenőrizzük a tananyagban.

Egy fejezet tanulásának befejezése

A tananyag egyes, nehezebb részeinek elsajátítását, megértését fejezetenként **gyakorlati feladatok** segítik. Ezeket először próbáljuk meg segítség nélkül megoldani. Ha ez nem sikerül (vagy bizonytalanok vagyunk a megoldás helyességében), ismételjük át a megfelelő tananyagrészeket, és megint kíséreljük meg a megoldást. A feladatok megoldását az **internetes tutor** is segíti, illetve ellenőrzi. A beküldendő gyakorlatokat a tutorok pontozzák és beszámítják a vizsgaeredményekbe.

A tananyag egyes fejezeteinek befejezését követően oldjuk meg az **önellenőrző tesztfeladatok**at. Ha eredményünk nem megfelelő, nincs értelme a következő fejezetek feldolgozását megkezdeni, mivel az ismeretek összefüggései és egymásra épültsége miatt a tananyag egy idő után érthetetlené válna.

A tantárgy tanulásának befejezése

A tananyag összes fejezetének a feldolgozását követően oldjuk meg a vizsgára való felkészülést önellenőrző tesztet. Ennek eredményétől függően szükségessé válhat az ismétlés illetve a vizsgaelőkészítő tantermi konzultáció.

A tananyag elsajátítását segíti a **vizsgaelőkészítő konzultációk**on való részvétel, mert az ott elhangzottak áttekintést adnak a tananyag összefüggéseiről, segítséget nyújtanak a bonyolultabb részek megértéséhez. Az ezeken való személyes megjelenést különösen akkor ajánljuk, ha a hallgató az önellenőrzés során nem éri el az 70%-os eredményt.

A sorszámozott fejezet-, alfejezetcímek a böngészőablak tetején láthatók. A legalsó alfejezetek címeit kapták a lapok címként, a címek végén jelezve, hány lapból áll az adott fejezet, illetve azok közül hányadikat látjuk.

A törzsanyag fekete betűs szöveg.

A számonkérendő tananyag (amelyre ellenőrző kérdés vonatkozik) dőlt, barna színű betűkkel van kiemelve.

Az oktatóprogramban az ugrási címek (linkek) aktív állapotban (feléjük húzva az egeret) zöld színűre változnak, ha az ILIAS-on belüli címre mutatnak, és pirosra, ha azon kívülire.

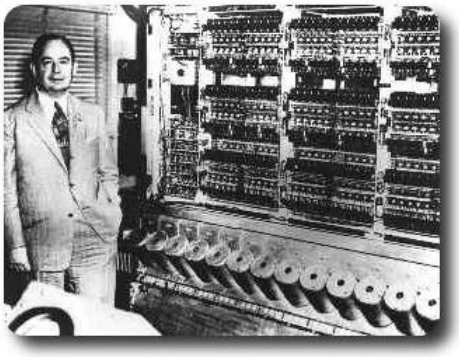
A kiegészítő anyagrészeket szürke háttérszínen apró, fekete betűs szövegrészek jelölik.

2 Informatikai alapismeretek

2.1 Alapfogalmak

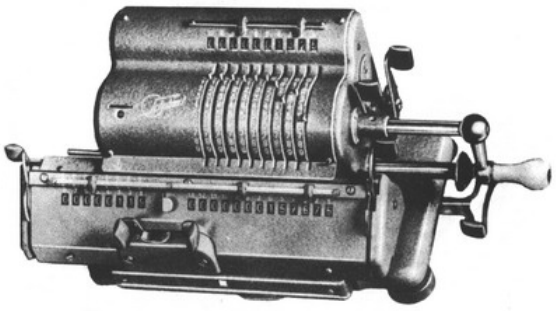
2.1.1 A Neumann-elvű számítógép

Ha számítástechnikával kezdünk foglalkozni, az első kérdés amit fel kell tennünk magunknak az, hogy mi az oka a számítógépek rohamos elterjedésének az élet csaknem minden területén. Mit tudnak a számítógépek, amivel az elmúlt évtizedekben nélkülözhetetlenné váltak a számunkra?



Neumann János és az ENIAC 1946

Az ENIAC 17.468 elektroncsövet tartalmazott, több mint 100 kW elektromos energiát fogyasztott, és 450 m^2 helyet foglalt el



Mechanikus számológép az 1960-as évekből

Valóban képesek-e a számítógépek az emberhez hasonlóan gondolkodni, vagy csak nagyon gyors működésű számológépek?

E kérdés megválaszolásához vizsgáljuk meg, hogy a számítógép működése mennyiben tér el elődjeitől, a különböző elveken működő számológépektől. Nézzük meg például, hogyan számíthatjuk ki a kör területét e két eszközzel.

$$T = R^2 \bullet \pi$$

Számológép

1. lépés A feladat megfogalmazása
2. lépés: R bebillentyűzése
3. lépés: Szorzás műveleti jelének beadása
4. lépés: R bebillentyűzése
5. lépés: Szorzás műveleti jelének beadása
6. lépés: π bebillentyűzése
7. lépés: Az eredmény kijelzése

Látható, hogy a **számológép** esetében a műveleti utasításokat és adatokat kívülről, lépésenként (és emberi sebességgel) közöljük a géppel. A végrehajtást követően a műveleti sorrendet a számológép nem tárolja.

Számítógép

1. lépés: Az algoritmus megfogalmazása a gép nyelvén és mentése a tárolóba:
   - R beolvasása a számítógépbe
   - π beolvasása a számítógépbe
   - $T = R \bullet R \bullet \pi$

- az eredmény kiíratása

2. lépés: A program futtatása, R bebillentyűzése

3. lépés: Az eredmény kijelzése

A számítógép esetében a műveletsorozatot és az [could not resolve link target: il_0_git_37] okat is a gép memóriájában tároljuk, a műveletek végrehajtása automatikusan, emberi beavatkozás nélkül történik.

Az animáció elindításához nyomja meg az 1. lépés gombot.

Látható, hogy a **számológép** esetében a műveleti utasításokat és adatokat kívülről, lépésenként (és emberi sebességgel) közöljük a géppel. A végrehajtást követően a műveleti sorrendet a számológép nem tárolja.

A számítógépesetében a műveletsorozatot és az adatokat is a gép memóriájában tároljuk, a műveletek végrehajtása automatikusan, emberi beavatkozás nélkül történik. Példánkból is kitűnik, hogy a számítógép nem képes gondolkodni, viszont azokat a műveletsorozatokat, más néven programokat, amelyeket az ember megfogalmaz számára (és amilyen műveletek végrehajtására felkészítették) képes tárolni és önállóan végrehajtani.

Ennyi előkészítés után megfogalmazhatjuk, hogy mit értünk számítógépen (computer, computer system). Számítógépnek nevezzük azokat az eszközöket, amelyek adatok és az ember által megfogalmazott műveletsorozatok (programok) tárolására és automatikus végrehajtására képesek. A számítógép fogalmába beleértjük a működéséhez szükséges programok összességét is. A számítástechnika fejlődésének kiindulópontját jelentő Neumann elvek a számítógépek legfontosabb jellemzőit rögzítik.

Érdemes megemlíteni, hogy Neumann János az emberi gondolkodás analógiái alapján alakította ki javaslatait. Ezt érzékeltetik a következő ábrák.

Az emberi agyműködés

Az **ember** gondolkodó élőlény. Az élővilág legfejlettebb tagja, a mai rendszerezés szerint a főemlősök rendjén belül az emberfélék családjának jelenleg élő tagja. Az embert minden más élőlénytől megkülönbözteti az ún. második jelzőrendszer, a nyelv és a gondolkodás, a tagolt, értelmes beszéd. Az ember tudatos, termelő tevékenységet folytató, kultúrateremtő társadalmi lény.

Az **agy** mintákat tárol és hív elő. Minden érzékszervi benyomás - legyen az látvány, hang, érintés vagy illat - mintasorozatokká alakul.

EMBERI AGY

Érzékszervek	Irányítás	Számítás	Eredmény közlése
Üzeneteket fognak fel a külvilágból és azt továbbítják az agyba.	Egy rendszer működésének meghatározott cél elérése, érdekében való befolyásolása. Az irányítás a folyó munkaműveletek mérése, elemzése és a tervekkel való összehasonlítása. Az irányítási rendszer lehet kézi vagy automatikus. Technikai értelemben két fajtáját különböztetjük meg, a szabályozási és a vezérlésit.	A valós számok közötti alapműveletekkel (összeadás, kivonás, szorzás, osztás), a hatványozással és a gyökvonással, ill. ezek felhasználásával foglalkozik.	Emberek közti kapcsolat, érintkezés. Legtöbbször beszéddel és írással történik.
Látás			
Hallás			
Szaglás			
Tapintás			
Ízlelés			

Emlékezet

Amikor új dolgokkal, helyzetekkel találkozunk, az agy megvizsgálja a korábban eltárolt tapasztalatokat, és megjósolja, mi fog történni. Számára a múlt nem más, mint tárolt minták sorozata.

A számítógép funkcionális felépítése

A számítógép funkcionális egységei:

- a feldolgozást a központi feldolgozó egység (CPU = mikroprocesszor) végzi, mely 2 részből áll:
  - vezérlőegység (CU = Control Unit)): értelmezi (dekódolja) az utasításokat, majd előállítja a végrehajtáshoz szükséges vezérlőjelet;
  - aritmetikai és logikai egység (ALU = Aritmetic Logic Unit): elvégzi a matematikai és a logikai műveleteket, ez a processzor műveletvégző egysége;
- operatív tár (memória): adat- és műveletsorozat-tároró;
- be- és kiviteli egységek.

KÖZPONTI FELDOLGOZÓ EGYSÉG (CPU = mikroprocesszor)

A processzor olyan funkcionális egység, amely utasításokat értelmez és hajt végre.

Gyakran azonosnak tekintik a központi egységgel.

Processzor

ADAT- ÉS MŰVELETSOROZAT-TÁROLÓ

A programból közvetlenül címezhető, írható-olvasható (RAM) és csak olvasható (ROM) részekből álló tár. Tartalmazza a végrehajtás alatt álló programrészeket, az ezekhez szükséges adatokat és az operációs rendszer aktív részeit.

RAM

ROM

HÁTTÉRTÁROLÓK

A számítógéphez csatlakoztatott, adatok hosszabb ideig történő tárolására alkalmas berendezés (mágneslemez-tár, mágnesszalag-tár).

Rendszerint azokat a programrészeket, adatokat és az operációs rendszer olyan részeit tárolja, amelyeknek a pillanatnyi feldolgozásban nincs szerepük. A háttértár kapacitása jóval nagyobb, mint a főtáré, de a hozzáférési idő is nagyobb.

Háttértárolók például az alábbiak:

Flopi

Merevlemez

CD

Streamer

ADATBEVITELI EGYSÉG

A beviteli egységek (input eszközök) segítségével visszük be a számítógépbe mindazokat az információkat, amelyekre a feldolgozáshoz szükség van, tehát a feldolgozandó adatokat és programokat. Ezeknek az eszközöknek nemcsak az adatmozgatás a feladata, hanem az is, hogy az adatokat az ember által értelmezhető formáról átalakítják a gép által értelmezhető formára.

Billentyűzet és egér

Vonalkódolvasó

Szkenner

Mikrofon

ADATKIVITELI EGYSÉG

A kiviteli egységek (output berendezések) a nevükből következően az adatok számítógépből történő kihozatalát, megjelenítését szolgálják. Az adatmozgatás mellett az adatokat át is alakítják a gépben tárolt bináris formáról az ember által értelmezhető analóg alakba.

Monitor

Nyomtató

Plotter

Hangszóró

A számítógépek felépítésének és működésének Neumann János által megfogalmazott elvei a következők:

- a tárolt program elve,
- a címezhetőség elve,
- önálló adat be-/kiviteli egység, vezérlő és művelet végrehajtó egység,
- teljesen elektronikus gép, amely a kettes (bináris) számrendszer alkalmazásával működik,
- a soros utasítás-végrehajtás elve.

A tárolt program elve azt mondja ki, hogy az elvégzendő műveleteket először be kell vinni a számítógép tárolójába, azután a gép vezérlését át kell adni ennek a műveletsornak, azaz a programnak. Ezt követően már a számítógép tárolójában lévő program irányítja a gépet:

- beolvassa az adatokat,
- elvégzi a műveleteket, ehhez tárolja a részeredményeket
- és kijelzi a végeredményt.

A címezhetőség elve azt jelenti, hogy az utasítások és az adatok ugyanolyan módon megcímzett (címmel, sorszámmal ellátott) memóriarekeszekben helyezkednek el, azaz nincsen külön hely az utasításoknak és az adatoknak.

A tárolt programokkal működő számítógép elvi felépítése funkcionálisan megfelel a ma használatos számítógépeknek:

- az adatok bevitelére és megjelenítésére önálló egységek szolgálnak;
- a gép tartalmaz egy vezérlőegységet, amely a tárolt program alapján irányítja a teljes gép működését;
- a gépnek része egy önálló műveletvégző egység, amely képes az összes fontosabb aritmetikai és matematikai logikai művelet végrehajtására;
- a számítógép rendelkezik egy belső tárolóval, ahol az aktuálisan szükséges programok és adatok tárolhatók.

A másik jelentős újítást a korábbi számológépek többségével szemben a bináris (kettes számrendszer alapján történő) adat (szám) és programutasítás tárolás és műveletvégzés jelentette.

A kettes számrendszer két számjegyéhez (0-hoz ill. 1-hez) ugyanis jól hozzárendelhetők két állapotú jelenségek (pl. van áram, vagy nincs áram egy ponton; mágnesezett vagy nem mágnesezett egy pont), az aritmetikai műveletek visszavezethetők a kétértékű (igaz, hamis) matematikai logika műveleteire, amelyek viszont logikai áramkörökkel modellezhetők.

Neumann János harmadik javaslata a számítógép soros utasítás végrehajtása volt. Ez azt jelenti, hogy a számítógép az utasításokat szigorúan egymás után, sorban hajtja végre (control flow). Ettől eltérés logikailag csak egy állapotjelző tartalmának vizsgálata alapján lehetséges, de ez is csak azt jelenti, hogy az utasítások végrehajtása a műveleti sor egy másik pontján folytatódik. „Természetesen a soros adatfeldolgozás lassúbb, mint a párhuzamos, de jóval kevesebb eszköz kell hozzá, és az eljárás lassúságát az igen gyors elektronika kiegyenlíti” – vélekedett Neumann János.

A soros utasítás-végrehajtásra példaként nézzük meg, milyen logikai műveletsorozattal tudjuk kiszámítani egy $ax^2 + bx + c = 0$ másodfokú egyenlet gyökeit.

Az ismert $x_{12} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ képlet alapján a következő műveletsorozatot kell alkalmaznunk:

1. START - program indítása
2. "a", "b", "c" beolvasása
3. Ha „a” értéke 0, kiírás: „Nullával nem lehet osztani”
4. Folytasd a 12. lépéssel
5. Ha „a” értéke nem 0: $d = b^2 – 4ac$ kiszámítása
6. Ha „d” értéke kisebb, mint 0, kiírás: „Nincs valós gyök”
7. Folytasd a 12. lépéssel
8. Ha „d” értéke nagyobb vagy egyenlő 0, folytasd a 9. lépéssel
9. x_1 kiszámítása
10. x_2 kiszámítása
11. x_1, x_2 kiírása
12. STOP - kilépés

Az oldalon megtekintheti a másodfokú egyenlet (algoritmus) folyamatábrájának az animációját.

A nagyító segítségével nagyítsa fel a programot.

A 1. gomb jelentése: a=0 esetén a folyamat. $ax^2+ 8x + 3 = 0$

A 2. gomb b^2-4ac < 0 esetén a folyamat. $4x^2+ 4x + 3 = 0$

A "harmadik" gomb tartalmazza a kivételek nélküli másodfokú egyenletet $4x^2+ 8x + 3 = 0$

Végül térjünk vissza fejezetünk kiinduló kérdésére, hogy mivel magyarázható a számítógépek gyors elterjedése, miért alkalmazzák ezeket az eszközöket számtalan szakterületen?

- Ha egyszer a számítógép számára az ember megfogalmazott egy műveletsorozatot, akkor ezt a gép igen gyorsan képes végrehajtani, műveletvégző képessége az emberének sokszorosa. A tárolt programozás miatt az emberi adatközlés nem lassítja le a számítógép működését.
- Ha egy feladatot lefordítottunk a számítógép nyelvére, akkor az ennek megfelelő műveletsorozatot megtestesítő programot a számítógép bármeddig tárolni képes; ezt bármikor újra végre lehet hajtani eltérő kiinduló adatokkal is.

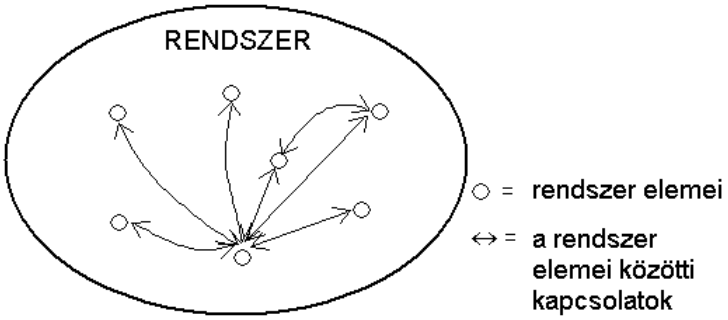
Így például, ha a másodfokú egyenletet megoldó előző programunkat tároljuk a számítógépen, akkor ezt felhasználva bármilyen másodfokú egyenlet megoldására alkalmassá tettük a gépet. (Ehhez csak a különböző „a”, „b” és „c” értékeket kell közölnünk a számítógéppel.) Arra is fontos felhívni a figyelmet, hogy ezt a másodfokú egyenletmegoldó programot bárkinek a számítógépére át lehet másolni, így a másodfokú egyenlet megoldását az összes számítógépet használó ember számára lehetővé tettük.

A számítógépeknek tehát az előzőekben részletezett tulajdonságai lehetővé teszik, hogy a megoldott feladatoknak megfelelő programokban megtestesülő szellemi értékek évről-évre felhalmozódjanak és így a számítógépek egyre több területen felhasználhatóvá válnak.

2.1.2 A rendszer fogalma

Rendszernek nevezzük a rendszer részeit alkotó elemek és ezek kapcsolatainak olyan együttesét, amelyek meghatározott ismérvek (vizsgálati célok) szempontjából összetartoznak.

A **rendszer elemei** alatt a rendszernek azokat az alkotóit értjük, melyeket a rendszer vizsgálata során már további részekre nem bontunk.



A rendszer szimbolikus ábrázolása

Látható, hogy definíciónk igen általános. Ilyen értelmezésben rendszernek tekinthetjük például a társadalmat, egy konkrét embert, egy gyárat vagy a UNIX operációs rendszert egyaránt.

A definícióval összefüggésben a következőkre hívjuk fel a figyelmet:

- Ha egy rendszert vizsgálunk, akkor mindig a valóság egy modelljét hozzuk létre. Ezért az, hogy mit tekintünk egy rendszernek, az a vizsgálat céljától függő elvonatkoztatás eredménye. Így például, ha egy számítógépes felhasználói programrendszert vizsgálunk, akkor ez a rendszer állhat a programegységekből és ezek funkcióiból, mint elemekből (például, ha a vizsgálat tárgya a rendszer felhasználói szolgáltatásai), de állhat programutasításokból is, mint elemekből (pl. ha hibakeresés a vizsgálat tárgya).
- Hangsúlyozni kell, hogy a rendszer nem egyszerűen elemeknek a halmaza, mindig beleértjük a rendszer fogalmába elemkapcsolatainak összességét is. Az elemkapcsolatok közül csak azokat tekintjük a rendszer részének, amelyek az adott vizsgálati cél szempontjából lényegesek. Így például, ha egy munkaszervezet működési modelljét vizsgáljuk, mint rendszert, amelynek elemei az adott szervezethez tartozó emberek, nem elemkapcsolat az a viszony, hogy a dolgozók közül ki született azonos városban.

Önszerveződő és adaptív egy rendszer, ha képes arra, hogy elemkapcsolatait külső beavatkozás nélkül megváltoztassa és ezáltal a környezetéhez alkalmazkodjon.

Önszabályzó egy rendszer, ha külső beavatkozás nélkül képes belső folyamatait irányítani.

Ilyen rendszerekre jó példa az emberi szervezet, egy gazdasági vállalkozás vagy egy automata gép. Az emberi szervezet adaptivitását mutatja például a meleg hatására való izzadás, az önszabályozást pedig a véráramlásunk folyamatos fenntartása.

A rendszer állapotába való külső beavatkozás típusa szerint megkülönböztetünk:

- irányítást, amikor a külső beavatkozás a rendszer állapotának változását céltudatosan befolyásolja,
- szervezést, amikor a külső beavatkozás a rendszer felépítését (struktúráját) változtatja meg.

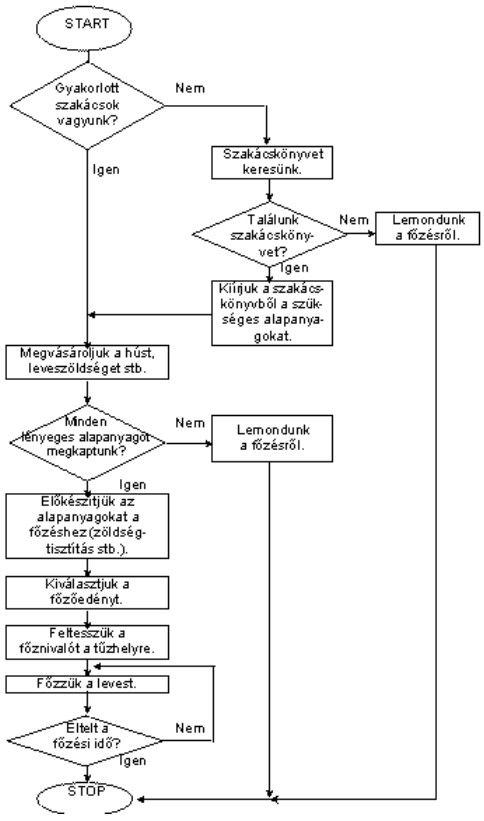
A külső beavatkozást közvetítő információt nevezzük input-nak, a rendszer erre adott reakcióját megtestesítő információt pedig output-nak.

2.1.3 Az algoritmus és a program fogalma

Ha egy feladatot kell megoldanunk, akkor ezt általában több részfeladatra bontjuk fel, amelyekben különböző műveleteket végzünk el, feltételeket értékelünk ki, és így jutunk el célunkhoz.

Példaként nézzük meg, ha húslevest akarunk főzni, akkor milyen résztevékenységeken keresztül jutunk el az eredményhez, és ennek során milyen döntéseket kell hoznunk.

Ezt egy úgynevezett folyamatábrán fogjuk szemléltetni, ahol téglalappal jelöljük a részműveleteket és fekvő rombusszal a döntési pontokat.



A feladat elemzése alapján határozhatjuk meg azokat a lépéseket és döntési pontokat, amelyeken keresztül végül elérhetjük a kívánt eredményt. Ezt a lépéssorozatot – beleértve a döntési helyzeteket is – szokták algoritmusnak nevezni. Ennek megfelelően tehát a pontos definíció a következő:

Algoritmusnak nevezzük egy feladat megoldását eredményező véges számú lépésben véget érő, egyértelmű szabályokkal megfogalmazható műveletsorozatot. Definíciónk értelmében tehát egy jól működő számítógépes program utasítássorozata és a π 5 tizedes pontosságú kiszámítására való matematikai eljárás is algoritmus.

A számítógépek lényegében mindig algoritmusokat végrehajtva működnek. Ebben az esetben azonban az alkalmazható elemi lépéseket a gép felépítése határozza meg.

Ha egy algoritmust a számítógép által értelmezhető és végrehajtható lépésekből építünk fel, akkor ezt programnak nevezzük, az elemi lépéseket pedig utasításnak. A program tehát, a számítógép számára értelmezhető és végrehajtható utasítások sorozata. Ezért a programkészítésnél olyan elemi lépésekre kell bontanunk a feladatot, amelyek a számítógép nyelve által megengedett utasításokból állnak (lásd 5. fejezetben a számítógép utasításkészletét).

2.1.4 Információ és adat

Ha a környezetünkben zajló eseményeket befolyásolni akarjuk, azokra reagálni kívánunk, meg kell ismernünk azokat. Ezek az ismeretek hírek vagy közlemények formájában jutnak el hozzánk. A közleményeket mindig valamilyen hordozó közeg (fény, hang, szag stb.) által érzékeljük. Ha a közleménynek van olyan tartalma, amelynek révén olyan ismereteket szerzünk, amivel korábban nem rendelkeztünk, akkor a közlemény számunkra **információt** hordoz.

A közlemények meghatározott ideig rögzített formában is létezhetnek (például az újságban kinyomtatva, vagy a számítógép memóriájában tárolva), a hétköznapi életben ekkor beszélünk **adatokról**.

Hangsúlyozni kell, hogy az adat csak a közlemény formai oldalát jelenti, információt az ember számára csak értelmezésével jelenthet. Így például ugyanazon hangfrekvenciákat tartalmazó csengőhang az iskolában jelentheti az óra végét, a tűzoltóságnál pedig a tűzriadót.

Az elkövetkezendőkben az adat és az információ fogalmának pontosabb meghatározásával fogunk foglalkozni és bemutatjuk e két fogalom viszonyát.

Az IBM adatfeldolgozási szótára szerint az adat tények, fogalmak, eligazítások olyan formai megjelenése (képe), amely alkalmas az emberi vagy az automatikus eszközök által történő kommunikációra, értelmezésre vagy feldolgozásra.

Az ISO szabványa szerint az információ az adatnak tulajdonított jelentés. Az információt úgy is szokták definiálni, hogy olyan tény, amely a befogadó ember számára új ismeretet tartalmaz, és ezáltal bizonytalanságunkat csökkenti.

Ezek szerint, pl. Kovács Béla APEH-nyilvántartásban tárolt vezeték- és utóneve adat, ugyanez Kovács Béla számára a személyi igazolványában szintén adat, míg az őt igazoltató rendőr számára (ha korábban nem ismerte Kovács Bélát) egyúttal információ is.

Az információ definíciójával összefüggésben lényeges hangsúlyozni, hogy az információ mindig kötődik az információt befogadó emberhez, az adat viszont személytelen, objektív ismeret. Ezt emeli ki a következő definíció: az adat értelmezhető (észlelhető, érzékelhető, felfogható és megérthető) ismeret, az információ pedig új ismeretként értelmezett adat.

Tehát például, ha nem tudunk kínaiul, akkor egy kínai könyvben szereplő mondatok számunkra nem jelentenek információt, mert nem vagyunk képesek értelmezni azokat. Ugyanakkor ez a könyv természetesen adatokat tartalmaz, mert kínaiul tudó emberek számára értelmezhető.

Az információ mérésére a matematikában (információelmélet) mérőszámot is kidolgoztak. Ennek megértéséhez vegyük példának azt az eseményt, hogy az utcán találkozunk egy emberrel és megkérdezzük az utónevét. Ha véletlenszerűen választunk, annak valószínűsége, hogy partnerünk a JÁNOS vagy az UBUL nevet fogja válaszolni, nyilván a nevek gyakoriságától függ. E nevek előfordulásának valószínűségét a következőképpen számíthatjuk ki:

$$p_J = \frac{\text{JÁNOS nevet viselők száma}}{\text{összes ember száma}} \qquad p_U = \frac{\text{UBUL nevet viselők száma}}{\text{összes ember száma}}$$

Ha valakiről csak azt tudjuk, hogy JÁNOS-nak hívják, nehezebben tudjuk megkeresni, beazonosítani az összes ember között, azaz ez kevesebb információt jelent számunkra. Ennek az az oka, hogy a $p_J > p_U$. Azaz egy esemény egyedi információtartalmának nagysága (jelöljük I-vel) fordítottan arányos az adott esemény előfordulásának valószínűségével: $I_J < I_U$. Ezért logikusnak tűnik, hogy az

I_J arányos $\frac{1}{p_J}$ $\qquad$ I_U arányos $\frac{1}{p_U}$ szabályt állítsuk fel.

E gondolatmenetet követve az információelmélet kidolgozójának, Shannon-nak a javaslata szerint egy esemény bekövetkezésének információtartalmát a

$$-\log_2 p = \log_2 \frac{1}{p}$$

mérőszámmal határozhatjuk meg. (Itt a 2-es alapú logaritmus csak a „skálabeosztást” befolyásolja.)

Végül ismételten érdemes arra felhívni a figyelmet, hogy az információ az értelmezőhöz, tehát az emberhez kötődő fogalom. Az adatokat feldolgozó eszköz, például a számítógép számára teljesen közömbös az adatok jelentéstartalma, vagyis az információ. Itt csak az a lényeges, hogy a számítógép a program formájában testet öltött algoritmust a szabályoknak megfelelően végrehajtsa. Tehát a számítógép mindig adatokkal és nem információkkal dolgozik.

2.1.5 Információtechnológia, információs és adatfeldolgozó rendszer

Az információtechnológia magába foglalja mindazon módszereket és eszközöket, amelyek az információ előállítását, feldolgozását és továbbítását szolgálják.

Ebben az értelemben információtechnológiai eszközöknek minősülnek:

- a hírközlő, kommunikációs és média eszközök,
- a számítástechnikai eszközök,
- az irodatechnikai eszközök (szervező eszközök, mikrofilm, másolók, iratmegsemmisítők stb.).

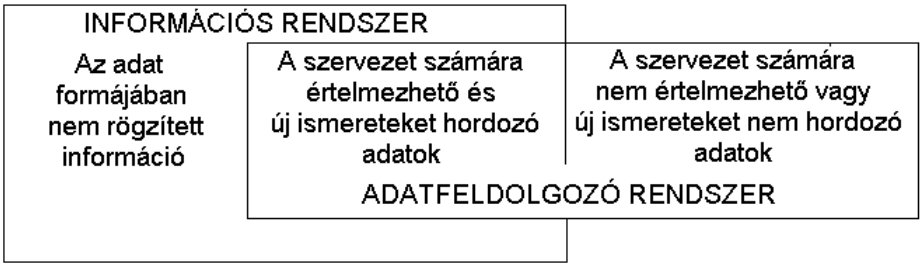
Az információtechnológia fogalmának megjelenése híven kifejezi azt, hogy a fenti három eszközcsoport vonatkozásában az utóbbi években határozott integrációs tendencia észlelhető.

A telefonközpontok ma már lényegében számítógépek, egy számítógépes rendszernek is szerves része a távadat-feldolgozó hálózat stb.

Az információs rendszer mindazon módszerek, eljárások, folyamatok és eszközök szervezett együttese, amellyel a szervezet tevékenységéhez információt állít elő, befogad, tárol, feldolgoz és továbbít.

Az adatfeldolgozó rendszer mindazon módszerek, eljárások és eszközök szervezett együttese, amellyel a szervezet adatot fogad, rögzít, feldolgoz, tárol, továbbít és megsemmisít.

Ha feltételezzük, hogy a szervezet céljaival és feladataival, valamint humán erőforrásaival teljes mértékben összhangban van a kialakított adatfeldolgozó rendszer (azaz az adatok mindig értelmezhetők és információt hordoznak), akkor az információs rendszernek mindig része az adatfeldolgozó rendszer. Ez azonban a valóságban legtöbbször nem így van, ezt szemlélteti a következő ábra:



Az adatfeldolgozó és információs rendszer viszonya

Vélhetően sokan tudnának gyakorlati példákat felsorolni olyan számítógépes rendszerekre, amelyek esetenként senki által nem hasznosítható adatokat állítanak elő, vagy a szolgáltatott adatokat a kellően nem kiképzett felhasználók nem tudják értelmezni.

A definícióval kapcsolatban hangsúlyozni kell, hogy minden szervezetnek létezik információs és adatfeldolgozó rendszere, függetlenül attól, hogy a szervezet alkalmaz-e információtechnológiai eszközöket, vagy feladatait hagyományos nyilvántartások alapján végzi.

Most már eljutottunk az informatika fogalmának meghatározásához, bár meg kell jegyezni, hogy e téren még sok a terminológiai vita.

Informatikának nevezzük az információs rendszerek fejlesztésének, működtetésének, hasznosításának törvényszerűségével foglalkozó tudományágat (szakmát).

Sokan az informatikát a számítástechnikával azonosítják. Ez több szempontból is hiba:

- az információs rendszert leszűkítik az azt kiszolgáló eszközre, a számítógépes rendszerre,
- mint láttuk, a számítástechnika az informatika eszközoldalának, az információtechnológiának csak egy részterülete.

Ez a helytelen szemlélet Magyarországon az elmúlt években igen komoly károkat is okozott. A rendszerfejlesztésben az eszközoldalra koncentrálnak, és ennek során sokszor elfelejtették, hogy a szervezeti céloknak megfelelően olyan információrendszert kell kialakítani, amelyet a szervezet képes befogadni. (A szervezet informatikai fogadóképessége a felhasználók képzettségétől kezdve az informatikusok szakértelmén keresztül, a vezetők szemléletéig számtalan tényezőtől függ.) Emiatt sok, például fejlettebb országokban bevált számítástechnikai eszköz hatékony alkalmazása is meghiúsult.

2.1.6 Kódolás, kódrendszer, kódkészlet

Az adatok megjelenési formája nagyon sokféle lehet, de a leggyakoribb az írott vagy nyomtatott szöveg; ennek egységelemét karakternek nevezzük.

A karakterek lehetnek

- kis és nagybetűk (A, B, C … a, b, c … y, z),
- számjegyek (0, 1, 2 … 9),
- különleges jelek (+, -, ? … stb.).

A csak számjegyekből álló karaktersorozatokat numerikusnak, a csak betűkből állókat alfabetikusnak, a fenti három típust vegyesen tartalmazó karaktersorozatokat pedigalfanumerikusnak nevezzük.

Például a „1233” karakterből álló adat numerikus, a „1+3=C6” karakterekből álló pedig alfanumerikus adat.

Az adatokat a különböző adathordozókon olyan formában célszerű rögzíteni, hogy helyigényük ne legyen túlzott mértékű, és minél kevesebb hibát kövessünk el rögzítéskor. Ezt többek között az adatok kódolásával érhetjük el.

Ezen túlmenően például az adatok számítógépes feldolgozásához az ember által megszokott formátumú tízes számrendszerű számokat és karaktersorozatokat is át kell alakítani a számítógép működésének megfelelő 0-ból és 1-ből álló jelsorozattá.

Az egyes hírekben és közleményekben lévő adatok formai átalakítását kódolásnak nevezzük. Ennél általában követelmény, hogy az adatok kódolás előtti és utáni formája között kölcsönösen egyértelmű megfeleltetés legyen.

A kódolás tehát áttérést jelent egy jelkészlet és ezzel összefüggő szabályrendszer használatáról egy másik jelkészletre és szabályrendszerre.

Erre mutat példát a következő ábra:

Fekete	←→	0
Kék	←→	1
Zöld	←→	2
Világoskék	←→	3
Piros	←→	4
Lila	←→	5
Sárga	←→	6
Fehér	←→	7

a)

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

b)

Az ábra b) részében bemutatott bináris kódolás esetén, ha a közlemény a 1910 karaktersorozat, akkor ennek kódolás utáni megfelelője:

0 0 0 1	1 0 0 1	0 0 0 1	0 0 0 0
1	9	1	0

a) Színek kódolása;
b) A tizedes számrendszer számjegyeinek bináris (kettes számrendszerű) kódolása

A kódolás során használt jelkészletet és formai szabályrendszert együttesen kódrendszernek nevezzük.

Az előző ábra a) részének példájában az alkalmazott jelkészlet (1, 2, 3, 4, 5, 6, 7), az áttérés formai szabálya pedig az ábráról olvasható le (fehér ↔ 1, kék ↔ 2 stb.).

2.1.7 A hardver, szoftver és a förmver fogalma

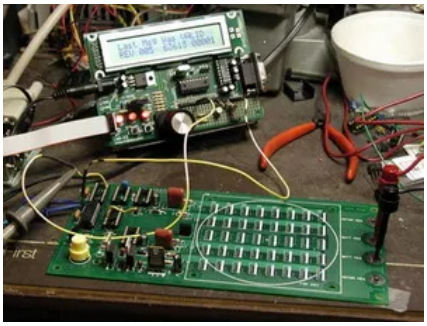
A **hardver** eredeti angol jelentése „kemény áru” (vas-áru). Lényegében ez a számítógépet alkotó, kézzel fogható eszközök összefoglaló neve, a számítógép elektronikus áramköreit, mechanikus berendezéseit, kábeleit, csatlakozásait és perifériáit nevezzük így.

A hardver önmagában egy működésképtelen eszközhalmaz. Ahhoz, hogy a számítógép egy feladatot meg tudjon oldani, azt algoritmus formájában meg kell fogalmazni és a számítógéppel az általa értelmezhető utasítások formájában közölni kell. A számítógépet működőképessé tevő programok összességét **szoftver**nek (eredeti jelentése: „lágy áru”) nevezzük.

A számítógépek általános, gyakran ismétlődő vezérlési feladatait végző programokat legtöbbször kisebb, csak olvasható memóriákban (ROM – Read Only Memory) helyezik el. Ezeket a programokat, a tárolóeszközzel együtt förmvernek (firmware) nevezik.

Hardver

Szoftver



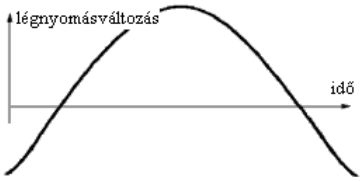
Förmver

2.2 Az analóg és digitális technika

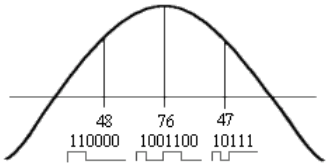
A digitális technikával abból a világból, amiben a megszokott mércerendszerünk „analóg” a valóságosan érzékelhető mennyiségekkel, átlépünk a számoknak egy olyan világába, ahol a folyamatosan változó mennyiségeket diszkrét módon, pl. számjegyekkel ábrázolva dolgozzuk fel.

Talán a legjobb példa erre az emberi hang.

Az a) ábrán látható az emberi hang ábrázolása, ahogy az a valóságot megközelíti, a b) ábrán pedig számjegyekkel meghatározott értékekkel megjelenítve ugyanaz a hang.



a) Emberi hangot ábrázoló görbe



b) Emberi hangot ábrázoló görbéből vett minták alapján előállított digitális (azaz számjegyekkel meghatározott értékekkel megjelenített) jelek

Az ábrák alapján is egyszerűen megítélhető, hogy mennyivel nagyobb nehézségi fokot jelent az emberi hang alakhű ábrázolása annál, mintha csak azt kell előállítanunk valamilyen elektronikus eszközzel, hogy van jel vagy nincs jel. Ezzel érzékelhetjük azt, hogy mi a különbség az analóg és digitális jel között és hogyan alakíthatjuk át az egyik jelet a másikba.

2.2.1 Analóg technika

Analógnak nevezzük az olyan eszközöket, eljárásokat stb., amelyek folytonos mennyiségeket ábrázolnak, illetve dolgoznak fel. (A szó alapjelentése: hasonlóságon alapuló.) Az analóg jelek ábrázolási vagy feldolgozási tartománya megfeleltethető a valós számok egy intervallumának.

Az analóg jelenségeket ezért folytonos fizikai mennyiségek segítségével modellezhetjük. (Egy analóg jel bizonyos határok között a jeltartományon belül tetszőleges értéket felvehet.)

Analóg elven működő áramkörök:

- lineáris erősítők: műveleti erősítők, hangfrekvenciás erősítők, videó erősítők, széles sávú erősítők stb.,
- nem lineáris erősítők: keverők, modulátorok, függvényeket megvalósító áramkörök, komparátorok stb.,
- forrásáramkörök: oszcillátorok, stabilizátorok stb.,
- jelátalakító áramkörök: analóg-analóg jelátalakító, analóg-digitális jelátalakító, digitális-analóg jelátalakító

Példák analóg eszközökre:

Oszcilloszkóp

Fényképezőgép

Telefon

Diktafon

2.2.2 Digitális technika

Digitálisnak nevezzük az olyan adatokat, eszközöket, eljárásokat stb., amelyek változó mennyiségeket számjegyekkel, diszkrét módon ábrázolnak, illetve dolgoznak fel.

Példák digitális eszközökre:

Fényképezőgép

Monitor

MP3 lejátszó

Az analóg-digitális átalakítók (A/D átalakító, A/D konverter, ADC, digitalizáló) olyan áramkörök, illetve eszközök, amelyek a bemenetükre adott analóg jelet digitális jellé alakítják át.

A
digitális ábrázolás valamely változó értékének diszkrét ábrázolása számjegyekkel. (A digitális jel számjegyekkel azért ábrázolható, mivel egy ilyen jel csak véges számú diszkrét értéket vehet fel.)

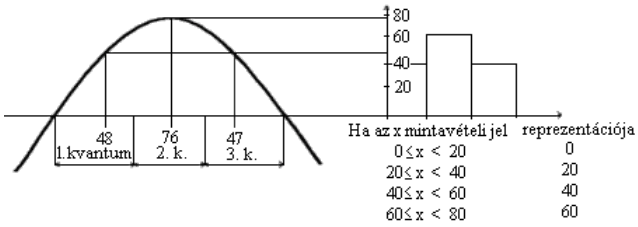
A digitális elven működő áramkörök a digitális ábrázolású adatok feldolgozására szolgálnak. A digitális áramkörök impulzusok alakjában lévő adatokat dolgoznak fel. A pozitív és negatív, azaz bináris (binary) impulzusokat feldolgozó áramkörök kombinációs vagy szekvenciális működésűek lehetnek.

2.2.3 Jelátalakítás

2.2.3.1 Analóg–digitális átalakítás

Az analóg–digitális (A/D) átalakítás olyan eljárás, amely az analóg jelből előállítja az annak megfelelő digitális jelet. Általában ez két műveletből áll: a mintavételből és a kvantálásból. Nézzük, hogy esetünkben mi is az a minta? Mintának nevezzük valamely jel valamely jellemzőjének megmért értékét egy adott pillanatban**. A mintavétel az az eljárás, amellyel egy folytonosan változó mennyiséget csak bizonyos időpontokban (a mintavételi időpontokban) felvett értékeivel jellemzünk.**

**Ez az alapja az analóg jelek digitális átalakításának, amikor is az analóg jelet (pl. beszéd) kellő mintavételi frekvenciával letapogatják. (A mintavételi frekvencia, vagyis a mintavétel gyakorisága, a másodpercenként vett minták számát jelenti.) A mintavétel után a digitalizálás következő művelete a kvantálás. A kvantálás során a változó értéktartományát egymásba nem nyúló, nem feltétlenül egyenlő hosszúságú véges sok intervallumra osztják, és minden intervallumot egy kijelölt elemével
reprezentálnak. (Ettől lesz a jel szakaszos.)**



A mintavétel során kapott értékek digitalizálása kvantálással

2.2.3.2 Digitális–analóg átalakítás

A digitális–analóg (D/A) átalakítás olyan eljárás, amely a digitális jelből előállítja az analóg jelet.

A digitális/analóg átalakító (D/A átalakító, D/A konverter, DAC) a digitális ábrázolású adatot analóg ábrázolású adattá alakítja, vagyis olyan áramkör, amelynek bemenetére (bemeneteire) digitális jelet (jeleket) adnak és kimenetén a bemeneti jel számértékének megfelelő nagyságú analóg jel jelenik meg.

Erre jó példa a számítógépen digitális formában tárolt adatok megjelenítése analóg elektronsugárral a monitor képernyőjén.

Példák D/A átalakítókra:

Digitális-analóg multiméter

Digitális-analóg PCI hangkártya

Digitális-analóg kamera

2.2.4 Logikai áramkörök

2.2.4.1 Bináris logika

A logikai áramkörök diszkrét értékű bemenő jelekből, meghatározott matematikai logikai függvényeknek megfelelő kimenőjelet állítanak elő. Ezen logikai értékeket az áramkör valamely elektromos jellemzőjéhez rendeljük hozzá.

Egy logikai áramkör kapcsolási funkciót lát el, vagyis megengedi vagy megakadályozza az áram folyását. A számítógépen belül ez a funkció automatikusan végbemegy, amikor például az órajel megjelenik. Kapcsolóáramkörök kombinációja logikai műveletek végrehajtására alkalmas.

Általában kétértékű logikai rendszereket használnak, ahol leggyakrabban a feszültség a választott elektromos jellemző. Ezek a kapcsolóéhoz hasonló jellegű működéssel a logikai döntéseket elektromosan valósítják meg. Ez a bináris működés minimális követelményeket támaszt az alkatrészek tűrésével szemben, ezért a digitális áramkörök integrált formában jól megvalósíthatók.

Az olyan logikai rendszert, amely kétállapotú elemekkel valósítja meg a kívánt logikai kapcsolatokat, bináris logikának nevezzük.

A két különböző állapot szokásos elnevezései:

H (High=Magas)	L (Low=Alacsony)	A kétértékű logikai rendszerekben két logikai szint lehetséges: a feszültség abszolút értelemben nagyobb értéke a logikai magas szint, jele H (H=High), kisebb értéke a logikai alacsony szint, jele L (L=Low), a kettő közötti átmeneti tartomány biztosítja a jelek jó szétválaszthatóságát.
„1”	„0”	
„IGAZ”	„HAMIS”	A pozitív logikai konvenció a fizikai változókhoz (áram, feszültség) a H=1 és L=0 szerint, a negatív logikai konvenció a H=0 és az L=1 szerint rendeli hozzá a logikai értékeket.
„IGEN”	„NEM”	
„TRUE”	„FALSE”	A logikai áramkör bemenetei és kimenetei a két szint közötti értéket hibátlan működés esetén csak átmeneti (tranziens) állapotban vehetnek fel.

A logikai áramkörök a három alapvető logikai funkciót, az „ÉS” (AND), a „VAGY” (OR) és a „NEM” (NOT) műveleteket végzik el. Ezzel a három alapművelettel összetettebb logikai kifejezések írhatók fel.

2.2.4.2 Boole-algebra

George Boole (1815–1864) angol matematikus alakította ki azt az algebrai rendszert, amely logikai feladatok megoldására is alkalmas. Ezzel a rendszerrel megteremtették a mai digitális elektronikus berendezések kialakításának elméleti alapjait.

A Boole-algebrában egy nem üres M halmazból indulunk ki, és feltesszük a következőket:

M-en értelmezve van két kétváltozós művelet, az egyiket nevezzük összeadásnak (+), a másikat szorzásnak (·). Ez azt jelenti, hogy M bármely A és B elemére A + B is, A · B is eleme M-nek.

M-en értelmezve van egy egyváltozós művelet, nevezzük komplemens képzésnek ($\bar{}$), vagyis M bármely A eleme esetén $\overline{A}$ is eleme M-nek.

Az M halmazt az előző lapon tárgyalt műveletekkel együtt Boole-algebrának nevezzük, ha M bármely A, B, C elemére fennállnak a következők:

(1) A · (B · C) = (A · B) · C;

(2) A + (B + C) = (A + B) + C;

(3) A · B = B · A;

(4) A + B = B + A;

(5) A · (B + C) = (A · B) + (A · C);

(6) A + (B · C) = (A + B) · (A + C);

(7) A · (A +B) = A;

(8) A + (A · C) = A.

(9) Létezik M-nek olyan, a Φ-val jelölt eleme, hogy minden M-beli A-ra Φ · A = Φ Φ + A = A.

(10) Létezik M-nek olyan, a Φ-tól különböző, I-vel jelölt eleme, hogy minden M-beli A-ra I · A = A; I + A = I

(11) Bármely M-beli A esetén A · A = Φ A + A = I.

A fenti (különben nem független) axiómáknak több konkrét matematikai struktúra is megfelel.

A Boole-algebráknak alapvető szerepük van a halmazelméletben, a valószínűségszámításban, továbbá a mi tárgyalásunkat közelebbről érintő kétértékű logikai változókkal kapcsolatos műveleteknél is.

A logikai algebra számunkra legfontosabb alkalmazása a G. E. Shannon által kidolgozott kapcsolás-algebra, amely a digitális áramkörök működésének alapját képezi. Itt az M halmaznak két eleme van, a 0 és az 1.

2.2.4.2.1 Boole-művelet

A Boole-művelet
olyan logikai művelet, amely logikai értékekhez (a művelet operandusaihoz) logikai értékeket (a művelet eredményét) rendel hozzá. Pl.: a kétoperandusú ÉS művelet az A és B logikai változókhoz az alábbi eredményt (A · B) rendeli:

A	B	X = A · B
0	0	0
0	1	0
1	0	0
1	1	1

A legfontosabb Boole-műveletek az egyoperandusú (egyváltozós) NEM művelet (negáció) és a kétoperandusú (kétváltozós) ÉS, VAGY, illetve KIZÁRÓ VAGY műveletek.

2.2.4.2.2 Igazságtáblázat

A Boole-műveleteket a logikában általában ún. igazságtáblázatukkal adják meg. Ez egy olyan táblázat, amely az operandus(ok), más néven változók minden lehetséges értékkombinációjához megadja az eredmény értékét.

A lehetséges egyváltozós Boole-műveleteket az alábbi táblázat foglalja össze:

Bemenet értéke	Kimenet értéke			
	0 állandó	EGYEN-ÉRTÉKŰ-SÉG	NEM	1 állandó
0	0	0	1	1
1	0	1	0	1

Egyváltozós Boole-műveletek igazságtáblázatai

Kétváltozós műveletek igazságtáblázata (két bemenő változó esetén a lehetséges logikai függvények):

A bemenet értékei		Konstans 0	NEM-VAGY	A tiltása ÉS	Egyváltozós NEM (A)	B tiltása ÉS	Egyváltozós NEM (B)	Kizáró VAGY (ANTIVALENCIA)	NEM-ÉS	ÉS	Egyenértékűség (EKVIVALENCIA)	Egyváltozós egyenértékűség (B)	Ha A akkor B (IMPLIKÁCIÓ)	Egyváltozós egyenértékűség (A)	Ha B akkor A (IMPLIKÁCIÓ)	Megengedő (VAGY)	Konstans 1
A	**B**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Egy- és kétváltozós Boole-műveletek igazságtáblázatai

A logikai függvények táblázatai tehát megadják, hogy a bemeneti jelek különféle variációi a kimeneten milyen jelet eredményeznek.

Bebizonyítható, hogy minden logikai művelet kifejezhető az ÉS, a VAGY és a NEM műveletek alkalmas kombinációjával.

Egyes lehetséges kimeneteknek az áramköröknél fontos KIZÁRÓ VAGY (jelölése $\oplus$) az ÉS (jelölése $\bullet$), VAGY (jelölése +) és NEM (jelölése $\overline{\ }$) műveletekkel az alábbi függvények feleltethetők meg:

Kimenet sorszáma	Függvény	Kimenet sorszáma	Függvény
0	0	8	$A \cdot B$
1	$\overline{A + B}$	9	$A \overline{\oplus} B$
2	$\overline{A} \cdot B$	10	B
3	$\overline{A}$	11	$A + B$
4	$A \cdot \overline{B}$	12	A
5	$\overline{B}$	13	$A + B$
6	$A \oplus B$	14	$A + B$
7	$\overline{A \cdot B}$	15	1

Egy- és kétváltozós logikai függvények

Az igazságtáblázatokkal előbb megadott műveletek eleget tesznek a Boole-algebra szabályainak.

2.2.4.3 Kapuáramkörök, logikai áramkörök

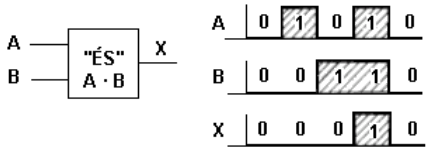
ÉS (AND) kapu

Az ÉS kapu olyan logikai áramkör, amely az ÉS műveletet valósítja meg, vagyis az adott bemeneti logikai változókra az ÉS művelet eredményét képezi.

A logikai szorzást végző ÉS kapuáramkör olyan több-bemenetű logikai elem, amelynek kimenetén akkor és csak akkor jelenik meg az 1-et (logikában az igazat) fizikailag képviselő jelszint (továbbiakban H szint), ha az összes bemenetén H szint van.

A	B	$X = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

A logikai szorzás (konjunkció) igazságtáblázata



A logikai szorzás jelképi jelölése és idődiagramja

A kapuáramkörök olyan egy vagy több-bemenetű logikai áramkörök, amelyek egy meghatározott logikai műveletet valósítanak meg. A logikai felépítéstől függően a kapuk lehetnek kombinációs, vagy szekvenciális (sorrendi) áramkörök.

A kombinációs áramkör olyan logikai áramkör, amely nem tartalmaz tárolóelemet, és amelynek kimeneti értékei csakis a bemeneti értékek pillanatnyi értékétől függenek.

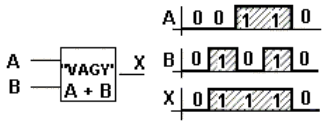
A szekvenciális (sorrendi) áramkörök következő állapotát a jelen bemeneti állapot és a hálózat belső állapota együttesen határozza meg.

VAGY (OR) kapu

A logikai összeadást végző VAGY kapuáramkör olyan több-bemenetű logikai elem, amelynek kimenetén akkor és csak akkor jelenik meg H szint, ha legalább egy bemenetén H szint van.

A	B	X = A+B
0	0	0
0	1	1
1	0	1
1	1	1

A logikai összeadás (diszjunkció) igazságtáblázata



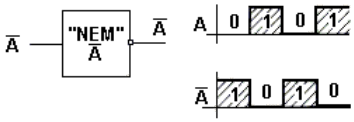
A logikai összeadás jelképi jelölése és idődiagramja

INVERTER vagy NEM (NOT) kapu

A NEM kapu olyan logikai áramkör, amely a NEM műveletet valósítja meg, azaz bemenetére adott logikai változóra a NEM műveletének eredményét képezi.
A NEM kapuáramkör egybemenetű logikai elem, amelynek a kimenetén mindig a bemenettel ellentétes szint jelenik meg (a H szinttél ellentétes szint jele: L).

Ha A = 1, akkor $\overline{A}$ = 0 (L szint),

ha A = 0, akkor $\overline{A}$ = 1 (H szint).



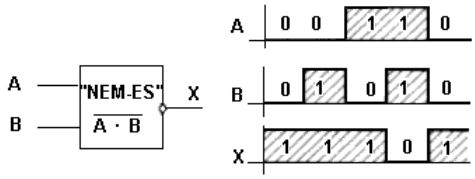
Az invertálás jelképi jelölése és idődiagramja

NAND vagy NEM-ÉS (NOT-AND) kapu

A NAND kapu olyan logikai áramkör, amely a NEM-ÉS műveletet valósítja meg, vagyis az adott bemeneti logikai változókra a NEM-ÉS művelet eredményét képezi.
A NAND kapuáramkör több-bemenetű logikai elem, amelynek kimenetén akkor és csak akkor jelenik meg L szint, ha az összes bemenetén H szint van.

A	B	X = $\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

A NEM-ÉS művelet igazságtáblázata



A NEM-ÉS művelet jelképi jelölése és idődiagramja

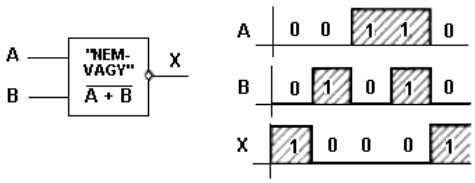
NOR vagy NEM-VAGY (NOT-OR) kapu

A NOR kapu olyan logikai áramkör, amely a NEM-VAGY műveletet valósítja meg, vagyis az adott bemeneti logikai változókra a NEM-VAGY művelet eredményét képezi.

A NOR kapuáramkör több-bemenetű logikai elem, amelynek a kimenetén akkor és csak akkor jelenik meg L szint, ha legalább egy bemenetén H szint van.

A	B	$X = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

A NEM-VAGY művelet igazságtáblázata



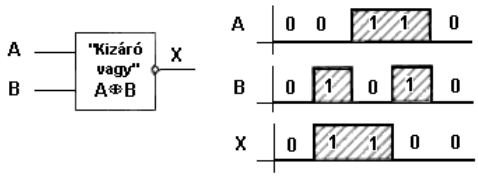
A NEM-VAGY művelet jelképi jelölése és idődiagramja

A KIZÁRÓ VAGY (EXCLUSIVE OR, XOR)

A kizáró VAGY olyan logikai áramkör, mely az ún.ANTIVALENCIA műveletet valósítja meg; a bemenetére adott logikai változók az ANTIVALENCIA műveletének eredményét képezik.
A kizáró VAGY kétbemenetű logikai elem, amelynek kimenetén akkor és csak akkor jelenik meg H szint, ha csak egy bemenetén van H szint.

A	B	$X = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

A logikai kizáró VAGY igazságtáblázata



A logikai kizáró VAGY jelképi jelölése és idődiagramja

Az animáció elindításához nyomja meg a megfelelő kapu nevét.

A kombinációs döntési hálózat általában olyan digitális hálózat, amelynek bármely bemeneti állapotához, jelkombinációjához egy definiált kimeneti jelkombináció tartozik.

A mikroszámítógépek általában nem rendelkeznek az összes alap kapuáramkörrel, de kombinációs hálózatokkal minden szükséges logikai függvény előállítható.

Az

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

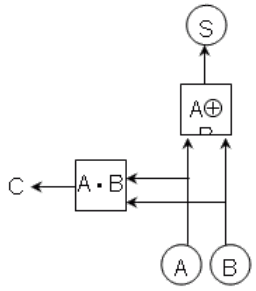
$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

ún. de Morgan-féle szabályok segítségével például az ÉS művelet előállítható a VAGY és NEM művelet segítségével:

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

A számítógépek működésében meghatározó jelentőségű az aritmetikai műveletek végrehajtása. Ezért nagyon fontos, hogy aritmetikai műveleteket végrehajtó áramköröket is kialakíthatunk kombinációs áramkörökből.
Erre példát a következő 1-bites összeadó egység áramköri felépítése mutat.

A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1



Jelölések:
A, B a két összeadandó érték
S az összeg
C a következő helyi értékre való átvitel

1-bites összeadás igazságtáblázata és az 1-bites összeadóegység áramköri felépítése

2.3 Digitális adatábrázolás a számítógépekben

2.3.1 Számrendszerek

A számrendszerek valós ábrázolására szolgáló jelek és az ezek alkalmazására vonatkozó szabályok összessége.

Legelterjedtebben a helyi értékes rendszereket használják, amelyekben a valós számot karakterek sorozata úgy ábrázolja, hogy minden számjegypozícióhoz valamely helyi érték van hozzárendelve, és a valós szám értékét az egyes helyi értékek és a hozzájuk tartozó alaki értékek szorzatainak az összege adja.

A digitális technikában a számrendszereknek alapvetően fontos gyakorlati szerepük van. A mennyiségeket (valós számokat) a helyi értékes írásmódban a választott alapszám hatványaival írjuk fel. Az alapszám bármely 1-nél nagyobb egész szám lehet.

2.3.1.1 A tízes (decimális) számrendszer

A tízes számrendszer alapszáma a 10.

A tízes számrendszerben a valós számot karakterek sorozata ábrázolja úgy, hogy az egyes számjegypozíciókhoz a tizedesjeltől balra a 10^0, 10^1, 10^2 stb. helyi értékek, míg a tizedesjeltől jobbra a 10^{-1}, 10^{-2}, 10^{-3} stb. helyi értékek vannak hozzárendelve, és a valós szám értékét az egyes számjegypozíciókon álló számjegyek és a hozzájuk tartozó helyi értékek szorzatainak összege adja.

Pl.: a 135, illetve a 123,45 karaktersorozat értéke a tízes számrendszerben:

$135 = 1 \times 10^2 + 3 \times 10^1 + 5 \times 10^0$

$123{,}45 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$

2.3.1.2 A kettes (bináris) számrendszer

A kettes számrendszer alapszáma a 2. A kettes számrendszerben egy számjegyet két számjegy, a „0" és az „1" segítségével lehet felírni, a helyi értékkel tüntetve fel azt, hogy a jegyet a kettő hányadik egész kitevőjű hatványával kell megszorozni. Például:

Decimális írásmód	Írásmód a 2 alapú hatványok összegével	Bináris írásmód
1	1×2^0	1
2	$1 \times 2^1 + 0 \times 2^0$	10
3	$1 \times 2^1 + 1 \times 2^0$	11
4	$1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	100
5	$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$	101
6	$1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$	110
7	$1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$	111
8	$1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	1000
135	$2^7 + 2^2 + 2^1 + 2^0$	10000111
0,25	2^{-2}	0,01

Mint említettük, az elektronikus digitális számítógépek a kettes számrendszer alapján működnek, amit az a tény indokol, hogy bizonyos áramköri elemek és áramkörök tulajdonságai alapján két állapot jól meghatározható, pl.: mechanikus vagy elektronikus kapcsoló (félvezető eszközök, flip-flop áramkör, kriotron, ferrit stb.) eszközökkel.

A kettes számrendszerben megadott számokat vagy sorosan impulzussorozatokkal, vagy párhuzamosan, az egyes helyi értékeknek megfelelő impulzusokkal lehet reprezentálni.

2.3.1.3 A tizenhatos (hexadecimális) számrendszer

A tizenhatos (hexadecimális) szám olyan helyi értékes számrendszerben ábrázolt szám, amelynek alapszáma 16. A hexadecimális karakterek szokásos jelölése a következő:

- **decimálisan: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15**
- **hexadecimálisan: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**

Tehát a 10 decimális számjegy mellett még az ABC első hat betűjét is felhasználják a 16 számjegy jelölésére. Egy hexadecimális számjegy négy bináris számjeggyel ábrázolható ($16=2^4$).

Példa: Legyen egy decimális szám: 26, ennek bináris ábrázolása: 11010, hexadecimális számmal ábrázolva: 1A. A számítógépek tárolóegységét, a bájt 8 bitjét, két hexadecimális számjeggyel lehet leírni.
Például: a 1001 1111 bájt értéke hexadecimálisan kifejezve: 9F.
Példák a számok különböző számrendszerekben való ábrázolására:

Decimális 10-es	Bináris 2-es	Hexadecimális 16-os
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Példák számok 10-es, 2-es és 16-os számrendszerben történő

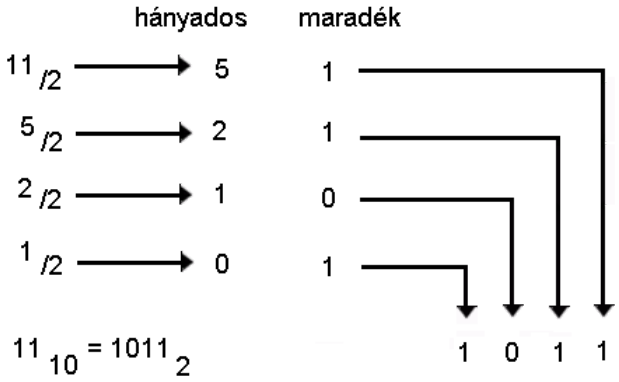
ábrázolására

2.3.1.4 Számrendszerek átalakítása

2.3.1.4.1 Decimális-bináris átalakítás

A decimális–bináris átalakítás az az eljárás, amelynek során a tízes számrendszerben írt számokat azokkal egyenlő értékű kettes számrendszerbeli számokká alakítják át.

Az átalakítás algoritmusát az alábbi ábra szemlélteti.



Számok átalakítása

10-es számrendszerből 2-es számrendszerbe

2.3.1.4.2 Bináris-decimális átalakítás

A bináris–decimális átalakítás az az eljárás, amelynek során a kettes számrendszerben írt számokat azokkal egyenlő értékű tízes számrendszerbeli számokká alakítják át.

Például a 1011 bináris szám esetén az eljárás a következő:

$$1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 = 11_{10}$$

2.3.1.5 Műveletvégzés bináris számokkal

2.3.1.5.1 Bináris számok összeadása

A lehetséges kombinációk két bináris szám összeadása esetén:

1. Példa:

a + b	=	összeg	átvitel
0 + 0	=	0	0
0 + 1	=	1	0
1 + 0	=	1	0
1 + 1	=	10	1

Az átvitel a következő magasabb helyi értékű pozícióhoz kerül hozzáadásra.

2. Példa: a decimális és bináris számok összeadásának összehasonlítása:

	decimális összeadás	bináris összeadás
	208	11010000
	+ 92	+ 1011100
összeg:	300	100101100
átvitel:	11	11 1

2.3.1.5.2 Bináris számok kivonása

Két szám különbségét a számítógép ún. komplemens képzési és összeadási művelettel állítja elő.

A számítógépben a bináris számok ábrázolására korlátozott számú számjegypozíció, másnéven bit áll rendelkezésre. Legyen az adott számú pozíción ábrázolható legnagyobb szám M. Ekkor egy tetszőleges bináris szám egyes komplemensének az M és az adott szám különbségét értjük. Kettes komplemensnek nevezzük az előző különbségnél eggyel nagyobb számot.

A meghatározásból következik, hogy ha egy számhoz hozzáadjuk a kettes komplemensét, éppen azt a legkisebb számot kapjuk, amely az adott számú pozíción már nem fér el.

Például: Tegyük fel, hogy a számok ábrázolásához 4 bit áll rendelkezésre. Határozzuk meg a 11 bináris kódú szám egyes és kettes komplemensét. A négy biten ábrázolható legnagyobb szám: 1111.

Egyes komplemens: 1111 − 0011 = 1100 Kettes komplemens: 1100 + 1 = 1101

Ellenőrizzük, hogy a kettő összege valóban az a legkisebb szám, ami nem fér el 4 biten.
A legkisebb szám, ami már 4 biten nem fér el: 10000.

Az eredeti szám: 0011
Kettes komplemens: + 1100
10000

A kettes komplemens tehát a számot éppen arra a legkisebb számra egészíti ki, amely az adott számú pozíción már nem fér el. A fenti összeadást érdemes még egyszer szemügyre venni. Ha a műveletet a gép végzi, az eredmény 0000 lesz, hiszen csak 4 bit áll rendelkezésre, és a gép a bal oldali egyest egyszerűen levágja. Ha tehát egy számhoz hozzáadjuk a kettes komplemensét, eredményül 0-át kapunk (feltéve, hogy korlátozott számú pozíció áll rendelkezésre). A kettes komplemenst tehát tekinthetjük az eredeti szám belső ábrázolású ellentettjeként. A gondot most már csak az jelenti, hogy hogyan különböztetjük meg egymástól a pozitív és negatív számokat. Erre szolgál az ún. előjelbit.

A továbbiakban a számot negatívnak tekintjük, ha a bal szélső bit értéke 1, pozitívnak vesszük, ha 0. Az előző összeadásban a túlcsordult 1-es éppen az előjelbit helyén van. Az előjelbittel természetesen összeadási műveletet nem végzünk. Példaként ábrázoljuk a decimális 22 és -22 számokat 6 biten.

Egyes komplemens: 1111 − 0011 = 1100 Kettes komplemens: 1100 + 1 = 1101

(A -22 belső kódját a 22 kettes komplemenseként kaptuk.)

Kivonás
Példák:

a)

	bináris kivonás	bináris kivonás számítógéppel	
Kisebbítendő:	10001	10001	
Kivonandó:	- 01011	+ 10101	(a kivonandó 2-es komplemense)
Különbség:	00110	100110	

↓

Pozitív eredmény esetén az utolsó átvitelt törölni kell.

b)

	bináris kivonás	bináris kivonás számítógéppel	
Kisebbítendő:	101	101	
Kivonandó:	- 11011	+ 00101	(a kivonandó 2-es komplemense)
Különbség:	10110	01010	

↓

01010-nak a 2-es komplemense = 10110

Ha az utolsó átvitel = 1, a kivonás eredménye pozitív, ha az utolsó átvitel = 0, akkor a kivonás eredménye negatív.
A bináris számok szorzása és osztása ugyanolyan lépések sorozatából áll, mint a decimális számok szorzása és osztása.

2.3.2 A számítógépes számábrázolás

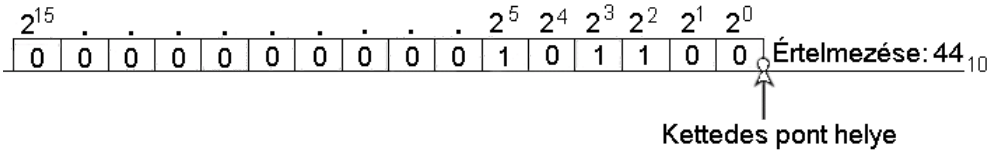
2.3.2.1 Fixpontos számok

Fixpontos számtárolási formánál a szám kettes számrendszerbeli együtthatóit tároljuk helyi értékeiknek megfelelően egy rögzített nagyságú memóriaterületen. Ez leggyakrabban

- **1 bájt méretű,**
- **2 bájt méretű szó (word),**
- **4 bájt méretű duplaszó (dword).**

A fixpontos számoknál fontos kérdés, hogy melyik pozíción helyezkedik el a szám egész- és törtrészét elválasztó jel, melyet a tizedesvessző analógiájára kettedes pontnak nevezhetünk.

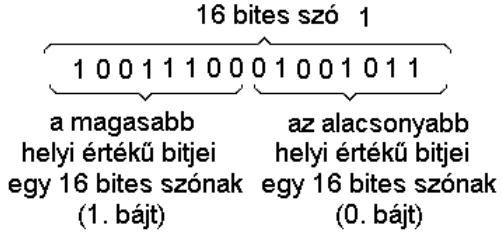
A „kettedes pontról" kapta ez a számábrázolási forma az elnevezését, mivel az angolszáz szóhasználatban tizedesvessző helyett a tizedespontot alkalmazzák.
Minden számítógépnél a kettedes pontnak rögzített a helye, a gép a tárolt bitsorozatot ennek megfelelően értelmezi. A leginkább elterjedt a fixpontos számok egész számként való értelmezése, ebben az esetben a kettedes pont a legalacsonyabb helyi érték után helyezkedik el.



Fixpontos egész szám (előjel nélküli szó)

Ha 8 bites a fixpontos szám, akkor ez a bináris számjegyek (0,1) 256 (2^8) kombinációját engedi meg, az ábrázolható számok tartománya tehát 000000002 = 0_{10}-tól 11111111_2 = 255-ig terjed.

A 8 bites adatok kezelhetők párban is, egy 16 bitből álló fixpontos szám például:



Ebben a szóban (előjel nélkül) a legkisebb ábrázolható érték 0, a legnagyobb pedig 2^{16}-1=65535.

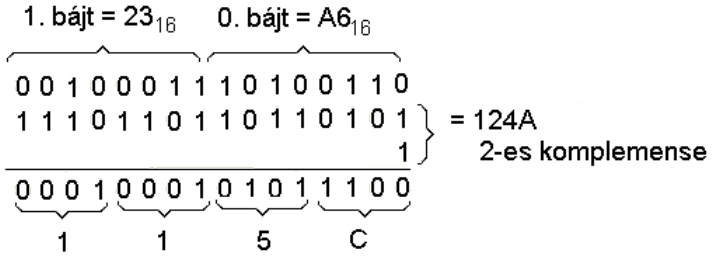
A fixpontos számokkal a 2-es számrendszerben végezhetünk műveleteket. Például, két 16 bites szám összeadása a következő szerint történik.

10011101	10000110
+ 00101010	11010100
11001000	01011010

A kivonást úgy végezzük, hogy a kivonandó 2-es komplemensét hozzáadjuk a kisebbítendőhöz.

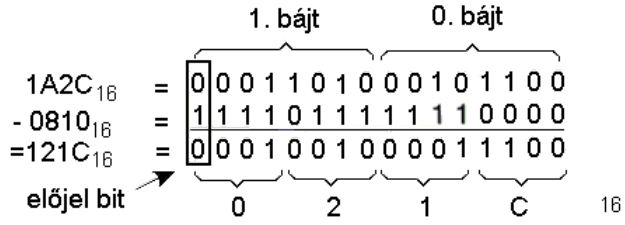
Példa: $23A6_{16} - 124A_{16} = 115C_{16}$

Végezzük el a kivonást 16 bites memóriaszavakkal.



Összeadási példa előjel bittel (1 = negatív, 0 = pozitív)

$1A2C_{16}$ + (-0810_{16}) = $121C_{16}$



A -0810_{16} előállítása:

0810_{16} =	**0000 1000 0001 0000**
A 0810_{16} 1'-es komplemense =	**1111 0111 1110 1111**
	+1
A 0810_{16} 2'-es komplemense =	**1111 0111 1111 0000**

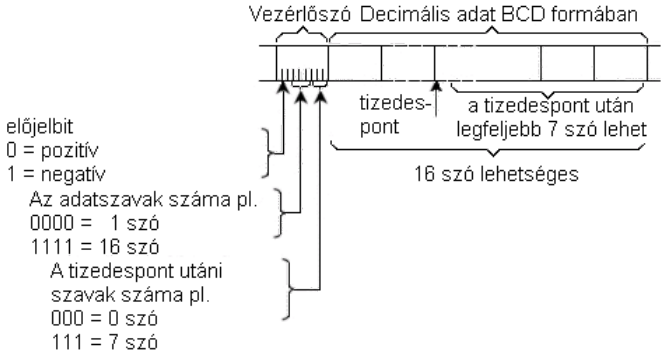
2.3.2.2 Binárisan kódolt decimális számok (BCD kódok)

A BCD kódolás decimális számjegyek olyan bináris ábrázolási módszere, ahol minden decimális számjegyet helyi értékenként 4 bittel ábrázolnak. (Más szóval: a 0–9 számjegyeket 2-es számrendszerben fejezik ki, mindegyikben 4 helyi értéket véve.)

Bináris kód (szám)		Decimális kód (szám)
0000	=	0
0001	=	1
0010	=	2
0011	=	3
0100	=	4
0101	=	5
0110	=	6
0111	=	7
1000	=	8
1001	=	9

Előjeles BCD szám az eddigi szabályok szerint nem képezhető, ezért a BCD számok előjelének jelzésére speciális, úgynevezett vezérlőszó szükséges.

A vezérlőszó egyik lehetséges formátuma:



Vezérlőszó formátum

BCD aritmetikai műveleteknél speciális szabályokat kell alkalmazni. Nézzük meg, milyen az összeadás és a kivonás a binárisan kódolt decimális számokkal. Ennek lényege, hogy az egyes számjegyeknek megfelelő 4-bites egységeket összeadjuk vagy kivonjuk a 2-es számrendszer szabályai szerint, de az átviteleket a 10-es számrendszer szabályai szerint kell képezni a 4 bites egységek között.

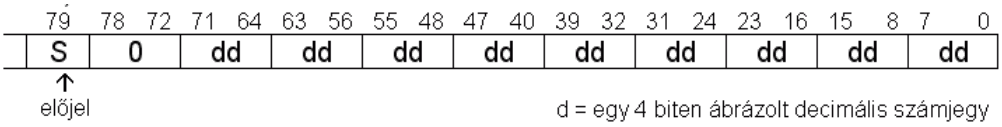
Példák az összeadásra:

			29		87	
			+ 34		+ 79	
			= 63		= 166	
Összeadandó	=	(29)	00101001	(87)	10000111	
66_{16}	=		+ 01100110		+ 01100110	
Összeg (1. lépés)	=		10001111		11101101	
Összeadandó	=	(34)	+ 00110100	(79)	+ 01111001	
Összeg (2. lépés)	=		11000011		01100110	
C/IC	=		0 1		1 1	
Faktor	=		10100000		00000000	
2. lépés összege	=		+ 11000011		+ 01100110	
Válasz	=		01100011		01100110	
			6 3		6 6	
C (2. lépés után)	=		0		1	
Eredmény	=		36		166	

Példák a kivonásra:

			75		25
			21		-71
			= + 54		= - 46
Kisebbítendő	=		00100001		01110001
A kisebbítendő 2'-es komplemense	=		11011111		10001111
Kivonandó	=		+ 01110101		+ 00100101
Az 1. lépés összege	=		01010100		10110100
	=		1 1		0 1
	=		00000000		10100000
A 2. lépés összege	=		+ 01010100		+ 10110100
	=		01010100		01010100
			5 4		5 4
C (átvitel a 2. lépés után)	=		1		0
			1-es átvitel esetén az eredmény = 54		0-ás átvitel esetén az eredmény = 46 (100-54=46)

Az IBM kompatibilis PC-knél a BCD kódban ábrázolt számok 18 jegyű decimális egész számok, amelyeket 10, egymást követő bájton tárolunk tömörítetten. A legfelső bájt legfelső bitje az előjel. Ha értéke 1, negatív a szám, ha 0, akkor pozitív. A legfelső bájt többi része kihasználatlan. A többi bájt egyenként két BCD jegyet tárol (lásd az ábrát alább).



BCD egész számok

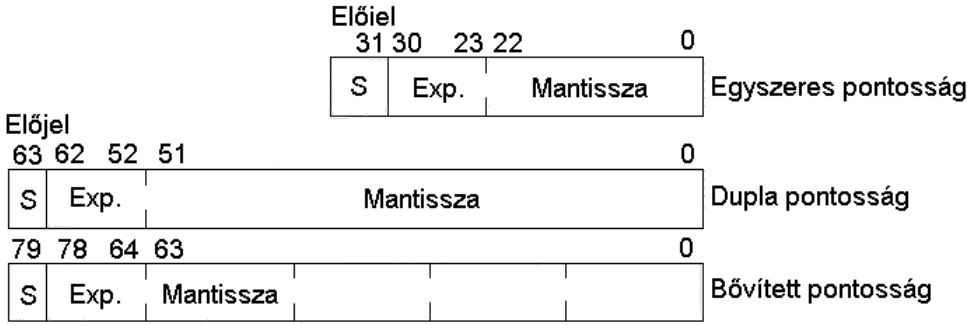
A formátum értéktartománya 0-tól ±999.999.999.999.999.999-ig tart.

2.3.2.3 Lebegőpontos számok

A hagyományos decimális lebegőpontos ábrázoláshoz (pl. $4{,}74 \times 10^3$) hasonló a bináris lebegőpontos ábrázolás. Az általános formátum: $\pm f \times 2^{\pm e}$, ahol f egy bináris tört, a mantissza vagy törtrész, az e pedig a 2 hatványkitevője, exponense. Három mező alkot tehát egy lebegőpontos számot: az előjel, az exponens és a mantissza.

Tehát a lebegőpontos számok számítógépes ábrázolása során 2 fixpontos számot, a mantisszát, és az exponenst kell együtt kezelnünk.

A lebegőpontos számok pontos tárolási formátumát pontosságuk határozza meg. Leggyakrabban az egyszeres, dupla és bővített (kiterjesztett) pontosságú lebegőpontos számokat használjuk, ezek formátumát mutatja be az alábbi ábra. Az exponens növelt (az ábrázolási tartomány felével, például 127-tel) értékét tároljuk, hogy mindig pozitív legyen.



Lebegőpontos adatformátumok a számábrázolás pontossága szerint

Az „S"-el jelölt előjel mező egyetlen bit, értéke 1, ha a lebegőpontos szám negatív; 0, ha a szám pozitív.

Az exponens mező a hatványkitevőt tartalmazza (Exp-el jelölve), ennek mérete:

- **8 bit egyszeres pontosságú számnál,**
- **11 bit dupla pontosságú számnál,**
- **15 bit bővített pontosságú számnál.**

A mantissza mező tartalmazza a szám törtrészét. Ennek mérete:

- egyszeres pontosságú számnál 23 bit,

- dupla pontosságú számnál 52 bit,

- bővített pontosságú számnál 64 bit.

A lebegőpontos számok számítógépes ábrázolására nemzetközi szabványokat dolgoztak ki (ANS/IEEE 754 és 854-es szabvány). Ez egyrészt meghatározza a lebegőpontos számok gépi ábrázolásának adatformátumát, másrészt pontosan leírja a lebegőpontos számokkal végezhető műveletek algoritmusait.

2.3.3 Karakterek kódolása

A számítógépben szöveg és más, nem szám jellegű adat kezelésére binárisan kódolt karaktereket alkalmazunk.

Ehhez általában a következő karakterek szükségesek:

26 kisbetű
26 kisbetű
kb. 25 speciális jel
+ 10 decimális szám
= 87

Ennek alapján a karakterkészlet 87 karakterből áll, amelynek ábrázolására 7 bináris digit szükséges (2^7=128).

A számítógépekben általánosan ASCII kódot (American Standard Code for Information Interchange) alkalmazzuk az adatok ábrázolására. Az ASCII kódnál egy jelkombináció 8 bitből (1 bájtból) áll: 7 bit + 1 paritásbit.

A paritásbit a karakterkódot ellenőrző, hibajelző bitje. A paritásbit értéke 1, ha a karakterben a bitek között páros számú az 1-es bitek száma, és 0, ha ez a szám páratlan.

Paritásbitek felhasználásával egy karakter jelkombinációjában az 1-esek számát hibajelzés céljából egy rendszerre vonatkozóan tetszés szerint változtathatjuk vagy párossá, vagy páratlanná.

Paritás	Krakterek paritásbit	Az 1-es bitek száma
páratlan	1 0001100	3
	0 1110011	5
páros	0 0000000	0
	1 0001110	4

Példa: ASCII kóddal ábrázolt karakterek páros paritásellenőrzés esetén:

hibás	E	n	r	e
0010000	11000101	11101110	01110010	01100101

Az ASCII kódolást a gyakorlatban kódtáblázatok segítségével hajtjuk végre (lásd a lenti ábrát).

Például a '6' karakter kódja (függőleges + vízszintes)

decimális : 48 + 6 = 54
hexadecimális: 30 + 6 = 36
bináris: 00110000 + 0110 = 00110110

DECIMÁLIS			0	16	32	48	64	80	9(
	HEXADECIMÁLIS		00	10	20	30	40	50	6(
		BINÁRIS	0000 0000	0001 0000	0010 0000	0011 0000	0100 0000	0101 0000	0' (
0	0	0000	NUL	DLE	SPACE	0	@	P	
1	1	0001	SOH	DC1	!	1	A	Q	
2	2	0010	STX	DC2	"	2	B	R	)
3	3	0011	ETX	DC3	#	3	C	S	
4	4	0100	EOT	DC4	$	4	D	T	
5	5	0101	ENQ	NAK	%	5	E	U	
6	6	0110	ACK	SYN	&	6	F		
7	7	0111	BEL	ETB	'	7	G		
8	8	1000	BS	CAN	(	8			
9	9	1001	HT	EM		9			
	A	1010							

Az ASCII kódtábla alkalmazása

A számítógépek elterjedésével fogalmazódott meg az igény a kódtáblázat olyan kiegészítésére, mely a speciális nemzeti karakterek (például a magyar Á, É, Í stb.) kódolását is lehetővé teszi. Magyarországon az IBM által ajánlott ún. „852”-es kódlapot szabványosították, melyet a melléklet tartalmaz.

2.4 Ellenőrző kérdések

1. Mi a legfontosabb különbség a számológép és a számítógép működése között?
2. Határozza meg a számítógép fogalmát!
3. Milyen analógia alapján határozta meg Neumann János a számítógépek működési elvét?
4. Sorolja fel a Neumann elveket!
5. Fejtse ki a tárolt programozás elvének lényegét!

6. Milyen funkcionális egységekből épül fel a tárolt programozás elvén működőszámítógép?
7. Mit jelent a kettes számrendszer alkalmazásának elve?
8. Mit jelent a soros utasításvégrehajtás elve?
9. Milyen tényezőkkel magyarázható a számítógépek gyors elterjedése és széleskörű elhasználhatósága?
10. Mit nevezünk rendszernek?
11. Hogyan függ a rendszer a vizsgálat céljától?
12. Milyen elemkapcsolatokat értünk a rendszer fogalmán?
13. Mit jelent az irányítás és a szervezés?
14. Mit nevezünk inputnak és outputnak?
15. Mit nevezünk algoritmusnak?
16. Mit nevezünk programnak és utasításnak?
17. Határozza meg az adat fogalmát!
18. Határozza meg az információ fogalmát!
19. Hogyan határozhatjuk meg az adat és az információ fogalmát a befogadó ember szerepeszerint?
20. Mit értünk információtechnológián?
21. Határozza meg az információs rendszer fogalmát!
22. Határozza meg az adatfeldolgozó rendszer fogalmát!
23. Milyen összefüggés van az adatfeldolgozó és az információs rendszer között?
24. Mit nevezünk informatikának?
25. Milyen összefüggés van az informatika és a számítástechnika között?
26. Mit értünk karakteren és milyen típusú karaktereket ismer?
27. Mit jelent a kódolás?
28. Mit nevezünk kódrendszernek?
29. Határozza meg a hardver fogalmát!
30. Határozza meg a szoftver fogalmát!
31. Mit értünk analóg adatokon, eszközökön?
32. Mikor nevezzük digitálisnak az adatokat, eljárásokat, az eszközök működését?
33. Mi jellemzi a digitális áramköröket?
34. Határozza meg a kombinációs és szekvenciális áramkör fogalmát!
35. Milyen részműveletekből áll az analóg–digitális átalakítás?
36. Mit értünk mintavétel alatt?

37. Mit értünk kvantálás alatt?

38. Hogyan működik a digitális–analóg átalakító?

39. Mi jellemzi a logikai áramköröket és milyen funkciókat látnak el?

40. Mi a bináris logika?

41. Mi a pozitív, illetve a negatív logikai konvenció?

42. Mi teremtette meg a mai digitális berendezések kialakításának elméleti alapjait?

43. Mi a kapcsolásalgebra?

44. Mit értünk Boole-műveleten?

45. Mivel határozzuk meg a Boole-műveleteket?

46. Mit nevezünk kapuáramkörnek?

47. MI jellemzi az "ÉS" kaput?

48. Mi jellemzi a "VAGY" kaput?

49. Mi jellemzi az "INVERTER" kaput?

50. Adja meg a NAND kapu igazságtáblázatát!

51. Adja meg a NOR kapu igazságtáblázatát!

52. Adja meg a "kizáró vagy" igazságtáblázatát!

53. Mit értünk kombinációs logikai hálózaton?

54. Hogyan lehet összeadó áramkört felépíteni?

55. Hogyan ábrázoljuk a számokat kettes számrendszerben? Mutasson erre két példát!

56. Hogyan ábrázoljuk a számokat tizenhatos számrendszerben? Mutasson erre két példát!

57. Számítsa át kettes számrendszerbe a 135 számot!

58. Számítsa át tízes számrendszerbe a következő számot:
1 1 0 1 1 0 1 1

59. Adja össze a következő két számot kettes számrendszerben!
0 1 0 1 1 1 0 1
+ 0 1 1 1 1 1 0 1

60. Mit nevezünk egy szám kettes komplemensének?

61. Mit nevezünk fixpontos számnak és milyen bájt méretben használjuk ezeket?

62. Mit értünk BCD kód alatt?

63. Adja meg a $123_{(10)}$ BCD kódját!

64. Mi a lényege a BCD kódú műveleteknek?

65. Hogyan ábrázoljuk a BCD kódú számok előjelét IBM kompatibilis PC-k esetén?

66. Milyen részekből épül fel a lebegőpontos szám?

67. Mit értünk egyszeres, dupla és bővített pontosságú lebegőpontos számon?

68. Mire használható és mit jelent az ASCII kód?

69. Mi a paritásbit?

70. Mennyi az ASCII kód jelkészlete és egy karaktert hány bittel kódolunk?

3 A mikroszámítógép

3.1 Bevezetés

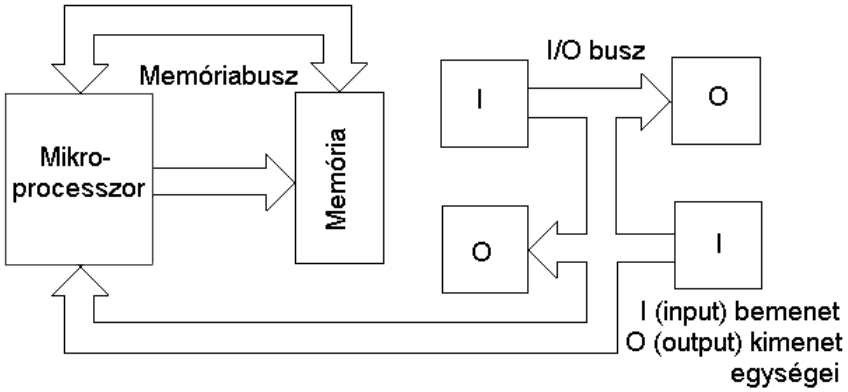
A mikroszámítógép fogalma a mikroprocesszorok megjelenését követően alakult ki egy olyan mikroprocesszort tartalmazó számítógépet jelölve, amely a kezdeti mikroprocesszorok teljesítmény-korlátainak megfelelően kis teljesítményű és alacsony árfekvésű volt. Jelenleg, amikor a számítógépek egyre nagyobb hányada tartalmaz mikroprocesszort, a mikroszámítógépek elnevezést egyre kevésbé használják. Szerepét, megváltozott jelentéstartalommal a házi, a személyi, a professzionális számítógép, illetve a „mérnöki munkahely" fogalmak veszik át. (PC = Personal Computer, Workstation stb.)

3.2 A mikroszámítógép felépítése

3.2.1 A központi egység (CPU = Central Processor Unit)

Minden mikroszámítógépnek négy alapvető funkcionális egysége van:

- **A bemeneti egység, amely az adatok és a program bevitelét biztosítja.**
- **A főtár (memória), amely a műveletek elvégzéséhez szükséges adatokat és programokat, valamint az eredményt tárolja későbbi felhasználás céljából.**
- **A mikroprocesszor, amely a memóriából kapott adatokon a programnak megfelelő logikai és számítási műveleteket elvégzi.**
- **A kimeneti egység, amelyen keresztül az eredmény eljut a felhasználóhoz.**



Mikroszámítógép alapegységei

A funkcionális egységek közötti kommunikációs kapcsolatokat, azaz a címek, adatok, vezérlőinformációk átvitelét, a mikroszámítógép busz- vagy sínrendszere biztosítja.

A busz egy vezetékrendszer, melynek fontosabb fajtái:

- **a belső busz, mely a mikroprocesszoron belüli részegységeket köti össze,**
- **a memóriabusz, mely a processzort köti össze a tárral,**
- **az I/O vagy rendszerbusz, mely a mikroszámítógép processzorát a be-/kiviteli egységekkel kapcsolja össze.**

A mikroszámítógép processzorát és főtárolóját a sínrendszerrel együtt a számítógép alaplapján szokták elhelyezni.

3.2.2 Perifériák

A perifériák a számítógépnek adatbevitelre, illetve -kivitelre vagy átmeneti külső adat- és programtárolásra szolgáló egységei.

Ezek tipikusan a perifériavezérlő egységeken keresztül kapcsolódnak a központi egység rendszerbuszához.

A leggyakrabban használt perifériatípusok:

Beviteli perifériák:

- **billentyűzet,**
- **egér,**
- **fényceruza,**
- **optikai letapogató (scanner).**

Billentyűzet

Fényceruza

Szkenner

Kiviteli perifériák:

- **monitor (display),**
- **nyomtatók,**
- **rajzolók,**
- **akusztikus kiviteli eszközök.**

Monitor

Nyomtató

Rajzgép

Hangfal

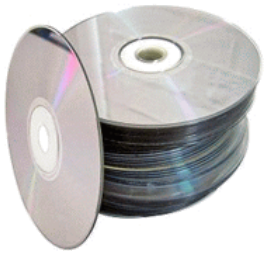
Adattárolásra és bevitelre/kivitelre is használt perifériák:

- **mágneslemezes tár (merevlemezes – Winchester, vagy hajlékonylemezes – floppydiszk),**
- **mágnesszalagos tár (sztrímer),**
- **optikai lemezes tár (CD-ROM, DVD-ROM, DVD-RAM).**

Merevlemez

Flopi

Sztrímer



CD, DVD

3.3 A mikroszámítógép működése

3.3.1 Órajel és gépi ciklus

A mikroszámítógép folyamatos működését periodikusan kiadott jelek – órajelek – biztosítják, amelyek egyrészt az ütemnek megfelelően engedélyezik az adatok jeleinek belépését az áramkörökbe, másrészt szinkronizálják az áramkörök állapotváltozásait. Ezekkel az órajelekkel találkozhatunk akkor, ha egy processzorról például azt olvassuk egy hirdetésben, hogy 400 MHz-es.

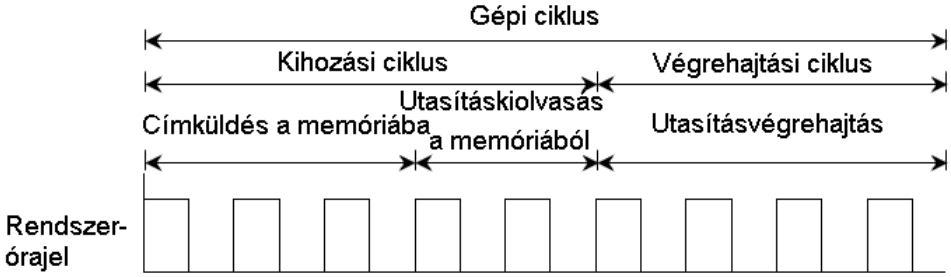
Egy gépi utasítás végrehajtása általában több óraciklus alatt megy végbe.

A gépi ciklus az az időtartam, amely egy processzor-alapművelet végrehajtásához szükséges.

Egy processzor-alapművelet: a belső tárban tárolt adatok kiküldése (kikapuzása) a belső buszokra, a kiválasztott alapművelet (pl. összeadás, logikai művelet) végrehajtása a processzor műveletvégző egységeiben és az eredmények beküldése (bekapuzása) – a buszon keresztül – a kijelölt belső tárolóba.

Egy gépi ciklus tehát két fázisból áll (lásd az alábbi ábrát):

- **az utasítás-kihozási ciklus, mely alatt (az utasításszámláló által) meghatározott tárolóhelyről a processzor kiolvassa az utasítást;**
- **az utasítás végrehajtásának ciklusa.**



A gépi ciklus fázisai

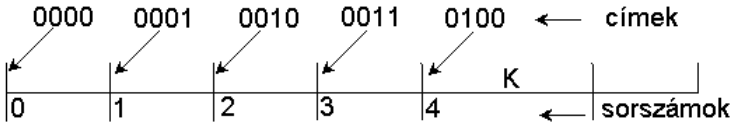
3.3.2 Tárak, tárolók (memória)

A számítógép az adatok és az utasítások bitjeit egy, az adott gépre jellemző rekeszekben tárolja. A rekeszeket felépítő bitek száma a mikroszámítógép típusára jellemző állandó szám. A legelterjedtebb a bájt, ez 8 bitből áll.

Tár minden olyan eszköz, amely adatokat őriz meg a későbbi kiolvasás céljára. A tár az elektronikus berendezéseknek az a része, amelyben az adatok, a program és az eredmény tárolása történik, általában elektromos vagy mágneses működés alapján.

Az adatok beírása és kiolvasása a tárolókból a címezhetőség elve alapján történik. Ez azt jelenti, hogy minden egyes rekesznek (bájtnak) van egy sorszáma, mely a gépi utasításokban a rekeszt egyértelműen azonosítja.

A rekeszek vagy bájtok sorszámát bináris formában a rekesz vagy bájt abszolút (vagy fizikai, illetve lineáris) címének nevezzük. (A sorszámozást 0-tól kezdjük.)



A tároló rekeszek címei

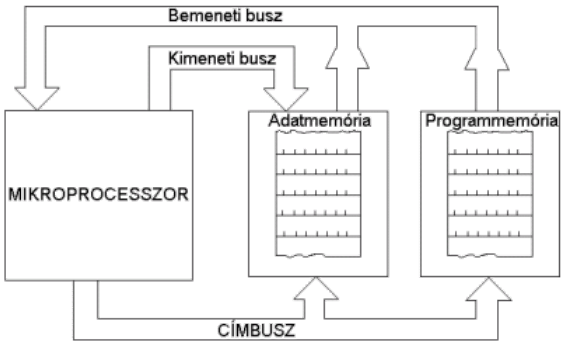
A tárkapacitást a tárolható rekeszek, bájtok vagy bitek számával mérik.

A legfontosabb mértékegységek:

- **a Kbájt (kilobájt), mely 2^{10} = 1024 bájtot jelent (például 4 Kbájt = 2^{10}•4 bájt = 4096 bájt)**
- **a Mbájt (Megabájt), mely 2^{20} bájtot jelent.**

A tárolókat a buszrendszerrel csatlakoztathatjuk a processzorhoz.

Ennek elvi megoldását mutatja be az alábbi ábra.



A tárak csatlakoztatása a processzorhoz

A különböző tárolókat

- **az adatok elérése,**
- **az adatok átírhatósága,**
- **a tárolók funkciója**
- **és fizikai működési elvük**

szerint osztályozhatjuk.

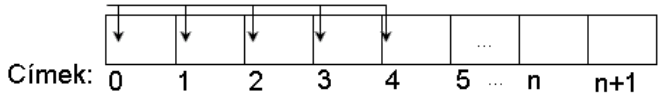
Az adatok elérése szempontjából megkülönböztetünk

- **soros és**
- **közvetlen hozzáférésű**

tárolókat.

A soros tárolóban a tárolóhelyek időben egymást követően válnak hozzáférhetővé az információ beírásakor és kivitelekor. Egy rekesz hozzáférési ideje függ az utoljára hivatkozott (kikeresett) rekesz és a keresett rekesz viszonylagos címeitől. Erre példa a hetvenes évek fő háttérmemória típusa, a mágnesszalag.

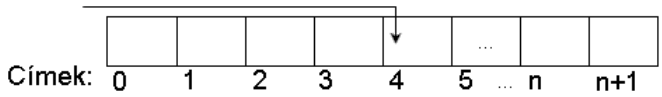
A soros hozzáférésű tárolónál, ha ki akarjuk olvasni pl. a 4. című adatot, akkor sorba végig kell olvasni a 0, 1, 2, 3 című adatokat is.



A soros hozzáférésű tároló működése

A közvetlen hozzáférésű tárolóban az adatokat felvitelük sorrendjétől függetlenül lehet elérni, bármelyik tárolt adat beírása körülbelül azonos idő alatt történik meg, valamennyi rekesz hozzáférési ideje lényegében egyenlő. Közvetlen hozzáférésű tárolóra példa a mágneslemez (a Winchester, vagy a floppy diszk).

A közvetlen hozzáférésű tárolóban azonnal kiolvashatjuk a 4. című adatot, az előtte elhelyezett adatoktól függetlenül.



A közvetlen hozzáférésű tároló működése

Az asszociatív tároló a tartalma alapján címezhető. Az asszociatív tárolóban tehát a tároló bemenetére egy konkrét adatértéket írunk, és a tároló kimenetén megkapjuk, hogy ezt az adatértéket a tároló tartalmazza-e vagy sem. Vagyis ez olyan tárolótípus, melynél a rekeszekben tárolt információ nem a rekeszek címei alapján, hanem a tárolt információ egy része (keresési kulcs) alapján olvasható ki.

Az asszociatív tár gyors működésű, a keresési kulcs összehasonlítása a rekeszekben tárolt tartalommal az összes rekeszre nézve egyidejűleg történik meg. Ez természetesen csak úgy valósítható meg, hogy az asszociatív tár minden rekeszhez külön-külön összehasonlító áramköröket tartalmaz. Ezért ez a tárolótípus a többihez képest lényegesen drágább.

A tárban tárolt adatok átírhatósága szempontjából megkülönböztetünk:

- **csak olvasható tárakat (ROM = Read Only Memory)**
- **írható-olvasható (programozható) tárakat (RAM = Random Access Memory)**

Az adatok rögzítése – beírása – a memóriában lehet végleges (fix memóriák, például: ROM), vagymódosítható (például: RAM, EPROM).

A ROM típusú memóriából az egyszer – általában technológiai módszerekkel – beírt információ csak kiolvasható, tartalma nem változtatható meg.

Az írható/olvasható tárolók(RAM) tartalma módosítható, azaz író üzemmódban a külső adatokkal azonossá tehető a memóriarekeszek tartalma.

Az írható/olvasható tárolók lehetnek dinamikus vagy statikus működésűek. A dinamikus tároló a beírt adatokat csak periodikus jelregenerálással képes tetszés szerinti ideig tárolni. A statikus tároló a beírt adatokat elvileg korlátlan ideig megőrzi. Természetesen mindkét eset folyamatos áramellátást feltételez, a tápfeszültség kimaradása esetén mind a dinamikus, mind a statikus tároló tartalmát elveszti.

Az újraprogramozható „csak olvasható" tárak a számítógép üzemszerű működése során csak olvashatóak. Ugyanakkor speciális eszközökkel (például EPROM esetében ultraibolya fénnyel) ezeknek a tárolóknak a tartalma törölhető, és ezt követően új adatokkal feltölthető.

A feladataik szerint a tárolók lehetnek:

- **operatív tárak (belső tár, főtár, központi tár, kissé pontatlanul RAM) és**
- **háttértárak (külső tárak).**

Az operatív tárban tároljuk azt a programot, melyet a számítógép aktuálisan végrehajt. A többi programot a háttértárakon helyezzük el, és akkor olvassuk be az operatív tárba, ha a megfelelő programot végre akarjuk hajtatni. (Ezt a program futtatásának is nevezik.)

Működési elvük, megvalósításuk alapján az alábbi főbb tártípusok különböztethetők meg:

- **mágneslemezes tár (floppy diszk, Winchester),**
- **mágnesszalagos tár,**
- **félvezető tár,**
- **optikai tár.**

Háttértárként jelenleg még döntően a mágneslemezes tárak használatosak, de terjedőben vannak az optikai tárak is (csak olvasható ill. írható/olvasható tárként).

Archív tárként, mentési céllal speciálisan kialakított mágnesszalagos tárak (sztrímerek) használhatóak előnyösen.

A félvezető tárolók alapeleme a két stabil állapotú billenőkör a flip-flop, amely 1 bit információ tárolására alkalmas. A tárolt bit értéke lehet "0" vagy "1". Állapotát csak külső jel hatására változtatja meg.

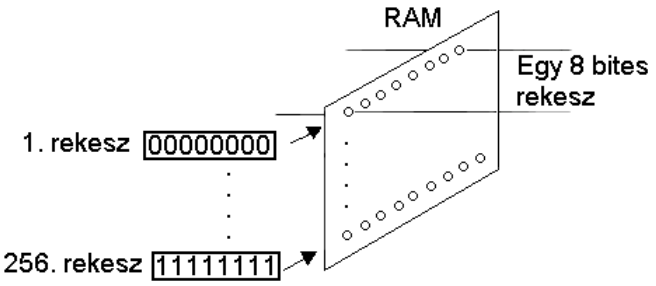
3.3.3 Memóriacímzés

Az adatokat a memóriából kiolvashatóvá, vagyis kikereshetővé, és ezért megcímezhetővé kell tenni. A számítógépeken a legkisebb címezhető adategység (amit a memóriában elérhetünk) – mint már említettük – a rekesz, ami általában egy bájt. Ennek megfelelően minden bájtnak saját címe van, ez a memórián belül a bájt sorszáma (az első bájtnak van a legalacsonyabb címe: zérus, az utolsó bájtnak van a legmagasabb címe). A legmagasabb cím értékét a memória mérete határozza meg.

A tárolóknak azt a részét, melyet a tárolás, az átvitel, a feldolgozás szempontjából a számítógép egy egységként kezel, szónak nevezzük. Ez lehet 1, 2, 4, 8 bájt, azaz 8, 16, 32 vagy 64 bitből állhat.

A következő példákban a memória címzését, az adatok kiolvasását, valamint a gépi utasítás végrehajtását leegyszerűsítve mutatjuk be (például 16 bites memóriacímek), koncentrálva az eljárás lényegére. Természetesen ez a mai számítógépekben jóval bonyolultabb módon történik. Ennek ellenére az eljárás elve napjainkban is lényegében azonos.

Először – az egyszerűség kedvéért – tételezzük fel, hogy a memória egy chipből álló részekből, vagyis egy chipes memóriamodulokból épül fel (lásd az alábbi ábrát).



Egy chipből álló memóriamodul

Ekkor chipen belüli rekeszek címzéséhez szükséges címbitek száma a chip méretétől függ. Például a fenti ábrán látható 256 memóriarekesz kapacitás esetén a rekesz címe 8 bitből áll.

Különböző számrendszerekben ábrázolva:

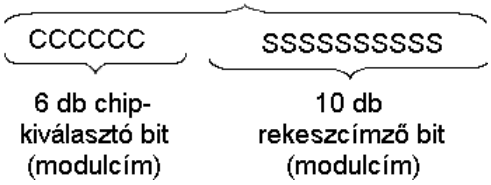
- **a legkisebb cím = 0000 0000_2 = 00_{16} = 00_{10}**
- **a legnagyobb cím = 1111 1111_2= FF_{16} = 255_{10}**

Egy írható/olvasható memóriamodul általában több chipből áll. Ezért minden memóriacím tartalmazza egyrészt a szükséges chipek, másrészt az aktuális rekesz kiválasztására vonatkozó adatokat. A chipkiválasztó bitek kiválasztják a memórián belül azt a memóriamodult, amely azokat a chipeket tartalmazza, amelyben az adatot tárolják. A rekesz kiválasztására szolgáló címbitek pedig egy-egy rekeszt azonosítanak a memóriamodulon belül.

Például, ha egy mikroszámítógép 65536_{10} (2^{16}) memóriarekesz címzésére alkalmas, akkor 16 bit kell a legnagyobb memóriacím kifejezéséhez ($FFFF_{16}$)

Ha a memóriachip 1024 bites, akkor egy rekesz címzéséhez 10 bitet kell felhasználni, így a chipek kiválasztására 6 bitünk maradt, ezzel (2^6) 64 memóriamodult lehet kiválasztani.

Így egy 16 bites memóriacím a következő részekből áll:



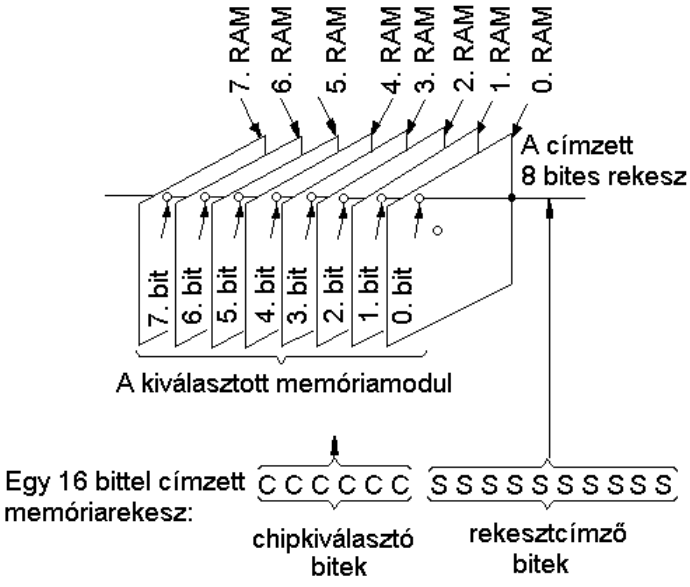
A tervezőtől függ az, hogy a címzéshez rendelkezésre álló bitek hogyan oszlanak meg a rekeszek és a chipeket tartalmazó memóriamodulok címzése között.

Egy konkrét memóriatípus esetén a következő döntéseket kell hozni:

- **egy chip hány bitből áll, ez meghatározza a rekeszek címzéséhez szükséges bitek számát,**
- **egy rekesz hány bitből áll, ez meghatározza hogy egy memóriamodult hány chipből kell felépíteni,**
- **a memória maximálisan hány memóriamodult tartalmazhat. Ez meghatározza a modulok címzéséhez szükséges chipkiválasztó bitek számát.**

Ha egy rekesz 8 bitből, azaz 1 bájtból áll (lásd az alábbi ábrát), akkor az előző példában bemutatott memória maximális kapacitása

8 • 1024 • 64 bit = 64 Kbájt.



Egy 16 bittel címzett rekesz azonosítása a memóriában

Példa: címzés 16 bittel.

A rekesz címe : 0110101110_2 = $1AE_{16}$

A chip címe : 000111_2 = 07_{16}

chip **rekesz**

07_{16} **$1AE_{16}$**

A teljes cím : 0 0 0 1 1 1 0 1 1 0 1 0 1 1 1 0 = $1DAE_{16}$

1 D A E

3.3.4 A gépi utasítások végrehajtásának lépései

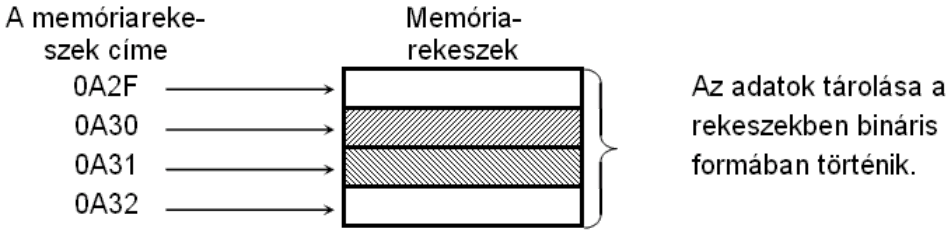
Egy, a processzor által végrehajtható számítógépes program gépi utasítások sorozatából áll, melyeket bináris formában tárolunk a memóriában. Minden gépi utasítást – összeadás, szorzás, vezérlés-átadás stb. – további elemi lépésekre bontva hajt végre a számítógép.

Vizsgáljuk meg, hogy például egy bináris összeadás elvégzéséhez milyen elemi lépések végrehajtása szükséges.

Legyen a feladat az, hogy a memória $0A30_{16}$ címén tárolt számhoz hozzá kell adni a $0A31_{16}$ címén tárolt számot, és az eredményt a $0A30_{16}$ címen kell tárolni.

Ha a memóriából az összeadásnak megfelelő gépi utasítást már kiolvasta és értelmezte a processzor, akkor a feladat végrehajtásához a következő lépések szükségesek:

1. **Az utasításban lévő cím alapján azonosítani kell azt a memóriarekeszcímet, amelyen az első összeadandó van**
2. **Az azonosított memóriarekesz tartalmát át kell vinni a mikroprocesszorba.**
3. **Azonosítani kell azt a memóriarekeszcímet, amelyen a második összeadandó tárolva van.**
4. **A második lépésben átvitt szóhoz hozzá kell adni a harmadik lépésben kikeresett szót.**
5. **Azonosítani kell annak a memóriarekesznek a címét, amelyben az összeget tárolni fogjuk.**
6. **Át kell vinni az összeadási művelet eredményét az ötödik lépésben kikeresett memóriarekeszbe.**



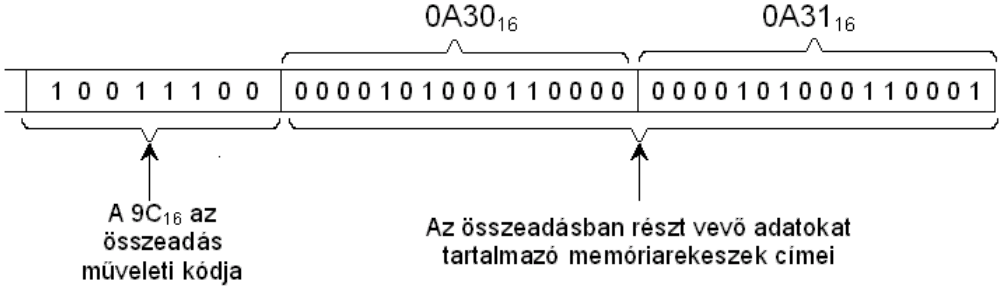
Az adatok elhelyezkedése a memóriában

Az összeadáshoz tehát hat elemi lépés szükséges.

A példánkban az összeadás gépi utasításkódja 5 bájtot foglal el (lásd III.10. sz. ábra).

Ez tartalmazza

- **a műveleti kódot (Mit kell végrehajtani?),**
- **az adatok címeit a memóriában (Mivel kell a műveletet elvégezni?).**



Az összeadás gépi utasításkódjának ábrázolása

Példánk annyiban általánosítható, hogy a számítógép gépi utasítása mindig tartalmaz egy műveletikód részt (mely az elvégzendő műveletet azonosítja) és egy címrészt (mely a műveletben részt vevő adatok címét tartalmazza).

3.4 A mikroszámítógép szoftvere

Az előző fejezetben láttuk, hogy a számítógép a számára bináris formában megadott gépi kódú utasításokat képes csak végrehajtani.

A számítógépprogram bináris számsorozat alakjában is megírható, de ekkor igen nagy az olyan hibák valószínűsége (például 0 vagy 1 felcserélése), amelyek igen nehezen találhatók meg.

A programot hexadecimális számjegyekkel írva kevesebb a lehetőség hiba elkövetésére, mivel négy bináris számjegy (négyjegyű bináris szám) megadható egy hexadecimális számjeggyel.

A hexadecimális számjegyekkel megjelenített programban a hibák is könnyebben megtalálhatók. Egyszerűbb egy rossz helyre írt hexadecimális számjegyet észrevenni, mint egy szemkápráztató bináris mintában a téves 0 vagy 1-et felismerni.

Végül azonban a programot át kell alakítani bináris jelsorozattá, mivel a memóriában így tárolják, és a processzor csak ebben a formában képes a program végrehajtására.

Az első számítógépeket (ezek még nem voltak a mai értelemben vett számítógépek) még áramköri szinten a vezetékek „dugaszolásával" programozták, később a gépi kódú utasításokat bináris formában lyukszalagra kellett rögzíteni. (A lyukszalagon egy lyuk léte vagy nemléte felelt meg a bináris 1 vagy 0 jelnek.) Ennek során némi könnyítést jelentett, hogy a lyukasztógép klaviatúráján nyolcas számrendszerben (3 bit egy nyolcas számrendszerbeli számjegy) kellett a számokat beütni.

A számítógépek elterjedésével egyre nyilvánvalóbbá vált, hogy ez a módszer nem elég hatékony. Először az ún. assembly programozási nyelv alakult ki, ahol már a műveleti kódokat az értelmüknek megfelelő angol szavak rövidítésével jelölték (mnemonik) és a gépi utasításban lévő címeknek is lehetett ún. szimbolikus nevet (címet) adni.

Így például az előző fejezetben bemutatott összeadás bináris kódja helyett a programozók az utasításokat szimbolikus formában fogalmazhatták meg.

gépi kód	10011100	0000101000110000 | 0000101000110001
assembly nyelv	ADD	ADAT1, ADAT2

Összeadási utasítás assembly nyelven és gépi kódban

Felmerül a kérdés, hogy a szimbolikus műveleti kódot és címeket hogyan fogja megérteni a számítógép, ha a processzor csak a bitek nyelvén ért?

A válasz (ha már rájöttünk a megoldásra) tulajdonképpen egyszerű: egy olyan programot kell készítenünk, mely ha beolvassa az ADD betűknek megfelelő karakterek bináris értékét, akkor ehhez mindig az összeadás bináris műveleti kódját rendeli hozzá, és ezt a tárolóban letárolja, és ehhez hasonlóan jár el az ADAT1, ADAT2 címek esetében is.

Azokat a programokat, melyek a szimbolikus programnyelveken megfogalmazott utasításokból a processzor számára gépi kódként értelmezhető bitsorozatot állítanak elő, fordítóprogramoknak nevezzük.

Tehát az előző fejezet utolsó ábráján bemutatott bináris formában tárolt gépi kódú utasítást a fordítóprogram állítja elő, a

$$\text{ADD} \Rightarrow 9C_{16} = 100111000$$
$$\text{ADAT1} \Rightarrow 0A30_{16} = 0000101000110000$$
$$\text{ADAT2} \Rightarrow 0A31_{16} = 0000101000110001$$

hozzárendelések segítségével.

A szimbolikus programnyelvek segítségével felgyorsult a programozás és az emberek egyre több feladatot fordítottak le a számítógépek nyelvére. Egyre jobban kihasználták a számítógépnek azt a képességét is, hogy a programokat lehetőség van a számítógép tárolóeszközein megőrizni, és szükség esetén ismételten felhasználni. Ha például valaki egy másodfokú egyenlet megoldását már beprogramozta és ezt tárolták, ezt bárki átvehette, végrehajtathatta a számítógéppel vagy saját programjába beépíthette.

Így a számítógépet működtető programok összességében, vagyis a szoftverben évről-évre egyre több szellemi érték halmozódott fel.

Az 1950-es évektől kezdve egyre-másra jöttek létre az emberi nyelvhez egyre inkább közelítő, ún. magasszintű programnyelvek. (FORTRAN 1954, ALGOL 60, COBOL 1959, BASIC 1964, PASCAL 1968, C 1974), melyekkel már sokkal könnyebben lehetett gyorsan áttekinthető és hibamentes programokat írni. Természetesen ennek az ára az volt – mivel a magasszintű programnyelveknél egy utasítás általában már több gépi kódú utasításra lesz lefordítva –, hogy a fordítóprogramok is egyre bonyolultabbá váltak.

Az egyre több feladat megoldására képes szoftver egyes részei a fejlődés során a programok felhasználási célja szerint is kezdtek elkülönülni.

Így alakult ki a gép alapműködtetését a felhasználás céljától függetlenül biztosító, általános (például egy eszköz hibás működésének behatárolását, erről a felhasználó tájékoztatását is elvégző) programok együttese, az operációs rendszer.

Ezt a számítógépet gyártó cég általában a géppel együtt szállította.

Az operációs rendszerek a fejlődés során egyre több segédprogrammal (utility) egészültek ki (másolók, rendezők stb.), melyek tovább segítették a számítógépet felhasználók munkáját.

Még az 1970-es években is jellemző volt, hogy ha egy szervezet egy meghatározott célra egy számítógépet vásárolt, akkor az adott feladatnak megfelelő programok elkészítését is vállalnia kellett.

Az egy-egy konkrét alkalmazás céljára kifejlesztett programokat felhasználói programoknak nevezzük.

A felhasználói programok fejlesztésében az idők folyamán célszerű munkamegosztás alakult ki az egyes speciális szoftver szakterületekre szakosodott cégek és a számítógépet alkalmazó szervezetek között. Nyilvánvalóan gazdaságosabb volt, ha az egyes sajátos, több felhasználónál is azonos módon jelentkező feladatokat megoldó szoftvereket erre szakosodott szoftverházak fejlesztik ki és gondozzák.

Erre az egyik legjellemzőbb példa azon szoftverek kialakulása, melyekkel egységes módon meg lehetett oldani a felhasználók adattárolással és visszakereséssel kapcsolatos feladatait.

A felhasználás szakterülete szerint specializálódott szoftverekre néhány további példa:

- **irodaautomatizálási (office) szoftverek,**
- **mérnöki tervezőrendszerek (CAD),**
- **elektronikus levelező szoftverek stb.**

Az eddigieket összefoglalva a mikroszámítógépet működőképessé tevő szoftverek fontosabb típusai a következők:

- **az operációs rendszer és segédprogramjai,**
- **az adatbáziskezelő rendszer,**
- **a programnyelvek fordító és könyvtárkezelő programjai,**
- **az általánosan használt gyári programcsomagok (szövegszerkesztők, elektronikus levelezőprogramok stb.),**
- **speciális célú gyári programcsomagok (pl. a számítógépes tervező, CAD rendszer),**
- **a felhasználó által írt programok.**

3.5 Ellenőrző kérdések

1. Milyen funkcionális részekből épül fel a mikroszámítógép?
2. Mi a feladata a be/kimeneti egységeknek?
3. Mi a feladata a mikroprocesszornak és a tárnak?
4. Mivel lehet biztosítani a kommunikációs kapcsolatokat a mikroszámítógép funkcionális egységei között?
5. Milyen részegységeket tartalmaz a mikroszámítógépek alaplapja?
6. Mit nevezünk perifériának?
7. Sorolja fel a beviteli perifériákat!
8. Sorolja fel a kiviteli perifériákat!

9. Sorolja fel az adattárolásra használt perifériákat!
10. Mit biztosítanak az órajelek?
11. Mi a gépi ciklus és milyen részekből áll?
12. Mit nevezünk tárnak?
13. Mit nevezünk címnek?
14. Mit nevezünk bájtnak, Kbájtnak és Mbájtnak?
15. Milyen szempontok szerint osztályozhatjuk a mikroszámítógépek tárolóit?
16. Mi jellemzi a soros tárolóeszközök működését?
17. Mi jellemzi a közvetlen hozzáférésű tárolókat?
18. Hogyan működik az asszociatív tár?
19. Milyen tárolókat különböztetünk meg a tárolt adatok átírhatóságát figyelembe véve?
20. Mi a ROM és a RAM?

21. Mi a különbség a statikus és a dinamikus tárak között?
22. Mi a feladata az operatív tárnak?
23. Milyen elven működő tárolótípust használunk háttértárként?
24. Mire használhatók a sztrímerek?
25. Mi határozza meg a tárolóban a legkisebb és legnagyobb címet?
26. Milyen részekből épül fel egy memóriacím?
27. Mit nevezünk memóriamodulnak?
28. 1024 bites memóriachip esetében hány bitet kell felhasználni a memóriarekesz címzéséhez?
29. Miből áll a processzor által végrehajtható program?
30. Milyen elemi lépésekből áll egy gépi utasítású összeadás végrehajtása?
31. Milyen részekből áll egy gépi kódú utasítás?
32. Mi a különbség a gépi kódú és az Assembly nyelvű utasítás között?
33. Mit nevezünk fordítóprogramnak?
34. Mi jellemzi a magasszintű programnyelveket?
35. Mit értünk operációs rendszeren?
36. Mit nevezünk felhasználói programnak?
37. Sorolja fel a mikroszámítógépes szoftverek fontosabb típusait!

4 Hardver alapismeretek

4.1 Mikroprocesszor

4.1.1 Bevezetés

A legtöbb egychipes mikroprocesszort MOS-technológiával állítják elő, amivel a kívánt nagy bonyolultsági fok (LSI, VLSI) elérhető. Cél, hogy egyetlen chipen belül minél komplexebb processzor legyen megvalósítható.

4.1.2 A mikroprocesszor feladata

A mikroprocesszor olyan nagy bonyolultságú (LSI, sőt VLSI) félvezető eszköz, amely a digitális számítógép központi egységének (CPU) a feladatait végzi el. A mikroprocesszor dekódolja és végrehajtja az utasításokat, vezérli a műveletek elvégzéséhez szükséges belső adatforgalmat és a csatlakozó perifériális berendezések tevékenységét.

4.1.3 A mikroprocesszor funkcionális egységei

A mikroprocesszoroknak három alapvető funkcionális blokkjuk van:

- **a regiszterek,**
- **az aritmetikai logikai egység (ALU = Arithmetical Logical Unit),**
- **a vezérlőegység.**

Ezeket a processzor belső buszrendszere köti össze.

4.1.3.1 Regiszterek

Egy CPU általában több gyors működésű tárolót, úgynevezett regisztert tartalmaz abból a célból, hogy a memóriából kihozott adatokat tárolja. Ezek tehát gyors írható-olvasható munkatárak. A különböző regiszterek meghatározott feladatokhoz vannak hozzárendelve. Ezek elvégzése közben a szükséges adatokat átmenetileg – rövid ideig – a regiszterekben tároljuk.

Egy regiszter tárolókapacitása többnyire egy szó, a regiszterek hosszát a benne tárolható bitek, bájtok vagy szavak számával mérik.

A regiszterek egy része szigorúan meghatározott feladatokhoz van hozzárendelve, és a felhasználói programok által nem használhatók. Ezeket rendszerregisztereknek nevezzük.

A rendszer (speciális funkciójú) regiszterek mellett még adattárolási, indexelési, címzési stb. célokra használatosak az ún. általános (univerzális) regiszterek. Ezeket a felhasználói programok is alkalmazhatják.

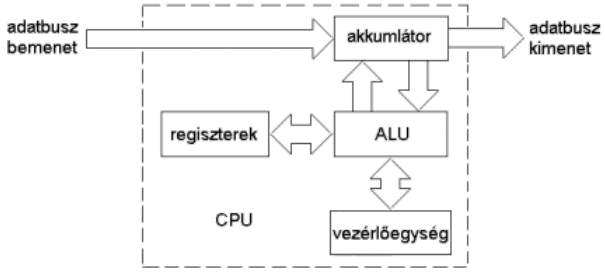
A regiszterek általában statikus memóriaelemek (például flip-flop áramkörök) valamilyen rendszeréből épülnek fel. (Ahol az összes regisztert statikus elemekből építik fel, az óraimpulzusok sorozatát leállíthatjuk anélkül, hogy a tárolt információkban veszteség keletkeznék.)

4.1.3.1.1 Tipikus regiszterek

Akkumulátor (Accumulator = A)

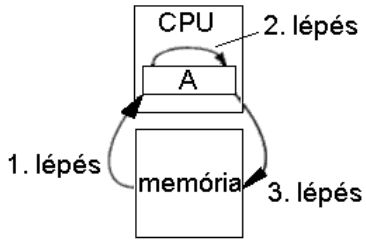
Az akkumulátor a számítógép megkülönböztetett számítási regisztere. Az aritmetikai és logikai műveletek operandusait, vagyis a műveletek tárgyát képező mennyiségeket, illetve ezeknek az eredményeit a CPU központi regiszterében, az akkumulátorában tároljuk.

A különböző műveleteket a CPU az akkumulátorban tárolt adatokon, vagyis az akkumulátor tartalmán végzi el (alábbi ábra).



Adatszófolyam a mikroprocesszoron keresztül

Az alábbi ábrán látható, hogy az adatok kihozása a memóriából (olvasás) és az akkumulátorban történő elhelyezése, a művelet elvégzése és az eredmény bevitele (írás) a memóriába, három elemi lépést jelent.



A mikroprocesszor (CPU) akkumulátora és a memória között létrejövő adatforgalom lépései

Azt az időtartamot, ami alatt az adatokhoz hozzáférünk, vagyis ami a címkijelöléstől az olvasás, vagy írás műveletek befejezéséig eltelik, hozzáférési, illetve elérési időnek nevezzük.

A jelenlegi számítógépekben a regiszterek hozzáférési ideje pár nanoszekundum. (1 nanoszekundum a másodperc ezer milliomod része (10^{-9} sec), ez alatt az idő alatt a fény mindössze 30 cm-t tesz meg.)

A CPU sebességét érdemes szemléletesen is összehasonlítani például az emberével. Ha a regiszter hozzáférési idejét 1 másodpercnek feleltetjük meg, akkor ehhez viszonyítva az az időtartam, amíg az ember leüt egy billentyűt a számítógép klaviatúráján kb. 30 év.

Megjegyzés: A legtöbb CPU-ban több akkumulátor, vagy akkumulátor típusú regiszter is van. Ekkor megfelelő szervezéssel elérhető, hogy egy memória hozzáférési idő alatt több CPU művelet is elvégezhető legyen.

A korszerű számítógépekben rendszerint egy vagy több regiszter-tömb is megtalálható 8, 16, ... max. 512 darab általános célú regiszterrel, melyet regisztertárnak nevezünk. E regiszterekben helyezhetők el ideiglenesen a műveletek operandusai és eredményei. Így csökkenthető a tárhoz fordulások száma, ezáltal növelhető az utasítás végrehajtás sebessége.

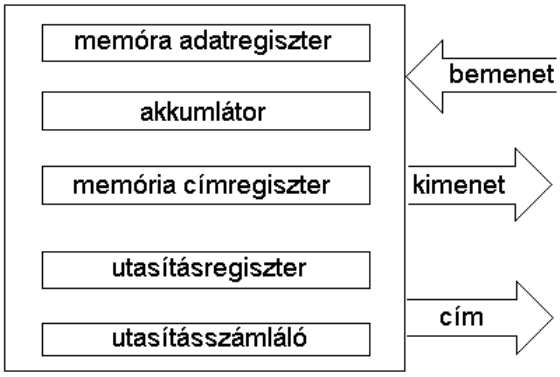
Utasításregiszter(Instruction Register = IR),
Az utasításkódok kezeléséhez szüksége van a CPU-nak az utasításregiszterre, amelyben a program aktuálisan végrehajtott utasítását átmenetileg tárolja.

Utasításszámláló (Program Counter = PC)
Az utasításszámláló regiszter a soron következő utasítás címét tárolja. Az utasításszámláló tartalmát a program maga is módosíthatja. (Ez történik például a program által végrehajtott elágazási utasítások esetén.)

A program az utasításkódok címeinek sorrendjében hajtódik végre. Vagyis az utasításszámláló által címzett első memóriarekesz elérésekor kihozunk a memóriából egy utasítást, melynek következtében az utasításszámláló tartalma a kihozott utasítás hosszával megnő, és így a memóriának most azt a rekeszét címzi, ahol a program szerinti következő utasításkód található.

Memória címregiszter (Memory Adress Register = MAR)
Az adatok kiolvasásakor vagy beírásakor azonosított memóriarekesz címét a memória címregiszter tárolja. Ennek mérete függ a mikroprocesszor által címezhető memóriakapacitástól.

Memória adatregiszter (Memory Data Register = MDR)
A memória adatregiszter a memóriából kiolvasott vagy beírni kívánt adatok átmeneti tárolója, úgynevezett pufferregiszter (buffer register). Általában a pufferregiszter feladata, hogy az adatok áramlási ütemének egyenetlenségeit, vagy az események időbeli előfordulásának különbségét kiegyenlítse abban az esetben, amikor az adatokat különböző sebességű hardveregységek között kell mozgatni. Az MDR a CPU és a főtár sebességének különbségét egyenlíti ki, mivel a főtár RAM memóriája kb. egy nagyságrenddel lassúbb a regiszterénél.



Egy egyszerű mikroszámítógép tipikus regiszterei

A CPU-nak még két fontos regisztere van: az állapotregiszter és a veremmutató regiszter, ezeket a felhasználásukkal foglalkozó ebben és ebben a fejezetben fogjuk bemutatni.

4.1.3.1.2 A regiszterek használata, az utasításvégrehajtás elemi lépései

Hogyan használjuk a CPU regisztereit?

Annak érdekében, hogy a CPU regisztereinek használatát megértsük, még egyszer végigkövetjük az összeadás elemi lépéseit, lépésről-lépésre részletesebben bemutatva.

Figyeljük meg a regiszterek tartalmának változását az utasítások végrehajtásának menete során.

1. lépés: Az utasításszámlálóban található cím bekerül a memória címregiszterbe. Ez alapján kiolvassuk a memóriából az összeadó utasítás gépi kódját, ez pufferelésre kerül a memória adatregiszterben. Innen a következő ütemben átkerül az utasításregiszterbe.

2. lépés: Az utasításszámláló regiszter tartalmát megnöveljük a kiolvasott utasítás hosszával (így ez most a következő végrehajtandó utasításra mutat).

3. lépés: A CPU értelmezi az utasításregiszterben található utasítás kódját, megállapítja, hogy az összeadás műveletét kell végrehajtani.

4. lépés: A CPU az utasításregiszterben található címek alapján meghatározza az első összeadandó címét a memóriában, és ezt beírja a memória-címregiszterbe.

5. lépés: A CPU kiolvassa az első összeadandó értéket a memóriából, ez bekerül a memória-adatregiszterbe, majd onnan az akkumulátorba.

6. lépés: A CPU az utasításregiszterben található címek alapján meghatározza a második összeadandó címét a memóriában, és ezt beírja a memória-címregiszterbe.

7. lépés: A CPU kiolvassa a második összeadandót a memóriából, és ez bekerül a memória-adatregiszterbe.

8. lépés: A CPU műveletvégző egységében (aritmetikai és logikai egységben – ALU) megtörténik az összeadás, és ennek eredménye képződik az akkumulátorban

9. lépés: A művelet eredménye visszaírásra kerül a memóriába, az első összeadandó címének megfelelő helyre. (Az első összeadandó címe átkerül a memória címregiszterbe, az eredmény a memória-adatregiszterbe és innen kerül beírásra a memóriába.)

4.1.3.2 Aritmetikai logikai egység (ALU)

Az aritmetikai logikai egység a CPU-n belül a számítási és logikai műveleteket végzi el.

Az ALU alkalmas:

- **bináris összeadásra,**
- **Boole-algebrai (logikai) műveletek végzésére,**
- **komplemensképzésre,**
- **bitsorozatok léptetésére (bitenként jobbra, balra).**

Minden más aritmetikai és logikai művelet (például lebegőpontos számokkal végzett műveletek), amelynek elvégzése a CPU feladata, felbontható a fenti alapműveletekre.

Ennek megfelelően az ALU rendszerint fixpontos bináris összeadóból, komplementálóból, léptető regiszterből és a logikai műveleteket végző részből áll.

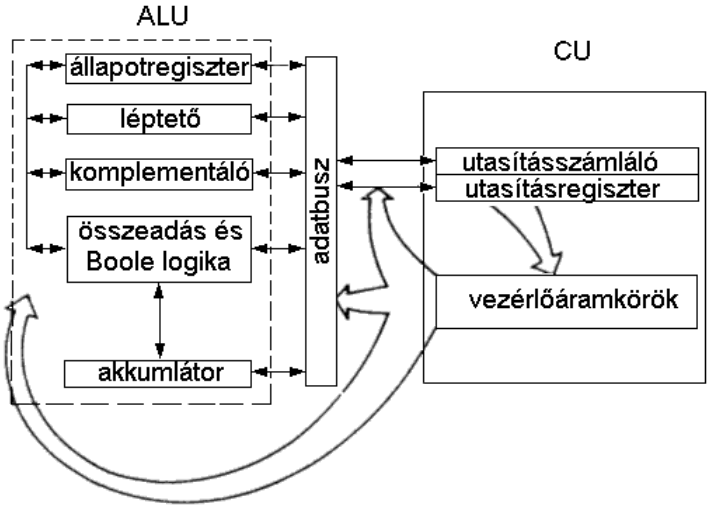
4.1.3.3 Vezérlőegység (CU)

A vezérlőegység funkciója bizonyos szempontból analóg a bábjátékos feladatával, aki úgy manipulálja a bábukat mozgató zsinegeket, hogy a bábu a megfelelő lépések szerint táncoljon. Egy mikroprocesszorban a regiszterek és az ALU a bábu, míg a vezérlőegység a bábjátékos.

A legtöbb mikroprocesszorchipben a vezérlőegység a teljes chipnek több mint a felét foglalja le, és rendkívül összetett funkcionális egység.

A vezérlőegység (Control Unit = CU) a CPU-n belül értelmezi az utasításokat és végrehajtásuk céljából összehangoltan vezérli a számítógép többi egységének működését úgy, hogy az események a programnak megfelelő helyes sorrendben és időben következzenek be. Biztosítja, hogy a megfelelő adatok, a megfelelő helyen és időben rendelkezésre álljanak. Indítja az áramkörök működését. Az utasítás-számláló tartalma alapján a vezérlőegység kiolvastatja a memóriából annak a rekesznek a tartalmát, amely a soron következő utasítást tárolja. Ezt az utasítást értelmezi, azaz az utasítás műveleti kódrésze alapján meghatározza, hogy sorrendben milyen műveletet, milyen elemi lépésekben kell végrehajtani. Az utasítás címrésze alapján értelmezi, hogy milyen címen találhatók a műveletben résztvevő adatok, vezérli ezek kiolvasását, és a megfelelő regiszterbe történő továbbításukat. Az ALU-val végrehajtatja a megfelelő műveletet, biztosítja az eredmény megőrzését és beállítja az utasításszámláló új tartalmát.

A vezérlőegység és az aritmetikai logikai egység működésének lényegét az alábbi ábra mutatja be.



A mikroprocesszor funkcionális ábrázolása

4.1.3.4 Állapotinformáció

Az állapotinformációval a számítógép pillanatnyi állapotát jellemezzük. Ezeket az állapotokat egy regiszter bitjeinek 0-ra vagy 1-re állításával kódoljuk.

Jellegzetes állapotinformációk például a következők:

előjelbit: **értéke a numerikus adatokon végzett műveletek eredményének előjele szerint kerül beállításra. A bit értéke 1, ha a művelet eredménye negatív.**

túlcsordulásbit: **a bit értéke 1, ha a numerikus adatokon végzett művelet eredménye túlcsordulás (azaz a művelet eredménye nem fér be a regiszterbe; pl. két szám szorzásánál)**

átvitelbit: **értéke 1, ha van átvitel az előjelbit helyi értékére**

nulla bit: **a bit értéke 1, ha a művelet eredménye 0**

paritásbit: **paritáshiba esetén kerül beállításra 1-re**

Az állapot-(státusz) információt az aritmetikai logikai egység hozza létre.

Az állapotbitek értéke általában feltételes elágaztató utasításokkal (lásd később) kérdezhető le és ettől függően változtatható meg az utasítás-végrehajtás sorrendje.

4.1.3.4.1 A jelzőbit

A jelzőbit (flag) olyan állapotinformáció, mely egy rendszer (hardverrendszer vagy program stb.) működési módjáról, működési feltételeiről vagy a működés közben előálló állapotokról (pl. eredmény=0, kiviteli/beviteli műveletek eredményes vagy eredménytelen végrehajtása stb.) ad felvilágosítást. Különbséget kell tenni hardverúton vagy szoftverúton (programváltozóként, illetve logikai változóként) megvalósított jelzőbitek között. A jelzőbitek beállíthatók, lekérdezhetők, illetve módosíthatók.

4.1.3.4.2 Állapotregiszter

Az állapotregiszter olyan regiszter, amely a központi egység vagy a perifériaeszközök állapotával kapcsolatos információkat tartalmazza.

A CPU állapotregiszterének bitjei (status register) az ALU műveleti eredményei alapján kerülnek beállításra (előjelbit, túlcsordulásbit stb.).

4.1.4 A processzor utasításkészlete, címzési eljárások

A processzorok egyik fontos jellemzője – ami egy-egy feladatra való alkalmasságukat döntően meghatározza –, hogy hány és milyen típusú gépi utasítás végrehajtására képesek. A processzor számára értelmezhető utasítások összességét nevezzük utasításkészletnek.

4.1.4.1 A gépi utasítás szerkezete

Az utasításkészletben különböző típusú gépi utasítások lehetnek, ezek felépítésükben is eltérhetnek.

Ezt fejezi ki többek közt az utasítás szerkezete, mely meghatározza a processzor számára, hogy a gépi utasítás mely részét hogyan kell értelmezni.

Általában az utasítások felépítése az alábbi ábra szerinti.

műveleti rész	módosító rész	címrész

A gépi utasítás felépítése

A műveleti rész (operation code) az elvégzendő feladatot határozza meg a processzor számára. (Tehát például megadja, hogy két számot össze kell adni.)

A címrész (address field) a művelet végrehajtásához szükséges adatok helyét határozza meg a számítógép tárolójában.

A módosító rész esetenként a műveleti rész, illetve a címzés értelmezéséhez ad kiegészítő információkat a processzor számára.

4.1.4.2 Címzési módszerek

A gépi utasítások abban is különböznek egymástól, hogy az adatokra való hivatkozást milyen formában tartalmazzák. Ezeket összefoglaló néven címzési eljárásoknak szokták nevezni.

Az utasítások címrésze általában nem a műveletben résztvevő adatok pontos rekeszsorszámát, az úgynevezett abszolút címet tartalmazza, hanem ezt a processzornak a címzési eljárásra utaló kiegészítő adatok alapján ki kell számítania.

Az elkövetkezendőkben a címzési eljárások közül áttekintjük

- **az abszolút címzést,**
- **a relatív címzést,**
- **a közvetlen adat (literális) címzést,**
- **a veremcímzést,**
- **a közvetett címzést,**
- **az indexelt címzést.**

4.1.4.2.1 Az abszolút címzés

Abszolút címzés esetében az utasítás címrésze, a műveletben részt vevő adatok, az operandusok valódi címét, azaz a memóriarekesz sorszámát tartalmazza, illetve a processzor egy regiszterére hivatkozik.

Példák:

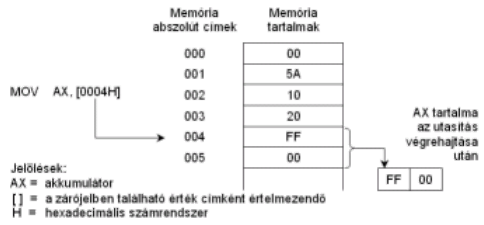
1. **MOV AX, DX**

Jelentése: A DX regiszter tartalmát vidd át az akkumulátorba.

2. **MOV AX, [0 0 4 H]**

Jelentése: A tároló 004 hexadecimálisan értelmezendő (ezt jelenti a H) címéről két bájtot tölts be az akkumulátorba.

Az abszolút címzést mutatja be a következő lapon az ábra, melynél a szerepeltetett számok hexadecimálisan jelölnek bináris értékeket.



A regiszterek abszolút címzését a programokban gyakran alkalmazzuk, az abszolút memóriacímzést viszont csak nagyon ritkán.

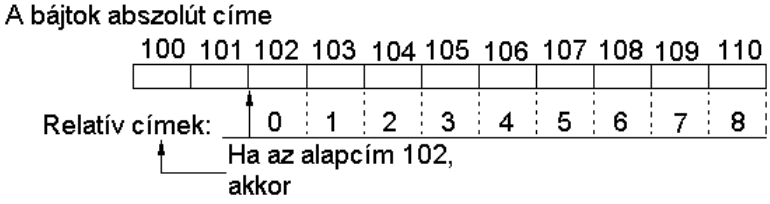
Abszolút memóriacímzés

Ennek az az oka, hogy a programokat nem mindig azonos memóriacímre töltjük be (hanem például onnan kezdődően, ahol van szabad hely a memóriában), így az egyes programbetöltési címtől függően a programutasításokban megcímzett adatok fizikai helye is megváltozik.

Az előző példánkban szerepelő Intel Assembly utasítás 004H címe is lényegében relatív címzést takar (szegmensen belüli relatív címzés).

4.1.4.2.2 Relatív címzés

Relatív címzés esetén a gépi utasítás-címrésze az adatnak valamilyen alapcímhez vagy báziscímhez viszonyított relatív címét tartalmazza. A tényleges fizikai memóriacímet ekkor a relatív cím és a báziscím összeadásával kapjuk meg.



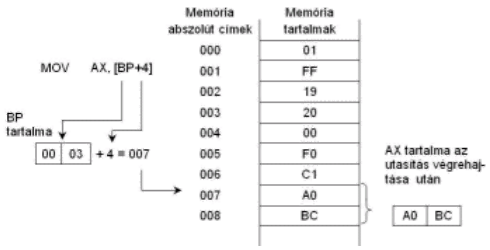
Az alapcím lehet

- **egy kijelölt regiszterben, melyet bázisregiszternek nevezünk,**
- **a program kezdetének (töltési helyének) a címe**
- **az éppen végrehajtott utasítás (ekkor a bázis címet az utasításszámláló regiszter tartalmazza).**

Példa:

MOV AX, [BP + 4]

Jelentése: **A BP bázisregiszterben található címértékhez adj hozzá 4-et, és ettől a címtől kezdve 2 bájtot tölts be az akkumulátorba.**



Relatív memóriacímzés

A relatív címzést alkalmazzuk akkor, ha biztosítani szeretnénk, hogy programjaink a memória tetszőleges helyére betöltve is futóképesek legyenek. Ezt elérhetjük azzal, hogy a program utasításaiban csak a program elejéhez (töltési címéhez) viszonyított relatív címeket alkalmazunk, és a program betöltését követően a BP bázisregiszterbe azonnal betöltjük a program elejének tárbeli címét.

4.1.4.2.3 Közvetlen adatcímzés

Ezt a címzési eljárást még közvetlen értékadó címzésnek, álcímzésnek és literálcímzésnek is szokták nevezni. A közvetlen adatcímzés lényege, hogy ebben az esetben az utasításban maga az az adat található meg, amellyel a műveletet végre kell hajtani.

Példák:

1. **MOV AX, 0**

Jelentése: **Tölts be 0-t az akkumulátorba.**

2. **ADD AX, 010110B**

Jelentése: **Add hozzá az akkumulátorban található értékhez a 010110 bináris számot.**

Ezt a címzési eljárást a programokban például a regiszterek konkrét számértékekkel konstansokkal való feltöltésére használhatjuk.

4.1.4.2.4 Veremcímzés (STACK-címzése)

Van egy olyan memóriacímzési mód, amelyet majdnem minden számítógép alkalmaz valamilyen formában, ez a veremcímzés.

A verem (stack) vagy egy regisztertár a CPU-ban, vagy egy kijelölt memóriaterület a főtárban, melyből az utoljára beírt adatokat lehet először kiolvasni.

Az adatforgalom a bemeneti sorrenddel ellenkező kimeneti sorrendű, így például az először beírt adatot lehet majd utoljára kiolvasni a veremből.

Az ilyen tulajdonságú memóriákat LIFO (LAST IN FIRST OUT = „utolsóként be, elsőként ki") szervezésűnek nevezzük. A verem egyik legfontosabb alkalmazása a CPU állapotának (az állapot- és az utasításszámláló regiszter tartalma) elmentése abban az esetben, ha egy program végrehajtását átmenetileg fel kell függeszteni.

4.1.4.2.4.1 Memóriaverem

Az a verem legegyszerűbb megvalósítása, amikor a főtár egy részét tartalékolják veremműveletekre. A verem megcímzése ekkor egy regiszterrel történik, ezt nevezik veremmutatónak (STACK POINTER).

4.1.4.2.4.2 Veremtár műveletek

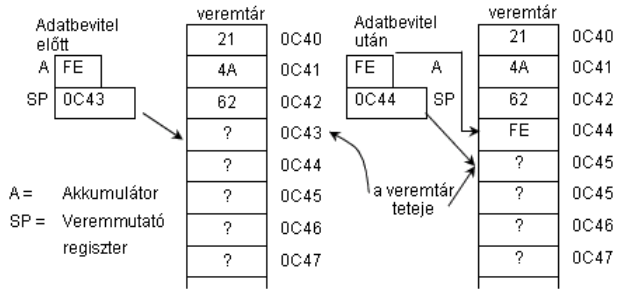
Két verem művelet van:

- **írás a verem felső részébe (a tetejére), ezt nevezik adatbevitelnek, vagy PUSH-nak;**
- **olvasás a verem felső részéből, ezt nevezik „POP"-nak, vagy adatkihozatalnak.**

Adatbevitel (PUSH)

A művelet a következő ábra segítségével jól követhető:

Az adatbevitel művelete (írás a verembe) azt eredményezi, hogy az akkumulátorban (vagy más CPU regiszterben) lévő adatokat beírják abba a memóriarekeszbe, amelyet a veremmutató címez meg. Ezt követően a veremmutató tartalma automatikusan megnő (inkrementálódik), hogy megcímezze a következő üres tárolóterületet a verem felső részén.



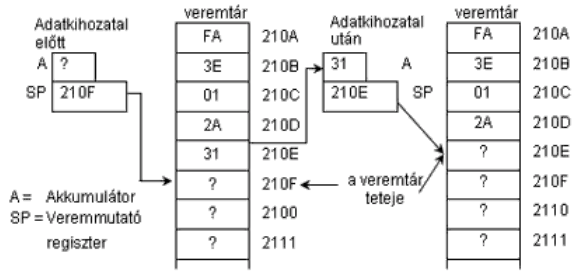
Adatbevitel – PUSH – művelet

Adatkihozatal (POP)

Az adatkihozatal művelete ellentétes az adatbevitellel: először a veremmutató tartalmát eggyel csökkenti, hogy az a verembe utolsónak beírt adatot megcímezze, azután a veremmutató által megcímzett memóriarekesz tartalmát továbbítják az akkumulátorba vagy más CPU regiszterbe. Ezt az alábbi ábra mutatja be.

Figyeljük meg, hogy az adatkihozatal-művelet végén a veremmutató ismét megcímzi a verem felső részén lévő első szabad memória-tárolóhelyet, ugyanis azt tételezzük fel, hogy ha egy adatot kiolvasunk a verem felső részéből, akkor az adattárolóhely „kiürül".

A verem-memóriacímzés esetén a veremmutató tartalmát növelni (inkrementálni) kell a beírás után és csökkenteni (dekrementálni) kell a kiolvasás előtt.



Természetesen semmi akadálya nincs annak, hogy a műveletet ne a verem felső részén, hanem a verem alján végezzék el. Ebben az esetben azonban a beírás előtt dekrementálni és a kiolvasás után inkrementálni kell.

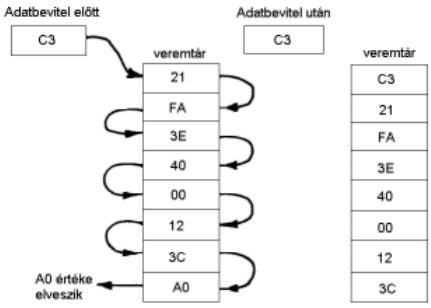
Adatkihozatal – POP – művelet

4.1.4.2.4.3 A kaszkád verem

Létezik egy másik veremszerkezet is. Ennél a megoldásnál a verem a CPU-ban van, és korlátozott számú regiszterből áll.

Adatbevitel (PUSH)

Ha egy adatbájtot beírunk a verembe, az az alábbi ábrán látható módon halad végig.

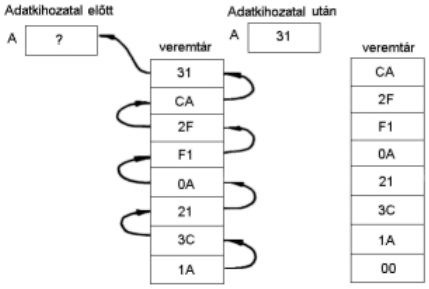


Adatbevitel a kaszkádba kapcsolt verembe

Adatkihozatal (POP)

Ha az adatkihozatal műveletét végezzük el a verem legfelső részéről, az adatok az alábbi ábrán látható módon haladnak végig.

Ehhez a veremszerkezethez nincs szükség veremmutatóra, mivel minden esetben az adatokat a verem legfelső részébe írják be, vagy onnan olvassák ki.



Adatkihozatal a kaszkádba kapcsolt veremből

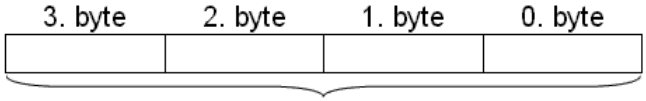
4.1.4.2.4.4 Mire használható a veremtár?

Szubrutinok

A legtöbb programban vannak olyan gyakran alkalmazott utasítás-sorozatok, amelyeket többször felhasználunk a program végrehajtása során, de csak egyszer tárolunk a programmemóriában. Ezeket a programrészeket (rutinokat) szubrutinnak nevezzük.

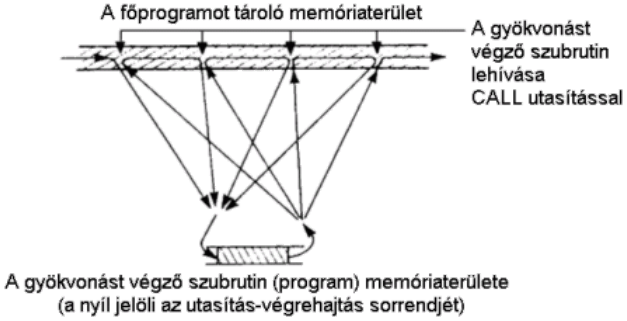
Nézzünk egy példát

Tegyük fel, egy programban többször előfordul, hogy olyan 32 bites számokból, amelyek négy egymással összefüggő bájtot foglalnak el, négyzetgyököt kell vonnunk.



Egy 32 bites szám

Az ilyen típusú aritmetikai művelet leghatékonyabban úgy végezhető el, ha külön programot írunk a gyökvonás elvégzésére. Minden esetben, amikor a programban el akarjuk végezni a gyökvonást, egy olyan utasítást alkalmazunk, amely lehívja (CALL) a gyökvonást végrehajtó szubrutint. A szubrutinlehívás az alábbi ábrán látható módon ábrázolható.

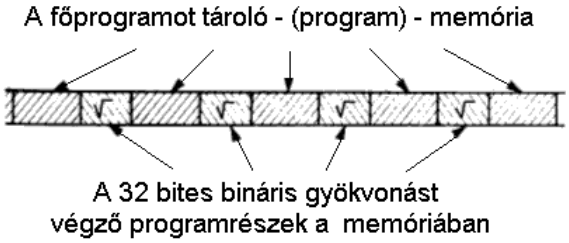


Szubrutin lehívás – CALL – menete

Látható, hogy a szubrutinnak mindig át kell adni a 32 bites szám kezdőcímét, valamint egy másik tárolóhely címét, ahova a szubrutin a kiszámított négyzetgyök értékét elhelyezheti. Ezeket a szubrutin paramétereinek nevezzük.

Mivel – mint ezt az alábbi ábrán is láthatjuk – a főprogramot mindig a CALL utasítást követő utasítással akarjuk folytatni, a paraméter átadás, átvétel mellett meg kell jegyezni a főprogramba való visszatérés címét is.

Fontos felhívni arra a figyelmet, hogy ha nem alkalmazunk szubrutinokat, akkor a gyökvonás elvégzésére szolgáló utasítássorozatot minden alkalommal meg kell ismételni a programban (tehát például a programozónak annyiszor ismételten le kell írni), amikor a főprogramban 32 bites gyökvonást akarunk elvégezni. Így a program szubrutin nélkül a következő lenne:

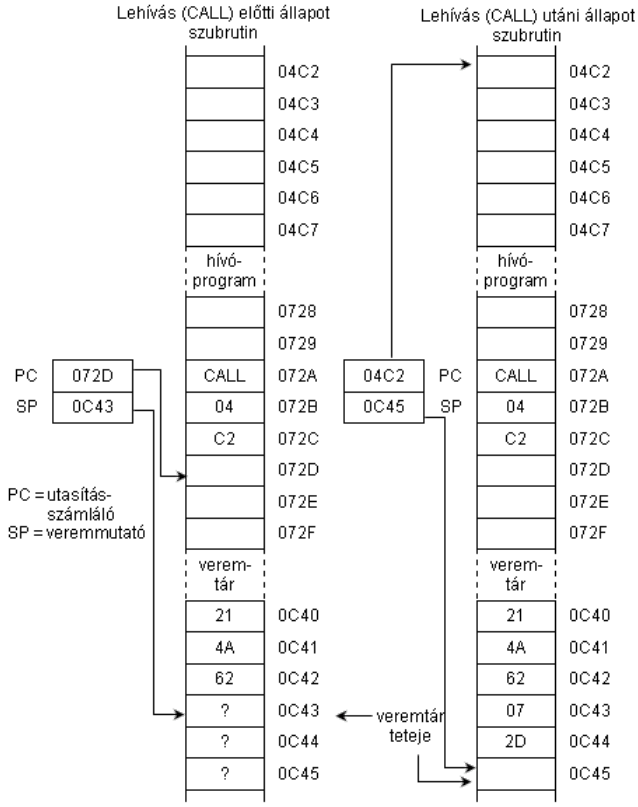


32 bites gyökvonás a főprogramban szubrutin nélkül

Látható, hogy ez a változat a gyökvonást végző utasítások többszöri leírása mellett több memóriát is elfoglalna.

A szubrutin visszatérési címeket (azaz melyik címen kell a főprogramot folytatni a szubrutin befejezése után) legtöbbször veremben tároljuk.

A gyökvonás szubrutinjának lehívása ekkor a következő módon hajtható végre (lásd az ábrát). (Az ábra példájában a memóriarekeszek 8 bitesek, a címek mind 16 bit hosszúak, így két-két bájt szükséges minden egyes cím tárolására. Ezért a CALL utasítás 3 bájt helyet foglal el.)



32 bites gyökvonás szubrutinjának lehívása veremhasználattal

A CALL szubrutinhíváskor

- **az utasításszámláló regiszter (PC) tartalma (mely a CALL utasítás utáni 072D címre mutat) a SP veremmutató regiszterben lévő 0C43 címre (ez a verem aktuális teteje) 2 bájton elmentésre kerül,**
- **a veremmutató-regiszterben lévő címet két bájttal megnöveljük, az új 0C45 érték lesz a verem aktuális tetejének címe,**
- **azután az utasításszámláló regiszterbe bekerül a szubrutin 04C2 címe, ahol folytatódik az utasítások végrehajtása.**

A szubrutinból való visszatéréshez csak a verem legfelső részéből kell a felső két bájtot az utasításszámlálóba átvinni. A program végrehajtása ezután a CALL utasítás után következő utasítással folytatódik.

4.1.4.2.4.5 Egymásba illesztett – egymásba ágyazott – szubrutinok és a veremtár

A veremtár alkalmazásának hasznosságát különösen az egymásba ágyazott szubrutinok esetében érzékelhetjük igazán. Ezek olyan szubrutinok, amelyek maguk is egy szubrutint hívnak meg.

Semmi szokatlan nincs abban, ha egy szubrutint egy másik szubrutin hív le. Valójában igen gyakran öt vagy annál több szubrutin is lehet egymásba illesztve. Sőt, léteznek olyan matematikai rutinok is, amikor a szubrutin önmagát hívja le. Az olyan típusú szubrutint, amely le tudja hívni önmagát, rekurzív szubrutinnak nevezik.

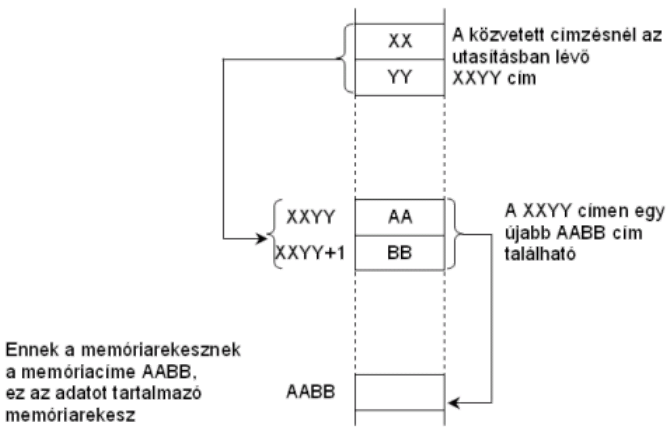
Mindaddig, amíg a vermet alkalmazzák a visszatérési címek tárolására, a szubrutinok bármilyen módon egymásba illeszthetők, vagy lehívhatják egymást: ha biztosítva van, hogy a visszatérés útja pontosan követi a lehívás útját, a megfelelő visszatérési cím mindig a verem aktuális „tetején" lesz.

A mikroszámítógép működésének és programozásának megértéséhez tényként kell elfogadni, hogy a verem biztosítja azt, hogy a szubrutinból való visszatérés útja pontosan fordítottja legyen a szubrutin lehívások útjának.

4.1.4.2.5 Közvetett címzés

A közvetett cím egy olyan memóriaterületet címez meg, amely az adat tényleges címét tartalmazza (lásd az alábbi ábrát).

A közvetett címet sokszor pointernek (mutatónak) is szokás nevezni.

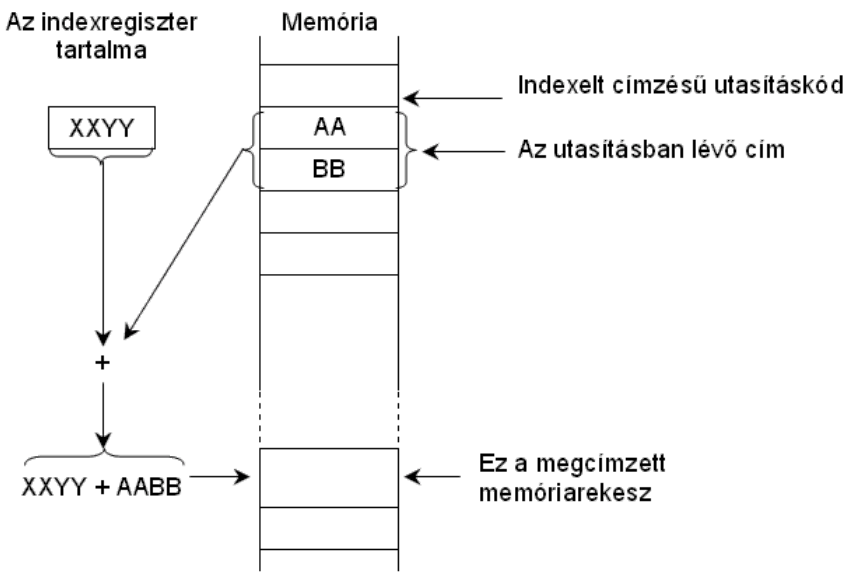


Közvetett memória címzés

A közvetett címzéshez a gépi utasításokban gyakran regisztereket alkalmazunk. Ekkor az utasításban egy regiszterre hivatkozunk, melyben a keresett adat címe megtalálható.

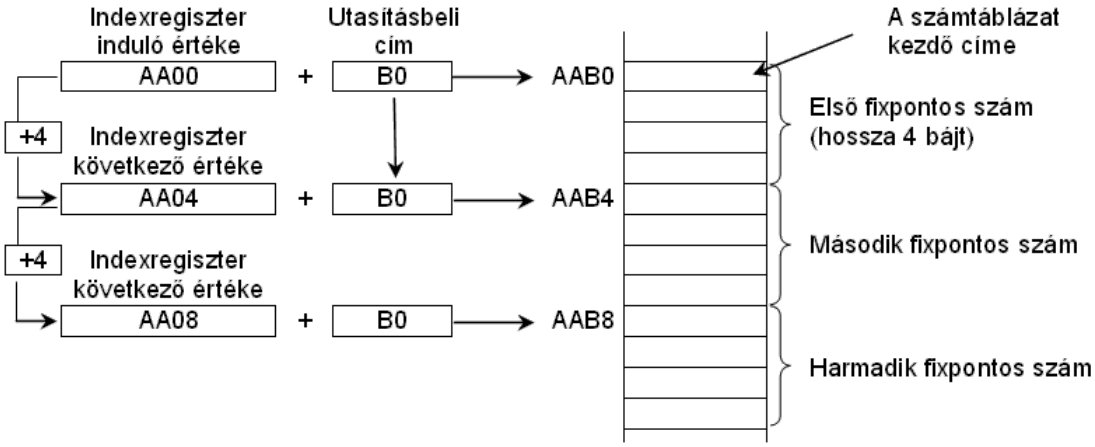
4.1.4.2.6 Indexelt címzés

Indexelt címzésnél a tényleges címet úgy kapjuk meg, hogy az utasításban lévő címhez (ez például egy vektortáblázat kezdőcíme a memóriában) hozzáadjuk az indexregiszterben lévő értéket. Ez a következő módon ábrázolható:



Indexelt címzés

Az indexelt címzést leggyakrabban több azonos típusú elemből (például 4 bájtos fixpontos számból) álló táblázatok (ún. tömbök) feldolgozására alkalmazzuk (lásd az alábbi ábrát).



Számtáblázat címzése indexeléssel

Látható az ábrán, hogy minden egyes lépésben 4 bájttal, a táblázatban lévő fixpontos számok hosszával megnöveljük az indexregiszter értékét, így az indexelt címzéssel a táblázatban lévő összes szám címét lépésről-lépésre előállítjuk.

4.2 A mikroprocesszor alapú rendszer

Az eddigiek során röviden ismertettük a mikroprocesszorok fontosabb jellemzőit. Most megvizsgáljuk, hogy a mikroprocesszor család elemeiből hogyan történik a rendszerépítés. Mint már említettük, a mikroprocesszor önmagában működésképtelen, ahhoz, hogy céljainknak megfelelően használható digitális rendszerünk legyen, a mikroprocesszorainkat ki kell egészíteni memóriákkal, bemeneti-kimeneti egységekkel, más elemekkel. Az ily módon felépített rendszert mikroszámítógépnek nevezzük. A gyártó cégek az egyes kiegészítő funkcionális egységeket, amelyből a különböző igényeknek megfelelő mikroszámítógépek felépíthetőek, készen, tokozva forgalmazzák. Kialakult a családelv. Egy-egy család – ami mikroprocesszorból, memóriákból, bemeneti- kimeneti (I/O) egységekből, óragenerátorból, vezérlőkből stb. állhat – különféle mikroszámítógép konfiguráció megvalósítását teszi lehetővé. A családtagok egymással kompatibilisek.

4.2.1 Memóriák csatlakoztatása a mikroprocesszorhoz

4.2.1.1 ROM (Read Only Memory) csak olvasható tár

A ROM üzemszerű használatkor csak olvasható tároló, tartalmát általában a gyártó írja be. Leggyakrabban általános vezérlési funkciókat megvalósító programok tárolójaként alkalmazzák. Egyik megvalósítási formája, pl. planáreljárás során megfelelő maszk segítségével programozott, MOS tranzisztorokból álló tár.

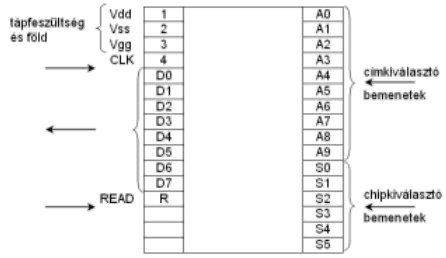
A ROM memóriákat nagyon egyszerűen lehet a központi egységhez (CPU) illeszteni. Ha megvizsgálunk egy ROM tokot, akkor általában a következő lapon található ábrán feltüntetett alábbi csatlakozópontokat találjuk rajta:

- **címkiválasztó bemenetek (A0-A9);**
- **memóriachip-kiválasztó bemenetek (S0-S5);**

- **adatkimenetek (D0-D7);**
- **READ (olvasás) jelbemenet (R);**
- **órajelbemenet (CLK);**
- **tápfeszültség- és föld-csatlakozók (Vdd, Vss, Vgg)**

A cím- és memóriachipet kiválasztó bemeneteken keresztül tudjuk elérni (kiolvasni) az éppen igényelt információt, vagyis egy rekesz tartalmát. A megcímzett memóriarekesz tartalma a sínrendszer adatvezetékein keresztül kerül átvitelre a központi egységbe (CPU). Maga a kiolvasási művelet a READ (olvas) vezérlőjel hatására megy végbe. A műveletvégzés alatt a CPU és a ROM működése szinkronban van, a szinkronizálást a CLK órajel végzi.

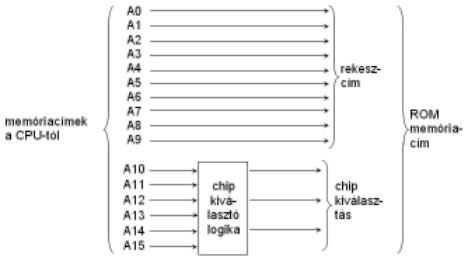
Ha a mikroszámítógépünk csak egy mikroprocesszorból (CPU) és egy ROM-ból áll, a két eszköz nagyon egyszerűen – közvetlenül – összekapcsolható egymással. Általában azonban több eszköz szükséges ahhoz, hogy az igényeinknek megfelelő mikroszámítógép-rendszerünket megalkossuk, amihez a szükséges jelkapcsolatokat egy, az eszközöket összekötő sínrendszeren – szokásos elnevezése szerint buszrendszeren – keresztül lehet célszerűen létrehozni. A továbbiak során az egyszerűség kedvéért 16 bites címzésű és 8 bites memóriarekeszeket tartalmazó számítógépet fogunk feltételezni (ez 16 bites címsínt és 8 bites adatsínt feltételez).



A mai számítógépek buszrendszere természetesen ennél jóval több adat egyidejű továbbítását is lehetővé teszi. Például a Pentium esetében a sín címvonalainak száma 32, az adatsín vonalainak száma pedig 64. Ez 64 bitből álló adatok egyidejű továbbítását biztosítja, és ennek megfelelően a memóriachipek csatlakozópontjainak száma is 32 bit cím- és 64 bit adatkimenetre módosul.

A ROM (fixmemória) csatlakozó pontjai

A CPU 16 címző jele közül 10 jel a rekesz címzésére, 6 jel pedig annak a ROM chipnek a kiválasztására szolgál, amelyben a rekesz tárolva van. A címvezetékeknek ebből a felosztásából azonnal következik, hogy a ROM chip csak 1024 rekesz kapacitású lehet. Ha a CPU nagyobb helyi értékű 6 címvezetékének (A10-A15) jelei megegyeznek a ROM eszközünk azonosító kódjával (S0-S5), akkor a tárolót kezelő logika a kisebb helyi értékű címvezetékeken (A0-A9) található jelek alapján megjeleníti az adatvezetékeken (D0-D7) az 1024 közül kiválasztott egy memóriarekesznek a tartalmát.



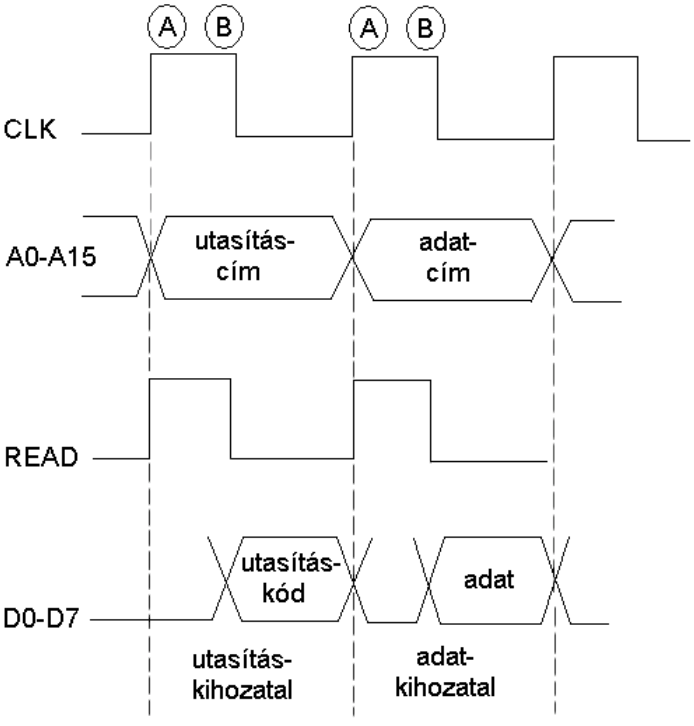
ROM- és/vagy RAM-kiválasztó logika

Vizsgáljuk meg egy ROM-CPU rendszerben a memória olvasási – READ – utasítás időzítését.

A vizsgált pontokat A és B jellel jelöltük (lásd az ábrát).

Minden művelet egy logikai 1-es bemenő CLK órajellel kezdődik („A" pont) és amikor a CPU kimenetein egy cím megjelenik, ugyan-abban az időpontban a READ vezérlő jel logikai 1-re vált. Ezeket a jeleket a ROM a busz-(sín) rendszeren keresztül (lásd az ábrát) fogadja.

Ha a nagyobb helyi értékű 6 címvezeték – A10-A15 – jelei megegyeznek a ROM kiválasztó – S0-S5 – kóddal, akkor a ROM chip logikája az A0 A9 által címzett memóriarekesz tartalmát – ami utasítás vagy adat lehet – abban az időpontban helyezi rá a D0-D7 adat-buszra (adatsínre), amikor az alábbi ábra szerint a „B" pontba érünk. Ekkor a CLK órajel és a READ jel logikai 0-ra vált. Az adat addig marad a D0-D7 adatbuszon, amíg a CLK órajel újra logikai 1 nem lesz.



A READ (memóriaolvasás) utasítás időzítése

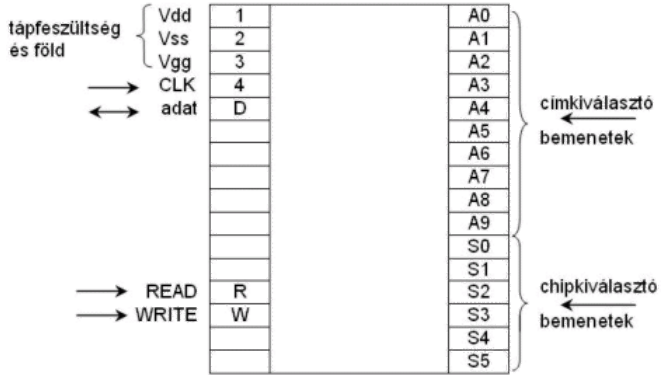
4.2.1.2 RAM (Random Access Memory) közvetlen elérésű

A RAM-ba üzemszerűen beírható és belőle üzemszerűen kiolvasható az adat, ezért a RAM-ot kezelő illesztő logika sokkal bonyolultabb, mint ahogy azt a ROM memóriáknál láttuk.

A RAM-nak működéséből adódóan képesnek kell lennie arra, hogy az adatokat felvegye a külső vezetékekről, és elhelyezze azokat a címzett memóriarekeszbe, valamint arra is, hogy a kívülről címzett memóriarekesz tartalmát kiolvassa és elhelyezze a buszrendszer adatvezetékein.

Egy RAM chip tokján általában a következő lapon található ábrán bemutatott alábbi csatlakozó pontokat találjuk:

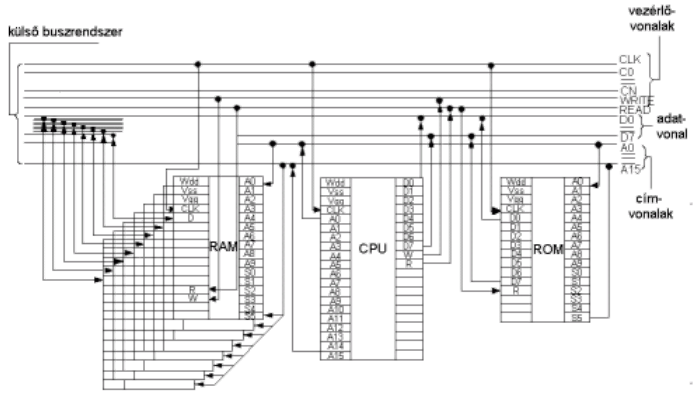
- **címbemenetek (A0-A9);**
- **memóriachip-kiválasztó bemenetek (S0-S5);**
- **adat-bemenetek/kimenetek (D);**
- **READ (olvasás) jelbemenet (R);**
- **WRITE (írás) jelbemenet (W);**
- **órajelbemenet (CLK);**
- **tápfeszültség- és föld-csatlakozók (Vdd, Vss, Vgg).**



RAM (írható/olvasható memória) csatlakozó pontjai

A csatlakozások funkciói jórészt megegyeznek a ROM-nál ismertetettekkel. Eltérés az írási művelet elvégzéséből adódik, ami miatt az adatbusz (D) kétirányú adatátvitelt tesz lehetővé a CPU és a RAM között. A beírási művelet a WRITE (ír) vezérlőjel hatására megy végbe.

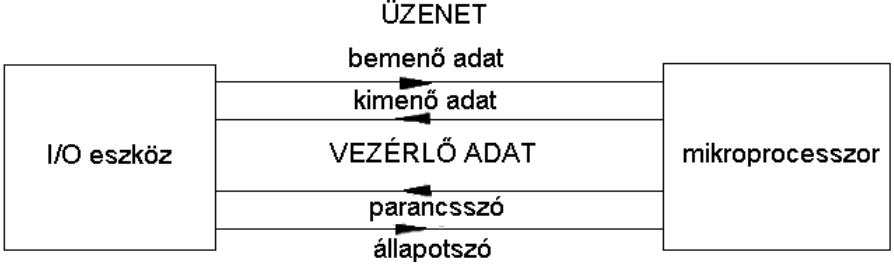
Egy rekeszhez tartozó bitek tárolására általában több RAM chipet használnak. Az alábbi ábrán bemutatott konfigurációban (8 bites rekeszhosszúság) 8 db RAM chip van. Egy RAM chipnél egy adatcsatlakozót (D) használnak fel, a RAM chipek csak egy-egy adatvezetékkel csatlakoznak az adatbuszra (D0-D7 adatvonalakra). Ezért a chipkiválasztó (beazonosító) kódjuk megegyezik, így a 8 db chipből kialakított RAM memória egy összefüggő egységet, memóriamodult képez. Tehát a 8 bites rekesz adatainak tárolását 8 chip végzi, mindegyik 1 bit tárolásáért felelős.



CPU, RAM és ROM csatlakoztatás a buszrendszeren keresztül

4.2.2 Adatátvitel a mikroszámítógép és a hozzá csatlakoztatott perifériális

Az input/output (I/O) eszközök és a mikroprocesszor között a sínrendszeren keresztül különböző adatok és vezérlőjelek áramlanak. Ezt szemlélteti az alábbi ábra.



Adatfolyam a mikroprocesszor és a bemeneti/kimeneti (I/O) eszközök között

4.2.2.1 Interfész, I/O PORT

A mikroszámítógép központi része és a hozzá csatlakozó I/O egységek közötti adatátvitel céljából biztosítani kell az egyes egységek illesztését és a rendszer összehangolt működését. Ezért a mikroszámítógép a bemeneti és a kimeneti készülékekhez vezérlő és illesztő – interfész – áramkörökön keresztül csatlakozik.

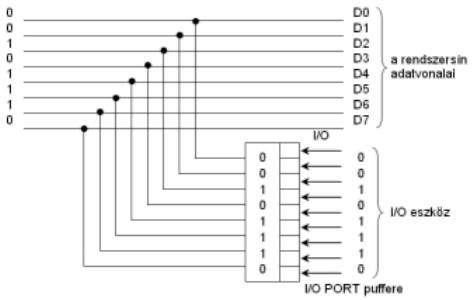
Az interfész két funkcionális egység összekapcsolhatóságát és együttműködését biztosító előírások összessége. Ezen előírások kiterjednek többek között – hardver esetén – a fizikai-mechanikai jellemzőkre (pl. csatlakozók), a definiált jelekre, azok elektromos jellemzőire, az egyes funkciókat realizáló jelfrekvenciákra, valamint – szoftvernél is – a definiált műveletek (pl. kapcsolatfelvétel, adatátvitel, szétkapcsolódás) megvalósítására.

Az interfész tehát két rendszer (vagy egy rendszer egyes egységei) közötti képzelt felületnek az a része, amelyen keresztül az adatok átvitele az igényelt illesztés biztosításával (kódátalakítás, sebességváltoztatás stb.) történik. Az interfészrendszer: kábelek, csatlakozók, meghajtó- és vevő-áramkörök, jelvezeték-előírások, időzítési és vezérlési egyezmények olyan készlete, amely biztosítja az egyes rendszertechnikai egységek közötti egyértelmű, rendezett és szervezett információcserét.

A PORT a mikroszámítógépnek olyan interfésze, amely a perifériális eszközökkel tart kapcsolatot. Ez biztosítja a szabványos csatlakozást a CPU és a perifériális egységek között, a rendszersín (külső busz) közbeiktatásával. Az I/O PORT pufferei a külső buszrendszer adatvezetékeihez csatlakoznak (lásd a következő lapoon az ábrát).

Az I/O PORT nincs állandóan aktív kapcsolatban az adatvezetékekkel, mert általában azokon keresztül bonyolódik le például a CPU és a memóriák közötti adatforgalom is. A CPU az I/O eszközzel való műveletvégzés igényeinek megfelelő időben választja ki csak az I/O PORT-ot.

Amikor a külső I/O eszköz adatot továbbít a mikroszámítógépbe, azt az I/O puffer tárolja. Az adatátvitel ezután az alábbi ábrán látható módon a sínrendszer segítségével valósul meg.



I/O PORT csatlakoztatása a rendszersín adatvezetékeihez és egy I/O eszközhöz

Az I/O PORT pufferek éppen úgy címezhetők a sín címvonalaival, mint a memóriarekeszek. Egyszerűen megoldható a címzés, ha kikötjük, hogy ha mikrogépünknek a legnagyobb helyi értékű címvezetéke (A15) logikai 0 értékű, akkor ez memóriamodult, ha logikai 1, akkor pedig I/O PORT puffert választ ki.

Sok mikroszámítógép rendszerben használnak különálló logikát az I/O PORT-ok kiválasztására, e célból két külön vezérlővonal csatlakozik a CPU-hoz. Ekkor

- **Az IOSEL vezérlővonal jelszintje jelzi, hogy a címvezetékek az I/O PORT-puffer-kiválasztó kódot tartalmazzák.**
- **Az IORW vezérlővonallal az írás (WRITE) és az olvasás (READ) különböztethető meg. Ha az IORW logikai 1, akkor a külső adatbuszon lévő adatot az I/O PORT puffernek ki kell olvasnia, ha logikai 0, akkor a kiválasztott I/O PORT puffer tartalma kerül a külső adatbuszra.**

Ha a mikroszámítógép kimenő jelei a külső I/O eszközök számára parancsokat továbbítanak, akkor ezeket I/O vezérlő jeleknek hívjuk.

A külső eszköz ugyanakkor a mikroszámítógép felé I/O állapotokat (I/O Status) jelző adatokat továbbít. Más szóval a mikroszámítógép processzorának kimenetei vezérlik a külső eszközt, ez viszont csak az állapotára vonatkozó információkat tudja eljuttatni a processzorhoz. Ez ezeket értelmezi, és így el tudja dönteni, hogy melyik I/O PORT-hoz kapcsolt eszköz van olyan állapotban, hogy az adatátvitelt végre lehet hajtani.

4.2.2.2 Az adatátvitel típusai

A mikroszámítógépek és a perifériák közötti adatátvitelt három fő módszerrel oldhatjuk meg:

- **Programozott bemeneti/kimeneti (I/O) adatátvitelnél az összes adatátvitellel kapcsolatos műveletet a számítógép programja vezérli.**
- **Megszakítással kezdeményezett bemeneti/kimeneti (I/O) adatátvitelnél az adatátvitelt egy programmegszakítás előzi meg. Ekkor egy I/O eszköz „kényszeríti" a processzort, hogy félbeszakítsa az éppen futó program végrehajtását, majd az I/O eszköz kérését végrehajtsa.**
- **Közvetlen memóriahozzáférés esetén (DMA = Direct Memory Access) közvetlen adatátvitel jön létre a memória és a bemeneti/kimeneti (I/O) készülékek között a CPU igénybevétele nélkül.**

4.2.2.3 Programozott adatátvitel (PIO mód)

A programozott I/O adatátviteli műveletek végrehajtása általában a következő lépésekből áll:

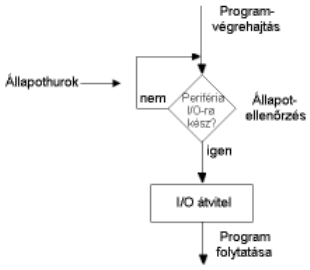
- **a periféria állapotát leíró adatok kiolvasása és vizsgálata;**
- **adatküldés a számítógépből a perifériához, vagy adatátvétel a perifériától.**

Ehhez az I/O interfészek tartalmaznak egy állapotregisztert, mely a perifériák állapotára vonatkozó jelzőbiteket tárolja, és egy adatregisztert (puffert), melyen keresztül az adatcsere a perifériával megvalósulhat.

Ha az adattovábbítás igénye az I/O eszköztől származik, az eszköz ezt azzal jelzi, hogy a megfelelő állapotbiteket az állapotregiszterben beállítja. Ha programmal kell megállapítani, hogy egy eszköz a műveletre készen áll-e vagy sem, akkor állapotregiszter bitjeit addig kell folyamatosan vizsgálni, míg abból az eszköznek az I/O művelet végrehajtására való alkalmassága meg nem állapítható.

Ez az állapotellenőrző hurok (azaz az állapotbitek sokszor ismételt ellenőrzése) drasztikusan leállítja a program érdemi részének végrehajtását, és így a hasznos processzoridő elfogadhatatlan pazarlását idézi elő (lásd az ábrát).

Programvezérelt I/O adatátvitel; állapothurok

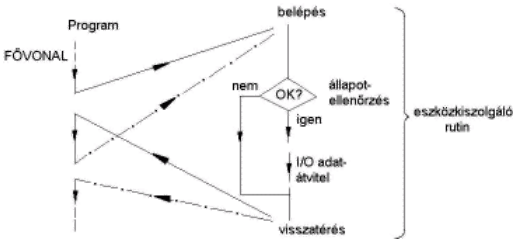


A processzoridő nem hatékony felhasználását különösen akkor érzékelhetjük, ha belegondolunk a működési sebességekben mutatkozó igen jelentős különbségekre. (Egyes perifériák több milliószor lassúbbak, mint a processzor.)

Ha például a processzor egy regiszter-regiszter műveletéhez szükséges időt 1 másodpercnek feleltetjük meg, akkor például a nyomtatás ehhez viszonyítva évekig tart. Így egy olyan helyzet áll elő az állapothurok esetében, hogy a processzor pár másodpercig dolgozik, azután például egy évig az állapotbiteket vizsgálja feleslegesen.

Mi lehet a megoldás erre a problémára?

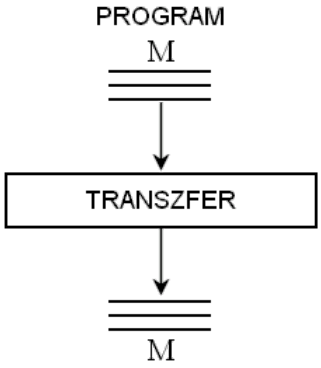
Kézenfekvőnek látszik olyan algoritmust keresni, hogy a processzor az állapotbiteket ne folyamatosan ellenőrizze, hanem csak időszakosan. Ha két ellenőrzés közötti időtartamot ügyesen úgy választjuk meg, hogy az a periféria lassúságával összhangban legyen, akkor a processzor a két állapotellenőrzés közötti időt hasznos utasítások végrehajtásával ki tudja tölteni. Ezt a lehetőséget mutatja be az ábra.



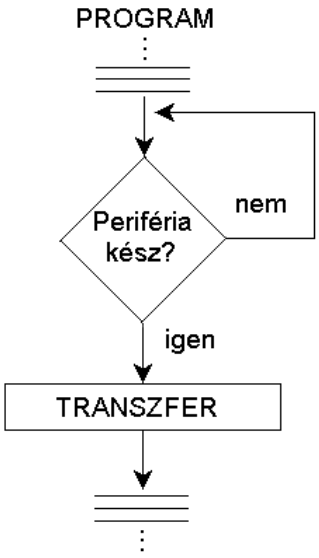
Programvezérelt I/O adatátvitel a „beillesztés” művelettel

Az állapothurok problémájára akkor kell különösen odafigyelni, ha a processzor egyidejűleg több adatátvitelt igénylő eszközzel is kapcsolatban áll, mivel ekkor az időszakos állapotellenőrzéseket mindegyik eszközön végre kell hajtani.

A programozott I/O adatátviteli művelet lehet feltétlen vagy feltételes (azaz valamilyen feltételtől függő). A feltétlen adatátvitel az adatok küldésére vagy fogadására mindig kész perifériát tételez fel (lásd az ábrát), ekkor tehát nem szükséges az állapotellenőrzés.



Feltétlen adatátvitel



A feltétlen adatátvitelre jó példa a számítógépen levő jelzőlámpák felgyújtása (LED kijelzők).

Feltételes adatátvitel esetén a CPU először megvizsgálja a periféria állapotregiszterét, és az adatközlés vagy vétel csak akkor következik be, ha a periféria arra készen áll (lásd az ábrát).

Ekkor alakulhat ki az előzőekben részletezett állapothurok.

Feltételes adatátvitel

4.2.2.4 Programmegszakítással (interrupt) történő adatátvitel

Az előzőekben láttuk, hogy a programvezérelt adatátvitel nem igazán hatékony megoldás, ha a CPU sebességét hatékonyan ki akarjuk használni. Ez az egyik oka annak, hogy a mikroszámítógépet a tervezők programmegszakítási lehetőséggel is ellátják.

Ekkor a számítógép rendelkezik egy olyan elektronikai részegységgel (megszakításvezérlő), amely lehetővé teszi az I/O eszközök számára, hogy I/O adatátviteli igényüket jelezzék a processzornak, ez pedig képes megszakítani a futó programot, és ezt követően kiszolgálni az I/O eszközt.

Mikor lehet szükség a mikroszámítógépünk programjának időszakos megszakítására?

A válasz egyszerű: amikor a jelenleg futó programnál a teljes rendszer hatékonyságának növelése miatt fontosabbá válik egy másik program végrehajtása.

4.2.2.4.1 A programmegszakítás okai

A programmegszakítás kérésének az I/O eszközök adatátviteli igénye mellett más oka is lehet, így például:

- **a jelenleg futó program folytatása valamilyen akadály miatt nem lehetséges, az akadály elhárításához szoftver beavatkozás szükséges. (Például egy utasítás nullával akar osztani, vagy paritáshiba lépett fel);**
- **a számítógépet kezelő ember közölni kíván valamit a rendszerrel, például leüt egy billentyűt.**

4.2.2.4.2 A megszakítás kiszolgálása

A megszakításkérés kiszolgálás során a következőket kell végrehajtania a CPU-nak:

- **meg kell állítani a jelenleg futó programot;**
- **tárolni kell a jelenleg futó programra vonatkozó olyan adatokat, melyeket a megállított program későbbi folytatásához meg kell őrizni (például az utasításszámláló és az állapotregiszter tartalma);**
- **el kell indítani a megszakítást kiszolgáló rutint, amely például elvégzi az I/O eszköz által igényelt adatátvitelt;**
- **be kell fejezni a megszakítást;**
- **vissza kell állítani az eredeti állapotot, vagyis a megszakított program folytatásához a processzort vissza kell állítani abba az állapotba, amelyben az a megszakítás időpontjában volt.**

Programmegszakítást csak úgy lehet kiszolgálni, ha gondoskodunk azokról a hardver- és szoftvereszközökről, melyek az előbbi lépések végrehajtását biztosítják.

4.2.2.4.3 A megszakítások típusai

A megszakítások a következő három fő típusba sorolhatók be:

- **külső megszakítások, amelyeket egy vagy több I/O eszköz generál;**
- **belső megszakítások, amelyeket maga a mikroprocesszor hoz létre abból a célból, hogy sajátos feltételek vagy hibák előfordulását jelezze, például:**
  - **áramkimaradást,**
  - **a rendszer egyes elemeinek meghibásodását,**
  - **az adattovábbításban előforduló különböző hibákat stb.;**
- **a szoftver által generált, szimulált megszakítások, amelyek például elősegíthetik a programhibák behatárolását azáltal, hogy meghatározott címeken megszakítjuk a program futását és megjelenítjük a tároló tartalmakat. (Ezt nyomkövetésnek, vagy DEBUG-nak nevezzük.)**

A különböző megszakítási okok között fontossági sorrendet is fel kell állítani. Így például néhány megszakítást azonnal ki kell szolgálni, mások esetleg késhetnek, mert fontosabb feladat végrehajtása folyik. Így például a súlyos hardverhibákat jelző megszakításokat nyilvánvalóan azonnal ki kell szolgálni.

Ezért a megszakítási rendszernek prioritásuknak megfelelően különbséget kell tennie a különféle megszakítás források között, és a processzornak fontosságuk (prioritásuk) szerinti sorrendben kell végrehajtania a megszakítások kiszolgálását.

4.2.2.5 Adatátvitel közvetlen memóriahozzáféréssel (DMA)

A DMA vezérlő egy leegyszerűsített CPU-hoz hasonló hardveregység, amely az adatmozgatást közvetlenül irányítja a memória és az I/O készülék között. Ez lényegében úgy is felfogható, hogy az ebben résztvevő periféria közvetlen memóriahozzáféréssel rendelkezik.

A DMA alkalmazásának egyik célja a CPU mentesítése az adatátvitel közvetlen vezérlése alól. Emellett a DMA vezérléssel nagysebességű közvetlen adatátvitel érhető el a memória és az I/O készülékek között.

A CPU, az I/O készülékek, valamint a memória közös buszra vannak rákapcsolva, ezért a CPU és az I/O készülékek azonos ciklusban nem férhetnek hozzá a memóriához. Emiatt valamilyen formában a CPU és a DMA vezérlő között meg kell osztani a sín használatát.

Ennek megoldási formája szerint különböző DMA adatátviteli eljárásokat különböztethetünk meg. Ezek: a CPU leállítása (CPU halt), a memória-időszelet eljárás és a cikluslopás.

4.2.2.5.1 DMA adatátviteli eljárások típusai

A CPU leállítási eljárásnál a DMA kérésére a CPU leáll és lekapcsolódik a buszról a DMA adatátvitel tartama alatt. Ez a módszer lassítja a CPU működését, bár a DMA átvitel nyilván ebben az esetben a leggyorsabb.

A memória-időszelet eljárásnál a memóriaciklust két részre bontják fel: az egyik a CPU-é, a másik a DMA-é. Ez a módszer nagy CPU végrehajtási és nagy DMA adatátviteli sebességet eredményez, mivel CPU és DMA memóriahozzáférés minden ciklusban van. Hátránya, hogy megvalósítása igen nagy sebességű, drága memóriát igényel.

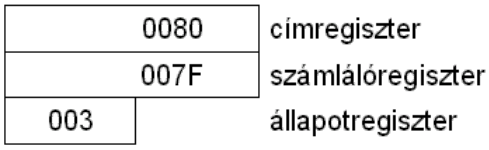
A cikluslopásos eljárás kompromisszum az előző kettő között, ekkor a CPU és a DMA vezérlő átlapolva használja a buszt. Ha a CPU-nak és a DMA-nak azonos időben lenne szüksége memóriára, akkor a DMA-nak prioritása van a CPU-val szemben, és a CPU addig vár, amíg a DMA ciklus be nem fejeződik. Ez a CPU működését lassítja, de nem állítja le, ezért ez a megoldás a legtöbb esetben kielégítő.

4.2.2.5.2 A DMA regiszterei

Egy DMA vezérlő a következő három regisztert használja:

- **a címregisztert, amely azt a memóriarekesz-címet tárolja, amely a következő írási/olvasási műveletben részt vesz;**
- **a számlálóregisztert, amely az átvitt memóriarekeszeket számlálja;**
- **az állapot- (status) regisztert, amelynek tartalma meghatározza az adatáramlás irányát, vagyis a DMA üzemmódját.**

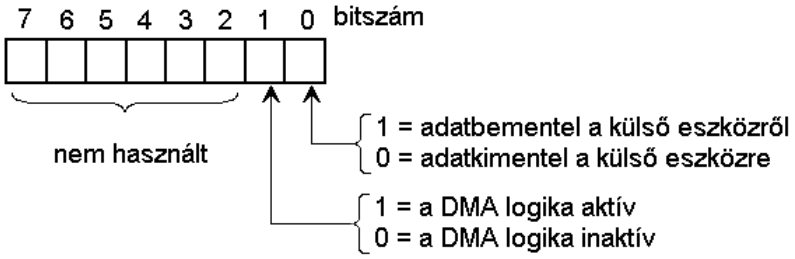
A CPU szempontjából nézve ezek a regiszterek úgy kezelhetők, mint az I/O PORT-ok vagy a címezhető memóriarekeszek. A DMA adatátvitel megkezdése előtt az átvitelhez szükséges adatokkal a CPU a cím-, a számláló- és az állapotregisztereket feltölti (lásd az alábbi ábrát). Ezt követően engedélyezi a CPU a DMA átvitel megkezdését.



Az ábrából leolvasható, hogy az átviendő bájtszám $007F_{16}$ (vagyis, hogy az adatpuffer $7F_{(16)}$ bájt hosszú), és a külső eszközről érkező adat elsőként a $0080_{(16)}$ című memóriarekeszbe kerül tárolásra.

A DMA regisztereinek beállítása (példa)

Az ábrán az állapotregisztert, vagyis a hardverüzemmód beállítását mutatjuk be. Azzal, hogy a $03_{(16)}$ érték az állapotregiszterbe kerül, meghatározottá válik, hogy külső eszköz adata kerül a bemenetre, és a DMA eszköz aktivizálódik.



A DMA az általa kezdeményezett műveleteket a CPU kezdeti segítségével hajtja végre. A CPU betölti az adatokat a DMA regisztereibe, vagyis a DMA vezérlő működési feltételeit a CPU állítja be, kiküldi az olvasási parancsot, az induló memóriacímet, valamint a szószámra és a memóriacímre vonatkozó adatokat.

A DMA művelethez a következő lépéseket kell végrehajtani:

1. **Be kell tölteni az induló címet a DMA címregiszterébe.**
2. **Be kell tölteni az átviendő rekeszszámra vonatkozó adatokat a DMA számlálóregiszterébe.**
3. **Be kell tölteni egy vezérlőkódot (példánkban a $03_{(16)}$-at) a DMA állapotregiszterébe. A vezérlőkód meghatározza az adatátvitel irányát, és annak megfelelően állítja be a DMA-t.**

4.2.2.5.3 A DMA művelet végrehajtása

Vizsgáljuk meg, mi történik, ha DMA lehetőséget építünk be a mikroszámítógép-rendszerbe. Amikor a DMA vezérlő egy adatszót fogad az I/O interfésztől, kéri a rendszerbusz használatát a DMA átvitelhez. Amint megkapta a buszt, kiküldi a memóriarekesz címét és az adatrekesz tartalmát. A "memória kész" jel vételkor a DMA vezérlő megvizsgálja a számlálóregiszterben a rekeszszámot. Ha ez nem egyenlő zérussal, akkor a számlálóregiszter tartalmát 1-gyel csökkenti, a címregiszter tartalmát pedig növeli. Ennek hatására a perifériapufferből a memóriába egy újabb adat átvitele megy végbe. Ha a számlálóregiszter tartalma eléri a zérust, a DMA vezérlő befejezi az adatátvitelt, és közli a CPU-val, hogy az adatátvitel kész.

4.2.2.5.4 DMA vezérlőfunkciók

Egy DMA adatátvitel végrehajtásához a következő vezérlési feladatokat kell ellátni:

Címvonalvezérlés: a DMA-t tartalmazó rendszerekben a memóriacímbuszt vagy a CPU, vagy a DMA hajtja meg attól függően, hogy az adott ciklusban a memóriát melyik eszköz használja. A DMA ciklusban a DMA vezérlőnek a kívánt DMA művelet elvégzéséhez szükséges címet kell a címbuszra adni.

Adatátviteli vezérlés: a DMA vezérlőnek a memória és az I/O készülék közötti közvetlen adatátvitelhez - megfelelő időzítéssel - vezérlőjeleket kell szolgáltatni. E vezérlőjelek csak a DMA ciklusban kapcsolódnak a vezérlőbuszra.

Címtárolás: a DMA vezérlő címregisztere tartalmazza a következő írásra vagy olvasásra kerülő adat címét. Ezt minden átvitel után inkrementálni vagy dekrementálni kell.

Adatszámlálás: a DMA adatátvitel indításakor a CPU betölti a DMA vezérlő számlálóregiszterébe az átvitelre kerülő rekeszek számát. A DMA adatátvitel alatt a DMA vezérlő számlálja az átvitt rekeszeket, és a megadott számú adat átvitele után befejezi az adatátvitelt.

Üzemmódvezérlés: a DMA hardver üzemmódvezérlő regiszterét a CPU tölti fel az adatátvitel előtt.

A DMA vezérlő helye szerint két megoldást különböztetünk meg:

- **Elosztott DMA: az I/O készülékekben van a DMA vezérlő.**
- **Centralizált DMA: egyetlen DMA vezérlő szolgál valamennyi I/O készülék vezérlésére.**

4.2.2.6 Külső buszrendszer (system bus)

A mikroszámítógép-rendszer sínrendszere kapcsolja össze a CPU-t az interfészeken keresztül a perifériákkal. A külső buszrendszer logikailag

- **adatvezetékekből vagy adatsínből (32 vagy 64 bit egyidejű átvitelére),**
- **címvezetékekből vagy címsínből (általában 32 bit egyidejű átvitelére),**
- **vezérlővezetékekből vagy vezérlősínből áll.**

Az adat- és címvezetékek száma a belső és külső buszon általában azonos, napjainkban a legtöbb mikroszámítógép 32 címvonallal rendelkezik.

A vezérlősín a mikroszámítógép részegységei közötti vezérlőadatok átvitelét biztosítja. Ezek lehetnek

- **I/O eszközvezérlő jelek**
- **megszakítási rendszer vezérlőjelei**
- **DMA vezérlőjelei**
- **sínvezérlőjelek, melyek például a sínhasználat kérésére és visszaigazolására szolgálnak.**

A tervezések során két véglet figyelhető meg a CPU-val megvalósított rendszerekben:

- **Az egyik véglet szerinti megoldásnál "okos" CPU-t és "buta" perifériákat használnak, vagyis a vezérlést teljes egészében a CPU hajtja végre. A CPU mondja meg a perifériáknak, hogy mit kell tenniük, és a perifériák szolgaian végrehajtják az utasításokat.**
- **A másik véglet feltételezi, hogy a csatlakozó készülékek, perifériák jelentős mennyiségű belső logikát tartalmaznak. Így a CPU-nak csak elemi vezérlőjeleket kell kiadnia, amelyeket a csatlakozó készülékek értelmeznek, és "saját fejük" szerint hajtanak végre.**

Amikor a CPU teljes mértékben ural egy rendszert, akkor a hozzá csatlakoztatott készülékek számos vezérlőjelet fogadnak, amelyek értelmezik az adatbuszon végbemenő eseményeket. A másik véglet, amikor a CPU csak két vezérlőjelet ad ki. Az egyik jelzi, hogy I/O (in-put/output) vagy memóriaművelet van-e folyamatban, a másik az adatforgalom irányát jelzi, vagyis, hogy adatbevitel vagy adatkihozatal történik-e.

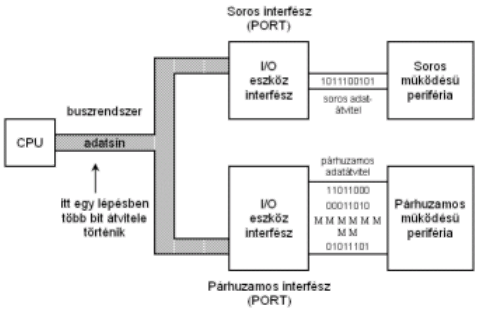
4.2.2.7 Soros adatátvitel

4.2.2.7.1 A soros és párhuzamos adatátvitel

Az előzőekben a CPU és a periféria interfészek közötti adatátvitel megszervezésének kérdéseivel és lehetséges formáival foglalkoztunk.

Hogyan történhet az adatátvitel a periféria interfészek és a perifériák között?
Erre két, lényegében különböző megoldás létezhet:

- **a periféria interfész és a periféria között az adatokat bitenként sorba egymás után visszük át, ezt soros adatátvitelnek nevezzük,**
- **a periféria interfész és a periféria között az adatokat bitcsoportonként egyszerre visszük át, ezt párhuzamos adatátvitelnek nevezzük.**



Soros és párhuzamos adatátvitel

A soros működésű I/O eszköz interfészt soros, a párhuzamos működésűt pedig párhuzamos portnak is szokták nevezni.

Nyilvánvalóan a párhuzamos adatátvitelnek megvan az az előnye, hogy gyorsabb a sorosnál, mivel egy lépésben egy bitcsoportot viszünk át. Ehhez viszont legalább annyi vezetéket (adatutat) kell biztosítani az adatok átviteléhez, mint ahány bitből áll az átvitt bitcsoport.

A soros átvitelhez viszont szélsőséges esetben egy vezetékpár is elegendő az adatátviteli összeköttetéshez, ami jelentősen csökkentheti a költségeket.

Az előbbieket figyelembe véve a párhuzamos adatátvitelt csak a számítógép közelébe elhelyezhető perifériák, így például nyomtató esetében szokták alkalmazni.

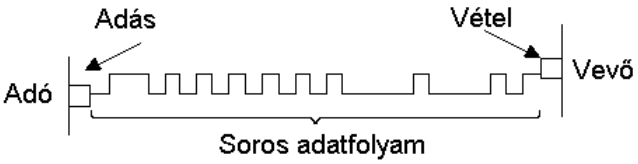
Az elmúlt évek technikai fejlődésének eredményeként egyre nagyobb sebességű soros adatátvitel megvalósítására nyílt lehetőség, így a párhuzamos adatátvitel gyorsasága ma már nem jelent előnyt. Erre példa, hogy a mikroszámítógépek körében gyorsan terjed az úgynevezett univerzális soros buszrendszer (USB = Universal Serial Bus), mely az összes kis és közepes teljesítményű periféria csatlakoztatását biztosítja a számítógéphez egy nagysebességű soros adatátvitellel. Az USB a jövőben várhatóan kiváltja a mai számítógépek soros és párhuzamos portját.

Az előbbiek mellett a soros adatátvitelt alkalmazzák a távadatfeldolgozás során a számítógéphálózatokban is.

Ezek a tényezők is indokolják, hogy a soros adatátvitel alapfogalmait és eljárásait kissé részletesebben is megvizsgáljuk.

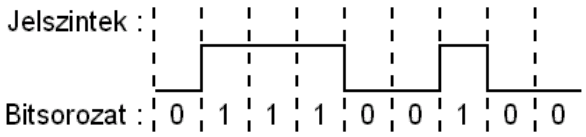
4.2.2.7.2 A soros adatátvitel bitjeinek felismerése

A soros adatátvitelben mindig két eszköz vesz részt, egyik az adó, a másik a vevő szerepkörét tölti be.



Adó-vevő csatlakozó pontok

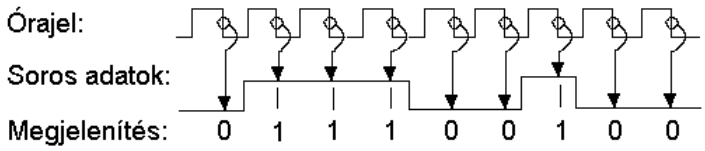
A soros adatfolyamban bitsorozatot viszünk át, minden bitet a jel feszültségszintjével adunk meg. Például, ha az átvitt bitsorozat 011100100, akkor ez jelszintekben a következő ábra szerint jellemezhető:



A soros adatok értelmezése

Vegyük észre, hogy az ábrán a jelszintek értelmezése csak úgy lehetséges, ha ismerjük egy bit időtartamát. Például, ha a bitek időtartama az ábrán láthatónak csak a fele lenne (azaz a szaggatott függőleges vonalak kétszer sűrűbben helyezkednének el), akkor a soros adatfolyamot 001111110000110000 birsorozatnak értelmeznénk.

Ezért a soros adatfolyam értelmezéséhez a vevőnek fel kell ismernie az adatbitek határait. Erre a célra órajelet alkalmazhatunk az alábbi ábra szerint.



A soros adatok értelmezése

Az ábra mutatja, hogy az órajel lefelé menő éle az az időpillanat, amikor a vevő mintát vesz a soros adatok jelszintjéből, és eszerint értelmezi a bitsorozatot. Az is nyilvánvaló, hogy a vevő órajelekkel értelmezi a soros adatfolyamot, akkor az adónak is azonos frekvenciájú órajelekkel kell létrehozni a soros adatok jelszintjét.

Megjegyzés: Nem szabad összetéveszteni a soros adatátvitel szinkronizálásához szükséges órajelet a processzor órajelével, a kettő között több tízezerszeres is lehet a sebességkülönbség, nyilvánvalóan a processzor javára.

4.2.2.7.3 Az adó és vevő szinkronizálása

Láttuk, hogy a soros adatfolyam értelmezéséhez az szükséges, hogy az adó és a vevő azonos frekvenciájú órajelet alkalmazzon. Most már csak az a kérdés, hogy hogyan lehet azt biztosítani, hogy az órajelek generálását a vevő az adóval összehangoltan képezze és például a két órajel ne „csússzon" el egymástól. Erre az egyik megoldás az lehet, hogy az adatok átvitelét szolgáló jelvezeték mellett még egy vezetéket alkalmazunk, melyen az órajelet is átvisszük az adó és a vevő között. Ez viszont nyilvánvalóan nem a leggazdaságosabb.

Hatékonyabb az az eljárás, amelyben az átvitt bitsorozatot használjuk fel az adó és vevő órajelének összehangolásához, szinkronba hozásához. Ez az jelenti, hogy előírunk egy speciális bitsorozatot, amit szinkronizáló jelnek nevezünk, aminek feladata az adó és vevő működésének szinkronizálása, az órajelképzés időbeli összehangolása.

A szinkronizáláshoz megfelelő szabályrendszert is ki kell alakítani, például azt, hogy a szinkronizáló bitmintának mindig meg kell előznie az érdemi adatbiteket. Ez vezet el a kommunikációs protokoll fogalmához.

4.2.2.7.4 Kommunikációs protokoll

Az adatátviteli szabályok, megállapodások összességét kommunikációs protokollnak nevezzük. Ez mint láttuk tartalmazza az adó és vevő oldal szinkronizálásához szükséges szabályokat, de ennél jóval többre is kiterjed. Így például ide tartoznak

- **azok az eljárások, melyekkel az adó jelzi, hogy adatot kíván küldeni, a vevő pedig visszaigazolja, hogy képes fogadni az adatokat,**
- **az adatátvitel során fellépő hibák felismerési, illetve kijavítási szabályai.**

A soros adatvonalakat – szemben a párhuzamos adatátvitellel – nem kísérik minden esetben olyan vezérlővonalak, melyekkel az adatátvitel vezérléséhez szükséges összes információ az adó és vevő között átvihető lenne. Ezért minden soros adatátvitelhez egy kommunikációs protokollra van szükség, amely a vevő és az adó összehangolt működését biztosítja.

A kommunikációs protokollok egyrészt szabványok, másrészt egy konkrét számítógépnél különböző hardver- és szoftverelemek, melyekben a szabványban megfogalmazott szabályrendszer ölt testet.

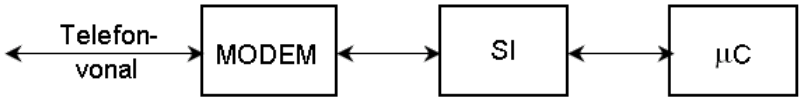
Így például a soros porton keresztül történő soros adatátvitel szabványa az Electronic Industries Association nevű szakmai szervezet által kialakított RS-232-C. Ez részletesen meghatározza az adatátvitel mechanikai, villamos, funkcionális és eljárási szabályrendszerét.

Egy kommunikációs protokoll nem csak néhány egyszerűen megfogalmazható szabály együttese. Például ezek legtöbbjének műszaki dokumentációja csak több ezer oldalban fogalmazható meg.

4.2.2.7.5 Soros adatátvitel távbeszélő vonalakon

Mivel a normál telefonvonalak a jeleket analóg formában viszik át, a soros adatátvitelnél viszont bináris digitek formájában adottak az adatok, ezért a távbeszélővonalakon történő adattovábbításhoz speciális eszközre, az úgynevezett MODEM-re is szükség van.

A MODEM (MODULÁTOR/DEMODULÁTOR) egy analóg-digitális és digitális-analóg konverziót megvalósító eszköz, amely a telefonvonal analóg jeleit digitális bitsorozattá alakítja át, illetve a digitális bitsorozatot a telefonvonalnak megfelelő analóg jelekké fordítja át. Ennek megfelelően a modemet közvetlenül a telefonvonal és a számítógép soros interfésze közé kell elhelyezni.



Adattovábbítás a távbeszélő hálózat és a számítógép között (SI = soros interfész és µC = mikroszámítógép)

A távbeszélőhálózat adatátviteli vizsgálatánál az első szempont annak a ténynek a szem előtt tartása, hogy az adatátvitelben résztvevő készülékek állandóan váltogatják szerepüket: egyszer adók, máskor vevők. Ez teljesen hasonló egy normál telefonbeszélgetéshez, ahol a beszélő az adó és a hallgató a vevő.

Az adatátvitel iránya és egyidejűsége szerint egy összeköttetés lehet:

- **szimplex (simplex), ha az adatátvitel csak egy irányban folyhat, azaz az adó és vevő szerepe nem változhat,**
- **fél-duplex (half duplex), ha az adatokat egy vezetéken mindkét irányba továbbíthatjuk, de az adatátvitel egy adott időben csak az egyik irányban folyik. Ebben az esetben tehát az adó és vevő időnként szerepet cserél,**
- **teljes duplex (full duplex), amikor az adatokat egy időben mindkét irányban továbbítani lehet. Ez két vezetéket tételez fel, és az adatok továbbítása ekkor mindkét irányban párhuzamosan történik.**

A telefonvonalon történő adattovábbítás során az adatok meghibásodhatnak, például a bitsorozatban egyes bitek ellenkezőjükre változhatnak. Ezért fontosak azok az eljárások, melyekkel a vevőkészülék képes ezeket a hibákat felismerni, illetve kijavítani. A hibafelismerésre használható a már megismert paritásbit ellenőrzés. Léteznek olyan eljárások is, melyekkel a hibák nemcsak felismerhetők, de azok egy része ki is javítható.

A soros adatátvitelnél az adatok tömörítése is fontos, mert ha sikerül az átviendő adatokat jelentősen összetömöríteni, akkor (mivel kevesebb mennyiségű bitet kell átvinni) az adatátvitel kisebb teljesítményű összeköttetések esetén is gyors lehet.

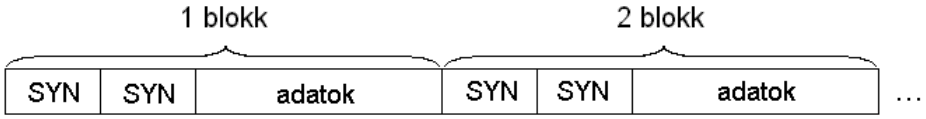
A modemek szabványos hibajavító és tömörítő eljárásai az MNP (Microcom Networking Protocol) protokollok.

Az adatátvitelnek fontos jellemzője a sebessége, melyet Baudban mérünk. 1 Baud sebességű az az átvitel, mely 1 másodperc alatt 1 jelet visz át. Ha az átvitt jelek bináris értékek, akkor a Baud megegyezik a bit/szekundummal

4.2.2.7.6 Szinkron és aszinkron soros adatátvitel

A soros adatátvitel két alaptípusa a szinkron és az aszinkron átvitel.

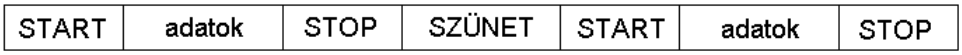
A szinkron adatátvitelnél az adó és vevő szinkronizálását a SYN bitcsoport = 01111110 érzékelése biztosítja a vevő részéről.
Az átviendő érdemi adatokat mindig két SYN jel vezeti be, az átvitt bitek mennyisége rögzített, azaz ezek blokkszervezésűek.



Szinkron soros adatátvitel

A szinkron adatátvitel órajellel vezérelt, azaz az egyes adatokhoz tartozó jelek csak egy meghatározott alapidőtartam egészszámú többszörösei lehetnek.

Az aszinkron adatátvitelnél az adatok elejét és végét két speciális jel, a START és STOP jel jelzi, az ezek között elhelyezkedő jelek értelmezendők adatként. Ebben az esetben az átvitt bitek mennyisége változó is lehet.



Aszinkron soros adatátvitel

Az aszinkron átvitelnél nem szükségképpen van folyamatos kapcsolat az adó és vevő között, ezek szinkronban csak az adatátvitel ideje alatt vannak. A szinkronizálást a START jel eredményezi.

Az adatátviteli eljárások lehetnek karakterorientáltak, ekkor az átvitel egysége az egy karakter kódjának megfelelő bitszám (ez még a távadatfeldolgozás kezdeteitől származtatható, amikor a karakterekből álló szöveges információ átvitele volt a jellemző).

A hálózatok, a hang- és képátvitel elterjedésével kerültek előtérbe a bitorientált eljárások, melyek már nemcsak a karakterkódoknak megfelelő számú bitsorozatok lépésenkénti átvitelét is biztosítják, hanem nagyobb mennyiségű bit (például képek) is továbbítható távadatfeldolgozással.

4.3 Ellenőrző kérdések

1. Milyen feladata van a mikroprocesszornak?
2. Milyen funkcionális egységei vannak a mikroprocesszornak?
3. Mi a regiszter?
4. Mi a különbség a rendszer- és az általános regiszterek között?
5. Milyen tipikus regisztereket ismer a processzorban?
6. Mit nevezünk elérési időnek?
7. Mit tartalmaz az utasításregiszter?
8. Mit tartalmaz az utasításszámláló?
9. Mi a pufferregiszter feladata?
10. Milyen elemi lépésekből áll egy bináris összeadás?
11. Mi a feladata az ALU-nak?
12. Milyen alapműveletek elvégzésére alkalmas az ALU?
13. Milyen részei vannak az ALU-nak?
14. Mi a feladata a vezérlőegységnek?
15. Mit jelent az utasításkód értelmezése?
16. Sorolja fel néhány állapotinformációs bit jelentését!
17. Mi a funkciója a jelzőbitnek (flag)?
18. Mit nevezünk a processzor utasításkészletének?
19. Mit jelent a gépi utasítás szerkezete?
20. Milyen részekből áll a gépi utasítás?

21. Mit határoz meg a műveleti kód a gépi utasításban?
22. Mit tartalmaz a gépi utasítás címrésze?
23. Milyen címzési eljárásokat ismer?
24. Mit jelent az abszolút címzés?
25. Mit tartalmaz a gépi utasítás címrésze relatív címzés esetén?
26. Mi a közvetlen adatcímzés lényege?
27. Mi jellemzi a vermet?
28. Jellemezze a LIFO szervezésű memóriát!
29. Mit nevezünk veremmutatónak?
30. Milyen veremműveletek vannak?
31. Jellemezze a POP utasítást?
32. Mi a hatása a PUSH műveletnek?
33. Hogyan működik a kaszkád verem?
34. Mit nevezünk szubrutinnak?
35. Mi történik a CALL szubrutinhívó utasítás hatására?
36. Hogyan lehet a szubrutinból visszatérni?
37. Mit jelent a rekurzív szubrutin?
38. Mit jelent a közvetett címzés?
39. Hogyan kapjuk meg a valódi címet indexelt ímzésnél?
40. Tipikusan mire használható az indexelt címzés?
41. Milyen részekből épül fel a mikroszámítógép?
42. Mi a ROM?
43. Milyen csatlakozóhelyei vannak egy ROM chipnek?
44. Mivel kapcsolható össze a processzor a memóriachipekkel?
45. Mi a RAM?
46. Milyen csatlakozóhelyei vannak egy RAM chipnek?
47. Mi a különbség a ROM és RAM chip csatlakozóhelyei között?
48. Egy 8-bites rekesz adatainak tárolására hány RAM chip szükséges? Indokolja meg a választ!
49. Mit értünk interfészen?
50. Milyen fizikai elemekből épül fel az interfész?
51. Mit nevezünk PORT-nak?
52. Mihez csatlakoznak áramkörileg az I/O PORT pufferei?
53. Milyen lehetőségek vannak az I/O PORT címzésére?
54. Mire használható a CPU számára az I/O eszköz állapotinformációja?
55. A CPU és a perifériák közötti adatátvitelnek milyen fontosabb típusai vannak?
56. Milyen lépésekből áll a programozott I/O adatátvitel?
57. Az állapotellenőrzés feladata milyen problémákat okoz a processzoridő hatékony felhasználásában?
58. Mondjon példát a feltétlen adatátvitelre a CPU és a perifériák között!
59. Milyen okai lehetnek a programmegszakításnak?
60. **[could not resolve link target: il_0_pg_14035]**
61. Milyen típusai vannak a megszakításoknak?
62. Miért van a megszakítási okoknak prioritása?
63. Mi a DMA vezérlő?
64. Milyen típusai vannak a DMA adatátviteli eljárásoknak?
65. Mi a lényege a memória–időszelet eljárásnak?
66. Mit jelent a ciklusloás?
67. Milyen feladatokat ellátó regisztereket tartalmaz a DMA vezérlő?
68. Hogyan történik meg egy DMA művelet végrehajtása?
69. Milyen vezérlőfunkciói vannak a DMA-nak?
70. Milyen részekből épül fel a külső (rendszer) busz?
71. Milyen jellegű vezérlőinformációkat viszünk át a buszrendszeren?
72. A CPU és a perifériák közötti munkamegosztásnak milyen szélsőséges esetei vannak?
73. Mit jelent a soros adatátvitel?
74. Mit nevezünk párhuzamos adatátvitelnek?
75. Miben különbözik egymástól a soros és párhuzamos port?
76. Milyen perifériákat kapcsolunk a párhuzamos porthoz?
77. Mi az USB?
78. Milyen adatátvitelt alkalmazunk a távadatfeldolgozás során?
79. Hogyan ismerjük fel soros adatátvitelnél az adatbitek határait?
80. Hogyan szinkronizálhatjuk az adó és vevő működését?
81. Mit nevezünk kommunikációs protokollnak?
82. Soroljon fel legalább két területet, melyet a kommunikációs protokollban szabályozni kell!

83. Mi a modem?
84. Mit értünk szimplex összeköttetésen?
85. Mit értünk fél-duplex összeköttetésen?
86. Mit értünk duplex összeköttetésen?
87. Mi a szerepe az MNP protokollnak?
88. Mi az adatátvitel sebességének mértékegysége?
89. Jellemezze a szinkron adatátvitelt!
90. Jellemezze az aszinkron adatátvitelt!
91. Mikor szinkronizált az aszinkron átvitel?

5 Szoftver alapismeretek

5.1 Bevezető

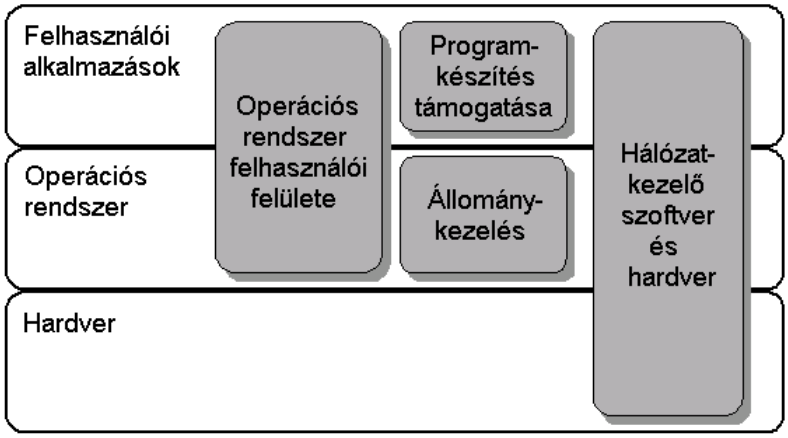
Mint az előbbiekben láttuk, a mikroprocesszor alapú rendszer egy olyan jól felkészült automata, amely precízen, gyorsan, de szolgaian hajtja végre azokat az utasításokat (programokat), melyeket memóriájában talál. Legtöbben azt szeretnénk, ha ezek az utasítások olyan algoritmust alkotnának, amelyekkel adatokat tudunk kezelni, válogatni, levelet tudunk írni, grafikonokat tudunk készíteni vagy segítségükkel barangolhatunk az interneten, azaz felhasználói programokat, vagy más szóval alkalmazásokat szeretnénk futtatni, esetleg írni.

A beszerezhető felhasználói szoftverek száma olyan nagy, hogy a teljesség igényével még csoportosításukkal sem kísérletezhetünk. Az egyes kategóriák hardver igénye is erősen eltérő, már a számítógép vásárlásánál érdemes elgondolkodni azon, hogy mire is szeretnénk használni gépünket.

Egy mai átlagos munkahelyen a következő szoftverekkel találkozhatunk:

- **Operációs rendszer, pl. Windows 95/98/NT, DOS, Linux, OS/2**
- **Irodai szoftvercsomag (szövegszerkesztő, táblázatkezelő, adatbáziskezelő, prezentációs grafika), pl. Microsoft Office**
- **Ügyviteli rendszerek (bérszámfejtés, raktárnyilvántartás, könyvelői rendszerek), pl. MaxBer, COBRA stb.**
- **Tervező rendszerek (CAD – Computer Aided Design), pl. PCAD, ArchiCAD, MathCAD**
- **Programfejlesztést támogató rendszerek, pl. Turbo Pascal, C, Java Builder, Visual Basic, Negyedik generációs nyelvek**
- **Számítógép hálózattal kapcsolatos szoftverek, pl. Pegasus Mail, Netscape, Internet Explorer**
- **Segédprogramok (tömörítő, víruskereső stb.), pl. PkZip, F-Prot**

Ebből a bő kínálatból azokat az elemeket válogattuk ki, melyek minden alkalmazás készítésénél vagy működtetésénél szerepet játszanak. (Ez a csoportosítás egyáltalán nem akar egyes szoftvereket mások fölé vagy alá rendelni.) Az következő ábra, amely a szerencsés kiemelt területeket helyezi el a számítógép-rendszer egészében talán annak illusztrálására is alkalmas, hogy nem is mindig olyan egyszerű meghúzni a határokat a felhasználói szoftver, az operációs rendszer és a hardver között.



A felhasználó alkalmazások, az operációs rendszer és a hardver összefüggései

Programfejlesztési támogatás

Alkalmazásainkat elkészíthetjük akár gépi kódban is, de ez igencsak fárasztó, és a számítógépet is részletekbe menően ismerni kell hozzá (nem elegendő csak a processzor utasításkészletének ismerete!). Már a kezdetekben létrejöttek a programkészítés megkönnyítése céljából az ún. magasszintű nyelvek. Segítségükkel az emberi nyelvhez, gondolkodáshoz többé-kevésbé közelebb álló módon hozhatunk létre alkalmazói programokat. Egy fordítóprogram (compiler) lefordítja a magasszintű utasításokat a gép számára emészthető kódokra, egy szerkesztőprogram (kapcsolatszerkesztő, linker) pedig összehangolja, összeilleszti a különböző időben, vagy különböző programozók által megírt programrészeket.

Operációs rendszer

Van azonban egy további megoldandó probléma is. Az elkészített, vagy vásárolt alkalmazási programokat nemcsak egy bizonyos gépen szeretnénk futtatni, hanem szeretnénk hazavinni, munkahelyünkön használni vagy akár eladni. Úgy kell tehát elkészíteni őket, hogy a számítógépek minél szélesebb körén működőképesek legyenek. A felhasználók nem minden programhoz vásárolnak új számítógépet, tehát ugyanazon a processzoron, illetve rendszeren az alkalmazások több generációjának is futnia kell. Az állítás fordítva is igaz, azaz egy bizonyos alkalmazás többféle rendszeren is működőképes kell legyen (a video vezérlők, hangkártyák és egyéb perifériák százain kell futnia), sőt a processzor konkrét típusától lehetőleg függetlenül (természetesen csak ha a „családon" belül maradunk). Szükség van tehát a szoftverek egy olyan csoportjára is, amely a felhasználói alkalmazások és a hardver közé ékelődve „eltakarja" az alkalmazások elől a hardver különbözőségét és számukra egységes „virtuális" gépként viselkedik. Ez a szoftver az operációs rendszer. Az alkalmazásokat tehát legtöbbször az operációs rendszerhez és nemcsak a processzorhoz készítik.

Hálózatok

A számítógépek összekapcsolása, helyi, majd világméretű hálózatba szervezése, szinte beláthatatlan lehetőségeket nyitott meg az informatika fejlődése előtt, lassan (?!) egész életünkre hatással lesznek (pl. internetes kávéfőző, mikrohullámú

sütő stb.). A hálózati szoftverek tárgyalásánál nem kerülhetünk meg bizonyos hardver vonatkozásokat sem, de általános tendencia, hogy ez az oldal fokozatosan a specialisták kis csoportjának érdeklődési körébe szorul. Ez a terület erősen feszegeti már magának a számítástechnikának a határait is (az internetes telefon számítástechnika vagy telekommunikáció?), ezért ma sokkal közkedveltebb az informatika, illetve a telematika kifejezés.

A továbbiakban az operációs rendszerekkel kapcsolatos legfontosabb fogalmakat, feladatokat tárgyaljuk, majd megismerkedhetünk a programok összeállításának, készítésének egy megközelítésével, végül a számítógéphálózatok világára vetünk egy pillantást.

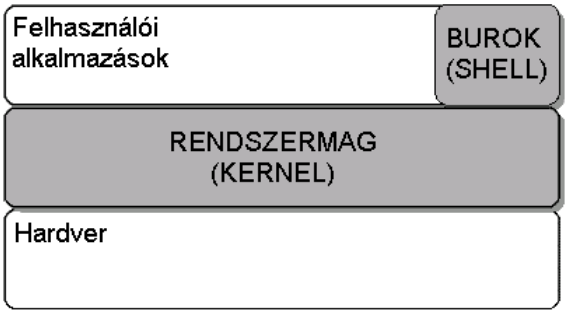
5.2 Az operációs rendszer

Az operációs rendszer egyik alapfeladata tehát az, hogy a felhasználót és a programozót megkímélje a közvetlen hardverkezelés keserveitől.

(A másik alapfeladat az erőforrások hatékony elosztása és a versenyhelyzetek kezelése, azonban ez a funkció az alkalmazások futtatása és a programkészítés szempontjából kevéssé érdekes, ezért itt nem is tárgyaljuk.)

5.2.1 Az operációs rendszer felépítése

Többfeladatos rendszerekben (ma már gyakorlatilag minden operációs rendszer ilyen) az operációs rendszer csak úgy tudja ellátni feladatát, ha áthatolhatatlanul beékelődik a felhasználói szoftverek és a hardver közé. Gondoljunk csak egy egyszerű példára: mi történne, ha egyszerre több futó alkalmazás reagálna az egér mozgására, vagy a programok minden koordinálás nélkül írogatnának a nyomtatóra. Tehát minden az operációs rendszeren keresztül, annak tudtával és beleegyezésével történik. Az operációs rendszereken belül meg szokták különböztetni a rendszer magját, a kernel-t, és a felhasználók felé közvetlen kapcsolatot biztosító részt, a burkot, a shell-t. Ez utóbbi tulajdonképpen felfogható egy olyan felhasználó alkalmazásként is, amely különösen kedves az operációs rendszer szívének.



A shell-ről a későbbiekben még lesz szó, vizsgáljuk meg alaposabban a kernel-t, pontosabban annak hardver, illetve szoftver felöli felületeit!

5.2.1.1 A rendszermag alsó (hardver) felülete

Az I/O eszközök általában megszakítás kéréssel jelzik, ha kiszolgálásra van szükségük, a processzor pedig – ha más, fontosabb dolga nem akad éppen – elindítja az operációs rendszer megfelelő rutinját. Azt, hogy éppen melyik rutin a megfelelő, a külön megalkotott megszakítás vezérlő dönti el annak alapján, hogy a kérést reprezentáló jel melyik vezetéken érkezett (PC-kben egyszerűen megszorozza a vezeték sorszámát 4-gyel, és az így kapott memóriacímről – a megszakítási vektortáblából – kiolvassa a rutin címét.) A kiszolgáló rutin az esetek többségében a hardvert kell vezérelje (például a lemez olvasófejét kell egy adott helyre irányítani). A hardver oldal azonban – mint már említettük – meglehetősen változatos képet mutat, ezért ebben a kérdésben (is) az operációs rendszer némi segítségre szorul. Mennyivel egyszerűbb lenne, ha minden periféria ugyanolyan képet mutatna! A probléma kezelésére több megoldás is született, nézzünk néhány tipikus példát!

1. **Klasszikus perifériák. Billentyűzete, monitora például majdnem minden számítógépnek van, attól függetlenül, hogy milyen operációs rendszer található rajta. Másrészt az operációs rendszer maga is lemezről töltődik be, tehát szükség van operációs rendszertől független lemezkezelésre is. Ezen alapvető funkciókat ellátó rutinokat az alaplapon olyan formában érdemes megvalósítani, hogy kikapcsoláskor is megmaradjon a programkód (Ezt a feladatot látja el a BIOS – Basic Input Output System – alapvető ki- és beviteli rendszer).**

2. **Egységes hardverfelület. Ha a merevlemezeket, CD-olvasókat belső felépítésüktől, tulajdonságaiktól függetlenül olyan elektronikával látjuk el, amely a többi egység felé már egyformán viselkedik, nagy gondot vehetünk le az operációs rendszer válláról. (IDE - Integrated Device Electronics - beépített eszközvezérlő elektronika.)**

3. **Eszközmeghajtók a gyártótól. Az operációs rendszer ez esetben egy egységes, szabványos felületet mutat az I/O eszközök felé (HAL - Hardware Abstraction Layer - elvonatkoztatott hardver felület), az eszközök ehhez a felülethez történő illesztését biztosító programocskát a hardvergyártó hivatott szállítani (Device Driver - eszközmeghajtó). Olykor – például a VGA kártyák esetén – az eszközmeghajtó program magán a kártyán helyezkedik el egy ROM-ban.**

4. **Intelligens perifériák. Készítsünk olyan interfészt, amely az operációs rendszer által feltett kérdésekre értelmes válaszokat tud adni, így meg lehet vele „beszélni" a szükséges beállításokat, az eszközvezérlés módját. Nem kell tehát semmi mást tenni, mint áramkörileg csatlakoztatni az eszközt, és (a kommunikáció után) már működik is (PnP – Plug and Play – csatlakoztasd és használd).**

5.2.1.2 A rendszermag felső (szoftver) felülete

A felhasználói alkalmazások kommunikációja a kernellel nagyon hasonlít a hardverkezelés folyamatához. Lényegében itt is megszakításkezelésről van szó, de a kérés közvetítője nem egy vezeték, hanem egy speciális gépi kódú utasítás (INT), melynek paramétere (szintén 4-gyel szorozva, azaz kettővel balra léptetve) határozza meg a kiszolgáló rutin címét. Ezt a szoftvereredetű megszakítást nevezzük rendszerhívásnak (system call). A paraméterátadásra, az eredmény visszaadására, valamint az esetleges alfunkció kiválasztására a regiszterek szolgálnak. A szoftvermegszakítások funkciója és sorszáma közötti kapcsolat természetesen rögzített, és az operációs rendszerek készítői – jól felfogott érdekükben – arra is ügyelnek, hogy a fejlettebb rendszerek elődeikkel kompatibilisek legyenek.

5.2.1.3 A rendszermag „magja” (röviden)

A rendszermag közvetít a hardver és a felhasználói szoftver között, itt történik az egyes kiszolgálás kérések koordinálása, a rendelkezésre álló erőforrások „igazságos” és hatékony elosztása. A közvetlen perifériakezelő funkciókon felül gazdálkodni kell a processzor idővel, a memóriával, készenléti sorokat kell fenntartani azon folyamatok számára, amelyek éppen várakozni kényszerülnek, illetve átmeneti tárolókat létesíteni az eszközök különböző sebességét kiegyenlítendő. Nagyon fontos, gyakran kiemelt feladat az állománykezelés, tehát érdemes ezt egy kicsit részletesebben is megvizsgálni.

5.2.2 Állománykezelés

Mivel a számítógépeket adatok, illetve programok kezelésére és tárolására használjuk, minden operációs rendszer nagy súly fektet ezek könnyű és gyors elérhetőségének biztosítására (pl. DOS – Disk Operating System, azaz lemezkezeléssel kiegészített operációs rendszer). Az adatok állományokba (fájlokba) csoportosíthatók így rájuk a továbbiakban egyszerűen egy logikai névvel hivatkozhatunk. Nem kell tehát a felhasználóknak azzal foglalkozniuk, hogy állományaik egy lemez melyik sávján, azon belül is melyik blokkban helyezkednek el, hanem nyugodtan rábízhatják magukat az operációs rendszerre, mely a megadott név alapján előkeresi számukra a megfelelő programot, vagy adatokat.

Állomány (fájl, file): Háttértárolón tárolt adatok együtt kezelt, névvel rendelkező összefüggő csoportja.

A fájl fogalmán általában tárolt adatokat értünk, de léteznek olyan operációs rendszerek (például a UNIX), ahol ezt a szemléletet minden külső adatfolyamra, így a képernyő tartalomra és a billentyűzetre is általánosították. A fájl fogalom kiszélesítése a programozó számára rendkívül kellemes, ezért egyes elemeit minden operációs rendszerben megtalálhatjuk. A programozó dolga nagyban leegyszerűsödik, mivel nem kell törődnie azzal, hogy programja milyen perifériáról kap majd bemenő adatokat, vagy eredményeit nyomtatóra, képernyőre, fájlba kell kiírnia, vagy csupán át kell adnia egy másik programnak, elegendő egy fájlra hivatkozni.

5.2.2.1 Elnevezések, hivatkozások

A felhasználók és programok fájlnevek segítségével hivatkoznak a kívánt adatcsoportokra. Ezeket a neveket általában a felhasználó adja, de előfordulhat az is – ideiglenes állományok esetén –, hogy az operációs rendszert kérjük fel névadásra, mely a többi elnevezés ismeretében ügyel az egyediségre.

A fájlneveket alkotó karakterek lehetséges halmaza, a név lehetséges hossza, valamint szerkezete az aktuális operációs rendszer függvénye. A nevek több összetevőből állhatnak, melyeket valamilyen speciális karakter választ el egymástól. A leggyakoribb a két összetevőből álló név, ahol az első az egyedi, a fájl tartalmára utaló rész, a második rész a fájl jellegére (szöveg, program, adatbázis) utal. Az elnevezésekre vonatkozó szabályok rendkívül változatosak, nézzünk néhány példát!

Az MS-DOS, személyi számítógépeken klasszikusnak számító operációs rendszer nagyon szigorú szabályokat követel meg. A fájlnév két komponensből áll, az első rész, a név, minimum 1, maximum 8 karakterből állhat, az elválasztó pont után következő kiterjesztés legfeljebb 3 karakterből állhat (itt alsó határ nincs, azaz el is maradhat). A fájlneveket alkotó karakterek az ASCII kódtábla nagybetűi, számai és egyes speciális karakterei (pl. ~!@#$%^&_-{}) közül kerülhetnek ki. A DOS nem tesz különbséget kis- és nagybetűk között, az operációs rendszer minden karaktert automatikusan nagybetűvé konvertál. Az újabb DOS változatok megengedik ugyan ékezetes karakterek használatát is, de ezek használata az egyes gépek eltérő konfigurációja, valamint az említett karakterkonverzió miatt hord némi veszélyeket magában.

A UNIX fájlnevek hossza maximum 255 karakter lehet, és tetszőleges számú, a DOS-hoz hasonlóan ponttal elválasztott összetevőkből állhatnak. A UNIX megkülönbözteti a kis- és nagybetűket, így a FÁJL, fájl és Fájl három különböző állományt jelöl. A speciális, vagy ékezetes karakterek használatára nagyjából a DOS szabályai érvényesek, annak veszélyeivel együtt, azaz nem szerepelhetnek olyan karakterek, melyek az operációs rendszer számára valamilyen speciális jelentéssel bírnak.

A Windows 95 speciális kettős elnevezésrendszert használ, részben a dokumentum-orientált szemlélet, részben a DOS kompatibilitás miatt. A Windows 95-ben minden fájlnak két neve van, egy rövid, és egy hosszú. A rövid fájlnevek követik a hagyományos DOS konvencióit, azaz 8+3 karakteres felépítésűek. Minden fájlhoz tartozik azonban egy legfeljebb 250 karakter hosszúságú, tetszőleges karakterekből álló név is, ugyancsak maximum három karakteres kiterjesztéssel. A Windows 95 alkalmazások ezt a hosszú nevet látják, ebből képezik a rövid nevet. Az első nyolc karakter automatikus választása azonban nem ad kielégítő megoldást, hiszen lehet sok olyan hosszú név is, melynek eleje megegyezik. A Windows 95 úgy oldja meg ezt a kérdést, hogy a hosszú fájlnévből eltávolítja szóköz karaktereket, majd az első 6 karakter után egy elválasztó jelet (~), azután egy sorszámot ad.

A fájlokra a felhasználói folyamatok általában a pontos nevük alapján hivatkoznak, de lehetséges a név egy részlete alapján is keresni egy, esetleg több, a mintára illeszkedő állományt. A többféle tartalommal is kitölthető, úgynevezett helyettesítő karakterekre (egyéb elnevezésük wildcard, joker, maszk, metakarakter) példa a DOS esetén a ‘?’ mely tetszőleges karakter egyetlen előfordulását jelenti, valamint a ‘*’, mely tetszőleges tartalmú és hosszúságú karakterlánc helyett állhat. A UNIX a fenti lehetőségeken kívül ismeri a karaktercsoport lehetőségét is, melynek lehetséges értékeit szögletes zárójelek között kell megadni. Az ‘[ABC]’ kifejezés például állhat egy ‘A’, egy ‘B’ vagy egy ‘C’ karakter helyett.

5.2.2.2 Az állományok egyéb jellemzői

A fájlokhoz a nevükön kívül egyéb információk is tartoznak, melyeket részben az operációs rendszer ad, részben a felhasználó. A többletadatok a különböző operációs rendszerekben a fájlnevekhez hasonló nagy változatosságot mutatnak.

Az utolsó módosítás időpontja mindig szerepel az adatok között. Újonnan alkotott, még nem módosított fájl esetén ez a paraméter természetesen megfelel a létrehozás időpontjának. A fájl mérete szintén nagyon fontos információ, általában bájtokban adják meg. Több felhasználós rendszerek esetén rögzítésre kerül a fájl tulajdonosa is.

A fájl állapotára utaló jelzőbiteket összefoglaló néven attribútumoknak nevezzük. (Az alábbiakban a DOS és a UNIX által használt jellemzők egyvelegét ismertetjük, melyek együtt talán képet adhatnak a lehetséges funkciókról). Az attribútumok jelezhetik az archiválást végző program számára, ha egy fájl megváltozott az utolsó mentés óta, azaz archiválandó (archive needed), az operációs rendszer többi folyamata számára, hogy a fájl csak olvasható (read only), rendszerfájl (system), rejtett állomány (hidden). A speciális tulajdonságú, adminisztrációs fájlokat is attribútumok különböztetik meg a felhasználói állományoktól, különböző jelzőbitjei vannak a katalógusoknak (directory) (amiket nemsokára tárgyalunk), a szimbolikus hivatkozásoknak (link), az ideiglenes, adatcsere fájloknak (pipe).

Néha (például a UNIX esetében) az felhasználók hozzáférési jogait is a fájlok jellemzői között találjuk, azaz itt kerül szabályozásra, hogy ki írhatja, olvashatja az állományt, vagy - programfájl esetén - hajthatja vére azt.

5.2.2.3 Katalógusok (directory)

A katalógus olyan speciális, adminisztratív célokat szolgáló fájl (ezért elnevezésére is ugyanazok a szabályok vonatkoznak, mint az állományokéra), amely a lemezen lévő fájlok adatainak listáját tartalmazza. Magát a katalógusfájlt a felhasználók

általában közvetlenül nem láthatják, tartalmukat rendszerhívások segítségével érhetjük el.

(Az eredeti angol elnevezést, a directory-t magyarul gyakran könyvtárnak is nevezik, de tartalmilag helyesebb, és a függvény- vagy szubrutin könyvtáraktól való megkülönböztetést is jobban szolgálja a katalógus szó.)

Katalógus (könyvtár, directory): Olyan speciális állomány, mely az adatokat tartalmazó állományok jellemzőit tárolja, lehetővé teszi csoportosításukat.

Ha egy folyamat egy fájlra hivatkozik, az operációs rendszer először a katalógust vizsgálja meg, hogy létezik-e egyáltalán ilyen. Pozitív válasz esetén jön a következő ellenőrzés, hogy a felhasználónak, illetve az általa indított folyamatnak van-e joga a kívánt művelethez. Amennyiben a fájl is létezik, és a jogosultságok is rendben vannak, akkor kezdődhet meg a katalógusban lévő, a fizikai elhelyezkedésre utaló információ alapján a művelet végrehajtása.

A mai operációs rendszerek szinte kivétel nélkül hierarchikus, vagy más néven fa struktúrájú katalógus rendszert használnak. A hierarchikus rendszer kiindulópontja a gyökérkönyvtár, mely tartalmazhat fájlokat és alkatalógusokat (subdirectory), ez utóbbiak szintén tartalmazhatnak fájlokat és alkatalógusokat és így tovább. Az elrendezés hasonlít egy fejre állított fára (innen az elnevezés), melynek gyökere a gyökérkönyvtár, ágai az alkatalógusok, levelei a fájlok. A hierarchikus felépítés a fájlok keresésének idejére is kedvező hatással van, mivel a lineáris rendszerekkel szemben itt alkalmazható a lényegesen gyorsabb bináris keresés.

5.2.2.4 Kötetek (volume)

A háttértárat az operációs rendszerek általában logikai kötetekként kezelik (pl. DOS, Windows, NetWare, de a Unix NEM!). A kötetek speciális nevet, esetleg címkét kaphatnak.

Kötet (meghajtó, volume, drive): Az a névvel ellátott logikai egység, mely segítségével az operációs rendszer a számítógép háttértárolóit kezeli.

A személyi számítógépek világában az angol ábécé 26 betűjét használják egy kettősponttal megtoldva (pl. a:, b:, c:, .. z:). Mindegyik kötethez tartozik egy gyökérkatalógus, ebben helyezkednek el a további alkatalógusok, illetve állományok adatai. A DOS hagyományokon alapuló operációs rendszerekben az elsődleges hajlékony lemez meghajtó az A:, az elsődleges merevlemez a C: nevet viseli, a hálózati meghajtók az F: .. Z: tartományt kapták.

Meg kell jegyezni, hogy hálózatok használata esetén a 26 betű kevésnek bizonyulhat, ilyenkor a kötetnév általában több karakterből áll, pl. SYS:, USER:.

5.2.2.5 Hivatkozások

Ha egy állományra hivatkozni akarunk, nemcsak a nevét kell megadni, hanem azt a kötetet, illetve katalógust is, amelyben megtalálható.

Az abszolút hivatkozás esetén a kötet megadása után a gyökér katalógustól kezdődően az összes közbülső katalógus nevének felsorolása után jutunk el a fájlhoz.

Az aktuális kötet / katalógus(current drive / directory) fogalmának bevezetésével egyszerűsíthető a hivatkozás, az eddigi, a gyökérkatalógustól induló abszolút címzéssel szemben alkalmazható az aktuális helyzethez viszonyított relatív címzés. Az aktuális kötet, illetve (kötetenként) az aktuális katalógus nevét az operációs rendszer saját adatterületén tárolja. Ez gyakran a képernyőn is megjelenik (pl. DOS prompt) vagy egyszerű paranccsal kiíratható (pl. Unix cwd).

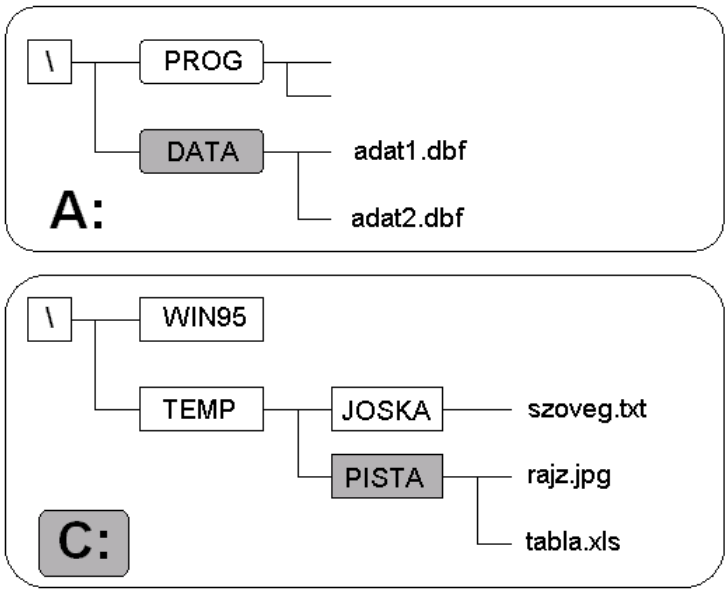
A relatív hivatkozás tehát a fájl megadásának az a módszere, amelyben a gyökér katalógus helyett az aktuális katalógus a kiinduló pont.

A gyökérkatalógus neve általában ‚\' (backslash), azaz egy fordított törtvonal.

Fa struktúrát alkalmazó rendszerekben az egyes katalógusoknak lehetnek szülei és gyermekei. A szülőkkel a szorosabb kapcsolat természetes, ezért igazi nevén felül, már csak szülői mivoltából is indokolt külön névvel illetni. A szülő katalógus neve általában (a gyermek szemszögéből) `..' (azaz két darab pont), az aktuális katalógus relatív neve `.' (azaz egy darab pont).

Az abszolút és relatív hivatkozás között a legszembetűnőbb formai különbség, hogy az abszolút hivatkozás (a meghajtó neve után) mindig tartalmaz ‚\' karaktert.

Nézzünk egy példát!



Példa fa struktúrájú katalógusrendszerre

A példában két kötetünk van, az aktuális meghajtó a `C:` , az aktuális katalógus az `A:` illetve `C:` meghajtón rendre a `\DATA`, illetve a `\TEMP\PISTA` (az ábrán sötétebb árnyalattal jelölve).

Abszolút címzések:

```
A:\DATA\ADAT1.DBF
C:\TEMP\PISTA\RAJZ.JPG
C:\TEMP\JOSKA\SZOVEG.TXT
```

Relatív címzések:

`A:ADAT1.DBF`	**(aktuális katalógus)**
`C:RAJZ.JPG`	**(aktuális katalógus)**
`\TEMP\JOSKA\SZOVEG.TXT`	**(aktuális kötet)**
`..\TEMP\JOSKA\SZOVEG.TXT`	**(aktuális kötet, katalógus)**

5.2.2.6 Közvetett elérés, keresési útvonal

A gyakran használt, de a logikai fájlrendszerben különböző helyeken található programok könnyed indítására a relatív hivatkozás lehetősége sem nyújt elfogadható megoldást. Legtöbb operációs rendszernél alkalmazható a közvetett elérés, vagy a keresési útvonal technikája.

Közvetett elérés (link) esetén létrehozható egy olyan kis méretű állomány, amely az indítandó program futtatásához szükséges adatokat tartalmazza. Ezek a kicsi fájlok aztán tetszőlegesen elhelyezhetők, csoportosíthatók (például kitehetők a Windows 98 „munkaasztalára").

A keresési útvonal (path) nem más, mint az operációs rendszer egy „változója". Egy parancs begépelése, vagy egyes fájlmegnyitások esetén az operációs rendszer először az aktuális köteten, az aktuális katalógusban keresi az állományt, de kudarc esetén sem esik kétségbe, végigjárja a PATH változóban felsorolt katalógusokat, és a hibaüzenet csak ennek eredménytelensége esetén várható.

Pl.

```
PATH C:\WINDOWS;C:\NORTON;C:\WINDOWS\COMMAND
```

5.2.2.7 Fájlok fizikai elhelyezése

A számítógépek kiemelkedő fontosságú háttértára a merevlemez (hard disk drive, winchester), ezért a fájlok elhelyezését ezen a példán mutatjuk be, ennek is azt a változatát, amelyet az IBM PC-k terjesztettek el. Az egyszerűség kedvéért még azt is feltételezzük, hogy csak egyetlen merevlemez található a képzeletbeli számítógépünkben.

A merevlemezeken a tárolható legkisebb kezelhető egységek a blokkok, azonban ezek gyakran értelmetlenül kicsinek bizonyulnak, így az operációs rendszer számára a legkisebb egység a több blokkból álló fürt, az úgynevezett cluster. A lemezeket ezentúl úgy kezelhetjük, mintha egy adott számú clusterből álló tárolósorozat lenne (a fejek mozgatását a lemezkorongok között bízzuk az operációs rendszerre).

Az operációs rendszer elhelyezkedése, betöltése

A merevlemezek részekre, úgynevezett partíciókra oszthatók. (A partíciók maximális száma 4.) A felosztásra vonatkozó adatok, azaz az egyes partíciók kezdő, illetve utolsó clusterének sorszáma a lemez legelején lévő táblázatban, az úgynevezett partíciós táblában találhatók. Ugyanitt van az a kis programocska is, a Master Boot Record (MBR), amely a táblázat adatait értelmezni tudja, és a megfelelő partícióról elindítja az operációs rendszer betöltését. (Ha az operációs rendszer fogalmát tágabb értelemben használjuk, helyesebben azt kellene mondani, hogy az operációs rendszer maradékának betöltését, hiszen a BIOS már rendelkezésre áll!) De a sok közül melyik a megfelelő partíció? Egy lemezen – különböző partíciókban – több operációs rendszer is lehet, azonban közülük csak az egyik aktív. Az aktivitást szintén a partíciós tábla egy mezője tárolja, amelyet az operációs rendszer segédprogramjával (pl. FDISK), vagy különböző betöltést felügyelő (boot menedzser) programokkal lehet megváltoztatni. Minden partíció elején megtalálhatjuk a Boot Recordot, és a hozzá tartozó adattáblákat, melyek azonban – az MBR-rel ellentétben – már operációs rendszer specifikusak, és a konkrét operációs rendszer betöltését végzik.

Tehát az operációs rendszer betöltésének lépései:

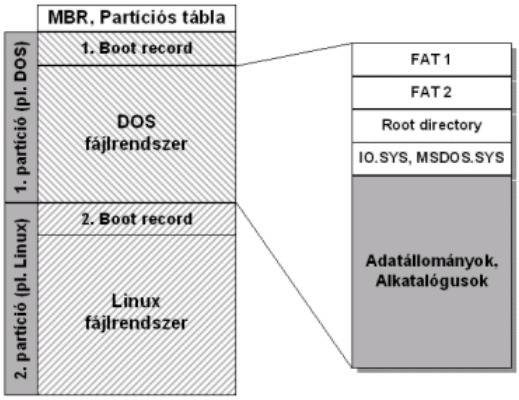
1. **Elindul az MBR-ben lévő program, és a partíciós tábla adatai alapján betölti az aktív partíció Boot rekordjában található programot, és átadja annak a vezérlést.**
2. **A Boot rekord programja az adott partícióra (azaz az adott operációs rendszerre) jellemző paraméterek segítségével megkeresi az operációs rendszer első állományát (pl. IO.SYS) és továbbadja a vezérlést.**
3. **A többi már az operációs rendszer dolga ...**

Az operációs rendszer állományai (a partíció határokhoz képest) állandó helyen találhatók, azonban a felhasználói állományok esetén ez a módszer természetesen nem alkalmazható.

Felhasználói állományok elhelyezése

Minden partíció (ritka kivétellel) egy logikai kötetként jelenik meg. A továbbiakban egy partícióra szorítkozunk, ezen belül vizsgáljuk a fájlok elhelyezését.

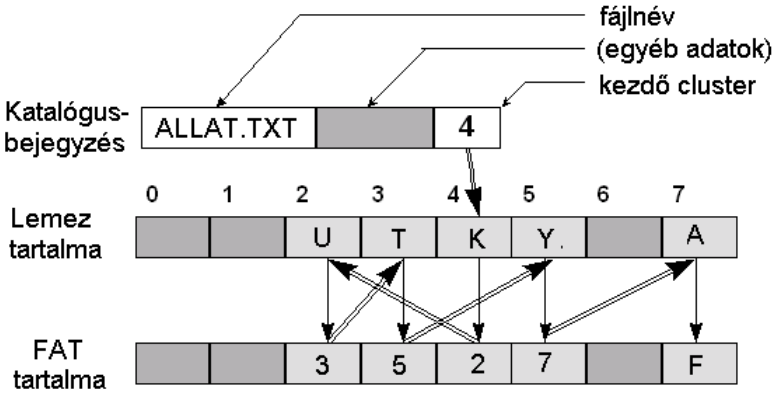
Ahhoz, hogy adatokat, programokat tárolhassunk a köteten, azt elő kell készíteni, létre kell hozni rajta az adminisztrációhoz feltétlenül szükséges táblázatokat. Ez az eljárás a lemez formázása, melynek során felmásolódhat az operációs rendszer magja, lefoglalódik a hely a gyökérkatalógusnak (root directory), amely a fájlrendszer kiinduló pontja lesz, és létrejön (biztos, ami biztos két példányban) a fájlok elhelyezését regisztráló táblázat, a FAT (File Allocation Table). Nézzük, hogy néz ki egy partíció!



Állományok elhelyezése a lemezen

Az állományok fizikai elhelyezésére több módszer létezik, de az alábbiakban csupán a PC-s világban leginkább elterjedt láncolt elhelyezést mutatjuk be. A katalógusban csak a fájl kezdő clusterét kell megadni, az összes többi adatot a fájl elhelyezési tábla (FAT) tartalmazza. A táblázatnak ugyanannyi eleme van, mint ahány cluster a lemezen és minden rekesz tartalma a fájl következő clusterére mutató sorszám, ha van ilyen, csupa egyes (azaz 16 bites FAT esetén FFFF), ha ez volt az utolsó blokk. Ezzel elérhető, hogy az állományoknak nem feltétlenül kell folytonosan elhelyezkedniük a lemezen, mivel az egymást követő clusterek sorszáma a FAT alapján megállapítható.

Legyen példánkban egy cluster mérete mindössze 1 bájt, és legyen csupán 8 clusterünk. A memóriában lévő 5 bájtot `(KUTYA) ALLAT.TXT` néven szeretnénk lemezre menteni. (Ebben az esetben ez a művelet az operációs rendszernek nem túl jól sikerült, de a nyilakat figyelmesen követve felfedezhetjük az eredeti szót.)



Állományok láncolt elhelyezése a lemezen

A FAT meglehetősen nagy táblázat lehet, és szerepe döntő. Ezen kívül nagyon sűrűn kell használni, ezért a memóriában kell tartani, ami még szűkösebb, mint a háttértár. A FAT sérülése esetén nagy valószínűséggel a kettészakadt fájlt visszaállítani nem lehet. A láncolási módszert alkalmazó operációs rendszerek, például a DOS, a Windows és a NetWare a biztonság kedvéért két ilyen táblázatot tartanak fenn.

5.2.3 A felhasználói felület

A legelső dolog, amivel a felhasználó találkozik, az operációs rendszerek felhasználói felülete.

Az eddigiekben megmutattuk, hogy az operációs rendszerek szinte minden mozzanatot felügyelnek. Kiszolgálják, ütemezik a folyamatokat, biztosítják számukra a megfelelő erőforrásokat. A mai rendszerek túlnyomó többsége interaktív, tehát a felhasználó is igényli, hogy befolyásolhassa a rendszer működését. Miért ne biztosíthatnák ezeket a szolgáltatásokat nemcsak a felhasználói folyamat, hanem közvetlenül a felhasználók számára is?

Van még ezen kívül egy egyáltalán nem elhanyagolható szempont. Lehet, hogy egy operációs rendszer sokat tud, de egyet biztosan nem. Nem tudja kitalálni, hogy egy felhasználó mit akar, milyen programot szeretne indítani. A felhasználói felületre tehát feltétlenül szükség van.

5.2.3.1 Karakteres felhasználói felület, a parancsértelmező

A parancsértelmezők az interaktív rendszerek kialakulásával jelentek meg, a parancsnyelvek továbbfejlesztéseként. Feladatuk az operációs rendszer szolgáltatásainak biztosítása az interaktív felhasználó számára. A funkció többféle elnevezésével találkozhatunk attól függően, hogy a feladatkör mely részét tekinthetjük elsődlegesnek. Gyakori a shell (burok, héj) elnevezés (Unix, DOS) mely arra utal, hogy a felhasználói felület burokba zárja, eltakarja a rendszer magját, a kernelt. A command interpreter (parancsértelmező) kifejezést azok használják, akik számára az operációs rendszerek fő feladata az adott parancsok kiszolgálása. Olykor találkozhatunk még a monitor (felügyelő) névvel, ha a folyamatok kezelése, nyomon követése az elsődleges. A grafikus rendszerek (pl. a Windows-család tagjai) a feladatkört több részre osztják, így létezik program, fájl-, nyomtató-, feladat (task) kezelő moduljuk.

A továbbiakban a megjelenés formáitól eltekintünk, a funkció megjelölésére a „shell" nevet használjuk. A shell-ek, mint már említettük, tulajdonképpen olyan felhasználói programok, amelyek az operációs rendszer lelkéhez közel állnak. Működésük nem igényel kivételes, privilegizált módot. Bizonyos esetekben - ha egy felhasználó mindig kizárólag ugyanazt a programot futtatja - a shell el is maradhat, a felhasználó számára a felületet maga a felhasználói program jelenti.

Más esetben az egyes felhasználók, vagy alkalmazási módok más-más felhasználói felületet kedvelnek. A többféle felület a UNIX-nál természetes (C, Bourne-shell), de ha egy kicsit tágabb értelmezést is megengedünk, shell-nek tekinthető a DOS esetén a Norton Commander, a PCShell vagy a DOS-Shell, Windows esetén a Fájl-kezelő és a Programkezelő is. A shell alapvető feladatai tehát:

- **programindítás, programkezelés**
- **egyéb, operációs rendszer funkciók felhasználói szintű biztosítása (általános értelemben vett fájlkezelés)**

A DOS parancsok általános felépítése a következő (a szögletes zárójel azt jelenti, hogy az így jelölt elem olykor elmaradhat):

```
parancs [paraméterek] [/kapcsolók]
```

A parancs határozza meg az elvégzendő feladatot, a paraméterek mutatják, hogy azt mely objektumokon kell végrehajtani, a kapcsolók (switch) kissé módosítják a parancs működését.

A legfontosabb DOS parancsok:

`programnev`	**a `PROGRAMNEV` nevű program indítása**
`FDISK`	**a partíciós tábla módosítása**
`FORMAT drive:`	**kötet formázása**
`drive:`	**az aktuális meghajtó megadása**
`MD directory`	**alkatalógus létrehozása**
`CD directory`	**az aktuális katalógus megadása, váltása**
`RD directory`	**(üres) katalógus törlése**
`DIR`	**aktuális katalógus tartalmának listázása**
`COPY honnan hova`	**állomány másolása**
`REN mit mire`	**állomány átnevezése**
`DEL file`	**állomány törlése**
`TIME`	**idő kiírása/beállítása**
`DATE`	**dátum kiírása/beállítása**

A fenti parancsok ismerete általában elegendő. Ha valakinek mégis hiányérzete támad, a `HELP` parancs, vagy a parancs kiadása a /? kapcsolóval sokat segíthet.

5.2.3.2 Grafikus felhasználói felületek

A felhasználó és a számítógép párbeszéde kezdetben a lyukkártyákra rögzített, majd a konzolírógépen begépelt parancsok segítségével történt, a közeli jövőben talán a kimondott szavak, majd a gondolatok irányíthatják gépeink működését, azonban jelenleg a leginkább elterjedtek, a legnépszerűbbek a grafikus felhasználó felületek. Az IBM kompatíbilis gépeken a Microsoft Windows különböző variációival találkozunk, a Apple a MacOS-t használja, a Unix, és általában a nagygépes operációs rendszerek az X-Window rendszert valósították meg.

A karakteres világ parancsértelmezője nem sokban különbözik egy egyszerű felhasználói programtól. Egyfeladatos rendszerekben (single task, pl. DOS) a parancsértelmező fut, amikor éppen semmi más nem történik, háttérbe vonul, amikor egy felhasználói program kap vezérlést, és annak végeztével újra előbukkan, és várja a következő parancsot.

Többfeladatos, karakteresfelület-használó rendszerekben (multitask, pl. Unix) ugyan futhatnak folyamatok a háttérben, azonban ezek működését csak meglehetősen nehézkesen lehet nyomon követni, illetve befolyásolni. Milyen jó lenne, ha minden folyamatra nyithatnánk egy ablakot, amelyen keresztül megfigyelhetjük a történéseket, beavatkozhatunk, vagy ha úgy tartja kedvünk, az ablakot be is csukhatjuk (ez persze nem jelent azt, hogy a folyamat is leáll)!

Az ablakozó technikák tehát a többfeladatos rendszerekhez kötődnek, fejlődésük kezdete a 80-as évek elejére tehető.

5.2.3.3 Segédprogramok, alrendszerek

Már a programfejlesztői támogatás esetén is nehéz eldönteni, hogy hol húzzuk meg az operációs rendszer határait, még inkább így van ez a felhasználó által közvetlenül használt programok esetén.

A felhasználói programok leggyakrabban használt példája az interaktív rendszerekben használt tolmács nyelv, a parancs-interpreter. A parancsnyelvet gyakran nevezik shell-nek (héj, burok), mivel a felhasználó csak ezen keresztül érintkezhet a maggal, a kernellel. A parancsnyelv nem más, mint a rendszerhívások, illetve rendszerhívások sorozatának kiadására szolgáló magas szintű eszköz, ezért vitathatatlanul az operációs rendszerhez tartozik, annak ellenére, hogy azonos funkciók ellátására más shellek is írhatók.

A gyakran használt operációs rendszer szintű feladatok megoldására szolgáló segédprogramok (lemezformázás stb.) szintén közel állnak az operációs rendszerhez.

Gyakori, hogy egy speciális felhasználói igényt nem az operációs rendszer módosításával elégítenek ki, hanem egy a segédprogramok körét kiegészítő, úgynevezett alrendszerrel bővítik az operációs rendszer magas szintű szolgáltatásait. A programfejlesztési támogatás is felfogható alrendszerként, de egyre terjednek a grafikus és adatbázis alrendszerek is. Az alrendszerek alkalmazásának előnye, hogy nem kell módosítani hozzá az operációs rendszert, hátrányuk, hogy egy újabb szint ékelődik a felhasználó és az operációs rendszer közé. Az első interaktív rendszerek is ilyen alrendszerként kerültek megvalósításra.

A segédprogramok, alrendszerek leggyakoribb működési területei a következők:

Állományok kezelése. Az állományok másolása, áthelyezése, átnevezése, törlése, valamint a katalógusok létrehozása mellett, a segédprogramok között megtalálhatók a lemezek partícionálására, formázására szolgálók is. Tipikus segédprogram a Norton Commander, mely a leggyakrabban használt 10 fájlművelet végrehajtását teszi hallatlanul kényelmessé.

Programfejlesztés. Szinte valamennyi operációs rendszer tartalmaz egyszerű szövegfájlok előállítására alkalmas programot (editor), valamint szerkesztőt (linker). A Unix alapú rendszereknek gyakran szerves része a C fordító is.

Adatbáziskezelés. A számítógépes rendszerek jó része adatbázisokat kezel. Ennek támogatására olykor az operációs rendszerek külön eszközöket is tartalmaznak, néha, pl. az OS400 esetén szinte elválaszthatatlanul a többi funkciótól.

Kommunikáció. A hálózatok kezelése napjainkban egyre fontosabb. Az alapvető hálózati rétegeket megvalósító folyamatok ma már szinte kivétel nélkül az operációs rendszer szerves részét alkotják.

A segédprogramok, alrendszerek és felhasználói programok lényegében nem különböznek egymástól, gyakorlatilag csak azok eredete, súlya vagy funkciója dönti el, melyik programot vagy program csoportot melyik kategóriába soroljuk.

5.3 Programkészítés

A operációs rendszerek egyik feladata – mint említettük – a hardver elrejtése a felhasználók elől. A rendszermag a felhasználói folyamatok oldaláról úgy látszik, mintha egy olyan számítógép (virtuális gép) lenne, amely speciális, jellemző erőforrás kezelő utasításokkal rendelkezik. A felhasználói folyamatok írói, a programozók a hardver kezelést csak kernel szolgáltatásainak igénybevételével végezhetik. Ilyen feltételek mellett természetes, hogy az operációs rendszer készítőinek, a rendszer legjobb ismerőinek a rendszermag funkcióinak megvalósításán kívül támogatniuk kell a felhasználói programok készítését is. A rendszerhívások jól definiált rendszeréhez az assembly programozók közvetlenül, a magasabb szintű nyelvek kedvelői összetettebb eljárásokon keresztül férhetnek hozzá.

5.3.1 A forráskód elkészítése

A forráskód elkészítésére szinte minden rendszer biztosít egy szövegszerkesztőt (editor), amely a konzolról begépelt szöveget egy állományba menti. A programozási nyelv megválasztása erősen függ az elvégzendő feladattól.

A programozási nyelveket gyakran csoportosítják annak alapján, hogy a hardverhez milyen kapcsolat fűzi őket. Ebből a szempontból léteznek hardver orientált nyelvek (tipikus példája a Assembly), valamint probléma-orientált nyelvek (például Clipper, VBA stb.) Az előbbi csoport nyelveiben a programozás nem nagyon egyszerű, de az így született program a lehető leghatékonyabban képes a hardver lehetőségeit kihasználni (persze csak, ha a programozó is képes rá!). A probléma-orientált nyelvek fejlesztésénél arra törekedtek, hogy az adott feladatcsoport logikáját a lehető legjobban megvalósítsák, így a programozást megkönnyítsék, azonban ez a hatékonyság romlásával járhat. A két csoport között helyezkednek el az általános célú nyelvek (pl. C, Pascal), melyek segítségével a feladatok széles spektrumát lehet megoldani.

5.3.2 Fordítás

A forráskód alapján azt a gépi kódú programot, amely az adott processzor által ismert utasításokat, valamint a rendszerhívásokat megvalósító szoftver megszakításokat tartalmazza a fordítóprogram (compiler) készíti el. Az így keletkező tárgykódú (object - OBJ) modul az ugrások, változók abszolút címei helyére általában még a modul elejéhez viszonyított relatív címet helyettesít, úgy mintha feltételezné, hogy a program modul a 0 címtől kezdődne. Az abszolút címek már csak azért sem szerepelhetnek a tárgykódban, mert egy program rendszerint több, külön fordított modulból áll. Az egyes programrészleteket természetesen készítheti a felhasználó, de az operációs rendszer is tartalmazhat előre elkészített, rendszerkönyvtárakba (library - LIB) rendezett, elsősorban perifériakezelő, vagy a felhasználói felület kialakítását segítő rutinokat. A modern operációs rendszereknek csak a leginkább hardver közeli részei íródnak gépi kódban, nagy részük a kifejezetten e célra kifejlesztett C nyelven készül, így, ha más nem is, de a C fordító az operációs rendszer születésével egyidőben már rendelkezésre áll.

E fejezetben nem foglalkozunk az interpreterekkel, azaz a soronként fordító és végrehajtó magasszintű nyelvekkel.

A kernel programozói szempontból egy függvény- vagy eljáráskönyvtárnak tekinthető. A programok készítéséhez szükséges információ leírásának és a segédprogramoknak összefoglaló neve a legtöbb esetben API (Application Programming Interface).

Az alkalmazások írói számára biztosított rutinok általában erősen függnek a szolgáltatások szintjétől. A DOS kb. 100 eljárása alapvető operációs rendszer funkciók elérését teszi lehetővé, míg a Windows több mint 1000 függvénye a felhasználók által közvetlenül megtapasztalt felület kialakítását is támogatja. A grafikus felületek egyedülálló népszerűségének éppen ez az egyik fő oka: a programozók - többek között saját jól felfogott érdekükben - az operációs rendszer által kínált magas szintű lehetőségeket használják fel, így mind a látvány, mind a működési mód hasonló. Ez a programozási stílus a végfelhasználó számára is nagyon előnyös. Elég egy programot megtanulni, az összes többi szinte magától értetődő, a lényegtől nem vonják el a figyelmet a technikai részletek.

5.3.3 Szerkesztés

A szerkesztő (linker) feladata a tárgykódú modulok címeinek összehangolása, kereszthivatkozások feloldása, a betölthető program (executable - EXE) előállítása. (A szerkesztőt gyakran kapcsolatszerkesztőnek nevezik, azért, hogy véletlenül se lehessen összekeverni a szövegszerkesztővel.) A szerkesztett program még mindig nem tartalmazhat konkrét memória címeket. Mivel a rendszerben egyszerre több folyamat fut, előre nem tudható, hogy az elkészített program a memória mely tartományára kerül. Az operációs rendszer dolgát mindenesetre célszerű megkönnyíteni úgy, hogy amennyiben a processzor lehetővé teszi, a címeket egy, a program kezdetén beállítandó alaphoz (bázis) képest adjuk meg.

A programkészítés fent vázolt menetét a korszerű integrált fejlesztőrendszerek észrevehetetlenné teszik. Látszólag egy lépésben végzik, sőt program belövési támogatással (lépésenkénti végrehajtás, változók, memóriatartalom kiírása) egészítik ki.

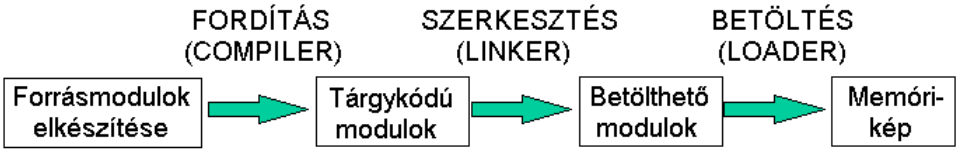
5.3.4 Betöltés, dinamikus könyvtárak

A betöltő (loader) program már az operációs rendszer magjához tartozik, feladata a végrehajtható program elhelyezése a memóriában, a bázis cím kitöltése a megfelelő értékkel, a folyamatleíró blokk elkészítése. A betöltéssel válik egy program folyamattá.

Az így elkészített, illetve betöltött program hátránya, hogy pazarló módon minden részlete egyszerre kerül a memóriába, akkor is, ha ez nem szükséges. További hátrány, hogy azok a modulok, amelyeket több program is használ, mint például a rendszerkönyvtár elemei annyiszor kerülnek betöltésre, ahány program használni kívánja őket. Az eddigi statikus szemlélettel szemben a problémát a dinamikusan szerkeszthető könyvtárak (Dynamic Link Library - DLL) segítségével lehet megoldani. A dinamikus könyvtárak csak akkor kerülnek a memóriába, ha azokra hivatkozás történik, bekerülésük után viszont több program is használhatja egyszerre őket.

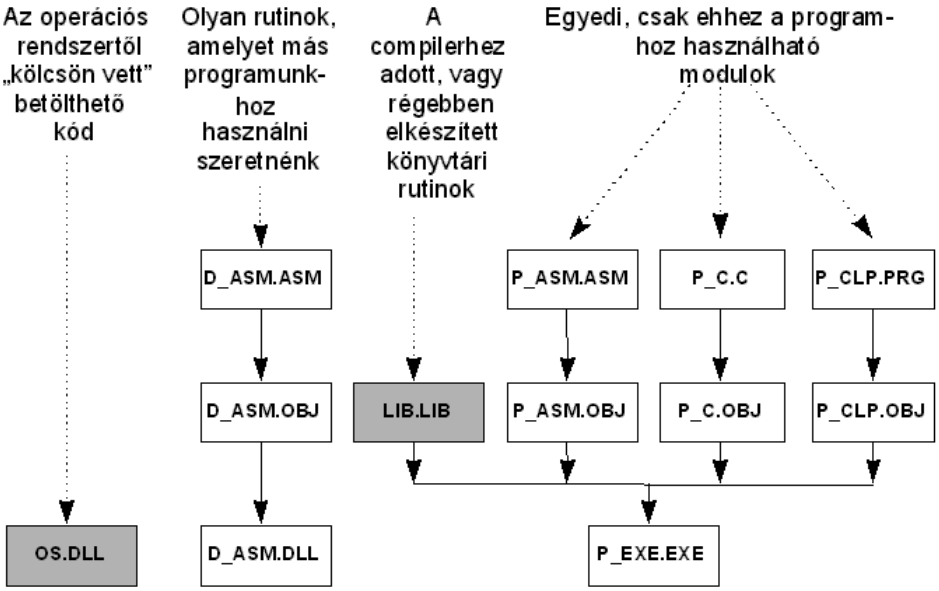
5.3.4.1 A program összeállítás teljes folyamata

A forráskód megírásától a memóriába való betöltődésig tartó folyamat tehát a következő.



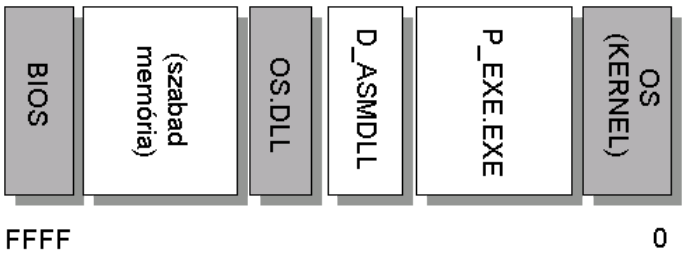
A programkészítés folyamata

Nézzünk végül egy példát! A következő lapon látható feladat lehetne például egy beléptető rendszer programja, amelyhez felhasználjuk az operációs rendszer kernelje és a BIOS által nyújtott szolgáltatásokat (rendszerhívások), valamint az OS.DLL szubrutinjait, függvényeit. Az elkészítendő program tartalmaz adatbázis funkciókat (erre megfelelő a Clipper), hardverkezelő rutinokat (ezek Assembly-ben készülnek) a főprogram pedig C-ben készül. Az Assembly rutinok egy részét más, egyidejűleg futó programjainkhoz is szeretnénk felhasználni, ezért ezeket dinamikusan betölthető formában hozzuk létre.



Egy komplex program összeállítása

A betöltődés után kialakult memória kép a következő lesz (természetesen a DLL-ek csak szükség esetén töltődnek be).



Egy program részeinek elhelyezkedése a memóriában

5.4 Röviden a hálózatokról

A számítógép-hálózatok alapvetően megváltoztatták a világot. Eddig minden törekvés az ember és a számítógép kapcsolatának finomítására irányult, a számítógép-hálózatok kialakulásával egyre fontosabb szerepet kap a gép-gép kapcsolat. Nem egyszerűen csak számítógépek halmazáról van szó, hanem gépek rendszeréről, melyben – az emberi szervezethez hasonlóan – egyes részek bizonyos célokra specializálódnak. Általános trend, hogy az intelligencia súlypontja az egyedi számítógépről áttolódik a hálózatra.

5.4.1 Mire jó a számítógép-hálózat?

Képzeljük el a következő helyzetet! Legyen két független számítógépünk, amelyeken ugyanaz az adatbázis, illetve a hozzá illeszkedő adatbázis kezelő található (két független példányban), és mindkét géphez tartozik nyomtató is. Ilyen lehet például egy bolt két pénztárgépe. Az adatbázis ebben az esetben az árukészlet, illetve az eladások adatait tartalmazza.

Nézzük meg, hogy a fenti rendszernek – a számítógép használatából adódó természetes előnyök mellett – milyen hátrányai vannak!

1. **Ha az egyik nyomtató elromlik, az érintett konfiguráció használhatatlan, hiszen nem tud számlát előállítani.**

2. **Mindkét számítógépnek nagy háttértárra van szüksége, hiszen az egész adatbázist külön-külön is tárolniuk kell.**
3. **Záráskor, vagy ha lehet gyakrabban, szinkronizálni kell az adatokat, azaz biztosítani kell, hogy mindkét gép ugyanazon adatbázis alapján dolgozzon.**
4. **Ha az egyik gépen megsérül az adatbázis, ki kell deríteni, hogy melyik változat az „igazi" – ez általában nem is olyan egyszerű –, és át kell azt másolni a másikra.**
5. **Az egyik nyomtató – feltételezve, hogy nincs mindig csúcsforgalom – általában csak várakozik, nem végez hasznos munkát.**

Foglaljuk össze a problémákat kissé általánosabban!

1. **Az erőforrások (nyomtató, háttértár, adatbázis) kihasználtsága rossz.**
2. **Egy-egy erőforrás kiesése esetén – szerencsétlen esetben – a rendszer működőképes marad ugyan, de meglehetősen leromlott (50%-os) hatékonysággal.**
3. **Erőfeszítéseket kell tenni az adatok megbízhatóságának (szinkronitásának, integritásának) biztosítására, ez munkaerőt köt le.**

Vagy még általánosabban, azért, hogy rendszerünket hatékonyabbá, és egyben megbízhatóbbá tegyük:

- **csökkenteni kell a redundanciát (az adatbázis elegendő csak egyetlen példányban, az esetek nagy többségében elég egyetlen nyomtató is) annak érdekében, hogy rendszerünk hatékonyabb legyen;**
- **növelni kell a redundanciát (tartalék nyomtató, az adatbázis biztonsági másolata stb.) annak érdekében, hogy rendszerünk megbízhatóbb legyen.**

(Redundanciának nevezzük egy rendszerben (vagy közleményben) azt a többletet, amely új szolgáltatást (vagy információt) nem biztosít.)

De hiszen a fenti példában van több nyomtatónk, van több adatbázis másolatunk, hogy lehetséges, hogy a rendszer mégsem megbízható?

Egy valami még hiányzik: a kommunikáció, a kapcsolat, mely a független rendszerelemek halmazából működő rendszert alkot.

5.4.2 A számítógépek közötti kommunikáció

A kapcsolathoz tehát kell (legalább) egy közvetlen vagy közvetett fizikai összeköttetés és (legalább) egy közös nyelv, amit mindkét számítógép ért. Ha pedig csak egy kicsit tovább bővítjük a kört, és további számítógépeket kapcsolunk a rendszerhez, szükségünk lesz egyedi címekre is, melyek egyértelműen azonosítják az üzenetek címzettjét vagy címzettjeit, illetve a feladót.

A gép–gép kommunikáció feltételei

5.4.2.1 Az összeköttetés

A kapcsolat szempontjából (mostani szempontjaink szerint) lényegtelen, hogy az milyen fizikai csatornán valósul meg. Használhatunk rézvezetéken átvitt elektromos impulzus sorozatot (koaxiális kábel, csavart érpár), valamilyen módon modulált rádióhullámokat (kábel TV, V-SAT), vagy szaggatott fénysugarat (üvegszál, lézer), lényeg az, hogy a megvalósított csatornán biteket kell átvinni szépen egymás után sorban. Ha mindegyik számítógépet mindegyikkel összekötnénk egy-egy csatornával nagyon rossz kihasználtságot érhetnénk el (és lépni sem lehetne a sok vezetéktől), hiszen egy gép általában csak egy másikkal „beszélget" és ez a beszélgetés is nagyon szakaszos, egyetlen aktív vezetéken az idő legnagyobb részében semmi sem történne.

Megoldás lehet, ha mindegyik számítógép ugyanazt a csatornát használná, és a kommunikációhoz használt üzenetcsomag önmaga tartalmazná a címzettre (és persze a feladóra) vonatkozó információt.

Gyakran nincs is más megoldás. Az 1970-es években A Xerox, a DEC és az Intel kapott olyan megbízást, hogy a Hawaii-szigetek több száz kikötője között megbízható adatkapcsolatot létesítsen. Vezetékek szóba sem jöhettek, a fénysugár alkalmazása a nagyobb távolságok miatt lehetetlen volt, egyedül az „éter"-ben terjedő rádióhullámok maradtak. A létrehozott rendszert ezért elnevezték Ethernetnek.

Az adatok tehát egy csomag részei, melynek fejléce tartalmazza a címzett és a feladó címét. Ezt a technikát nevezzük csomagkapcsolásnak (szemben a hagyományos telefon esetén alkalmazott vonalkapcsolással). Az üzenetet tehát a hálózat minden résztvevője veszi, de csak az értelmezi, aki a saját címét a megfelelő mezőben felismeri. Íme egy tipikus csomag (keret, frame):

Csomag típusa	Adó címe	Vevő címe	Hasznos adatok	Ellenőrző összeg

Tipikus csomag felépítése

Meg kell oldani azonban azt a feladatot is, hogy ezen a közös csatornán egyszerre mindig csak egy kapcsolat legyen aktív, azaz ne „beszéljenek" egyszerre az állomások, mert az átláthatatlan káoszt eredményezne. Az erre szolgáló átvitel-vezérlésnek lényegében két formája alakult ki.

- **CSMA/CD-nek (Carrier Sense Multiple Access / Collision Detection) nevezzük azt a rendszert, melyben minden állomás (multiple access) folyamatosan figyeli az átvivő közeget (carrier sense) és ellenőrzi annak „tisztaságát", azaz azt, hogy nem ad-e már éppen valaki. Egy adó csak akkor kezdeményezhet üzenettovábbítást, ha a csatornát éppen senki sem használja. Előfordulhat persze, hogy két, vagy több adó gondolkodik egyformán, de ilyenkor sincs nagy baj, hiszen minthogy figyelik a csatornát, azonnal észreveszik az ütközést (collision detection), és megismételhetik az adást. De ha egyformán működnek, akkor egyformán ismétlik, és akkor a gond nem csökken. A CSMA/CD technika úgy küszöböli ezt ki, hogy az ütközést érzékelt adók nem azonnal, hanem – a további ütközést**

megelőzendő – véletlenszerű idő múlva kezdik újra adásukat. Az ilyen rendszer tipikus képviselője a helyi hálózatok között napjainkban leginkább elterjedt Ethernet.

- **Token Passing, azaz a vezérjel átadás alapján működik például az Arcnet, Token Ring és az FDDI (Fiber Distributed Data Interface, optikai szálas hálózati protokoll; működhet csavart érpáron is) hálózat. Az állomások ebben az esetben – fizikai elhelyezkedésüktől függetlenül – egy logikai gyűrű mentén helyezkednek el. Az adók egy speciális üzenetet, a token-t (vezérjelet) adják körbe-körbe, és az üzenet csak ehhez fűzhető. Mivel a token egyszerre csak egy állomás birtokában lehet, automatikusan kizárható az ütközés lehetősége.**

5.4.2.2 Közös nyelv (protokoll)

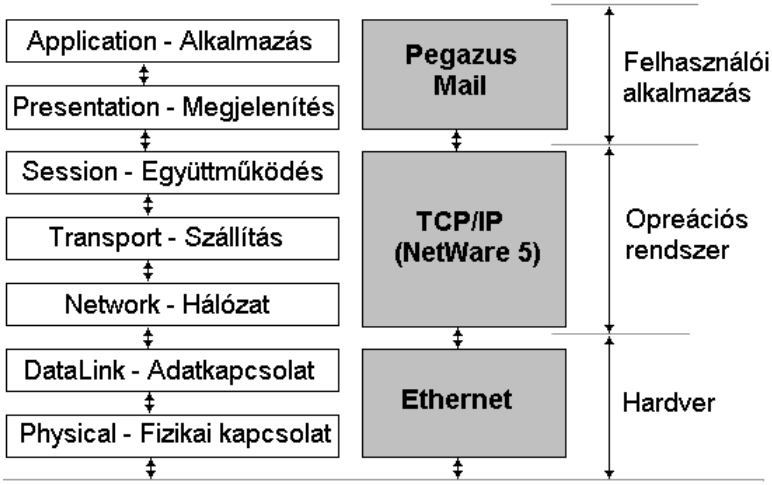
A hálózatok szerencsére olyan intenzíven kezdtek terjedni, hogy a hardver és szoftver gyártók elemi érdeküknek érezték, hogy olyan egységes rendszereken munkálkodjanak, amelyek más berendezésekkel is jó együtt tudnak működni. Az erőfeszítés eredményeként létrejött egy nemzetközi ajánlás, amit ISO-OSI (International Standard Organisation – Open System Interconnection) néven ismerünk, amely vezérfonalul szolgált minden további fejlesztéshez. A szabvány egy réteges felépítésű rendszert definiál, minden réteghez meghatározott feladatot rendel.

Az egyes rétegek szabványos felületen kommunikálnak a szomszédos rétegekkel, valamint a másik gép ugyanazon rétegével. Az azonos rétegek kapcsolati szabályainak összességét nevezzük protokollnak, illetve ezek összefüggő rendszerét protokoll családnak (gyakran előforduló protokoll családok például az IPX/SPX, a TCP/IP, az SNA, és a DECNET).

A gyakorlatban elterjedt rendszerek persze nem követik „szó szerint" a szabvány ajánlásait, egy-egy protokoll több réteget is megvalósíthat, egy rétegben több protokoll is helyet kaphat, illetve az is előfordul, hogy egy réteg csak félig-meddig valósul meg egy protokoll által.

Nézzünk egy példát!

A következő ábrán három számítógép réteges hálózati felépítése látható. A bal oldalon az eredeti, és a magyar elnevezésekkel, sötétebb árnyalattal jelölve pedig egy Novell hálózaton levelezésre használt számítógép példáján. A képen az is látható, hogy egy hálózati alkalmazás a hardver, operációs rendszer és felhasználói szoftver szintet egyaránt átöleli.



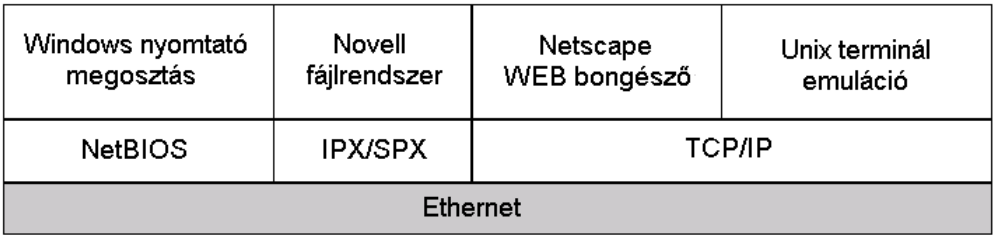
Az ISO-OSI ajánlás, ill. egy megvalósulása

Vizsgáljuk meg egy levél elküldésének példáján a modell működését!

1. **Megírom a levelet a levelező program beépített szövegszerkesztője segítségével (Felhasználói alkalmazás - Alkalmazás réteg - Pegasus Mail)**
2. **Ellátom digitális aláírással, titkosítom, hogy az illetéktelenek ne nézegethessék (Felhasználói alkalmazás - Megjelenítés réteg - Pegasus Mail)**
3. **Kapcsolatba lépek azzal a számítógéppel, amelyik majd továbbítja a levelemet (Operációs rendszer - Együttműködési réteg - Novell TCP/IP)**
4. **Megbízható, nyugtázott átvitelt kérek (Operációs rendszer - Szállítási réteg - Novell TCP/IP)**
5. **Megkeresem és megadom a célállomás címét, valamint a hozzá vezető útvonalat (Operációs rendszer - Hálózati réteg - Novell TCP/IP)**
6. **Megfelelő méretű darabokra tördelem a levelet, és előkészítem a legközelebbi megálló (azaz a végső útvonal felé vezető útvonal első állomása) felé történő továbbításra (Hardver - Adatkapcsolati réteg - Ethernet kártya)**
7. **Feszültség-impulzusokká alakítom a levelemből készült csomagok bitjeit és továbbítom őket a fizikai csatornán (Hardver – Fizikai réteg – Ethernet kártya)**

Az ellenállomáson minden ugyanígy lépésről lépesre, de természetesen minden visszafelé, játszódik le.

Éppúgy, mint az emberi nyelvek, vagy a programozási nyelvek esetén a számítógépek is beszélhetnek több nyelvet, azaz ismerhetnek több protokollt. Az alábbi példa egy olyan számítógép operációs rendszerének hálózati szerkezetét mutatja, mely egyszerre képes igénybe venni az Internet, egy Novell szerver és a Windows-os hálózat szolgáltatásait.



Munkaállomás hálózati rétegei

5.4.2.3 Címek

Ahhoz, hogy a kapcsolat teljessé váljon, szükség van egyedi címekre is. Mivel minden réteg egy másik ugyanolyan réteggel kommunikál, ilyen cím több is lehet. A leggyakoribb címtípusokat az előző fejezet konkrét példáján mutatjuk be a fizikai rétegtől az alkalmazási réteg felé haladva.

5.4.2.3.1 Fizikai címek

A helyi hálózatokon legjobban az ethernet hálózatok használatosak. Ebben az esetben a fizikai cím egy gyárilag beállított, a felhasználó által nem változtatható hat bájtos szám, amit – mint általában mindig – hexadecimális alakban szoktunk megadni. Például:

0000C0 – AB7416

A hexadecimális számjegyek csoportosítása nem kizárólag a könnyebb olvashatóságot szolgálja. Az első hat számjegy a gyártó kódja (ebben a példában az SMC), egyediségére nemzetközi szervezetek ügyelnek, a második hat számjegy egyedi azonosító.

Más, kevésbé használt hálózattípusoknál előfordul, hogy a kártyán lévő kapcsolóval állítható a fizikai cím, de ennek alkalmazása nagyobb hálózatoknál feltétlenül kerülendő. A fizikai cím másik szokásos elnevezése a MAC (Media Acces Control) cím.

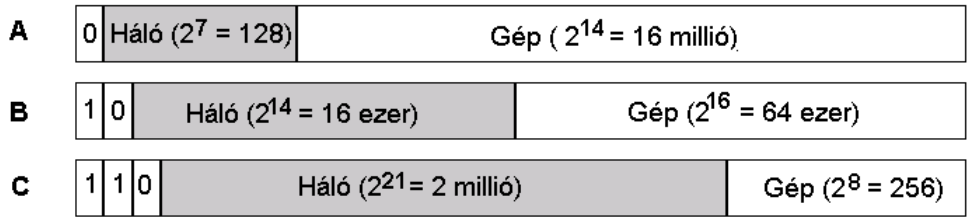
5.4.2.3.2 IP címek

A TCP/IP protokoll családban használatos címek 4 bájtos bináris számok.

Ez óriási nagy szám, hiszen 32 biten több mint 4 milliárd számítógép címezhető meg, de egyrészt a címtartomány nem teljesen kihasználható, másrészt ha elképzeljük, hogy minden háztartási gép belátható időn belül címet kaphat máris nem tűnik olyan soknak. A közeljövőben elterjedő, IPv6 szabvány 128 bites címmezője hoz vár-hatóan hosszú távú megoldást

A címet két részre oszthatjuk. Az első rész magának a hálózatnak a címe, a második pedig az egyedi gépé. Ha a gép címe helyén csupa egyest találunk (broadcast), azt jelenti, hogy az üzenet a hálózat minden állomására vonatkozik. A csak nullából álló cím tesztelésre szolgál, az üzenet az ilyen címet használó gépről nem jut ki.

Háromféle címosztály létezik:



„Klasszikus" IP címosztályok

Természetesen ezeket a címeket sem lenne kellemes kettes számrendszerben ábrázolni. Az IP címek körében a bájtokat egy-egy ponttal elválasztott tízes számrendszerbeli számmal adjuk meg. Például:

`1100001 11100000 10001101 11110000 → 193.224.141.240`

Ez az ábrázolás már barátságosabb, de még mindig elég nehezen megjegyezhető. A probléma megoldása érdekében alakították ki az úgynevezett tartománynév rendszert (DNS - Domain Name System), mely legegyszerűbb megközelítésben olyan számítógépek (Name Server) rendszere, melyek egy név alapján táblázataikban megkeresik a névhez tartozó számot, és azt visszajuttatják a kérőnek. Az előző példában (vegyük észre, hogy az első három bináris számjegy tanúsága szerint ez egy C osztályú cím) a `193.224.141` azonosítja a hálózatot, a 240 a számítógépet. Rendeljük hozzá a `gdf-ri.hu` a karaktersorozatot (domain name) a `193.224.141` címhez (és természetesen ezt jegyeztessük be a megfelelő szervezettel), és nevezzük el erebus-nak a gépünket (hostname), és ezt jegyezzük be saját rendszerünkben. Ettől kezdve, ha a világon bárhol az `erebus.gdf-ri.hu-ra` hivatkoznak, a legközelebbi Name szerver kitalálja, hogy ez a mi hálózatunk, a saját Name szerverünk pedig az üzenetet a megfelelő gépre irányítja (erebus). Tehát:

`193.224.141.240 → erebus.gdf-ri.hu`

5.4.2.3.3 Levelezési címek

Ha a számítógépet meg tudjuk címezni, már nem nehéz azt a felhasználót sem elérni, akinek gépen található a postafiókja. A hagyományos névkonvenció a következő (például):

`info@erebus.gdf-ri.hu`**, illetve** `info@193.224.141.240`

5.4.3 Hálózattípusok

Ma az egyes intézmények hálózatatinak többsége kapcsolatban áll egymással, egységes címrendszert és protokollt használnak, mégis érdemes megkülönböztetni a viszonylag kis kiterjedésű lokális hálózatokat (LAN – Local Area Network) a nagy kiterjedésű hálózatoktól (WAN – Wide Area Network), melynek szinte egyeduralkodó képviselője az Internet. (Régebben használatos volt a városi hálózat (MAN – Metropolitan Area Network) kategória is, ami a LAN és a WAN között helyezkedik el, azonban ez az egységesedés miatt egy időre háttérbe szorul és csak napjainkban kezd visszalopakodni a szaknyelvbe, igaz, jócskán megváltozott értelemmel.)

A LAN/WAN megkülönböztetés oka elsősorban a sebesség. A világhálózat által biztosított óriási erőforrás tömeg egy kis részét (azt, amit a leggyakrabban szeretnénk használni) célszerű fizikailag is közelebb hozni, hiszen könnyebb rövid távon nagy sebességű átviteli csatornát létesíteni, változtatni a hardver , illetve szoftver konfiguráción, a rendszergazdához fordulni segítségért, vagy a szomszéd szobába átmenni a kinyomtatott levélért. A LAN fizikai összeköttetések jóval nagyobb átviteli sebességet tesznek lehetővé (10 Mbps – 1000Mbps) speciális kábeleiken, mint az egyéb célra készült kommunikációs hálózatokat (telefon, kábel TV) is használó WAN kapcsolatok (10kbps – 2 Mbps).

Felvetődhet a kérdés, ha a sebesség ennyire fontos, a kapcsolat másik két összetevője, a címzés és az alkalmazott protokoll miért egyezik meg a két hálózattípus esetén? Az ok a kényelem szeretete! Az elsősorban Internet céljaira kidolgozott technológiák ugyan lassabbak, mint a lassan kiszoruló LAN technikák, viszont a menedzselés szempontjából óriási előnyt jelent az egységesítés. Az egységes technológia az elnevezésekben is megmutatkozik, az Intranet tulajdonképpen nem más, mint LAN fizikai összeköttetést megvalósító, annak jó menedzselhetőségét öröklő, de az Internettől ellesett protokollokat használó hálózat.

5.4.4 Munkamegosztás a hálózaton

Az egymással kapcsolatban lévő, kommunikáló számítógépektől tehát azt várjuk, hogy tegyék lehetővé erőforrásaik megosztását, növeljék a rendszer megbízhatóságát, illetve biztosítsák a gyors kommunikáció lehetőségétfelhasználóik között. E feladatok meg-valósítása során kialakult egy specializáció a számítógépek között, létrejöttek olyan gépek, amelyek felépítése, illetve a rajtuk futó szoftver egy bizonyos részfeladat ellátására tette őket különösen alkalmassá. Ezek a gépek részben vagy teljes egészében arra szakosodtak, hogy kiszolgálják a hálózat többi résztvevőjének igényeit. Ma a hálózatba kapcsolt gépek többsége a szerver (kiszolgáló) – kliens (ügyfél) modell szerint működik.

Gyakori, hogy a szervert azonosítják a nagy hardver erőforrásokkal rendelkező gépekkel, pedig nem ez a lényeg, hanem a feladat. Előfordulhat, hogy szappanos doboz méretű doboz teszi lehetővé egy jelentős méretű hálózatnak, hogy közösen használhassanak egy nyomtatót.

Melyek a legfontosabb szolgáltatások? (Az alábbi vázlatos felsorolásban a szolgáltatás nevét kerek zárójelek között követi az a hálózattípus, melyre jellemző, a rövid leírást pedig szögletes zárójelben a szolgáltatáshoz kapcsolódó jellegzetes protokollok sora zárja.)

- **Elektronikus levelezés (LAN/WAN) Elektronikus üzenet továbbítása, esetleg csatolt állományokkal együtt [SMTP, POP3, TCP/IP]**
- **Fájlszerver (LAN) Célja állományok gyors elérése abból a célból, hogy azt a kliens memóriájába betölthessük (tehát nem elsősorban a másolás), program esetén futtathassuk. [IPX/SPX, TCP/IP]**
- **Nyomtatószerver (LAN) Nyomtatók közös használatát teszi lehetővé. A nyomtatók megosztásának célja általában a takarékosság, de nem mindig! Ha egy gépről háromnál több nyomtatótípust szeretnénk használni, nem is igen van más lehetőség. [IPX/SPX, TCP/IP, NetBIOS]**
- **FTP szerver (WAN) Fájlok másolása (elsősorban) távoli szerverek és kliensek között. [FTP, TCP/IP]**
- **WEB szerver (LAN/WAN) Hipertext (azaz további szövegekre, multimédia elemekre utaló hivatkozást tartalmazó) szövegek továbbítása. Napjainkban ez a legnépszerűbb Internet alkalmazás. [HTTP, TCP/IP]**
- **Terminál szerver (LAN/WAN) Egy számítógép minden erőforrása igénybe vehető így, beleértve a processzort és a memóriát is. A programok a terminál szolgáltatást biztosító számítógépen futnak, a kliens felé mindössze a képernyőre szánt adatok jutnak el, illetve a kliens által megjelenített látszólagos terminál billentyűzete vezérli a távoli gépet. [Telnet, RLOGIN, SSH, TCP/IP]**
- **Adatbázis szerver (LAN/WAN) Nagy adatbázisokban történő keresések leghatékonyabban úgy valósíthatók meg, ha a hálózaton csak a kérdés (pl. SQL parancs), illetve a válasz (kigyűjtött adatok) kerülnek átvitelre, és nem az egész adatbázis, illetve kezelő programja. Ezt a feladatot látja el az adatbázis szerver, manapság igen gyakran Web felületen keresztül.**
- **Proxy szerver (WAN) Tulajdonképpen a lassan elérhető világháló, és a helyi hálózatok közötti gyorsítótár. Segítségével a gyakran letöltött fájlok, web oldalak a lokális hálózat sebességén érhetők el. [HTTP, FTP, TCP/IP]**
- **Tűzfal szerver (WAN) A lokális hálózat és az Internet elválasztására létesített szolgáltatás (a proxy-hoz hasonlóan), azonban célja a biztonság. Szabályozni lehet a belülről kifelé szóló kéréseket, illetve a kívülről befelé folyó forgalmat. [HTTP, FTP, TCP/IP]**

5.5 Ellenőrző kérdések

1. Milyen program típusokkal találkozhatunk leggyakrabban egy személyi számítógépen?
2. Mi az operációs rendszer alapfeladata?
3. Mit különbség és az összefüggés a shell és a kernel között?
4. Mi a feladata a kernelnek a hardver kezelés tekintetében?
5. **Hogyan, milyen módszerekkel egységesíthető a hardver-kernel interfész?**
6. Mi a Plug and Play technika lényege?
7. Milyen kapcsolat van a szoftver eredetű megszakítások, és a rendszermag között?
8. Ismertesse a rendszermag fő funkcióit!
9. Mi az állomány (fájl)?
10. Milyen fájl elnevezési lehetőségeket ismer?
11. Sorolja fel az állományok fő jellemzőit!
12. Mi a katalógus (directory)?
13. Milyen katalógus rendszert használnak a mai operációs rendszerek?
14. Mi a kötet (volume)?
15. Milyen kötet elnevezéseket használnak a DOS eredetű operációs rendszerek?
16. Írjon néhány példát az abszolút fájl hivatkozásra?
17. Mi a relatív fájl elérés lényege?
18. Ismertesse az aktuális katalógus, aktuális kötet fogalmát!
19. Mi a keresési útvonal? Mire használható?
20. Mit nevezünk (a fájlrendszerekkel kapcsolatosan) clusternek?
21. Hogyan töltődik be az operációs rendszer a lemezről?
22. Mi a Master Boot Record, és mi köze a Partíciós táblához?

23. **Mutassa be egy DOS kötet felépítését!**
24. Milyen előnyei és hátrányai vannak a láncolt fájl (FAT) elhelyezés alkalmazásának? Hogyan mérsékelhetők a hátrányok?
25. Mi a parancsértelmező feladata?
26. Milyen elemekből épül fel egy DOS parancssor? Mi az egyes részek szerepe?
27. Ismertesse a DOS legfontosabb, fájlokra vonatkozó parancsait!
28. Mik az ablakozó rendszerek (pl. Windows) előnyei?
29. Miért alakítanak ki az operációs rendszerek mellett alrendszereket is?
30. Melyek a programkészítés alapvető lépései?
31. Mit nevezünk API-nak?
32. Mi a fordítóprogram (compiler) feladata?
33. Milyen feladatot lát el a kapcsolat szerkesztő (linker)?
34. Mi jellemzi az integrált fejlesztő rendszereket?
35. Mi a feladata a betöltőnek (loader)?
36. Mi az összefüggés egy rendszer redundanciája és hatékonysága között?
37. Hogyan befolyásolja a redundancia a megbízhatóságot?
38. Mi szükséges (minimálisan) ahhoz, hogy két számítógép egymással kommunikálni tudjon?
39. Mit nevezünk csomagkapcsolásnak?
40. Milyen – számítógép-hálózatokban alkalmazott – átvitelvezérlési technikákat ismer?
41. Ismertesse a CSMA/CD lényegét!
42. Mit nevezünk számítógép-hálózati protokollnak?
43. Mire vonatkozik az OSI modell, és milyen részekből áll?
44. Rajzoljon le egy konkrét hálózati rétegszerkezet megvalósítást!
45. Ismertesse az IP címek képzésére vonatkozó szabályokat!
46. Miért van szükség tartomány nevekre? Mi a DNS (Domain Name System)?
47. Mi a fő különbség a LAN és a WAN között?
48. Melyek a szerver–kliens modell alkalmazásának előnyei?
49. Mi az Intranet?
50. **Ismertesse a számítógép-hálózatok alapfeladatait!**
51. Melyek a legfontosabb számítógép-hálózati szolgáltatások?
52. Mi az összefüggés és a különbség a Tűzfal és a Proxy szerver között?
53. Mi az összefüggés és a különbség a Fájl és az FTP szerver között?
54. Mikor használhatók ki igazán a nyomtató szerver alkalmazásának előnyei?

Melléklet A: Fogalomtár

Adat (DATA)

Tények, fogalmak, eligazítások olyan formai megjelenése (képe), amely alkalmas az emberi vagy az automatikus eszközök által történő kommunikációra, értelmezésre vagy feldolgozásra.

fogalomtár

Egy szakterület vagy tantárgy szakfogalmait és ezek definicioit tartalmazó szótár az ILIAS-ban.

Hajlékonylemezes tár= Flopidiszk

(FLIPPY-FLOPPY, FLOPPY, FLOPPY-DISK, FLOPPY-DISK SYSTEM, DISKETTE)
Mágneses jelrögzítéssel működő tárolóeszköz, amely az adatokat hajlékony anyagból készült lemezen tárolja.

hőmérséklet (műsz., fiz.)

a termikus kölcsönhatás jellemző →intenzív mennyisége. Az abszolút ~ jele: T, mértékegysége: 1 K. A Celsius-skálán mért hőmérséklet jele: ϑ, mértékegysége: 1 °C és 0-pontja: T=273,15 K-nél van. (Megj.: a víz ún. hármaspontjának abszolút ~e: 273,16 K.)

Kommunikációs protokoll

Két eszköz közötti adat és vezérlő információcsere szabályainak összessége.