Számítógépek architektúrája

Matijevics István

2014

A tananyag a TÁMOP-4.1.2.A/1-11/1-2011-0104 "A felsőfokú informatikai oktatás minőségének fejlesztése, modernizációja" c. projekt keretében a Pannon Egyetem és a Szegedi Tudományegyetem együttműködésében készült.

ELŐSZÓ

A jegyzet mérnökinformatikus hallgatók számára készült, tartalma követi a Szegedi Tudományegytem, Természettudományi és Informatikai Kar Informatikai Tanszékcsoporthoz tartozó B.Sc. mérnökinformatikus-képzés „Digitális Architektúrák" kurzus tantervét és előadásait.

A tudományterület részben klasszikus, részben gyorsan változó tudáson alapul, így ezt a jegyzetben is figyelembe kellett venni. Az anyag célja megalapozni a hallgatók számítógépekkel kapcsolatos tudását, előkészíteni őket a későbbi számítógépes szaktantárgyak elsajátítására.

A fejezetek a főbb részeket külön-külön mutatják be, de ezek a fejezetek egymásra épülve íródtak.

Az anyag megértését a számos ábra mellett néhány interaktív szimuláció és néhány szemléletes animáció is elősegíti.

Köszönetet mondok Bodnár Péter Ph.D. hallgatónak az ábrák kidolgozásában nyújtott segítségért.

Bízom benne, hogy a hallgatók egy olyan jegyzetet és segédanyagot kapnak kézbe, amivel könnyebben tudják elsajátítatni a számítógépekkel kapcsolatos tudást, ismereteket.

A Szerző

1. A SZÁMÍTÓGÉPEK TÖRTÉNETE, FEJLŐDÉSE

1.1. BEVEZETÉS

Napjainkban egyértelműen meghatározó szerepe van az informatikának. Gyakorlatilag az összes tudományágban, a gazdaságban, egészségügyben, a közlekedésben (repülő, vonat, hajó stb.) és az élet minden területén ma már lehetetlen nélküle meglenni. Fejlődése töretlen, egyre összetettebb rendszereket hoznak létre. Az informatika ilyen mértékű elterjedése csak úgy lehetséges, ha megfelelő műszaki-technológiai háttérre támaszkodik, ez pedig a számítógép-technika, ebben is döntő szerepe van a digitális, bináris elektronikának, illetve számítógép-technikának. A mai értelemben vett számítógép-technika kezdete nagyjából az 1950-es évek elejére tehető, de ez nem jelenti azt, hogy ezen időpont előtt nem voltak olyan eszközeink, amelyek számítógép-technikai feladatokatt láttak volna el.

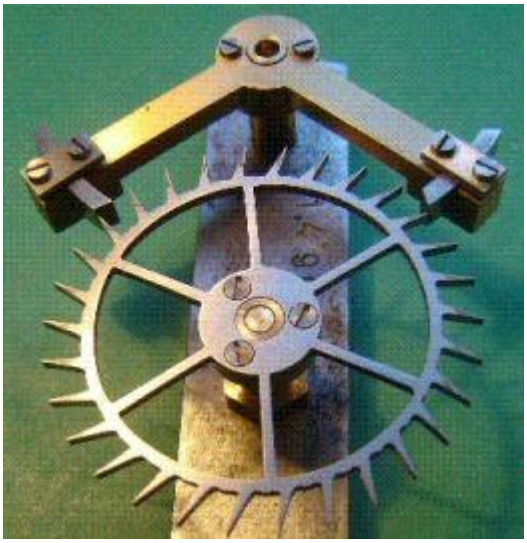
A fenti szövegben számítógép-technikáról és nem számítástechnikáról beszélünk, a kettő között lényeges különbség van, igaz átfedés is van közöttük. Sajnos a magyar nyelv nem tükrözi teljesen a két

fogalom közötti különbséget. A számítástechnika és eszközei számítások (pl. összeadás, szorzás stb.) elvégzését teszi lehetővé, ellentétben a számítógép-technika már olyan eszközök megépítését biztosítja, amelyek algoritmizálható feladatok megoldására alkalmasak, vagyis programozhatók. Ugyanakkor ne felejtsük el, hogy a számítógép számításokat is végez (fordítva nem igaz).

Ennek a mai, modern számítógép-technika fejlődésnek egyik alappillére az 1947-ben megépített tranzisztor volt. Az 1.1. ábrán a Bell Laboratóriumban megépített első működő tranzisztor látható, illetve egy mai tranzisztor. Ebben az eltelt nagyjából 60 évben a számítógép-technika is generációváltásokon ment keresztül.

1.1 ábra. Az első tranzisztor (Bell Laboratory) és egy mai tranzisztor.

Amióta az ember alkotómunkát végez, igyekszik a rá háruló feladatokat mással elvégeztetni, vagy valamilyen más segédeszközzel elvégezni. Az alkotómunka egy része igényli a számítások végrehajtását, valamint az adatok tárolását. A gazdaság vezetése, biológiai modellezés, vagy az utazások (nagy felfedezések) pontos és gyors számításokat igényeltek és igényelnek. Igen sok kísérlet történt ezen feladatok megoldására, nagyon sok találmány született (pl. az idő mérésére szolgáló mechanikus óra, a hajózásban használt helymeghatározó eszközök stb.), mely találmányok jó részét ma is használjuk. A 1.2. ábrán egy órát, illetve a 1.3. ábrán egy szextánst látunk.



(Forrás: http://syzygyastro.hubpages.com/hub/The-Clock-Through-History).

1.2. ábra. Óra időzítő szerkezete.

(Forrás: http://www.clipperlight.com/SEXTANTARTICLE/sextant1.jpg)

1.3. ábra. Szextáns.

A számításokat végző eszközöktől mindig azt várták, hogy nagy mennyiségű adathalmazon gyorsan (sokkal gyorsabban, mint az ember arra képes) végezzenek pontos számításokat, tévedések, hibázások nélkül, nem utolsósorban automatikusan, valamilyen meghatározott program szerint. Tehát a „számító

gépek” (számítástechnika) fejlődése, a velük szemben támasztott igények vitték a fejlődést a számítástechnika irányából a számítógép-technika felé.

Jelenlegi ismereteink szerint az írás kialakulásával egy időben (kb. i. e. 4000-3500 táján) a kialakuló kereskedelem a mérést is igényelte, meg kellett határozni a mértékegységeket, ugyanakkor az értékekkel számolni kellett, de azokat le is kellett jegyezni, tartósan kellett tárolni az adatokat.

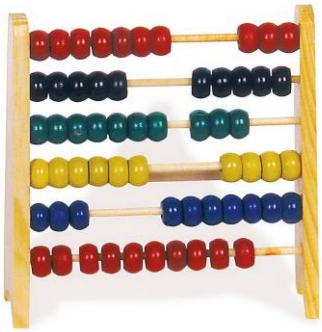
Természetesen az informatika első „számítástechnikai eszköze” az ember ujjai voltak, erre az is utal, hogy latinul az ujj elnevezése digitus. Mivel az angol nyelv a latin nyelv szókincséből igen sokat vett át, így átkerült a digit (számjegy) szó is.

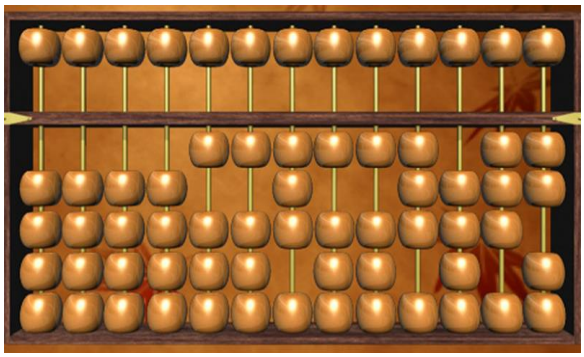
Persze, ahogy a mennyiség nőtt, a 10 (20) ujj már nem volt elég, ekkor köveket, kagylókat és egyéb hasonló tárgyat használtak, a mennyiségek további növelése pedig az átváltásos rendszer bevezetését igényelte (nagyobb értékek számossági megjelenítése).

A 10 ujj egyértelműen meghatározta a számrendszer alapját, de nagyon sok ókori és későbbi kultúra is más számrendszert is alkalmazott, ezek közül még ma is sok használatos.

Így ismert a 12-es számrendszer (angol, német és francia nyelv 11 és 12 elnevezése, de magyarul is használjuk a „tucat” elnevezést). A maják az 5, 10 és 20 számokat is ismerték, külön kifejezésük volt rájuk. Egyiptomból, Mezopotámiából ismert a 60-as számrendszer, vagy a kör 360 fokos osztása.

A számrendszer meghatározása, a számolási alapműveletek ismerete (összeadás, kivonás, szorzás és osztás) önmagában nem elég a gyors, hatásos számoláshoz, kellett valamilyen számolási eszközt (számítástechnika) is kitalálni. Igen sok kísérlet volt ennek kiépítésére, például táblázatos adatfeldolgozás, vagy először a kínaiak által megalkotott abakusz. Az abakusz egyik műszaki megoldása a sínekben elhelyezett kövecskék, ezek mozgatása, mivel a kövecskéket latinul calculus-nak nevezzük, ez adta a kalkulátor elnevezést a hasonló eszközök megnevezésére. Az abakusz japán változata a szorobán, amelyik hasonló elven működik.

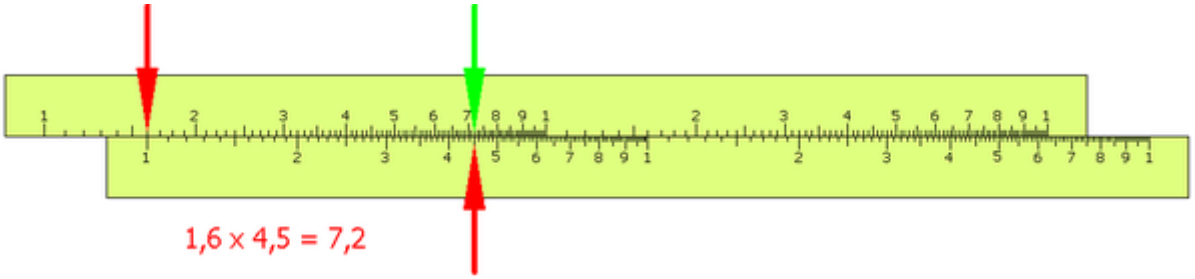




1.4. ábra. Abakusz és szorobán.

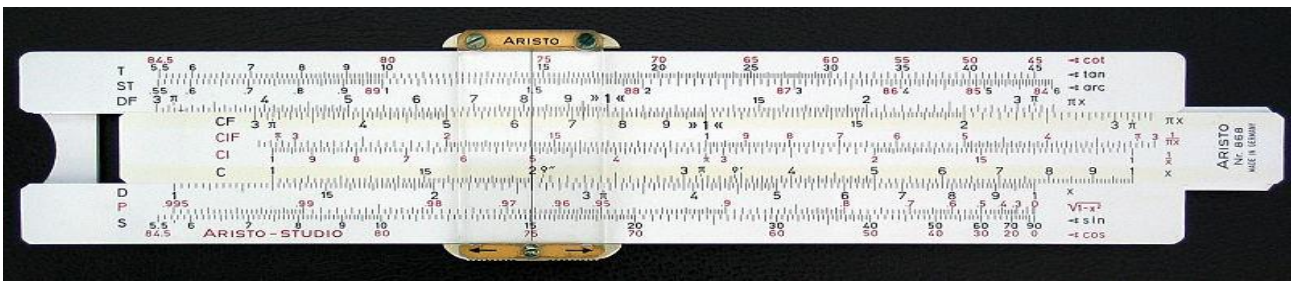
Külön meg kell emlékezni egy nagyon hosszú ideig forgalomban levő eszközről, a logarlécről. Nagyjából az 1970-es évek közepéig a mérnökök, a mérnöki számítások igen fontos eszköze volt. Először csak két lineárisan beosztott skálát tologattak egymáshoz képest, ezzel csak összeadni és kivonni lehetett. Ez az eszköz egy mechanikus analóg számítógép.

1620-ban Edmund Günter (1581-1626) szerkeszti meg az első logarlécet, amelynek elve igen egyszerű, két lécen logaritmikusan (10-es alapú logaritmus) visszük fel az értékeket, ezeket egymáshoz képest tologatjuk. Ekkor a tologatás valójában szorzást jelent. A logarlécek (1.5., 1.6. és 1.7. ábra.) általában 125 mm, 250 mm, ritkán 500 mm hosszúak. Ablak hozzáadásával tovább növelik a számítások körét. A 250 mm hosszú léccel kb. 1 % pontosságot lehetett elérni. Természetesen szorzás-osztás mellett gyökvonásra, szögfüggvényekkel való számításokra is alkalmas volt a logarléc, de tartalmazott természetes logaritmus skálát is. A pontosság mellett a nagyságrend meghatározása a felhasználó feladata volt (tizedes vessző helye). A lineáris logarlécnek létezik kör alakú változata is, melyet Oughtred 1632-ben alkotott meg, egy modern eszköz látható az 1.5. ábrán.



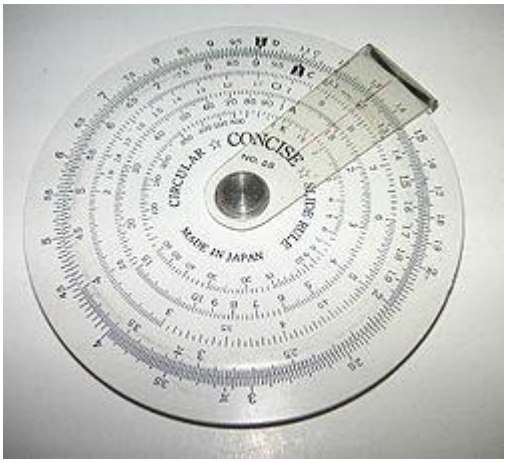
(Forrás: http://hu.wikipedia.org/wiki/Logarl%C3%A9c.)

1.5. ábra. A szorzás elve logarléccel.



(Forrás: http://hu.wikipedia.org/w/index.php?title=F%C3%A1jl:Sliderule_2005.jpg&filetimestamp=20110110214138).

1.6. ábra. 250 mm-es „modern” logarléc.

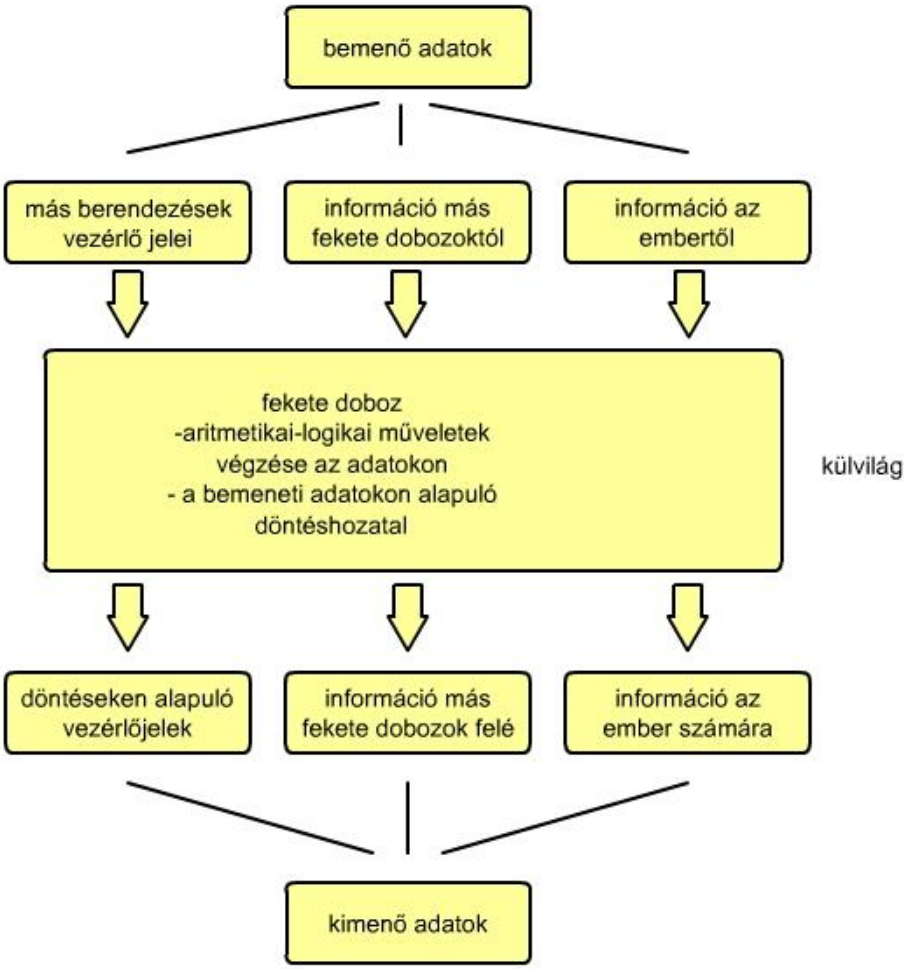


(Forrás: http://hu.wikipedia.org/wiki/Logarl%C3%A9c).

1.7. ábra. Körlogarléc.

A fenti rövid bevezetés után nézzük meg, mit értünk ma számítógép-technika, illetve számítógép alatt.

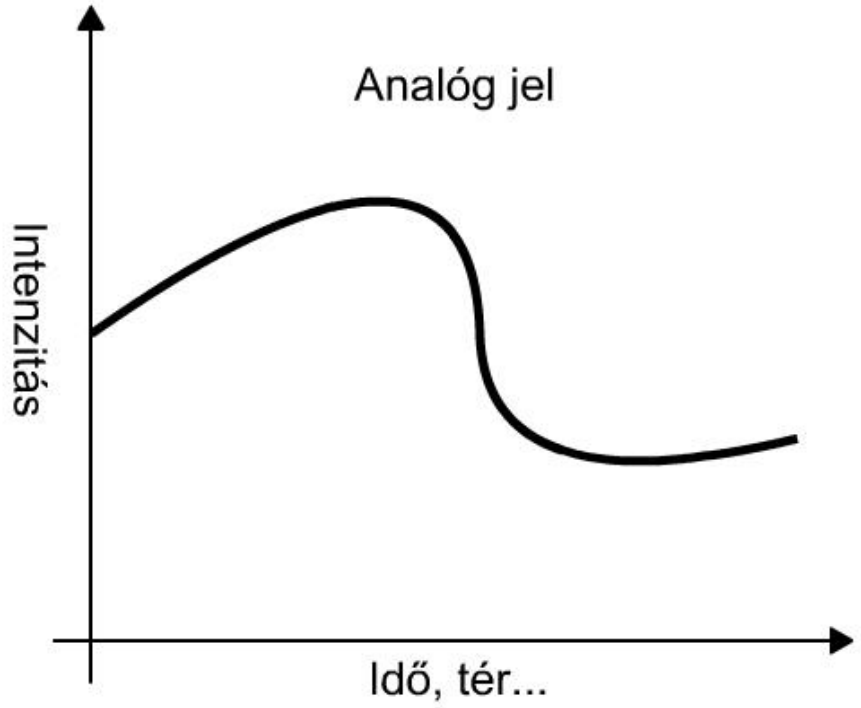
A számítógép valamilyen bemenő adatok alapján előre definiált lépések sorozatával (program) kimenő adatokat állít elő. A bemenő adatok különböző forrásból jöhetnek, embertől, géptől, ipari folyamatból stb. A kimenő adatok vagy közvetlenül értelmezhetők az ember számára (kép, szám, grafikon stb.), vagy valamilyen más berendezés bemenetére kerülnek (ez lehet egy másik számítógép is). Egy ilyen berendezés az 1.8. ábrán látható.



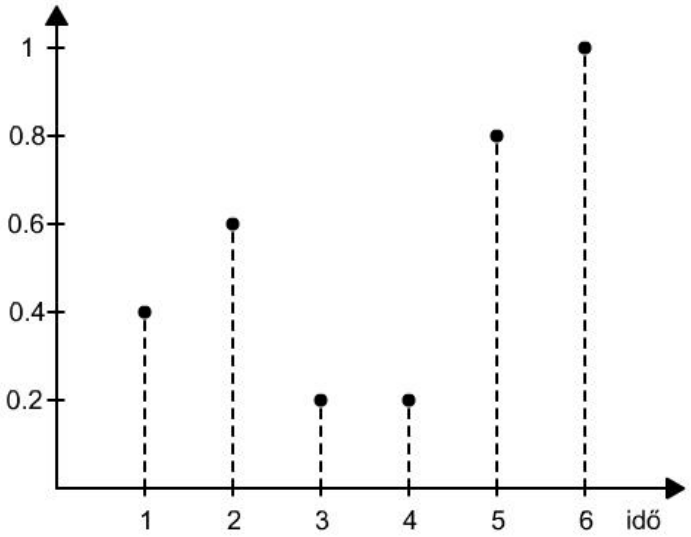
1.8. ábra. Számítógép kapcsolata a külvilággal, más számítógépekkel.

A bemenő jelek ugyanúgy, mint a kimenő válaszjelek a lehető legkülönbözőbb fizikai jelek lehetnek. Két csoportba sorolhatók:

- folytonos, analóg jelek (hőmérséklet, nyomás, szélsebesség stb.), 1.9. ábra és
- diszkrét jelek (kapcsoló ki-be állása, van nyomás nincs nyomás stb., 1.10. ábra.



1.9. ábra. Folytonos jel.



1.10. ábra. Diszkrét jel.

Mint ahogy látható, az analóg és diszkrét jelek között jelentős a különbség. Az analóg jeleket általában időtartományban ábrázoljuk és vizsgáljuk, egy adott tartományban értelmezzük azokat. A meghatározott tartományon belül a jel folytonosan változik.

A diszkrét jeleket is általában időtartományban ábrázoljuk, de csak előre jól meghatározott értékeket vehetnek fel, például ha 10 szint van, akkor 0, 1, 2 stb. 8 és 9. A gyakorlati életben leggyakrabban a kétértékű (bináris) rendszereket használjuk (0 és 1).

A számítógép-technika eszközei (berendezései) valamilyen fizikai jelenségen alapuló berendezések. Így igen gyakran használunk:

- mechanikai,
- elektromos,
- pneumatikus,
- fény,
- mágneses stb.

elven működő eszközöket, illetve ezek kombinációit.

Gyakorlatilag háromféle számítástechnikát különböztetünk meg:

- analóg,
- digitális és
- hibrid.

Ha van valamilyen vizsgálandó jelenségünk, akkor annak részletes, mély vizsgálata sokszor a rendszer bonyolultsága miatt lehetetlen. Ezért modelleket hozunk létre, mely modellek többé-kevésbé fizikailag világos viszonyok között pontosan írják le a valós rendszert, de a számos elhanyagolás miatt könnyebben kezelhetők. Többféle lehetőség áll rendelkezésünkre egy-egy modell létrehozására, ilyen pl. a matematikai modell, vagy a fizikai modell.

A matematika modellben az egyes részek közötti kapcsolatot írjuk le matematikai egyenletekkel.

A fizikai modell megépítése nem hasonlít a matematikai modellre, ezeknél például a valós rendszer kicsinyített, esetleg nagyított modelljét építjük meg, vagy létrehozunk egy olyan fizikai rendszert, amely helyettesíti az eredeti modell fizikai változóit egy könnyen kezelhető mennyiséggel, ez lehet

mechanika, de ma már az elektronika szolgál erre. A fizikai modell létrehozására használjuk az analóg számítógépet, melynek segítségével a megvalósított modellt valós időben működtetjük. Az összes létrejövő folyamatot (legyen az a bemeneten, a modellben, vagy a kimeneten) oszcilloszkópon vagy rajzgépen felrajzoljuk.

A digitális számítógépek alkalmasak nagyon bonyolult matematikai modellek kezelésére, programokat írunk a matematikai egyenletek leírására. Jellemző a digitális gépekre a nagyobb pontosság elérése, a nagy mennyiségű adat tárolása, ezen a két területen az analóg gép sokkal (lényegesen) gyengébb teljesítményt nyújt, viszont az analóg számítógép bizonyos egyenleteket valós időben, gyorsabban oldhat meg, mint digitális társa.

A két módszer, figyelembe véve mindkettő előnyeit és hátrányait jól kiegészítheti egymást, ezért használnak (inkább használtak) hibrid számítógépeket, melyek egymással analóg-digitális és digitális-analóg átalakítókkal kommunikálnak. Mivel az analóg számítógép elemeinek száma korlátozott, valamint összekötésük időigényes, illetve adattároló képességük kicsi, ezeket a feladatokat a digitális gép jobban oldja meg, ugyanakkor sokszor egy analóg modell létrehozása egyszerűbb a digitálisnál, a két gép összekötése egy hatásos számítástechnikai eszközt ad, a hibrid számítógépet.

Ahhoz, hogy környezetünket jobban megismerjük, megpróbáljuk azt modellezni, a modellen paramétereket beállítani és futtatni a modellt. Ezek a modellek sokszor visszavezethetők egyenletek megoldására, sokszor differenciálegyenletekre. Így minden területen, a kutatásban, előrejelzésben, a biológiában, közgazdaságban, mechanikában, elektrotechnikában stb. megtalálhatók.

1.2. AZ ANALÓG SZÁMÍTÓGÉP

Ma kicsi a jelentőségük, ritkán, speciális (főleg rendkívül gyors) estekben alkalmazzuk ezeket a megoldásokat. A kezdetekben mechanikus eszközökkel kísérleteztek, speciális feladatokat megoldó célszámítógépeket építettek, legfejlettebb formájukat az általános feladatokat megoldó elektronikus analóg számítógépek jelentették.

Azon az elven működnek, hogy a fizikai jelenségeket matematikailag írjuk le, szimuláljuk a folyamatokat, be- és kimenetük valamilyen fizikai jellemző (elmozdulás, feszültség, hőmérséklet, nyomás).

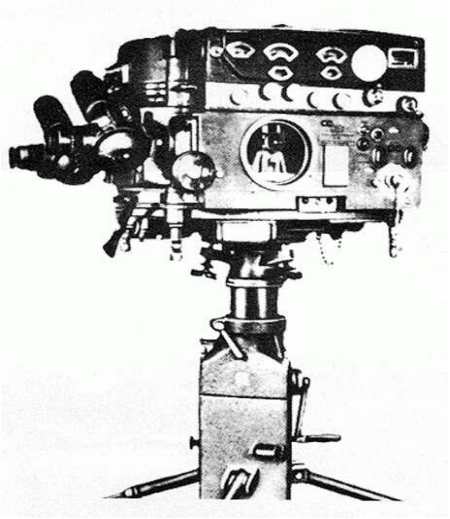
A számítandó mennyiség és annak valamely fizikai mennyiséggel való hasonlósága (pl. elmozdulással arányos feszültség) alapján működő számítógép, mely differenciál- és integrálszámításra közvetlenül alkalmas.

A XX. században már bonyolult analóg gépeket építettek, igaz a század elején még ezek mechanikai megoldások voltak. Ezek a modellek tengelyek és fogaskerekek megfelelő mértékben történő elforgatásával működtek. Egyenletek numerikus megoldásait számolták, más módon nem megoldható egyenleteket lehetett így kezelni.

Ilyen elven Josef Nowak 1910-ben ötismeretlenes lineáris egyenletrendszerek megoldására készített analóg számítógépet.

Igen széles körben és hosszú ideig használatos analóg számítógép született 1914-ben, alkotója Udo Knorr mérnök volt, aki a német vasút számára elkészítette a menetrendkészítő diagráfot. A készülék egy-egy vasútvonal mentén kiszámította a menetsebességet és a menetidőt a vasútvonal profiljának, a mozdony típusának és a szállított teher súlyának figyelembe vételével. Ezt az eszközt tökéletesítették, egészen az 1970-es évekig Németországban használták is.

Háborús célok érdekében, repülők lelövése, torpedók kilövése, bombák célra vezetése stb. érdekében fejlesztettek már az I. világháború alatt, de különösen a II. világháború alatt analóg számítógépeket. Ezek célberendezések voltak, ilyen például a lőelemképző szerkezet. A lövedékek, illetve repülő objektumok ballisztikus egyenletekkel írhatók le, gyorsan kell kiszámítani a repülőgép röppályáját, erre alkalmas az analóg számítógép, ugyanis a szerkezet azonnal adja az eredményt. Egy ilyen, a Gamma-Juhász lőelemképző készülék, ez látható az 1.11. ábrán.



(Forrás: Haditechnika, 1995, 1. szám, 48. oldal.).

1.11. ábra. A Gamma-Juhász féle lőelemképző készülék.

Nagy előrelépés volt az analóg számítástechnika területén a Massachusettes Institute of Technology-n (MIT) Vannevar Bush és munkatársai által elkészített első univerzális (tehát nem cél) analóg számítógép. Pontossága 0.1 % volt (ez nem érte el az elektronikus analóg számítógépeknél elérhető 0.001 %-ot).

1930-ban Vannevar Bush és kollégái elkészítették a differenciálanalizátor nevű készüléküket, amit egyszerűbb differenciálegyenletek megoldására használtak. Ez volt az első univerzális

analóg számítógép. A készülék 0,1% pontossággal dolgozott és évtizedekig konkurencia nélkül uralta a piacot. A mérnök és gépe látható a 1.12. ábrán.

(Forrás:http://www.science.uva.nl/museum/vbush_tbl.php)

1.12. ábra. Univerzális analóg számítógép, MIT, 1930.

1.3. A DIGITÁLIS SZÁMÍTÓGÉP

Ezek a számítógépek az analóg gépekétől teljesen eltérő módon működnek, mind az adatok, mind a feladatok számjegyekre (digitekre) vannak bontva, amit kétállapotú logika segítségével ábrázolunk (0 és 1). Itt kell letisztázni azt, hogy van kettes számrendszer és van kétállapotú logika. A két fogalom között van különbség, de formailag mindkét rendszer két állapotot, például 0-át és 1-et különböztet meg.

Ha összehasonlítjuk a két technika, a két módszer egyes tulajdonságait, akkor azt látjuk, hogy a digitális megoldásnak három területen van előnye az analóggal szemben:

- a számítások pontossága, analóg technikánál (főleg az elektronikus megoldásnál) csak 0.001 % lehet, szemben a bináris megoldással, ahol egy bit hozzáadása kétszerezi a szám nagyságát, vagy kétszer nagyobbá teszi a felbontást, így mindig egy előre meghatározott számnagyságot illetve pontosságot érhetünk el,
- adattárolás analóg módon gyakorlatilag lehetetlen (csak a kondenzátor alkalmas erre, de mivel nem ideális elem, idővel veszíti az információt), szemben a bináris digitális rendszerek akár félvezetők segítségével, akár optikai vagy mágneses módon történő tárolási technikájával és
- a számítógép gyors programozása, átprogramozása analóg módon nehézkes, vagy lehetetlen, míg a digitális rendszereknél ez egy egyszerű gyors folyamat.

A digitális számítógép-technika kialakulása egy folyamat, bizonyos találmányok, technikai újítások megjelenése ugrásszerűen megváltoztatja a rendszereket. Jelenleg a kifejezetten elektronikus, bináris, digitális számítógép (például a PC személyi számítógép) nem csak elektronikus elemeket tartalmaz, mechanika (Winchester, DVD), mágneses elv (Winchester), optika (DVD, száloptika), az ipari számítógépeknél pneumatika és hidraulika egészíti ki az elektronikát. A fő elemek, melyek a számításokat végzik, a programokat hajtják végre döntően elektronikusak, hiszen a felsorolt technikákat összehasonlítva az elektronika mérete a legkisebb, a feldolgozási sebessége a legnagyobb, a fogyasztása a legkisebb és a megbízhatósága a legnagyobb.

Ebben a bevezetőben egy rövid áttekintést nyújtunk a számítógép-technika fejlődéséről, de ne felejtsük el, hogy rengeteg ponton átfedés van a számítástechnikával, úgyhogy nem szétválasztható a két terület. Ma, amikor a számítógép szó elhangzik, nagyon sokan a PC személyi számítógépre gondolnak, de ez csak egy szelete a számítógépeknek, ebben a jegyzetben tágabb értelemben dolgozzuk fel a területet.

Többféle módon lehet tárgyalni a fejlődést, mi itt generációkra osztjuk fel, fontosabb változásokat évekhez kötünk, de ezek a mérföldkövek nem máról-holnapra történő élesen bekövetkező változások, sok megoldás gyorsan eltűnik, bizonyos megoldások pedig hosszú ideig léteznek, átívelnek több számítógép generáción.

1.3.1. A 0. SZÁMÍTÓGÉP GENERÁCIÓ (- 1946)

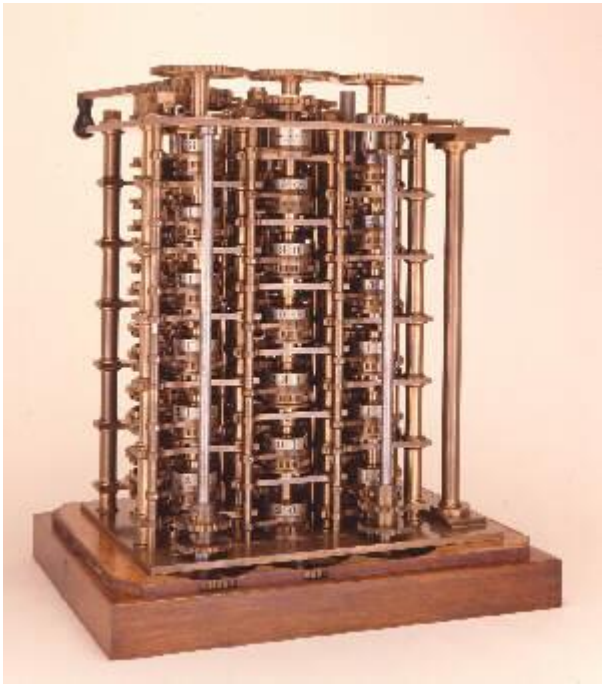
Ez a 0. generáció valójában a legelső számítástechnikai és számítógép-technikai cél- és általános gépeket öleli fel, egészen 1946-ig. A teljesség igénye nélkül említünk meg néhány érdekes megoldást, illetve olyan működő rendszert, amelyek valamilyen módon hatással voltak mai rendszereinkre.

Már szóltunk a hajózásban sokáig nélkülözhetetlen óráról és szextánsról, de feltétlenül meg kell emlékezni Pascal (1623-1662) és Leibniz (1646-1716) munkásságáról, mindketten mechanikus elven működő számológépet alkottak, Pascal gépe (Pascaline - 1642) csak összeadni-kivonni tudott, fogaskerekekből épült fel, 6 digites számábrázolása volt. Leibniz gépe már szorozni-osztani is tudott, de megjelenésére kb. 31 évet kellett várni.

Az 1800-as években Babbage (1791-1871) munkássága is figyelemre méltó, megkísérli a számítógépek elméletét kidolgozni. Ő az első tudós, aki kísérletet tesz a programozható számító-gépek megalkotására. Több elkezdett gépe van, sajnos gyakorlatilag teljesen egyet sem fejezett be. Hajózási navigációs adatokat szeretett volna meghatározni differenciáló gépével. 20 digitesre tervezte azt, de soha nem tudta teljesen megépíteni. Érdekesség, hogy később (az 1990-es években) megépítették tervei alapján a differenciáló gépet, ami tökéletesen működött. Ötletei, megoldásai még ma is használatosak a modern számítógépekben. Analitikai számítógépe négy fő részből állt, ezek:

- tárolóegység,
- számításokat végző egység,
- lyukkártya-olvasó és
- lyukkártya lyukasztó (vagy nyomtató).

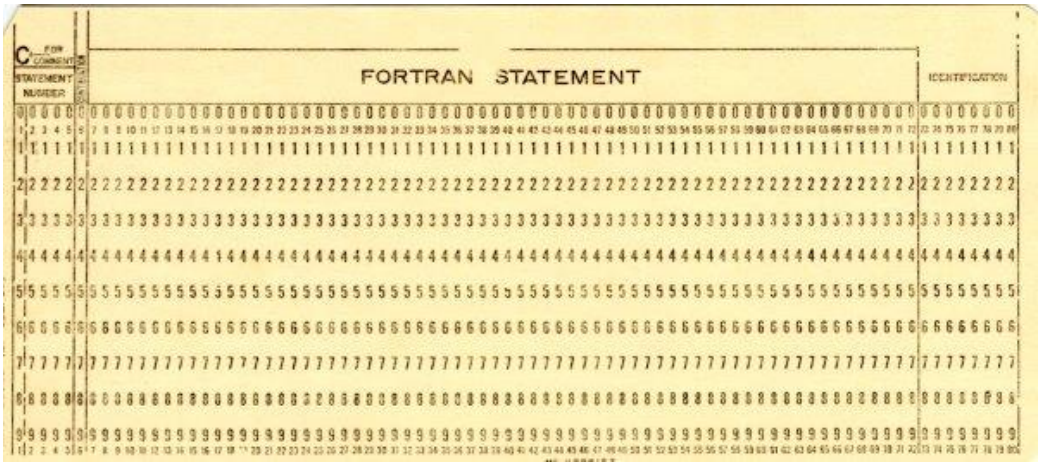
Egyáltalán nem nehéz ráismerni a mai számítógépeknél használt memóriára, központi egységre, bemeneti- és kimeneti eszközre.



(Forrás: http://www.charlesbabbage.net/).

1.13. ábra. Babbage differenciáló gépe.

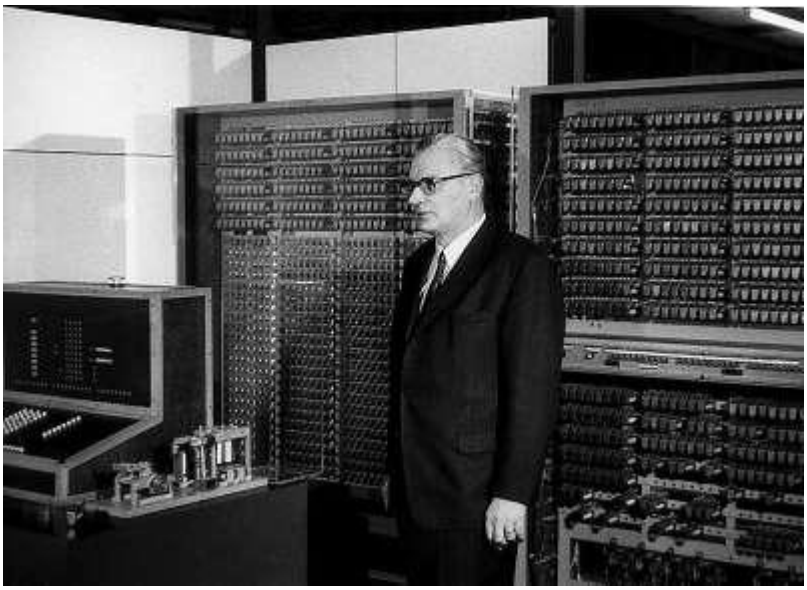
Igen érdekes helyet foglal el Hollerith (1860-1929) a számítástechnikában, eredetileg automatizálni szerette volna a népszámlálási adatok feldolgozását, amit sikerrel oldott meg egy lyukkártyás rendszerrel. Készített egy villanymotoros hajtású, elektromos osztályozó-feldolgozó rendszert, amely találmányából a lyukkártya-szabvány alakult ki, adatok és programutasítások tárolására használták, még az 1970-es évek végén is sok helyütt használatban voltak.



(Forrás: http://it.wikipedia.org/wiki/File:Hollerith_card.jpg).

1.14. ábra. A Hollerith-féle lyukkártya.

Meg kell emlékeznünk Zuse (1910-1995) német mérnökről, aki közvetlenül a II. világháború előtt (1938) elkészíti a Z1 számítógépet, ez még mechanikus, majd 1941-ben a Z2-t, mely már elektro-mechanikus elven működik (jelfogókkal), amit követ a Z3, ahol programvezérelt megoldásokat alkalmaz megalkotója, ráadásul már kétállapotú (bináris) logikán alapult a gép működése.



(Forrás: http://radio-weblogs.com/0105910/2004/06/07.html).

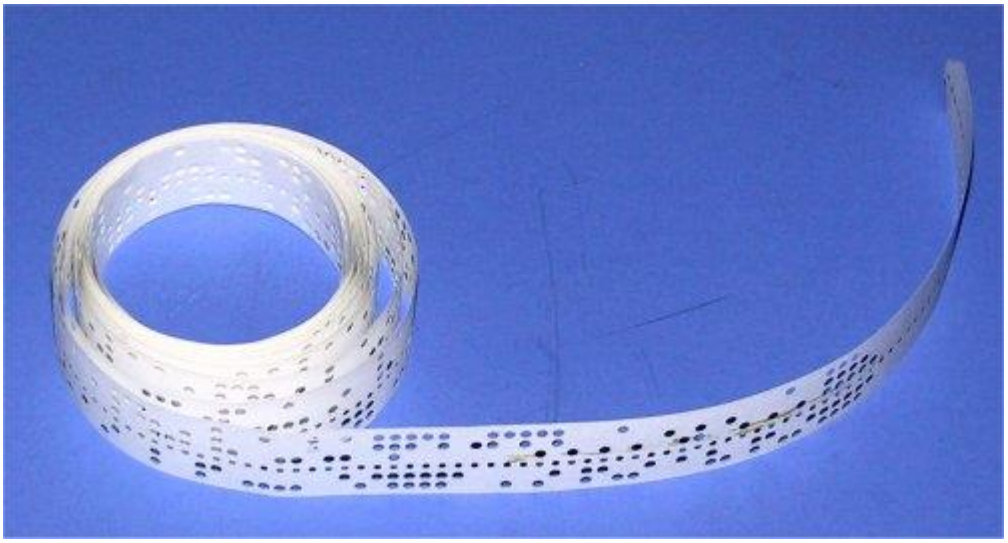
1.15. ábra. Konrad Zuse mérnök Z3 számítógépe előtt.

Zuse megelőzte korát, hiszen csak 1944-ben tudott Aiken (1900-1973), Lake (1888-1958) és Hamilton hasonló paraméterű gépet megalkotni, a MARK I-et, igaz ez 10-es számrendszerben dolgozott, Babbage elvei alapján építették meg. Aiken tökéletesítette a gépet, ez a MARK-II volt, 1947-re készült el.

(Forrás: http://infostory.wordpress.com/2011/08/07/computing-milestones/).

1.16. ábra. A MARK I számítógép.

Csak érdekességképpen jegyezzük meg, a gép 15.3 m hosszú, 2.4 m magas, 31,500 kg nehéz volt, 800 km huzal volt benne. A kapcsolatok száma 3 millió volt. A programutasításokat lyukszalagról olvasta be.



(Forrás: http://retropages.uw.hu/theme_45.html).

1.17. ábra. Lyukszalag.

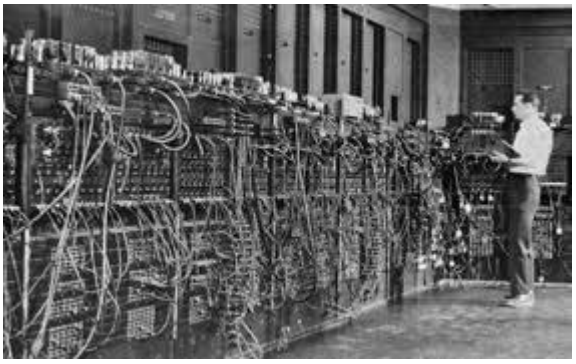
A MARK II is igen lassú gép volt (ez is jelfogókkal épült meg), két számot 0.5 mp alatt adott össze, 6 mp alatt szorzott össze és 15 mp alatt osztott el.

1.3.2. AZ 1. SZÁMÍTÓGÉP GENERÁCIÓ (1946 - 1952)

Ekkor kezdik el alkalmazni a számítógép-technikában az elektroncsöveket. Ez egy üvegcső, vagy vákuum, vagy valamilyen kis nyomású gáz van benne. Aktív eszköz, dióda, trióda stb. készíthető belőle. A számítógép-technikában mint kapcsolóelemet alkalmazták. Érdekes, már az 1800-as évek végén voltak működőképes példányaik, a 20. században pedig hangtechnikai erősítőkben, visszacsatolt erősítőkben, mérőműszerekben alkalmazták, de számítógépekben való alkalmazásukra igen későn került sor. Ennek egyszerű a magyarázata, megbízhatóságuk hosszú ideig igen kicsi volt, így olyan berendezésekben, ahol nagy számban fordulnak elő, a gyakori hibák miatt lehetetlen volt gyakorlati felhasználásuk. Tökéletesítésük után nyílt meg az út alkalmazásukra.

A 2. világháború nagy lökést ad a számító-gépek fejlődésének. Elkészül Angliában az első elektronikus digitális számítógép, a COLOSSUS a német ENIGMA kódok megfejtésére. Sajnos ezt a projektet 30 évre titkosítják, az itt alkalmazott megoldások nem hatnak ki a későbbi számítógépek fejlődésére.

Az elektronikus, digitális programozható számítógép az ENIAC (Electronic Numerial Integrator And Computer), 1946 február 14-én készült el, tízes számrendszerben dolgozott (tízjegyű előjeles számok). Eredetileg ez is katonai célokat szolgált volna, de csak a háború befejezése után volt működőképes. A programozás gépi kódban történt, kapcsolótáblán állították be. 17.468 elektroncsövet használtak benne mint kapcsolóelemet, tartalmazott 72.000 kristálydiódát és 1.500 jelfogót, 5 millió forrasztással oldották meg az elemek összekötését. Hossza 40 m, magassága 2.5 m, súlya pedig 30.000 kg. Az összeadást és kivonást a MARK II-höz képest 500-szor gyorsabban végezte el. Az elektroncsövek megbízhatósági paraméterei miatt moduláris volt a rendszer, általában 15-20 percenként történt meghibásodás, körülbelül kétnaponta kellett modult cserélni, ez egy 20 perces művelet volt. Mivel hadiipari célokat (is) szolgált, ballisztikus pályák számításánál differenciálegyenletek megoldására is alkalmas volt.



(Forrás: http://blog.peopleanswers.com/blog/2012/3/13/yottabytes-now-thats-big-data.html).

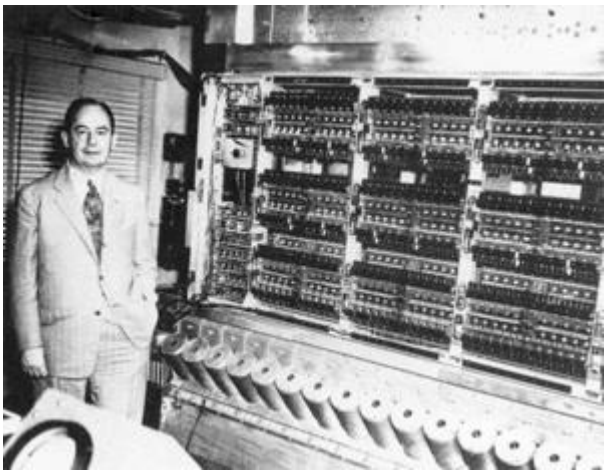
1.18. ábra: Az ENIAC elektroncsövekkel megépített számítógép.

A hardver fejlesztése mellett egyéb fejlesztések is megindulnak, így erre a periódusra tehető a fixpontos számábrázolás szabályainak meghatározása.

A fejlesztések lehetővé tették a sorozatgyártás irányába való elmozdulást, igaz ekkor még a számítógépek árai igen borsosak voltak, főleg nagy cégek vásárolhatták, illetve nagy egyetemek. Az első sorozatban gyártott számítógép az UNIVAC (Universal Automatic Computer), mely 1951-ben jelent meg (1951.06.05.). 1955-ig ebből a típusból 46 gépet helyeztek üzembe.

1951 az az év, amikor is olyan elven működő számítógép kerül megépítésre, amelyik mai napig meghatározza a működési elvüket, ennek az elvnek megalkotása, illetve a gépnek a megépítése

Neumann Jánoshoz (1903-1957) fűződik. Vegyészmérnök, matematikus, tanulmányait több helyen végezte.



Forrás: Wikipedia

1.19. ábra. Neumann János az IAS számítógéppel.

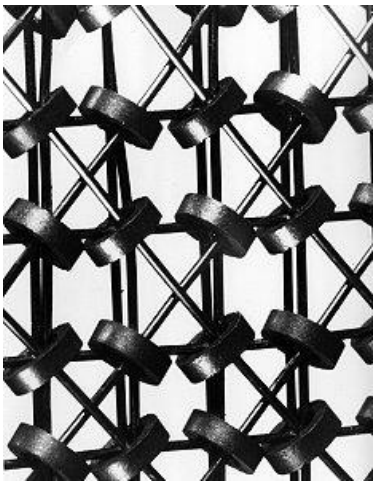
A Neumann-elv szerint a számítógépben kettes (bináris) számrendszert kell alkalmazni, a memória adatokat és programot tárol kétállapotú logikával, a vezérlőegység utasításokat hajt végre. Ez az elv a mai számítógépek működését is jellemzi. Neumann sikeresen működött együtt Kemény Jánossal (1926-1992), aki megalkotta a BASIC programoíási nyelvet.

Meg kell még emlékezni az 1. generáció IBM által gyártott 370-es sorozatáról, egy széleskörűen használt sikeres számítógéprendszerről, 1953-tól vezették be a piacra.

1.3.3. A 2. SZÁMÍTÓGÉP GENERÁCIÓ (1955-1964)

Forradalmasítja az elektronikát és így a számítógép-technikát is a bevezető elején említett tranzisztor felfedezése, illetve megépítése. Az ott felsorolt tulajdonságai miatt a számítógépek elterjedése (többek között anyagi okok miatt) ugrásszerűen növekszik.

A 2. generációra jellemző a ferritgyűrűs főtár alkalmazása is. Ezek kis állandómágnes (ferrit) gyűrűk, melyek lehetővé tesznek viszonylag gyors működésű, nagy kapacitású főtár kiépítést. Ebben az időben a nagy kapacitás néhány kB (kolobájt) méretet jelentett.



(Forrás: http://www.computerhistory.org/revolution/memory-storage/8/310).

1.20. ábra: Ferritgyűrűs főtár.

Ezekben az években terjed el a fixpontos aritmetika mellett a lebegőpontos alkalmazása is, ez lehetővé teszi a számok ábrázolásánál a nagy- és kis számok kényelmesebb ábrázolását és kezelését.

A háttértárak iránti nagyobb követelmények (tárolási kapacitás, elérési sebesség, megbízhatóság, kis fogyasztás) miatt az érdeklődés a mágneses elv felé fordul, megépítik a mágnesszalagos és a mágneslemezes egységeket. A mágnesszalag kifejlesztése sok gondot vetett fel, nem volt előzménye. Az IBM készítette el 1952-re és hozta forgalomba az első működő típust. Elterjedése a második generációs gépek korszakára esik. Lényegesen gyorsabb volt mint lyukkártyás és szalagos társa, kapacitása is messze meghaladta azokét. Igen nagy mennyiségű adatot vittek át a lyukkártyákról mágneses háttértárolókra.

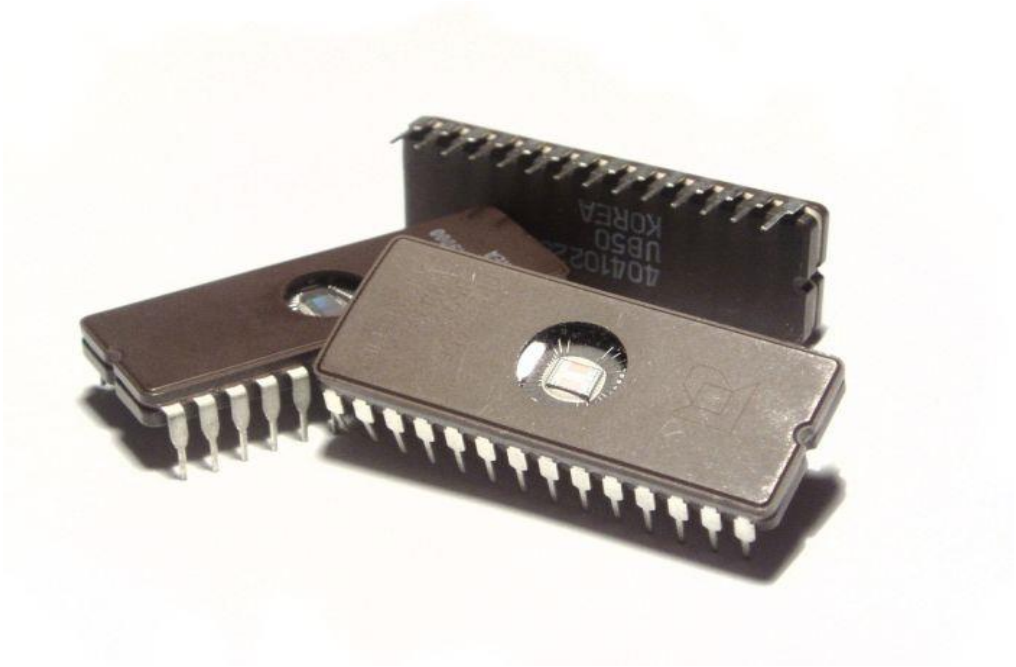
(Forrás: http://ibm.web4.hu/cikk/a-lyukkartyatol-a-totalis-adattarolasig).

1.21. ábra. Az IBM 702 számítógép-technika.

A második generációs számítógépek használata igényelte a magas szintű programozási nyelveket, ezek között kiemelkedő helyet foglalt el a FORTRAN.

1.3.4. A 3. SZÁMÍTÓGÉP GENERÁCIÓ (1965-1974)

Az elektronikus berendezések méreteinek és súlyának csökkentése érdekében ekkor kezdik el egyetlen lapkán kialakítani az áramköröket, vagyis megindul az integrált áramkörök korszaka. Kis- és nagy bonyolultságú áramkörök kerülnek egy-egy ilyen műanyag, vagy kerámia házba. Ilyen esetben már rendszertervezésről és rendszerépítésről lehet beszélni.

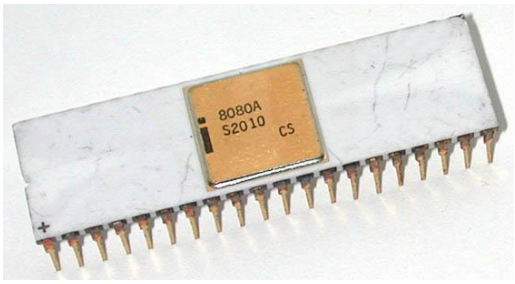


1.22. ábra. Integrált áramkörök.

A számítógép-elmélet fejlődése létrehozza a multiprogramozást, így jobban lehet kihasználni a nagyobb sebességű, kapacitású számítógépeket, megjelennek az operációs rendszerek, melyek biztosítják a hardver és szoftver közötti együttműködést. Ekkor kezdik el fejleszteni az időosztásos üzemmódot. Létrehoznak több új magasszintű programozási nyelvet (COBOL, PL/1, BASIC, PASCAL stb.).

Megjelenik a közvetlen hozzáférésű merevlemez-tár, ez biztosítja a nagy mennyiségű adat tárolását. Megjelennek a monitorok, nyomtatók rajzgépek. Mindezek a fejlesztések olyan rendszereket biztosítanak, amelyek áruk miatt széleskörűen hozzáférhetők.

1973-ban elkészül az első valódi mikroprocesszor, amely ára és műszaki paraméterei miatt széleskörűen felhasználható ipari alkalmazásokban is. Két cég kezdi el gyártani, az egyik az Intel, a 8080A jelzésű mikroprocesszorral (1.23. ábra), a másik pedig a szintén 8 bites Motorola 6800-as (1.24. ábra) jelenik meg a piacon.



1.23. ábra. Intel 8080A mikroprocesszor.

1.24. ábra: Motorola 6800 mikroprocesszor.

1.3.5. A 4. SZÁMÍTÓGÉP GENERÁCIÓ (1975-1985)

Tovább folytatódik a miniatürizálás, ekkor az LSI (Large Scale Integration- nagyfokú integráltság) mellett létrejön a VLSI (Very Large Scale Integration – nagyon nagy fokú integráltság) is.

Lehetővé válik a szuperszámítógépek megépítése, amelyek alkalmasak bonyolult fizikai és matematikai feladatok megoldására, nagy adatbázisok kezelésére.

Ekkor alakulnak ki a számítógépes rendszerek is.

A mikroszámítógépek piacán megjelennek a házi számítógépek, Commodore, ZX Spectrum stb., de ezek nem voltak szabványosítva, így problémát okozott az adatcsere közöttük.

1.25. ábra. A ZX Spectrum házi számítógép.

1978-ban építi meg az Intel a 8086-os processzort (1.26. ábra), ami alapját képezi az IBM XT személyi számítógépnek. A Motorola is megjelenteti 16 bites processzorát (1.27. ábra.), amelyik erősebb, jobb paraméterekkel rendelkezett, mint az Intel 8086, de más paraméterek (pl. ár, gyártási jog) miatt más berendezésekben alkalmazzák.

1.26. ábra. Intel 8086 processzor.

1.27. ábra. Motorola 68000 processzor.

1.28. ábra. IBM XT személyi számítógép.

1.3.6. AZ 5. SZÁMÍTÓGÉP GENERÁCIÓ (1985-2012)

Az előző, 4. számítógép generáció IBM PC számítógépe tovább fejlődik, mind a processzorok, mind a perifériák tökéletesebbé válnak. Az asztali gépek mellet notebookok (1.29. ábra) és netbookok (1.30. ábra.) terjednek el. A netbook kisebb teljesítményű processzorral kevesebbet fogyaszt, de könnyen hordozható mérete miatt (általában 10 ” képernyőmérettel készül).

1.29. ábra. Notebook.

1.30. ábra. Netbook.

Megjelennek a különféle processzor-sebesség növelő technológiák, egészen a többmagos processzorokig. A hardver fejlődésével párhuzamosan a szoftver oldal is igen erőteljesen fejlődik. Megjelenika a multimédia, az internet.

Jellemző különböző technológiák összefonódása, a mobil telefonok okos telefonná válnak (1.31. ábra), teljes számítógép funkciókat látnak el (jellemzően 3.5 ” – 5 ” közötti képernyőméret, de ma már vannak

6, sőt 7 ” méretűek is). Nagyobb testvérük a táblagép (1.32. ábra.) pedig igen hatásos, hordozható eszközt biztosít (jellemzően 7 ” – 10 ” közötti képernyőméret).

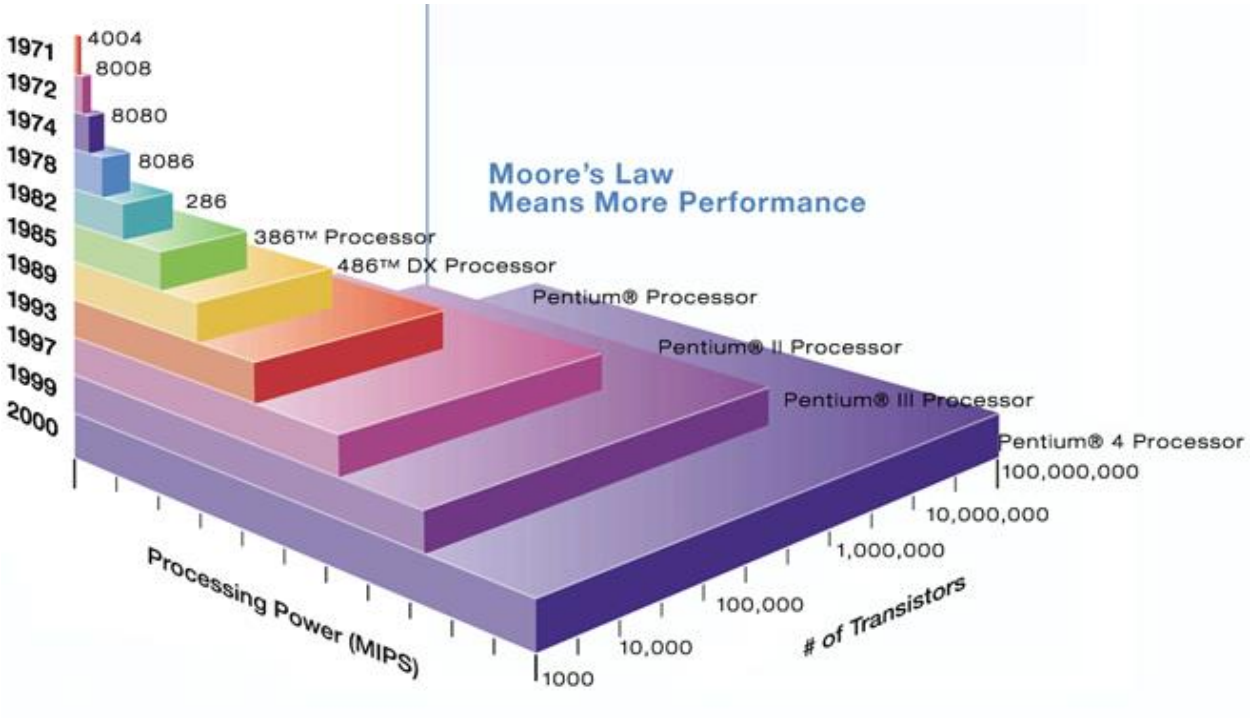
1.31. ábra. Okostelefon.

1.32. ábra. Táblagép.

A televíziózás magába integrálja a számítógép-technikát is. A 3D mellett internetezni is lehet már a TV-vel, okso televíziók (smart TV) teszik kényelmesebbé mindennapjainkat.

A kijelzőknél külön figyelmet igényel az LCD monitorok megjelenése, ez fogyasztásban, méretben és minőségben is döntően átformálta a számítógép-táblagép piacot.

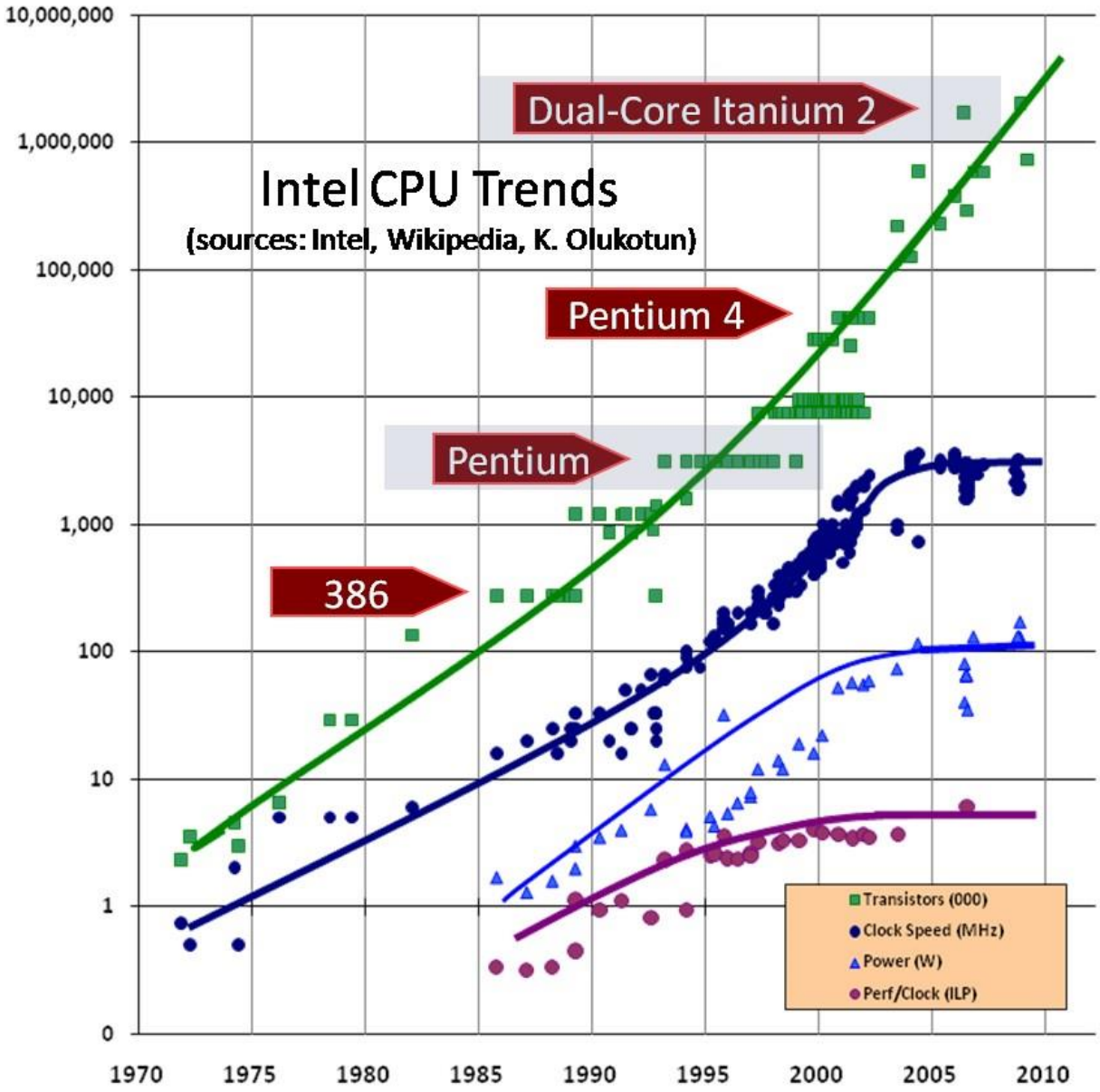
1965-ben Gordon Moore angol tudós kísérletet tett a fejlődés ütemének meghatározására. Szerinte 14-18 hónap alatt megduplázódik a lapkákon elhelyezett elemek (tranzisztorok) száma és a lapkák teljesítménye.



(Forrás: http://software.intel.com/en-us/articles/de-mystifying-software-performance-optimization).

1.33. ábra. Moore törvénye a félvezetőgyártásban.

A fenti ábrán szemléletesen láthatjuk Moore törvényének igazát, az évek függvényében (függőleges tengely) a tranzisztorok számának növekedése (logaritmikus skála) és a számítási teljesítmény látható.



1.34. ábra. Moore törvénye órajellel és fogyasztással kiegészítve.

A fenti ábrából leolvasható egy érdekes összefüggés, igaz az elemek számának mésfél évenkénti növekedése, de már nem növekszik jelentősen a feldolgozási sebesség (órajelfrekvencia) kb. 10 éve, illetve ezzel párhuzamosan a fogyasztás sem.

A jelenleg elérhető csíkszélesség az integrált áramkörök gyártásakor már a 30 nm alatti tartományban van (16-18 nm), ez gyakorlatilag már nagyon nehezen csökkenthető a technológiai korlátok miatt. Egyes tudósok szerint egészen atomi méretekig el lehet menni, de ez bizonyos elektronikai-technológiai paraméterek miatt valószínűleg nem, vagy igen komoly erőfeszítésekkel oldható meg.

Külön probléma az, hogy a memóriáknál és háttértáraknál ez a szabály nem érvényes, így össze kell hangolni a processzor és memóriák közötti sebességkülönbségeket.

Gyakorlatilag a lapkák síkban elhelyezett áramköröket tartalmaznak, a méret további csökkentése, illetve a teljesítmény növelése 3D technológiával lehetséges, már vannak igéretes megoldások laboratóriumokban. Itt nemcsak az elemszám-növelés az érdekes, hanem az egymás felett elhelyezett rétegekben levő blokkok összekapcsolását biztosító vezetékek hosszának rövidülése, ami teljesítménynövekedést eredményez.

Meg kell még említeni gazdasági problémákat is, a mind vékonyabb csaíkszélességű technológia egyre drágább beruházásokat igényel. Egy-egy fejlesztés akár 3-5 évig is eltarthat, míg kétszeres elemszám-növekedés lapkánként kb. másfél év, így sokszor megtörténik a csak néhány hónapos késésnél is, hogy az új áramkör eldhatatlan.

2. A SZÁMÍTÓGÉPEK MATEMATIKAI ÉS LOGIKAI ALAPJAI

Ebben a fejezetben megismerkedünk a legalapvetőbb digitális technikai fogalmakkal, annak érdekében, hogy megértsük a bináris számrendszerben dolgozó, kétállapotú logikai elemekkel megépített digitális rendszer, a számítógép működését. Ezt a teljesség igénye nélkül fogjuk megtenni, csak annyi ismeretet feldolgozva, ami elegendő a számítógépek felépítésének és működésének megértéséhez.

A fejezet elméleti ismeretei a jegyzethez tartozó animációs mellékletben szimulálhatók, így könnyebb az anyag megértése, megtanulása. Ez a rész a 01. és 02. MELLÉKLET-ben található meg.

A számítógépek építőelemei logikai áramkörök, amelyek diszkrét értékű jelekből a logikai kapcsolás szerint logikai kimenő jeleket állítanak elő. A logikai áramkörök építőelemei a logikai kapuk, amelyek valamilyen fizikai építőelem állapotának két értékét vehetik fel, az elektronikus számítógépeknél ez két különböző feszültség-érték. A logikai kapuk tulajdonképpen kapcsolóelemek, amelyek valamilyen függvény szerint engedik, vagy tiltják a kimeneten a jel megjelenését. Bizonyos esetekben ez a változás órajelhez kötött.

Mai számítógépeink a bináris állapotot használják az adatok és az utasítások ábrázolására, így:

Logikai 0, 0 V, illetve a TTL szabvány szerint 0 – 0.8 V közötti tartomány, (LOW, FALSE, NEM),

Logikai 1, 5 V, illetve a TTL szabvány szerint 2.4 – 5 V közötti tartomány, (HIGH, TRUE, IGEN).

Két szint különválasztása könnyebb, mint 3, 4 stb. szint szétválasztása, ezért terjedt el a kétállapotú logika.

2.1. SZÁMRENDSZEREK, SZÁMOK ÁBRÁZOLÁSA

A számrendszerek jelképek, melyek alkalmasak valós számok ábrázolására, figyelembe véve az alkalmazásra szolgáló szabályokat.

A helyiértékes számrendszer a legalkalmasabb a valós számok ábrázolására. A valós számot karakterek sorozatával ábrázolják, úgy, hogy minden számjegypozícióhoz valamely helyi érték van rendelve, értékét az egyes helyi értékek és a hozzájuk tartozó alaki értékek szorzatainak összege határozza meg.

Mind a digitális technikában, mind a számítógép technikában a számrendszerek fontos szerepet játszanak. Megfelelő alapszám választása után a valós számot az alapszám hatványával írjuk fel. Az alapszám 1-nél nagyobb egész szám, leggyakrabban 2, 10 és 16.

2.1.1. TÍZES, VAGY DECIMÁLIS SZÁMREDSZER

A mindennapi élet legelterjedtebb számrendszere, alkalmas negatív és pozitív egész- és tört számok ábrázolására, alapszáma a 10. A karakterkészlet elemei: 0, 1, 2, 3, 4, 5, 6, 7, 8 és 9.

A 987.65 karaktersorozat a tízes számrendszerben:

$$987{,}65 = 9 \cdot 10^2 + 8 \cdot 10^1 + 7 \cdot 10^0 + 6 \cdot 10^{-1} + 5 \cdot 10^{-2}$$

2.1.2. KETTES, VAGY BINÁRIS SZÁMRENDSZER

A számítógép-technika számrendszere, alkalmas negatív és pozitív egész- és tört számok ábrázolására, alapszáma a 2. A karakterkészlet elemei: 0 és 1.

Mivel a gyakorlatban a tízes számrendszert használjuk, viszont a számítógépeink binárisak, ezért a két számrendszer között meg kell határoznunk a kapcsolatot. A következő táblázatban néhány szám tízes-, illetve kettes számrendszerbeli értékét láthatjuk.

2.1. táblázat. Kapcsolat a tizes és a kettes számok között.

Tízes számredszerbeli írásmód	Írásmód 2-es alapú hatványok összegével	Kettes számrendszerbeli írásmód
0	$0 \cdot 2^0$	0
1	$1 \cdot 2^0$	1
2	$1 \cdot 2^1 + 0 \cdot 2^0$	10
.	.	.
9	$1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$	1001
3.25	$1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}$	11.01

2.1.3. TIZENHATOS, VAGY HEXADECIMÁLIS SZÁMRENDSZER

A számítógép-technika kettes számrendszerének 4 bitenként való ábrázolása az áttekinthetőség érdekében ($16 = 2^4$), alkalmas negatív és pozitív egész- és tört számok ábrázolására, alapszáma a 16.

A karakterkészlet elemei decimálisan: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 és 15.

A karakterkészlet elemei hexadecimálisan: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E és F.

A számítógépeknél az adatok, utasítások, címek stb. binárisak, ezek ábrázolása 2-es számrendszerben bonyolult és nehezen áttekinthető, ezért 4 bitenként összefogva az értékeket a hexadecimális számrendszerben tömörebben, áttekinthetőbben ábrázolhatjuk azokat.

Például, ha az 11000101 bájt-tartalmát szeretnénk hexadecimálisan ábrázolni, csak 4 bitenként kell azt csoportosítani, így kapjuk 1100 helyett a C, 0101 helyett pedig az 5-öt, vagyis $11000101_2 = C5_{16}$ (az indexben szereplő szám, 2 és a 16 a számrendszer alapját jelöli). Szokásos még az indexben levő szám helyett a szám utáni betűjelölés is, a bináris számoknál b, a tizedes számoknál a d (decimális) és a hexadecimális ábrázolásnál a h használata, így az előző példa: 11000101b = C5h.

2.1.4. BCD, BINÁRISAN KÓDOLT DECIMÁLIS SZÁMOK

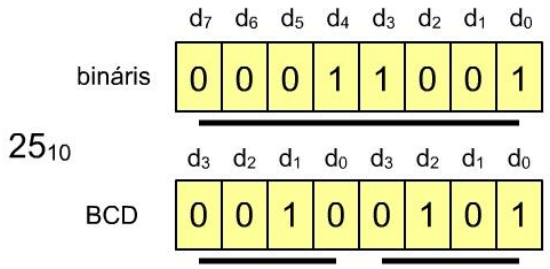
Minden tízes számrendszerbeli számjegyet egyenként 4 biten kódolunk. Más szóval, egy 8 bites szám BCD értéke 4-4 bit decimális értéke.

A 2.1. ábra szemlélteti 4 bit lehetséges értelmezését, a hexadecimális számokat, illetve a BCD számokat.

	d_3	d_2	d_1	d_0	
0	0	0	0	0	BCD
1	0	0	0	1	BCD
2	0	0	1	0	BCD
3	0	0	1	1	BCD
4	0	1	0	0	BCD
5	0	1	0	1	BCD
6	0	1	1	0	BCD
7	0	1	1	1	BCD
8	1	0	0	0	BCD
9	1	0	0	1	BCD
A	1	0	1	0	HEX
B	1	0	1	1	HEX
C	1	1	0	0	HEX
D	1	1	0	1	HEX
E	1	1	1	0	HEX
F	1	1	1	1	HEX

2.1. ábra. Hexadecimális és bináris számok ábrázolása.

A baloldali oszlop a tízes számrendszerbeli értéke a BCD (0-tól 9-ig) és a hexadecimális (0-tól 15-ig) számoknak. A d jelölés a data (adat) szó első betűje, míg az index jobbról balra 0-tól, 3-ig. Az index értékének és sorrendjének megválasztása nem véletlen, így kapjuk meg a természetes bináris kódot, ahol is a 2 hatványa az index értéken adja a szám súlyozott helyértékét. A legkisebb helyi érték jobb oldalon 1, balra haladva előző érték kétszerese, és így tovább. Az adott helyértéknél levő 1 (egy) jelzi, hogy figyelembe kell venni az érték kiszámításánál a bitet, a 0 (nulla), hogy nem.



2.2. ábra. Példa egy tízes számrendszerbeli szám bináris és BCD ábrázolására.

2.1.5. SZÁMOK ÁBRÁZOLÁSA REGISZTEREKBEN

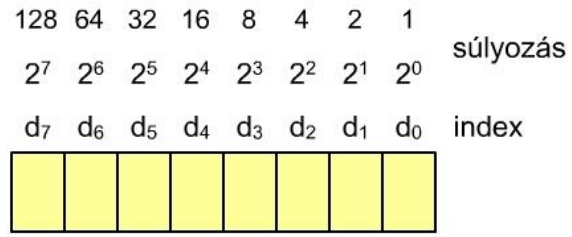
A bináris számábrázolás alapja a bit, ez két érték ábrázolását, 0 és 1 teszi lehetővé. Nagyon sok esetben akár a processzoron belül is találunk olyan változókat, amelyek ezt a két értéket vehetik fel, illetve egy kapcsoló, vagy LED (VILÁGÍTÓ DIÓDA) is kétállapotú.

A következő alapegység a 4 bit, amely vagy hexadecimális számok ábrázolását teszi lehetővé, vagy a BCD ábrázolásánál játszik szerepet (lásd előző, 2.1.4. fejezet). Első esetben összesen 16 számjegy, második esetben 10 számjegy ábrázolható.

A következő alapegység a bájt (byte) mely 8 bitből áll, egységesen kell kezelni, de sokszor a bájt 2 hexadecimális számból, vagy két BCD számból áll, illetve vannak olyan esetek, amikor egy bájton belül a biteket egyenként kell tudni lekérdezni, vagy állítani (írni, törölni, komplementálni). Maga a bájt 8 bit, de a benne levő tartalom több minden lehet, így szám (tizes számrendszerbeli érték binárisan ábrázolva), nyomtatandó karakter (betű, szám és írásjel - pl. ASCII kódolás szerint – 0.3. MELLÉKLET), de lehet egy kép-, vagy hangfelvétel része is.

Ha a bájt számok ábrázolására szolgál, akkor is több értelmezést köthetünk hozzá, vegyük ezeket sorra:

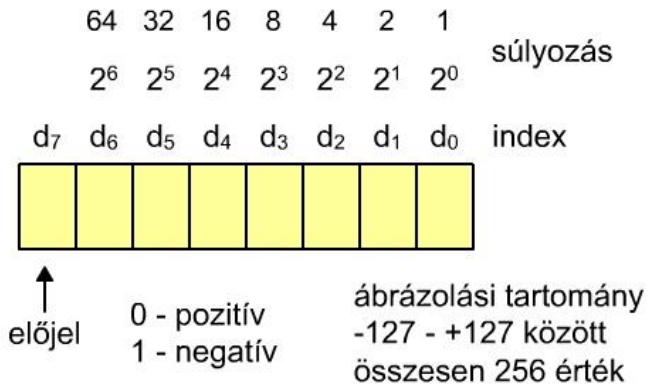
2.1.5.1. EGÉSZ POZITÍV SZÁMOK ÁBRÁZOLÁSA



2.3. ábra. A bájt mérete és az egyes bitek jelölése, súlya.

A d jelölés a data (adat) szó első betűje, míg az index jobbról balra 0-tól, 7-ig. Az index értékének és sorrendjének megválasztása nem véletlen, így kapjuk meg a természetes bináris kódot, ahol is a 2 hatványa az index értéken adja a szám súlyozott helyiértékét. A legkisebb helyi érték jobb oldalon 1, balra haladva előző érték kétszerese, és így tovább. Az adott helyiértéknél levő 1 (egy) jelzi, hogy figyelembe kell venni az érték kiszámításánál a bitet, a 0 (nulla), hogy nem. Ezek szerint egy bájt, pozitív egész számként 0 és 255 között összesen 256 különböző értéket különböztet meg.

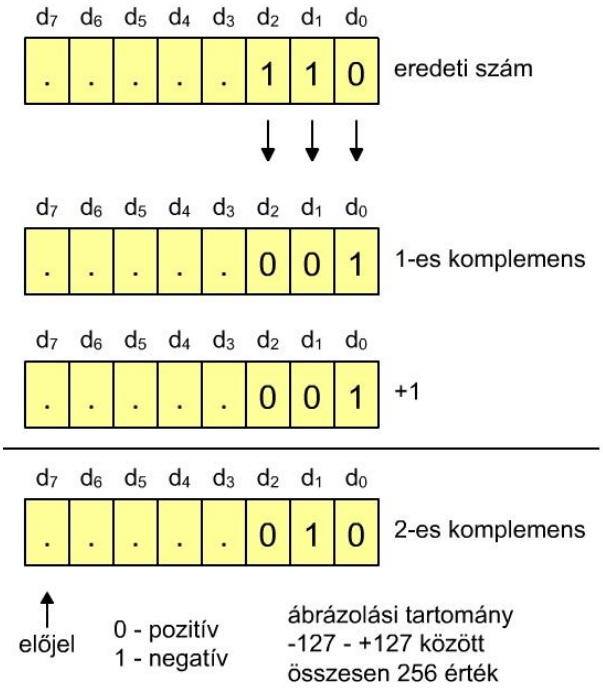
2.1.5.2. EGÉSZ POZITÍV ÉS NEGATÍV SZÁMOK ÁBRÁZOLÁSA



2.4. ábra. Pozitív és negatív egész számok ábrázolása.

A bájtban a bit csak 0 és 1 értékű lehet, ezért a pozitív és negatív érték megkülönböztetése is csak így lehetséges. Megegyezés szerint a 0 a "+", pozitív előjelnek, míg az 1 a "-", mínusz előjelnek felel meg. ekkor csak 7 bit marad a szám bináris ábrázolására, az ábra szerinti tartományban. Meg kell jegyezni, hogy így tulajdonképpen van egy +0 és egy -0, ami ugyanaz, tehát csak 255 értékes szám létezik.

2.1.5.3. KETTES KOMPLEMENSŰ SZÁMÁBRÁZOLÁS

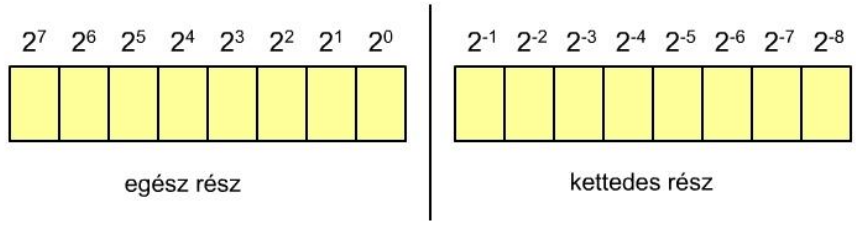


2.5. ábra. Egy bináris szám és átalakítása kettes komplemensű bináris számmá.

Gyakorlatilag ez az ábrázolás a legkényelmesebb, hiszen ilyenkor a számítógép összeadó egysége közvetlenül tud az ábrázolt számokkal (negatív és pozitív) számolni, vagyis a hardver az összeadás mellett a kivonást is el tudja végezni, mint összeadást. (lásd később az aritmetikai-logikai egység - ALU – tárgyalásánál, 4.4. fejezet). A szorzáshoz sem kell az ismételt összeadáson kívül semmilyen más műveletet végrehajtani, ugyanúgy az osztás is ismételt kivonásra (az imént láttuk, hogy ez is összeadás) vezethető vissza.

A kettes komplemens meghatározása igen egyszerű, venni kell a szám egyes komplemensét, ami nem más, mint minden egyes bit komplementálása (0-ból 1, 1-ből 0) és az így kapott értékhez hozzá kell adni egyet. Ekkor is megmarad a 256 különböző szám, -128 és +127 között.

2.1.5.4. FIXPONTOS SZÁMÁBRÁZOLÁS (TÖRT-, VAGY TIZEDES SZÁMOK ÁBRÁZOLÁSA)



Kettedes pont, elképzelt választóvonal az egész- és a törtrész között

2.6. ábra. 16 bites fixpontos számábrázolás.

A regiszterekben ugyanúgy ábrázoljuk a tizedes számokat, mint az egész számokat, de most egy elképzelt helyen, két bit között lesz a választóvonal az egész és a tört rész között, ez a kettedes pont. Ettől az elképzelt vonaltól balra egész számok, jobbra a 2 negatív hatványai, 0.5, 0.25, 0.125 stb. találhatók (tehát ½, ¼, stb.) Most az egész részen, a 8 biten ugyanúgy 256 különböző érték van, a tört rész 8 bites, a legkisebb felbontás $1/2^{-8}$. Ez a technika a természetes bináris számoknál mind az egész számú számábrázolásra, mind a negatív számok ábrázolására érvényes (kettes komplemensre is). Leggyakrabban 16, vagy 32 bites ábrázolást használunk.

A gyakorlatban a számok ábrázolásánál két dologra kell figyelnünk:

- a számok nagyságára és
- az ábrázolt szám pontosságára.

A fenti két feltétel kielégítése a regisztermére helyes meghatározásától és a kettedes pont helyétől függ. A kettedes (bináris pont) mozgatásának következményei balra tolás esetén:

- csökken a számábrázolási tartomány,
- nő az ábrázolás pontossága és
- ha a kettedes pont a regiszter bal szélére kerül, akkor a szám egy fixpontos törtszám.

A kettedes (bináris pont) mozgatásának következményei jobbra tolás esetén:

- nő a számábrázolási tartomány,
- csökken az ábrázolás pontossága és
- ha a kettedes pont a regiszter jobb szélére kerül, akkor a szám egy fixpontos egészszám.

2.1.5.5. FESZÍTETT-, ELTOLÁSOS-, VAGY ALAPÉRTÉKES SZÁMÁBRÁZOLÁS

Lényege az ilyen számábrázolásnak, hogy egy előre rögzített konstanst adunk a negatív számhoz, ekkor a legkisebb ábrázolható szám –konstans lehet. Az eredmény mindenféleképpen 0 vagy pozitív szám. Használata előnyös a lebegőpontos számábrázolás kitevőjénél.

2.1.5.6. LEBEGŐPONTOS SZÁMÁBRÁZOLÁS

Valós számot ábrázolhatunk a következő alakban:

$3 \cdot 10^7$.

Az ilyen számábrázolást a valós számok normálalakjának hívjuk. Itt 3 rész van:

- szorzó (a példában 3),
- a hatványalap (a példában 10) és
- a kitevő (a példában 7).

A feladat az, hogy az adott számot $r \cdot 2^m$ bináris alakra írjuk át. Először is r szorzót normalizálni kell, vagyis 0 és 1 közé kell hozni. Matematikailag, $0 \leq r < 1$ és m egész szám.

Ha most 32 bites lebegőpontos számból indulunk ki, akkor egy előjelbit van, a kitevő m 8 bit, míg 23 bit lesz a valós szám szorzója. Egy dolgot még figyelembe kell venni, ez pedig az, hogy a kettes számrendszerben r-nek $1/2 \leq r < 1$ közé kell esni, de ilyenkor r mindig egy, amit el szoktunk hagyni. Ennek az 1-esnek a neve implicitbit.

A fent ismertetett módszertől sokszor eltérnek, más számú bit szolgál az egyes értékek ábrázolására.

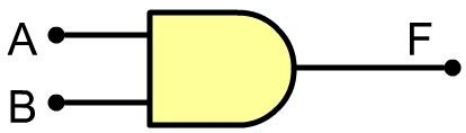
2.2. DIGITÁLIS TECHNIKAI ALAPISMERETEK

A Boole-algebra törvényein alapuló digitális technika ismereteket csak annyiban érintjük, hogy az architektúra alapismereteket megértsük. Két változó között összesen 16 különböző függvény határozható meg, ezek között három segítségével az összes többi kifejezhető. Ezek a következők:

- ÉS függvény, másképpen logikai szorzás, konjunkció (AND),
- VAGY függvény, másképpen logikai összeadás, diszjunkció (OR) és
- INVERTER, NEM kapu (NOT)

2.2.1. ÉS KAPU (AND)

Két változó közötti ÉS kapcsolatot. Rajza a 2.7. ábrán látható, míg az igazságtáblázat a 2.8. ábrán látható.



2.7. ábra. 2 bemenetű logikai ÉS kapu.

	A	B	F
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

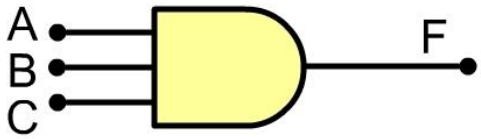
2 bemenetű

ÉS kapu

2.8. ábra. 2 bemenetű ÉS kapu igazságtáblázata.

Az igazságtáblázatban A és B a független változó, bemenet, F jelöli a kimenetet. A sorszám növekvő sorrendet mutat, értéke az egyes független változóknál használt index alapján határozható meg.

Három változó közötti ÉS kapcsolatot. Rajza a 2.9. ábrán látható, míg az igazságtáblázat a 2.10. ábrán látható.



2.9. ábra. 3 bemenetű logikai ÉS kapu.

	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

3 bemenetű

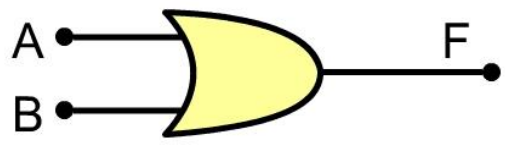
ÉS kapu

2.10. ábra. 3 bemenetű ÉS kapu igazságtáblázata.

Látható, hogy az ÉS kapu kimenetén csak akkor jelenik meg logikai 1-es, ha az összes bemenetén egyidejűleg 1-es van.

2.2.2. VAGY KAPU (OR)

Két változó közötti VAGY kapcsolatot. Rajza a 2.11. ábrán látható, míg az igazságtáblázat a 2.12. ábrán látható.



2.11. ábra. 2 bemenetű logikai VAGY kapu.

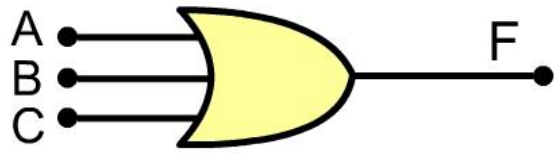
	A	B	F
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

2 bemenetű

VAGY kapu

2.12. ábra. 2 bemenetű VAGY kapu igazságtáblázata.

Három változó közötti VAGY kapcsolatot. Rajza a 2.13. ábrán látható, míg az igazságtáblázat a 2.14. ábrán látható.



2.13. ábra. 3 bemenetű logikai VAGY kapu.

	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

3 bemenetű

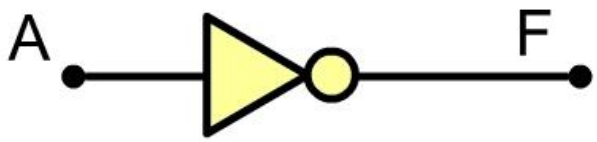
VAGY kapu

2.14. ábra. 3 bemenetű VAGY kapu igazságtáblázata.

Látható, hogy a VAGY kapu kimenetén akkor jelenik meg logiai 1-es, ha legalább egy logikai változó 1-es.

2.2.3. INVERTER, NEM KAPU (NOT)

Egy bemenetű kapu, a bemenet logikai értékét megfordítja. Rajzjele a 2.15. ábrán, az igazságtáblázata pedig a 2.16. ábrán látható.



2.15. ábra. INVERTER rajzjele.

	A	F
0	0	1
1	1	0

Negálás

(invertálás)

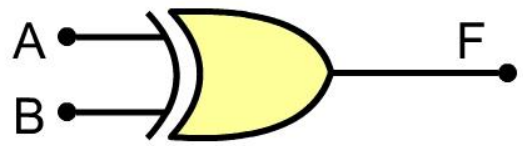
2.16. ábra. Az INVERTER igazságtáblázata.

2.2.4. KIZÁRÓ VAGY kapu (EXCLUSIVE OR)

Ez a kapu hardverben gyakran használt elem, az előző pontokban ismertetett ÉS, VAGY és INVERTER segítségével is felépíthető. A kapu egyenlete az előző kapuk függvényeivel kifejezve:

$$A \oplus B = \bar{A} \cdot B + A \cdot \bar{B} .$$

A kapu szimbolikus rajza az 2.17. ábrán, míg igazságtáblázata a 2.18. ábrán látható.



2.17. ábra. A KIZÁRÓ-VAGY kapu rajzjele.

	A	B	F
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

Kizáró

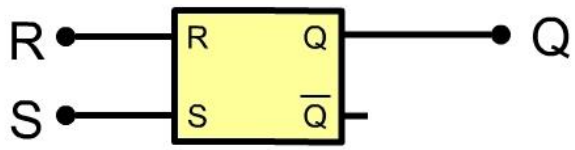
VAGY kapu

2.18. ábra. A KIZÁRÓ-VAGY kapu igazságtáblázata.

Azonos bemeneti értékek mellét a kimeneten 0 érték, míg különböző bemeneti értékek mellett logikai 1 jelenik meg.

2.2.5. RS FLIP-FLOP (MEMÓRIACELLA)

Az RS flip-flop, vagy másképpen RS tároló már az úgynevezett sorrendi hálózati elemek közé tartozik, a kimenet (Q) állapot nemcsak az S (set) - író- és R (reset) törlő jel értékétől, hanem a kapu előző állapotától is függ (visszacsatolás). Szerepe főleg a számlálók és regiszterek megépítésénél van. Több regiszterből pedig memóriát építhetünk. Az elem szimbolikus rajza a 2.19. ábrán, míg az elem átmeneti táblája az 2.20. ábrán látható.



2.19.. ábra. RS tároló szimbolikus ábrázolása.

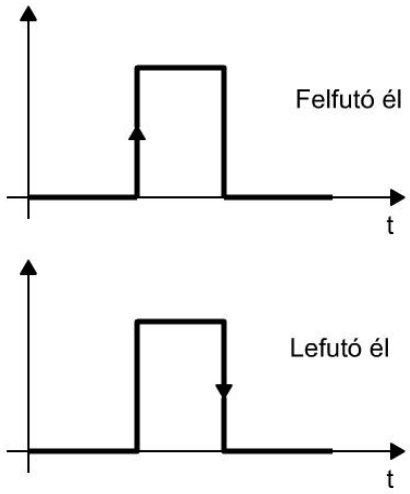
		jelenlegi állapot	következő állapot	leírás
R	S	Q_t	Q_{t+1}	
0	0	0	0	HOLD
0	0	1	1	HOLD
0	1	0	1	SET
0	1	1	1	SET
1	0	0	0	RESET
1	0	1	0	RESET
1	1	0	X	tiltott
1	1	1	X	tiltott

2.20. ábra. RS tároló átmeneti táblája.

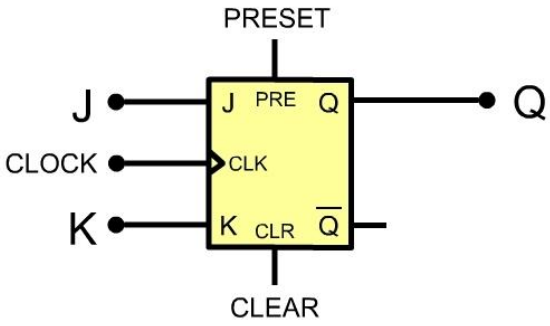
A szimbolikus rajzból és a táblából elemezhető működése. Ez most egy úgynevezett aszinkron megoldás, órajel nélkül működik, a Q kimenet következő állapota R és S bemenetektől és Q kimenet mostani állapotától függ. Az elemnek van órajellel vezérelt változata is. Ha R is és S is 0, nincs változás, a kapu tartja a mostani állapotát (HOLD). R = 0 és S = 1 esetén a Q kimenet mostani állapotától függetlenül Q következő állapota 1 lesz, ezt SET műveletnek hívjuk (írás). R = 1 és S = 0 esetén a Q kimenet mostani állapotától függetlenül Q következő állapota 0 lesz, ezt RESET műveletnek hívjuk (törlés). Az R = S = 1 állapot a bemeneteken nem megengedett, az áramkör ilyenkor ellenőrizetlenül oszcillálna.

2.2.6. JK FLIP-FLOP (MEMÓRIACELLA)

A JK flip-flop, vagy másképpen JK tároló is az úgynevezett sorrendi hálózati elemek közé tartozik, a kimenet (Q) állapot nemcsak a J (set) - író- és K (reset) - törlő - jel értékétől, hanem a kapu előző állapotától is függ (visszacsatolás). Tartalmaz még egy órajelet is az áramkör, amely órajel periodikus, vagy aperiodikus négyszög-jelekből áll. Az áramkör az órej fel-, vagy lefutó élének megjelenésekor vált állapotot. Gyakrabban lefutóél-vezéreltek a sorrendi hálózatok (2.21. ábra). Szerepe a kapunak a számlálók és regiszterek megépítésénél van. Az elem szimbolikus rajza az 2.22. ábrán, míg az elem átmeneti táblája az 2.23. ábrán látható. Az ábrán szereplő kapu CLEAR vezérlőjele az elem bekapcsolásakor a Q kimeneten 0-, míg a PRESET lábon levő jel 1 állapotot állít be.



2.21. ábra. Órajelek (lefutó- és felfutó él).



2.22. ábra. JK tároló szimbolikus ábrázolása

		jelenlegi állapot	következő állapot	leírás
K	J	Q_t	Q_{t+1}	
0	0	0	0	HOLD
0	0	1	1	HOLD
0	1	0	1	SET
0	1	1	1	SET
1	0	0	0	RESET
1	0	1	0	RESET
1	1	0	1	TOGGLE
1	1	1	0	TOGGLE

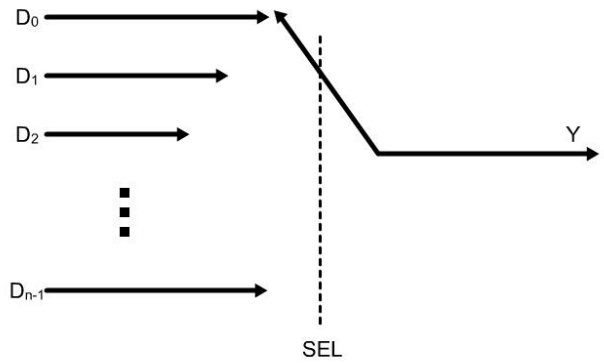
2.23. ábra. JK tároló átmeneti táblája

A szimbolikus rajzból és a táblából elemezhető működése. Ez most egy úgynevezett szinkron megoldás, órajellel működik, a Q kimenet következő állapota J és K bemenetek állapotától és Q kimenet mostani állapotától függ. Ha J is és K is 0, nincs változás, a kapu tartja a mostani állapotát (HOLD). J = 0 és K = 1 esetén a Q kimenet mostani állapotától függetlenül Q következő állapota 1 lesz, ezt SET műveletnek hívjuk (írás). Ez az állapot csak az órajel lefutó élének megjelenésekor történik meg (szinkronizálás). J = 0 és K = 1 esetén a Q kimenet mostani állapotától függetlenül Q következő állapota 0 lesz, ezt RESET műveletnek hívjuk (törlés). Ez a változás is csak az órajel lefutó élének megjelenésekor történik meg. A J = K = 1 állapot a bemeneteken megengedett, az áramkör ilyenkor az oszcillátor frekvenciájának ütemében vált állapotot (0 → 1 és 1 → 0). Ha jobban megfigyeljük, akkor ez az elem ebben az esetben a kimenetén (Q) az órajel frekvenciájának a felével fog oszcillálni, vagyis a frekvenciaosztóként működik.

2.2.7. MULTIPLEXER - DEMULTIPLEXER DIGITÁLIS TÖBBÁLLÁSÚ KAPCSOLÓ

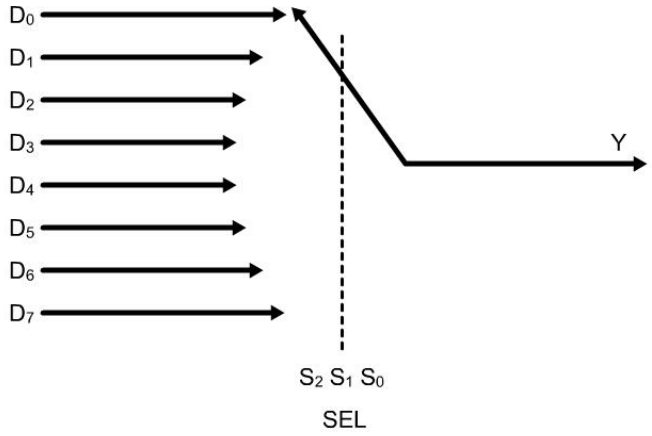
Ez a digitális elem mint egy fokozatkapcsoló működik, több forrásból érkező digitális jelet továbbít egy kimenő vezetéken. A forrás kiválasztása egy kombinációs hálózat segítségével

történik. A 2.24. ábrán egy n választóvezetékkel (SELECT) ellátott kapcsolás látható, ahol 2^n bemenet, 0 és 2^{n-1} között számozott.



2.24. ábra. 2^n bemenetű multiplexer.

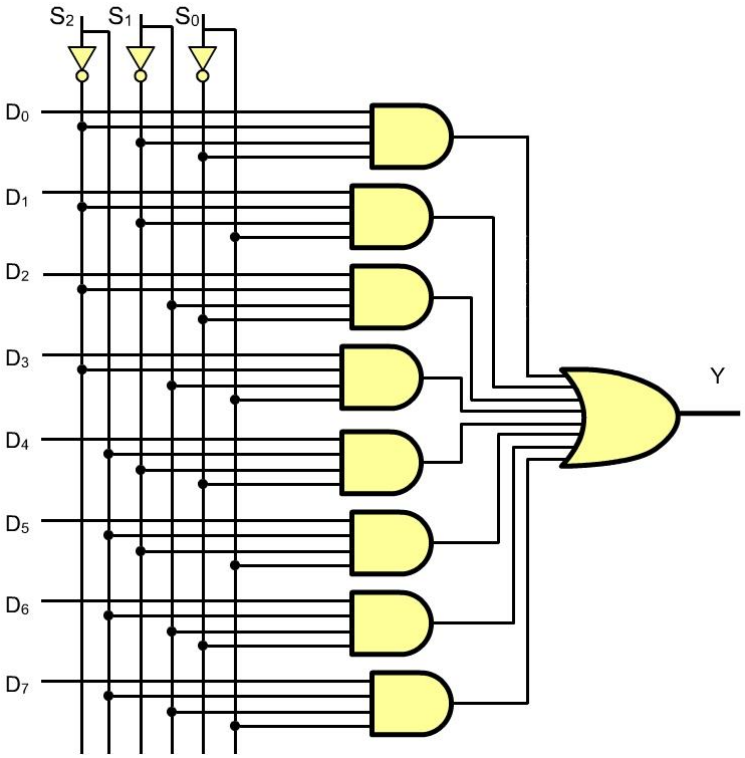
Amennyiben a választó vezetékek száma n = 3, akkor összesen 8 bemenetről jöhet jel, 0 és 7 között számozva (2.25. ábra). Az elem igazságtáblázata az 2.26. ábrán látható, a kapcsolás kapukkal pedig az 2.27. ábrán van.



2.25. ábra. n = 3 választóvezetékes multiplexer

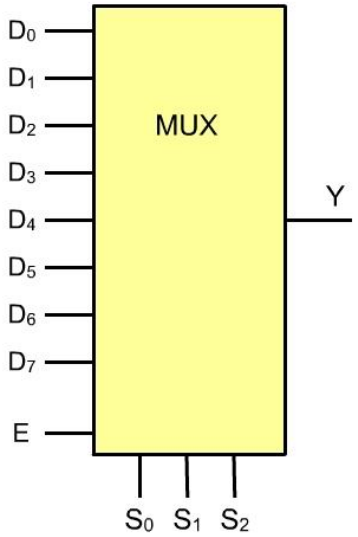
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

2.26. ábra. n = 3 választóvezetékes multiplexer



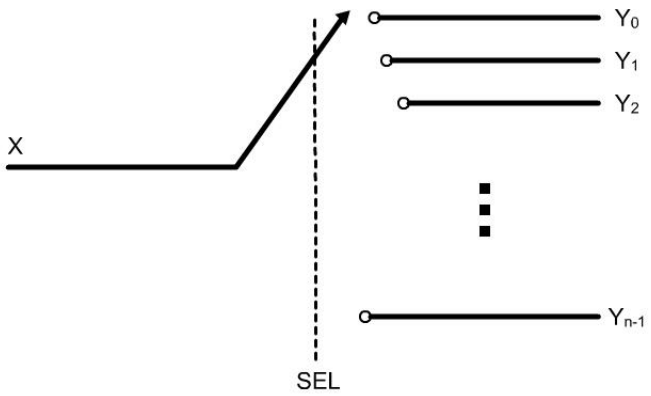
2.27. ábra. 8 bemenetű multiplexer felépítése logikai kapukkal.

A 2.28. ábrán az áramkör szimbolikus ábrája látható. E bemenet egy engedélyező (ENABLE) jel, aminek segítségével lehet a bemenet jelét a kimenetre (Y) továbbítani, vagy tiltani.

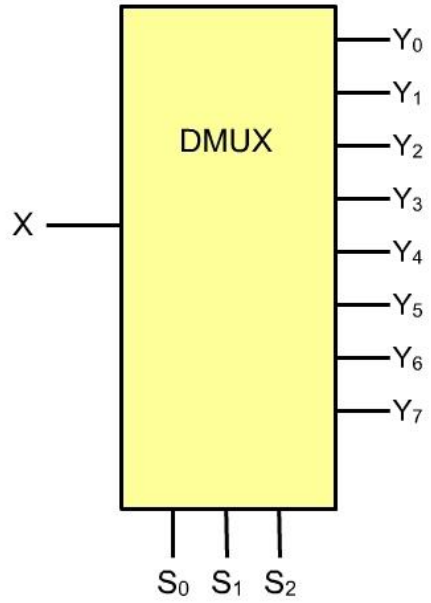


2.28. ábra. 8 bemenetű multiplexer rajzjele.

A demultiplexer fordított elrendezésű elem, mint a multiplexer, egy bemenetre érkező jelet oszt szét adott számú kimenetre (2.29. ábra és 2.30. ábra).

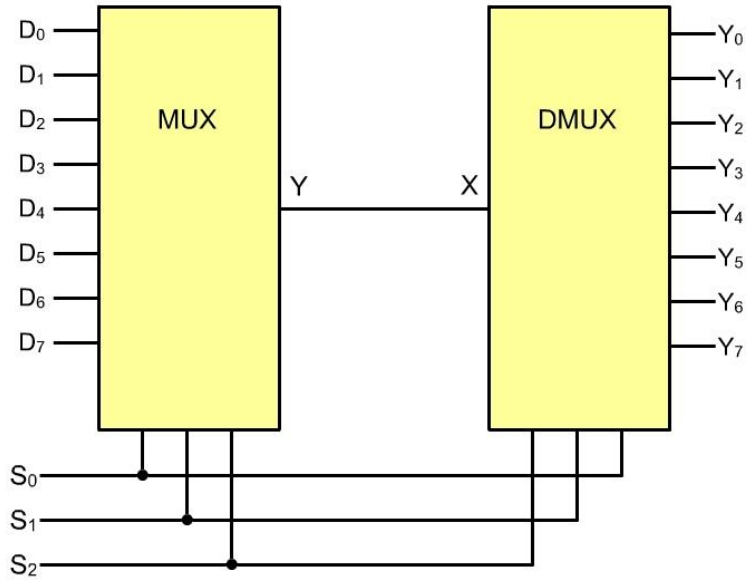


2.29. ábra. Demultiplexer elvi kapcsolása.



2.30. ábra. Demultiplexer szimbolikus rajzjele.

A két elemet megfelelően összekötve (2.31. ábra) egy olyan kapcsoláshoz jutunk, amelyikkel 2^n helyről 2^n helyre juttatunk jeleket, egyetlen vezetéken keresztül. Természetesen mindkét elem ugyanarra a SELECT választójelre válaszol.



2.31. ábra. Multiplexer és demultiplexer összekapcsolása.

2.2.8. DEKÓDOLÓ ÁRAMKÖRÖK

A dekódoló áramkörök több bemenetű és több kimenetű áramkörök. Ezekben az áramkörökben minden megengedett bemeneti kombináció más és más kimenetet kapcsol be. A dekódolók lehetnek:

- teljes dekódolók - **n** bemenethez 2^n kimenő függvény tartozik és
- nem teljes dekódolók - **n** bemenethez kevesebb mint 2^n kimeneő függvény tartozik, a bemenetek nem engedélyezik a lehetséges összes kombinációt (tiltott kombinációk).

A dekódoló olyan áramkör, amelyik a következő logikai függvényeket valósítja meg:

$$\begin{aligned} D_0 &= \overline{X_1} \cdot \overline{X_2} \cdot .....\overline{X_n} \cdot E, \\ D_1 &= \overline{X_1} \cdot \overline{X_2} \cdot .....X_n \cdot E, \\ &. \\ &. \\ D_{2^n-1} &= X_1 \cdot X_2 \cdot ...X_n \cdot E \end{aligned} \qquad (2.1.)$$

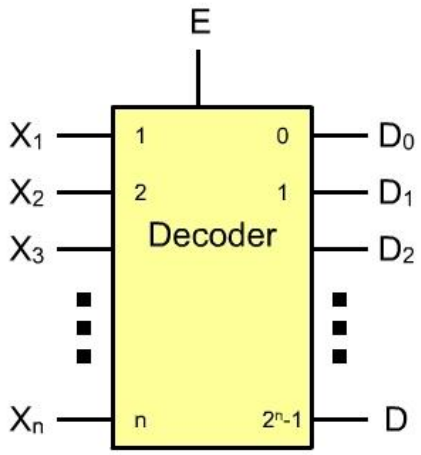
jelölések:

$X_1, X_2, ...X_n$ - bemenetek,

E - engedélyező jel (ENABLE),

$D_0, D_1, ...D_{2^n-1}$ - kimenetek.

Az E engedélyező jel ha nulla (E = 0), akkor minden kimenet tiltott állapotban van, vagyis $D_i = 0$, amennyiben E jel értéke egyes (E = 1), akkor az összes D_i kimenet közül a bemeneti kombinációtól függően csak egy, a bemeneti kombinációnak megfelelő kimenet értéke 1. Azt, hogy melyik ez a kimenet, X_i bemenetek kombinációja szabja meg, hiszen ez n változó teljes szorzatát jelenti. A dekódoló áramkör szimbóluma az 2.32. ábrán látható.



2.32. ábra: a dekódoló szimbólikus rajza

Példa:

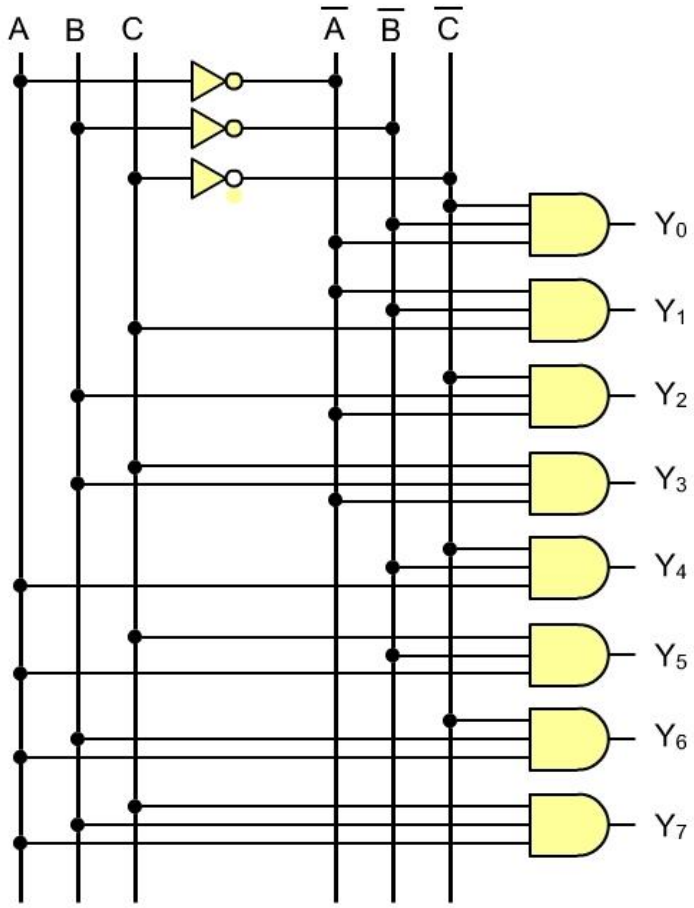
Tervezzünk egy **n=3** bemenettel rendelkező dekódoló áramkört, ahol a bemeneteket **A**, **B** és **C** betűvel jelöljük.

Ha a bemenetek száma n=3, akkor a bemeneten összesen $2^3 = 8$ kombináció jöhet létre. Ezek után kitölthetjük a dekódoló áramkör igazságtáblázatát (2.33. ábra).

s.sz.	A	B	C	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0.	0	0	0	0	0	0	0	0	0	0	1
1.	0	0	1	0	0	0	0	0	0	1	0
2.	0	1	0	0	0	0	0	0	1	0	0
3.	0	1	1	0	0	0	0	1	0	0	0
4.	1	0	0	0	0	0	1	0	0	0	0
5.	1	0	1	0	0	1	0	0	0	0	0
6.	1	1	0	0	1	0	0	0	0	0	0
7.	1	1	1	1	0	0	0	0	0	0	0

2.33. ábra: 3 bemenetű dekódoló áramkör igazság táblázata

Az igazságtáblázatból az derül ki, hogy a dekódoló áramkör minden kimenetét egy-egy logikai szorzat valósítja meg, így minden kimenet ÉS kapuval és inverterrel valósítható meg, ugyanakkor az is látszik, hogy a hálózat nem egyszerűsíthető. A dekódoló áramkör kapcsolási rajza az 2.34. ábrán látható.



2.34. ábra: A bináris dekódoló áramkör kapcsolási rajza

A dekódoló egyéb elnevezései:

- 3/8 dekódoló vagy
- 1 a 8-ból dekódoló.

3. A SZÁMÍTÓGÉPEK OSZTÁLYOZÁSA, FELÉPÍTÉSE

Számítógépekkel egymástól nagyon különböző feladatokat oldunk meg, de ennek ellenére a számítógépek felépítésében és működésében hasonló, illetve megegyező elemeket találunk. Egy asztali számítógép, egy okostelefon, vagy egy ipari szabályozó azonos, vagy hasonló felépítésű.

Mai modern számítógépeink az úgynevezett INPUT – PROCESS –OUTPUT (beolvasás – feldolgozás – kiírás) elven működnek.

3.1. ábra. Az INPUT – PROCESS – OUTPUT modell.

Az INPUT (adatbeolvasás) adatbevitelt jelent, ennek nagyon sok módja van:

- billentyűzet,
- lapolvasó,
- mikrofon,
- kamera,
- hőmérséklet,
- nyomás,
- stb.

Mindamellett, hogy ezek a módszerek nagyon különböznek egymástól, műszakilag hasonló megoldásokról van szó, a legutolsó fázisban bináris, 0 és 1 értékkel történik az adatok bevitele, majd ábrázolása.

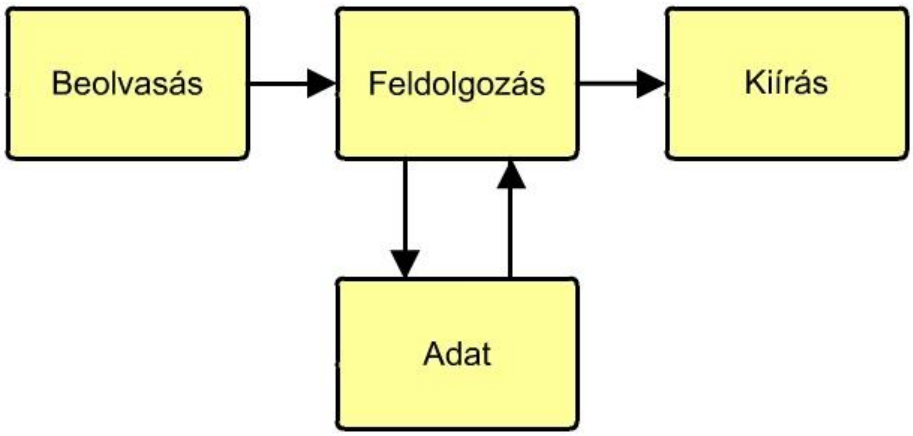
A PROCESS, vagy adatfeldolgozás a beolvasott, és/vagy már tárolt adatokon való valamilyen műveletsor végrehajtását jelenti, ami nagyon sokféle módszer alkalmazását jelenti. Így lehet egyszerűbb, vagy bonyolultabb matematikai eljárás, táblázatban adatok keresése, vagy oda beírása, adatok konvertálása egyik rendszerből a másikba, stb. Ezeket a lépéseket, feladatokat a számítógép processzora hajtja végre azokon az adatokon, amelyek már a főtárba kerültek. A feldolgozásnál sokszor előfordul egymás utáni lépések sorozata, párhuzamosan, egyszerre végrehajtott lépések és az előzőek kombinációja. Ezeket a feladatokat a hardver és a rajta futó szoftver együttesen hajtja végre.

AZ OUTPUT, az adatkiírás történhet tárolóba, illetve a felhasználó felé, megfelelő formában:

- nyomtató,
- képernyő,
- hangszóró,
- motor bekapcsolása,
- szelep elfordítása,

- stb.

Tekintettel arra, hogy adatokat sokszor tartósan, vagy ideiglenesen tárolni kell, szükséges háttértárak alkalmazása is, így az 3.2. ábra tárolókkal bővítve a következő lesz:



3.2. ábra. Háttértárral bővített IPO (INPUT – PROCESS – OUTPUT) modell.

A háttértárakat még másodlagos táraknak is nevezik, míg a főtár más elnevezése az elsődleges tár. Az adatfeldolgozás során jelentkező fenti három lépés tovább bontható kisebb, önmagukban értelmes részeke, ezek tovább bonthatók, míg nem jutunk el már tovább nem bontható lépésekig. Néhány ilyen lépés, kép megjelenítése képernyőn, adatátvitel számítógép és végrehajtóegység (pl. motor) között, keresés adatállományban stb. Ezeket az elemi részeket olyan bonyolultságig bontjuk le, míg nem keletkeznek számítógépes programozással megoldható feladatok. Tehát a program a felhasználó igényei szerint elemi lépésekkel megvalósított utasítássorozat. Később fogjuk látni, hogy a számítógép alapműveletei meglehetősen egyszerűek, de ezekkel az elemi lépésekkel nehézkes lenne bonyolult feladatokat megoldani, ezért magasszintű programozási nyelveket használunk, ahol az IPO modell egyes lépései nem részletezve, nagyobb, összetettebb feladatokat hajtanak végre.

3.1. A SZÁMÍTÓGÉP ÁLTALÁNOS FELÉPÍTÉSE

A fentieket figyelembe véve négy fő komponenst találhatunk minden számítógépben:

- hardver,
- szoftver,
- adattárolás
- kommunikáció.

A számítógép hardver fizikai elemek összessége, biztosítja az adatok bevitelét és tárolását, valamint az eredmények megjelenítését. Fontos, hogy a hardverelemek működését vezérlőelektronika hangolja össze.

A számítógép szoftver a hardver által értelmezhető utasítások (nullák és egyesek) összessége, amelyeket megfelelő sorrendben, a feladat által meghatározva válogatunk össze.

Az adattárolás különféle adatok (számok, betűk, hangok, képek stb.) ideiglenes illetve tartós adatok tárolása, olyan alakban, amelyet a számítógép és a perifériák is értelmezni tudnak, vagy 0 és 1 alakban.

Tekintettel arra, hogy a számítógép szakterület eléggé összetett, valamint arra, hogy igen intenzív a fejlődés, megjelennek különféle osztályozási rendszerek, megkísérlik a felépítést, működést, felhasználást valamilyen rendszer szerint osztályozni. Ez azt eredményezi, hogy átfedések is jelentkeznek ezek között. Példa erre a fenti négyes felosztás (hardver, szoftver, adattárolás, kommunikáció), hosszú ideig a kommunikáció része volt a három területnek, de mai, külön tárgyalását az indokolja, hogy ma már gyakorlatilag a térben elosztott rendszerek uralják mindennapjainkat.

Végül is, mit nevezünk számítógép architektúrának? Hagyományosan a hardver és a szoftver alkotja, mi is a jegyzet keretén belül így foglalkozunk vele, de nem szabad elfelejteni, hogy az adatok külső hordozón (vagy másként másodlagos hordozón) való tárolása, illetve a kommunikáció is szervesen kapcsolódik e két területhez.

3.1.1. A HARDVER

Az eredeti angol hardware kifejezés „kemény árút” jelent, a magyar nyelvben már átírásban szerepel, mint hardver. Minden számítógépben megfogható eszközt, alkatrészt jelöl. Ezek az eszközök igen nagymértékben különböznek egymástól, gondoljunk csak arra, hogy egy csatlakozó is és egy több milliárd elemet tartalmazó integrált áramkör is ide sorolható.

A bonyolult eszközök leírása, de tervezése is hierarchikus módon lehetséges, vagyis rétegeket definiálunk, a felsőbb réteg mindig az alsóbb réteg felhasználásával valósul meg. Az 3.3. ábrán egy lehetséges hierarchikus számítógép-modellt láthatunk.

3.3. ábra. Hierarchikus számítógép rétegmodell.

Az egyes szintek jelentése a következő:

ÁRAMKÖRI SZINT: mivel ma döntően elektronikus számítógépekről beszélünk, ezért az elektronika alkatelemei (ellenállás, kondenzátor, induktivitás, dióda, tranzisztor stb.) folytonos mennyiségekkel jellemezhetők, mint a feszültség, áram, ellenállás stb.

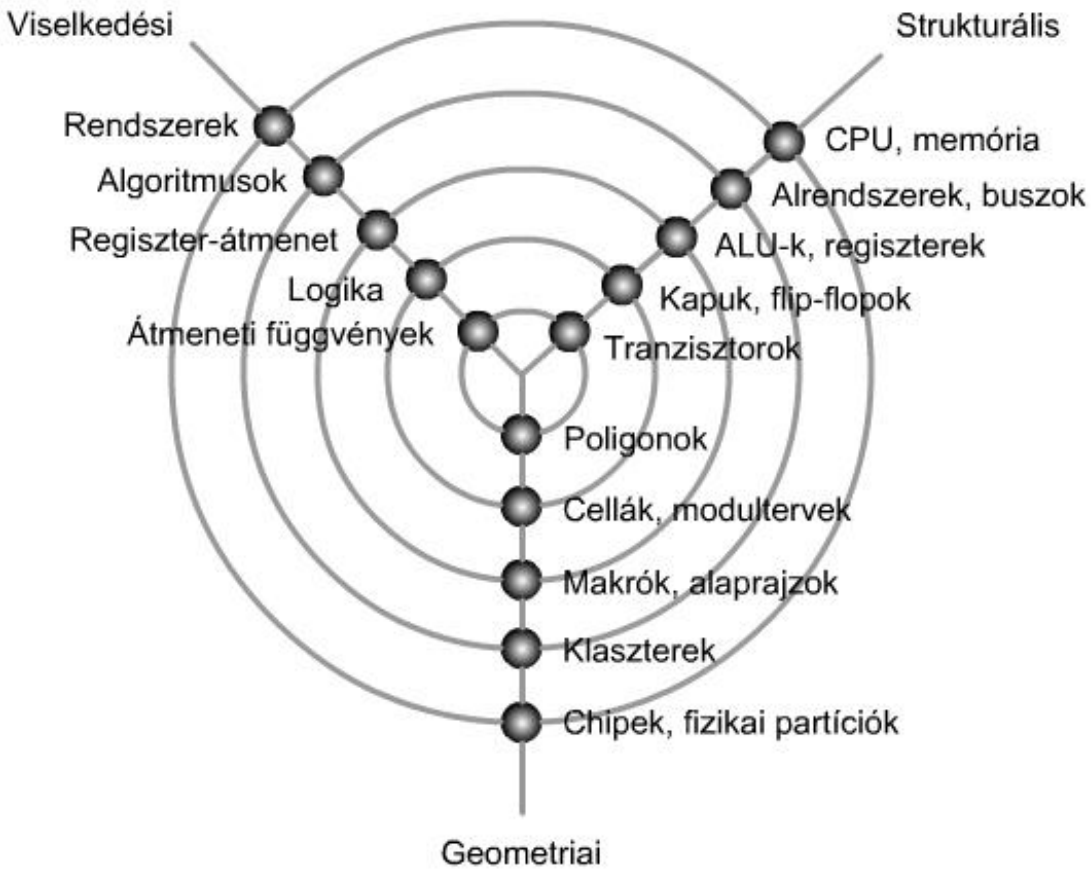
LOGIKAI SZINT: számítógépeink bináris (kétállapotú) logikai kapuelemekből épülnek fel, melyek működését és összekapcsolásukat a Boole-algebra szabályai szerint végezhetjük el. Ezen a szinten nem folytonos, hanem diszkrét jelekkel írjuk le a rendszert.

FUNKCIONÁLIS BLOKKOK SZINTJE: ezen a szinten hardver modulok szerepelnek, úgy mint regiszterek, számlálók, memóriák, összeadók stb. Lényeges a működésük is, de fontos az egymás közötti adatcsere is. Itt már időzítések, szinkronizálások is döntő módon kihatnak a működésre.

ALGORITMIKUS SZINT: a hardver modulok ezen a szinten már több funkcionális blokkból állnak, a blokkoknál megjelenő bitsorozatok jelentését algoritmikusan írjuk le.

RENDSZERSZINT: itt a részegységek jellemzése funkcionálisan (feladat) és protokolljaik leírásával történik.

A fenti modell egyes szintjei különböző bonyolultságú eszközökből állnak, legegyszerűbb az áramköri szint, legbonyolultabb természetesen a rendszerszint. A tantárgyon belül áramköri szinttel egyáltalán nem, logikai szinttel pedig csak igen korlátozott módon foglalkozunk.



3.4. ábra. Gajski-Kuhn Y számítógép rétegmodell.

A diagram kapcsolatot teremt a számítógép egyes részei és a következő szempontok között:

- Strukturális felépítés – a hardver elemek mi módon épülnek fel az alacsonyabb szinten levő elemekből,
- Viselkedési modell – az egyes rétegek elemeinek működési módja és
- Fizikai, geometriai jellemzők – az egyes szintek elemeinek formája, mérete.

3.1.2. SZÁMÍTÓGÉP-ARCHITEKTÚRÁK

Mint azt a 3.1. fejezetben láttuk, az architektúrát többféleképpen is lehet definiálni, mi a fogalom alatt a hardvert és a szoftvert értjük.

Maga a hardver fogalma több részből tevődik össze:

- a hardver funkcionális felépítése,
- az egyes elemek közötti kommunikációs kapcsolat és
- a rendszer specifikációja.

A számítógép-architektúrák állandó (tovább)fejlesztésének az az oka, hogy mindig újabb és újabb kihívások, igények jelennek meg, gondoljunk például a multimédia igényeire. Egyre nagyobb és nagyobb teljesítményű számítógépek előállítása válik lehetővé.

Tekintettel arra, hogy a bináris logikai elemek műszaki megvalósítása és működtetése során több nem-ideális jelenséggel is találkozunk, ezek miatt az egyes elemekben történő 0 → 1 és 1 → 0 átváltás nem azonnal, hanem késéssel történik meg, ráadásul minden elemnél ez az érték más és más, így az egyes elemek közötti szinkronizálást órajel felhasználásával biztosítják. Az órajel frekvenciájának növelésével növeljük az időegység alatt végrehajtott műveletek számát, de a jelenlegi elemméretek (főleg tranzisztorok, csíkszélesség, csíkszélesség távolság) parazita kapacitást, induktivitást, ellenállást, tranzisztoráram szivárgást, stb. okoznak, ezért ez ma már nem nagyon növelhető tovább. Ez látszik a bevezetőben található Moore-diagramon is (1.3.6.5. fejezet, 1.33. és 1.34. ábra).

3.1.3. A SZOFTVER

Természetesen a szoftver meghatározására is több definíció létezik. Egy egyszerű meghatározás az, ha arról beszélünk, hogy a szoftver meghatározza a hardverelemek működését, azok egymás utáni sorrendjét, amit utasítások megfelelő rendezésével (sorba rakásával) érünk el. Valójában ahhoz, hogy számítógépet kapjunk, olyan hardver-elemeket kell felhasználnunk és megfelelően összekötni, amelyeket 0 és 1 alakban kódolt és megfelelő sorrendbe összefűzött utasítások (a program) működtetnek.

A szoftver szolgáltatást nyújt, vagyis funkciója van, igaz hogy az egyes szoftver-részek között kapcsolat és átlapolódás van, de felosztható két nagy területre:

- operációs rendszer és
- felhasználói, vagy alkalmazói szoftver.

Az operációs rendszer egy rendszerszoftver, feladata az, hogy a számítógép felhasználók a számítógépet működtetni tudják, vagyis a hardver erőforrásokat megfelelő időben működésre bírják. Gyakorlatilag a legegyszerűbb számítógéptől kezdve a legbonyolultabbig mindegyik csak operációs rendszerrel tud működni (sokszor más neve is lehet, de a feladata mindig ugyanaz).

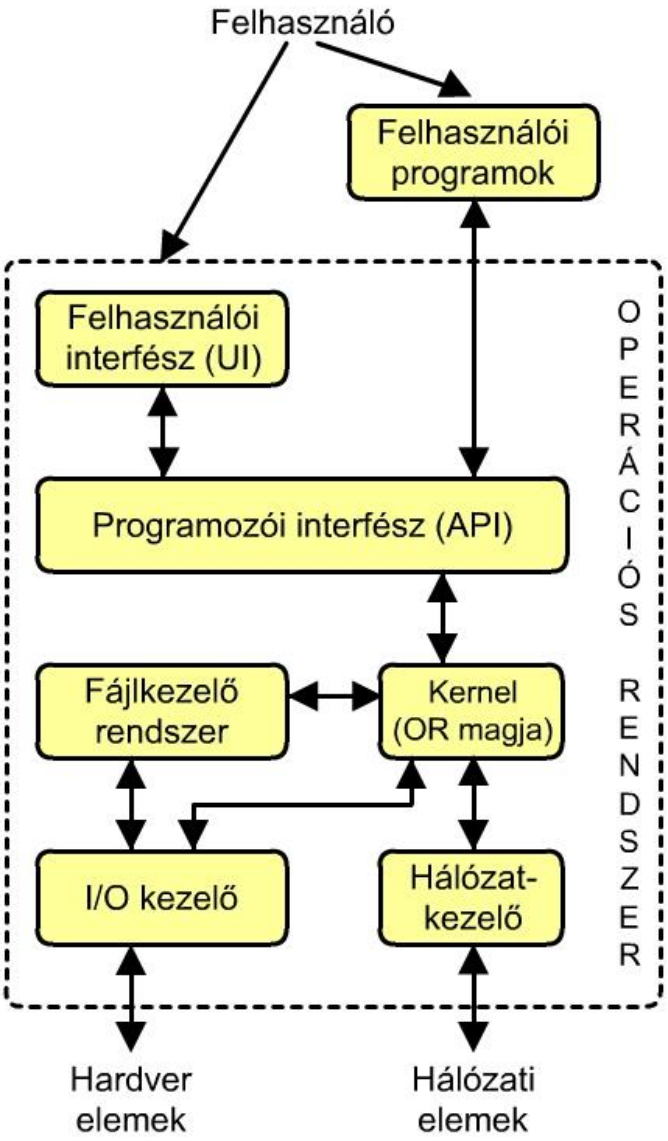
A felhasználói, vagy alkalmazói szoftver adott feladatkör problémáit oldja meg. Lehetnek széles körben használt kész szoftverek, mint pl. szövegszerkesztők, szimulációs programok, böngészők stb., vagy speciális feladatokat megoldó, egy-két felhasználó számára elérhető szoftverek. Természetesen mi is írhatunk felhasználói szoftvereket.

Az operációs rendszer és a felhasználói szoftver között nincs éles határ, sokszor feladatok a felhasználói felületről átkerülnek az operációs rendszerekbe.

Az operációs rendszer két irányban nyújt szolgáltatást, vezérli a hardver elemeit, valamint a felhasználó igényeit is ki kell hogy elégítse, így megkülönböztetünk:

- felhasználói interfészt, aminek a segítségével közvetlenül tudunk az operációs rendszertől szolgáltatást kérni, vagy parancssoros üzemmódban, vagy valamilyen grafikus felületen keresztül (ablakos operációs rendszerek),
- programozói interfészt, aminek a segítségével programokat futtathatunk, ahol a programok is elérhetik az operációs rendszer szolgáltatásain keresztül a megfelelő hardver elemeket.

Tehát az operációs rendszer szolgáltatást nyújt egyrészt az alkalmazások, másrészt a felhasználónak. A következő, 3.5. ábrán észre lehet venni, hogy mindkét esetben egy úgynevezett programozói interfészen keresztül történik mindez.



3.5. ábra: Az operációs- és felhasználói rendszer és azok kapcsolatai.

3.1.4. A SZÁMÍTÓGÉP TELJESÍTMÉNYE

Mint ahogyan azt az előző 3.1.1. HARDVER illetve 3.1.3. SZOFTVER fejezetben láttuk, a sok erőforrás (hardver elemek és szoftver elemek) együttes, jól megszervezett munkája csak jó szervezéssel, az operációs rendszerrel oldható meg. Egymástól különböző operációs rendszerek, vagy azok különböző változatai különféle módon oldják meg ezt a feladatot. Ebből következik, hogy ugyanaz a hardver más és más operációs rendszerrel más és más hatékonyságot mutathat.

A számítógép teljesítményének meghatározására a következő képlet használható:

$$F = C \cdot T \cdot U \qquad (3.1.)$$

Ahol:

F – egy feladat végrehajtási ideje, teljesítménye,

C – az utasítás ciklusszáma,

T – ciklusidő és

U – az utasítások száma az adott feladatnál.

Sok mindent elárul a képlet a teljesítmény növeléséről. C csökkentése az architektúra továbbfejlesztésével érhető el, az utasítás-végrehajtás párhuzamosításával. T csökkentése a frekvencia növelésével, de ezt az imént láttuk, már nem nagyon tudjuk tovább növelni. U csökkentése pedig szoftverkérdés, hatékony program írása és a fordítás optimalizálása.

A számítógépek teljesítménye ugyan leírható a fenti képlettel, de ez az egyes megoldások közötti (sokszor lényeges) eltérések miatt nem mindig tudunk valódi, értékelhető következtetéseket levonni. Ezért hoztak létre olyan mérőszámokat, amelyek utalnak valamilyen feladat végrehajtására.

Ezek a következők:

MIPS (Million Instructions per Seconds) – a másodpercenkénti utasítások száma,

MOPS (Million Operations per Seconds) - a másodpercenkénti műveletek száma,

MFLOPS (Millions of Floating Point Operations per Seconds) - a másodpercenkénti lebegőpontos műveletek száma.

A fenti mérőszámok a számítógép teljesítményéről ugyan adnak információt, de a számítógépeken futtatott különböző feladatok igen eltérő módon használják a hardvert, ezért szokták még használni a bizonyos feladatokhoz rendelt „Benchmark” teszteket. A mérnöki tudományoknál, számításoknál fellépő igények például a Whestone teszttel, míg a rendszerprogramozás a Dhrystone teszttel értékelhető ki.

3.1.5. SZÁMÍTÓGÉPEK OSZTÁLYOZÁSÁNAK, BESOROLÁSÁNAK SZEMPONTJAI

Tekintettel arra, hogy a hasonló, vagy kis mértékben egymástól eltérő hardverrel rendelkező számítógépek nagyon sokszor nagymértékben eltérő feladatokat oldanak meg, ezért a legkülönfélébb szempontok alapján sorolhatjuk be azokat csoportokba, a legkülönfélébb szempontok alapján osztályozhatjuk azokat. Néhány besorolási szempont:

- működési elv,
- a kiszolgált felhasználók száma, illetve a kiszolgálás időbeosztása,
- utasításkészlet tulajdonságai

- utasítás- és adatfolyam,
- számítási teljesítmény,
- stb.

Nézzünk meg néhány csoportosítási szempontot részletesebben.

3.1.5.1. SZÁMÍTÓGÉPEK OSZTÁLYOZÁSA MŰKÖDÉSI ELV SZERINT

Minden számítógépet együttesen határozza meg a hardver és a szoftver architektúrája. Egyik a másik nélkül nem létezhet. Igaz, hogy több elv is ismeretes ezen a szakterületen, de végül is mindegyik visszavezethető a Neumann-elvre.

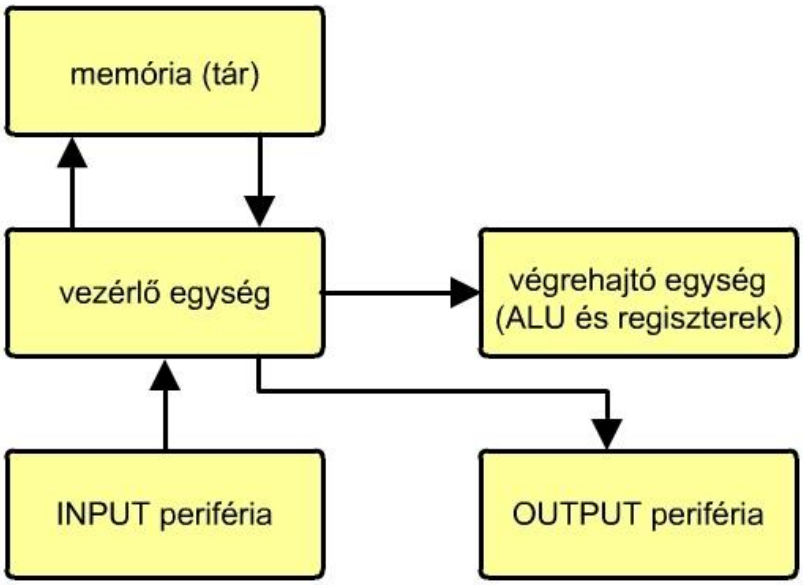
A Neumann modell tulajdonságai a következők:

- Mind az utasítások, mind az adatok ugyanolyan formában (0 és 1) alakban egy közös tárban helyezkednek el. A memória (tár, vagy elsődleges tár) lineárisan címezhető (egydimenziós), továbbá a korábbi értékek újakkal felülírhatók.
- A következő utasítás helyét a tárban a processzorban levő regiszter (utasításszámláló) jelöli ki. Nem tudható, hogy azon a helyen ténylegesen utasítás, vagy esetleg adat van, a gép mindenféleképpen utasításnak értelmezi a tartalmat.
- Az utasítások egymás utáni sorrendben hajtódnak végre (szekvenciális utasítás-végrehajtás), ez egy vezérlésáramlásos utasítás-végrehajtás.

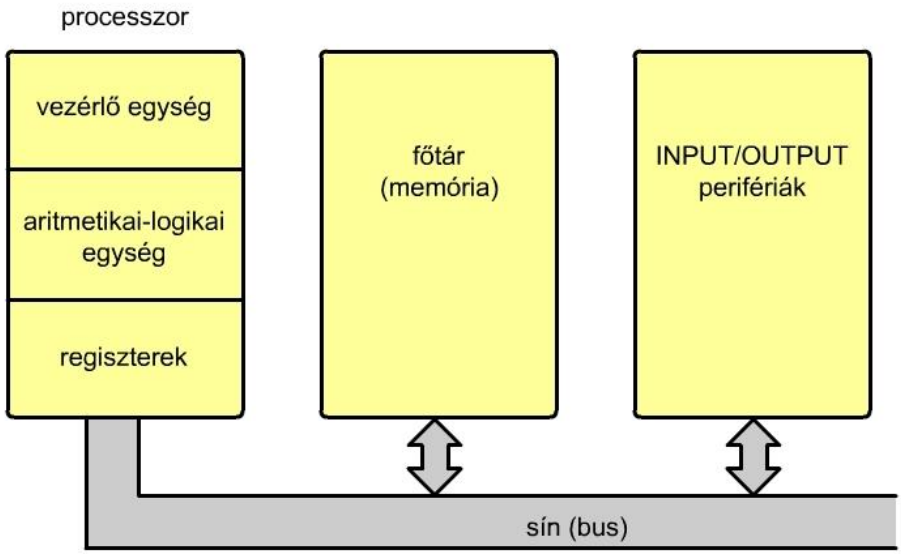
3.1.5.1.1. A SZÁMÍTÓGÉP NEUMANN MODELLJE

A Neumann elvű számítógép működése:

- 0 és 1 alakban a memóriarekeszek tartalmazzák a végrehajtandó gépikódú utasításokat, valamint a hozzájuk tartozó adatokat.
- A programszámláló tartalma alapján a processzor lehívja (kiolvassa) a memóriából az uatsítást, dekódolja és végrehajtja azt.
- Az aritmetikai-logikai egység (ALU – Aritmetic Logical Unit) végrehajtja a dekódolt utasítást.
- A keletkezett eredmény tárolódik, vagy a processzor valamelyik regiszterében, vagy a memória egyik rekeszében.
- Eggyel nő a programszámláló tartalma és a folyamat elölről kezdve ismétlődik.



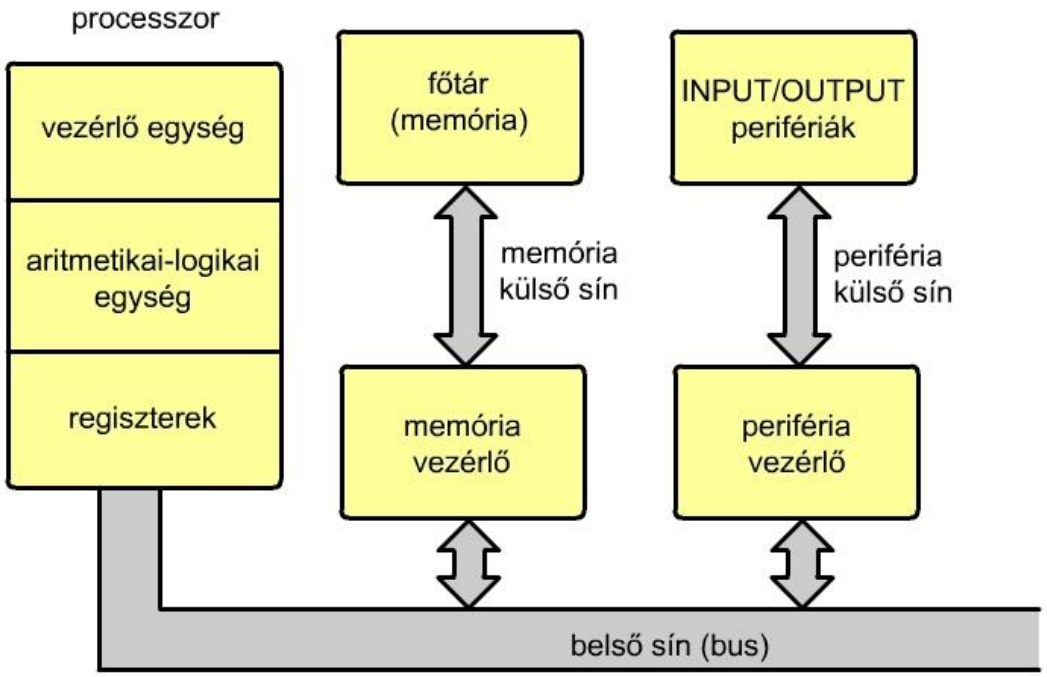
3.6. ábra. Eredeti Neumann-modell.



3.7. ábra. Módosított Neumann-modell.

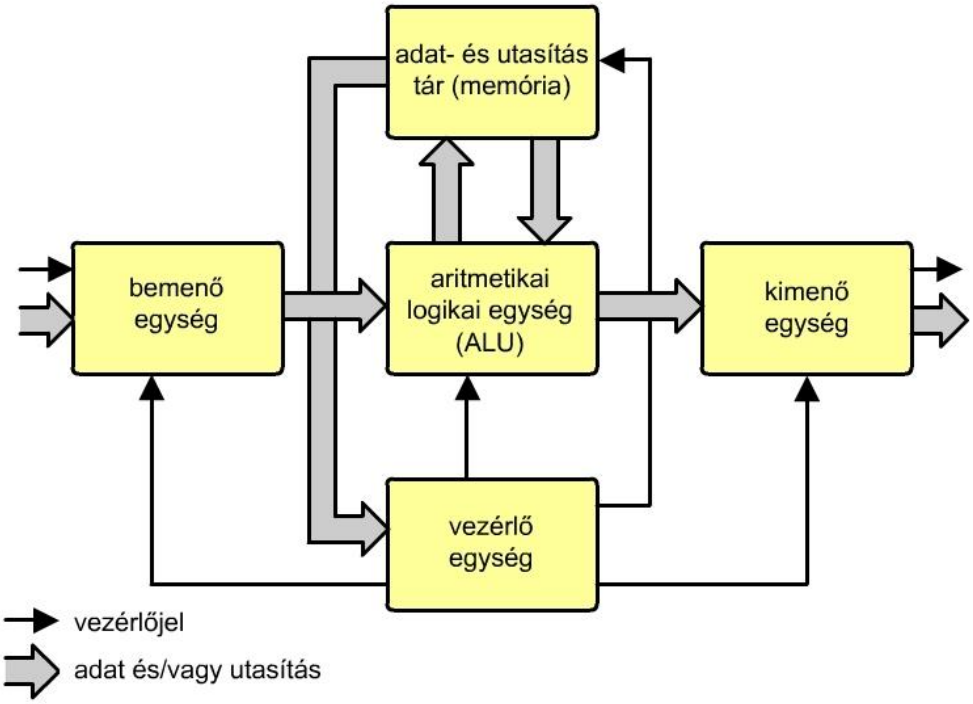
Az eredeti Neumann-modell a 3.6. ábrán, míg a valós modell a 3.7. ábrán látható. Ennek az architektúrának a fő jellegzetessége az, hogy a számítógép három fő egysége (processzor, főtár és perifériák) egy sínen (BUS) keresztül vannak egymással kapcsolatban. Ez az architektúra jellemzően az 1. és 2. generációs gépek megoldása. Mind a memória-, mind a perifériavezérlés összes lépését ilyenkor a processzor látja el.

A 3. generációtól kezdve a feladatok bizonyos része átkerült a memória- illetve perifériavezérlő egységekre, ezek lehetnek processzorok, vagy processzor bonyolultságú megoldások is. Egy ilyen megoldás látható az 3.8. ábrán. Itt a belső sin mellett megjelenik a külső sín is.



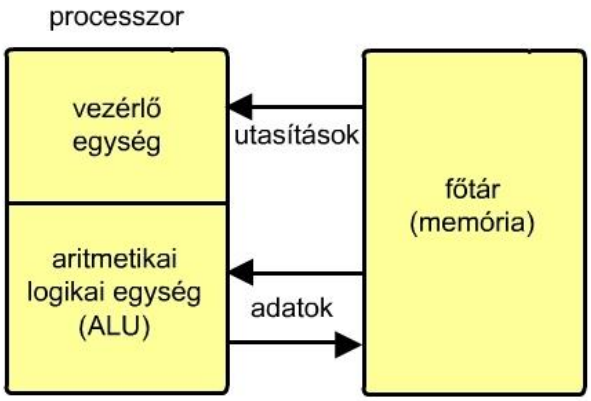
3.8. ábra. Memória- és perifériavezérlővel ellátott architektúra (külső sin).

A 3.9. ábrán egy egyszerűsített Neumann-modell látható, ahol jól látszanak az egyes építőelemek közötti kapcsolatok (adatáramlás és vezérlőjelek).



3.9. ábra. Egyszerű Neumann modell.

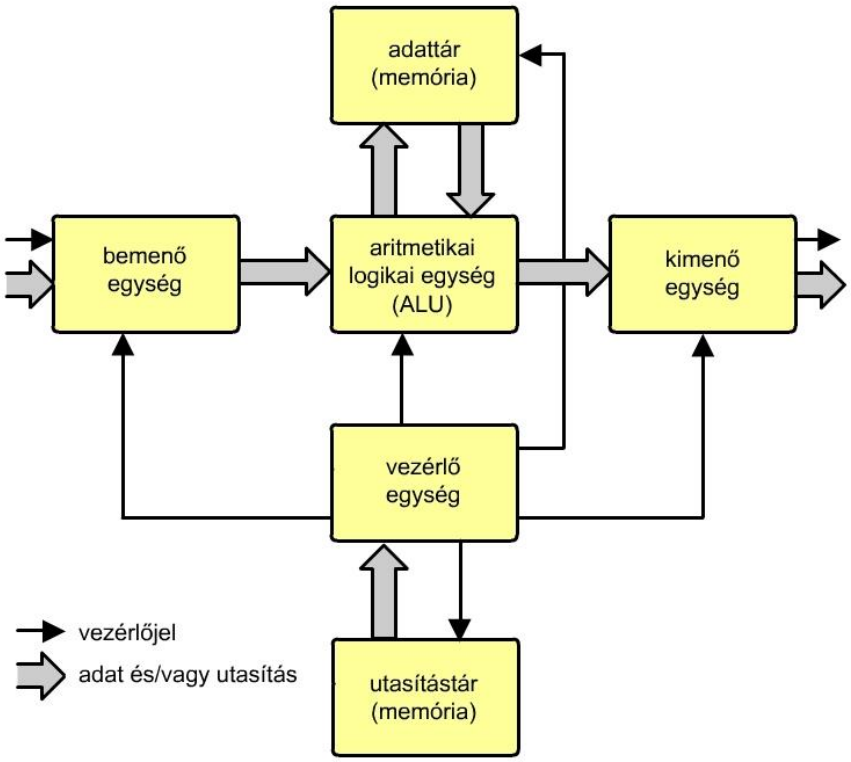
A modell hátránya az, hogy a memória felépítése főleg az adatábrázolás szempontjából optimalizált, ugyanakkor ez a bájt szóhosszúság az utasítások szempontjából már nem ideális. Még hátránya a modellnek, hogy véletlenül (de szándékosan is) működés közben a memóriában tárolt program felülírható, emiatt a programrész újbóli végrehajtásakor már a módosított utasítást dekódolja és hajtja végre a vezérlő egység. Hátrány még az is, hogy utasítás olvasásakor a belső sínek foglaltak, más művelet nem hajtható végre, ez igaz az adattovábbításra is. Ez a probléma világosan látszik az 3.10. ábrán, ahol egy nagyon leegyszerűsített kapcsolat látható. A modell a SISD típusú számítógépek csoportjába tartozik (lásd később, 3.1.7. fejezet).



3.10. ábra. Erősen leegyszerűsített Neumann modell.

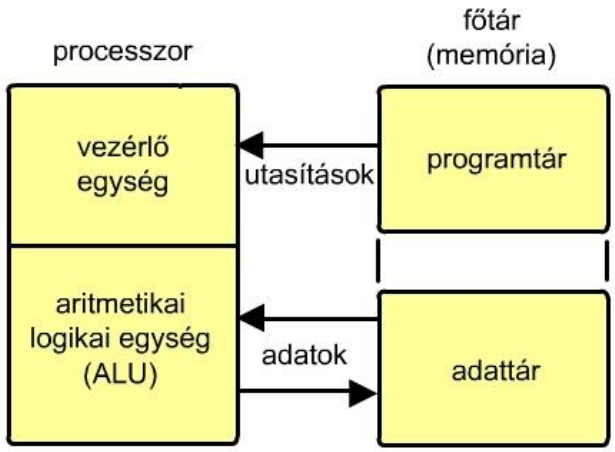
3.1.5.1.2. HARVARD MODELL

A Harvard modell már kiküszöböli a Neumann modell imént ismertetett két hátrányát. A harvard modell a 3.11. ábrán látható.



3.11. ábra. Harvard modell.

Ennél a megoldásnál jól látható, hogy a tár (memória) két elkülönített részből áll, a felső továbbra is hagyományosan bájt struktúrában megépített, míg az ábra alján levő programtár az utasítások igénye szerint került kialakításra (szóhosszúság). Még az is következik az ábrából, hogy adatforgalom esetén párhuzamosan (egyidőben) utasításlehívás is történhet, így gyorsítva a processzor működését. Itt nem fordulhat elő, hogy (véletlen, vagy szándékos) beírás történjen a programtárba, ahol az utasítások vannak. A leegyszerűsített Harvard modellen világosan láthatók a fentiekben leírtak (3.12. ábra). A modell a SISD típusú számítógépek csoportjába tartozik (lásd később, 3.1.7. fejezet).

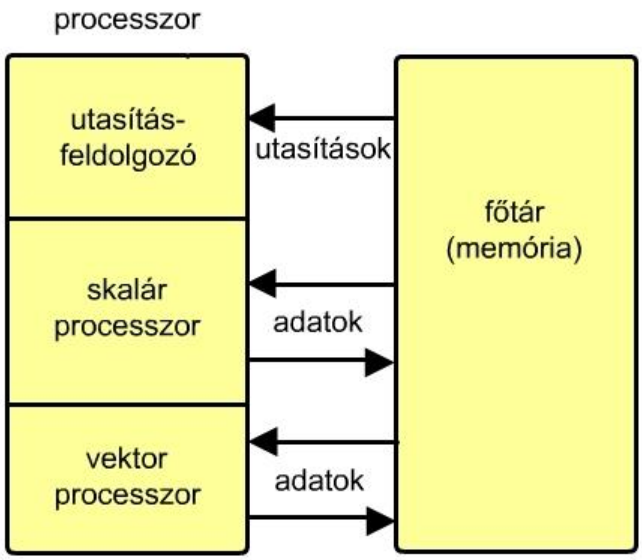


3.12. ábra. Erősen leegyszerűsített Harvard modell.

A Neumann megoldás a nagyobb, míg a Harvard a kisebb teljesítményű processzorokra jellemző. A kisebb teljesítményű processzorok tipikus alkalmazási területe az ipari berendezések feladatait megoldó vezérlő-, irányító és szabályozó berendezések. Tekintettel arra, hogy gyakorlatilag az összes többi megoldás egyforma a két modellnél, az architektúra tárgyalása során nem teszünk különbséget, csak abban az esetben, ha a fent említett két részletnél van eltérés.

3.1.5.1.3. VEKTOR-SZÁMÍTÓGÉP MODELL

Bizonyos matematikai-tudományos számításoknál vektorokkal számolnak. Ezeknél a vektoroknál sokszor ugyanazt a műveletet kell végrehajtani a vektor egyes elemeinél, ez megoldható átlapolt utasítás-végrehajtással. Az utasítást csak egyszer kell lehívni, valamint az átlapolás gyorsítást jelent. Műszakilag a módosított vektorelemek gyors visszatöltése a memóriába a műveletvégző egység előtt és után regiszterláncok felhasználásával oldják meg. A skalár számok feldolgozása ilyen módon lassítja a feldolgozást, ezért a processzor tartalmaz még egy skalárprocesszort is (3.13. ábra). Ilyenkor SIMD architektúráról beszélünk (lásd később, 3.1.7. fejezet).



3.13. ábra. Vektorprocesszor erősen leegyszerűsített modellje.

3.1.6. SZÁMÍTÓGÉPEK OSZTÁLYOZÁSA TELJESÍTMÉNYÜK SZERINT

Ez az osztályozás a gépek teljesítményét veszi alapul, de mivel – mint az imént láttuk – különböző módon lehet a teljesítményt mérni, az egyes csoportok között nem lehet éles határt húzni, Három csoportot különböztetünk meg:

Nagygépek – mind sebességük, mind háttértár kapacitásuk nagy, alkalmasak nagymennyiségű adat gyors feldolgozására. Gyakran használják tudományos feladatok megoldására. Nemcsak a hardver, hanem a szoftver is, így az operációs rendszer is nagyteljesítményű. Többgépes

rendszerekben a központi gép szerepe jut nekik. Általában a processzor több integrált áramkörből áll, maga a gép mérete is nagy, akár egy vagy több szobányi.

Minigépek – az előző nagygépeknél kisebb teljesítmény, kevesebb adattal dolgoznak. Alkalmazási területük pl. a folyamatirányítás, CAD/CAM tervezés.

Kisgépek – más néven mikrogépek, általában önállóan dolgoznak, csak hálózaton keresztül vannak összekötve. A mikro szó főleg a méretükre utal. Szerényebb perifériákkal rendelkeznek, mint az előző két csoport gépei. A processzor egyetlen integrált áramköri tokban helyezkedik el. Egyetlen házban, kompakt berendezésként használjuk őket. Alkalmazási területük a személyi számítógép, de irodai, ipari stb. feladatoknál is használjuk őket. Leggyakrabban ezekkel a számítógépekkel találkozunk, valójában mindennapi életünk mára szerves részét képezik. Itt ne csak kifejezetten számítógépekre gondoljunk, hanem egyéb, processzorokkal ferszerelt eszközökre is, például az okostelefonok, de környezetünk berendezéseiben is számtalan más helyen működnek (gépkocsik, repülők, mérőberendezések, ipari folyamatok stb.).

3.1.7. SZÁMÍTÓGÉPEK OSZTÁLYOZÁSA UTASÍTÁS- ÉS ADATFOLYAM SZERINT

M.J. Flynn 1966-ban publikálta a nagyobb teljesítményű gépek csoportosításának egy lehetséges módszerét:

SISD (Single Instruction Stream Single Data Stream) – egy utasításfolyam egy adatfolyam gépek, ezek egyenként hajtják végre az utasításokat, egy-egy adaton. Ide tartoznak a hagyományos Neumann elven működő gépek. Egy vezérlőegység, egy aritmetikai egység jellemzi ezeket a gépeket.

SIMD (Single Instruction Stream Multiple Data Stream) – egy utasításfolyam többszörös adatfolyam gépek. Egy vezérlőegység párhuzamosan több aritmetikai egységgel dolgozik, tehát párhuzamosan dolgozza fel az adatokat.

MISD (Multiple Instruction Stream Single Data Stream) – több utasításfolyam egy adatfolyam gépek. Ez a csoport nem létezik, de meg kell jegyezni, hogy néha a csővezeték (pipeline) gépeket egyes szakemberek ide sorolják.

MIMD (Multiple Instruction Stream Multiple Data Stream) – több utasításfolyam többszörös adatfolyam gépek. Ide sorolják a többprocesszoros gépeket, ahol az erőforrások (memóriák, szoftverek stb.) nagy részét közösen használják a processzorok.

3.1.8. SZÁMÍTÓGÉPEK OSZTÁLYOZÁSA UTASÍTÁSKÉSZLETÜK ALAPJÁN

Minden processzorhoz tartozik egy rá jellemző utasításkészlet, amelyből a programozó kiválaszthatja a számára megfelelő, az adott feladatot végrehajtó utasításokat, azokat a megfelelő sorrendbe összerendezve kapja a programot. Az utasítások úgynevezett gépi kódú utasítások, melyek tárolása az adatokhoz hasonlóan 0 és 1 logikai értékekkel történik. A programok (vagy azok részei) végrehajtáskor a főtárban vannak. A program utasításait a processzor innen egymás után lehívja és végrehajtja. Az utasításhoz tartozik még egy címrész is, amely arra szolgál, hogy meghatározza az operandus helyét (forrás - honnan és cél - hova).

Egy adott processzorhoz tartozó utasításkészlet utasításainak száma meghatározza tárolásához szükséges szóhosszúságot is. Például 234 különböző utasítás kódolása nem lehetséges 7 biten, hiszen $2^7 = 128 < 234$, tehát 8 bit kell, ugyanis $2^8 = 256 > 234$.

A gyakorlatba két utasításkészlet-típus alakult ki:

- CISC (Complex Instruction Set Computer) - összetett utasításkészletű gépek és
- RISC (Reduced Instruction Set Computer) - csökkentett utasításkészletű gépek.

3.1.8.1. A CISC UTASÍTÁSKÉSZLETTEL RENDELKEZŐ SZÁMÍTÓGÉPEK

Kialakulásának két oka van, egyrészt a memória mérete a kezdeti időkben viszonylag szerény volt (műszaki problémák mellett az ár is számított), másrészt az összetettebb feladatokat ellátó utasításokkal a programozó könnyebben állított össze rövidebb programokat (hatásosabb programokat). Az utasításkészlet sok utasítást tartalmaz. Elterjedését az is támogatta, hogy a nehezen továbbfejleszthető huzalozott logikájú vezérlőegységet felváltotta az egyszerűbb továbbfejlesztést biztosító mikroprogramozott vezérlőegység. A mikroprogramozott vezérlőegység valójában egy igen egyszerű Neumann-elvű hardver, tehát processzor a processzorban elven működik a központi egység. A huzalozott, illetve mikroprogramozott vezérlőegység részlétesen az 4.4.1. fejezetben található.

3.1.8.2. A RISC utasításkészlettel rendelkező számítógépek

Mivel a mikroprogramozott vezérlőegység egy határon túl már gátlótényezővé válik a processzorok teljesítménynövelésénél, ezért elkezdtek olyan megoldásokat keresni, ahol az utasításkészlet csak a legegyszerűbb utasításokat tartalmazza és azok végrehajtását nem mikroprogramozott technikával oldották meg. Ekkor a fordítás közvetlenül történik magas szintű programozási nyelvről gépi szintre.

A következő 3.1. táblázatban látható néhány kritérium a kétféle utasításkészlet jobb megértésére.

3.1. táblázat. A CISC és RISC utasításkészlet összehasonlítása

	CISC	RISC
Az utasításkészlet bonyolultsága	Sok, összetett utasítás	Kevés, egyszerű utasítás
Az utasítások száma	100-300	32-128
Végrehajtáskor igényelt gépi ciklusok száma	Több, jellemző 1-3, de előfordul akár 20 is	Csak 1 gépi ciklus
Utasításhossz	Változó, bonyolult utasítások hosszabbak	Rögzített utasításhossz
Címzési mód	Sok, 8-20	Csak 2-4 (írás-olvasás)

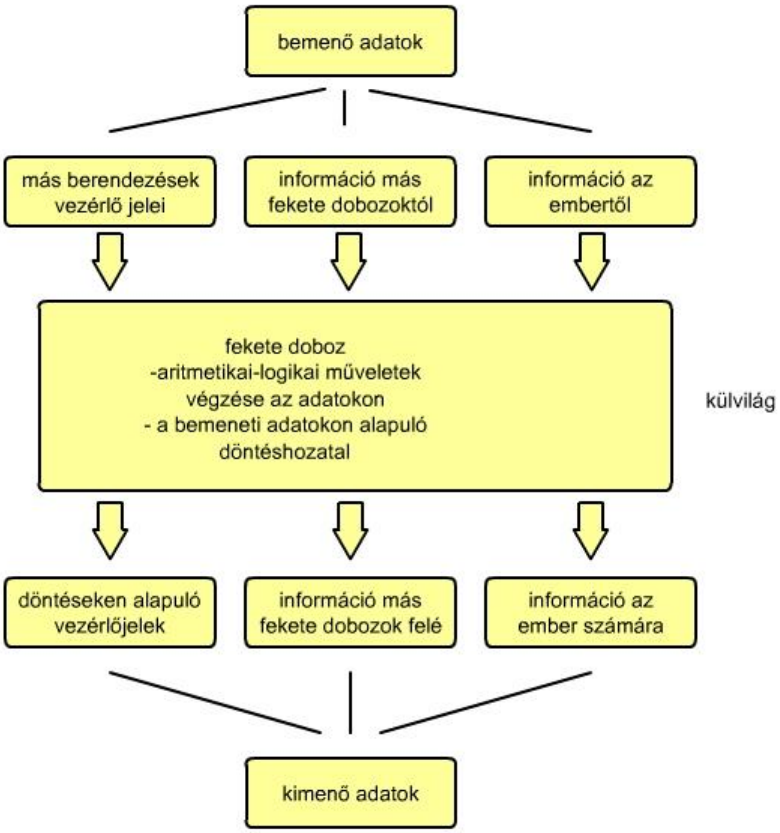
Gyorsítás futószalag technikával (pipelining)	Nem nagyon alkalmazható	Kifejezetten támogatja, hatásos pipelining
Regiszterek száma	kevés	Sok (regisztertár)
Memóriavédelem	Hardver megoldás	Szoftver megoldás

Előfordulnak olyan processzor megoldások is, ahol az utasításkészlet CISC típusú, de a hardver ezeket lefordítja RISC alakúra és egy RISC mag hajtja végre, így érhető el nagyobb feldolgozási sebesség.

3.2. A SZÁMÍTÓGÉP ÉS A SZÁMÍTÓGÉP-RENDSZER FELÉPÍTÉSE, MŰKÖDÉSE

Az 3.1.6. fejezetben, a számítógépek csoportosításánál találkoztunk a nagy-, közepes és kisszámítógép fogalmakkal, ebben a fejezetben döntően a kisszámítógépek csoportjába tartozó mikroszámítógépekkel fogunk foglalkozni, a működés szerinti felosztásnál alkalmazott Neumann modell szerint. Ez azért is logikus, mert gyakorlatilag döntően ezekkel a gépekkel fogunk találkozni, például ilyen az IBM PC személyi számítógép, vagy az ipari alkalmazásoknál elterjedt mikrovezérlő is. A megértést fokozatosan építjük fel, egyszerű fogalmaktól, működési elvektől kezdve a bonyolultabb felé.

A mikro (μ) görög betű jelentése kicsi, apró, ami itt az elemek méretére utal és nem a teljesítményére. Először nézzük meg mi is egy mikroszámítógép-rendszer. A 3.14. ábra fekete doboza a mikroszámítógép-rendszer, ami egy ‘intelligens’ (‘okos’) berendezés a feldolgozandó, ‘nyers’ adatok és a kimenetekre kerülő adatok, vezérlések között. Mivel a fekete doboz az elektronikus digitális elemek mellett még mechanikus, pneumatikus, analóg elemeket is tartalmazhat, szükséges megvilágítani a tartalmát.



3.14. ábra. A bemenetek és kimenetek között levő fekete doboz, a mikroszámítógép-rendszer.

A fenti fekete doboz feltétlenül kell hogy tartalmazzon:

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

Ez a két egység teszi ezt a berendezést (a mikroszámítógép-rendszert) ‘intelligenssé’. A gyorsabb és könnyebb feladatvégzés érdekében a fekete doboznak tartalmaznia kell adattárolásra és utasítástárolásra memóriát (tárat).

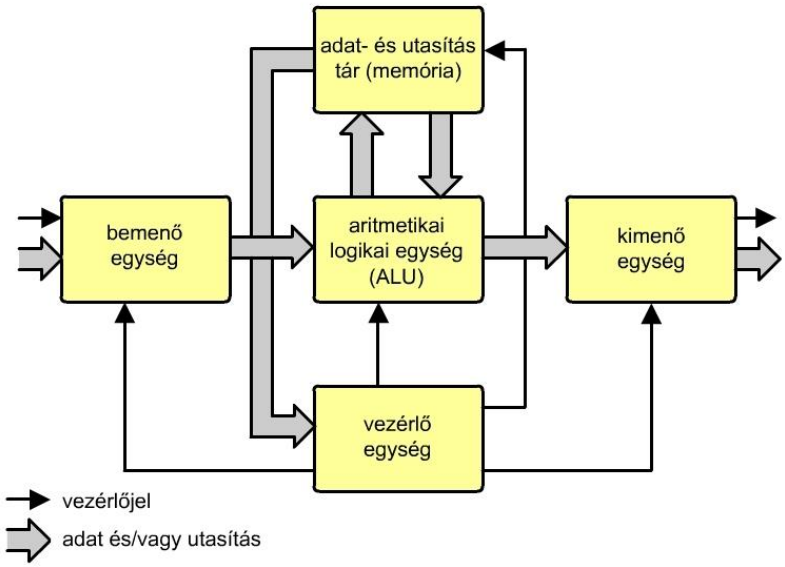
Ha az adatok és utasítások számára ugyanazt a tárat (memóriát) használja a gép akkor Neumann-féle számítógépről beszélünk, ha viszont a program tárolására is és az adatok tárolására is elkülönített tárakat használ a gép, akkor Harvard-típusú számítógépről van szó.

Mind az adatok beviteléhez, mind az adatok kiírásához fizikai hordozókra, egységekre van szükség. A 3.2. táblázatban felsoroltunk néhány ismert bevitelt biztosító egységet, a 3.3. táblázatban pedig a kimenő adatok fizikai megjelenítését szolgáló eszközt. Midkét táblázatban látható, hogy három tipusú külvilági kapcsolatfajta létezik, egyrészt az ember, másrészt a másik számítógépek, de nem utolsó sorban a környezet felé. A felsorolás nem teljes, sok egységet, mely sokáig igen fontos volt a számítástechnikában túlhaladott az idő, a számítástechnika fejlődése pedig újabbnál újabb megoldások alkalmazását hozza magával.

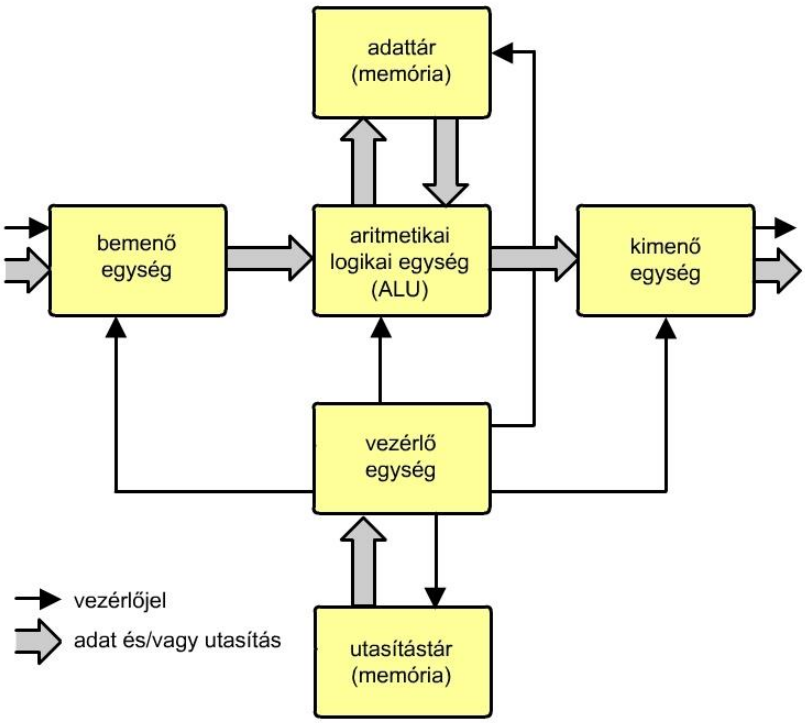
3.2. táblázat. Bemeneti adthordozók (információhordozók).	
A bemeneti adat	Bemeneti adthordozó
az embertől	kapcsolók, klaviatúra (billentyűzet), teleprinter, fényceruza, egér, mikrofon, kamera, stb
más számítógépektől	CD/DVD, modem, soros átvitel (RS 232, RS 485), párhuzamos átvitel (CENTRONICS), internet, stb
a környezetből	érintkezők analóg-digitális átalakítók, stb.

3.3. táblázat. Kimeneti adthordozók (információhordozók).	
A kimeneti adat	**Kimeneti adathordozó**
az ember felé	kijelző tábla, numerikus kijelző-egység, nyomtató, képernyő, hangszóró, projektor, stb.
más számítógépek felé	modem, soros átvitel (RS 232, RS 485), párhuzamos átvitel (CENTRONICS), hálózati csatlakozó, stb.
a környezetbe	digitális-analóg átalakító, jelfogó léptetőmotor, stb.

A 3.15. ábrán a Neumann-, míg a 3.16. ábrán a Harvard-típusú számítógép modelljei láthatók. Jól látszik közöttük a különbség, míg a Neumann gép adat- és programtárolásra ugyanazt a memóriát használja, addig a Harvard gép különbséget tesz az adat és utasítás között, így más tipusú tárat igényel ezek tárolására.



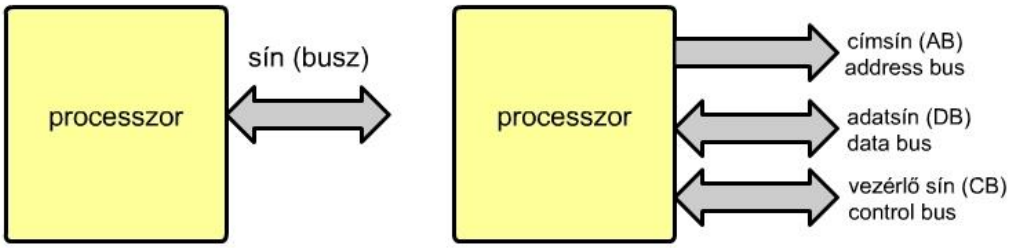
3.15. ábra. Neumann-féle számítógépmodell.



3.16. ábra. Harvard-féle számítógépmodell.

Mindkét fent ismertetett modell (Neumann és harvard) az ábrákon az egyes egységek összekapcsolásával egy funkcionális modellt alkot. Egyszerűen érthető meg a rendszer működése. Ha azonban belemélyedünk a részletekbe, akkor levonhatjuk a következtetést, hogy az egyes egységek nehezen kivitelezhetők az itt ábrázolt módon egyszerűbb felépítést kellene alkalmazni. Egy példa, mindkét modellnél a főtár két csatlakozási ponttal rendelkezik, adatok olvasására és írására is külön-külön. Ezt elkerülendő, a processzornál a külvilág felé egy csatlakozási felületet alakítanak, amely felület nem más, mind több be-/ki vezeték (3.17. ábra

bal oldala, a) rész). Részletesebben a sin 3 részből áll, mint ahogy az az 3.17. ábra b) részén látható.



3.17. ábra. A processzor sematikus ábrázolása (**a** - egyszerű, **b** - részletes).

Az ábra a) részén csak egy a külvilággal kapcsolatot tartó sinrendszer (angolul bus, innen ered a magyarul is használatos busz elnevezés) látható, ami a processzor és más elemek (memóriák, illesztő körök) közötti kapcsolatot biztosítja. Ez nem más mint lábak, illetve vezetékek összessége. Az ábrán a sín kétirányú, ami kétirányú adatáramlást tesz lehetővé, de nem azonos időben, hanem a feladattól függően egyszerre csak egyirányút.

A 3.17. ábra b) részén ez a sín már három részből áll, itt:

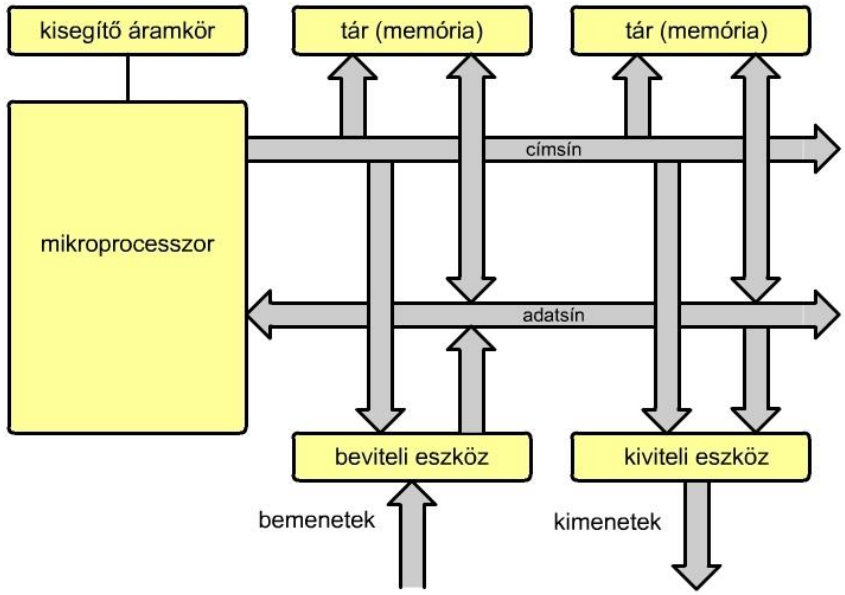
- AB címsin (Address Bus),
- DB adatsin (Data Bus) és
- CB vezérlő sin (Controll Bus).

A címsin csak egyirányú adatforgalmat engedélyez, mégpedig a processzortól más elemek (perifériák, memóriák) felé, az adatsínen már kétirányú adatáramlás lehetséges, de egyidőben csak egyik irányban. A vezérlő sín kétféle irányban dolgozhat, de egy vezeték mindig csak egyik irányban, mig a másik tipus a másik irányban.

A nagyobb teljesítményű számítógépekben a központi egységet processzornak, míg a kisebb teljesítményű gépekben mikroprocesszornak nevezzük. A (mikro)processzor olyan integrált áramkör (IC), amely egy számítógép központi egységének megfelelő funkciókat lát el, vagyis a memóriából (tárból) kapott adatokon a programnak megfelelően logikai és/vagy számítási műveleteket végez, valamint az eredmények alapján kiválasztja a következő lépést.

3.3 A SZÁMÍTÓGÉP ÉS A MIKROSZÁMÍTÓGÉP.

A 3.18. ábrán egy számítógép, illetve egy mikroszámítógép sematikus rajza látható, ahol is a központi egység, a processzor, vagy mikroprocesszor köré néhány elem került (perifériák és memóriák), mégpedig a cím-, adat- és vezérlő sín felhasználásával.



3.18. ábra: A számítógép, illetve a mikroszámítógép felépítése.

A számítógép nyitott struktúrát képez, így kiépítése, bővítése az igényektől, illetve a tervezőmérnöktől függ. Mindig a sín (busz) köré épül fel egy gép. Korlátot a gép kiépítettségének a rendelkezésre álló címezhető terület szab.

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

Az ábrán egy bemeneti-, valamint egy kimeneti illesztőegység van, a kiviteli és beviteli eszközök számát szintén a rendelkezésre álló címzési terület nagysága szabja meg. A perifériák száma lényegesen kisebb, mint a memóriacelláké, ezért kevesebb címet használnak. Minden elemnek, egységnek külön, saját címe van, az esetleg egy címhez rendelt két vagy több egység adatütközést okoz, ami lehetetlenné teszi a rendszer működését.

Mivel a központi egység, a CPU (processzor) tartalmazza a vezérlő egységet, ezért a címet mindig a processzor állítja elő, ami aktív elem, ellentétben a rendszer többi elemével, amelyek passzívak (kivétel a memória- és perifériavezérlő fejlettebb architektúrájú rendszereknél).

Az egységek az adatközléskor a kétirányú adatsínt használják információcserére, ami egyes esetekben egyirányú is lehet, mint pl. az ábrán a bemenő illesztő áramkör és adatsín között csak a (mikro)processzor felé történik adatáramlás.

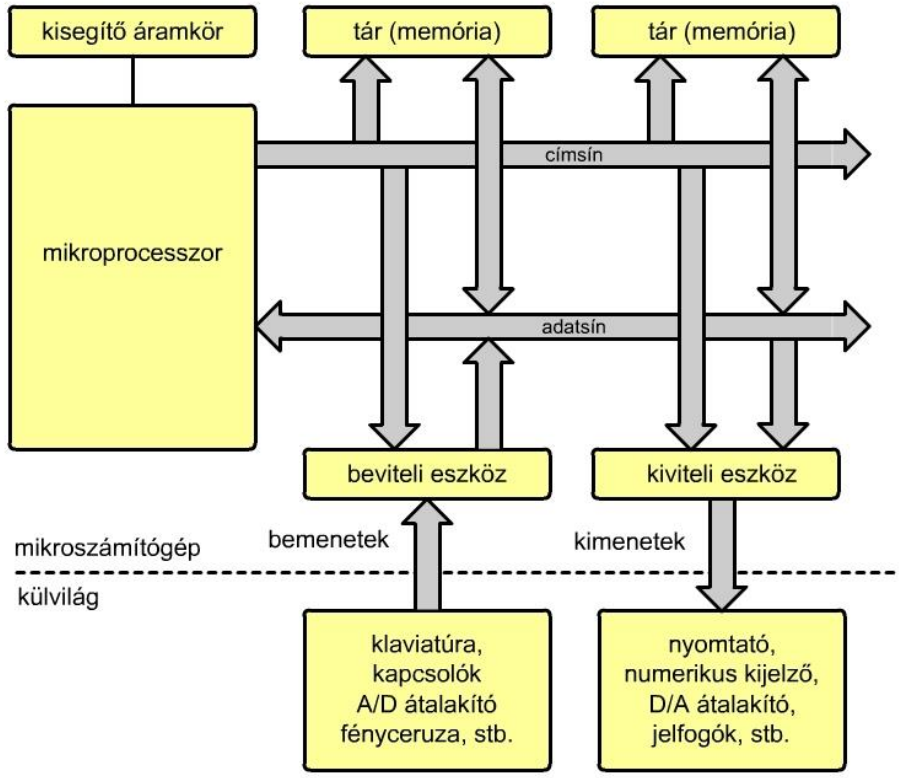
3.4. A SZÁMÍTÓGÉP-RENDSZER

A számítógép feladata az, hogy a külvilágból kapott adatokat és információkat, valamint vezérlő jeleket a beírt program segítségével átalakítsa a felhasználó által megkívánt alakú adatokká, információkká és vezérlőjelekké.

A számítógépet olyan elemekkel kiegészítve, amelyek egy adott műszaki feladathoz illeszkednek kapjuk meg a számítógép-rendszert. A külvilágból származó adatok, információk,

jelek különböznek a gépen belüli jelektől, pl. analóg jel a hőmérséklet, nyomás, sebesség stb.. Diszkrét bináris jel esetében is lehet eltérés a jelek között, jelszintben (feszültség) is. Ezért szükséges a jeleket elektromosan illeszteni. A 3.19. ábrán látható egy számítógép-rendszer sematikus rajza.

Mind bemenő- mind kimenő illesztő-áramkörök széles skálájával találkozhatunk egy számítógép-rendszernél.



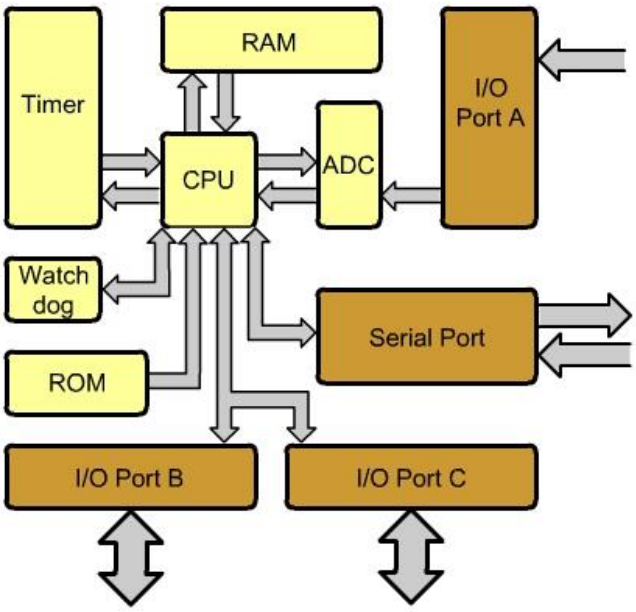
3.19. ábra. A számítógép-rendszer sematikus ábrázolása.

3.5. A MIKROSZÁMÍTÓGÉP ÉS A MIKROVEZÉRLŐ ÖSSZEHASONLÍTÁSA

A 3.2. fejezetben ismertetett (mikro)processzor általában egy integrált áramköri tokban (IC) helyezkedik el, ehhez hozzáépítve különböző típusú, kapacitású és darabszámú elemet egy mikroszámítógép építhető fel. Általában ezek az elemek is, mint a memória, kimenő- és bemenő illesztőegységek szintén egy-egy IC-ben (integrált áramköri tokban) helyezkednek el. Az elemek közötti kapcsolat, vagyis adatcsere a sínrendszeren keresztül történik. Ez a sínrendszer, mindamellett hogy biztosítja a rendszeren belüli adatáramlást, illetve információáramlást, a további bővítéshez is lehetőséget biztosít. Ezt a bővítést egyrészt a felhasználó igénye, másrészt az elemek (mikroprocesszor, memória, ki- bemenő egységek) műszaki korlátai szabják meg. Természetesen egy-egy feladat megoldásakor más és más igények merülnek fel sebesség, kapacitás szempontjából is, ilyenkor ezt is figyelembe kell venni.

A műszaki haladás állandó, újabb és újabb technológiai megoldások jelennek meg, így egy-egy IC (integrált áramkör) egyre több és több elemet tartalmaz. Ez azt is magával hozza, hogy a mikroprocesszor mellé, ugyanarra a lapkára már más elemek (funkcionális egységek) is felkerülnek. Ekkor is érvényesek azonban a fentiekben ismertetett architektúrák, kapcsolatok, valamint működési elvek.

A kis kapacitású, tehát viszonylag egyszerű, néhány bemenetet, kimenetet és kis kapacitású memóriát tartalmazó rendszer kiépítése egyedi elemekből bonyolult hardvert és architektúrát igényelne a hagyományos mikroprocesszor, memória, ki-, és bemenő egységek felhasználásával. Ennek a problémának a kiküszöbölésére építették meg az úgynevezett mikrovezérlőket, amelyek egy IC tokon (integrált áramkörön) belül még memóriát és ki-bemenetet tartalmaznak. Néhány egyéb kiegészítő áramkörrel együtt (például számláló, soros kommunikációs hardver stb.) kapjuk a teljes számítógépet. Mivel az így kapott elem teljes értékű számítógép, valamint a kivezetések száma is korlátozott, már a külvilágba nem is vezetik ki a síneket, így a rendszer nem is bővíthető (léteznek olyan mikrovezérlők, amelyeknél biztosított a bővítés). Ugyanakkor a belső felépítés, működés továbbra is megegyezik a mikroszámítógépnél leírtakkal. A 3.20. ábrán egy mikrovezérlő blokkvázlata látható.



3.20. ábra. Mikrovezérlő blokkvázlata.

Az ábra egyes elemei:
Timer – időzítő/számláló
RAM (Random Access Memory) – véletlen elérésű felejtő memória
I/O Port A, B és C – ki- bemenő bináris illesztő kapuk
CPU (Central Processor Unit) – központi egység, processzor
ADC (Analog to Digital Converter) – analóg-digitális átalakító
Watch dog – önellenőrző egység
ROM (Read Only Memory) – csak olvasható memória, nem felejtő memória

Serial Port – soros kommunikációs hardver egység.

4. A SZÁMÍTÓGÉP KÖZPONTI EGYSÉGE, A PROCESSZOR

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

1. a program kezdőcímének beállítása,
2. az itt található utasítás lehívása a processzorba (a vezérlőegységbe) és végrehajtása,
3. lépés a következő címre,
4. visszalépés a 2. pontra.

A 3. pontban szereplő címmeghatározás lehet feltétel nélküli, ekkor gyakorlatilag a programban levő következő címről van szó, vagy valamilyen feltételhez kötött, amely valamilyen jelzőbit-érték alapján történik, ami a programban egy ugrást jelent.

A számítógép-programozás szempontjából a számítógép legfontosabb eleme az aritmetikai és logikai műveletek elvégzésére szolgáló ALU (Arithmetic Logic Unit) - aritmetikai-logikai egység, az akkumulátor (A - Accumulator), az utasításokat értelmező és végrehajtó CU (Control Unit) - vezérlőegység, a programot tartalmazó programmemória, valamint az adatokat tartalmazó adatmemória.

4.1. A PROCESSZOR (PROCESSOR)

Ez az egység a számítógép alapegysége. Feladata a programtárból beolvasott utasítások dekódolása és végrehajtása, amelyekből új adatokat és vezérlőjeleket hoz létre a számítógép egységes működtetéséhez.

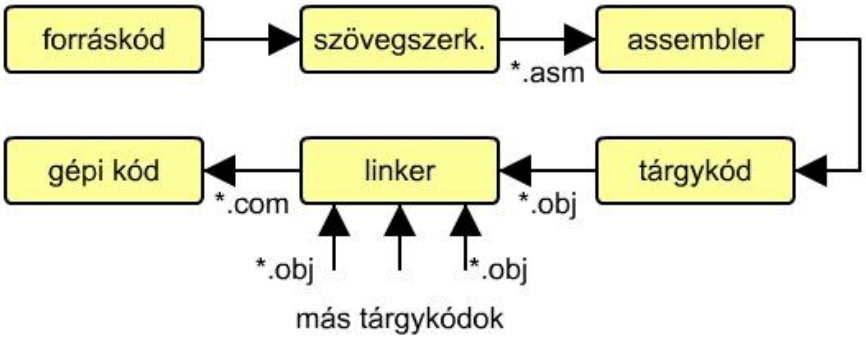
Tekintettel arra, hogy gyakorlatilag a ma használatos processzorok legkisebb szóhosszúsága 8 bit, illetve a nagyobb szóhosszúságú gépek is valamilyen módon kötődnek a 8 bithez (16 bit, 32 bit és 64 bit), az itt bemutatott modell az egyszerűség kedvéért is 8 bites. Ez a struktúra bővíthető bármilyen módon, így megérthető ennek a 8 bites modellnek a segítségével bármely más szóhosszúságú számítógép működése is.

4.2. AZ UTASÍTÁSOK SZEREPE, FELÉPÍTÉSE

Minden számítógép más és más, kismértékben, vagy jelentősen különböző utasításkészlettel oldja meg az elemi feladatokat. Az utasításkészlet kialakításánál a tervezéskor a következő tényezőket veszik figyelembe a tervezők:

- tapasztalat, leggyakrabban milyen típusú utasításokra van szükség,
- a programtárolás gazdaságossága,
- az utasítás lehívásának módja a főtárból a központi egységbe,
- az utasításon belül a műveleti kódra felhasznált mező nagysága akkora kell hogy legyen, hogy a kívánt számú utasítás kódolható legyen és
- a címezhető memória mérete.

Egy-egy feladat lépéseit programutasításokkal adjuk meg, vagyis a számítógépen belül, a memóriában elemi lépések egymásután való elhelyezésével programot írunk, ezt beírjuk a főtárba, a központi, vagy más néven operatív memóriába. A gépi kódokat, a bináris alakban tárolt programot különböző segédprogramokkal hozhatjuk létre. Például egy magasszintű programozási nyelven megírt program először egy szövegszerkesztőbe kerül (begépelés), ez eltárolható valamelyik háttértárolón (leggyakrabban merevlemezen), majd egy fordítóprogram ezt a forráskódot szabványos tárgykóddá konvertálja, egyidejűleg létrehozva egy "listát", amelyik az eredeti forrásprogramot a lefordított címekkel és binárisan kódolt gépi utasításokkal együtt tárolja. Ha vannak hasonlóan lefordított más szabványos programrészek, akkor az összefűző program ezekből a szabványos tárgykódokból végrehajtható programot hoz létre, összekötve az egyes részprogramokat (modulokat). A fordítás és szerkesztés folyamata látható a következő ábrán.

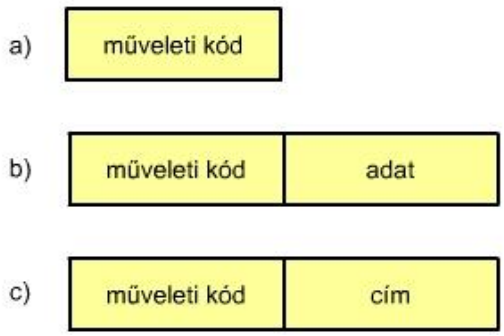


4.1. ábra. A progarmírás, fordítás és szerkesztés folyamata.

Egy utasítás gépi kódú általános alakja látható a 4.2. ábrán.

4.2. ábra. A gépi kódú utasítás általános alakja.

A gépi kódban a műveleti kód mező, mint az utasítás többi része is, kettes számrendszerben kódolt információ, ami meghatározza a végrehajtandó feladatot, tehát az utasítás „feladata” van ide beírva. Az utasítás harmadik mezőjében levő cím, vagy adat információ határozza meg a műveletben szereplő adatot, vagy cím esetén az adatnak a helyét a főtárban. A 4.3. ábra az utasítások néhány lehetséges megoldását láthatjuk. (Itt most nem vesszük figyelembe a „címzési mód” mezőt).



4.3. ábra. Egyszerű utasítás három lehetséges alakja.

Az a esetben processzoron belüli műveletről van szó, nincs szükség egyéb információra. A b esetben maga az utasítás tartalmazza a műveletben szereplő adatot (implicit címzés), a c esetben pedig az utasítás az operandus helyét adja meg a főtárban, cím alakjában.

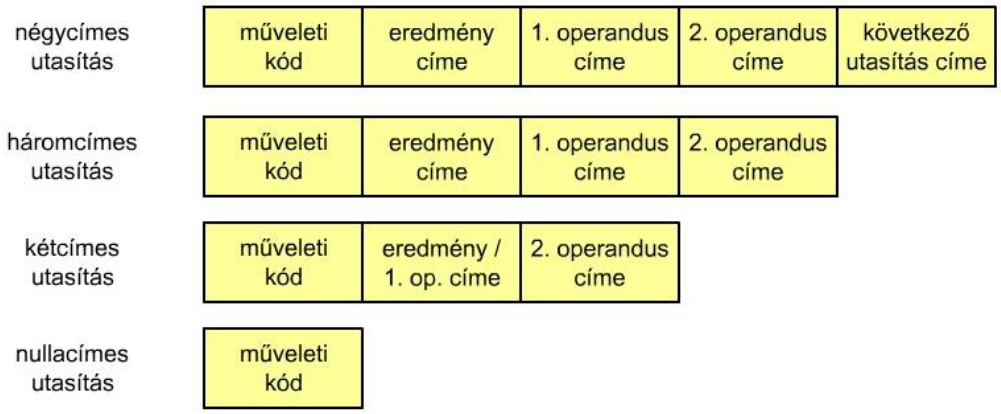
A következő 4.4. ábrán a 4-, 3-, 2-, 1- és 0 címes utasításábrázolás látható.

A 4 címes utasítás a leghosszabb, a műveleti kód mellett tartalmazza az eredmény-, az első- és második operandus helyét a főtárban, illetve még azt a címet is, ahol a következő utasítás található. Tekintettel arra, hogy a programjaink kb. 85 %-a lineáris struktúrájú, vagyis fizikailag egymást követik a tárban az utasítások, a „következő utasítás címe" mező elhagyható, de helyette a processzornak kell rendelkezni egy olyan regiszterrel, amely mutatja az aktuális utasítás címét a memóriában. Ez a regiszter a „programszámláló" (Program Counter – PC) regiszter, amelyet néha „utasítás mutatónak", IP (Instruction Pointer) is neveznek.

Az így kapott 3 címes utasítás egyrészt rövidebb, mint a 4 címes, ugyanakkor, mivel a következő utasítás címe a processszorban van, nem kell azt beolvasnia a memóriából. Tekintettel arra, hogy processzor-processzor műveletek körülbelül tízszer gyorsabbak, mint a processzor-főtár adatcsere, még ez is gyorsabbá teszi a háromcímes utasítás végrehajtását, mint a négycímesét.

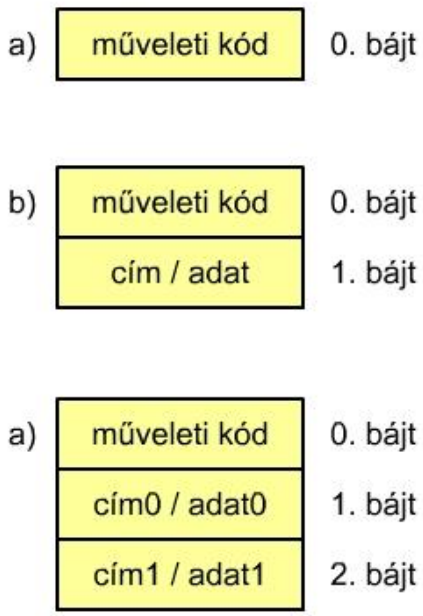
Tovább lehet csökkenteni az utasítás hosszát, ha áttérünk a kétcímes utasításokra, ahol az első operandus címe ugyanaz, mint az eredmény címe. Ez gyakorlatilag azt jelenti, hogy utasításvégrehajtás előtt a memória adott helyén az első operandus van, az utasítás végrehajtása után az eredmény ugyanebbe a memóriacellába kerül, felülírva az első operandus értékét (ez gyakorlatilag azt jelenti, hogy végleg elveszik az első operandus).

Még egy egyszerűsíést végrehajtva jutunk el az egycímes utasításokhoz. A kétcímes utasításban kihagyjuk „az első operandus/eredmény" mezőt, azzal, hogy helyette egy processzoron belüli regisztert alkalmazunk, az akkumulátort (A - Accumulator). Néhány processzornál ennek a regiszternek a neve munka regiszter, W (Work Register). Ekkor az utasításhossz már elfogadható számú bitből áll, de, mint az imént a PC-nél láttuk, az A, lévén hogy a processszoron belül van, kb tízszer gyorsabban hajt végre műveletet, mintha processzor-főtár művelet lenne. Amennyiben az utasítás olyan, hogy nincs operandus, megkapjuk a 0 címes utasítást, ez csak műveleti kódból áll.



4.4. ábra. A négy-, három-, két-, egy- és nullacímes utasítások

Műszaki, technológiai és megbízhatósági korlátok miatt egyszerűbb és olcsóbb kisebb szóhosszúságú processzorokat gyártani. Ma az alsó határ a 8 bit, illetve ennek általában egészszámú többszöröse. Ha a nyolcbites gépnél elemezzük azt, hogy hogyan férhet el egy bájtban a 4.4. ábra szerint értelmezett utasítás, akkor a válasz az, hogy gyakorlatilag sehogy, ugyanis a 256 különböző állapot kevés utasítás létrehozását, illetve igen kis címmezhető memóriacella elérését teszi lehetővé. Ezért célszerű több bájt felhasználása egy utasítás tárolására. Itt is többféle megoldás lehetséges, ezek közül például a 4.5. ábra szerinti megoldás egy lehetséges.



4.5. ábra. Utasítások kódolása bájtokon.

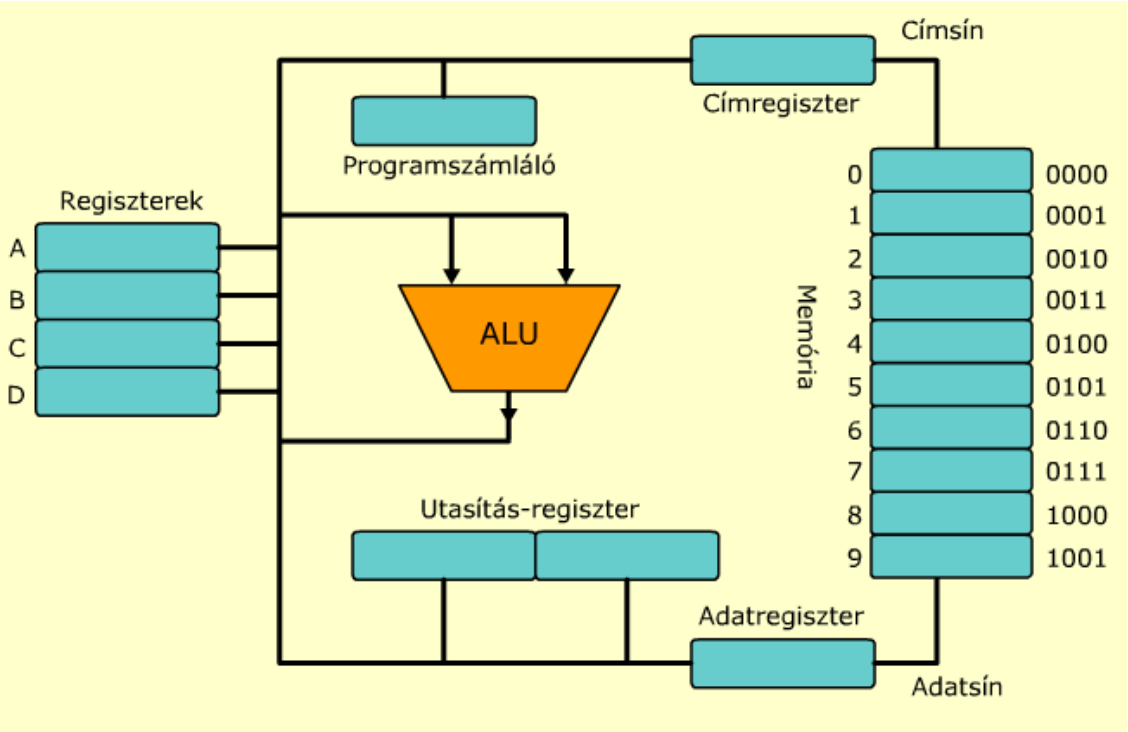
A 4.5. a) ábra utasítás processzoron belüli művelet jelent, a b) utasítás egy nyolcbites közvetlen (implicit) adattal végzett művelet, vagy egy nyolcbites címmel címezhető elem (például periféria) művelete. A 4.6. c) ábra kétbájtos (16 bites) címet ad, ami 0 és $2^{16}-1=65535$címek között tesz lehetővé adatelérést (0000h – FFFFh). Közvetlen adat esetén ez 0 és 65535 közötti egész számot jelent.

A 4.5. ábra mindegyik utasításának első bájtja a műveleti kód, összesen 256 kombináció ábrázolható egy bájton, ez lesz a felső korlátja a lehetséges utasítás-számnak.

4.3 EGYSZERŰ PROCESSZORMODELL

A processzor-modell több eleme és az előző, 4.2. fejezetben tárgyalt utasításábrázolás között szoros kapcsolat van. Amennyiben az utasításokat bájt-struktúrában ábrázoljuk, mint ahogy ezt tettük az előző fejezetben, akkor a bájtok továbbítása, tárolása is a bájt mérethez igazodik, így például a processzor belső sínje, az A akkumulátor, de értelemszerűen a nagyobb méretű (nagyobb szóhosszúságú) elemek is a bájt mérethez kötődnek, ennek kétszeresei, négyszeresei, ilyen elem mondjuk a PC programszámláló is.

A 4.6. ábrán egy processzormodell látható, ahol nem tüntettük fel külön ezeket az adatokat, az egyszerűség kedvéért. Az animációs modellen végig lehet követni a teljes modell működését egy mintapéldán keresztül, ez a 04. MELLÉKLET-ben található meg részletesen.



4.6. ábra. Egyszerű processzor-modell.

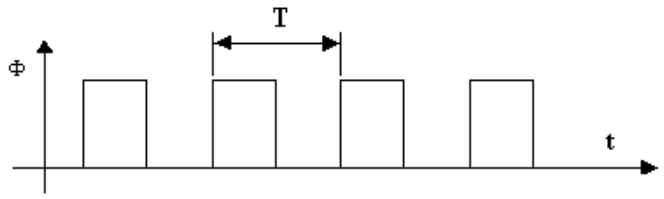
Az ábra egyes elemei:

- A (Accumulator) – akkumulátor, általános célú regiszter, legtöbb utasítás végrehajtásában van szerepe,
- B regiszter – általános célú regiszter, szűkebb szerepköre van mint az akkumulátornak,
- C regiszter – általános célú regiszter, szűkebb szerepköre van mint az akkumulátornak,
- D regiszter – általános célú regiszter, szűkebb szerepköre van mint az akkumulátornak,
- PC (Program Counter) – programszámláló, mutatja az éppen aktuális, végrehajtás alatt álló utasítás címét a memóriában,
- IR (Instruction Register) – utasításregiszter, a memóriából ebbe a regiszterbe kerül a végrehajtandó utasítás (gépi kód és a kiegészítő adat, például érték, vagy cím),
- MAR (Memory Address Register) – memória címregiszter, ezen keresztül történik a főtár adott cellájának címzése,
- MDR (Memory Data Register) – memória adatregiszter, ezen keresztül történik az adatok, műveleti kódok és címek továbbítása a processzor és a főtár között, mindkét irányban és
- FŐTÁR, vagy MEMÓRIA – adatok és utasítások tárolása.

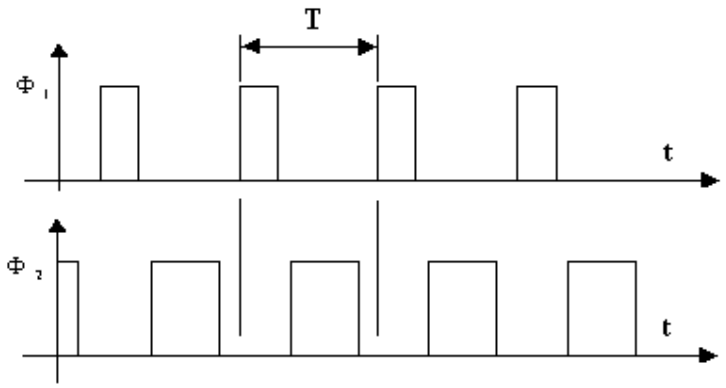
Az utasítás végrehajtása két fázisban történik:

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

A gyakorlatban leggyakrabban két típusú órajel létezik, az úgynevezett egyfázisú (4.7. ábra) és kétfázisú (4.8. ábra) órejel. Ennek az órajelnek a frekvenciája szabja meg a mikrogép működési sebességét. Meghatározásakor figyelembe kell venni az egyes elemek működési sebességét.



4.7. ábra. Egyfázisú órajel.



4.8. ábra. Kétfázisú órajel.

4.4. A PROCESSZOR ÉS RÉSZEI

A számítógép három fő eleme, a processzor, memória és periféria közül először a processzor felépítésével és a működésével ismerkedünk meg. Mint ahogy már említettük, mi az elektronikus, digitális bináris rendszerrel foglalkozunk, vagyis a digitális technika elvei szerint felépített és kétállapotú logikai kapukat magában foglaló eszközzel.

A kérdés a következő, hogyan lehetséges aritmetikai, tehát számtani műveleteket (összeadás, kivonás, szozás és osztás) elvégezni logikai kapukkal, úgy, hogy még negativ számokkal is tudjon a rendszer dolgozni. Először nézzük meg két egyszámjegyű bináris szám összeadását (4.9. ábra):

	A	B	A+B	átvitel
0.	0	0	0	0
1.	0	1	1	0
2.	1	0	1	0
3.	1	1	0	1

4.9. ábra. Két egyszámjegyű bináris szám összeadása.

Mind A, mind B 0 és 1 értéket vehet fel, így tulajdonképpen 4 kombináció lehetséges, az összeadás eredménye az „A + B”, míg a keletkező átvitel az „átvitel oszlopban” látható. Ha most kielemezzük mindkét oszlop tartalmát, arra a következtetésre jutunk, hogy a logikai kapuk között az „A + B” oszlop tartalmával megegyező kimenettel rendelkezik a KIZÁRÓ VAGY kapu (2.2.4. fejezet), az „átvitel” oszlop tartalmával megegyező kimenetet ad a logikai ÉS kapu (2.2.1. fejezet). A KIZÁRÓ VAGY KAPU igazságtáblázata az 4.10. ábrán, míg az ÉS kapu igazságtáblázata az 4.11.. ábrán látható.

	A	B	A⊕B
0.	0	0	0
1.	0	1	1
2.	1	0	1
3.	1	1	0

4.10. ábra. KIZÁRÓ VAGY igazságtáblázata.

	A	B	A•B
0.	0	0	0
1.	0	1	0
2.	1	0	0
3.	1	1	1

4.11. ábra. ÉS kapu igazságtáblázata.

Tehát egy aritmetikai művelet, az öszeadás megvalósítható KIZÁRÓ VAGY és ÉS kapukkal. Továbbgondolva a logikai kapuk felépítését, láthatjuk, hogy a KIZÁRÓ VAGY kapu felírható és megvalósítható a következő egyenlettel:

$$A \oplus B = \bar{A} \cdot B + A \cdot \bar{B} \qquad (4.1)$$

Az úgynevezett De Morgan azonossággal ez a kifejezés átalakítható olyan alakká, ahol csak logikai szorzás (ÉS kapu) és INVERTÁLÁS van, vagyis, két egyjegyű bináris szám összeadása az analógia miatt (szám 0 és 1, valamint logikai állapot 0 és 1) logikai kapukkal (ÉS kapuk és INVERTEREK) megépíthető. A 2.1.1. fejezetben megismerkedtünk a negativ és pozitiv számok ábrázolásával, ezen belül a kettes komplemenssel. A kettes komplemens létrehozása egyszerű, minden bitet invertálunk, majd egyet adunk a számhoz, megvan a negativ érték. Ekkor már a kivonás összeadássá alakult át. Nézzünk meg egy példát:

3 – 2 = 3 + (-2).

Tehát a számítógépnek élég ha csak összeadni tud, hiszen a negítiv szám invertálással (INVERTEREK) és 1 hozzáadásával (ÖSSZEADÓ) megkapható. A szorzás is visszavezethető összeadásra, ismételt össszeadásra, az osztás pedig ismételt kivonásra.

A kettes számrendszer is helyiértékes, mint a tízes, itt is ugyanazok a szabályok vannak, így, ha nagyobb értékeket akarunk ábrázolni, akkor több bitet kell egymásssal összekapcsolni. A kisebb helyiérték átvitele (ami leht 0 vagy 1) befolyásolja az eggyel nagyobb helyiértéken történő bitek összeadását, ezért az úgynevezett teljes össszeadó A_i és B_i mellett még egy bemenetet, C_{i-1} -et is tartalmazza (4.12. ábra).

C_{be}	a	b	C_{ki}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

4.12. ábra. A teljes összeadó igazságtáblázata.

Az igazságtáblázatból kiírható a teljes összeadó egyenlete:

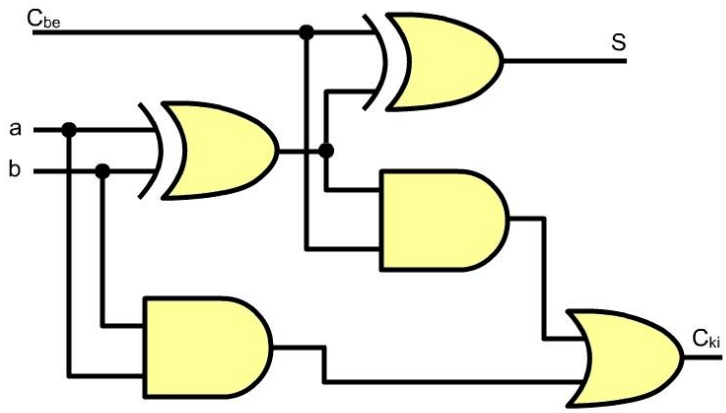
Az összeg:

$$S_i = \overline{C_{i-1}} \cdot \bar{A} \cdot B + \overline{C_{i-1}} A \cdot \bar{B} + C_{i-1} \cdot \bar{A} \cdot \bar{B} + C_{i-1} \cdot A \cdot B = A \oplus B \oplus C_{i-1} \qquad (4.2)$$

Az átvitel:

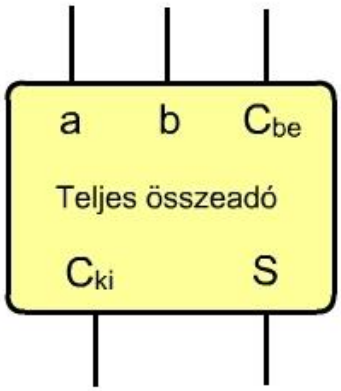
$$C_i = \overline{C_{i-1}} \cdot A \cdot B + C_{i-1} \cdot \bar{A} \cdot B + C_{i-1} \cdot A \cdot \bar{B} + C_{i-1} \cdot A \cdot B = (A \oplus B) \cdot C_{i-1} + A \cdot B \qquad (4.3)$$

Az áramkör logikai kapukkal az 4.13. ábrán látható.



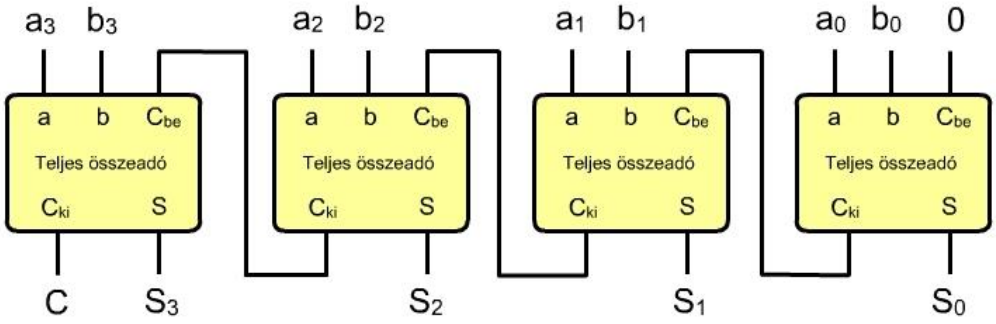
4.13. ábra. Teljes egy bites összeadó logikai sémája.

A teljes összeadó jelölése az 4.14. ábrán látható.



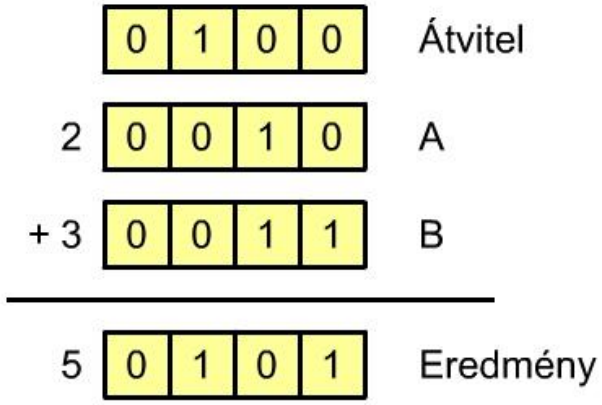
4.14. ábra. Teljes 1 bites összeadó jelölése.

A számokat bináris alakban a 2.1.1. fejezetben tárgyaltak szerint ábrázoljuk, egy 4 bites soros összeadó látható a következő 4.15. ábrán.



4.15. ábra. 4 bites soros összeadó.

Egy példán látható két szám összeadása 4 bites soros összeadóval:



4.16. ábra. Két bináris 4 bites szám összeadása, példa.

A soros összeadó legalacsonyabb helyiértékénél levő 1 bites teljes összeadójánál a bejövő átvitel 0, a többi számjegynél az előző helyiérték eredménye szerinti. Mivel az összeadók sorba vannak kapcsolva, ezért a következő helyiérték számolása csak a nála kisebb helyiérték átvitelének kiszámítása után lehetséges. Az összeadók késleltetése 1 ns – 10 ns nagyságrendben van, így ennek a megoldásnak az a hátránya, hogy nagyobb bit szóhosszúság esetén nagy a késleltetés. Ilyenkor jobb más megoldást, például párhuzamost alkalmazni.

Miután megvizsgáltuk, hogyan tud számolni a számítógép, nézzük meg most a processzor felépítését. Itt általános alapelveket tárgyalunk, a Neumann modellnél definálták szerint. A

mai processzoroknál ezeket az alapelveket felhasználva, de sok gyorsítási technikát alkalmazva tervezik a különböző processzorokat.

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

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

A hagyományos Neumann és Harvard struktúrájú mikrogépek nagyon hasonló belső architektúrával rendelkeznek, csak részletekben térnek el egymástól (3.1.5.1.1. fejezet és 3.1.5.1.2. fejezet).

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

Egy CISC gép főbb elemei a következők:

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

4.4.1. A VEZÉRLŐ EGYSÉG (CU – CONTROL UNIT)

Feladata a mikroszámítógép összes elemének működését összehangolni, szinkronizálni. Egyrészt az egyes elemi lépéseket egymás után végre kell hajtani ahhoz hogy elvégezze a mikrogép a feladatát, másrészt az elemek műveletvégzési sebességei különböznek, az egymás

közötti adatátvételt kell megfelelő időpillanatokban biztositania, vagyis időzítési problémákat kell megoldani.

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

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

Ez a legalapvetőbb két lépés egy utasítás végrehajtásakor, ezt lehet tovább finomítani, egyég lépéseket beiktatni, amik abban nyújtanak segítséget, hogy hogyan lehet párhuzamosítani a működést.

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

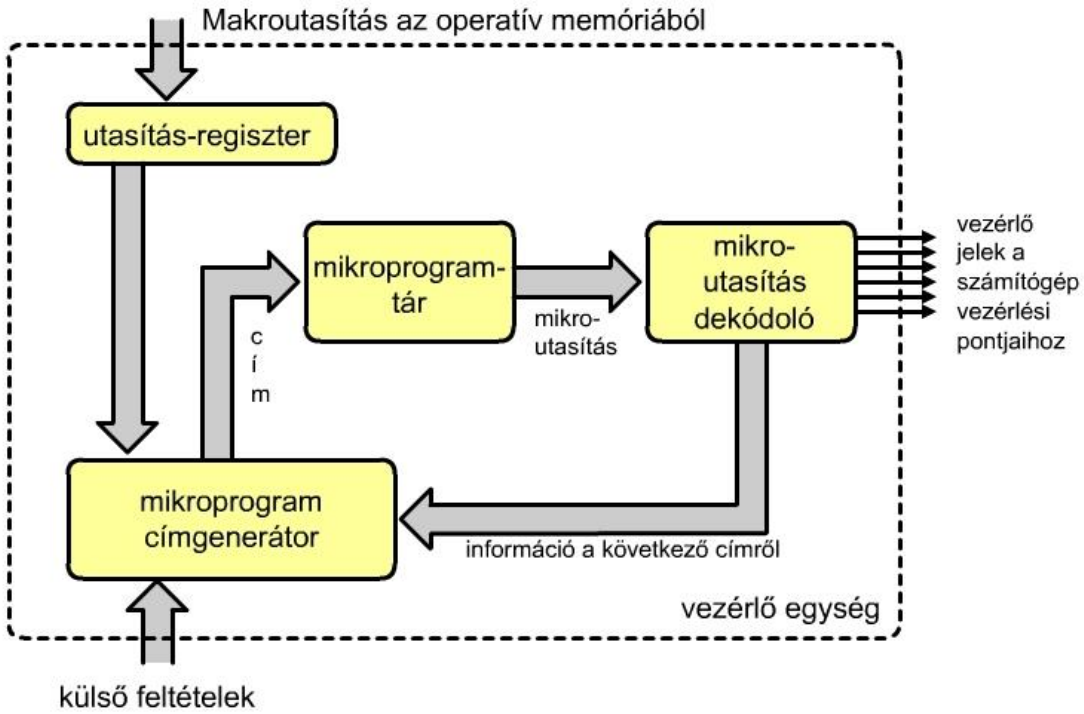
- huzalozott, vagy hardver kiépítés, amelynél az utasításvégrehajtás elemi lépéseit kombinációs és sorrendi áramkörök oldják meg, ahol fontos szerepet játszik a vezérlő jelek sorrendje mellett azok időzítése is. Ez a megoldás gyors vezérlőjel generálást jelent, de a rendszer merev, nem könnyű a módosítása,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

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

A vezérlőegység feladata a vezérlő jelek előállítása, amelyek lehetnek:

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

Az 4.17. ábrán egy olyan vezérlőegység látható, amely mikroprogram segítségével állítja elő a vezérlőjelek sorozatát az utasítás előkészítési fázisban IR utasításregiszterbe beolvasott műveleti kód alapján.

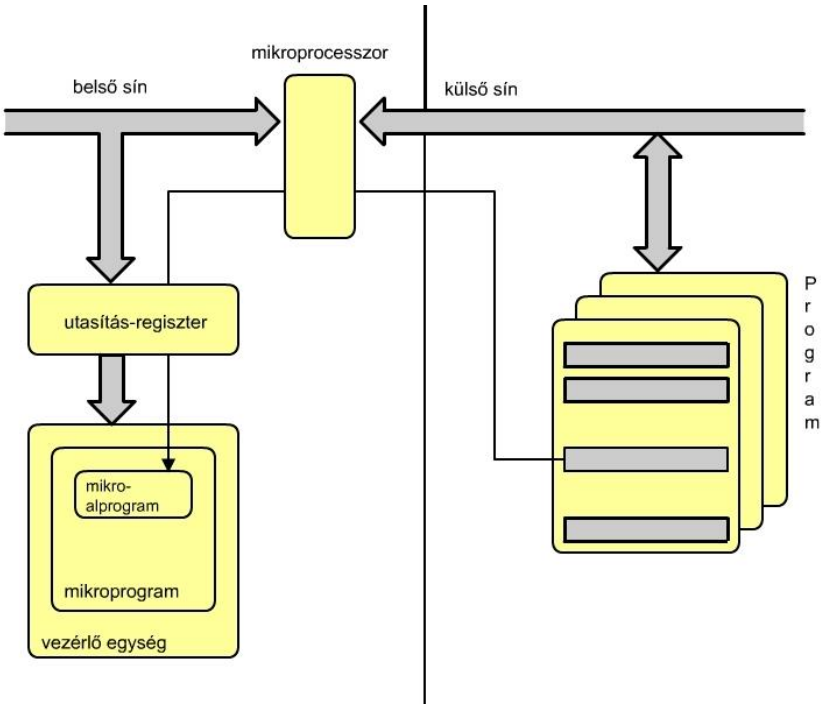


4.17. ábra. Mikroprogramozott vezérlőegység.

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

Egy (mikro)processzor, mikrovezérlő adat-, vagy utasítás szóhosszúsága független a mikroprogramtár szóhosszúságától.

A 4.18. ábrán látható a makroprogram (gépi kódú program) és a mikroprogram közötti kapcsolat. Látható az ábrán, hogy egy gépi kódú utasításhoz, végrehajtásához több alacsonyabb szintű mikroutasítás tartozik. A mikroutasítások száma függvénye a makroutasítás bonyolultságának. Ezeknél a mikroprogramoknál fellelhetők ugyanazok az elemek, mint a gépi kódú programoknál, alprogram, feltételes és feltétel nélküli ugrás, stb.



4.18. ábra. A makroprogram (gépi kódú progaram) kapcsolata a mikroprogrammal.

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

Az utasítás feladata az, hogy az A akkumulátor minden bitjét ellenkező értékűre állítja, ami egyidejűleg történik meg minden biten.

Az utasítás végrehajtását felbonthatjuk három lépésre, ezek a következők:

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

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

Ezek a megegyező részek ugyanazt az alprogramot használják.

A COM A utasításhoz tartozó mikroalprogram a következő alakú:

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

4.19. ábra: A COM A utasítás mikroalprogramja

Az egyes mezők jelentése:

- a – mező a vezérlő egységen belül ható mikroutasítást jelöli,
- b – az ALU aritmetikai-logikai egység belső elemeinek parancsszavai, például 100 a komplementáló egység működik, 111 azt jelenti, hogy az ALU nem aktiv, stb.),
- c – mező a célregiszter címét adja (0100 az A akkumulátor, 0010 a komplementáló egység, 0110 az IR utasításregiszter címrészének címe, 1001 pedig a PC programszámláló címe, stb. címe),
- d – mező a forrásregiszter címét adja,
- e – mező információt tartalmaz a következő utasítás címéről (pl. 000, következő cím) és

- f – megadja azt, hogy melyik külső feltétel hat ki a következő cím meghatározására (például 01 az átvitelbit, stb.).

Látható, hogy a COM A utasítás 3 darab18 bites mikroutasításból áll (nem számolva az utasítás lehívását biztosító részt, amelyik minden utasításnál ugyanaz)

Egy másik példa, elemezzük a feltétel nélküli ugrás mikroprogramját. Az utasítás mnemonikus kódja: JP cím, amely két részből áll, egy műveleti kódból és a cím bináris alakjából. Ez a két érték (mint egységes utasítás) kerül be az IR utasításregiszterbe, egyrészt a műveleti kód, másrészt cím alakban. Ez az utasítás nagyon egyszerű, mindössze a beolvasot címet kell a programszámlálóba áthelyezni:

Mikroprogramtár						
Cím	Tartalom					
	a	b	c	d	e	f
002B	00	111	1001	0110	000	00

4.20. ábra. A JP CIM utasítás mikroalprogramja.

Látható, hogy a művelet végrehajtása közben az ALU működése tiltott.

4.4.2. AZ ARITMETIKAI-LOGIKAI EGYSÉG (ALU – ARITMETIC LOGIC UNIT)

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

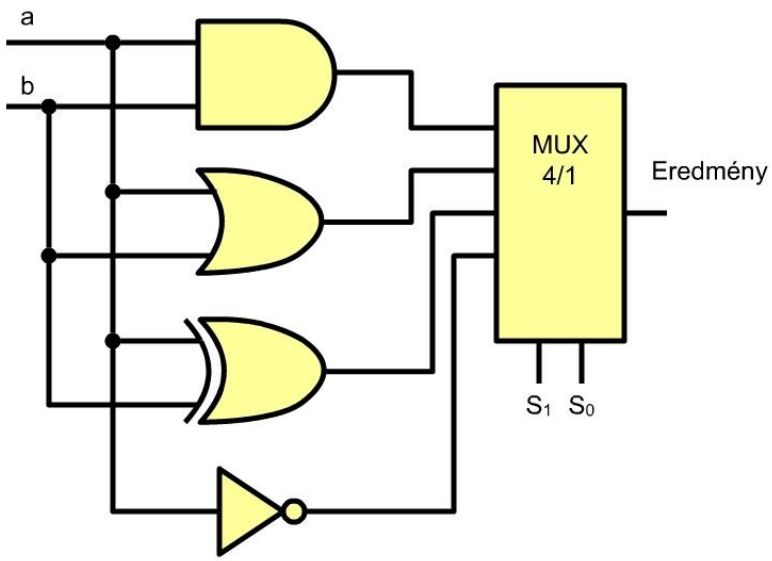
- komplementálás (minden bit ellenkező értéket vesz fel),
- léptetés (általában léptetés és körbeforgatás),
- összeadás (2 operandus, egyszerűbb processzoroknál bináris, fixpontos),
- inkrementálás (az érték növelése eggyel),
- dekrementálás (az érték csökkentése eggyel) és
- átvitel.

Vizsgáljuk meg az ALU logikai részét, hogyan épül fel és működik. Természetesen itt csak egy egyszerű példát elemzünk. Legyen a feladat egy olyan hardver megtervezése és megépítése, amelyik két bit A és B között végez logikai műveletet, ÉS, VAGY, KIZÁRÓ VAGY és INVERTÁLÁS. A négy feladat igazságtáblázata a következő:

S_1	S_0	Y
0	0	ab
0	0	a + b
0	1	$a \oplus b$
0	1	$\bar{a}$

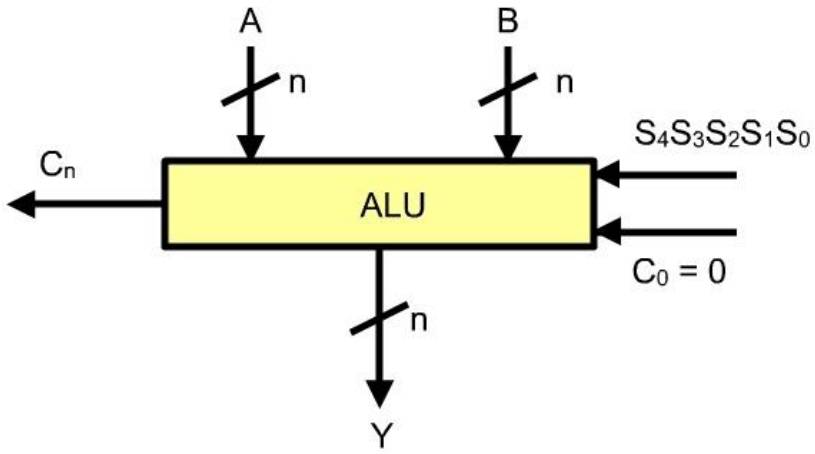
4.21. ábra. Négy logikai feladat igazságtáblázata.

Mivel egyszerre csak egy logikai műveletet kell elvégezni, ezért egy multiplexerrel lehet ezt megoldani:



4.22. ábra. A négy logikai függvény és a multiplexer.

A következőkben egy 16 aritmetikai és 16 logikai műveletet végrehajtó ALU felépítését elemezhetjük. Az összesen 32 művelet olyan multiplexerrel oldható meg, amelyiknek 5 SELECT vezetéke van:



4.23. ábra. 32 (16 aritmetikai és 16 logikai) művelet végrehajtása.

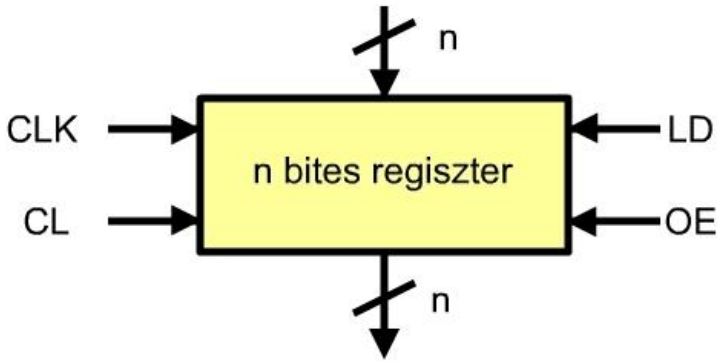
Az áramkör igazságtáblázata:

S_3	S_2	S_1	S_0	$S_4 = 1$	$S_4 = 0$
0	0	0	0	$Y = \overline{A}$	$Y = A$ plus C_0
0	0	0	1	$Y = \overline{A + B}$	$Y = (A + B)$ plus C_0
0	0	1	0	$Y = \overline{A}B$	$Y = (A + \overline{B})$ plus C_0
0	0	1	1	$Y = 0000$	$Y = 0$ minus C_0
0	1	0	0	$Y = \overline{AB}$	$Y = A$ plus $(A\overline{B})$ plus C_0
0	1	0	1	$Y = \overline{B}$	$Y = (A + B)$ plus $(A\overline{B})$ plus C_0
0	1	1	0	$Y = A \oplus B$	$Y = A$ minus B minus $\overline{C_0}$
0	1	1	1	$Y = A\overline{B}$	$Y = (A\overline{B})$ minus $\overline{C_0}$
1	0	0	0	$Y = \overline{A} + B$	$Y = A$ plus (AB) plus C_0
1	0	0	1	$Y = \overline{A \oplus B}$	$Y = A$ plus B plus C_0
1	0	1	0	$Y = B$	$Y = (A + \overline{B})$ plus (AB) plus C_0
1	0	1	1	$Y = AB$	$Y = (AB)$ minus $\overline{C_0}$
1	1	0	0	$Y = 1111$	$Y = A$ plus A plus C_0
1	1	0	1	$Y = A + \overline{B}$	$Y = (A + B)$ plus A plus C_0
1	1	1	0	$Y = A + B$	$Y = (A + \overline{B})$ plus A plus C_0
1	1	1	1	$Y = A$	$Y = A$ minus C_0

4.24. ábra. 32 (16 aritmetikai és 16 logikai) művelet igazságtáblázata.

Az ALU-ban kombinációs hálózati megoldások mellett sorrendi elemek is szerepelnek, a regiszterek.

A párhuzamos regiszter:



4.25. ábra. Párhuzamos beírású és olvasható regiszter

Ahol:

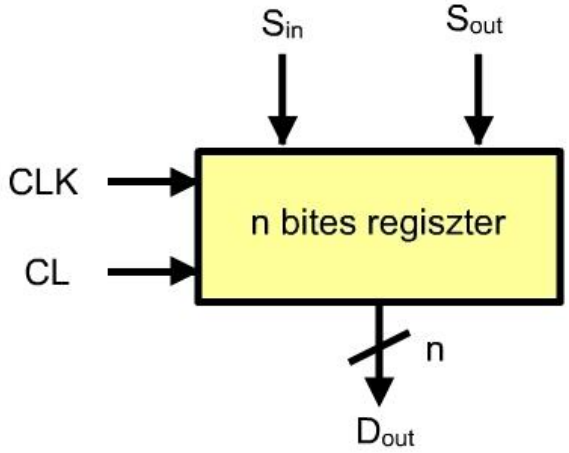
CLK (CLOCK) – ÓRAJEL

CL (CLEAR) - TÖRLÉS

LD (LOAD) - BEÍRÁS

OE (OUTPUT ENABLE) - KIOLVASÁS

Soros léptető regiszter:



4.26. ábra. Soros léptető regiszter.

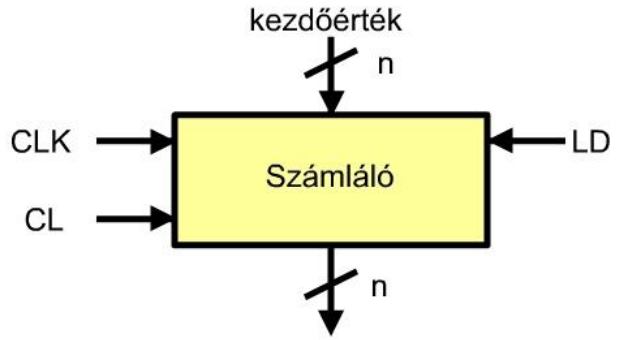
Ahol:

Sin (SERIAL IN) – SOROS ADATBEMENET

Sout (SERIAL OUT) – SOROS ADATKIMENET

Dout (DATA OUT) – PÁRHUZAMOS ADATKIMENET

Számlálók:



4.27. ábra. Számláló szimbolikus jelölése.

Ahol:

CLK (CLOCK) – ÓRAJEL

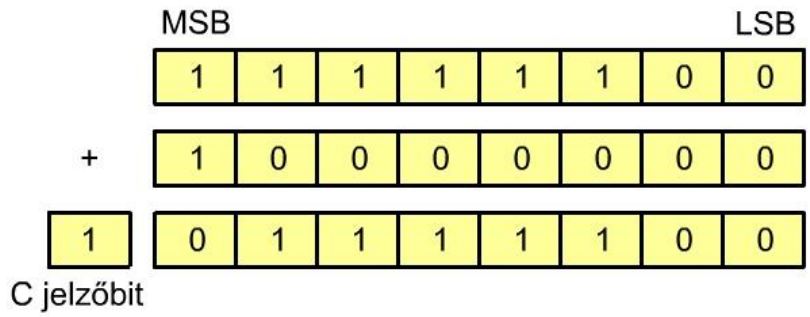
CL (CLEAR) - TÖRLÉS

LD (LOAD) - BEÍRÁS

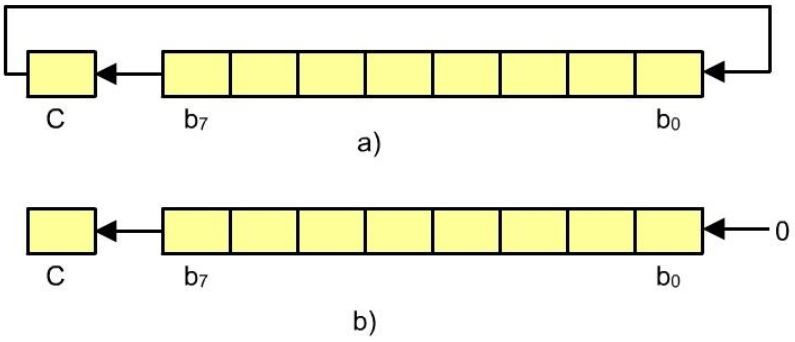
Az aritmetikai-logikai egységhez kapcsolódik több bistabil (flip-flop) amelyek a műveletvégzés eredményétől függően tárolják a kapott eredmény jellemzőit. Ezeket nevezik jelzőbiteknek is, vagy angolul a zászló, flag szóval. Összefogva a bistabilokat egy állapotjelző regiszter keletkezik. Az állapotregiszter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

A (mikro)processzoroknál, mikrovezérlőknél leggyakrabban használt jelzőbitek a következők:

- C – átvitel (carry) jelzőbit, kéttős szerepe van:
  - Ha az eredmény legmagasabb helyértékén átvitel keletkezik C értéke 1-re állítódik, erre példa a 4.28. ábrán látható. A D7 helyen történő átvitel D8 helyre kerülne, de ilyen nem létezik egy 8 bites regiszternél, e helyett az átvitel a C átvitelbitbe kerül. Ezt bizonyos utasításokkal vizsgálhatjuk, C értéke alapján elágazást (feltételes ugrást) lehet létrehozni a programban.
  - A C jelzőbitet használhatjuk léptetésnél és forgatásnál is, amire példát a 4.29. ábrán láthatunk.

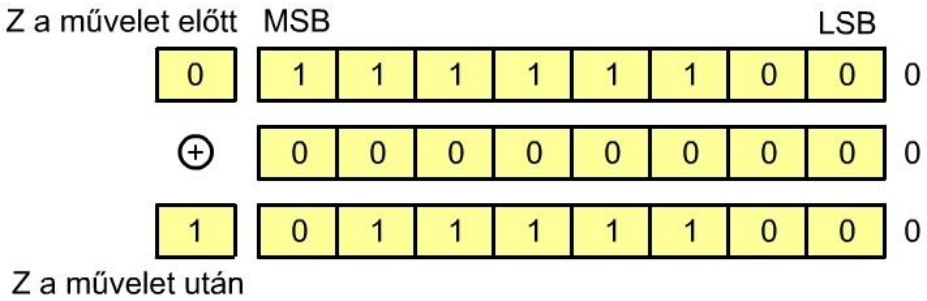


4.28. ábra. Példa olyan összeadásra, ahol átvitel keletkezik.



4.29. ábra. Körbeforgatás (rotálás) **a**) és léptetés balra **b**)

- V – túlcsordulás (overflow) jelzőbit, a V = 1 azt jelzi, hogy egy matematikai művelet eredménye nem értelmezhető, hibás eredmény keletkezett. Ez matematikailag a kettes komplemens előjelbit hibája: $V = C_s \oplus C_p$, ahol C_p az előjelbitbe az átvitel, vagyis b7, C_s pedig az előjelbitből való átvitel. Tulajdonképpen egy nyolcbites regiszternél a művelet eredménye kilépett a –128 , + 127 tartományból.
- N - a negatív számot jelöli, csak előjeles számábrázolásnál, kettes komplemensnél van értelme, megegyezik a jelzőbit értéke megegyezik b7 értékével.
- Z – jelzi, hogy a regiszter tartalma 0 (Z = 1), illetve nem nulla, bármi más (Z = 0). Értéke aritmetikai műveletek, valamint összehasonlító utasítások végrehajtása után állítódik, adatmozgatás után nem változik értéke a tartalomtól függően. A 4.30. ábrán a példában, a kizáró vagy művelet elvégzése utáni állapot látható.
- H – 'félátvitel', tulajdonképpen ugyanaz, mint a C átvitel, de 4 bit után, vagyis b3-ról b4-re. Mivel a BCD (Binary-Coded Decimal) számokat 4 biten lehet ábrázolni, ezért szükséges figyelni az átvitel jelentkezését. A BCD számoknál 9 után újból 0 következik (és egy átvitel), ugyanakkor a 4 bit 16 különböző számot ábrázolhat, de ebből csak a 0 és 9 közé eső bináris kombinációknak van értelme.

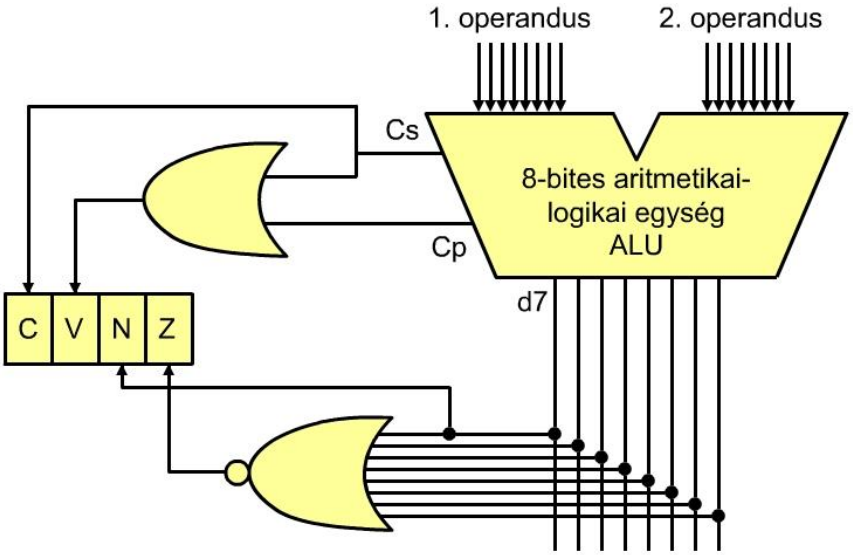


4.30. ábra. Mintapélda a Z jelzőbit beállítására (kizáró vagy kapuval).

A fent felsorolt jelzőbiteken kívül processzortípustól függően találhatunk még egyéb

indikátorbiteket, ez mindig az adott géptől függ.

A 4.31. ábrán az ALU aritmetika-logikai egység és a különböző jelzőbitek közötti kapcsolat látható.



4.31. ábra. A jelzőbitek és az aritmetikai logikai egység kapcsolata.

4.4.3. AZ A AKKUMULÁTOR ÉS AZ ÁLTALÁNOS CÉLÚ REGISZTEREK

Mint már a (mikro)processzor modellnél is látható volt, a processzor egyik regisztere, az A akkumulátor kitüntetett szerepet játszik a műveletek végrehajtásának nagy részénél. Mint már említettük, bizonyos esetekben munka regiszter a neve, W (Work Register). Az akkumulátor mellett a processzor tartalmaz még néhány regisztert, amelyek száma igen eltérő lehet típustól függően, néhánytól néhány százig terjedhet számuk (gyakorlatilag a programozó számára a CISC processzoroknál néhány érhető el, míg a RISC processzorokra jellemző a nagyobb szám). A regiszterek adatelérési ideje az 1 ns - 10 ns nagyságrendben van, míg az operatív memóriához való hozzáférés kb. egy nagyságrenddel lassúbb. Rendszerint több regiszter egy processzoron belüli memóriát alkot. A műveletek egy része hasonló módon használja ezeket a regisztereket, mint az akkumulátort, de korlátozottak a lehetőségek. Például a C jelzőbitet nemcsak az A akkumulátorral végzett műveletek állíthatják, hanem bizonyos általános célú regiszteren végzett művelet is módosíthatja az értékét. Az adott processzor utasításkészlete tartalmazza ezeknek a lehetőségeknek a leírását.

4.4.4. CÍMREGISZTEREK

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

A következő címregiszterek találhatók meg egy processzorban:

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

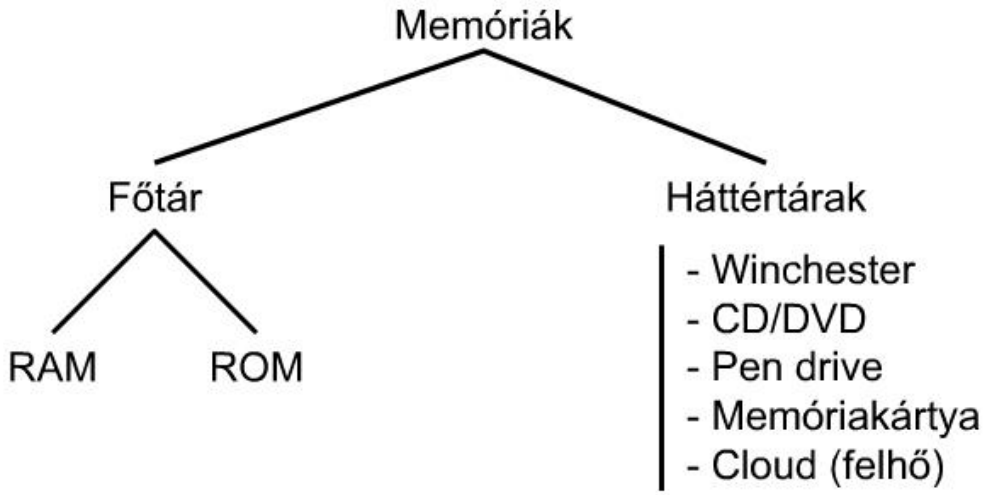
Sok processzornál az a lehetőség is fennáll, hogy valamilyen általános célú regiszter címzésre is használható.

5. MEMÓRIÁK

A digitális bináris elektronikus rendszerek elterjedése az analóg elektronikával szemben számottevő, ennek két fő oka van. Az egyik az, hogy az ábrázolt mennyiségek közötti arány (a pontosság) analóg rendszereknél 1 : 1000 lehet, míg a kétállapotú számábrázolásnál az érték nagysága, illetve a pontosság a megfelelő számú bit meghatározásával érhető el. A másik ok az adat- és utasítástárolás hatékonysága. Analóg elektronikai eszközökkel ez gyakorlatilag lehetetlen, míg a bináris logika sorrendi elemei, a flip-flopok (2. fejezet) alkalmasak egy bit tárolására (0 vagy 1).

Tekintettel arra, hogy az egyes számítógép-rendszerek teljesítményben, méretben, bonyolultságban eléggé eltérnek egymástól, maga az adat- és programtárolás is lényeges különbségeket mutat egyrészt méretben, másrészt abban, hogy fizikailag hol helyezkedik el az adott tár (memória).

Több szempont szerint is elemezhetjük a memóriákat, itt most aszerint tekintjük át ezeket az elemeket, hogy közvetlen, gyors elérést biztosítanak a processzor számára, vagy nagy mennyiségű adatot, programot tárolnak, igaz lassúbb eléréssel.



5.1. ábra. Memóriák felosztása elérés szerint.

Minden számítógép-rendszer rendelkezik főtárral, az ábra jobb oldalán felsorolt háttértárak pedig részei a közeopes, illetve nagy gépeknek. A számítógép technika fejlődése során többféle megoldás jelent meg az adatok, utasítások tárolására, ebből néhány tartósan megmaradt, néhány pedig túlhaladottá vált, gyakorlati alkalmázra alkalmatlan.

A mai adattárolási technikákra jellemző a szilícium alapú félvezető megoldás, a mágneses elv és az optika.

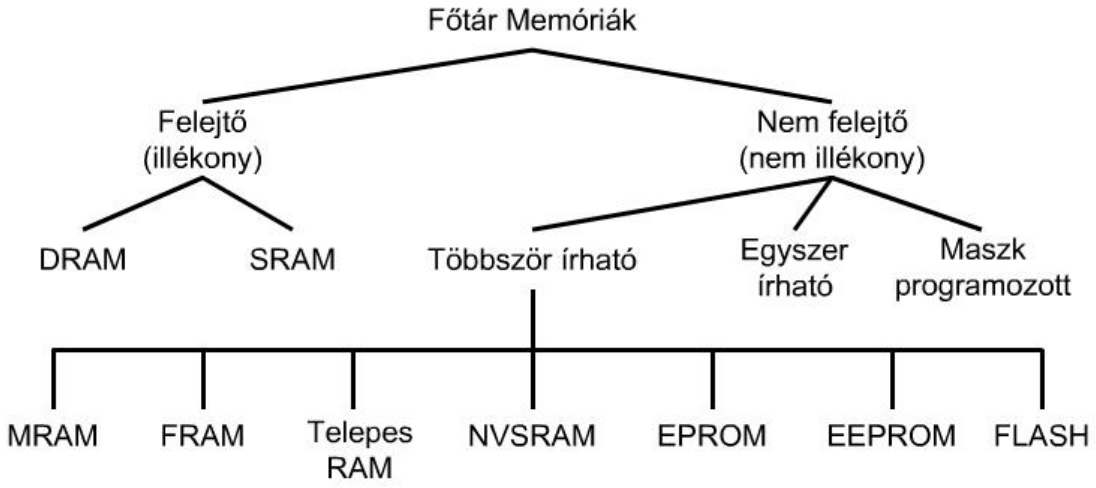
Milyen elvárások vannak a memóriákkal szemben?

- Tekintettel a számítógép működési elvére, a bináris logikára, ugyanilyen alakban célszerű tárolni az adatokat, vagyis bináris elektronikus úton.
- Célszerű, fóleg a méretek miatt a nagy adatsűrűség biztosítása, vagyis minél kisebb felületen (köbtartalomban) a lehető legtöbb cellát kialakítani.
- Legyen az adttárolás tartós, tehát addig őrizze a hordozó az információt, ameddig a felhasználás során szükséges azt.
- Az információ hordozóra való írása a lehető legrövidebb ideig tartson.
- Kiolvasáskor is legyen a kiolvasási idő a lehető legrövidebb.
- Elvárjuk az adathordozótól, hogy nagyszámú újraírást bírjon ki, mielőtt meghibásodik.
- Az egy bitre jutó költség a lehető legkisebb legyen.
- Szükséges a fogyasztás minimalizálslása. Itt sokszor megkülönböztetünk normál használatot és ideiglenes használaton kívüli úgynevezett „stand by”, vagyis rendelkezésre álló állapototot.

Egyértelmű, hogy a fenti feltételeket egyetlen megoldás sem biztosítja. Valamilyen elvek szerint kompromisszumos megoldást kell találni.

Mivel a processzorhoz fizikailag és logikailag a legközelebbi memória a főtár, vagy más néven az operatív tár, először ezzel foglalkozunk. Itt jegyezzük meg, hogy a gyorsítási technikáknál használják a cache nevű gyorsító tárat, amelyek a processzor és a főtár közé kerül, de ezzel később foglalkozunk az 5.7. és 5.8. fejezetben (kisebb teljesítményű processzoroknál, mikrovezérlőknél természetesen nem alkalmazzák).

A főtár logikailag egy egységesen címezhető, minden cellát (memóriarekeszt) ugyanannyi idő alatt elérő (írás/olvasás) része a rendszernek, függetlenül attól, hogy fizikailag eltérő elven megépített kisebb egységekből építjük fel. Vizsgéljuk meg először azt, a technika, technológia mai állása szerint mely megoldások jöhetnek számba. A következő ábrán azok a megoldások láthatók, melyek számba jöhetnek egy processzor főtárának kialakításánál, bár fizikai és használati paramétereikben van különbség.



5.2. ábra. Főtárban alkalmazható memória technológiák

A fenti ábrán döntően félvezető technológiában megépített memóriák vannak ábrázolva. Csekély jelentőségük miatt nem foglalkozunk három típussal, ez az MRAM, mely mágneses elven működik, szintén nem tárgyaljuk a ferromágneses FRAM és a két dielektrikumos NVSRAM típusokat.

Két fő csoportot különböztetünk meg, ezek:

- A felejtő memóriák (illékony memóriák) – a beírt információ a tápfeszültség lekapcsolásával elvész, új tartalom normál használat közben kerül a cellákba, ahonnan bármikor kiolvasható – és
- Nem felejtő memóriák (nem illékony memóriák) – programozó készülékkel lehet új tartalmat beírni, egyes típusoknál pedig a lapraszerelt integrált áramkörbe működés közben, tápfeszültség kikapcsolása után is megtartja a beírt tartalmat.

5.1. RAM (RANDOM ACCESS MEMORY) – VÉLETLEN ELÉRÉSŰ MEMÓRIÁK

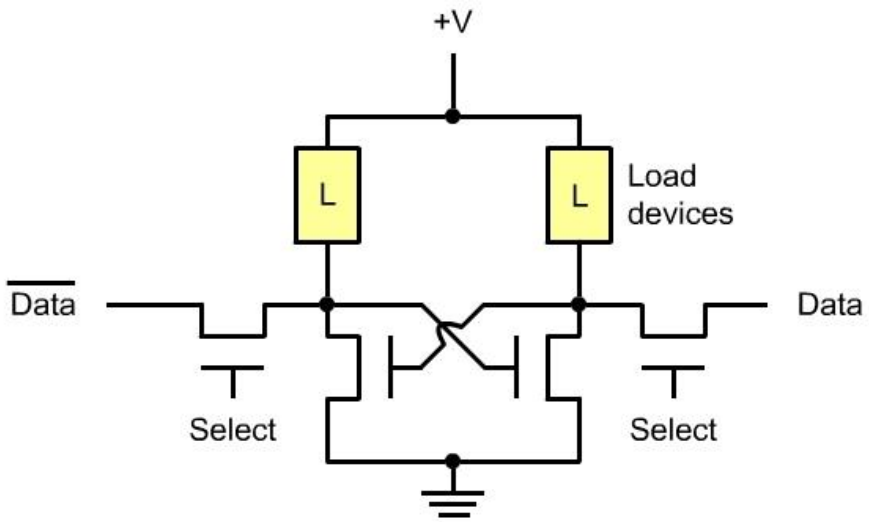
RAM (Random Access Memory) - tetszőleges (közvetlen, véletlen) hozzáférésű memória:

- közvetlenül írható és olvasható, adatok átmeneti tárolására szolgál.
- Információ tartalmuk a tápfeszültség megszűnésével elvész.

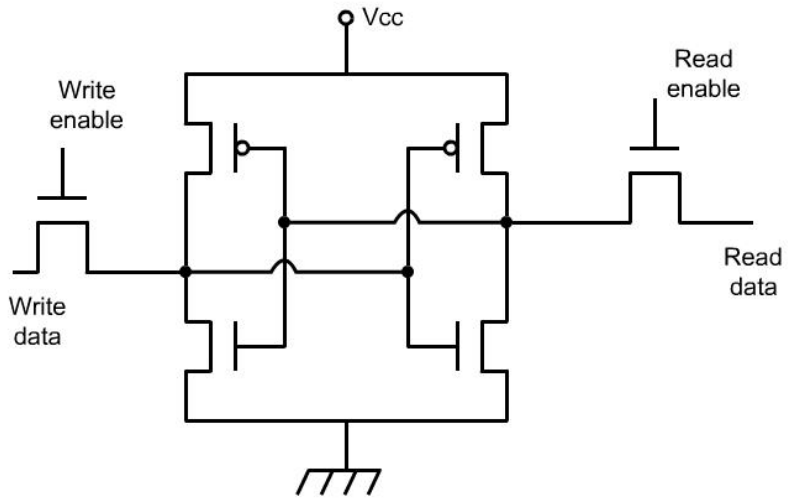
A közvetlen, vagy véletlen elérés kifejezés megtévesztő, valójában a lényeg abban van, hogy a memória teljes területe ugyanúgy és ugyanannyi idő alatt érhető el, akár írás, akár olvasás céljából. Két típusa van:

- statikus RAM (tároló elem: flip-flop (RS tároló)),
- dinamikus RAM (tároló eleme: kondenzátor).

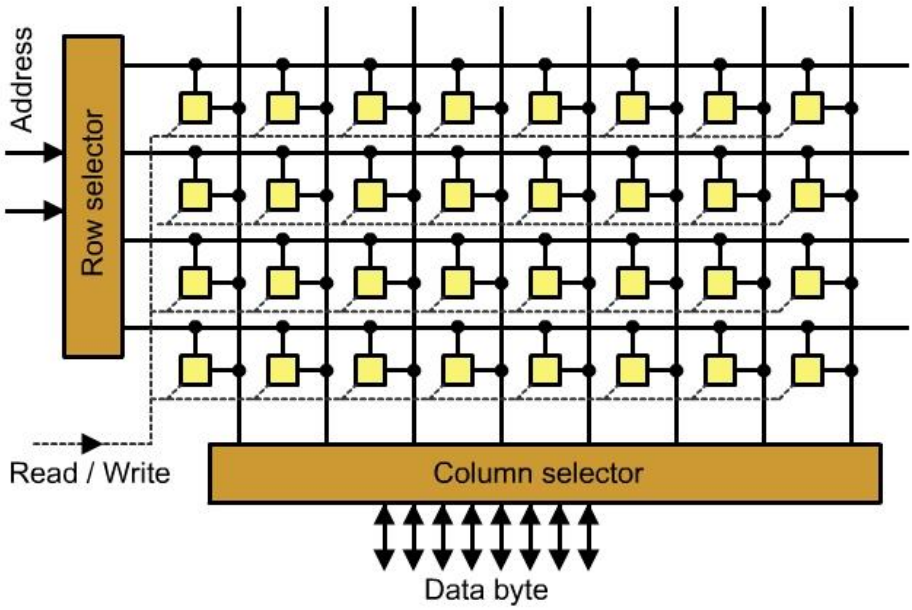
5.1.1. STATIKUS RAM MEMÓRIÁK



5.3. ábra. Statikus RAM kapcsolási rajza 4 tranzisztorral, 2 ellenállással.



5.4. ábra. Statikus RAM kapcsolási rajza 6 tranzisztorral.



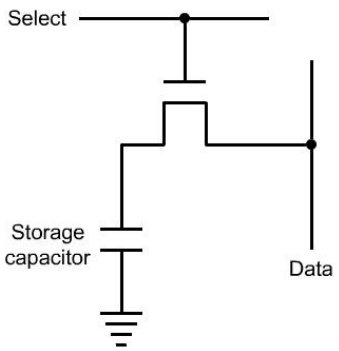
5.5. ábra. Statikus RAM szervezése.

A memória tulajdonságai:

- statikus RAM (tároló elem: flip-flop (RS tároló, 4-6 tranzisztor)),
- kicsi adatsűrűség (viszonylag sok elem miatt),
- nagy sebesség (elérési idő), 1 ns – 100 ns,

- nagy megbízhatóság,
- kis fogyasztás,
- rossz kapacitás/ár-viszony,
- elsősorban gyorsító (cache) memóriaként használják.

5.1.2. DINAMIKUS RAM MEMÓRIÁK



5.6. ábra. Dinamikus RAM memória kapcsolási rajza

- dinamikus RAM (tároló elem: kondenzátor-tranzisztor),
- nagy adatsűrűség (viszonylag kevés elem miatt),
- kis sebesség (elérési idő), 10 ns – 200 ns,
- kis megbízhatóság,
- kis fogyasztás,
- jó kapacitás/ár viszony,
- 2 ms – 4 ms frissítés

A két memória típus összehasonlítása a következő táblázatban látható:

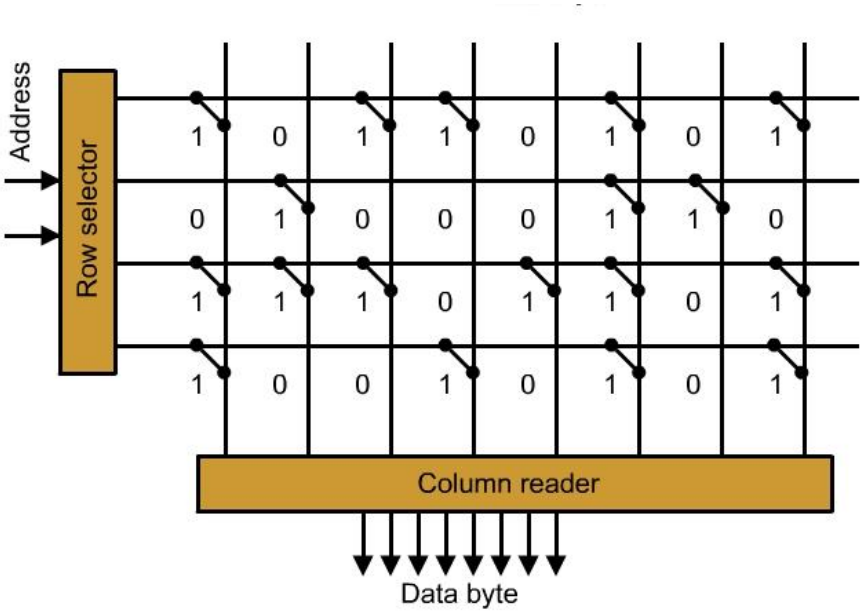
5.1. táblázat. A statikus és dinamikus RAM összehasonlítása		
TULAJDONSÁG	STATIKUS RAM	DINAMIKUS RAM

sebesség	jóval gyorsabb	
kapacitás		lényegesen nagyobb
fogyasztás	kevesebbet fogyaszt	
ár		olcsóbb
a hiba valószínűsége	megbízhatóbb	
írások/olvasások száma	gyakorlatilag végtelen	gyakorlatilag végtelen

5.2. ROM (READ ONLY MEMORY) – CSAK OLVASHATÓ (NEM FELEJTŐ) MEMÓRIÁK

A csak olvasható memóriák csoportjába tartoznak, tartalmuk nem vész el a tápfeszültség megszűntekor. A következő jellegzetes változatokkal találkozhatunk:

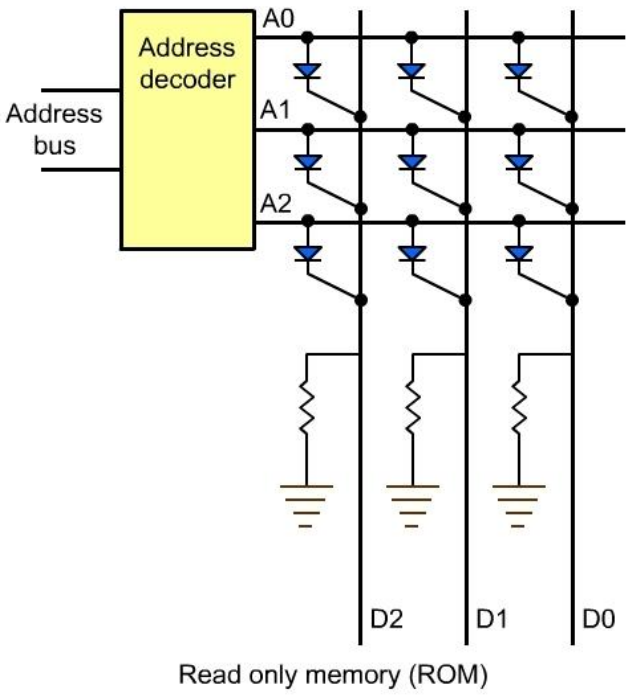
Maszkprogramozott ROM – gyártáskor alakítják ki a memória-mátrixot, nagy sorozatban gyártott elemeknél nagyon olcsók. Az ábrán látható, hogy a sorok-oszlopok metszéspontjaiban a szakadás 0, a rövidzár 1 értéket ad.



5.7. ábra. Maszkprogramozott ROM felépítése.

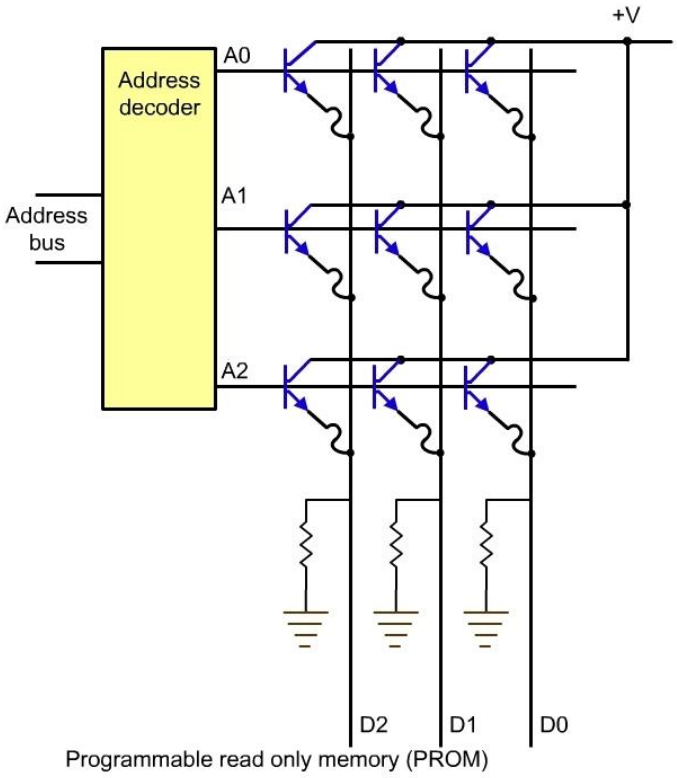
Mivel így rövidzár keletkezne, ezért diódával kell ezt meggátolni, a következő ábra szerinti

módon:



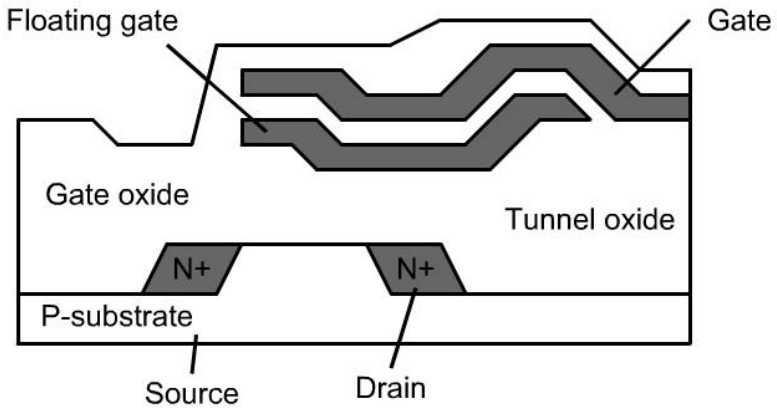
5.8. ábra. Maszkprogramozott ROM memória diódákkal.

Programozható ROM – (PROM - Programmable Read Only Memory) – programozó készülékkel írható a tartalmuk. Az információ beírása egy fémből készült biztosíték, „fuse” kiégetésével történik. A biztosíték anyaga NICr, Ti, W, Pt, A programozáshoz, az átégetéshez 5 – 20 mA áram és 10..15 V feszültség kell, általában bipoláris áramkörökben alkalmazzák ezt a technológiát. A programozás után az eredeti, üres állapot már nem állítható vissza.

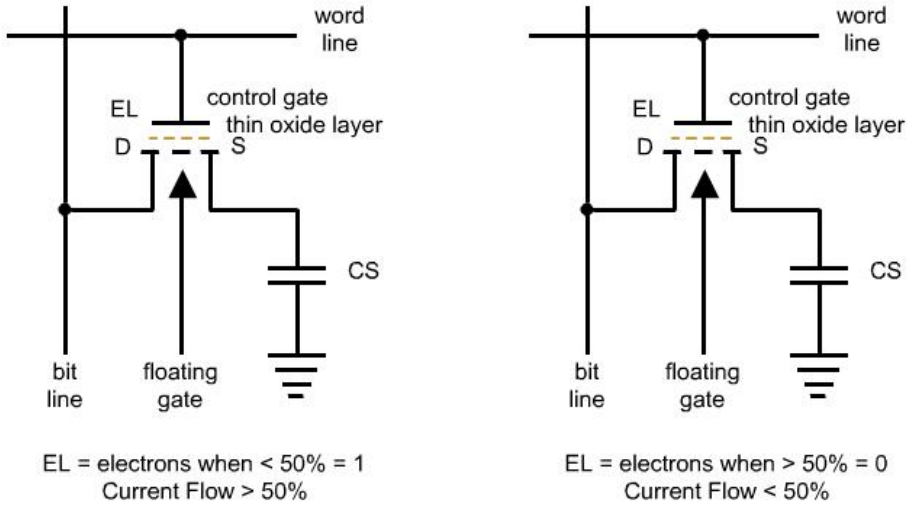


5.9. ábra. Programozható ROM, PROM memória.

Írható – törölhető ROM (EPROM - Erasable Programmable Read Only Memory) – programozó készülékkel elektronikusan írható, törlése egy meghatározott hullámhosszú, ultraibolya fénnyel lehetséges. Az írás-olvasás lehetőségét az úgynevezett lebegő GATE teszi lehetővé, mint ahogy a következő ábrán látható, a szigetelő rétegben egy kis elzárt vezető van. Programozáskor a source-t földeljük, a gate-re és a drainre +25 V feszültséget kapcsolunk, aminek hatására lavinaletörés jön létre a csatornában, a nagy energiájú elektronok pedig keresztüljutnak az oxid potenciálgátján (3.2eV) és a lebegő elektródára kerülnek,ezután a lebegő elektródán lévő negatív töltés ott marad, ezáltal a tranzisztor küszöbfeszültsége megnő, és akkor sem nyit ki, ha a gate –re tápfeszültséget kapcsolunk.



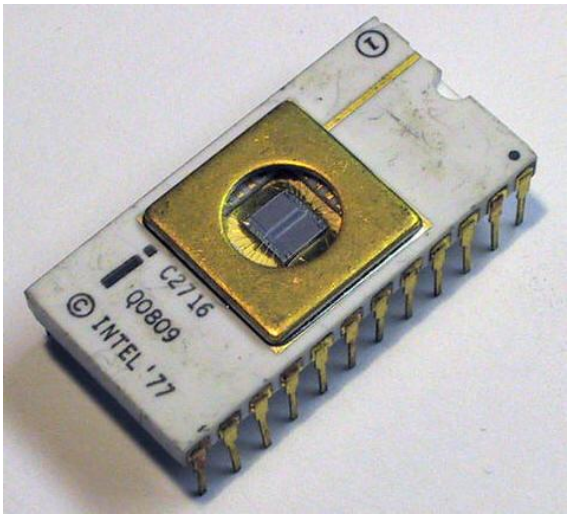
5.10. ábra. Írható-törölhető ROM, EPROM keresztmetszete.



5.11. ábra. Írható-törölhető ROM, EPROM memória működése.

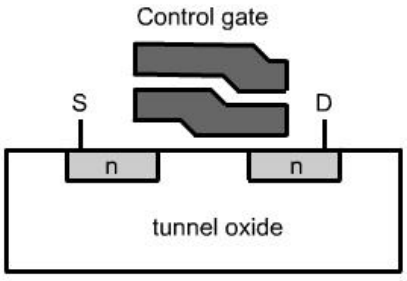
A memória tartalmának törlése egy az integrált áramkörre épített üvegablakon keresztül történik ultraibolya fénnyel. A fény hatására a csapdába ejtett elektronok kikerülnek a lebegő gate-ből, aminek hatására visszaáll az eredeti állapot.

Az írás-törlés folyamat csak korlátozott számban ismételhető, egy bizonyos számú írás-törlés után a szigetelő károsodik és tönkremegy. Adatot a ciklusszámra az adott típusnál találhatunk. Ez a memória még érzékeny a statikus elektromosságra is.

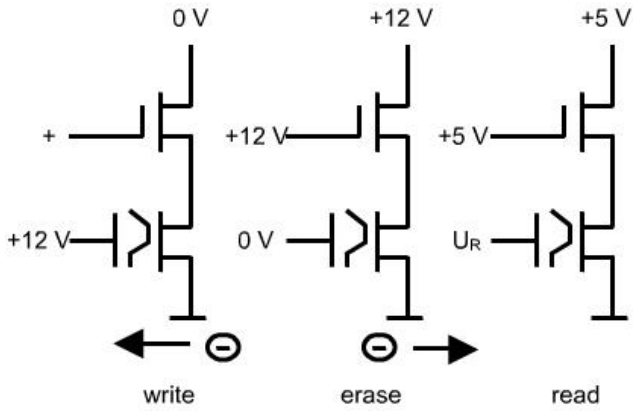


5.12. ábra. EPROM memória.

Írható – törölhető ROM (EEPROM - (**E**lectrically **E**rasable **P**rogrammable **ROM**) – programozó készülékkel elektronikusan írható, a tartalom törlése szintén elektronikusan lehetséges. Maga az írás általában jóval lassúbb folyama, mint az olvasás. Az írás-olvasás lehetőségét az úgynevezett lebegő GATE teszi lehetővé, mint ahogy a következő ábrán látható, a szigetelő rétegben egy kis elzárt vezető van. Programozáskor az alagút- hatást használjuk ki, vékony, tökéletesen szigetelő, feszültségbíró és töltésmentes oxidrétegben levő GATE-re, az elektronok az alkalmazott feszültség polaritásától függően mindkét irányból át tudnak „tunnelezni”.



5.13. ábra. Írható-törölhető ROM, EPROM memória keresztmetszete.



5.14. ábra. Írható-törölhető ROM, EEPROM memória működése.

Az írás-törlés folyamat csak korlátozott számban ismételhető, egy bizonyos számú írás-törlés után a szigetelő károsodik és tönkremegy. Adatot a ciklusszámra az adott típusnál találhatunk.

Mindamellett, hogy programozó készülékkel is felprogramozható a csip, lehetséges a nyomtatott áramkörön is (nem kivéve abból) működés közben átprogramozni. Ez nagy előny az adatok tartós átírására, nincs szükség külön készülékre.

FLASH memória – megegyezik az előző EEPROM memóriával, de szervezése nem bájtonkénti, hanem nagyobb blokkokban van (ennek programozáskor van jelentősége).

A programozó készülék

PROM, EPROM és EEPROM memóriák programozására szolgál, EEPROM esetében tartalomtörlés is lehetséges.

5.15. ábra. A Microchip cég PROM, EPROM, EEPROM és FLASH programozója

5.3. A FŐTÁR MEMÓRIATÍPUSAI

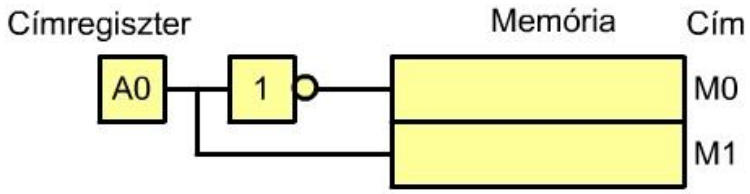
A kisebb rendszereknél nincs háttértár, egy, vagy néhány egyszerű program fut, nagyon egyszerű operációs rendszer, vagy monitor alatt. Itt kifejezetten célprogramok találhatók, például egy szabályozó, irányító rendszer, adott feladathoz speciális célprogram. Ezeknél a megoldásoknál a címezhető terület valamilyen feltételek alapján RAM és ROM részből áll, ezek logikailag egy közös, címezhető területet jelentenek. Természetesen a ROM memóriában van a program. A bonyolultabb, nagyob teljesítményű rendszereknél már megjelenik a háttértár, hiszen itt sokszor cserélgetjük a programokat, ilyenkor inkább a RAM memóriának van nagyobb szerepe.

A program állandóan a főtár valamelyik rekeszére hivatkozik, egy adott szóhosszúságú címmel. Mivel egy rendszeren belül egy egységnek (most memóriarekesznek, cellának) csak egy címe lehet, valamint egy cím hatására csak egy egységnek szabad aktívvá válni, a kiküldött címet egy logikának azonosítania kell és ha megegyezik az egység címével, azt aktívvá kell tennie. Erre szolgál a címdekódolás.

5.4. MEMÓRIA VÁLASZTÓ LOGIKA (TELJES CÍMDEKÓDOLÁS)

A Neumann típusú mikroszámítógépnél egy közös memória (főtár) tartalmazza a programot és az adatokat, változókat és ugyanennek a közös memóriának lehet a része a verem (stack) memória is. A Harvard típusú számítógépnél már külön memóriában található a program, más memóriában az adatok. A memóriában levő rekeszeket, amelyek több bit szóhosszúságúak (pl. 8, 16 vagy 32 stb.) címezni kell.

Induljunk el az alapoknál. Ha 1 bit szolgál a címzésre, akkor azzal két különböző rekesz (vagy más elem, periféria) különböztethető meg (5.16. ábra). Az így létrehozott címválasztó-logika igazságtáblázata a 5.17. ábrán található.

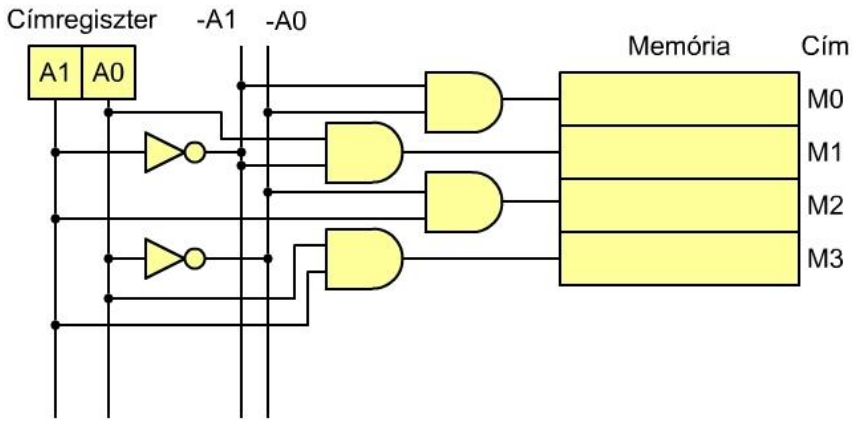


5.16. ábra. Egy bit szóhosszúságú címregiszter és címzési tartománya.

A0	M1	M0
0	0	1
1	1	0

5.17. ábra. Egybites cím igazságtáblázata.

Két bit szóhosszúságú címregiszter már 4 különböző címet állit elő (2^2), ennek logikai rajza az 5.18. ábrán látható és igazságtáblázata az 5.19. ábrán látható.



5.18. ábra. Két bit szóhosszúságú címregiszter és címzési tartománya.

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

5.19. ábra. Kétbites cím igazságtáblázata.

A fentiekből következik, hogy tetszőleges szóhosszúságú címregiszterhez megtervezhető a címválasztó logika, a címzett terület nagysága pedig a következő képlettel számítható ki:

$$\text{összes cím} = 2^n \qquad (5.1.)$$

ahol:

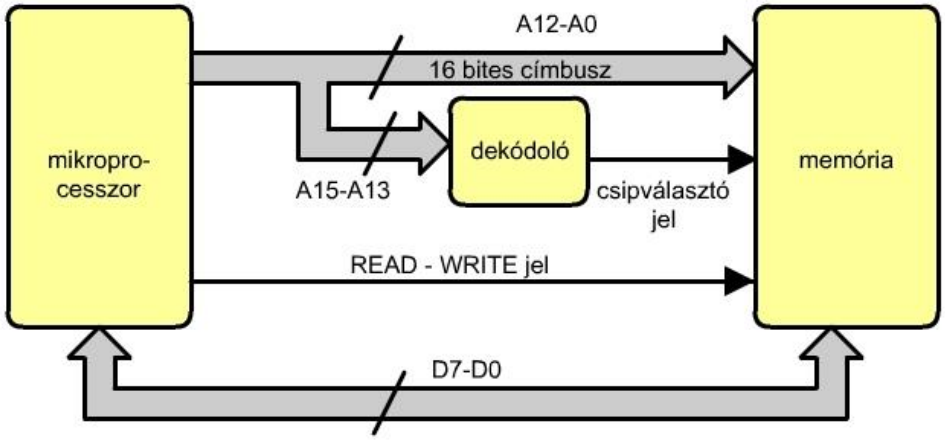
- összes cím -- a címezhető memóriarekeszek száma,
- n -- a címregiszter szóhosszúsága bitekben.

Ha egy (mikro)processzor például 16 bites címzéssel rendelkezik, akkor az 5.1. képlet szerint 65536 különféle elem (rekesz) különböztethető meg a rendszeren belül, ami a 16-os (hexadecimális) számrendszerben FFFFh. A szám utáni h betű utalás a hexadecimális számrendszerre. Ez az érték még 64 kB (64 kilobájt) alakban is megadható. A 65536 és a 64 kB közötti eltérés oka abban van, hogy a 2^{10} értéke 1024 a tízes számrendszerben és nem 1000.

Rövidebb program, vagy kevesebb adat esetén nem szükséges a számítógép illetve a mikrovezérlő teljes címezhető területét kiépíteni. Ugyanakkor egy memóriaterület felépítésekor különböző kapacitású memóriaelemek között válogathatunk. Ezek a kapacitások különböző értékűek, de mindnél közös az, hogy 2 hatványaként fejezhetőek ki, így például lehet 1 kB, 2 kB, 4 kB, 8 kB, 16 kB, 32 kB és 64 kB stb. nagyságú. Ezek a kapacitások a mai személyi számítógépeknél használt GB méretekhez képest elenyészően kis ételek, de ma is iparban használt egyszerű, kis teljesítményű rendszereknél elfogadott érték. A logika ugyanaz a kB-nál is, mint a G-nál. Az 1 kB-nál kisebb kapacitásnak ma már nincs gyakorlati jelentősége. Ezek a memóriakapacitások egymás kapacitásainak a kétszeresei, négyszeresei, nyolcszorosai, stb. Kialakításuk olyan, hogy beépített címdekódolóval rendelkeznek, így a címdekódolást

elvégzik, egy adott számú címvezetékkel rendelkeznek. Ez pl. a 8 kB-os memória esetén 2^{13} = 8096 rekesz, vagyis 13 címvezetékkel (A12-től A0-ig) rendelkezik a memória. A 64 kB nagyságú memóriaterületre így 64 kB / 8 kB = 8 db. ilyen IC köthető be.

Ha maradunk a 8 kB-os memóriánál, akkor az 5.20. ábrán megérthetjük a memóriacímzés és memória-választó logika lényegét, az 5.21. ábrán pedig a címek szerepét találhatjuk.



5.20. ábra: Példa a memória-választó logikára és címzésre, 8 kB memória esetén.

A memóriakapacitás növelésének érdekében több 8 kB-os memóriát is rákötethetünk a címsínre. A processzor adatsínjére párhuzamosan csatlakozik rá az összes memória adatvezetéke, ugyanígy a címsín A12 - A0 vezetékeire minden 8 kB-os memória 13 címvezetéke. A két, vagy több memória megkülönböztetése ekkor az 5.20. ábra 16 bites címregiszterének 'címdekódolóra' nevű mezője szolgál. Ez a példában **3 bit**, ami 2^3 = 8 elem megkülönböztetését teszi lehetővé (8 x 8 kB = 64 kB). A 8 jel közül kell kiválasztani azt a vezetéket, amelyik a memória **CS** (chip select), csipválasztó vezetékén keresztül engedélyezi, vagy tiltja az adott memória működését. Egy ilyen csipválasztó (dekódoló) logika igazságtáblázata látható az 5.22. ábrán levő táblázatban.

MSB LSB

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

5.21. ábra. 8 kB címzése és kiválasztása 16 bites címbusz esetén.

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

5.22. ábra. 8 darab 8 kB-os memória csip dekódolása.

Ekkor az 5.23. ábra szerinti memória kiosztást kapjuk. A baloldali oszlop a beépített memóriák kezdő- és utolsó címét jelöli, míg a jobb oldali oszlop az egyes memóriák sorszámát mutatja.

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

5.23. ábra. 64 kB memóriaterület kitöltése 8 darab 8 kB kapacitású memóriával.

Az 5.24. ábra a 0. 8 kB, az 5.25. ábra az 1. 8 kB, míg az 5.26. ábra a 6. 8 kB memória címeit mutatja, ahol x értéke 0 vagy 1 lehet. Az előzőek alapján a 0. memória 0000 kezdő- és 1FFF utolsó címe között helyezkedik el pl. 0B5A, de a 2B5A az 1. 8 kB memória címe, míg a CB5A már a 6. 8 kB memóriában van.

MSB LSB

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

5.24. ábra: A 0. 8 kB címei.

MSB LSB

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

5.25. ábra. Az első 8 kB címei.

MSB LSB

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

5.26. ábra. A 6. 8 kB címei.

A memóriák kijelöléséhez szükséges hardver megtervezése az 5.22. ábrán látható táblázat alapján történik. M0 vezetéket az 5.20. ábrán látható memória CS bemenetéhez kell csatlakoztatni. Az itt megjelenő 1 jel engedélyezi a memória működését, a 0 pedig teljesen leválasztja azt a mikroprocesszorról (az adatsínről). Matematikailag ez a következő alakban adható meg:

$$M0 = \overline{A_{15}} \cdot \overline{A_{14}} \cdot \overline{A_{13}} \qquad (5.2)$$

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

$$M1 = \overline{A_{15}} \cdot \overline{A_{14}} \cdot A_{13} \qquad (5.3)$$

$$M6 = A_{15} \cdot A_{14} \cdot A_{13} \qquad (5.4.)$$

Az A12 - A0 címek megjelennek ugyan mindegyik memórián, de csak a CS vezetéken keresztül engedélyezett memóriában címeznek egy rekeszt.

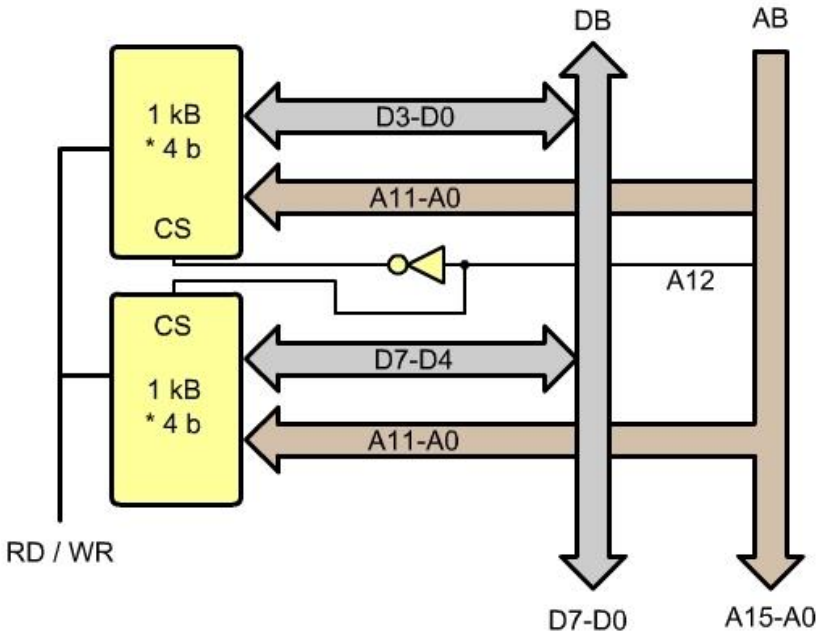
Különböző kapacitású memóriák is használhatók ugyanazon a memóriatartományon belül, de figyelembe kell venni néhány szabályt a memóriák elhelyezésére.

5.5. MEMÓRIA VÁLASZTÓ LOGIKA (RÉSZLEGES CÍMDEKÓDOLÁS)

Az előző fejezetben az úgynevezett teljes címválasztó-kód meghatározása volt látható. Ilyenkor minden cím egy adott fizikai elem (memóriarekesz) működését teszi lehetővé, ami a teljes címzés felhasználását jelenti.

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

Tartalmazzon egy rendszer két darab 4 kB kapacitású memóriát. Mint látható volt, a 4 kB címzése 12 vezetéket, vagyis A11 - A0 vezetékeket használja. Ugyanígy a másik 4 kB is ugyanezen vezetékeken keresztül címezi az egyes rekeszeket. Az A12 bit segítségével szétválaszthatjuk a két területet (5.27. ábra).



5.27. ábra. Részleges címdekódolás két darab **4 kB**-os memória esetében.

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

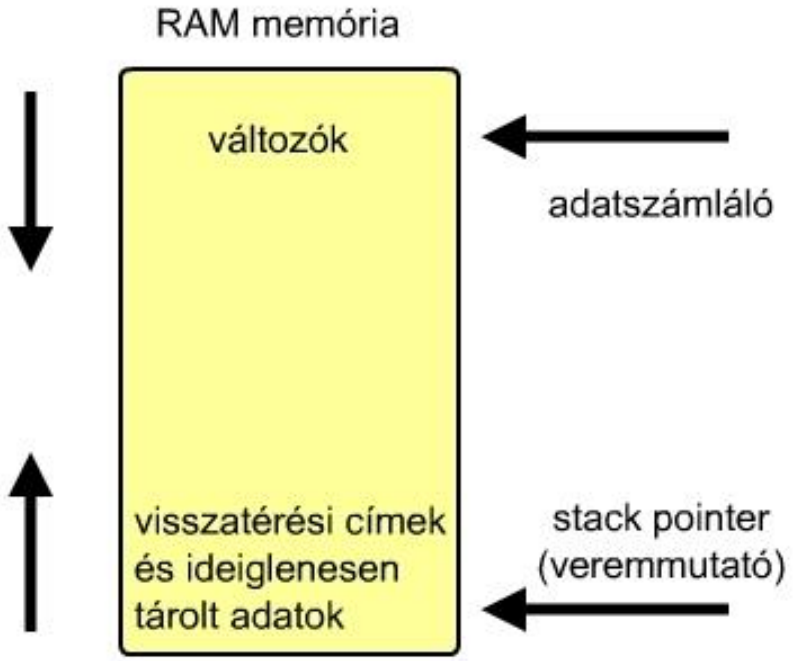
5.6. A VEREMTÁR (STACK) MEMÓRIA

Az alacsony- és magas szintű programozási nyelveknél megismert alprogramok és megszakítással indított alprogramok, eljárások visszatérési címének tárolását biztosító memóriát nevezik verem, vagy stack memóriának. A gépi szintű programozás is használ alprogramokat, ezért szükséges ennek a memóriatípusnak a tárgyalása. Fontos még az alprogramoknál, megszakításoknál a megszakított program adatainak, paramétereinek ideiglenes megőrzése is, ez is lehetséges a verem (stack) memóriában is.

Kétféle megoldás terjedt el a gyakorlatban:

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

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



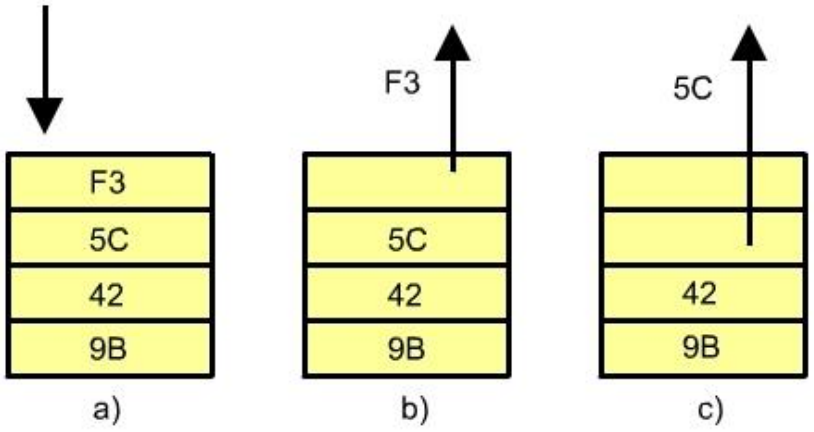
5.28. ábra. RAM memória felosztása változók- és verem adatok tárolására.

A fenti ábrán egy olyan megoldás látható, ahol a rendszer RAM memóriáját együtt használjuk a változók tárolására, ez a memória felső része, az új változók tárolása növekvő címek felé történik. A RAM memória alsó részén helyezkedik el a veremtár, új adatok beírása a veremtár-mutatót (stack pointer) a kisebb címek felé mozdítja el. Ez azért jó, mert így a két terület (változók és veremtár) egymás felé közeledik. Természetesen ügyelni kell arra, hogy ne történjen átlapolódás a két rész között. Vannak olyan megoldások, ahol nagyobb RAM terület áll rendelkezésre, hogy teljesen szétválaszthatók legyenek az egyes területek feladat szerint, így külön program-, adat-, verem- és különleges memóriaterület van kialakítva, nem lehetséges egyikből a másikba véletlenül adatot átvinni.

A programozott verem viszonylag nagy kapacitású memória, de a processzor és verem közötti adatcsere időigénye körülbelül tízszerese a processzoron belüli regiszter-regiszter műveletek időigényének. A hardver verem nagy sebességű adathozzáférést biztosít, hiszen processzoron belül helyezkedik el, igaz kapacitása kisebb a másik típusénál.. A kisteljesítményű mikrovezérlőknél inkább a hardver megoldást alkalmazzák, míg például adatfeldolgozásnál a szoftver megoldást (a PC gépeknél is).

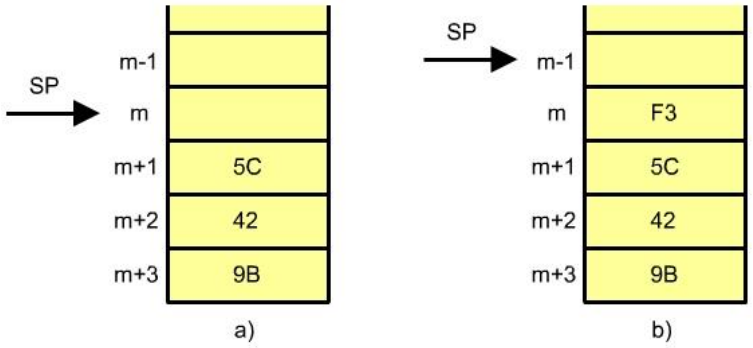
Általában a mikroprocesszorok rendelkeznek a stack memóriába adatot beíró utasítással, amit rendszerint PUSH utasításnak hívnak, valamint adatot a memóriából kiolvasó utasítással, ami POP (néha PULL). Ez a memória LIFO (Last In First Out) struktúrájú, ami azt jelenti, hogy a

legutoljára beírt adat kiolvasása történik meg legelőször, a következő kiolvasott adat még korábban került a memóriába. Az 5.29. ábrán látható, hogy a PUSH utasítással a veremtár csúcsára beírt (5.29. (a) ábra) adat a POP utasítással (5.29. (b) ábra) olvasható ki, szintén a veremtár csúcsáról. A következő POP a régebben beírt adatot távolítja el (5.29. (c) ábra).



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

Az 5.30. ábra szemlélteti a stack memória és a stack pointer (veremtár mutató) közötti kapcsolatot. Ha programozott veremtár használatáról van szó, akkor a veremtár mutató szóhosszúsága rendszerint a 8 bites gépeknél 16 bit, hardver megoldásnál értelemszerűen kisebb szóhosszúság elegendő. Az 5.30. ábrán programozott megoldás látható. A veremtár mutató 'sétál' a memória rekeszein, címezve azokat, minden stack memóriába való íráskor értéke csökken, adatkivételnél értéke növekszik. Létezik fordított megoldás is, vagyis adatbeírás esetén a veremtár mutató értéke eggyel nő, kiolvasáskor eggyel csökken.

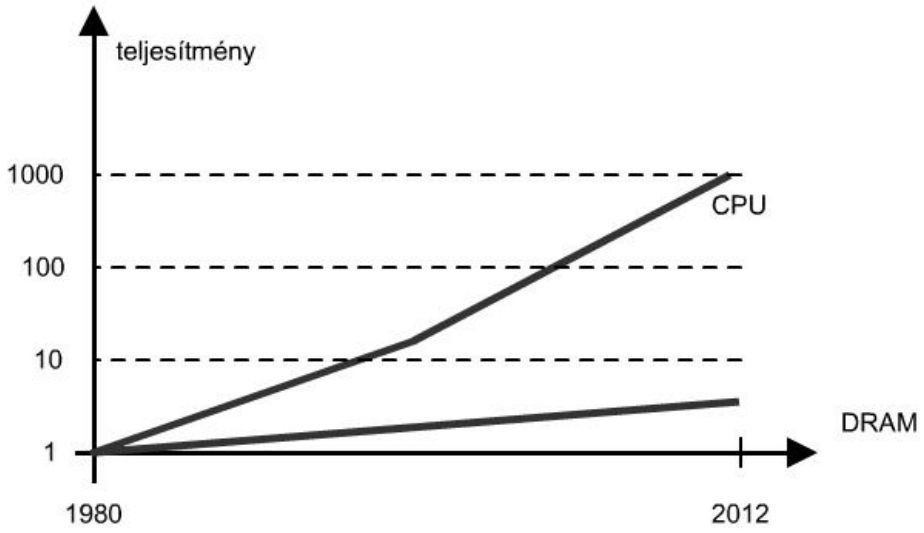


5.30. ábra. A veremtár mutató (**SP**) és a veremtár viszonya a **PUSH** utasítás végrehajtása előtt (a ábra) és után (b ábra).

5.7. MEMÓRIÁK ELÉRÉSI IDEJÉNEK VIZSGÁLATA

A mai modern számítógép-technika fejlődése immár több mint 60 évre tekint vissza. Ez idő alatt mind az architektúránál, mind a technológiánál születtek olyan megoldások, amelyek a mai napig alkotóelemei számítógépeinknek, meg természetesen ezen megoldások, technológiák egyes részei túlhaladottakká is váltak.

Egyik teljesítménynövelő, nem strukturális megoldás az órajel-frekvencia növelése, amely ma már a legnagyobb értékeknél tart, nem nagyon tudjuk ezt a sebességet tovább növelni. Mi a helyzet az egyes elemeknél elért fejlődéssel? A következő ábrán több mint 20 évet tekintünk át, egyrészt a processzorok időzítését, órajel-frekvenciáját, másrészt a DRAM (dinamikus RAM) félvezető memóriák elérési idejét összehasonlítva. A teljesítménynövelést logaritmikus skálán ábrázoljuk. Látható, hogy ez idő alatt a processzoroknál három nagyságrenddel, tehát ezerszeresen megnőtt az órajel frekvenciája, míg ugyanezen idő alatt a memóriáknál egy hét-, nyolcszoros gyorsulást sikerült elérni.



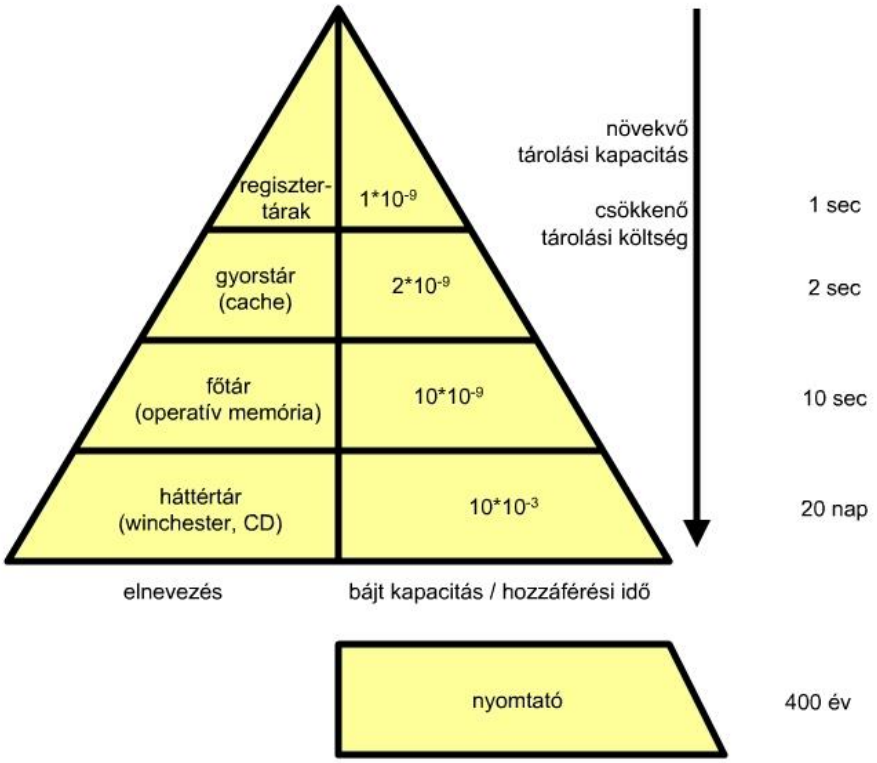
5.31. ábra. Processzorok és memóriák teljesítményének növekedése

Ennek a nagy időbeli különbségnek következményei vannak, a processzor képes időegység alatt jóval több műveletet végrehajtani, mint amire a DRAM memóriák képesek, tulajdonképpen a memória szabja meg a számítógép teljesítményét.

Kisebb teljesítményű, iparban használt processzoroknál a megoldás az, mivel itt sok, kis teljesítményt igénylő feladatot kell megoldani, hogy a memória elérési sebességéhez idomítjuk a processzorok sebességét.

Ott, ahol mégis cél kihasználni a processzor képességeit, egy olyan megoldást kell találni, amely közelebb hozza a két értéket. Először nézzük meg a következő ábrán, hogy milyen arányok vannak a memóriakapacitások és az elérési idők között egyes memóriáknál. Az ábra

csak szemléltető jellegű, piramis alakú, ezzel a lefelé szélesedő jelleggel a nagyobb kapacitást jelöljük. Értéket elég nehéz beleírni, mert a fejlődés olyan gyors, hogy rövid idő alatt akár nagyságrenddel is nőhet a kapacitás. A sebesség viszont aránylag stabil érték, így az fel van tűntetve, de mivel ezeket az értékeket nem tudjuk érzékelni, az ábra jobb oldalán egy olyan skála látható, amely 1 másodpercnek tűnteti fel a regiszter-regiszter műveletet, ehhez arányosan pedig a többi memória sebességét. Érdekességképpen a nyomtató egy karakter nyomtatásához szükséges idejét is feltüntettük, ami esetünkben 200 év!



5.32. ábra. Memória elérési idők összehasonlítása.

A fenti ábrán a processzor (regiszter-tárak) és a főtár között megjelenik egy főtárnál kisebb, de regiszterek számánál nagyobb kapacitású, regisztereknél lassúbb, de főtárnál gyorsabb, viszonylag drága gyorsítótár, a cache. Szerepe az, hogy a gyakran használt adatokhoz, illetve programrészekhez gyorsabban jusson a processzor. Megjelenésükkor még közös adat-program cache volt használatos, azután szétválasztották külön adat és külön program gyorsítótárra, majd kétszintű, később megjelentek a háromszintű megoldások is.

A program-cache esetében a program épp egy aktuális része kerül a cache-be, ahonnan így a processzor ötször gyorsabban tudja kiolvasni az utasításokat, mint a főtárból. Valójában a főtár tartalmát hardver megoldással a gép blokkokra osztja, mindig azt a blokkot bent tartva a

cache-ben, amelyik aktuális (ahol a program fut). Amennyiben a következő utasítás máshol van, új blokk másolódik be, egy lépésben.

Az adat-cache alkalmazásakor fordított irányú blokkmozgatásra is szükség van, hiszen a cache-be visszaírt értékek a főtárba is vissza kell hogy kerüljenek.

Program-, illetve adatlokalitásról akkor beszélünk, ha a programutasítások, illetve az adatok kis időintervallumban a memória egy kis területét foglalják el.

Időbeli lokalitásról akkor beszélünk, ha adatra-, illetve utasításra rövid időn belül hivatkozunk.

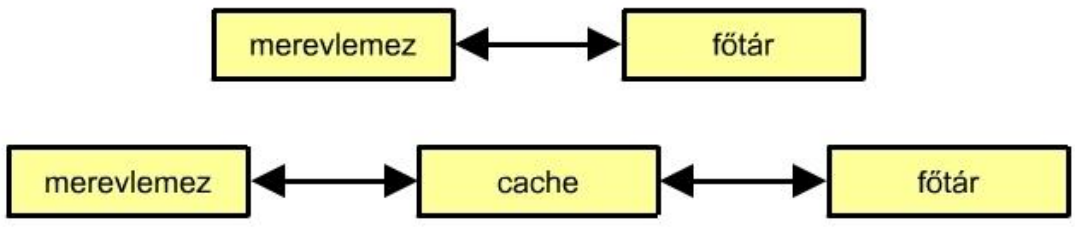
A helyi lokalitás azt jelenti, hogy az adatra-, illetve utasításra való hivatkozásnál a következő értékek ezek közvetlen környezetében levő adatok, illetve utasítások. Például egy vektor elemeinek egymás utáni feldolgozása, vagy a programnál a lineáris programszerkezet miatt a következő utasítás a következő memóriacellában van.

Fizikailag, ha a gyorsító tár a csipen belül van, akkor ON-CHIP, a csipen kívül van akkor OFF-CHIP megoldásról beszélünk.

Ha hivatkozáskor az adat, illetve utasítás a cache-ben van, akkor a CACHE-HIT fogalmat, ha nincs a cache-ben, akkor CACHE-MISS fogalmat használjuk. A szervezés hatékonysága miatt a CACHE-HIT/CACHE-MISS arány 90 % felett van.

5.8. LEMEZGYORSÍTÓ CACHE TÁRAK

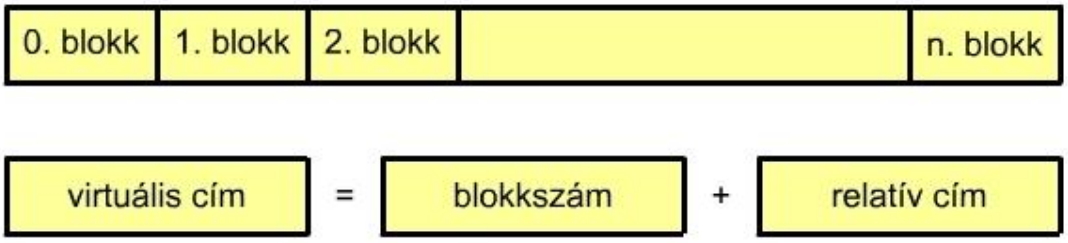
Természetesen a számítógép-rendszereken belül másutt is vannak olyan kapcsolatok, ahol a két egység működésében lényeges időbeli eltérések vannak. Ilyenek a háttértárak. Itt is lehet a működést optimalizálni, lemezgyorsító gyorsító tárak alkalmazásával. Ezek nagysága a több tíz MB méretet is meghaladja. A következő ábrán láthatjuk a gyorsító tár nélküli és az azzal történő megoldást. Látható, hogy itt is nagy a sebességbeli különbség a háttértár és főtár között (5.33. ábra).



5.33. ábra. Lemezgyorsító cache megoldás

5.9. VIRTUÁLIS MEMÓRIA, VIRTUÁLIS TÁRKEZELÉS

Szeretnénk, ha a számítógép például a háttértár teljes tartalmát „látná”, vagyis egyszerre tudjon bármikor bármit gyorsan elérni. Természetesen a főtár az a része a gépnek, amellyel a processzor a gyorsító tár(ak)on keresztül kapcsolatban van. Ilyenkor az operációs rendszer és a hardver közösen egy olyan technikát alkalmaz, amelynél a processzor teljes címzési tartománya a háttértároló területe, az úgynevezett MMU (Memory Management Unit) a virtuális címet a főtár címévé alakítja. A cache memóriánál megismert logikához hasonló módon történik ez az átalakítás. Több megoldás létezik, egyik az, amikor egyforma méretű blokkok kerülnek a háttértárolóról a főtár bizonyos részébe.



5.34. ábra. A blokk szerepe a virtuális memóriánál

Amikor új blokkra van szükség, a hardver és operációs rendszer felülír egy régi tartalmat egy új blokkal, itt is több megoldás van, gyakran használják az LRU (Least Recently Used – legritkábban használt) technikát.

Alkalmaznak még szegmentálási technikát is a blokkok helyett, de itt mivel az egyes tartalmak különböző hosszúságúak, figyelni kel az átlapolódásra.

6. PERIFÉRIÁK, KI- ÉS BEMENETI INTERFÉSZEK, EGYSÉGEK

A számítógép alapfeladata a bemeneti adatok és információk beolvasása, feldolgozása és a kapott eredmények, vezérlőjelek visszajuttatása a folyamatba, emberhez, illetve másik számítógéphez, számítógépekhez. Ebben a folyamatban fontos szerepet játszik a számítógép és periféria-egységek közötti adat- és információáramlás. Ilyen peifériák a billentyűzet,

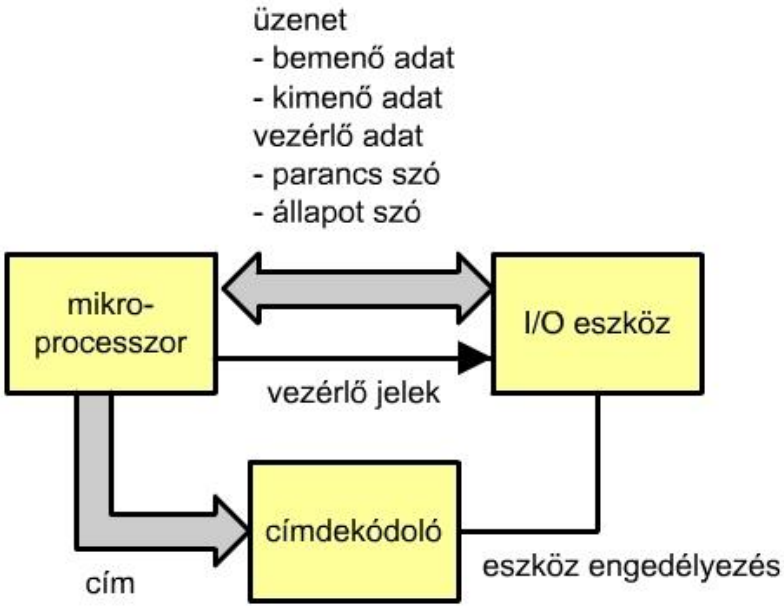
képernyő, érintőképernyő, egér, jelfogó, léptetőmotor, vagy az analóg jelek beolvasását lehetővé tevő A/D átalakító, mikrofon, hangszóró stb.

Az I/O periféria-eszközök LSI (Large Scale Integration) – nagy bonyolultságú – és VLSI (Very Large Scale Integration) – nagyon nagy bonyolultságú technológiában megvalósított eszközök, amelyek rendszerint programozhatóak, pontosabban bizonyos paramétereik állíthatóak. Függetlenül attól, hogy mint önálló csipek képezik egy mikrogép építőelemét, vagy szerves részét alkotják egy mikrovezérlőnek, működési elvük, felépítésük ugyanaz. Esetleg különbség csak ott van közöttük, hogy addig míg a számítóépnél a tervezőmérnök hozzárendeli egy címhez az eszközt, addig ez a mikrovezérlőn belül már adott, ezen változtatni nem lehet.

A számítógéphez csatlakoztatott I/O periféria-eszköz szerves részét képezi a rendszernek, feltétlenül szükséges használatakor is biztosítani a számítógép, számítógép-rendszer összehangolt működését. Ezt teszi lehetővé az illesztő áramkör, az interfész (interface). A számítógépekhez az I/O eszközt ugyanolyan logika szerint kapcsolhatjuk hozzá, mint a memóriáknál látott teljes, illetve részleges címdekódolás (címkiválasztás).

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

Az 6.1. ábrán látható blokkvázlat szemlélteti, hogy elvileg milyen adat-, illetve információáramlást kell biztosítani a processzor és az I/O eszköz között.



6.1. ábra. A periféria csatolása a processzorhoz.

Meg kell jegyezni, hogy a következőkben az egyszereű, viszonylag kis kapacitású számítógépekről, mikrovezérlőkről van szó, ahol a perifériák vezérlését is a processzor végzi, tehát itt nics külső sin alkalmazva, mint a komolyabb számítógépeknél.

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

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

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

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

Mint már az előző fejezetekben láttuk, a CISC struktúrájú gépekben az adatok és programok

ábrázolása párhuzamosan történik 8-, 16-, 32- vagy 64 biten. Amikor két eszköz között egy így ábrázolt adat átvitele válik szükségessé, kézenfekvő a szóhosszúságnak megfelelő bitszámú párhuzamos adatátvitelt, csatornát alkalmazni. Ekkor valóban az adatátvitel sebessége nagy, de például egy 32 bit szóhosszúságú információcsere 32 párhuzamos vezetéket igényel, és még egy közös földvezetéket, nem beszélve arról, hogy ilyenkor a zavarszűrés érdekében minden egyes vezetékkel párhuzamosan összesodorva kell vezetni a közös, földvezetéket, ami így már legalább 64 eret jelent. Ezt a módszert rövid távolságok áthidalására lehet alkalmazni, általában 8 bit szélességben.

Nagyobb távolságokra való adatátvitelnél rendszerint soros átviteli csatornát használunk. Ennek egyik oka az, hogy ilyenkor mindössze egy érpárra van szükség az adatátvitelhez, ami lényegesen egyszerűsíti a kábel kivitelezését, de a soros adatátvitel a már meglevő rendszereknél is használható. Ilyen például a telefonvonal. Ekkor már lényegesen lecsökken az adatátviteli sebesség, hiszen minden egyes bitet külön-külön kell továbbítani, valamint az adó és vevő között szinkronizációt, együttfutást kell biztosítani.

Vezetékes soros adatátvitel mellet még alaklmaznak soros infravörös adatátvitelt is, ahol egy infravörös tartományban a fényt kibocsátó dióda modulált jelét egy fototranzisztor fogadja. Természetesen a külvilág zavaró jeleit szűrni kell, továbbá a közeg (levegő, levegőben levő pára, por, szennyezés) az átvihető távolságot is befolyásolja.

Rádiós soros adatátvitel a széleskörűen elterjedt, főleg a 2,4 GHz-es tartományban dolgozó Bluetooth technológia, vagy a wifi hálózat.

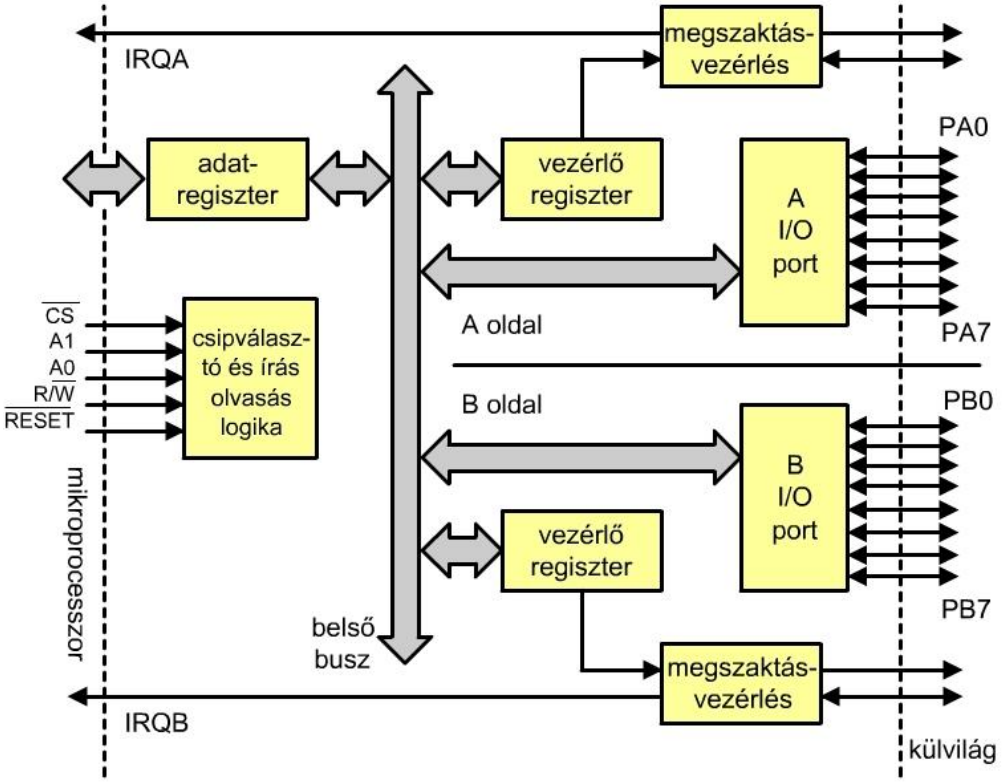
A soros és párhuzamos adatátviteli elv kombinálható is, például 16 bites adat átvitele megoldható négy lépésben egy négy bit szóhosszúságú adtátviteli csatornával.

A különleges I/O eszközök használata célszerű gyakran használt, bonyolult adatátvitelnél, ahol nagy mennyiségű adat mozgatását kell biztosítani gyorsan, például háttértárak (hajlékonylemez-meghajtó, Winchester stb.).

6.1. PÁRHUZAMOS I/O VEZÉRLŐ-ILLESZTŐ ESZKÖZÖK

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

A 6.2. ábrán egy programozható I/O párhuzamos interfészt belső, logikai felépítése látható.



6.2. ábra. Párhuzamos I/O interfész belső felépítése, blokk-sémája.

A két vezérlőregiszter (A port és B port) biztosítja az áramkör programozhatóságát. Az ide beírt parancsszó határozza meg az adatátvitel paramétereit, megszakítás-engedélyezést, stb. Ugyanezen információ határozza meg az I/O vonalak adatáramlási irányát, tehát azt, hogy bemenetkét, vagy kimenetként használja a program ezeket a csatlakozópontokat. Egyes I/O eszközöknél a kimenetek és bemenetek csak csoportonként határozhatók meg, de léteznek olyan párhuzamos elemek is, ahol minden egyes csatlakozási pont tetszőlegesen használható kimenetnek, vagy bemenetnek. A mikrovezérlőknél hasonló szervezésű portok vannak egybeintegrálva a processzorral, memóriákkal. Lábanként lehet az irányt a portokon programozni. Elektromos szempontból olyan meghajtóval rendelkeznek, hogy a bemenet nem igényel külön tápfeszültséget, mindössze egy kapcsoló, nyomógomb vagy érintkező adja a jelet, kimenetként pedig mind logiai 0, mind logikai 1 állapotban 15 mA-es jelet biztosítanak.

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

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

- $\overline{CS}$ a csipkiválasztás jele,
- $R/\overline{W}$ (olvasás/írás) jel, aminek megléte is szükséges az adatáramláshoz, ez a jel szinkronizálja az adatátvitelt a processzor és az eszköz között, ha például $R/\overline{W}$ = 1, akkor az eszköz felől a mikroprocesszor felé van adatátvitel, $R/\overline{W}$ = 0 esetén a processzor küld adatot az I/O eszközbe,
- $\overline{RESET}$ jel biztosítja bekapcsolásnál az eszköz alapállapot-beállítását, ez a jel tulajdonképpen hardver jel, a táplálás bekapcsolásakor jelentkezik,
- A_0 és A_1 címvezetékek, amik meghatározzák a vezérlőregisztereket, itt a két vezeték 4 belső vezérlő regiszter meglétét teszi lehetővé,
- **IRQA** és **IRQB** vezetékek megszakítást kérhetnek a mikroprocesszortól akkor, ha azt a parancs-szó lehetővé teszi (tehát a programíró engedélyezte a külső megszakítás elfogadását).

A bekapcsoláskor fellépő $\overline{RESET}$ hardver jel csak bizonyos belső regisztereket, illetve a kimeneteket állítja alaphelyzetbe. Az alaphelyzet nem mindig 0, illetve nem minden eszköznél veszik fel a kimementi pontok a 0 értéket, van ahol ez az érték 1-es, ennek oka, hogy ha a kimenetek INPUT állapotba kerülnek, akkor a bemeneten biztosítja a hardver a tápot, hogy a kpcsolónak ne kelljen kükön, kívülről ezt megkapni.

A programozó feladata, hogy megírjon egy inicializáló programrészt, amely a főprogram indulásakor a hardver-szoftver-igényeknek megfelelő kezdőhelyzetet hoz létre. Ez főleg a kimenetek, illetve bemenetek meghatározása, de ide tartozik a megszakítás engedélyezése, vagy tiltása is. Ez a kezdőhelyzet beállítás két részből áll:

- az eszköz alapállapotba hozása $\overline{RESET}$ jellel, vagy programutasításokkal és
- az új parancsszó beírása az I/O eszköz parancsregiszterébe.

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

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

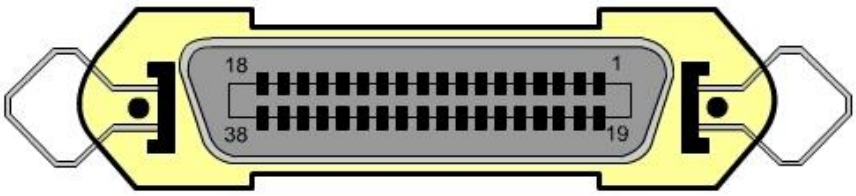
Az első adtátvitel ugyan a legegyszerűbb, de ilyenkor minden feltétel nélkül a kimeneti eszközre adat kerül, annak vizsgálata nélkül, hogy az működőképes, illetve foglalt, vagy nem. Ugyanez vonatkozik adatbeolvasásra is, feltétel nélküli adatbeolvasásra kerül sor, annak ellenőrzése nélkül, hogy az adott bemeneteken valóban megtalálható-e a stabil, helyes adat. Látható, hogy az ilyen adatforgalom hardvere egyszerű, ám a megbízhatósága is kicsi.

Az előző adatátviteli mód hibáinak kiküszöbölésére szolgál a második tipusú, ahol megjelenik egy, vagy néhány vezérlő jel az adatátvitel szinkronizálására. Ilyen vezérlőjelek például az adat stabil, eszköz foglalt, vagy egy konkrét példa, a nyomtatóban nincs papir stb. A bonyolultabb hardver általában képes megoldani a szinkronizációs feladatokat, amennyiben nem, ezt szoftver eszközökkel kell pótolni.

A harmadik adtátviteli tipus hasonlít a másodikra, a különbség csak ott van, hogy nincs állandó szoftveres vezérlőjellekérdezés, hanem a hardver eszközön létrejövő változás hatására programmegszakítási igény lép fel, amit az I/O eszköz továbbít a processzor felé.

6.1.1 A CENTRONICS PÁRHUZAMOS ADATÁTVITEL

Nagyon sokszor szükséges vézérlő-, mérő-, adatgyűjtő, vagy más rendszereket az igen elterjedt IBM PC személyi számítógéphez kötni adatcsere céljából. A berendezés és PC között párhuzamos adatcserelehetőség valósítható meg az úgynevezett CENTRONICS párhuzamos porton keresztül, ami gyakorlatilag minden PC tartozéka. Ez a port szolgált valamikor a párhuzamos nyomtatók személyi számítógéphez való kapcsolására. Még ma is a biztos, hibamentes adatátvitele miatt CNC vezérlésű szerszámgépek x, y és z tengelyét ezen keresztül vezérlik. Az 6.3. ábrán a PC személyi számítógép hátoldalán levő csatlakozó képe látható. A kivezetéseket, a lábakat az ábra szerint számozzák 1 és 38 között.



6.3. ábra. A PC személyi számítógép hátoldalán található CENTRONICS párhuzamos port.

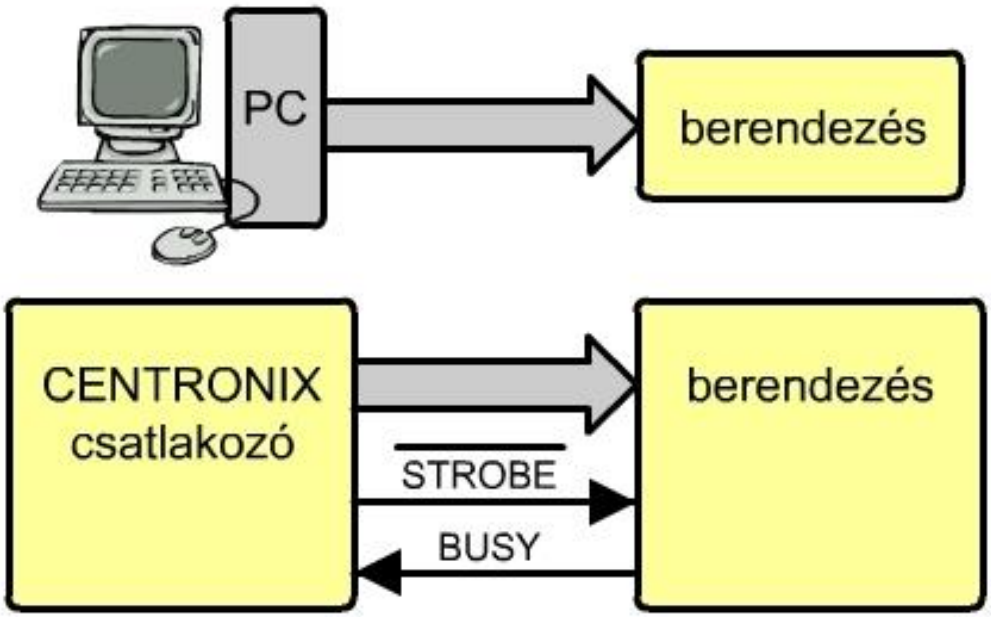
A 38 csatlakozási pont megnevezése és feladata a 6.1. táblázatból olvasható le. A táblázat irány oszlopa a csatlakozóhoz kapcsolt berendezés (például nyomtató) szempontjából értendő.

6.1. táblázat. A CENTRONICS párhuzamos csatlakozó lábkiosztása

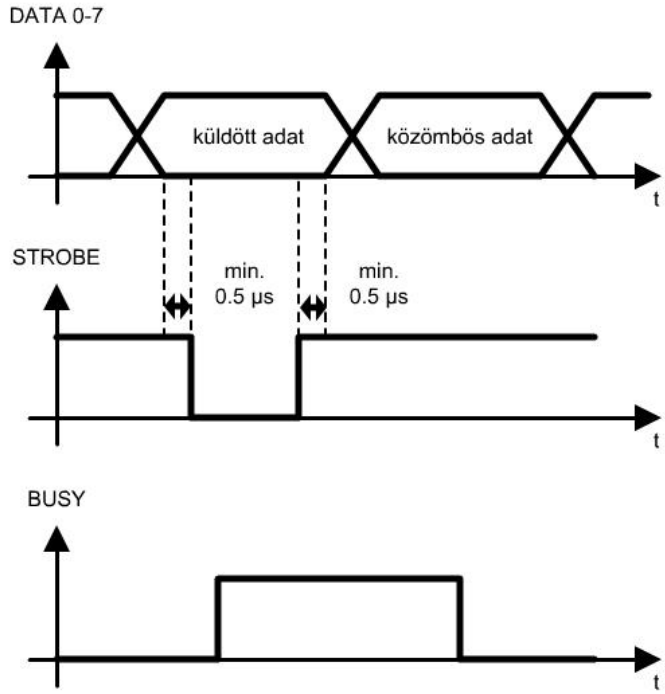
Láb	Jelölés	Irány	Leírás
1	$\overline{STROBE}$	bemenet	0 esetén beolvassa a DATA adatot (DATA7 – DATA0). Az impulzus legalább 0.5 µs kell hogy legyen
2	DATA 0 (LSB)		
3	DATA 1		
4	DATA 2		
5	DATA 3	bemenet	A nyolc párhuzamos adtbit
6	DATA 4		
7	DATA 5		
8	DATA 6		
9	DATA 7 (MSB)		
10	$\overline{ACKNOWLEDGE}$	kimenet	A jel 0 aktiv lesz, ha az adatátvitel befejeződött. Ekkor újabb adatátvitel kezdődhet.
11	BUSY	kimenet	Jelzi a PC felé, hogy foglalt a berendezés.
12	PE	kimenet	
13	SELECT	kimenet	Ha értéke 0, nem vehető a 8 adatbit
14	$\overline{AFD}$	bemenet	
15	NC		Nincs kihasználva.
16	OV		
17	CHASSIS GND		A berendezés házának földje.
18	+ 5 V	kimenet	Legfeljebb 50 mA-os áramforrás.
19-30	GND		Jel földelés.

31	$\overline{INPUT\ PRIME}$	bemenet	
32	$\overline{FAULT}$	kimenet	
33	GND		
34	NC		Nincs kihasználva.
35	+ 5 V	kimenet	
36	SLCT IN	bemenet	

A PC személyi számítógép és a mi általunk tervezett berendezés között kialakítható egy párhuzamos adatátvitel (6.4. ábra). Az eljárást megérthetjük az 6.5. ábrán látható idődiagramból.



6.4. ábra. Párhuzamos adatátvitel PC-ből berendezésbe CENTRONICS párhuzamos csatlakozóval.



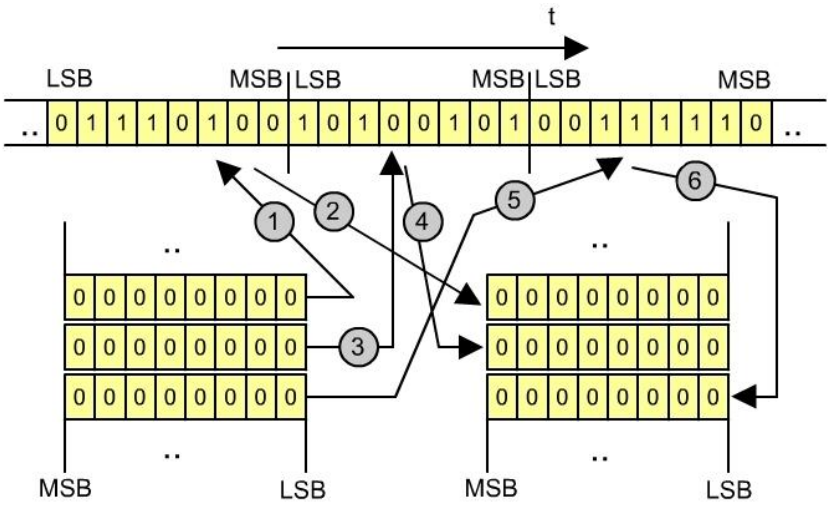
6.5. ábra. A párhuzamos adatátvitel idődiagramja PC-ből berendezésbe CENTRONICS párhuzamos csatlakozóval

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

6.2. SOROS I/O VEZÉRLŐ-ILLESZTŐ ESZKÖZÖK

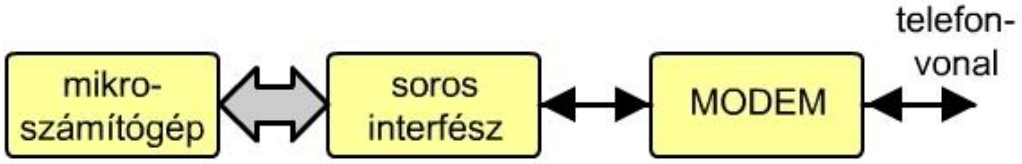
A soros adatátvitelnél az adóoldali I/O eszköz feladata az, hogy a párhuzamos (rendszerint 8 bites) adatot soros impulzusok sorozatává alakítsa át, azért, mert a soros adatátvitel egy

vonalon keresztül történik. A vevőben található soros I/O eszköz a sorosan vett impulzusok sorozatát alakítja vissza párhuzamos adattá (6.6. ábra).



6.6. ábra. Párhuzamos adatok sorossá való átalakítása és visszaalakítása párhuzamos adatokká.

A soros adatátvitel igen elterjedt, mindössze egy vonal (érpár) szükséges megvalósításához. Főleg térben távoli helyek között történik ily módon az átvitel. Igen gyakran a már meglevő telefonvonal használható adatátvitelre (MODEM közbeiktatásával), aminek segítségével Interneten keresztül bármely távoli helyen levő számítógép elérhető. A 6.7. ábrán látható a számítógép és telefonvonal összekapcsolása.



6.7. ábra. Adatátvitel mikroszámítógép és távbeszélővonal között.

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

Egyéb berendezések, például adatgyűjtő eszközök, szabályozók stb. is kapcsolhatók soros interfész alkalmazásával a mikroszámítógépen, ahogy az a 6.8. ábrán látható.

6.8. ábra. Eszköz csatlakoztatása soros interfésszel a mikroszámítógéphez.

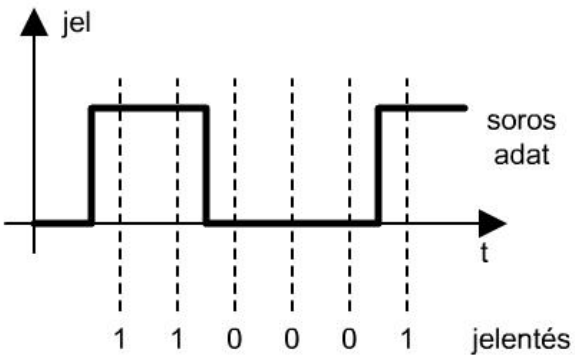
6.2.1. SOROS SZINKRON ÉS ASZINKRON ADATÁTVITEL

Soros adatátvitel esetén megkülönböztetünk:

- szinkron és
- aszinkron adatátvitelt.

6.2.1.1. SZINKRON SOROS ADATÁTVITEL

Ennél az adatátvitelnél egy órajelnek van kiemelt szerepe, az órajel megadása után minden egyes óraimpulzus alatt egy bitet kell átvinni. Ekkor az átvitt információ pontosan értelmezhető. A 6.9. ábrán látható a soros adatátvitel bináris alakban, illetve fészültség (áram) értékekben kifejezve.

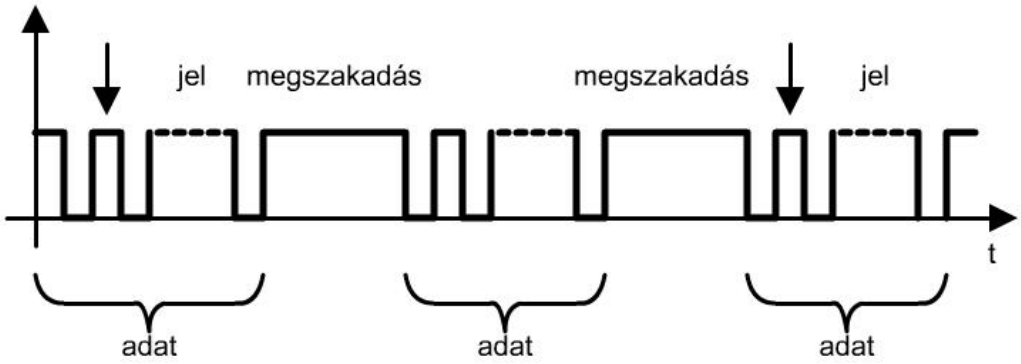


6.9. ábra. Szinkron soros adatátvitel

Mivel általában mindig bájtokat visz át a soros egység, szükséges a bájtok közötti határ megtalálása. Amennyiben ez nem történik meg, teljesen használhatattlan adatok kerülnek feldolgozásra. A bájthatár megtalálásának módja az, hogy az értékes adatok átvitele előtt az adó két szinkronizációs jelet küld ki.

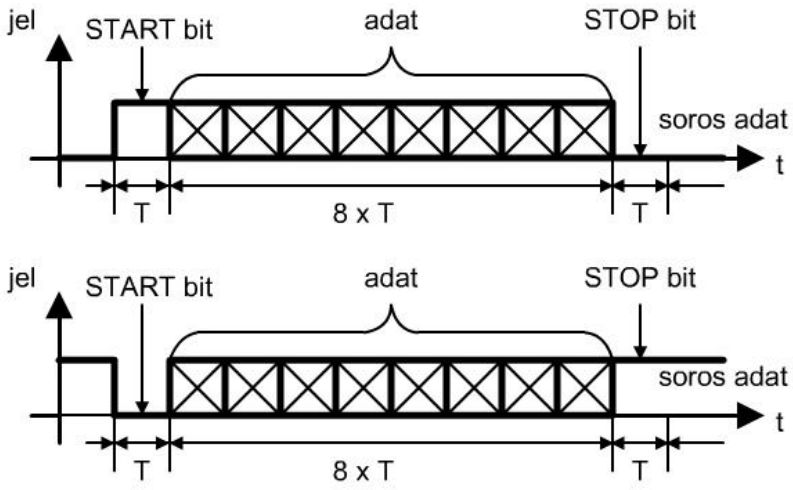
6.2.1.2. ASZINKRON SOROS ADATÁTVITEL

Az aszinkron adatátvitel abban különbözik a szinkron átviteltől, hogy akkor, amikor nincs adat, nem történik semmilyen jelátvitel, a karakterek (bájtok) között egy szakadás van (BREAK). Amennyiben a logikai 0-hoz 0 V feszültséget, a logikai 1-hez például 5 V feszültséget rendelünk, akkor nem lehetne különbséget tenni egy valódi fizikai vezetékszakadás és egy jelmegszakadás között. Ezért rendszerint fordított logikát használunk, vagyis a logikai 0 lesz 5 V érték, míg a logikai 1-es 0 V. Így akkor, amikor nincs adatátvitel állandó magas szint van a vezetéken. Egy soros aszinkron adatátvitel látható a 6.10. ábrán.



6.10. ábra. Aszinkron soros adatátvitel.

A két adat közötti idő tetszőleges hosszúságú lehet, úgyhogy valami módon meg kell oldani a szinkronizálást az adó és a vevő között. Ehhez szükséges egy adadtátvitelt indító jelzőbitet bevezetni, illetve az adatok végét jelző bitet. A valódi 8 bites adatot megelőzi egy START bit, valamint lezárja (legalább) egy (vagy több) STOP bit. Így valójában 1 bájt, vagyis 8 bit átviteléhez legalább 10 bitre van szükség. Az aszinkron adatátvitel látható a 611. ábrán, a felső ábrán a logikai 0 0 V feszültségnek, míg a logikai 1-es 5 V feszültségnek felel meg, az ábra felső részén pedig fordítva. Értelemszerűen a START és STOP bitek is megfordulnak, ellenkező értékűek. A 8 adat logikai 0, illetve 1 értékű lehet, ezt az ábrán a jelben levő áthúzással jelöltük.



6.11. ábra. Aszinkron adatátvitel szervezése.

6.2.2. RS-232C ASZINKRON FESZÜLTSÉGSZÍNTŰ SOROS ÁTVITELI CSATORNA

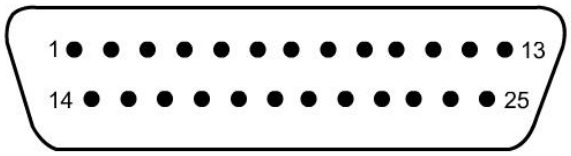
A **CCITT** nemzetközi távközlési bizottság a **V.24**-es és **V.28**-as ajánlásaiban meghatározta a soros adatátviteli interfész feladatait és elektromos jellemzőit. Az adatátvitel a **DTE** (Data Terminal Equipment) berendezés, ami a számítógép, illetve számítógépes terminál és a **DCE** (Data Communications Equipment) távközlési, vagy egyéb berendezés között történik. Az amerikai **EIA** szervezet egy szabványt fogalmazott meg az interfész létrehozására, aminek neve **RS-232C**. Ezen szabvány szerint soros összeköttetés valósítható meg bármilyen két berendezés között. A berendezés lehet számítógép, mérőfej, megjelenítő eszköz stb. Aszinkron üzemmód mellet az RS-232C interfésszel szinkron kapcsolat is létrehozható.

Az interfész csatlakozóból, kábelből és elektromos elemekből áll. Az interfész feszültségértékei az 6.2. táblázatban láthatók. A szabványban használt TTL jelszintnél magasabb feszültségértékek csökkentik az átvitelnél zavarólag ható zavarjelek hatását.

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

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

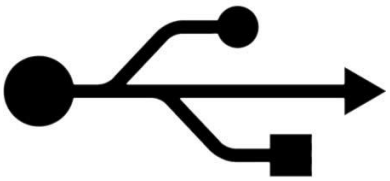
A csatlakozó lábkiosztása a 6.12.. ábrán látható, ennek neve D25.



6.12. ábra. **D25**-ös **RS-232C** csatlakozó lábkiosztása.

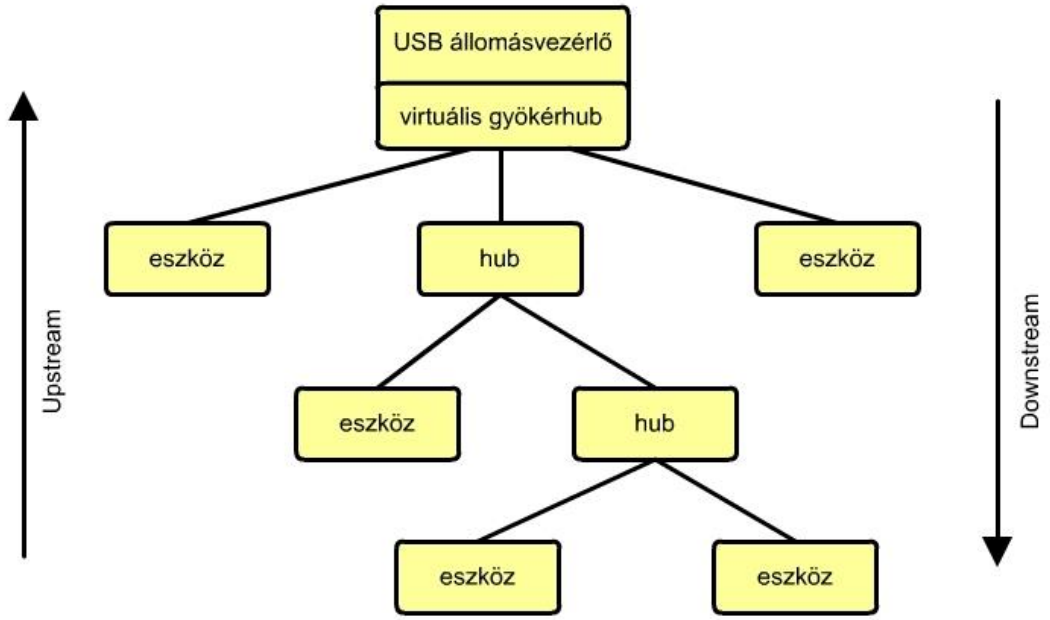
6.3. USB (UNIVERSAL SERIAL BUS)

Mind az RS232 soros, mind a CENTRONICS párhuzamos porthoz való csatlakozás csak feszültségmentes állapotban lehetséges. Ezzel szemben az USB csatlakozás PLUG AND PLAY, ami annyit jelent, hogy bármikor, működés közben is csatlakoztatható, illetve leválasztható a berendezés. A csatlakozó jele a következő:



6.13. ábra. USB csatlakozó jele

A csatlakozás további előnye, hogy fa struktúra alakjában további USB csatlakozóval ellátott készülékek kapcsolhatók a számítógéphez.



6.14. ábra. USB csatlakozás lehetséges hierarchiája.

Az ábrán a HUB nevű USB osztópont kevesebb darabszámú és kisebb teljesítményű további USB eszköz csatlakozását teszi lehetővé, ekkor passzív, tehát nem igényel további teljesítményt, illetve lehet aktív HUB is, amikor nagyobb fogyasztású, vagy több USB eszközt csatlakoztatunk.

Megjelenése óta több szabványos típusa jelent meg, néhány adat a különböző megoldásokra:

Low speed: max 150 kB/s, USB-1.1, USB-2.0, USB-3.0

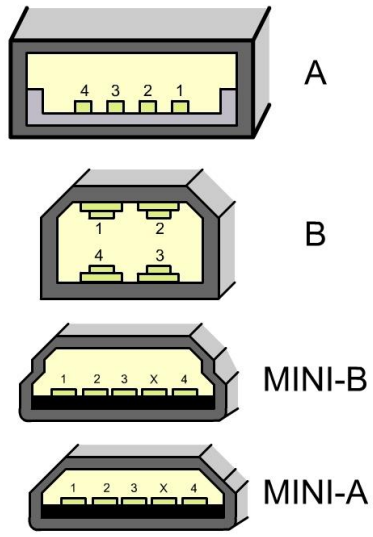
Full speed: max 1,2 MB/s, USB-1.1, USB-2.0, USB-3.0

Hi speed: max 48 MB/s, USB-2.0, USB-3.0

Super speed: max 400 MB/s, USB-3.0

A PC személyi számítógéphez csatlakoztatott különböző berendezések sokszor IRQ vonal megszakítást igényelnek (például az RS 232), ennek a technikának erre nincs szüksége, tehát nem köt le IRQ megszakítást.

A következő ábrákon néhány USB csatlakozó fizikai felépítését láthatjuk.



6.15. ábra. USB csatlakozó változatok

Az egyes lábak szerepe a következő táblázatban láthatók. A két adat vezeték (D^+ és D^-) mellett még a 0 – 5 V táplálás is megtalálható, ami egy csatlakozásnál általában 1 A terhelét engedélyez a készülék számára, vagyis az ennél kevesebbet fogyasztó csatlakoztatott eszközök nem igényelnek plusz táplálást.

6.3. táblázat. Az USB csatlakozási pontjai

Pin	Elnevezés	Szín	Leírás
1	VCC	Piros	+5 VDC
2	D-	Fehér	Data -
3	D+	Zöld	Data +
4	GND	Fekete	Ground

6.4. A SZÁMÍTÓGÉP ÉS PERIFÉRIÁK KÖZÖTTI ADATFORGALOM SZERVEZÉSE

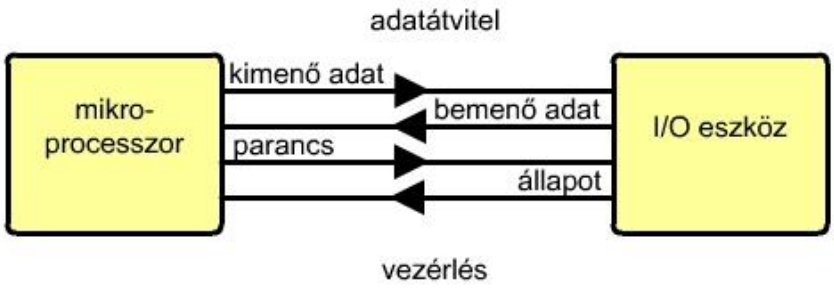
A számítógép hardver a periférián keresztül tud külvilágból analóg, vagy digitális jelet feldolgozás céljából beolvasni, illetve a szoftver, program alapján adatot, információt kiküldeni. A hardver működését programozással tudjuk hatásosan kihasználni, mindig a megfelelő biztonsági, átviteli sebességi követelmények figyelembe vételével.
A számítógépek fő feladata adatokon műveleteket végezni annak érdekében, hogy egy, vagy több faladatot végrehajtsanak. Ennek során állandó adatcserére kerül sor a külvilág és a mikroszámítógép, mikrovezérlő között. A külvilág (ember, másik számítógép, illetve folyamat)

a legkülönbözőbb típusú jeleket állítja elő (analóg és digitális, feszültség, nyomás, hőmérséklet, elmozdulás, szög stb.), ugyanúgy a legkülönfélébb jeleket kell a számítógépnek visszaküldenie az adott rendszerbe.

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

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

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



6.16. ábra. Processzor és I/O eszköz közötti adatáramlás és vezérlés.

Az I/O eszköznek négy feladatot kell megoldania:

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

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

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

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

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

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

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

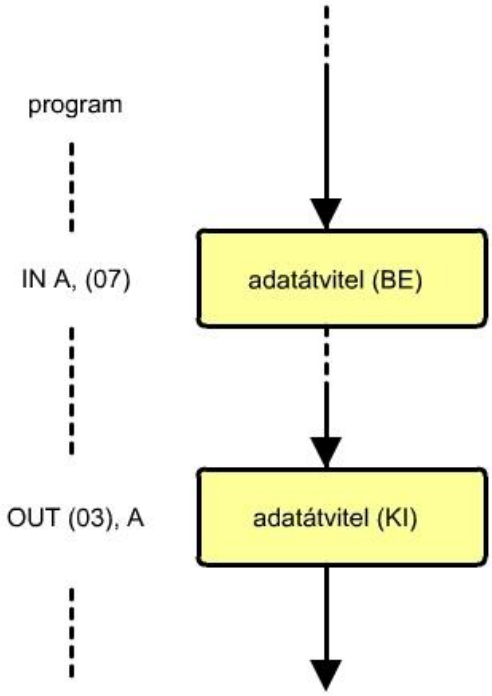
6.4.1. PROGRAMOZOTT I/O ADATÁTVITEL

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

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

6.4.1.1. FELTÉTEL NÉLKÜLI PROGRAMOZOTT I/O ADATÁTVITEL

Csak olyan esetben használatos, amikor a külső periféria válaszideje ismert. Ennél a megoldásnál a külső eszköznek mindig késznek kell lenni az adatátvitelre (akár küldés, akár fogadás) a programban szereplő megfelelő utasítás végrehajtásakor. Egyszerűségéből következik, hogy a program, valamint a hardver felépítése is igen egyszerű. A 6.17. ábrán látható a feltétel nélküli adatátvitel vázlata.



6.17. ábra. Feltétel nélküli I/O adatátvitel folyamatábrája.

A programban szereplő IN A,(07) utasítás hatására a bemenő eszköz portján levő 8 bit bekerül a processzor akkumulátorába. Az IN utasítás (07) része a beviteli eszköz címét adja meg, itt ennek értéke 7. Ezt követően egy programrész végrehajtására kerül sor, amely rész az ábrán pontokkal van jelölve. A másik adatátviteli utasítás, az OUT (03),A hatására az akkumulátorban

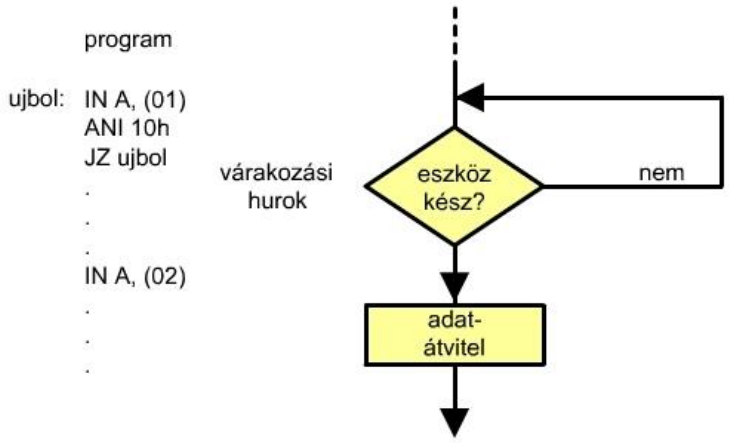
levő 8 bit átkerül a (03) címmel ellátott kiviteli eszközre, ahol az eszközben levő regiszter tárolja az új adatot egy újabb adatkiírásig, adatváltoztatásig.

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

6.4.1.2. FELTÉTELES PROGRAMOZOTT I/O ADATÁTVITEL

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

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



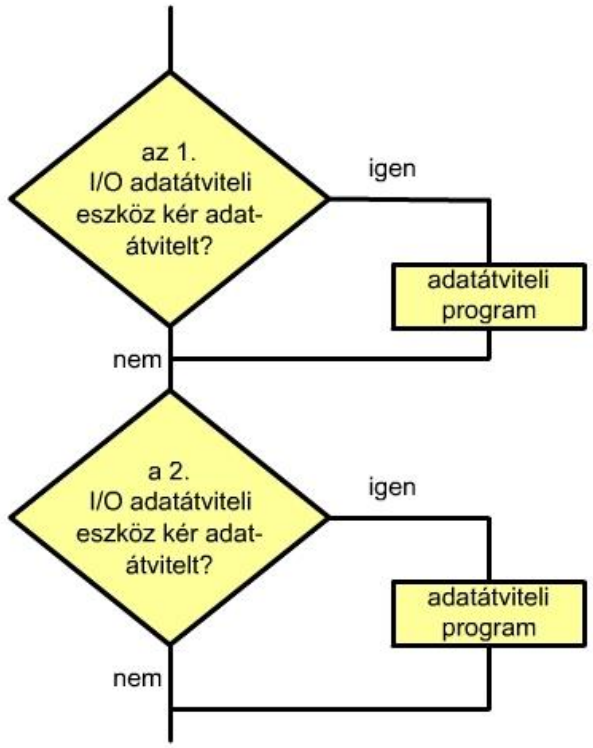
6.18. ábra. Feltételes I/O adatátvitel folyamatábrája.

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

szükséges adattranszfert, vagyis a példánkban az akkumulátorba olvassa a kettes címre kötött perifériából az adatot.

6.4.1.3. FELTÉTELES, KIVÁLASZTÁSOS (POLLING) I/O ADATÁTVITEL

Több I/O eszköz esetében a feltételes adatátvitel nem működik, hiszen egy eszköz már leállíthatja a többi periféria adatátvitelét. Ennek kiküszöbölésére használjuk a kiválasztásos, feltételes adatátviteli módszert, amely módszer a 6.19. ábrán látható.

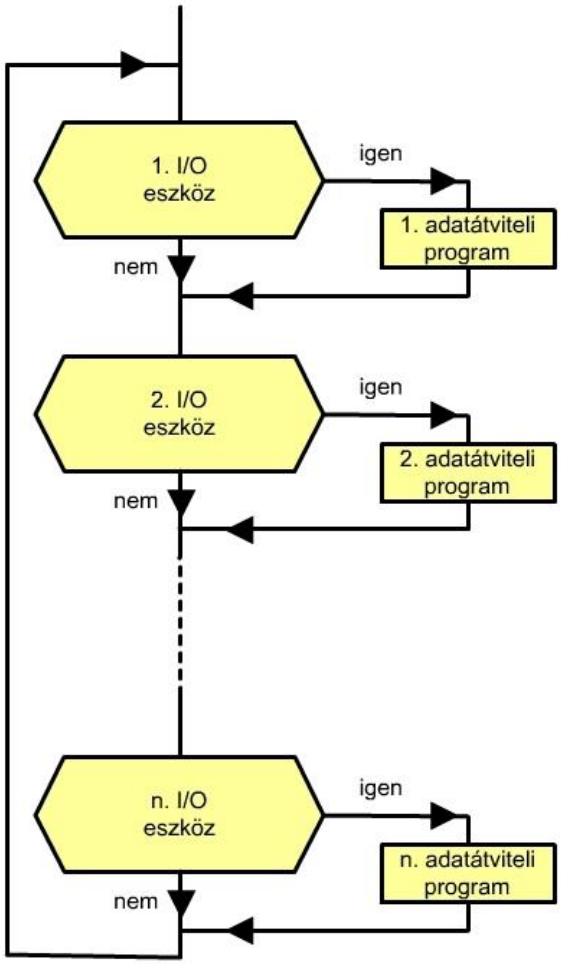


6.19. ábra. Feltételes, kiválasztásos I/O adatátvitel folyamatábrája.

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

A 6.20. ábrán látható egy teljes ciklus több I/O eszköz kiszolgálására a kiválasztásos (polling), feltételes I/O adatátviteli módszerrel. Ennek a módszernek előnye az, hogy minden eszközzel csak addig foglalkozik a program, míg meg nem állapítja, hogy szabad-e a periféria, viszont ez a módszer hátránya is, hiszen minden ciklusban minden eszközről meg kell állapítani ezt az

adatot, ami sok időt emészt fel. Ez különösen akkor zavaró, ha a perifériák ritkán képesek adatátvitelre.



6.20. ábra. A kiválasztásos, feltételes ciklikus adatátvitel folyamatábrája.

6.4.1.4. MEGSZAKÍTÁSSAL KEZDEMÉNYEZETT ADATÁTVITEL

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

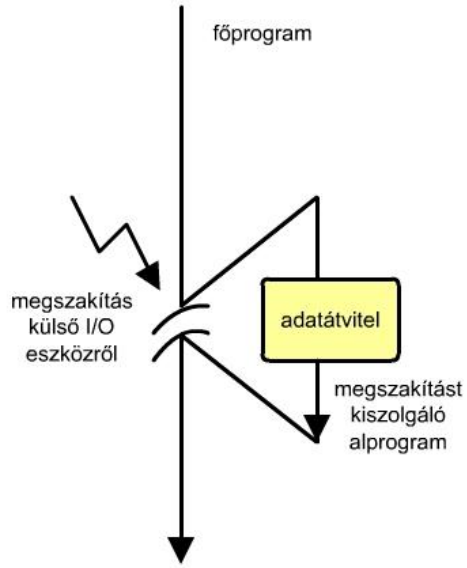
Ezért célszerűbb az adatátvitelnél a megszakítással kezdeményezett adatátvitelt alkalmazni, itt ugyanis csak akkor kerül sor adatátvitelre, ha azt az I/O eszköz megszakítás-igénnyel jelezte.

A főprogram akadálytalanul végrehajthatja az ott előírt feladatait, anélkül, hogy időt pazarolna a perifériák lekérdezésére.

Megszakítással kezdeményezett adatátvitelnél a főprogram végrehajtása, amennyiben ez engedélyezett megszakad, a vezérlés átkerül egy alprogramra, amely ellátja az igényelt adatátvitelt. Ez a folyamat a 6.21. ábrán látható.

A mikroszámítógép megszakításos rendszere aszinkron eseményekre is tud válaszolni, vagyis a külső I/O eszközök igényeire, úgy, hogy nem várakozik ciklusban az eszköz szabaddá válására. Mivel a mikroprocesszoroknak több megszakítást fogadó bemenetük van, illetve ha

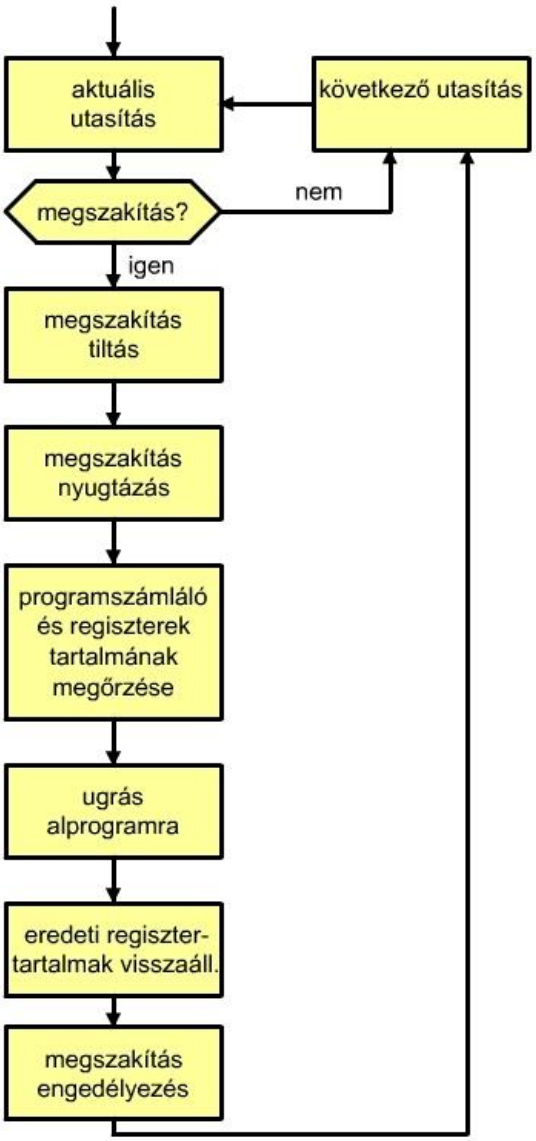
kevés a bemenetek száma az külső logikával bővíthető, több periféria kiszolgálása is lehetővé válik.



6.21. ábra. Megszakítással kezdeményezett adatátvitel vázlata.

A 6.21. ábrán követhető a megszakítással kezdeményezett adatátvitel folyamata, ahol feltételezzük, hogy előzőleg már programból engedélyeztük a megszakítás elfogadását:

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

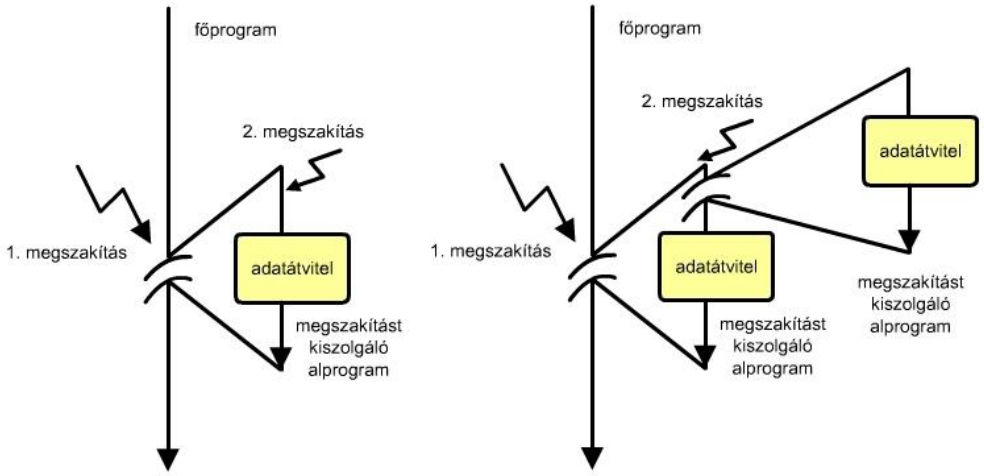


6.21. ábra. Megszakítással kezdeményezett adatátvitel folyamatábrája.

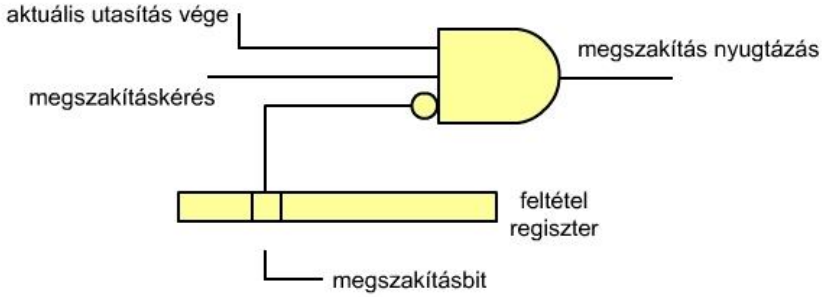
A számítógépek, mikrovezérlők lehetővé teszik a megszakítások maszkolását, valamint a megszakítások közötti prioritás, fontossági sorrend meghatározását. A 6.22. a) ábrán egy olyan eset látható, amikor két megszakításból csak az 1. engedélyezett (maszkolással), ilyenkor hiába keletkezik hardver jel formájában igény a megszakításra, ezt a számítógép nem fogadja el. A 6.22. b) ábrán két megszakítás engedélyezett, ez maszkolással biztosítható, valamint az első megszakítást a második megszakíthatja, vagyis a 2. megszakításnak nagyobb a prioritása.

Az I/O eszköz megszakítási igénnyel fordul a mikroprocesszorhoz, hogy az szakítsa meg az éppen futó programot és egy megfelelő adatátviteli alprogram segítségével végezze el az adatátvitelt. Amennyiben programból engedélyezett a megszakítás elfogadása, valamint az aktuális utasítás végrehajtása befejeződött, a processzor áttér a megszakítás feldolgozására.

Erről az eseményről értesíti a perifériát egy ún. megszakítás-nyugtázó jellel. A nyugtázó jel létrehozása a 6.23. ábra szerinti feltételek teljesülésével jön létre.



6.22. ábra. Megszakítások egymásba-ágyazása



6.23. ábra. A megszakítást nyugtázó jel feltételei

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

A módszer hátránya abban van, hogy a többforrásos és több prioritási szinttel rendelkező megoldások hardver kiegészítő elemeket igényelnek. Lassítja a megszakításos kiszolgálást az is, hogy minden megszakítás-kiszolgálás esetén program-számláló és regisztermentési műveletet kell végrehajtani, illetve a főprogramba való visszatéréskor ezen adatok

visszamentése is kötelező. A PC és regisztertartalmak mentése memóriareferens művelet, ami időigényes folyamat.

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

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

6.4.1.5. A MEGSZAKÍTÁS-FORRÁS AZONOSÍTÁSA

Mint már volt róla szó, a processzoroknak, mikrovezérlőknek általában több megszakítás-vonaluk van. Ha minden egyes megszakításhoz csak egy I/O eszközt rendelünk hozzá, a

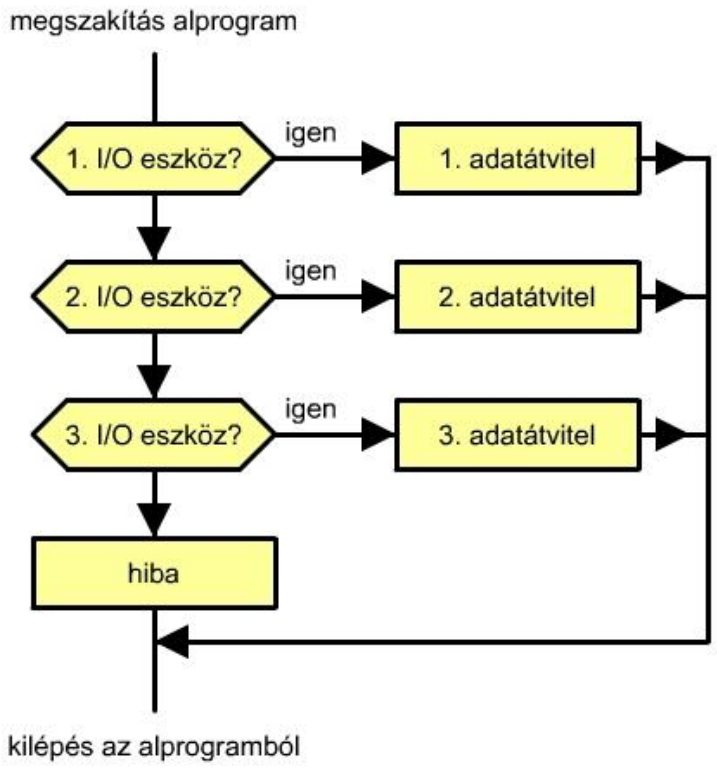
rendszer könnyen kezeli a kéréseket, hiszen minden megszakításnak címe van, könnyen azonosítható.

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

- az eszközök lekérdezése és
- vektoros megszakítás módszere.

6.4.1.5.1. A MEGSZAKÍTÁS-FORRÁS AZONOSÍTÁSA – ESZKÖZÖK LEKÉRDEZÉSE

Ennél a módszernél az adott megszakításhoz egy alprogram tartozik, amely a megszakítás hatására elindul, lekérdezéssel kideríti a megszakítás forrását, majd kiszolgálja az adott eszközt, vagyis végrehajtja az adatátvitelt. A folyamat a 6.24. ábrán követhető nyomon.



6.24. ábra. Megszakítás-forrás azonosítása lekérdezéssel

6.4.1.5.2. A MEGSZAKÍTÁS-FORRÁS AZONOSÍTÁSA – VEKTOROS MÓDSZER

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

6.4.1.5. ADATÁTVITEL KÖZVETLEN MEMÓRIAHOZZÁFÉRÉSSEL – DMA (DIRECT MEMORY ACCESS)

6.4.1.5.1. BEVEZETÉS

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

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

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

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

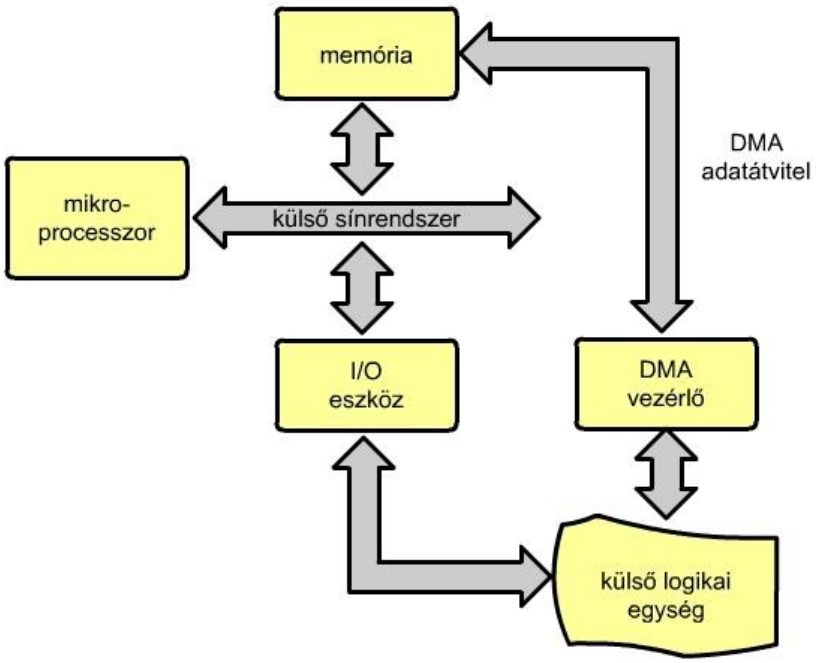
6.4.1.5.2. A KÖZVETLEN MEMÓRIA-HOZZÁFÉRÉS SZERVEZÉSE

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

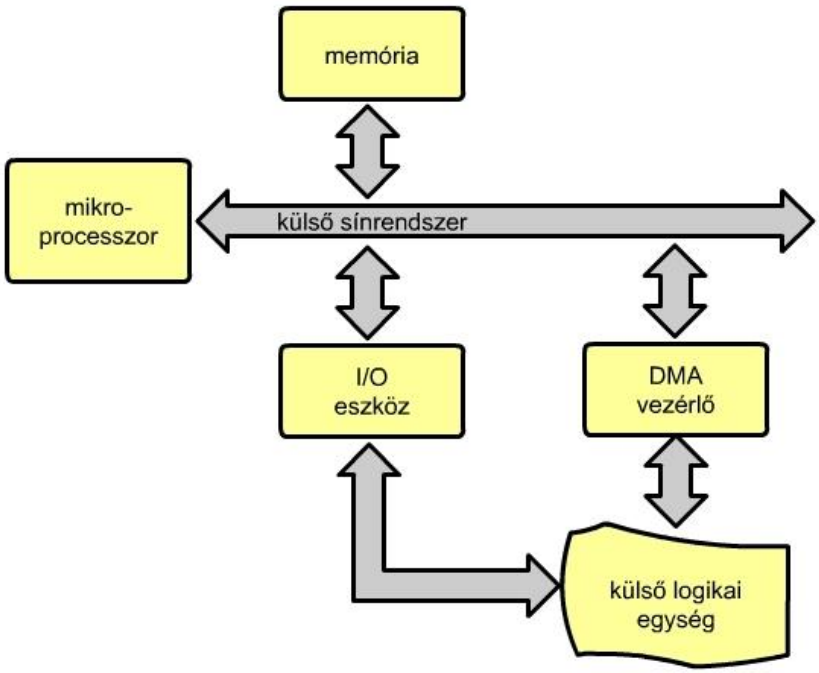
A 6.25. ábrán látható egy DMA egységet tartalmazó mikroszámítógép. Látható, hogy a periféria és a főtár közvetlenül van összekapcsolva, nincs kötődés a processzorhoz, illetve a buszrendszerhez.

Gyakorlati okok miatt, például amiatt, hogy a memóriáknak nincs, csak egy adatsínjük, ha lenne, akkor pedig amiatt, hogy a sínrendszert még egyszer meg kellene valósítani a nyomtatott lapon, valamint amiatt, hogy már eleve létezik a számítógépben a címsínt, adatsínt és

vezérlősínt tartalmazó sínrendszer, kihasználják a feladat megoldására ezeket a megoldásokat. A 6.26. ábrán ezen feltételek figyelembe vétele mellett kialakított DMA struktúrát láthatunk.



6.25. ábra. A DMA (Direct Memory Access) sematikus ábrázolása

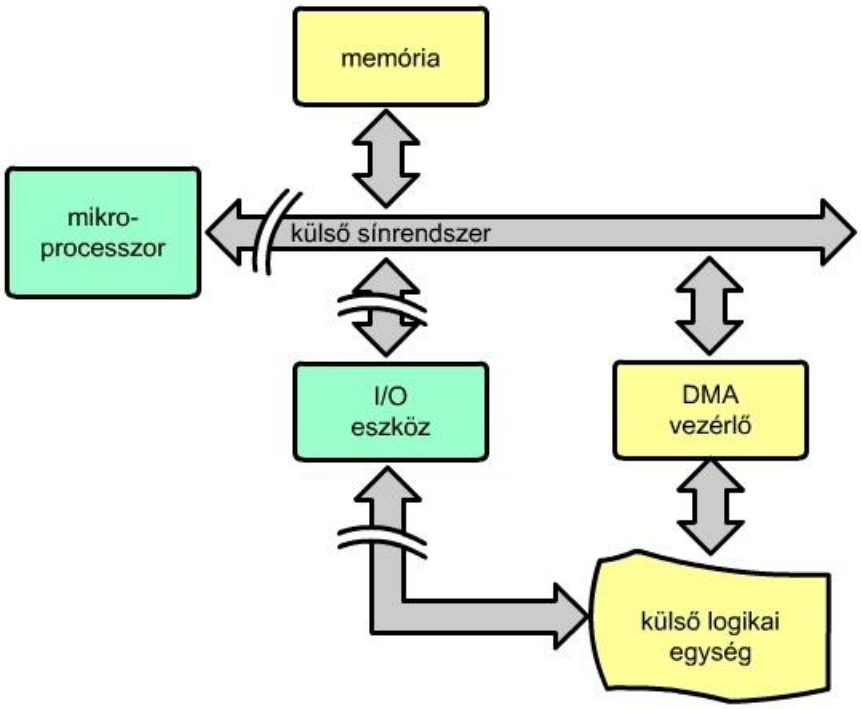


6.26. ábra. Egy valós, a meglevő sínrendszerre épülő DMA.

Amikor a mikroprocesszor végzi a rendszer vezérlését, vagyis közönséges programvégrehajtásról van szó, akkor a DMA vezérlő elektromosan a háromállású logika (three state) segítségével lekapcsolódik a mikroszámítógépről. Ez az állapot a 6.26. ábrán látható.

Akkor, amikor DMA típusú adatátvitel igénye merül fel, a különböző paraméterek, címek, adatok beállítását a CPU kezdeményezi és hajtja végre programok és adatok alapján. Megállapítja a DMA állapotát az állapotszó alapján, majd átküldi a DMA egységbe a memória kezdőcímét, az átvitelre kijelölt adatblokk hosszát, valamint a működéshez szükséges parancsszó tartalmát. Ezen információk átvitele után a CPU elindítja a DMA segítségével a közvetlen adatátvitelt. Ekkor a 6.27. ábra szerinti állapotba kerül a számítógép, látható, hogy a DMA az adatátvitel befejezéséig aktív egységként viselkedik. Az adatátvitel után újból I/O eszközként viselkedik (6.26. ábra).

Többféle módon lehet végrehajtani a DMA adatátvitelt:
DMA adatátvitel a processzor teljes leállítása mellett,
DMA adatátvitel ciklusloPásos elven,
DMA az előző két módszer egyesítésével és
DMA adatátvitel a CPU/DMA műveletek multiplexálásával.

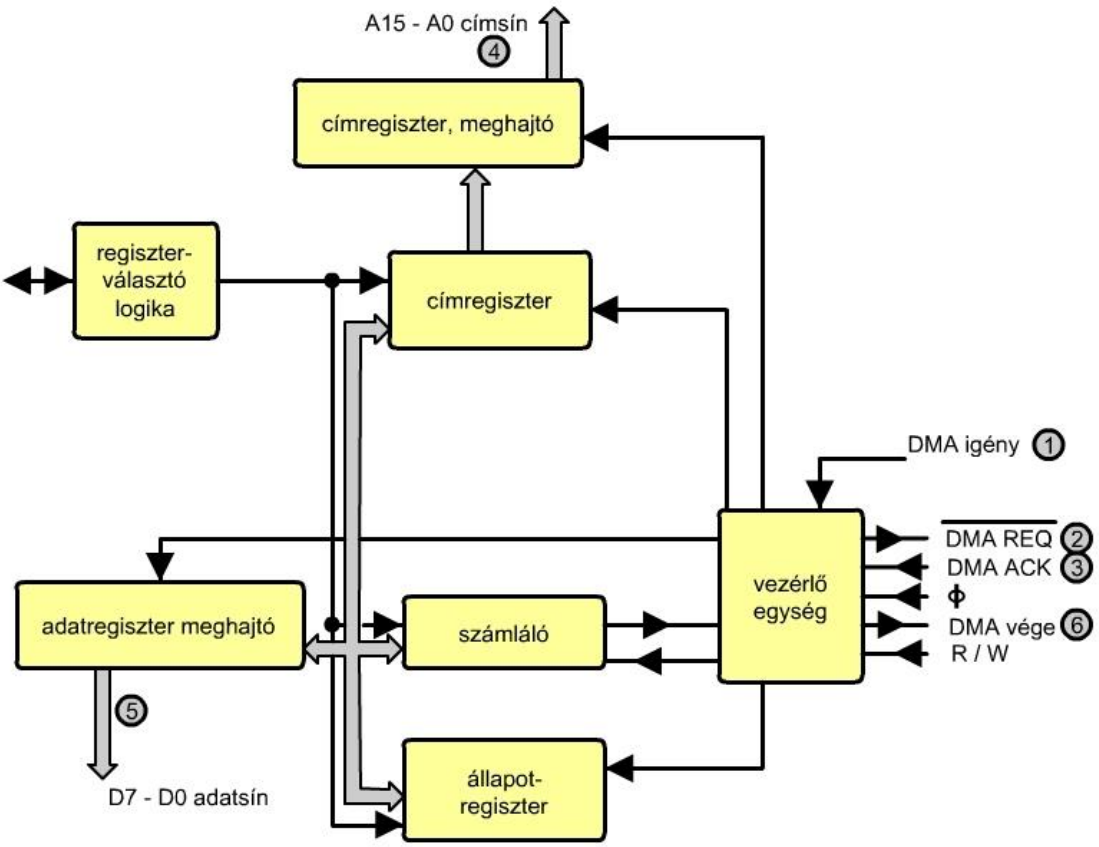


6.27. ábra. DMA adatátvitel.

- Ahhoz, hogy létrejöjjön a DMA adatátvitel a következő feladatokat kell megoldani:
- a címbusz vezérlése,
- a címbusz mikroprocesszorhoz, illetve DMA vezérlőhöz való hozzárendelése,

- az adatbusz vezérlése és
- a memória címzése.

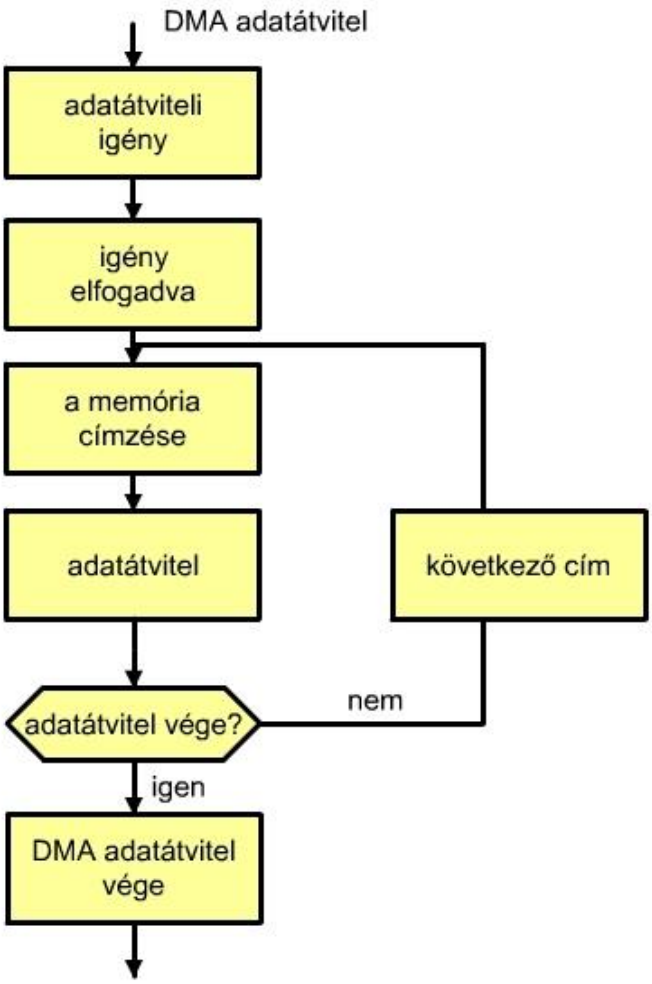
A CPU a programszámláló segítségével címezi a megfelelő memóriarekeszt. Ugyanezt a feladatot a DMA-nak is el kell látnia, ezért a DMA is tartalmaz egy címzésre alkalmas regisztert, illetve a regisztert inkrementáló logikát. Az átvitt adatok mennyiségének ellenőrzése, ez az átvitt adatblokk hosszának ellenőrzése, a folyamat befejezése, magának a DMA vezérlési módnak a meghatározása egy parancsregiszter segítségével történik. Ugyanitt található információ az átvitel irányára, arra, hogy aktív-e a DMA átvitel stb.



6.28. ábra. DMA vezérlő belső felépítése.

A DMA adatátvitel megérthető a 6.28. ábráról, ahol az egyes lépések számokkal vannak ellátva, a folyamat a 6.29. ábrán látható folyamatábrán követhető nyomon. Az egyes lépések a következők:

- az I/O eszköz logikai elemei létrehozzák a közvetlen memória hozzáférés igényét,
- az igényt továbbítja a logika a mikroprocesszor felé,
- nyugtázó jelet küld a DMA az adatátviteli igény elfogadásáról,
- memóriacímzés,
- adatátvitel a memória és a periféria között,
- a DMA adatátvitel vége.



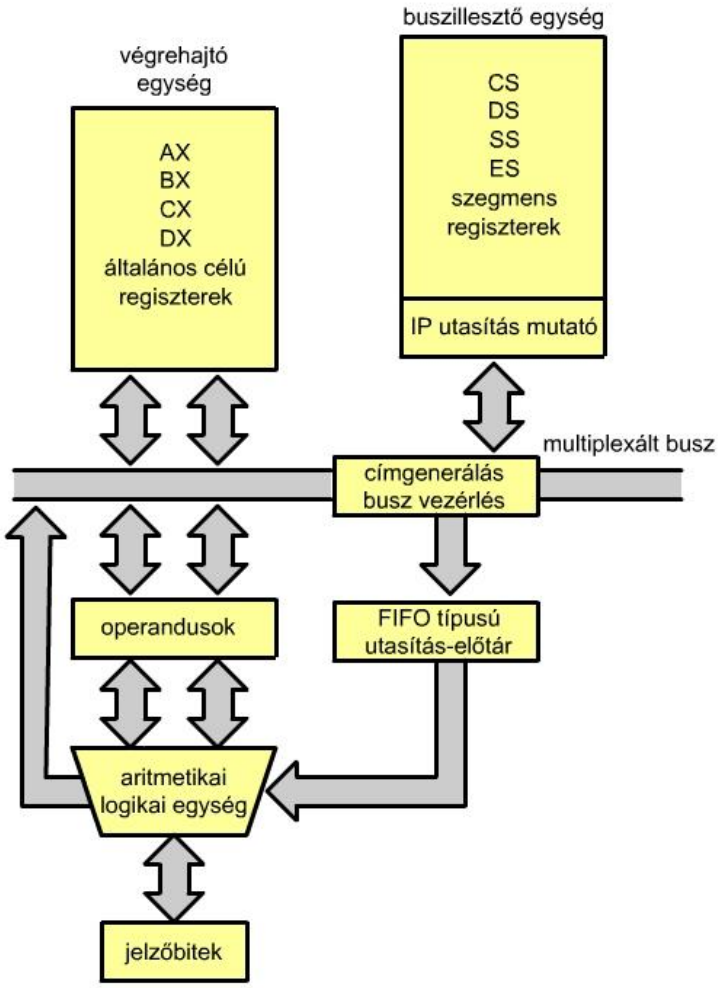
6.29. ábra. DMA adatátvitel folyamata.

7. AZ UTASÍTÁSOK

A 4.2. fejezetben már bemutattuk a gépi kódok felépítését, kapcsolatát a hardverrel. Ebben a fejezetben az Intel 8086-os architektúra segítségével pedig megnézzük, hogy a legalacsonyabb szinten hogyan lehet az utasításokkal az egyes hardver-elemek felhasználásával egyszerű műveleteket végrehajtani.

7.1. A 8086 HARDVER EGYSZERŰSÍTETT MODELLJE

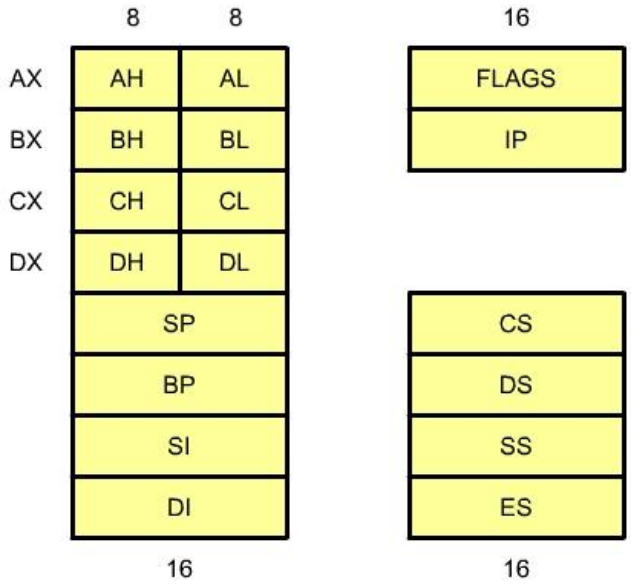
A processzor egyszerűsített, programozás számára kialakított modellje látható a 7.1. ábrán.



7.1. ábra. A 8086 egyszerűsített modellje.

A végrehajtó egységben levő AX, BX, CX és DX regiszter segítségével végezhetünk műveleteket, míg a négy szegmensregiszter tartalmát az operációs rendszer határozza meg:
CS (Code Segment) – kódszegmens, a program helye a fizikai memóriában,
DS (Data Segment) – az adatok helye a fizikai memóriában,
SS (Stack Segment) – a veremtár helye a fizikai memóriában és
ES (Extra Segment) – különböző tartalmak (például videomemória) a fizikai memóriában.

A programíráshoz a következő regiszterekre van szükségünk:

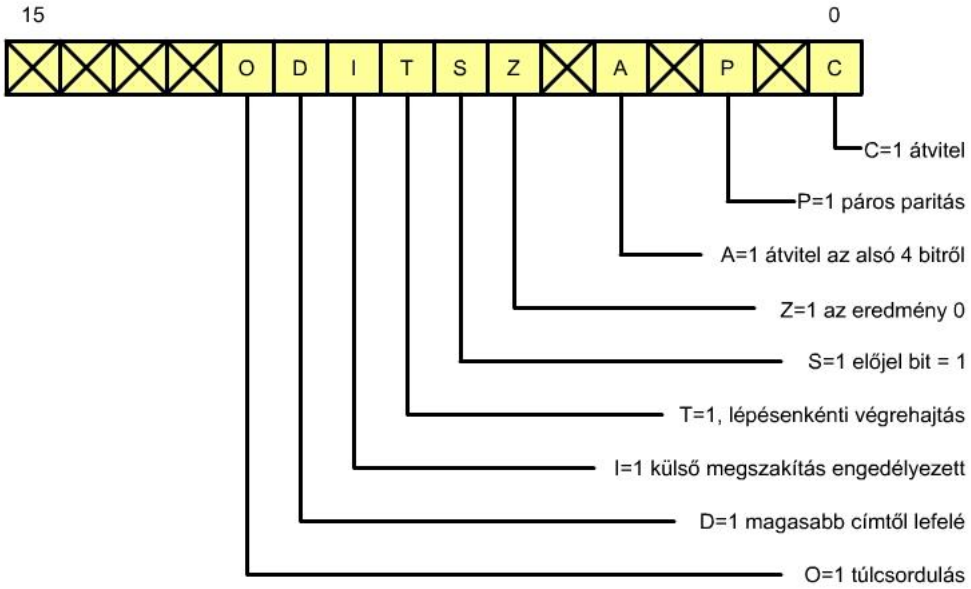


7.2. ábra. A programozáshoz használt regiszterek.

A processzor regiszterei a következők:

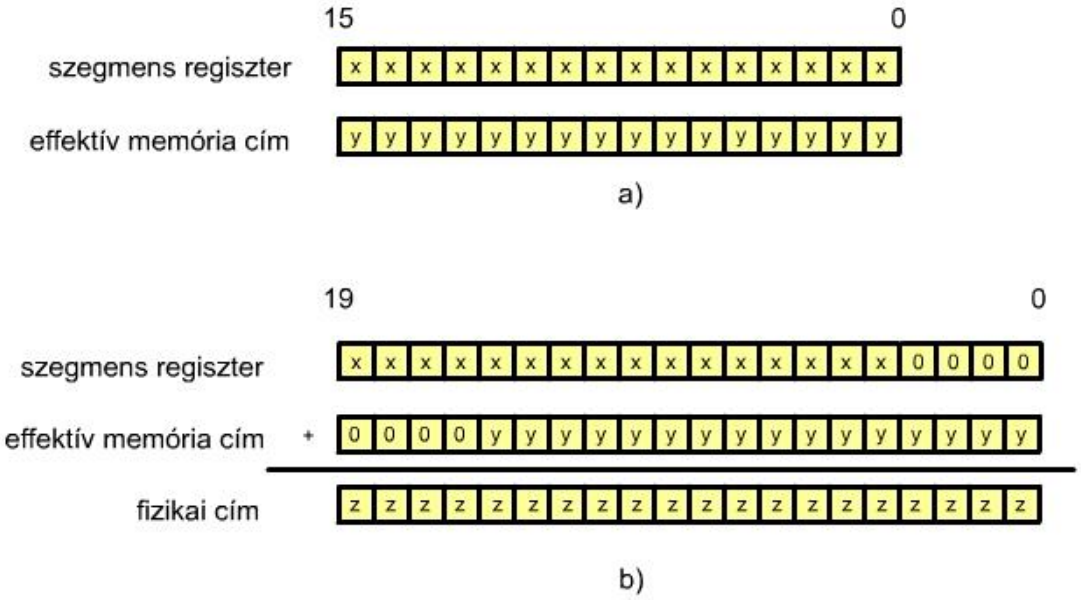
- AX – általános célú 16 bites akkumulátor, használható mint 2 8 bites akkumulátor is (AH és AL),
- BX – 16 bites bázis (base) regiszter, lehet mint 2 8 bites regisztert is használni (BH és BL),
- CX – 16 bites számláló regiszter (count), mint 2 8 bites regiszter is használható (CH és CL),
- DX – 16 bites adatregiszter (data), mely két 8 bites regiszterként is használható (DH és DL),
- SP – 16 bites veremtár mutató regiszter (Stack Pointer),
- BP – 16 bites bázismutató regiszter (Base Pointer),
- SI – 16 bites forrásindex regiszter (Source Index),
- DI – 16 bites célindex regiszter (Destination Register),
- FLAGS – 16 bites jelzőbit regiszter és
- IP – 16 bites utasítás mutató regiszter (Instruction Pointer),

Mint látható, a 16 bites AX, BX, CX és DX regiszter egyes utasításoknál nyolcbites regiszterként kezelhetők. A FLAGS nevű regiszter gyűjtőregiszterként tartalmazza a jelzőbiteket (negativ, átvitel, tólcsordulás stb.).
A jelzőbitek jelentése a következő:



7.3. ábra. Az egyes jelzőbitek jelentése és helye.

Mivel a 16 bites CS kódszegmens regiszter csak 64 kB memóriát címezhet a rendelkezésre álló 1 MB fizikai memóriából, ezért a teljes memória-címzés megvalósítása a következő ábra szerint történik:

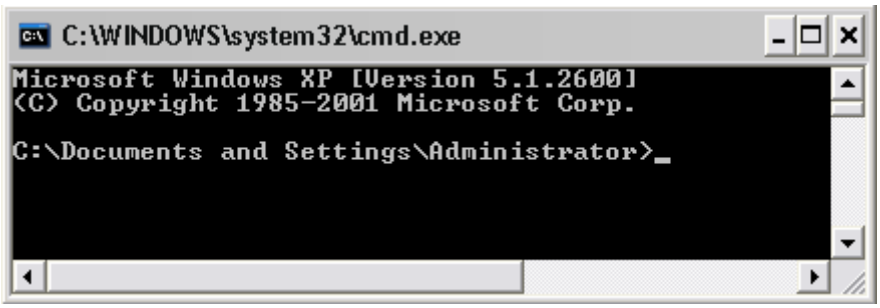


7.4. ábra. A fizikai 1 MB memória címzése eltolással.

Amint az ábrán látható, a hardver a CS szegmensregiszert balra tolja 4 bittel, ez tizenhattal való szorzást jelent (64 kB x 16 = 1 MB) , ehhez adja hozzá az aktuális programutasítás helyét (ezt IP utasítás mutató regiszter tartalmazza).

7.2 PROGRAMÍRÁS KÖZVETLENÜL A HÁTTÉRTÁRRA DOS SEGÍTSÉGÉVEL

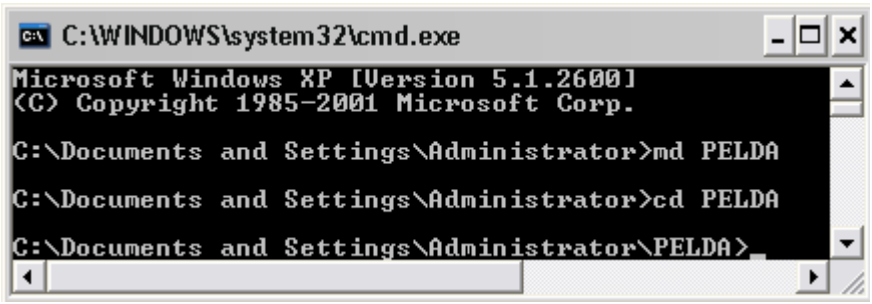
A futtatható program kiterjesztése exe kell hogy legyen, ezért létrehozzuk az elso.exe állományt. A Windows operációs rendszer START (bal alsó sarok) ikonjára kattintva megjelenik a FUTATÁS parancssor, ahova beírjuk a cmd parancsot és OK ikonra kattintunk. Megjelenik a következő, DOS ablak (természetesen valami hasonló, de ezzel nem teljesen megegyező szöveggel):



7.5. ábra. DOS ablak.

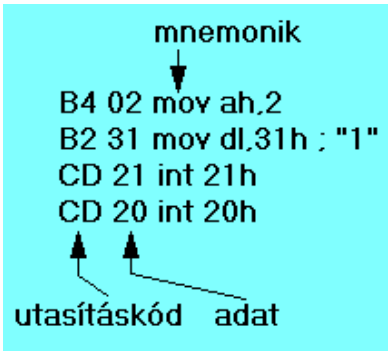
Ezután az md (make directory) paranccsal létrehozunk egy PELDA nevű alkönyvtárat és átváltunk erre az alkönyvtárra a cd (change directory) DOS utasítással:

md PELDA ENTER, cd PELDA ENTER, ami után a következő ablak jön létre:



7.6. ábra. A PELDA alkönyvtár.

A program, amelyet megírunk egy 1 szám nyomtatása. Ezt a programot következőképpen lehet megírni assembly nyelven:



7.7. ábra. Mintaprogram 1 szám nyomtatására.

A 05. MELLÉKLET-ben találhatók meg a 8086-os rendszer utasításai, csoportosítva. Itt az adatmozgató utasításoknál az első helyen szerepel a mov utasítás, ami adatmozgatást biztosít a rendszer elemei között. A mov ah,2 utasítás 02h értéket ír be az AX regiszter magasabb helyértékű helyére, ami egy parancs a harmadik sorban levő int 21h beépített alprogramnak, azt közli vele, hogy a képernyőre írás következik, a DX regiszter alcsonyab helyén levő érték ASCII kódja alapján. A mov dl,31h utasítással írjuk be a nyomtatandó karakter kódját a DX alcsonyabb helyértékű helyére. Az int 21h alprogram a képernyő következő helyére kiírja a karaktert, esetünkben az 1-et, az int 20h alprogram pedig biztosítja a szabályos visszatérést a DOS-ba.
Az ASCII (American Standard Code for Information Interchange) kódtáblázat a 03. MELLÉKLET-ben található, itt az 1 számnál az oszlopban a 011 érték található, a sorban pedig a 0001. Ez így csak hét bit, a 011 elé teszünk még egy nullát, így 0011 lesz, összekapcsolva a 0001 értékkel a 00110001b = 31h = 49d érték keletkezik, ami az 1 számnak az ASCII kódja.

A fenti (7.7. ábra) programot csak gépi kódban tudjuk közvetlenül a fájlba beírni, ezért szükséges ezek meghatározása, itt most nem közöljük a részletes értékeket, interneten ez megtalálható, a 7.7. ábrán levő első két hexadecimális szám adja ezt az értéket, mi ezt csak decimálisan tudjuk beírni. Átalakítva ezeket a kódokat tízes számrendszerbeli számokká, a következő értékeket kapjuk: 180, 2, 178, 49, 205, 33, 205, 32

Vigyük be az ELSO.EXE programba ezeket a számokat. Ezt a copy con elso.exe ENTER utasítással tehetjük meg, úgy, hogy az ALT billentyű folyamatos tartása mellett beírjuk a számot, ALT elengedése beírja a kódot, azzal, hogy a képernyőn az adott kódhoz rendelt karakter jelenik meg. Az utolsó szám bevitele után CTRL Z utasítással zárjuk be a fájlt, ami a PELDA alkönyvtárba másolja. Ellenőrizzük az elso.exe fájl meglétét a DIR paranccsal, a type elso.exe DOS parancs pedig a beírt kódokat kilistázza a képernyőre, ami számunkra érthetetlen jelek sorozata.

Ez a programunk gépi kódja, ezek után írjuk be az elso programnevet. Ha mindent jól csináltunk, megjelenik egy 1 szám a képernyőn.

7.3. A BEÉPÍTETT DEBUG PROGRAM HASZNÁLATA

A DEBUG programmal egyszerűen és gyorsan lehet létrehozni rövid assembly programokat, melyeket közvetlenül a főtárba tudunk beírni és ott futtatni. Indítása a DEBUG fájlnévvel történik, ami után egy ’-’ (godolatjel), a prompt jelenik meg. A ’?’ (kérdőjel) és ENTER begépelése megmutatja a parancsokat.

Az ’a’ parancs az assembly indítása, megjelenika CS tartalma és a program helye Itt kell megadnunk a mnemonikus kódokat az operandusokkal együtt, ami ennél a programnál csak hexadecimális érték lehet . Az ENTER után a következő sor beírása lehetséges, egészen a CTRL C befejező parancskódig. Mintapéldánk beírása után a kőpernyőn a következő jelenik meg:

```
164C:0100 mov ah,02
164C:0102 mov dl,31
164C:0104 int 21
164C:0106 int 20
164C:0108 ^C
-g
1
Program terminated normally
```

7.8. ábra. Példaprogramunk beírása a DEBUG programmal.

Az 164C cím a kódszegmens-cím, a 0100, 0102 stb. A gépi kód címe, ami most azért lép 2-vel, mert minden utasításunk 2 bájtos. A '-' prompt utáni 'g' és ENTER hatására a program végrehajtódik és a képernyőre kiíródik az '1' szám, majd egy üzenetet küld a képernyőre a DEBUG, a program helyesen fejeződött be, vagyis sikeresen visszaadta a vezérlést a DEBUG-nak. Ha hibásan fejezzük be a programunkat, akkor 'elszáll' a számítógép, vagyis a mi esetünkben nem tud DOS üzemmódban tovább dolgozni, a WINDOWS visszaveszi a vezérlést.

A DEBUG alkalmas a gépi kódokból visszaállítani az assembly programot, ahogyan azt a következő ábrán láthatjuk:

```
-u
164C:0100 B402          mov ah,02
164C:0102 B231          mov dl,31
164C:0104 CD21          int 21
164C:0106 CD20          int 20
164C:0108
```

7.9 ábra. a DEBUG disassembly utasítása.

A DEBUG program alkalmas a legalacsonyabb szinten, a fizikai memóriában való programfuttatásra, akár lépésenként is, továbbá memóriatartalmat listáz, módosít, regisztertartalmakat ír ki. Nem célunk részletesen ismertetni, itt csak néhány egyszerű példán mutattuk be használatát.

8. AZ OPERÁCIÓS RENDSZER (OPERATING SYSTEM – OS)

8.1. AZ OPERÁCIÓS RENDSZER FELADATA

A 3.1.3. fejezetben már foglalkoztunk az operációs rendszerrel. Ott adtunk egy rövid definíciót, ami így szólt:

„Az operációs rendszer egy rendszerszoftver, feladata az, hogy a számítógép felhasználók a számítógépet működtetni tudják, vagyis a hardver erőforrásokat megfelelő időben működésre bírják."

Nézzük meg részletesebben az operációs rendszer néhány főbb tulajdonságát. A hardver működtetését az operációs rendszernek a rendszerprogramok segítségével úgy kell biztosítani, hogy az könnyen történjen meg a felhasználó részéről, az adott szoftver az erőforrásokat sokoldalúan hozza működésbe, természetesen nagy figyelmet fordítva a biztonságra is. A felhasználó az egyes felhasználói programokat egységesen kell, hogy tudja használni, természetesen a programoknak is egységes támogatást kell, hogy nyújtson a szoftver.
Az operációs rendszer több, egymástól erősen eltérő feladatot kell, hogy megoldjon, ezek:

- memóriakezelés,
- perifériakezelés,
- állományok kezelése,
- a felhasználónak egységes, könnyen kezelhető felület,
- a különböző folyamatok ütemezése, végrehajtása,
- a működés közben fellépő hibák kezelése és a
- a folyamatok naplózása.

8.1.1. MEMÓRIAKEZELÉS

A programok merevlemezen helyezkednek el, amikor indítjuk azokat, akkor a főtárba töltődnek be. Mint ahogyan azt láttuk az 5.9. fejezetben, a fizikai memória (főtár) mérete kisebb, mint amennyire szükség lenne az összes elindított program számára, ezért kerül sor a virtuális memória megszervezésére, ezt végzi el az operációs rendszer.

8.1.2. PERIFÉRIAKEZELÉS

A perifériák, tehát a be- kiviteli eszközök a külvilág és a számítógép között tartják a kapcsolatot, adatcserét végezve azzal. Ezeket az eszközöket vezérli az operációs rendszer.

8.1.3. ÁLLOMÁNYOK KEZELÉSE

A perifériákról jövő adatok alakja, időzítése, szervezettsége erősen eltér, ezt kell valamilyen szabványos formába ölteni, ezt a feladatot látja el az állományok kezelése.

8.1.4. FELHASZNÁLÓI FELÜLET

Itt egyrészt az operációs rendszer a felhasználónak biztosít egy átlátható, könnyen kezelhető felületet, de ugyanakkor a futtatandó programoknak is, hiszen egyszerre több programot is kell indítani és működtetni.

8.1.5. FELADATOK ÜTEMEZÉSE, VÉGREHAJTÁSA

Az elindított programok a felhasználó számára párhuzamosan futó programok, az operációs rendszer pedig egy soros feldolgozású számítógépen kell hogy ezeket futtassa, ennek szervezését kell elvégeznie, mikor melyik programrész mikor futhat.

8.1.6. HIBÁK KEZELÉSE

Mind a hardvernél, mind a programoknál (szoftvernél) felléphet hiba, ennek, vagy ezeknek a hibáknak nem szabad, hogy veszélyeztessék a számítógép működését, tehát nem állhat le a számítógép. Ide tartozik a véletlen, vagy szándékos hibás adatkezelés, nem megengedhető, hogy elvesszenek, vagy idegen kézbe kerüljenek adataink, programjaink.

8.1.7. ESEMÉNYEK NAPLÓZÁSA

Az előző fejezetekben felsorolt események feljegyzése, naplózása azért fontos, mert ebből visszakereshetők az események okai, pontos megjelenési ideje, sorrendje, amiből a jövőre tekintettel vonhatunk le következtetéseket, ezek elkerülésére.

8.2. OPERÁCIÓS RENDSZEREK BESOROLÁSA

A felhasználók számára legismertebb a PC személyi számítógép, illetve a ma már az okostelefon és tablet is, de azért ne feledkezzünk meg arról, hogy igen sok helyen egészen más bonyolultságú és képességű számítógépek terjedtek el. A következőkben többféle szempont alapján csoportosítjuk az operációs rendszereket, főleg a PC személyi számítógépeket véve figyelembe.

Egy lehetséges felosztás a felhasználok száma és a futtatott folyamatok száma alapján:

- egy feladat, egy felhasználó,
- több feladat, egy felhasználó és
- több feladat, több felhasználó.

Felosztható az operációs rendszerek a felhasználás alapján:

- asztali gépek és
- szerverek.

Besorolhatjuk az operációs rendszereket a terjesztésük alapján:

- szabad terjesztés és
- kereskedelmi terjesztés.

8.3. A LEGELTERJEDTEBB OPERÁCIÓS RENDSZEREK

A PC személyi számítógépeknél három rendszer jelenleg a legelterjedtebb:
WINDOWS, Microsoft termék, fizetős,
LINUX, szabad terjesztésű, ingyenes és
MAC OS X, fizetős.

Érdemes még megemlíteni a LINUX rendszeren alapuló JOLI OS, JOLICLOUD operációs rendszert is, mely felhasználóbarát, sok régebbi gépen is hibátlanul fut.

A tablet, okostelefon kategóriában a következő három rendszer a legelterjedtebb:

IPHONE OS, APPLE gépek operációs rendszere,
ANDROID, LINUX alapú operációs rendszer és
WINDOWS okostelefon operációs rendszer.

Mind a PC-nél, mind az okostelefon, tablet kategóriában a gyártók ügyelnek arra, hogy a különböző adatállományok (szöveg, zene és kép) mindegyik operációs rendszer által kezelhető legyen és az adott gépek szoftverei kezeljék azokat.

8.3.1. WINDOWS OPERÁCIÓS RENDSZER

Legelterjedtebb a PC személyi számítógépeknél, a WINDOWS név az ablakos megjelenítésre utal, amely könnyű kezelést tesz lehetővé. Folyamatosan fejlesztik, mindig új változatok kerülnek forgalomba.

8.3.2. LINUX OPERÁCIÓS RENDSZER

A WINDOWS operációs rendszerrel ellentétben grafikus csomagok nélkül is telepíthető, így parancs-soros kapcsolatot biztosít a felhasználóval, ezt a megoldást inkább szervereknél alkalmazzák. Asztali gépeknél természetesen az ablakos alkalmazás az elterjedtebb. Több verziója van, létezik verzió, melyet cégek írnak, de internet közösségek is tartanak fel LINUX verziókat.

8.3.3. MAC OS X OPERÁCIÓS RENDSZER

Az APPLE cég gépeinek operációs rendszere, ez a legkevésbé elterjedt, egyik oka a számítógép viszonylag magas ára.

8.3.4. IPHONE OS OPERÁCIÓS RENDSZER

Tulajdonképpen a MAC OS X okostelefonra implementált változata, az okostelefon mellett az APPLE IPAD készülékének is az operációs rendszere.

8.3.5. ANDROID OPERÁCIÓS RENDSZER

Gyakorlatilag egy LINUX felett futó, szintén ablakos rendszer. Okostelefonoknál és tableteknél találkozunk vele, alacsonyabb ára miatt ma már elterjedtebb, mint az APPLE IPHONE.

8.3.6. WINDOWS MOBIL OS OPERÁCIÓS RENDSZER

A WINDOWS mobil telefonokra implementált változata, jóval kevésbé elterjedt, mint az előző két operációs rendszer.

8.4. AZ OPERÁCIÓS RENDSZER FELHASZNÁLÓI FELÜLETE

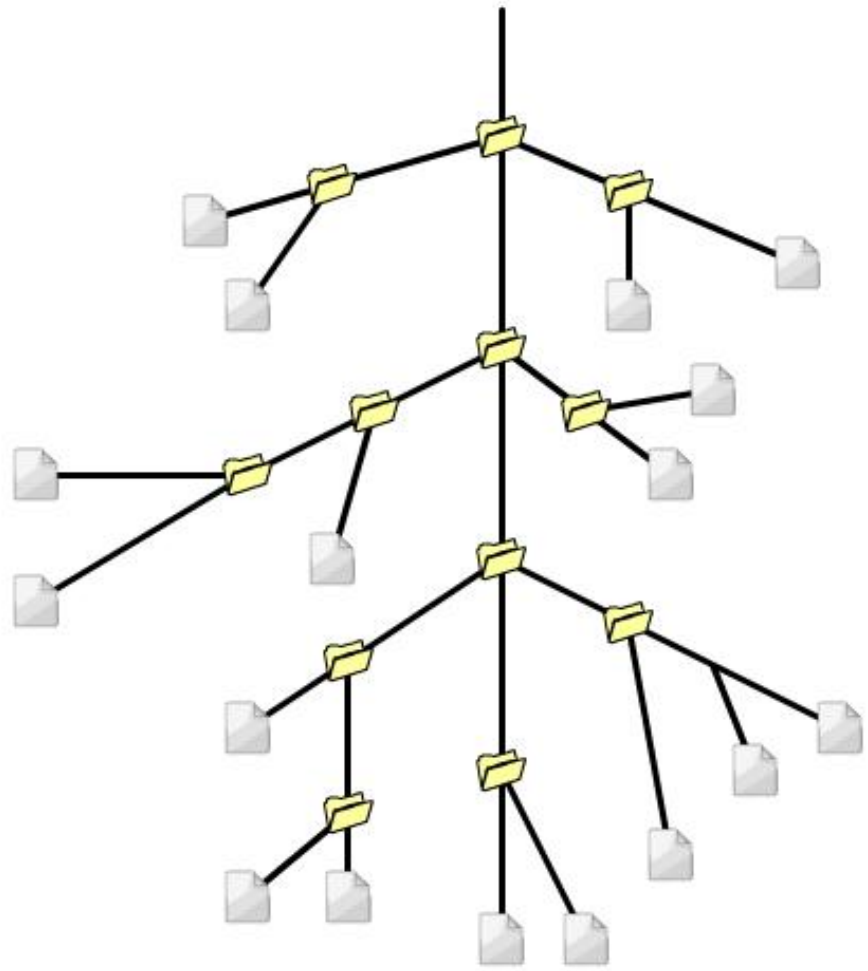
A felhasználó a gép bekapcsolása után ezzel a felülettel találkozik először, a képernyőn ikonok jelennek meg, melyek mögött programok és állományok vannak. A programok, állományok indítása egérrel, billentyűzettel, érintő felülettel történhet meg. Tableteknél, okostelefonoknál érintőképernyő segítségével, ceruzával, újjal történhet a programok indítása.

Szervereknél általában parancssorba történő kulcsszavak, paraméterek begépelésével indítható a megfelelő feladat.

Meg kell említeni az olyan számítógép alapú berendezéseket is, amelyek nem rendelkeznek sem képernyővel, sem billentyűzettel, sem egérrel, de igénylik a felprogramozást, működtetést. Ilyenek a különböző hálózati eszközök, kamerák, műholdvevők, hálózati nyomtatók stb., de az ipari berendezések egy része is, PLC-k, szabályozók, stb. . Általában valamilyen kapcsolaton keresztül, például hálózat, rákapcsolódunk a PC gépünkkel az adott berendezésre, annak vezérlője egy képernyőt, menü-rendszerben jelentet meg a PC képernyőn, ahol elvégezhetjük a beállítást, vagy kezelést.

8.5. KÖNYVTÁR, ALKÖNYVTÁR, FÁJL ÉS FÁJLRENDSZER

A különböző típusú állományok (programok, adatállományok) tárolását a merevlemezen igényeink szerint szervezhetjük fa struktúra alakjában. Itt fájlokról van szó, ezek programok, szöveges állományok, képek, hangok stb. lehetnek. Nevük mellett, ponttal elválasztva kiterjesztésüket is meg kell adni (vagy a program eleve hozzárendeli a fájlhoz a neki megfelelő kiterjesztést). Valamilyen közöttük levő logikai kapcsolat alapján hozzuk létre a fájlrendszert, könyvtárakat, az alattuk levő alkönyvtárakat. Az alkönyvtárak tovább ágaztathatók, ez mind egy fára hasonlító struktúrát alakít ki.



8.1. ábra. Fa-struktúra merevlemezen állományok tárolására.

9. SZÁMÍTÓGÉP-PERIFÉRIÁK

Nem teljesen egységes a témakör tárgyalása, eltérő definíciók találhatók az irodalomban a periféria meghatározásra. A problémát bonyolítja, hogy nagyon sok felhasználónak (joggal) a számítógép szó hallatán a hagyományos PC (személyi számítógép) jut az eszébe, hiszen a felhasználók nagy része csak ezzel találkozik. A valóságban a számítógép, mint láttuk sokkal elterjedtebb, de nagyon sok alkalmazás rejtve marad a számunkra, gondoljunk csak egy mai, modern gépkocsira, ahol kis, egyszerű feladatok helyi kezelésére akár többtíz számítógép-alapú berendezés is található.

Mi a jegyzetben minden számítógéphez kapcsolt eszközt, amelyik biztosítja a kapcsolatot az ember, más számítógépek és más berendezések felé, perifériának nevezzük, így a monitort, klaviatúrát stb. is ide soroljuk.

Ezek szerint a periféria egy olyan számítógépes hardver, amivel egy számítógép képes adatforgalmat lebonyolítani emberrel/másik számítógéppel/másik berendezéssel. Természetesen a szükséges perifériák mellett opcionális perifériák is felhasználásra kerülnek,

amelyek sokszor csak ideiglenes, szükség szerint kapcsolódnak a számítógéphez (hordozható merevlemez stb.).

Ezek az eszközök számítógépes buszon keresztül csatlakoznak a számítógéphez, a buszok egy része csak „dobozon", házon belül érhető el (pl PCI busz), míg egy részük külső csatlakozóval is rendelkezik (pl. USB).

A perifériák egy része hosszú ideig használatban van (pl. a billentyűzet), de van köztük olyan is, amely viszonylag gyorsan kikerül a használatból (Lyukszalag).

A fejezetben a következő felosztás szerint tárgyaljuk ezeket az eszközöket:

- háttértárak,
- bemenetek,
- kimenetek és
- hálózati eszközök.

9.1. HÁTTÉRTÁRAK

Fizikai elhelyezkedésük szerint megkülönböztetünk a számítógép-házon belüli és azon kívüli eszközöket. Természetesen sok esetben egy-egy periféria mindkét kivitelben is megtalálható, gondoljunk csak a merevlemezre.

9.1.1. BELSŐ HÁTTÉRTÁRAK

9.1.1.1. MEREVLEMEZ (HDD)

Ez az eszköz egy kör alakú tárcsán, hordozón elhelyezkedő mágneses felületen tárolja a biteket, a mágneses anyag úgynevezett keménymágnes, ami azt jelenti, hogy a kis részek, domének íráskor 0 és 1 értéktől függően egyik, vagy másik irányban mágneseződnek, ebből bármikor visszaolvasható a 0 vagy 1 érték. A lemezhez közel, 1 nm-es légpárnán egy kombinált író-olvasó fej helyezkedik el. Ez a távolság rendkívül kicsi, csak pormentes környezetben szerelhető össze a berendezés. Maga a tárcsa állandóan forog, leggyakrabban a következő sebességekkel:

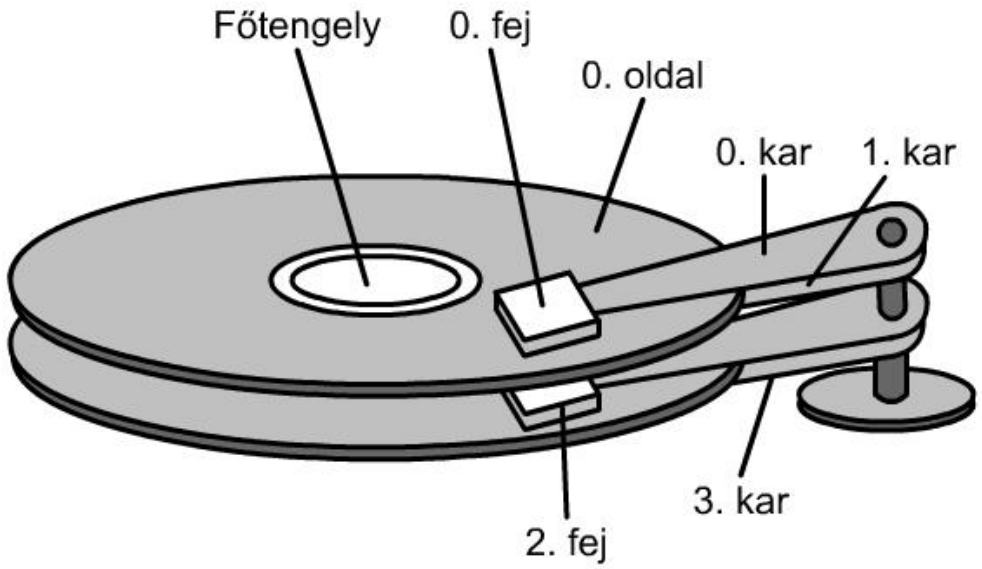
5400,

7200,

10.000 vagy

15.000 fordulat/perc (rpm – rotations per minute).

A jobb kihasználtság érdekében a lemez mindkét oldala mágneses, így alul-felül Író/olvasó fej található, valamint ugyanazon a tengelyen több tárcsa is van.



9.1. ábra. Merevlemez mechanika.

Mivel a merevlemeznél a tárcsák ugyanazon a tengelyen vannak, ezért minden tárcsán azonos központú, különböző sugarú kör van, a sávok (trackek). A sávok azonosítása számokkal történik, kívülről kezdve, a 0-ás sorszámútól. Az egymás alatt elhelyezkedő sávok cilindert alkotnak. A sávon belül találhatók a szektorok. Ezek számozása eggyel kezdődik. A winchester 2-4 szektort (a szektoroknak a száma mindig 2-nek valamelyik hatványa) együtt kezel, ezek alkotják a klasztert (cluster-t).

A merevlemez-meghajtó tárolási kapacitása a technológia fejlődésével egyre jobban nő, a kezdeti néhány MB (megabájt) kapacitás ma már eléri TB (terrabájt) nagyságrendet. A viszonylag nagy tárolókapacitás mellett lassú elérési idő jellemzi az egységet, ez persze függvénye a lemezek forgási sebességének is, valahol a ms, 10 ms nagyságrendben található. Mint ahogyan azt az 5.7. és 5.8. fejezetben láttuk, periféria-gyorsító cache memória beépítésével lehet ezt az adatot javítani.

A merevlemez egység csatlakoztatása is fontos kérdés, különféle megoldásokat alkalmaznak, az ATA (PATA) - párhuzamos, SATA (SATA I, SATA II, SATA III) - soros, SCSI, SAS (Serial Attached SCSI), FC (Fiber Channel).

A fizikai méretük részben összefügg alkalmazási területükkel is, így jellemzően 5400 fordulat/perc sebességű és 2.5 hüvelykes a hordozható számítógépekbe épített változat, 3.5 hüvelykes és 7200, vagy 10.000 fordulat/perc sebességű az asztali gépek merevlemez egysége.

Felvetődik a kérdés, hogyan lehetséges az, hogy még ma is, amikor már aránylag olcsó, nagy kapacitású félvezető háttértárakat építenek, merev-lemezt használunk? Ezeknél az egységeknél nincs probléma az újraírhatósággal, gyakorlatilag végtelen sok ciklust bírnak ki, ezzel ellentétben a félvezetőknél még problémás a többszöri újraírás (5.2. fejezet). Erősen terjednek

a kombinált, hibrid megoldások, ahol a gyors beolvasást a félvezető rész biztosítja, míg a nagymennyiségű adat továbbra is mágneses lemezen tárolódik.

A merevlemez egység particionálásával több logikai egység hozható létre ugyanazon a fizikai egységen. Formattálással kell az egységes mágneses felületen kialakítani a sávokat és a szektorokat. Idővel megjelenik a töredezettség, ami azt jelenti, hogy az egységes adathalmazok különböző, távoli fizikai helyekre kerülnek (a szektorokban sok üres hellyel), ami miatt lassú a beolvasásuk, ezt célszerű időnként megszűntetni.

9.1.2. CSERÉLHETŐ MÉDIA

Sokszor van igény egyik számítógépről másikra adatot átvinni, vagy tartósan egy-egy állományt megőrizni. Ekkor olyan megoldást kell találni, amelynél könnyen kezelhető, kis méretű és fizikai terhelést elviselő hordozót alkalmazunk. Ezek közül tárgyalunk néhányat.

9.1.2.1. CD (COMPACT DISC) - KOMPAKTLEMEZ

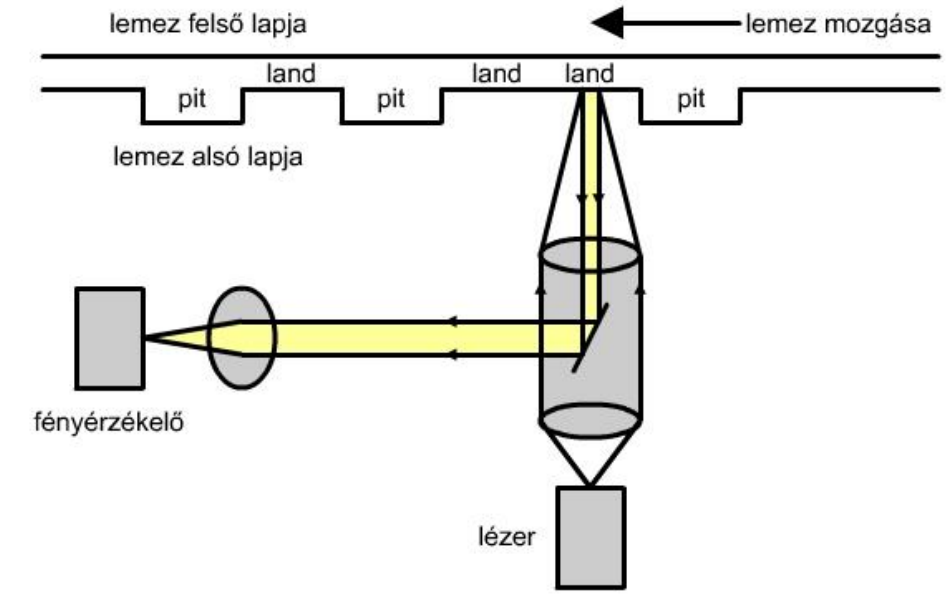
Egy 120 mm (ritkán 80 mm) átmérőjű és 1.2 mm vastagságú polikarbonát tárcsa felülete alatt vékony szerves festékréteg helyezkedik el. Adatírásra lézerdióda szolgál, 780 nm-es hullámhosszúságú fénnyel égetik be az információt. A fókuszált lézersugár az anyagba pit-et, kis pontot, krátert éget be. Kiolvasáskor csökkentett energiájú lézerfénnyel történik a megvilágítás, a visszavert fényt optoelektronika veszi, a pit-ről kevésbé verődik vissza a fény, mint a környezetéről. A hirtelen fényváltozás logikai 1 értéket jelent.

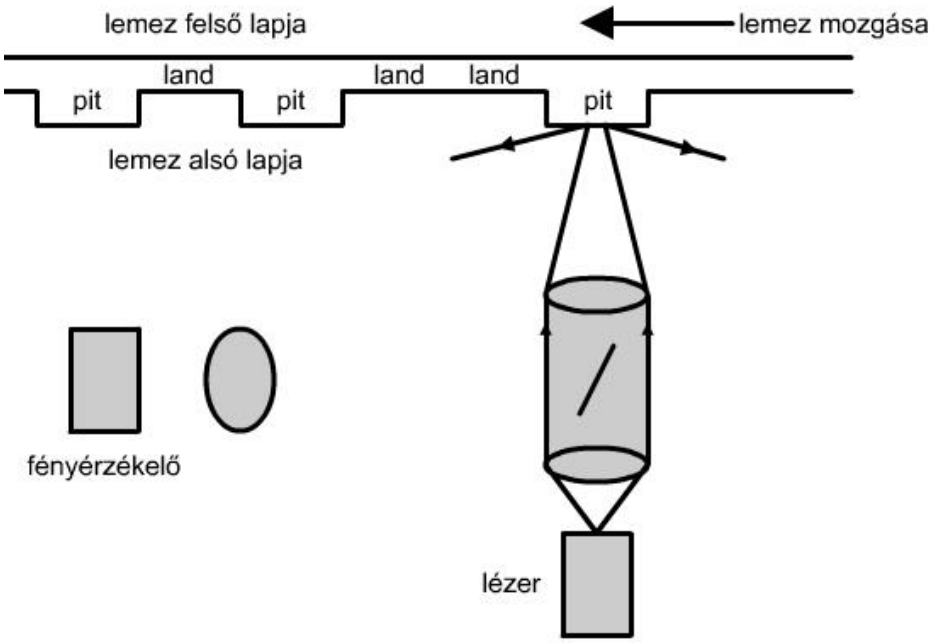
9.2. ábra. CD lemezek

Az audio (hanganyagot tartalmazó) CD kissé különbözik a digitális jeltárolástól. A következő kivitelben készül a digitális CD:

- préselt (csak olvasható),
- CD-R, egyszer írható és
- CD –RW, előző tartalom törölhető és újraírható.

A CD-RW típusú lemez olvasófeje különbözik a fent leírt CD-R olvasófejétől. A lemez anyaga íráskor megváltozik, de törléskor visszaalakul az eredeti formájúvá. Ezeknek a CD-knek az anyaga egy tükröződő felület, ami különleges festékkel van bevonva. Az író/törlő/olvasó fej három, különböző erősségű lézersugárral rendelkezik. Olvasáskor gyenge, a CD anyagát nem módosító sugárra van szükség, íráskor erős, a felületet módosító, míg törléskor közepes, az eredeti állapotot visszaállító energiájú sugarat kell használni. Az író lézer felmelegíti a festékréteget, ami az opálossá válik, erről a lézersugár nem tud visszaverődni. A törlő lézersugár is felmelegíti a réteget, de csak annyira, hogy az lehüléskor ismét átlátszó legyen.





9.3. ábra. A CD-ről való információ olvasása

A CD-ROM (Compact Disc Read Only Memory) meghajtó a számítógépbe épített egység, amelybe helyezzük a CD korongot.

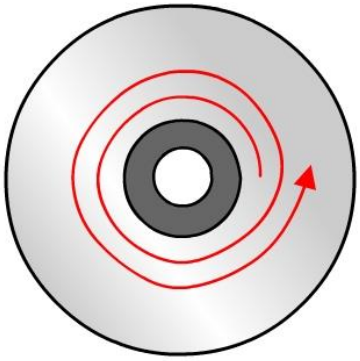
Itt is, mint más eszközöknél a technológia fejlődése egyre gyorsabb működési sebességet eredményez. Eredetileg 200 – 530 fordulat/perc forgási sebesség jellemezte a készüléket, de ezt erősen megnövelték, adatokat a következő 9.1. táblázat tartalmaz.

9.1. táblázat. CD egységek sebessége és olvasási sebessége.

Sebesség	Olvasási sebesség [kB/s]	Elérési idő [ms]
1X	150	400–600
2X	300	200–400
3X	450	180–240
4X	600	150–220
6X	900	140–200
8X	1200	120–180
10X	1500	100–160
12X	1800	90–150
16X	2400	80–120
20X	3000	75-100
24X	3600	70–90
32X	4800	70–90
40X	6000	60–80
52X	7800	60–80

Egy CD-re írható adatmennyiség 700 MB, valójában azonban ennél több információt tartalmaz az adatbiztonság miatt.

Az adatok egy úgynevezett radiális csigavonalon helyezkednek el, az adatolvasás mindig belülről kifelé történik. Az író/olvasó fejet egy léptető motor mozgatja, mikron pontossággal.



9.4. ábra. CD olvasása

9.1.2.2. DVD (DIGITAL VERSATILE DISC) – DIGITÁLIS SOKOLDALÚ LEMEZ

Előző, CD-hez hasonló optikai adathordozó, felülről kompatibilis vele. Külső méreteiben megegyezik a CD-vel. Hasonlóan a CD-hez, itt is megkülönböztetünk:

- préselt (csak olvasható),
- DVD-R, egyszer írható és
- DVD –RW, előző tartalom törölhető és újraírható.

Meg kell jegyezni, hogy többféle megoldás i létezik az előző három csoportban, a tárolható adatmennyiség 7-25-szöröse a CD-nek. Két 0.6 mm-es lemez összeragasztásával készül, így mindkét oldalon tárolható adat. A következő kivitelben és tárolási kapacitásban léteznek:

- 1 oldalas, 1 rétegű dvd 4.7 Gbyte-os
- 1 oldalas, 2 rétegű dvd 8,54 Gbyte-os
- 2 oldalas, 1 rétegű dvd 9,4 Gbyte-os
- 2 oldalas, 2 rétegű dvd 17,08 Gbyte-os .

9.1.2.3. PENDRIVE

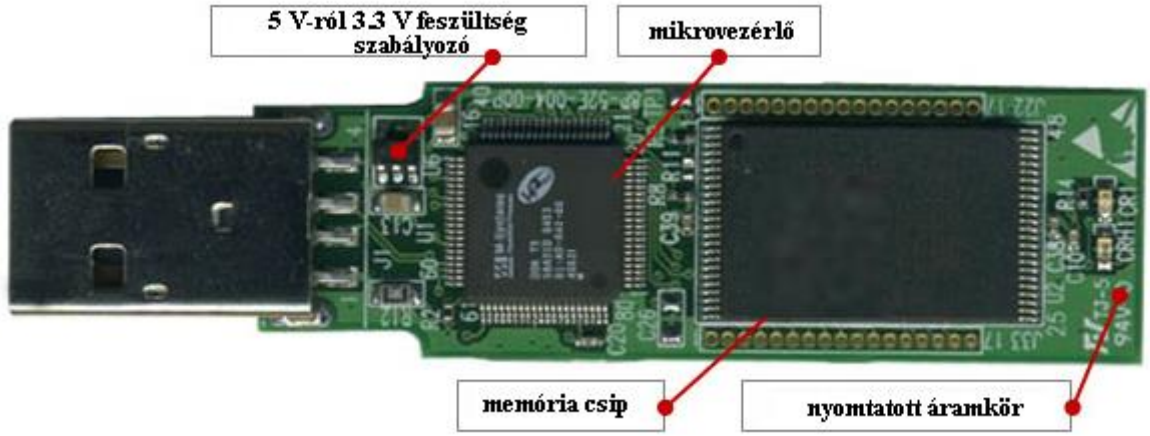
Az eszköz FLASH típusú félvezető memórián őrzi az adatokat. A FLASH memória működése részletesen az 5.2. fejezetben található meg. Többféle elnevezéssel lehet találkozni, leggyakrabban a PENDRIVE, vagy USB PENDRIVE megnevezéssel jelölik. Gyors elterjedését és széleskörű felhasználását abban kereshetjük, hogy kisméretű, viszonylag nagymennyiségű adat tárolására alkalmas, de nem utolsó sorban csatlakozása PLAG AND PLAY "A" típusú USB csatlakozóval (6.3. fejezet). Különböző kapacitású memóriákkal kapható, ez 8 MB-tól 256 GB-ig terjedhet. Az irodalomban találkozni olyan adattal, hogy akár 10 évig is megtarthatja a beírt adatokat, valamint százazer és egymillió közötti írási ciklust is kibír. Fontos az is, hogy míg kezdetekben az operációs rendszer külön meghajtót (drivert)

igényelt a kezeléséhez, ma már erre nincs szükség, az operációs rendszerek támogatják használatát, azonnal felismerik.

Az adatátviteli sebesség függ a szabványtól, a következő táblázat tartalmazza az értékeket:

9.2. táblázat. USB átviteli sebességek.		
Elnevezés	Adatátvitel legfeljebb [MB/s]	Szabvány
Alacsony sebesség	0.150	USB-1.1, USB-2.0, USB-3.0
Teljes sebesség	1.2	USB-1.1, USB-2.0, USB-3.0
Nagy sebesség	48	USB-2.0, USB-3.0
Szuper nagy sebesség	400	USB-3.0

A következő ábrán az eszköz belső felépítését láthatjuk.



(Forrás: http://usbthumbdrivebootable.blogspot.com/2012/01/inside-usb-flash-drive.html)

9.5. ábra. PENDRIVE belső felépítése.

Látható, hogy ennél a típusnál az adatforgalmat egy mikrovezérlő, tehát egy számítógép bonyolítja le. Az ábra bal oldalán, a fémes felület szolgál a mechanikus csatlakozásra, valamint ezen a fémburkolaton belül van a négy vezeték csatlakozója (GND, VCC, D+ és D-).

Az eszközt a legkülönbözőbb alakban találhatjuk meg, hagyományos kiviteltől kezdve a legelképesztőbb megoldásokig.

9.6. ábra. Néhány PENDRIVE megoldás

9.1.2.4. MEMÓRIAKÁRTYÁK

Nagyon sokszor van szükség viszonylag nagy kapacitású memóriákra, melyeket nem túl gyakran helyezünk be egy eszközbe, vagy távolítunk el onnan, ellentétben az előző fejezetben tárgyalt PENDRIVE-val szemben. Okostelefonok, fényképezőgépek, multimédia-központok, kisebb beágyazott rendszerek háttértár-kapacitását lehet így megoldani. A felsorolt készülékek eleve kisméretűek, ezért a memóriának is kis fizikai mérettel kell rendelkezni. A következő alfejezetekben ezek közül tárgyalunk néhányat.

9.1.2.4.1. COMACT FLASH (CF) MEMÓRIAKÁRTYA

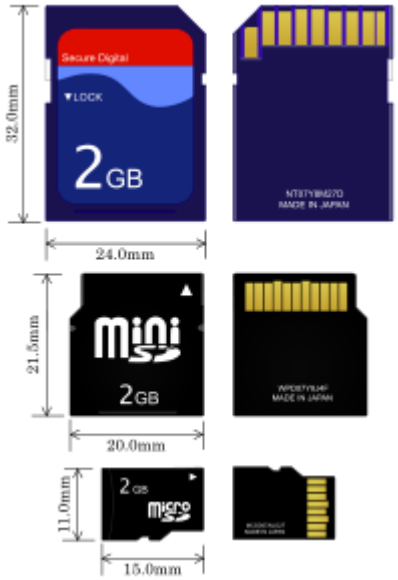
Ezt a típust kezdték el gyártani a legkorábban, Type I. és Type II. elnevezéssel (ma csak Type II létezik). Köztük a különbség az adatírási- és adatolvasási sebességben van. Kapacitásuk 1 MB és 100 GB között lehet. Mint ahogyan a nevében is benne foglaltatik, ez is FLASH félvezető memóriát tartalmaz.

9.7. ábra. Egy COMPACT FLASH memória.

9.8. ábra. COMPACT FLASH memóriakártya beágyazott rendszeren.

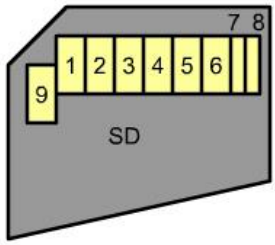
9.1.2.4.2. SECURE DIGITAL (SD) MEMÓRIAKÁRTYA

Ma az egyik legelterjedtebb memóriakártya, viszonylag nagy kommunikációs sebessége miatt. Ezek a kártyák kisebb készülékekben használatos, PDA, digitális fényképezőgép, kamera, navigációs eszközök stb. Létezik egy kisebb, miniSD változata, mely nem túlságosan terjedt el, szemben a microSD változattal, amely igen népszerű.

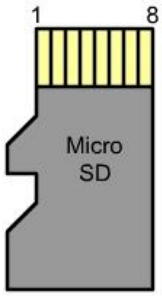


9.9. ábra. SD, miniSD és microSD memóriakártya és méretei.

A lábkiosztás az összes SD (és a következő fejezet SDHC) kártyáira érvényes.



Pin	SD	SPI
1	CD/DAT3	CS
2	CMD	DI
3	VSS1	VSS1
4	VDD	VDD
5	CLK	SCLK
6	VSS2	VSS2
7	DAT0	DO
8	DAT1	X
9	DAT2	X



Pin	SD	SPI
1	DAT2	X
2	CD/DAT3	CS
3	CMD	DI
4	VDD	VDD
5	CLK	SCLK
6	VSS	VSS
7	DAT0	DO
8	DAT1	X

9.10. ábra. SD, SDHC és microSD kártyák lábkiosztásai.

A CS – chip select az eszközválasztó jel, DI – data in, DO – data out, a SCLK – serial clock, vagyis soros órajel, a VSS a föld, a VDD pédig a táplálás az ábrán.

9.1.2.4.3.SECURE DIGITAL HIGH CAPACITY (SDHC) MEMÓRIAKÁRTYA

Tulajdonképpen ez egy SD 2.0 szabvány, mely a kapacitást 2GB – 32 GB érték között határozza meg, továbbá a Class 2, Class 4 stb. felirattal közli az adatátviteli sebességet, ami azt jelenti, hogy a szám a Class felirat után MB/s értéket jelent. Egy példa, a Class 6 6 MB/s sebességet jelent. Alkalmazásakor győződjünk meg arról, hogy a készülékünk támogatja-e az átviteli sebességet, mérete, csatlakozása megegyezik az egyszerű SD kártyával. Már megjelentek a legalább 64 GB méretű, 20 MB/s adatátviteli sebességet biztosító változatok is.

9.1.2.5. EGYÉB MEMÓRIAKÁRTYÁK

Természetesen léteznek egyéb típusú memóriakártyák is, sokszor egy nagyobb, ismert cég próbál eszközeihez fejleszteni ideálisabban illeszkedő megoldást, ez vagy sikeres, vagy elég hamar kiszorul a piacról.

9.1.2.5.1. SOLID STATE DRIVE (SSD) – SZILÁRDTEST-MEGHAJTÓ

Valószínűleg a mágneses elven működő merevlemez-meghajtót felváltó eszközről van szó. Itt is, mint az előző fejezetekben tárgyalt memóriakártyáknál FLASH félvezető típusú memóriával oldják meg az adattárolást, ugyanakkor szoktak még SRAM, vagy DRAM memóriát is használni, ekkor RAM DRIVE-ról beszélünk. Nincs mozgó alkatrész, gyakorlatilag viszont egy ugyanolyan, ugyanúgy a számítógéphez csatlakozó háttértárról (másodlagos tárolóról) van szó, mint a Winchester. A FLASH memóriás megoldásnál nem kell gondoskodni állandó táplálásról, míg a jobban elterjedt DRAM egység állandó táplálást igényel, amit akkumulátorral oldanak meg.

Természetesen a mechanika hiánya miatt gyorsabb az adatátvitel.

A következő táblázatban hasonlítjuk össze a hagyományos, mágneses-mechanikus merevlemez néhány tulajdonságát a szilárdtest meghajtóval.

9.3. táblázat. Hagyományos és szilárdtest meghajtó összehasonlítása.

	hagyományos merevlemez-meghajtó	szilárdtest-meghajtó
indulási idő	akár 1 s	nincs
mozgó alkatrész	van	nincs
olvasási várakozási idő	5.5 – 12 ms	12.5 µs
írási várakozási idő	5.5 – 12 ms	33 µs
olvasási sebesség		520 MB/s
írási sebesség		320 MB/s
áramfelvétel	nagy	kicsi
zaj	mechanika miatt van (fej)	nincs
mechanikai igénybevétel	érzékeny ütésre, vibrációra, nyomásra	nem érzékeny ütésre, vibrációra nyomásra
környezeti hőmérséklet	5 C° – 55 C°	-40 C° - 85 C°
írási-olvasási teljesítmény	változó	állandó
méret	nagyobb	kisebb

Meg kell jegyezni, hogy a tárolt egységnyi adatmennyiség még drágább, mint a hagyományos merevlemeznél, de előbb felsorolt előnyei miatt már laptopokban is, egyre jobban terjed az alkalmazása, tabletekben, okostelefonokban pedig egyértelműen ezt alkalmazzák.

Léteznek olyan merevlemez megoldások, ahol a hagyományos merevlemezt kiegészítik az SSD-vel, a gyors adatcsere miatt, a nagymennyiségű adat tárolása pedig a hagyományos, mágneses részen történik.

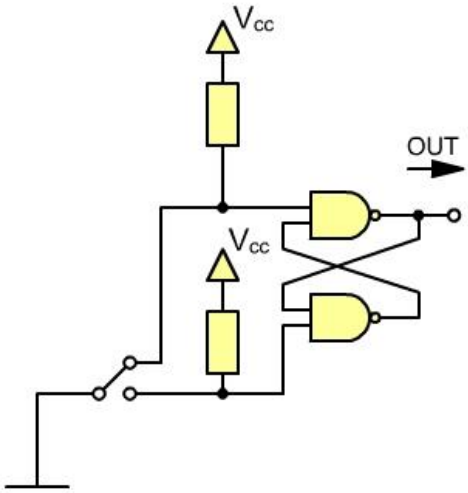
9.11. ábra. Egy 3.5 ”-es hagyományos merevlemez és egy 2.5 ”-es szilárdtest merevlemez.

9.2. BEMENETI PERIFÉRIÁK

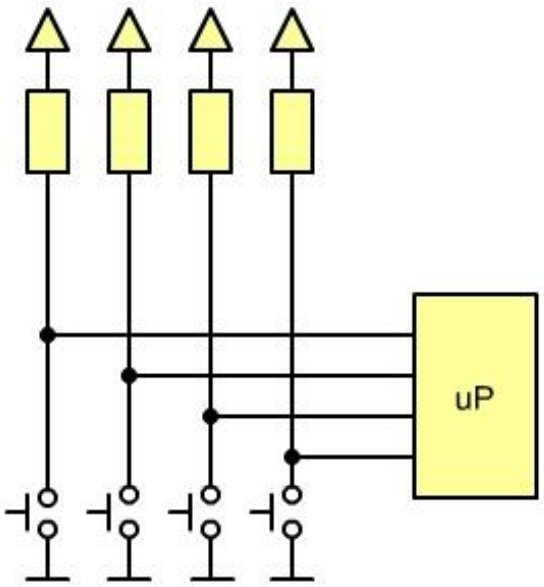
A számítógép a külvilágból igen különböző típusú és mennyiségű információt olvas be. A bemeneten analóg és digitális jelek lehetnek, néhány bittől egészen nagy mennyiségű adatig. A következő alfejezetekben ezek közül a legjellegzetesebb egységekkel foglalkozunk, figyelembe véve a személyi számítógépnél szokásosan használt egységeket, de kitérünk néhány ipari megoldásra is. A felsorolás nem teljes.

9.2.1. ÉRINTKEZŐK, KAPCSOLÓK (BILLENTYŰK)

Egyszerűbb, mikrovezérlőt tartalmazó berendezések sokszor csak néhány érintkezőről olvasnak információt. Ilyenkor az érintkezőről, kapcsolóról jövő jel nem ideális, pereg. A pergésmentesítést hardver, vagy szoftver úton tudjuk megoldani. Az áramkörök minimalizálása miatt ma gyakrabban alkalmazzuk a szoftveres megoldást.



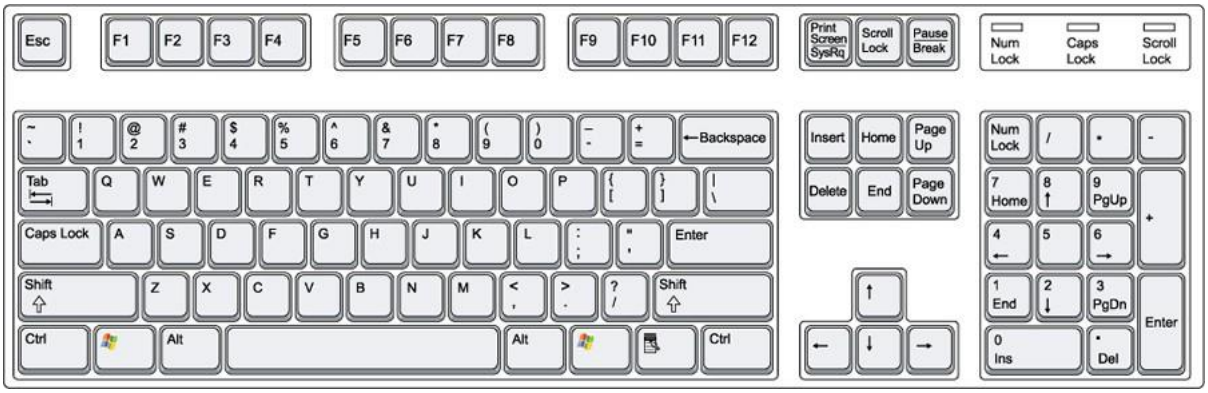
9.12. ábra. Információ olvasása érintkezőről, hardver pergésmentesítés.



9.13. ábra. Felhúzó ellenállás alkalmazása érintkezőről jövő információ olvasására

9.2.2. KLAVIATÚRA, SZÁMÍTÓGÉPES BILLENTYŰZET

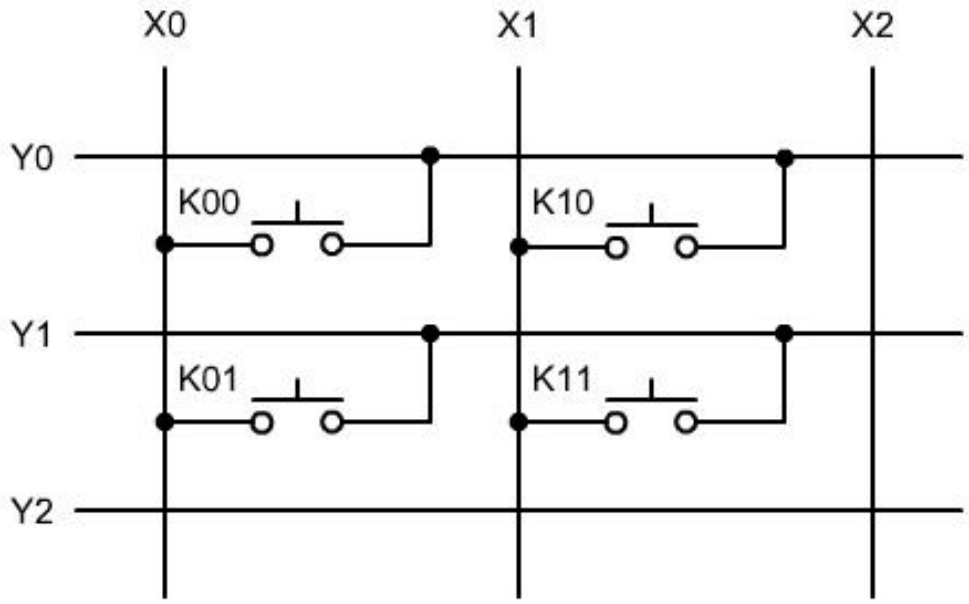
Formáját, elrendezését az írógéptől örökölte. Ezen keresztül tudunk szöveges információt, betűk, számok és írásjelek formájában bevinni. Alkalmas még számítógép-parancsok végrehajtására a speciális billentyűk segítségével. Az eredeti kiosztás az angol nyelvhez igazodik, de szoftver segítségével adott nyelvhez átrendezhető, így a magyar nyelvhez is. Általában egy billentyű több karakter bevitelét is ellátja, valamilyen speciális kiegészítéssel.



9.14. ábra. PC billentyűzet

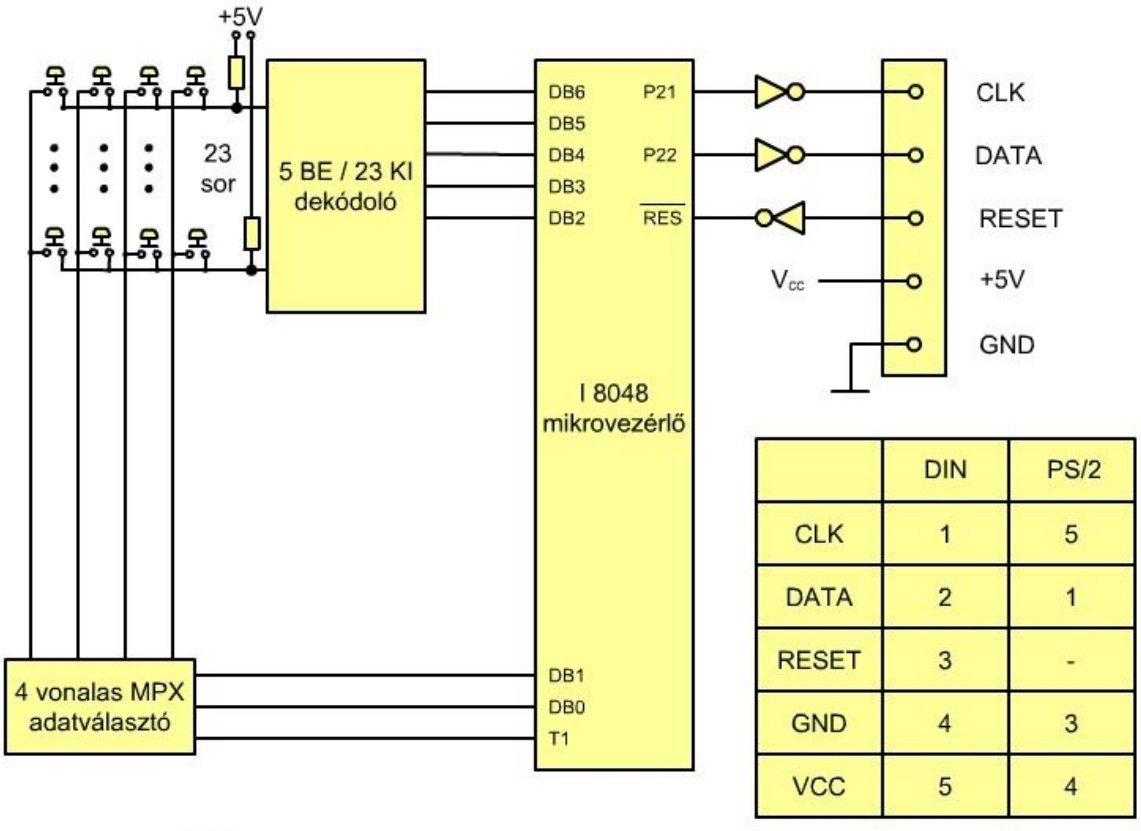
A PC billentyűzet egy Intel 8048-as mikrovezérlővel ellátott egység, amelyik sorosan kommunikál a PC-vel. A billentyű általában négyszögletes alakú, rugó nyomja vissza a leütött gombot, biztosítva az érintkezést elektronikusan, mikrokapcsolóval, mágneses, vagy optikai elven. Az egyszerű klaviatúrától kezdve egészen az ütés- és vízálló változatok nagyon sokféle kialakítása van. Mivel a PC klaviatúrákon általában 100 feletti elemet használnak, ezért az aktivált billentyűt nem egyenkénti kereséssel azonosítják be, hanem n X m-es mátrixba kötve,

ez lényegesen csökkenti a leolvasást és csökkenti a leolvasás időtartamát. A mátrix egy-egy sora kap 1-es információt, a sorhoz tartozó billentyű beazonosítása után a következő sor kapja meg az egyest és így tovább.



9.15. ábra. Klaviatúra-mátrix kapcsolási rajza.

A fenti billentyűzetet összekötve a mikrovezérlővel kapjuk a következő kapcsolási rajzot:

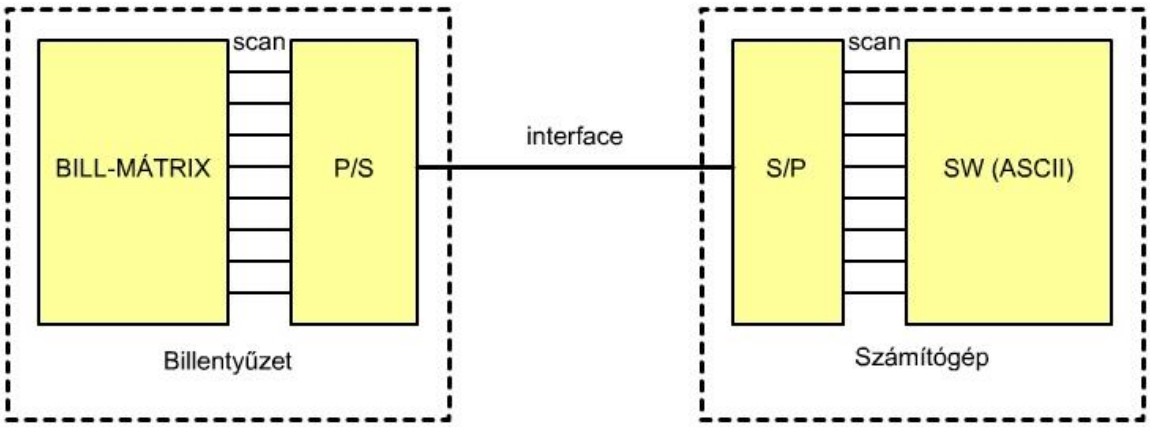


	DIN	PS/2
CLK	1	5
DATA	2	1
RESET	3	-
GND	4	3
VCC	5	4

9.16. ábra. A billentyűzet jeleinek feldolgozása és továbbítása mikrovezérlővel.

Az aktivált billentyűhöz a mikrevezérlő hozzárendeli a billentyűkódot (a kód neve SCAN-kód), megszakítást kér és megszakításos adatátvitellel (6.4.1.4. fejezet) küldi a PC processzorába a kódot. Amennyiben a PC fogadja a megszakítást, a SCAN kódot a mikrovezérlő ASCII kóddá alakítja (0.3. MELLÉKLET) és továbbítja. Ha a PC képtelen fogadni a kódot, egy 16 bájtos FIFO típusú memóriába kerül, ha ez megtelik sípoló hangot ad. Szoftverben beállítható többek között egy ismétlési késleltetés, amivel a billentyűzet lenyomott gombjának tartásával ismételten generálja a karaktert és küldi tovább.

Maga az adatátvitel soros, 11 bitből áll (START, 8 bit, paritás, STOP bit).



9.17. ábra. Adatátvitel PC billentyűzet és PC között.

A vezetékes kapcsolat mellett még használnak rádiós, Bluetooth és optoelektronikai átvitelt is. Az optoelektronika (infravörös tartomány) ma már nem használatos.

A PC fizikai csatlakozója régebben PS2 szabványú volt, ma már USB rendszerű.

9.18. PS2 és USB csatlakozópontok, valamint egy PS2 – USB átalakító.

Létezik úgynevezett virtuális klaviatúra, ezt kivetítik az asztalra, optikailag pedig le lehet olvasni a billentyűzet „leütését”.

9.19. ábra. Virtuális klaviatúra.

9.2.3. EGÉR

Az egér egy kézi mutatóeszköz, amely lehetővé teszi egy felületen a gyors pozícióváltást, ezt egy virtuális felületen ki is jelzi (például a képernyőn megjelenített tervezői felületen) valamilyen szimbólummal. Legegyszerűbb esetben nyíllal, de tervezőprogramoknál például lehet egy szimbolikus ceruza, vagy más is. Gyakorlatilag síkban működik, relatív elmozdulást érzékel, felemelve, máshol lehelyezve ugyanott folytatja a mozgást a kivetített területen, ahol abbahagyta az imént, vagyis a számítógép nem érzékeli az elmozdulást. Szükséges a számítógépnek jelzést adni arról, ha valahol valamilyen műveletet el szeretnénk kezdeni, ez egy kis kapcsolóval oldható meg. Kényelmi szempontok miatt három, vagy több ilyen kézre álló kapcsoló és sokszor egy görgő is található az egereken. Ezekkel az elemekkel kiegészítve egy egyszerűen, kellemesen használható eszközhöz jutunk.

Gyakorlatilag két típusuk létezik, az elektromechanikus és az optikai. A mechanikus egér egy keménygumi gömböt tartalmaz, ez a felületen görgetve x és y irányú mechanikai áttételen keresztül elmozdulás jeladót mozgat. A keletkezett forgómozgást egy kódtárcsa érzékeli. Hátránya az optikai egérhez képest az elpiszkolódás és a kisebb pontosság (felbontás).

Az optikai egér egy érzékelő (szenzor) segítségével sorozatos képeket készít az egér alatti sík területről. Az keletkezett képek között a mozgástól függően eltérés keletkezik, ezeket az eltéréseket egy áramkör elemzi ki és az eredményt a két tengelyhez (x és y) viszonyított elmozdulássá alakítja. Többféle felbontással készülnek:

9.4. táblázat. Optikai egerek felbontása [dpi].

kis felbontás	közepes felbontás	nagy felbontás	extra felbontás
20-30-50	100-200	250-350	400

A dpi a Dot Per Inch rövidítés, ami azt jelenti, hogy 25.4 mm-re hány képpont esik.

9.2.4. BOTKORMÁNY (JOYSTICK)

A botkormány rögzített talapzattal rendelkezik, amiből egy kényelmesen marokra fogható kar nyúlik ki. A kar egy kúp alakú mozgást engedélyez, amit szögértékké alakít át az elektronika. Különböző parancsokat is indíthatunk a karra szerelt nyomógombokkal. A PC-nél főleg a játékoknál van jelentősége, viszont fontos eszköze az iparnak, ahol segítségével nagy gépeket, berendezéseket tudunk pozícionálni, illetve parancsokkal ellátni.

Két típusa van:

- folytonos potenciométeres és
- érintkezős.

A folytonos potenciométeres megoldásnál egy csuklón két potenciométer van elhelyezve, úgy, hogy közöttük 90 ° -os eltérés van, így x – y koordinátákat tudnak visszaadni. Ezeket a jeleket a számítógépbe A/D (analóg-digitális) átalakítóval lehet bevinni a további feldolgozás érdekében.

Az érintkezős megoldásnál négy érintkező található 90 °- ra egymástól (kettő-kettő szemben). Ezeket az információkat egyszerűbb beolvasni, mint a potenciométeres megoldásnál alkalmazott analóg jelet.

Mindkét típusnál kiegészítő (például a kar tetején) nyomógombok vannak, indító-leállító folyamatok létrehozására.

9.2.5. LAPOLVASÓ (SZKENNER)

Képek, szövegek beolvasását teszi lehetővé. Létezik:

- kézi,
- lapáthúzós és
- asztali kivitel (síkágyas szkenner).

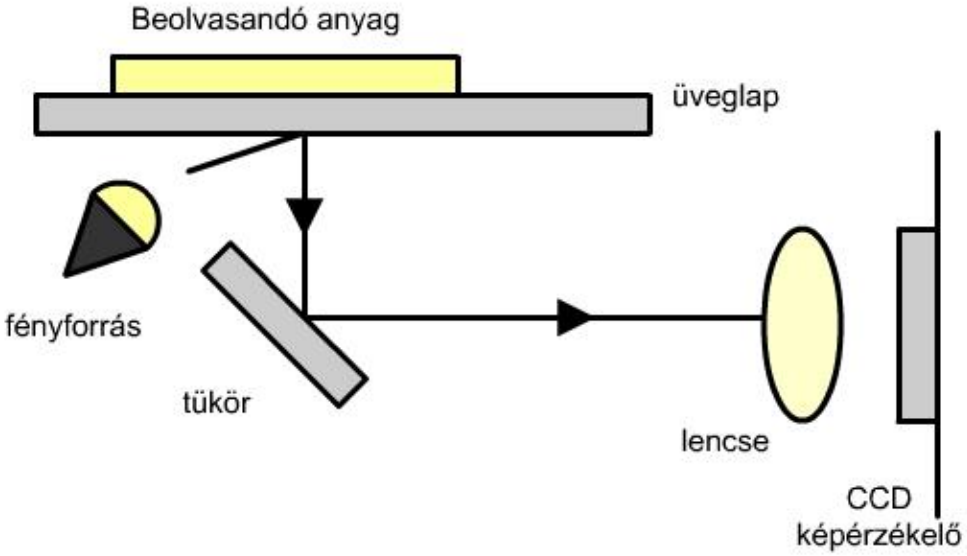
A kézi, hordozható lapolvasót a kívánt felületen (ahol a beolvasandó szöveg, vagy kép van) kell végighúzni, míg az asztali kivitelnél a fekvő készülék üveglapjára helyezett felületet egy mozgó fej tapogatja le. A lapáthúzós szkenner a lapot behúzza, ez nem minden esetben alkalmazható technika. A kézi szkenner egyik problémája, hogy nem lehet teljesen egyenletesen, egyforma sebességgel letapogatni az oldalt.

Műszakilag két működési elvet különböztetünk meg:

CCD (Charge-coupled Device - töltés-csatolt eszköz) és

CIS (Compact Image Sensor – kompakt képérzékelő).

A síkágyas, vagy asztali szkennernél egy léptetőmotor egy lámpát (fényforrást, fénycsövet) vízszintesen végigmozgat alulról a dokumentumon egy tükörrel együtt, amely tükör a CCD érzékelőbe vetíti a képet egy lencsén keresztül. A lencse kicsinyíti le a képet a CCD számára. Ezt a képet azután az elektronika dolgozza fel tovább digitális alakúvá.



9.18. ábra. Síkágyas (asztali szkenner).

A kompakt képérzékelő, vagyis a CIS elvileg más felépítésű, egyrészt LED-es megvilágítású, másrészt nincs tükör, így sokkal kisebb a mérete. Háromszínű (RGB – vörös, zöld és kék) LED megvilágítással dolgozik a rendszer.

Ha összehasonlítjuk a két rendszert, akkor láthatjuk, hogy a CIS ugyan méretében kisebb, viszont a fényerő miatt közelebb kell a fejnél lenni az anyaghoz, a LED fényspektruma pedig szűkebb, mint a fénycsőé, így gyengébb a színhűség.

9.3. KIMENETI PERIFÉRIÁK

A számítógép a külvilág felél igen különböző típusú és mennyiségű információt tud továbbítani. A kimeneten analóg és digitális jelek lehetnek, néhány bittől egészen nagy mennyiségű adatig. A következő alfejezetekben ezek közül a legjellegzetesebb egységekkel foglalkozunk, figyelembe véve a személyi számítógépnél szokásosan használt egységeket, de kitérünk néhány ipari megoldásra is. A felsorolás nem teljes.

9.3.1. NYOMTATÓK

A nyomtatók feladata, hogy digitális alakban tárolt információt (szöveg, számok, kép) nem elektronikusan jelenítsen meg, leggyakrabban papíron, valamilyen műanyag fólián, esetleg más felületen. A nyomtatás minőségét a DPI (Dot Per Inch) határozza meg, ami szó szerinti fordításban azt jelenti, hogy egy 25.4 mm (hüvelyk) hosszúságú vonalon hány pontot lehet elhelyezni. Ez egyrészt attól függ, milyen kicsi pontot tudunk nyomtatáskor létrehozni, másrészt milyen közel tudjuk a pontokat elhelyezni.

Fontos paraméternek számít a lap/perc teljesítmény, ami gyakorlatilag a nyomtatási sebességet jelenti.

A fejlődés során többféle technika alakult ki nyomtatók gyártására, ezek közül ma már többet nem használunk, vagy esetleg igen korlátozottan, speciális megoldásoknál.

Többféle besorolást használunk a nyomtatótípusok meghatározására:

- ütőnyomtató és
- nem ütőnyomtató.

Feloszthatjuk a nyomtatókat:

- színes és
- szürkeállományú nyomtatókra.

Feloszthatjuk őket a papíron egyszerre hagyott nyom szerint:

- karakternyomtatóra (írógép-technikából örökölt megoldás), lehet betűkerekes, vagy gömbfejes,
- sornyomtatóra, egy sor nyomtatása egyszerre történik meg és
- lapnyomtatóra, ahol egy lap kinyomtatása történik egyszerre.

A nyomtatók csatlakoztatása a számítógéphez történhet

- vezetékesen és
- vezeték nélkül.

A vezetékes megoldásnál elterjedt:

- a soros port (RS 232 C),
- párhuzamos port (CENTRONICS),
- USB és
- ETHERNET (hálózat).

Ma gyakorlatilag csak az USB csatlakozó létezik a nyomtatóknál, illetve gyakori a hálózatra kötött nyomtató, melyet UTP kábellel csatlakoztatunk. Már otthoni alkalmazásoknál is érdemes hálózati nyomtatót használni, ilyenkor csak a nyomtatót és az adott számítógépet kell bekapcsolni, míg USB használata esetén a nyomtatóval kapcsolatban levő számítógép és az is, amelyről nyomtatunk bekapcsolt állapotban kell hogy legyen.

- Vezeték nélküli megoldások között vannak:

- WIFI,
- Bluetooth és
- IRDA (infravörös) megoldások.

Nem ütőnyomtató a:

- hőnyomtató és
- a rajzgép, vagy plotter.

A hőnyomtató a drága papír miatt ma már nem használatos. A rajzgépnél egy számítógépből vezérelt motoros-mechanika valamilyen rajzolóeszközt, tollat mozgat, de más nyomtatók használata háttérbe szorítja ezeket a berendezéseket.

Legelterjedtebb két nyomtatótípus a:

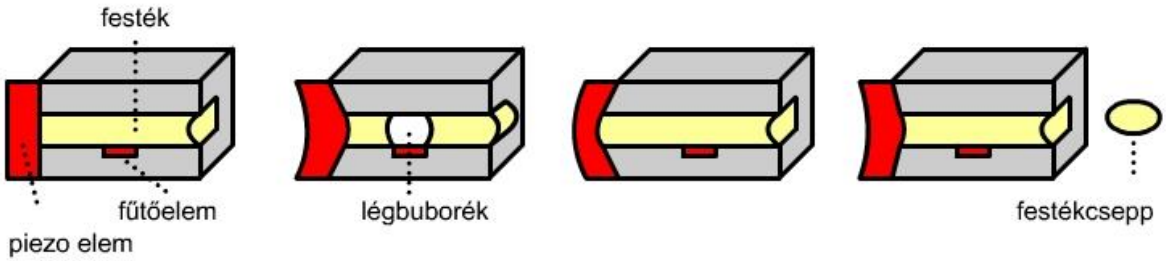
- tintasugaras- és
- a lézernyomtató.

Ezen kívül ma már igen jó minőségű 3D nyomtatók jelentek meg, amelyek segítségével valós tárgyakat lehet létrehozni.

Leggyakrabban négy színes patronból juttatnak apró festékcseppeket a papírra. Két tecnológia terjedt el:

termáltechnológia és

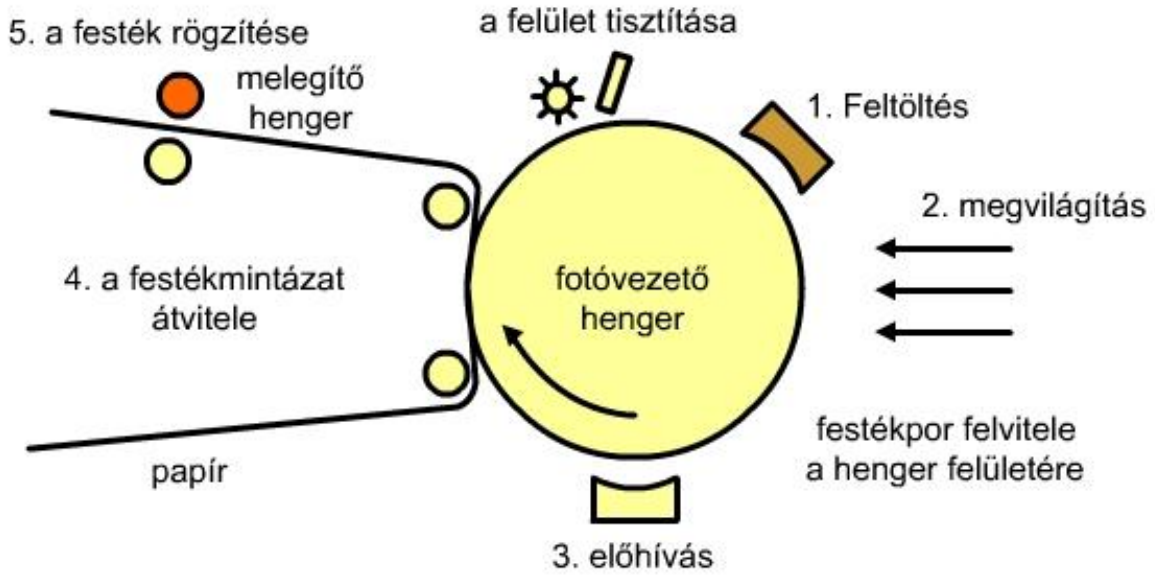
piezotechnológia.



9.19. ábra. Termál- és piezotechnológia alkalmazása a tintasugaras nyomtatóknál

A termál technológiánál a festékcsatorna (1) alatt egy fűtőtest (2) gázbuborékot (3) hoz létre, ami kilöki a papírra a festékcseppet.

A piezotechnolóiánál elektromos jellel a piezokristályt (5) kidomborítják, ezzel festéket szív be a csatornába, majd ugyanezt a piezokristályt homorítják ellentétes polarizálással, ami kilöki a festékcseppet.

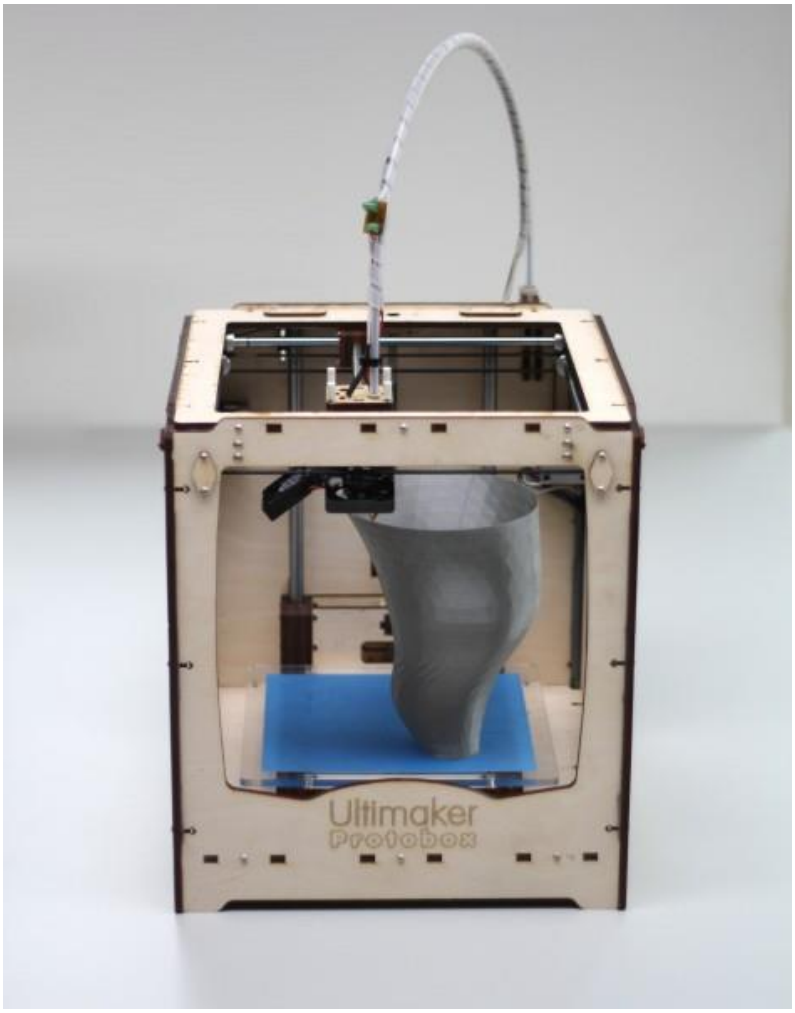


9.20. ábra. Lézernyomtató működése

A lézernyomtató egy hengert tartalmaz, amelyet bevonnak fényérzékeny anyaggal, majd a hengert elektrosztatikusan feltöltik. Lézerfénnyel a sötét részeket jelölik a hengeren, ezeken a helyeken vagy semleges, vagy ellentétes töltés alakul ki. Ezután a festék megtapad azokon a helyeken, ahol a lézersugár megváltoztatta a töltést, ezt a festéket papírral érintkezve egy körülbelül 200 C° -os melegítéssel a papírra égetik.

9.3.2. 3D NYOMTATÓK

Iparban már régóta létező technológia ára erősen zuhan, ma már elérhető áron hozzá lehet jutni ezekhez a nyomtatókhoz. Többféle technológia létezik, a kiindulás vagy egy CAD programmal megtervezett tárgy, vagy egy valós tárgy 3D-ben történő beszkennelése. A technológia hasonlít a tintasugaras technológiához, azzal a különbséggel, hogy az elkészítendő tárgy vékony szeletekben készül, pl megolvasztott kis cseppek x – y felületre juttatva z (magassági koordináta állandó növelésével) irányban rétegenként. A réteg vastagsága adja meg a felbontást, illetve a pontosságot. Jelentősége abban van, hogy kis sorozatban próba képpen valós tárgyak készíthetők, de elméletileg már lehetséges email-ben elküldeni bárhová az adatokat és másutt is létre lehet hozni az eredetihez hasonló tárgyakat.



9.21. ábra. 3D nyomtató.

9.3.3. VIZUÁLIS KIMENET

Az ember látószerve, a szem viszonylag könnyen jó minőségű információt szerez a külvilágból. Természetesen a számítógépek is rendelkeznek olyan kimenetekkel, amely kimenetek a szem számára küldenek információt, sokszor nagyon egyszerűt, egy két lámpa, fix felirat felgyújtásával, eloltásával. Bonyolultabb vizuális információ megjelenítése egyszerűbb esetben lehetséges síkban monitorral, illetve projektorral, de térhatású effektek is elérhető különféle technológiákkal. Közvetlenül a számítógép nem alkalmas az előbb említett információmegjelenítésre, ehhez közbe kell iktatni a grafikus kártyát.

9.3.3.1. GRAFIKUS (VIDEO) KÁRTYÁK

Ez a kártyatípus is, a számítógép egyéb perifériáihoz hasonlóan, óriási fejlődésen ment át. Nincs értelme a fejlődés összes lépését részletesen leírni, csak annyit, hogy az első PC típusú számítógépek fekete-fehér (monokróm) monitorokat tudtak meghajtani, két üzemmódban:

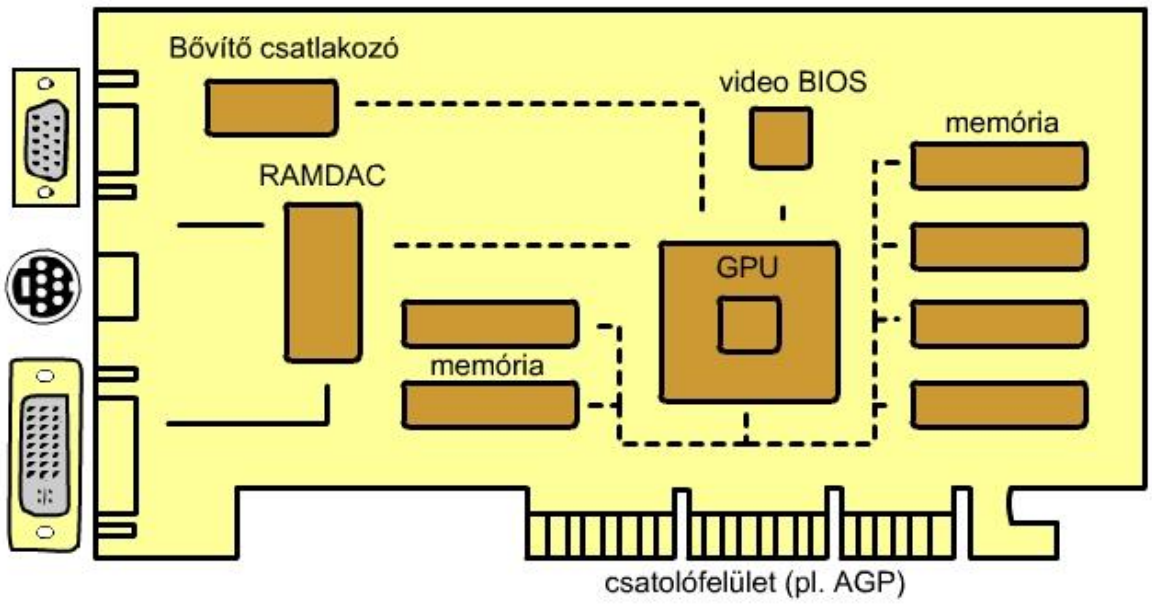
- szöveges - 80 oszlop x 25 sor, ahol egy karaktert 9 x 14 képpont alkotott és

- grafikus - (az első megoldásnál nem is volt), 640 x 200 képpont később.

Az alaplapokon ma már egy egyszerűbb megoldású videokártya is megtalálható, így nem szükséges külön vásárolni, de természetesen ez kikapcsolható és mint periféria egy nagyobb teljesítményű videokártya implementálható a rendszerbe.

A mai, modern, nagyteljesítményű grafikus kártyák már 3 D effektusok létrehozására is képesek. Természetesen ez bonyolítja a helyzetet a tiszta kétdimenziós megoldásokhoz képest, hiszen itt mélységi adatokat is meg kell jeleníteni (illetve tárolni is). Több gyártó két részre bontja a kártya feladatait, az egyik feladat az úgynevezett elfedett felületek és az árnyékok létrehozása, a processzor másik feladata pedig a felületekhez tartozó mintázatok létrehozása, azok színezése. Nem ritka a több busz alkalmazása sem, de döntő a memóriakapacitás nagysága (ez természetesen a kártyán kialakított memória), sebessége. A mérnöki tervezőprogramok igénylik ezeket a kártyákat, de természetesen a játékprogramok egyre látványosabb megoldásai is hozzájárulnak a fejlődéshez.

A grafikus kártyákon gyakorlatilag egy célprocesszor található, amelyiknek a számítási teljesítménye messze meghaladja egy CPU teljesítményét, de természetesen a CPU-hoz képest lényegesen szűkebb utasításkészlettel rendelkezik. Ennek a központi egységnek GPU (Graphics Processing Unit) a neve. A feladatot a CPU-tól kapja.

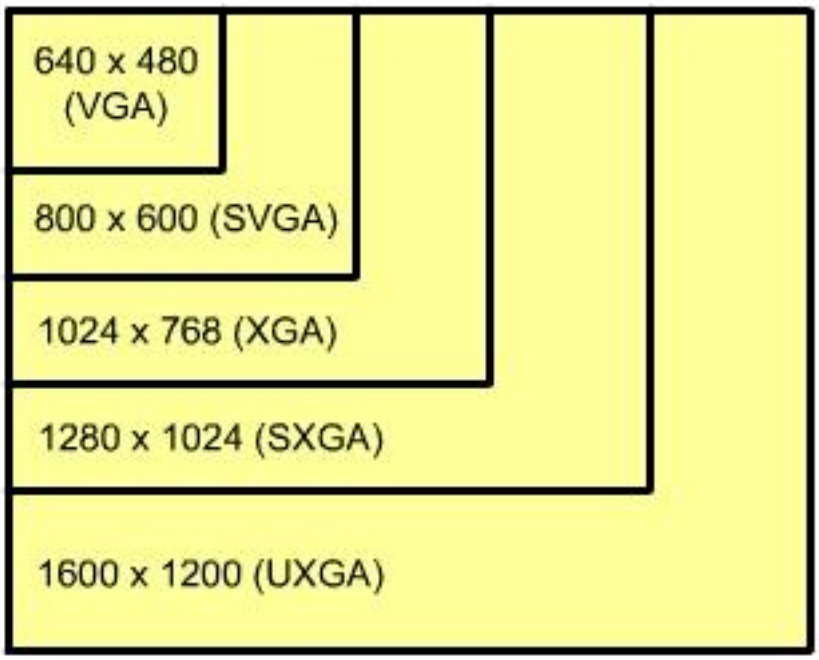


9.22. ábra. Egy videokártya felépítése.

A videokártya fő egységei:

- GPU (Graphics Processing Unit) - Grafikus feldolgozó egység - bonyolult számításokkal állítja elő a 2D, vagy 3D képet.
- RAMDAC (RAM - Digital Analog Converter) RAM - digitális-analóg átalakító - a számítógép digitális bináris jeleit analóg jellé konvertálja.
- Video BIOS (Basic Input Output System) - biztosítja a vezérlőáramkörök sajátos képességeinek kezelését és illesztését a rendszerhez.
- A video memória - a monitoron megjelenítendő képek tárolása.

A videokártyák különböző képarányokat és felbontásokat támogatnak. A következő ábrán egy 4:3 arányú képernyő szabványos képfelbontásait láthatjuk.



9.23. ábra. 4:3 arányú képernyő néhány szabványos felbontása

A 16:9 széles képfelbontás (Wide Screen) két szabványt támogat:

- WXGA (1280x800) és
- WUXGA (1920x1080) - a HD (High Definition) televíziózás felbontása.

Létezik még egy úgynevezett nagyon széles képernyőfelbontás is, ahol:

- QWXGA (2048x1152) és
- WQUXGA (3840x2400) található.

9.4. HÁLÓZATI PERIFÉRIÁK

A személyi számítógépek ugyanúgy, mint a hordozható eszközök, a mobil telefonok, vagy a tabletek gyakorlatilag állandóan egy közös információs térben vannak, vagyis egymással, különböző kapcsoló eszközökkel összeköttetésben vannak. Ez ma már megtalálható az ipari alkalmazásoknál is.
A kapcsolat a berendezések között lehet:

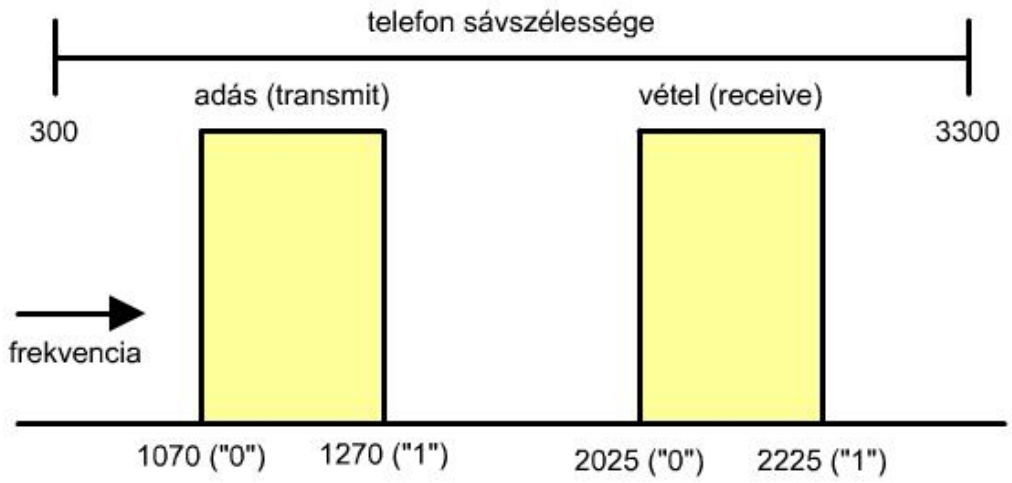
- vezetékes és
- vezeték nélküli.

Leggyakrabban a következő megoldásokkal találkozunk:

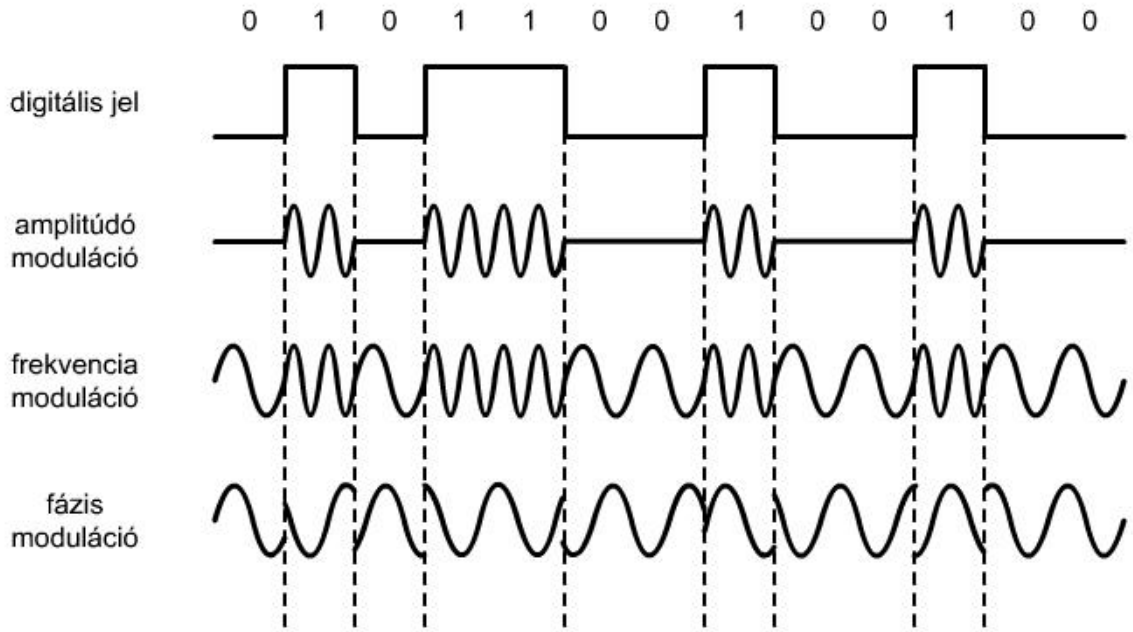
- MODEM,
- hálózati vezetékes eszközök (Ethernet) és
- WLAN.

MODEM (modulátor – demodulátor)

Meglevő analóg telefonvonalat használnak pont-pont adatátvitelre. A telefonvonal sávszélessége 0.3 kHz és 3 kHz között van, így bináris digitális jel közvetlenül nem vihető át. Helyette 1 kHz és 2 kHz frekvenciájú szinusz jelek viszik a 0 és 1 jelet.



9.24. ábra. Full duplex (kétirányú adatátvitel).



9.25. ábra. A modemek néhány modulációs technológiája.

9.4.1. HÁLÓZATI KAPCSOLAT

Biztosítja a kapcsolatot a hálózatokhoz, ma már az alaplap eleve tartalmazza a kártyát, de létezik külön kártyán levő változata is. Régebben koaxiális kábelen keresztül csatlakozott, ma sodrott érpár használatos (UTP csatlakozóval).

9.26. ábra. Hálózati kártya

9.4.2. WLAN (WIRELES LAN, WiFi – VEZETÉK NÉLKÜLI KAPCSOLAT)

Rádiós megoldás, helyi kapcsolat létrehozására. A WiFi az IEEE 802.11 szabvány szerint működik, 2.4 GHz frekvencián 11 Mbit/s és 54 Mbit/s sávszélességgel.

01. MELLÉKLET

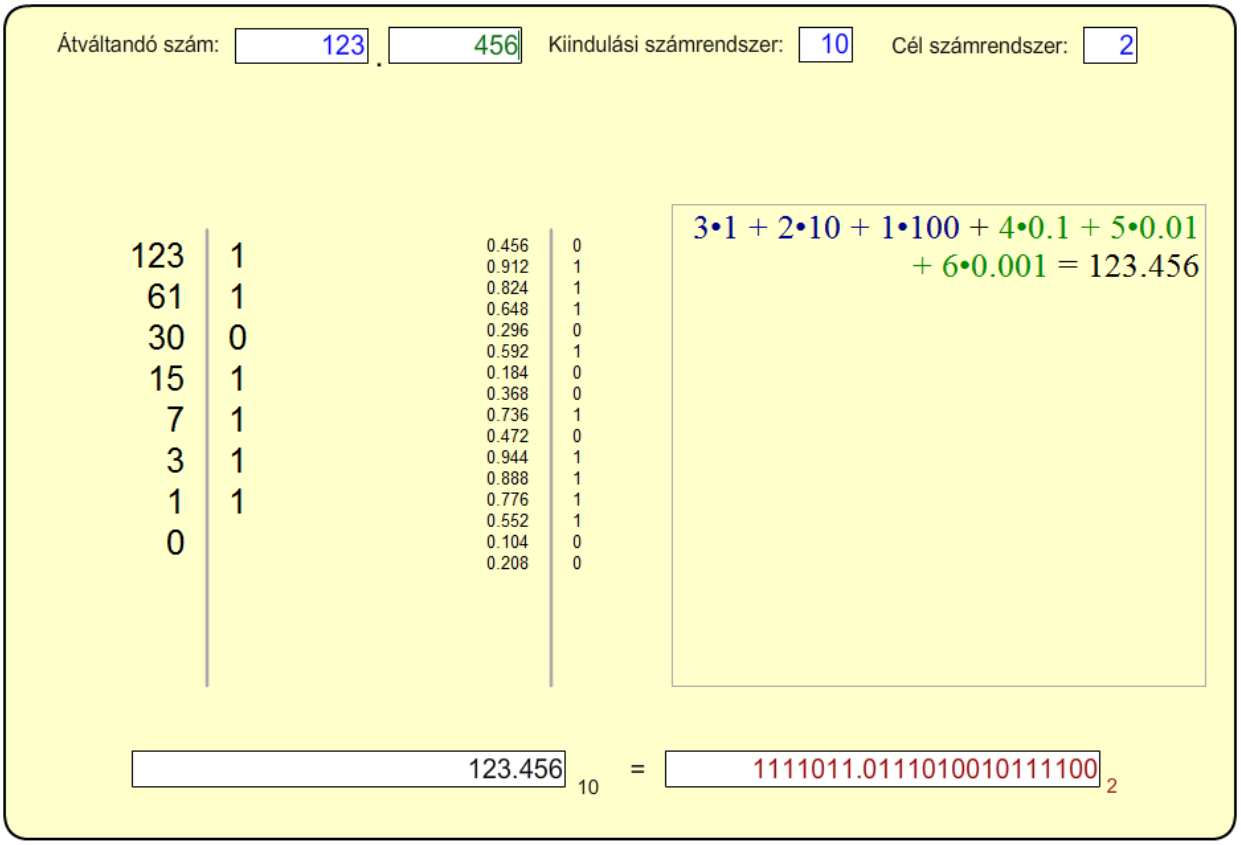
A SZÁMÍTÓGÉPEK MATEMATIKAI ALAPJAI

A csatolt fájlrendszerben a „szamrendszerek.html”-re kattintva megjelenik egy weboldal a számítógépen beállított keresőben:

Induláskor a kiindulási számrendszer alapja 10 (tízes számrendszer), a célszámrendszer alapja pedig 2 (bináris számrendszer). Ez természetesen bármikor tetszőlegesen megváltoztatható.

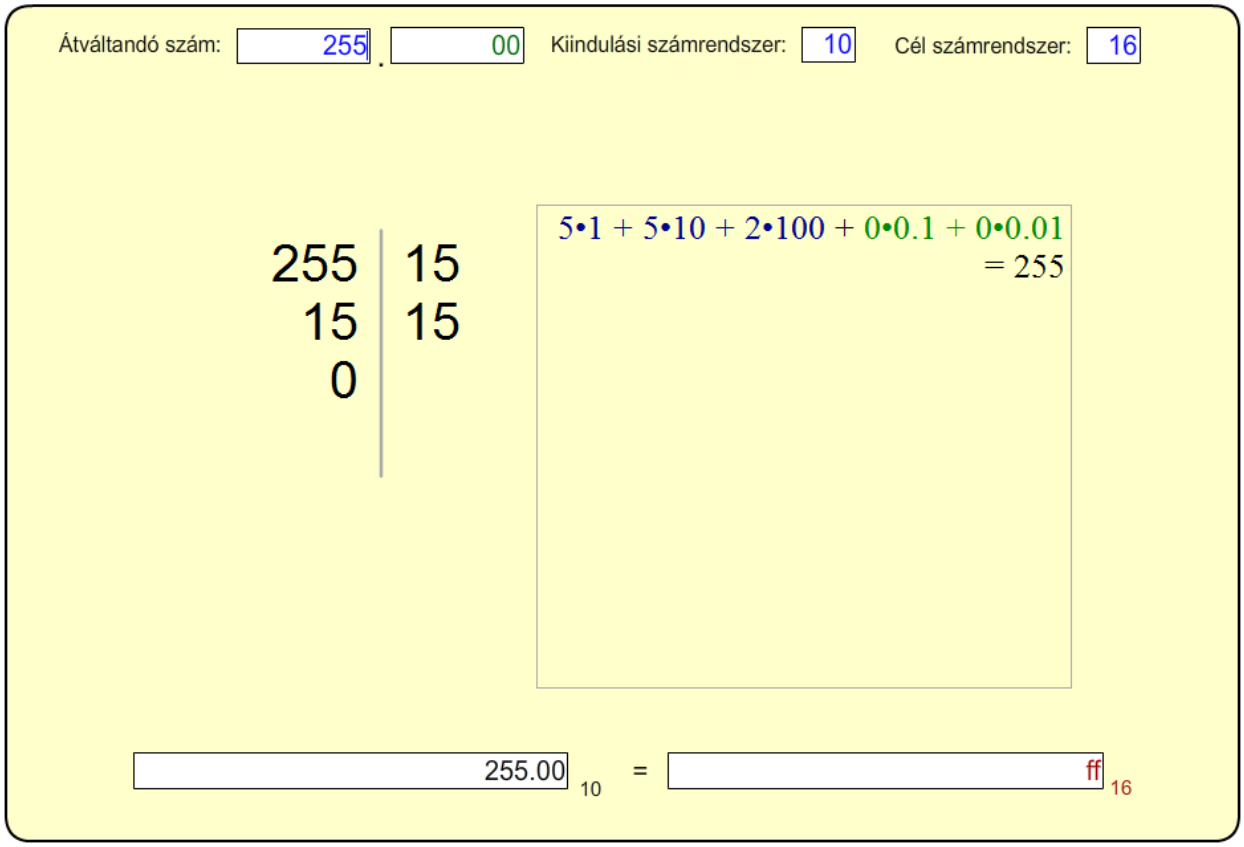
Példák:

Alakítsuk át a 123.456 tízes számrendszerbeli számot binárissá, az eredmény a következő ábrán látható:



Alakítsuk át a tízes számrendszerben megadott 255-öt 16-os (hexadecimális számrendszerbe:

Átállítjuk a célszámrendszert 16-ra, majd beírjuk a 255-öt a kiindulási számrendszer ablakba, eredmény az ábra szerint.

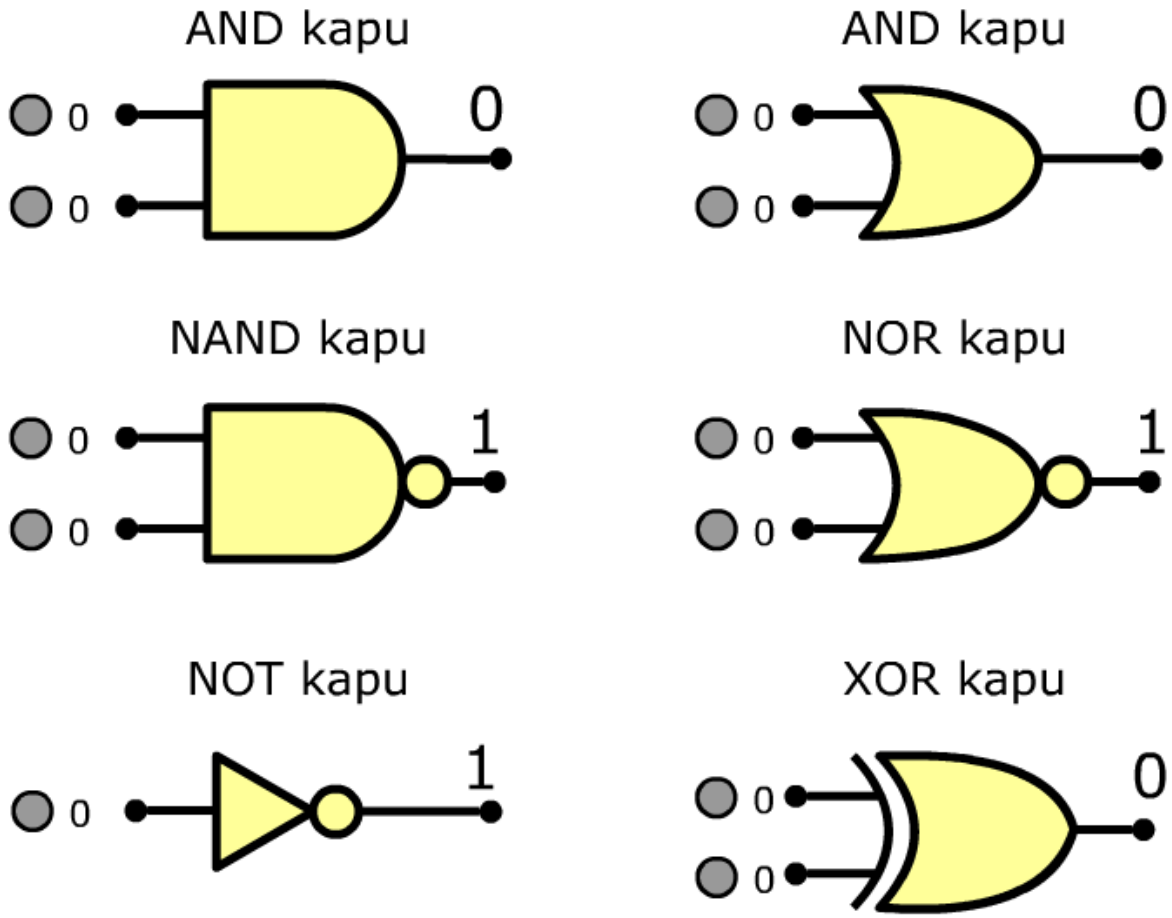


02. MELLÉKLET

A SZÁMÍTÓGÉPEK LOGIKAI ALAPJAI

02.1. ELEMI LOGIKAI KAPUK

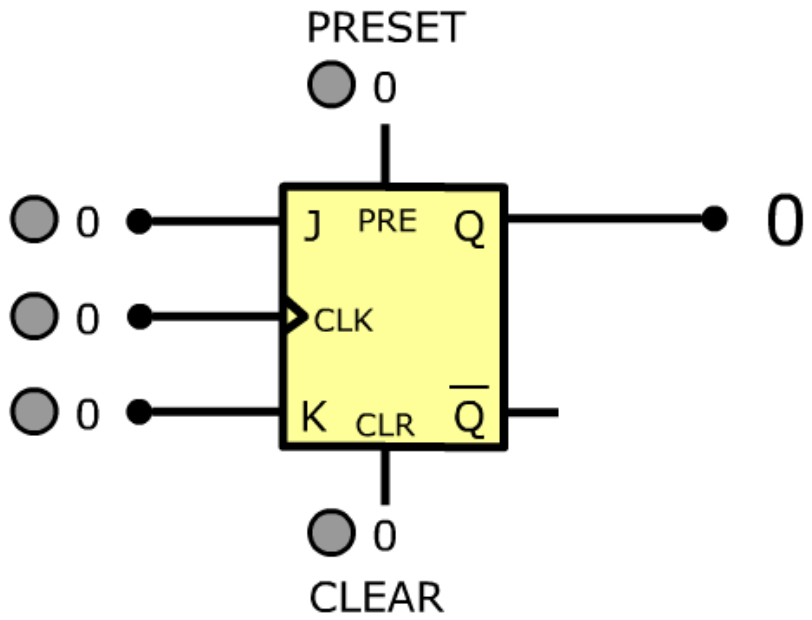
A kapuk2.html fájlra kattintva, vagy a kapuk2.swf fájlt egy keresőbe húzva indíthatjuk el a következő ábrán levő animációt. A bemenetek előtt található körre kattintva logikai 0 vagy logikai 1 állítható be, aminek a hatására a kimeneten megjelenik az adott kapu logikai értéke.



02.1. ábra. Néhány logikai kapu.

02.2. JK FLIP_FLOP MŰKÖDÉS

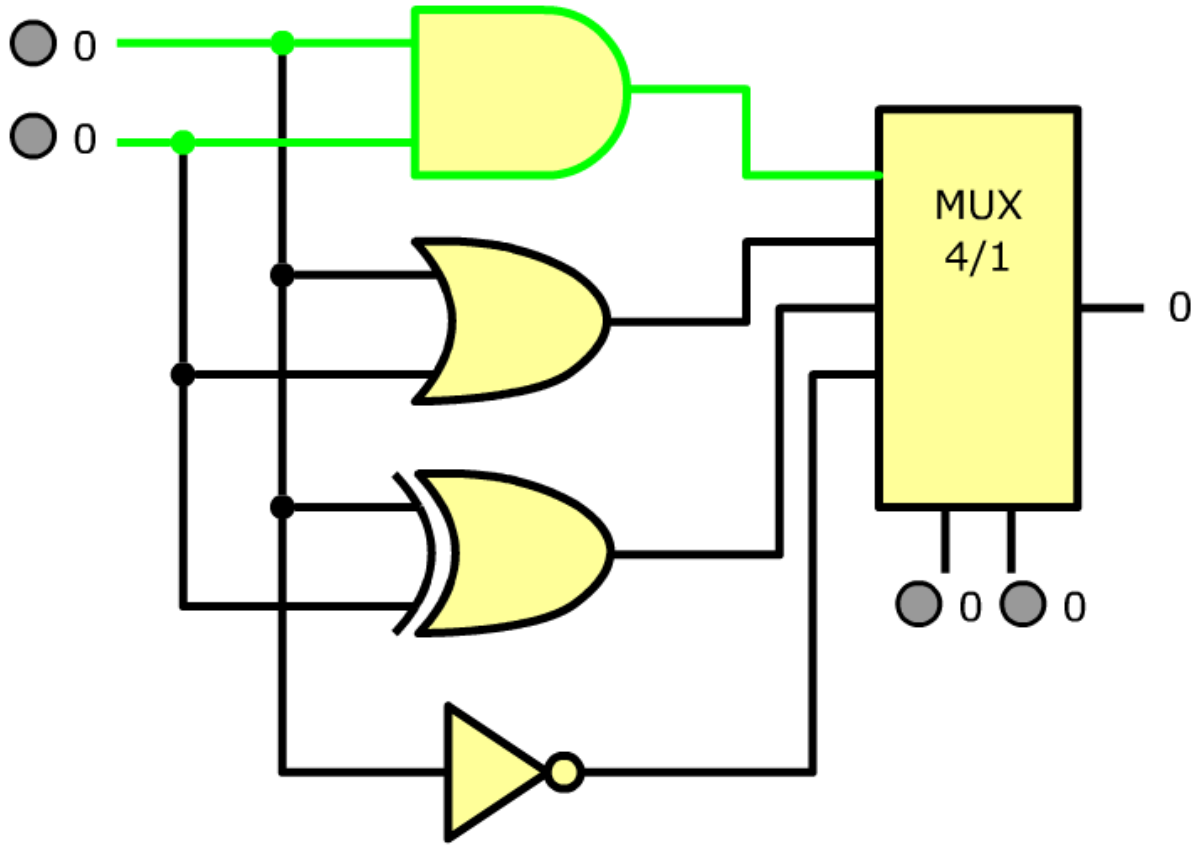
A jkflipflop.html fájlra kattintva, vagy a jkflipflop.swf fájlt egy keresőbe húzva indíthatjuk el a következő ábrán levő animációt. A bemenetek előtt található körre kattintva logikai 0 vagy logikai 1 állítható be, aminek a hatására a kimeneten megjelenik a kimeneti érték. A CLK (clock) jel lefutó éle (1 → 0 átmenet) váltja ki a változást.



02.2. ábra. A JK tároló.

02.3. ALU (ARITMETIKAI-LOGIKAI EGYSÉG) VIZSGÁLATA

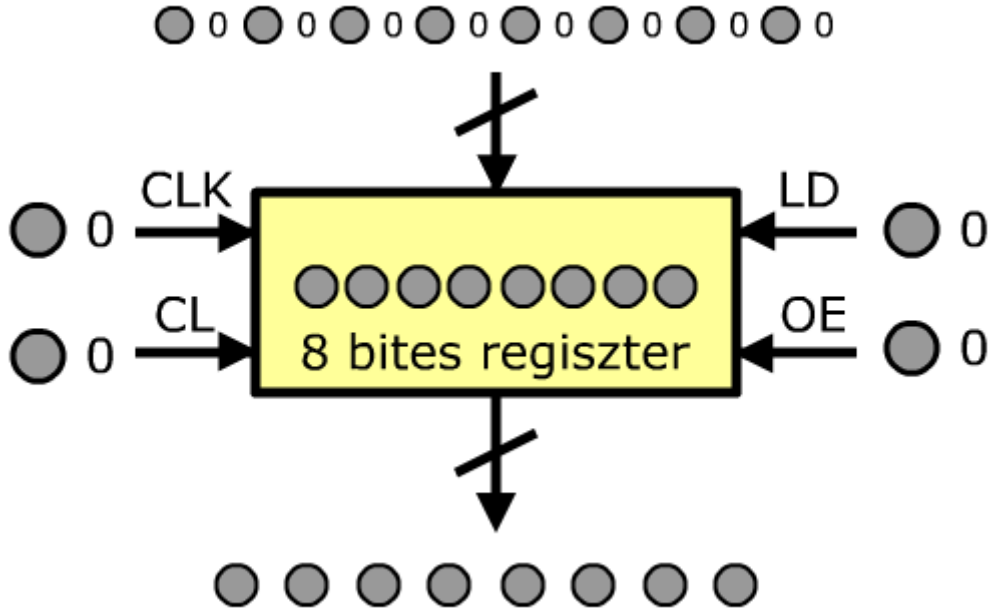
Az alu.html fájlra kattintva, vagy az alu.swf fájlt egy keresőbe húzva indíthatjuk el a következő ábrán levő animációt. A bemenetek előtt található körre kattintva logikai 0 vagy logikai 1 állítható be, aminek a hatására a kimeneten megjelenik a kimeneti érték.



02.3. ábra. ALU (Aritmetikai-logikai egység) vizsgálata.

02.4. REGISZTER VIZSGÁLATA

A regiszter.html fájlra kattintva, vagy a regiszter.swf fájlt egy keresőbe húzva indíthatjuk el a következő ábrán levő animációt. A bemenetek előtt található körre kattintva logikai 0 vagy logikai 1 állítható be, aminek a hatására a kimeneten megjelenik a kimeneti érték. A CLK (clock) jel lefutó éle (1 → 0 átmenet) váltja ki a változást.



0.3. AZ ASCII (AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE) TÁBLÁZAT

02.1. táblázat: az ASCII kódtáblázat.

	000	001	010	011	100	101	110	111
0000	NUL	DLE	blank	0	@	P	’	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EDT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}

1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

A táblázat tartalmazza a betűk, számok és írásjelek 7 bites kódjait, valamint parancskódokat is (például kocsi vissza – CR, vagy soremelés – LF).

Ha szeretnénk meghatározni egy karakter kódját, ki kell keresnünk a karakternek megfelelő bit-kombinációt az oszlopokból és sorokból.

Példa: „A” betű kódja az 100 oszlophoz és 0001 sorhoz tartozik, így az érték:

0 100 0001 bitekből áll össze, vagyis 01000001b = 41h = 65d lesz.

A 7-es szám kódja a 011 oszlophoz és 0111 sorhoz tartozik, így az érték:

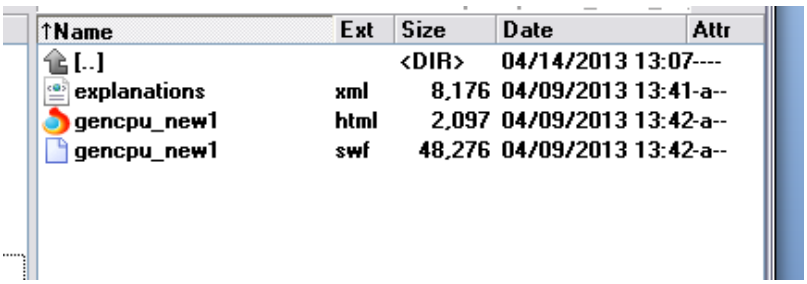
0 011 0111 bitekből áll össze, vagyis 00110111b = 37h = 55d lesz.

04. MELLÉKLET

SZÁMÍTÓGÉP-MODELL

Az interaktív számítógép indítása a következőképpen történik:

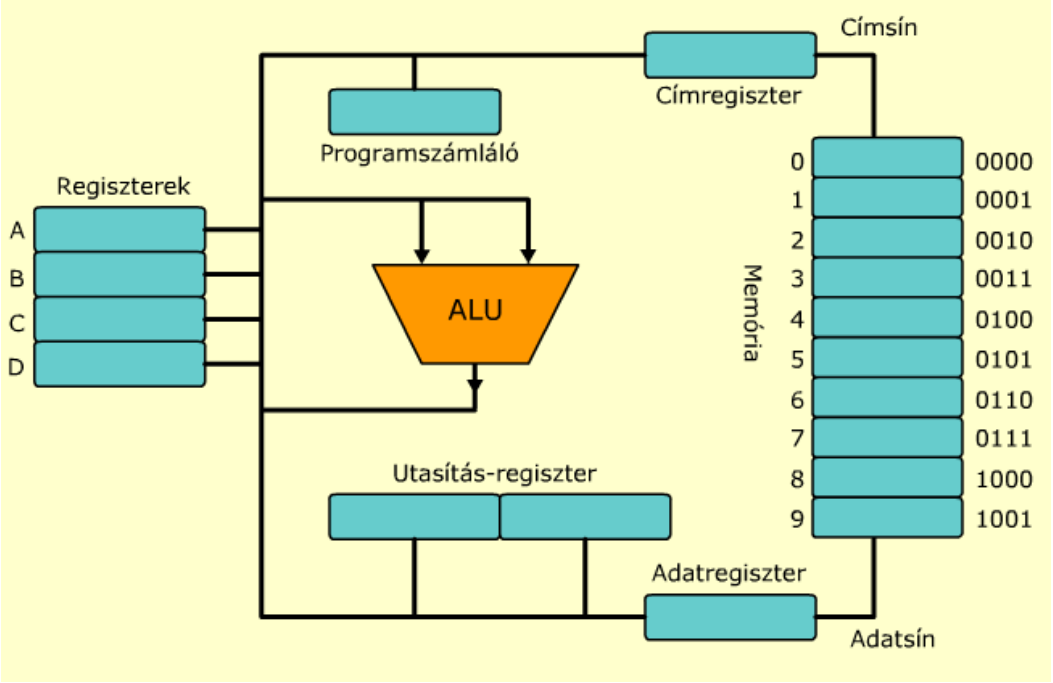
Az alkönyvtár 3 fájlt tartalmaz:



04.1. ábra. Az alkönyvtár tartalma

a) A gencpu_new1.html fájlra kattintással az alapértelmezett keresőben elindul a szimulációs-interaktív egyszerű számítógép modell,
b) vagy egy kinyitott keresőbe egyszerűen áthózzuk a gencpu_new1.swf fájlt.

A modell működését a modell mellett látható ablakban követhetjük, itt jelennek meg a magyarázatok.



04.2. ábra. A számítógép-modell

0.5. MELLÉKLET

INTEL 8086 ÖSSZEFOGLALÓ

i8086 16 bites regiszterei	
- általános célú regiszterek ax (al,ah): akkumulátor bx (bl,bh): bázis regiszter cx (cl,ch): számláló regiszter dx (dl,dh): adat regiszter	- index regiszterek si : forrás index regiszter di : cél index regiszter
- stack kezelés regiszterei sp : stack pointer bp : bázis (frame) pointer	- szegmens regiszterek ds :adat szegmens regiszter cs :program szegmens regiszter ss :stack szegmens regiszter es :extra adat regiszter
- speciális regiszterek	

ip : utasitás számláló status: státusz és vezérlő regiszter	

Cimzési módok

Az i86 szegmentált memóriaképet valósit meg a moduláris programozás támogatására.

fizikai cim = szegmens * 16 + offset

Hosszú (FAR) cimzésnél a szegmens cimet explicite meg kell adni. Rövid (SHORT) cimzésnél a szegmens cim egy szegmens regiszter tartalma. A vonatkozó szegmens regisztert meg kell adni, vagy az értelemszerű alapértelmezést használhatjuk:

vonatkozó szegmensregiszterek	
adatműveletek	ds
stack és BP relativ műveletek	ss
ugrások, szubrutin hivások	cs
blokk cél operandusok	es

Pl.:

```
sub   cx, kivon       ;ds
and   cx, es:valt     ;es
stosb                 ;es
push  ax              ;ss
mov   dx, [bp-8]      ;ss
inc   cs:valt1        ;cs
call  rutin           ;cs
```

Az offset kialakitásához használható cimzési módok és példák	
-memória direkt:	mov BYTE PTR cim, 1
-regiszter direkt:	xor ax, ax
-regiszter indirekt:	cmp [bx], 2 (csak si,di,bx és bp)
-bázis relativ:	inc [bp-6] (csak si,di,bx és bp)
-indexelt:	and tomb[si](csak si,di,bx és bp)
-bázisrelativ indexelt:	dec [bx+si+3](csak si,di,bx és bp)

Utasitás készlet (real mód)

Az utasítás alakja:
cimke mnemonik cél-, forrásoperandus ; művelet

pl: add ax, bx ----------> ax = ax + bx

ahol forrásoperandus lehet konstans, pl. sub dx, 1

Adatmozgató utasítások	
MOV op1, op2	op1 = op2
XCHG op1, op2	op1 és op2 felcserélése
LEA reg, cim	reg = cim;
LDS reg, cim	reg = *(cim) ; ds = *(cim+2)
LES reg, cim	reg = *(cim); es = *(cim+2)

LAHF	ah = flag-ek
SAHF	flag-ek = ah
XLAT	al = *(bx + al)

Stack kezelő utasitások	
PUSH op	SP -= 2; stack[SP] = op;
POP op	op = stack[SP]; SP += 2;
PUSHA, POPA	általános, index és stack kezelő regiszterek mentése és visszaállitása
PUSHF, POPF	flag-ek mentése visszaállitása

I/O műveletek:	
IN al, portcim	portcim-ről beolvasás
IN al, dx	*(dx)-ről beolvasása
OUT portcim, al	portcim-re kiirás
OUT dx, al	*(dx)-re kiirás

Aritmetikai utasitások: carry, auxiliary-carry, parity, zero, sign, overflow bitek állítása	
ADD op1, op2	op1 = op1 + op2
ADC op1, op2	op1 = op1 + op2 + carry
SUB op1, op2	op1 = op1 - op2;
SBB op1, op2	op1 = op1 - op2 - carry;
INC op	op+1;
DEC op	op-1;
NEG op	op = 2-s komplemens(op);
NEG op	op = 1-s komplemens(op);
CMP op1,op2	flag-ek = (op1 - op2) tulajdonságai
CBW	ax = előjelkiterjesztés(al)
CWD	dx:ax = előjelkiterjesztés(ax)
Előjel nélküli számokra:	
MUL byteop	ax = al * byteop
MUL wordop	(dx,ax) = ax * wordop
DIV byteop	al = ax / byteop; ah = ax % byteop;
DIV wordop	ax = (dx,ax)/byteop; dx = (dx,ax)%byteop;
Előjeles számokra:	
IMUL byteop	ax = al * byteop
IMUL wordop	(dx,ax) = ax * wordop
IDIV byteop	al = ax / byteop; ah = ax % byteop;
IDIV wordop	ax = (dx,ax)/byteop; dx = (dx,ax)%byteop;

Logikai műveletek zero, carry, overflow flagek állítása	
NOT op	op = ~op;
AND op1, op2	op1 = op1 & op2;
OR op1, op2	op1 = op1 | op2;

XOR op1, op2	op1 = op1 ^ op2;
TEST op1, op2	flag-ek mint AND op1, op2
SHL op1,op2	op1 <<= op2; (előjel nélkül)
SHR op1,op2	op1 >>= op2; (előjel nélkül)
SAL op1,op2	op1 <<= op2; (előjel marad)
SAR op1,op2	op1 >>= op2; (előjel marad)

Ciklikus léptetések: ROL, RCL, ROR, RCR

BCD aritmetika: AAA, AAD, AAM, AAS, DAA, DAS

Blokk (string) kezelő utasitások:	
MOVSB	(es:di)byte = (ds:si)byte; si+= dsi; di+= ddi;
MOVSW	(es:di)word = (ds:si)word; si+= dsi; di+= ddi;
CMPSB	CMP (es:di)byte, (ds:si)byte; si+= dsi; di+= ddi;
CMPSW	CMP (es:di)word, (ds:si)word; si+= dsi; di+= ddi;
SCASB	CMP al, (es:di)byte; si+= dsi; di+= ddi;
SCASW	CMP ax, (es:di)word; si+= dsi; di+= ddi;
LODSB	MOV al, (es:di)byte; si+= dsi; di+= ddi;
LODSW	MOV ax, (es:di)word; si+= dsi; di+= ddi;
STOSB	MOV (es:di)byte, al; si+= dsi; di+= ddi;
STOSW	MOV (es:di)word, ax; si+= dsi; di+= ddi;
INSB	IN (es:di)byte, dx; di+= ddi;
INSW	IN (es:di)word, dx; di+= ddi;
OUTSB	OUT dx, (ds:si)byte, dx; si+= dsi;
OUTSW	OUT dx, (ds:si)word, dx; si+= dsi;

Vezérlés átadó utasitások:	
JMP cim	feltétel nélküli ugrás
JZ, JNZ	Logikai művelet után
JE, JNE, JL, JG, JLE, JGE	Előjeles aritmetikai művelet után
JE , JNE, JB, JA, JBE, JAE	Előjel nélküli műveletek után
JCXZ, JNO, JS, JNS, JP, JPO	Speciális

Feltételes ugrást csak 128 byte-tal nem messzebbi cimkére lehet kiadni.

Szubrutinhivás:	
CALL cimke	cimkén kezdődő rutin meghivása
CALL op	indirekt szubrutinhivás
RET	szubrutinból visszatérés
INT vector	software interrupt
IRET	visszatérés software és hardware interrupt rutinból

Processzorvezérlő utasítások	
CLC-törlés, STC-beállitás, CMC-komplementálás	Carry flag:

CLD-törlés, STD-beállitás	Direction flag:
CLI-tiltás, STI-engedélyezés	Interrupt Engedélyezés:
HLT	Processzor leállitás:
NOP	üres operáció: