8. Mikroszámítógépek

8.1. Bevezetés

Az emberi tevékenység egyik fő célkitűzése a bennünket körülvevő világ megismerése. A természeti törvények és törvényszerűségek felfedezésének és tanulmányozásának gyökerei az ókori görögökig nyúlnak vissza. Ők kezdték el a mai értelemben vett tudomány alapjait lerakni. Különösen Püthagoraszról és a róla elnevezett pithagoreusok tevékenységéről kell megemlékeznünk. Az egész pithagorászi szemlélet a következő állításban foglalható össze: a dolgok természete, lényege a szám. A valóság és a matematika kapcsolatába vetett hitet konkrét eredményekkel támasztották alá. Püthagorasznak és iskolájának köszönhetjük a Pitagorasz-tételt, az irracionális számok létezésének bizonyítását, a lebegő, gömb alakú Föld képét, a zenei hangok és a húrok hossza közötti összefüggést, hogy csak a legfontosabbakat említsük.

Szintén a görögök nagy érdeme az a felismerés, hogy a törvények felírásához absztrakció, idealizáció kell. Úgy tűnik, hogy az absztrakció nehéz, de nagyon fontos lépés volt a tudományok fejlődésében. Az absztrakció a jelenség olyan leegyszerűsítése, amely a jelenség alapvető jellegét nem változtatja meg, ugyanakkor kvantitatív tárgyalásra alkalmas.

A matematikai modellalkotás szerepét elsőnek Galileo Galilei (1564–1642) ismerte fel. Az 1623-ban megjelent „Il Saggiatore" című munkájában a következőket írja: „*A filozófia abban a nagy könyvben van írva, amely nyitva áll mindenkor szemeink előtt: az univerzumra gondolok; de nem olvashatjuk mindaddig, míg meg nem tanultuk a nyelvét, és nem barátkoztunk meg a jelekkel, amelyekkel írva van. A matematika nyelvén van írva, és a betűi háromszögek, körök és más geometriai alakzatok, amelyek ismerete nélkül lehetetlen egyetlen szót is megérteni.*" A matematikai modellen végzett számításokról később James Clerk Maxwell (1831–1879) így ír: „*Minthogy minden matematikai tudomány a fizika törvényei és a számok törvényei közötti relációkon alapul, így az egzakt tudomány célja az, hogy a természetben fellépő problémákat mennyiségek meghatározására redukálja – számokon végzett műveletek segítségével.*"

A számítási eljárások gyakorlati megvalósításának megkönnyítésére az emberiség az idők folyamán, a civilizációs foknak megfelelően, különböző bonyolultságú segédeszközöket szerkesztett. Az első ilyen segédeszköz az úgynevezett abakusz volt, amelyet az ókori görögök és főként a rómaiak használtak. Bizonyos számolóköveket a helyi érték szerint csúszóvájatba illesztettek, és ide-oda tologatásukkal végezték el az egyszerű műveleteket.

A XVI. és különösen a XVII. században beindult órásmesterség erőteljes fejlődése adta az első döntő impulzust a számolási segédeszközök célszerűsítéséhez és tökéletesítéséhez. Az első, mai értelemben vett számológép születése az ismert francia tudós, Blaise Pascal (1623-1662) nevéhez fűződik. Pascal gépe, amelyet tizenkilenc évesen, 1642-ben néhány tehetséges órással együttműködve épített meg, egy folyamatos tízes áttételű mechanikus összeadó- és kivonógép volt. Később, 1671-ben G. W. Leibniz (1646-1716) egy olyan számológépet szerkesztett, amely szorozni is tudott.

Továbbá megemlítjük Ch. P. Babbage (1792-1871) angol matematikus munkásságát; ő 1822-ben készített egy olyan gépet, amellyel táblázatokat lehetett kiszámítani. Később, 1825-ben, egy ún. „analitikus gép" tervét vetette fel, amely már sok, napjaink elektronikus számítógépjeire jellemző megoldást tartalmazott.

A gépnek 1000 szavas tára lett volna, egyenként 50 decimális számjegy befogadóképességgel. Adatbevitelre a J. M. Jacquard által feltalált és szövőgépek vezérlésére használt lyukkártyákat alkalmazta volna. A terv élete végéig foglalkoztatta; de a tervezést befejezni nem tudta, tervének gyakorlati kivitelezés is korlátokba ütközött, de szellemileg egy századdal megelőzte korát.

A modern számítógépig vezető úton haladva elsősorban A. M. Turing matematikust említhetjük meg. Turing 1936-ban fejlesztette ki az automatikus tárolt programozású univerzális számítógép elméleti (matematikai) modelljét. Bebizonyította, hogy az olyan gép, amely el tud végezni néhány alapműveletet, elvileg bármilyen számítás végrehajtására alkalmas. Turing használta először a „*to compute*" (kiszámítani) igét. Innen származik a „*computer*" (számítógép) elnevezés is. Ezután, 1948-ban Norbert Wiener a „Cybernetics or Control and Communication in the Animal and the Machine" (A kibernetika, az élő és élettelen rendszerek folyamatainak irányítás-elméletében) című művében megírja, hogy még 1940-ben kidolgozott öt alapelvet, amelyek később a programvezérlésű elektronikus, számítógépekben megvalósultak.

A legelső kizárólag elektronikus elemekkel működő számítógép az ***ENIAC*** (**E**lectronic **N**umerical **I**ntegrator and **C**alculator) volt. Ezt Neumann János (1903-1957) matematikus elgondolása alapján J. N. Brainerd, J. P. Eckert, J. W. Mauchly és H. M. Goldstine építették meg, és 1944 végén állították működésbe. Neumann 1946-ban tartott előadásaiban tette közzé a korszerű számítógép megépítésének alapelveit. Először is rámutatott arra, hogy az akkori gépek, amelyekben még lényeges szerepet játszották a mechanikus alkatrészek, sem túlságos lassúságuk, sem megbízhatatlanságuk miatt nem felelhettek meg rendeltetésüknek. Ezért teljesen elektronikus gépet javasolt. Másodsorban, kifejtette, hogy míg a tízes számrendszer a mechanikus szerkezeteknél tökéletesen megfelelő volt, az elektronikus gépeknél a kettes számrendszert sokkal célszerűbb alkalmazni. Ez a számrendszer kétállapotú elektronikus eszközök segítségével, könnyen megvalósítható, és ezenkívül a száminformációk tárolása minimális alkatrészt igényel. A fenti elvek alkalmazásával a gép számítási gyorsasága olyan nagy lesz, hogy értelmetlenné válik az emberi beavatkozás a gép működésébe, ez ugyanis tetemesen több időt venne igénybe; mint a műveletek elvégzése: Logikusan következik a *harmadik javaslat*: a belső memória (tár) létrehozása. Ebben a számítási részeredmények tárolhatók, és így a gép több műveletsorozatot is automatikusan el tudna végezni. A legfontosabb javaslat a *negyedik* volt, amely a számítógépek logikai irányítására vonatkozott. Ha a gép belső memóriája nemcsak adatokat tárol, hanem műveleti utasításokat is, akkor képes lépésről lépésre önállóan haladni. Minden lépés után a gépet saját memóriája utasítja a további teendőkre anélkül, hogy emberi beavatkozásért. kellene időznie.

Az utasításoknak azt a sorozatát, amely meghatározott feladatot old meg, ***programnak*** hívják. A számítógép programozhatósága tekinthető az igazán lényeges áttörésnek, amely megkülönbözteti a modern számítógépet a régi számológépektől. A tárolt program koncepciója vezetett azután az önmagukat módosító számítógépek kifejlesztéséhez.

A Neumann felsorolt alapelvei szerint működő első gép az ***EDVAC*** (**E**lectronic **D**iscrete **V**ariable **A**utomatic **C**omputer) volt, amelyet 1949-ben állítottak működésbe. Az EDVAC volt az első elektronikus digitális komputer, amit belső programtárolási koncepciónak megfelelően építettek meg. Az EDVAC-ot a Moore School of Electrical Engineering munkatársai tervezték, hasonlóan mint az ENIAC-et, de az EDVAC igen jelentősen különbözött attól. A legfontosabb eltérés az volt, hogy ez a gép tárolt programozású volt. Az EDVAC-ba beépítettek 19 különböző típusú, 3563 vákuumcsövet, 8000 kristály diódát, 1325 mágneses elemet, közelítőleg 5500 kondenzátort, 12000 ellenállást és 320 neont.

Az EDVAC 50 kW-ot fogyasztott és közel 50 négyzetméter területet foglalt el. A berendezés közel 8 tonna súlyú volt.

A mikroprocesszorokkal megvalósított mikroszámítógépek az elvi működés szempontjából elég kevés eltérést mutatnak a Wiener, valamint Neumann javaslatai alapján megépült, tárolt programozású univerzális számítógépektől. Ellenben annál nagyobb a különbség a megvalósítási technológia terén. Ennek a jelentőségét nem hanyagolhatjuk el, ugyanis a technológia fontossága már a számítógépek fejlődésének történetében megmutatkozott. A XVII. században a finommechanika fejlődése tette lehetővé Pascalnak az első mechanikus számológép megalkotását. A mechanikus és a későbbi elektromechanikus számológépek korszakának a végét a következő felfedezések jelentették: az elektronemisszió (Edison, 1883-ban), a mágnesesvezérlésű elektroncső (R. von Lieben, 1906-ban) és az audiondetektor (L. de Forest, 1907-ben). Az első elektronikus számítógép, az *ENIAC*, 18000 elektroncsövet tartalmazott. A gép gyorsasága körülbelül ezerszeresen felülmúlta az elektromechanikus gépek gyorsaságát. Ennek ellenére megmutatkoztak az elektroncsövek hátrányai. Az *ENIAC* 18 000 elektroncsövének felszerelésére 500000 ponton kellett forrasztani. Csupán a forrasztási helyek előkészítése két évig tartott. A gép 100-150 kW elektromos energiát fogyasztott. Elhelyezésére 9 × 15 méter nagyságú teremre volt szükség, súlya pedig 30 tonna körül volt.

Talán semmihez nem kötődik úgy a mai számítástechnika, mint a félvezetők fizikájához, technológiájához. A mikroprocesszorok technológiai előzményei 1948-ig nyúlnak vissza, amikor a félvezetők több évtizedes kutatása eredményeként John Bardeen, Walter H. Brattain és William Shockley, a Bell Laboratórium munkatársa felfedezték a tranzisztort. Ez nagy jelentőségű fordulópont volt, ugyanis az elektroncső működése a vákuumban mozgó elektronok vezérlésén, míg a tranzisztoré – amely majd később teljesen kiszorította az elektroncsöveket – a félvezető kristályszerkezetében lejátszódó jelenségeken alapszik. Érdekes megemlíteni, hogy 1948-ban John W. Turkey először használja a számítástechnika egyik legfontosabb és leggyakrabban használt egységét, a ***bitet***, azaz a „**bi**nary digi**t**”-ből rövidített fogalmat. A tranzisztor és a félvezetőipar fejlődésében igen jelentős eseménynek tekinthető, hogy 1952-ben az amerikai ipar nagyértékű megbízást kap a hadseregtől mind a gyártásra mind a fejlesztésre. Ez már szükségessé tette részben a gyártás automatizálását, részben a szabványosítást. A kutatás, a technológiafejlesztés, a gyártásautomatizálás és a tömegtermelés itt kezd szoros szimbiózisban élni, hatnak egymásra, táplálják egymást. A félvezetőkből készült elektronikus eszközök nemcsak képesek mindarra, amire az elektroncsövek, hanem azokhoz képest több döntő előnyük is van:

- térfogatuk az elektroncső térfogatának törtrésze;
- sokkal nagyobb a tartósságuk és a megbízhatóságuk;
- messzemenően kisebb energiát igényelnek működésükhöz;
- nagyobb sebességgel tudnak dolgozni.

Félvezetők nélkül a mai napig sem lett volna sima leszállás a Vénuszra és a Holdra. A valódi. miniatürizáláshoz főképpen a rakéta és űrrakétaprogramok adtak ösztönzést. Érthető okokból a szakemberek rá voltak kényszerülve, hogy kicsi és könnyű elektronikus berendezéseket építsenek. A növekvő követelményeket a nyomtatott áramkörök sem elégítették ki. Újabb nehézségeket is ki kellett küszöbölni: a nagyfokú zavarérzékenységet és a csökkenő megbízhatóságot, amelyeket elsősorban a nagyszámú forrasztási pont okozott.

A miniatürizálás és a performanciák, valamint a megbízhatóság terén további nagy ugrást jelentett az *integrált áramkör*. Az integrált áramkörben a különböző rendeltetésű aktív és passzív áramköri építőelemeket, valamint a hozzájuk tartozó összekötéseket egyetlen gyártási folyamatban közös félvezető alapon állítják elő. Az integrált áramkört 1960-ban fejlesztették ki a Fairchild és a Texas Instruments cégek szakemberei: R. Noyce, valamint J. St. Clair Kilby. G. Moore 1965-ben megállapítja, hogy évenként megduplázódik az egy félvezető (szilícium) lapkán levő integrált tranzisztorok száma. Az ekkor megjósolt tendencia napjainban sem veszített érvényességéből.

Az integrált áramkörök technológiájának kimagasló eredménye a mikroprocesszor megvalósítása. Az első mikroprocesszort az Intel cég szakemberei fejlesztették ki 1971-ben. ***Mikroprocesszoron*** *egy (vagy legfeljebb néhány) olyan integrált áramkörből álló rendszert értünk, amely elvégzi a programozható számítógép központi egységének a feladatkörét. A mikroprocesszoron alapuló* ***mikroszámítógép*** *pedig az a rendszer, amely a mikroprocesszoron kívül rendelkezik a szükséges memóriával, valamint ki-, illetve bemeneti információcserére alkalmas egységgel.*

A számítógépekkel kapcsolatosan két szorosan összefüggő, de mégis jól elhatárolható részt különböztetnek meg: a ***hardvert*** és a ***szoftvert*** (az angol „hardware" és „software" kifejezések kiejtésének megfelelő írásmódja). A hardver a számítógép összes műszaki (fizikai) míg a szoftver a számítógép működését és a feladatok megoldását biztosító programokat foglalja magába.

A számítási sebesség növelése céljából a legújabb típusú számítógépek felépítése nem követi Wiener és Neumann javaslatait. 1983-ban a NEC Corporation japán cég kifejlesztette a világ első nem Neumann típusú, ultrasebességű számítógépét. E számítógép, amely nem volt nagyobb, mint egy klasszikus Neumann-típusú mikroszámítógép, másodpercenként 53 millió számítási műveletet végzett, vagyis 50-100 -szor olyan gyorsan számolt, mint az előbbi. A sebességét annak köszönhette, hogy míg a Neumann-típusú számítógépeknél minden számítási műveletnél az adatokat a memóriából el kell juttatni az aritmetikai egységbe és onnan vissza, addig az új típusú számítógépnél nem kell minden egyes számítási műveletnél adattovábbítást végezni.

A nyolcvanas évek elejétől a számítástechnika és az informatika minden képzeletet felülmúló fejlődésének vagyunk szemtanúi. Egyre gyorsabb, nagyobb teljesítményű és kisebb méretű digitális számítógépek és mikroprocesszoros rendszerek jelennek meg. Ezeknek a rendszereknek az ára – a nagy sorozatban való gyártásnak, az automatizálásnak és a magas fokú integráltságnak köszönhetően – fokozatosan csökken. A különböző mikroprocesszoros rendszerek ezáltal a mindennapi élet nélkülözhetetlen részeivé válnak.

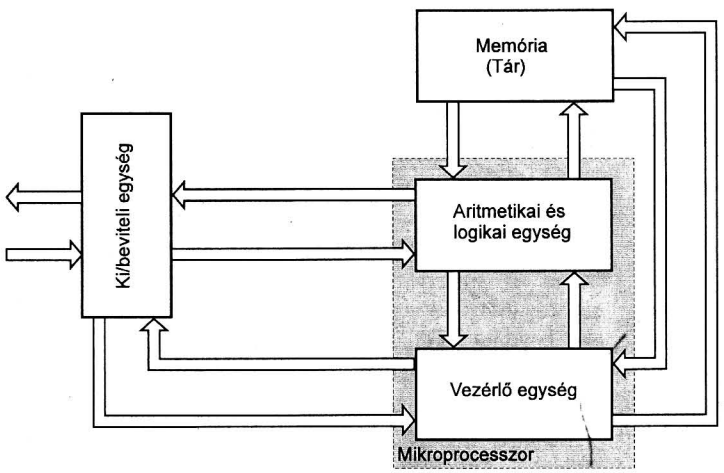
8.2. A mikroszámítógépek felépítése és működése

A mikroprocesszoros mikroszámítógépek működésének alapelveit a klasszikus értelemben vett (Neumann-féle) univerzális számítógép felépítésének és működésének tanulmányozásával érthetjük meg.

Az egymásba láncolt feladatmegoldási lépéssorozatok elemzésével a számítógép működése könnyen érthetővé válik. Adott feladat megoldása a megfelelő matematikai modell felállításával kezdődik. Ezután meg kell szerkeszteni a modellen alapuló megoldási lépések szekvenciáját. Ezt az úgynevezett programot a számítógépnek tudnia kell tárolni. Tárolásra szorulnak a feldolgozás alatt, valamint az utána keletkező rész-, illetve végeredmények. Mielőtt a gép hozzálátna a feladat megoldásához, biztosítani kell a megfelelő adatok betáplálását és tárolását. A feladat megoldása után a számítógépnek az eredményt könnyen érthető és kezelhető formában kell közölnie a külvilággal.

A felsorolt feladatok elvégzését biztosítja a számítógép négy alapegysége (*8.1. ábra*):

- az aritmetikai és logikai egység;
- a memória- vagy táregység;
- a vezérlőegység;
- a ki/bevitelí egység.



8.1. ábra. A klasszikus felépítésű univerzális számítógép rendszer-tömbvázlata

Az alábbi részletes elemzés megvilágítja az egységek működését, valamint a számítógépen belüli egymáshoz való funkcionális viszonyulását.

8.2.1. Az aritmetikai és logikai egység

Az aritmetikai és logikai egység (***ALU*** – **A**rithmetic and **L**ogic **U**nit), amint az elnevezése is mutatja, azon aritmetikai és a logikai műveletek végrehajtását teszi lehetővé, amelyekkel a program által meghatározott számolási és logikai műveletek sorozata végezhető el.

Minél többféle művelet elvégzésére képes egy számítógép aritmetikai és logikai egysége, annál könnyebben tudja megoldani a bonyolultabb feladatokat. Általában az aritmetikai és logikai egység a következő alapvetőnek számító bináris műveleteket végezheti el: két szám összeadása, valamint kivonása, egy szám jobbra vagy balra léptetése (helyérték eltolás), két számmal végzett logikai *ÉS*, *VAGY*, valamint *KIZÁRÓ-VAGY* művelet, egy szám komplemensének képzése és két szám összehasonlítása. E műveletek segítségével más komplexebb műveletek is elvégezhetők. Például a szorzást ismételt összeadás és helyérték eltolás segítségével, az osztást pedig ismételt kivonás és helyértékeltolás segítségével lehet elvégezni.

A számok összehasonlítása, valamint az aritmetikai és logikai műveletek együttese a legbonyolultabb döntést (feltételes ugrást) meghatározó műveletek elvégzését teszi lehetővé. A döntési műveletek a számítógép „intelligens" működésének alapját képezik.

Az aritmetikai és logikai egység központi regisztere az úgynevezett ***akkumulátor*** (angolul: *accumulator*). Ebben a művelet végrehajtása előtt az egyik operandus, a művelet végrehajtása után pedig az eredmény található. Az akkumulátoron kívül az aritmetikai és logikai egységben még egy ***regisztertömb*** (angolul: *scratch pad, register array*) is van. Ennek a regiszterei bizonyos utasítások végrehajtásában vesznek részt. Egy-egy operandust vagy eredményt tárolnak. Az akkumulátor és a regisztertömb a memóriából és a ki/beviteli egységen keresztül kapja az adatokat. Feladatuk elvégzése után tartalmukat a memóriában tárolják, vagy a ki/beviteli egységen keresztül szolgáltatják ki.

Egy műveletet a számítógép aritmetikai és logikai egysége egy vagy több utasítás segítségével végez el. Az utasítás a program legkisebb funkcionális egysége.

8.2.2. A memória

A memória vagy táregység (angolul: Memory Unit, Store Unit) nagy funkcionális fontosságú, és rendszerint a számítógép legnagyobb egységének tekinthető. Memórián azokat az eszközöket értik, amelyek az információkat tetszés szerinti ideig megőrzik.

A memória a program és az adatok tárolását teszi lehetővé. A programmemóriában található a programot alkotó utasítássorozat. Az adatmemóriában találhatók a feladat megoldásához szükséges kezdeti adatok, a program futtatása során keletkezett részeredmények, valamint a program befejezése utáni végeredmények.

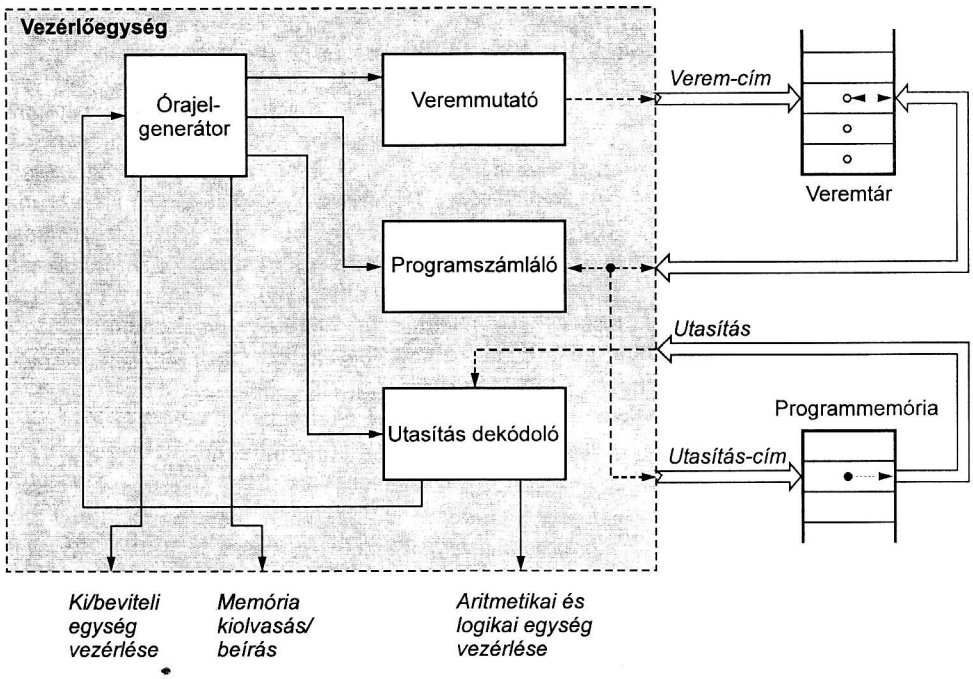
A memória egy-egy utasítás vagy adat tárolására kijelölt részét ***rekesznek***, a rekesz azonosítására szolgáló sorszámot pedig ***címnek*** nevezik. A memória által tárolható információmennyiség mutatója a ***kapacitás***. Egy-egy cím megadása után, amíg a rekesz tartalmát ki lehet olvasni vagy adatot lehet beírni, bizonyos időnek kell eltelnie. Ez az úgynevezett *hozzáférési* vagy *elérési idő*. A kapacitás és a hozzáférési idő a memóriák lényeges jellemzői.

A *8.1. ábrán* bemutatott tömbvázlatban levő memória a számítógép úgynevezett ***operatív memóriája*** (angolul: *main memory*). Használatosak a központi, belső vagy elsődleges memória elnevezések is. A számítógép teljesítményének növelése céljából az operatív memóriát kiegészítik úgynevezett ***külső*** vagy ***másodlagos memóriával*** is (secondary memory). Ennek a memóriának az elérése a ki/beviteli egységen keresztül valósul meg. Szükség esetén a külső memóriában tárolt információt át lehet vinni az operatív memóriába, vagy fordítva. a külső memória kiegészíthető a nagy adattömeget tároló ***háttér memóriával*** (angolul: *mass storage*). Ennek kapacitása nagyobb, de az előbbiekhez képest a hozzáférési ideje is jóval nagyobb.

8.2.3. A vezérlőegység

A vezérlőegység (angolul: *Control Unit*) szerepe a számítógép működésének, tehát a műveletek program szerinti végrehajtásának az irányítása is. Ez az egység rendszerint a következő négy részegységből épül fel (*8.2. ábra*):

- a programszámláló;
- a veremtár- (memória-cím) mutató;
- az utasításdekódoló;
- az órajelgenerátor.

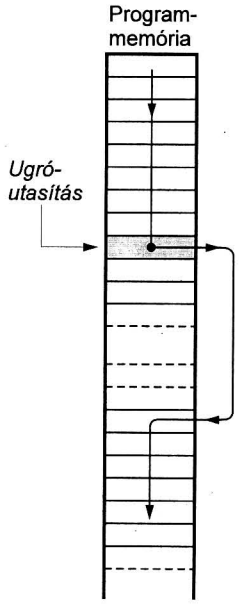


8.2. ábra: A vezérlőegység felépítése és funkcionális összeköttetései

Az egység működését a felsorolt részegységek funkcionális elemzése alapján érthetjük meg.

8.2.3.1. A programszámláló

A programszámláló (***PC*** – **P**rogram **C**ounter) a soron következő utasítás címét jelöli ki. Minden egyes utasítás kiolvasása után az órajelgenerátor a programszámláló tartalmát eggyel megnöveli. A programszámláló így biztosítja az utasítások lépésről lépésre való elérését. Abban az esetben, ha a gép egy ún. ***elágaztató*** vagy ***ugróutasításhoz*** ér, amelynek az a szerepe, hogy megszakítsa az utasítások növekvő sorrendű elérését, akkor a programszámláló tartalma egy adott címre cserélődik.



8.3. ábra. Programelágazás egy ugróutasítás esetén

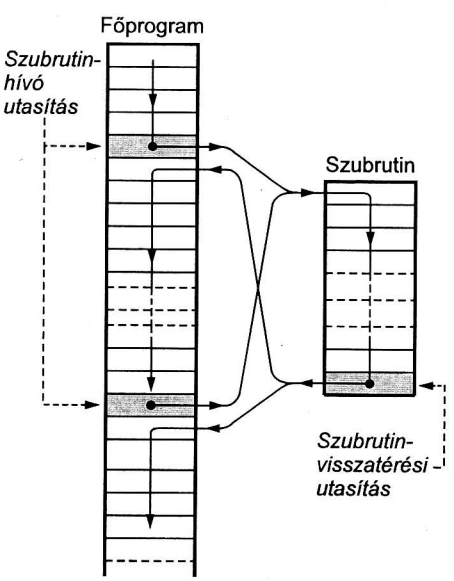
Ez után az ugrás után a programszámláló az új címtől kezdve lépésről lépésre tovább növeli tartalmát (*8.3. ábra*).

Az ugróutasítások lehetnek feltételhez kötöttek vagy feltétel nélküliek. A ***feltételes ugróutasítást*** a számítógép egy döntési művelet után hajtja végre. A döntés eredményétől függően, amely – mint láttuk – egyszerűbb vagy bonyolultabb összehasonlítási műveletekből származik, programelágazás jöhet létre. A ***feltétlen ugróutasítás*** esetén a gép program végrehajtásában feltétel nélküli ugrás következik.

8.2.3.2. A verem (memória-cím) mutató

A veremmutató (***SP*** – **S**tack **P**ointer) és a veremtár (*Stack*) szerepe ***alprogramok*** (más néven ***szubrutinok***) alkalmazása esetén nyilvánul meg. A programozás egyszerűsítésének legfontosabb segédeszköze a *szubrutin*. Különböző vagy azonos feladatot leíró programban megtörténhet, hogy azonos részek többször fordulnak elő. Ezeket úgy célszerű felállítani, hogy a program bármelyik részén közvetlenül fel lehessen használni. Azt a programrészt, amelyet a program különböző helyein használnak fel, de csak egyszer programoznak be, ***szubrutinnak*** nevezik (*8.4. ábra*).

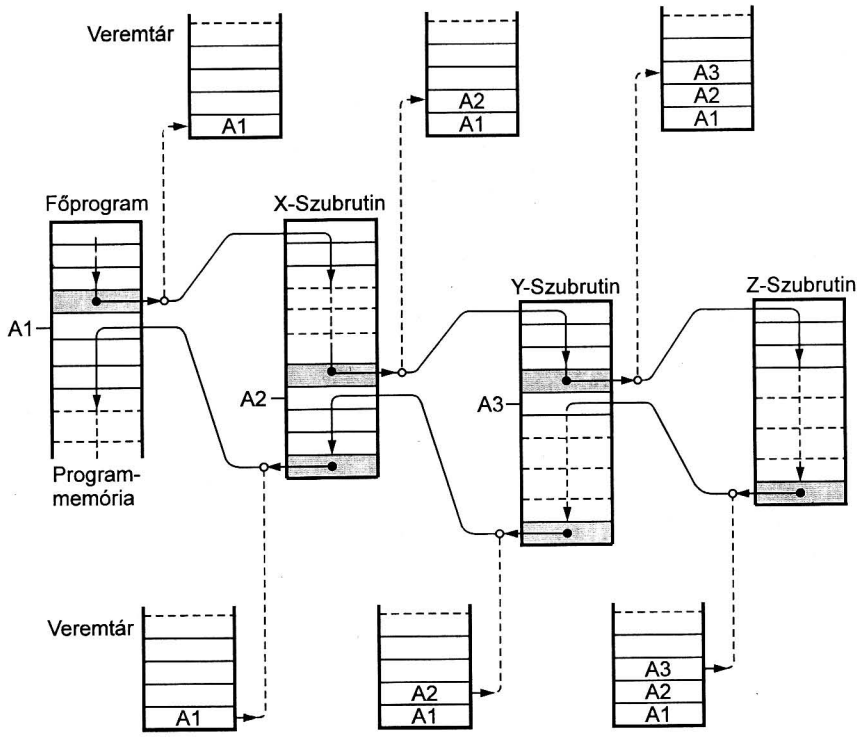
A szubrutint a főprogramba két utasítás segítségével ékelik be: a ***szubrutinhívó*** és a ***szubrutin-visszatérési*** utasítással. A szubrutinhívó utasítás a programszámlálóba a szubrutin kezdő címét írja be, a szubrutin-visszatérési utasítás pedig a főprogramnak azt a címét, ahová vissza kell térni. Ez a cím rendszerint a szubrutinhívó utasítás utáni cím, amelyet még a szubrutinra való ugrás előtt szükséges tárolni. Ez a tárolás a veremmemóriában történik.



8.4. ábra. Egy szubrutin felhasználása a főprogram különböző helyein

A veremtár beírási sorrendben tárolja a címeket, kiolvasáskor pedig fordított sorrendben adja vissza őket. Ezért a veremmemória egy ún. LIFO (**L**ast – **I**n, **F**irst – **O**ut: az utolsó – be, az első – ki) típusú memória. A legutolsó beírt adat címét, tehát a legelsőnek kiolvasandó adatcímet a veremmutatóból lehet kiolvasni (*8.2. ábra*).

A veremmemória és a veremmutató hasznossága főleg a több szintű egymásba skatulyázott szubrutinok esetén nyilvánul meg (*8.5. ábra*).



8.5. ábra. A veremtár szerepe a többszintű, egymásba skatulyázott szubrutinok használatával

Ebben az esetben a szubrutinhívó utasítások hatására a veremtár a visszatérési címeket a szubrutin hívások sorrendjében tárolja, visszatéréskor pedig fordított sorrendben adja vissza őket. Ily módon a visszatérés mindig az utoljára megszakított programra (szubrutinra vagy főprogramra) történik. A veremmemóriában nemcsak címeket lehet tárolni, hanem adatokat is, mint például az akkumulátor és a regiszterek tartalmát.

8.2.3.3. Az utasításdekódoló

Az utasításdekódoló (angolul: Instruction Decoder) a vezérlőegység legfontosabb része. Szerepe abban nyilvánul meg, hogy az utasításokat ábrázoló kódszámokat megfelelő vezérlőjelekké alakítja át. A vezérlőjelek egy része közli az aritmetikai és logikai egységgel a végrehajtandó művelet típusát. A vezérlőjelek másik része a számítógépen belüli információáramlást ellenőrzi és szabályozza.

8.2.3.4. Az órajelgenerátor

Az órajelgenerátor (Clock Generator, Timing Unit) állítja elő a gép időbeni működéséhez szükséges vezérlőjeleket. Ezek rendeltetései a következők: az aritmetikai és logikai egység vezérlése (az utasításdekódoló által jelzett művelet elvégzése), az információk kiolvasása és beírása a memóriába, a ki/beviteli egység működésének vezérlése és a vezérlőegység időbeni működésének irányítása.

Az órajelgenerátor által szolgáltatott vezérlőjeleket részben a végrehajtandó utasítás határozza meg. Ezért a dekódoló egység irányító információkat közöl az órajelgenerátorral.

A számítógép időbeni működését rendszerint a következő négy gépi ciklus írja le:

1. ciklus – az ún. címregiszterbe beíródik a programszámláló tartalma, amely a végrehajtandó utasítás programmemóriabeli címe;
2. ciklus – a programszámláló tartalma eggyel növekszik; ezzel előkészül a következő utasítás megcímzésére;
3. ciklus – az 1. ciklus alatt megcímzett utasítás beíródik az ún. utasításregiszterbe;
4. ciklus – az utasításregiszter tartalmát értelmezi az utasításdekódoló, amely biztosítja az utasítás által meghatározott művelet végrehajtását.

Abban az esetben, ha az előbbi utasítás nem egy leállító utasítás volt, az órajelgenerátor indítja a következő 1. gépi ciklust.

8.2.4. A ki/beviteli egység

A számítógép és az ember, valamint a számítógép és az általa vezérelt berendezés közti kapcsolatot a ki/beviteli egység teszi lehetővé (Input/Output Unit). Általában a ki/beviteli egység felelős a számítógép és a külvilág közti információcseréért.

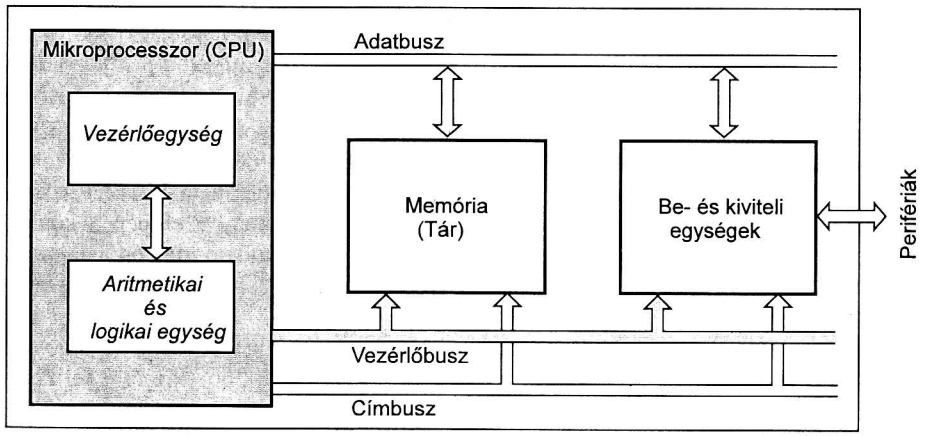
A beviteli egység segítségével visszük be a számítógép operatív memóriájába a megoldandó feladatot leíró programot és az ehhez szükséges adatokat. A bevitelre kerülendő információ előzőleg lyukszalagon, lyukkártyán, mágneslemezen, mágneskártyán vagy mágnesszalagon lehet tárolva. Innen ezt az információt a sajátos leolvasómű a gép számára kompatibilis elektromos jellé alakítja át. Ezen kívül a számítógép alkalmas az információ kézi (manuális) betáplálására is. Erre szolgál a billentyűzet vagy más nevén a klaviatúra. Abban az esetben, ha a számítógép ipari folyamatot szabályoz, a beviteli egységnek a folyamat paramétereit figyelő érzékelők és átalakítók által szolgáltatott elektromos jeleket is kell; tudnia venni. Legtöbbször ez a jel a megfigyelés alatt álló paraméterrel arányos analóg jel. Ilyenkor analóg-digitális konverter (átalakító) alkalmazása szükséges.

A kiviteli egységnek a számítási folyamat eredményeit további felhasználásra is alkalmas formában kell megadnia. Ezt elektromos írógép, nyomtató, rajzolókészülék, lyukszalag-, vagy lyukkártya-lyukasztó segítségével lehet megvalósítani. A számítógépeknél népszerű a katódsugárcsöves megjelenítő használata is. Ennek segítségével nemcsak írott szövegként (számok és betűk) kaphatjuk kézhez az eredményt, hanem rajz formájában is. Ipari folyamatok szabályozása esetén a kiviteli egység olyan elektromos jeleket is kell, hogy szolgáltasson, amely motorokat, elektromechanikus csapokat és más vezérlőműveket is képes működtetni.

A számítógép ki/beviteli egységéhez kapcsolt ki-, illetve beviteli készülékeket (például: lyukszalagolvasó, billentyűzet, egér, nyomtató, katódsugárcsöves megjelenítő) perifériáknak nevezik. A perifériák egy sajátos illesztőegységen, ún. interfészen keresztül csatolódnak a ki/beviteli egységhez.

8.3. Mikroprocesszorok

A mikroprocesszor tulajdonképpen a számítógép „agya". A mikroprocesszor szó a számítógépes terminológiában már korábban is ismert volt. A mikroprogramozható számítógépek mikroprogramját feldolgozó központi egységének a processzorát értették mikroprocesszoron. Jelenleg ezt a kifejezést tágabb értelemben használják. Tekintsünk vissza az univerzális számítógép elvi felépítéséhez.



8.6. ábra. A mikroszámítógépek egyszerűsített tömbvázlata

A *8.6. ábrán* bemutatott mikroszámítógép tömbvázlata a *8.1. ábrán* látható klasszikus architektúrájú univerzális számítógép tömbvázlatából megfelelő átcsoportosítással alakítható ki. A mikroszámítógép három alapvető egységét lehet megkülönböztetni:

- a memória (Memory),
- a központi feldolgozó egység (CPU – **C**entral **P**rocessing **U**nit) és
- a ki-, illetve beviteli egység (Input/Output Unit).

A *mikroprocesszor* elnevezés a központi feldolgozó egységre vonatkozik. Ez egy olyan LSI áramkör, amely egy vagy néhány tokba van beépítve. Felépítésében két alapvető egység különböztethető meg: az *aritmetikai és logikai egység*, valamint a *vezérlőegység*.

A mikroszámítógép felépítéséhez a mikroprocesszort elsősorban memória- és ki-, illetve beviteli áramkörökkel, valamint többé kevésbé a vezérlést kisegítő áramkörökkel (mint például órajelgenerátorral, állapotdekódolóval, meghajtókkal, stb.) kell kiegészíteni.

A mikroszámítógép építőelemeit párhuzamos vezetékcsoport-együttesből álló, ún. ***busz-*** vagy ***sínrendszer*** köti össze. Az adatáramlás az adatbuszon bonyolódik le; a címeket a mikroprocesszor a címbuszon küldi ki; és végül a vezérlőjeleket a vezérlőbusz juttatja el a mikroszámítógép összes egységéhez. A *8.6. ábra* egy egy adatbusszal rendelkező, hagyományos Neumann-féle, ún. ***SISD*** (**S**ingle **I**nstruction – **S**ingle **D**ata Stream = egy utasítás – egy adatáramlás) mikroszámítógépet szemléltet. Gyorsabb adatfeldolgozás és adattárolás céljából bonyolultabb szervezésű számítógépeket fejlesztettek ki, mint például az

SIMD (**S**ingle **I**nstruction – **M**ultiple **D**ata Stream = egy utasítás – sokszoros adatáramlás) vagy az ***MIMD*** (**M**ultiple **I**nstruction – **M**ultiple **D**ata Stream = többszörös utasítás – többszörös adatáramlás).

Az első mikroprocesszort 1971-ben készítették el az Intel cég szakemberei. Ez az Intel 4004 mikroprocesszor, 4-bites adatok feldolgozására volt képes, és egy japán asztaliszámológép-gyártó cég rendelése nyomán valósították meg. Számítási teljesítménye közepesen aluli volt, inkább egy számológépre emlékeztetett. A következő az Intel cég 8008-as mikroprocesszora volt. Ezt 1972-ben, egy katódsugárcsöves megjelenítő részére dolgozták ki. Közben a megrendelő bipoláris áramkörökkel oldotta meg a feladatát, és lemondta a megrendelést. Utólag az Intel cég ezt a 8-bites processzort piacra dobta, és a nagy igénylés nyomán kidolgozta a második 8-bites típust – a 8080-as mikroprocesszort. Ez 1974-ben került forgalomba, és az első sikeres mikroprocesszornak tekinthető. Egy évvel később megjelent a Motorola cég 6800-as, néhány évvel később a Zilog cég Z80-as mikroprocesszora, hogy csak az elterjedtebb 8-bites mikroprocesszorokat említsük. A hetvenes évek végén megjelentek az egytokos mikroszámítógépek (Microcomputer on a chip). A processzoron kívül a chipen található a memória, valamint a ki- illetve beviteli áramkör is. Az egyik legbeváltabb típus volt az Intel 8048, amely a chipen a következő egységeket tartalmazza: 8-bites CPU, 1 kbájtos ROM programmemória, 64 bájtos RAM adatmemória, három 8-bites ki/beviteli kapu, 8-bites időjelgenerátor, órajelgenerátor és megszakításvezérlő.

A mikroprocesszorok teljesítőképessége az egyidejűleg feldolgozandó adatszó hosszával növekedik. Így jelentek meg a 12, 16 (pl. Intel 286), 32 (pl. Intel 386, 486 és Intel Pentium család), valamint a 64 bites mikroprocesszorok.

A mikroprocesszorok teljesítménye növelhető a ***segédproceszszorokkal*** (Co-processor). Példaként megemlítjük az Advanced Micro Devices cég Am 9511A és Am 9512A valamint az Intel cég 8087, 287, 387 jelű aritmetikai műveletekre specializált segédprocesszorait. Segítségükkel 16, 32, illetve 64 bites fixvesszős, valamint 32, illetve 64 bites formátumú lebegővesszős számokkal a következő műveletek végezhetők el: összeadás, kivonás, szorzás, osztás, négyzetgyökvonás, hatványra emelés, logaritmus és trigonometriai függvények számítása, valamint átalakítások: a fixvesszős formátumból a lebegővesszősbe és fordítva. Ezek a műveletek a segédprocesszorral sokkal gyorsabban végezhetők el, mint nélküle. Ezen kívül számottevően csökkenthető a program memória kapacitása azáltal, hogy nincs szükség a műveleteket elvégző szubrutinokra.

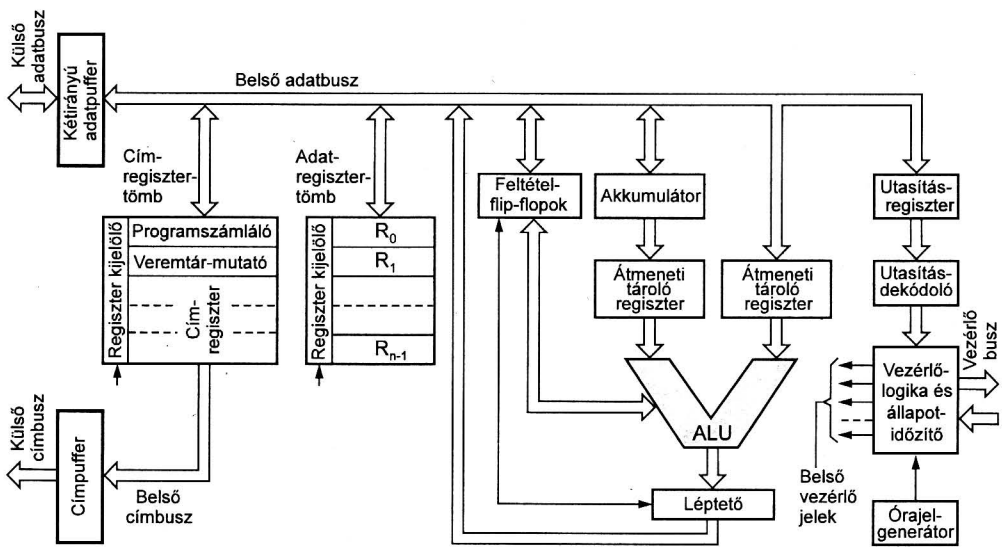
Különböző adatfeldolgozási és vezérlési műveletekre specializált processzorokat is kifejlesztettek. Például az Intel 2920 jelprocesszorát, amely az analóg jelek valós idejű, digitális feldolgozására alkalmas. Az analóg bemeneti jel mintavételezését és digitálissá való átalakítását a chipen levő áramkörök végzik. A digitális jelet a processzor, az ugyancsak a chipen levő 192 × 24 bites EPROM programmemóriába betáplálható algoritmus alapján dolgozza fel. A kimenet előtt levő digitális/analóg átalakító az így kapott digitális jelet analóggá alakítja át. A TRW cég TDC1008/1009/1010 8-, 12, illetve 16 bites gyors jelprocesszorai segédprocesszorként is használhatók. Sebességüket a gyors összeadást, kivonást, valamint szorzást végző aritmetikai egységük biztosítja. Segítségükkel valós idejű gyors Fourier-transzformálás, digitális szűrés és vektorszorzás valósítható meg.

Végül a ***bitszelet mikroprocesszorokat*** (angol elnevezése: *bit-slice microprocessor*) említjük meg, amelyek a mikroprogramozás eszközei. A mikroprocesszor tulajdonképpen úgy tekinthető, mint egy sorrendi logikai hálózat. Bonyolultabb sorrendi hálózatok tervezése legtöbbször komoly nehézségekbe ütközik. Ezenkívül a rugalmasságuk meglehetősen alacsony, és a továbbfejlesztési lehetőségeik korlátozottak. Ezek a hátrányok egy

mikrovezérelt rendszerrel küszöbölhetők ki. A vezérlőinformációt úgynevezett vezérlőmemória tárolja. A pillanatnyi vezérlőmemória-cím tartalma adja az aktuális vezérlési állapotot. A következőt pedig az előző állapot, a vezérlőmemória tartalma és a külső feltételek szabják meg. A mikrovezérelt rendszer vezérlőmemória tartalmának az adott feladat függvényében való előállítását mikroprogramozásnak nevezik. A legbeváltabb az Advanced Micro Devices Am 2900 4-bites bitszelet mikroprocesszor családja. A kívánt szó hosszúságnak megfelelően több bitszelet processzor kapcsolható párhuzamosan.

8.3.1. A mikroprocesszorok felépítése és működése

A *8.7. ábra* a mikroprocesszorok közös funkcionális tulajdonságait veszi figyelembe. A továbbiakban ismertetésre kerülő elemzés alapján a különböző típusok felépítése és működése könnyen érthetővé válik.



8.7. ábra. Tipikus mikroprocesszor-tömbvázlat

8.3.1.1. A buszrendszer (Bus System)

Megkülönböztethető a ***külső*** és a ***belső buszrendszer***. A külső buszrendszer, a mikroszámítógép mikroprocesszorán kívül levő építőelemeit köti össze. A belső buszrendszer a mikroprocesszor belső egységeit köti össze és a chipen van kialakítva.

A buszrendszer három részre tagolódik.

1. Az adatbusz (*Data bus*) a különböző egységek közötti kétirányú adatátvitelt bonyolítja le. A belső adatbusz a kétirányú adatátvitelre alkalmas adatpufferen keresztül csatlakozik a külső adatbuszhoz. Ha a mikroprocesszor adatot kap, akkor az adatpuffer bemenő állapotban van. Adatkiküldés esetén az adatpuffer kimenő állapotba kerül. A külső adatbusz-meghajtók az ún. ***közvetlen memóriahozzáférés*** (DMA – **D**irect **M**emory **A**cces) alatt kerülnek a

harmadik, nagy impedanciájú állapotba. Ekkor a közvetlen memóriahozzáférést lebonyolító egység kerül összeköttetésbe a memóriával.

2. A címbusz (*Address bus*). A címpuffer csatlakoztatja a belső címbuszt a külsőhöz. A címpuffer csak egyirányú jelátvitelre alkalmas. A mikroprocesszor az általa előállított cím segítségével jelöli ki azt az egységet, amellyel éppen érintkezésbe fog lépni. A közvetlen memóriahozzáférés alatt a mikroprocesszor felszabadítja vezérlése alól a címbuszt. A buszmeghajtók a harmadik, nagy impedanciájú állapotba kerülnek. Ezáltal a. memóriát a közvetlen hozzáférést lebonyolító egység címezheti meg.

3. A vezérlőbusz (*Control bus*) szerepe a különböző vezérlő és szinkronizáló jelek továbbítása. A mikroszámítógép vezérlését a mikroprocesszor által előállított jelek biztosítják. Válaszként a számítógép áramkörei állapotelismerő jeleket küldenek vissza. A mikroprocesszor műveletvégrehajtást vezérlő vagy állapotkérő jeleket is kaphat a perifériás áramköröktől. A kérő jel (*request signal*) átvételét elismerő jellel (acknowledge signal) nyugtázza.

A mikroprocesszorok buszmeghajtóit általában egy TTL vagy 5-10 MOS áramkörrel lehet leterhelni. Ezért – az egészen kis számítógépek kivételével – a buszrendszerre teljesítmény-meghajtókat kapcsolnak.

8.3.1.2. Az aritmetikai és logikai egység (ALU – Arithmetic and Logic Unit)

Az aritmetikai és logikai egység bizonyos számú aritmetikai és logikai művelet végrehajtását teszi lehetővé. A nagy teljesítményű mikroprocesszorok aritmetikai és logikai egysége többféle művelet elvégzésére képes, mint a kis teljesítményűeké, általában mindkettő a következőkre:

- összeadás és kivonás (aritmetikai műveletek),
- komplementálás, ÉS, VAGY és KIZÁRÓ-VAGY (logikai műveletek).

A műveletek egy vagy két operandusra vonatkoznak. Az operandus vagy az operandusok egyike az *akkumulátorban* (accumulator) van. A műveletek eredménye is az akkumulátorba kerül. Minden mikroprocesszornak legkevesebb egy akkumulátora van, de létezik olyan is, amelyet több akkumulátorral látnak el. Az operandusok az aritmetikai és logikai egységhez egy-egy átmeneti tárolóregiszteren keresztül jutnak el. Ezek elszigetelik az egység kimenetét a bemeneteitől. Három belső adatbusszal rendelkező mikroprocesszoroknál az átmeneti tárolóregiszterek fölöslegessé válnak. Az operandusok két különálló buszon jutnak az aritmetikai és logikai egységhez. Az eredmény a harmadik buszra kerül.

Az akkumulátor mellett levő ún. feltétel- vagy státuszregiszter (angol elnevezése: Flag, Status-register) flip-flopjainak állapota az éppen végrehajtott művelet eredményétől függően alakul. Ezért jelző biteknek is szokták nevezni. Feltételes ugróutasítások előtt a mikroprocesszor egy adott feltételbit értékét hasonlítja össze a logikai változó **0** vagy **1** értékével. Az összehasonlítás eredményének függvényében alakul a program további végrehajtása.

A feltételregiszter általában a következő bitekből áll:

1. **CY** (Carry) – átvitelbit, amely úgy tekinthető mint az n-bites akkumulátort bővítő $(n+1)$ legnagyobb helyértékű bit. Ez a bit tartalmazza a legnagyobb helyértékű oszlop összeadása után keletkező átvitelt. Segítségével n-nél nagyobb szóhosszúságú adatok összeadása is lehetséges.

2. **V** (Overflow) – a túlcsordulásbit, amelyet **O**-val is szoktak jelölni, két előjeles szám összeadása után keletkező átvitelt mutatja. Ha értéke **1**, akkor a két n-bites 2-es komplemensű szám összege egy $(n+1)$-bites szám. A legnagyobb helyértékű oszlop, valamint a közvetlenül előtte levőnek az összeadása után keletkező átvitelekkel végzett KIZÁRÓ-VAGY művelet eredményezi a túlcsordulást.

3. **N** (Negative) vagy **S** (Sign) – az előjelbit. Ez a 2-es komplemensben ábrázolt számok előjelét mutatja. Ha értéke l, akkor az akkumulátor negatív számot tartalmaz. Ha **0**, akkor ellenkezőleg, az akkumulátor tartalma pozitív vagy nulla. A 2-es kiegészítős számábrázolás figyelembevételével az előjelbit egyenlő az akkumulátor legnagyobb helyértékű bitjével.

4. **H** (Half-carry) vagy **AC** (Auxiliary-carry) a BCD kódolású számokkal végzett műveleteknél játszik szerepet, és az első elnevezése a 8-bites mikroprocesszorokra vonatkozik. Két decimális számjegyű két BCD szám legkisebb helyértékű számjegyének az összeadásánál keletkező átvitelt jelöli. Tulajdonképpen a legkisebb helyértékű decimális számjegy legnagyobb helyértékű oszlopának (a 8-bit 4. bitje) az összeadásánál keletkezik.

5. **Z** (Zero) – a zerobit, amelynek értéke **1**, ha egy művelet eredményeként az akkumulátor tartalma nullává válik.

6. **P** (Parity) – a paritásbit. Ez a bit jelzi az akkumulátorban levő **1**-esek páros vagy páratlan számát. Páros számú egyesek esetében a paritásbit értéke **1**. A paritásbit fontossága az adatátvitelnél nyilvánul meg. A külső zavarok miatt hamisan érkezett információ detektálását teszi lehetővé. Ha a beérkezett adat paritásbitje azonos a küldött adatéval, akkor a helyességének valószínűsége nagy.

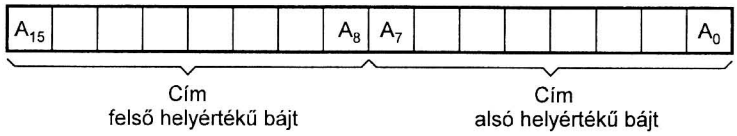
7. **I** (Interrupt) – megszakításbit, amelynek szerepe a program megszakítás-ellenőrzésénél nyilvánul meg.

Az aritmetikai és logikai egység léptetési műveleteket (angolul: shift, rotate) is végez. Ezeket a ***léptető*** (shifter) hajtja végre.

8.3.1.3. A mikroprocesszor belső regiszterei (Internal registers)

A mikroprocesszorok két regisztertömbbel (Register array) rendelkeznek, ezek: az ***adatregisztertömb*** (Data registers, Scratch pad) és a ***címregisztertömb*** (Address registers).
Az adatregisztertömb rendeltetése egy-egy operandus vagy eredmény átmeneti tárolása, és az akkumulátorral azonos hosszúságú regiszterekből tevődik össze. A regiszterkijelölő áramkör a végrehajtás alatt álló utasítás függvényében a regiszterek egyikét választja ki, és az adatbuszhoz kapcsolja. A regisztereket általában kettesével vagy négyesével is össze lehet kapcsolni. Ilyenkor a regiszterkijelölő egy kétszer vagy egy négyszer hosszabb regisztert választhat ki. Ezáltal könnyebb a nagyobb szóhosszúságú adatok feldolgozása.

A címregisztertömb legfontosabb regisztere a ***programszámláló*** (Program counter), és ez akármelyik mikroprocesszorban megtalálható. A programszámláló a soron következő utasítás címét tartalmazza, amelyet a címpuffer juttat a külső címbuszra. A 8-bites mikroprocesszorok esetén a címszó általában 16 bites. Ezzel összesen 2^{16} = 65 536 címhely különböztethető meg.



8.8. ábra. 8-bites mikroprocesszorok 2-bájtos címe

A felső helyértékű címbájttal (*8.9. ábra*) 256, ***lap***nak is nevezett memóriamező címezhető meg. Ezért a felső helyértékű címbájtot ***lapcím***nek is szokták nevezni. Az alsó helyértékű címbájttal egy lap keretén belül 256 rekesz címezhető meg, Mivel egy rekeszben egy adatszó tárolható, az alsó helyértékű címbájtot ***szócímnek*** is nevezik. A címet rendszerint hexadecimális karakterekkel ábrázolják (*8.2. táblázat*).

LAPCÍM	00		01		FF	
SZÓCÍM	00	00	00	00	00	00
		01		01		01
		02		02		02
		.		.		.
		.		.		.
		.		.		.
		.		.		.
		.		.		.
		09		09		09
		0A		0A		0A
		0B		0B		0B
		0C		0C		0C
		0D		0D		0D
		0E		0E		0E
		0F		0F		0F
		10		10		10
		11		11		11
		12		12		12
		.		.		.
		.		.		.
		.		.		.
		.		.		.
		.		.		.
		FE		FE		FE
		FF		FF		FF

8.2. táblázat. A 8-bites mikroprocesszorok 16-bites címei hexadecimális karakterekkel kifejezve

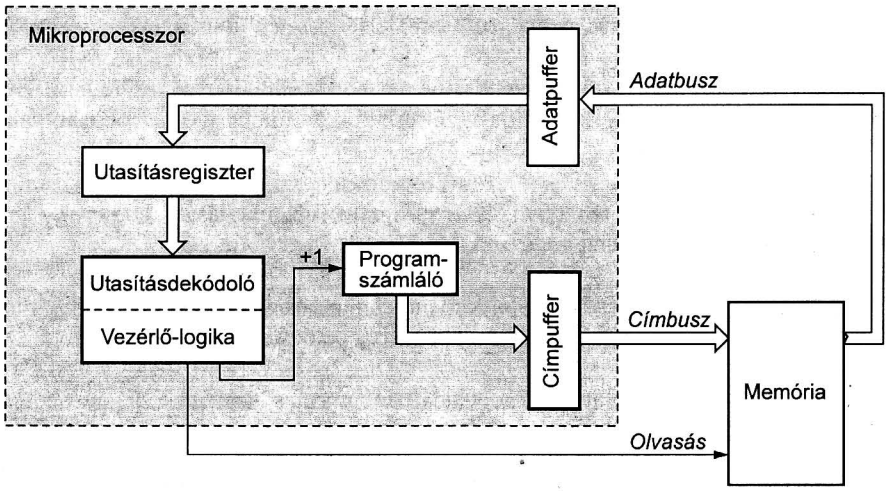
A ***veremtár-mutató*** (angolul: Stack Pointer) egy másik fontos címregiszter. A veremtár-mutató a veremmemória (stack memory) legfelső rekeszének címét tartalmazza.

A címregiszterek közül még megemlítjük az ***indexregisztert***. Ennek szerepével a továbbiakban, a címzési módszerek bemutatásánál ismerkedhetünk meg.

8.3.1.4. Az utasításregiszter (Instruction Register) és az utasításdekódoló (Instruction Decoder)

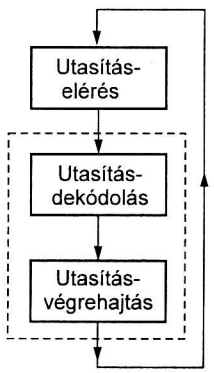
A mikroprocesszor időbeni működésére jellemző az *utasításciklus*. Ezt két egymásutáni utasítás kezdete határolja el, és három alapvető fázisra tagolódik:

- az ***utasításelérés***,
- ***utasításdekódolás*** és az
- ***utasításvégrehajtás***.



8.9. ábra. Az utasítás kiolvasása és dekódolása

Az *utasításelérés* alatt a mikroprocesszor a programszámláló által kijelölt címről kiolvassa az utasítást. Ez az adatpufferen keresztül eljut az utasításregiszterbe (*8.9. ábra*). A következő fázisban az utasításdekódoló az utasításregiszter tartalmát dekódolja. Végül, az utolsó fázisban, a vezérlő logika a dekódolt utasítás függvényében irányítja a mikroprocesszor, valamint a mikroszámítógép egységeit az utasítás végrehajtása céljából. Az utasításdekódolás folyamata, ha a mikroprocesszort rendszer-alkatelemként tekintjük, elkülöníthetetlen a végrehajtás folyamatától (*8.10. ábra*).



8.10. ábra. A mikroprocesszor tipikus utasítás ciklusa

Egyes mikroprocesszorok működési sebességének növelése céljából az átlapolt processzorciklusú (angol elnevezése: overlapped processing cycle) megoldást használják. Ez nagyjából abban áll, hogy a mikroprocesszor, miközben az utasítást a program-memóriából olvassa ki, az azelőtti utasítás végrehajtását is végzi.

8.3.1.5. A vezérlőlogika és állapotidőzítő (Control and Timing Unit)

A vezérlőlogika és állapotidőzítő egy szinkron sorrendi hálózat. Ennek az órajelét az órajelgenerátor szolgáltatja. Bemeneteire a mikroprocesszorhoz érkező külső vezérlőjelek, valamint az utasításdekódoló kimenőjelei vannak csatolva. A következő négy külső vezérlőjelet említjük meg mint legfontosabbakat:

- **RESET** – *jel*, a kezdeti feltételek beállítására (mint például a programszámláló nullázása, az akkumulátor és a státusregiszter törlése);
- **READY** – *jel*, a visszajelzés arról, hogy a memória vagy a ki/beviteli áramkör kész a program további végrehajtására; nemleges visszajelzés esetén a mikroprocesszor WAIT (várakozási) – állapotba kerül;
- **HOLD** – *állapot kérő jel*, a közvetlen memóriahozzáférést teszi lehetővé; a HOLD-állapotban a mikroprocesszor felszabadítja vezérlése alól a címbuszt, az adatbuszt, valamint a vezérlőbusz egy részét;
- **INTERRUPT** – *megszakításkérő jel* (a program megszakítására); a mikroszámítógép megszakíthatja az éppen folyamatban levő program végrehajtását, hogy egy másik, pillanatnyilag fontosabb műveletet végezzen el.

A vezérlőlogika az utasításdekódoló kimenőjeleinek segítségével határozza meg az utasítás végrehajtásának lépéseit. Feladatkörét az alábbi példák szemléltetik:

– regiszterek közötti adatátvitelt meghatározó utasítás esetében lehetővé teszi a forrásregiszterből a célregiszterbe való adatáramlást a belső adatbuszon;
– aritmetikai vagy logikai művelet esetében biztosítja az operandusok eljutását az aritmetikai és logikai egység tárolóregisztereibe, a kért művelet végrehajtását, és végül az eredmény átvitelét az akkumulátorba (kivételes esetekben egy regiszterbe vagy egy memóriarekeszbe);
– memóriából való adatkiolvasás esetében a memóriarekesz címét a címbuszra, az olvasó jelet a vezérlőbuszra és végül a kiolvasott adatot az adatbuszon a célregiszterig juttatja;
– memóriába való adatbeírás esetében a memóriarekesz címét a címbuszra, a tárolandó adatot az adatbuszra és végül az írásjelet a vezérlőbuszra juttatja.

A vezérlőlogika és állapotidőzítő az utasítások végrehajtásán kívül növeli a programszámláló tartalmát, amely a soron következő utasítás címe. Kivételt képeznek az ugróutasítások.

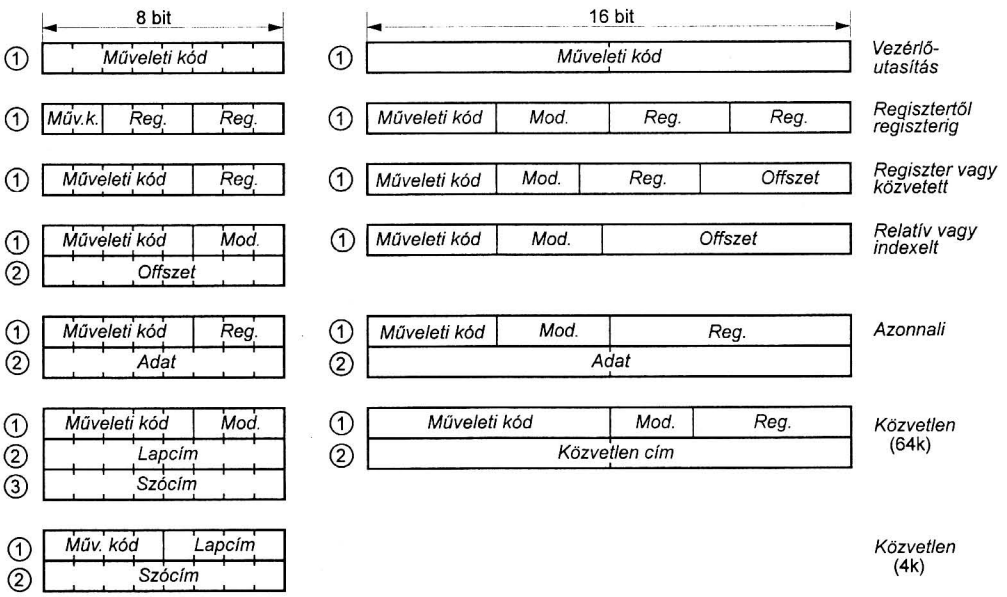
8.3.1.6. Az órajelgenerátor (Clock generator)

Egyes mikroprocesszorok órajelgenerátorral vannak ellátva. Másoknál az órajelgenerátor különálló egység. Az órajel frekvenciáját a legtöbb esetben egy kvarckristály adja meg. Amikor a frekvencia-stabilitási igények nem túl nagyok, akkor megfelel egy egyszerű RC áramkör is.

8.3.2. A mikroprocesszorok utasításai

Az utasítások (*Instruction*) a programmemóriában az adatokhoz hasonlóan bináris formában vannak tárolva. A szoftvertervezésnél az ***emlékeztető szimbólumokkal*** (*Mnemonics*) való jelölésüket használják. Ezek az utasítások angol nyelvű elnevezésének megfelelő rövidítései. A programmemóriába való beírásának megkönnyítése végett az utasítások hexadecimális ábrázolását használják (amint láttuk, a hexadecimális karakterekkel ábrázolt számot könnyen át lehet alakítani binárissá).

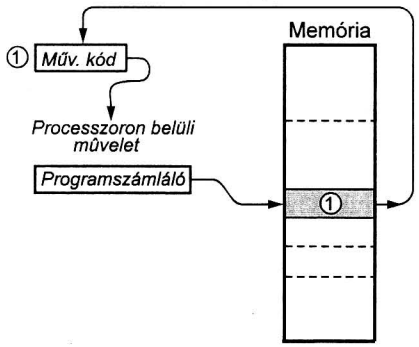
Az utasítások hossza változó: egyszavas, kétszavas és háromszavas utasításokat különböztethetünk meg (*8.11. ábra*). Legfontosabb részük a ***műveleti kód*** (Op code). Általában az utasítás tartalmazza az operandusok címeit is. Az operandusok vagy a mikroprocesszor regisztertömbjében, vagy a memóriában, vagy a beviteli áramkörben találhatók. Az utasítás formátumát a mikroprocesszor architektúrája határozza meg, amely lehet regiszter-, illetve memória-orientált. A regiszterorientált architektúrájú mikroprocesszorok utasításai a regiszterek megcímzését, míg a memóriaorientált architektúrájúaké a memória megcímzését teszik lehetővé.



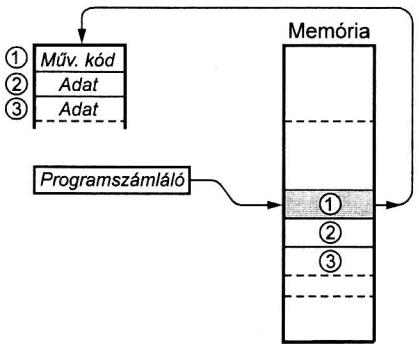
8.11. ábra. Tipikus 8-bites és 16-bites utasításformátumok

Az alábbiakban az ismertebb címzési módszerek kerülnek bemutatásra (a vonatkoztatások a 8-bites mikroprocesszorokra érvényesek, de kis eltéréssel kiterjeszthetők a 16 vagy 32-bitesekre is).

Az utasításban foglalt címzés (*Inherent-, Implied addressing*) esetében az utasításból egyértelműen következnek az operandusok. Ezt a típusú címzést használó utasítások egybájtosak (*8.12. ábra*). Általában egy processzoron belüli műveletet hajtanak végre. Ezért a hatékonyságuk a regiszterorientált mikroprocesszoroknál nyilvánul meg.



8.12. ábra. Utasításban foglalt címzés



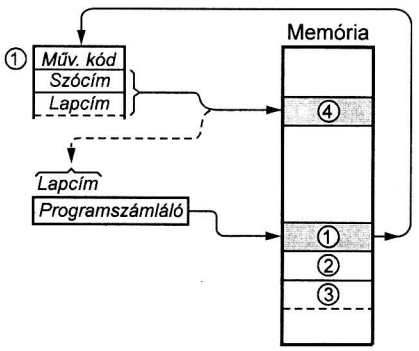
8.13. ábra. Közvetlen beírás

A közvetlen beírás (*Immediate addressing*). Az ezt a típusú címzést használó utasítások két- vagy- hárombájtosak (*8.13. ábra*). A műveleti kód után következő egy- vagy két bájt az operandust képviselő egy-, illetve kétbájtos adat. A ROM típusú programmemóriájú mikroszámítógépeknél ez az adat csak egy fix érték lehet.

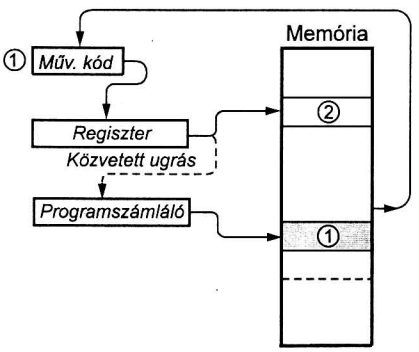
A közvetlen címzés (*Direct addressing*). A közvetlen címzést használó utasítások a memóriát közvetlenül a műveleti kód vagy az utána következő bájt, illetve bájtok segítségével címezik meg (*8.14. ábra*). Általában a második bájt az adat címének alsó helyértékű bájtja (szócíme), a harmadik pedig a felső helyértékű bájtja (lapcím). Egyes mikroprocesszoroknál, ha az adat és az utasítás azonos lapon vannak, nem szükséges az adat lapcímét is megadni. Ez a rövidített közvetlen címzés.

A közvetett címzés (*Indirect addressing*). Ezt a címzést használó utasítások a memóriát közvetetten, egy regiszter vagy egy memória-rekesz segítségével címezik meg (*8.15. ábra*). A regisztert, illetve a memóriarekeszt, amely az adat címét tartalmazza, az egybájtos műveleti kód jelöli ki. Az *azonnali címzés* a közvetett címzés sajátos esete, amelynél az adat címét tartalmazó regiszter a programszámláló. Egy különleges közvetett címzés az, amelynél a regiszter (vagy a memória-rekesz) tartalmát a programszámláló veszi át. Ez egy közvetett ugróutasítás.

A közvetett címzést használó utasítások nagyon hatékonyak. Egy egybájtos utasítással a mikroszámítógép teljes operatív memóriája megcímezhető.



8.14. ábra. Közvetlen címzés

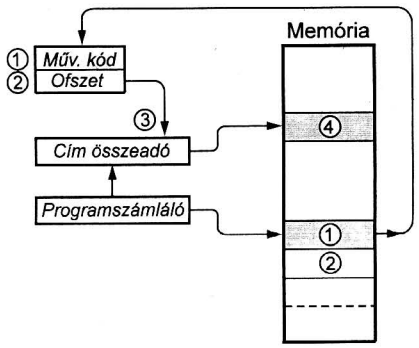


8.15. ábra. Közvetett címzés

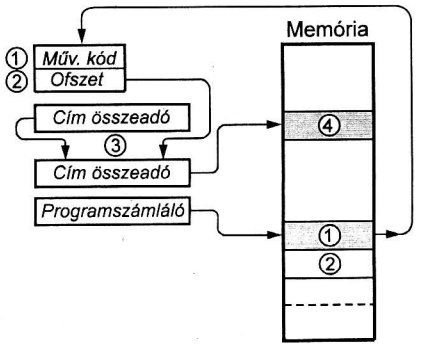
A relatív címzés (*Relative addressing*) esetében az adatot tartalmazó memóriarekesz címét a mikroprocesszor számítás után kapja meg úgy, hogy egy referenciacímhez hozzáadja az ofszet-, vagy eltoláscímet. A referenciacím származhat a programszámlálóból, egy címregiszterből vagy egy memóriarekeszből. Ha a programszámlálóból származik, akkor ***program-relatív címzésről*** beszélünk (*8.16. ábra*). Ez nagyon előnyös az *áthelyezhető programok* (Relocatable Program) szerkesztésénél, amelyek az ugróutasítások által megadott címek megváltoztatása nélkül a programmemória bármelyik részébe áthelyezhetők.

Ha a referenciacím egy címregiszterből vagy egy memóriarekeszből származik, akkor ***bázis-relatív címzésről*** beszélünk. Ennek a fontossága az adatkezelés esetében nyilvánul még. Segítségével áthelyezhető adatkezelő programok szerkeszthetők. Adattárolás céljából fenntartott bizonyos memóriamező az operatív memória akármelyik részébe áthelyezhető. Ezt ***dinamikus memóriafelosztásnak*** (Dynamic Memory Allocation) nevezik.

A relatív címzést használó utasítások végrehajtási idejének csökkentése végett a címet az adatkezeléssel elfoglalt aritmetikai és logikai egységtől különálló címösszeadó végezheti el.



8.16. ábra. Program-relatív címzés

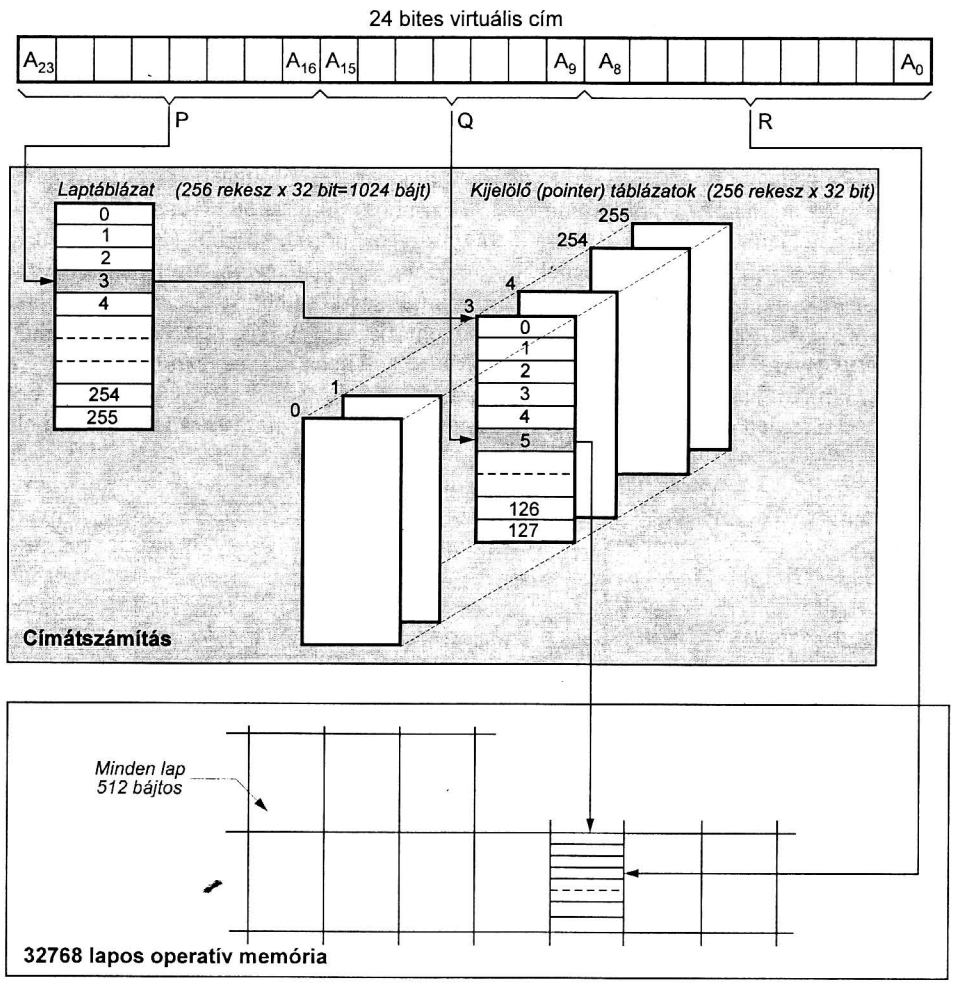


8.17. ábra. Indexelt címzés

Az indexelt címzés (*Indexed addressing*) (*8.17. ábra*) a bázisrelatív címzés egy változata. A bázis-relatív címzésnél a referenciacím fix, míg az indexelt címzés esetében a referenciacím növelhető vagy csökkenthető. A referenciacímet tartalmazó regisztert ***indexregiszternek*** nevezik.

A virtuális címzés (*Virtual addressing*) a nagyobb teljesítményű, 16-bites és 32-bites mikroprocesszorokra jellemző. A virtuális memória és a fizikailag meglevő operatív memória azonos nagyságú lapokra van felosztva (a 16 bites mikroprocesszoroknál a lap általában nagyobb, mint a 8-biteseknél szabványos 256 bájt). Az egész virtuális memória tartalma a háttérmemóriában van tárolva. Az operatív memóriába a virtuális memória azon lapjai kerülnek, melyek az éppen futó programban közvetlenül szükségesek. A virtuális memória segítségével a felhasználó hosszabb programokat szerkeszthet anélkül, hogy figyelembe venné az operatív memória fizikai határait.

A virtuális címzést a *8.18. ábra* szemlélteti. A 24 bites címszó felső 15 bitje ($A_{23}A_{22}....A_{11}A_{10}A_9$) egy lap virtuális címét tartalmazza. A virtuális címből a valós cím kiszámítása a laptáblázatok alapján történik. A virtuális cím ***P*** része a laptáblázat 256 rekesze közül egyet jelöl ki. Ennek a rekesznek a tartalmával a 256 kijelölő táblázatokból egy választható ki.



8.18. ábra. Virtuális címzés

A virtuális cím ***Q*** része a kijelölő táblázat egyik rekeszét címezi meg. Ennek a rekesznek a tartalma a fizikailag meglevő memória egyik lapját jelöli. A virtuális cím utolsó, ***R*** része ($A_8A_7....A_2A_1A_0$) a kijelölt lap egyik rekeszét címezi meg.

A továbbiakban a jellegzetes mikroprocesszor-utasításokat ismertetjük. A különböző mikroprocesszor-utasítások emlékeztető szimbólumainak jobb megértése végett az utasítások angol nyelvű elnevezését használjuk.

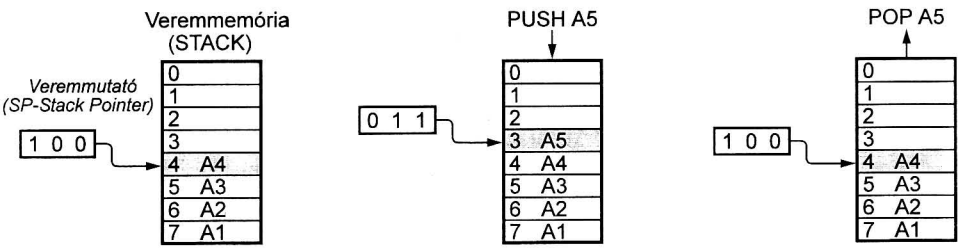
8.3.2.1. Adatmozgató utasítások

Load, Move utasítások hatására a processzor a kijelölt forrásregiszter tartalmát átmásolja a kijelölt célregiszterbe.

Store utasítás eredményeképpen a processzor a forrásregiszter tartalmát egy memória-rekeszben tárolja.

Exchange utasítás után két regiszter tartalma helyet cserél.

PUSH és **POP** (vagy **PULL**) két, speciális adatmozgató utasítás, melyek a veremmemóriába való adatbeírást, illetve kiolvasást eredményezik. Szemléltetésük a *8.19. ábrán* levő hipotetikus 8-rekeszes veremmemóriával történik. A PUSH utasítás hatására, az A5 adat beíródik a veremmutató által megcímzett veremmemória 3. rekeszébe.



8.19. ábra. A veremmemória és a veremmutató a PUSH, valamint a POP művelet után

A POP utasítás hatására a mikroprocesszor a veremmutató által megcímzett veremmemória 3. rekeszében tárolt A5 adatot kiolvassa. A POP utasítás végrehajtása után, a veremmutató a soron következő 4. rekesz címét tartalmazza.

Input utasítás egy adatbeviteli áramkör regiszterének tartalmát egy regiszterbe vagy egy memóriarekeszbe írja be.

Output utasítás egy regiszter vagy egy memóriarekesz tartalmát egy adatkiviteli áramkör regiszterébe írja be.

8.3.2.2. Aritmetikai utasítások

Add az összeadás műveletét jelzi.

Add with carry utasítás is az összeadást jelzi. Az operandusok összegéhez az ezelőtti művelet során keletkező átvitelbitet is hozzáadja.

Subtract a kivonás műveletét jelzi.

Subtract with carry utasítás is a kivonást jelzi. Az operandusok különbségéből az ezelőtti művelet során keletkező átvitelbitet is levonja.

Increment utasítás hatására egy regiszter vagy egy memória-rekesz tartalma 1-gyel növekszik.

Decrement utasítás hatására egy regiszter vagy egy memória-rekesz tartalma 1-gyel csökken.

Multiply a szorzás műveletét jelzi, és csak a nagyobb teljesítményű mikroprocesszorok utasításkészletében lelhető fel.

Divide az osztás műveletét jelzi, és fentihez hasonlóan csak a nagyobb teljesítményű mikroprocesszorok utasításkészletében lelhető fel.

Decimal Adjust a decimális helyesbítést végző utasítás. Segítségével a tiszta bináris számokkal dolgozó aritmetikai és logikai egység BCD kódolású számokkal is tud műveleteket végezni.

8.3.2.3. Logikai utasítások

And a logikai *ÉS* műveletet jelzi. Az eredmény minden egyes bitjét a két operandus megfelelő helyértékű bitjei között elvégzett *ÉS* művelet adja.

Or a logikai *VAGY* műveletet jelzi.

Exclusive – Or a logikai KIZÁRÓ-VAGY műveletet jelzi.

Complement az operandus komplemensét képezi.

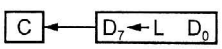
Compare az összehasonlítás utasítása. Az összehasonlítás eredményét a feltételbitek megfelelő beállítása jelzi.

Shift, Rotate a léptetési és forgatási műveleteket jelzi. Ezek az akkumulátor tartalmára és a carry átvitelbitre vonatkoznak.

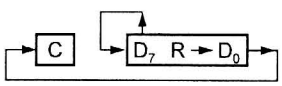
Az alábbiakban a 8-bites mikroprocesszorokra jellemző léptetési és forgatási műveleteket ismertetjük (*8.20. ábra*).

– *Aritmetikai léptetés balra* (Arithmetic shift up): az akkumulátor összes bitjei egyet lépnek balra, a D_7-bit átlép a *CY* átvitelbe és D_0-ba **0** íródik be. Ha a léptetés után CY átvitelbit és D_7-bit azonosak, akkor az akkumulátorban levő szám egy előjeles szám, amely a léptetés előttinek a kétszerese. Ha a léptetés előtti akkumulátor tartalmát előjel nélküli számnak tekintjük, akkor a $CY = \mathbf{1}$ azt jelenti, hogy a kétszeres érték túllépi az akkumulátor kapacitását.

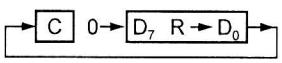
– *Aritmetikai léptetés jobbra* (Arithmetic shift down): az akkumulátor összes bitjei egyet lépnek jobbra, a D_0 bit átlép a *CY* átvitelbe, és a D_7-bit változatlan marad. A léptetés eredménye fele a léptetés előtti számnak. Ha a léptetés után $CY = \mathbf{0}$, akkor a felezés eredménye maradék nélküli. Ellenkező esetben, ha $CY = \mathbf{1}$, akkor a felezés nem maradék nélküli. Az eredmény lefelé kerekített.



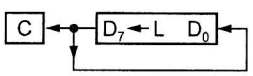
a) aritmetikai léptetés balra (szorzás 2-vel)



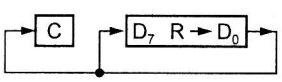
b) aritmetikai léptetés jobbra (osztás 2-vel)



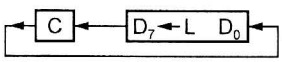
c) logikai léptetés jobbra



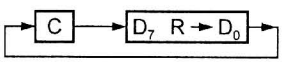
d) ciklikus léptetés (forgatás) balra



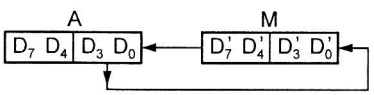
e) ciklikus léptetés (forgatás) jobbra



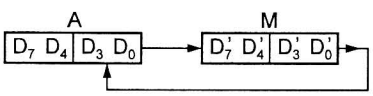
f) ciklikus léptetés az átvitelbiten keresztül balra



g) ciklikus léptetés az átvitelbiten keresztül jobbra



h) ciklikus tetrád-léptetés balra



i) ciklikus tetrád-léptetés jobbra

8.20. ábra. Léptetési műveletek

– *Logikai léptetés jobbra* (Logical shift right): ez a léptetési művelet abban különbözik az előbbitől, hogy a D_7-be **0** íródik be. A léptetés eredménye fele a léptetés előtti előjel nélküli számnak. A maradékot, az előbbihez hasonlóan, a *CY* átvitelbit tartalmazza.

– *Ciklikus léptetés (forgatás) balra* (Cyclic shift left, Rotation left): az akkumulátor összes bitjei ciklikusan lépnek egyet balra, a D_7-bit átlép a D_0-ba, és átmásolódik a *CY* átvitelbe.

– *Ciklikus léptetés (forgatás) jobbra* (Cyclic shift right, Rotation right): ez a léptetési művelet az előbbihez hasonló, csak a léptetési irány különbözik.

– *Ciklikus léptetés (forgatás) az átvitelbiten keresztül balra* (Rotation through carry left): a *CY* átvitelbitet az akkumulátor 9. bitjeként kezeli. A léptetési műveletben D_7 és D_0 között helyezkedik el. Az összes bitek egyet lépnek balra, a *CY* átvitelbit D_0-ba lép, az átvitelbitbe pedig D_7 lép át.

– *Ciklikus léptetés (forgatás) az átvitelbiten keresztül jobbra* (Rotation through carry right): ez a léptetési művelet az előbbihez hasonló, csak a léptetési irány különbözik.

– *Ciklikus tetrádléptetés balra*: ez az utasítás ciklikusan léptet három tetrádot balra. Ezek közül kettő egy memóriarekeszben van, a harmadik pedig az akkumulátorban. Az akkumulátor legnagyobb helyértékű tetrádja változatlan marad.

– *Ciklikus tetrádléptetés jobbra*: ez az utasítás hasonló az előbbihez, a BCD kódolású számokkal végzett műveleteknél van jelentősége, és csak a léptetési irány különböző.

8.3.2.4. Vezérlő utasítások

A vezérlő utasítások lehetnek *feltételesek* és *feltétel nélküliek*. A feltételes utasítás végrehajtása egy feltétel flip-flop állapotától függ.

Jump' Skip utasítások a programszámláló folyamatos növekedésében egy ugrást iktatnak be. A Jump ugróutasítás a legtöbb mikroprocesszornál feltétel nélküli, míg a Skip feltételhez kötött. Ha a feltétel teljesül, akkor a program további végrehajtása az ugróutasítással jelzett címtől folytatódik. Ellenkező esetben, ha a feltétel nem teljesül, akkor az ugrás nem következik be, a program végrehajtása a soron következő utasítással folytatódik tovább.

Call Subroutine' Jump Subroutine szubrutinhívási utasítások. Ezek is lehetnek feltételesek és feltétel nélküliek. A programszámláló tartalmát, amely a soron következő utasítás címe, a veremmemória a szubrutinra való ugrás előtt elmenti. Ezután a programszámlálóba a szubrutin kezdő címe íródik be.

Return a szubrutinból való visszatérési utasítás. Ez is lehet feltételes vagy feltétel nélküli. A programszámlálóba a szubrutinra való ugrás előtti, a veremmemóriába elmentett tartalom íródik be.

A Halt utasítás megállítja a programszámlálót, és ennek következtében a program további végrehajtását is. A megállítás addig tart, amíg egy külső jel hatására a mikroprocesszor kikerül a *Halt* állapotból, és újraindítja a programszámlálót.

8.3.3. A megszakítás

Miközben a mikroprocesszor a főprogram végrehajtásával van elfoglalva, előállhat egy fontosabb feladat megoldása, amelynek eredménye történetesen kihathat a főprogramra is. Ebben az esetben a főprogramot meg kell szakítani, hogy a mikroprocesszor áttérjen arra a programra, amely az éppen előállt feladat megoldására szolgál.

A mikroprocesszor kap egy megszakításkérő jelet (*Interrupt Request*). A megszakítást a megszakítás-elismerő jellel nyugtázza (*Interrupt Acknowledge*). A megszakítás idejére a programszámláló valamint a belső regiszterek tartalmát elmenti a veremmemóriába.

Bonyolult rendszer esetében a mikroprocesszor több helyről is kaphat megszakításkérést. A megszakítási módszer lehet ***lekérdező*** (*Polled Interrupt*) vagy **vektorizált** (Vectorized Interrupt). Az első esetben a mikroprocesszor a perifériás áramkörök megszakításkérő kimeneteit sorra lekérdezi. Ha valamelyikben megjelenik a megszakításkérő jel, akkor végrehajtja a megfelelő megszakítási programot, és továbblép. A megszakítás elsőbbségét a periferikus áramkörnek a lekérdezési szekvenciában levő relatív helyzete határozza meg. A vektorizált megszakítás hatékonyabb, mint a lekérdezéses. Ebben az esetben a mikroprocesszor válaszként a megszakításkérésre az adatbuszon egy periféria-azonosító címet küld ki. Ez az ún. *vektor*. A megszakításkérő periféria azonosítása után elvégzi a megfelelő megszakítási programot.

8.3.4. Jellegzetes mikroprocesszor típusok

A jelenleg gyártott sok különböző mikroprocesszor közül két nagy családot emelhetünk ki. Az egyik a Motorola 6800-as típusára épül, a másik az Intel 8080-as áramkörére.

A *8.1. táblázat* – a teljesség igénye nélkül – a 6800-as család típusait foglalja össze. A 6802-es típus a 6800as továbbfejlesztése. A 6802-es utasításkészlete azonos, ezért a programok változtatás nélkül csereszabatosak. A 6802-es típust a gyártók órajelgenerátorral és 128 byte RAM-mal egészítették ki.

A 6809-esnek két címregisztere van. Utasításkészlete különösen indexelt címzésnél jelentősen kibővült.

A 68000-es típusok teljesen új generációhoz tartoznak. Minden programsíkon azonos utasításkészletük van, ami a 68010 és 68020 típusoknál csupán néhány kiegészítéssel bővül. A 68000-es típusok belül 32 bites mikroprocesszorként épülnek fel. A 68000 és 68010 esetén a külső adatbusz 16 bit széles. A 68008-nál a buszillesztő 8 bites adatbuszhoz illeszkedik. Ezért számottevő hardver-változtatás nélkül 8 bites rendszerben is működhet. A 68000 és 68010, 24 bit hosszú címbuszával 2^{24} byte = 16 Mbyte cím címezhető. Ezzel már nagyobb folyamatirányító számítógép teljesítőképességének nagyságrendjét érjük el. Utasításkészletének különleges jellegzetessége a viszonylag kevés utasítás kombinációja sok címzésmóddal. Tehát nagyon egyszerű és mégis hatékony programozásra alkalmas.

A 68010 és 68020 típusok virtuális táras üzemmódban is működtethetők (*Virtual Memory, VM*). Ebben az üzemmódban a felhasználó úgy fordulhat a háttértár adataihoz (pl. merevlemezről vagy floppy lemezről), mintha azok közvetlenül a RAM-ban lennének tárolva. A háttértár kezelését az operációs rendszer teljesen automatikusan végzi. A program futása közben előfordulhat, hogy olyan operandus kell, amelyik nem a RAM-ban, hanem külső háttértárban van. Ebben az esetben az utasítás végrehajtása félbeszakad, az operációs rendszer azt az adatszegmenst, melyben az operandus van, áttölti a háttértárból a RAM-ba, és ezután a megszakított utasítás végrehajtása is befejeződik.

A 68020-as típus 32 bit szóhosszú belső adatbusza kívülről is hozzáférhető. Ez kétszeres adatátviteli sebességet eredményez a 16 bit szóhosszal működő típusokhoz képest olyan esetekben, amikor az utasítások és az operandusok szóhossza nagyobb 16 bitnél.

A 68020-as típus ezenkívül „aritmetikai társprocesszor interfésszel" (*Coprocesszor Interface*) is rendelkezik, mellyel speciális számítógépekkel párhuzamosan képes működni, ilyen pl. a 68881 aritmetikai processzor (numeric data processor). A két processzor az utasításkód alapján felismeri, hogy melyiknek kell az utasítást végrehajtani.

Típus	Tipikus teljesítmény-felvétel (mW)	Adatbusz szóhossz bit	Cím-tartomány, byte	Adat-regiszter, bit	Cím-regiszter, bit	Utasítások száma	Órajel frekven-ciája, (MHz)	Különleges tulajdonságok
6800	600	8	64 k	2 × 8	3 × 16	72	1 ÷ 2	
6802	600	8	64 k	2 × 8	3 × 16	72	1 ÷ 2	128 byte RAM
6809	650	8	64 k	2 × 8	5 × 16	59	1 ÷ 2	
68000	1200	16	16 M	2 × 32	10 × 32	56	6 ÷ 12	
68008	1200	8	1 M	2 × 32	10 × 32	56	6 ÷ 12	
68010	1500	16	16 M	2 × 32	11 × 32	58	6 ÷ 12	VM
68020	1500	32	4 G	2 × 32	11 × 32	92	12 ÷ 16	VM, CP, CA

8.1. táblázat. A 6800-as mikroprocesszor-család áttekintése

Típus	Tipikus teljesítmény-felvétel (mW)	Adatbusz szóhossz bit	Cím-tartomány, byte	Adat-regiszter, bit	Cím-regiszter, bit	Utasítások száma	Órajel frekven-ciája, (MHz)	Különleges tulajdonságok
8080	780	8	64 k	8 × 8	6 × 8	78	2 ÷ 3	
8085	650	8	64 k	8 × 8	2 × 16	80	3 ÷ 5	
8086	1200	16	1 M	4 × 16	6 × 8	104	5 ÷ 10	CP
8088	1200	8	1 M	4 × 16	9 × 16	104	5 ÷ 8	CP
80186	1500	16	1 M	4 × 16	9 × 16	114	8 ÷ 10	CP, T, DMA, IRC
80188	1500	8	1 M	4 × 16	9 × 16	114	8 ÷ 10	CP, T, DMA, IRC
80286	2000	16	16 M	4 × 16	9 × 16	129	8 ÷ 10	CP, MMU, VM
80386		32	4 G	4 × 32	9 × 32		16 ÷ 40	CP, MMU, VM
80486		32	16 G				16 ÷ 133	ACP, MMU, VM

CP=koprocesszor interfész; T=időzítő (timer); DMA=közvetlen tárhozzáférés (Direct Memory Access); IRC=megszakításvezérlő (Interrupt Controller); MMU=tárkezelőegység (Memory management unit); VM=virtuális tár (Virtual memory); CA=cache memória; ACP=beépített aritmetikai társprocesszor

8.2. táblázat. A 8080-as mikroprocesszor-család áttekintése

A 8080-as családról a *8.2. táblázat* ad áttekintést. A 8080A alaptípus még egy régebbi technológiával készült, melynél az áramkör még három tápfeszültséget igényelt. Ezt a típust teljesen kiszorította utódja, a 8085-ös típus amely – két utasítástól eltekintve – azonos utasításkészlettel és azonos gépi kóddal működik. A 8085-ös processzor hátránya, hogy bizonyos utasítások hiánya miatt aritmetikai teljesítménye igen korlátozott.

A 8086-os típus adatbusza és aritmetikája 16 bites. Az utóbbi a szorzást és osztást is tartalmazza. A 8088 utasításkészlete azonos, azonban adatbusza 8 bites. A 80186 esetén a 8086-hoz képest néhány vezérlő funkciót is integráltak, amit különösen nagyobb rendszerekben a felhasználónak kellene csatlakoztatni. Tartalmaz továbbá egy időzítőt (*timer*), egy DMA-vezérlőt (közvetlen tárhozzáférés vezérlőt) a gyors adatátvitelhez és egy megszakításvezérlőt több megszakítás (*interrupt*) feldolgozásához.

A 80286 a 8086 továbbfejlesztett változatának tekinthető. Hasonlóan a 68000-es családhoz különleges operációs rendszerutasításokkal működhet. Ezenkívül a 80286-ot ellátták tárkezelő-egységgel (angolul: memory management unit), és a 68020-hoz hasonlóan virtuális tárkezelésre is alkalmas. A 8086-os processzor család minden típusa aritmetikai társprocesszorral is képes együttműködni, mint pl. a 8087 aritmetikai vagy a 8099 ki/beviteli (I/O) processzorral.

A 80386-nál a cím- és adatbusz szóhossza 32 bit, amely a 80286 továbbfejlesztése. Ez megfelel a 68020-nak és majdnem eléri a hatékonyságát is.

A 80486 esetén az aritmetikai társprocesszor nem külső áramkörként csatlakozik a rendszerhez, hanem ugyanazon a chip-en van integrálva. A 80486-os áramkör a 80386 továbbfejlesztésének tekinthető.

A korszerű mikroprocesszorok teljesítményfelvétele nagyságrendekkel kisebb, mint a korábbi TTL technikával készült processzoroké. Ennek ellenére telepes működés számára mégis túl nagy a teljesítményfelvétel. Ezért nagyon fontos, hogy a legtöbb egyszerű mikroprocesszor hardver- és szoftverkompatibilis CMOS változatban is készül, amelyekből néhányat a *8.3. táblázat* foglal össze.

NMOS típusok	CMOS típusok	Teljesítmény-felvétel (mW)	Az órajel frekvenciája (MHz)	Gyártók
6802	MD 68SC02	75	5	Mitel
6809	HD 6309	80	5	Hitachi
8085A	MSM 80C85	50	3	Oki
8085	HD 80C86	150	8	Harris
8088	HD 80C88	150	8	Harris

8.3. táblázat. A 6800-as és a 8080-as család CMOS mikroprocesszorai

A CMOS technológiával készült processzoroknak belső statikus táruk van. Ezért megengedett, hogy órajel frekvenciáját tetszőlegesen csökkentsük. Mivel az áramfelvétel arányos a frekvenciával, ezért kisebb frekvenciájú órajellel áramot takaríthatunk meg. Az órajel frekvenciája akkora kell legyen, hogy a működési sebesség még éppen kielégítse a követelményeket.

8.5.1. Programozható logikai vezérlők (PLC)

A mikroszámítógépek másik fontos ipari alkalmazása a technológiai folyamatok vezérlése. Az üzemben levő minden fontosabb berendezés működését egy kisebb teljesítményű specializált mikroszámítógép irányítja. Ennek egyik fontos feladata, hogy a feldolgozott adatok egy részét közölje a központi folyamatirányító számítógéppel. Ez a nagyteljesítményű számítógép csak bizonyos helyzetekben lép közbe.

A tárprogramozott vezérlőberendezések olyan számítógépes készülékek, amelyeket programozható ipari vezérlési feladatokra fejlesztettek ki. Teljesítőképességüket alapvetően a szoftver határozza meg.

A tárprogramozott vezérlések feldolgozó egységeként PLC-t (**P**rogramable **L**ogic **C**ontroller = programozható logikai vezérlőt) mikroszámítógépet vagy folyamatterminált alkalmaznak. A PLC-k olyan elektronikus üzemi eszközök, amelyek egy alkalmazásorientált nyelven programozhatók és képesek ellátni vezérlési és szabályozási feladatokat is.

A programozható vezérlőlogikát a PLC programtárolójában, mint vezérlőutasításokat lehet megadni. Ezek felváltják a kapcsolatprogramozott vezérléseknél használt logikai elemeket (pl. *ÉS, VAGY*). A vezérlőutasításokat a PLC értelmező szoftvere (interpretere) dolgozza fel. A programozható logikai vezérlők utasításkészlete a következő vezérlőutasításokat mindenképpen tartalmazza:

- logikai kapcsolatok (pl. *ÉS, VAGY*),
- számlálás (pl. impulzusszám meghatározása),
- késleltetés (pl. egy jelet csak egy beállított idő elteltével ad ki),
- tárolás (pl. egy kimenet vagy bemenet logikai értékének tárolása).

A programozható logikai vezérlők megvalósítása lehet *moduláris* vagy *kompakt*. A moduláris PLC-k az adott üzemi követelményekhez illeszthetők. Az egyes modulok bővíthetők vagy kicserélhetők. A kompakt kiépítésű PLC-k olyan vezérlési feladatokhoz alkalmasak, amelyeknél előreláthatólag a be- és kimenetek száma lényegesen nem változik a későbbiek folyamán.

A vezérlőlogikát három különböző módon határozhatjuk meg (vagyis az eszköz programozására három nyelvet fejlesztettek ki):

- utasításlista (**St**atement **l**ist – STL),
- létradiagram (**La**dder **d**iagram – LAD),
- funkcionális terv (**C**ontrol **s**ystem **f**lowchart – CSF)

Az egyes gyártók különböző eszközöket bocsátanak rendelkezésre a PLC-hez a vezérlőprogram bevitelére. Ezek lehetővé teszik a vezérlőprogram megadásának legalább az egyik módját a három közül. A jelenleg legelterjedtebb eszköz a programozás és programfejlesztés megvalósítására a megfelelő szoftverrel ellátott számítógép (PC – Personal Computer). A gyártók gyakran használnak egyedi vezérlőutasításokat a programok (utasításlisták, létradiagramok vagy funkcionális tervek) beviteléhez.

A *8.31. ábra* egy PLC tipikus felépítését mutatja be. A főbb részegységek a következők:

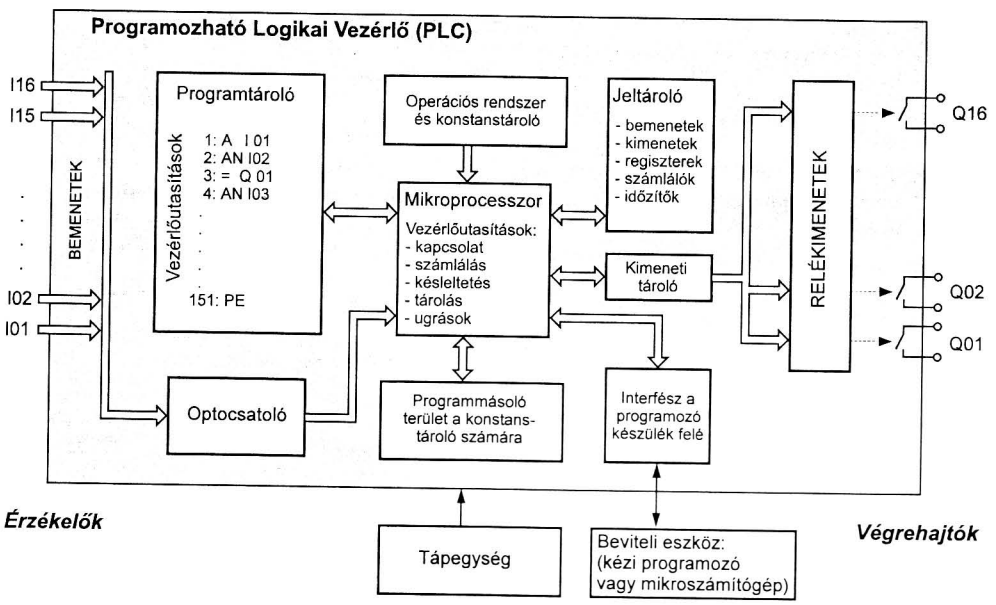
- feldolgozóegység:
  - a mikroprocesszorral, a konstanstárolóval, a jeltárolóval, a programtárolóval;
- bemeneti interfész;
- kimeneti interfész;
- kimeneti interfész a programozókészülék felé;
- hálózati csatlakozás.

A bemenetek – galvanikus leválasztást biztosító – optocsatolón keresztül kapcsolódnak a mikroprocesszor bemeneti interfészéhez. Ezen keresztül valósul meg az érzékelők jelszintjének illesztése a PLC üzemi feszültségéhez. A relékimenetek látják el ebben az esetben a teljesítmény interfész feladatát. Meg kell jegyezni, hogy a PLC-k kimeneti- és bemeneti interfésze – kiépítéstől függően – képes lehet analóg és digitális jelek kibocsátására ill. fogadására is. Ebben az esetben az analóg bemeneti jelek feldolgozását A/D átalakítók végzik. A mikroprocesszor digitális vezérlőjeleit D/A átalakítók alakítják át analóg kimenő jellé.

Az utasításlistát (vezérlőlogikát) a beviteli egység (más néven programozókészülék) az interfészen keresztül juttatja el a PLC programtárolójába. A konstanstárolóban levő speciális operációs rendszer vezérli úgy a mikroprocesszort, hogy a kapott adatok a programtárolóba kerüljenek. Ez az operációs rendszer döntően meghatározza a PLC teljesítőképességét.

A vezérlőprogram indításakor a bemenetek aktuális logikai állapota eltárolásra kerül (ezeket az értékeket veszi figyelembe a vezérlőlogika a program végrehajtása során) és az utasításszámláló az első utasításssorra áll (az ábrán az 1 sor). A mikroprocesszor kiolvassa az első utasítássort a programtárolóból és az operációs rendszer segítségével, végrehajtja azt. Az "**A I01**" parancs hatására az **I01** bemenet – előzőleg már letárolt – logikai állapota betöltődik, és a jeltárolóba kerül. Ezt követi a következő utasítássor végrehajtása ("**AN I02**"). Az **I02** bemenet logikai állapota is a jeltárolóba kerül. A második sorban lévő "**A**" utasítás hatására az operációs rendszer a két bemenet eltárolt állapotát logikai *ÉS*-kapcsolatba hozza.

Az **I01** és **I02** bemenet *ÉS*-kapcsolatának eredményét (amely logikai **1** vagy **0** értéket vehet fel) szintén eltárolja a mikroprocesszor. Ezután a következő utasítássor kerül végrehajtásra ("**= Q01**"). Az utasítás a logikai *ÉS*-kapcsolat eredményét a **Q01** kimenethez rendeli hozzá. Az eredmény betöltődik a jeltárolóba. Közben a kimeneti tároló kiadja a jelet (logikai 0 vagy 1) a **Q01** kimenet számára. Ha az **I01** és **I02** bemenet logikai **1**-es szintet kap az érzékelőktől az **Q01** kimeneten levő relé behúz, vagyis a terhelési oldal áramköre záródik. Ezzel a programtárolóban levő első "logikai mondat" végrehajtása befejeződik.



8.31. ábra. A PLC alapvető felépítése

A mikroprocesszor folytatja a negyedik, ötödik, stb. sorban levő utasítás végrehajtását mindaddig, amíg a program végére nem ér. A processzor számára a program végének elérését a **PE** (**P**rogram **E**nd) utasítás jelzi. Az utasításszámláló ismét a program elejére áll vissza. A PLC elkezdi ismét az utasításlista soronkénti végrehajtását. A **Q01** kimeneten addig áll fenn a logikai **1**-es jel, amíg az **I01** vagy **I02** bemenet bármelyikén logikai **1**-es szint van

Megállapítható, hogy a PLC működését a program határozza meg. A program nem más, mint utasítások sorozata, amelyeket folyamatosan egymás után (szekvenciálisan) hajt végre a mikroprocesszor.

8.5.1.1. A PLC programozása

A nemzetközi szabványok nem rögzítik a programozható logikai vezérlők utasításkészletének sem a legkisebb sem a legnagyobb terjedelmét. A különböző PLC-gyártók (pl. Siemens, AEG, Festo, Omron, Mitsubishi) utasításkészletei ezért gyakran eltérnek egymástól. Az adott PLC kézikönyvében a gyártók megadják a megfelelő programozási előírásokat.

Programozás utasításlista (STL) segítségével

Az utasításlistával történő programozás esetén a működési egyenleteket előírt formában soronként kell megadni. Az utasításlista tulajdonképpen a vezérlőutasítások egy sorozata. A *8.7. táblázatban* látható egy utasítássor felépítése. Az **01**-el és **02**-vel jelölt bemenet (**I**) *ÉS*-kapcsolata (**A**) valamint *ÉS-NEM*-kapcsolata valósul meg a példában.

A különböző programozható logikai vezérlők utasításkészlete a *8.8. táblázatban* feltüntetett parancsokat (műveleteket) mindenképpen tartalmazza. A régebbi gyártású (általában 1993 előtti) PLC-k esetében egy utasításlista sornak mindenképpen az **L** (töltés) paranccsal kell kezdődnie és a **PE** (program vége) paranccsal kell végződnie. Az újabb fejlesztésű PLC-k esetében az első sorban levő **A** (*ÉS*) utasítás automatikus programbetöltést eredményez.

Műveleti rész	operandus rész	
	operandus jel	*paraméter*
A	I	01
A	I	02
AN	I	01
AN	I	02

8.7. táblázat. Egy utasításlista-sor formai megjelenése

Jelentés	Műveleti rész	Megjegyzés
töltés	L	gyártóspecifikus
ÉS	A	
NEM-ÉS	AN	
ÉS nyitózárójel	A(	
VAGY	O	
NEM-VAGY	ON	
VAGY nyitózárójel	O(	
Záró zárójel	)	
Záró zárójel NEM	)N	
Értékadás	=	
Értékadás NEM	=N	
Idő-bemenet	=T	gyártóspecifikus
Beírás	S vagy SL	gyártóspecifikus
Törlés	R vagy RL	gyártóspecifikus
Üres művelet	NOP vagy NO	gyártóspecifikus
Program vége	PE	gyártóspecifikus

8.8. táblázat. A műveleti rész és annak jelentése

A PLC gyártók a dokumentációban meghatározzák, hogy melyik parancs szükséges az indításhoz.

A be- és kimenetek mellett a műveletek operandusaként szóba jöhetnek a különböző regiszterek, számlálók és időzítők.

A ***regiszterek*** itt 1-bites tárolók, amelyeket arra használnak, hogy a logikai kapcsolatok kiértékelésének eredményeit eltárolják a következő utasítások számára.

Időzítők felhasználásával a folyamatirányító berendezés időterv-vezérlésekre is alkalmassá válik. Ha a beállított idő letelt, az időzítő logikai **1**-es jelet ad ki.

Számlálók felhasználásával különböző impulzusok számlálása valósítható meg. Ha a számláló bemenetére kerülő impulzusszám túllépi a beállított értéket a számláló kimenetén logikai **1**-es jel jelenik meg.

Jelentés	Operandus-jelölések
bemenet	I (pl. I01-tól I16-ig)
kimenet	Q (pl Q01-től Q16-ig)
regiszter	M (M01-től M256-ig)
időzítő	T (T01-től T1024-ig)
számlálló	C (C01-tól C256-ig)

8.9. táblázat. Operandusjelölések és azok jelentései

Ahogy a *8.7. táblázatban* is megfigyelhető volt, minden operandus-jelet egy megfelelő paraméternek kell meghatároznia. A paraméter adja meg a bemenet, a kimenet, a regiszter vagy az időzítő sorszámát. A rendelkezésre álló paraméterek számát a PLC típusa és kiépítettsége (pl. a bemeneti- és kimeneti modulok száma) határozza meg. A *8.9. táblázat* az operandusok tipikus jelölését és jelentését szemlélteti.

Programozás létradiagram (LD) segítségével

A létradiagram szerkezete egy adott vezérlés áram-út tervének elvét követi. Az áramút-tervtől abban különbözik, hogy az áramutakat vízszintesen rendezi, és más szimbólumokat használ, amelyek a számítógépes ábrázolásban könnyen és szemléletesen megvalósíthatók. Ezzel a létradiagramokat egyszerűen lehet megjeleníteni a képernyőn vagy a nyomtatón. A *8.10. táblázat* a létradiagramok tipikus szimbólumait mutatja be.

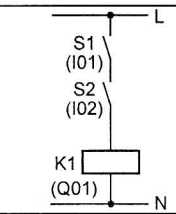
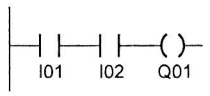
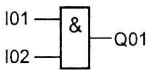
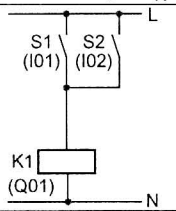
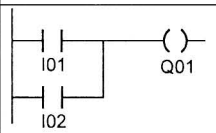
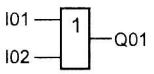
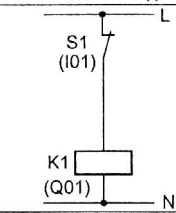
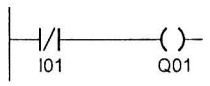
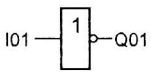
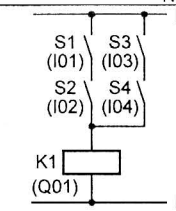
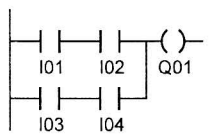
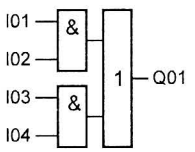
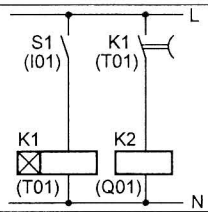
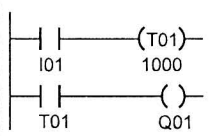
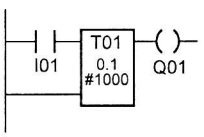
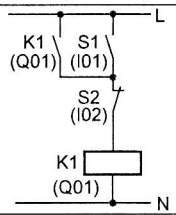
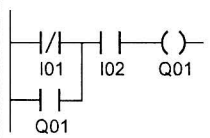
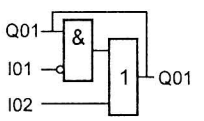
A *funkcionális terv* alapjaiban a logikai tervvel egyezik meg. Ezzel terjedelmi okokból részletesen itt nem foglalkozunk.

A programozható logikai vezérlők tehát utasításlistával, létradiagrammal vagy funkcionális tervvel programozhatók. A három programozási mód tulajdonképpen ugyanannak a vezérlő-programnak a különböző meghatározási módja.

Művelet	***Létradiagram-szimbólumok***	***Jelentés***
L, A, O	—| |—	Alapállapotban nyitott (NO) érintkező (bemenet)
AN, ON	—|/|—	Alapállapotban zárt (NC) érintkező (bemenet)
Q, M, T	—| |—	Alapállapotban nyitott (NO) érintkező (bemenet)
S vagy SL	—|S|—	Beíró bemenet
R vagy RL	—|R|—	Törlő bemenet
=	—( )—	Kimenet

8.10. táblázat. Létradiagramok szimbólumai

A *8.11. táblázat* az *ÉS*, *VAGY* és *NEM* logikai műveletek és további példák különböző technológiai megvalósítását mutatja be.

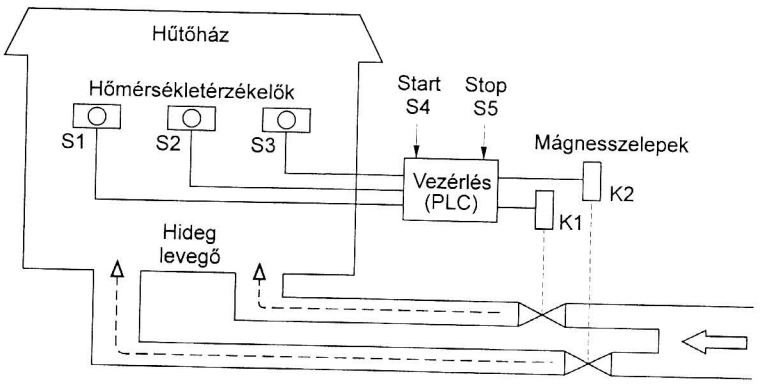
Jelölés Működési egyenlet	Elektromechanikus megvalósítás	PLC		
		STL	LAD	CSF
ÉS $I01 * I02 = Q01$		A I01 A I02 = Q01		
VAGY $I01 + I02 = Q01$		A I01 O I02 = Q01		
NEM $\overline{I01} = Q01$		AN I01 = Q01		
Programozás zárójelekkel $(I01 * I02) +$ $+ (I01 * I02) = Q01$		A(A I01 A I02) O(A I03 A I04) = Q01		
Késleltetett bekapcsolás (a késleltetés 1000*100 ms)		A I01 = T01 AN T01 = Q01		
Egy kimenet öntartása		A I01 O Q01 AN I02 = Q01		

8.11. táblázat. Tipikus programozási modulok

Megoldott feladat:

Egy hűtőházban (*8.32. ábra*) a **K1** mágnesszelep csak akkor engedi a hideg levegő bevezetését, ha az **S1**, **S2**, **S3** hőmérsékletérzékelők közül legalább kettő a beállított hőmérsékleti határérték túllépését jelzi. Ha a három érzékelő együtt jelez, akkor egy második **K2** mágnesszelepen további hideg levegő áramolhat be.

A vezérlés öntartó bekapcsolását az **S4** kézi nyomógomb (záró), kikapcsolását pedig az **S5** kézi nyomógomb (bontó) kell, hogy biztosítsa. A hűtőház vezérlését programozható logikai vezérlő (*PLC*) alkalmazásával szeretnénk biztosítani.



8.32. ábra. A hűtőház vezérlésének egyszerűsített tömbvázlata

a) Tervezze meg a vezérlés binárisan kódolt vezérlőtábláját, bekapcsolt állapotot feltételezve!

b) Rajzolja meg - az a) pontban kapott vezérlőtáblát felhasználva - a megfelelő logikai tervet!

c) Készítse el a vezérlés hozzárendelési listáját! A programozható logikai vezérlő felhasználható elemei és jelölései a hozzárendelési lista készítésénél a következők:

- I 01 ... I 16 bemenetek;
- Q 01 ... Q 16 kimenetek;
- M 01 ... M 16 regiszterek.

d) Valósítsa meg a vezérlés működését biztosító vezérlőprogramot létradiagramban!

e) Készítse el a vezérlés működését biztosító vezérlőprogramot utasításlistás formában!

Megoldás:

a) A vezérlés vezérlőtábláját a *8.12. táblázat* szemlélteti:

S1	S2	S3	K1	K2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	**1**	0
1	0	0	0	0
1	0	1	**1**	0
1	1	0	**1**	0
1	1	1	**1**	**1**

8.12. táblázat. Vezérlőtábla

b) A K1 logikai függvény Karnaugh-tábláját (a megfelelő tömbösítéssel) a *8.33. ábra* szemlélteti.

S1 \ S2S3	00	01	11	10
0	0	0	1	0
1	0	1	1	1

8.33. ábra. Karnaugh-tábla

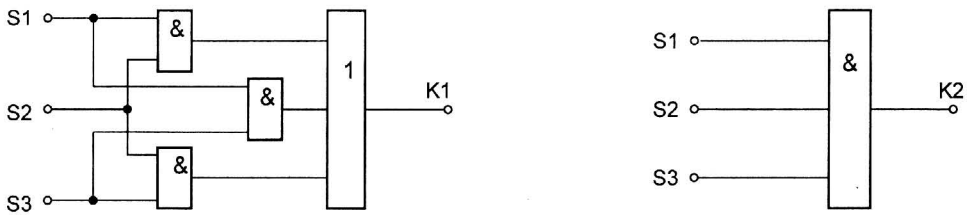
Az egyszerűsített függvény, vagyis a K1 vezérlését biztosító működési egyenlet:

$$K1 = S2 \cdot S3 + S1 \cdot S3 + S1 \cdot S2 .$$

A K2 működési egyenlete:

$$K2 = S1 \cdot S2 \cdot S3 .$$

A vezérlés logikai tervét a *8.34. ábra* mutatja.



8.34. ábra. A vezérlés logikai terve

c) A hozzárendelési listákat a *8.35. ábra* mutatja.

Érzékelő	S1	S2	S3	S4 (Start)	S5 (Stop)
PLC bemenet	I 01	I 02	I 03	I 04	I 05

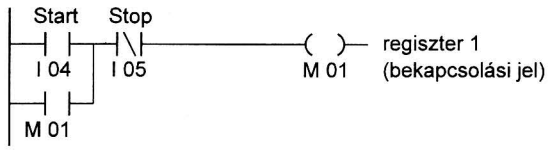
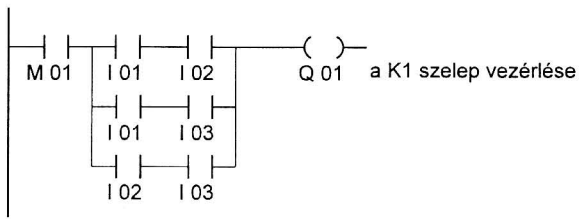
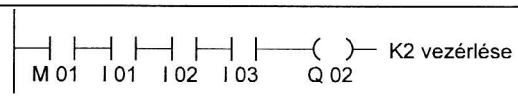
a) bemenetek hozzárendelési listája

Végrehajtószerv	K1	K2
PLC kimenet	Q 01	Q 02

b) kimenetek hozzárendelési listája

8.35. ábra. Hozzárendelési lista

d), e)
A vezérlés működését biztosító vezérlőprogramot utasításlistában és létradiagramban a *8.36. ábra* mutatja.

Utasításlista Sor Utasítás Operandus	Létradiagram
1: A I 04 2: O M 01 3: AN I 05 4: = M 01	
5: A M 01 6: A(7: A I 01 8: A I 02 9:) 10: O(11: A I 01 12: A I 03 13:) 14: O(15: A I 02 16: A I 03 17:) 18: = Q 01	
19: A M 01 20: A I 01 21: A I 02 22: A I 03 23: = Q 02	

8.36. ábra. A vezérlőprogram

Összefoglaló kérdések és feladatok:

1. Milyen alapegységei vannak a Neumann-féle univerzális számítógépnek?
2. Milyen feladatokat lát el az aritmetikai logikai egység és hogyan működik együtt a vezérlőegységgel?
3. Mi a szerepe a programszámlálónak?
4. Magyarázzuk el a veremtár elvi működését!
5. Mit nevezünk szubrutinnak?
6. Rajzolja fel egy mikroszámítógép egyszerűsített tömbvázlatát és magyarázza el a részegységek feladatát!
7. Mi a közvetlen memóriahozzáférés lényege?
8. Milyen jelző-biteket tartalmaz a feltételregiszter?
9. Milyen belső regiszterei vannak a mikroprocesszornak?
10. Milyen alapvető fázisokra tagolódik a mikroprocesszor utasításciklusa?
11. Mit nevezünk virtuális címzésnek?
12. Magyarázzuk meg a műveleti rész és az operandus rész fogalmakat?
13. Hogyan csoportosíthatók a mikroprocesszor utasításai?
14. Írjuk le egy programmegszakítás elvét!
15. Magyarázzuk meg mi történik a **MOV r1, r2** utasítás hatására!
16. Rajzolja le a szoftverfejlesztés folyamatábráját és magyarázza el a folyamatábra egyes lépéseinek lényegét!
17. Mi a különbség a gépi nyelv és magas szintű nyelv között?
18. Mi a különbség a forrásprogram és a tárgyprogram között?
19. Mit nevezünk programhuroknak és milyen részekből áll?
20. Mit nevezünk PLC-nek és milyen feladatokat tud ellátni egy PLC?
21. Milyen lehetőségek vannak egy PLC programozására?
22. Adja meg a következő vezérlőfüggvényt (logikai függvényt) utasításlistás és létradiagramos formában:

$$F^4 = A \cdot B \cdot C + \overline{B} \cdot C + A \cdot D + B \cdot \overline{D} \cdot \overline{C}$$

A PLC felhasználható bemenetei, kimenetei és lépéstárolói (regiszterei) a következők:

- bemenetek: I01 ÷ I16
- kimenetek: Q01 ÷ Q08
- regiszterek: M01 ÷ M256