SZAKDOLGOZAT

Borsi István Norbert

Debrecen

2007

Debreceni Egyetem
Informatika Kar

MIKROKONTROLLEREK AZ INFORMATIKA OKTATÁSÁBAN

Témavezető:
Szabó Zsolt
Intézeti mérnök

Készítette:
Borsi István Norbert
Informatika tanári szak

Debrecen

2007

Tartalomjegyzék

Bevezető

Témaválasztás indoklása

A dolgozat a mikrovezérlők és a középiskolai informatikai oktatás kapcsolatát tárgyalja, megpróbál rámutatni a tananyagok között fennálló kapcsolatokra. A következő néhány pontban szeretnék rávilágítani, ezen összefüggésekre.

- A számítógép (számítástechnika) összetett rendszert alkot, ezt a komplex rendszert, szerintem érthetőbbé teszi a mikrokontrolleres ismeret.
- A mikrokontrollek programozása egyszerűen elsajátítható, a kezdő assembly ismeretek elsajátítására ideális.
- A már ismert programstruktúrák (szekvencia, elágazás, ciklus) a legelemibb módon létrehozhatóak és tanulmányozhatóak.
- Az elkészített programok azonnali hardveres környezetben történő kipróbálása, nagy sikerélménnyel kísért.
- A főiskolák (egyetemek) informatikai karain már bevett tananyag, a mikrokontrolleres ismeret.
- A digitális technika világa már hétköznapi szinten is egyre inkább a programozható eszközök irányába tolódik el (pl. TV, MP3-as lejátszó), ezért ezek központi elemének ismerete alapvető fontosságú.

Célkitűzéseim

- Olyan tananyagot létrehozni, amely az informatika és digitális technika között teremt szorosabb kapcsolatot.
- Egyszerű programozható eszköz (PIC-es gyakorlópanel) készítése, amely alkalmas a mikrokontrolleres technika megismertetésére.
- Olyan gyakorlati (programozási) feladatokat előkészíteni (elkészíteni), amelyek jók a tanulók képességfejlesztésére.
- Középiskolai oktatásban használható tanmenet készítése, amely a illeszkedik a tanulók informatika (információ technológia) képzésébe.
- A mikrokontrolleres tananyagra vonatkozó tesztkérdések kidolgozása, amelyek egyszerűbbé teszik a tanulók értékelését.
- Az assembly ismereteken túl optimális lenne, egy magas szintű programozási nyelvet (C) is megtanítani, amely segítségével szintén lehetne kontroller programozást oktatni.

§ Az egész dolgozatban megpróbálom a mikrokontroller – személyi számítógép párhuzamot fenntartani.

A tananyag helye a képzési struktúrában

A mikrokontrollekhez kapcsolódó ismeretek oktatására a 12. évfolyamos informatika szakmacsoportos tanulók esetén nyílik optimális lehetőség, az információ technológia tantárgy keretein belül. Az érintett tantárgy az elmúlt években jelentős változáson ment keresztül, régebben ez a tárgy egyet jelentett az elektronika oktatásával. Manapság azonban a korszerű technológiáknak a hétköznapi életben történő begyűrűzésével (mobiltelefon, GPS) a korszerű ismeretek tanítása elkerülhetetlen. A központi programok nem elég rugalmasak, ezért helyi szinten kell a megfelelő lépéseket megtenni a korszerűsítés érdekében.

A 12. osztályban tanév 33 hétre bontható (az érettségi miatt), és a tárgy heti 4 órában kerül oktatásra. Egy osztály három egyforma létszámú csoportra van bontva, egyszerre három tanárnak adva munkát. A csoportlétszámok osztálylétszámoktól függően általában 8-10 fő körül alakul, a csoportlétszám ideális gyakorlati ill. programozási feladatok oktatására. Az információ technológia tananyag elosztásában a vezető elv az volt, hogy az informatikában használható tudást lehessen átadni a legegyszerűbb módon. Ezért az oktatás három teremben és három különböző tartalommal folyik, a csoportok forgószínpadszerűen 11 hetenként váltva egymást vándorolnak teremről-teremre. A tartalmak (modulok): mikrokontrolleres ismeretek, elektronikai alapmérések és számítógép konfiguráció (rendszer adminisztráció). Ez a fajta modulfelosztás a tanulók részére a változatosságot biztosítja, a tanárok szempontjából pedig az állandóságot tartja szem előtt.

A mikrokontrolleres tananyag tartalmi felépítése

Ismétlésszerű tartalom, az eddigi informatikai és elektronikai ismereteket kívánja felidézni, megpróbálja tisztázni a számítógépről alkotott elképzeléseket. Ilyen a processzormodell, memóriák felépítése, program struktúrák és a programfejlesztés lépései.

Új tananyag feldolgozását jelenti, a tantárgy specifikus tartalom, tehát a mikrovezérlők felépítése, assembly programozási ismeretek és a szoftver – hardver összekapcsolási lehetőségek.

A számonkérések, a kifejtős és a tételszerű számonkéréseket száműzve, tisztán tesztdolgozatra épülnek, természetesen a tesztalapú számonkérések minden sajátosságával.

A tanagyag részekre bontása

A tanagyag 11 foglalkozásra bontható az előzőkben már említett okok miatt, a ténylegesen megtartható foglalkozásszám sajnos ennél csak kevesebb lehet.

1. Foglalkozás

- Számítógép általános modellje;
- Memória áramkörök felépítése;
- Neumann-elvek;
- Neumann-elvű mikroprocesszorok működési elve.

A tananyagrész vegyesen tartalmaz régi és új ismereteket, a cél, hogy a foglalkozás végére megértsék a processzor működését.

2. Foglalkozás

- Alapvető processzor architektúrák összehasonlítása;
- Mikroszámítógép felépítése;
- Mikrovezérlő fogalma;
- Nem Neumann-elvű rendszer bemutatása, a Harvard architektúra.

A foglalkozás célja, hogy bevezesse a mikrovezérlő fogalmát, valamint hogy a tanulók megismerjék a mikrokontrollerek alapját képező Harvard architektúrát.

3. Foglalkozás

- Mikrovezérlők általános jellemzői;
- Konkrét áramkörtípus ismertetése, architektúrájának áttekintése;
- A mikrokontrollerek utasítás feldolgozásának elve;
- Pipe-line elv.

A tanulók egy konkrét mikrovezérlő típus megismerése után képesek lesznek az ilyen áramköröket tartalmazó adatlapok önálló értelmezésére, specifikációk megismerésére.

4. Foglalkozás

- Mikrovezérlő utasítások felépítése;
- Utasítások csoportosítása, áttekintése;
- Mikrovezérlők memóriaszervezése;
- Mikrovezérlők program és adatmemóriájának felépítése, jellemzői.

A tanulók megismerik egy konkrét mikrovezérlőn keresztül az assembly programozás alapjainak tekinthető utasítás mnemonikokat. Tisztában lesznek a mikrovezérlők memória

korlátaival. Fontos a számítógépes környezetben megismert programmemória és operatív memória szemlélet átörökítése mikrovezérlős környezetbe.

5. Foglalkozás

- Mikrovezérlők speciális fájlregisztereinek áttekintése;
- Indirekt címzés fogalma, használata;
- I/O portok felépítése, működése.

Az előző órákon megismert mikrokontroller – számítógép párhuzam továbbvitele, memóriaszervezés és a portok vonatkozásában. Az indirekt címzés magyarázatánál megjelenik a folyamatábra, mint általános és jól használható program leírási módszer.

6. Foglalkozás

- Számláló, időzítő egységek működése;
- A/D átalakítók működési elvének az ismerete;
- A mikrokontroller megszakítási rendszere;
- Reset folyamat lezajlása mikrovezérlős környezetben;
- Speciális mikrokontrolleres áramkörök áttekintése.

Az egyik legösszetettebb foglalkozás, bár itt a legtöbb a kapcsolódási pont a hagyományos számítógépes környezettel. Timer – rendszeróra, A/D – hangkártya, megszakítási rendszer – IRQ, reset folyamat, órajel előállítás.

7. Foglalkozás

- Első tesztdolgozat;
- Programkészítési módszerek;
- Assembly alapismeretek;
- MPASM assembler;
- MPASM direktívák (előzetes fordítási utasítások) áttekintése;
- Programkészítési ismeretek assembly nyelven.

A tanulók elsajátítják, a programfejlesztés lépései assembly nyelven. Megismerik a program fordítás folyamatát, hogyan lesz a forrásnyelvi állományból futtatható (letölthető) állomány. Az itt megszerzett ismeretek jól hasznosíthatóak majd a tanulók számára a további programozási feladatok megoldásában akár más programozási nyelveken is (C, Java).

8. Foglalkozás

- Elemi programozási feladatok megoldása assembly nyelven;
- Változó deklaráció, értékadás;
- Szekvenciális elemek;
- Feltétel nélküli vezérlésátadás;
- Elágaztató szerkezetek;
- Ciklusszervezési lehetőségek;
- Alprogramok létrehozása;
- Szoftverkésleltetés létrehozása;
- Programfejlesztés MPASM rendszerben, integrált fejlesztői környezet használata.

A tanulók megértik, azt hogy a programkészítés valódi tervet igényel, nem eseti próbálkozással kell egy programozási feladatot megoldani. A folyamatábrák segítségével egyszerűen megfogalmazhatóak az egyes programozási lépések. A tanulók megismerik, az MPLAB IDE felületét és mikrokontrollerre programot lesznek képesek írni.

9. Foglalkozás

- Programozási feladat egyszerű mintaprogram készítése: matematikai, és bit manipulációs utasítások felhasználásával;
- Program szimuláció MPASM rendszerben;
- 1 LED-es villogó programjának elkészítése, szimulációja;
- PIC-es gyakorlópanel megismerése, elérhető perifériaelemek listájával;
- Programletöltő használata.

A tanulók az eddigi ismereteiket összegezve működő programokat készítenek, ellenőrzik a programjuk működését szimulációval és valós környezetben is.

10. Foglalkozás

- Második tesztdolgozat;
- Programozási feladatok:
    - Futófény program elkészítése a PORTC alsó 6 bitjére forgató utasításokkal és ütemezéssel;
    - Jelzőlámpa modell programozása ütemezéssel;
    - 1 LED-es villogó megszakításkezeléssel.

Az elemi programozási feladatok után egyszerűbb programok készítése a feladat. A programok írását megkönnyíti, a tervezést segítő folyamatábrák használata.

11. Foglalkozás

- 7 szegmenses kijelző kezelésének alapjai a táblakezelés és a multiplex kijelzés;
- Nyomógomb kezelés, egyszerű szintvizsgálattal;
- Csoport értékelése, összefoglalás az elvégzett munka áttekintése.

Az elemi programozási feladatok után összetettebb programok készítése a feladat. A tanulók megértik a 7 szegmenses kijelző működését. Képesek lesznek kezelni a nyomógombot, mint bemeneti perifériát.

A csoport munkája lezárul, az elvégzett munkát közösen értékeljük.

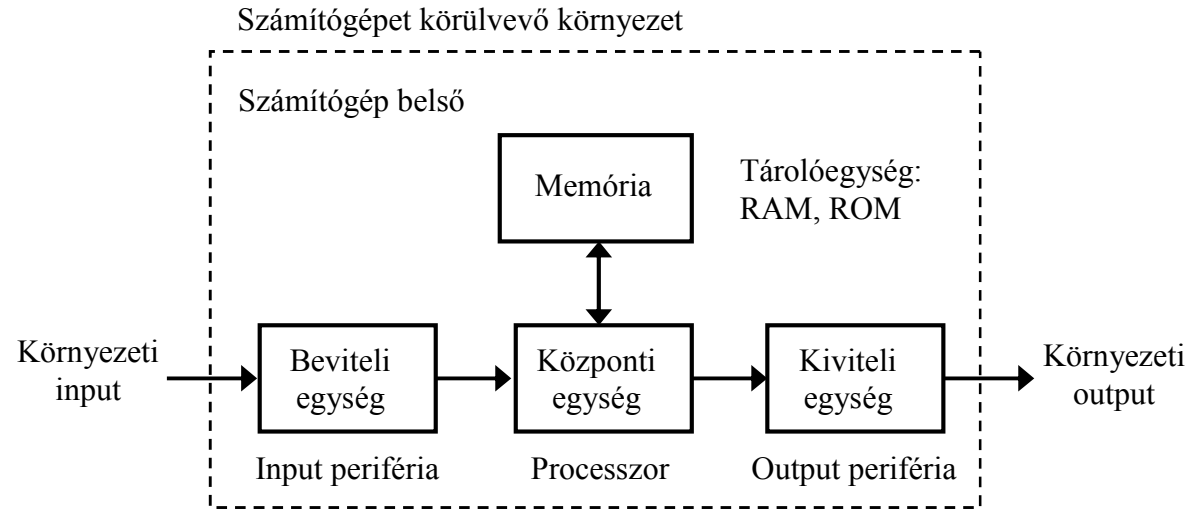
A tananyag

1. Foglalkozás

Számítógép általános modellje

A számítógép a hardver (fizikai rész) és az azt működtető programok, a szoftver (szellemi termék) szerves egységének tekinthető.

A számítógép egyszerűsített tömbvázlata:



1. ábra

A számítógép a környezetből származó jeleket a bemeneti egységén keresztül fogadja, a központi egysége feldolgozza (transzformálja) és előállítja a kimeneti jeleket, amelyeket a kiviteli egység bocsát a felhasználó birtokába (1. ábra). A memóriára azért van szükség, hogy a feldolgozást irányító processzor innen vegyen utasításokat (instrukciókat) a jelátalakítás módjára, és hogy a keletkezett átmeneti adatokat el tudja ide menteni a központi egység. A számítógép funkcionális egységei speciális vezetékkötegekkel, úgynevezett buszáramkörökkel vannak összekötve. [**1.** 44. o]

Neumann-elvek

Neumann János magyar származású amerikai tudós volt az, akinek a tervei és útmutatásai alapján megépítették az első tárolt programú elektronikus digitális számítógépeket pl. az EDVAC-ot (Electronic Discrete Variable Automatic Calculator).

Neumann 1945-ben fogalmazta meg egyik művében a számítógépek építésével (felépítésével) kapcsolatos téziseit. Az elképzelései annyira helyesnek bizonyultak, hogy napjainkban is ezen elvek alkotják a számítástechnika alapjait.

Röviden megfogalmazva a Neumann-elvek:

1. A számítógép legyen teljesen elektronikus;
2. A gép a bináris (2-es) számrendszert használjon;
3. Az adatok és a programok a gép belső tárolójában helyezkedjenek el;
4. A vezérlőegység emberi beavatkozás nélkül értelmezze és hajtsa végre az utasításokat;
5. A számítógép tartalmazzon egy olyan egységet, ami képes elvégezni az alapvető logikai és aritmetikai műveleteket.

A számítógépek csoportosítása

Generációs elv:

1. **generációs számítógépek:** 40-es évek, aktív alkatrészei elektroncsövek, gépi kódban programozhatóak, lyukszalagot ill. lyukkártyát alkalmaztak adat be és kivitelre.
2. **generációs számítógépek:** 50-es évek, aktív alkatrészei tranzisztorok, assembly nyelven programozhatóak, megjelenik a mágnesszalag.
3. **generációs számítógépek:** 60-es évek, integrált áramkörökből épültek fel, már magas szintű programozási nyelven is programozhatóak, mágneslemezeket használnak adattárolásra, megjelenik az operációs rendszer.
4. **generációs számítógépek:** 70-es évektől, nagy integráltságú áramkörökből (VLSI) épülnek fel, a számítógépek méretei a mai szintre csökkenek, e generációs gépeket használja az emberiség ma is.
5. **generációs számítógépek:** a 80-es évektől nagy pénzeket fektettek be (tipikusan Japánban) a tisztán tudás alapú számítógépek kifejlesztésére, a projektet sikeresnek ítélték, az elterjedésére, azonban még várni kell.

Használati felosztás:

1. **generációs számítógépek:** csak szűk tudósréteg alkalmazta főleg katonai alkalmazásokban tudományos számításokra, a nemzetgazdaságokban különféle adatfeldolgozásokra használták.
2. **generációs számítógépek:** már megjelenik a számítógép ipari folyamatirányító (gyártó) rendszerekben is.
3. **generációs számítógépek:** ettől a generációtól kezdve a számítástechnikai eszközök már általános elterjedéssel bírnak, a felhasználásuknak igazán semmi nem szab határt.

Memória áramkörök felépítése

A memória áramkörök a programozható berendezések (pl. a processzor) nélkülözhetetlen egysége, feladata az adattárolás.

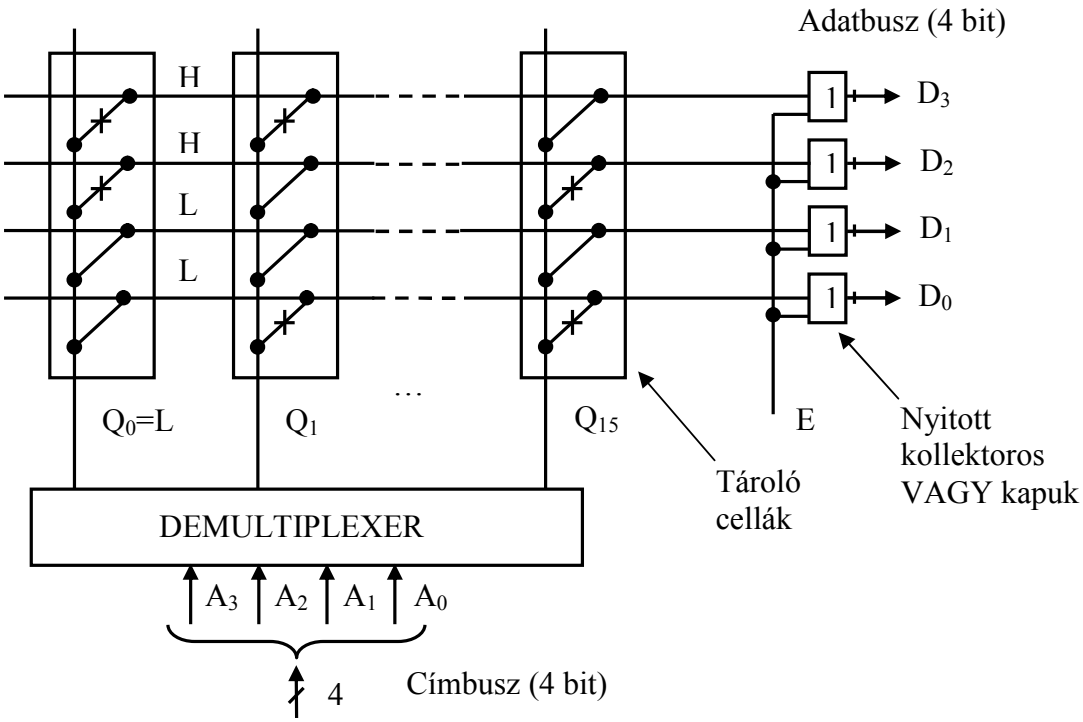
Memória áramkörök csoportosítása felhasználás szempontjából.

Csak olvasható tárak: feladatuk programtárolás.

- **ROM** (**R**ead **O**nly **M**emory): a félvezető gyár programozza be a kívánt tartalommal, a felhasználó nem tudja felülírni a tartalmát.
- **PROM** (**P**rogrammable **ROM**): a félvezető gyár egy üres ROM áramkört bocsát rendelkezésére, a tartalmat a felhasználó helyezheti el az áramkörben, de csak egyszer.
- **EPROM** (**E**resable **PROM**): olyan PROM, amely törölhető, ezért természetesen újra lehet programozni, kezelése nehézkes, mert a memória törlése a félvezető részre bocsátott UV fénysugárzás segítségével történik.
- **EEPROM** (**E**lectrically **PROM**): olyan EPROM amely már egyszerű módon elektromos jellel törölhető, számos programozható eszköz használja, mint programtároló. A személyi számítógépek alaplapjai a BIOS-t ilyen tárban tárolják.
- **Flash memória**: egy speciális EEPROM, amely gyorsabb működésre képes, mint a hagyományos EEPROM. A Flash memória legelterjedtebb felhasználási területe a különböző memóriakártyák és a pendrive.

Felhasználás szempontjából a ROM-ok feladata a hosszú idejű adattárolás, ezért ideálisak a programtárolásra.

ROM áramkörre példa: 8 bájt kapacitású ROM (2. ábra).



2. ábra

Elemei:

- **Adatbusz:** az adatbuszon keresztül adatok tudnak mozogni, ide kerül a tárból kiolvasott és a tárba beírandó információ is, a legtöbbször bájtos szervezésű.
- **Címbusz:** a címbusz segítségével lehet azt a memóriarekeszt kiválasztani, amellyel írási (RAM) vagy olvasási műveletet kell végezni, a címbusz szélességét legtöbbször a memória kapacitása határozza meg.
- **Demultiplexer:** olyan digitális áramkör, amely több kimenettel (Q) és cím bemenettel (A) rendelkezik. A kimenetei közül mindig csak egy van aktív logikai szinten (ez lehet L és H szint is), hogy ez melyik legyen azt a címbemenet, határozza meg.
- **Engedélyezőjel:** ez a jel határozza meg azt, hogy az adatbuszra mikor kerüljön ki a kiolvasott adat.

Működés:

Az információ tárolásának az elve az, hogy a keresztben elfutó adatvonalak és a függőleges vezetékek között kialakított rövidzár a logikai alacsony, a szakadás pedig a magas szintet kódolja. A címbuszon megjelenő cím (0000) a demultiplexer, megfelelő kimenetét aktivizálja (Q_0 = L). A kapcsolatok, kialakításától függ a kiolvasott információ bitértéke (HHLL). Az adatbuszra csak akkor kerülhet ki a kiolvasott érték, ha azt az engedélyezőjel lehetővé teszi (E = H).

Írható-olvasható tárak:

A **RAM** (**R**andom **A**ccess **M**emory), úgynevezett véletlen elérésű írható és olvasható memória.

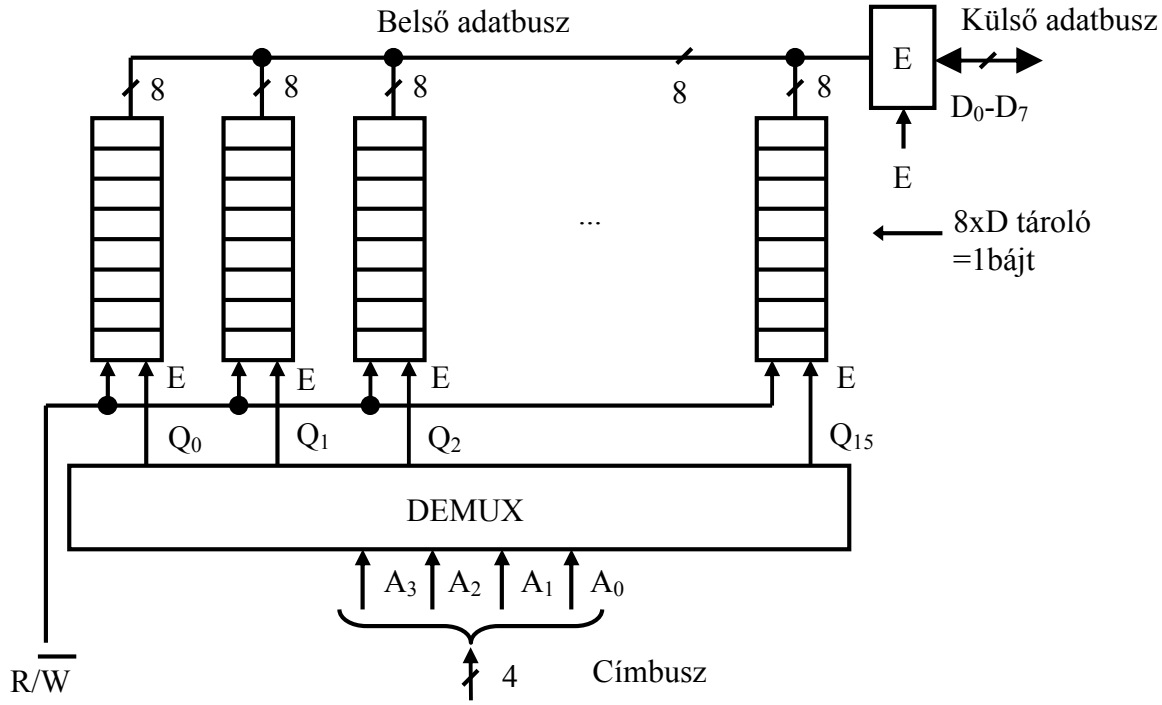
- **SRAM** (**S**tatic **RAM**): statikus RAM, a benne elhelyezett információ az áramkör tápfeszültségének a megszűnéséig megőrzi (a tárolóeleme RS vagy D tároló).
- **DRAM** (**D**inamic **RAM**): dinamikus RAM, a benne elhelyezett információ csak bizonyos ideig őrződik meg, a tárat frissíteni (emlékeztetni) kell a már tárolt információra (a tárolóeleme egy kis kapacitású kondenzátor).

A RAM-ot általában, mint átmeneti tárat alkalmazzák, a számítások közben keletkezett ideiglenes adatok tárolására. A RAM nevében a „Random” (véletlen) jelző arra utal, hogy bármely (véletlenül kiválasztott) elemét egyazon idő alatt lehet elérni (ez az asszociatív tár jellemzője).

RAM áramkörre példa: 16 bájt kapacitású RAM (3. ábra).

Elemei:

- **Belső adatbusz:** a belső busz kapcsolja össze a tárolócellákat;
- **Külső adatbusz:** a memória áramköri csatlakozó pontja, ezen keresztül lehet elérni kívülről a belső adatbuszt;
- **R/W:** a memória olvasását illetve írását engedélyező jel.



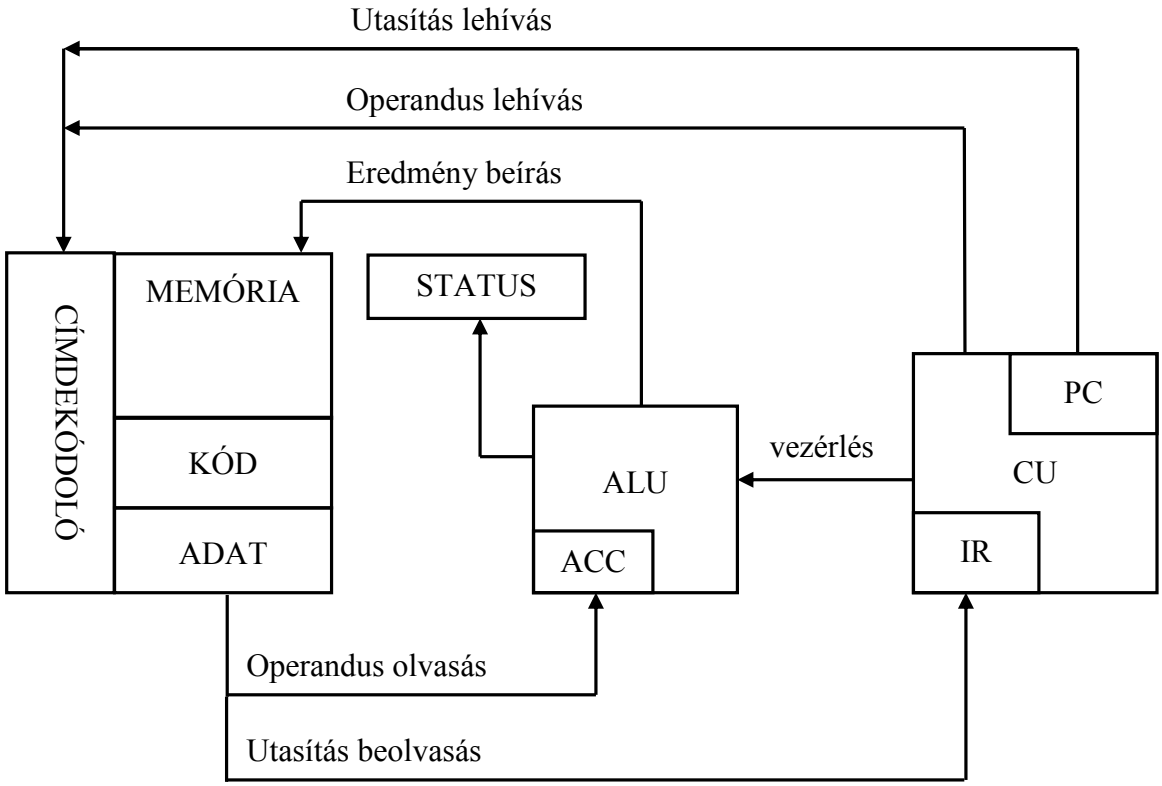
3. ábra

Neumann-elvű mikroprocesszorok működési elve

Működési modell (4. ábra) elemei:

- **ALU:** (**A**ritmetic **L**ogic **U**nit) aritmetikai-logikai egység, matematikai és logikai műveleteket végez el;
- **ACC:** (**ACC**umulator) akkumulátor, speciális regiszter, a legtöbb ALU-t érintő művelet egyik operandusát és a művelet eredményét tartalmazza;
- **CU**: (**C**ontrol **U**nit) vezérlőegység, a program végrehajtását biztosítja;
- **PC:** (**P**rogram **C**ounter) programszámláló, mindig annak az utasításnak a címét tartalmazza, amit a mikroprocesszornak a tárból le kell hívnia;
- **IR:** (**I**nstruction **R**egister) utasításregiszter, a tárból lehívott utasítás bináris kódját és az adat címét fogadja és ebből hozza létre a logikai vezérlőjeleket;
- **STATUS:** (**S**tatus **R**egister) a processzor által végrehajtott műveletekről ad visszajelzést. Legfontosabb az átvitel bit (**C**arry bit), amely akkor keletkezik, ha a számítási művelet eredménye nagyobb, mint 255 és a nulla (**Z**ero bit) bit, amely a műveletvégzés 0 eredményét jelzi;
- **Címdekódoló:** kiválasztja a megfelelő memóriarekeszt;
- **Kódmemória:** itt helyezkedik el a program;

§ **Adatmemória:** adatokat tartalmaz, amelyekkel a processzor műveletet végez.



4. ábra

A processzor modell működése:

1. **PC változtatása:** a PC értéke egyértelműen kijelöl egy tárcímet a programmemóriában;
2. **Utasítás lehívás:** a címet a címdekódoló értelmezi és kiválasztja a megfelelő memóriarekeszt és a tartalmát az IR-be tölti;
3. **Dekódolás:** az IR-ben a kódnak megfelelő vezérlőjelek jönnek létre, amelyek a többi egységet vezérlik;
4. **Adatlehívás és végrehajtás:**
   - § Ha a kód tartalmazza az adatot, akkor azon az ALU végrehajtja a kijelölt műveletet;
   - § Ha a kód csak az utasítás címét tartalmazza, akkor a processzornak ismét a memóriához kell fordulni, és az adat címének az ismeretében, a megfelelő memória helyről az ALU-ba kerül a tényleges adat. Az ALU ezután végrehajtja a kijelölt műveletet az adaton.

A fenti műveleteket a processzor a programnak megfelelően ciklikusan hajtja végre, azaz minden végrehajtott utasítás után megváltoztatja a PC értékét (a legtöbbször növeli eggyel). Egy utasítás végrehajtási idejét a processzor gépi ciklusának nevezik.

A processzor nem tud bármilyen utasítást „megérteni” csak azokat, amelyeknek az elvégzésére lehetővé tették. Az adott processzor által végrehajtható, értelmezhető utasításokat a processzor utasításkészletének nevezik. A processzor utasításkészlete természetesen előre rögzített. [**1.** 47. o]

Ellenőrző kérdések

1. Sorolja fel a Neumann-elveket.
2. Csoportosítsa a félvezető alapú tárolóelemeket.
3. Röviden fogalmazza meg a Neumann-elvű processzorok működési elvét.

2. Foglalkozás

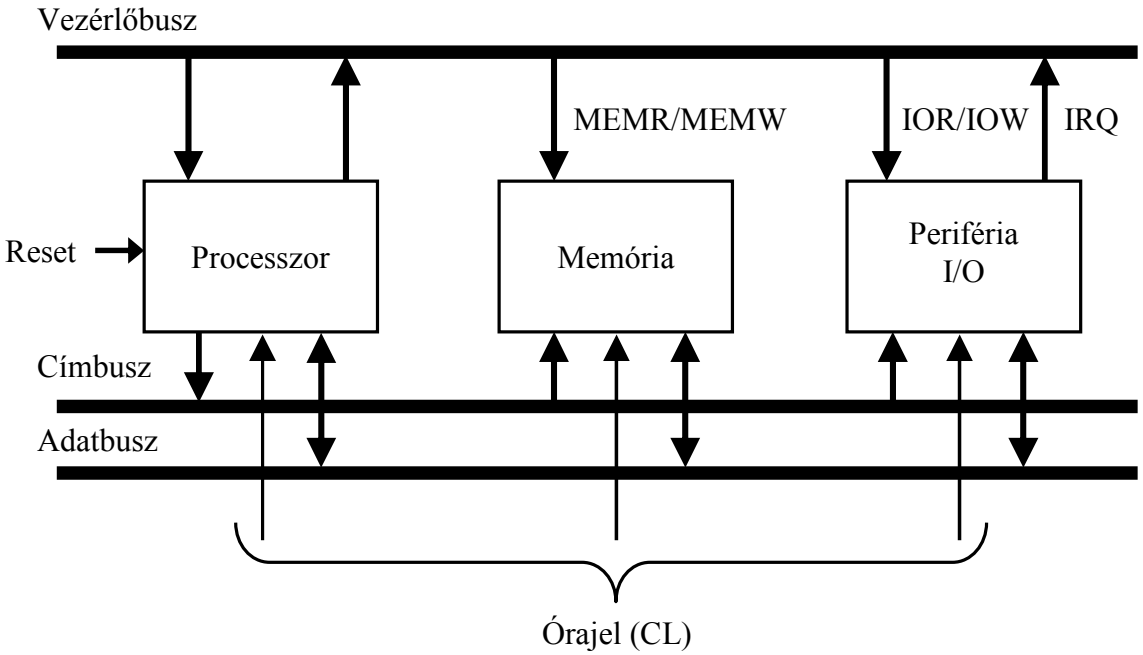
Számítógép architektúrák

A processzoroknál, két fajta utasításrendszer fejlődött ki:

1. **CISC** (**C**omplex **I**nstruction **S**et **C**omputer), összetett utasításkészletű processzor.
   - § Utasításkészlete összetett, sok és bonyolult utasítást tartalmaz.
   - § Bonyolult a címzési rendszere, az utasítások változó hosszúságúak.
   - § Utasítások változó gépi ciklusúak (az egyes utasítások végrehajtási ideje különböző lehet).
   - § Az assembly programozás egyszerű (sok utasítás miatt).
   - § Kevés regisztert tartalmaz.
   - § Pl. Intel 286/386/486/PI-IV, DEC VAX, Motorola 68000.
2. **RISC** (**R**educed **I**nstruction **S**et **C**omputer), csökkentett utasításkészletű processzor.
   - § Utasításkészlete egyszerű, csak a legalapvetőbb utasításokat tartalmazza.
   - § Egyszerű címzéssel és fix kódhosszúsággal rendelkezik.
   - § Minden utasítás azonos gépi ciklusú (minden utasítás végrehajtási ideje azonos).
   - § Az assembly programozás bonyolult (kevés utasítás miatt).
   - § Sok regisztert tartalmaz, ezért a működés gyors.
   - § Pl. DEC ALPHA, SUN SPARC, IBM POWER PC.

Mikroszámítógép felépítése

A mikroszámítógép a lehető legkevesebb alkatelemből álló, működőképes programozható számítógépes konfigurációt jelenti (5. ábra).



5. ábra

Mikroszámítógép részei:

1. **Processzor:**
   - Biztosítja a vezérlőjeleket;
   - Adatmozgatást végez;
   - Dekódolja és végrehajtja az utasításokat;
   - Vezérli a periféria elemeket.
2. **Memória:** A működtető programot és az adatokat tárolja.
3. **Periféria (I/O) egység:** A külvilággal tart kapcsolatot.
4. **Busz vezetékek (sínek):**
   - **Vezérlőbusz:** ezen keresztül utasítja a processzor a többi elemet a megfelelő működésre, kétirányú;
   - **Címbusz:** a memória (vagy a periféria) megfelelő tároló rekeszét (portját) címzi, egyirányú működésű;
   - **Adatbusz:** adatok mozognak rajta, kétirányú.
5. **Legfontosabb vezérlőjelek:**
   - **Órajel** (**CL**ock): a mikroszámítógép működését ütemezi, minden jelváltás, kommunikáció csak az órajel változásának ütemében történhet;
   - **Reset:** a számítógépet alaphelyzetbe állító bemenet;
   - **MEMR/MEMW: MEM**ory **R**ead/**W**rite, olyan jel, amely a memória olvasását vagy írását jelzi;
   - **IOR/IOW: I**nput **O**utput **R**ead/**W**rite periféria olvasását vagy írását jelzi;
   - **IRQ: I**nterrupt **R**e**Q**uest, megszakítás kérése. A megszakítás egy változás a vezérlés menetében, amit a környezet vagy egy periféria (belső esemény) kezdeményez. [**1.** 50. o]

Mikrovezérlő fogalma

Régen:

A 70-es évek elején megjelent mikroprocesszorokat és a segítségükkel felépített mikroszámítógépeket a 70-es évek közepére már széles körben alkalmazták, de a felhasználók felől egyre újabb igények léptek fel. Ebben az időben a mikroprocesszorok többsége 1, 4, 8 bites szóhosszúságú volt, a működtető órajel frekvencia 0,5 és 4 MHz közötti értékkel bírt.

A mikroszámítógépekkel szemben a felhasználóknak három jól elkülöníthető elvárása fogalmazódott meg:

- **Gyorsabb működés**, órajel frekvencia növekedése;
- **Nagyobb szóhosszúság**ú, nagyobb bitszélességű adatokon elvégzett művelet gyorsabb működést jelent;
- **Tömörebb (kompaktabb) felépítés**, egy áramkörben legyen megtalálható a mikroszámítógép összes eleme (CPU, memória, perifériaelemek).

Most:

A fent megfogalmazott igényeket külön-külön próbálták a gyártók teljesíteni (személyi számítógép CPU: 3 GHz, 64 bit; videokártya GPU: 512 bit). A kompakt felépítésű áramköröket kezdetben egy csipes mikroszámítógépnek nevezték, ezek ma használatos elnevezése a mikrovezérlő ill. mikrokontroller.

A mikrokontrollerek, egyetlen integrált áramkörrel (**IC**: **I**ntegrated **C**ircuit) megvalósított, mikroszámítógépek. Az integrált funkciókból adódó előnyöknek köszönhetően a mai elektronika elképzelhetetlen a mikrovezérlők nélkül. A manapság gyártott elektronikai készülékek szinte kivétel nélkül mikrovezérlő alapúak: TV, DVD, MP3 lejátszó.

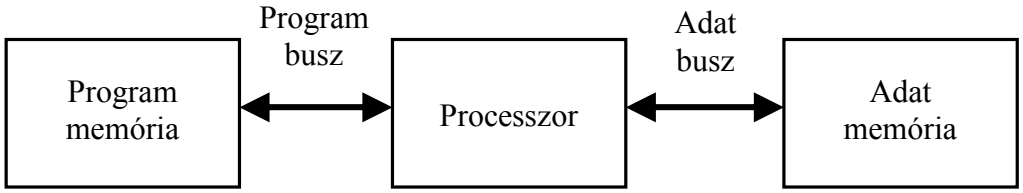
Mikrovezérlők felépítése

A mikrovezérlők RISC jellegű processzormagra épülő, változatos memória és periféria elemekkel rendelkező áramkörök. Nagyon sok neves és akár kevésbé ismert félvezetőgyártó készít egyedi, illetve egymással kompatibilis (helyettesíthető) áramköröket.

Nem teljes körű felsorolás a gyártókról, egy-egy ismertebb gyártmányukkal együtt:

- Microchip – PIC mikrovezérlők;
- Atmel – AVR gyártmányok;
- Intel – 8051-es sorozat;
- Texas Instrumens – TMS sorozat.

A mikrokontrollerek felépítése sajátos, eltér a Neumann-i alapelvektől, a mikrovezérlők számára sokkal kedvezőbb úgynevezett Harvard felépítéssel bírnak (6. ábra).



6. ábra

A Harvard architektúra alapját, azaz elképzelés alkotja, hogy a mikrokontrollerben a program és az adatmemória külön egységként legyen kialakítva. Ez számos a működést, segítő tulajdonságot kölcsönöz a mikrovezérlőknek.

Harvard architektúra jellemzői:

- Minden utasítás 1 szavas, a programmemória szóhosszúsága tetszőleges lehet, így nem jelent korlátozó tényezőt az adatmemória általában bájtos kialakítása;
- A speciális kialakítás miatt nem alakulhat ki versenyhelyzet az adat és kód elérésnél, akár egyszerre (párhuzamosan) is kezelhetőek a memóriák;
- A fenti tulajdonságok miatt ez a fajta felépítés nagyon gyors működést biztosít.

Mikrovezérlők általános jellemzői

- Egy csipes áramkörök, amelyek valamilyen mikroszámítógép konfiguráció minden alkotóelemét, CPU, memória (RAM, ROM), I/O elemek, rendszer órajel generátor, tartalmazzák.
- Túlnyomórészt Harvard felépítésűek, vagyis külön program és adat memóriával rendelkeznek. A program memória 0,5-16 kszó, az adat memória is viszonylag kicsi, maximum 128 bájt-8 kbájt nagyságrendben mozog.
- RISC magra épülnek, ezért utasításkészletük egyszerű, programozásuk mégis könnyen elsajátítható.
- Külső memóriával általában nem bővíthetők, vagy ha igen, az az I/O vonalak felhasználásával történik, és így a szabadon maradó I/O vonalak száma csökken.
- Jellegzetes erőforrásokkal rendelkeznek. (Watchdog timer, számláló/időzítő (Timer) áramkörök, digitális be- és kimeneti vonalak, analóg bemenet).
- A kivezetések számának csökkentése céljából többcélú kivezetéseik vannak (egy kivezetéshez több funkció van rendelve).
- Számos külső és belső megszakítás vonallal rendelkezhetnek.
- Általában tartalmaznak kommunikációs portot (pl. UART), amely a programozásukat megkönnyíti, valamint lehetővé teszi beillesztésüket egy számítógépes környezetbe.
- Fogyasztásuk kicsi, ezért telepes (akkumulátoros) készülékben is alkalmazhatóak.

Ellenőrző kérdések

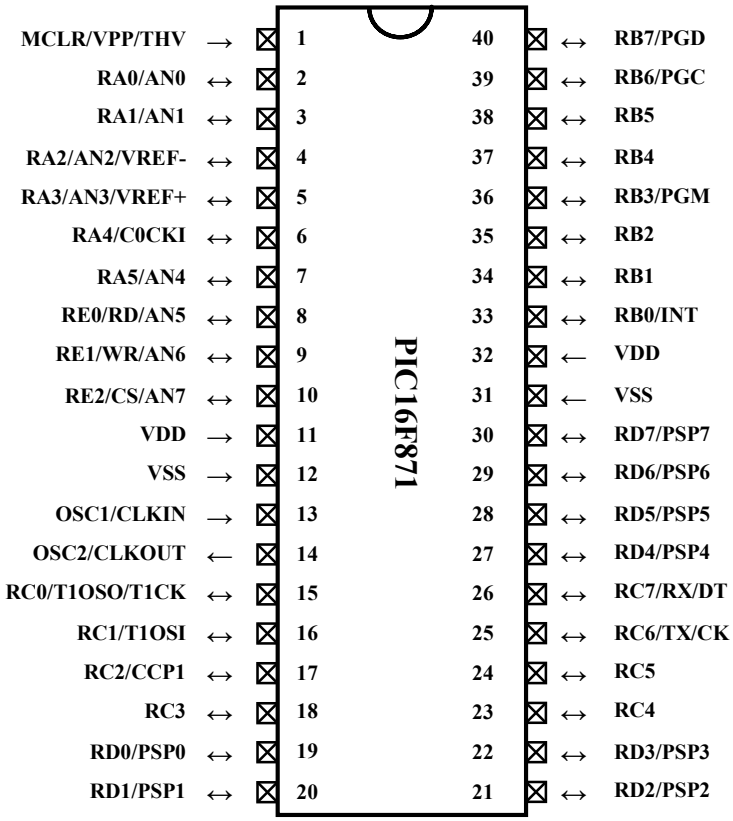
1. A CISC és a RISC architektúrák összehasonlítása.
2. Mikrokontroller fogalma.
3. Mikroszámítógép vezérlőjeleinek a csoportosítása.
4. Mi a Harvard architektúra lényege?
5. Mikrovezérlők általános jellemzői.

3. Foglalkozás

Egy PIC (16F871) konkrét jellemzői

CPU jellemzők:

- § Nagy teljesítményű RISC CPU;
- § Könnyen megtanulható 35 db utasítás;
- § Minden utasítás egy utasítás ciklus alatt végrehajtódik, kivéve az elágazó utasításokat, ezek két ciklust igényelnek;
- § A működési sebesség:
  - DC-20 MHz órajel;
  - DC-200 ns gépi ciklus idő;
- § 2048 x 14 szó kapacitású program memória (Flash EEPROM);
- § 128 bájtos adat memória (SRAM);
- § 64 bájtos EEPROM adat memória;
- § A lábkivezetése teljesen kompatibilis a PIC 16CXXX sorozattal;



7. ábra

- § 11 megszakítási forrással rendelkezik;
- § 8 szintű hardver verem;
- § Direkt, indirekt és relatív címzési mód;
- § Bekapcsolási reset (POR);
- § Bekapcsolási időzítő (PWRT) és oszcillátor indítási időzítő (OST);
- § Watchdog időzítő, saját RC oszcillátorral;
- § Programozható kódvédelem;
- § Alacsony fogyasztású alvó üzemmód (SLEEP);
- § Választható oszcillátor konfiguráció;
- § Alacsony teljesítményigényű nagy sebességű CMOS Flash/EEPROM technológia alkalmazása;
- § Áramkörön belüli soros programozási (ICSP) lehetőség;
- § Egyszerű 5V-os, áramkörön belüli soros programozási lehetőség;
- § A processzor képes a programmemória írására és olvasására;
- § Széles tápfeszültség tartomány: 2,0 – 5,5 V;
- § Kivezetések magas 25 mA-es áramterhelhetősége H és L szintben is;
- § Kapható kommersz és ipari kivitelben is;
- § Alacsony teljesítményfelvétel:
  - < 1,6 mA, 5 V-os tápfeszültség és 4 MHz-es órajel mellett;
  - 20 μA, 3 V-os tápfeszültség és 32 kHz-es órajel mellett;
  - < 1 μA, készenléti (alvó) üzemmódban.

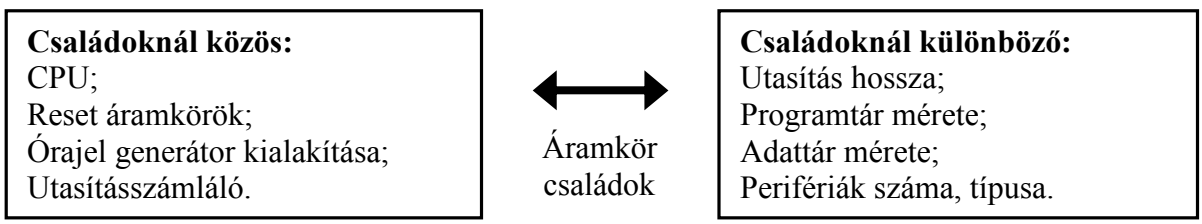
Periféria jellemzők:

- Timer0: 8-bites időzítő/számláló 8 bites előosztóval;
- Timer1: 16-bites időzítő/számláló 8 bites előosztóval, amely képes alvó üzemmódban is működni;
- Timer2: 8-bites időzítő/számláló 8 bites elő és utóosztóval;
- Egy Capture (mintavevő), Compare (összehasonlító) és PWM modul (CCP);
  - A Capture modul 16 bites, maximum 12,5 ns-os érzékenységgel;
  - A Compare modul 16 bites, maximum 200 ns-os érzékenységgel;
  - 10 bites érzékenységű PWM modul;
- 8 csatornás 10 bites érzékenységű A/D modul;
- Univerzális soros port (USART);
- 8 bites párhuzamos port (PSP);
- Tápfeszültség csökkenést érzékelő áramkör és reset (BOR). [**4.** 1-3. o]

Kulcsjellemzők	PIC16F871
Működési frekvencia	DC – 20 MHz
Reset lehetőségek	POR, BOR, PWRT, OST
14 – bites szóhosszúságú EEPROM memória	2048 szó
Adat memória	128 bájt
EEPROM adat memória	64 bájt
Megszakítási források száma	11 db
I/O portok	A (6), B (8), C (8), D (8), E (3) portok
Számláló/időzítők	3 db
Mintavevő/Összehasonlító/PWM modul	1 db
Soros kommunikáció	USART (RS-232)
Párhuzamos kommunikáció	PSP
10 bites A/D modul	8 csatorna
Utasítás készlet	35 db utasítás
Tokozás	40 lábas DIL tokozás (7. ábra)

Áramköri családelv

Azért, hogy egy mikrokontroller gyártó cég (Microchip, Atmel) minél szélesebb felhasználási igényt ki tudjon elégíteni egy termékével (PIC, AVR), több változatban (kialakításban) gyártják, akár egy konkrét típuson belül is létrehoznak más, esetleg speciális tudású egyedeket (8. ábra). [**1.** 52. o]



8. ábra

A legfontosabb különbség az utasítások hosszában mutatkozik meg, ezért ez a tulajdonság lett az egyértelmű megkülönböztető jel az áramkör családokra.

Egyszerű megfogalmazásban egy utasítás felépítése:

utasítás = műveleti kód + operandus cím

- Műveleti kód: az utasítás bináris kódja (mit tegyen a CPU);

- Operandus cím: az adat bináris címe (az adat, amin műveletet végez a CPU).

Utasításhossz szerinti megkülönböztetés (PIC):

- 12 bites: 7 bit műveleti kód 5 bit operandus cím 16C5XX család;
- 12/14 bites: 12XXX család;
- 14 bites: 7 bit műveleti kód 7 bit operandus cím 16XXX család;
- 16 bites: 8 bit műveleti kód 8 bit operandus cím 17CXX és a 18XXX család.

A családok felülről kompatibilisek egymással pl. a 17-es családnál alkalmazhatóak a 16-os kódjai (fordítva nem biztos, hogy igaz!). A személyi számítógépek kompatibilis jelzője is ezt takarja, ezért elméletileg egy XT gép 8086-os processzorára írt gépi kód minden további nélkül futtatható a legújabb többmagos processzoron is!

A továbbiakban csak a 16-os sorozat 16F871-es áramkörével foglalkozunk részletesen.

PIC 16F871 architektúra

Egyszerűsített működési elv

A programot a 2k szó kapacitású 14 bites Flash EEPROM tároló tartalmazza (2048 x 14bit). A kódmemóriát a programszámláló címzi meg. A memóriahelyről kiolvasott utasítás az utasításregiszterbe, majd az utasítás dekódoló áramkörbe kerül. A dekódolás után a PIC végrehajtja az utasítást. Amennyiben szükséges használni az adatmemóriát vagy valamelyik perifériaelemet akkor az is a működés részévé válik (9. ábra). [**4.** 6. o]

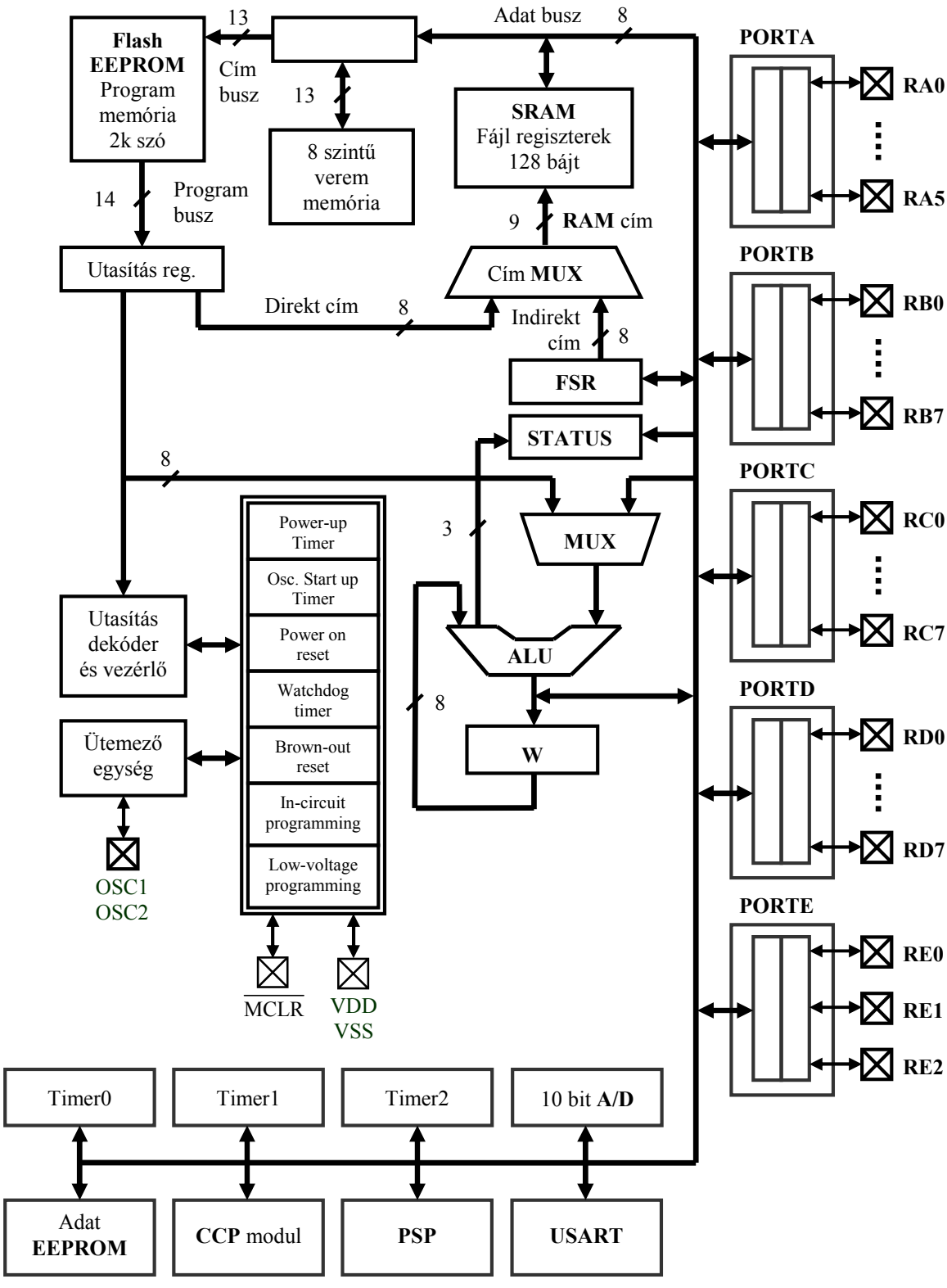
Egy egyszerű aritmetikai utasítás végrehajtása

Az utasítás legyen egy egyszerű inkrementáció (`INCF ADAT1,1`), az utasítás végrehajtása után az ADAT1 fájl regiszter értéke eggyel növekszik (magas szintű programozási nyelven ezzel ekvivalens utasítások: ++ADAT1 vagy az ADAT1:=ADAT1+1).

A végrehajtás lépései:

1. A PC értéke a címbuszon keresztül megcímzi azt a programtár rekeszt, amelyben a vizsgált utasítás található;
2. A programbuszon keresztül az utasítás bináris kódja az utasítás regiszterbe kerül;
3. Mivel az utasításban direkt utalás történik RAM területre (ADAT1) a kívánt adattár rekesz kiválasztódik;
4. Az utasítás dekódolódik az „Utasítás dekóder és vezérlő” áramkörben, ennek hatására a direkt címről betöltődik az ADAT1 tartalma a W munkaregiszterbe;
5. Az ALU végrehajtja az utasítást, a W-hez hozzáad 1-et;
6. Amennyiben szükséges az ALU megváltoztatja a STATUS regiszter értékeit;
7. A kapott eredmény visszaíródik a RAM területre, az ADAT1 regiszterbe;
8. A mikrokontroller kész a következő utasítás végrehajtására, inkrementálódik a PC.

A PIC16F781 belső felépítése

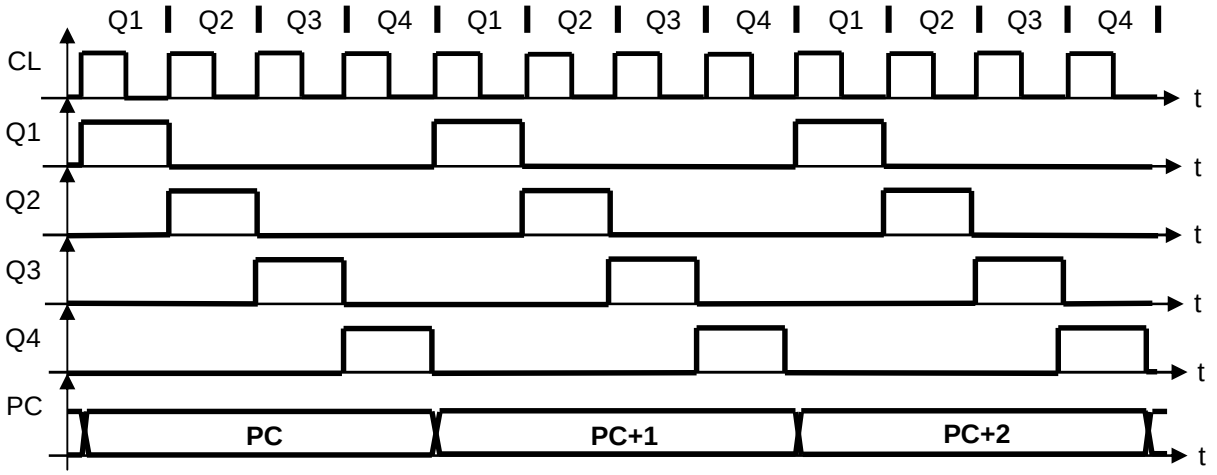


9. ábra

Utasítás feldolgozás és végrehajtás

Minden digitális számítógéphez hasonlóan a mikrokontrollerek utasítás végrehajtását (és minden egyéb műveletet is) a rendszer órajel (CL) vezérli. Egy utasítás végrehajtása természetesen több lépésből áll (lásd az előző pontot), ezért az órajelnek csak valamilyen arányú leosztásával lehet a programvégrehajtást alapjaiban meghatározó PC értékét megváltoztatni (pl. csak minden negyedik órajel periódus után, változik a PC értéke).

Az utasítás végrehajtását a legszemléletesebben idődiagramban lehet bemutatni.



10. ábra

A PIC-ek esetében, egy utasítás végrehajtása négy órajel periódust (ill. ütemet: Q1, Q2, Q3, Q4) igényel. A négy ütem együttesen alkot egy gépi ciklust (10. ábra).

Tevékenységi sor egy gépi ciklus alatt:

- Q1: PC inkrementálódása;
- Q2: programtár megfelelő tárrekeszének a címzése;
- Q3: az utasítás kiolvasása a programtárból;
- Q4: az utasítás bináris kódja az utasításregiszterbe kerül.

A következő gépi ciklus alatt lezajlódó folyamat:

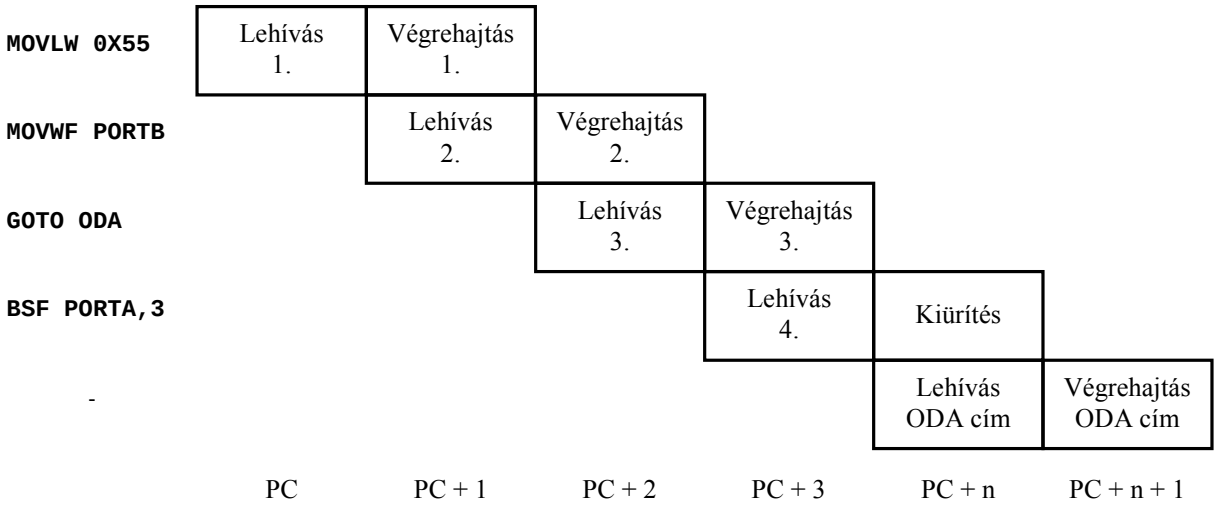
- Q1: utasítás dekódolódik;
- Q2: az operandus kiolvasása a fájlregiszterből (ha kell);
- Q3: utasítás végrehajtása;
- Q4: eredmény visszaírása a fájlregiszterbe.

A fent leírt folyamat elméletileg azt tételezi fel, hogy egy utasítás két gépi ciklus alatt (8 órajel periódus) zajlódik le. A valóságban egy utasítás valóban egy gépi ciklust igényel, ezt az úgynevezett utasítás előolvasás (pipe-line elv) fogja biztosítani.

Átlapolt utasítás végrehajtás

Az utasítások végrehajtásának gyorsítása érdekében a mikrokontroller alapértelmezésben felhasználja a pipe-line (csővezeték) elvet, amely azt jelenti, hogy az éppen aktuális utasítás végrehajtásával párhuzamosan a vezérlő egység beolvassa a soron következő utasítást is. Ilyen módon egy utasítás végrehajtása így valóban csak egy gépi ciklust igényel. Kivételt képeznek ez alól az elágaztató utasítások (pl. `CALL`, `GOTO`) ezekben, az esetekben az előolvasási sor tartalmát el kell dobni (kiürítés), s az új (pl. a szubrutinban levő) utasítást kell lehívni és végrehajtani. [**1.** 56. o]

Példa az átlapolt utasítás végrehajtásra



11. ábra

Utasítások rövid magyarázata (11. ábra)

- `MOVLW 0X55`: **MOV**e **L**iteral to **W**ork register ↔ mozgass egy (az argumentumban megadott) állandót (a hexadecimális 55-öt) a munkaregiszterbe (W-be);
- `MOVWF PORTB`: **MOV**e **W**ork register to **F**ile register ↔ mozgassd a munkaregiszter (W) tartalmát egy (az argumentumban megadott) fájlregiszterbe (a PORTB-be);
- `GOTO ODA`: ↔ a vezérlés átadása az ODA címkén elhelyezkedő utasításra;
- `BSF PORTA,3`: **B**it **S**et **F**ile register ↔ állítsd logikai magas szintbe (azaz 1-be) egy (az argumentumban megadott) fájlregiszter megadott helyértékű bitjét (a 3-at).

Működés

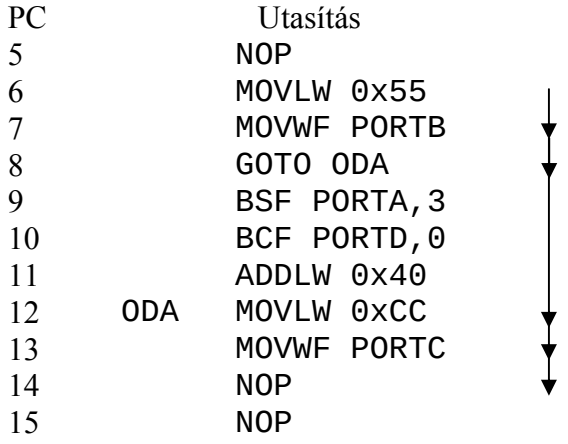
A PC értékének megfelelően az első utasítás lehívódik (`MOVLW 0X55`), a következő gépi ciklusban a PC értéke növekszik eggyel, ekkor lehívódik a következő utasítás (`MOVWF PORTB`) és ezzel párhuzamosan végrehajtódik az első. A PC következő inkrementálódása után beolvasásra kerül a `GOTO ODA` utasítás, miközben a már megismert módon végrehajtódik az előző 2. utasítás. A következő beolvasott utasítás a soron következő negyedik a `BSF PORTA,3`. A következő végrehajtandó utasítás azonban nem a sorban következő lesz, mert a programszámláló értéke a feltétel nélküli ugró utasítás hatására n eltolási értékkel meg fog változni. A fenti folyamatból következik hogy a 4. utasítás előolvasási sora elromlott, mert az utasítás nem volt végrehajtható. A program futása az ugrási helytől fog folytatódni.

Az átlapolt utasítás végrehajtás nem esetleges esemény, hanem a PIC-ek működésének az alapját jelentő tevékenység.

Tisztán látszik, hogy az átlapolt utasítás végrehajtás a különálló program és adatmemóriával rendelkező és általában RISC processzorokat tartalmazó Harvard architektúra kedvező tulajdonsága.

Megjegyzendő, hogy a pipe-line elv a mikroprocesszorok (CPU, GPU) esetén is természetes módon alkalmazott működési elv, a különbség annyi, hogy az átlapolódás a nagyságrendekkel nagyobb memória (cache) méret miatt többszöröse a mikrokontrollereknél alkalmazottnak.

A példa szemléltetése:



12. ábra

Ellenőrző kérdések

1. Adja meg a PIC16F871 kulcsjellemzőit.
2. Fogalmazza meg a családelvet.
3. Mi az összefüggés a Harvard architektúra és az átlapolt utasítás végrehajtás között?
4. Próbálja meg kitalálni a `BCF PORTB,5` utasítás működését.

4. Foglalkozás

PIC utasítások felépítése és típusai

A 16F-es sorozatú mikrokontrollerek utasításszó hossza a már megismert 14 bit. Az alkalmazható utasítások száma 35, ez kevésnek tűnhet, azonban nem szabad elfelejteni a RISC jellegből fakadó hatékonyságot és a fejlesztőkörnyezet (a későbbiekben megismerendő MPLAB) által biztosított plusz (ún. pszeudo) utasításokat sem.

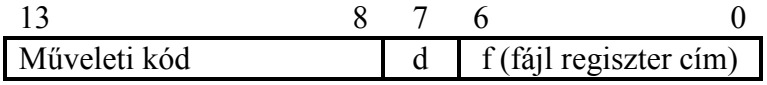
Az utasítások csoportosítása:

- Bájt orientált műveletek;
- Bit orientált műveletek;
- Állandót tartalmazó utasítások;
- Vezérlésátadó utasítások;
- Speciális utasítások.

Minden utasítás egy gépi ciklus alatt végrehajtódik, kivéve a tesztelő és a vezérlésátadók, amelyek két ciklust igényelnek.

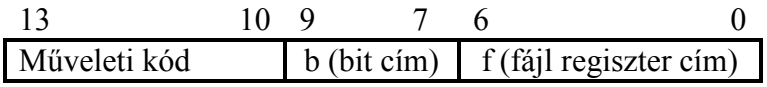
Az utasítások felépítése

1. Bájt orientált fájlregiszter utasítások:



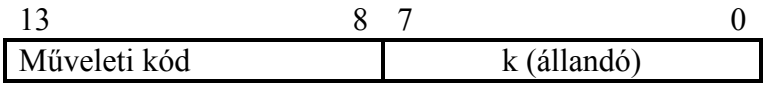
   - d = 0, ha W a cél;
   - d = 1, ha az adott f fájlregiszter a cél, 0 esetén a W;
   - f = az f fájlregiszter 7 bites címe.

2. Bit orientált fájlregiszter utasítások:



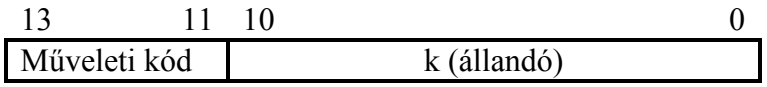
   - b = 3 bites bitcím (a fájlregiszteren belüli cím);
   - f = az f fájlregiszter 7 bites címe.

3. Állandót tartalmazó utasítások:



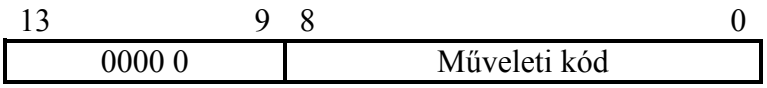
   - k = 8 bites állandó.

4. Vezérlésátadó utasítások:



   - k = 11 bites állandó (célcím).

5. Speciális utasítások:



PIC 16F871 utasításkészlete

Mnemonic		Leírás	Ciklus	14 bites műveleti kód				Flag
operandus				Msb			Lsb	
Bájt orientált fájlregiszter utasítások								
ADDWF	f, d	W + f → d	1	00	0111	dfff	ffff	C,DC,Z
ANDWF	f, d	W & f → d	1	00	0101	dfff	ffff	Z
CLRF	f	0 → f	1	00	0001	1fff	ffff	Z
CLRW	-	0 → W	1	00	0001	0xxx	xxxx	Z
COMF	f, d	/f → d	1	00	1001	dfff	ffff	Z
DECF	f, d	f – 1 → d	1	00	0011	dfff	ffff	Z
DECFSZ	f, d	f – 1 → d, átlép ha 0	1 (2)	00	1011	dfff	ffff	
INCF	f, d	f + 1 → d	1	00	1010	dfff	ffff	Z
INCFSZ	f, d	f + 1 → d, átlép ha 0	1 (2)	00	1111	dfff	ffff	
IORWF	f, d	W || f → d	1	00	0100	dfff	ffff	Z
MOVF	f, d	f → d	1	00	1000	dfff	ffff	Z
MOVWF	f, d	W → f	1	00	0000	dfff	ffff	
NOP	-	-	1	00	0000	0xx0	0000	
RLF	f, d	f(n) → d (n + 1), C → d(0), f(7) → C	1	00	1101	dfff	ffff	C
RRF	f, d	f(n) → d (n – 1), C → d(7), f(0) → C	1	00	1100	dfff	ffff	C
SUBWF	f, d	f – W → d [f + /W +1 → d]	1	00	0010	dfff	ffff	C,DC,Z
SWAPF	f, d	f (0-3) ↔ f (4-7)	1	00	1110	dfff	ffff	
XORWF	f, d	W ⊕ f → d	1	00	0110	dfff	ffff	Z
Bit orientált fájlregiszter utasítások								
BCF	f, b	0 → f(b)	1	01	00bb	bfff	ffff	
BSF	f, b	1 → f(b)	1	01	01bb	bfff	ffff	
BTFSC	f, b	Ha f(b) = 0 akkor átlép	1 (2)	01	10bb	bfff	ffff	
BTFSS	f, b	Ha f(b) = 1 akkor átlép	1 (2)	01	11bb	bfff	ffff	
Állandót tartalmazó utasítások								
ADDLW	k	k + W → W	1	11	111x	kkkk	kkkk	C, DC, Z
ANDLW	k	k & W → W	1	11	1001	kkkk	kkkk	Z
SUBLW	k	W – k → W	1	11	110x	kkkk	kkkk	C, DC, Z
IORLW	k	k || W → W	1	11	1000	kkkk	kkkk	Z
XORLW	k	k ⊕ W → W	1	11	1010	kkkk	kkkk	Z
RETLW	k	k → W, TOS → PC	2	11	01xx	kkkk	kkkk	
Vezérlő utasítások								
GOTO	k	k → PC (11 bit)	2	10	1kkk	kkkk	kkkk	
CALL	k	PC + 1 → TOS, k → PC	2	10	0kkk	kkkk	kkkk	
RETURN	-	TOS → PC	2	00	0000	0000	1000	
RETFIE	-	TOS → PC	2	00	0000	0000	1001	
CLRWDT	-	0 → WDT (az előosztó is, ha van)	1	00	0000	0110	0100	/TO, /PD
SLEEP	-	0 → WDT (oszcillátor stop)	1	00	0000	0110	0011	/TO, /PD

Megjegyzések:

- A / jel a **NEM** műveletet, a || a logikai **VAGY**, a **&** a logikai **ÉS**, a ⊕ a **KIZÁRÓ VAGY** jelzésére szolgál;
- **TOS**: **T**op **O**f **S**tack: a veremmemória legfelső (elérhető) eleme;
- **C**: átvitel (**C**arry) bit, **DC**: félbájt (**D**igit **C**arry) átvitel bit, **Z**: (**Z**ero) zéró bit;
- **/TO**: **T**ime **O**ut bit, **/PD**: **P**ower **D**own bit. [**4.** 103-104. o]

PIC utasítások csoportosítása

Logikai:

ANDWF, ANDLW, IORWF, IORLW, XORWF, XORLW, COMF; (7)

Aritmetikai + léptető:

ADDWF, ADDLW, SUBWF, SUBLW, CLRF, CLRW, DECF, DECFSZ, INCF, INCFSZ, RLF, RRF, SWAPF; (13)

Bitkezelő:

BSF, BCF, BTFSC, BTFSS; (4)

Adatmozgató:

MOVF, MOVWF, MOVLW; (3)

Programvezérlő:

CALL, GOTO, NOP, RETURN, RETFIE, RETLW; (6)

Rendszervezérlő:

SLEEP, CLRWDT. (2)

Utasítások működése

- **Logikai utasítások:** a 8 bites adatok bitenkénti logikai kapcsolatát végzik el: ANDWF (ANDLW): ÉS, IORWF (IORLW): VAGY, XORWF (XORLW): XOR, COMF: NEM.
- **Aritmetikai és léptető:** 8 bites aritmetika alkalmazása. ADDWF (ADDLW): összeadás, SUBWF (SUBLW): kivonás, CLRF (CLRW): törlés, DECF: regisztertartalom 1-el csökkentése, INCF: regisztertartalom 1-el növelése, DECFSZ (INCFSZ): regisztertartalom 1-el csökkentése (növelése) és a következő utasítás átlépése, ha a regiszter nullázódik. RRF: léptetés jobbra a C biten keresztül, RLF: léptetés balra a C-n keresztül, SWAPF: regisztertartalom alsó és felső 4 bitjének cseréje.
- **Bitkezelő:** fájlregiszter bitek vizsgálatát végzik el. BSF: bit 1-be írása, BCF: bit törlése, BTFSC (BTFSS): a következő utasítás átlépése, ha a vizsgált bit 0 (illetve 1).
- **Adatmozgató:** fájlregiszterek, és a W közötti adatcserét valósít meg. MOVF: fájlregiszter W-be töltése, MOVWF: W-fájlregiszterbe mozgatása; MOVLW: állandó W-be töltése.
- **Programvezérlő:** vezérlésátadó utasítások. GOTO: ugrás a megadott címre, CALL: szubrutin hívása, RETURN: visszatérés szubrutinból; RETFIE: visszatérés megszakítási szubrutinból, RETLW: visszatérés szubrutinból egy állandó betöltésével, NOP: „üres” utasítás.
- **Rendszervezérlő:** SLEEP: a processzor SLEEP (alvó) üzemmódba kerül, CLRWDT: a törli a Watchdog timer-t.

Mikrovezérlők memóriaszervezése

A Harvard felépítés miatt a program és az adatmemória teljesen elkülönítve kezelhető. A mikrokontrollerek memóriaszervezését alapvetően a családelv határozza meg.

Memóriaszervezési lehetőségek (13. ábra)

- **Program memória:** feladata a mikrovezérlőt működtető program tárolása. A memória típusa általában EEPROM, de előfordulnak típusok EPROM és hagyományos PROM kivitelben is. A PROM-ot tartalmazó kontrollereket, úgynevezett egyszer programozható (**OTP**: **O**ne **T**ime **P**rogrammable) eszközöknek is nevezik. A program memória szervezése szórendszerű, a szó mérete (bitszáma) utalhat az eszköz teljesítményére, ez hétköznapi típusoknál 12-16 bit. A program memória kapacitása

általában pár kszó, csak nagyobb teljesítményű eszközöknél jöhet szóba a külső programmemória használata.

Program memória EEPROM EPROM PROM	**Adatmemória**		
	Általános célú RAM SRAM	Speciális Funkciójú Regiszterek (SFR) SRAM	EEPROM

13. ábra

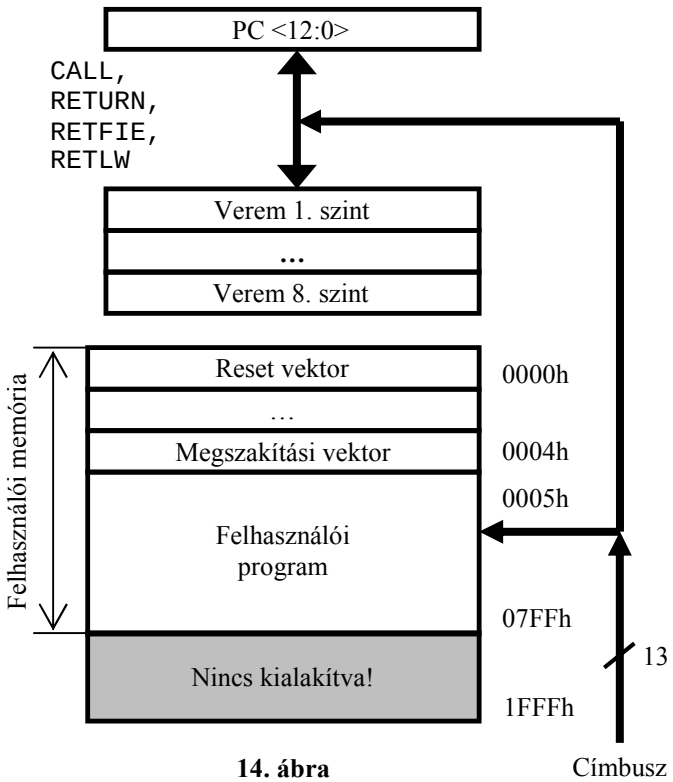
§ **Adatmemória:** több egymástól független területre bontható. Általában két memóriatípust alkalmaznak, az SRAM területen alakítják ki a fájlregisztereket (általános és speciális) és alkalmazhatnak EEPROM-ot hosszú adatmegőrzési idejű tárként. Az EEPROM terület alkalmazása főleg régi típusok esetén nem volt általános, a mai típusok kivétel nélkül tartalmazzák.

PIC 16F871 kontroller memóriaszervezése

Programmemória

A PIC 16F871-es áramkört 2 kszó 14 bites Flash EEPROM programmemóriával látták el. A kontroller 13 bites programszámlálóval rendelkezik, így összesen 8 kszó programmemória lenne megcímezhető, ezt a tárméretet azonban a PIC 16F877 típus kapta meg. A programtár adatmegőrzési idejét a gyártó cég 100 év körülire garantálja.

Blokkvázlat



14. ábra

Reset vektor: címe a 0000h, a reset vektor az a programmemória cím ahonnan a kontroller a legelső utasítást, a bekapcsolását követően végrehajtja.

Megszakítási vektor: címe a 0005h, a megszakítási vektor az a programmemória cím ahová a vezérlés átadódik egy megszakítási esemény bekövetkezése esetén. A megszakításhoz az esemény hozzárendelése (annak eldöntése, hogy mi okozta a megszakítást) szoftverből történik.

Veremmemória: a 8 szintes és 13 bites veremmemória kezelését a kontroller végzi (a megfelelő utasítások hatására), a felhasználó a vermet közvetlenül nem érheti el.

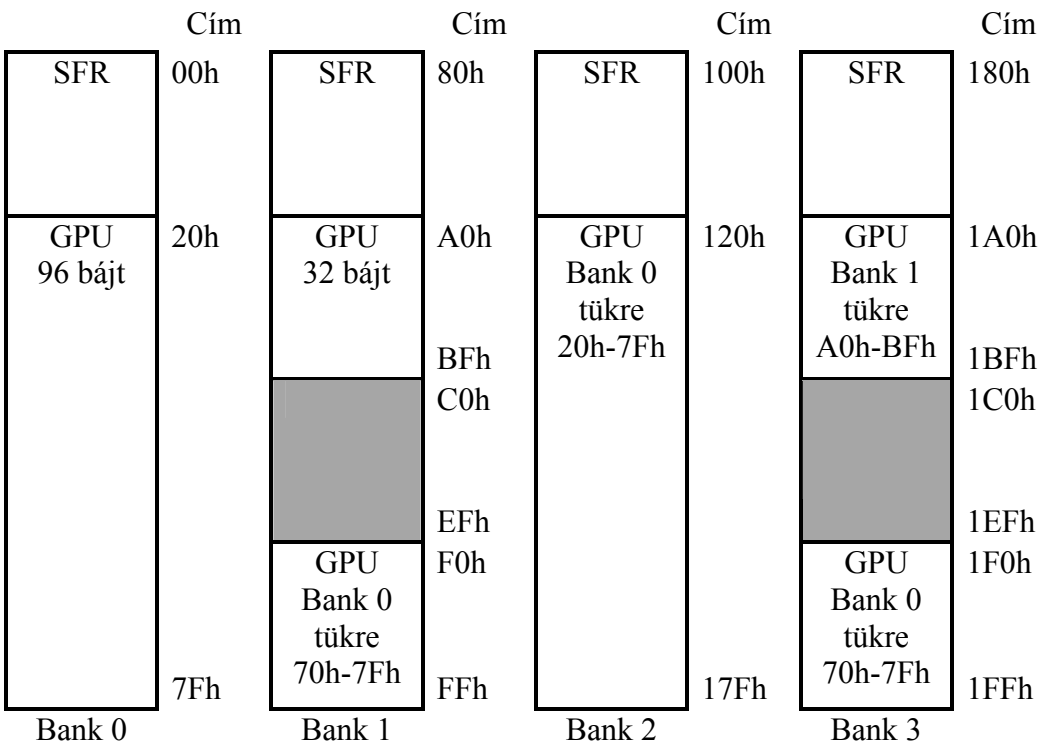
Adatmemória

A PIC16F871-es áramkör adatmemóriája a PIC mikrovezérlőknél már jól bevált módon 128 bájtos részekre (bankokra) osztott. A Microchip a bankok használatával biztosítja a kontrollerei nagyfokú kompatibilitását. A bankok felső részében alakítják ki a speciális funkciójú regisztereket (**SFR**: **S**pecial **F**unction **R**egisters), az alsó részben a felhasználói szinten használható regisztereket (**GPR**: **G**eneral **P**urpose **R**egisters) hozzák létre. Az SFR regiszterek bankonként változó tartalmúak lehetnek, a GPR területek azonban mindig azonos értékeket tárolnak (ezt az úgynevezett memóriatükrözéssel oldják meg).

Az aktuális bank kiválasztása a bankválasztó RP1 (STATUS <6>) és az RP0 (STATUS <5>) bitekkel történik. Ezek beállításáról a programozónak kell gondoskodni. A bankváltás elmaradása nem várt problémák, és rejtélyes hibák forrása lehet. [**4.** 11-12. o]

Blokkvázlat

RP <1:0>	Bank
00	0
01	1
10	2
11	3



Ellenőrző kérdések

1. Csoportosítsa a PIC16F871 utasításait több szempont szerint is.
2. Mi lehet az EPROM programmemória használatának az előnye ill. hátránya?
3. Milyen az OTP kialakítású kontroller, miért alkalmazhatják?
4. Miért biztosíthat kompatibilitást a bankok alkalmazása az adatmemóriában?
5. Milyen konkrét hibát okozhat a bankváltás elmaradása?

5. Foglalkozás

Mikrovezérlők fájlregisztereinek áttekintése

A mikrovezérlő teljes memóriaterülete elérhető direkt és indirekt módon. Az indirekt elérést a **F**ile **S**elect **R**egister (**FSR**) biztosítja (a továbbiakban lesz részletezve). Mint már szerepelt, a speciális funkciójú regiszterek, és a bankok használata biztosítja a PIC mikrovezérlők magas fokú kompatibilitását. Az **SFR** szerepe kettős, tartalmazza a processzor közeli regisztereket (pl. W, STATUS) és a memóriába ágyazott perifériaelemeket is. Ez a technika a személyi számítógépeknél is alkalmazott, hiszen a különböző perifériaelemeknek memóriacímük van (például videó RAM terület).

Legfontosabb SFR regiszterek

STATUS regiszter

A STATUS regiszter az ALU jelzőbitjeit, a bankválasztó biteket, valamint a CPU állapotáról tájékoztató biteket tartalmazza.

STATUS REGISTER (03h, 83h, 103h, 183h)

7.							0.
IRP	**RP1**	**RP0**	**/TO**	**/PD**	**Z**	**DC**	**C**

7. bit **IRP**: Bank választó bit (indirekt címzés esetén):
   0: Bank 0, 1 (00h – FFh);
   1: Bank 2, 3 (100h – 1FFh).

6-5. bit <**RP1:RP0**>: Bank választó bitek (direkt címzés esetén):
   00: Bank 0 (00h – 7Fh);
   01: Bank 1 (80h – FFh);
   10: Bank 2 (100h – l7Fh);
   11: Bank 3 (180h – 1FFh).
   Minden bank 128 bájtos.

4. bit **/TO**: Time-out bit:
   0: ha a WDT túlcsordul;
   1: a tápfeszültség bekapcsolásakor, `CLRWDT` vagy a `SLEEP` utasítás hatására.

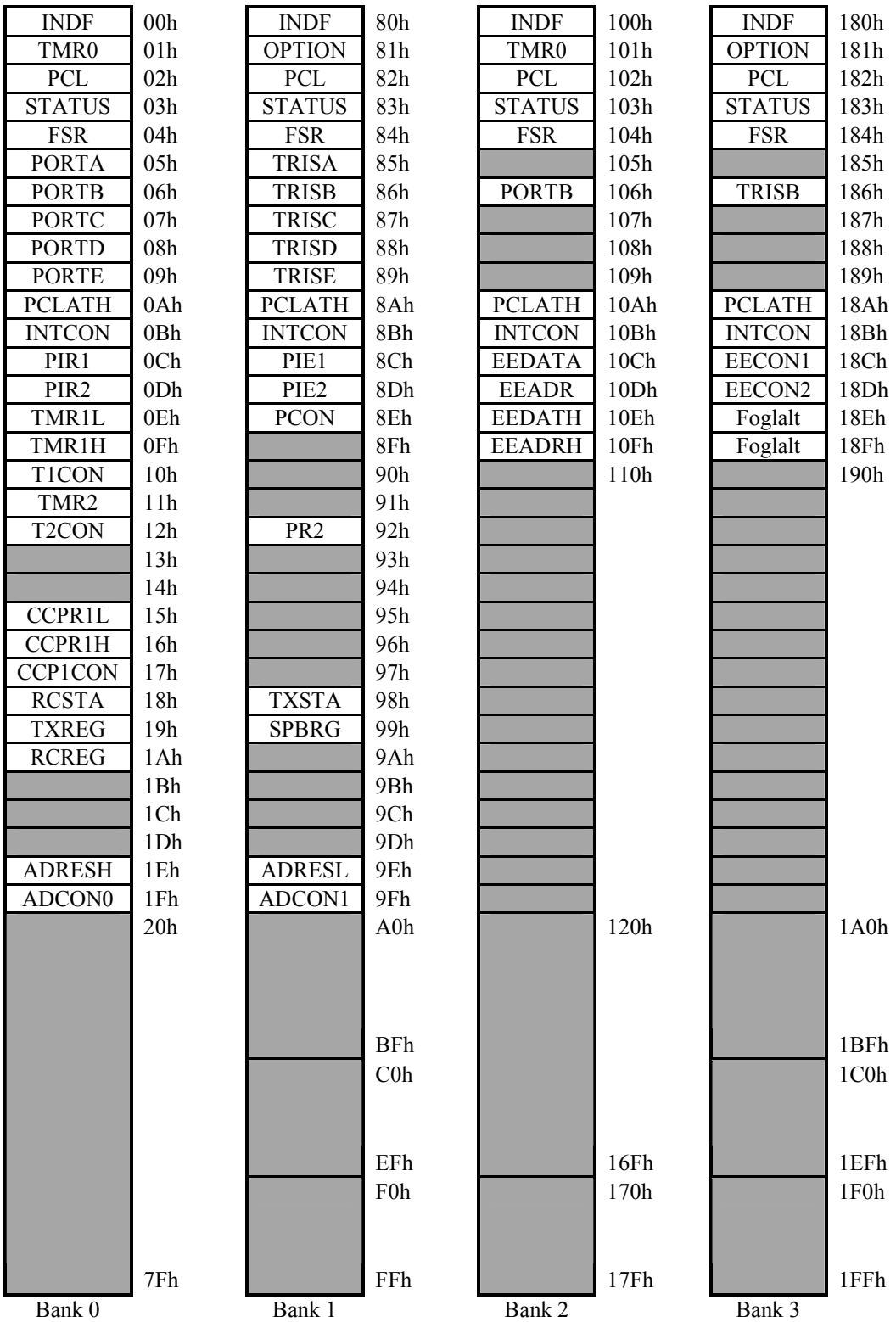
3. bit **/PD**: Power-down bit:
   0: a `SLEEP` utasítás hatására;
   1: a tápfeszültség bekapcsolása után és a `CLRWDT` utasítás hatására.

2. bit **Z**: Zéró bit:
   0: ha valamely aritmetikai vagy logikai művelet eredménye nem nulla;
   1: ha valamely aritmetikai vagy logikai művelet eredménye nulla.

1. bit **DC**: Digit Carry bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` utasítások kezelik):
   0: ha, nincs átvitel a bájt a negyedik bitjénél;
   1: ha, van átvitel a bájt a negyedik bitjénél.

0. bit **C**: Carry bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` utasítások kezelik):
   0: 0 ha nincs átvitel a legnagyobb súlyozású bitnél;
   1: 1 ha, van átvitel a legnagyobb súlyozású bitnél.

SFR regiszterek



OPTION regiszter

A mikrokontroller alapvető perifériáinak a beállítására szolgáló regiszter.

OPTION REGISTER (81h, 181h)

7.							0.
/RBPU	**INTEDG**	**T0CS**	**T0SE**	**PSA**	**PS2**	**PS1**	**PS0**

7. bit **/RBPU**: a PORTB felhúzó ellenállásait engedélyező bit:
   0: a felhúzó ellenállások engedélyezve;
   1: a felhúzó ellenállások tiltva.
6. bit **INTEDG**: külső megszakítás él választása:
   0: a megszakítás RB0/INT láb lefutó élére aktív;
   1: a megszakítás RB0/INT láb felfutó élére aktív.
5. bit **T0CS**: TIMER0 órajel forrását kiválasztó bit:
   0: belső utasításciklus lépteti TMR0-t;
   1: az RA4/T0CKI láb az órajel forrása.
4. bit **T0SE**: TIMER0 léptető élét kiválasztó bit:
   0: a Timer0 az RA4/T0CKI lábon történt felfutó élre növekszik;
   1: a Timer0 az RA4/T0CKI lábon történt lefutó élre növekszik.
3. bit **PSA**: Előosztó kiválasztó bit:
   0: az előosztó a Timer0-hoz rendelt;
   1: az előosztó a WDT-hez kapcsolódik.
2-0. bit <**PS2:PS0>**: az előosztó osztásarányát kiválasztó bitek.

PS2 PS1 PS0	Timer0	WDT
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

INTCON regiszter

A mikrokontroller a megszakításkezelésének beállításához kötődő regiszter.

INTCON REGISTER (0Bh; 8Bh, 10Bh, 18Bh)

7.							0.
GIE	**EEIE**	**T0IE**	**INTE**	**RBIE**	**T0IF**	**INTF**	**RBIF**

7. bit **GIE**: általános megszakítás engedélyező bit:
   0: megszakítások tiltva;
   1: megszakítások engedélyezve.
6. bit **EEIE**: EEPROM írás kész, megszakítás engedélyező bit:
   0: EEPROM írás kész megszakítás tiltva;
   1: EEPROM írás kész megszakítás engedélyezve.
5. bit **T0IE**: Timer0 túlcsordulás engedélyező bit:
   0: Timer0 megszakítás tiltva;
   1: Timer0 megszakítás engedélyezve.
4. bit **INTE**: RB0/INT külső megszakítás engedélyező bit:
   0: RB0/INT külső megszakítás tiltva;
   1: RB0/INT külső megszakítás engedélyezve.

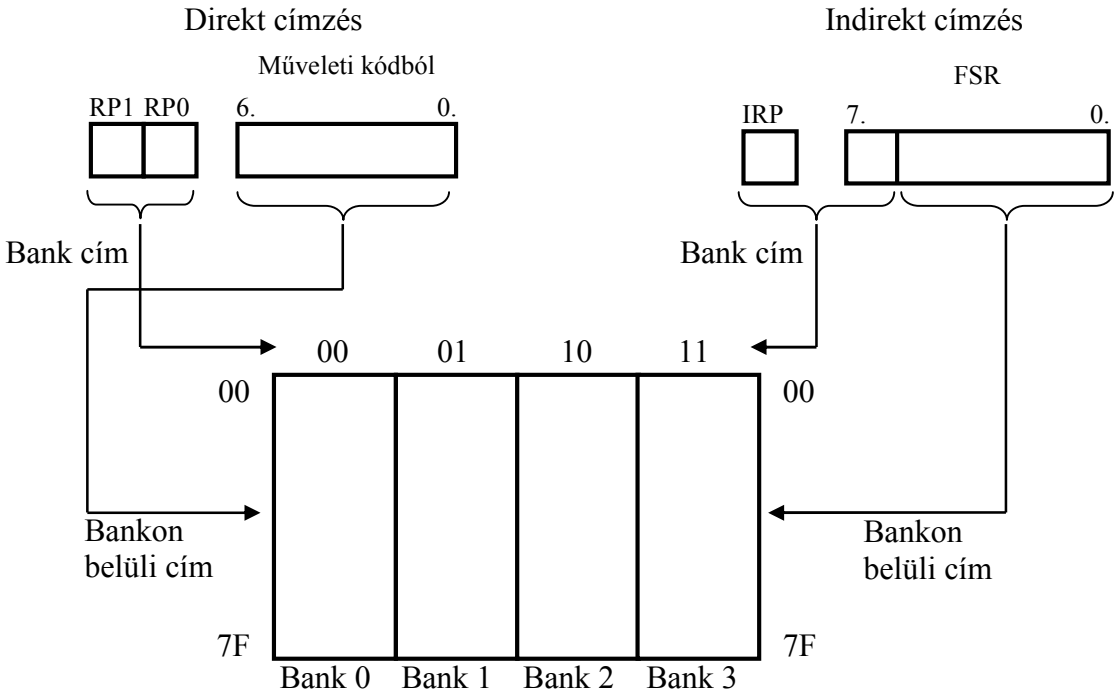
3. bit **RBIE**: a PORTB felső 4 bitjén létrejövő megszakítás engedélyező bit:
   0: PORTB változás megszakítás tiltva;
   1: PORTB változás megszakítás engedélyezve.
2. bit **T0IF**: Timer0 túlcsordulás megszakítást jelző bit:
   0: nincs túlcsordulás;
   1: Timer0 túlcsordult (szoftverből törölni kell!).
1. bit **INTF**: RB0/INT külső megszakítást jelző bit:
   0: nem volt megszakítás;
   1: RB0/INT bemenetre megszakítás érkezett (szoftverből törölni kell!).
0. bit **RBIF**: PORTB felső 4 bitjén a megszakítást jelző bit:
   0: nem volt változás;
   1: változás a lábakon (szoftverből törölni kell!).

Indirekt címzés fogalma, használata

A mikrokontrollerek kétféle címzési rendszert alkalmaznak a fájlregiszterek eléréséhez, ez a közvetlen (direkt) illetve közvetett (indirekt) címzési mód (15. ábra).

Közvetlen címzés esetén az utasítás tartalmazza a megcímzendő fájlregiszter címének alsó 7 bitjét, a megfelelő bank kiválasztását az <**RP1:RP0**> bitekkel lehet biztosítani.

Közvetett címzés esetén az utasításban a közvetlen regisztercím helyett az FSR (File Select Register) regisztert kell alkalmazni, mert ez tartalmazza a megcímzendő fájlregiszter címét. Az FSR által kiválasztott (mutatott) tárrekeszt az INDF regiszteren keresztül lehet elérni. A megfelelő lapválasztásról a 8 bites FSR tartalom és az IRP bit biztosítja mikrokontrollert.



15. ábra

Az indirekt címzés természetesen nem ismeretlen fogalom a magas szintű programozási nyelvekben sem, a C nyelv alapját képezi az indirekció és a mutatók használata. [**4.** 16-18. o]

Direkt és indirekt címzés összehasonlítása egy példa alapján

A feladat legyen az, hogy a 20h-tól 2Fh-ig terjedő fájlregiszter területet töröljünk ki, azaz minden itt található regiszter 00h értéket vegyen fel.

Direkt címzéssel

Egyesével venni kell a tárcímeket, és alkalmazni kell rájuk a már megismert fájlregiszter törlő utasítást, összesen 16-szor.

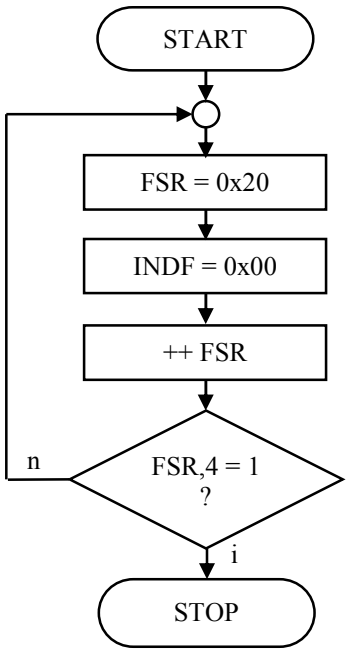
```
CLRF    0x20
CLRF    0x21
CLRF    0x22
…
CLRF    0x2E
CLRF    0x2F
```

A megoldás hátránya, hogy 16 szó programmemória területet igényel. Nagyobb tárterület ilyen módon történő törlése vagy adott értékkel való feltöltése nem ajánlott.

Indirekt címzéssel

Az indirekt címzéssel a feladat megoldására tömörebb, kevesebb tárhelyet, igénylő megoldást lehet adni. A megoldáshoz az FSR-t a törlendő memóriaterület elejére kell állítani, alkalmazni kell a regisztertörlő műveletet az INDF regiszterre, majd az FSR-t a következő törlendő regiszter címére kell állítani. Így a törlő utasítást csak egyszer kell szerepeltetni, viszont 16-szor kell lefuttatni. Természetesen a feladatot így csak valamilyen ciklusszervező utasítás segítségével lehet megoldani (16. ábra). A ciklus befejeztetéséről egy feltétel beteljesülése gondoskodhat, a példában ezt a legegyszerűbben úgy lehet megfogalmazni, hogy a ciklus addig fusson, amíg az FSR 4. bitje 1 értékűvé nem válik, mert ekkor átvitel keletkezik a 3. és a 4. bit határon. (A 16 binárisan ’B’01111, a 17 ’B’10000.)

A program folyamatábrája és kódja.



16. ábra

Az assembly kódrészlet:

```
            …
            MOVLW   0x20
            MOVWF   FSR
TOVABB      CLRF    INDF
            INCF    FSR
            BTFSS   FSR,4
            GOTO    TOVABB
FOLYTAT     …
```

I/O portok felépítése, működése

A PIC 16F871 5 általános I/O porttal rendelkezik, a legtöbb port kivezetésre igaz az, hogy többcélú kialakításúak, azaz több feladat ellátására is alkalmassá tették őket.

A port, olyan alapperiféria, amelynek alapvető feladata a bináris információ be és kivitele a kontroller és a környezete között. A kimenetre a CMOS, a bemenetekre a TTL jelszintek definiáltak. A működés előtt a port kivezetéseket konfigurálni kell az adatirány szempontjából, erre szolgálnak az adatirányt kiválasztó TRIS regiszterek. A portok megfelelő TRIS regiszterbe a kívánt bit helyre elhelyezett 1 érték az adott port kivezetést bemenetként, a 0 érték pedig kimenetként definiálja. Minden port magas áramterhelhetőséggel (25 mA) rendelkezik, így közvetlenül képes meghajtani kijelzőket, LED-eket. A mikrokontroller adatkivitel esetén egy úgynevezett latch (tároló) áramkört állít be a kívánt jelszintre, az adatbeolvasás közvetlenül a lábkivezetésről történik. Egyes portok Schmitt-triggeres (ST) bemenetként működnek, ez nagyobb zavarvédelmet biztosít a hagyományos TTL-es megoldástól.

Portok áttekintése

PORTA

Megnevezés	Puffer	Funkció
RA0/AN0	TTL	I/O vagy analóg input
RA1/AN1	TTL	I/O vagy analóg input
RA2/AN2	TTL	I/O vagy analóg input
RA3/AN3/V_{REF}	TTL	I/O vagy analóg input vagy analóg referencia
RA4/T0CKI	ST	I/O vagy külső órajel forrása a Timer0-nak, nyitott drain-ű
RA5/AN0	TTL	I/O vagy analóg input

A PORTA elsődlegesen, mint általános I/O port, illetve másodlagos funkcióként az A/D áramkör bemeneti pontjaiként működik. Egyedi minden más port kivezetéstől különböző kialakítású az RA4-es, mert ez a digitális technikában már megismert nyitott kimenettel rendelkezik, illesztése ezért különbözik a hagyományos TTL vagy CMOS áramkörökétől. Ez a port kivezetés biztosítja a Timer0 külső órajellel történő léptetését.

PORTB

Megnevezés	Puffer	Funkció
RB0/INT	TTL/ST	I/O vagy külső megszakítás
RB1	TTL	I/O vagy analóg input
RB2	TTL	I/O vagy analóg input
RB3/PGM	TTL/ST	I/O vagy alacsony feszültségű programozási mód
RB4	TTL	I/O
RB5	TTL	I/O
RB6/PGC	TTL/ST	I/O vagy soros programozás órajele
RB7/PGD	TTL/ST	I/O vagy soros programozás adat

A PORTB másodlagos funkciói megszakítási forrásként (RB0: külső megszakítás, RB <7:4>: megszakítás változás esetén) használatosak, illetve az eszköz beprogramozását teszik lehetővé (RB3 és RB <7:6>).

PORTC

Megnevezés	Puffer	Funkció
RC0/T1OSO/T1CKI	ST	I/O vagy Timer1 oszcillátor kimenete vagy órajel bemenete
RC1/T1OSI	ST	I/O vagy Timer1 oszcillátor bemenet
RC2/CCP1	ST	I/O vagy Capture1 bemenet, Compare1 kimenet. PWM1 kimenet
RC3	ST	I/O
RC4	ST	I/O
RC5	ST	I/O
RC6/TX/CK/	ST	I/O vagy UART adás, szinkron órajel
RC7/RX/DT	ST	I/O vagy UART vétel, szinkron adatjel

A PORTC másodlagos funkciói a Timer1 működtetését (RC <1:0>) a speciális CCP modul elemeit illetve az adatkommunikációt szolgálják (RC <7:6>).

PORTD

Megnevezés	Puffer	Funkció
RD0/PSP0	ST/TTL	I/O vagy párhuzamos port 0. bit
RD1/PSP1	ST/TTL	I/O vagy párhuzamos port 1. bit
RD2/PSP2	ST/TTL	I/O vagy párhuzamos port 2. bit
RD3/PSP3	ST/TTL	I/O vagy párhuzamos port 3. bit
RD4/PSP4	ST/TTL	I/O vagy párhuzamos port 4. bit
RD5/PSP5	ST/TTL	I/O vagy párhuzamos port 5. bit
RD6/PSP6	ST/TTL	I/O vagy párhuzamos port 6. bit
RD7/PSP7	ST/TTL	I/O vagy párhuzamos port 7. bit

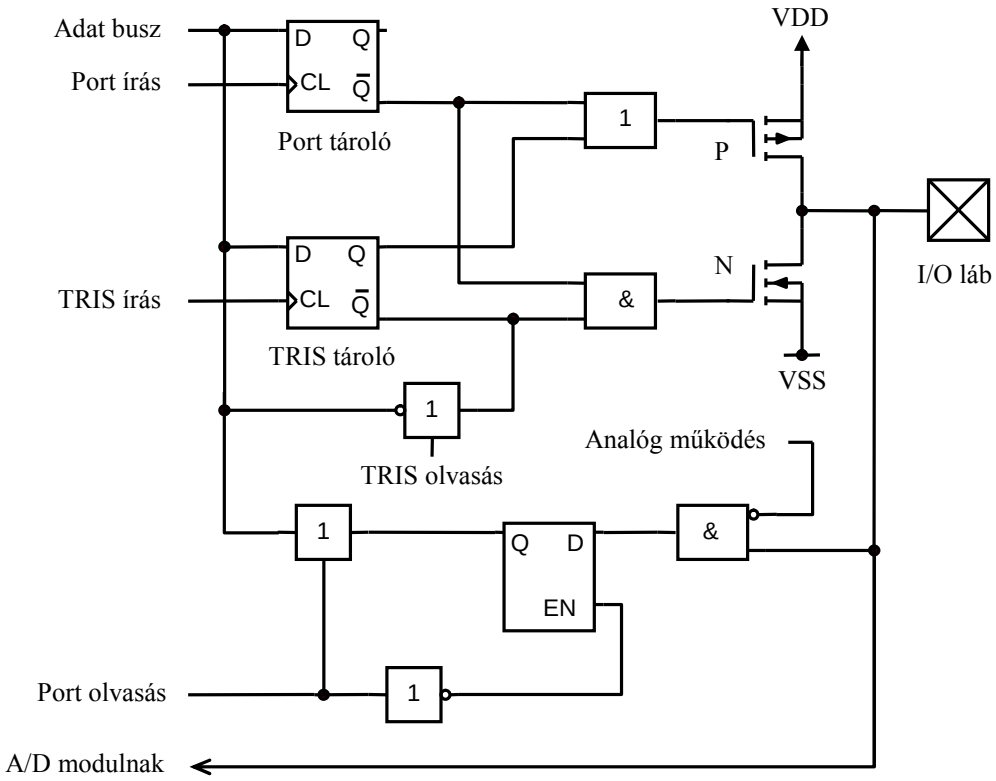
A PORTD másodlagos funkciói a kontroller memória bővítését lehetővé tevő párhuzamos port adatbitjeit tartalmazza. A párhuzamos portra csatlakozó memória elemeket a mikrovezérlő úgy képes kezelni, mint ahogy ezt egy mikroprocesszoros rendszer teszi.

PORTE

Megnevezés	Puffer	Funkció
RE0/RD/AN5	ST/TTL	I/O vagy adat olvasás jel bitje a PSP-nek vagy analóg input
RE1/WR/AN6	ST/TTL	I/O vagy adat írás jel bitje a PSP-nek vagy analóg input
RE2/CS/AN7	ST/TTL	I/O vagy eszköz kiválasztás jel bitje a PSP-nek vagy analóg input

A PORTE másodlagos funkciói a párhuzamos port vezérlő bitjeit illetve az analóg bemenet még kimaradt elemeit tartalmazza. [**4.** 33-41. o]

Egy tipikus port kialakítást (PORTA <3:0>) mutat be a következő kapcsolási rajz (17. ábra).



17. ábra

Az áramkör különösebb magyarázata nélkül jól elkülöníthetőek a különböző funkciók: a CMOS kimeneti fokozat, az adattárolók, amelyek a port a TRIS és a bemeneti logikai érték tárolását végzik el. [**4.** 33. o]

Ellenőrző kérdések

1. Adjon még példát az indirekt címzés használatára.
2. Milyen logikai jelszintekkel dolgozik a mikrokontroller?
3. Miért különbözőek a portok kialakításai?
4. Mi a szerepe a TRIS regisztereknek?

6. Foglalkozás

Számláló, időzítő egységek működése

A számlálók általános feladata a bemenetükre jutó (óra) jeleknek a megszámlálása. A számlálókat sok szempont szerint lehet csoportosítani, ebből az egyik legfontosabb a szóhosszúság, amely a számláló végértékének a nagyságát határozza meg. A mikrokontrollerek tipikusan tartalmaznak belső periféria elemként programozható számlálót, típusonként akár többet is. A számlálók a végértéküknek az eléréséről általában egy megszakítással adnak jelzést. A számláló egységekből úgy lehet időzítő egységet készíteni, hogy a számlálást egy ismert periódusidejű jelre kell elvégezni. A számlálók számlálási tartományát elő illetve utóosztóval lehet kibővíteni, így nagyobb szóhosszúsággal működhet, illetve hosszabb ideig időzíthet. Mikrokontrollerek időzítő moduljai programozhatóak, ez azt jelenti, hogy a számláló tartalma tetszőleges értékre beállítható, értéke megváltoztatható, törölhető.

Mikrokontrollerek (PIC 16F871) időzítő moduljai

- **Timer0 modul:** 8 bites számláló, működtethető belső ($F_{OSC}/4$) és külső órajellel (itt programozható az aktív él). A számláló számlálási ciklusa a WDT 8 bites programozható előosztójával bővíthető.
- **Timer1 modul:** alapban 16 bites számláló, külső és belső impulzus is léptetheti a számlálót. A Timer1 modul együttműködik a Capture (mintavevő) és a Compare (összehasonlító) modulokkal. Ez azt jelenti, hogy a számláló tartalma külső jel hatására tetszőleges időpontban kiolvasható ill. a tartalma egy másik regiszterrel összehasonlítható.
- **Timer2 modul:** 8 bites számláló programozható elő- és utóosztóval rendelkezik. A CCP modul PWM egységével képes együttműködni.
- **CCP modul:** Capture, Compare, PWM egység.
  Lehetséges üzemmódjai:
  - **16 bites kiolvasás (Capture):** külső jel hatására (CCP1(2) kivezetés) a Timer1 számláló értéke betöltődik a CCPR1(2)H ill. a CCPR1(2)L regiszterpárba. Ez a működés kitűnően alkalmazható jelek periódusidejének ill. frekvenciájának vizsgálatára.
  - **16 bites összehasonlítás (Compare):** ebben az üzemmódban a Timer1 értéke folyamatosan összehasonlításra kerül a CCPR1(2)H ill. a CCPR1(2)L regiszterpár értékével. Az egyezés megszakítási eseményt illetve A/D konverziót is elindíthat.
  - **8 (10) bites impulzusszélesség-moduláció (PWM):** PWM üzemmódban a Timer2 értéke kerül összehasonlításra egy 10 bites értékkel ill. egy 8 bites periódus regiszterrel (PR2). Az összehasonlítás eredményeként a CCP1(2) kivezetésen jelenik meg az ún. impulzus-szélesség modulált jel.

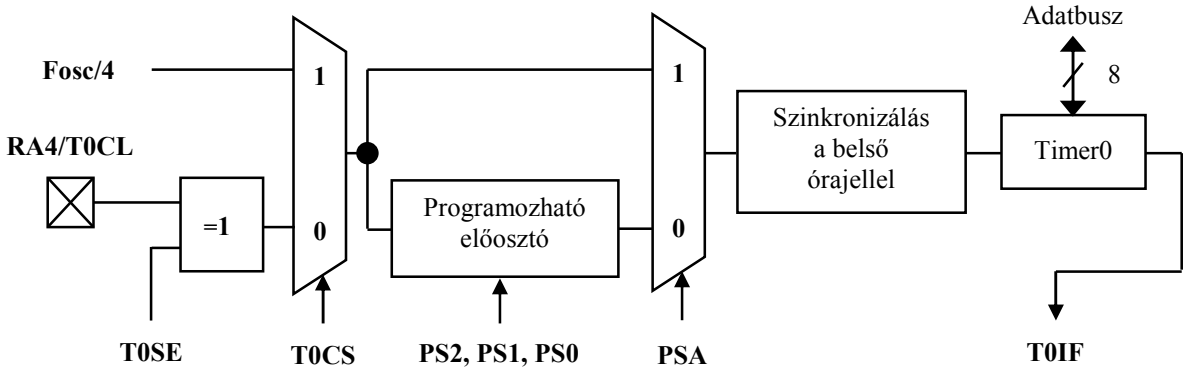
A Timer0 modul

Jellemzők:

- **Belső vagy külső órajel forrás:** (választás a T0CS bittel)
  - **Belső órajel:** a számlálót a kontroller órajel frekvenciájának a negyede léptetheti (pl. 4 MHz esetén 1 MHz);
  - **Külső órajel:** a számlálót az RA4/T0CL port lábra kötött külső impulzussorozat működteti, a T0SE konfigurációs bit segítségével lehet beállítani, hogy a külső jel felfutó ill. felfutó éle léptessen e.

- **Programozható előosztó:** a WDT előosztója a PSA bit segítségével lehet a Timer0-hoz rendelni. Az előosztó osztásarányát a PS2, PS1, PS0 bitekkel lehet konfigurálni.
- **Szinkronizáló áramkör:** általában a külső léptető impulzus frekvenciája (fázishelyzete) a kontroller órajel frekvenciájával nem egyezik meg, ez az áramkör ezt a szinkronitási problémát oldja meg.

A Timer0 egy 8 bites, közvetlenül írható és olvasható fájlregiszter (memóriába ágyazott periféria). A számláló túlcsordulása (amikor FFh-ból 00h-ba lép át) megszakítási forrása lehet a kontrollernek, ekkor a T0IF megszakítást jelző flag magas szintű lesz. Természetesen nem szükséges a megszakítási lehetőséget kihasználni, szoftverből folyamatos lekérdezéssel (polling) is lekezelhető (18. ábra). [**2.** 51. o]



18. ábra

Timer0 beállítása:

OPTION regiszter beállítása:

7.							0.
/RBPU	**INTEDG**	**T0CS**	**T0SE**	**PSA**	**PS2**	**PS1**	**PS0**
1	0	0	0	0	1	1	1

Az assembly kódrészlet:

```
…
;TIMER0 INICIALIZÁLÁSA
          CLRB    STATUS,RP0        ;BANK0-BAN VAN A TMR0
          CLRB    STATUS,RP1        ;BANK0-BAN VAN A TMR0
          CLRF    TMR0              ;TMR0 REGISZTER TÖRLÉSE
          CLRF    INTCON            ;IT-K TILTÁSA
          SETB    RP1               ;BANK1-BEN VAN AZ OPTION
          MOVLW   B'100000111'      ;TMR0 -> BELSŐ,(OSC/4) 1:256-OS ELŐOSZTÓ
          MOVWF   OPTION_REG        ;BEÁLLÍTÁSOK ELHELYEZÉSE
          BSF     INTCON,T0IE       ;TMR0 IT ENGEDÉLYEZÉS
          BSF     INTCON,GIE        ;MINDEN IT EGEDÉLYEZÉSE
…
;TIMER0 SZOFTVERES FIGYELÉSE
T0TULCS   BTFSS   INTCON,T0IF       ;VÁRAKOZÁS TÚLCSORDULÁSRA
          GOTO    T0TULCS           ;VISSZA AMIG T0 NEM CSORDUL TÚL
…
```

A Timer0 megszakításos kezeléséről a továbbiakban még lesz szó.

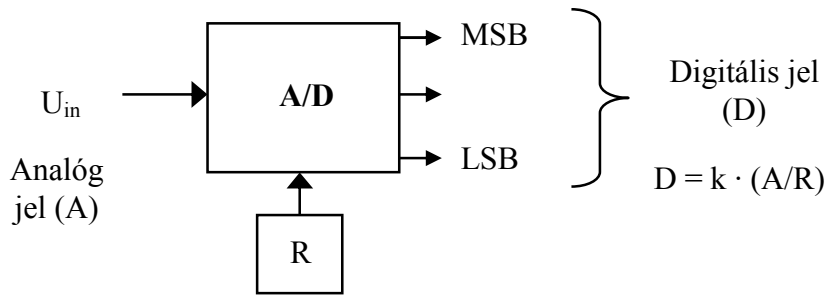
A/D átalakítók működési elve

Az analóg digitális átalakítók olyan áramkörök, melyek a bemenetükre érkező analóg jelet digitális jellé alakítják át. A használatuk nagyon fontos, hiszen a fizikai jellemzők (pl. hőmérséklet) érzékelését általában ellenállás változásra így analóg jelértékre lehet visszavezetni. Természetesen léteznek úgynevezett digitális érzékelők is, amelyek közvetlenül digitális jelértéket biztosítanak, de a belső áramköreikben szintén megjelenik az A/D átalakítás problematikája. A személyi számítógépek is rendelkeznek ezzel az áramkörrel, hiszen minden a hangkártyára bemeneti jelként érkező jelet először digitalizálni kell, hogy azt a háttértárra el lehessen menteni.

A/D átalakítás alapfogalmai

- **Kvantálás:** analóg mennyiség értékváltozási tartományának felosztása meghatározott számú egységre (kvantumra).
- **Bitszám:** az átalakítás végeredményeként kapott digitális jel szóhosszúsága (felbontása): n = 8, 10, 14.
- **MSB:** legmagasabb (legértékesebb) helyi értékű bit.
- **LSB:** legkisebb (legértéktelenebb) helyi értékű bit.
- **Megkülönböztethető állapotok száma:** a digitális jel szóhosszúságától függ, $N = 2^n$.
- **Referencia jel:** egy nagy stabilitású egyenfeszültség, amely az átalakítás pontosságát is befolyásolja.
- **Legkisebb analóg lépcső:** az a legkisebb analóg jelérték változás, amit a D/A átalakító fel tud dolgozni $\Delta A = R/2^n$.
- **Konverziós idő:** az átalakításhoz szükséges időtartam.

Blokkvázlat



19. ábra

A mikrokontrollerek A/D modulja

Legalább 8 bites szóhosszúsággal dolgoznak az átalakítási elvük általában fokozatos közelítésű (successive approximation). A konverziós idejük az órajel periódus 2-32-szerese, így nagyon gyors működést nem lehet elvárni (nagyfrekvenciás jelek mintavételezésénél sajnos külső áramkör szükséges). A mikrovezérlőkben többcsatornás A/D átalakítókat alakítanak ki, ez azt jelenti, hogy több analóg jelet tudnak szimultán kezelni.

PIC 16F871 A/D átalakítója

Az átalakító fokozatos közelítésű, 10 bites eredmény szolgáltat és nyolc független csatornával rendelkezik. Az átalakító még SLEEP üzemmódban is működik, mert belső RC oszcillátorral rendelkezik. A referencia jel lehet külső jel, illetve lehet a tápfeszültség is. A bemeneti analóg jeltartománya a V_{SS}-V_{DD} sáv lehet.

A PIC 16F871 mikrokontroller megszakítási rendszere

A PIC 16F871-es kontroller összesen 11 megszakítási forrással rendelkezik. A megszakítás globálisan tiltható (engedélyezhető) illetve minden forrás megszakítási lehetősége egyénileg is engedélyezhető.

A megszakítási források két részre oszthatóak:

1. általános megszakítás;
2. periféria megszakítás.

Megszakítási esemény általános kiszolgálási modellje:

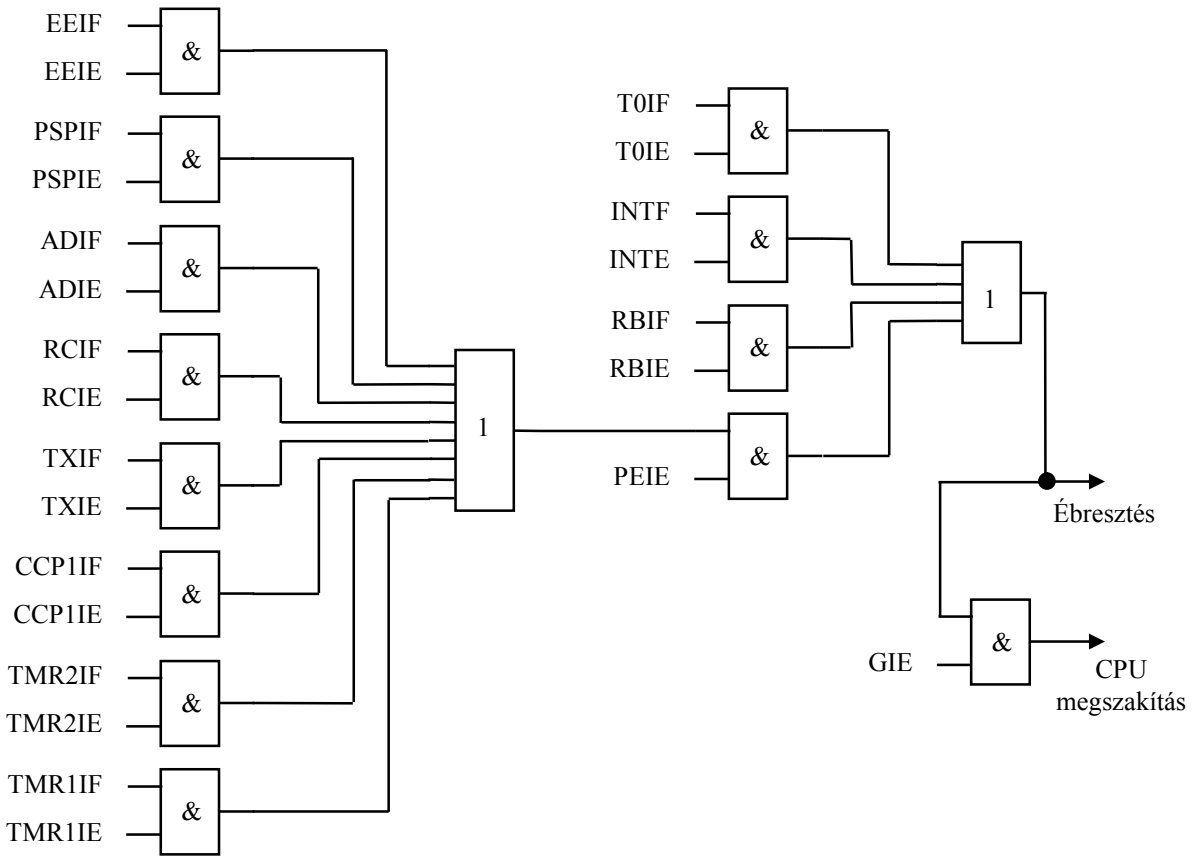
1. Megszakítások konfigurálása és engedélyezése (megfelelő xxIE = 1 ill. GIE = 1);
2. Általános feladatok (a megszakításon kívüli főprogram végrehajtása);
3. Amennyiben, valamely forrás megszakítási igényt jelent be akkor a következő folyamat zajlódik le:
   - § Az aktuálisan végrehajtás alatt álló utasítás végrehajtódik;
   - § A PC aktuális értéke a verembe mentődik;
   - § A vezérlés a 0004h megszakítási vektorcímre adódik át, innen kezdve a megszakítási rutin fog futni;
   - § Le kell tiltani a további megszakítást GIE = 0 (a megszakítás ne legyen esetleg újból megszakítható, ez problémás működéshez vezet);
   - § A megszakítást jelző flagek végigkérdezésével (pollinggal) meg kell keresni azt a megszakítási flaget, amely a megszakítás forrását jelzi (xxIF), ha csak egy van, akkor természetesen a polling elkerülhető;
   - § Lefut a megszakítási rutin érdemi része;
   - § Törölni kell a megszakítási flaget, ha ez elmarad a kontroller egy állandó megszakítási folyamatba fog ragadni (xxIF = 1);
   - § Engedélyezni kell a globális megszakítást (GIE = 1), ez el is maradhat, mert a RETFIE parancs úgyis 1-be állítja;
   - § A vezérlő visszatölti a veremből a visszatérési értékét a PC-nek.
4. Folytatódik a főprogram végrehajtása.

A megszakítások kezelésénél probléma keletkezhet:

- § Amennyiben a megszakítások kizárólagosan lekötik a programvégrehajtást, akkor a főprogram nem tudja az elvárt működést garantálni.
- § Ha a megszakítási rutin használja a munkaregisztert (W) ill. módosítja az állapotjelző biteket akkor ez a főprogramban sok előre nem látható hiba forrása lehet, ezért a megszakítási rutin elején menteni kell, a végén pedig vissza kell tölteni, a közösen használt regisztereket.

A PIC 16F871 megszakítási rendszerének teljes kapcsolási rajza (20. ábra)

A kapcsolási rajzból könnyen belátható a működés és az engedélyezési lehetőségek. A személyi számítógépek megszakítási rendszere eltér a PIC-eknél alkalmazott rendszertől. Több prioritási szint létezik és nem egy belépési pont (vektorcím) létezik, hanem több, minden forrásonként egy-egy. [**4.** 97. o]



20. ábra

Reset folyamat lezajlása mikrovezérlős környezetben

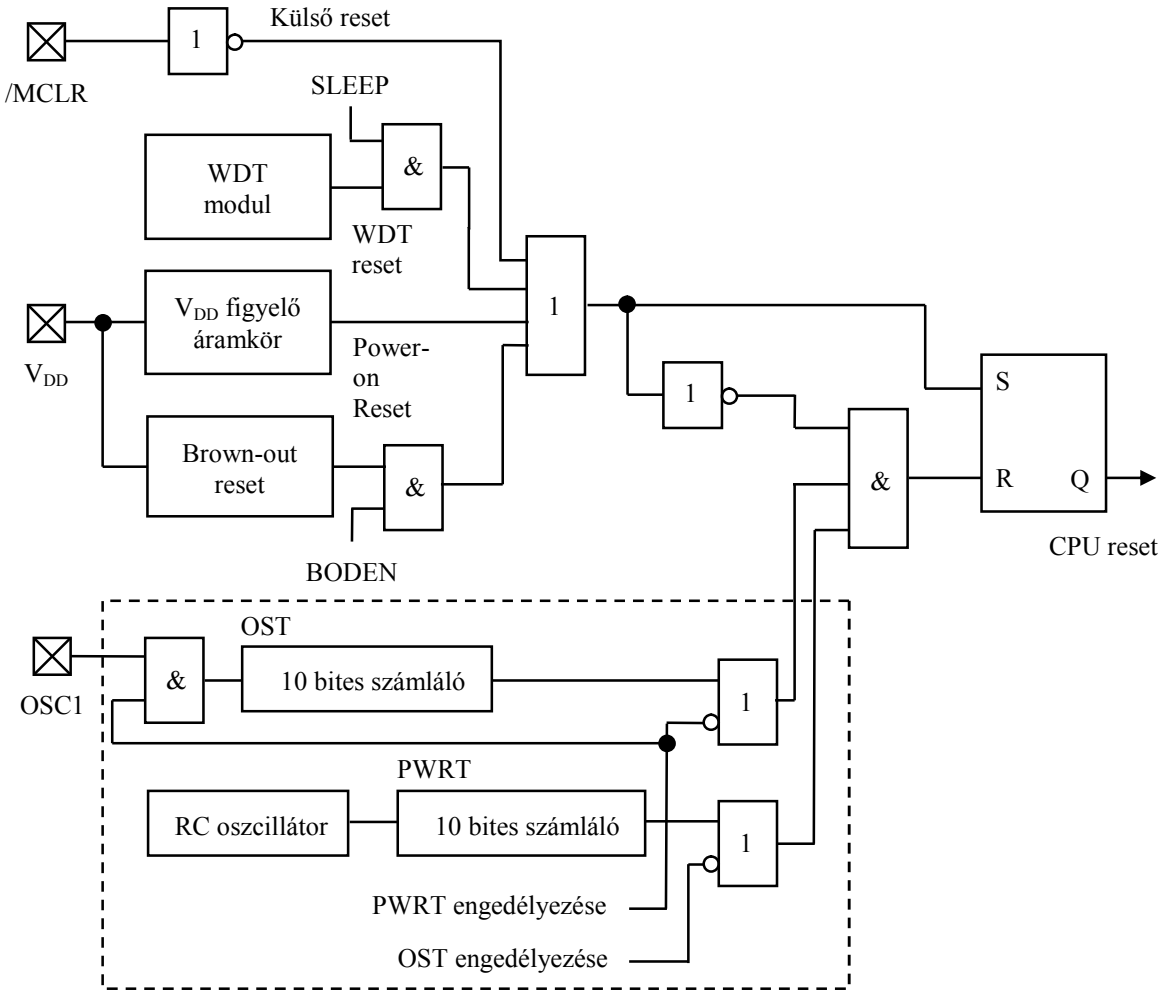
A reset a számítógépes környezetben egy olyan folyamat, amely a tápfeszültség bekapcsolásától az első utasítás végrehajtásáig tart. Feladata, a számítógép (programozható eszköz) alapbeállításainak a biztosítása. A reset folyamat a tápfeszültség mellett is lezajlódhat, ez általában valamilyen hiba következménye szokott lenni. Mikrovezérlők esetében a legfontosabb ellenőrizendő dolgok: a tápfeszültség és a stabil órajel megléte.

A PIC 16F871-es mikrovezérlő reset folyamata:

- **Külső reset:** az /MCLR kivezetésre adott 0 logikai érték váltja ki a resetet;
- **Power On Reset (POR):** tápfeszültség küszöbérték figyelő áramkör;
- **Power-up Timer: (PWRT):** általában egy 72 ms-os késleltetésű időzítő, a tápfeszültség bekapcsolási tranzienseit hidalja át;
- **Oscillator Start-up Timer (OST):** addig reseteli a CPU-t amíg az oszcillátor jele stabil nem lesz;
- **Internal reset:** a belső reset mindaddig blokkolja a mikrovezérlő működését, amíg a fenti folyamatok le nem zajlanak.

A CPU mindaddig blokkolva van, amíg a reset folyamat le nem zajlik. A belső reset csak akkor szűnik meg, ha az R-S tároló törlődik, ehhez az kell, hogy az előbbiekben ismertetett bekapcsolási folyamat sikerrel lezajlódjon, ezt az R bemenetre kötött ÉS kapu biztosítja. A kapcsolás tartalmazza a WDT áramkört is, mint resetet kiváltó belső eszközt. A WDT azaz egység, amely programhiba elleni védelmet szolgálja (21. ábra). [**4.** 91. o]

Reset logika áramköri megvalósítása



21. ábra

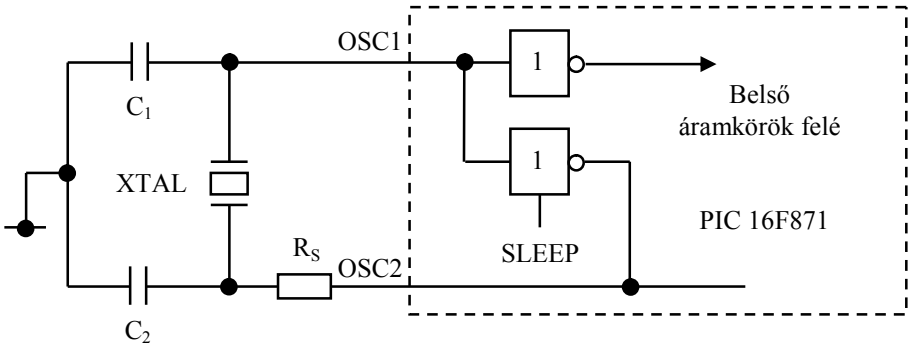
Oszcillátor konfigurációk

A számítógépek esetén az oszcillátor, azaz áramköri egység, amely előállítja a nagy frekvencia stabilitású órajelet. Az órajel nagy stabilitását általában a kvarc kristállyal működtetett oszcillátorok tudják biztosítani. Mikrokontrolleres környezetben nem minden feladat igényel nagyon pontos órajelet, ezért itt más megoldások is szóba jöhetnek.

Mikrokontrolleres órajel források:

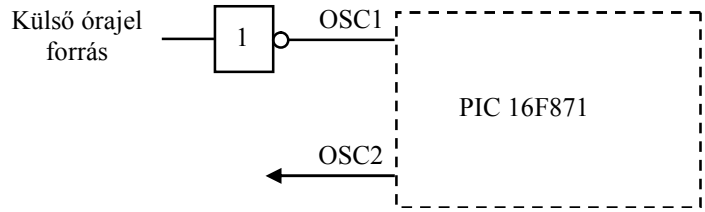
- **Kvarckristály (kerámia rezonátor):** a kontroller megfelelő kivezetéseihez kapcsolt rezgő kvarccal történik (22. ábra);
- **Külső áramkörből származó órajel:** a kontroller egy nagyobb áramköri egység része, fontos a szinkronitás a többi áramkörrel, ezért közös az órajel is (23. ábra);
- **RC oszcillátor:** igénytelen esetben, egy egyszerű RC taggal a kívánt frekvencia beállítható (24. ábra);
- **Belső órajel forrás:** nem kell külső alkatrész, a kontroller saját belső RC hálózata hozza létre az ütemező jelet.

Kvarckristály



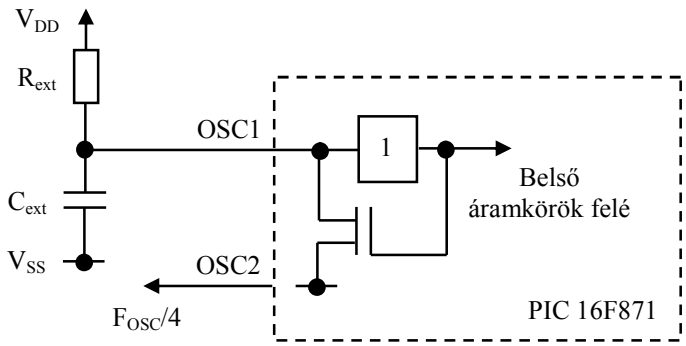
22. ábra

Külső órajel



23. ábra

RC oszcillátor



24. ábra

A PIC 16F871 négy különböző oszcillátor módban dolgozhat. A felhasználó két bit segítségével <**FOSC1:FOSC0**> tudja kiválasztani a megfelelő üzemmódot:

- **LP**: alacsony fogyasztású kvarc illetve kerámia rezonátoros oszcillátor 32-200 kHz;
- **XT**: kvarc illetve kerámia rezonátoros oszcillátor 0,5-4 MHz;
- **HS**: nagysebességű kvarc illetve kerámia rezonátoros oszcillátor 4-20 MHz;
- **RC**: RC rezonátoros.

Egyéb speciális PIC áramkör

Watchdog Timer (WDT)

A Watchdog egy olyan speciális belső egység amely, a programvégrehajtás közben fellépett hiba ellen véd, saját belső RC oszcillátora van, ami növeli a megbízhatóságát. Működésének alapja, hogy a szabadon futó WDT timer túlcsordulása resetet idéz elő, ezért a program bizonyos pontján (`CLRWDT`) gondoskodni kell a számláló törléséről. Amennyiben valamilyen hiba következtében (pl. végtelen ciklus) nem kerül sor a törlésre a kontroller megpróbálja egy resettel újraindítani a rendszert. A WDT használata ajánlott hibát nem tűrő rendszerekben.

Energiatakarékos vagy alvó (SLEEP) működés

A mikrovezérlő fogyasztását minimalizálni lehet ($P = 1\mu W$; $V_{DD} = 2V$), az úgynevezett SLEEP üzemmóddal. Ebben az üzemmódban a kontroller órajel generátora leáll, belső események megszűnnek. Alvó módban is működik a WDT ill. a megszakítási rendszer is működőképes, valamint a port lábak is vezérelnek. Az alvó módból a WDT túlcsordulás vagy megszakítás ébreszteti fel a CPU-t.

Konfigurációs szó (14 bit)

A konfigurációs szó a kontroller működését meghatározó konfigurációs biteket tartalmazza. Az elhelyezkedése a kódmemória vége a 2007h cím, a beállítása a felhasználói programban történik, az úgynevezett konfigurációs biztosítékokkal, ezért a továbbiakban nem megváltoztatható. [**4.** 88. o]

Felépítése:

13.												0.
CP1	CP0	DEBUG	-	WRT	CPD	LVP	BODEN	CP1	CP0	/PWRTE	WDTE	FOSC1	FOSC0

13-12 (5-4). bit <**CP1:CP0**>: Programmemória kódvédelmének beállítása:
00: Kódvédelem bekapcsolva;
01: Nem támogatott beállítás;
10: Nem támogatott beállítás;
11: Kódvédelem kikapcsolva.

11. bit **DEBUG**: Áramkörben történő DEBUG beállítása:
0: DEBUG kikapcsolva, RB6 és az RB7 normál I/O kivezetésként működik;
1: DEBUG bekapcsolva, az RB6 és az RB7 lábakon keresztül lehet a programvégrehajtást nyomon követni.

10. bit Nincs létrehozva, olvasva 1-et ad.

9. bit **WRT**: programmemória írásának engedélyezése:
0: A nem védett kódmemória nem írható az EECON segítségével;
1: A nem védett kódmemória írható az EECON regiszter segítségével.

8. bit **CPD**: Adat EEPROM kódvédelme:
0: A kódvédelem bekapcsolva;
1: Nincs kódvédelem.

7. bit **LVP**: Alacsony feszültségű soros programozás engedélyezése:
0: RB3 digitális I/O kivezetés, a magas feszültségű programozás a /MCLR lábon keresztül történik;
1: Az RB3 mint, PGM kivezetés működik, alacsony feszültségű programozás bekapcsolva.

6. bit **BODEN**: Brown-out reset engedélyezése:
0: BOR tiltva;
1: BOR engedélyezve.

3. bit **/PWRTE**: Power-up Timer engedélyezése:
0: PWRT tiltva;
1: PWRT engedélyezve.

2. bit **WDTE**: Watchdog timer engedélyezése:
0: WDT tiltva;
1: WDT engedélyezve.

1-0. bit <**FOSC1:FOSC0**>: Oszcillátor konfiguráció beállítása:
00: LP oszcillátor;
01: XT oszcillátor;
10: HS oszcillátor;
11: RC oszcillátor.

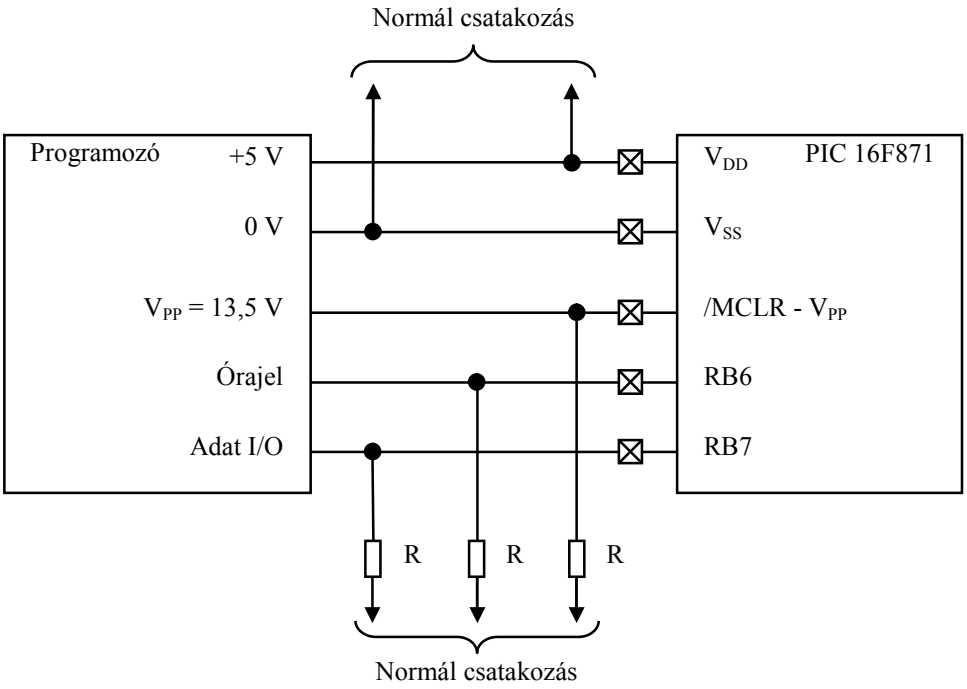
Áramkörön belüli programozás

A PIC beprogramozásához (a kódmemória programmal való feltöltéséhez) szükséges:

1. Megfelelő hardver, amelyben a kontrollert be lehet programozni;
2. Programozó (égető) szoftver.

A programozás végrehajtható a célkörnyezetben is, ezt az áramkörön belül történő (in-circuit) programozásnak nevezik.

Áramköri felépítés (nagyfeszültségű programozás esetén)



25. ábra

Ellenőrző kérdések

1. Hasonlítsa össze a különböző Timer modulokat.
2. Milyen lehet a PWM modul kimeneti jele, mire alkalmazhatják?
3. Ha az `OPTION_REG = 0x94`, akkor milyen üzemmódban működik a Timer0?
4. Miért kell a megszakítási jelzőbitet törölni?

7. Foglalkozás

Programkészítési módszerek

Mikrokontrolleres fejlesztések esetén a hardver és a szoftver megtervezése szorosan összefügg, nem lehet különválasztani egymástól, mert egy mikrovezérlős alkalmazásfejlesztés a legtöbbször valamilyen speciális feladatot ellátó céleszköz létrehozására irányul.

A programozás nem más, mint problémamegoldás, mert a programnak az eddig ismeretlen dolgokat kell az ismert tudásanyagból meghatározni. A program az ismert és az ismeretlen között teremt kapcsolatot. A program lényegi része az algoritmus, amely véges számú lépésből álló tevékenységsorozat, amely mindig a probléma megoldás felé viszi programot. Az algoritmus további részekre bontható: adatbevitelre, adatkivitelre, számítási- és döntési folyamatokra.

Programtervezési módszerek

Strukturált programozás

A legjobban ismert programtervezési eljárás. A programokat három alapvető szerkezetből lehet felépíteni:

1. **Sorrendi (szekvenciális) szerkezet**: utasítások szorosan vett egymásutánisága;
2. **Feltételes (szelekciós) szerkezet**: a program bizonyos pontjain elhelyezett döntési pontok határozzák meg a program folytatásának a helyét;
3. **Ismétlődéses (iterációs) szerkezet**: bizonyos utasításoknak a megismétlését jelenti, ez lehet feltételtől függő vagy előírt lépésszámú.

A strukturált programozás száműzi a feltétel nélküli vezérlésátadást (GOTO), hiszen bizonyítható, hogy a fenti szerkezetek használatával bármilyen program elkészíthető. A strukturált program műveleteit könnyű követni, a programban hibát keresni. Az egyszerű szerkezetekből történő építkezés miatt a bonyolultabb szerkezetek nehezebben megvalósíthatóak, komoly programozási tudást igényel.

Moduláris programozás

A programot több elemi részre (modulra) kell felbontani. Az egyes részek funkciója jól behatárolt, minden modulnak van egy a többi modullal kapcsolatot teremtő interfésze, amelyen keresztül kommunikálnak. A modulok természetesen lehetnek strukturált felépítésűek. A modulrendszerű építkezésnek sok előnye van: egy modul többször is felhasználhatóvá válik, a modulokból összeálló programot több programozó is fejlesztheti (team munka).

Felülről-lefelé történő építkezés elve (top-down)

A program elkészítése során először a program vázát kell elkészíteni, az egyes pontokon kialakítandó struktúrákat csak jelölni kell. A tervezés következő lépése, a kijelölt részfeladatok megoldása és az esetleges újabb részfeladatok kijelölése. A lépések finomításával előáll a program. A tervezés erősen feladatorientált, ezért az így megírt program részeinek a felhasználhatósága és hatékonysága is igen kérdéses. [**1.** 110. o]

Programozási lehetőségek

A programfejlesztésre három lehetséges út kínálkozik

1. **Gépi kód:** a processzor számára szóló bináris utasítássorozat, a legelemibb programozási lehetőség. Az utasítás minden esetben műveleti kódból és operandusból épül fel. Nagyon hatékony kódok hozhatók létre, azonban a programírás nagyon nehéz és bonyolult feladatot jelent a programozó számára. A processzort a saját gépi kódú utasításaival lehet programozni, ezért minden processzor esetén meg kell tanulni

(ismerni) az utasításkészletet. Egyszerű és hatékony programok létrehozására használható, ahol nagyon fontos tényező a futásidő és a tárkapacitás.

2. **Magas szintű programozási nyelv:** a magas szintű programozási nyelven történő programfejlesztés egyszerű, a programozandó processzortól függetlenül azonos módon történik az egyes rendszerekben. A nyelv az eszközei, mögé elrejti a programozandó eszközt. A programozási nyelveknek, mikrokontrolleres fejlesztésekben történő alkalmazását mindig meg kell fontolni, mert bizonyos feladatok elvégzéséhez (pl. lebegőpontos aritmetika) ideális lehet. Az így elkészített programok általános jellemzője a tárterület pazarló felhasználása, a kis hatékonyságú és nagy futásidejű program. Minden kontrolleres rendszerre létezik legalább egy olyan magas szintű nyelv, amellyel program fejleszthető az adott eszközre (BASIC, C).
3. **Assembly nyelv:** a fenti két programozási lehetőséget foglalja magába, hiszen gépközelinek mondható és számos eszközrendszert biztosít a programozó számára. Az assembly a legegyszerűbb programozási nyelvnek tekinthető. Az utasítás a gépi kódhoz hasonlóan itt is műveleti kódból és operandusból épül fel, azonban a bináris alak helyett egy az emberhez közelebb álló funkcióra utaló kódnévvel (mnemonik – emlékeztető) történik a programírás.

Az operandusok által betölthető szerepek:

- **Állandók:** olyan konstansok, amelyek értéke a program futása alatt nem változik (π, e, g);
- **Adat regiszter címek:** olyan elemek, amelyeknek a tartalmával műveletet lehet végezni;
- **Program memória címek:** a program struktúrájának kialakításához fontos operandus, a program végrehajtási sorrendet lehet a segítségével definiálni (ugró utasítás).

Az assembly programban történő hivatkozásokat az azonosítók használata biztosítja. Az azonosítók szerepe, hogy a programozó a saját programozói eszközeit megnevezze vele és, ezután ha szükséges, a program tetszőleges részéről hivatkozzon rá [**3.** 19. o].

Azonosítók csoportosítása:

- **Foglalt (fenntartott) azonosító:** utasítások, direktívák (fordítónak szóló utasítások), SFR regiszterek vannak megnevezve, ezeket megváltoztatni (újradefiniálni) nem szabad és nem is érdemes;
- **Felhasználói azonosító:** olyan azonosító, amelynek a felhasználó tulajdonít jelentőséget, ilyenek a GPR regiszter elnevezések, címkék.

Fordítóprogramok

A fordítóprogramok a magas szintű programozási nyelven megírt forrásprogramot gépi kódúvá alakítja, hogy a processzor azt végre tudja hajtani. A forrásállomány a legtöbb esetben, egy egyszerű szöveges állomány, amely bármely általánosan használt szövegszerkesztővel elkészíthető. Az assembly forráskódot gépi kóddá alakító folyamatot assemblálásnak nevezik. Az assemblálás folyamán a programban használt azonosítók, assembly utasítások és a fordítónak szóló utasítások, az úgynevezett direktívák kerülnek értelmezésre.

Az assembly nyelv

Az assembly nyelvben megírt programoknak hasonlatosan a magas szintű programozási nyelvekhez, két fontos feltételnek kell eleget tenniük:

1. Szintaktikai helyesség: a program a nyelv szabályainak megfelelően épüljön fel;
2. Szemantikai helyesség: a program helyesen működjön, a problémát oldja meg.

A szintaktikai helyesség betartásában a fordítóprogram hiba és figyelmeztető jelzésekkel segíti a programozót.

Assembly program felépítése

A forrásállomány sorokból áll, minden sornak kötött szerkezete van. Egy program sor a következő négy nyelvi elemből építkezhet (nem kell feltétlenül mindegyiket tartalmaznia):

1. **Címke:** a programsor megnevezésére szolgáló azonosító, amelyet a programozás során szükséges vezérlésátadó utasításoknál lehet felhasználni;
2. **Utasítás:** utasítások mnemonikjai és a direktívák találhatóak;
3. **Operandus:** azok a program elemek, amelyekre az utasítás vonatkozik, a számuk egy programsorban lehet 0, 1 vagy akár 2 is. Az operandusokat általában vessző választja el egymástól;
4. **Megjegyzés:** a programozó a programsorokat magyarázó, szöveget helyezhet el, az aktuálisan írt program még jól átlátható, azonban pár hét múltán már nem biztos. A megjegyzést a program többi részétől általában valamilyen jel, a legtöbbször a pontosvessző választja el.

A jól átlátható felépítés miatt érdemes a programsorokban szereplő elemeket tabulátorral, táblázatos rendszerben létrehozni:

Pl.

```
NULLA   MOVWF   ADAT     ;az ADAT regiszterbe betolti a W-t
```

Az assembler működése

Az assemblálás több lépésben végrehajtódó folyamat. Először elemzésre kerülnek az állandó és a felhasználói azonosítók, ezek egy-egy táblázatba kerülnek, amely tartalmazza a címüket és az elhelyezkedésüket. A következő lépésben a forrásprogram szemantikai és szintaktikai ellenőrzése után, amennyiben nem lép fel hiba, létrejön a tárgykód. Az elkövetett hibákról a programozó jelzést kap. A tárgykód mellett létrejöhet egy úgynevezett fordítási lista is, amely nem más, mint a programfordítás közbeni információkkal kibővített forrásprogram. Az egyes rendszereknek általában létezik disassemblere is, amely segítségével a tárgykód visszaalakítható forrásállománnyá. [**1.** 116-117. o]

MPASM assembler

Az MPASM a Microchip mikrokontrolleres assemblere, amely egy számítógépes program és tetszőleges IBM kompatibilis számítógépen futtatható.

Az MPASM jellemzői:

- Univerzális, minden Microchip által gyártott eszközhöz használható: PIC 14/16/17/18 típusok;
- Windows alatt futtatható, ingyenes program;
- Folyamatos programfrissítéssel rendelkezik, az újabb eszközök támogatásával;
- Támogatja a forrásprogramba beilleszthető forrás állományokat tartalmazó include fájlok használatát;
- Létrehozhatóak makrók;
- A fordítási folyamatot feltételekhez lehet kötni;
- A programfejlesztés a mai programnyelvekhez hasonlóan projekt szemléletben történik.

MPASM direktívák

A direktíva egy előzetes fordítási utasítás, amely nem a kontrollernek, hanem a fordítóprogramnak szól és segítségével a fordítási folyamatot, lehet irányítani. [**1.** 119. o]

Fontosabb direktívák

Vezérlő és definíciós direktívák		
Megnevezés	Funkció	Szintaxis
#DEFINE	Szöveget helyettesítő címke létrehozása	#DEFINE <név> [argumentum]
END	Program végét jelzi	END
EQU	Állandó (konstans) létrehozása	<címke> EQU <kifejezés>
INCLUDE	Forrásfájl beillesztése	INCLUDE <fájlnév>
ERRORLEVEL	Hibaszint beállítása	ERRORLEVEL 0|1|2 <+-> <üzenet>
LIST	Lista opciók állítására	LIST [<opció>, […,<opció>]]
ORG	Következő utasítás helyét határozza meg	<címke> ORG <kifejezés>
SET	Assembler változó létrehozása (újradefiniálható)	<címke> SET <kifejezés>
Adat direktívák		
CBLOCK	Adat definíciós blokk kezdete	CBLOCK [<kifejezés>]
ENDC	Adat definíciós blokk vége	ENDC
__CONFIG	Konfigurációs bitek beállítására	__CONFIG <kifejezés>
DB	Bájt definiálása	DB <kifejezés>[…,<kifejezés>]
DT	Táblázat létrehozása	DT <kifejezés>[…,<kifejezés>]
__IDLOCS	Azonosító definiálása	__IDLOCS <kifejezés>
Bankváltó és programmodul kezelő direktívák		
BANKSEL	Direkt RAM címhez, bankválasztás	BANKSEL <címke>
BANKISEL	Indirekt RAM címhez, bankválasztás	BANKISEL <címke>
EXTERN	Külső címke létrehozása	EXTERN <címke> [, <címke>]
GLOBAL	Más modulban deklarált címke	GLOBAL <címke> [, <címke>]
Makró direktívák		
MACRO	Makró definíció kezdete	<címke> MACRO [<arg>,…,<arg>]
EXITM	Kilépés a makróból	EXITM
ENDM	Makró definíció vége	ENDM
LOCAL	Makróban helyi változó létrehozása	LOCAL <címke> [,<címke>]

Példák a direktívák használatára:

#DEFINE

```
  #DEFINE   ON      BE      ;az ON-t a BE helyettesíti
  #DEFINE   PORTC,0 LED0    ;a PORTC,0-át a LED0 helyettesíti
```

EQU

```
  VEGERT    EQU     0x37    ;a VEGERET címke 37h értékű lesz
  ADAT1     EQU     0x20    ;a 20h című GPR ADAT1 lesz
```

SET

```
  KEZDO     SET     0x42    ;a KEZDO 42h értékű lesz
  …
  KEZDO     SET     KEZDO+5 ;a KEZDO új értéke 47h
```

ORG

```
  ORG       0x04    ;a következő utasítás a 4h címen lesz
```

END

```
            …
            END             ;eddig fordít
```

LIST

Opció	Alapérték	Funkció
F = <format>	INHX8M	Kimeneti hexa fájl formátuma
P = <type>	nincs	Processzortípus
R = <radix>	HEX	Számformátum

```
LIST      P = 16F871, R = DEC
```

INCLUDE

```
#INCLUDE  <P16F871.INC>   ;az SFR regisztereket tárolja
```

CBLOCK

```
CBLOCK    0x20    ;definíció a 20H című regiszternél kezdődik
          AREG    ;AREG, a 20H című GPR
          BREG    ;BREG, a 21H című GPR
          CREG    ;CREG, a 22H című GPR
ENDC              ;az adat definíciós blokk vége
```

CONFIG

```
__CONFIG_XT_OSC & _WDT_OFF & _CP_ON __IDLOCS 1234
```

DT

```
DT    „0123”  ; azonos a RETLW 0,…, RETLW 3 utasításokkal
```

BANKSEL

```
BANKSEL   TRISB   ;Bank1 kiválasztása
CLRF      TRISB   ;PORTB minden bitje kimenet
BANKSEL   PORTB   ;Bank0 kiválasztása
CLRF      PORTB   ;Minden PORTB bit 0 értékű
```

A makrók használata

A makró egy előre definiált utasítássorozat, amely névvel rendelkezik és akár argumentumokkal is ellátható. A programban tetszőleges helyen, és tetszőleges számban meghívható. A makrózás az assembly programozás során gyakran használt eszköz, mert nagyon sokszor függvényt helyettesítő funkcióval bír. A makrók használata átláthatóvá teszi a programszerkezetet, a gyakran használt makrók akár külön include fájlba is rendezhetőek. Megjegyzendő, hogy a makró a jó tulajdonságain kívül sajnos jelentős tárterületet igényel, hiszen minden hívásakor a program adott helyére befordítódik.

Pl.:

Deklaráció:

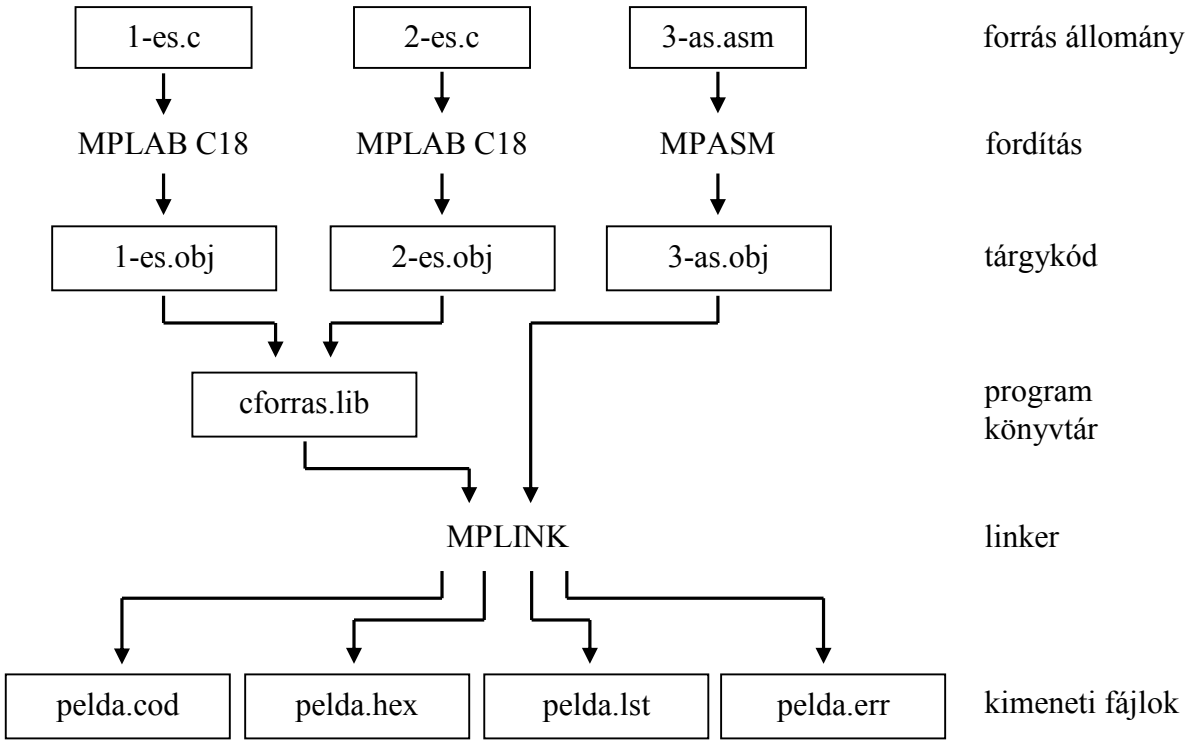
```
LED_BE    MACRO   ;LED_BE makró definíció
BSF       PORTA,1 ;PORTA 1-es bitjére kötött LED világít
ENDM              ;makró vége
```

Használat:

```
…
LED_BE            ;LED világít
…
```

A programmodulok használata

Az assemblálás közben létrejövő object tárgykód segítségével, lehetőség nyílik többmodulos program írására is. Ez azt jelenti, hogy akár a különböző fordítóprogramokkal és nyelven megírt kódok egy közös futtatható állományba szerkeszthetőek. A tárgykódok összeillesztését a linker program végzi el (26. ábra). A Microchip linker programja az MPLINK, ez minden fordítási esemény esetén meghívásra kerül. A modulok kapcsolódási pontjai az EXTERN és a GLOBAL címkék használatával történik. [**1.** 132. o]



26. ábra

Az Intel hexa formátum

A programozható eszközökbe (ROM, EEPROM) az információ, az úgynevezett Intel hexa formátumban történik elhelyezésre. A fájl egy ASCII kódolású, ezért platform független, egyszerű szöveges állomány, minden szövegszerkesztővel olvasható, módosítható. Az állomány az adatokat rekordos formátumban tárolja. Az MPLINK kimeneti állományában található „hex" kiterjesztésű fájl is ilyen formátumú. A 16-os sorozatú PIC-ek az INHX8, a 18-as sorozata az INHX32-es formátumot használja. [**1.** 133. o]

A INXH formátum általános felépítése:

:HHCCCCTTDD…DDEE

Ahol:

- **:** rekordelválasztó mező;
- **HH**: rekordhossz mező; két karakteren ábrázolt;
- **CCCC**: betöltési cím mező; négy karakteres és megadja, hogy az adatbájtot milyen címtől kell betölteni;
- **TT**: típusmező; adatrekord esetén 00, fájlvége esetén 01 az értéke;
- **DD**: adatmező; az adatokat ASCII karakterekre konvertálva tárolja, a magasabb helyi értékű szerepel először;
- **EE**: ellenőrzőösszeg-mező; a 2-5. mezők hexadecimális összegének 8 bitre csonkított összegének kettes komplemense ASCII formátumban.

Példa Intel hexa formátumra:

```
:020000040000FA
:0A0006000428831603138701 8312F8
:0800100003130714071009286F
:00000001FF
```

Programdokumentálás

Az assembly nyelven megírt programokra fokozottan érvényes a megfelelő programdokumentáció elkészítése, hiszen a programokban alkalmazott különböző apró fogások, trükkök hamar kiesnek a programozó gondolatvilágából. A megjegyzések használata ezért minden programozói szinten elvárt tevékenység. [**1.** 135. o]

A megfelelő programdokumentáció biztosítja:

- A program későbbi továbbfejlesztési lehetőségét;
- Azt, hogy más programozók könnyen tudják a program gondolatmenetét követni (team munka);
- Az egyszerű hibakeresést.

Általános tanácsok a programok írására:

- A programok legyenek megtervezve, a programnak legyen folyamatábrája;
- Világos, strukturált, a lehető legegyszerűbb felépítés;
- A lehető legkevesebb feltétel nélküli vezérlésátadó utasítás használata (GOTO);
- A programban használt címkék, változók tükrözzék a funkciójukat („beszélő” nevek);
- Legyen egy olyan mintafájl, amely szerkezetileg alapot szolgáltat a további programok elkészítéséhez.

Ellenőrző kérdések

1. Hasonlítsa össze a programtervezési módszereket.
2. Milyen hasonlóságokat fedezett fel az MPASM és az Ön által már ismert programnyelvek között?
3. Az informatikai tanulmányaiban hol találkozott még a makró fogalmával?
4. Mit tartalmazzon a mintafájl?

8. Foglalkozás

Változó deklaráció, értékadás

Deklarációt a név és cím összerendelést, a legegyszerűbben az EQU direktíva segítségével lehet elvégezni.

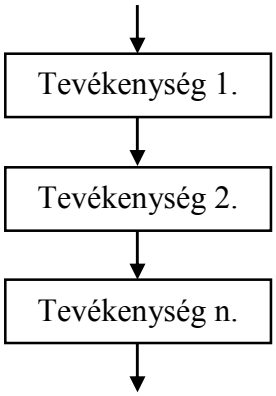
```
AREG   EQU  0x20  ;AREG, a 20H című GPR
BREG   EQU  0x21  ;BREG, a 21H című GPR
```

A deklarált változónak értéket adni, az adatmozgató utasítások segítségével lehet.

```
MOVLW  0x12       ;a W-be 12h kerül
MOVWF  AREG       ;a W-ből az AREG-be átkerül a 12h
```

Szekvencialitás

Az utasítások kötött sorrendiségét jelenti, a kódban nincs elágazás (27. ábra).



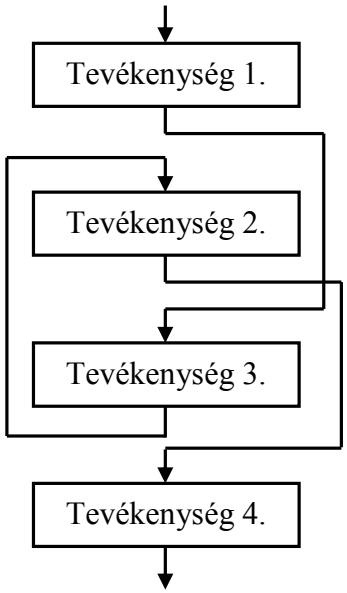
27. ábra

Kódrészlet:

```
…
CLRW                ;W = 0
SETB    PORTC,0     ;PORTC,0 = 1
CLRF    PORTB       ;PORTB = 0
…
```

Feltétel nélküli vezérlésátadás

Feltétel nélküli vezérlésátadás: a vezérlés az utasításban megadott címkén folytatódik, a strukturált programfelépítés nem támogatja a használatát, assembly nyelvben azonban megkerülhetetlen a használata (28. ábra). [**3.** 57. o]



28. ábra

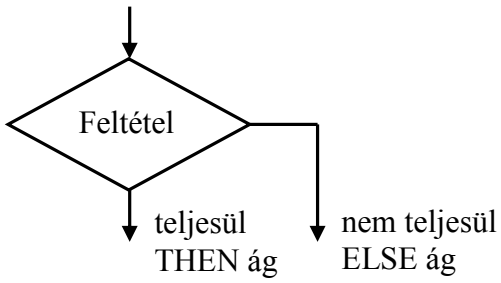
Kódrészlet:

```
…
   TEV_1   MOVLW    0x12
   GOTO    TEV_3
   TEV_2   MOVWF    AREG
   GOTO    TEV_4
   TEV_3   CLRW
   GOTO    TEV_2
   TEV_4   SETB     PORTC,0
…
```

Elágaztató szerkezetek

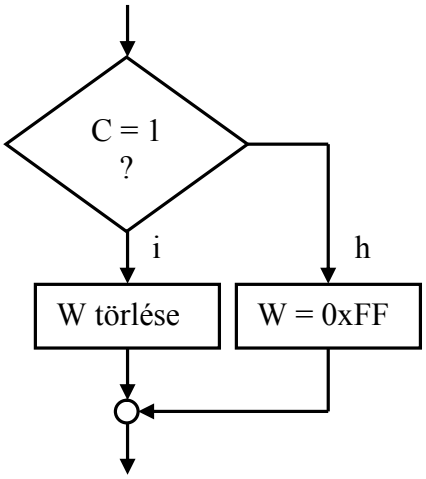
Kétirányú elágaztató utasítás (feltételes utasítás), arra szolgál, hogy a program egy adott pontján két tevékenység közül választani lehessen. [**3.** 58. o]

Általános alakja: IF feltétel THEN tevékenység ELSE tevékenység. A feltétel IGAZ értéke mellett a THEN utáni tevékenységsorozat, HAMIS érték mellett pedig az ELSE utáni fog végrehajtódni (29. ábra).



29. ábra

Kontrolleres környezetben a legegyszerűbben a bit állapottól függő elágaztatást lehet megvalósítani. A BTFSC és BTFSS bittesztelő utasítások, egy vagy két utasításciklus alatt végrehajtják az elágazást. A BTFSC (**B**it **T**est **F**ile-register, **S**kip if **C**lear) az utasításban megjelölt regiszter kiválasztott bitjét vizsgálja, ha az "0" értékű, akkor átugorja a soron következő utasítást. A BTFSS (**B**it **T**est **F**ile-register, **S**kip if **S**et) utasítás ugyanezt teszi, azzal a különbséggel, hogy "1" értékre tesztel (30. ábra).



30. ábra

Kódrészlet:

```
…
            BTFSS     STATUS,0 ;C = 0?
            GOTO      HAMIS    ;C = 0
IGAZ        CLRW               ;C = 1
            GOTO      TOVABB
HAMIS       MOVLW     0xFF
TOVABB      …
…
```

Ciklusszervezési lehetőségek

A ciklusszervező utasítások teszik lehetővé azt, hogy a program egy adott pontján egy tevékenységet tetszőleges számban meg lehessen ismételni. A ciklus általában három részből, fejből, magból és végből áll. Az ismétlődésre vonatkozó információk a fejben vagy a végben szerepelnek, a mag az ismétlendő tevékenységet tartalmazza. [**3.** 62. o]

Feltételes ciklus

Az ismétlődés egy feltétel igaz, vagy hamis értéke határozza meg. A feltételes ciklus a feltétel helyétől függően lehet kezdő vagy végfeltételes (31. ábra).

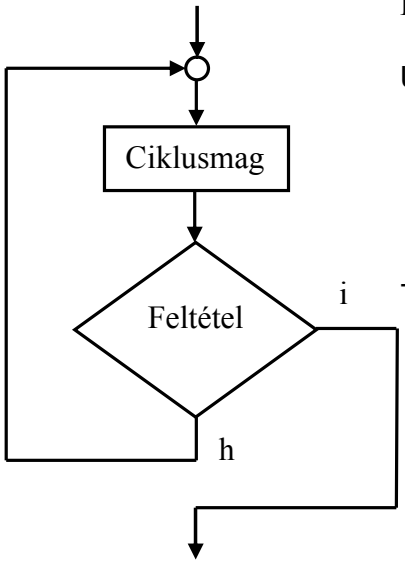
Kezdőfeltételes (elöltesztelő) ciklus: WHILE (feltétel) tevékenység.

A ciklus a feltétel tesztelésével kezdődik, ha igaz végrehajtódik a ciklusmag, majd újra kiértékelődik a feltétel, ez mindaddig tart, amíg a feltétel hamissá nem válik. A feltételt befolyásoló utasításoknak a magban kell szerepelniük. A kezdőfeltételes ciklus a feltételtől függően lehet üres, ha feltétel egyszer sem teljesül, és lehet végtelen is, ha a feltétel mindig igaz.

Végfeltételes (hátultesztelő) ciklus: DO tevékenység WHILE (feltétel).

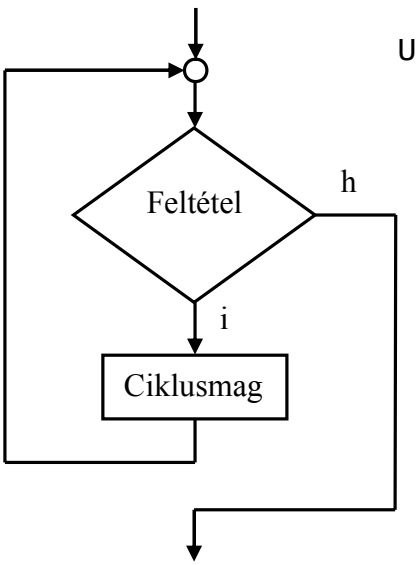
Először végrehajtódik a ciklusmag, majd a feltétel tesztelésével folytatódik, ha hamis a feltétel, akkor újra végrehajtódik a ciklusmag. Az ismétlődés addig tart, amíg a feltétel igazzá nem válik. A feltételt befolyásoló utasításoknak a magban kell szerepelniük. A végfeltételes ciklus soha nem lehet üres, de végtelen igen.

A hátultesztelő ciklus megvalósítható egyetlen elágaztató szerkezettel, amelyet a ciklusmag után kell elhelyezni, és amely a feltétel hamis értékének hatására visszalép a ciklusmag elejére.



Kódrészlet: elöltesztelő ciklus

```
        …
UJRA    BTFSC   JELZO,0   ;jelzőbit vizsgálata
        GOTO    TOVA      ;JELZO,0 = 1, kilép
        …                 ;a ciklusmag kezdete
        …                 ;a ciklusmag
        …                 ;a ciklusmag vége
        GOTO    UJRA      ;JELZO,0 = 0,vissza
TOVA    …                 ;JELZO,0 = 1
        …
```



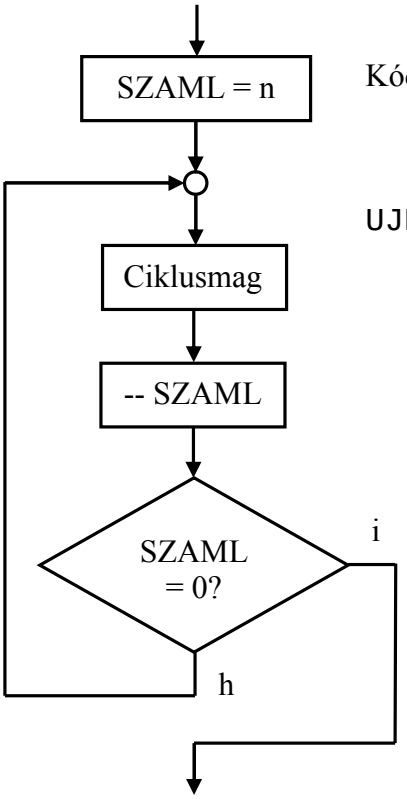
Kódrészlet: hátultesztelő ciklus

```
        …
UJRA    …
        …                 ;a ciklusmag kezdete
        …                 ;a ciklusmag
        …                 ;a ciklusmag vége
        BTFSC   JELZO,0   ;jelzőbit vizsgálata
        GOTO    UJRA      ;JELZO,0 = 0,vissza
        …                 ;JELZO,0 = 1
```

31. ábra

Előírt lépésszámú ciklus: FOR

Az ilyen típusú ciklusok esetén a ciklus megkezdése előtt már ismert, hogy hányszor kell a ciklusmagot végrehajtani, ezt az információt az úgynevezett ciklusváltozó tartalmazza (32. ábra).



Kódrészlet: előírt lépésszámú ciklus

```
        …
        MOVLW  0x0A      ;W = 10D
        MOVWF  SZAML     ;ciklusváltozó
UJRA    …                ;a ciklusmag kezdete
        …                ;a ciklusmag
        …                ;a ciklusmag vége
        DECFSZ SZAML,1   ;SZAML-1; SZAML = 0 ?
        GOTO   UJRA      ;SZAML ≠ 0, vissza
        …                ;SZAML = 0
```

32. ábra

Alprogramok assembly nyelvben

Az alprogramok (szubrutinok) olyan programrészletek, amelyek a program bármely pontjáról meghívhatóak. Az alprogramok akkor alkalmazhatóak, ha a program különböző pontjain ugyanaz a programrész többször ismétlődik, így a program hossza csökkenthető, átláthatósága javítható. A programrészt csak egyszer kell létrehozni, a program megfelelő részein csak hivatkozni kell rá. Az alprogram rendelkezik névvel és rendelkezhet paraméterlistával.

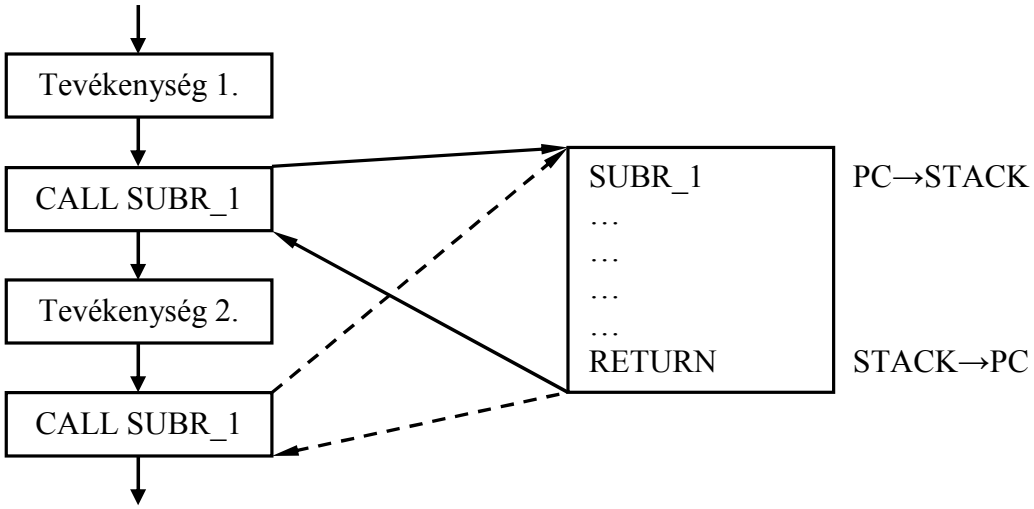
Alprogram típusok:

- **Függvény:** feladata egyetlen érték meghatározása, amelyet általában a hívás helyén a nevével közvetít (értékadás jobb oldalán szerepelhet);
- **Eljárás:** olyan alprogram, amely valamilyen tevékenységet hajt végre. Az eljárás a programban kifejtett hatását a tevékenységei által illetve a megváltoztatott paramétereivel közvetíti a hívás helyén. [**3.** 74. o]

CALL

Assembly nyelvben a legegyszerűbben eljárás hozható létre, a `CALL` szubrutinhívó utasítás segítségével. A `CALL` utasítás hatására a programszámláló értéke a nyolc-szintes hardver verembe kerül, és a programfutás a szubrutinnal és a benne szereplő tevékenységek, végrehajtásával folytatódik (33. ábra). A szubrutin utolsó utasítása, a `RETURN` vagy `RETLW` szolgál arra, hogy a programfutást visszairányítsa a hívás helyére. Ezek hatására a veremből, az ott elhelyezett programszámláló érték (visszatérési cím) átkerül a programszámlálóba, és a program a szubrutinhívás utáni utasítással folytatódik. A `RETLW` utasítás a visszatéréskor egy, az utasításban megadott, 8 bites konstanst tölt be a W munkaregiszterbe.

A megszakítások kezelése az úgynevezett megszakítási szubrutinok segítségével történik, a működés lényege megegyezik a fenti folyamattal, a különbség abban van, hogy a szubrutinnak a hívását nem a programozó kezdeményezi, hanem valamilyen külső vagy belső megszakítási esemény.



33. ábra

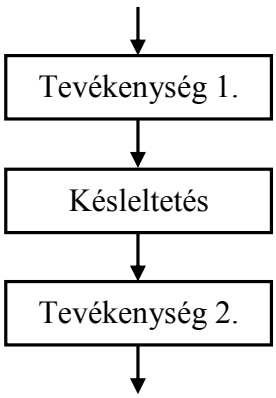
Szoftverkésleltetés létrehozása

Szoftver késleltetések

A hardverműködtetések során számos esetben szükség van bizonyos idejű késleltetése két tevékenység végrehajtása között. A késleltetést célszerű egyszerű szoftveres eszközökkel megoldani. A szoftveres késleltetés működési alapja az, hogy az időzítési időtartam alatt a processzor a két tevékenységgel össze nem függő egyéb műveleteket (pl. NOP) hajt végre. A késleltetést a többször felhasználhatóság és tártakarékosság miatt általában szubrutinban érdemes létrehozni.

Rövid késleltetések NOP utasításokkal

A NOP üres utasítás végrehajtása során nem történik műveletvégzés, viszont eltelik egy utasításciklusnyi idő (34. ábra). (Az utasításciklus hossza az órajel frekvenciájától függ: gépi ciklus = $4/f_{CL}$.) A rövid idejű késleltetéseket általában szimulációban illetve jelátvitel esetében bitidőzítés létrehozására használnak.



34. ábra

Kódrészlet: rövid idejű késleltetés

```
        …
        CALL   WAIT  ;a szubrutin hívása
        …
        …
WAIT    NOP          ;a késleltető szubrutin
        NOP
        NOP
        RETURN       ;a szubrutin vége
```

Az utasításciklusok száma (k) egyszerű, N db NOP-ot tartalmazó szubrutin esetén.

$$k = 2 + N + 2 = 4 + N$$

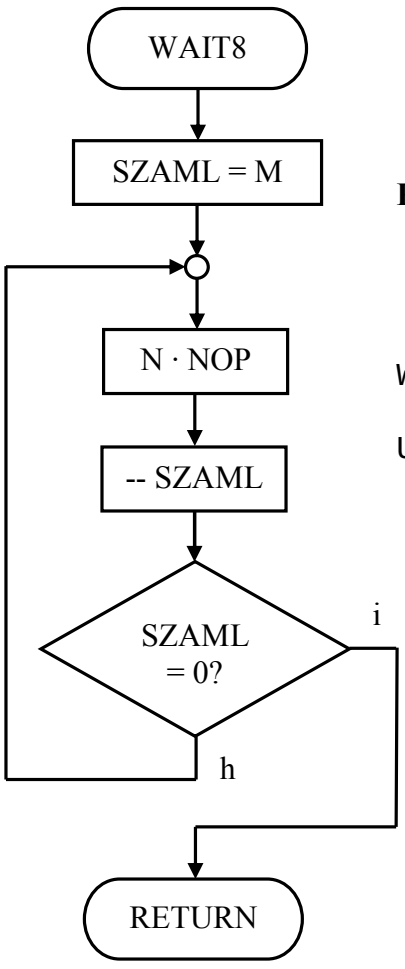
Hosszabb idejű késleltetések ciklusszervezéssel

A NOP egyszerű használatával csak rövid (µs-os) késleltetések hozhatók létre gazdaságosan. Az f_{CL} = 10 MHz esetén, 0,5 s-os időzítéshez 1 249 996 db NOP-ot kellene a fenti szubrutinnak tartalmaznia! A tártakarékos szoftver késleltetéseket ciklusutasítások (előírt lépésszámú) segítségével lehet létrehozni. A ciklusmag általában itt is NOP-okat tartalmaz.

1. 8 bites késleltető ciklus

A legegyszerűbb megoldást a `DECFSZ` utasítással felépített 8 bites előírt lépésszámú ciklus adja (35. ábra). A ciklus megkezdése előtt a SZAML regiszterbe be kell tölteni a kezdeti értéket (M és $1 \le M \le 255$), ami az ismétlések számát adja meg, ez a késleltetés hosszát fogja egyértelműen meghatározni.
Az utasításciklusok száma (k) M ismétlődésű és N db NOP-ot tartalmazó szubrutin esetén:

$$k = 2 + 2 + M \cdot (3 + N) + 2 = 6 + M \cdot (3 + N)$$



35. ábra

A NOP-ok természetesen el is maradhatnak, ill. M = 0 esetén, a ciklus 256-szor fog lefutni. Az f_{CL} = 10 MHz esetén, 9 db NOP-ot használva és M = 255 érték mellett, a késleltetés időtartama, közelítőleg 1,2264 ms.

Kódrészlet: 8 bites késleltetés

```
        …
        CALL    WAIT8   ;a késleltető
        …               ;szubrutin hívása
        …
WAIT8   MOVLW   0x0A    ;M = 10
        MOVWF   SZAML   ;SZAML = M
UJRA    NOP             ;N = 2
        NOP
        DECFSZ  SZAML,1 ;SZAML–1 = 0?
        GOTO    UJRA
        RETURN          ;a szubrutin vége
```

2. 16 bites késleltető ciklus

A nagyobb késleltetési idők létrehozására több számlálót kell bevonni a késleltető ciklusba. A 8 bites ciklusokat egymásba ágyazva már komoly késleltetési idők hozhatóak létre.
A ciklus megkezdése előtt a SZAML1 és SZAML2 regiszterekbe kell betölteni a kezdeti értéket (L és $1 \le L \le 255$ ill. M és $1 \le M \le 255$).

Az utasításciklusok száma (k) két egymásba ágyazott 8 bites, L és M ismétlődésű, N db NOP-ot tartalmazó szubrutin esetén (az L a külső, M a belső ciklus alapértéke).

$$k = 2 + 2 + L \cdot (M \cdot (3 + N) + 3 + 2) + 2 = \\ = L \cdot M \cdot (3 + N) + L \cdot 5 + 6$$

Az f_{CL} = 10 MHz esetén, 9 db NOP-ot használva és M = L = 100 érték mellett, az utasításciklusok száma: 120 506 db, így a késleltetés időtartama, közelítőleg 48,2 ms.

Kódrészlet: 16 bites késleltetés

```
        …
        CALL    WAIT16    ;a késleltető szubrutin hívása
        …
        …
WAIT16  MOVLW   0x64      ;L = 100
        MOVWF   SZAML1    ;SZAML1 = L
ALAP    MOVLW   0x64      ;M = 100
        MOVWF   SZAML2    ;SZAML2 = M
UJRA    NOP               ;N = 2
        NOP
        DECFSZ  SZAML2,1  ;SZAML2–1 = 0?
        GOTO    UJRA
        DECFSZ  SZAML1,1  ;SZAML1–1 = 0?
        GOTO    ALAP      ;SZAML1 alapérték
        RETURN            ;a késleltető szubrutin vége
```

Programfejlesztés MPASM IDE rendszerben

A Microchip az MPLAB IDE integrált fejlesztői környezetet biztosítja a mikrokontrollereinek a programozására. A fejlesztői környezet az eszközrendszereivel alapvetően az assembly programfejlesztést támogatja. Sokrétű szolgáltatási közül csak azok kerülnek bemutatásra, amelyek a fejlesztéshez alapvetően szükségesek. [**1.** 225. o]

1. Program indítása:
   - MPLAB IDE parancsikon segítségével;
   - Start menü: Microchip MPLAB IDE mappájából;
   - „C:\Program Files\MPLAB IDE\dlls\MPlab.exe” útvonalon.
2. Az MPLAB IDE (v7.52) fájlrendszere:

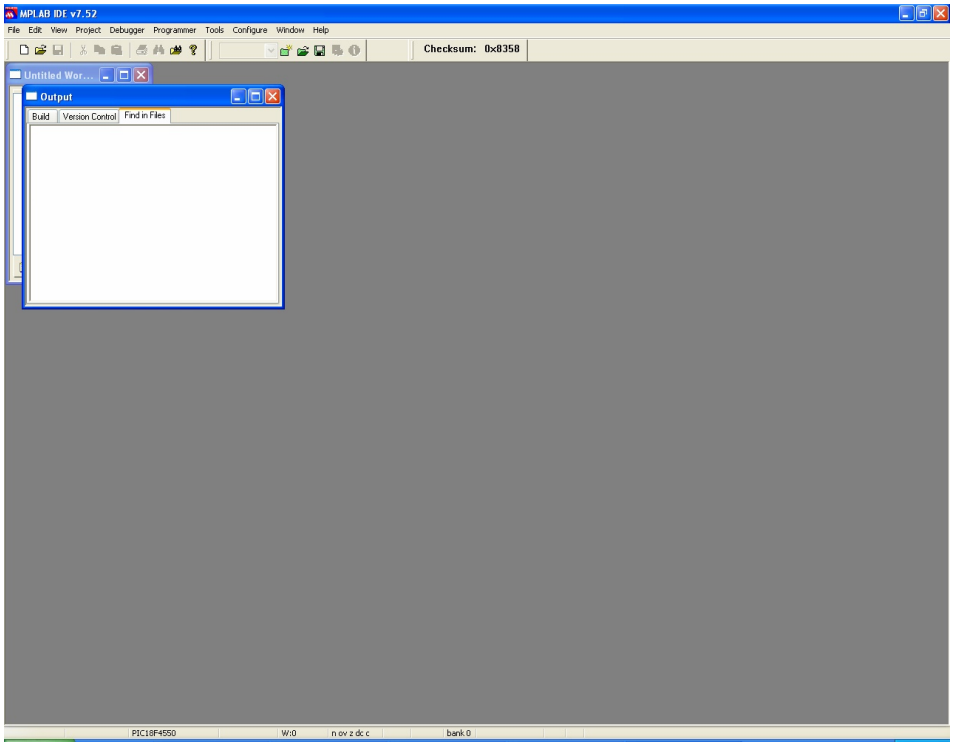
   A fejlesztői környezet a manapság divatos projekt szemléletet használja, tehát egy programhoz több különböző funkciójú állomány társítható. A programfejlesztéskor ezért ajánlott az azonos programhoz tartozó fájlokat azonos néven, egy közös mappában tárolni, így a programok közötti esetlegesen előforduló fájlkeveredés elkerülhető.

   Fontosabb fájlok:

   - Project fájl (*.mcp), az adott fejlesztéshez tartozó fájlokat fogja össze;
   - Munkaterület (workspace) fájl (*.mcw), a használt munkafelületet tárolja: fordítási, szimulációs, ablak beállításokat;
   - Forrásfájl (*.asm), tartalmazza a program assembly forráskódját, az MPASM ezt fordítja le;
   - Programozó fájl (*.hex), a sikeres fordítás eredménye, ennek a fájlnak a tartalma kerül a kontrollerbe (az ún. Intel-Hexa formátum);
   - Tárgykód (*.COD), a fordítás utáni tárgykód;
   - Hibafájl (*.err) a fordítás közben fellépő hibákat, figyelmeztetéseket tárolja;
   - Listafájl (*.lst) a fordítás listafájlja.

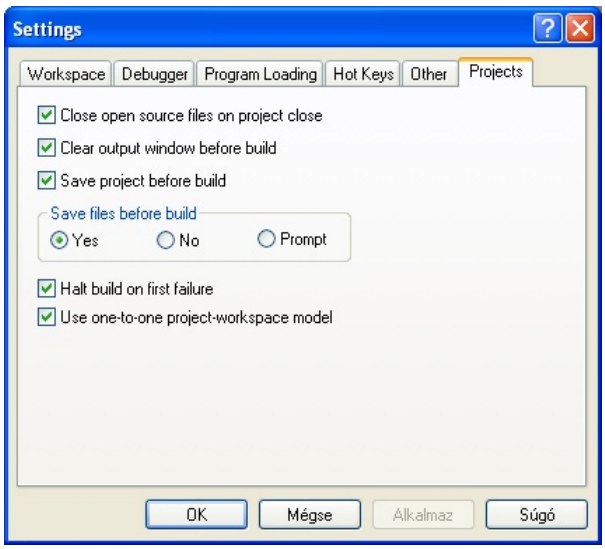
Programfájlok létrehozásának (*.asm, *.mcp, *.mcw) lépései:

1. MPLAB-IDE indítása (36. ábra);

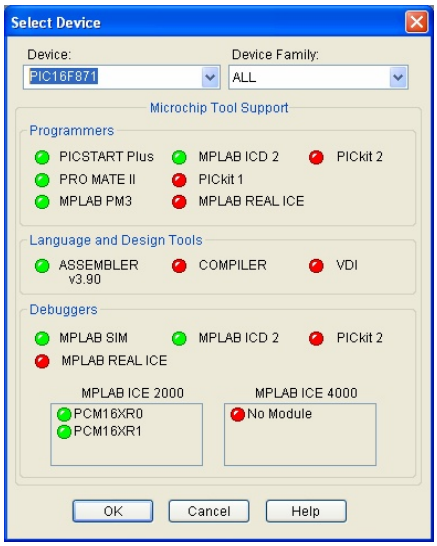


36. ábra

2. Configure │Settings │Projects → Use one-to-one project-workspace modell (37. ábra);
3. Configure │Select Device → PIC16F871 (38. ábra);

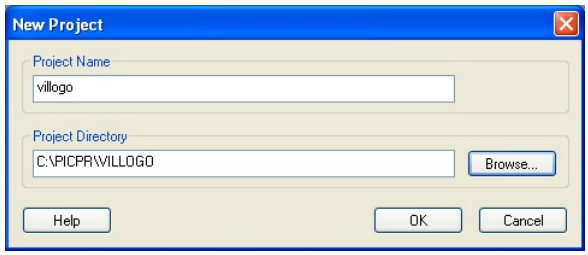


37. ábra



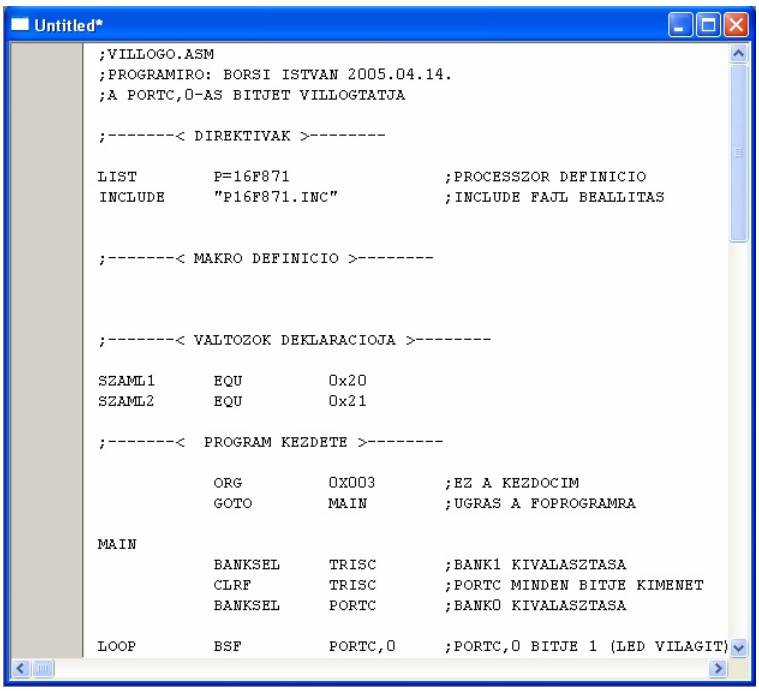
38. ábra

4. Új project készítése: Project │New → project név megadása, helyének a kijelölése (pl. „C:\PICPR\VILLOGO”) (39. ábra);



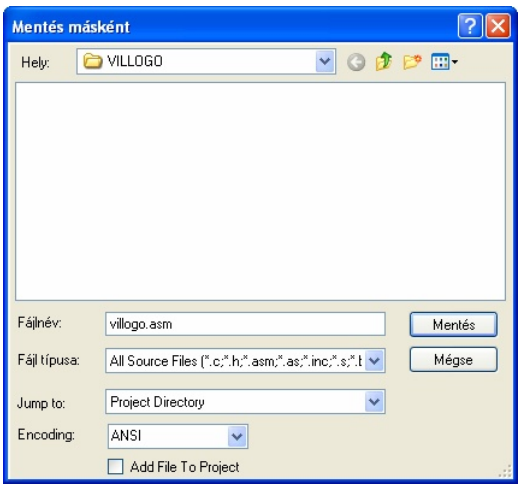
39. ábra

5. Új forrásfájl készítése: File │New → „untitled”;
6. Programírás (ill. forrásfájl átemelés) (40. ábra);



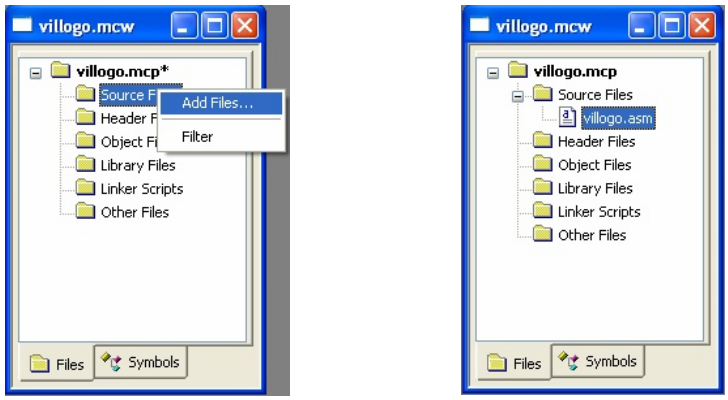
40. ábra

7. Forrásfájl mentése: File │Save → „villogo.asm” (a munkakönyvtárba) (41. ábra);



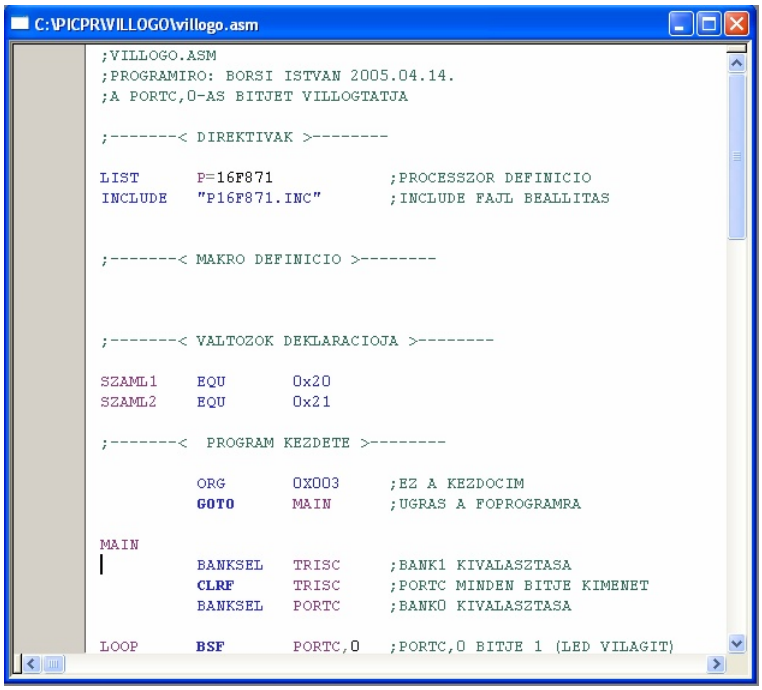
41. ábra

8. Forrásfáj – project (munkaterület) összerendelés: villogo.mcp → Source Files (jobb egérgomb) → Add Files... → a forrásfájl „betallózása” (villogo.asm) (42. ábra);



42. ábra

9. Programírás (43. ábra);

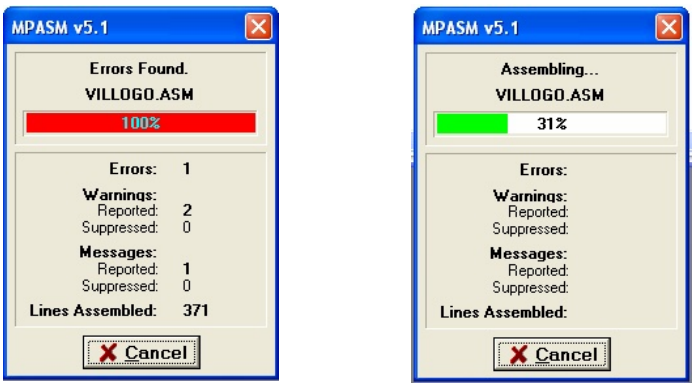


43. ábra

10. Mentés: File | Save All ill. File | Save Workspace. (A még nem mentett állományt a fájlnév mellett megjelenő csillag jelzi.)

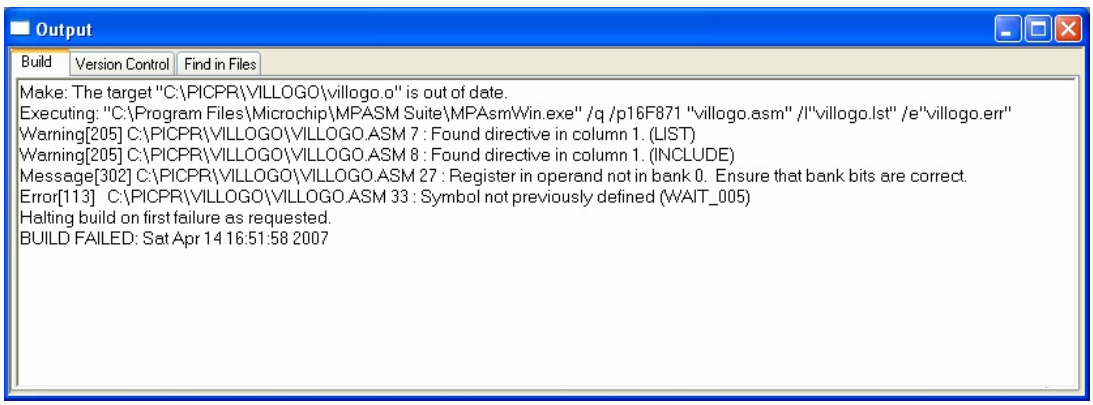
Programfordítás (*.hex) lépései:

Project | Build All (Ctrl+F10) vagy eszköztár Build All ikon, ha nincs hiba, létrejön a hex fájl. (Csak zöld állapotjelző vonalat lehet látni.) Amennyiben hiba lép fel, akkor ki kell javítani, majd újra kell fordítani (44. ábra).



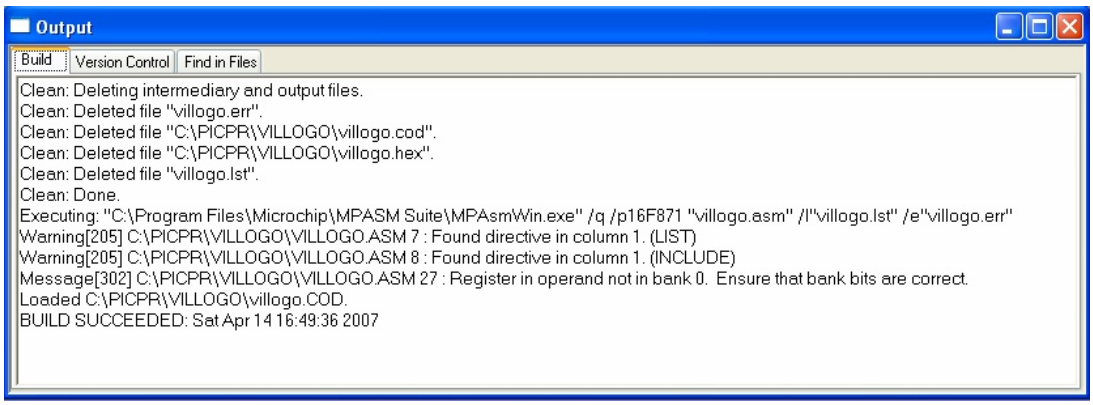
44. ábra

Sikertelen fordítás Output ablaka (45. ábra)



45. ábra

Sikeres fordítás Output ablaka (46. ábra)



46. ábra

Ellenőrző kérdések

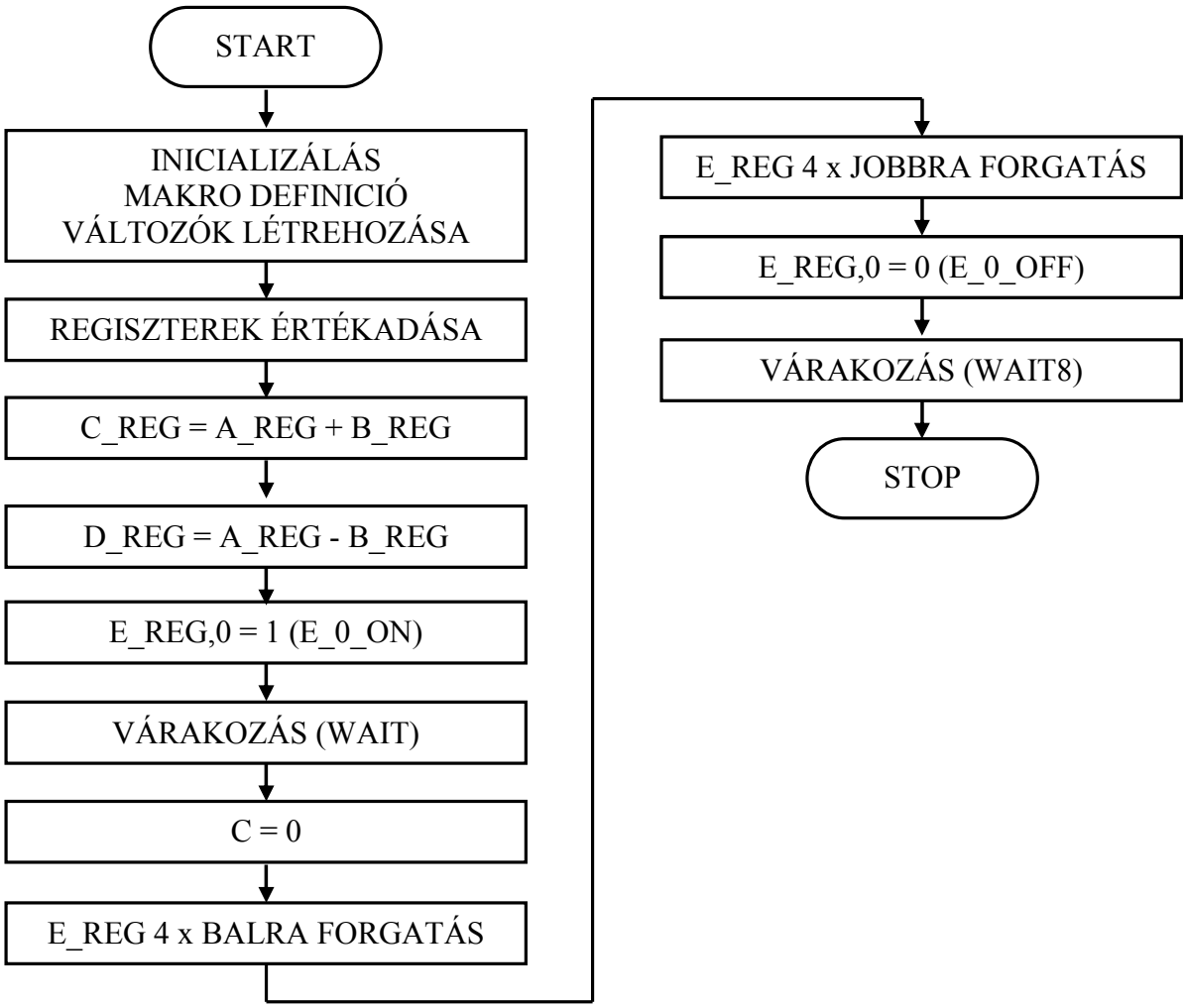
1. Méretezzen 0,5 s-re késleltető rutint.
2. Hogyan képes eljárás, az MPASM-ban paramétert átvenni és átadni?
3. Az interneten keresse meg az MPASM IDE legújabb verzióját, és adja meg a verziószámát.

9. Foglalkozás

Programozási feladat

Készítsen el a PIC 16F871 mikrokontrollerre, egy olyan demonstrációs programot, amely az ismeretek összefoglalásaként, alapvető programozás technikai tartalommal bír. A feladatot a megadott folyamatábra alapján készítse el. A program forráskódja a függelékben elérhető (`SZAMOL.ASM`).

Az elkészítendő program folyamatábrája.



47. ábra

Segítő információk:

- A felhasználói regiszterek (A_REG – E_REG) folytonosan helyezkedjenek le a memóriában a 20h címtől;
- Az E_0_ON és az E_0_OFF makrók, amelyek az E_REG,0-át kezelik;
- A WAIT, egyszerű késleltető szubrutin, amely pár db NOP-ot tartalmaz;
- A WAIT8, 8 bites késleltető szubrutin, a ciklusok számát 10 körülire válassza.

Program szimuláció MPASM rendszerben

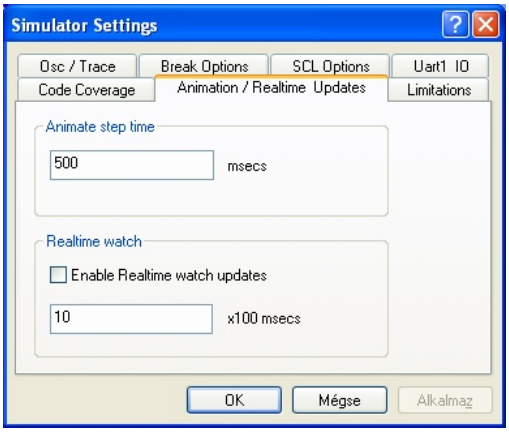
A program szimuláció célja, hogy a céleszközbe történő programbetöltés előtt, az elkészített program működőképességéről meg lehessen győződni. Az MPASM eszközrendszerével, természetesen támogatja ezt a lehetőséget.

Tesztelés (debug) lépései:

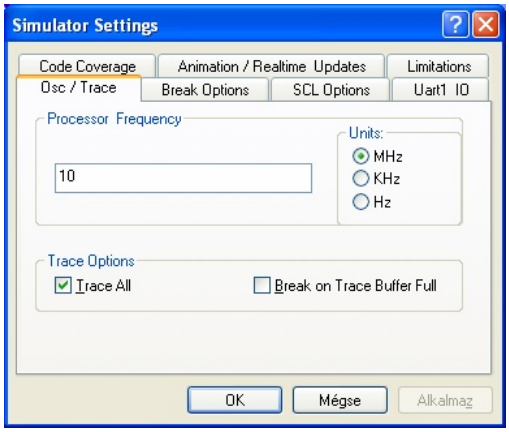
Természetesen a szimuláció, csak sikeres fordítás után lehetséges.

1. Szimuláció bekapcsolása: Debugger | Select Tool | MPLAB SIM; Az eszköztár bővült a debug elemeivel;

48. ábra

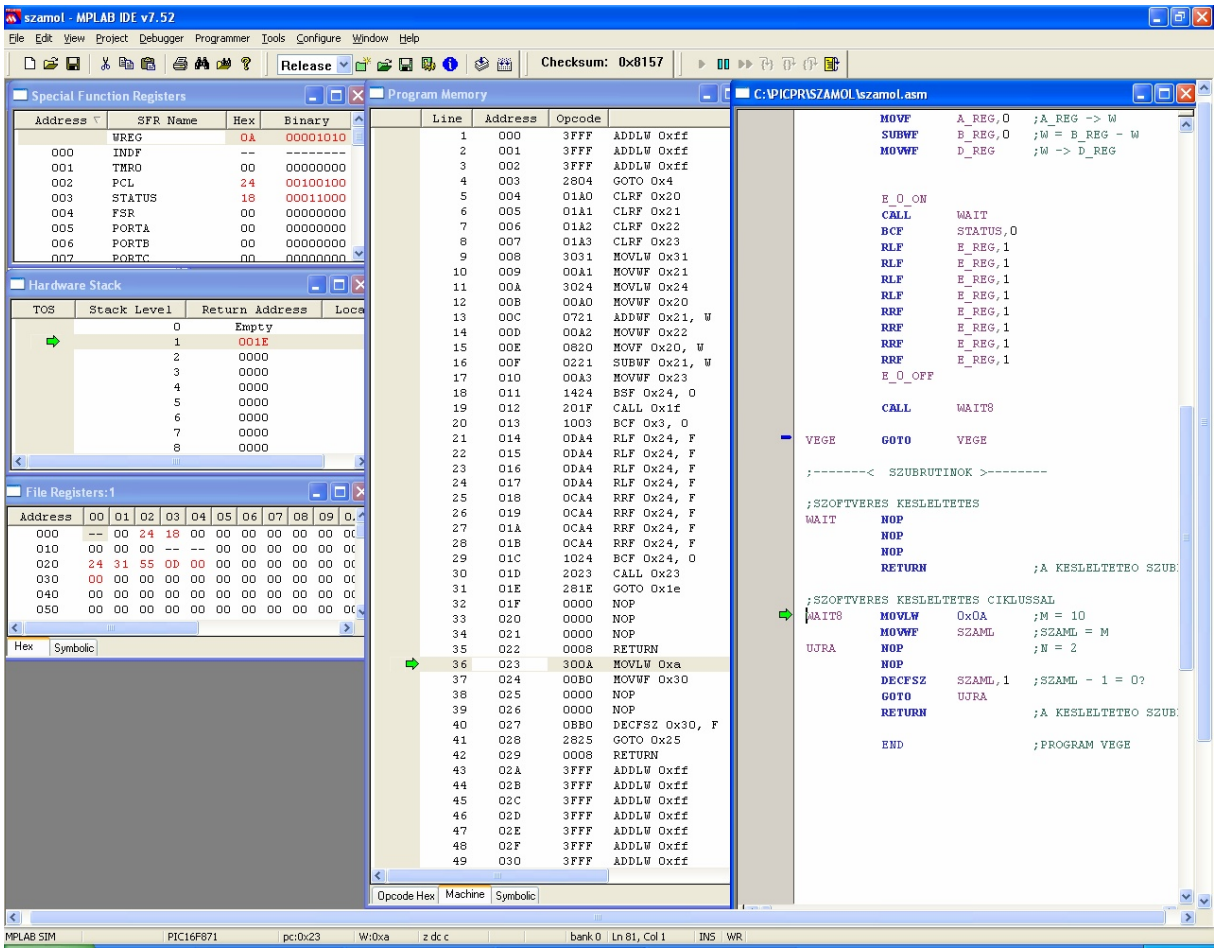


49. ábra



50. ábra

2. Animáció sebességének a beállítása: Debugger | Settings... → Debugger animation → Animate step time: 500ms (49. ábra);
3. Órajel beállítása: Debugger | Settings... → Clock: 10MHz (50. ábra);
4. Szimuláció ablakai: Wiew | Program Memory, File Registers, SFR (az ablakok elhelyezésénél a láthatóságra oda kell figyelni) (51. ábra);
5. Lépésenkénti szimuláció: Debugger | Step Into (F7) vagy Step Into ikon;
6. Folyamatos szimuláció: Debugger | Animate vagy Animate ikon;
7. Szimuláció leállítása: Debugger | Halt (F5) vagy Halt ikon;
8. Reset: Debugger | Reset → Processor Reset (F6) vagy Reset ikon;
9. Memória törlése: Debugger | Clear Memory → All Memory;
10. Munkaterület mentése: File | Save Workspace.



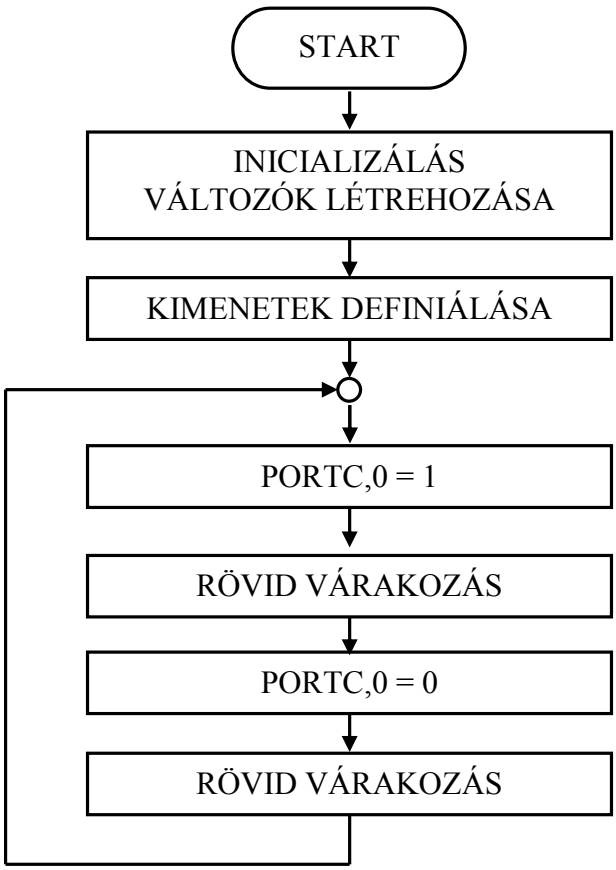
51. ábra

1 LED-es villogó programjának elkészítése, szimulációja

Az 1-ledes villogók a mikrokontrolleres fejlesztésekben olyan fontossággal bírnak, mint a magas szintű programozási nyelvekben (pl. C) a „Hello World!” típusú alkalmazások. Alapvető programírási gyakorlatot és sikeres fordítást feltételez.

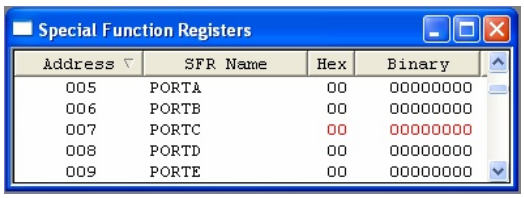
Az elkészítendő 1 LED-es villogó programjának folyamatábrája (52. ábra). A program forráskódja a függelékben elérhető (`VILLOGO.ASM`).

A program egyszerűen működik, egy programhurokban kell ciklikusan változtatni a kimeneti portbit (PORTC,0) értékét 0 és 1 között. Az egyes kimeneti állapotok közé késleltetést kell beilleszteni, mert ez fogja biztosítani a láthatóságot (különben túl gyors lenne és az emberi szem, nem tudná követni). A késleltetés nagyságrendje szimulációban néhány utasításciklusnyi, a kontrollerre történő szoftverfejlesztés esetén pedig néhányszáz milliszekundumtól másodpercekig terjedhet.

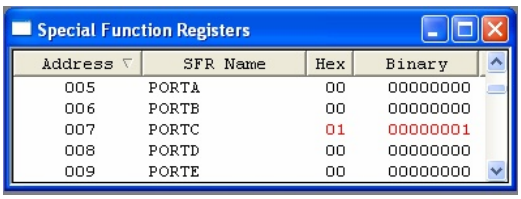


52. ábra

A szimuláció során az figyelhető meg, hogy a PORTC legkisebb helyi értékű bitje nulla (53. ábra) és egy (54. ábra) között billeg, tehát ha oda egy LED-et kötnénk, az folyamatosan villogna.



53. ábra



54. ábra

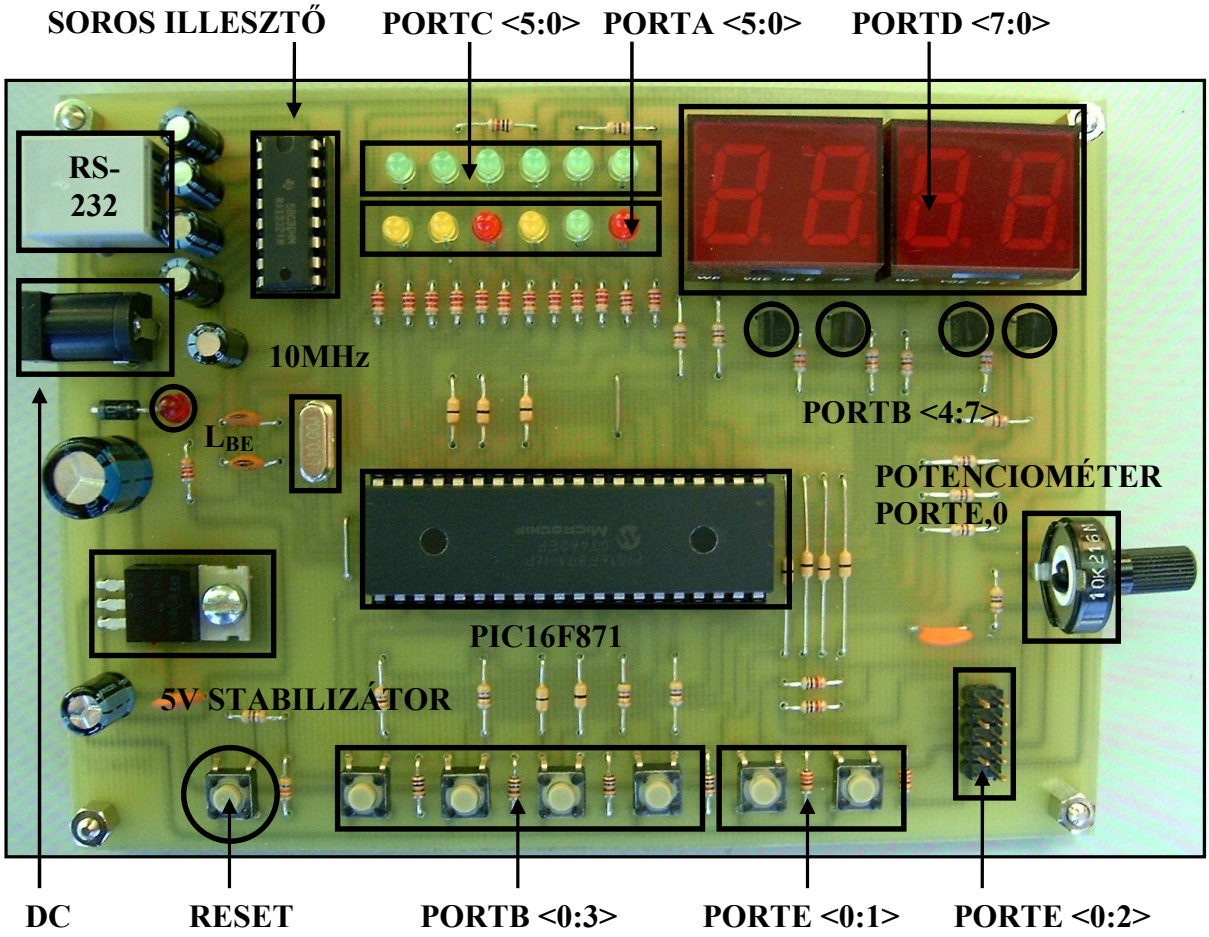
PIC-es gyakorlópanel

A gyakorlópanel e tananyag programozási részéhez készült, mint egy olyan univerzálisan használható áramkör, amelyen az alapvető programozási feladatok gyakorolhatóak.

A gyakorlópanel felépítését tekintve tartalmaz:

- Mikrovezérlőt: PIC 16F871-es mikrokontrollert.

§ Perifériaelemeket:
- kimeneti: LED sor, 4 digites 7 szegmenses kijelző;
- bemeneti: nyomógombok, potenciométer;
- kommunikációs: RS-232-es csatoló.



55. ábra

PIC-es gyakorlópanel segítségével lehet az MPASM-ban kifejlesztett mikrokontrolleres programokat valós környezetben futtatni.

PIC-es gyakorlópanel periféria címei:

Megnevezés:	**Megnevezés**	**Port kivezetés:**
Világító diódák (LED-ek)	LC0 – 5	PORTC <0:5>
Világító diódák (LED-ek)	LA0 – 5	PORTA <0:5>
Nyomógomb	NB0 – 3	PORTB <0:3>
Nyomógomb	NE0 – 1	PORTE <0:1>
Tüskesor	PORTE <0:2>	
Potenciométer	PORTE,2	
Soros vonal	PORTC <6:7>	
Kijelző anódok	digit0 – 3	PORTB <7:4>
Kijelző szegmensek	dp, g, b, f, a, c, e, d	PORTD <0:7>

Megjegyzés:

- A LED-ek logikai magas szintre világítanak (de a PORTA,4-es port 0 szintre aktív);
- A nyomógombok lenyomása 0 szintet ad;
- A kijelző anódokat logikai 0 kapcsolja be:
  digit3 = PORTB,7; digit2 = PORTB,6; digit1 = PORTB,5; digit0 = PORTB,4.
- A szegmensek logikai 0 szintre aktívak:
  dp = PORTD,0; g = PORTD,1; b = PORTD,2; f = PORTD,3;
  a = PORTD,4; c = PORTD,5; e = PORTD,6; d = PORTD,7.
- Az L_{BE} LED az áramkör tápfeszültségének a meglétét jelzi.

Program letöltés a PIC-es gyakorlópanelbe

1. Feszültséget kell adni a gyakorlónak, amit az L_{BE} LED jelez, a tápfeszültséget egy 12V-os hálózati dugasz tápegység szolgáltatja;
2. A PC-t és a gyakorlót össze kell kötni egy speciális soros kábellel (az RS-232-es összeköttetés az RX, TX és a GND bekötését jelenti, a port többi funkcióját nem használja a PIC-es gyakorló);
3. El kell indítani a programletöltő, a **dwnldr.exe** programot, és be kell állítani az elérhető soros portot (COM1-4), a letöltendő hexa fájlt (programnév.hex), az adatátviteli sebességet (9600 baud) és a mikrokontroller típusát (PIC 16F87X);
4. A RESET gombot nyomva tartva, meg kell nyomni, az NB0 gombot, el kell engedni a RESET-et, majd végül az NB0-át is fel kell engedni. Ekkor az LC0-ás LED kettőt villanva jelzi, hogy a gyakorló kész a letöltendő programot fogadni;
5. A Letöltés gombot lenyomva a folyamatjelző mutatja a letöltés folyamatát, a gyakorlón ezt az LC0-ás LED villogása mutatja;
6. A sikeres letöltés esetén az LC0 LED hármat villan és kialszik, ami után a RESET gomb működtetésével elindítható a saját letöltött program futása. Sikertelen letöltés esetén az L0-ás LED folyamatosan világít.

Megjegyzés:

A program letöltését a mikrovezérlőbe előzetesen beégetett úgynevezett betöltő program (bootloader) vezérli. A bootloader a programmemória végén található, ennek a programrésznek az aktivizálása történik a kezdeti RESET és NB0-ás nyomógombok működtetésével. A programfejlesztésnél figyelni kell arra, hogy a saját program szigorúan csak a 0003h címen kezdődhet, a RESET vektor 0000h-ás helyét elfoglalja egy ugró utasítás, amely a betöltőre adja át a vezérlést. A letöltő programot először a Microchip publikálta a TB025-ös anyagában, a dwnldr.exe program is ennek a projektnek lett a folytatása. [**1.** 206. o]

Ellenőrző kérdések

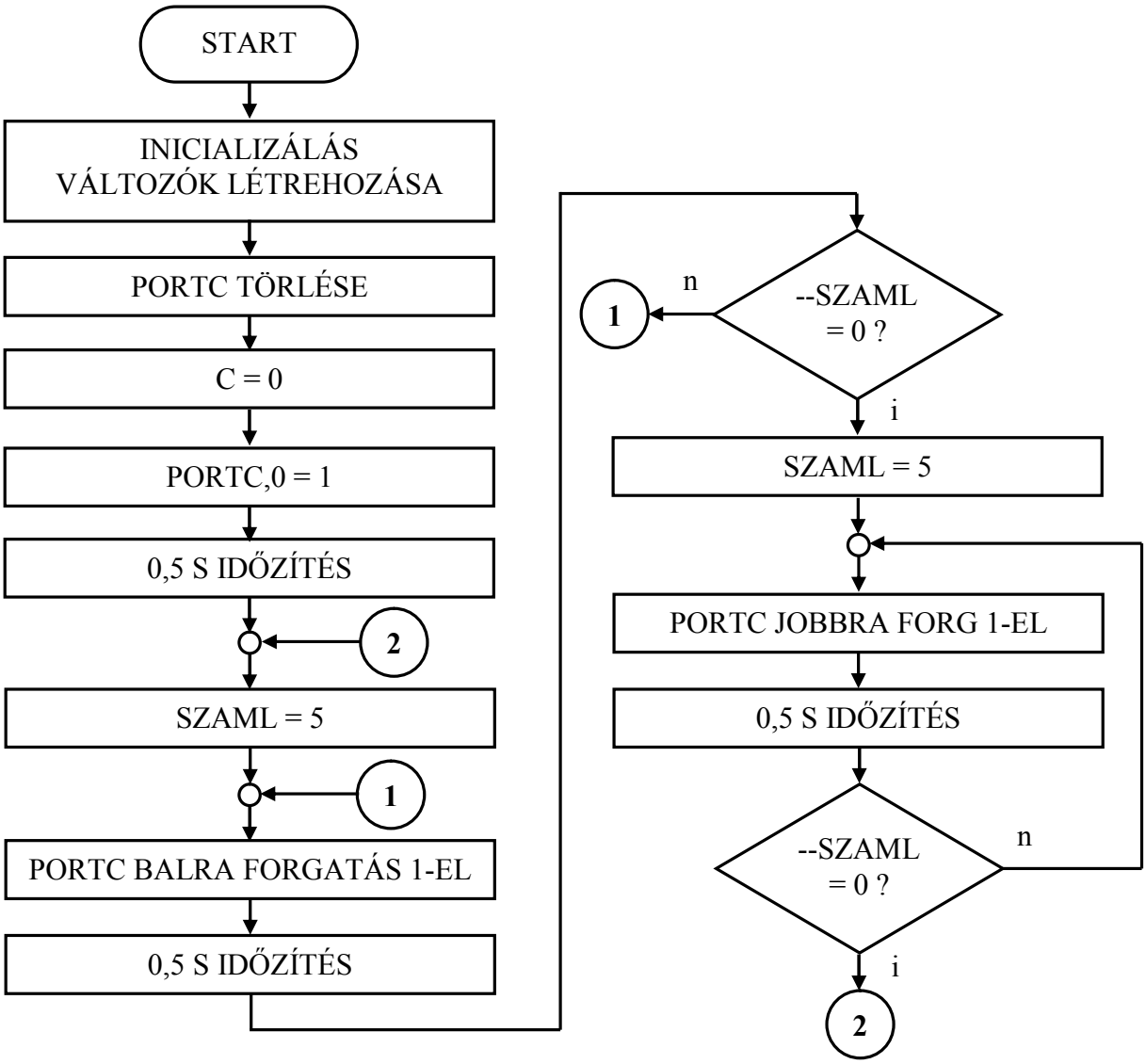
1. A PORTE,2-es portra csatlakozó potenciométernek mi lehet a feladata?
2. Keresse meg a 7 szegmenses kijelző szegmens hozzárendelését (tavalyi tananyag).
3. Az egyes perifériaelemekhez milyen elképzelt feladatokat tudna rendelni?

10. Foglalkozás

1. Programozási feladat – futófény program 1.

Futófény program elkészítése a PORTC alsó 6 bitjére forgató utasítások (RLF és a RRF) használatával. A program forráskódja (FFENYSH.ASM) a függelékben elérhető.

Az elkészítendő program folyamatábrája.



56. ábra

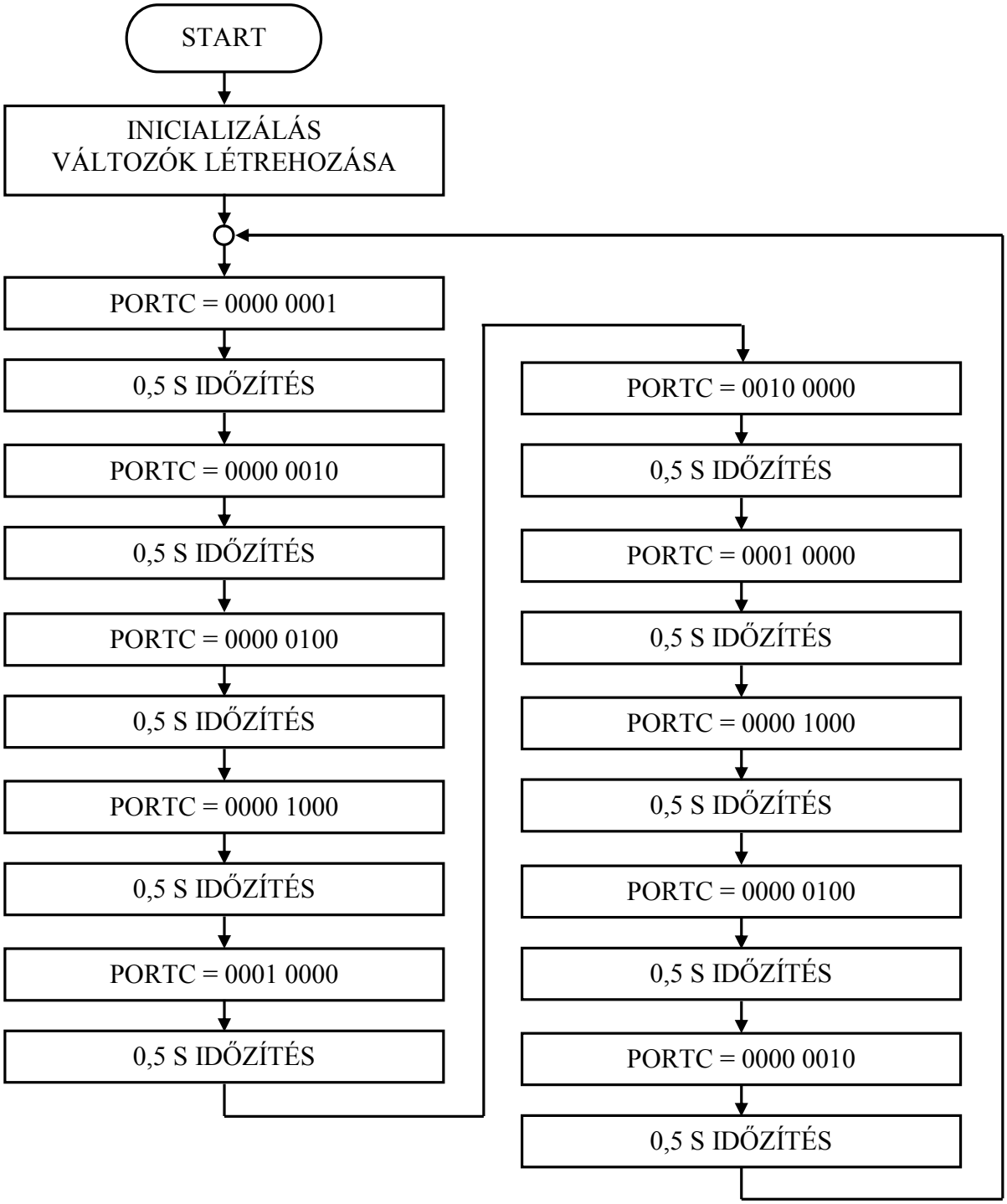
Megjegyzés:

- A program elkészíthető egyszerűbben is, a ciklusutasítások elhagyásával (tisztán szekvenciális módon lásd a következő feladatban);
- Az időzítő szubrutinként az előzőekben tárgyalt 16 bites változatot kell alkalmazni;
- A PORTC használatánál a TRISC értékének a megfelelő beállításáról gondoskodni kell;
- A feladat megoldásánál először szimulációval kell ellenőrizni a programhelyességet.

2. Programozási feladat – futófény program 2.

Futófény program elkészítése a PORTC alsó 6 bitjére ütemezéssel. A program forráskódja (`FFENYUT.ASM`) a függelékben elérhető.

Az elkészítendő program folyamatábrája.

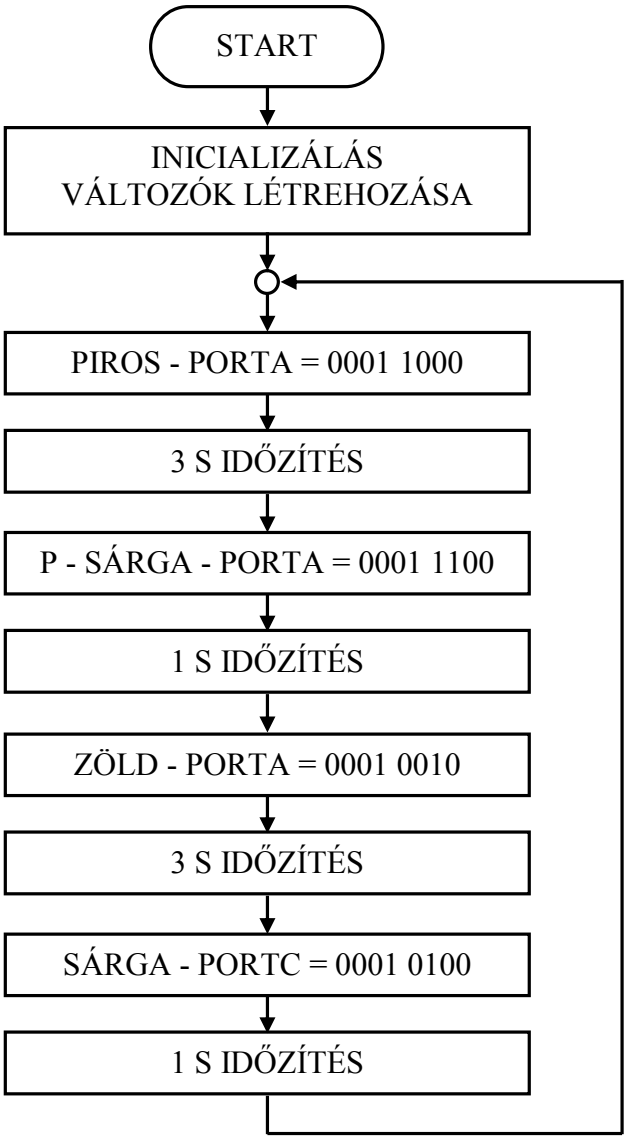


57. ábra

3. Programozási feladat – jelzőlámpa modell

Jelzőlámpa modell programozása ütemezéssel. A program forráskódja (`JELZOL.ASM`) a függelékben elérhető.

Az elkészítendő program folyamatábrája.

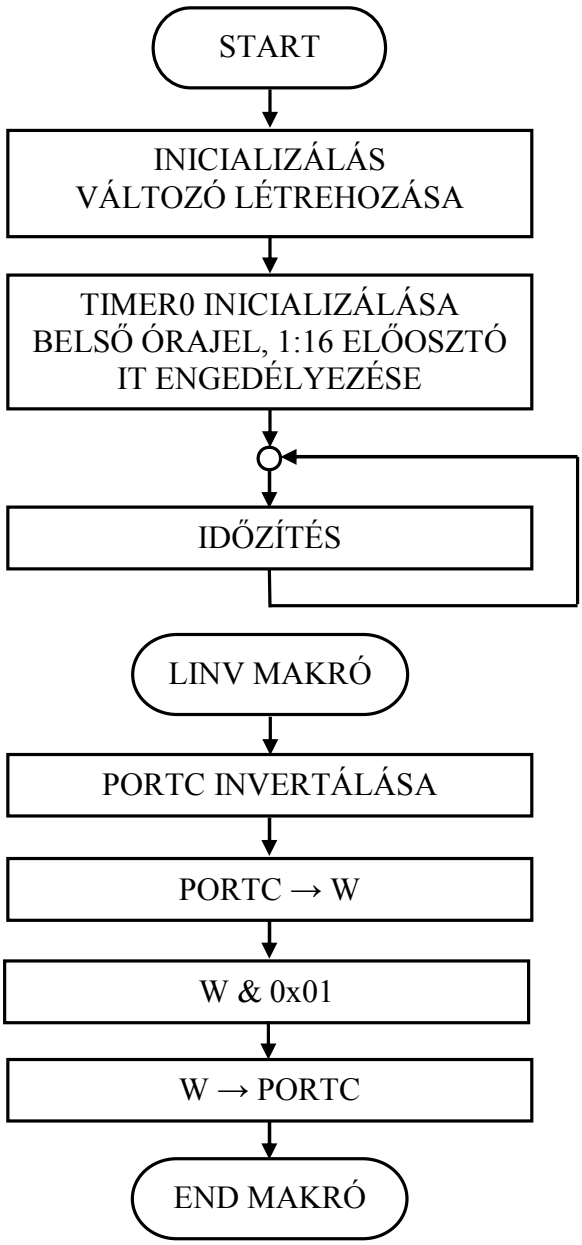


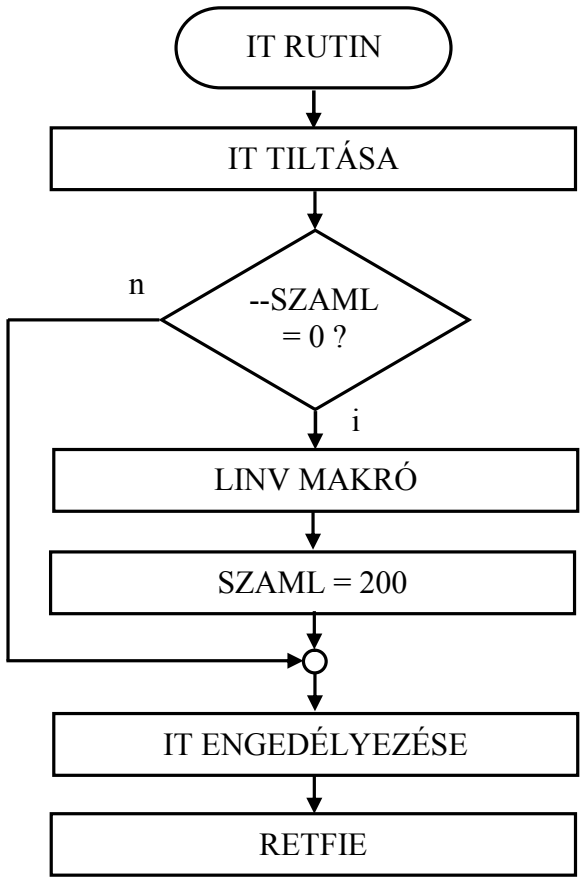
58. ábra

4. Programozási feladat – 1 LED-es villogó megszakításkezeléssel

1 LED-es villogó készítése megszakításkezeléssel. A program forráskódja (`VILLOGIT.ASM`) a függelékben elérhető.

Az elkészítendő program folyamatábrái.





Megjegyzés:

- A programot érdemes részekre bontani, az LINV makró helyett szubrutint is lehet írni;
- A megszakítás konfigurációjánál használni kell az elméletben tanult ismereteket.

59. ábra

Ellenőrző feladatok

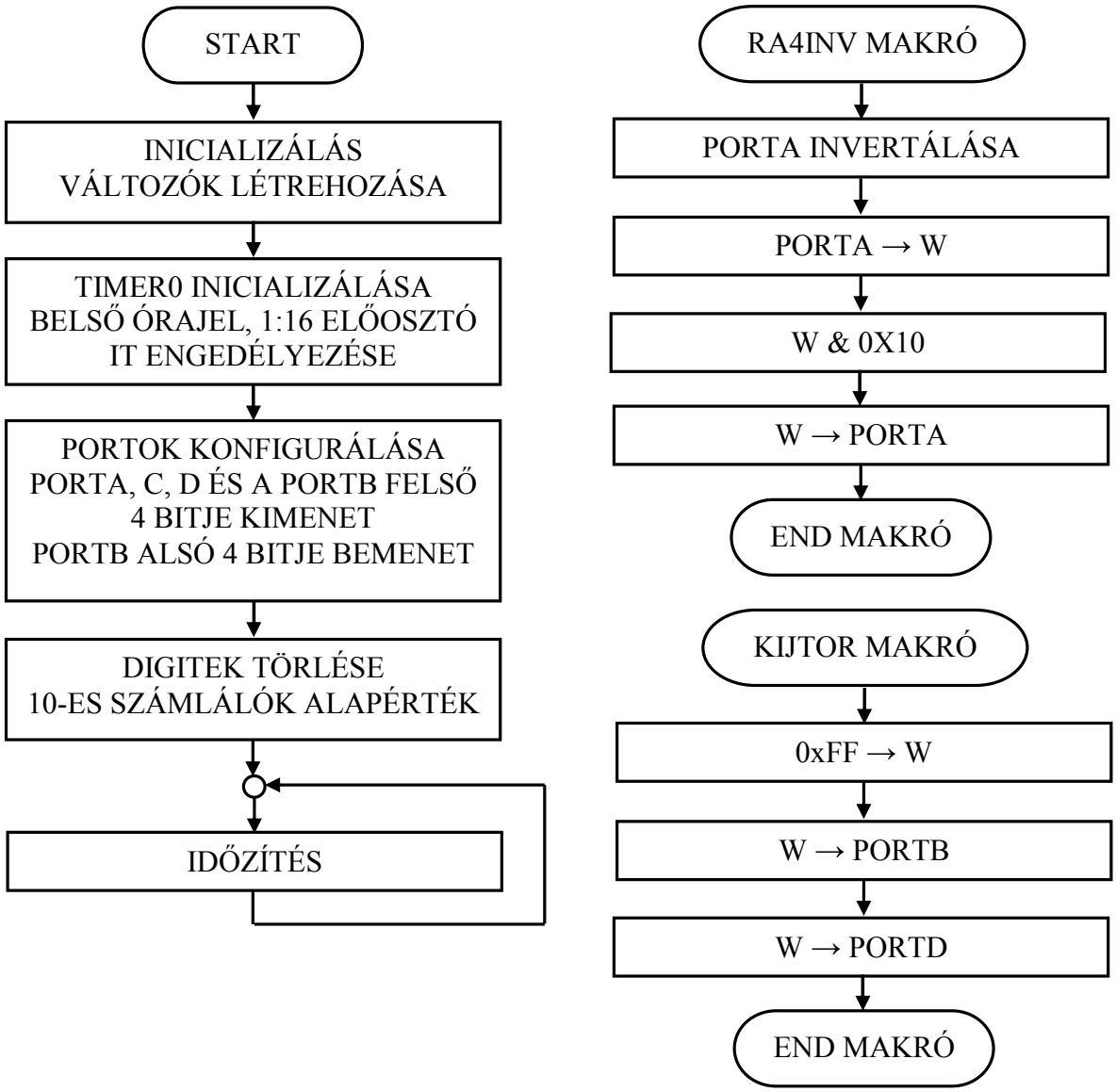
1. Készítsen váltva-villogót a PORTC,0 és PORTC,5 port kivezetésekre (vasúti átkelő);
2. Írjon programot, amely folyamatosan a PORTC-n egy LED-et balról-jobbra, ezzel egy időben a PORTA-n szintén egy LED-et, de jobbról-balra kigyújt;
3. Az IT kezelő programot írja át úgy, hogy az a PORTC-n egy bináris felfelé számlálót valósítson meg.

11. Foglalkozás

1. Programozási feladat – 7 szegmenses kijelző kezelése

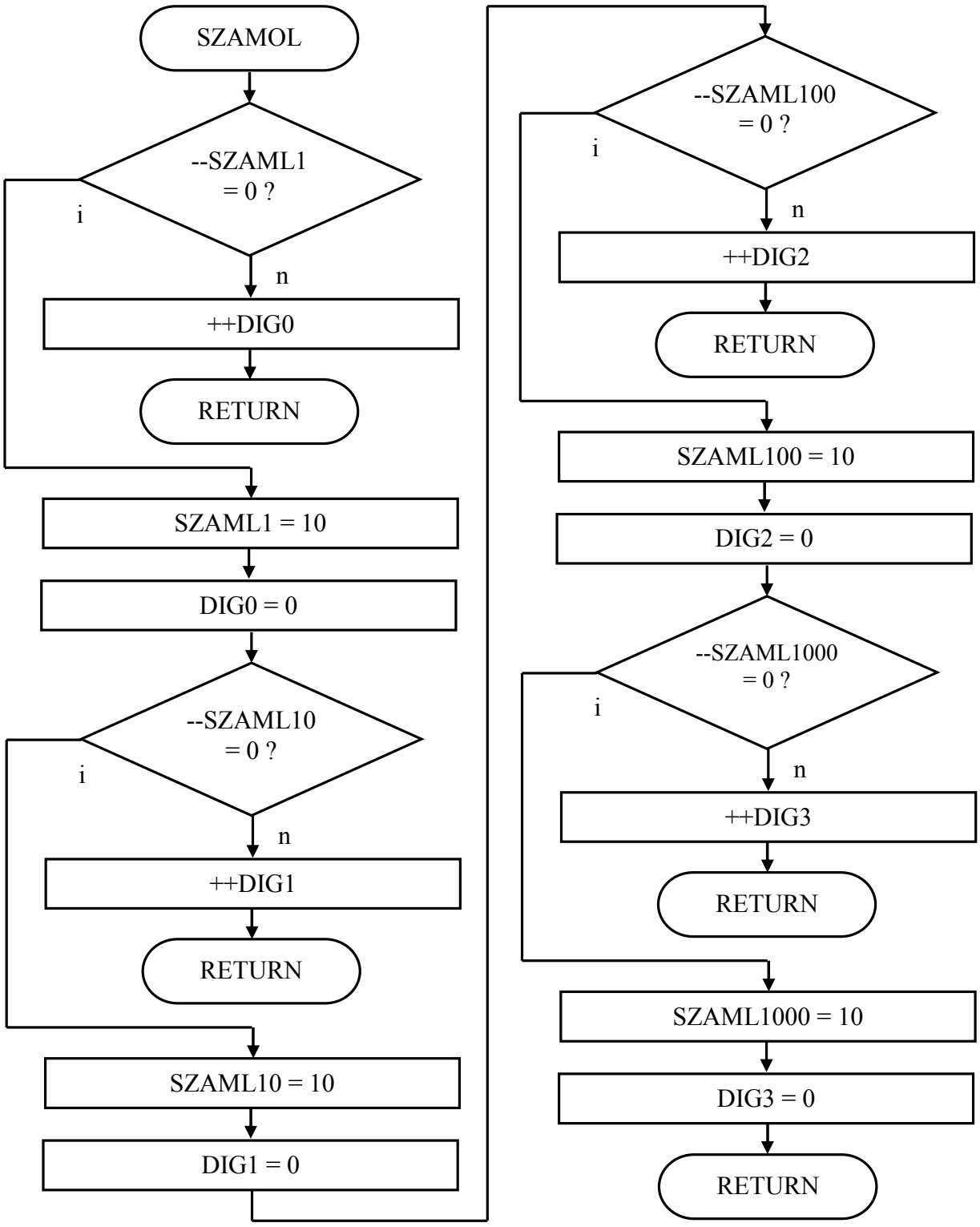
7 szegmenses kijelző kezelésének alapjai a táblakezelés és a multiplex kijelzés fogalma. A program forráskódja (`KIJMPX.ASM`) a függelékben elérhető.

A program folyamatábrái.



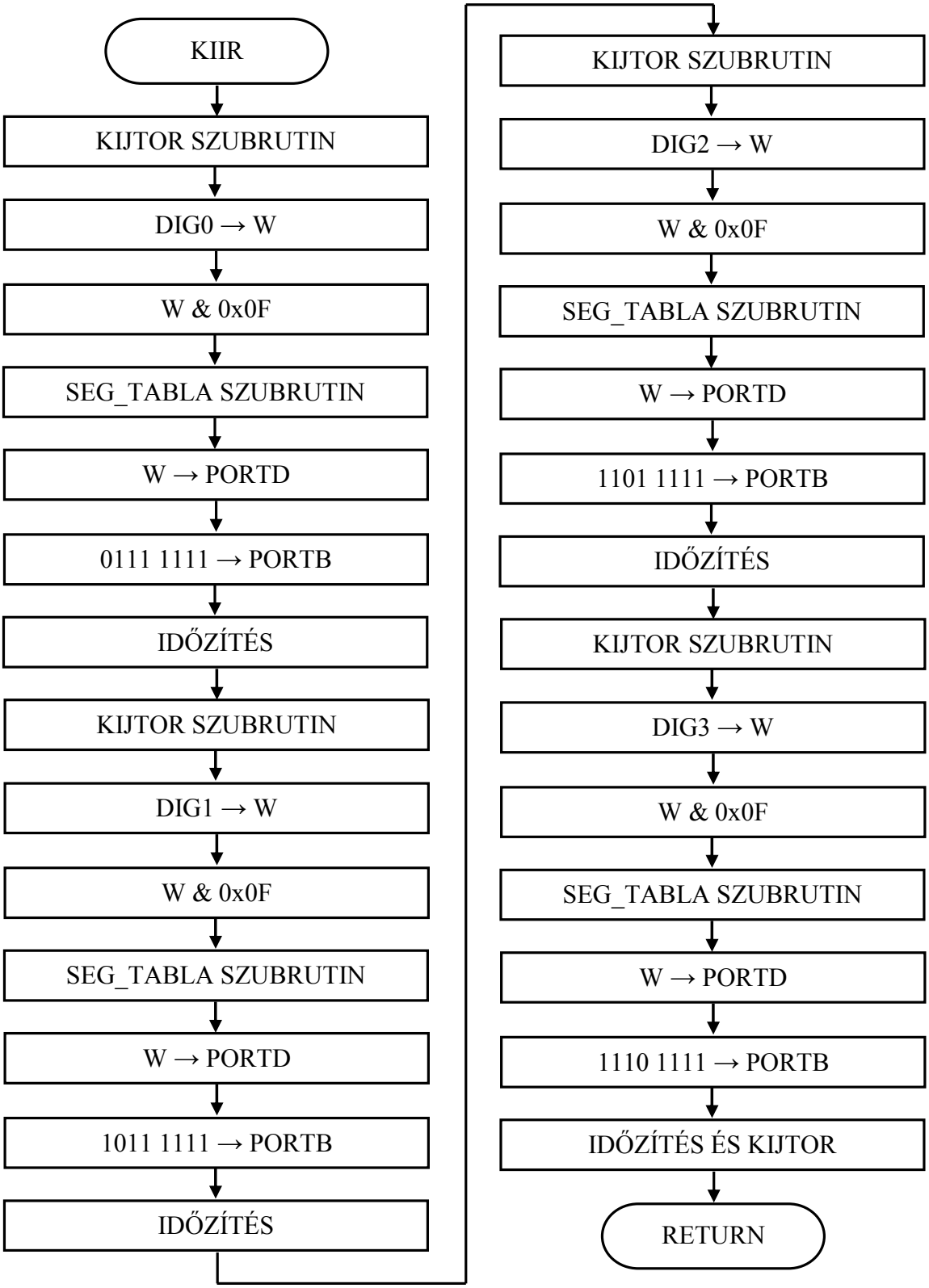
60. ábra

A 10-es számláló szubrutin.



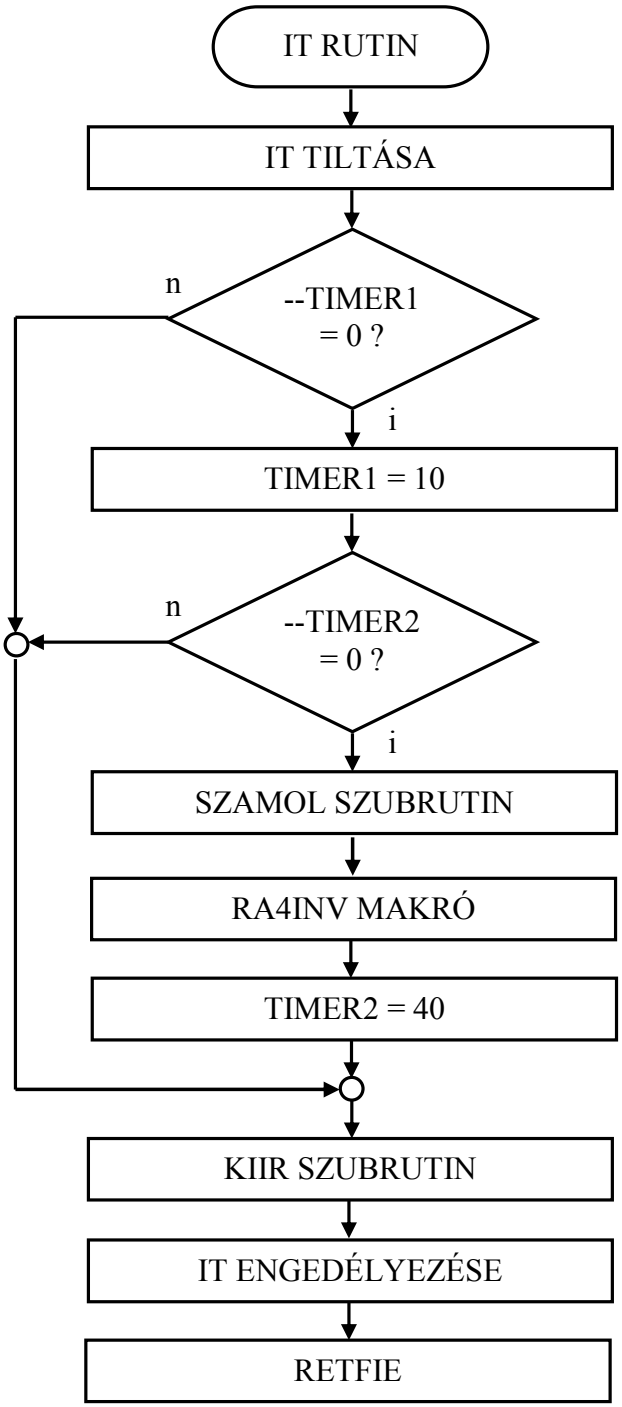
61. ábra

A 7 szegmenses kijelzőre kiíratő szubrutin.



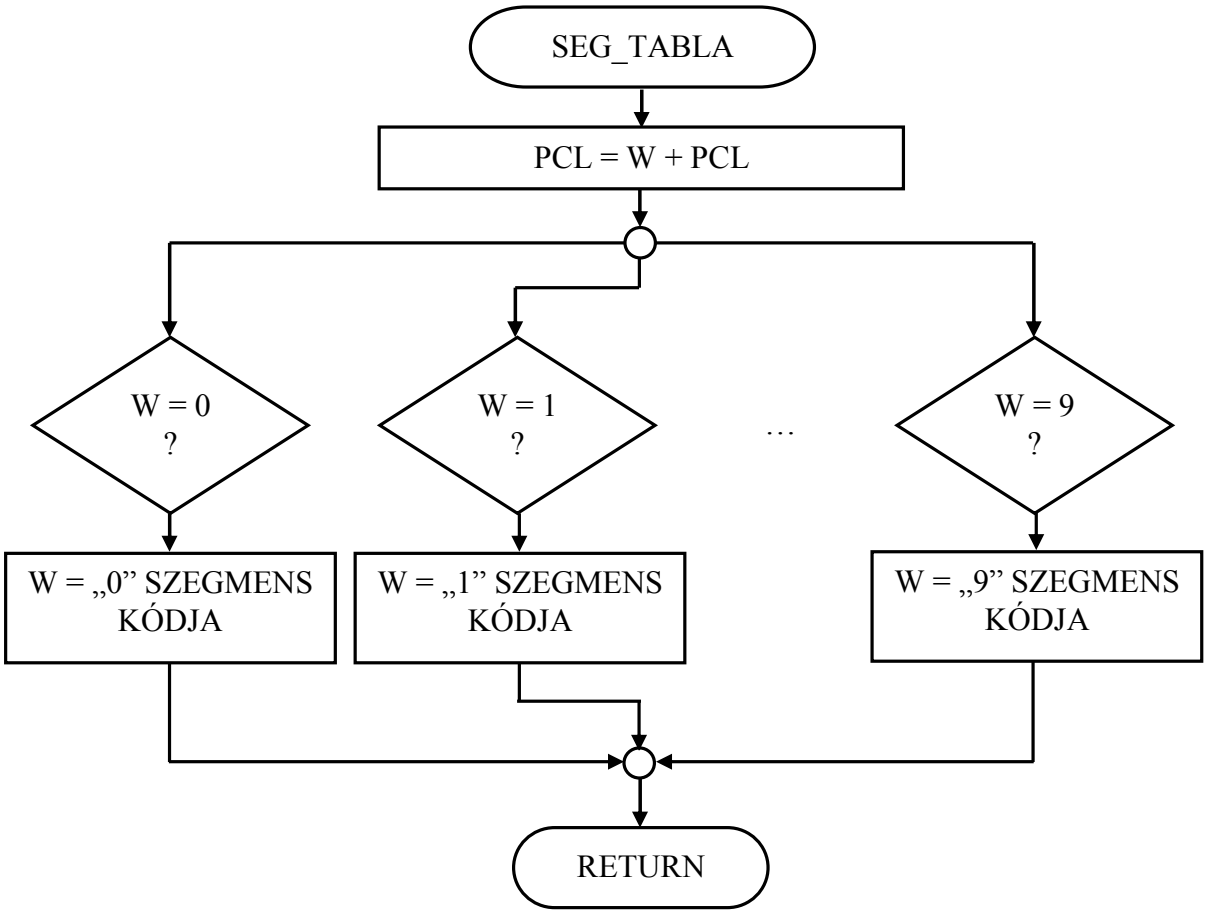
62. ábra

Megszakítást kezelő szubrutin.



63. ábra

A szegmens táblát kezelő szubrutin.



64. ábra

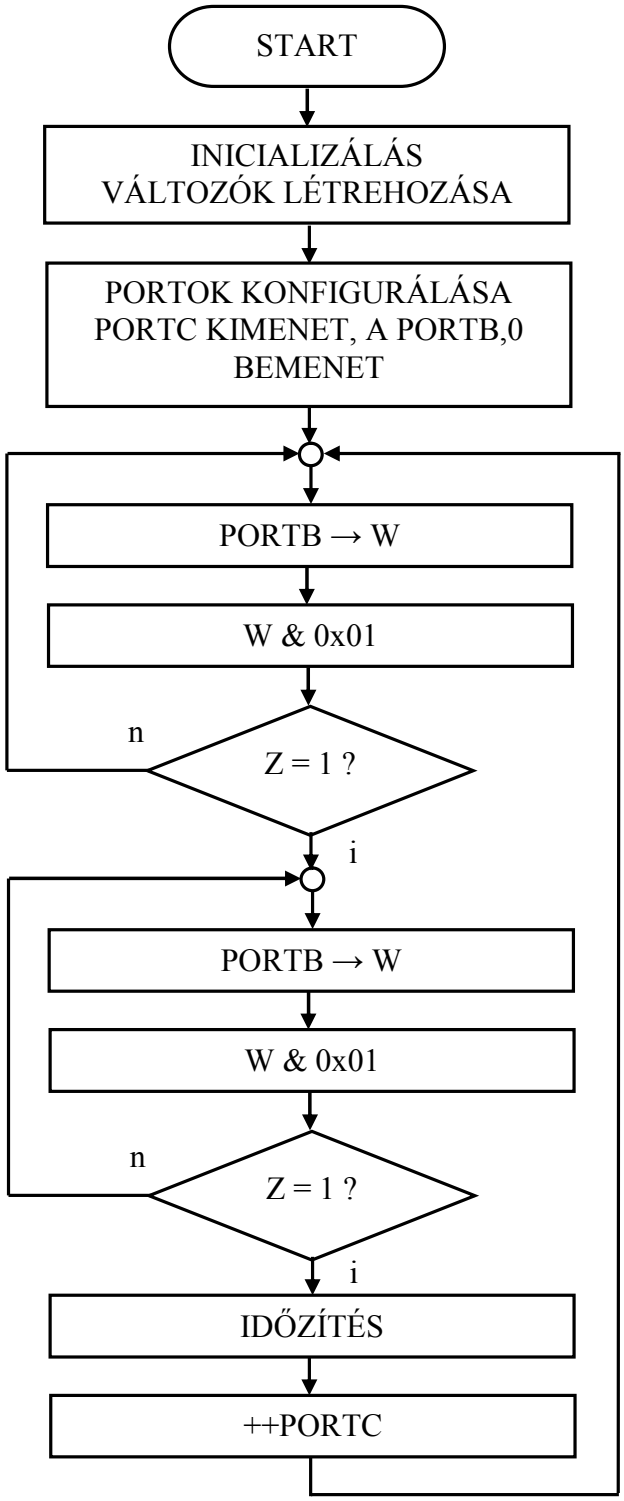
Megjegyzés:

- A feladat az eddigi ismereteket foglalja össze;
- A program bonyolultsága miatt, érdemes a programot, a folyamatábrákat szem előtt tartva lépésről-lépésre megoldani.

2. Programozási feladat – nyomógomb kezelés

Nyomógomb kezelés, egyszerű szintvizsgálattal. A program forráskódja (`NYGEGYSZ.ASM`) a függelékben elérhető.

Az elkészítendő program folyamatábrája.



65. ábra

Összefoglalás

A szakdolgozatom a középiskolai oktatás határait is figyelembe véve, dolgozza fel a mikrovezérlőkkel kapcsolatos ismereteket. Az eredeti célkitűzéseim többségét sikerült megvalósítani.

A mikrokontrollereket feldolgozó tananyag a jelenlegi formájában elkészült, az oktatáshoz eleddig egy ennek az írásműnek az alapjait jelentő prezentációt használtam. A továbbiakban szeretném a szakdolgozat tananyagrészét valódi tananyagként használni, és a csoportlétszámnak megfelelően sokszorosítani és bekötni. A segédanyag a továbbiakban az írás (jegyzetelés) nehézségétől mentesíti a tanulókat, ezért a programozásra így több idő maradhat.

A mikrokontrolleres gyakorló áramkör elkészült, az áramkör elkészítéséhez természetes módon felhasználtam az eredeti villamosmérnöki végzettségemet. A kapcsolás és az azt megvalósító nyomtatott áramköri panel is az én munkám. A továbbiakban szeretném az elkészült két panelből álló gyakorlókészletet kibővíteni, legalább öt áramkörre. Az órákon eddig is használtunk gyakorlópanelt, de az csak korlátozottan alkalmas a dolgozatban bemutatott programokhoz, mert nem tartalmaz bizonyos perifériaelemeket (7 szegmenses kijelző, LED sor).

A magas szintű programozási nyelv használatával azonban még adós maradtam. A témára fordítandó időt kevésnek találtam, a témakörök többsége olyan, hogy akár egy foglalkozás anyaga is kitölthetné az egész témára szánt oktatási időt. A tervbe vett C programozási nyelv oktatásához egy másik, sokkal bonyolultabb kontroller típust kell felhasználni (PIC 18F452), sajnos ennek az eszköznek a bemutatására már nem volt elegendő idő és hely.

Az érdeklődő diákok számára a következő tanévtől kezdődően egy mikrokontrolleres szakkör beindítását tervezem. A délutáni foglalkozások lehetővé teszik a programozási tananyagrészből kimaradó ismeretek feldolgozását és a PIC mikrovezérlőkön túl esetlegesen más kontroller típus (pl. Atmel) megismerését.

A tanulók a mikrovezérlőkkel kapcsolatos ismereteket hasznosnak tartják. Nagy kedvvel készítik a programokat, főleg az első működőképes alkalmazás okozza a legnagyobb sikerélményt. A tananyagrészek közül a programozási illetve a gyakorlati tananyagrészek azok, amelyek a leginkább kedveltek, ezért a továbbiakban szeretném a tananyagot legalább fele arányban, ebben az irányban továbbfejleszteni.

Több volt tanulóm, akik már főiskolán, egyetemen tanulnak, jelezték, hogy az ezeken, az órákon kapott ismerteteket közvetlen (villamosmérnök és informatikus hallgatók) és közvetett módon (programozó matematikus hallgatók) is fel tudták használni a tanulmányaikban.

Irodalomjegyzék

1. PIC mikrovezérlők alkalmazástechnikája – Dr. Kónya László ChipCad Kft.
2. A PIC16C mikrovezérlők – Dr. Madarász László GAMF.
3. Programozás 1 – Dr. Juhász István mobiDIÁK könyvtár.
4. DataSheet_30569b.pdf (PIC 16F870/871 28/40-pin, 8-Bit CMOS FLASH Microcontrollers) – Microchip Technology Inc.

Függelék

```
;SZAMOL.ASM
;PROGRAMIRO: BORSI ISTVAN 2007.04.14.
;DEMONSTRACIOS PROGRAM

;-------< DIREKTIVAK >--------

LIST            P=16F871                    ;PROCESSZOR DEFINICIO
INCLUDE         "P16F871.INC"               ;INCLUDE FAJL BEALLITAS


;-------< MAKRO DEFINICIO >--------

E_0_ON          MACRO                       ;A MAKRO DEFINIALASA
                BSF         E_REG,0         ;E_REG,0 = 1
                ENDM                        ;MAKRO VEGE

E_0_OFF         MACRO                       ;A MAKRO DEFINIALASA
                BCF         E_REG,0         ;E_REG,0 = 0
                ENDM                        ;MAKRO VEGE


;-------< VALTOZOK DEKLARACIOJA >--------

CBLOCK          0X20
                A_REG
                B_REG
                C_REG
                D_REG
                E_REG
ENDC
SZAML           EQU         0x30

;-------< PROGRAM KEZDETE >--------

                ORG         0X003           ;EZ A KEZDOCIM
                GOTO        MAIN            ;UGRAS A FOPROGRAMRA

MAIN
                CLRF        A_REG
                CLRF        B_REG
                CLRF        C_REG
                CLRF        D_REG

                MOVLW       0X31            ;W-BE 31H
                MOVWF       B_REG           ;B_REG-BE A W-T (31H)
                MOVLW       0X24            ;W-BE 24H
                MOVWF       A_REG           ;W-T AZ A_REG-BE (24H)
                ADDWF       B_REG,0         ;A_REG + B_REG -> W
                MOVWF       C_REG           ;W -> C_REG
                MOVF        A_REG,0         ;A_REG -> W
                SUBWF       B_REG,0         ;W = B_REG - W
                MOVWF       D_REG           ;W -> D_REG


                E_0_ON
                CALL        WAIT
                BCF         STATUS,0
                RLF         E_REG,1
                RLF         E_REG,1
                RLF         E_REG,1
                RLF         E_REG,1
                RRF         E_REG,1
                RRF         E_REG,1
                RRF         E_REG,1
                RRF         E_REG,1
                E_0_OFF

                CALL        WAIT8

VEGE            GOTO        VEGE

;-------< SZUBRUTINOK >--------
```

```
;SZOFTVERES KESLELTETES
WAIT            NOP
                NOP
                NOP
                RETURN                  ;A KESLELTETEO SZUBRUTIN VEGE

;SZOFTVERES KESLELTETES CIKLUSSAL
WAIT8           MOVLW     0x0A          ;M = 10
                MOVWF     SZAML         ;SZAML = M
UJRA            NOP                     ;N = 2
                NOP
                DECFSZ    SZAML,1       ;SZAML – 1 = 0?
                GOTO      UJRA
                RETURN                  ;A KESLELTETEO SZUBRUTIN VEGE

                END                     ;PROGRAM VEGE


;VILLOGO.ASM
;PROGRAMIRO: BORSI ISTVAN 2007.04.14.
;A PORTC,0-AS BITJET VILLOGTATJA

;-------< DIREKTIVAK >--------

LIST            P=16F871                ;PROCESSZOR DEFINICIO
INCLUDE         "P16F871.INC"           ;INCLUDE FAJL BEALLITAS


;-------< MAKRO DEFINICIO >--------



;-------< VALTOZOK DEKLARACIOJA >--------

SZAML1          EQU       0x20
SZAML2          EQU       0x21

;-------< PROGRAM KEZDETE >--------

                ORG       0X003         ;EZ A KEZDOCIM
                GOTO      MAIN          ;UGRAS A FOPROGRAMRA

MAIN
                BANKSEL   TRISC         ;BANK1 KIVALASZTASA
                CLRF      TRISC         ;PORTC MINDEN BITJE KIMENET
                BANKSEL   PORTC         ;BANK0 KIVALASZTASA

LOOP            BSF       PORTC,0       ;PORTC,0 BITJE 1 (LED VILAGIT)
                CALL      WAIT          ;0,5S-OS IDOZITES
                BCF       PORTC,0       ;PORTC,0 BITJE 0 (LED KIALSZIK)
                CALL      WAIT          ;0,5S-OS IDOZITES
                GOTO      LOOP          ;AZ ELEJERE UGRAS

;-------< SZUBRUTINOK >--------

;ROVID SZOFTVERES KESLELTETES
WAIT            NOP                     ;A SZIMULACIO MIATT
                RETURN

;0,5S-OS SZOFTVERES KESLELTETES - PIC-ES GYAKORLOBA
WAIT_05         MOVLW     0xFF          ;L = 255
                MOVWF     SZAML1        ;SZAML1 = L
ALAP            MOVLW     0xFF          ;M = 255
                MOVWF     SZAML2        ;SZAML2 = M
UJRA            NOP                     ;N = 16
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
```

```
                NOP
                NOP
                NOP
                DECFSZ      SZAML2,1        ;SZAML2 – 1 = 0?
                GOTO        UJRA            ;VISSZA
                DECFSZ      SZAML1,1        ;SZAML1 – 1 = 0?
                GOTO        ALAP            ;SZAML1 ALAPERTEK
                RETURN                      ;A KESLELTETEO SZUBRUTIN VEGE

                END                         ;PROGRAM VEGE

;FFENYSH.ASM
;PROGRAMIRO: BORSI ISTVAN 2007.04.14.
;A PORTC ALSO 6 BITJEN FUTOFENY

;-------< DIREKTIVAK >--------

LIST            P=16F871                    ;PROCESSZOR DEFINICIO
INCLUDE         "P16F871.INC"               ;INCLUDE FAJL BEALLITAS

;-------< MAKRO DEFINICIO >--------



;-------< VALTOZOK DEKLARACIOJA >--------

SZAML1          EQU         0x20
SZAML2          EQU         0x21
SZAML           EQU         0x22

;-------< PROGRAM KEZDETE >--------

                ORG         0X003           ;EZ A KEZDOCIM
                GOTO        MAIN            ;UGRAS A FOPROGRAMRA

MAIN
                BANKSEL     TRISC           ;BANK1 KIVALASZTASA
                CLRF        TRISC           ;PORTC MINDEN BITJE KIMENET
                BANKSEL     PORTC           ;BANK0 KIVALASZTASA
                CLRF        PORTC           ;PORTC TORLESE

                BCF         STATUS,C        ;CY TORLESE, NE FOROGJON BE 1
                BSF         PORTC,0         ;PORTC,0 BITJE 1 (LED VILAGIT)
                CALL        WAIT_05         ;IDOZITES INDITASA
BAL_F           MOVLW       0x5             ;FORGATASOK SZAMA
                MOVWF       SZAML           ;ALAPERTEK BETOLTESE
BALRA           RLF         PORTC,1         ;1-EL BALRA
                CALL        WAIT_05         ;IDOZITES INDITASA
                DECFSZ      SZAML           ;--SZAML3
                GOTO        BALRA           ;SZAML3 = 0
                GOTO        JOBB_F          ;SZAML3 <> 0

JOBB_F          MOVLW       0x5             ;FORGATASOK SZAMA
                MOVWF       SZAML           ;ALAPERTEK BETOLTESE
JOBBRA          RRF         PORTC,1         ;1-EL JOBBRA
                CALL        WAIT_05         ;IDOZITES INDITASA
                DECFSZ      SZAML           ;--SZAML3
                GOTO        JOBBRA          ;SZAML3 = 0
                GOTO        BAL_F           ;SZAML3 <> 0

;-------< SZUBRUTINOK >--------

;ROVID SZOFTVERES KESLELTETES
WAIT            NOP                         ;A SZIMULACIO MIATT
                RETURN

;0,5S-OS SZOFTVERES KESLELTETES - PIC-ES GYAKORLOBA
WAIT_05         MOVLW       0xFF            ;L = 255
                MOVWF       SZAML1          ;SZAML1 = L
ALAP            MOVLW       0xFF            ;M = 255
                MOVWF       SZAML2          ;SZAML2 = M
UJRA            NOP                         ;N = 16
                NOP
                NOP
                NOP
                NOP
                NOP
```

```
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                DECFSZ      SZAML2,1          ;SZAML2 – 1 = 0?
                GOTO        UJRA              ;VISSZA
                DECFSZ      SZAML1,1          ;SZAML1 – 1 = 0?
                GOTO        ALAP              ;SZAML1 ALAPERTEK
                RETURN                        ;A KESLELTETEO SZUBRUTIN VEGE

                END                           ;PROGRAM VEGE

;FFENYUT.ASM
;PROGRAMIRO: BORSI ISTVAN 2007.04.14.
;A PORTC ALSO 6 BITJEN FUTOFENY UTEMEZESSEL

;-------< DIREKTIVAK >--------

LIST            P=16F871                      ;PROCESSZOR DEFINICIO
INCLUDE         "P16F871.INC"                 ;INCLUDE FAJL BEALLITAS

;-------< MAKRO DEFINICIO >--------



;-------< VALTOZOK DEKLARACIOJA >--------

SZAML1          EQU         0x20
SZAML2          EQU         0x21

;-------< PROGRAM KEZDETE >--------

                ORG         0X003             ;EZ A KEZDOCIM
                GOTO        MAIN              ;UGRAS A FOPROGRAMRA

MAIN
                BANKSEL     TRISC             ;BANK1 KIVALASZTASA
                CLRF        TRISC             ;PORTC MINDEN BITJE KIMENET
                BANKSEL     PORTC             ;BANK0 KIVALASZTASA

LOOP            MOVLW       B'00000001'       ;AZ 1. UTEM
                MOVWF       PORTC             ;PORTC,0 BITJE 1 (LED VILAGIT)
                CALL        WAIT_05           ;IDOZITES INDITASA
                MOVLW       B'00000010'       ;AZ 2. UTEM
                MOVWF       PORTC             ;PORTC,1 BITJE 1 (LED VILAGIT)
                CALL        WAIT_05           ;IDOZITES INDITASA
                MOVLW       B'00000100'       ;AZ 3. UTEM
                MOVWF       PORTC             ;PORTC,2 BITJE 1 (LED VILAGIT)
                CALL        WAIT_05           ;IDOZITES INDITASA
                MOVLW       B'00001000'       ;AZ 4. UTEM
                MOVWF       PORTC             ;PORTC,3 BITJE 1 (LED VILAGIT)
                CALL        WAIT_05           ;IDOZITES INDITASA
                MOVLW       B'00010000'       ;AZ 5. UTEM
                MOVWF       PORTC             ;PORTC,4 BITJE 1 (LED VILAGIT)
                CALL        WAIT_05           ;IDOZITES INDITASA
                MOVLW       B'00100000'       ;AZ 6. UTEM
                MOVWF       PORTC             ;PORTC,5 BITJE 1 (LED VILAGIT)
                CALL        WAIT_05           ;IDOZITES INDITASA
                MOVLW       B'00010000'       ;AZ 7. UTEM
                MOVWF       PORTC             ;PORTC,4 BITJE 1 (LED VILAGIT)
                CALL        WAIT_05           ;IDOZITES INDITASA
                MOVLW       B'00001000'       ;AZ 8. UTEM
                MOVWF       PORTC             ;PORTC,3 BITJE 1 (LED VILAGIT)
                CALL        WAIT_05           ;IDOZITES INDITASA
                MOVLW       B'00000100'       ;AZ 9. UTEM
                MOVWF       PORTC             ;PORTC,2 BITJE 1 (LED VILAGIT)
                CALL        WAIT_05           ;IDOZITES INDITASA
                MOVLW       B'00000010'       ;AZ 10. UTEM
                MOVWF       PORTC             ;PORTC,1 BITJE 1 (LED VILAGIT)
                CALL        WAIT_05           ;IDOZITES INDITASA
                GOTO        LOOP              ;VISSZA AZ ELEJERE
```

```
;-------< SZUBRUTINOK >--------

;ROVID SZOFTVERES KESLELTETES
WAIT            NOP                         ;A SZIMULACIO MIATT
                RETURN

;0,5S-OS SZOFTVERES KESLELTETES - PIC-ES GYAKORLOBA
WAIT_05         MOVLW       0xFF            ;L = 255
                MOVWF       SZAML1          ;SZAML1 = L
ALAP            MOVLW       0xFF            ;M = 255
                MOVWF       SZAML2          ;SZAML2 = M
UJRA            NOP                         ;N = 16
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                DECFSZ      SZAML2,1        ;SZAML2 – 1 = 0?
                GOTO        UJRA            ;VISSZA
                DECFSZ      SZAML1,1        ;SZAML1 – 1 = 0?
                GOTO        ALAP            ;SZAML1 ALAPERTEK
                RETURN                      ;A KESLELTETEO SZUBRUTIN VEGE

                END                         ;PROGRAM VEGE


;JELZOL.ASM
;PROGRAMIRO: BORSI ISTVAN 2007.04.14.
;A PORTA ALSO 3 BITJEN JELZOLAMPA MODELL
;PORTA,3 PIROS; PORTA,2 SARGA; PORTA,1 ZOLD;

;-------< DIREKTIVAK >--------

LIST            P=16F871                    ;PROCESSZOR DEFINICIO
INCLUDE         "P16F871.INC"               ;INCLUDE FAJL BEALLITAS
ERRORLEVEL      2

;-------< MAKRO DEFINICIO >--------



;-------< VALTOZOK DEKLARACIOJA >--------

SZAML1          EQU         0x20
SZAML2          EQU         0x21

;-------< PROGRAM KEZDETE >--------

                ORG         0X003           ;EZ A KEZDOCIM
                GOTO        MAIN            ;UGRAS A FOPROGRAMRA

MAIN

                BANKSEL     PORTA           ;PORTA KIVALASZTAS
                CLRF        PORTA           ;PORTA TORLES
                CLRF        PORTC           ;PORTC TORLES
                BANKSEL     ADCON1          ;PORTA KONFIGURALASA
                MOVLW       0x06            ;MINDEN PORTA
                MOVWF       ADCON1          ;DIGITALIS I/O
                CLRF        TRISA           ;TRIS ERTEK BETOLTES
                BANKSEL     PORTA           ;PORTA KIVALASZTASA


LOOP            MOVLW       B'00011000'     ;AZ 1. UTEM
                MOVWF       PORTA           ;PORTA,3 BITJE 1 (PIROS)
                CALL        WAIT_05         ;IDOZITES INDITASA
```

```
                CALL        WAIT_05          ;IDOZITES INDITASA
                CALL        WAIT_05          ;IDOZITES INDITASA
                CALL        WAIT_05          ;IDOZITES INDITASA
                CALL        WAIT_05          ;IDOZITES INDITASA
                CALL        WAIT_05          ;IDOZITES INDITASA
                MOVLW       B'00011100'      ;AZ 2. UTEM
                MOVWF       PORTA            ;PORTA,3 ES 2 BITJE 1 (PIROS-SARGA)
                CALL        WAIT_05          ;IDOZITES INDITASA
                CALL        WAIT_05          ;IDOZITES INDITASA
                MOVLW       B'00010010'      ;AZ 3. UTEM
                MOVWF       PORTA            ;PORTA,2 BITJE 1 (ZOLD)
                CALL        WAIT_05          ;IDOZITES INDITASA
                CALL        WAIT_05          ;IDOZITES INDITASA
                CALL        WAIT_05          ;IDOZITES INDITASA
                CALL        WAIT_05          ;IDOZITES INDITASA
                CALL        WAIT_05          ;IDOZITES INDITASA
                CALL        WAIT_05          ;IDOZITES INDITASA
                MOVLW       B'00010100'      ;AZ 4. UTEM
                MOVWF       PORTA            ;PORTA,2 BITJE 1 (SARGA)
                CALL        WAIT_05          ;IDOZITES INDITASA
                CALL        WAIT_05          ;IDOZITES INDITASA
                GOTO        LOOP             ;VISSZA AZ ELEJERE

;-------< SZUBRUTINOK >--------

;ROVID SZOFTVERES KESLELTETES
WAIT            NOP                          ;A SZIMULACIO MIATT
                RETURN

;0,5S-OS SZOFTVERES KESLELTETES - PIC-ES GYAKORLOBA
WAIT_05         MOVLW       0xFF             ;L = 255
                MOVWF       SZAML1           ;SZAML1 = L
ALAP            MOVLW       0xFF             ;M = 255
                MOVWF       SZAML2           ;SZAML2 = M
UJRA            NOP                          ;N = 16
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                DECFSZ      SZAML2,1         ;SZAML2 – 1 = 0?
                GOTO        UJRA             ;VISSZA
                DECFSZ      SZAML1,1         ;SZAML1 – 1 = 0?
                GOTO        ALAP             ;SZAML1 ALAPERTEK
                RETURN                       ;A KESLELTETEO SZUBRUTIN VEGE

                END                          ;PROGRAM VEGE


;VILLOGIT.ASM
;PROGRAMIRO: BORSI ISTVAN 2007.04.16.
;A PORTC,0-AS BITJET VILLOGTATJA TIMER0 IT HASZNALATAVAL

;-------< DIREKTIVAK >--------

                LIST        P=16F871         ;PROCESSZOR DEFINICIO
                INCLUDE     "P16F871.INC"    ;INCLUDE FAJL BEALLITAS
                ERRORLEVEL  2                ;CSAK HIBAUZENETEK JELENNEK MEG

;-------< MAKRO DEFINICIO >--------

LINV            MACRO                        ;A PORTC,0-AT VILLOGTATJA
                BANKSEL     PORTC            ;PORTC KIVALASZTASA
                COMF        PORTC            ;PORTC INVERTALASA
                MOVF        PORTC,0          ;PORTC A W-BE
                ANDLW       B'00000001'      ;ES-ELNI KELL A W-T
                MOVWF       PORTC            ;PORTRC-RE KIHELYEZI A W-T
```

```
                ENDM

;-------< VALTOZOK DEKLARACIOJA >--------

TIMER           EQU         0x20            ;BELSO SZAMLALOHOZ

;-------< PROGRAM KEZDETE >--------

                ORG         0X003           ;EZ A KEZDOCIM
                GOTO        MAIN            ;UGRAS A FOPROGRAMRA
                GOTO        ITSERV          ;UGRAS AZ IT SZUBRUTINRA

MAIN
                BANKSEL     TRISC           ;BANK1 KIVALASZTASA
                CLRF        TRISC           ;PORTC MINDEN BITJE KIMENET
                BANKSEL     PORTC           ;BANK0 KIVALASZTASA

;TIMER0 INICIALIZÁLÁSA
                BANKSEL     TMR0            ;BANK0-BAN VAN A TMR0
                CLRF        TMR0            ;TMR0 REGISZTER TORLESE
                CLRF        INTCON          ;IT-K TILTASA
                BANKSEL     OPTION_REG      ;BANK1
                MOVLW       B'10000011'     ;TMR0 -> BELSO,(OSC/4) 1:16 ELOOSZTO
                MOVWF       OPTION_REG      ;BETOLTES
                BSF         INTCON,T0IE     ;TMR0 IT ENGEDELYEZES
                BSF         INTCON,GIE      ;MINDEN IT EGEDELYEZESE
                MOVLW       D'200'          ;BELSO SZAMLALO ERTEKE
                MOVWF       TIMER           ;BETOLTESE


LOOP            CALL        WAIT            ;EZ A FOPROGRAM!
                GOTO        LOOP

;-------< SZUBRUTINOK >--------

;ROVID SZOFTVERES KESLELTETES
WAIT            NOP                         ;KIS KESLELTETES
                RETURN


;-------< MEGSZAKÍTÁSI SZUBRUTINOK >--------

ITSERV
                BANKSEL     INTCON          ;BANKVALTAS
                BCF         INTCON,GIE      ;MINDEN IT TILTASA
                DECFSZ      TIMER           ;SZAMLALJA A MEGSZAKITASOK SZAMAT
                GOTO        KI              ;KI HA NEM VEGERTEK
                LINV                        ;LEDVILLOGTATO MAKRO
                MOVLW       D'200'          ;ALAPERTEK
                MOVWF       TIMER           ;SZAMLALO ALAPERTEKENEK VISSZATOLTESE
KI              BCF         INTCON,T0IF     ;FLAG TORLESE
                BSF         INTCON,T0IE     ;MINDEN IT ENGEDELYEZESE
                RETFIE                      ;MEGSZAKITAS VEGE

                END                         ;PROGRAM VEGE


;KIJMPX.ASM
;PROGRAMIRO: BORSI ISTVAN 2007.04.16.
;A 7 SZEGMENSES KIJELZON FELFELE SZAMLALO KIALAKITASA

;-------< DIREKTIVAK >--------

                LIST        P=16F871        ;PROCESSZOR DEFINICIO
                INCLUDE     "P16F871.INC"   ;INCLUDE FAJL BEALLITAS
                ERRORLEVEL  2               ;CSAK HIBAUZENETEK JELENNEK MEG

;-------< MAKRO DEFINICIO >--------

RA4INV          MACRO                       ;A PORTA,4-ET VILLOGTATJA
                BANKSEL     PORTA           ;BANKVALTAS
                COMF        PORTA           ;PORTA INVERTALASA
                MOVF        PORTA,4         ;PORTA A W-BE
                ANDLW       B'00010000'     ;ES-ELNI KELL A W-T
                MOVWF       PORTA           ;PORTRA-RA KIHELYEZI A W-T
                ENDM
```

```
KIJTOR          MACRO                       ;LETORLI A KIJELZOT
                BANKSEL     PORTD           ;BANKVALTAS
                MOVLW       B'11111111'     ;MINDEN MAGASZINTU
                MOVWF       PORTD           ;EGYIK SZEGMENS SEM VILAGIT
                MOVWF       PORTB           ;EGYIK DIGIT SEM AKTIV
                ENDM

;-------< VALTOZOK DEKLARACIOJA >--------

CBLOCK          0X20
                DIG0
                DIG1
                DIG2
                DIG3
                TIMER1
                TIMER2
                TIMER3
                SZAML1
                SZAML10
                SZAML100
                SZAML1000
                MENT
ENDC

;-------< PROGRAM KEZDETE >--------

                ORG         0X003           ;EZ A KEZDOCIM
                GOTO        MAIN            ;UGRAS A FOPROGRAMRA
                GOTO        ITSERV          ;UGRAS AZ IT SZUBRUTINRA

MAIN

;TIMER0 INICIALIZALASA
                BANKSEL     TMR0            ;BANK0-BAN VAN A TMR0
                CLRF        TMR0            ;TMR0 REGISZTER TORLESE
                CLRF        INTCON          ;IT-K TILTASA
                BANKSEL     OPTION_REG      ;BANK1
                MOVLW       B'10000001'     ;TMR0 -> BELSO,(OSC/4) 1:2 ELOOSZTO
                MOVWF       OPTION_REG      ;BETOLTES
                BSF         INTCON,T0IE     ;TMR0 IT ENGEDELYEZES
                BSF         INTCON,GIE      ;MINDEN IT EGEDELYEZESE
                MOVLW       D'10'           ;BELSO SZAMLALO ERTEKE
                MOVWF       TIMER1          ;BETOLTESE
                MOVLW       D'40'           ;BELSO SZAMLALO ERTEKE
                MOVWF       TIMER2          ;BETOLTESE

;PORTOK KONFIGURALASA
                BANKSEL     TRISA           ;BANK1 KIVALASZTASA
                MOVLW       0x06            ;MINDEN PORTA
                MOVWF       ADCON1          ;DIGITALIS I/O
                CLRF        TRISA           ;PORTA MINDEN BITJE KIMENET
                CLRF        TRISC           ;PORTC MINDEN BITJE KIMENET
                CLRF        TRISD           ;PORTD MINDEN BITJE KIMENET
                MOVLW       B'00001111'     ;PORTB IRANYVALASZTASAHOZ
                MOVWF       TRISB           ;PORTB ALSO4 BEMENET FELSO4 KIMENET
                BANKSEL     PORTA           ;PORTA KIVALASZTAS
                CLRF        PORTA           ;PORTA TORLES
                CLRF        PORTC           ;PORTC TORLES

                CLRF        DIG0            ;0-AS DIGIT TORLESE
                CLRF        DIG1            ;1-ES DIGIT TORLESE
                CLRF        DIG2            ;2-ES DIGIT TORLESE
                CLRF        DIG3            ;3-AS DIGIT TORLESE

                MOVLW       D'10'           ;MINDEN SZAMLALO 10-ES
                MOVWF       SZAML1          ;SZAMLALO ALAPERTEKEK
                MOVWF       SZAML10
                MOVWF       SZAML100
                MOVWF       SZAML1000


LOOP            NOP                         ;EZ A FOPROGRAM!
                GOTO        LOOP

;-------< SZUBRUTINOK >--------

;8 BITES SZOFTVERES KESLELTETES A KIIRATASHOZ
```

```
WAIT            NOP
                NOP
                NOP
                DECFSZ      TIMER3
                GOTO        WAIT
                RETURN

;EGYMASBA AGYAZOTT 10-ES SZAMLALOK
SZAMOL          DECFSZ      SZAML1          ;SZAML1 - 1
                GOTO        SZAMOL1         ;HA NEM A 10. AKKOR LEPTETI A DIG0-AT
                GOTO        ALAP1           ;A 10. EZERT EGY DIGITTEL FELJEBB LEP

SZAMOL1         INCF        DIG0            ;LEPTETI FELFELE A DIG0-AT
                GOTO        VEGE            ;NINCS TOBB FELADAT

ALAP1           MOVLW       D'10'           ;SZAML1 ALAPERTEK
                MOVWF       SZAML1          ;A KOVETKEZO CIKLUSHOZ
                CLRF        DIG0            ;DIG0 TORLESE
                DECFSZ      SZAML10         ;SZAML10 - 1
                GOTO        SZAMOL10        ;HA NEM A 10. AKKOR LEPTETI A DIG1-ET
                GOTO        ALAP10          ;A 10. EZERT EGY DIGITTEL FELJEBB LEP

SZAMOL10        INCF        DIG1            ;LEPTETI FELFELE A DIG1-ET
                GOTO        VEGE            ;NINCS TOBB FELADAT

ALAP10          MOVLW       D'10'           ;SZAML10 ALAPERTEK
                MOVWF       SZAML10         ;A KOVETKEZO CIKLUSHOZ
                CLRF        DIG1            ;DIG1 TORLESE
                DECFSZ      SZAML100        ;SZAML100 - 1
                GOTO        SZAMOL100       ;HA NEM A 10. AKKOR LEPTETI A DIG2-OT
                GOTO        ALAP100         ;A 10. EZERT EGY DIGITTEL FELJEBB LEP

SZAMOL100       INCF        DIG2            ;LEPTETI FELFELE A DIG2-OT
                GOTO        VEGE            ;NINCS TOBB FELADAT

ALAP100         MOVLW       D'10'           ;SZAML100 ALAPERTEK
                MOVWF       SZAML100        ;A KOVETKEZO CIKLUSHOZ
                CLRF        DIG2            ;DIG2 TORLESE
                DECFSZ      SZAML1000       ;SZAML1000 - 1
                GOTO        SZAMOL1000      ;HA NEM A 10. AKKOR LEPTETI A DIG3-AT
                GOTO        ALAP1000        ;A 10. EZERT EGY DIGITTEL FELJEBB LEP

SZAMOL1000      INCF        DIG3            ;LEPTETI FELFELE A DIG3-AT
                GOTO        VEGE            ;NINCS TOBB FELADAT

ALAP1000        MOVLW       D'10'           ;SZAML1000 ALAPERTEK
                MOVWF       SZAML1000       ;A KOVETKEZO CIKLUSHOZ
                CLRF        DIG3            ;DIG3 TORLESE
VEGE            RETURN                      ;VISSZA


;KIIRATO RUTIN
KIIR
                KIJTOR                      ;KIJELZO TORLESE MAKRO
                MOVFW       DIG0            ;A JOBB SZELSO DIGIT
                ANDLW       0X0F            ;MASZKOLAS AZ ELSO 4 BITEN A SZAMJEGY SORSZAMA
                CALL        SEG_TABLA       ;A SEG_TABLA-BOL A SORSZAMNAK MEGFELELO ERTEK
                MOVWF       PORTD           ;A SZEGMENSERTEK A KIJELZORE
                MOVLW       B'01111111'     ;A DIG0 KIVALASZTASA (0-RA AKTIV)
                MOVWF       PORTB           ;A DIG0 AKTIV
                CALL        WAIT            ;KESLELTESTES, HOGY LATSZODJON

                KIJTOR                      ;KIJELZO TORLESE MAKRO
                MOVFW       DIG1            ;JOBBROL KOVETKEZO DIGIT
                ANDLW       0X0F            ;MASZKOLAS AZ ELSO 4 BITEN A SZAMJEGY SORSZAMA
                CALL        SEG_TABLA       ;A SEG_TABLA-BOL A SORSZAMNAK MEGFELELO ERTEK
                MOVWF       PORTD           ;A SZEGMENSERTEK A KIJELZORE
                MOVLW       B'10111111'     ;A DIG1 KIVALASZTASA (0-RA AKTIV)
                MOVWF       PORTB           ;A DIG1 AKTIV
                CALL        WAIT            ;KESLELTESTES, HOGY LATSZODJON

                KIJTOR                      ;KIJELZO TORLESE MAKRO
                MOVFW       DIG2            ;JOBBROL KOVETKEZO DIGIT
                ANDLW       0X0F            ;MASZKOLAS AZ ELSO 4 BITEN A SZAMJEGY SORSZAMA
                CALL        SEG_TABLA       ;A SEG_TABLA-BOL A SORSZAMNAK MEGFELELO ERTEK
                MOVWF       PORTD           ;A SZEGMENSERTEK A KIJELZORE
                MOVLW       B'11011111'     ;A DIG2 KIVALASZTASA (0-RA AKTIV)
```

```
                MOVWF       PORTB           ;A DIG2 AKTIV
                CALL        WAIT            ;KESLELTESTES, HOGY LATSZODJON

                KIJTOR                      ;KIJELZO TORLESE MAKRO
                MOVFW       DIG3            ;UTOLSO, LEGBALOLDALIBB DIGIT
                ANDLW       0X0F            ;MASZKOLAS AZ ELSO 4 BITEN A SZAMJEGY SORSZAMA
                CALL        SEG_TABLA       ;A SEG_TABLA-BOL A SORSZAMNAK MEGFELELO ERTEK
                MOVWF       PORTD           ;A SZEGMENSERTEK A KIJELZORE
                MOVLW       B'11101111'     ;A DIG3 KIVALASZTASA (0-RA AKTIV)
                MOVWF       PORTB           ;A DIG3 AKTIV
                CALL        WAIT            ;KESLELTESTES, HOGY LATSZODJON

                KIJTOR                      ;KIJELZO TORLESE MAKRO NE MARADJON AZ UTOLSO
                RETURN                      ;VISSZATERES

;SZEGMENS TABLAZAT
SEG_TABLA                                   ;A W-BEN A KIIRATANDO TARTALOM A PC EZZEL AZ
                ADDWF       PCL,1           ;ERTEKKEL ELTOLODIK A MEGFELELO SORRA
;                             decafbgp      ;SZEGMENS HOZZARENDELES
                RETLW       B'00000011'     ;W-BEN A 0 SZEGMENS KOMBINACIO, EZT ADJA VISSZA
                RETLW       B'11011011'     ;W-BEN A 1 SZEGMENS KOMBINACIO, EZT ADJA VISSZA
                RETLW       B'00101001'     ;W-BEN A 2 SZEGMENS KOMBINACIO, EZT ADJA VISSZA
                RETLW       B'01001001'     ;W-BEN A 3 SZEGMENS KOMBINACIO, EZT ADJA VISSZA
                RETLW       B'11010001'     ;W-BEN A 4 SZEGMENS KOMBINACIO, EZT ADJA VISSZA
                RETLW       B'01000101'     ;W-BEN A 5 SZEGMENS KOMBINACIO, EZT ADJA VISSZA
                RETLW       B'00000101'     ;W-BEN A 6 SZEGMENS KOMBINACIO, EZT ADJA VISSZA
                RETLW       B'11001011'     ;W-BEN A 7 SZEGMENS KOMBINACIO, EZT ADJA VISSZA
                RETLW       B'00000001'     ;W-BEN A 8 SZEGMENS KOMBINACIO, EZT ADJA VISSZA
                RETLW       B'01000001'     ;W-BEN A 9 SZEGMENS KOMBINACIO, EZT ADJA VISSZA


;-------< MEGSZAKÍTÁSI SZUBRUTINOK >--------

;TIMER0 IT RUTINJA
ITSERV
                MOVWF       MENT            ;W MENTESE, NE OKOZZON PROBLEMAT
                BANKSEL     INTCON          ;BANKVALTAS
                BCF         INTCON,GIE      ;MINDEN IT TILTASA
                DECFSZ      TIMER1          ;SZAMLALJA A MEGSZAKITASOK SZAMAT
                GOTO        KI              ;KIIRATAS ES IT VEGE

                MOVLW       D'10'           ;ALAPERTEK
                MOVWF       TIMER1          ;SZAMLALO ALAPERTEKENEK VISSZATOLTESE
                DECFSZ      TIMER2          ;SZAMLALJA A MEGSZAKITASOK SZAMAT
                GOTO        KI              ;KIIRATAS ES IT VEGE
                CALL        SZAMOL          ;A 10-ES SZAMLALOK LEPTETESE
                RA4INV                      ;LEDVILLOGTATO MAKRO
                MOVLW       D'40'           ;ALAPERTEK
                MOVWF       TIMER2          ;SZAMLALO ALAPERTEKENEK VISSZATOLTESE

KI              CALL        KIIR            ;KIJELZOKRE A SZAMLALO TARTALMAK
                BCF         INTCON,T0IF     ;FLAG TORLESE
                BSF         INTCON,T0IE     ;MINDEN IT ENGEDELYEZESE
                MOVFW       MENT
                RETFIE                      ;MEGSZAKITAS VEGE

                END                         ;PROGRAM VEGE


;NYGEGYSZ.ASM
;PROGRAMIRO: BORSI ISTVAN 2007.04.14.
;NYOMOGOMB KEZELES A NB0 LENYOMASARA SZAMLALAS PORTC-N

;-------< DIREKTIVAK >--------

LIST            P=16F871                    ;PROCESSZOR DEFINICIO
INCLUDE         "P16F871.INC"               ;INCLUDE FAJL BEALLITAS


;-------< MAKRO DEFINICIO >--------



;-------< VALTOZOK DEKLARACIOJA >--------

SZAML1          EQU         0x20
SZAML2          EQU         0x21
```

```
;-------< PROGRAM KEZDETE >--------

                ORG         0X003           ;EZ A KEZDOCIM
                GOTO        MAIN            ;UGRAS A FOPROGRAMRA

MAIN
                BANKSEL     PORTB           ;BANK1 KIVALASZTASA
                CLRF        PORTB
                CLRF        PORTC
                BANKSEL     TRISB           ;BANK1 KIVALASZTASA
                MOVLW       1               ;0. BIT KIVÁLASZTASA
                MOVWF       TRISB           ;BEMENET:0. KIMENET:1-7.
                CLRF        TRISC           ;PORTC MINDEN BITJE KIMENET
                BANKSEL     PORTC           ;BANK0 KIVALASZTASA
                CLRF        PORTC

LENYOM                                      ;NYOMOGOMB KEZELES (GOMB LENYOM->PORTB.0 = 0)
                MOVF        PORTB,W         ;A VIZSGALAT MIATT PORTB->W-BE
                ANDLW       0X01            ;CSAK A 0. BITET KELL VIZSGALNI
                BTFSS       STATUS,Z        ;HA MEGNYOMTUK, FELENGEDRE LEP
                GOTO        LENYOM          ;VISSZA A LENYOM-RA (NINCS NYOMOGOMB VALTOZAS)

FELENGED
                MOVF        PORTB,W         ;A VIZSGALAT MIATT PORTB->W-BE
                ANDLW       0X01            ;CSAK A 0. BITET KELL VIZSGALNI
                BTFSC       STATUS,Z        ;HA FELENGEDTÜK KIKERÜLI AZ ELŐREUGRÁST
                                            ;(FELENGED-ET)
                GOTO        FELENGED        ;VISSZA UGRIK A FELENGED-RE
                CALL        WAIT_05         ;0,5S-OS SZOFTVERES KESLELTETES

                INCF        PORTC,1         ;PORTC NOVELESE
                GOTO        LENYOM          ;VISSZA A LENYOMRA

;-------< SZUBRUTINOK >--------

;0,5S-OS SZOFTVERES KESLELTETES - PIC-ES GYAKORLOBA
WAIT_05         MOVLW       0xFF            ;L = 255
                MOVWF       SZAML1          ;SZAML1 = L
ALAP            MOVLW       0xFF            ;M = 255
                MOVWF       SZAML2          ;SZAML2 = M
UJRA            NOP                         ;N = 16
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                DECFSZ      SZAML2,1        ;SZAML2 – 1 = 0?
                GOTO        UJRA            ;VISSZA
                DECFSZ      SZAML1,1        ;SZAML1 – 1 = 0?
                GOTO        ALAP            ;SZAML1 ALAPERTEK
                RETURN                      ;A KESLELTETEO SZUBRUTIN VEGE

                END                         ;PROGRAM VEGE


;NYGELV.ASM
;PROGRAMIRO: BORSI ISTVAN 2007.04.16.
;NYOMOGOMB KEZELES ELFIGYELESSEL

;-------< DIREKTIVAK >--------

LIST            P=16F871    ;PROCESSZOR DEFINICIO
INCLUDE         "P16F871.INC"               ;INCLUDE FAJL BEALLITAS
ERRORLEVEL      2

;-------< MAKRO DEFINICIO >--------
```

```
;-------< VALTOZOK DEKLARACIOJA >--------

CBLOCK            0X20
                            FEL
                            LE
                            UJ
                            REGI
                            ELEK
ENDC

;-------< PROGRAM KEZDETE >--------

                  ORG       0X003             ;EZ A KEZDOCIM
                  GOTO      MAIN              ;UGRAS A FOPROGRAMRA

MAIN
                  BANKSEL   TRISB             ;BANK1
                  MOVLW     B'00001111'       ;PORTB ALSO 4 BIT BEMENET
                  MOVWF     TRISB
                  CLRF      TRISC             ;PORTC MIND KIMENET
                  BANKSEL   PORTC                  ;BANK0

                  CLRF      PORTC             ;REGISZTEREK TORLESE
                  CLRF      UJ
                  CLRF      REGI
                  CLRF      ELEK
                  CLRF      LE
                  CLRF      FEL

LOOP              CALL      ELFIGY            ;ELFIGYELO RUTIN
                  BTFSS     FEL,0             ;NG0 LENYOMVA?
                  GOTO      NY1               ;NINCS, NG1 TESZTELESE
                  INCF      PORTC             ;PORTC NOVELESE
                  GOTO      LOOP              ;VISSZA

NY1               BTFSS     FEL,1             ;NG1 LENYOMVA?
                  GOTO      LOOP              ;NINCS, VISSZA
                  DECF      PORTC             ;PORTC CSOKKENTESE
                  GOTO      LOOP              ;VISSZA

;-------< SZUBRUTINOK >--------

;ELFIGYELO RUTIN
ELFIGY
                  BANKSEL   PORTB             ;BANKVALASZTAS
                  MOVF      PORTB,0           ;PORTB -> W
                  MOVWF     UJ                ;W -> UJ
                  XORWF     REGI,0            ;W -> W XOR REGI
                  MOVWF     ELEK              ;W -> ELEK
                  ANDWF     REGI,0            ;W -> W & REGI
                  MOVWF     FEL               ;W -> FEL
                  MOVF      UJ,0              ;UJ -> W
                  ANDWF     ELEK,0            ;W -> W & ELEK
                  MOVWF     LE                ;W -> LE
                  MOVF      UJ,0              ;UJ -> W
                  MOVWF     REGI              ;W -> REGI
                  RETURN

                  END                         ;PROGRAM VEGE
```

Név: ..... Osztály: Pontszám:...../**20p** Jegy: **1 2 3 4 5**

Mindenütt csak **EGY** jó válaszlehetőség van!

Javítás a tesztben **NEM LEHETSÉGES**!

1. A számítógép periféria feladata:
   a. Kapcsolattartás a külvilág felé;
   b. Adatmozgatás vezérlése;
   c. Számítási feladatok elvégzése;
   d. Csak az információk megjelenítése.

2. A kódmemória helye általában:
   a. RAM;
   b. SRAM;
   c. DRAM;
   d. ROM.

3. Nem lehet az adatmemória helye:
   a. EPROM;
   b. RAM;
   c. Flash memória;
   d. EEPROM.

4. Mi nem igaz a Neumann elvre:
   a. Tízes számrendszer;
   b. Elektromos működés;
   c. Közös kód és adatmemória;
   d. Vezérlőegység és ALU megléte.

5. Az ALU:
   a. Képes logikai műveletek elvégzésére;
   b. Tud adatokat mozgatni;
   c. Nem tud aritmetikai műveleteket elvégezni;
   d. PIC16F84-ben 14 bites számokkal dolgozik.

6. A PC:
   a. Program Controll;
   b. Program Cache;
   c. Adat számláló;
   d. A köv. végrehajtandó ut. címét tartalmazza.

7. Az IR:
   a. Az utasításregiszter;
   b. Értelmezi az utasításokat;
   c. Az ALU része;
   d. Integrated Relay.

8. A címdekódoló általában:
   a. Utasításokat értelmez;
   b. Utasításokat hajt végre;
   c. Megfelelő memóriarekeszt kiválaszt;
   d. Vezérli a PIC portjait.

9. Az átvitelbit:
   a. Az úgynevezett Borrow bit.
   b. Az úgynevezett Carry bit.
   c. Akkor 1 ha egy számítás 0 eredményt ad.
   d. Legkisebb helyi értéken jön létre.

10. A RISC processzor:
    a. Viszonylag sok utasítással rendelkezik;
    b. Viszonylag kevés regisztert tartalmaz;
    c. Összetett utasításkészletű;
    d. Assembly programozása bonyolult.

11. Melyik busz nem létezik:
    a. Cím;
    b. Flash;
    c. Adat;
    d. Vezérlő.

12. A RESET:
    a. Törli a kódmemóriát;
    b. Mindig valamilyen hiba eredménye;
    c. Alaphelyzetbe állítja a processzort;
    d. Nem törli az adatmemóriát.

13. Harvard arhitectúra:
    a. A cím és az adatbusz szélessége azonos;
    b. Jellemző a CISC processzorokra;
    c. Ugyanaz, mint a Neumann;
    d. Különálló pr. és adat memóriát tartalmaz.

14. A PIC-ekre jellemző:
    a. A legtöbb utasítás min. 2 szavas;
    b. Az órajel általában 0 és 20MHz közötti;
    c. Csak egycélú kivezetésekkel rendelkeznk;
    d. A TTL felépítés.

15. A PIC-ek gépi ciklusa:
    a. Négy órajelet igényel;
    b. Hét órajelet igényel;
    c. Hét ütemet igényel;
    d. Az adat elérési ideje.

16. Válassza ki a rendszervezérlő utasításokat:
    a. SLEEP, CLRWDT;
    b. BSF, BCF;
    c. GOTO, CALL;
    d. ADD, SUB.

17. Válassza ki a programvezérlő utasításokat:
    a. GOTO, CALL;
    b. AND, OR;
    c. BTFSC, BTFSS;
    d. RL, RR, SWAP.

18. A MOVWF utasítás:
    a. Adatot mozgat a W-ből egy fájl regiszterbe;
    b. Adatot mozgat egy fájlregiszterből a W-be;
    c. Két gépi ciklust igényel;
    d. Két fájlregiszter között mozgat adatot.

19. Az SFR:
    a. A kódmemória része;
    b. Az adatmemória része;
    c. Általános célú;
    d. EEPROM felépítésű.

20. A megszakítási vektor helye PIC-ek esetén:
    a. 0004h;
    b. 0040h;
    c. 03FFh;
    d. 0000h.

Név: ..... Osztály: Pontszám:...../**20p** Jegy: **1 2 3 4 5**

Mindenütt csak **EGY** jó válaszlehetőség van! Javítás a tesztben **NEM LEHETSÉGES**!

9. A 0x20 szám:
   a. A decimális 16;
   b. A bináris 11000011;
   c. Az okta 0123;
   d. A hexa 20.

10. A -4 decimális szám kettes komplemens értéke:
    a. 0001;
    b. 1100;
    c. 0100;
    d. 0101.

11. Tároló cellái kondenzátorokból állnak:
    a. EPROM;
    b. SRAM;
    c. ROM;
    d. DRAM.

12. A RISC processzor:
    a. Komplex utasításkészletű;
    b. Mikroprogramtárat használ;
    c. Egyszerű utasításkészletű;
    d. Nem tartalmaz ALU-t.

13. Az ACC:
    a. Aritmetic Compare Circuit;
    b. Tud adatokat mozgatni;
    c. Az ALU része;
    d. Csak címeket tartalmaz.

14. A MEMR\MEMW jel:
    a. Az adatbusz jele;
    b. Egy megszakításjel;
    c. A vezérlőbusz jele;
    d. A perifériával kommunikál.

15. A PIC mikrovezérlők:
    a. CISC felépítésűek;
    b. Gyártója az Intel^{R};
    c. Neumann felépítésűek;
    d. RAM mérete maximum egy-két Kilobájt.

16. A PIC16F84 1 program szavának mérete:
    a. 8 bites;
    b. 16 bites;
    c. 14 bites;
    d. 12 bites.

9. A Watchdog Timer:
   a. Programhiba ellen véd;
   b. Minimális tápáram felvételt biztosít;
   c. Univerzális időzítő;
   d. Illetéktelen program kiolvasás ellen véd.

10. Melyik nem a gépi ciklus üteme:
    a. Utasítás dekódolása;
    b. Jelzőbitek állítása;
    c. Utasítás végrehajtása;
    d. Eredmény visszaírása.

11. Az átlapolt utasítás végrehajtás:
    a. Több utasítást hajt végre egy időben;
    b. A processzort lassítja;
    c. Mindig jól működik;
    d. A pipe-line elv.

12. A Status Regiszter tartalma:
    a. Vezérlő és konfigurációs bitek;
    b. Megszakításkezelő bitek;
    c. Indirekt címzést végez;
    d. Az ALU jelzőbitjei.

13. A PIC16F871-re nem igaz:
    a. Az 1kszó méretű programmemória;
    b. Kisáramú kimenetek;
    c. 35db egyszerű utasítás;
    d. Program EEPROM 1000 írási/törlési ciklusú.

14. A BSF AREG,2 utasítás:
    a. Törli az AREG 2. bitjét;
    b. Egybe állítja az AREG 3. bitjét;
    c. Átlépi a következő utasítást, ha AREG.2 = 1;
    d. Az AREG 2. bitje egy lesz.

15. A SWAP BREG utasítás:
    a. Felcseréli a BREG alsó felső 4 bitjét;
    b. A BREG értéke csökken 1-el;
    c. A BREG törlődik;
    d. A BREG inkrementálódik.

16. Az ADDLW 0x37 utasítás:
    a. A W értéke 37H lesz;
    b. A W értéke 37H-val nő;
    c. A W értéke csökken 37H-val;
    d. A W-hez hozzáadja a 37H című regisztert.

17. Az emulátor:
    a. Egy számítógépes program;
    b. Egy programozó;
    c. Egy hardver eszköz;
    d. Szimulációs eszköz, programozáshoz.

18. Az AREG EQU 0x20 direktíva után:
    a. Az AREG értéke 20H;
    b. Az AREG címe 20D;
    c. Az AREG egy 8 bites regiszter;
    d. Az AREG egy 16 bites regiszter.

19. A makró:
    a. Előre definiált utasítás sorozat;
    b. Olyan, mint a függvény;
    c. Helytakarékos megoldás a programozásban;
    d. CALL utasítással kell meghívni.

20. Az MPASM rendszerben:
    a. A forrásfájl PRJ kiterjesztésű;
    b. Nem lehet szimulálni;
    c. A programozó fájl kiterjesztése HEX;
    d. Nincs MAKE project módszer.

Köszönetnyilvánítás

Köszönetemet szeretném kifejezni a volt Kandós főiskolai tanáraimnak, akik a mikrokontrolleres technikával megismertettek, a Kossuth Zsuzsanna Műszaki Szakközépiskolának és Gimnáziumnak, amiért az informatika tanári képzésemet és a szakdolgozatom elkészítését támogatta. Külön köszönet illeti a szakdolgozat egyetemi konzulensét, Szabó Zsoltot.