

***A 8031/51 mikrokontroller bázisú***

***mikrogépek programfejlesztése***

**Összeállították: Zalotay Péter**  
főiskolai docens  
**Lamár Krisztián**  
főiskolai adjunktus

## 1. A programozás célja

A mikrogép működését meghatározó programot az ún. **code-**, vagy **programmemóriába** kell tárolni az alkalmazott mikroprocesszor, vagy mikrokontroller - a továbbiakban **processzor** - típusától függő gépi kódok sorozatával. A program végrehajtásakor a processzor innen olvassa ki az utasításokat, és paramétereiket.

A programozás célja tehát.

- egy olyan **kódsorozat** létrehozása, amelyet
- egy **mikrogép** ( mikroprocesszor bázisú számítógép ) **programmemóriájába** töltve
- meghatározza a mikrogép **feladat** szerinti **működését**.

## 2. A programfejlesztés lépései

A programfejlesztés az alábbiakban tömören ismertetett lépésekből áll.

- a. A program **felépítésének** megtervezése. Ezt segíti a **folyamatábra** megrajzolása.
- b. A szükséges **memória-szegmensek** deklarálása
- c. A programban használt **változók, konstansok, periféria-címek** meghatározása.
- d. szöveges formában - valamilyen szövegszerkesztő (editor) segítségével - kell megírni a programozási-nyelv " helyesírási " (szintaktikai), és tartalmi (szemantikai) szabályai szerint. Az így megírt szöveg a **program forrásnyelvi** formája. A számítógépek, mikrogépek alkalmazásának széleskörű elterjedése a különböző **programnyelv** választékot is bővítette. A programnyelveket **gépközei-** (assembly), és a **magas szintű** nyelvek csoportjába sorolhatjuk.
  - **Gépközei** az a programozás, amelynél minden **programlépés** – programsor - az alkalmazott processzor egy-egy **utasítása**. Mivel minden processzornak saját utasításkészlete van, ezért a programírás is processzorfüggő. Az ilyen programírást nevezzük **assembly** vagy más közelítésben **utasítás – szerkezetű** programírásnak.
  - A **magas szintű** programozási nyelvek ( Pascal, C, stb ) rendszerint a nemzetközileg elfogadott **matematikai, logikai** műveletek, és különböző **általános**, a programszerkezetet meghatározó **utasítások** segítségével írják le a feladatot. Ezért e nyelveken a programozás független attól a géptől, amelyen a programot futatni akarjuk. A magas szintű nyelven történő programírást **művelet - központú** programozásnak is nevezhetjük.
- e. A forrásnyelvi programból a különböző **fordító** programok (assembler, compiler) állítják elő a **program tárgykódú** (object) alakját, illetve a programlistát, amelyek már a processzor utasításkódjait is tartalmazzák. (A leírtakból következik, hogy a magas szintű nyelveknél is a fordítóprogram már processzortól függő.)
- f. A tárgykódú programból, vagy programokból (több modul esetében) a **szerkesztő** (linker) program állítja elő a **futtatható** programváltozatot.
- g. Az így előállított programot kell beírni (letölteni) a programmemóriába. A programmemória lehet fix (ROM, EPROM), illetve írható-olvasható (RAM) kialakítású. A beégetés, vagy letöltés történhet teljes kódú (bináris), vagy tömörített (pl. HEX) alakú fájlból. Ennek megfelelően a programozás utolsó lépése a kívánt formátum-konverzió elvégzése.

### 3. A programfejlesztés eszközei

A programfejlesztés **leglényegesebb** eszköze az **ember**, mivel a feladatot megvalósító program **algoritmusait, felépítését, változókat, konstansokat** meghatározását stb. csak alapos **tervező, elemző** munkával lehet, és kell elvégezni

A programfejlesztés **tárgyi** eszközei ma kizárólag általános célú számítógépek **PC-k**, és az ezeken futó **fejlesztő-programok** segítségével történik. Természetesen e programok csak a technikai háttérrel adják. A teljesség igénye nélkül az alábbi szoftverek elengedhetetlenül szükségesek:

- a **rajzoló** programok (folyamatábra rajzolás),
- a **szövegszerkesztők**, editorok (forrásnyelvű állomány írásához),
- az **assembler**ek, **compillerek** ( a szöveges állomány fordításához),
- a **linkerek** ( az abszolút tárgykódú állomány szerkesztéséhez),
- az **obj-hex, obj-bin** átalakítók (formátum-konverzióhoz),
- a **szimulátorok** (a program-szimulációhoz),
- **emulátorok, terminál-monitor** programok (a valós idejű teszteléshez)

A programfejlesztés módszerei – a sokszor az adottságoktól függően – különbözőek. A PC-k elterjedésének kezdetén a fejlesztési lépések **egyedi** végrehajtása, esetleg ún. **parancs-fájlok** (batch) segítségével segítették a hatékonyságot. A számítógépek, a szoftvergyártás rohamos fejlődése hozta az ún. **integrált fejlesztői környezetek** kialakítását. A nagy szoftvergyártó cégek különféle környezeteket (**DOS**, majd **Windows** operációs rendszerekben) fejlesztettek. Ezek első-sorban az elterjedt magas szintű nyelvekhez, úgymint a **Pascal, C++** stb. készültek. A különböző multitask-os operációs rendszerekben egymás után jelentek meg az „automatizált” programfejlesztő programcsomagok pl. **DELFI, BILDER** stb.

A **8031/51** mikrokontroller család programfejlesztéséhez a **Keil Elektronik** cég másfél évtizede készít fejlesztő programokat ( a51.exe, c51.exe, l51.exe, lib51.exe, ohs51.exe), illetve integrál fejlesztő környezetet (**UV2**). A főiskolai oktatásban is ezeket a fejlesztői szoftvereket alkalmazzuk. Az oktatási feltételeket, és igényeket kielégítő fejlesztő környezet-családot (az **XE2** különböző változatait oktatónk irányításával a hallgatók szakdolgozataikban dolgoztak ki. A továbbiakban a legújabb változatú – **XE251** elnevezésű – fejlesztői környezet működését, és használatát írjuk le.

### 4. Programszerkezetek

A mikroprocesszoros berendezések programjainak legkisebb egységei az **utasítás-kódok**, és paramétereik. Nagyobb egységek azok a **program-blokkok**, utasítás-csoportok, amelyek egy adott jellegű művelet végrehajtásához tartoznak. A következőekben összefoglaljuk az alapvető programblokkokat, és szemléltetjük folyamatábrával, illetve assembly nyelvű példával.

• **Műveleti blokk,**

; az OP1 és OP2 összeadása

```
MOV    A,OP1
ADDC   A,OP2
MOV    ERED,A
      .
      .
```

egy logikailag összetartozó (matematikai, logikai, stb.) utasítás -csoport.

• **Feltétel nélküli programugrás**

; ugrás a folytatásra

```
MOV    ERED,A
LJMP   FOLYT
      .
      .
FOLYT: MOV    DPTR,#NGS
      ..
```

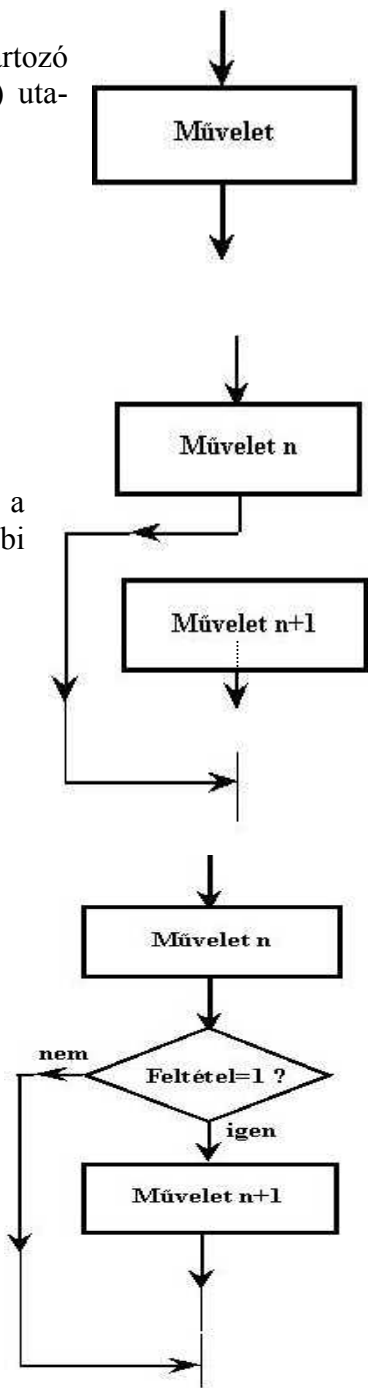
az utasítás beolvasása a programmemória távolabbi címéről történik

• **Feltételes programugrás**

; ha az A=0 ugrás a folytatásra

```
ADDC   A,OP2
JZ     FOLYT
MOV    ERED,A
      .
      .
FOLYT: MOV    DPTR,#NGS
```

egy változó (bájtos, bites) aktuális értékétől függ, hogy az utasítás beolvasása a programmemória következő címéről, vagy távolabbi címéről történik.



A matematikai, logikai összefüggésekkel leírható feladatok **elemi programszerkezetek** segítségével leírhatók. Ezek a már megismert **program-blokkokból** építhetők. Jellegetes elemi program-szerkezetek:

• **Lineáris** (lefutó) program

elemi programblokkok sorozat, amely a program lefutása, (az elemi blokkok végrehatása) után nem tér vissza a program elejére. A program futási ideje alatt csak egyszer hajtódik végre. Pl. inicializálás stb.

• **Ciklikus** program

ismétlődő - elemi blokkokból álló – programcsoport (ciklustörzs). A program futási ideje alatt végrehajtása ismétlődik. Pl. a főprogram (a main) stb.

- **Feltételes program végrehajtás** egy művelet, érték-összehasonlítás, stb. eredményétől függ a programcsoport végrehajtása.

Miután minden szoftver ciklikus szervezésű, ezért ezt részletezzük. Minden programciklus alapvetően a következő elemeket tartalmazza:

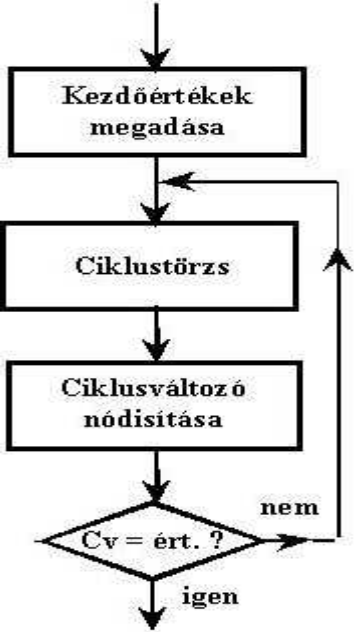
- **ciklustörzs** az a programcsoport, amely ismétlődik,
- **ciklus változó** az a változó, amelynek értékének tartománya meghatározza a ciklustörzs ismétlését,
- **értékelés** a ciklusváltozó értékének összehasonlítása a ciklusban maradás feltételeivel,
- **döntés** az értékelés eredményétől függő programelágazás.

A feladattól függően több változatban szervezhetők programciklusok. A szerkezet függ a ciklusváltozótól, az értékelés cikluson belüli helyétől. Jellegzetes ciklus típusok:

- **tól l-ig .. (for to) ciklus**

```
; 16 bájt törlése
MOV R7,#16
MOV R0,#BEM
ISM: MOV @R0,#0
      INC R0
      DJNZ R7,ISM
      .
```

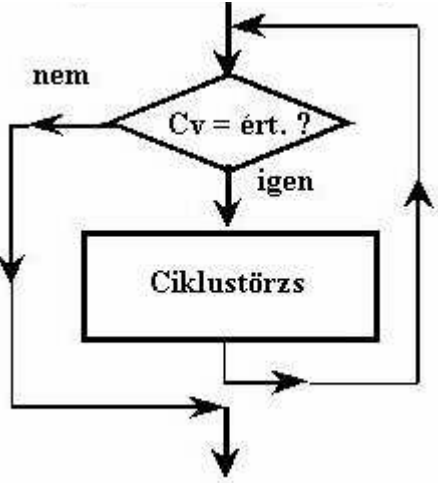
a ciklusba lépés előtt ismert az ismétlések száma,



- **amíg .. (while) ciklus**

```
ISM: CJNE @R1,#'b',TOV
      .
      .
      .
      AJMP ISM
```

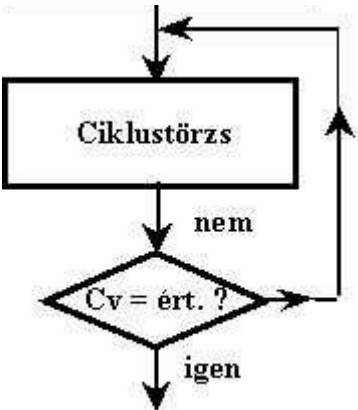
amíg a ciklusváltozó adott értéktartományba esik ismételtén hajtsd végre a ciklustörzset ( a ciklusváltozó a törzsben módosul). A ciklus **elől-tesztelt**.



- *majd-amíg...*( do-while)

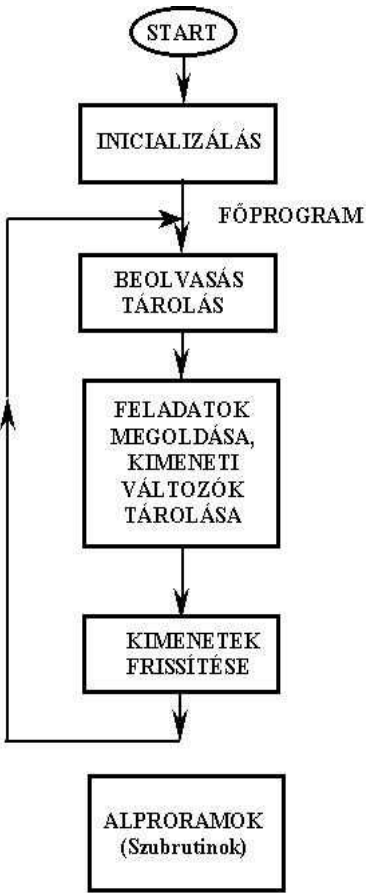
```
ISM:      ; ciklustörzs
          .
          .
          .
          CJNE  A,BEM,ISM
          .
```

először hajtsd végre a ciklustörzset , majd amíg a ciklusváltozó adott értéktartományba esik ismételd ( a ciklusváltozó a törzsben módosul). A ciklus *hátraltesztelt*.



A technikai feladatokat megvalósító programok struktúrája az ismertetett elemi program-szerkezetek használatával kialakítható. Egy alkalmazói program a következő alapvető egységekből áll:

- inicializálás a változók kezdőértékének, programozható perifériák üzemmódjának, megszakítási rendszer szükség szerinti beállítása, stb., lefutó típusú,
- főprogram (main) a feladatot ismétlődően végrehajtó program mindig (ciklikus), amely három fő részre bontható:
  - a bemeneti értékek frissítése, tárolása
  - a feladat szerinti műveletek megoldása az aktuális bemeneti értékekkel, és az új kimeneti változók tárolása.
  - kimenetek frissítése a ciklusban meghatározott kimeneti változókkal.
- alprogramok (szubrutinok) a részfeladatokat megoldó programrészek lefutó típusúak.



A leírt általános felépítést szemlélteti a folyamatábra.

## 5. Programfejlesztés az XE251 környezetben.

A fejlesztői környezet menürendszerű kialakítású, és támogatja a programfejlesztés technikai részét a forrásnyelvű állomány írásától a valósidejű programellenőrzésig. A fejlesztői környezet adja a keretet, és az XE251.INI elnevezésű inicializáló fájlba kell megadni a fejlesztéshez használt programokat. Az XE251-környezetben mind assembly, mind pedig C forrásnyelvű programok fejleszthetők.

A program indítása az XE251.BAT parancsfájl aktiválásával történik, amely után az 1. ábra szerinti képet kapjuk



1. ábra

Az egyes menük megnyitása történhet:

- a *cursor - vezérlő gombok* mozgásával + **Enter** leütésével,
- az **Alt** + a menü *nagybetűjének* együttes leütésével,
- az *egér* segítségével.

.Bármely menüből az **Esc** segítségével lehet kilépni.

A továbbiakban tekintsük végig az egyes menük és almenük funkcióit



5.1. Fájl (Fájlkezelés menü)

A menü a különböző *fájlok* (állományok) *kiválasztását, megtekintését* szolgálja. A menü megnyitásával megnyíló ablakban - 2. ábra – látható almenük közül lehet választani.



2.ábra.

mely programvázként szolgál. Külön almenüből választhatunk, hogy Assembly vagy C programhoz illetve Projekt-fájlhoz előkészített sablont kívánunk használni. A \*.\*-ot választva üres fájlt hozhatunk létre.

A *Takarítás* almenü aktiválásával törölhetjük a segéd-állományokat a használt munka-könyvtárból. ( a forrásállomány és a projekt fájl marad meg).

A *Kilépés /Quit* almenü aktiválásával léphetünk ki.

A 3.ábra mutatja a *Betölt* menüpont megnyitása utáni képet. Itt választhatjuk ki a fejlesztésben alkalmazott



3.ábra

A 4. ábrán láthatók a *Megtekint* megnyitása utáni választások

A *Betölt* almenüben választható ki az állomány – a forrásnyelvű fájl -, amellyel dolgozni ( írni, módosítani ) fogunk.

A *Megtekint* almenüben választhatjuk ki a fejlesztés során keletkező állományokat.

A *Könyvtár létrehozása* menüpontot megnyitva könyvtárat hozhatunk létre a C:\XE251\MUNKA\ alá.

A *Fájl létrehozás* megnyitásával egy új munkafájlt hozhatunk létre az előzőben megnyitott könyvtárban. A létrehozandó fájl *előre elkészített sablon* alapján készül, így egy kezdeti általános (és hasznos) tartalommal rendelkezik,

- |       |                                   |
|-------|-----------------------------------|
| *.A51 | assembly nyelvű, vagy             |
| *.C   | C nyelvű forrásállományt,         |
| *.PRJ | projekt fájlt,                    |
| *.*   | egyéb kiterjesztésű állományokat. |

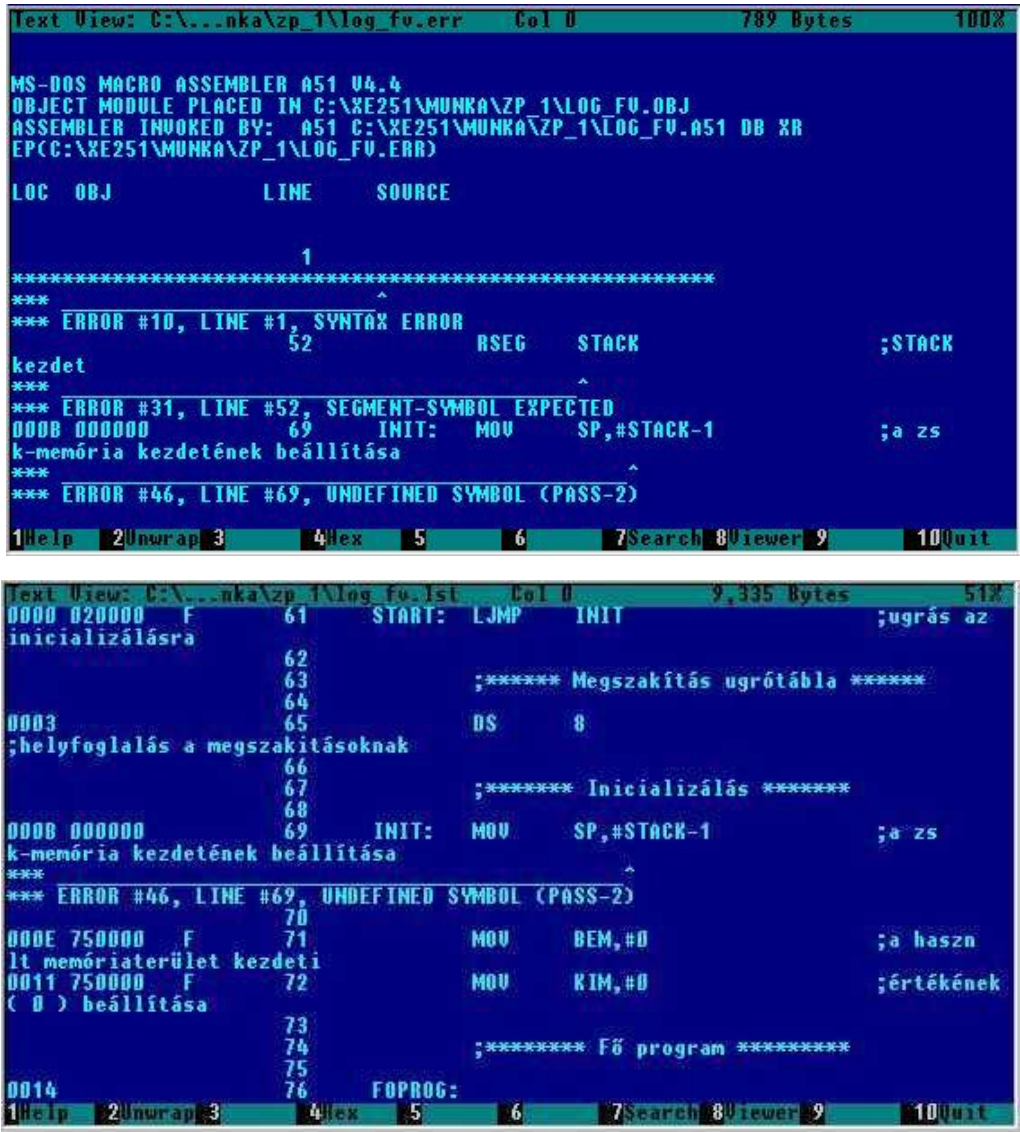




4. ábra

- A **Hiba-fájl** a fordításnál keletkezik, és a forrás-állomány (szintaktikai) hibáit, és leírását tartalmazza
- A **Lista-fájl** ugyancsak a fordításnál keletkezik, és mutatja a forrásállomány sorait és a lefordított kódokat. A hibás sorok – nál jelzi azokat, és jellegüket
- A **Map-fájl** a szerkesztés (linkelés) eredménye, és a teljes memóriatérképet, szimbólumokat mutatja.
- A **Kevert C lista** csak a C nyelvű programoknál, a linkelés eredményeként jön létre. Mutatja a C sorokból lefordított assembly listát.

Az 5. ábra Hiba-, és Lista -fájl részleteket szemléltet.



5. ábra

### 5.2. Editor (ASCII szöveges fájlok megtekintése, módosítása)

A menüpontban írható, módosítható a kiválasztott (Fájl / betölt..) állomány. A menü megnyitása után a 6.ábrán látható kép jelenik meg.



6.ábra

A **Forrás** almenüben a forrásnyelvi fájl, míg

A **Project** almenüben a munkakönyvtárban lévő project fájl írható, módosítható. A project fájl nevére nincs megkötés (célszerűen a Fájl menülétrehozás almenüjével kell létrehozni), azonban egy munkakönyvtárban, kizárólag egyetlen project-fájl lehet. A project fájlal kapcsolatos részleteket ld. a Súgó menü Project filozófia című almenüje alatt.

A **MEGSZ.A51**, **STARTUP.A51** és **STARTUP0.A51** menüpontokban a C nyelvű programoknál szükséges segédfájlok megnyitása szerkesztésre végezhető. Ezek mindegyikének az adott munkakönyvtárban kell lenniük. Ha a kiválasztott fájl nem létezik, akkor azt az XE251 a szerkesztés megkezdése előtt létrehozza egy előre elkészített **sablon** szerint, melyben a szükséges módosítások gyorsan elvégezhetők. A módosított fájlt automatikusan újrafordítja a rendszer a szerkesztőből való kilépéskor.

### 5.3. Assembler / Compiler (Fordítás menü)

A menümegnyitása után (7.ábra) választható ki



7.ábra

a forrásnyelvű állomány **fordítása**, amely létrehozza az elsődleges tárgykódú, \*.OBJ, a lista \*.LST, és a hiba \*.ERR állományokat.

A **Linker** almenüben történik a tárgykódú fájl(ok) **szerkesztése** abszolút object állománnyá.

Az abszolút object állomány letölthető **Intel-Hex** formátumú fájl generálása. Létrehozza a \*.HEX állományt.

A Linker almenü megnyitása újabb választási lehetőségeket kínál, amelyet a 8. ábra mutat.



**Egyedi (4000H-ra)**: Egyedi (egymodulos) forrásfájl **linkelése** és automatikus **relokálása** 4000H-ra

**Egyedi (0000H-ra)**: Egyedi (egymodulos) forrásfájl **linkelése** és automatikus **relokálása** 0000H-ra

**Fontos**: A belső IDATA RAM mérete 256 bájt-ra van megadva, ezért 8051-es mikrokontroller esetén vigyázni kell, mert annak csak 128 bájtos IDATA memóriája van. A fordító nem fog hibaüzenet adni, ha ezt túllépjük!



**Egyedi + Startup (4000H-ra):** Egyedi (egymodulos) forrásfájl *linkelése* és automatikus *relokálása* 4000H-ra. Hozzálinkeli még a munkakönyvtárban lévő, a C nyelvű programoknál szükséges *STARTUP.A51* és *MEGSZ.A51* segédfájlokat.

**Egyedi + Startup (0000H-ra):** Egyedi (egymodulos) forrásfájl *linkelése* és automatikus *relokálása* 0000H-ra. Hozzálinkeli még a munkakönyvtárban lévő, a C nyelvű programoknál szükséges *STARTUP0.A51* és *MEGSZ.A51* segédfájlokat.

**Project:** Többmodulos forrásfájl *linkelése* **PROJECT** fájl alapján

5.4. Tesztelés (programellenőrzés)

A menümegnyitása után (9. ábra) választható ki



9. ábra

A **Szimulátor** almenü megnyitásakor a Simula5x szoftver szimulátor segítségével ellenőrizhető a fejlesztett program.

A **Terminál** almenü megnyitásakor a mikrogép és a PC között – választható soros vonalon – kommunikációs kapcsolat jön létre. A letöltött program lépésenkénti, és valós idejű futtatása segítségével ellenőrizhetjük a programot.



10. ábra



11. ábra

A **Disassembler** almenübe újabb ablak nyílik meg, és felkínálja a forrás állomány visszafordításának (disassemblál) két változatát. Az első az assembly forrást adja, míg a második a C nyelven írt program kevert visszafordítását szolgáltatja (11. ábra).

5.5. A súgó

A 12. ábrán látható menüben ismertető leírásokat hívhatunk elő a képernyőre. A programfejlesztés közben segítséget kaphatunk a munka elvégzéséhez



12. ábra

Az MCS-51 megnyitása után a 13. ábra szerinti kép mutatja, hogy az *utasításkészlet* is szerepel a menükben.



13. ábra