Die Bedienung des Assemblers ASM51

Wolfgang SCHARL, TGM, NT/EL

Für viele, die sich im Zuge des Unterrichts oder privat im Rahmen der Clubaktion mit dem 8051 Controller auseinandersetzen, stellt sich natürlich das Problem, selbstgeschriebene Programme zu assemblieren und zu linken. In diesem Zusammenhang haben sich einige immer wiederkehrende Fragen herauskristallisiert, die ich an Hand eines kleinen Beispiels - speziell für Anfänger - darstellen möchte.

Grundsätzlich seien Syntax und Befehlsvorrat des ASM51 hier einmal vorausgesetzt (Mnemonics 8051: TGM-LIT-003, Assemblerdirektiven: in diesem Heft). Geschrieben werden Assemblerprogramme mit irgend einem Editor, der unformatierte ASCII Texte erstellen kann. Von Sidekick bis zum Textverarbeitungssystem ist so ziemlich alles, mehr oder weniger komfortabel, verwendbar. Bei Textsystemen wie Word muß darauf geachtet werden, daß die Textdatei "unformatiert" abgespeichert wird.

Eine derart erstellte Textdatei hat am Ende grundsätzlich ein ^Z als "end of file"-Markierung. Die irritiert den Assembler, und er reagiert mit einer Fehlermeldung. Das ist an sich belanglos, wenn Sie das stört, können Sie mit einem kleinen Hilfsprogramm (EOF-Killer) das ^Z aus der Datei herauslöschen. [Beim Editieren mit WORD wird kein ^Z an die Datei angefügt, es kann aber bei Bedarf mit ALT+NUM2,NUM6 generiert werden.]

Als Beispiel möchte ich das Demoprogramm TEST1.ASM, das auf der Diskette "µPROFI-Grundpaket" enthalten ist, verwenden. Dieses Programm ist auf der Diskette auf eine falsche Adresse gelinkt und läuft daher auch nicht.

TEST1.ASM

```
NAME    Test1
C_Test1 SEGMENT CODE
        RSEG C_Test1
Start:  mov     a, #01h         ;ACC lowest Bit setzen
        clrc    c               ;Carry löschen
Loop:   rlc     a               ;Accu Bit nach links rotieren
        jnc     Loop            ;solange nach Loop bis Carry 1
Ende:   clr     c               ;Carry löschen
END
```

Für den µPROFI ist es erforderlich, die Programme im Downloadbereich auf der Adresse a000h beginnen zu lassen. Dies kann mit dem Befehl ORG 0a000h am Beginn des Programmcodes erfolgen. Diese Methode hat aber den Nachteil, daß ein Verschieben des Progammes im Adreßbereich, sowie das Zusammenlinken mehrerer Programmodule nicht möglich ist. Man sollte daher stets einen Segmentcode definieren und die Startadresse dann beim Linken festlegen.

Der erste Schritt ist jetzt das Assemblieren des Programmes mit folgender Eingabe:

```
asm51 test1.asm db ep(test1.err)
```

Die einzelnen Teile dieser Kommandozeile haben folgende Bedeutung:

`asm51` ruft den Assembler auf, der natürlich im aktuellen Verzeichnis oder im Suchpfad zu finden sein muß.

`test1.asm` ist der Name des Sourcecodes unseres Programms.

`db` weist den Assembler an, ein Symbolfile zu erstellen. Mit diesem Symbolfile können Sie später im FSD51 die Labels Ihres Programmes einblenden (so Sie welche verwendet haben).

`ep(test1.err)` weist den Assembler an, ein Errorfile mit dem Namen TEST1.ERR zu erstellen. In diesem steht nur der Sourcecode mit ev. Fehlern. Alles andere wird unterdrückt. Sehr hilfreich zur Fehlersuche, besonders bei längeren Programmen.

Nun sollten zusätzlich zur Datei TEST1.ASM folgende Dateien entstanden sein:

TEST1.OBJ

TEST1.OBJ ist als Textdatei nicht darstellbar.

TEST1.ERR

```
DOS 3.30 (033-N) MCS-51 MACRO ASSEMBLER, V2.2
OBJECT MODULE PLACED IN TEST1.OBJ
ASSEMBLER INVOKED BY:  ASM51.EXE TEST1.ASM DB EP(TEST1.ERR)

LOC  OBJ            LINE     SOURCE
REGISTER BANK(S) USED: 0
ASSEMBLY COMPLETE, NO ERRORS FOUND
```

TEST1.LST

```
MCS-51 MACRO ASSEMBLER    TEST1                                                 03/21/91   PAGE    1
DOS 3.30 (033-N) MCS-51 MACRO ASSEMBLER, V2.2
OBJECT MODULE PLACED IN TEST1.OBJ
ASSEMBLER INVOKED BY:  ASM51.EXE TEST1.ASM DB EP(TEST1.ERR)

LOC  OBJ            LINE     SOURCE
                       1     NAME    Test1
                       2
                       3     C_Test1 SEGMENT CODE
----                   4             RSEG C_Test1
                       5
                       6
0000 7401              7     Start:  mov     a, #01h         ;ACC lowest Bit setzen
0002 C3                8             clr     c               ;Carry l^schen
0003 33                9     Loop:   rlc     a               ;Accu Bit nach links rotieren
0004 50FD             10             jnc     Loop            ;solange nach Loop bis Carry 1
0006 C3               11     Ende:   clr     c               ;Carry l^schen
                      12
                      13     END
```

```
MCS-51 MACRO ASSEMBLER    TEST1                                                  03/21/91   PAGE    2
SYMBOL TABLE LISTING
------ ----- -------
N A M E     T Y P E   V A L U E      A T T R I B U T E S
C_TEST1. .  C SEG     0007H          REL=UNIT
ENDE . . .  C ADDR    0006H   R      SEG=C_TEST1
LOOP . . .  C ADDR    0003H   R      SEG=C_TEST1
START. . .  C ADDR    0000H   R      SEG=C_TEST1
TEST1. . .    ----    ----
REGISTER BANK(S) USED: 0
ASSEMBLY COMPLETE, NO ERRORS FOUND
```

Als Nächstes wird auf die richtige Adresse gelinkt. Die Startadresse für den µPROFI ist a000h . Es ist dabei zu beachten, daß Hexzahlen immer mit einer Ziffer beginnen müssen, sonst werden sie von Assembler und Linker nicht erkannt (daher 0a000h statt a000h). Die Kommandozeile für den Linker sieht daher so aus:

```
rl51 test1.obj to test1.bin code(c_test1(0a000h))
```

Nun sind folgende Dateien entstanden:

TEST1.BIN

TEST1.BIN ist wieder als Textdatei nicht darstellbar. Weiters entsteht die Symboldatei:

TEST1.M51

```
RL51                                                    PAGE    1
DATE : 03/21/91
DOS 3.30 (033-N)
MCS-51 RELOCATOR AND LINKER V3.0, INVOKED BY:
RL51.EXE TEST1.OBJ TO TEST1.BIN CODE(C_TEST1(0A000H))
INPUT MODULES INCLUDED
  TEST1.OBJ(TEST1)

LINK MAP FOR TEST1.BIN(TEST1)
TYPE    BASE      LENGTH    RELOCATION   SEGMENT NAME
----    ----      ------    ----------   ------------
REG     0000H     0008H                  "REG BANK 0"
        0000H     A000H                  *** GAP ***
CODE    A000H     0007H     UNIT         C_TEST1

SYMBOL TABLE FOR TEST1.BIN(TEST1)
VALUE           TYPE            NAME
-----           ----            ----
-------         MODULE          TEST1
C:A000H         SEGMENT         C_TEST1
C:A006H         SYMBOL          ENDE
C:A003H         SYMBOL          LOOP
C:A000H         SYMBOL          START
-------         ENDMOD          TEST1
```

Die Binärdatei TEST1.BIN könnte nun so wie sie ist in ein Eprom geschrieben werden und müßte funktionieren. Die meisten Programmiergeräte und auch unser FSD51 erwarten diese Datei aber in einem besonderen, fehlersicheren Format genannt "Intel-Hex". Dieses Format wird mit dem Programm OH erzeugt:

```
oh test1.bin
```

es entsteht nun ein

TEST1.HEX

```
:07A000007401C33350FDC3DE
:00000001FF
```

Diese Datei kann nun mit dem FSD51 in den µPROFI geladen und mit Singlestep oder Run ausgeführt werden. Wird auch das Symbolfile TEST1.M51 in den FSD51 geladen, so erscheinen zur besseren Übersicht auch die Labels am Bildschirm.

Die weitere Vorgangsweise ist im Benutzerhandbuch des µPROFI51 ab Seite 25 ausführlich dargestellt.

Zusammenfassung

zum Assemblieren und Linken des Programmes TEST1.ASM sind folgende Eingaben erforderlich:

```
asm51 test1.asm db ep(test1.err)

rl51 test1.obj to test1.bin code(c_test1(0a000h))

oh test1.bin
```

Das erste Gesetz des wissenschaftlichen Fortschritts

Variante 1: Der Fortschritt einer Wissenschaft ist daran zu messen, in welchem Tempo sich Ausnahmen zu früheren Regeln ansammeln.
Variante 2: Ausnahmen sind grundsätzlich zahlreicher als Regeln.
Variante 3: Von allen anerkannten Ausnahmen gibt es Ausnahmen.
Variante 4: Wenn man die Ausnahmen endlich im Griff hat, erinnert sich niemand mehr der Regeln, für die sie angeblich gelten.