Disassembler...

. . . für den Junior-Computer

Die Entwicklung von Software für Computer ist ein Abenteuer, die Analyse häufig eine Entdeckungsreise. Ein Kompaß ist zur Orientierung unentbehrlich: das Disassembler-Programm. Damit werden einzelne Bytes zu Befehlen, einschließlich genauer Operanden-Informationen, verarbeitet. Das Programm steckt in einem EPROM, das man auch vom ESS programmieren lassen kann.

Die Redaktion erhält zuweilen Programme für den Junior-Computer zugeschickt. Eine gute Portion Skepsis (Redakteurspflicht) veranlaßt uns, diese Programme auszuprobieren. Also tippen wir die Daten ein. Das geht sehr schnell mittels PME. Anschließend "entschlüsseln" wir die Daten mit Hilfe des Disassemblers. Das Ergebnis ist nicht immer das einer Entdeckungsreise, hat allerdings manchmal auch etwas mit der Tätigkeit eines Sherlock Holmes zu tun.
Ein weiteres Beispiel. Im vorigen Heft wurde die Wandlung des KIM-Basic zum Junior-Basic beschrieben. Ohne Disassembler hätte diese Prozedur kaum stattfinden können.
Zum dritten. Ein Disassembler leistet gute Dienste bei der Dokumentation selbst"gestrickter" Software, eventuell auch als Vorstufe einer umfangreichen Editor/Assembler-Dokumentation.

Tabelle 1.

```
FC4E
FC4E A9 R
VALID COMMANDS: A D H L P R SP

D
DISASSEMBLE: 200,22F
L,P,SP ?
L
0200 A9 00        LDA #$00
0202 AD 01 02     LDA $0201
0205 A5 03        LDA $03
0207 A1 04        LDA ($04,X)
0209 B1 05        LDA ($05),Y
020B B5 06        LDA $06,X
020D BD 07 08     LDA $0807,X
0210 B9 09 0A     LDA $0A09,Y
0213 B6 0B        LDX $0B,Y
0215 20 0C 0D     JSR $0D0C
0218 4C 0E 0F     JMP $0F0E
021B 6C 10 11     JMP ($1110)
021E 77           @@@
021F FF           @@@
0220 00           BRK
0221 00           BRK
0222 CA           DEX
0223 C8           INY
0224 E8           INX
0225 0A           ASL  A
0226 F0 12        BEQ $023A
0228 D0 FE        BNE $0228
022A B0 34        BCS $0260
022C 90 EE        BCC $021C
022E FA           @@@
022F 00           BRK
```

Einzelheiten

Das Disassembler-Programm steckt in einem 2716-EPROM, das man beispielsweise als ESS511 programmieren lassen kann. Dabei geht's um die Adressen F800 . . . FFFF. Das EPROM kann auf eine RAM/EPROM-Karte oder auch auf die in diesem Heft beschriebene Mini-EPROM-Karte gesetzt werden. Die Speicherplätze F800 . . . FDD9 enthalten die Disassembler-Software; die Plätze FDDA . . . FFF9 sind mit "EPROM PROGRAMMING UTILITIES" gefüllt (die Beschreibung erfolgt im nächsten Heft); und in FFFA . . . FFFF stecken die bekannten Vektor-Daten.
Was bewirkt solch ein Disassembler eigentlich? Zunächst zu Tabelle 1. Dazu genügt eine kurze und bündige Beschreibung. Nach dem Start (Adresse FC4E über PM eingeben!) folgt eine Rückmeldung mit einer Auflistung der Funktions-Tasten. Mit Taste D gibt man zwei Adressen ein, die die Grenzen des zu disassemblierenden Speicherbereichs festlegen (abschließen mit CR). Im Beispiel der Tabelle 1 also 0200 bis einschließlich 022F. Achtung: Hier muß man die letzte Adresse eingeben, nicht wie es vielfach noch vorkommt die letzte Adresse + 1.
Auf die Eingabe der beiden Adressen folgt die Rückmeldung "L, P, SP?" Nach Betätigung der Taste L wird das gesamte Programm disassembliert. Drückt man Taste P, dann geschieht das in Blöcken von 15 Befehlen (ein voller Bildschirm, die oberste Zeile wird als letzte vor der Betätigung von P ausgedruckt). Mit der SP-Taste kann Befehl für Befehl, also Zeile für Zeile, disassembliert werden.
Das "aufgedröselte" Programm von Tabelle 1 gibt ein repräsentatives Beispiel dafür wieder, was man nach einer Disassemblierungs-Prozedur erwarten kann. In jedem Fall also die Adresse mit dem OpCode des Befehls. Außerdem das Byte oder die Bytes des Befehls. Auf einigen Zwischenraum folgt das "Befehls-Stenogramm", genannt Mnemonics. Falls relevant, wird als letzte die Operanden-Information ausgedruckt. Bedingte Sprungbefehle sind in die Sprungadresse "übersetzt".
Daten, die nicht als OpCode eines Befehls erkannt werden, erhalten ein Mnemonic aus drei @-Zeichen. Gleichzeitig wird diesen Daten eine Befehlslänge von Eins zugeordnet. Die Daten FF erkennt der Disassembler **nicht** als Label-Opcode an.
Ein Druck auf Taste R, und man befindet sich wieder in PM. Bequemer geht's nicht!
Schließlich die Tasten H und A. Die Betätigung von H löst die gleiche Funktion aus, als wenn die Taste M während des Laufes von PM gedrückt wird. Kurzum: Nach Eingabe von zwei Adressen, gefolgt von CR, gibt der Computer ein Hex-Dump-Listing aus. Warum doppelt moppeln? Weil die dafür benötigte Software wegen der Taste A sowieso vorhanden sein muß. Das "A" steht nämlich für "ASCII-Dump". Was bedeutet das nun wieder? Nun – es handelt sich um einen Hex Dump, dessen auszudruckende Daten als ASCII-Kode eines auszudruckenden Zeichens interpretiert werden. Falls die Daten in den Bereich von 20 bis einschließlich 7E fallen, wird das entsprechende ASCII-Zeichen ausgedruckt. Falls nicht, gibt's nichts zu sehen. Der Sinn des Ganzen? Man kann in der zu prüfenden Software leicht Auszudruckendes, wie beispielsweise Rückmeldungen, lokalisieren. Falls Sie also die Disassembler-Software mit Hilfe des Disassemblers disassemblieren möchten (können Sie noch folgen?), dann werden Sie feststellen, daß die Software reichlich viele Texte enthält.

Und das noch:

Man kann den Ausdruck eines Dump oder eines Listings durch Drücken der BRK-Taste unterbrechen. Der BRK-Sprungvektor führt den 6502-Prozessor nach einem zentralen Punkt, wo auf die Betätigung einer (neuen) Taste gewartet wird.
Von den beiden Adressen zur Begrenzung des Listings oder des Dump muß die zweite Adresse höher als die erste sein. Falls das bei der Eingabe nicht berücksichtigt wurde, muß man die Adressen neu, und zwar jetzt in der richtigen Reihenfolge, eingeben!
Der Disassembler arbeitet nicht nur auf den bekannten Speicherplätzen in den Seiten 00 und 1A, sondern auch auf 0010 . . . 0027. Die restliche ESS511-Software enthält außerdem den Speicherplatz 0028. Das zu disassemblierende Programm darf also diese Adresse nicht verwenden.
Wir wünschen Ihnen viel Spaß als Software-Amateur-Detektiv!