Ein Disassembler für den 6502

In Ergänzung zu einem Decompiler (siehe "Ein Decompiler...." in dieser Vierten Dimension) benötigt man oft eine Möglichkeit, Maschinencode listen zu können. Damit kann man sich ein Bild von den Kernal-Routinen und anderen Code-Definitionen machen. Häufig verfügt der Computer über einen eingebauten oder zuladbaren Maschinensprache-Monitor, der einen Disassembler enthält. Für alle FORTH-Freunde, die mit einem 6502-System arbeiten und keinen Disassembler besitzen, der von FORTH aus ansprechbar ist, folgt hier ein Disassembler, der vollständig in FORTH formuliert ist. Er läßt sich relativ einfach in den vorgenannten Decompiler einbinden.

Untersucht man die Opcodes des 6502 anhand einer hexadezimal sortierten Befehlsliste, so fällt sofort auf, daß alle Befehle, deren niederwertiges Bit Nr. 0 gesetzt ist, eine große Regimäßigkeit aufweisen. Diese Hälfte der Opcoes kann man einfach weiter unterteilen: alle Opcodes, deren Bit Nr. 1 ebenfalls gesetzt ist, ($03, $07, $0B, $0F, $13,...) sind keine gültigen Befehle. Das andere Viertel der Opcodes mit gesetztem Bit Nr. 0 und nicht gesetztem Bit Nr. 1 ($01, $05, $09, $0D,...) sind die 8 Befehle: ORA, AND, EOR, ADC, STA, LDA, CMP und SBC mit je 8 Adressierungsarten (ind,X), Zero-Page, immediate, absolut, (ind),Y , Zero-Page,X , absolut,Y und absolut,X . Es gibt nur die eine logische Ausnahme: einen Befehl STA ZZ gibt es nicht.

Bei der anderen Hälfte der Opcodes mit nicht gesetztem Bit Nr. 0 läßt sich eine Regelmäßigkeit nicht so einfach feststellen. Der folgende Disassembler faßt deshalb diese Hälfte der Opcodes in der Tabelle SHORTCODE mit ihren wesentlichen Daten zusammen. Die Daten der sytematischen Hälfte der Opcodes sind in die beiden kurzen Tabellen SCODE und ADRMODE gefaßt, die von dem Wort SHORTCODE1 ausgewertet werden. Das Wort SHORTCODE0 schließlich liefert die für alle Opcodes benötigten Informationen.

Das Wort DIS schließlich disassembliert ab der Adresse, die auf dem Stack liegt, Zeile für Zeile, bis das Disassembling mit RETURN abgebrochen wird. Es benutzt dazu außer SHORTCODE einige Tabellen mit ASCII-Zeichen. Noch eine Bemerkung zur Art und Weise, wie das Wort TABELLE die Zahlen einliest: Die Übergabe der Zahlen auf dem Stack ist in FIG-6502-Systemen wegen der bedauerlich kleinen Stacktiefe nicht möglich, deshalb der mühsame Weg über BL WORD HERE NUMBER DROP.

von Georg Rehfeldt

Screen nr.:26

```
 0( Disassembler für 6502     -          30.03.84re )
 1
 2 HEX   TOOLS   DEFINITIONS
 3
 4 : TABELLE      ( +n -- )
 5               <BUILDS   0 DO
 6                         BL WORDS HERE NUMBER DROP , DROP LOOP
 7               DOES>     SWAP + COUNT SWAP C$ ;
 8
 9
10 86 TABELLE   SHORTCODE0        ( 8b1 -- 8b2 +n )
11( Tabelle folgt aus Platzgründen im nächsten Screen )
12( Vorsicht beim abtippen . )
13
14 -->
15
```

Screen nr.:27

```
 0( Disassembler für 6502     -          30.03.84re )
 1
 2Alles folgenden Zahlen sind HEXADEZIMAL.
 3Die neuen Worte werden in das ( bereits definierte !!!)
 4Vocabulary TOOLS geschrieben.
 5
 6
 7 TABELLE        ( +n -- )
 8 Ein Definitionswort, das einehn Suchbegriff ins Wörterbuch
 9 schreibt, dann +n Worte durch Leerzeichen von einander getrennt
10 aus dem Eingangstext (INPUTSTREAM), in Zahlen umwandelt und
11 diese als 16b Werte ins Wörterbuch (Kapitel: TOOLS ) schreibt.
12 Ein mit TABELLE definiertes Wort braucht vor dem Aufruf die mit
13 2 multiplizierte Nummer des gewünschten 16b - Wertes und lie-
14 fert diesen getrennt in 2 8b-Werten. Oben auf dem Stapel liegt
15 das höherwertige Byte . ( 8b1 - 8b2 +n )
```

Screen nr.:28

```
 0 0810 0000 0000 0341 2510 0320 0000 0332
 1 0AC1 0000 0000 03A1 0E10 0000 0000 0362
 2 1502 0000 0741 2341 2710 2820 0732 2832
 3 08C1 0000 0000 23A1 2610 0000 0000 2862
 4 2A10 0000 0500 2141 2410 2120 1C32 3132
 5 0CC1 0000 0000 21A1 1010 0000 0000 2162
 6 2810 0000 0000 2941 2610 2920 1CD2 2932
 7 0CC1 0000 0000 29A1 2F10 0000 0000 2962
 8 0000 0000 3241 3141 1710 3610 3232 3132
 9 0000 04C1 32A1 3181 3810 3710 0000 0000
10 3051 1F51 2041 1F41 3410 3310 2032 1F32
11 05C1 0000 20A1 1FB1 1110 3510 2062 1F72
12 1451 0000 1441 1541 1B10 1610 1432 1532
13 09C1 0000 0000 15A1 0F10 0009 0000 1562
14 1C51 0006 1341 1941 1A10 2210 1332 1932
15 08C1 0000 0000 19A1 2E10 0000 0000 1962     --->
```

Screen nr.:29

```
 0( Disassembler für 6502     -          30.03.84re )
 1
 2 SHORTCODE0     ( 8b1  - 8b2 +n )
 3 Ein Wort, definiert durch TABELLE, das 180 16b - Werte zur Ver-
 4 fügung stellt. Es dientzur Ermittlung von Informationen für die
 5 wenig systematischen OPcodes des 6502. Nach Aufruf liefert +n
 6 einen Index auf das Mnemonic. 8b2 trägt 2 Informationen. Das
 7 obere Nibble ( Halbbyte ) ist die Nummer der Adressierungsart,
 8 das untere Nibbel ist die Befehlslänge in Byte - 1. 8b1 muß ei-
 9 ne gerade Zahl sein und ist der mit 2 multiplizierte Index in
10 die Tabelle. Siehe auch TABELLE
11
12
13
14
15
```

Screen nr.:30

```
 0( Disassembler für 6502     -          30.03.84re )
 1
 2 0 VARIABLE SCODE -2 ALLOT
 3 23 C,  2 C,  12 C,  1 C,  30 C,  1E C,  12 C,  2C C,
 4
 5 0 VARIABLE  ADRMOD -2  ALLOT
 6 81 C,  41 C,  51 C,  32 C,  91 C,  A1 C,  72 C,  62 C,
 7
 8 : SHORTCODE1  ( 8b1  --  8b2 +n )
 9               2/ DUP 1 AND
10               IF DROP 0 0
11               ELSE 2/ DUP 7 AND ADRMODE + C$ SWAP
12                    2/ 2/ 2/ 7 AND SCODE + C$
13               THEN ;
14( VARIABLE MODE     0 VARIABLE  LENGTH
15-->)
```

Screen nr.:31

```
 0( Disassembler für 6502     -          30.03.84re )
 1 SCODE -  ( --  addr )
 2 Eine Tabelle von acht 8b Werten, die einen Index auf ein Mnemo-
 3 nic liefern. Nur von SHORTCODE1 benutzt. Siehe dort.
 4
 5 ADRMODE   (  -- adr )
 6 eine Tabelle von acht 8b-Werten, die 2 Informationen tragen:
 7 das obere Nibbel ist ein Index für eine Adrssierungsart, das
 8 untere Nibbel ist die Befehlslänge in Byte - 1. Siehe SHORTCODE
 9
10 SHORTCODE1  ( 8b1  -- 8b2 +n )
11 Ein Wort zur Ermittlung des Index auf die Mnemonictabelle,
12 genauso wie SHORTCODE0 aber nur für ungerade Zahlen.
13 Die Bytes bedeuten dasselbe wie in SHORTCODE0.
14 MODE  ( -- addr ) Variable für die Adressierungsart
15 LENGHT ( -- addr ) Variable für die Befehlslänge in Bytes - 1
```

```
Screen nr.:32
 0( Disassembler für 6502     -          30.03.84re )
 1
 2: SHORTCODE     ( 8bl -- +n )
 3                DUP 1 AND IF DUP 89 =   ( ungerade Codes ? )
 4                          IF DROP 2 THEN SHORTCODE1
 5                        ELSE SHORTCODE0 ( sonst gerade Codes )
 6                          THEN SWAP DUP 3 AND LENGHT !
 7                2/ 2/ 2/ 2/ MODE !   ;
 8
 9: TEXTAB        ( char +n 8b -- )
10          <BUILDS C, 0 DO DUP WORDS HERE COUNT
11                 >R HERE R CMOVE R> ALLOT LOOP DROP
12          DOES> COUNT >R SWAP R * + R> TYPE ;
13
14-->
15
```

```
Screen nr.:33
 0( Disassembler für 6502     -          30.03.84re )
 1
 2 SHORTCODE      ( 8b  -- +n )
 3 Ein Wort zur Ermittlung der nötigen Informationen für alle 6502
 4 OPcodes. 8b ist der 6502-OPcode. +n ist der Index auf die Mne-
 5 monictabelle. Seiteneffekte sind das setzten der Variablen
 6 LENGHT auf die zugehörige Befehlslänge -1 und der Variablen
 7 MODE auf die zutreffende Adressierungsart. s.a. SHORTCODE0 (1)
 8
 9 TEXTTAB        ( char +n 8b -- )
10 Ein Definitionswort, das einen Suchbegriff ins Wörterbuch
11 schreibt, mit 8b als Count-Byteund anschließend +n Worte aus
12 dem Eingangstext, die durch char voneinander getrennt sind, als
13 ASCII-Zeichen ins Wörterbuch schreibt. char und Count werden
14 auf alle +n Worte angewandt. Ein mit TEXTTAB definiertes Wort
15 erwartet den Tabellenindex und druckt das entspr. Wort aus.
```

```
Screen nr.:34
 0( Disassembler für 6502     -          30.03.84re )
 1
 2 BL 39 3 TEXTTAB  .MNEMONIC   ( +n -- )
 3 >BY ADC AND ASL BCC BCS BEQ BIT BMI BNE BPL BRK BVC BVS
 4 CLC CLD CLI CLV CMP CPX CPY DEC DEX DEY EOR INC INX INY
 5 JMP JSR LDA LDX LDY LSR NOP ORA PHA PHP PLA PLP ROL ROR
 6 RTI RTS SBC SEC SED SEI STA STX STY TAX TAY TSX TXA TXS TYA
 7
 8?F 0E 1 TEXTTAB .VOR    ( +n  -- )
 9/ / /A/ /2/8/ / /(/ (/2/2/ /(/
10
11?F 0E  3  TEXTTAB  .NACH   ( +n -- )  /    /    /    /    /
12:    /,X /,Y /,X)/),Y/,X /,Y /     /)     /
13
14: 4U.R  ( u -- )   0 <# # # # #> TYPE ;
15: 2U.R  ( u -- )   0 <# # # #> TYPE ;     -->
```

```
Screen nr.:35
 0( Disassembler für 6502     -          30.03.84re )
 1.MNEMONIC       ( +n -- )
 2 Die Mnemonictabelle von $39 Worten zu 3 Byte Länge, durch TEXT-
 3 TAB definiert. Druckt aus dem Index +n das entsprechende Wort
 4 aus. Die Tabelle der 6502-Mnemonics ist alphanumerisch sortiert
 5
 6.VOR    ( +n -- )
 7 Diese ebenfalls durch TEXTTAB definierte Tabelle von $0E Worten
 8 von 1 Byte Länge, enthält die VOR dem Operanden in Abhängigkeit
 9 von der Adressierungsart anzuzeigenden ASCII-Zeichen.
10
11 .NACH ( +n --.)
12 selbiges wie .VOR für die Zeichen NACH dem Operanden.
13
14 4U.R  ( u -- ) druckt u als vierziffrige Zahl mit führender 0
15 2U.R  ( u -- ) druckt u als zweiziffrige Zahl mit führender 0.
```

```
Screen nr.:36
 0( Disassembler für 6502     -          30.03.84re )
 1 FORTH DEFINITIONS
 2 : DIS  ( addr -- )  BASE @ >R  HEX
 3   BEGIN TOOLS CR DUP 4U.R SPACE DUP C@ DUP 2U.R SPACE
 4         SHORTCODE >R
 5         LENGTH @ DUP IF OVER 1+ C@ 2U.R SPACE THEN
 6         DUP 2 =       IF OVER 2+ C@ 2U.R SPACE THEN
 7         2 SWAP - 3 * SPACES R> MNEMONIC SPACE1+ MODE @ DUP .VOR
 8         0C = IF DUP C@ DUP 80 AND IF 100 - THEN OVER + 2+ 4U.R
 9              ELSE LENGTH @ DUP 2 SWAP - 2* SPACES
10                   ?DUP IF 2= IF DUP @ 4U.R
11                             ELSE DUP C@ 2U.R THEN
12                        THEN  THEN MODE @ .NACH LENGTH @ +
13       KEY 0D =
14    UNTIL DROP R> BASE !  FORTH ;
15
```

```
Screen nr.:37
 0( Disassembler für 6502     -          30.03.84re )
 1
 2 DIS ( addr -- )
 3 Das Wort, das ab  addr disassembliert. Ein Maschinenbefehl wird
 4 in der Form :
 5       addr   OPcode   (byte) (byte) Mnemonic  Operand
 6
 7 Beispiele: 0FA5  6C 70 09 JMP  (0970)
 8            0123  EA       NOP
 9Nach dem listen eines Befehls, wartet DIS auf einen Tastendruck,
10CR bricht das Disassembling ab, jede andere Taste listet den
11nächsten Befehl in der nächsten Zeile.
12
13
14
15
```

Und nun einige Beispiele für diesen 6502 - Dissasembler :

" dup " :
dup compiling cfa f9e § fa0 code

0fa0 dis
0fa0 b5 00 lda z 00,x
0fa2 48 pha
0fa3 b5 01 lda z 01,x
0fa5 4c 70 09 jmp 0970

" r> " :
r> compiling cfa e7f § e81 code

0e81 dis
0e81 ca dex
0e82 ca dex
0e83 68 pla
0e84 95 00 sta z 00,x
0e86 68 pla
0e87 95 01 sta z 01,x
0e89 4c 77 09 jmp 0977

(P.S.: Alle Zahlen sind Hexadezimalzahlen. Die kleinen Buchstaben stehen als deutliches Zeichen für den C64 - Alltag. d.RED.)