Meno Sellschopp

6502-Programme verschiebbar

Wer ein 6502-Programm schreibt, weiß entweder vorher, in welchem Adreßbereich es arbeiten soll, oder setzt nachträglich mit Hilfe des Assemblers jene Adressen um, die ins Programm selbst und in seine Tabellen zeigen. Soweit der Normalfall. Wenn man aber das Programm in ein PROM brennen und trotzdem die Möglichkeit offenhalten will, es in verschiedene Sockel zu stekken, dann wird es schwieriger.

Zu diesem Thema hat mc schon zwei Programme veröffentlicht [1, 2], die aber ein ortsfestes Hilfsprogramm benötigen und nicht alle Operationscodes verarbeiten können. Hier folgt eine Lösung, die solche Nachteile vermeidet. Das Programm (*Bild*) wird vor das Anwenderprogramm gespannt und mit ihm zusammen in das PROM gebrannt. Es erzeugt seinen „Archimedischen Punkt" selbst und verschiebt alle 3-Byte-Befehle bis auf den indirekten Sprung JMP ($...). Entwickelt wurde es auf CBM. Aber es wurden bis auf elf Byte keine Systemadressen verwendet. Nur sie müssen auf anderen 6502-Systemen anders gewählt werden. Und so arbeitet das Programm:

Wo bin ich?

Der Prozessor wird durch JSR auf ein selbst gesetztes RTS veranlaßt, die Rücksprungadresse in den Stack zu schreiben. Damit sie nicht gleich nach dem Rücksprung bei einem System-Interrupt überschrieben werden kann, muß vorher SEI gesetzt werden. Diese Adresse wird vom Stack gelesen und aus ihr die tatsächliche Lage von „Reloc" und „Programm" abgeleitet. Die Anfangsadresse von „Programm" wird in ADR0 festgehalten, die Adresse von „Reloc" mit einem davorgesetzten JMP ins „Relais" geschrieben. „Relais" dient als feste Anlaufstelle für alle späteren Aufrufe von „Reloc".

Adressierung im Anwenderprogramm

Das Anwenderprogramm beginnt unabhängig von seiner tatsächlichen Lage mit der virtuellen Adresse $0000, d. h. die Argumente aller 3-Byte-Befehle, die in das Anwenderprogramm zeigen, sind nicht auf die Ablageadresse, sondern auf „Programm" als 0-Adresse zu beziehen.

Vor jedem dieser Befehle muß ein „JSR Relais" erfolgen. Der Sprung auf „Programm" und das Programmbeispiel zeigen das.

Wohin soll ich?

„Reloc", über „JSR Relais" aufgerufen, rettet zunächst alle Registerinhalte. Dann liest es wieder die Rücksprungadresse aus dem Stack und schreibt eine um drei Stellen höhere Adresse in den Stack zurück. SEI muß diesmal nicht gesetzt werden, weil anders als in „Wo bin ich?" der Rücksprung noch nicht erfolgt ist. Der relativ zu „Programm" adressierte 3-Byte-Befehl wird mit seinem Argument ins „Double" kopiert und dort auf seinen tatsächlichen Wert umgerechnet. Statt des Originals wird dieses Double bearbeitet, nachdem vorher noch alle Register wiederhergestellt sind. Der folgende Rücksprung ins Anwenderprogramm setzt hinter dem Original auf, weil ja der Stack entsprechend korrigiert ist. Nur wenn der verschobene Befehl ein JMP ist, muß die Rücksprungadresse ganz vom Stack genommen werden, sonst geht das nächste RTS in die Irre. Offsets, die aus dem Anwenderprogramm hinausführen, dürfen selbstverständlich nicht mit „JSR Relais" aktualisiert werden.
Die Umrechnung der Argumente während des Programmablaufs kostet natürlich Zeit, und zwar genau 170 Taktzyklen, das sind 0,2 ms, für jede Umrechnung. Bei klug kalkulierter Programmgestaltung wird man das meistens verschmerzen können gegenüber dem Vorteil, daß man jetzt wirklich beliebig verschiebbare PROMs herstellen kann.

Literatur

[1] *Kittel, Peter:* Große Sprünge mit dem 6502. mc 1981, Heft 1, Seite 51...53.
[2] *Mester, Roland:* Relokatible 6502-Programme. mc 1982, Heft 11, Seite 56...57.

```
0000 0000                       ;*********************************
0000 0000                       ;*                               *
0000 0000                       ;*   RELOCATIBLE ADRESSIERUNG    *
0000 0000                       ;*      IN 6502-PROGRAMMEN       *
0000 0000                       ;*      (C)  M. SELLSCHOPP       *
0000 0000                       ;*                               *
0000 0000                       ;*********************************
0000 0000                       ;
0000 0000           STACK       =$0100
0000 0000                       ;   (11 BYTES IM RAM TYPISCH FUER CBM)
0000 0000           ZEIGER      =$54    ;2 BYTES ZUR (.),Y-ADRESSIERUNG
0000 0000           ADR0        =$033A ;2 BYTES FUER ADRESSE 'PROGRAMM'
0000 0000           RELAIS      =$033C ;3 BYTES FUER 'JMP RELOC'
0000 0000           DOUBLE      =$033F ;4 BYTES FUER BEFEHLSKOPIE + RTS
0000 0000                       ;
0000 0000                       *=$040D ; OHNE AENDERUNG VERSCHIEBBAR!
040D 0000                       &=$1000
040D 1000                       ;
040D 1000           P00         ;   "WO BIN ICH?"
040D 1000                       ;   AKTUELLE ADRESSE AUS STACK HOLEN
040D 1000 78                    SEI
```

Mit diesem Programm als Vorspann sind Routinen frei im Speicher zu verschieben; ein Anwendungsbeispiel finden Sie ab Adresse $49C

```
040E 1001 A4 00              LDY $00
0410 1003 A9 60              LDA #$60
0412 1005 85 00              STA $00
0414 1007 20 00 00           JSR $0000
0417 100A BA         P01     TSX
0418 100B 84 00              STY $00
041A 100D CA                 DEX
041B 100E BD 00 01           LDA STACK,X
041E 1011 BC 01 01           LDY STACK+1,X
0421 1014 58                 CLI
0422 1015                    ;   'RELAIS' EINRICHTEN, BEZUGS-
0422 1015                    ;   ADRESSE 'PROGRAMM' IN ADR0
0422 1015 18                 CLC
0423 1016 69 2F              ADC #[RELOC-P01+1]
0425 1018 90 01              BCC P02
0427 101A C8                 INY
0428 101B 8D 3D 03 P02       STA RELAIS+1
042B 101E 8C 3E 03           STY RELAIS+2
042E 1021 18                 CLC
042F 1022 69 57              ADC #[PROGRAMM-RELOC]
0431 1024 90 01              BCC P03
0433 1026 C8                 INY
0434 1027 8D 3A 03 P03       STA ADR0
0437 102A 8C 3B 03           STY ADR0+1
043A 102D A9 4C              LDA #$4C
043C 102F 8D 3C 03           STA RELAIS
043F 1032                    ;
043F 1032                    ;   AUF 'PROGRAMM' SPRINGEN
043F 1032 20 3C 03           JSR RELAIS
0442 1035 4C 00 00           JMP $0000
0445 1038                    ;
0445 1038            RELOC   ;   "WOHIN SOLL ICH?"
0445 1038                    ;   ALLE REGISTERWERTE RETTEN
0445 1038 08                 PHP
0446 1039 48                 PHA
0447 103A 8A                 TXA
0448 103B 48                 PHA
0449 103C 98                 TYA
044A 103D 48                 PHA
044B 103E                    ;   RUECKSPRUNGADRESSE IN ZEIGER,
044B 103E                    ;   ADRESSE AUF STACK ERHOEHEN
044B 103E BA                 TSX
044C 103F 18                 CLC
044D 1040 BD 05 01           LDA STACK+5,X
0450 1043 85 54              STA ZEIGER
0452 1045 69 03              ADC #3
0454 1047 9D 05 01           STA STACK+5,X
0457 104A BD 06 01           LDA STACK+6,X
045A 104D 85 55              STA ZEIGER+1
045C 104F 69 00              ADC #0
045E 1051 9D 06 01           STA STACK+6,X
0461 1054                    ;   BEFEHL AUS PROGRAMM
0461 1054                    ;   IN 'DOUBLE' KOPIEREN UND
0461 1054                    ;   ANFANGSADRESSE ADDIEREN
0461 1054 A0 01              LDY #1
0463 1056 B1 54              LDA (ZEIGER),Y
0465 1058 8D 3F 03           STA DOUBLE
0468 105B C8                 INY
0469 105C 18                 CLC
046A 105D B1 54              LDA (ZEIGER),Y
046C 105F 6D 3A 03           ADC ADR0
046F 1062 8D 40 03           STA DOUBLE+1
0472 1065 C8                 INY
0473 1066 B1 54              LDA (ZEIGER),Y
0475 1068 6D 3B 03           ADC ADR0+1
0478 106B 8D 41 03           STA DOUBLE+2
047B 106E                    ;   RTS HINTER DOUBLE SETZEN
047B 106E A9 60              LDA #$60
047D 1070 8D 42 03           STA DOUBLE+3
0480 1073                    ;   REGISTERWERTE ZURUECKHOLEN,
0480 1073                    ;   AKKU UND STATUS IN ZEIGER RETTEN
0480 1073 68                 PLA
0481 1074 A8                 TAY
0482 1075 68                 PLA
0483 1076 AA                 TAX
0484 1077 68                 PLA
0485 1078 85 54              STA ZEIGER
0487 107A 68                 PLA
0488 107B 85 55              STA ZEIGER+1
048A 107D                    ;   WENN JMP, DANN KEIN RUECKSPRUNG
048A 107D AD 3F 03           LDA DOUBLE
048D 1080 C9 4C              CMP #$4C
048F 1082 D0 02              BNE REL1
0491 1084 68                 PLA
0492 1085 68                 PLA
0493 1086                    ;   AKKU UND STATUS ZURUECKHOLEN
0493 1086 A5 55      REL1    LDA ZEIGER+1
0495 1088 48                 PHA
0496 1089 A5 54              LDA ZEIGER
0498 108B 28                 PLP
0499 108C                    ;   AUF DOUBLE SPRINGEN, DORT DEN
0499 108C                    ;   KORRIGIERTEN BEFEHL AUSFUEHREN
0499 108C 4C 3F 03           JMP DOUBLE
049C 108F                    ;
049C 108F            PROGRAMM    ;** BEZUGSPUNKT FUER RELOC.ADRESSEN **
049C 108F                    ;   HIER NUR BEISPIEL FUER DIE
049C 108F                    ;   ANWENDUNG DES HILFSPROGRAMMS
049C 108F                    ;
049C 108F            BILDSCHIRM =$8000 ;(CBM)
049C 108F                    ;
049C 108F 20 3C 03           JSR RELAIS
049F 1092 20 07 00           JSR [UNTERPROGRAMM-PROGRAMM]
04A2 1095 60                 RTS ;  -> READY
04A3 1096            UNTERPROGRAMM
04A3 1096 20 3C 03           JSR RELAIS
04A6 1099 AE 1A 00           LDX [TABELLE-PROGRAMM]
04A9 109C 20 3C 03 U1        JSR RELAIS
04AC 109F BD 1A 00           LDA [TABELLE-PROGRAMM],X
04AF 10A2 9D 00 80           STA BILDSCHIRM,X
04B2 10A5 CA                 DEX
04B3 10A6 D0 F4              BNE U1
04B5 10A8 60                 RTS
04B6 10A9            TABELLE     ;   (CBM-BILDSCHIRMCODE)
04B6 10A9 1F A0 B6           .31,/ 6502 RELOCATIBLE ADRESSIERUNG /
04B9 10AC B5 B0 B2
04BC 10AF A0 92 85
04BF 10B2 8C 8F 83
04C2 10B5 81 94 89
04C5 10B8 82 8C 85
04C8 10BB A0 81 84
04CB 10BE 92 85 93
04CE 10C1 93 89 85
04D1 10C4 92 95 8E
04D4 10C7 87 A0
```

sten Zeile ein anderer Wert zuzuweisen (bei 32 kByte: 48860; bei 48 kByte: 65240).
Das Basic-Startprogramm wird bei der Texteingabe überschrieben. Nach der Erstellung muß es also erst einmal auf Kassette gesichert werden, ehe man es ausprobiert! Vor jeder neuen Texteingabe ist es also neu zu laden, oder aber man springt das Maschinenprogramm, das bis zum Ausschalten im Speicher stehen bleibt, mit SYSTEM / (Startadresse) direkt an. Natürlich läßt sich auch noch eine Zeile mit Befehlen zur Drucker-Voreinstellung in das Startprogramm einfügen. Leider schützt die CLEAR-Vorgabe in Zeile 9 das geladene Maschinenprogramm nicht vor sich selbst. Wenn man den Speicher bis zum Ende vollschreibt, frißt sich das Maschinenprogramm selbst auf. Programmabsturz und vollständiger Textverlust sind die Folge. Das wäre fatal! Daher sorgt eine kleine Testroutine dafür, daß der noch freie Speicherraum ständig überwacht wird. Nähert sich der Text zu sehr dem Speicherende, wird die Eingabe selbsttätig abgebrochen, die Zeile abgeschlossen und READY aufgerufen. Dadurch wird der Text gerettet.

Erweiterung für den AIM-65-Disassembler

Die CMOS-CPU 65CXX von Rockwell besitzt einen erheblich erweiterten Befehlssatz gegenüber der NMOS-Version (59 neue OP-Codes). Das im *Bild 1* aufgelistete Maschinenprogramm disassembliert die zusätzlichen Befehle. Es wird erst wirksam, wenn der AIM-Disassembler Befehle nicht interpretieren kann. Das Programm schaltet den Drucker zu Beginn ein und am Ende wieder aus. Der Ausdruck erfolgt linksbündig, da alle 20 Schreibstellen benötigt werden. Ein Umschreiben des Programms auf andere Mikrocomputer ist mit einem relativ hohen Aufwand verbunden, da umfangreiche Programmteile des AIM-65/PC100 benutzt werden.
Der Umgang mit dem Programm erfolgt so: ★ = 200 und <G>. Nach FROM bzw. TO gibt man die Start- bzw. Endadresse des zu untersuchenden Programms ein, jeweils gefolgt von Return. *Bild 2* zeigt die Disassemblierung eines Testprogrammes, das alle zusätzlichen Befehle enthält. *Heinrich Siggemann*

```
<0200> 20 13 EA 20 A3 E7 20 D7 E5 20 3B E8 20 A7 E7 20
<0210> 13 EA A9 00 8D 11 A4 A9 01 8D 19 A4 20 2B E7 AD
<0220> 69 A4 C9 BF F0 10 A2 00 BD 61 A4 9D 60 A4 E8 E0
<0230> 13 D0 F5 4C 55 03 AD 26 A4 85 F1 AD 25 A4 85 F0
<0240> D0 02 C6 F1 C6 F0 20 13 EA A5 F1 20 46 EA A5 F0
<0250> 20 46 EA 20 3E E8 A0 00 B1 F0 20 46 EA 20 3E E8
<0260> B1 F0 D9 F2 03 F0 1D C8 C0 1B D0 F6 29 8F D9 F2
<0270> 03 F0 11 C8 C0 1F D0 F6 20 D4 E7 20 D4 E7 20 D4
<0280> E7 4C 55 03 84 F2 B9 D3 03 AA A0 03 BD 11 04 20
<0290> 7A E9 E8 88 D0 F6 A6 F2 BD 4E 04 85 F5 20 A0 02
<02A0> 18 68 65 F5 85 F6 68 90 01 1A 85 F7 6C F6 00 4C
<02B0> 55 03 20 3E E8 20 78 03 4C 4D 03 20 3E E8 20 80
<02C0> 03 4C 45 03 20 3E E8 20 78 03 20 8E 03 4C 4D 03
<02D0> 20 3E E8 20 80 03 20 8E 03 4C 45 03 20 3E E8 A9
<02E0> 28 20 7A E9 20 78 03 A9 29 20 7A E9 4C 4D 03 20
<02F0> 3E E8 A9 23 20 7A E9 20 78 03 4C 4D 03 20 3E E8
<0300> A9 41 20 7A E9 4C 55 03 20 3E E8 A9 28 20 7A E9
<0310> 20 80 03 A9 29 20 7A E9 20 8E 03 4C 45 03 20 99
<0320> 03 20 78 03 20 3E E8 E6 F0 D0 02 E6 F1 20 AA 03
<0330> 4C 45 03 20 3E E8 20 AA 03 4C 4D 03 20 99 03 20
<0340> 78 03 4C 4D 03 EE 25 A4 D0 03 EE 26 A4 EE 25 A4
<0350> D0 03 EE 26 A4 A9 80 8D 16 A4 8D 11 A4 20 24 EA
<0360> 0E 11 A4 38 AD 1C A4 ED 25 A4 AD 1D A4 ED 26 A4
<0370> 90 03 4C 17 02 4C 82 E1 A0 01 B1 F0 20 46 EA 60
<0380> A0 02 B1 F0 20 46 EA 88 B1 F0 20 46 EA 60 A9 2C
<0390> 20 7A E9 A9 58 20 7A E9 60 A0 00 B1 F0 29 70 4A
<03A0> 4A 4A 4A 20 51 EA 20 3E E8 60 18 A5 F0 69 02 85
<03B0> F3 A5 F1 69 00 85 F4 18 A0 01 B1 F0 10 02 C6 F4
<03C0> 65 F3 85 F3 90 02 E6 F4 A5 F4 20 46 EA A5 F3 20
<03D0> 46 EA 60 00 10 02 1A 12 1D 20 06 17 27 2A 25 3A
<03E0> 3A 25 31 04 08 0D 34 37 27 2A 25 0B 3A 3A 2E 23
<03F0> 2C 15 80 12 32 52 72 92 B2 D2 F2 04 14 34 64 74
<0400> 89 1A 3A 5A 7A DA FA 0C 1C 3C 7C 9C 9E 07 87 0F
<0410> 8F 42 52 41 4E 44 45 43 4D 50 48 59 4A 4D 50 4C
<0420> 59 4F 52 41 44 43 42 42 53 42 43 45 4F 52 53 54
<0430> 41 4C 44 41 53 4D 42 49 54 53 42 54 52 42 42 52
<0440> 4D 42 49 4E 43 50 48 58 50 4C 58 53 54 5A 94 3D
<0450> 3D 3D 3D 3D 3D 3D 3D 13 13 25 13 25 50 5E 5E 10
<0460> 10 10 10 10 1C 1C 31 69 1C 31 9D 9D 7F 7F

FROM=0200 TO=0300  <75AF>
FROM=0300 TO=0400  <6724>
FROM=0400 TO=046D  <2068>
```

Bild 1. Mit Hilfe dieses Programmes kann der AIM-65 auch die Befehle der CMOS-CPU disassemblieren

```
2001 90 BCC 2043
2003 80 BRA 2017
2005 12 ORA (12)
2007 32 AND (12)
2009 52 EOR (12)
200B 72 ADC (12)
200D 92 STA (12)
200F B2 LDA (12)
2011 D2 CMP (12)
2013 F2 SBC (12)
2015 04 TSB 12
2017 14 TRB 12
2019 34 BIT 12,X
201B 64 STZ 12
201D 74 STZ 12,X
201F 07 RMB0 12
2021 17 RMB1 12
2023 27 RMB2 12
2025 37 RMB3 12
2027 47 RMB4 12
2029 57 RMB5 12
202B 67 RMB6 12
202D 77 RMB7 12
202F 87 SMB0 34
2031 97 SMB1 34
2033 A7 SMB2 34
2035 B7 SMB3 34
2037 C7 SMB4 34
2039 D7 SMB5 34
203B E7 SMB6 34
203D F7 SMB7 34
203F 89 BIT #56
2041 1A INC A
2042 3A DEC A
2043 5A PHY
2044 7A PLY
2045 DA PHX
2046 FA PLX
2047 0C TSB 1234
204A 1C TRB 1234
204D 3C BIT 1234,X
2050 7C JMP (1234),X
2053 9C STZ 1234
2056 9E STZ 1234,X
2059 0F BBR0 34 205C
205C 1F BBR1 34 206E
205F 2F BBR2 34 2072
2062 3F BBR3 34 2095
2065 4F BBR4 34 20B8
2068 5F BBR5 34 20DB
206B 6F BBR6 34 1FEE
206E 7F BBR7 34 2001
2071 8F BBS0 34 2014
2074 9F BBS1 34 2027
2077 AF BBS2 34 203A
207A BF BBS3 12 204D
207D CF BBS4 12 2058
2080 DF BBS5 12 2073
2083 EF BBS6 12 207E
2086 FF BBS7 12 2088
2089 5E LSR 1234,X
```

Bild 2. Ein kleines Testprogramm zeigt die neuen Befehle