6502 Programmieren - Teil 11#

von Uwe Röder#

Willkommen zum elften Teil des Assembler-Lehrganges!

Jetzt nachdem alle Befehle erklärt worden sind, werde ich praxisorientiert vorgehen und mich der Anwendung der Befehle bei irgendwelchen Problemen widmen.

Zuallererst werde ich Ihnen verschiedene Arten, eine Schleife zu realisieren, vorführen. Schleifen braucht man beim Programmieren in Assembler öfter als in Basic oder allen anderen Hochsprachen. Teilweise ist die Programmierung einer Schleife in Assembler aber bedeutend komplizierter als in einer Hochsprache.

Ich benutze als Zählregister am liebsten das X- oder Y- Register, da ich diese einfacher kontrollieren kann als eine Speicherstelle, und da der Akku in aller Regel zum Abarbeiten der Befehle innerhalb der Schleife benötigt wird.

Beispiel:

```
00010 S      LDA #0
00020        TAY
00030 .1     STA $5000,Y
00040        INY
00050        BNE .1
00060        RTS
```

Zu Beginn dieser Schleife werden Akku und Y-Register auf den Wert Null gesetzt. Es wird in der Schleife, die hier nur aus STA.., INY und BNE .1 besteht, jeweils die Adresse $5000+Y auf Null gesetzt. Wenn das Y-Register alle Werte von 0 bis 255 durchlaufen hat, wird es bekanntlich wieder 0, das heißt das Zero-Flag wird gesetzt und der Computer springt bei dem relativen, bedingten Sprung nicht an den Anfang der Schleife zurück.

Nun schön, die Schleife wurde also 256 mal abgearbeitet, doch was soll man tun, wenn man mehr oder weniger Durchläufe möchte?

Der zweite Fall, das weniger Durchläufe benötigt werden, ist sehr einfach: Die Durchlaufgrenze wird über einen CPY-Vergleich festgelegt.

Beispiel:

```
00010 S      LDA #0
00020        TAY
00030 .1     STA $5000,Y
00040        INY
00050        CPY #$80
00060        BNE .1
00070        RTS
```

Hier wird die Schleife verlassen, wenn Y den Wert $80 erreicht hat. Das heißt, dass innerhalb der Schleife Y die Werte von 0 bis $7F angenommen hat. Bei Y=$80 wurde die Schleife sofort verlassen. Die Adresse $5000,Y = $507F war also die letzte veränderte Adresse.

Wenn mehr Durchläufe als 256 erforderlich sind, muss man sich überlegen, was in Anbetracht des Schleifenkörpers angebracht ist. Wenn ich in unserem Beispiel nicht eine Page (256 Bytes) sondern drei Pages löschen möchte, so bleibt die Schleifenbedingung die gleiche:

```
00010 S      LDA #0
00020        TAY
00030 .1     STA $5000,Y
00040        STA $5100,Y
00050        STA $5200,Y
00060        INY
00070        BNE .1
00080        RTS
```

In diesem Fall konnte man mehr als 256 Durchläufen aus dem Weg gehen, aber bei vielen Problemen ist dies nicht möglich. Eine andere Methode, ist eine Schleife, die in einer anderen Schleife liegt, so dass X- und Y- Register Zählregister sind.

Beispiel:

```
00010 S      LDA #0
00020        STA $D0
00030        STA $D1
00040        TAX
00050        TAY
00060 .1     INC $D0
00070        BNE .2
00080        INC $D1
00090 .2     INY
00100        CPY #40
00110        BNE .1
00120        LDY #0
00130        INX
00140        CPX #$80
00150        BNE .1
00160        RTS
```

Die innere Schleife erhöht den Wert von $D0/D1 40 mal um 1, dies entspricht also einer einfachen Addition um 40. Die äußere Schleife wird $80 mal (0 bis 7F) abgearbeitet, also wird der Wert von D0/D1 insgesamt von 0 auf 128*40 = 5120 erhöht.

Es ist aber nicht immer erlaubt, oder günstig, in Schleifen die beiden Register zu benutzen, zum Beispiel bei Schleifen die Unterprogramme aufrufen, die die Register verändern. Hier müsste vor jedem Aufruf der Inhalt der Register zwischengespeichert und nach Rückkehr in die Schleife wieder zurückgeholt werden.

In solchen Fällen bietet es sich an, direkt Speicherstellen als Schleifenzähler einzusetzen.

Beispiel:

```
00010 S      LDA #0
00020        STA $D0
00030        STA $D1
00035        LDY #0
00040 .1     LDA ($D0),Y
00050        LDX #$0
00060        JSR DEZ0
00062        LDA #$9B
00064        JSR PUTCHAR
00070        INC $D0
00080        BNE .2
00090        INC $D1
00100 .2     LDA $D0
00110        CMP #$80
```

```
00120        BNE .1
00130        LDA $D1
00140        CMP #$4
00150        BNE .1
00160        RTS
```

DEZ0 und PUTCHAR sind hier die Ausgaberoutine für Dezimalzahlen von der Bibo-Assembler-Tooldisk 1.

Hier würden die Inhalte der Adressen von 0 bis $47F am Bildschirm ausgegeben werden. Nach jedem Durchlauf würde die Adresse in $D0/$D1 um eins erhöht und mit $480 verglichen werden. Da beide Adressen 16-Bit Adressen sind, müssen wir diese Byte für Byte miteinander vergleichen. Aus Zeitgründen empfiehlt es sich erst das Low-Byte zu testen. In unserem Fall passiert es nämlich nur viermal, dass beide Low-Bytes gleich sind, so dass die High-Bytes auch getestet werden. Würden erst die High-Bytes geprüft werden, so würden diese 128 mal identisch sein, so dass 127 überflüssige Low-Bytes-Tests gemacht werden müssten.

Allgemein lässt sich somit über Schleifen sagen, dass man unbedingt beachten sollte, dass die Zählvariablen nicht von Teilen der Schleife unbeabsichtigt verändert werden. Auch ist die Position der Durchlaufabfrage teilweise von entscheidender Bedeutung, wenn die Routine bei Nichterfüllung der Durchlaufbedingung nicht abgearbeitet werden soll, muss die Abfrage am Anfang des Schleifenkörpers stehen. Wenn die Routine trotzdem mindestens einmal ausgeführt werden soll, muss sie am Ende des Schleifenkörpers stehen.

Wenn man mehrere Schleifen schachtelt, muss man unbedingt die Struktur der Schleifen beachten, so dass wirklich eine Schleife in der anderen liegt und nicht zum Beispiel innerhalb der innersten Schleife ein Schleifenzähler der äußeren Schleifen verändert wird.

Wenn Sie dies alles beachten, dürfte es keine Probleme geben. Wichtig ist immer nur, dass Sie sich vor dem eigentlichen Programmieren schon über die Struktur der Schleife im klaren sein müssen.

Bis nächsten Monat.
Ihr Uwe Röder
CSM 06/1989

Der Artikel entstammt der Kursreihe ?6502 Programmieren? des Compy Shop Diskettenmagazins. Die Kursreihe besteht aus 14 Kursen, die im Laufe des Jahres 2011 in unregelmäßigen Abständen einzeln veröffentlicht werden, bzw. anschließend als Zusammenzug als ABBUC-Buch ?6502 Programmieren? erscheinen.
Koordination: Volkert Barr (volkert@nivoba.de)
Version 1.1 / 2011-01-23