6502 Programmieren - Teil 4#

von R. Wilde#

So so, Ihr wagt es also auch diesmal, in den Lehrgang reinzuschauen. Aber ernsthaft, nachdem wir in den ersten drei Lehrgangsteilen reine Theorie hatten, wollen wir uns diesmal in den harten Alltag der Praxis begeben.

Ich habe lange überlegt, womit wir im ersten Praxisteil anfangen sollen. Nachdem ich mir MEINE ersten Schritte in Assembler vor Augen führte, entschied ich mich für die Entwicklung einer einfachen Schleife, mit der eine Speicherseite im Computer mit einem beliebigen Wert gefüllt werden soll.

Zu diesem Zweck müssen wir Maschinenbefehle finden, mit denen Werte geladen und gespeichert werden können. Schauen wir uns die Liste der Maschinenbefehle (Teil 2) genauer an, finden wir sofort Befehle, die mit den Abkürzungen 'LD' und 'ST' beginnen. Diese Buchstaben stehen für 'LOAD' wie 'LADEN' und 'STORE' wie 'SPEICHERN'. Der Buchstabe dahinter bezeichnet das Register des 6502, mit dem gearbeitet werden soll - 'A' für 'ACCU', 'X' für das Indexregister 'X' und 'Y' für das Indexregister 'Y'. Wenn Ihr Euch die Beschreibung der Maschinenbefehle und der Adressierungsarten genau angesehen habt, könnt Ihr Euch sicher auch denken, warum die Register X und Y Indexregister genannt werden. Doch dazu kommen wir später noch.

Um die Erinnerung noch einmal aufzufrischen - LDA bedeutet Lade ACCU entweder mit einem festen Wert oder mit dem Inhalt der angegebenen Speicherstelle. STA bedeutet speichere den Inhalt des ACCU's in der angegebenen Speicherstelle. Mit diesen Befehlen können wir also schon einmal arbeiten. Für den Anfang laden wir den ACCU mit dem Wert 1 und speichern den Wert dann in der ersten Speicherstelle der Speicherseite 80 (in HEXadezimal $50).

```
10 LDA #1
20 STA $5000
30 RTS
```

Wie wir sehen ist das recht einfach. Unsere selbstgestellte Aufgabe lautete aber, dass wir eine ganze Speicherseite mit einem beliebigen Wert füllen wollen.

Ich sehe jetzt einige von Euch angestrengt nachdenken, was denn wohl eine Speicherseite ist. Also will ich das einmal kurz klären. Eine Speicherseite besteht aus 256 Speicherstellen. Warum aber '256'? Auch das ist recht einfach zu erklären. Wenn Ihr Euch immer brav alle Artikel in den Magazinen durchgelesen habt, sollte Euch noch der Artikel im Compy Shop Magazin April '88 über die Zahlensysteme in Computern in Erinnerung sein. Dort war zu erkennen, dass ein Computer vorzugsweise in dem Zahlensystem 'HEXADEZIMAL' arbeitet.

Dieses Zahlensystem beruht auf der Basis 16, das heißt, dass hier nicht wie beim dezimalen System bis 10 gezählt wird, bevor die nächst höhere Stelle heraufgezählt wird, sondern dass die nächst höhere Stelle erst dann um 1 erhöht wird, wenn der Wert 16 erreicht ist. Da aber die Zahlen 10 bis 15 zweistellig sind, wurden diese durch die Buchstaben A bis F ersetzt. Es wird also folgendermaßen gezählt:

0 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 usw.

Aus dem Artikel über die Zahlensysteme war auch ersichtlich, das eine Speicherstelle Werte bis 255, in HEX = $FF, aufnehmen kann. Nun, das ist auch die magische Grenze für eine Speicherseite, denn mit einem Byte, also einem Wert zwischen 0 und 255 (00 bis FF), kann der Prozessor auch nur 256 Speicherstellen ansprechen.

Alles klar? Nein? Auch nicht schlimm. Mit der Zeit beim praktischen Arbeiten in Maschinensprache wird sich das Verständnis schon einstellen.

Um nun wieder zum Thema zurückzukommen, bis jetzt haben wir nur eine Speicherstelle der Speicherseite gefüllt. Man kann jetzt natürlich die restlichen 255 Speicherstellen auch in der bewährten Weise mit dem Befehl 'STA' füllen. Das sähe dann aber folgendermaßen aus :

```
10 LDA #1
20 STA $5000
30 STA $5001
40 STA $5002
50 STA $5003
60 STA $5004
70 STA $5005
usw.
```

Sehr aufwendig! Nicht wahr?

Wie können wir das Problem also anders lösen ? Dazu wollen wir uns erst einmal die Problemlösung in BASIC anschauen:

```
10 FOR X=0 TO 255
20 POKE 20480+X,1
30 NEXT X
40 END
```

Hier wird die gesamte Speicherseite in einer Schleife gefüllt. Im POKE-Befehl wird zur angegebenen Adresse der Inhalt der Variablen 'X' zugezählt, dann haben wir die Adresse der Speicherstelle, in die der Wert 1 abgelegt wird. In ähnlicher Weise können wir eine Schleife in Maschinensprache aufbauen. Dazu benutzen wir ein Indexregister als Schleifenzähler. Wie das funktioniert, zeige ich erst einmal anhand einer einzelnen Schleife.

```
10 LDX #0
20 LOOP INX
30 BNE LOOP
40 RTS
```

Das ist die einfachste Schleife in Maschinensprache. Zuerst wird das X-Register mit dem Wert 0 geladen. Der INX-Befehl zählt das X-Register um den Wert 1 herauf. Solange Z nicht 0 ist, springt BNE zum Label LOOP und das X-Register erneut inkrementiert. Das Label LOOP ist eine Sprungmarke, die der Assembler in eine relative Sprung-Adresse umwandelt, so dass man diese nicht selbst errechnen muss.

Den BNE-Befehl können wir mit folgender BASIC-Anweisung vergleichen :

```
30 IF X<>0 THEN GOTO Zeile
```

Also sähe die Schleife in BASIC folgendermaßen aus :

```
10 X=0
20 X=X+1
30 IF X<>256 THEN 20
40 END
```

Der BNE-Befehl bedeutet also in dieser Schleife soviel wie - Verzweige zu der Stelle, die mit dem Label 'LOOP' bezeichnet ist, wenn das Ergebnis der Rechenfunktion 'INX' ungleich 0 ist. Auf diese Weise wird das X-Register von 0 beginnend solange um Eins herauf gezählt, bis das Ergebnis wieder 0 ist. Das geschieht, wenn im X Register der Wert 255 steht und das Register nochmals um 1 raufgezählt wird.

Da der Assembler beim Assemblieren nicht zeilenorientiert arbeitet, nach der BNEAnweisung aber ein Sprungziel stehen muss, wurden die sogenannten LABEL's erfunden. Solch ein LABEL haben wir also bereits in der Maschinen-Programm Schleife stehen.

Wie wir in der BASIC-Schleife gesehen haben, fehlt uns nun nur noch ein Befehl, der ähnlich wie der POKE Adr+X-Befehl arbeitet. Da drängt sich gerade der STA Adr,X - Befehl auf. Aus der Beschreibung der Maschinenbefehle ersehen wir, dass dieser Befehl praktisch genauso zu handhaben ist wie der BASIC-Befehl. Zu der Adresse, die hinter der STA-Anweisung steht, wird noch der Wert im X-Register zugezählt. Dann erst haben wir die Adresse, in die der Wert aus dem ACCU abgelegt wird.

Aus dieser Funktion ergibt sich auch die Bezeichnung des X-Registers als Indexregister, da das X-Register als Index auf die Speicherstelle verwendet wird, in der im Endeffekt der Wert 1 abgelegt wird.

Nun können wir die komplette Schleife zusammenfügen:

```
10 LDA #1
20 LDX #0
30 LOOP STA $5000,X
40 INX
50 BNE LOOP
60 RTS
```

Zum Schluss steht der RTS-Befehl. Der sorgt dafür, dass nach Beenden der Schleife dahin zurückgesprungen wird, von wo aus unser kleines Maschinenprogramm aufgerufen wurde.

Damit Ihr Euer erstes Maschinenprogramm auch einmal abarbeiten lassen könnt, müsst Ihr das komplette Listing (das letzte Beispiel mit den Zeilennummern 10-60) im 'BIBO-Assembler' eingeben. Ist das geschehen, muss noch die Anweisung 'ASM' gegeben werden, damit unser Assemblerlisting in die gültigen Maschinen-Anweisungen umgesetzt wird. Bei diesem Vorgang ist zu erkennen, dass das Maschinenprogramm ab der Adresse $4000 abgelegt wird. Zum Starten jetzt nur noch 'RUN $4000' eingeben. Sollte der Assembler sich nicht sofort wieder mit 'Edit' zurückmelden, habt Ihr irgendwo innerhalb der 6 Zeilen einen Fehler gemacht.

Wenn Ihr Euch das Ergebnis der abgearbeiteten Maschinenroutine anschauen wollt, müsst Ihr vom Assembler aus nur 'MON' aufrufen und '5000,,' eingeben. Soll die entsprechende Speicherseite mit einem anderen Wert gefüllt werden, muss dieser Wert nur statt der 1 in dem Befehl 'LDA #1' eingesetzt werden. Anschließend wieder 'ASM' und 'RUN $4000' eingeben.

Ich hoffe, dass Euch dieser erste praktische Teil gefallen hat. Ihr könnt Euch ja schon einmal überlegen, wie mehr als nur eine Speicherseite mit einem bestimmten Wert gefüllt werden kann. Eine Antwort darauf findet Ihr dann im nächsten Teil des Lehrganges.

Bis dahin viele Grüße, Euer R. Wilde.
CSM / 9.1988

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 1.2 / 2011-02-12