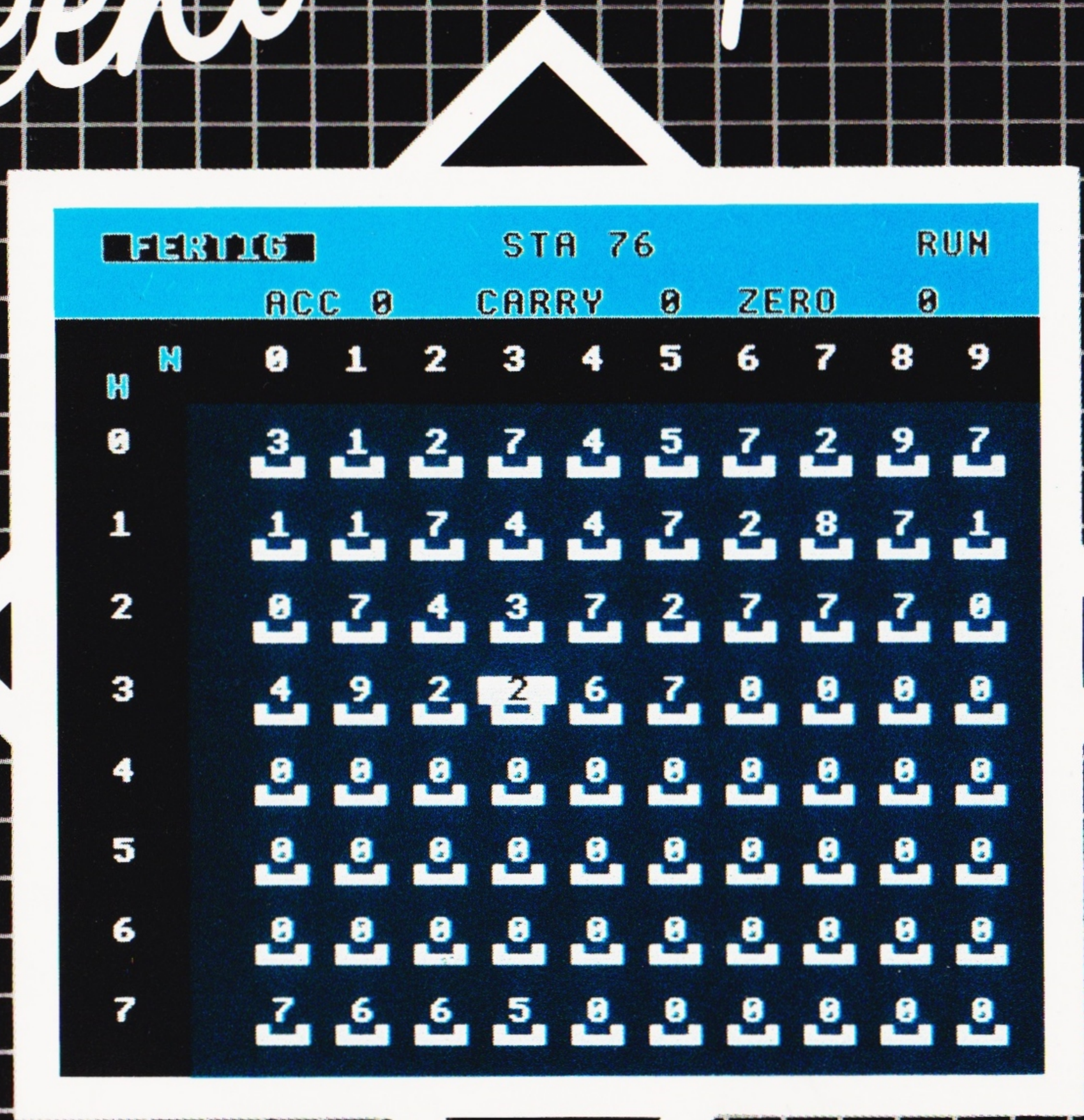
acorn electron

Peeko-Computer



Inhalt

		Seite
1	Es kann losgehen	1
	Einführung	1
	Die Beispielprogramme	1
	Das Peeko-Computer-Programm laden	2
	Was Sie auf dem Bildschirm sehen	2
	Informationen eintragen	3
	Den Speicher leeren	4
2	Einstellige Zahlen addieren	5
	Programmierbefehle	5
	Befehle eingeben	5
	Ein Programm eingeben	6
	Das Program überprüfen	7
	Ein Programm betreiben	8
	Fehler im Programm	9
	Ein Programm speichern	10
	Ein Programm laden	10
3	Zweistellige Zahlen	12
4	Rückwärts zählen	15
5	Vorwärts zählen	16
6	Multiplikation	18
7	Subtraktion	19
	Den Befehlsvorrat ändern	19
	Subtraktion mit zweistelligen Zahlen	20
8	Division	22

9 Zwei Datenzeilen addieren 24
LDA( ) und STA( ) 24
Datenzeilen addieren 26

10 Höchste Zahle ermitteln 28

11 Primzahlen generieren 31

12 Ausdrucken 33

Anhang A 34
Subtrahieren mit dem Peeko-Computer 34

Anhang B 37
Lösungsvorschläge zu den Übungsaufgaben 37

Anhang C 41
Tips für junge Programmierer 41

Anhang D 42
Weitere Übungsaufgaben 42

Referenzteil 43

1 Es kann losgehen

Einführung

Der Peeko-Computer von Acornsoft ist ein Computer im Computer. Er simuliert einen vereinfachten Mikroprozessor, den 6502, welcher das Herzstück Ihres Electron bildet, und er läßt sich beinahe wie der richtige 6502 programmieren.

Der Peeko-Computer hat jedoch einen ganz großen Vorteil: sein Speicher ist am Bildschirm Ihres Fernsehers oder Monitors sichtbar. Beim Eingeben und Betreiben eines Programms können Sie genau mitverfolgen, was im Innern des Speichers vor sich geht. Auf diese Weise kommen Sie sehr schnell dahinter, was es mit dem Programmieren auf sich hat; Sie sehen, wie der Computer auf Ihre Befehle reagiert, und können Programmfehler leicht beheben.

Den Kern dieses Buches bilden einige Beispielprogramme, anhand derer Sie nützliche Programmiermethoden und gleichzeitig die Befehle zur Verwendung mit dem Peeko-Computer kennenlernen. Die vollständige Liste der direkten Befehle und der Programmbefehle finden Sie am Schluß dieses Buches und ebenso auf der Referenzkarte, die Sie am besten auch während der Lektüre griffbereit neben sich legen.

Die Beispielprogramme

Ihre Peeko-Computer-Kassette enthält die folgenden Programme:

COUNT
FACTOR
COPY
ADD
PRIMES

Zum Laden und Betreiben dieser Programme erst den Peeko-Computer selbst laden und danach der Anleitung auf Seite 00. folgen.

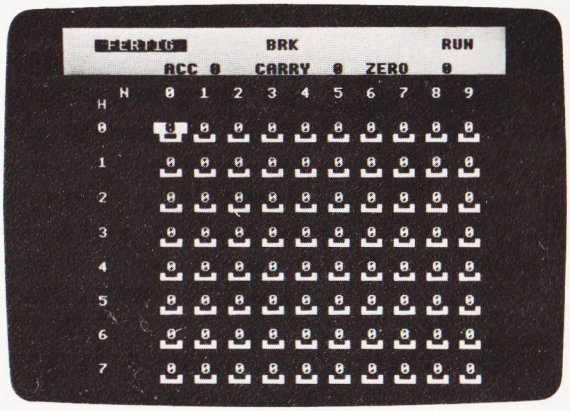
Das Peeko-Computer-Programm laden

Ihr Computer, Fernseher bzw. Monitor und das Kassettengerät sollten gemäß der Beschreibung in Kapitel 2 des *Handbuchs zum Acorn Electron* angeschlossen sein.

Legen Sie die Peeko-Computer-Kassette in das Kassettengerät und vergewissern Sie sich, daß das Magnetband voll zurückgespult ist.

Geben Sie dann den Befehl `CHAIN "PEEKO"` ein und drücken Sie RETURN. Sobald am Bildschirm die Meldung `Searching` erscheint, drücken Sie den Wiedergabeknopf (PLAY) am Kassettenrekorder.

Innerhalb weniger Minuten sehen Sie am Bildschirm eine Anzeige von der hier abgebildeten Art. Falls Ihr Kassettengerät nicht über automatische Fernsteuerung verfügt, sollten Sie es jetzt von Hand stoppen.



Was Sie auf dem Bildschirm sehen

Der größte Teil des Bildschirms wird eingenommen von einem Rechteck mit vielen "Postfächern", die mit Zahlen gefüllt sind. Das ist der Speicher des Peeko-Computers, und die einzelnen Postfächer sind die Speicherstellen (manchmal auch "Speicherplätze" oder "Speicherzellen" genannt). Jede Speicherstelle enthält ein Informationsbyte. Der Peeko-Computer hat 80 solche Speicherstellen und kann folglich bis zu 80 Bytes speichern, wobei jedes Byte eine beliebige Zahl zwischen 0 und 9 sein kann. Zu Anfang sind alle Bytes auf Null gesetzt.

(Im Speicher Ihres Electron finden etwa 64000 Bytes Platz, und jedes einzelne kann eine Zahl, einen Buchstaben oder ein Sonderzeichen – etwa ***** oder **$** – aufnehmen.)

Alle Postfächer im Peeko-Computer haben auch eine *Adresse* in Form einer zweistelligen Nummer zwischen 00 und 79. Die letzte dieser beiden Zahlen (die Einerstelle) ist die niederwertige Adressenziffer. Am Bildschirm werden diese Zahlen oberhalb des Speichers in der mit "N" markierten Zahlenreihe aufgeführt. Die erste der beiden Zahlen (die Zehnerstelle) ist die höherwertige Adressenziffer. Das ist die linke senkrechte Spalte, gekennzeichnet mit "H". Die Adresse einer Speicherstelle setzt sich zusammen aus erstens der H-Ziffer der betreffenden Reihe und zweitens der N-Ziffer, die zu der Spalte gehört.

Das helle Rechteck mit dem schwarzen Byte (Umkehranzeige) ist der Cursor, der Ihnen die momentane Speicherposition anzeigt. Das ist der Speicherplatz, auf den das nächste über die Tastatur hereinkommende Byte eingetragen wird, bzw. die Stelle, die der Computer beim Ausführen eines Programms erreicht hat. In unserer Abbildung befindet sich der Cursor auf Speicherstelle 00.

Direkt über der N-Reihe befindet sich ein Register – **ACC** (das steht für "Akkumulator"), und dahinter die beiden "Flags" **CARRY** und **ZERO**. Diese werden bei der Programmausführung gebraucht, und wir kommen später ausführlich darauf zurück.

In den beiden oberen Ecken des Bildschirms stehen die Signale **FERTIG** und **RUN**. Direkt nach dem Ladevorgang und wenn immer der Computer auf Befehle wartet, erscheint **FERTIG** in Umkehranzeige, d.h. helle Schrift auf dunklem Grund. **RUN** erscheint in Umkehranzeige, solange ein Peeko-Programm betrieben wird.

Und zu guter Letzt noch die Gruppe von drei Buchstaben in der Mitte der obersten Zeile (in der Abbildung **BRK**). Das sind die sogenannten mnemotechnischen Abkürzungen für die Peeko-Programmbefehle – kurz "Mnemoniks" genannt. Sie werden diese Befehle im Verlauf der einzelnen Lernschritte kennenlernen.

Information eintragen

Übung Nr. 1

Geben Sie dem Peeko-Computer ein paar Zahlen über die Tastatur ein, und beobachten Sie dabei, wie die betreffende Zahl jeweils in die aktuelle Speicherstelle eingetragen wird, und wie der Cursor gleich auf die benachbarte Position springt. Die Mehrzahl der nicht-numerischen Tasten wird vom Peeko-Computer nicht zur Kenntnis genommen – mit Ausnahme von einigen wenigen, die er als direkte Befehle akzeptiert. Davon später mehr. Falls Sie versehentlich eine solche Befehls-Taste drücken, machen Sie den Befehl mit ESCAPE wieder rückgängig und machen unbeirrt weiter.

VORSICHT: Durch Drücken der P-Taste veranlassen Sie den Peeko-Computer zum Ausdrucken des Speicherinhalts. Wenn kein Drucker angeschlossen ist, kann dies zu einem "Schwebezustand" des Programms führen, aus dem Sie sich jedoch mit ESCAPE wieder befreien können.

Der Cursor läßt sich auch mit den vier Pfeiltasten rechts an der Tastatur steuern.

Den Speicher leeren

Es ist nicht notwendig, den Speicher des Peeko-Computers vor Eingabe eines neuen Programms zu leeren. Wenn Sie dem Computer sagen, wo ein Programm endet, kümmert er sich nicht um den restlichen Speicherinhalt.

Dennoch kann es störend sein, überflüssige Informationen im Speicher zu haben, und zu diesem Zweck sind zwei Methoden zum Löschen des Speichers vorgesehen.

Die erste Methode besteht darin, die 0-Taste so lange gedrückt zu halten, bis alle Speicherstellen auf 0 gesetzt sind.

Die andere Methode macht sich die Tatsache zunutze, daß im Electron neun Bereiche zum Abspeichern Ihrer Peeko-Programme reserviert sind. Beim Einschalten des Peeko-Computers enthält jeder dieser Bereiche ein "Blanko-Programm" (ein leeres Programm). Den Speicher können Sie durch Laden eines solchen Blanko-Programms löschen: im `FERTIG` Zustand (Fertig-Modus) drücken Sie L (für Laden) und danach eine Zahl zwischen 1 und 9 (entsprechend des gewünschten Speicherbereichs).

(Da zu Beginn alle Speicherbereiche leer sind, spielt es keine Rolle, welche Nummer Sie wählen. Es empfiehlt sich jedoch, stets einen Speicherbereich für Löschvorgänge leer zu halten – z.B. Bereich Nr. 9.)

Übung Nr. 2

Löschen Sie den Speicher des Peeko-Computers mit Hilfe dieser beiden Methoden.

2 Einstellige Zahlen addieren

Dieses Kapitel beschreibt, wie man mit dem Peeko-Computer zwei einstellige Zahlen addiert, und wie Programme aufgebaut und betrieben werden.

Programmierbefehle

Unter einem Programm versteht man eine zusammengehörende Gruppe von Befehlen, die den Computer in die Lage versetzt, eine bestimmte Aufgabe zu lösen. Je nach Programmiersprache sind die Befehle verschieden; beim Peeko-Computer haben wir es mit zwanzig der häufigsten Befehle zum Programmieren des 6502 Mikroprozessors zu tun.

Für die Addition brauchen wir die folgenden Befehle:

```
CLC
LDA
ADC
BRK
```

Wie Sie sehen, bestehen die Befehle jeweils aus drei Buchstaben, einer Kurzform des (englischen) Befehls. Die ausgeschriebenen Befehle und eine deutsche Entsprechung finden Sie im Referenzteil und auf der Referenzkarte. Am besten beobachten Sie einfach, wie die Befehle innerhalb des Programms funktionieren.

Befehle eingeben

Wie bereits erwähnt, können die Bytes im Peeko-Computer Zahlenwerte zwischen 0 und 9 aufnehmen, nicht jedoch Buchstaben. Mit anderen Worten: alphabetische Befehle können nicht direkt eingegeben, sondern müssen vorerst in Form von einstelligen Zahlen verschlüsselt werden. Das nennt man den Code, und das ist die Form, in der Sie Ihre Befehle eingeben. Der Peeko-Computer wandelt den Code dann in die entsprechenden Befehle um und führt diese aus. Im Referenzteil finden Sie

für die in unserem ersten Programm benötigten Befehle die folgenden Zahlencodes:

Mnemonik	*Code*
CLC	3
LDA	1
ADC	4
BRK	Ø

Dabei ist Ihnen vielleicht auch aufgefallen, daß sich manche Befehle auf bestimmte Speicherstellen beziehen. Folglich müssen Sie dem Peeko-Computer Bescheid sagen, welche Speicherstelle zu benutzen ist. Dies geschieht mit Hilfe einer zweistelligen Adresse. Eine Besonderheit des 6502 Mikroprozessors (und daher auch des Peeko-Computers) ist, daß bei Adressen innerhalb von Programmen immer das niederwertige Byte als erstes anzugeben ist.

Der Code für den Befehl

```
LDA 7Ø
```

d.h. "Lade den Akkumulator mit dem Wert auf Speicherplatz 70", lautet:

`1Ø7` (und nicht etwa: `17Ø`)

Ein Programm eingeben

Peeko-Programme beginnen stets auf Speicherplatz 00. Bei der Ausführung eines Programms fängt der Computer im allgemeinen bei 00 an und schreitet dann vorwärts auf 01, 02, 03 usw., bis er auf eine `BRK` Anweisung stößt.

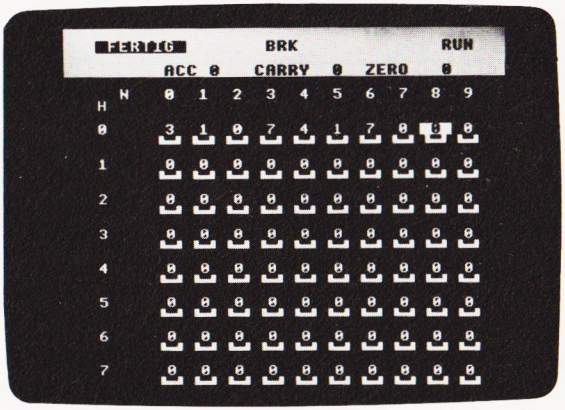
Zur Eingabe eines Programms müssen Sie den Cursor auf Speicherplatz 00 zurückholen, entweder mit Hilfe der Pfeiltasten oder durch Drücken der O-Taste ("O" steht für englisch "origin", d.h. "Ursprung").

Anschließend geben Sie Ihr Programm ein, indem Sie einen Code nach dem anderen schreiben. Der vollständige Code zum Addieren von zwei Zahlen lautet:

Speicherplatz	*Code*	*Mnemonik*
ØØ	3	CLC
Ø1	1Ø7	LDA 7Ø
Ø4	417	ADC 71
Ø7	Ø	BRK

(Die Speicherplätze haben wir hier nur angegeben, damit Sie leichter die Vollständigkeit Ihrer Eingabe überprüfen können. Sie sind nicht etwa abzuschreiben.)

Wenn Sie diese Eingabe vorgenommen haben, vergleichen Sie Ihren Peeko-Speicher mit der Abbildung:



Haben Sie bemerkt, wie jeweils bei Eintragung des Codes der entsprechende Kurzbfehl in der Mitte der obersten Bildschirmzeile ausgegeben wurde? Während Sie eine Befehlseingabe vornehmen, signalisiert der Peeko-Computer, daß weitere Angaben erwartet werden. Angenommen, Sie geben den Code 1 ein, dann erscheint auf der obersten Zeile

```
LDA ■■
```

zum Zeichen, daß der Computer Ihren Code als Ladebefehl erkannt hat und jetzt noch auf die Angabe der Ladeadresse wartet. Nach Eingabe der niederwertigen Ziffer 0 reagiert der Computer mit

```
LDA ■Ø
```

was bedeutet, daß er jetzt noch die höherwertige Ziffer erwartet.

Beachten Sie, daß die Adressenbytes bei den Kurzbefehlen in der "normalen" Reihenfolge angezeigt werden, also erst die Zehner, dann die Einer.

Das Programm überprüfen

Nach Eingabe eines Programms sollten Sie es stets auf seine Richtigkeit überprüfen.

Zu diesem Zweck verlagern Sie den Cursor wieder auf Platz 00 (am einfachsten durch Drücken von "O"). In der obersten Zeile erscheint der erste Befehl Ihres Programms, in unserem Beispiel der Befehl `CLC`.

Drücken Sie dann die N-Taste (N steht für "nächster Befehl"). Dabei springt der Cursor auf den Anfang des nächsten Befehls und gleichzeitig wird oben am Bildschirmrand die entsprechende Abkürzung angezeigt.

Vergleichen Sie die Mnemoniks sorgfältig mit der Vorlage in diesem Buch. Falls sich ein Fehler eingeschlichen hat, korrigieren Sie den Befehl durch nochmalige Eingabe.

(Zur Beachtung: Statt "N" zu verwenden, können Sie den Cursor auch mit Hilfe der Pfeiltasten bewegen. Dabei springt er jedoch nicht auf den Anfang des nächsten Befehls, sondern einfach auf die nächste Speicherstelle, die nicht unbedingt der Anfang eines neuen Programmbefehls ist. Das ist jedoch für den Peeko-Computer nicht erkennbar, und er gibt ein unsinniges Mnemonik aus, welches nicht beachtet werden muß. Bei dieser Methode gelten daher immer nur die Mnemoniks, die angezeigt werden, wenn der Cursor auf der Anfangsposition eines Befehls sitzt.)

Ein Programm betreiben

Das Programm, das Sie vorhin eingegeben haben, addiert die beiden Zahlen, welche es auf den Speicherplätzen 70 und 71 vorfindet. Bevor Sie das Programm betreiben, geben Sie die beiden Zahlen ein, die summiert werden sollen. Beginnen wir mit einer Addition, deren Resultat kleiner als 10 ist, also beispielsweise 3 auf Platz 70 und 4 auf Platz 71.

Zum Starten des Programms die Leertaste drücken. Dabei passieren zwei Dinge: der Cursor springt auf Platz 00, und in der obersten Zeile am Bildschirm ist jetzt nicht mehr das Signal `FERTIG` umgekehrt angezeigt, sondern `RUN` – zum Zeichen, daß das Programm am "Laufen" ist.

Jedesmal, wenn Sie die Leertaste drücken, springt der Cursor auf den Anfang des nächsten Befehls, und gleichzeitig wird in der Mitte der obersten Zeile der jeweilige Befehls-Kürzel angezeigt. Die Leertaste erfüllt also hier die gleiche Funktion wie die N-Taste, nur daß jetzt der Befehl nicht nur angezeigt, sondern auch *ausgeführt* wird, nämlich so:

Befehl	*Wirkung*
`CLC`	Das `CARRY` wird auf 0 gesetzt (wenn es bereits 0 war, ist keine Wirkung sichtbar)
`LDA 70`	Der Wert aus Speicherplatz 70 wird in den Akkumulator übertragen
`ADC 71`	Der Wert auf Speicherplatz 71 wird zum Wert im Akkumulator addiert

BRK Das Programm wird angehalten. Dabei wird wiederum das Signal **FERTIG** in Umkehranzeige dargestellt, zum Zeichen, daß der Peeko-Computer wieder empfangsbereit für neue Befehle ist.

Gehen Sie das Programm mit der Leertaste Schritt für Schritt durch und vergewissern Sie sich, daß der Peeko-Computer die einzelnen Befehle vorschriftsgemäß ausführt.

Als nächstes betreiben Sie das Programm mit zwei Zahlen, die zusammen mehr als 10 ergeben, sagen wir 6 und 7. Achten Sie dabei auf zwei Dinge:

1. Der **LDA** Befehl *ersetzt* jeden bereits im Akkumulator befindlichen Wert durch den neuen.
2. Nach Ausführung des **ADC** Befehls plaziert der Peeko-Computer die Einer des Ergebnisses in den Akkumulator und eine 1 in das **CARRY**, genau wie Sie das bei Additionsaufgaben tun, wenn Sie sie von Hand ausrechnen:

```
    6
  + 7
  ---
    3
merke 1
```

(Übertrag = carry)

Wir schlagen vor, daß Sie mit diesem Programm noch einige andere Additionen durchführen. Statt jedesmal jeden einzelnen Befehl schrittweise auszuführen, können Sie einfach den Go-Befehl verwenden: drücken Sie im FERTIG-Modus die G-Taste. Der Go-Befehl veranlaßt die automatische Ausführung eines Programms, beginnend mit dem Befehl auf Speicherplatz 00 bis zur Entdeckung eines **BRK** Befehls. Für lange Programme gibt es außerdem den F-Befehl zur beschleunigten Ausführung (F steht für "fast" = "schnell").

Fehler im Programm

Befehle, die einen Speicherplatz mit einer höheren Adresse als 79 enthalten, versteht der Peeko-Computer nicht. Dazu gehören z.B. Befehle wie **158**, was soviel heißt wie **LDA 85** (Lade den Akku mit dem Wert auf Platz 85).

Beim vergeblichen Versuch, einen solchen Befehl auszuführen, gibt der Peeko-Computer ein akustisches Warnsignal aus, und anstelle der Meldung **FERTIG** erscheint kurzfristig **FEHLER**. Danach springt der Cursor an den Anfang des fehlerhaften Befehls, damit Sie die Korrektur vornehmen können.

Ein Programm speichern

Wenn ein Programm zu Ihrer Zufriedenheit funktioniert und Sie es vielleicht zu einem späteren Zeitpunkt wieder verwenden wollen, können Sie es abspeichern.

Im FERTIG-Modus (`FERTIG` in Umkehranzeige) drücken Sie die S-Taste (für "Speichern") und danach eine Zahlentaste zwischen 0 und 9 zur Angabe des Speicherbereichs.

Tasten zwischen 1 und 9 bewirken, daß das Programm in einem der neun speziell für den Peeko-Computer reservierten Bereiche gespeichert wird. Das Programm kann dann jederzeit wieder durch Drücken von L, zusammen mit der entsprechenden Zahl, hervorgeholt werden. Auf diese Weise gespeicherte Programme gehen jedoch verloren, wenn Sie den Electron ausschalten, und auch nach Drücken von BREAK.

Um ein Programm auf Kassette abzuspeichern, drücken Sie die Taste 0.

Schreiben Sie den Namen, den Sie der Programmdatei geben wollen, und drücken Sie RETURN. Der Peeko-Computer gibt die Aufforderung `RECORD then RETURN` aus.

Vergewissern Sie sich, daß das Magnetband sich an einer noch nicht überspielten Stelle befindet, und notieren Sie sich den Zählerstand. Drücken Sie dann die Aufnahmetaste "RECORD" und drücken Sie RETURN auf der Electron-Tastatur.

(Eine ausführlichere Beschreibung zum Speichern von Dateien auf Kassette finden Sie im *Handbuch zum Acorn Electron* , Kapitel 9.)

Nach Abschluß des Speichervorgangs wird in der obersten Zeile am Bildschirm wiederum das Mnemonik angezeigt, welches der momentanen Cursorposition entspricht.

Ein Programm laden

Ein auf Kassette gespeichertes Peeko-Programm kann jederzeit wieder in den Computer geladen werden: drücken Sie L, gefolgt von 0, worauf der Dateiname angefragt wird. Geben Sie genau den Namen ein, den Sie beim Speichervorgang verwendet hatten, und drücken Sie RETURN. Angenommen, Sie möchten das Beispielprogramm namens `COUNT` laden, dann schreiben Sie den Befehl `LØ` und beantworten die Rückfrage nach dem Namen mit `COUNT`.

Sobald am Bildschirm die Meldung `Searching` erscheint, drücken Sie die Wiedergabetaste "PLAY" an Ihrem Kassettengerät (vorher sicherstellen, daß das Magnetband genügend weit zurückgespult ist). Das

Wiedererscheinen des Mnemoniks, welches der Cursorposition entspricht, zeigt an, daß der Ladevorgang abgeschlossen ist.

3 Zweistellige Zahlen addieren

Das einfache Programm aus dem letzten Kapitel kann nur einstellige Zahlen addieren, und das Ablesen des Ergebnisses ist einigermaßen umständlich, weil es sich teils im Akkumulator, teils im **CARRY** befindet.

Unser nächstes Beispiel ist eine Erweiterung dieses Programms, welche Additionen von Zahlen bis 99 ermöglicht. Außerdem soll gezeigt werden, wie die Ergebnisse zurück in den Peeko-Speicher übertragen werden können. Dazu führen wir zwei neue Befehle ein: **STA** und **LDA#**.

(Verwechseln Sie auf keinen Fall die beiden Befehle **LDA** und **LDA#**. **LDA** wird gefolgt von zwei Ziffern, welche eine Speicheradresse bezeichnen. Der Wert, der sich auf diesem Speicherplatz befindet, wird in den Akku plaziert. Anders bei **LDA#**, welches nur eine einstellige Ziffer nach sich zieht, welche als solche (d.h. direkt) in den Akku übertragen wird.)

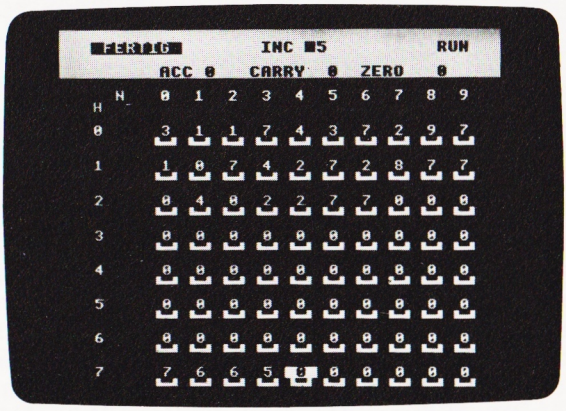
Das vollständige Programm hat diese Form:

Speicherplatz	*Code*	*Kurzbefehl*
00	3	CLC
01	117	LDA 71
04	437	ADC 73
07	297	STA 79
10	107	LDA 70
13	427	ADC 72
16	287	STA 78
19	70	LDA# 0
21	402	ADC 20
24	277	STA 77
27	0	BRK

Tragen Sie die codierten Befehle, beginnend bei Speicherplatz 00, ein und überprüfen Sie anschließend das ganze Programm mit Hilfe der N-Taste.

Dieses Programm addiert die Zahlen, die sich auf den Speicherplätzen 70–71 und 72–73 befinden, und plaziert die Summe auf den Plätzen 77–79.

Tragen Sie zwei Summanden ein, beispielsweise 76 auf Platz 70–71 und 65 auf 72–73. Der Speicher Ihres Peeko-Computers sieht dann so aus:



Ehe Sie das Programm betreiben, überlegen Sie sich, wie Sie eine derartige Addition von Hand lösen würden:

1.	76	2.	76	3.	76
	+65		+65		+65
	——		1		41
merke		merke	1	merke	1

Der Peeko-Computer geht ganz genau gleich vor, was Sie leicht nachvollziehen können, wenn Sie das Programm Schritt für Schritt mit der Leertaste verfolgen:

Befehl	*Wirkung*
CLC	**CARRY** wird auf 0 gesetzt.
LDA 71	Die Einerstelle des ersten Summanden wird in den Akku plaziert.
ADC 73	Die Einerstelle des zweiten Summanden wird in den Akku plaziert.
STA 79	Die Einerstelle des Resultats wird auf den vorgesehenen Speicherplatz übertragen.
LDA 70	Die Zehnerstelle des ersten Summanden wird in den Akku plaziert.
ADC 72	Die Zehnerstelle des zweiten Summanden wird in den Akku plaziert.

STA 78	Die Zehnerstelle des Resultats wird in den Peeko-Speicher übertragen (die Hunderterstelle ist im **CARRY**).
LDA #Ø	Der Akkumulator wird auf Null gesetzt.
ADC 2Ø	Die Null auf Platz 20 wird zusammen mit der Eins im **CARRY** zum Akku addiert. Dies bewirkt den Übertrag des Hunderters in den Akkumulator.
STA 77	Die Hunderterstelle des Resultats wird auf die vorgesehene Speicherstelle plaziert.
BRK	Das Programm hält an.

Zu beachten ist hierbei, daß es keinen Befehl für den 6502 Mikroprozessor (oder für den Peeko-Computer) gibt, der einen direkten Übertrag der Ziffer im **CARRY** auf den Speicher gestattet. Aus diesem Grund ist es erforderlich, den Akkumulator auf Null zu setzen (mit dem Befehl **LDA# Ø**), dann zu dem Carry **ADC** = "add with carry", d.h. addiere mit dem Carry) eine weitere Null zu addieren, um das Carry in den Akku zu plazieren. Danach wird der Akku gespeichert (mit **STA**). Im **ADC** Befehl hätte statt des Speicherplatzes 20 jeder andere Speicherplatz benutzt werden können, der ebenfalls eine Null enthielt.

Führen Sie noch einige Additionen mit Hilfe dieses Programms aus.

Übung 3

Erweitern Sie das Programm so, daß es auch dreistellige Summanden addieren kann, z.B. 304 und 817.

Übung 4

Schreiben Sie ein Peeko-Programm, welches drei 2-stellige Zahlen addiert, beispielsweise 76 + 65 + 89. Beachten Sie, daß der Peeko-Computer nur Überträge von 1 erlaubt. Sie müssen also das erste Summandenpaar addieren, das Resultat im Speicher ablegen und dann den dritten Summanden dazu addieren.)

Wenn Sie sich Lösungen ausgedacht und Ihre Programme geschrieben und getestet haben, können Sie sie mit den Lösungsmöglichkeiten im Anhang B vergleichen.

4 Rückwärts zählen

Unser nächstes Programm zählt rückwärts von 9 bis 0 und verwendet zu diesem Zweck zwei neue Befehle, nämlich **DEC** und **JMP**.

Und hier das Programm:

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	**507**	**DEC 70**	Subtrahiert 1 vom Wert auf Platz 70
03	**800**	**JMP 00**	Rücksprung auf Platz 00.

Geben Sie die Befehlscodes ein und den Anfangswert 9 auf Speicherplatz 70. Betreiben Sie dann das Programm mit Hilfe der Leertaste. Durch Festhalten der Leertaste läuft das Programm kontinuierlich. Wenn der Platz 70 den Nullstand erreicht hat, bewirkt die nächste Substraktion von 1 das Rücksetzen auf 9.

Da in diesem Programm kein **BRK** Befehl vorgesehen ist, bleibt **RUN** aktiv (d.h. auf Umkehranzeige), selbst wenn Sie aufhören, die Leertaste zu betätigen. Auch das Drücken beliebiger anderer Tasten bewirkt, daß das Programm weiterläuft. Gestoppt werden kann es nur mit E (für ESCAPE). E ist der einzige Peeko-Befehl, der funktioniert, während ein Programm läuft.

(Dieses Programm kann auch mit G kontinuierlich betrieben werden. Zum Anhalten benötigen Sie ebenfalls den E-Befehl.)

5 Vorwärts zählen

Und nun ein Programm zum Vorwärtszählen, von 00 bis 99. Dazu benötigen wir wiederum zwei neue Befehle, diesmal **INC** und **JNE**.

Auf den ersten Blick erscheint der **JNE** Befehl etwas verwirrend, doch anhand eines praktischen Beispiels werden Sie ihn schnell verstehen.

Der jeweilige Zählerstand ist auf Speicherplatz 70–71 enthalten. Hier das vollständige Programm:

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	617	INC 71	Addiere 1 zur Einerstelle
03	900	JNE 00	Wenn die Einerstelle 0 erreicht hat, zum nächsten Befehl übergehen; andernfalls zurück auf den Befehl im Speicherplatz 00.
06	607	INC 70	Addiere 1 zur Zehnerstelle.
09	900	JNE 00	Wenn die Zehnerstelle 0 erreicht hat, zum nächsten Befehl übergehen; andernfalls zurück auf den Befehl im Speicherplatz 00.
12	0	BRK	

Die codierten Befehle eintragen und sorgfältig überprüfen. Anschließend die Speicherplätze 70 und 71 auf Null setzen und das Programm mit G betreiben und seine Funktion testen. Das Programm läuft so lange weiter, bis auf den Plätzen 70 und 71 der Wert 00 erscheint.

Betreiben Sie anschließend das Programm nochmals, jetzt mit der Leertaste, damit Sie genau verfolgen können, was für Wirkungen die einzelnen Befehle haben. Achten Sie besonders darauf, was passiert, wenn die Einerstelle bei 9 anlangt. Der folgende **INC 71** Befehl setzt die Einerstelle auf 0, und das **ZERO** Flag wechselt auf 1. Bei dieser

Gelegenheit bleibt der **JNE ØØ** Befehl unberücksichtigt, und das Programm geht zu **INC 7Ø** über, womit die Zehnerstelle um 1 erhöht und das **ZERO** Flag auf 0 zurückgesetzt wird.

Das **ZERO** Flag des Peeko-Computers bezieht sich auf das zuletzt im Programm behandelte Resultat. Betrug dieses Null, steht das **ZERO** Flag auf 1; andernfalls, d.h. wenn das Resultat nicht Null war, steht das Flag auf 0. Die Bedeutung des **JNE** Befehls kann auch so formuliert werden:

Wenn das letzte Resultat nicht Null war, ist ein Sprung auf den nächsten Speicherplatz nach diesem Befehl auszuführen; wenn das letzte Resultat Null war, ist zum nächsten Befehl überzugehen.

(Ihre Peeko-Computer-Kassette enthält ein Programm namens COUNT, welches rückwärts von 99 auf 10 zählt und dann auf Platz 00 springt. Laden Sie dieses Programm gemäs der Anleitung am Ende von Kapitel 2 und betreiben Sie es für einen Vergleich mit diesem Zählprogramm.)

6 Multiplikation

Übung 5

Denken Sie sich selbst ein Multiplikationsprogramm aus, welches zwei einstellige Zahlen (außer 0) durch wiederholte Addition mit Hilfe der beiden Befehle **DEC** und **JNE** multipliziert. Gehen Sie davon aus, daß die beiden Faktoren auf den Speicherstellen 70 und 72 und ihr Produkt auf den Plätzen 78–79 eingetragen werden.

Ihre Lösung können Sie nach Betreiben und Testen des Programms mit dem Lösungsvorschlag in Anhang B vergleichen.

7 Subtraktion

Für Subtraktionsaufgaben sind zwei neue Befehle erforderlich: `SEC` und `SBC`. Sie gehören zum "alternativen Befehlsvorrat" des Peeko-Computers.

Der Peeko-Computer verfügt über 20 verschiedene Programmbefehle in Maschinensprache. Da jedoch jeder Befehl mit einer einzigen Code-Zahl ausgedrückt werden muß, können gleichzeitig immer nur zehn verschiedene Befehle eingesetzt werden (0-9).

Der Standard-Befehlsvorrat besteht aus den zehn Befehlen, die Sie in den 6 ersten Kapiteln kennengelernt haben. Sie werden im Referenzteil von 0 bis 9 numeriert und sind beim Einschalten des Peeko-Computers mit diesen Nummern codiert.

Um einen Befehl aus dem alternativen Befehlsvorrat zu gebrauchen, muß diesem erst eine Nummer zwischen 0 und 9 zugewiesen werden. Dies geht wie folgt vor sich:

Den Befehlsvorrat ändern

Im FERTIG-Modus die I-Taste drücken.

Anstelle der üblichen Bildschirmanzeige tritt eine Liste der momentan aktivierten Befehle, zusammen mit ihren Codenummern. Die Frage `Aenderung gewuenscht?` beantworten Sie mit `J` und drücken RETURN. Danach erscheint eine Anfrage, welcher Befehl geändert werden soll. Schreiben Sie `3` und drücken Sie RETURN.

Nun wird Ihnen am Bildschirm eine Liste aller verfügbaren Befehle mit Ihren Codes präsentiert. Geben Sie die Zahl `10` ein und drücken Sie RETURN. Sie sehen, daß das Kurzbefehl neben der Nr. 3 im Standardbefehlssatz durch den Befehl `SEC` ersetzt worden ist.

Wiederholen Sie diesen Vorgang und ersetzen Sie den Befehl `ADC` (Code Nr. 4) durch `SBC` (Befehl Nr. 11). Um die Änderung zu beschleunigen, können Sie die Frage `Aenderung gewünscht?` auch gleich mit `4` beantworten.

Nachdem Sie diese Änderungen vorgenommen haben, sollte Ihr derzeitiger Befehlsvorrat so aussehen:

```
0 BRK
1 LDA
2 STA
3 SEC
4 SBC
5 DEC
6 INC
7 LDA#
8 JMP
9 JNE
```

Bei Auffinden des Codes Nr. 3 führt der Peeko-Computer jetzt den `SEC` Befehl aus, und bei Nr. 4 den `SBC` Befehl.

(Die einfachste Methode, zum Standard-Befehlssatz zurückzukehren ist durch Eingabe von `S` auf die Frage `Aenderung gewuenscht?`.)

In den FERTIG-Modus des Peeko-Computers kehren Sie durch Eingabe von `N` auf die Frage `Aenderung gewuenscht?`.

Subtraktion mit zweistelligen Zahlen

Unser nächstes Programm subtrahiert den Subtrahenden in den Speicherstellen 72–73 vom Minuend in 70–71 und legt die Differenz in 78–79 ab.

Nachdem Sie die Änderungen im Befehlsvorrat vorgenommen haben:

```
3 SEC
4 SBC
```

geben Sie den folgenden Code ein:

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	3	SEC	CARRY auf 1 setzen.
01	117	LDA 71	Die Einerstellen
04	437	SBC 73	subtrahieren und das
07	297	STA 79	Resultat auf Platz 79 übertragen.
10	107	LDA 70	Die Zehnerstellen
13	427	SBC 72	subtrahieren und das
16	287	STA 78	Resultat auf Platz übertragen.
19	0	BRK	

Führen Sie einige Subtraktionsaufgaben mit diesem Programm durch und versuchen Sie dann, zu verstehen, wie es funktioniert. Um 56 von 78 abzuzählen, muß diese Eingabe vorgenommen werden:

Speicherplatz	*Daten*
70	7
71	8
72	5
73	6

Wenn der Wert im **CARRY** nach Ausführung des Programms 1 ist, enthalten die Speicherplätze 78-79 das korrekte, positive Ergebnis. Ist das **CARRY** Null, sollte ein negatives Resultat vorliegen; das endgültige Resultat errechnet sich aus der Zahl auf den Plätzen 78–89 minus 100.

Die Arbeitsweise der **SBC** Anweisung ist eine Folge der Konstruktion des 6502 Mikroprozessors und aus diesem Grund ziemlich kompliziert und kaum nachvollziehbar. Ein genaues Verständnis wird jedoch nicht vorausgesetzt, und solange Sie sich an die folgenden zwei Regeln halten, werden Sie keine Schwierigkeiten mit Peeko-Subtraktionen haben:

1. **CARRY** vor der Subtraktion mit **SEC** auf 1 setzen.

2. Sicherstellen, daß **CARRY** auch nach der Subtraktion noch auf 1 steht.

Wer sich für eine ausführlichere Beschreibung des **SEC** Befehls interessiert, sei auf Anhang A verwiesen.

8 Division

Als nächstes wenden wir uns einem Programm zu, welches einen zweistelligen Dividenden auf Speicherplatz 70–71 durch einen einstelligen Divisor auf Speicherplatz 73 teilt und das ganzzahlige Resultat (den Quotienten) auf den Plätzen 76-77 ausgibt. Ein etwaiger Rest wird auf Platz 79 abgelegt.

Eine Division als solche gibt es im Mikroprozessor nicht. Das Programm behilft sich daher mit einer wiederholten Subtraktion, bei der der Teiler solange subtrahiert wird, bis es ein `CARRY` mit Null entdeckt, welches signalisiert, daß die zuletzt aussgeführte Subtraktion ein negatives Resultat ergab. Neben den Befehlen, die Sie bereits kennen, wird hier zusätzlich der `JCC` Befehl eingesetzt (`JCC` steht für "Jump if `CARRY` clear", d.h. "Springe, wenn `CARRY` gelöscht).

Vor Eingabe des Programms sind die folgenden Änderungen am Standard-Befehlssatz vorzunehmen:

```
3 SEC
4 SBC
5 JCC
```

und danach die codierten Befehle zu schreiben:

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	70	LDA# 0	Damit werden die für das ganzzahlige Resultat und den Rest reservierten Plätze auf gesetzt.
02	267	STA 76	
05	277	STA 77	
08	297	STA 79	
11	117	LDA 71	Übertragen der Einerstelle des Zählers (Dividenden) auf Platz 79; wenn keine weiteren Subtraktionen möglich sind, ist das der Rest.
14	297	STA 79	

17	3	SEC	Subtraktion des Divisors vom
18	437	SBC 73	Dividenden, Ergebnis auf
21	217	STA 71	Plätze 70–71. Beachten Sie,
24	107	LDA 70	daß die Null auf
27	410	SBC 01	Platz 01 von der
30	207	STA 70	Zehnerstelle abgezogen wird, um die Berück-sichtigung des Übertrags zu garantieren.
33	584	JCC 48	Wenn das **CARRY** Null ist, war das Resultat der letzten Subtraktion negativ und damit ist die Rechenoperation zu Ende; Sprung auf **BRK** bei Platz 48.
36	677	INC 77	Addiere 1 zur Einerstelle
39	911	JNE 11	des Resultats, und, wenn
42	667	INC 76	dieses Null wird,
45	811	JMP 11	addiere 1 zur Zehnerstelle. Danach Rücksprung auf den Beginn der nächsten Subtraktion bei Platz 11.
48	0	BRK	

Wenn das Divisionsprogramm zu Ihrer Zufriedenheit funktioniert und Sie dahintergekommen sind, *wie* es funktioniert, schlagen wir vor, daß Sie das Beispielprogramm FACTOR von der Peeko-Kassette laden.

Dieses Programm ermittelt den höchsten Faktor unter 10 einer beliebigen 2-stelligen Zahl. Auch hier erfolgt die Division durch wiederholte Subtraktion, wobei mit einem Nenner (Divisor) von 9 angefangen wird. Wenn der Rest ungleich Null ist, wird die Operation wiederholt, diesmal mit einem Nenner von 8, usw., usw., bis der erste Faktor gefunden wird.

Vor Benutzung des Programms ist sicherzustellen, daß die Speicherstellen 76 und 77 auf Null stehen. Die Zahl, deren Faktor ermittelt werden soll, ist auf den Plätzen 78–79 einzutragen. Bei Anhalten des Programms ist der höchste Faktor auf Platz 77 zu entnehmen. Zur Ermittlung des nächsthöchsten Faktors kann das Programm erneut betrieben werden.

9 Zwei Datenzeilen addieren

Das Programm, das wir Ihnen in diesem Kapitel vorstellen, verwendet zwei neue Befehle aus dem alternativen Befehlssatz: `LDA ()` und `STA ()` Sie sind den Befehlen `LDA` und `STA` verwandt, die Sie ja bereits kennen. Die Unterschiede lassen sich am besten anhang eines Beispiels veranschaulichen.

`LDA()` und `STA()`

Nehmen Sie folgende Änderungen am Standard-Befehlssatz vor:

```
1 LDA (
2 STA (
```

und geben Sie dann die folgenden codierten Befehle ein:

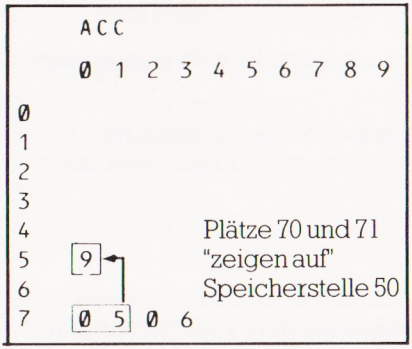
Speicherplatz	*Code*	*Kurzbefehl*
00	107	LDA (70)
03	227	STA (72)
06	0	BRK

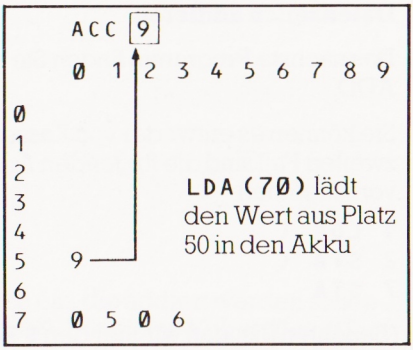
und diese Daten:

Speicherplatz	*Daten*
50	9
60	0
70	0
71	5
72	0
73	6

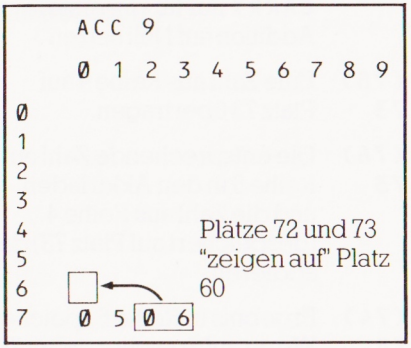
Betreiben Sie anschließend das Programm.

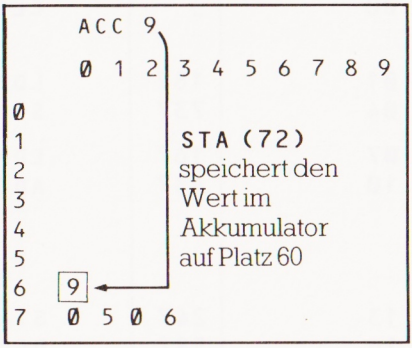
`LDA (70)` lädt den Akkumulator nicht etwa mit dem Wert auf Platz 70 (das wäre 0), sondern mit dem *Wert auf dem Speicherplatz, auf den die Speicherstellen 70 und 71 ZEIGEN.* Die Werte auf 70 und 71 sind 05 – das ergibt in der üblichen umgekehrten Notierung den Speicherplatz 50. Kurz gesagt: `LDA (70)` lädt den Akkumulator mit dem Wert auf Speicherplatz 50 (also mit 9).





Ganz ähnlich verhält es sich mit **STA (72)**: dieser Befehl *speichert den Wert im Akkumulator auf dem Speicherplatz, auf den die Plätze 72 und 73 ZEIGEN.* Die Werte auf 72-73 sind 06; folglich speichert **STA (72)** den Wert 9 auf Platz 60.





Verändern Sie die Werte auf den Speicherstellen 72 und 73 und überlegen Sie sich, was passiert, wenn Sie das Programm jetzt betreiben. Nun, war Ihre Vermutung richtig? Probieren Sie's dann einmal mit anderen Werten in 70 und 71.

Mit Hilfe der Befehle **LDA ()** und **STA ()** können Sie Rechenoperationen mit Daten aus verschiedenen Programmabschnitten durchführen, ohne das Programm neu schreiben zu müssen.

Ein etwas komplexeres Beispiel können Sie sich mit dem Demonstrationsprogramm COPY auf der Peeko-Kassette vorführen lassen. Hier wird eine ganze Reihe von Daten aus den Speicherplätzen 50-59 auf die Reihe 60–69 kopiert. Der **INC** Befehl ist zuständig für die Änderung der Speicherplätze, auf die sich die Befehle **LDA ()** und **STA ()** beziehen.

Datenzeilen addieren

Das nächste Programm finden Sie auf Ihrer Kassette unter dem Namen ADD.

Sie können es entweder von Kassette laden oder selbst eingeben. Im zweiten Fall sind die folgenden Änderungen am Standard-Befehlssatz vorzunehmen:

```
1 LDA (
2 STA (
7 STA
```

(Beachten Sie, daß wir hier den **STA** Befehl mit der Code-Nr. 7 anstatt seiner üblichen (Nr. 2) versehen haben. Jedem Peeko-Computer Befehl kann eine beliebige Code-Nummer zwischen 0 und 9 zugewiesen werden. Code zum Eingeben:

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	3	CLC	**CARRY** nur für die erste Addition auf Null setzen
01 04	187 737	LDA (78) STA 73	Eine Zahl aus Reihe 4 auf Platz 73 übertragen.
07 10	167 437	LDA (76) ADC 73	Die entsprechende Zahl auf Reihe 5 in den Akku laden und die Zahl aus Reihe 4 (gespeichert auf Platz 73) dazu addieren.
13 16 19 22	247 587 567 547	STA (74) DEC 78 DEC 76 DEC 74	Ergebnis in Reihe 6 speichern. Werte auf Speicherstelen 74, 76 und 78 so ändern, daß das Programm die nächsten Zahlen aus den Reihen 4 und 5 addiert.
25 28 31 34 37	910 187 405 247 0	JNE (01) LDA (78) ADC 50 STA (74) BRK	Wenn der Wert auf Platz 74 auf Null steht, zum Zeichen, daß alle Zahlen ddiert worden sind, ist das letzte "carry" auf Platz 60 anzuzeigen und das Programm zu stoppen. Andernfalls Sprung auf Platz 01 und Addieren der nächsten beiden Zahlen.

Eingabe der Daten:

Speicherplatz	*Daten*
74	9
75	6
76	9
77	5
78	9
79	4

Geben Sie dann die Zeilen mit den Zahlen ein, die addiert werden sollen – in die Speicherstellen 41 bis 49 und 51 bis 59. Die Plätze 40 und 50 sind leer zu lassen, damit Platz 60 das letzte "carry" anzeigen kann.

Beachten Sie, daß es für **ADC** keinen parallelen Befehl zu **LDA ( )** und **STA ( )** gibt. Im obigen Programm werden die Zahlen aus Reihe 4 erst an einen bekannten Speicherplatz (siehe Speicherplätze 01 bis 06) übertragen und dann mit dem normalen **ADC** Befehl zu der Zahl aus Reihe 5 addiert (siehe Plätze 07 bis 12).

10 Höchste Zahl ermitteln

Dieses Programm ermittelt die höchste aus einer Reihe von Zahlen und führt den **CMP#** Befehl ein. Zum leichteren Verständnis empfehlen wir als erstes das folgende Kurzprogramm.

Nehmen Sie am Standardbefehlssatz die folgenden Änderungen vor:

```
2 CMP#
6 INC A
```

und geben Sie dann diesen Code ein:

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	74	LDA# 4	Den Akku auf einen Anfangswert von 4 setzen.
02	25	CMP# 5	Den Wert im Akku mit 5 vergleichen.
04	6	INC A	Den Wert im Akku erhöhen.
05	820	JMP 02	Vergleich wiederholen.

Das Programm mit Hilfe der Leertaste betreiben und dabei nach jedem Vergleich die Werte von **CARRY** und **ZERO** überprüfen. Beobachten Sie, wie die Werte sich ändern, je nachdem, ob der Akku-Wert kleiner, gleich oder größer als der Vergleichswert ist.

Durch Überprüfen des Zustands von **CARRY** und/oder **ZERO** nach einem Vergleich ist es möglich, die relative Größe der beiden Zahlen zu ermitteln, die miteinander verglichen werden – wie dieses Programm veranschaulicht. .blank Nehmen Sie die folgenden Änderungen am Standard-Befehlssatz vor:

```
1 LDA (
4 JCC
5 JEQ
7 CMP#
```

und schreiben kopieren Sie dann den Code:

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	107	LDA (70)	Den ersten (bzw. den nächsten) zu prüfenden Wert laden.
03	562	JEQ 26	Wenn der Wert 0 ist, ans Programmende springen.
06	70	CMP# 0	Den Wert mit dem momentan höchsten Wert vergleichen (anfänglich auf 0 gesetzt).
08	441	JCC 14	Wenn der Akku-Wert höher (bzw. identisch mit) als der momentan höchste Wert ist, wird er als der neue Höchstwert anzusehen.
11	270	STA 07	
14	607	INC 70	Zeiger auf den nächsten zu prüfenden Wert setzen.
17	900	JNE 00	
20	617	INC 71	
23	800	JMP 00	
26	0	BRK	

und dazu diese Daten:

Speicherplatz	*Daten*
70	0
71	4

Überprüfen Sie die Richtigkeit des eingegebenen Programms und tragen Sie dann eine Reihe von einstelligen Zahlen ein, deren höchste Zahl sie ermitteln wollen. Die Serie muß auf Speicherplatz 40 beginnen. Das Programm testet die Zahlen in den Speicherplätzen 41, 42 und so weiter, bis es eine Null findet, und hält dann an. Die höchste Zahl wird auf Platz 07 angezeigt. (Zum Wiederbetreiben des Programms, den Platz 07 auf 0 zurücksetzen und die Plätze 70 und 71 auf 0 bzw. auf 4. .blank Beachten Sie, daß das Programm den "momentanen Höchstwert" nicht nur dann ersetzt, wenn der Wert im Akkumulator größer ist, sondern auch bei Gleichheit. Das ist zwar etwas unökonomisch, vereinfacht jedoch das Programm, insofern als nicht beide (die **CARRY** und die **ZERO**) Ziffern getestet werden müssen.

Versuchen Sie, das "Maximum"-Programm in folgender Hinsicht zu verbessern:

1. Ersetzen des "momentanen Höchstwerts" nur bei Gelegenheiten, wo der Akku-Wert diesen übersteigt.

2. Speichern des gefundenen Maximalwertes in Speicherzelle 79.

3. Das Programm so ändern, daß die Zahlenserie auch Null beinhalten kann. Zu diesem Zweck ist es erforderlich, dem Programm zu sagen, wieviele Zahlen es hat (damit es weiß, wann anhalten). Unter Umständen erfordert ein Programm, welches alle diese drei Verbesserungsvorschläge enthält, mehr als den erlaubten Vorrat von zehn Befehlen. In diesem Fall müßte entweder der Vorschlag 1 oder 2 wegfallen.

Für Lösungsvorschläge verweisen wir auf Anhang B.

11 Primzahlen generieren

Und zu guter Letzt ein Beispiel eines recht komplizierten Peeko-Computer Programms. Es dient zum Erzeugen von Primzahlen (Zahlen, die sich nur durch 1 und durch sich selbst teilen lassen) zwischen 10 und 99. .blank Das Programm testet der Reihe nach jede Zahl zwischen 10 und 99, und zwar versucht es, die betreffende Zahl nacheinander durch 2, 3 . usw. bis 9 zu teilen, wobei die Division in Form einer wiederholten Subtraktion vor sich geht. Sobald eine dieser Divisionsversuche klappt, ist die betreffende Zahl keine Primzahl, und das Programm geht zur nächsten Zahl über. Wenn keine Division aufgeht, ist es eine Primzahl und das Programm hält an.

Der Einfachheit halber können Sie sich das Programm direkt ab Kassette laden, wo es unter dem Namen PRIMES abgespeichert ist. Auf Wunsch schreiben Sie es selber: Voraussetzung sind die folgenden Änderungen am Standard-Befehlssatz:

```
3 SEC
4 SBC 5 JCS
8 JEQ
```

Codeeingabe:

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	70	LDA# 0	Speicherplatz 74 auf 0 setzen.
02	247	STA 74	
05	697	INC 79	1 zum Wert addieren, der
08	941	JNE 14	gerade auf Primzahl getestet
11	687	INC 78	wird.
14	72	LDA# 2	Auf Platz 75 die erste für die
16	257	STA 75	Division benötigte Zahl eintragen.
19	197	LDA 79	Die gerade im Test
22	277	STA 77	befindliche Zahl auf Zellen 76-
25	187	LDA 78	77 übertragen.
28	267	STA 76	

31	3	SEC 32	Den momentanen Divisor von der zu testenden Zahl subtrahieren.
32	177	LDA 77	
35	457	SBC 75	
38	277	STA 77	
41	167	LDA 76	
44	447	SBC 74	
47	267	STA 76	
50	995	JNE 59	Wenn die Division 0 ergibt, ist es keine Primzahl: auf Platz 05 zurückkehren und Zahl um 1 erhöhen.
53	177	LDA 77	
56	850	JEQ 05	
59	523	JCS 32	Wenn CARRY 1 ist, ist die Division nicht vollständig, auf Platz 32 zurückkehren und sie zu Ende bringen.
62	657	INC 75	Den Divisor um 1 erhöhen, wenn dies zu einem Null-Ergebnis führt, wurden alle Zahlen von 2 bis 9 getestet und das Programm kann gestoppt werden.
65	991	JNE 19	
68	0	BRK	

Einzugebende Daten:

Speicherplatz	*Daten*
78	0
79	9

Die Zahl, die gerade getestet wird, ist in den Speicherzellen 78-79 sichtbar. Bei Anhalten des Programms ist die dort befindliche Zahl eine Primzahl. Zur Ermittlung der nächsten Primzahl wird das Programm von neuem betrieben, ohne die betreffende Zahl zu ändern.

12 Ausdrucken

Der Speicherinhalt des Peeko-Computers kann auf Papier ausgedruckt werden, vorausgesetzt, daß Ihr Electron mit einem Drucker verbunden ist. Der Druckbefehl heißt **P** und ist im FERTIG-Modus einzugeben.

Zur Beachtung: Der Speicherinhalt wird in codierter Form ausgegeben, also genauso wie am Bildschirm. Eine Auflistung des Programms in mnemonischer Form ist nicht möglich.

Anhang A

Subtrahieren mit dem Peeko-Computer

Wie wir in Kapitel 7 gesehen haben, führt der Peeko-Computer Subtraktionen mit dem **SBC** Befehl aus.

SBC ist die Kurzform für "Subtract with **CARRY**" (subtrahiere mit CARRY), was eine Verwandtschaft zu **ADC** ("Add with **CARRY**") vermuten läßt. In Wirklichkeit sind die beiden Befehle jedoch ziemlich verschieden.

Bei Peeko *Additionen* beinhaltet das **CARRY** die Zahl, die jeweils von einer Stelle zur nächsten übertragen wird (wie wir das bei der manuellen Addition auch kennen). Anders ist es bei Peeko *Subtraktionen*: hier funktioniert das **CARRY** als ein "Flag" (eine "Signalflagge"), die dem Peeko-Computer etwas aussagt über den zuletzt ausgeführten Rechenvorgang.

Wert des CARRY flag	*Bedeutung*
1	Vorangehende Rechnungsphase: kein Übertrag
2	Vorangehende Rechnungsphase: Übertrag

Man könnte sich auch einen Mikroprozessor vorstellen, der eine eigene "Merke-Flagge" für "Überträge" besitzt. Da dies jedoch weder beim 6502 Mikroprozessor noch beim Peeko-Computer der Fall ist, muß eben das **CARRY** neben seiner normalen Additionsfunktion auch dafür herhalten.

Der Grund weshalb **CARRY** bei Beginn einer Subtraktion auf 1 gesetzt werden muß, ist also ganz einfach der, daß es keine "vorangehende Rechnungsphase" gibt und folglich auch keinen Übertrag. Dies wiederum hat zur Folge, daß bei Ausführung eines **SBC** Befehls mit **CARRY** auf 0 eine Eins vom Resultat abgezogen werden muß.

Um 154 von 226 zu subtrahieren, geht der Peeko-Computer wie folgt vor:

1. Bei Beginn der Subtraktion: kein Übertrag, also: **CARRY** auf 1 setzen.

```
   226
 − 154
 ----
carry flag   1
```

2. 6-4=2, ohne Übertrag also: **CARRY** auf 1 rücksetzen

```
   226
 − 154
     2
carry flag   1
```

3. 2 minus 5 geht nicht, bzw. 5 kann nicht auf 2 ergänzt werden, also müssen wir einen Hunderter "auflösen": 12−5 = 7. **CARRY** wird nun als "merke" auf 0 gesetzt

```
   226
 − 154
    72
carry flag   0
```

4. 2−1=1, aber der vorangehende Übertrag muß noch abgezogen werden (zur Erinnerung steht daß **CARRY** auf Null). Also wird nochmals eine Eins subtrahiert. Für diese Phase der Berechnung war kein Übertrag erforderlich – also wird das **CARRY** wieder auf 1 gesetzt. 1 ist der Endzustand des **CARRY** flag, und damit ist das Ergebnis positiv.

```
   226
 − 154
   072
carry flag   1
```

In unserem nächsten Beispiel wird beim 3. Schritt ein Hunderter aufgelöst und gleichzeitig das im 2. Schritt erforderliche "merke 1" abgezählt. Außerdem sehen Sie, was es bedeutet, wenn am Schluß der Subtraktion das **CARRY** auf 0 steht.

1. Beginn der Berechnung: **CARRY** auf 1 setzen.

```
    43
 −  59
 ----
carry flag   1
```

2. 3 minus 9 geht nicht, also muß ein Zehner aufgelöst werden. 13−9=4. Merke 1: **CARRY** wird auf 0 gesetzt.

```
    43
 −  59
     4
carry flag   0
```

3. 4 minus 5 geht nicht, also wird wieder eine 1 dazugeholt: 14−5=9, doch das **CARRY** steht auf 0 ("merke 1"): 9−1=8. Da wieder eine Eins geliehen wurde, wird **CARRY** wieder auf 0 gesetzt.

```
    43
 −  59
    84
carry flag   0
```

Wenn das **CARRY** am Schluß einer Rechenaufgabe auf 0 steht,

bedeutet das, daß ein Übertrag (ein "merke 1") nicht mehr abgerechnet werden konnte. In unserem Beispiel handelt es sich um eine Hunderterstelle; folglich ist das tatsächliche Resultat: 84−100=−16.

Mathematisch gesehen funktioniert der **SBC** Befehl wie folgt:

A=A−M(1−C)

If A>0; C=1 If A<0; A=A+10; C=0

wobei A= der Wert im Akkumulator, M= der Wert des Subtrahenden und C= der Wert des **CARRY** flag ist.

Anhang B

Lösungsvorschläge zu den Übungsaufgaben

Übung 3

Dieses Programm addiert die Zahlen in den Speicherzellen 70-72 und plaziert das Ergebnis auf 76-79.

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	3	CLC	
01	127	LDA 72	Einer addieren
04	457	ADC 75	
07	297	STA 79	
10	117	LDA 71	
13	447	ADC 74	Zehner addieren
16	287	STA 78	
19	107	LDA 70	
22	437	ADC 74	Hunderter addieren
25	277	STA 77	
28	70	LDA# 70	
30	492	ADC 29	Gegebenenfalls Tausender
33	267	STA 76	in den Speicher plazieren
36	0	BRK	

Übung 4

Dieses Programm addiert die Zahlen auf 70–71, 72–73 und 74–75 und legt das Resultat in den Speicherstellen 77–79 ab. Die Berechnung erfolgt in zwei Phasen, wobei die Ergebniszellen für das Zwischentotal verwendet werden.

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	3	CLC	
01	117	LDA 71	
04	437	ADC 73	
07	297	STA 79	
10	107	LDA 70	Addieren der ersten beiden Zahlen
13	427	ADC 72	
16	287	STA 78	
19	70	LDA# 0	
21	402	ADC 20	
24	277	STA 77	
27	157	LDA 75	
30	497	ADC 79	
33	297	STA 79	
36	147	LDA 74	Addieren der dritten Zahl zum Zwischentotal
39	487	ADC 78	
42	287	STA 78	
45	70	LDA# 0	
47	477	ADC 77	
50	277	STA 77	
53	0	BRK	

Übung 5

Dieses Programm multipliziert die Zahlen auf 70 und 72 und plaziert das Produkt auf 78–79. Zur Beachtung: das Programm funktioniert nicht, wenn der Multiplikator (Speicherplatz 72) 0 ist.

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	3	CLC	
01	127	LDA 72	Überträgt die zweite Zahl auf Platz 74
04	247	STA 74	
07	70	LDA# 0	
09	287	STA 78	Leert Plätze 78 und 79
12	297	STA 79	
15	197	LDA 79	
18	407	ADC 70	Addiert die erste der zu multiplizierenden Zahlen zum momentanen Wert auf Platz 78 und 79.
21	297	STA 79	
24	70	LDA# 0	
26	487	ADC 78	
29	287	STA 78	

32	547	DEC 74	Zählt von der zweiten Zahl 1 ab
35	951	JNE 15	Wenn die zweite Zahl 0 ist,
38	0	BRK	Programm anhalten.

Übung 6

1.

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	107	LDA (70)	
03	592	JEQ 29	
06	70	CMP# 0	
08	471	JCC 17	
11	571	JEQ 17	
14	270	STA 07	
17	607	INC 70	
20	900	JNE 00	
23	617	INC 71	
26	800	JMP 00	
29	0	BRK	

Sprung auf Rücksetzungszeiger, außer, wenn der Akku-Wert größer als der momentan höchste Wert ist.

2. Am Standardbefehlssatz diese Änderung vornehmen:

3 LDA

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	107	LDA (70)	
03	592	JEQ 29	
06	70	CMP# 0	
08	471	JCC 17	
11	571	JEQ 17	
14	270	STA 07	
17	607	INC 70	
20	900	JNE 00	
23	617	INC 71	
26	800	JMP 00	
29	370	LDA 07	Versetzt den höchsten
32	297	STA 79	Wert aus Platz 07 auf 79.
35	0	BRK	

3. Dieses Programm ist ohne die Verbesserungen 1 und 2. Nehmen Sie folgende Änderungen am Standardbefehlssatz vor:

```
1 LDA (
3 LDA
4 JCC
7 CMP#
```

Speicherplatz	*Code*	*Kurzbefehl*	*Wirkung*
00	347	LDA 74	Anzahl der zu prüfenden Eintragungen um 1 reduzieren (gespeichert auf 73 und 74).
03	921	JNE 12	
06	537	DEC 73	
09	983	JNE 38	
12	547	DEC 74	
15	107	LDA (70)	
18	70	CMP# 0	
20	462	JCC 26	
23	291	STA 19	
26	607	INC 70	
29	900	JNE 00	
32	617	INC 71	
35	800	JMP 00	
38	337	LDA 73	Testen, ob letzter Eintrag überprüft wurde.
41	79	CMP# 9	
43	921	JNE 12	
46	0	BRK	

Wie das Programm verwendet wird:
1. Die gewünschten einstelligen Zahlen eintragen, deren maximaler Wert gefunden werden soll. Beginnen auf Platz 50 (nicht 40). Es können bis zu 20 Zahlen eingegeben werden, einschließlich 0.

2. Anzahl der eingetragenen Zahlen auf 73 und 74 notieren.

3. Auf Platz 70 0 und auf Platz 71 5 eintragen.

4. Auf Platz 19 0 eintragen.

Das Ergebnis erscheint auf Speicherplatz 19.

Anhang C

Tips für junge Programmierer

Mit Hilfe des Peeko-Computers werden Sie schnell Ihren eigenen Programmierstil entwickeln. Dabei sind vielleicht die folgenden Tips nützlich:

1. Es empfiehlt sich, vor Schreiben des Programms Speicherplätze für alle erforderlichen Daten zu reservieren, d.h. sich Gedanken zu machen, wo die Anfangsdaten, wo die Resultate eingetragen werden sollen. Überlegen Sie auch, ob Zwischenrechnungen notwendig sind und ob Daten für Zählzwecke kopiert werden sollen.

2. Schreiben Sie Ihr Programm in mnemonischer Form, d.h. mit den Kurzformen der Befehle. Am besten notieren Sie sich jeweils dazu auch die Anfangsadresse (für etwaige Sprungbefehle).

3. Überprüfen Sie, ob sich unter den benutzten Befehle solche aus dem alternativen Befehlssatz befinden. Wenn ja, weisen Sie diesen Code-Nummern zu, die nicht bereits durch Standardbefehle in Ihrem Programm belegt sind.

4. Schreiben Sie jeweils hinter den Kurzbefehl den Code.

5. Geben Sie den Code sorgfältig in den Peeko-Computer ein.

6. Überprüfen Sie die Richtigkeit der Kurzbefehle am Bildschirm mit Hilfe des N-Befehls.

7. Betreiben Sie das Programm schrittweise, um sicherzugehen, daß es richtig funktioniert.

Anhang D

Weitere Übungsaufgaben

Und zum Schluß noch ein paar Anregungen für weitere Programme, die Sie für den Peeko-Computer erstellen könnten.

1. Ein Programm, welches das Minimum einer Zahlengruppe ermittelt.

2. Ein Programm, welches den Wert von n! ermittelt, d.h. n x (n-1) x (n-2) x .x1) für Werte zwischen n und 9.

3. Ein Divisionsprogramm, welches Zahlen bis 99 durch Teiler zwischen 1 und 9 teilen kann, und das Ergebnis bis zur nächsten ganzen Zahl aufrundet.

4. Ein Programm, welches den Wert der Reihe n + (n-1) + (n-2) + . + 1 für Werte bis zu 99 findet.

Referenzteil

Direkte Befehle

0 bis 9		Eingabe des entsprechenden Zahlenwerts an der momentanen Cursorposition im Speicher
Leertaste		Startet Programmdurchführung bzw. Ausführung des nächsten Befehls
E	ESCAPE	Abbruch der Programmausführung (im RUN-Modus)
F	Fast	Beschleunigte Programmausführung
G	Go	Programmausführung
I	Befehle	Änderung am Befehlsvorrat vornehmen. Aus dem Menu eine Option wählen: J – Änderung am aktuellen Befehlssatz N – Beibehalten des aktuellen Befehlssatzes S – Rückkehr zum Standard-Befehlssatz 0 – 9 Ändern des betreffenden Befehls
LO	Load	Laden eines Programms von Kassette
L1	bis 9	Laden eines Programms aus den Bereichen 1-9 aus dem Electron Arbeitsspeicher
N	Next	Anzeige des nächsten Befehls
O	Origin	Cursor auf "Ursprung" setzen: Platz 00
P	Print	Speicherinhalt des Peeko-Computers ausdrucken
SO	Save	Ein Programm auf Kassette speichern
S1 bis 9		Ein Programm in einen der reservierten Speicherbereiche des Electron (1-9) speichern.

Programmbefehle

Standard-Code	*Kurzbefehl*	*Bedeutung*
0	BRK	= BREAK: Abbruch, Stoppen des Programms
1	LDA xx	= Load Accumulator: Akkumulator laden. Den Wert auf Platz xx in den Akku laden.
2	STA xx	= Store Accumulator: Akkumulator speichern. Akku-Wert auf Speicherplatz xx plazieren.
3	CLC	= Clear Carry: Carry löschen. CARRY auf 0 setzen.
4	ADC xx	= Add with Carry: Mit Carry addieren. Wert auf Platz xx und CARRY zum Akku-Wert addieren.
5	DEC xx	= Decrease memory: Speicherwert reduzieren. Wert auf Speicherplatz xx um 1 reduzieren.
6	INC xx	= Increase memory: Speicherwert erhöhen. Wert auf Speicherplatz xx um 1 erhöhen.
7	LDA# n	= Load Accumulator immediately: Akku direkt laden. Akku-Wert durch n ersetzen.
8	JMP xx	= Jump always: unbedingt springen. Auf Speicherplatz xx springen.
9	JNE xx	= Jump if ZERO not equal to one: springen, wenn ZERO ungleich 1. Bedingter Sprung: wenn ZERO = 0, auf Platz xx springen; andernfalls zum nächsten Befehl.
10	SEC	= Set CARRY: CARRY setzen. CARRY flag auf 1.
11	SBC xx	= Subtract with CARRY: Subtraktion mit CARRY. Wert auf Platz xx vom Akku-Wert subtrahieren.
12	CMP# n	= Compare immediate: Sofortvergleich. Akku-Wert mit n vergleichen. Wenn: Akku<n – CARRY auf 0, ZERO auf 0 setzen Akku=n – CARRY auf 1, ZERO auf 1 setzen Akku>n – CARRY auf 1, ZERO auf 0 setzen
13	JCC xx	= Jump if CARRY clear: Springe, wenn CARRY gelöscht. Bedingter Sprung: Wenn CARRY 0 ist, erfolgt Sprung auf Platz xx; andernfalls Übergang zum nächsten Befehl.
14	JCS xx	= Jump if CARRY set: Springe, wenn CARRY gesetzt. Bedingter Sprung: Wenn CARRY 1 ist, erfolgt Sprung auf Platz xx; andernfalls Übergang zum nächsten Befehl.

15	JEQ xx	= Jump if ZERO equal to one: Springe, wenn ZERO gleich 1 ist. Bedingter Sprung: Wenn die ZERO flag 1 ist, erfolgt Sprung auf Platz xx; andernfalls Übergang zum nächsten Befehl.
16	DEC A	= Decrease Accumulator: Akku reduzieren. Akku-Wert um 1 reduzieren.
17	INC A	= Increase Accumulator: Akku erhöhen. Akku-Wert um 1 erhöhen.
18	LDA (xx)	= Load Accumulator indirectly: Akku indirekt laden. Akku-Wert durch den Wert ersetzen, der sich im Speicherplatz befindet, auf den xx und xx+1 zeigen.
19	STA (xx)	= Store Accumulator indirectly: Akku indirekt speichern. Akku-Wert an den Speicherplatz setzen, auf den xx und xx+1 zeigen.

ACORNSOFT

Acornsoft Limited, Betjeman House, 104 Hills Road,
Cambridge CB2 1LQ, England. Telephone (0223) 316039

SLC02/B