Arbeiten mit dem Stack

Der Stack ist einer der wichtigsten Konzepte von FORTH. Wir können uns den **Stack** vorstellen als einen Stapel von Zahlen. In der Tat heißt das englische Wort "stack" auf deutsch nichts anderes als **Stapel**. Wozu dient nun der Stack und wie wird er praktisch eingesetzt? Das soll in diesem Kapitel erklärt werden.

Schauen wir uns zunächst das Wort `stapeln` an:

```
: stapeln  11  22  33  44  55  ;
```

Wird dieses Wort ausgeführt, so werden die Zahlen `11`, `22`, `33`, `44` und `55` der Reihe nach auf den Stack gelegt. Anschaulich können wir uns das so vorstellen:

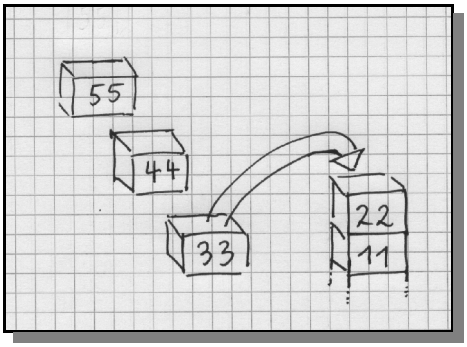


Abb. 1

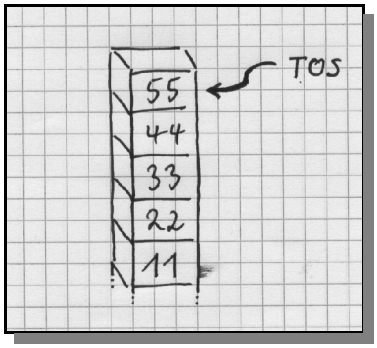


Abb. 2

Am Ende liegen unsere fünf Zahlen übereinander, die 11 zuunterst, die 55 ganz oben. Die oberste Zahl wird auch **TOS** (= Top Of Stack) genannt.

Wörter wie z. B. "." und `wait` greifen auf diesen Stapel zu. Das Wort "." holt sich z. B. den TOS vom Stapel und gibt diese Zahl auf dem Port B aus; das Wort `wait` greift sich auch den TOS und wartet entsprechend viele Sekunden. Wird das folgende Wort

```
: ausgabe . wait . ;
```

nach dem Wort `stapeln` ausgeführt, geschieht folgendes:

"." holt die Zahl 55 vom Stack und gibt sie auf Port B aus.

`wait` holt die Zahl 44 vom Stack und wartet entsprechend viele Sekunden.

"." holt die Zahl 33 vom Stack und gibt sie auf Port B aus.

Am Schluss befinden sich nur noch die Zahlen 11 und 22 auf dem Stack.

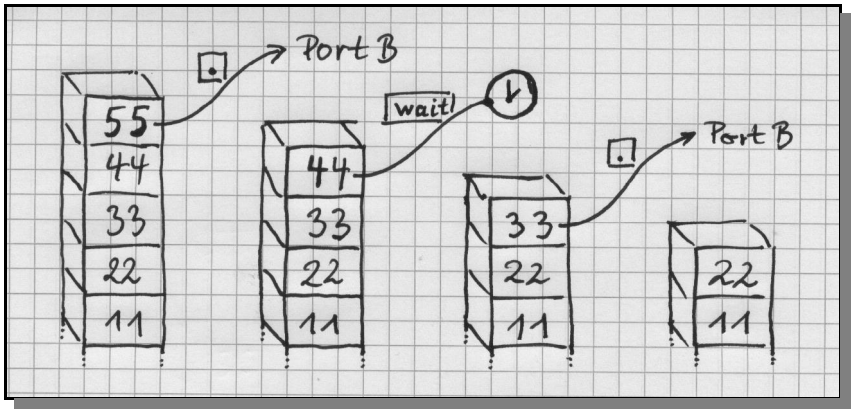


Abb. 3

Wörter verändern i. A. den Stack: Unser Wort `stapeln` legt 5 Zahlen auf den Stack, unser Wort `ausgabe` entfernt die obersten drei Zahlen. Manche Worte holen erst Zahlen vom Stapel und legen anschließend neue Werte auf dem Stapel ab. Dies gilt insbesondere auch für Rechenoperationen wie Plus und Minus.

Wir schauen uns einmal etwas genauer an, wie man mithilfe des Wortes "+" in FORTH zwei Zahlen addiert. Zum Beispiel sollen die Zahlen 4 und 9 addiert werden. Von den meisten Taschenrechnern, aber auch von vielen Programmiersprachen, ist man es gewohnt, die folgende Anweisung zu schreiben:

```
4 + 9
```

Das Rechenzeichen steht zwischen den beiden Summanden; man spricht hier von einer **Infix**-Schreibweise.

In FORTH schreibt man dies so:

```
4  9  +                    (Leerzeichen zwischen 4 und 9 nicht vergessen!)
```

Hier werden zuerst die beiden Summanden eingegeben und anschließend erst das Rechenzeichen; man spricht hier von einer **Postfix**-Schreibweise.

Was steckt dahinter? Natürlich unser Stapel! Zunächst werden die Zahlen 4 und 9 auf den Stapel gelegt; dann holt das Wort "+" diese beiden Zahlen vom Stapel, addiert sie und legt das Ergebnis (also 13) wieder auf den Stapel. Der Vorteil dabei: Das Wort "+" führt bei FORTH die Addition sofort aus; beide Summanden liegen ja bereits vor. Bei der Infix-Schreibweise ist das nicht so einfach möglich. Hier müssen sich Taschenrechner oder Computer das Pluszeichen zunächst merken; die eigentliche Addition kann erst ausgeführt werden, wenn nach dem zweiten Summanden noch ein weiterer Befehl z. B. in Form von "=" erfolgt.

Wenden wir unsere Kenntnisse nun an, um den Attiny mit FORTH die Rechenaufgabe 4 + 9 durchführen zu lassen. Unser Programm sieht so aus:

```
: main  4  9  +  . ;
```

Wir geben es in das Quelltext-Feld ein, interpretieren, kompilieren und übertragen es. Die Leuchtdioden am Port B zeigen tatsächlich die Zahl 13 an (&B00001101).

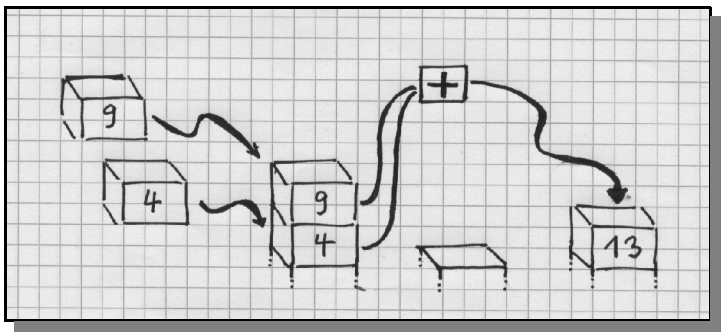


Abb. 4

Wir haben bereits gelernt, was hier im Einzelnen geschieht: Zuerst werden die Zahlen 4 und 9 auf den Stack gelegt; dann holt das Wort "+" diese beiden Zahlen vom Stack, addiert sie und legt das Ergebnis wieder auf den Stack. Das nächste Wort "." holt sich diese Zahl 13 vom Stack und gibt sie auf dem Port B aus.

Wir sehen: Der Stack ist eine Art Marktplatz, auf dem die einzelnen Wörter Zahlen holen oder auch abgeben können. Im Gegensatz zu einem echten Marktplatz können hier allerdings nur Zahlen gehandelt werden; außerdem gibt es hier nur einen einzigen Stand und an diesem Stand liegen die Zahlen nicht irgendwie nebeneinander, sondern ordentlich übereinander auf einem einzigen Stapel.

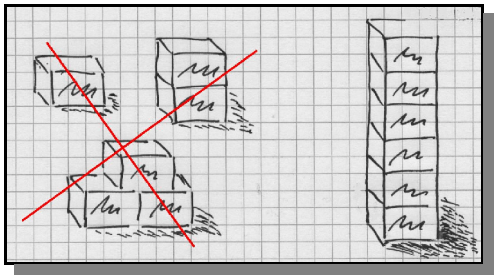


Abb. 5

An dieser Stelle sei schon verraten: FORTH stellt noch weitere Möglichkeiten zum Austausch von Daten zwischen den Wörtern zur Verfügung, z. B. sogenannte Variablen. Der Austausch über den Stack ist aber die wichtigste Methode. Deswegen wollen wir den Umgang mit dem Stack noch etwas üben.

Wie man einfache Rechenaufgaben mit FORTH lösen kann, haben wir schon kennen gelernt. Wie aber sieht es mit komplexen Termen aus?

1. Beispiel:

Term: (3 + 5) * 2

FORTH: `3 5 + 2 *`

2. Beispiel:

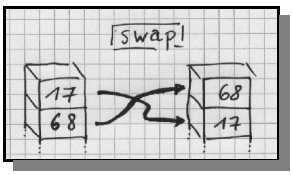
Term: 120 - 5 * 20

FORTH: `120 5 20 * -`

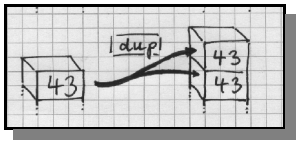
Bei dem letzten Beispiel könnten die Zahlen 120, 5 und 20 schon fertig auf dem Stack liegen; um das Ergebnis des Terms zu erhalten, müssten nur noch die Worte "*" und "-" hintereinander ausgeführt werden. Ginge das auch beim ersten Beispiel? Nein, auf keinen Fall wäre das so einfach wie im Beispiel 2: Da die Zahl 2 auf dem TOS liegt, würde jede Operation sich auf jeden Fall (auch) auf diese Zahl 2 beziehen. Die Klammern im Term verlangen aber, dass zunächst die Zahlen 3 und 5 verarbeitet (addiert) werden.

Es gibt aber eine Reihe von FORTH-Wörtern, die die Zahlen auf dem Stack manipulieren (vertauschen, entfernen oder verdoppeln) können:

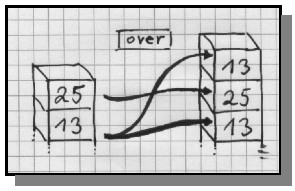
`swap` (a b – b a)
vertauscht die zwei obersten Stackwerte



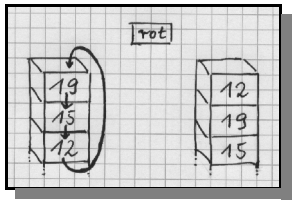
`dup` (a – a a)
dupliziert (verdoppelt) den obersten Stackwert



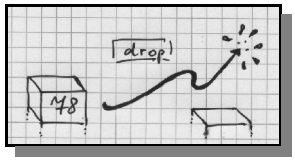
`over` (a b – a b a)
kopiert den zweitobersten Stackwert nach oben



`rot` (a b c – b c a)
rotiert die drittoberste Zahl nach oben



`drop` (a –)
entfernt die oberste Zahl vom Stack



Hier wurde auch eine für FORTH typische Beschreibung der Stackmanipulation benutzt: Die benutzten Stackwerte werden hinter dem FORTH-Wort in Klammern angegeben. Links vom Bindestrich stehen die Stackinhalte vor der Anwendung des FORTH-Wortes, rechts vom Bindestrich die Stackinhalte nach Ausführung des Wortes. Im Prinzip ist diese Schreibweise nur eine vereinfachte formale Darstellung unserer "Stapelbilder" wie in Abb. 4; bei der Klammerdarstellung ist das Stapelbild lediglich um 90° im Uhrzeigersinn gedreht.

Nehmen wir einmal an, auf dem Stack lägen schon die Zahlen a und b; Ziel wäre es, den Term

a * (a + b)

auszurechnen. Ein geeignetes Wort zur Berechnung des Terms könnte so aussehen:

```
: term over + * ;
```

Das macht die folgende Bildfolge deutlich.

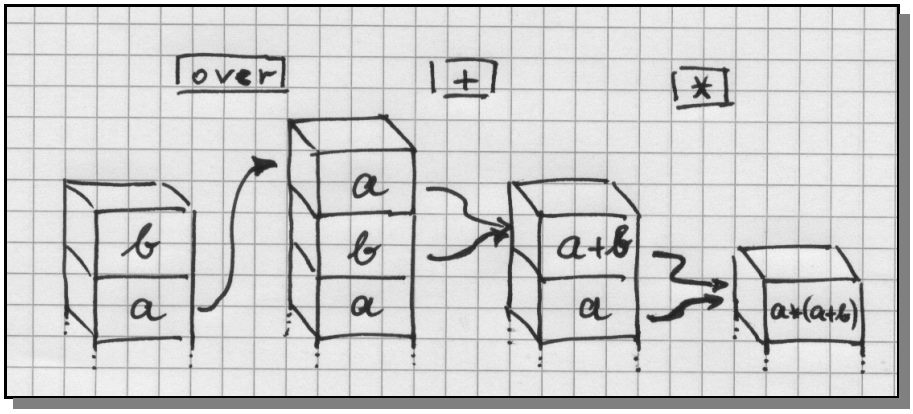


Abb. 6

Ließe sich auf ähnliche Weise auch ein geeignetes Wort für die Lösung unseres ersten Beispiels finden? Natürlich, und zwar so:

```
: beispiel1 rot rot + * ;
```

Zeichnen Sie dazu einmal selbst eine entsprechende Folge von Stapelbildern!

Sie sehen: Der Umgang mit den Stackmanipulationen ist im Prinzip einfach, aber gewöhnungsbedürftig. Deswegen hier einige Fingerübungen:

1. Überlegen Sie einmal selbst, wie man mit dem "*"-Wort das "quadrat" -Wort definiert; dieses soll zu einer Zahl auf dem TOS die zugehörige Quadratzahl liefern:

   `quadrat` (a – a^2)

2. `drop_2` (a b c ... – a c ...)

3. `antirot` (a b c – c a b)

4. `swap_2` (a b c – b a c)

5. `/oR` Division ohne Rest-Anzeige; “/” benutzen