MATHEMATISCHES KABINETT

Wie ein Taschenrechner funktioniert

Die Lösungen des Acht-Damen-Problems mit einem Taschenrechner hatten uns gezeigt, was diese Minicomputer lesisten können (siehe Mathematisches Kabinett in den Heften 10/78 und 1/79).

Voraussetzung war allerdings neben einem programmierbaren Taschenrechner die Entwicklung eines Algorithmus und eines Programms für die optimale Ausnutzung der verfügbaren Funktionen. Dieses Mal möchte Harald Nahrstedt aus Hamm in die Taschenrechner hineinschauen und die Frage nach dem *Wie* beantworten. Nicht die physikalische und elektronische Realisierung, sondern die funktionellen Arbeitsweisen der kleinen Rechenwerke sollen dabei interessieren.

Daß es Unterschiede bei den technischen Bausteinen und im Umfang der Funktionen gibt, ist jedem klar. Letztlich sind darin auch die Preisunterschiede begründet. Daß aber selbst bei einfachsten Rechnern, die nur die Grundrechenarten beherrschen, verschiedene Arbeitsweisen vorkommen, ertgeht den meisten Benutzern. Schließlich ist ja die Hauptsache, daß das Ergebnis stimmt.

Der Unterschied basiert auf der Schreibweise arithmetischer Ausdrücke. Im täglichen Umgang mit Zahlen und Rechenarten haben wir uns an die algebraische Notation (AN) gewöhnt. Ein Beispiel, das uns fortan begleiten soll:

Ergebnis: = A + B× (C – D) ÷ E (1)

Die Symbole (+ , – , × , ÷ ,) der zweistelligen (dyadischen) Operationen stehen je zwischen zwei Operanden (Infix-Darstellung). Klammern helfen, den Ausdruck eindeutig und übersichtlich zu machen. Bei komplizierteren Ausdrücken geht die Übersichtlichkeit aber schnell verloren, besonders wenn Teile mit Klammern ineinander verschachtelt sind.

Diese Klammern zu vermeiden war ein Hauptanliegen des Mathematikers Jan Lukasiewicz (1878 – 1956) aus der polnischen Schule der mathematischen Logik, als er die sogenannte „Polnische Notation" entwickelt hat. Die Operationszeichen stehen nicht zwischen, sondern vor oder hinter den Operanden, und entsprechend unterscheidet man die Präfix-Form

Ergebnis: = – + A × B ÷ C D E (2)

oder die Postfix-Form

Ergebnis: = A B C D ÷ + E ÷ – (3)

Beide Darstellungen entsprechen unserem Beispiel in der Form (1).

Lukasiewicz hatte sich für die Präfix-Form entschieden; über die Mathematik hinaus hat sich aber keine der beiden Schreibweisen durchgesetzt.

Die Postfix-Form – auch „umgekehrte Polnische Notation" (UPN) genannt – zeigt Vorteile bei der funktionellen Arbeitsweise von Rechenwerken. Sie spart Programmschritte gegenüber der AN-Form, wie wir noch später sehen werden. Und deshalb finden wir sie neben AN auch in Taschenrechnern realisiert.

Zunächst wollen wir aber die Ausdrücke (2) und (3) noch erläutern, sie also in die uns vertraute Form AN auflösen (Abbildung 1).

Bei der Präfixform beginnt man mit den letzten Operationszeichen (hier ein „÷") und den beiden unmittelbar danach stehenden Operanden (hier C und D). Das Zwischenergebnis Z, ist ein neuer Operand, der zum nächsten, weiter links liegenden Operationszeichen gehört. So löst man schrittweise von rechts nach links auf.

Bei der Postfix-Form funktioniert das alles umgekehrt: Man beginnt mit dem ersten Operationszeichen und den beiden links davor stehenden Operanden und verfährt analog weiter von links nach rechts bis das Ergebnis feststeht.

Kommen wir nun zur Realisierung der Notationen AN und UPN in Taschenrechnern. Einer umfassenden Darstellung wegen schließen wir die programmierbaren ein.

Sowohl AN- als auch UPN-Rechnertyp haben den in der Abbildung 2 schematisierten Aufbau. Sie verfügen also über Arbeits-, Programm- und Zahlenspeicher. Die Trennung der beiden letzteren wird neuerdings nicht mehr festeingehalten vorgenommen, man speichert vielmehr beide Datenarten (Befehle und Zahlen) in einem gemeinsamen Speicher. Dieser wird, wie die Abbildung 3 zeigt, von zwei Seiten gefüllt (dynamische Speicherverwaltung).

Über die Eingabe (Tasten) werden abhängig vom Zustand (Modus) des Rechners Zahlen in einen Arbeitsspeicher (RUN-Modus) oder Programmbefehle (LEARN-Modus) gegeben.

Zahlen aus den Arbeitsspeichern werden im Rechenwerk miteinander verknüpft oder es werden Zahlen verglichen. Die Ergebnisse werden dem Steuerwerk gemeldet, das den zeitlichen Ablauf aller Operationen steuert.

Die Zahlenspeicher werden je nach Anzahl mit einer oder zwei Ziffern numeriert (oudiert) und korrespondieren mit einem Speicher der Arbeitsspeicher-Einheit im einfachsten Fall durch die Befehle

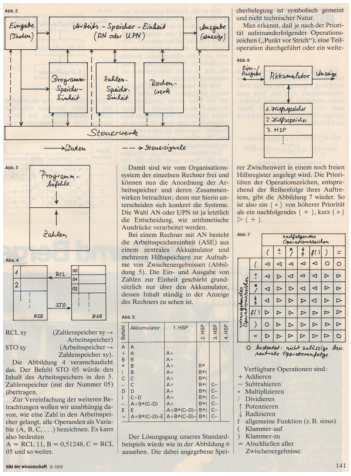

Abb. 2

Abb. 3

Abb. 4

RCL xy (Zahlenspeicher xy → Arbeitsspeicher)
STO xy (Arbeitsspeicher → Zahlenspeicher xy).

Die Abbildung 4 veranschaulicht das. Der Befehl STO 05 würde den Inhalt des Arbeitsspeichers in den 5. Zahlenspeicher (mit der Nummer 05) übertragen.

Zur Vereinfachung der weiteren Betrachtungen wollen wir unabhängig davon, wie eine Zahl in den Arbeitsspeicher gelangt, alle Operanden als Variable (A, B, C, . . .) bezeichnen. Es kann also bedeuten

A = RCL 11, B = 0,51248, C = RCL 05 und so weiter.

Damit sind wir vom Organisationssystem der einzelnen Rechner frei und können nun die Anordnung der Arbeitsspeicher und deren Zusammenwirken betrachten; denn nur hierin unterscheiden sich konkret die Systeme. Die Wahl AN oder UPN ist ja letztlich die Entscheidung, wie arithmetische Ausdrücke verarbeitet werden.

Bei einem Rechner mit AN besteht die Arbeitsspeichereinheit (ASE) aus einem zentralen Akkumulator und mehreren Hilfsspeichern zur Aufnahme von Zwischenergebnissen (Abbildung 5). Die Ein- und Ausgabe von Zahlen zur Einheit geschieht grundsätzlich nur über den Akkumulator, dessen Inhalt ständig in der Anzeige des Rechners zu sehen ist.

Abb. 5

Der Lösungsgang unseres Standardbeispiels würde wie in der Abbildung 6 aussehen. Die dabei angegebene Speicherbelegung ist symbolisch gemeint und nicht technischer Natur.

Man erkennt, daß je nach der Priorität aufeinanderfolgender Operationszeichen („Punkt vor Strich"), eine Teiloperation durchgeführt oder ein weiterer Zwischenwert in einem noch freien Hilfsregister angelegt wird. Die Prioritäten der Operationszeichen, entsprechend der Reihenfolge ihres Auftretens, gibt die Abbildung 7 wieder. So ist also ein (+) von höherer Priorität als ein nachfolgendes (+), kurz (+) > (+).

Abb. 6

Abb. 7

Verfügbare Operationen sind:

+ Addieren
– Subtrahieren
· Multiplizieren
/ Dividieren
↑ Potenzieren
√ Radizieren
f allgemeine Funktion (z. B. sinus)
(Klammer-auf
) Klammer-zu
= Abschließen aller Zwischenergebnisse.

MATHEMATISCHES KABINETT

Für unser Standardbeispiel ergeben sich die Auswertungsstufen wie in der Abbildung 8 gezeigt. Ein solcher Prioritätsausdruck wird von links nach rechts, ausgehend vom ersten Zeichenpaar < I >, ausgewertet. In dieser strengen Zuordnung liegt letztlich auch die UPN begründet.

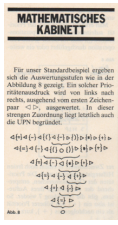

Abb. 8

Die an die Priorität gebundene Abspeicherung von Zwischenergebnissen bei Ausdrücken in der AN bedarf einer größeren Anzahl von Hilfsspeichern. Diese entfallen bei Rechnern mit UPN. Dafür haben wir es auch nicht mit einem zentralen Akkumulator, sondern mit mehreren Arbeitsspeichern zu tun. Sie sind zu einem Stapelspeicher zusammengeschlossen.

Wie der Name schon sagt, liegt der zuletzt eingegebene Wert „oben". Nur auf ihn kann wieder zurückgegriffen werden (Last-in – first-out = Prinzip). Schon vorhandene Werte werden beim Einspeichern eines neuen Wertes in die darunter liegenden Speicher transportiert. Die Abbildung 9 verdeutlicht dieses Speicherprinzip.

Abb. 9

Der Fachausdruck für das Verschieben der Werte heißt Stack-lift. Die Speicheranordnung selbst nennt man auch „Keller", und man spricht so auch vom „Einkellern".

In der Regel besitzt ein solcher Stapel vier Speicher. Wird also ein fünfter Wert eingegeben, fällt der zuerst eingegebene Wert unten heraus.

Die Abbildung 10 zeigt die zwischen den einzelnen Speichern möglichen Daten-Transformationen. Bei Durchführung einer Rechenoperation wird

Abb. 10

der Enter-Befehl umgekehrt. Der zuletzt eingegebene Wert wird zum Rechenwerk gebracht. Ein Stack-lift nach oben schiebt den vorletzten Wert nach oben. Er wird mit dem Wert im Rechenwerk verknüpft, das Ergebnis kommt zurück in den ersten Speicher des Stapels.

Die Abbildung 11 zeigt ein Additionsbeispiel. Ein automatischer Stack-lift sorgt nach Durchführung einer solchen Operation zuallererst dafür, daß ein neuer Wert eingegeben werden kann, ohne den Enter-Befehl zu benutzen; er wird bei der Zuweisung einer neuen Zahl automatisch durchgeführt.

Abb. 11

Stapelspeicher	vorher	nachher
1	X	X+Y
2	Y	Z
3	Z	U
4	U	U

In der Abbildung 12 kommen wir auf unser Standardbeispiel zurück. Sie

Abb. 12

Befehl	Stapelspeicher 1.	2.	3.	4.	Bemerkungen
A	A				
ent	A	A			
B	B	A			
ent	B	B	A		Das „Einkellern" der Werte muß beim Rechnen „von Hand" mit dem Enter-Befehl durchgeführt werden.
C	C	B	A		
ent	C	C	B	A	
D	D	C	B	A	
–	C-D	B	A	A	
*	B*(C-D)	A	A	A	
+	A+B*(C-D)	A	A	A	
E	E	A+B*(C-D)	A	A	Automatischer Stack-lift
÷	A+B*(C-D)÷E	A	A	A	

zeigt die Schrittfolge in einem Rechner mit UPN, wobei das „Einkellern" der Werte mit einem Enter-Befehl „von Hand" geschieht. Bei Benutzung eines Programms zur Ermittlung des Ergebnisses würden automatisch durchgeführte Stack-lift-Operationen die Enter-Befehle hinfällig machen und wir können zu der Schrittfolge in der Abbildung 13.

Vergleicht man die Befehlsfolge in der linken Spalte mit der Postfix-Form

Ergebnis: = (A + B + C) ÷ D – (E÷F)

sowohl Präfix- als auch Postfix-Form auf und bilden Sie die Schrittfolge in einem UPN-Rechner analog zur Abbildung 13.

Die Lösung können Sie im nächsten Heft vergleichen, und dann wollen wir Ihnen auch beweisen, warum ein UPN-Rechner zur Auflösung eines arithmetischen Ausdrucks weniger Befehle braucht als sein Rechner-Kollege mit der uns geläufigen traditionellen AN-Arbeitsweise.

Abb. 13

Befehl	Stapelspeicher 1.	2.	3.	4.	Bemerkungen
A	A				
B	B	A			Automatischer Stack-lift
C	C	B	A		Automatischer Stack-lift
D	D	C	B	A	Automatischer Stack-lift
–	C-D	B	A	A	
*	B*(C-D)	A	A	A	
+	A+B*(C-D)	A	A	A	
E	E	A÷B*(C-D)	A	A	Automatischer Stack-lift
÷	A+B*(C-D)÷E	A	A	A	

ergebnis in den ersten Speicher zurück. Durch den zwischenzeitlich erfolgten Stack-lift (vorletzter Wert nach oben) steht die nächste Variable im 2. Speicher zur nachfolgenden Verknüpfung bereit.

Man erkennt in der Abbildung 12 leicht die Grenzen des Stapels. Die Eingabe von mehr als vier Werten hintereinander ist nicht möglich. Auch daß bei Durchführung einer Operation der vorherige Inhalt des obersten Stapelspeichers verlorengeht, ist eine unangenehme Sache, da mitunter eine Variable mehrmals in einem Ausdruck verwendet werden könnte. Daher haben einige Rechner einen zusätzlichen Speicher (last X), wohin dieser Wert gerettet wird und für weitere Verknüpfungen zur Verfügung steht.

Das ist aber für das folgende Beispiel, das der Leser lösen soll, unbedeutend. Mit vier Speichern im Stapel kommt man aus. Stellen Sie für die AN-Form

n-Damen-Problem

Karl Müller aus Bremen hat einen Großrechner eingesetzt. Ein Taschenrechner hätte das Problem nicht mehr lösen können.

Für n > 8 hat er folgende Werte für die Anzahlen der Lösungen errechnen lassen:

n = 9	352 Lösungen
10	724 Lösungen
11	2 680 Lösungen
12	14 200 Lösungen
13	73 712 Lösungen
14	365 596 Lösungen

Prima! Bleibt nur noch zu hoffen, daß sein Rechnerprogramm richtig war.