Programmierung von Datenverarbeitungsanlagen

von

Dr. H. J. Schneider und **Dipl.-Math. D. Jurksch**

2. erweiterte Auflage

Mit 8 Tabellen und 14 Abbildungen

Sammlung Göschen Band 1225/1225a

Walter de Gruyter & Co · Berlin 1970

vormals G. J. Göschen'sche Verlagshandlung · J. Guttentag, Verlagsbuchhandlung · Georg Reimer · Karl J. Trübner · Veit & Comp.

Inhaltsverzeichnis

Literaturverzeichnis

[1] *Güntsch, F. R.:* Einführung in die Programmierung digitaler Rechenautomaten. 2. Aufl., Berlin, 1963

[2] *Steinbuch, K.* (Hrsg.): Taschenbuch der Nachrichtenverarbeitung. Berlin/Göttingen/Heidelberg, 1962

[3] *Naur, P.* (Hrsg.): Revised Report on the Algorithmic Language ALGOL 60.
Numer. Mathematik 4 (1963), 420—453 und
Comm. ACM 6 (1963), 1—17

[4] FORTRAN vs. Basic FORTRAN.
Comm. ACM 7 (1964), 591—625

[5] *Backus, J. W.:* The Syntax and Semantics of the Proposed International Language of the Zurich ACM-GAMM Conference, in: Information Processing 1959, München, 1960, 125—132

[6] *Zurmühl, R.:* Praktische Mathematik für Ingenieure und Physiker. 5. Aufl., Berlin/Göttingen/Heidelberg, 1965

[7] *R. Baumann:* ALGOL-Manual der ALCOR-Gruppe, München, 1968

[8] *D. E. Knuth* et al.: A proposal for input-output conventions in ALGOL 60
Comm. Assoc. Comp. Mach. 7 (1964), 273—283

[9] Anonym: Report on input-output procedures for ALGOL 60
Comm. Assoc. Comp. Mach. 7 (1964), 628—630

Einleitung

Der vorliegende Band soll den Leser in die Programmierung digitaler Datenverarbeitungsanlagen einführen. Als Beispiel für maschinenorientierte symbolische Programmierung wird ein kurzer Überblick über die wichtigsten Einzelheiten der weit verbreiteten Assemblersprache des IBM Systems /360 gegeben. Maschinenorientiert bedeutet, daß Assemblerbefehle in der Struktur den Maschinenbefehlen ähneln, jedoch für den Benutzer leichter lesbar sind. Im Hinblick auf die Vielfalt der verschiedenen Anlagen und die damit verbundene Vielfalt der Maschinensprachen stehen aber die problemorientierten Sprachen im Vordergrund. Darunter verstehen wir Programmiersprachen, die jeweils für einen Problemkreis entwickelt wurden. Wenn der Rahmen dieses Bandes nicht gesprengt werden soll, muß ein Problemkreis ausgewählt werden. Dies seien die Aufgaben der Mathematik, Naturwissenschaft und Technik, bei deren Lösung z. Z. besonders zwei Programmiersprachen im Vordergrund stehen: ALGOL 60 und FORTRAN. Diese beiden Sprachen sind im folgenden so behandelt, daß die Abschnitte auch getrennt gelesen werden können. (Benutzt ein FORTRAN-Abschnitt Kenntnisse aus einem ALGOL-Abschnitt, so ist ein entsprechender Hinweis vorhanden.) Die in Kleindruck gesetzten Textabschnitte kann ein Leser, der ohne Vorkenntnisse beginnt, zunächst einmal überschlagen.

1. Grundbegriffe und Programmvorbereitung

1.1 Programm und Rechenformular

Die numerische Auswertung mathematischer Aufgabenstellungen kann im allgemeinen von Hilfskräften bewältigt werden, die mit dem eigentlichen Problem

nicht vertraut sind. Dazu muß die Aufgabe zuvor in eine Reihe von einzelnen Rechenanweisungen (Befehle) zerlegt werden, die sich auf möglichst elementare Operationen beschränken, z. B. die vier Grundrechenoperationen und das Nachschlagen in Tabellen. Beschreibt eine solche Anweisungsfolge eine in sich abgeschlossene Aufgabe, so nennen wir sie ein Programm. Als Beispiel sei die Lösung eines linearen Gleichungssystems mit zwei Unbekannten betrachtet:

$$\begin{aligned} a\,x_1 + b\,x_2 &= e \\ c\,x_1 + d\,x_2 &= f \end{aligned}$$

Führen wir die Bezeichnungen

$$\begin{aligned} D &= ad - bc \\ D_1 &= ed - bf \\ D_2 &= af - ec \end{aligned}$$

ein, so ergeben sich für $D \neq 0$ die Lösungen aus der Cramerschen Regel zu

$$x_1 = \frac{D_1}{D} \text{ und } x_2 = \frac{D_2}{D}.$$

Ist jedoch $D = 0$, dann hat das Gleichungssystem im Fall $D_1 \neq 0$ und/oder $D_2 \neq 0$ keine Lösung, während im Fall $D_1 = D_2 = 0$ eine weitere Betrachtung zur Feststellung der Existenz von Lösungen durchgeführt werden muß. Diese Sonderbetrachtung soll hier aber übergangen werden.

Die nacheinander auszuführenden Operationen ergeben ein *Rechenformular* (Tab. 1). Auf Grund der angegebenen Operationen (im Beispiel ab Zeile 7) werden die in den bezeichneten Zeilen eingetragenen Zahlen (‹n› = Inhalt von Zeile n) miteinander verknüpft und das Ergebnis neben der entsprechenden Anweisung festgehalten.

Die Operationen in den Zeilen 17 bis 22 behandeln die Fragestellung $D_1 \neq 0$ und/oder $D_2 \neq 0$: Ist ‹12› $\neq 0$, dann interessiert ‹15› nicht mehr. Ist dagegen ‹12› $= 0$,

Tab. 1

Nr.	Rechenoperation	Zahlenwerte				Bemerkung
1	Ausgangswerte	−1	2	1	...	a
2		2	4	2	...	b
3		2	3	2	...	c
4		−4	2	4	...	d
5		1	14	3	...	e
6		1	13	6	...	f
7	‹2› × ‹3›	4	12	4	...	bc
8	‹1› × ‹4›	4	4	4	...	ad
9	‹8› − ‹7›	0	− 8	0	...	D
10	‹2› × ‹6›	2	52	12	...	bf
11	‹5› × ‹4›	−4	28	12	...	ed
12	‹11› − ‹10›	−6	−24	0	...	D_1
13	‹5› × ‹3›	2	42	6	...	ec
14	‹1› × ‹6›	−1	26	6	...	af
15	‹14› − ‹13›	−3	−16	0	...	D_2
16	Fortsetzung bei 23, wenn ‹9› ≠ 0					D ≠ 0?
17	Fortsetzung bei 21, wenn ‹12› ≠ 0					D_1 ≠ 0?
18	Fortsetzung bei 21, wenn ‹15› ≠ 0					D_2 ≠ 0?
19	Eintragen 'SB'			SB	...	Sonderbetr.
20	Fortsetzung bei 25					
21	Eintragen 'KL'	KL			...	keine Lösung
22	Fortsetzung bei 25					
23	‹12› / ‹9›		3		...	x_1
24	‹15› / ‹9›		2		...	x_2
25	Nächste Spalte beginnen bei 7					

so hängt es nur noch von ‹15› ab, ob keine Lösung existiert oder ob eine Sonderbetrachtung notwendig ist.

Überflüssig sind diese Operationen, wenn D ≠ 0 ist. Daher wird ihnen die Fragestellung in Zeile 16 vor-

ausgeschickt, so daß sie für ‹9› $\neq 0$ übergangen werden können.

1.2 Aufbau einer programmgesteuerten Rechenanlage

Die Anweisungen des Rechenformulars lassen sich recht einfach und ohne Kenntnis der ursprünglichen Problemstellung mit einer Tischrechenmaschine behandeln. Die zu verarbeitenden Zahlen werden aus dem Formular abgelesen, in der Maschine eingestellt und durch Druck auf die entsprechende Operationstaste miteinander verknüpft. Das Resultat kann dann abgelesen und im Formular notiert werden. Fassen wir noch den Transport der Zahlen vom Formular zur Maschine und zurück als Operationen auf, so ist bereits in großen Zügen der Aufbau einer programmgesteuerten Rechenanlage beschrieben (Abb. 1):

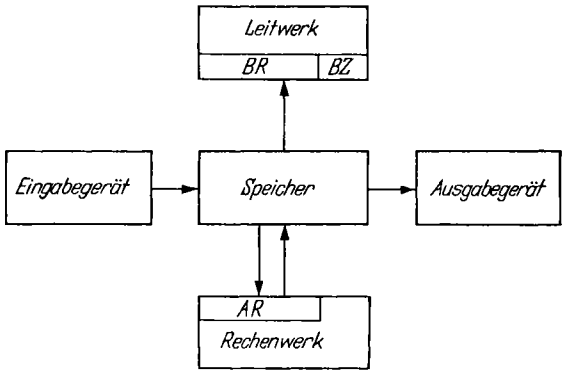

Abb. 1

Dem Rechenformular entspricht der *Speicher*. Er dient zum Aufbewahren von Informationen, z. B. Zahlen und Befehlen. Diese stehen den anderen Teilen der Anlage auf Anforderung zur Verfügung. Damit jede Information eindeutig erreichbar ist, ist der Speicher in

Einheiten unterteilt: die *Speicherzellen*. Sie entsprechen den Zeilen des Rechenformulars und sind durchnumeriert. Die Nummer einer Speicherzelle bezeichnen wir als ihre *Adresse*. Der Inhalt einer solchen Zelle heißt *Wort*. Ein grundlegender Unterschied zwischen dem Speicher und dem hier dargestellten Rechenformular besteht jedoch: Während nämlich im Rechenformular in jeder Zeile eine Operation und alle aus ihr resultierenden Ergebnisse eingetragen werden, kann eine Speicherzelle immer nur eine einzige Information — Rechenoperation oder Zahlenwert — aufnehmen. Wird eine Zelle beschrieben, so wird ihr alter Inhalt durch die neue Information ersetzt. Damit ein Programm sich nicht selbst zerstört, müssen die von ihm errechneten Zahlenwerte in anderen Speicherzellen untergebracht werden.

Das *Leitwerk* entspricht dem menschlichen Bediener. Es holt sich nacheinander die im Speicher stehenden Operationen — im folgenden auch *Befehle* genannt — in ein *Befehlsregister* BR und sorgt für ihre Ausführung. Die sequentielle Abhandlung der einzelnen Operationen wird mit Hilfe eines *Befehlszählregisters* BZ erreicht. Dieses enthält jeweils die Adresse der Speicherzelle, in der der nächste Befehl zu finden ist, und wird beim Transport dieses Befehls nach BR um 1 erhöht. Eine Sonderstellung nehmen die Sprungbefehle ein, die den BZ mit einer neuen Adresse füllen, so daß der sequentielle Programmablauf unterbrochen und das Programm an einer neuen Stelle fortgesetzt wird.

Je nach Art des Befehls werden im Speicher stehende Zahlen über ein *Rechenwerk* mit dem Inhalt eines *Akkumulatorregisters* AR verknüpft oder vom AR in den Speicher transportiert.

Ein- und *Ausgabegeräte* sorgen für die Kommunikation zwischen Rechenmaschine und Benutzer. Die Eingabegeräte besorgen den Transport der Befehle und Parameter in den Speicher, und die Ausgabegeräte übernehmen von dort die Ergebnisse. Meist werden diese

Ein- und Ausgabegeräte (kurz: E-A-Geräte) vom Leitwerk in Tätigkeit gesetzt.

Während das klassische Modell der programmgesteuerten Rechenanlage einen Speicher, ein Leitwerk und ein Rechenwerk hat, sind heute Rechner mit mehreren Speichern nicht mehr selten, und es gibt auch solche mit mehreren Rechen- oder Leitwerken. Diese erlauben, mehrere Operationen oder mehrere Programme gleichzeitig auszuführen. Im Rahmen dieser Einführung kann darauf jedoch nicht weiter eingegangen werden. Eine etwas ausführlichere Darstellung der dabei auftretenden Probleme findet sich z. B. bei [1] und [2].

1.3 Informationsdarstellung

Jede Information, sei es eine Zahl oder ein Befehl, wird durch eine nichtnegative N-stellige Zahl im Binärsystem dargestellt. N heißt Wortlänge. Die Angabe eines Unterscheidungsmerkmals zwischen Zahlen und Befehlen ist nicht notwendig: Die vom Leitwerk behandelten Speicherinhalte werden als Befehle und die ins Rechenwerk transportierten Informationen als Zahlen interpretiert.

1.3.1 Das Festpunktwort

Das *Festpunktwort* dient zur Darstellung ganzer Zahlen in der Form

$$z = b_{N-1}\, b_{N-2} \ldots b_2\, b_1\, b_0.$$

Dabei ist jedes b_i entweder 1 oder 0 und heißt *Binärstelle* oder *Bit*. z hat dann den Zahlenwert

$$z = \sum_{i=0}^{N-1} b_i\, 2^i.$$

Beispielsweise bedeutet 00...0011010 die Zahl $1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 16 + 8 + 2 = 26$. Die größtmögliche Zahl wäre

$$\sum_{i=0}^{N-1} 2^i = 2^N - 1,$$

aber es wird eine Stelle für das Vorzeichen benötigt.

Demnach ist bei einer Wortlänge N die größtmögliche darstellbare Zahl

$$2^{N-1} - (1.^1)$$

Um negative Zahlen auszudrücken sind zwei verschiedene Darstellungen gebräuchlich. Es gibt Rechenanlagen, die beide nebeneinander erlauben, in der Regel ist aber nur eine von ihnen möglich:

Die negativen Zahlen lassen sich einmal durch das Komplement bzgl. $2^N - 1$ ausdrücken, d. h. daß jede 0 in eine 1 und jede 1 in eine 0 verwandelt wird. -26 wird als 11...1100101 dargestellt. Positive Zahlen haben dann in der vordersten Stelle eine 0, negative eine 1. Die Zahl 0 ist zweimal vorhanden:

$$+0 = 00...0000$$
$$-0 = 11...1111.$$

Die zweite Darstellung der negativen Zahlen benutzt das Komplement bzgl. 2^N. Diese Darstellung erhält man aus der ersten durch Addition einer 1. Also wird -26 durch 11...1100110 ausgedrückt. Beim 2^N-Komplement gibt es nur eine 0. Dagegen läßt sich das Komplement der dem Betrage nach größten negativen Zahl 1000..00 nicht bilden. Es entsteht ein Überlauf. Denkt man sich an irgendeiner Stelle ein Komma gesetzt, so lassen sich die Speicherinhalte auch als gebrochen rationale Zahlen interpretieren.

Um für den Benutzer eine bequemere Lesbarkeit zu erzielen, ist es üblich, extern (d. h. außerhalb der Maschine) die binären Ziffern zu Gruppen zusammenzufassen und diese durch ihre dezimalen Werte zu ersetzen[2]. Dabei haben sich vor allem die *oktale* (Zusammenfassen von 3 Ziffern) und die *hexadezimale* (Zusammenfassen von 4 Ziffern) Schreibweise durchgesetzt. Z. B. ist

$$26_{\text{dezimal}} = 11\ 010_{\text{dual}}$$
$$= 3\quad 2_{\text{oktal}}$$

[1] Bei einigen Anlagen wird ein weiteres Bit für die Überlaufkontrolle (Überschreitung des Zahlenbereichs) verbraucht. Meistens ist hierfür jedoch eine gesonderte Anzeige üblich.

[2] Sofern die Zahl nicht ins Dezimalsystem konvertiert wird.

Im Hexadezimalsystem verwendet man für die Ziffernwerte 10 bis 15 die Buchstaben A bis F. So ergibt sich z. B.

$$26_{\text{dezimal}} = 1\,1010_{\text{dual}} = 1\quad A_{\text{hexadezimal}}$$

1.3.2 Das Gleitpunktwort

Wesentlich erhöhen läßt sich der Zahlenbereich durch die halblogarithmische Notierung der von 0 verschiedenen Zahlen in der Form:

$$z = p \times B^q \quad (B > 1)$$

p heißt Mantisse und q Exponent. Um die Darstellung eindeutig zu machen, wird p normiert:

$$\frac{1}{B} \leq |p| < 1$$

Für B werden je nach der bevorzugten externen Darstellungsweise die Werte 2 (dual), 8 (oktal) oder 16 (hexadezimal) genommen. Die Mantisse hat die duale Darstellung

$$|p| = \sum_{i=-(n-1)}^{-1} b_i\, 2^i.$$

Einschließlich des Vorzeichens braucht man dazu n Stellen des Wortes. Für den Exponenten q bleiben N − n Stellen übrig. Um beim Exponenten das Vorzeichen zu sparen, ist es üblich, statt q die Charakteristik $\bar{q} = q + 2^{N-n-1}$ zu betrachten, so daß sich für q eine Variationsmöglichkeit von

$$-2^{N-n-1} \leq q \leq 2^{N-n-1} - 1$$

ergibt.

Bezüglich der Anordnung von p und $\bar{q}$ innerhalb eines Wortes haben sich verschiedene Methoden eingebürgert. Wir wollen zwei von ihnen an Beispielen demonstrieren:

Speichert man zuerst p und im Anschluß daran $\bar{q}$, so ergibt sich z. B. für N = 32, n = 24 und B = 2

$26 = 0{,}8125 \times 2^5$:

$$\underset{\qquad\qquad p}{0110100000000000000000000} \quad \underset{\bar{q}}{10000101}$$

$0{,}05078125 = 0{,}8125 \times 2^{-4}$:

$$\underset{\qquad\qquad p}{0110100000000000000000000} \quad \underset{\bar{q}}{01111100}$$

Bei negativen Zahlen wird lediglich die Mantisse p komplementiert. Verwendet man das Komplement bezüglich 2^n, so erhält man z. B.

$-26 = -0{,}8125 \times 2^5$:

$$\underset{\qquad\qquad p}{1001011111111111111111111} \quad \underset{\bar{q}}{10000101}$$

Bei einer anderen Darstellung der Gleitpunktzahlen[3] ist die Reihenfolge von p und q vertauscht. Für das Vorzeichen v von p benutzt man das erste Bit des Wortes. Es wird $B = 16$ genommen, und für $N = 32$ ist $n = 25$:

$26 = 0{,}1015625 \times 16^2$:

$$\underset{v}{0} \quad \underset{\bar{q}}{1000010} \quad \underset{\qquad\qquad p}{00011010000000000000000}$$

$0{,}05078125 = 0{,}8125 \times 16^{-1}$:

$$\underset{v}{0} \quad \underset{\bar{q}}{0111111} \quad \underset{\qquad\qquad p}{11010000000000000000000}$$

Die negativen Zahlen erscheinen im Gegensatz zu der zuerst beschriebenen Darstellungsweise in der üblichen Form. Das bedeutet, daß z. B. bei einer Multiplikation mit -1 lediglich das Vorzeichenbit v komplementiert wird und alles andere unverändert bleibt:

$-26 = -0{,}1015625 \times 16^2$:

$$\underset{v}{1} \quad \underset{\bar{q}}{1000010} \quad \underset{\qquad\qquad p}{00011010000000000000000}$$

Extern wird die Mantisse für $B = 16$ in hexadezimaler Form angegeben:

$$|p| = \sum_{i=-\frac{n-1}{4}}^{-1} h_i \, 16^i \qquad (0 \leq h_i \leq 15),$$

[3] z. B. IBM System /360