Eurocomp

ACT I

Algebraischer Compiler und Übersetzer für LGP 21

Eurocomp GmbH · Elektronische Rechenanlagen
495 Minden/Westf. · Schillerstr. 72 · Tel. 0571/8 34 21

D 015/765/1

A C T I

Algebraischer Compiler und Übersetzer

für den LGP-21

PROGRAMMIERUNGSANLEITUNG

INHALTSVERZEICHNIS

Einleitung

Der Compiler ACT I (Algebraic Compiler and Translator) dient dazu, das Programmieren für den elektronischen Ziffernrechner LGP-21 erheblich zu vereinfachen. Der ACT I benutzt eine Sprache, die der vertrauten Schreibweise arithmetischer Formeln entspricht und darüber hinaus Elemente enthält, mit denen logische Entscheidungen in einfacher Weise programmiert werden können.

Der LGP-21 stellt sich aus dem in der ACT I - Sprache geschriebenen Programm das Objektprogramm her und speichert es auf der Magnetscheibe (Compiler-Phase). Nachdem die Unterprogramme, die benutzt werden sollen, eingelesen worden sind, kann sofort mit der Rechnung begonnen werden. Der Compiler wird von den Unterprogrammen überspeichert.

Bevor der ACT I beschrieben wird, soll er kurz mit dem ACT V, einem anderen Compiler für den LGP-21, verglichen werden. Mit dem ACT I kann man längere Objektprogramme herstellen als mit dem ACT V. Außerdem stehen für den ACT I im Gegensatz zum ACT V alle Unterprogramme für Gleitkomma- und zusätzlich für Festkommarechnung zur Verfügung. Ein wesentlicher Vorteil des ACT V besteht darin, daß Prozeduren benutzt werden können, daß sehr einfach Text geschrieben werden kann und daß Doppelindizes verwendet werden können. Der ACT V compiliert zwar schneller und die erstellten Objektprogramme werden etwas kürzer, aber in der Rechenphase benötigen die Programme mehr Zeit.

Kapitel I

Die Sprache des ACT I

Anweisungen

Das Quellenprogramm besteht aus einer Reihe von Anweisungen. Folgende Aufgaben können durch solche Anweisungen erledigt werden:

1. Auswertung von mathematischen Formeln
2. Abänderung des normalen Befehlsablaufs, z.B. Verzweigungen und schleifenerzeugende Anweisungen.
3. Eingabe-Anweisungen. Zahlen und Text können über Lochstreifen oder über Schreibmaschine eingegeben werden.
4. Ausgabe-Anweisungen. Zahlen und Text können über Schreibmaschine oder Stanzer ausgegeben werden.

Variable

Zwei Arten von numerischen Größen können im Quellenprogramm auftreten: Konstanten, deren numerischer Wert direkt im Quellenprogramm festgelegt wird oder Variable, deren Wert entweder eingegeben oder während des Programmablaufs errechnet wird.

Für Variable gelten die folgenden Regeln:

1. Der Name darf nicht mehr als 5 Zeichen lang sein.
2. Er darf nicht nur aus Zahlen bestehen.
3. Er darf nicht mit dem Zeichen für eine Operation identisch sein.
4. Er darf nicht aus einer Klammer oder einem Anweisungskennzeichen (s.u.) bestehen.

Nur die dritte Regel erfordert einige Aufmerksamkeit. Speziell ist "x" das Zeichen für die Festkomma-Multiplikation, darf also nicht für die unabhängige Variable x benutzt werden. Auch die Leertaste ist ein spezielles Zeichen. Die Namen "asq", "as q", "asq "und "a sq" bedeuten verschiedene Variable.

Gleitkommazahlen

Gleitkommazahlen werden in halblogarithmischer Darstellung mit Mantisse und Exponent geschrieben. Zur Eingabe müssen die Zahlen folgendes Format haben: Das erste Wort enthält das Vorzeichen urd die siebenstellige Mantisse. Das zweite Wort enthält eine zweistellige Zehnerpotenz von - 31 bis + 31 mit Vorzeichen. Pluszeichen können weggelassen werden.

Beispiele:

Zahl	Gleitkommadarstellung
123,45	1234500' 03'
0,000987	9870000'-03'
-0,6789	-6789000' 00'

Die zulässige Größe für Gleitkommazahlen liegt zwischen

$$0{,}1 \cdot 10^{-31} \leq a < 1 \cdot 10^{31}.$$

Generell werden bei der Rechnung 7 bis 8 Dezimalstellen (=24 binäre Stellen) mitgeführt. Umwandlungs- und Rundungsfehler sind gewöhnlich klein, können sich aber bei langen Rechnungen mit beträchtlichen Unterschieden in den Größenordnungen aufsummieren.

Der Stopcode

Beim Ablochen des Quellenprogramms muß jeder Variablenname, jede Konstante, jedes Operationszeichen und jedes Befehlssymbol von einem Stopcode beendet werden.

So ist z.B. a'/'b' nicht identisch mit a/b'.

Am Schluß einer jeden Anweisung folgt ein zusätzlicher Stopcode.

Gleitkomma-Operationen

f+, f-, fx, f/ sind die Symbole für die vier Grundrechenoperationen (das f steht für floating point).

Das Symbol f - bedeutet Subtraktion und nicht Vorzeichenwechsel. Vorzeichenwechsel erreicht man mit 0'f-'a' .

In der Algebra wird der Punkt für die Multiplikation oft weggelassen, im ACT I darf das Symbol f x niemals weggelassen werden.

fabs ist das Symbol für die Bildung des Absolutbetrages.

sqrt sqrt'a' bedeutet Quadratwurzel von a. a darf nicht negativ sein.

ln ln'a' bedeutet natürlicher Logarithmus von a; a muß positiv sein.

log log'a' bedeutet Logarithmus zur Basis 10 von a ; a muß positiv sein.

eexp eexp'a' bedeutet e^a .

10exp	10exp'a' ergibt entsprechend 10^a.
sin	sin'a' bedeutet Sinus zum Bogen a; a muß in Bogenmaß angegeben werden. Ist a in Grad gegeben, so ist der Sinus von a/57.29578 zu bilden.
cos	cos'a' bedeutet Cosinus zum Bogen a.
asin, acos, atan	asin'a', acos'a', atan'a' ist für die Umkehrfunktionen arcsin, arccos, arctang zu schreiben.

Das Zuordnungszeichen

Das Zeichen " : " wird als Zuordnungszeichen benutzt. Folgendes Beispiel möge zur Erläuterung dienen:

a'fx'a'f+'b'fx'b':'c"

Diese Anweisung errechnet a^2+b^2 und weist den Wert der Variablen c zu. Das nächste Beispiel zeigt deutlich, daß das Zeichen ":" nicht als algebraisches Gleichheitszeichen zu interpretieren ist:

sum'f+'a'fx'a':'sum"

Mehrfache Zuordnungsanweisungen sind möglich, z.B.

0':'sumx':'sumy':'sumz"

Festkommazahlen

Alle Festkommazahlen stehen bei q = 29. Festkommazahlen werden als ganze Zahlen gespeichert. Der Programmierer muß zur Festlegung des dezimalen Kommas Maßstabsfaktoren einführen. Beispielsweise kann die Zahl, die im LGP-21 als 12345 steht, vom Programmierer als 12,345 interpretiert werden. Die Zahl hat dann 3 Dezimalstellen hinter dem Komma (Q=3). Bei Addition und Subtraktion sind solche Faktoren anzubringen, daß beide Zahlen gleiche Kommastellung haben. Bei Multiplikation sind die Q-Werte zu addieren, bei Division ist der Q-Wert des Divisors von dem des Dividenden zu subtrahieren.

Beispiel:

a habe Q=2

b habe Q=3

c habe Q=2

Es soll gerechnet werden: $y = \frac{(a+b)\ c}{b}$

Dafür ist zu bilden:

$$y = \frac{(10a+b)c}{b}$$

Der Maßstabsfaktor von y berechnet sich zu:

$$Q=3+2-3=2$$

Für den Zahlenbereich für Festkommazahlen gilt:

$$-536\ 870\ 911 \leq a \leq 536\ 870\ 911 = 2^{29}-1$$

Die größte Zahl, die durch Multiplikation entstehen kann, ist $2^{27}-1=134\ 217\ 727$. Negative Festkommazahlen sind mit führenden Nullen zu schreiben, z.B. wird - 12345 eingegeben als - 0012345'.

Das Programmieren im Festkomma ist deshalb umständlich, weil man die Größenordnung aller Zahlen kennen muß, wenn man nicht dauernd Überläufe erhalten oder zu viele Stellen hinter dem Komma verlieren will. Es wird nur dann angewandt, wenn die Größenordnungen bekannt oder leicht zu überschauen sind und wenn auf hohe Rechengeschwindigkeit Wert gelegt wird. Indexrechnungen können mit Gleitkomma - oder Festkommazahlen vorgenommen werden. Die Indizes sdbst können nur ganze Zahlen sein.

Festkomma-Operationen

+, -, x, / sind die Symbole für die vier Grundrechnungsarten für Festkomma-Operationen.

abs ist das Symbol für die Bildung des Absolutbetrages.

Da in der Rechenphase entweder die Gleitkomma- oder die Festkommaprogramme gespeichert sind, werden dieselben Funktionssymbole benutzt. Dabei sind zwei Ausnahmen gemacht worden:

loge loge'a'bedeutet natürlicher Logarithmus von a

log10 log10'a' bedeutet Logarithmus zur Basis 10 von a

Eingabe und Ausgabe

Die Operationen für Ein- und Ausgabe sind:

read'a" Der Wert für die Größe mit dem Symbol a wird in Festkommaform eingelesen. Der Compiler ordnet dem Symbol a eine Adresse im Hauptspeicher zu, an der der eingelesene Wert gespeichert wird.

finp'a" Der Wert wird in Gleitkommadarstellung eingelesen.

aread'a" Es werden 5 alphanumerische Zeichen für die Größe mit dem Symbol a eingelesen.

rdhex'a" Ein hexadezimales Wort wird eingelesen und nach a gespeichert. Binärisieren entfällt hierbei. Es handelt sich bei diesen hexadezimalen Wörtern um spezielle Symbole oder um Speicherauszüge von vorher gerechneten Programmen.

n'print'a" Der Wert von a wird als Festkommazahl ausgeschrieben. n gibt an, wieviel Dezimalstellen hinter dem Komma geschrieben werden sollen.

fprt'a" Der Wert von a wird in Gleitkommaform ausgegeben. Minuszeichen stehen vor Mantisse bzw. Exponent.

aprt'a" Es werden die 5 alphanumerischen Zeichen der Größe a ausgeschrieben.

Die folgenden 2 Operationen dienen der Formatsteuerung beim Ausschreiben:

cr" bewirkt einen Wagenrücklauf der Schreibmaschine.

tab" bewirkt einen Tabulatorsprung der Schreibmaschine.

Mit dem stop-Befehl wird der Rechner angehalten: §top"

Bemerkung:

Diese drei Operationen müssen durch zwei Stopcodes abgeschlossen werden.

Wenn zwei Operationszeichen direkt aufeinander folgen, muß das zweite Operationszeichen in Klammern gesetzt werden.
Beispiel: fprt' ['sin'a'] ", a'fx'['fabs'a']'

Umwandlungen

Zur Umwandlung von Zahlen aus der Festkommadarstellung in die Gleitkommadarstellung und umgekehrt dienen die Operationen flo und unflo. Vor dem Operationszeichen steht immer eine positive Zahl, die angibt, um wieviel Dezimalstellen das Komma verschoben wird.

Beispiele:

2'flo'y" Der Zahlenwert der Größe y wird von der Festkommadarstellung in Gleitkommadarstellung umgewandelt; das Komma wird um 2 Stellen nach links verschoben.

3'unflo'y" Der Zahlenwert der Größe y wird von der Gleitkommadarstellung in Festkommadarstellung umgewandelt; das Komma wird um 3 Stellen nach rechts verschoben. Alle Stellen hinter dem Komma gehen verloren.

2'print'['2'unflo'y']" Der Zahlenwert der Größe y wird mit 2 Stellen hinter dem Komma als Festkommazahl ausgedruckt.

Gleitkommakonstante können aus Festkommakonstanten erzeugt werden, z.B.: 0'flo'3':'drei" und 6'flo'['31416'x'100'+'29']':'pi"

Vorrang und Klammern

Die Vorrangstufen der algebraischen Operationen entsprechen der gewöhnlichen Ordnungskonvention der Algebra. Bei Abwesenheit von Klammern werden die Funktionen (abs, sin, ln usw.) zuerst ausgeführt, haben also den höchsten Vorrang. Multiplikation und Division haben den nächst niedrigeren Vorrang, Addition und Subtraktion haben den niedrigsten Vorrang.

Wie in der Algebra können die Vorränge von Operationen durch das Setzen von Klammern abgeändert werden. Die innersten Klammern haben

den höchsten Vorrang, die äußeren den niedrigsten. In einer Klammer gelten die normalen Vorrangstufen. Wegen der Notwendigkeit, eine Anweisung immer in eine Zeile zu schreiben, müssen bei der Niederschrift unter Umständen mehr Klammern als in der Originalgleichung benutzt werden.

Beispiele:

a) $y = \frac{2a \cdot \sin(b+c) + d}{e^2 - 4ab}$

Die Anweisung im Quellenprogramm (Gleitkomma) muß lauten:

```
['zwei'fx'a'fx'['sin'['b'f+'c']']'f+'d']'
 f/'['e'fx'e'f-'vier'fx'a'fx'b']':'y"
```

Vor dieser Anweisung müssen die beiden Gleitkommakonstanten zwei' und vier' festgelegt werden:

```
0'flo'2':'zwei"  und  0'flo'4':'vier"
```

Die Operationen werden in der folgenden Reihenfolge ausgeführt:

1. `b'f+'c'`
2. `sin'['b'f+'c']'`
3. `zwei'fx'a'`
4. `zwei'fx'a'fx'sin'['b'f+'c']'`
5. `zwei'fx'a'fx'sin'['b'f+'c']'f+'d'`
6. `e'fx'e'`
7. `vier'fx'a'`
8. `vier'fx'a'fx'b'`
9. `e'fx'e'f-'vier'fx'a'fx'b'`
10. `['...]' f/'['...]':'y"`

b) $y = \frac{a}{b \cdot c}$

kann folgendermaßen geschrieben werden:

```
a'f/'['b'fx'c']':'y"  oder  a'f/'b'f/'c':'y"
```

Programmablauf

Die Anweisungen eines Quellenprogramms werden bei der Ausführung normalerweise in der Reihenfolge ihrer Niederschrift verarbeitet. Wenn dieser Ablauf abgeändert werden soll, so müssen Anweisungskennzeichen eingeführt werden. Unter Anweisungskennzeichen versteht man eine Markierung, die einer bestimmten Anweisung im Quellenprogramm zugeordnet ist. Ein Anweisungskennzeichen besteht aus dem Buchstaben S und irgendeiner Zahl zwischen 0 und 255. Es wird vor die Anweisung gesetzt, z.B. `S0'read'a"`. Mit Hilfe von unbedingten oder bedingten Sprungoperationen kann nach so markierten Anweisungen gesprungen werden.

Sprungoperationen und Vergleiche

use — `use'S100"` bewirkt einen unbedingten Sprung zu der Anweisung mit dem Kennzeichen S100.

trn — trn'S100" bewirkt einen bedingten Sprung. Ist der Inhalt des Akkumulators negativ, so erfolgt ein Sprung nach S 100. Andernfalls wird die nächste Anweisung ausgeführt. Findet Verwendung beim Vorzeichentest von Gleitkommazahlen.

' less' kleiner als

when'a'grt'b'trn'S30" Falls a größer als b ist,

equal' gleich

wird als nächste Anweisung die mit dem Kennzeichen S 30 genommen. Vergleiche dieser Art können nur mit Festkommazahlen ausgeführt werden.

ret — Die Rückkehroperation ret dient zum Einsetzen von Sprungadressen und wird bei der Verwendung von Unterprogrammen benutzt. Sie wird weiter unten an einem Beispiel erklärt.

Beispiele

Für die Berechnung einer Größe y gelten 3 verschiedene Formeln, je nachdem, ob die Summe a+b eine vorgegebene Grenze c überschreitet, unterschreitet oder ihr gleich ist. Die Anweisungen des Quellenprogramms lauten:

```
when'['a'+'b']'grt'c'trn'S3"
when'['a'+'b']'less'c'trn'S4"
a'x'a'+'b':'y"use'S8"
S3'a'x'b'-'b':'y"use'S8"
S4'a'+'b'x'b':'y"
S8'y'x'['...
```

Die erste Anweisung bewirkt einen Sprung zur Anweisung S3, wenn (a+b) > c, sonst geht das Programm zur nächsten Anweisung. Diese prüft, ob (a+b) ≤ c; wenn ja, erfolgt ein Sprung nach S4, wenn nein, wird die nächste Anweisung ausgeführt. Die nächsten 3 Anweisungen enthalten die drei verschiedenen Formeln. Aus allen drei Anweisungen kommt man mit Hilfe der unbedingten Sprungoperation use'S8" zur Anweisung S8, bei der das Programm normal weiterläuft.

Statt eines Symbols kann in der Vergleichsoperation auch eine Festkommakonstante stehen. Sind die zu vergleichenden Größen Ergebnisse von Rechnungen, so müssen sie in Klammern stehen, z.B.

```
when'['a'+'b']'grt'['c'+'d'x'f']'trn'S12"
```

Will man aus dem normalen Programmablauf herausspringen und später an die gleiche Stelle zurück, so benutzt man die Rückkehroperation ret.
Beispiel:

```
S0'read'a'read'b"
ret'S7'use'S6"
```

```
S3'a'+'b'x'd':'c"
        .
        .
        .
S12'use'S4"
S6'a'x'b':'d"
        .
        .
        .
S7'use'S0"
S4'. . .
```

Die Rückkehroperation setzt in Anweisung S7 anstelle von S0 die Adresse der übernächsten Operation nach ret (hier S3) ein. Danach erfolgt ein Sprung nach S6. Sobald das Programm nach S7 kommt, wird der Sprung zurück nach S3 ausgeführt. Es ist wichtig, daß das Kennzeichen in Anweisung S7 (in diesem Fall S0) vorher schon verwendet wurde.

Ein spezielles Hilfsmittel zur Programmierung von Programmverzweigungen sind die Programmsprungtasten. Eine solche Anweisung muß im Quellenprogramm lauten:

PS4'S10" — S10 steht für ein beliebiges Anweisungskennzeichen und gibt die Adresse an, zu der bei gedrückter PS-4-Taste gesprungen wird. Ist diese Taste nicht gedrückt, so wird die nächste Anweisung ausgeführt. Es gibt 4 Programmsprungtasten, sie haben die Bezeichnung PS-4, PS-8, PS-16 und PS-32.

Indizierte Größen

In ACT I Programmen können einfachindizierte Größen (=Bereichssymbole) werwendet werden.

1. Index-Anweisung

Die Anweisung index 'i'j'k" schafft drei Indexregister mit den Symbolen i, j und k. Es dürfen ein- bis fünfstellige Symbole genommen werden. Alle Indizes eines Programms müssen in einer einzigen Index-Anweisung definiert werden. Eine zweite Index-Anweisung in demselben Quellenprogramm wird ignoriert. Die Indizes können folgendermaßen angegeben werden:

a'i', a'7', b'j', b'20'.

Für die Bereichssymbole muß am Anfang des Programms ein Bereich im Speicher reserviert werden. Es können bis zu 61 Indizes definiert werden.

2. Bereichsreservierungen

Bereichsreservierungen werden mit der Anweisung dim vorgenommen. Unmittelbar hinter dim müssen das Bereichssymbol und die Anzahl der zu reservierenden Plätze folgen.

Beispiel: dim'a'36"

Diese Anweisung reserviert für das Bereichssymbol a im Speicher

36 Plätze. Das Symbol a darf nur für die Werte aus diesem Bereich benutzt werden.
Will man einen dieser Werte auswählen, so muß das durch eine nachgesetzte Konstante oder Index erfolgen. Das Bereichssymbol darf niemals ohne Konstante oder Index benutzt werden. Die interne Numerierung beginnt bei Null, der erste Platz wird immer für die Größe a'0' reserviert. Nennt man den ersten Wert a'1', und ist der letzte Wert a'n', so muß man n+1 Plätze reservieren. Der erste Platz wird dann nicht benutzt.

Doppelindizes können in Einfachindizes umgewandelt werden. Es sei die Matrix $\mathcal{A}$ mit (m+1)Spalten und (n+1) Zeilen und den Elementen a_{ik} (i=Zeilenindex, k= Spaltenindex) gegeben;

$$\mathcal{A} = \begin{pmatrix} a_{00} & a_{01} & a_{02} & \dots & a_{0m} \\ a_{10} & a_{11} & a_{12} & \dots & a_{1m} \\ \vdots & & & & \\ a_{n0} & a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix}$$

Im Programm werde das Element a_{ik} verlangt, wobei die Indizes i und k durch eine Indexrechnung definiert sind. Dann läßt sich ein einfacher Index j nach

$$i(m+1)+k+1=j$$

errechnen, so daß $a_{ik} = a_j$ ist.
Für den Bereich a müssen $(m+1)\cdot(n+1)+1$ Plätze reserviert werden.

Oft ist es auch notwendig, mit demIndex eines Index zu arbeiten. b_r bedeutet, daß der Index r des Bereichs b selbst ein Bereich ist $_s$ und daß sein (ganzzahliger) Wert durch den Index s in der Werteliste r_s festgelegt ist. Man wird also in der dim-Anweisung die Bereiche b und r reservieren und in der Indexliste den Index s und einen Hilfsindex t definieren. Dann leistet

```
r's':'t"
b't' . . .
```

das Gewünschte.

Es können insgesamt 11 verschiedene Bereichssymbole verwendet werden. Symbole und Adressen werden ab Zelle 6240 gespeichert, und zwar erst die Symbole, die durch ein Codewort beendet werden und anschließend die zugehörigen Adressen.

Die Beschränkung auf 11 Bereichssymbole ist oft lästig. Man kann sich aber dadurch helfen, daß man mit Hilfsindizes arbeitet. Sei dim'a'201" festgelegt; die ersten 100 Werte von a mögen dem Bereich a'j' entsprechen, die zweiten 100 Werte von a sollen dem Bereich b'j' entsprechen; für b'j' sei aber keine Reservierung möglich (12. Symbol). Durch j'+'100'; 'jj" und Aufruf a'jj' wird dann der fehlende Bereich b simuliert; allerdings muß die Umrechnung über den Hilfsindex jj bei jeder j-Schleife -meist am Anfang der Schleife- erfolgen.

Programmschleifen

Sollen bestimmte Programmteile mehrmals, z.B. mit verschiedenen Ele-

menten von Bereichssymbolen, durchlaufen werden, so benutzt man die iter-Anweisung. Diese Anweisung erhöht den Wert eines Symbols um einen vorgegebenen Betrag und prüft ihn gegen eine angegebene Grenze. In der Anweisung müssen drei Symbole oder Konstanten stehen: die zu ändernde Größe, der Betrag der Erhöhung und die Grenze. Am Ende schließlich muß stets ein Anweisungskennzeichen stehen, auf das gesprungen wird, wenn die Größe noch kleiner als die Grenze ist oder deren Wert erreicht hat. Wird der Grenzwert überschritten, so wird die nächste Anweisung ausgeführt.

Beispiel: iter'i'1'26'S16"

Die Größe mit dem Symbol i wird bei jeder Ausführung der Anweisung um den Wert 1 erhöht. Solange i kleiner oder gleich 26 ist, wird zur Anweisung S16 zurückgesprungen, ist i größer als 26, wird die nächste Anweisung ausgeführt. Nach Durchlaufen der iter-Anweisung ist i=27.

Statt der Konstanten können auch Symbole in der Anweisung stehen, z.B. iter'i'di'imax'S16". Hierbei ist zu beachten, daß diese Symbole irgendwann vorher schon einmal eingeführt sein mußten und daß sie im Festkomma stehen müssen. Gegebenenfalls sind sie mit "unflo" umzuwandeln. Die abzuändernde Größe wird meistens ein Index sein, es kann aber auch jede andere Größe geändert werden. Der Anfangswert ist vorher außerhalb der Schleife durch eine besondere Anweisung festzulegen.

Unterprogramme

Im Compiler ACT I sind eine Reihe von häufig gebrauchten Unterprogrammen eingebaut, die mit speziellen Symbolen versehen sind und mit diesen auf einfachste Weise aufgerufen werden. Sie können zusammen mit anderen Operationen in einer Anweisung benutzt werden und haben immer Vorrang 3.

Will man andere Unterprogramme benutzen, so muß man bei deren Abfassung besondere Vorschriften bezüglich der Rückkehradresse und der Verarbeitung von Mehrfachargumenten beachten (siehe Kapitel IV). Der Aufruf solcher Unterprogramme erfolgt mit der Anweisung sub, in der dann unmittelbar ein Symbol als Name und Kennzeichnung des Unterprogramms und ein oder mehrere Symbole folgen, die vom Unterprogramm wahlweise als Argument oder als Ergebnis verwendet werden können (Einzelheiten siehe Kapitel IV).

Beispiel: sub'tanh'alfa'tgha"

Diese Anweisung bewirkt einen Sprung in das Unterprogramm tanh für die Berechnung des hyperbolischen Tangens, wobei das Kennzeichen tanh willkürlich gewählt sein kann und mit Hilfe der Anweisung dim'tanh'123", die gleichzeitig die Speicherplätze für das Unterprogramm reserviert, definiert wird. Das Argument heißt alfa, das Ergebnis hat das Symbol tgha, mit dem dann normal weiteroperiert werden kann.

Kapitel II

Aufbau des Quellenprogramms

Allgemeines

Das Quellenprogramm setzt sich aus einer Vielzahl von Anweisungen zusammen. Eine Anweisung enthält eine oder mehrere Operationen, verbunden mit Konstanten, Symbolen, Klammern oder Anweisungs-Kennzeichen. Die Länge einer Anweisung ist begrenzt auf insgesamt 63 Operationen, Symbole, Klammern, Konstanten o.ä.

Als erste Anweisung im Quellenprogramm steht meistens dim'comp'512"; sie belegt 8 Spuren (50-57) für die Unterprogramme für Gleitkommaoperationen. Werden außerdem irgendwelche anderen Unterprogramme für die Funktionen benutzt, so heißt diese Anweisung dim'comp'(576+n)", wobei n so groß sein muß, daß die Plätze für das zu benutzende Unterprogramm (s. Tabelle auf Seite 23) auch wirklich reserviert werden. Der Anweisung dim'comp' xxx" müssen unmittelbar alle übrigen Bereichsreservierungen (für spezielle Unterpläne oder indizierte Variable) folgen.

Die Anweisung dim'comp'kann entfallen, wenn entweder keine sonstigen Bereichsreservierungen nötig sind oder wenn nur im Festkomma und außer Eingabe/Ausgabe ohne sonstige Unterprogramme gearbeitet wird.

Die Festlegung der Indizes muß unmittelbar nach den dim-Anweisungen mit index....erfolgen. Für die später mit Indizes zu versehenden Größen muß vorher ein Bereichssymbol festgelegt sein.

Die dann folgenden Anweisungen enthalten im allgemeinen Operationen zum Einlesen. Man muß darauf achten, daß alle Symbole, mit denen später gerechnet wird, vorher entweder durch Einlesen der entsprechenden Zahlenwerte oder durch Berechnung aus Konstanten festgelegt wurden.

Das Gleichheitszeichen : ordnet dem rechts von ihm stehenden Symbol eine neue Speicheradresse zu, wenn dieses Symbol bisher noch nicht vorkam und speichert den errechneten Wert an diesen Platz. Danach kann mit diesem Symbol als linkem Operand in einer neuen Anweisung weitergerechnet werden. In der gleichen Anweisung kann man nur dann weiterrechnen, wenn die Operationen links und das Symbol rechts des Gleichheitszeichens in einer Klammer eingeschlossen sind und diese Klammer als Operand für direkt anschließende Operationen steht.

Die Anweisungen werden normalerweise in der Reihenfolge, in der sie stehen, ausgeführt. Soll diese Reihenfolge durchbrochen werden, sind Sprunganweisungen vorzusehen. Diejenigen Anweisungen, die im Verlaufe des Programms durch Sprunganweisungen aufgerufen werden sollen, müssen mit Anweisungs-Kennzeichen versehen werden, die als e r s t e s Zeichen in der betr. Anweisung stehen müssen. Die Numerierung kann in beliebiger Weise (≤ 256) und Reihenfolge erfolgen. Man muß aber darauf achten, daß alle in den Anweisungen iter, trn, use gebrauchten Kennzeichen irgendwo im Programm einmal am Anfang einer Anweisung stehen.

Als letzte auszuführende Anweisung steht stets stop".

Äußere Form des Quellenprogramms

Zuerst sind immer die Bereichsreservierungen vorzunehmen, dim'comp'xxx" darf nicht vergessen werden. Unmittelbar darauf muß die index-Anweisung folgen. Es sind Anweisungs-Kennzeichen zu definieren. Bei dim und index dürfen keine Anweisungskennzeichen angegeben werden. Man spart Zeit beim Compilieren, wenn die Variablennamen als letztes Zeichen einen Buchstaben enthalten. Bereichssymbole müssen immer mit einer Konstanten oder einem Index angegeben werden. Wenn mehrere Operationszeichen aufeinander folgen, müssen Klammern gesetzt werden. Die speziellen Anweisungen dim, index, iter und sub dürfen nicht mit anderen Operationen in einer Anweisung verbunden werden. Die Anzahl der öffnenden und schließenden Klammern in einer Anweisung muß gleich sein. Wenn Rechenoperationen mit Entscheidungen oder Druckoperationen verbunden werden, müssen Klammern gesetzt werden. Mit der whan-Vergleichsanweisung können nur Festkommazahlen verglichen werden. Für Vergleiche von Gleitkommazahlen benutzt man die Operation trn.

Mit der Anweisung trn wird eine Verzweigung in Abhängigkeit vom Akkumulatorinhalt festgelegt. Ist der Inhalt negativ, so wird zu dem angegebenen Anweisungs-Kennzeichen gesprungen, andernfalls wird die nächste Anweisung ausgeführt.

Beispiel:

```
['tol'f-'['fabs'['y'k'f-'ex'k'],],]'trn'S10"use'S12"
```

Solange $|y_k - x_k| > tol$, wird nach S10 gesprungen, andernfalls nach S12.

Zahlenmäßge Begrenzungen

Symbole dürfen bis zu 5 Zeichen enthalten.
Konstante dürfen aus bis zu 5 Zahlen bestehen.

Folgende Höchstzahlen dürfen nicht überschritten werden:

1 Indexanweisung je Quellenprogramm

11 verschiedene Bereichssymbole

127 verschiedene Symbole

39 verschiedene Konstanten

256 Anweisungskennzeichen

60 Operationen maximal (wichtig für eventuelle Änderung der Operationstafel)

64 Stopcodes in einer Anweisung

6 ineinandergeschachtelte Klammern in einer Anweisung.

Kapitel III

Benutzung des Compilers

Lochen des Quellenprogramms

Das Lochen hat auf einem im LGP-21-Code arbeitenden Flexowriter zu erfolgen. Dabei sind folgende Regeln zu beachten:

1. Es muß jedem Symbol, Operationszeichen, Bereichssymbol, Anweisungs-Kennzeichen, jeder Konstanten und Klammer ein Stopcode folgen. Der Stopcode kennzeichnet das Ende eines solchen Zeichens. Jedes Zeichen wird für sich eingelesen und vom ACT I entschlüsselt. Ein zusätzlicher Stopcode muß am Ende jeder Anweisung stehen.

2. Nach der letzten Anweisung des Quellenprogramms sind insgesamt drei Stopcodes zu lochen.

3. Es ist zweckmäßig, das Quellenprogramm mit Kleinbuchstaben zu schreiben. Bei der Eingabe über Flexowriter gehen Steuerungssymbole (wie Größenumschaltung und Kleinumschaltung) nicht in den Rechner, sie werden ignoriert. Dagegen werden Steuerungssymbole bei Eingabe über Schnelleser eingelesen. Das ist bei Verwendung der Operation aread zu beachten.

   Beispielsweise sind die folgenden Zeichen äquivalent:

   : und ; Gleichheit
   ? und / Division
   [und , Klammer auf
   ] und . Klammer zu

   usw.

Außerdem ist zu beachten, daß das Zeichen "l" als Ziffer 1 interpretiert wird. Damit fehlt der Buchstabe l. Kommt in einem Formelplan die Konstante l vor, so ist im Quellenprogramm dafür z.B. el' zu schreiben. Es empfielt sich, jede Anweisung auf eine neue Zeile zu schreiben; der Wagenrücklauf wird nicht in den LGP-21 eingelesen. Beispiele von Quellenprogrammen finden Sie in Kapitel VI.

Compilieren eines Quellenprogramms

Um aus einem Quellenprogramm ein Maschinenprogramm herzustellen, hat man folgendermaßen vorzugehen:

1. ACT I - Streifen mit Hilfe eines Eingabeprogramms, J1-10.1 oder SF-J1-2014, in den LGP-21 einlesen. Eingabe und Aufruf dieser Eingabeprogramme wird in der allgemeinen Programmierungsanleitung bzw. in den speziellen Programmbeschreibungen erläutert.

2. Streifen mit Quellenprogramm in die Lesestation des Flexowriters einlegen.

3. Am Flexowriter MANUELL drücken.

4. Aufruf von J1-10.1.

5. Schreiben des Schlüsselwortes .0004000' auf Flexowriter (=(F)). Der Compiler beginnt bei 4000.

6. RECHNER START drücken (F).

7. MANUELL lösen (F).

8. START am Rechner drücken.

Das Quellenprogramm wird in Maschinenbefehle übersetzt. Nach Beendigung des Compilierens wird folgendes ausgedruckt:

i xxxx Anfangsadresse des Objektprogramms.

f xxxx Endadresse des Objektprogramms.

S01 xxxx Absolute Adresse von S1.

S02 xxxx Absolute Adresse von S2.

. .
. .
. .

Danach hält der Rechner an. Das Objektprogramm ist nun auf den Speicherplätzen i xxxx bis f xxxx gespeichert. Anschließend kann das Programm ausgestanzt werden, ein weiteres Quellenprogramm compiliert oder die Rechnung vorbereitet werden.

Das Ausstanzen und Ausdrucken des Objektprogramms erfolgt mit Hilfe des im Compiler enthaltenen Druckprogramms über Flexowriter.

a) Ausstanzen unmittelbar nach dem Compilieren.

1. LOCHEN EIN drücken (F).
2. BANDLAUF drücken (F).
3. START am Rechner drücken.

Das Objektprogramm wird hexadezimal ausgeschrieben und ausgestanzt.

b) Ausstanzen mit besonderem Aufruf über Flexowriter.

Ein besonderer Aufruf des Druckprogramms ist notwendig, wenn zwischen Compilieren und Ausstanzen der Inhalt des Befehlsregisters verändert wurde.

1. MANUELL drücken (F).
2. Aufruf von J1-10.1.
3. Schreiben des Schlüsselwortes .0005501 auf (F).
4. RECHNER START drücken (F).
5. LOCHEN EIN drücken (F).
6. BANDLAUF drücken (F).
7. START am Rechner drücken.

Das Objektprogramm wird hexadezimal ausgeschrieben und ausgestanzt.

Der Compiler erlaubt es, mehrere Quellenprogramme nacheinander zu compilieren. Die Objektprogramme sind dann nach dem Compilieren auszustanzen.
Bemerkungen:
Am Ende des Compiler-Streifens befindet sich eine Korrektur, die nur dann einzulesen ist, wenn bei den Anweisungen aread und rdhex über Flexowriter eingelesen werden soll.

Die Adressen der Anweisungs-Kennzeichen können mittels Sprung nach 5413 ausgestanzt werden.

Fehlersuche und Fehlerstops

Es gibt im ACT I eine sehr einfache Möglichkeit, Fehler aufzuspüren, indem man vor dem Compilieren, d.h. vor dem Sprung nach 4000, die PST-Taste am LGP-21 drückt. Dadurch wird im Objektprogramm nach jeder compilierten Anweisung eine Aufruffolge erzeugt, die das Unterprogramm Trace aufruft. Drückt man nun vor der Ausführung des Maschinenprogramms die PST-Taste, so wird nach jeder Anweisung die folgende Information ausgedruckt:

a) Die Adresse des ersten Befehls, der auf die ausgeführte Anweisung folgt. In Verbindung mit den Adressen der Anweisungs-Kennzeichen kann man feststellen, welche Anweisung des Quellenprogramms gerade ausgeführt wurde.

b) Das Ergebnis der ausgeführten Anweisung, das im Akkumulator steht. Es wird einmal in Festkommadarstellung und bei positiven Gleitkommazahlen zusätzlich in Gleitkommadarstellung ausgeschrieben.

Damit kann man nach jeder Anweisung kontrollieren, ob die Zwischenergebnisse noch stimmen. Das Ausdrucken erfolgt nur solange, wie die Sprungtaste gedrückt ist. Nimmt man sie heraus, so wird das Ausschreiben dieser Zwischenergebnisse unterbrochen.

Will man bei einem Testlauf die Ergebnisse einiger spezieller Anweisungen in dieser Form ausgeben, so hat man im Quellenprogramm hinter diese Anweisungen das Symbol"trace" zu setzen. Bei der Rechnung muß wie oben die Sprungtaste gedrückt sein. Es empfiehlt sich, das ausgetestete Quellenprogramm noch einmal ohne trace zu compilieren, da jede Operation trace zwei Speicherplätze belegt und Rechenzeit erfordert.

Während des Compilierens prüft der ACT I das Quellenprogramm auf Formfehler. Es können insgesamt 8 verschiedene Fehler festgestellt werden. Bei Auffindung eines solchen Fehlers wird ein Kennzeichen ausgeschrieben, und der Rechner hält an. Drückt man am Flexowriter EINGABE VON HAND und am Rechner START, so kann sofort über Flexowriter die verbesserte Anweisung eingegeben werden. Nach der Verbesserung ist EINGABE VON HAND zu lösen. In der folgenden Tabelle werden die Fehlerkennzeichen zusammengestellt und erklärt.

Kennzeichen	Bedeutung	Verbesserung
e 2	Maschinenprogramm ist zu groß, überlappt mit Bereichsreservierungen oder Endadresse > 2963	Programm ändern, evtl. Bereiche verkleinern
e 3	Wert in der Anweisung dim ist keine Zahl	Anweisung verbessern

Kennzeichen	Bedeutung	Verbesserung
e 4	Quellenprogramm hat zwei Index-Anweisungen	Programm verbessern; zu einer zusammenfassen
e 5	Auf-Klammern und Zu-Klammern ungleich	Betr. Anweisung verbessern
e 6	Anweisung ist zu groß	In 2 oder mehr Anweisungen aufteilen
e 7	Anweisungskennzeichen ist nicht definiert	Programm verbessern oder nachprüfen, ob Flexowriter richtig eingelesen hat. Bei schlechter Justierung wird u.U. das erste Zeichen einer neuen Zeile (oft ein S des Anweisungskennzeichens) nicht gelesen. Dann Flexowriter justieren. Durch Sprung auf 5413 kann man sich alle Anweisungskennzeichen ausgeben lassen
8	Falsches Bereichssymbol	Anweisung verbessern
e 9	Linker Operand falsch	Anweisung verbessern

Programm Ausführen

Soll das Objektprogramm nach dem Compilieren oder nach dem Ausstanzen ausgeführt werden, so ist folgendermaßen vorzugehen:

1. Die zu benutzenden Unterprogramme einlesen.
   Der Streifen ACT I - UPA enthält die Unterprogramme für:

   a) Gleitkommaoperationen (6 Spuren)
   b) Trace, Float, Unfloat (2 Spuren)
   c) Festkomma - Ein/Ausgabe (2 Spuren)

   Falls Funktionen wie sin, cos, usw. in dem Programm benutzt werden, muß außerdem ein Streifen ACT I - UPB eingelesen werden. Weitere Einzelheiten siehe Kapitel IV.

2. Lochstreifen mit den einzugebenden Daten in den Flexowriter oder Schnelleser einlegen.

3. Aufruf von J1-10.1 und Sprung zur Anfangsadresse des Maschinenprogramms.

4. RECHNER START drücken (F).
   Der Rechner wählt den Schnelleser für die Dateneingabe an. Soll über Flexowriter eingelesen werden, so hat man unter Punkt 1 die Änderung am Schluß des UPA-Streifens mit einzulesen.

Steht das Objektprogramm nicht mehr auf der Speicherscheibe, weil inzwischen ein anderes Programm gespeichert wurde, muß der hexadezimal ausgestanzte Lochstreifen ebenfalls eingelesen werden.

Kapitel IV

Benutzung von Unterprogrammen

In dem Compiler ACT I sind eine Anzahl von Unterprogrammen enthalten, die unmittelbar durch ein besonderes Symbol aufgerufen werden können. Sollen andere Unterprogramme benutzt werden, so muß das mit der Anweisung sub geschehen.

Zum ACT I gehörige Unterprogramme

Unterprogramme, die praktisch immer benutzt werden, sind

1. Trace, Float und Unfloat
2. Gleitkommaoperationen
3. Festkomma Eingabe - Ausgabe

Sie dienen der Eingabe und Ausgabe sowohl für Festkomma- als auch für Gleitkommazahlen. Ferner werden sie für alle Gleitkommaoperationen benötigt und für das Prüfen des Programms bei gedrückter Sprungtaste.

Diese Unterprogrammgruppe ist auf einem Lochstreifen zusammengefaßt, der mit ACT I - UP A gekennzeichnet ist.Sie wird auf den Spuren 50 - 59 gespeichert.

Bei Halt innerhalb dieser Unterprogramme liegt folgender Tatbestand vor:

Adresse, in der Halt erfolgt	Programm	
5110	Unfloat	Zahl der Dezimalziffern ist zu groß
5429	Float	Exponent ist 32

Am Ende des UPA-Streifens befindet sich eine Korrektur für Einlesen über Flexowriter. Die Korrektur ist mittels J1 - 10.1 oder SF-J1-2014 einzugeben und betrifft nur read und finp.

Zu dem UPA-Streifen gehört ein kurzer Korrekturstreifen für das Ausstanzen von Gleitkommazahlen in wiedereinlesbarer Form. Diese Änderung kann durch Einlesen des zweiten Teils dieses Streifens rückgängig gemacht werden.

Bei Division durch Null erfolgt kein Fehler-Stop. Man muß also beim Programmieren darauf achten, daß eine Division durch Null nicht auftreten kann.

Bei Problemen aus der Matrizenrechnung und Aufgaben aus der Statistik

kann ein spezieller UPA-Streifen verwendet werden. Durch Weglassen von Nulltests wird die Rechnung beschleunigt. Es können bis zu 12 % Rechenzeit eingespart werden. Dabei wird vorausgesetzt, daß in der Rechnung nur wenige Nullen vorkommen. Für die Gleitkoma - Null muß 0'-32' eingegeben werden.

Die übrigen zum ACT I gehörigen Unterprogramme sind in untenstehender Tabelle zusammengestellt.

Diese Unterprogramme werden unmittelbar mit dem Operationszeichen aufgerufen. Es können entweder nur die für Gleitkomma oder die für Festkomma benutzt werden. Man muß sich von vornherein für ein System entscheiden. Praktisch ist das meistens das Gleitkomma. Die Verbindungen (linkages) zu diesen Unterprogrammen stehen auf Spur 49. Sie sind für Gleitkomma und Festkomma verschieden.

Die Lochstreifen für diese beiden Unterprogrammgruppen enthalten getrennt nach GK und FK jeweils alle oben angeführten Unterprogramme sowie die Verbindungen. Die Streifen sind gekennzeichnet durch ACT I - UP B - GK für Gleitkomma bzw. ACT I - UP B - FK für Festkomma.

Tabelle zu den Unterprogrammen

Funktion	Gleitkomma		Festkomma	
	Operations-zeichen	Belegte Spuren	Operations-zeichen	Belegte Spuren
arc 'sin arc cos	ASIN ACOS	3300-3631	ASIN ACOS	4432-4563
Exponential-funktion	EEXP 10EXP	3632-3931	EEXP 10EXP	4100-4163
Logarithmus	LN LOG	3932-4163	LOGE LOG10	4200-4431
arctg	ATAN	4200-4463	ATAN	4600-4663
sin, cos	SIN COS	4500-4731	SIN COS	4700-4763
Quadratwurzel	SQRT	4732-4863	SQRT	4800-4863

Eingabe und Ausgabe für die Festkomma-Unterprogramme

Bei Benutzung der Festkomma-Unterprogramme ist das jeweilige Format von Argument und Ergebnis zu beachten.

Unterprogramm	Argument im Akkumulator	Ergebnis im Akkumulator
sin, cos	α in Grad $\cdot 10^5$ z.B. α =45 als 45 00 000	sin α, cos α $\cdot 10^7$
arctg	$a \cdot 10^5$	α in Grad $\cdot 10^5$
$\sqrt{\ }$	a mit 2 n Dezimalen	$\sqrt{a}$ mit n + 4 Dezimalen
arcsin arccos	$a \cdot 10^7$	α in Grad $\cdot 10^5$
$\exp_e$ $\exp_{10}$	$a \cdot 10^7$; $-1 \leq a \leq 1$	e^a bzw. $10^a \cdot 10^7$
ln $\log_{10}$	a mit n Dezimalen, $a > 0$, $0 \leq n \leq 8$ Aufruf: n'loge'a' bzw. n'log10'a'	ln a bzw. lg $a \cdot 10^6$

Die folgenden Ausführungen beziehen sich auf Festkommaunterprogramme. Die Funktionen sin und cos werden durch ein Polynom 9. Grades angenähert. Der maximale absolute Fehler beträgt $5 \cdot 10^{-7}$. Die Funktion arctg wird durch ein Polynom 15. Grades approximiert, der maximale Fehler beläuft sich auf $1 \cdot 10^{-5}$. Die Quadratwurzel wird iterativ bestimmt, der Fehler hängt wesentlich davon ab, mit wieviel Dezimalstellen man rechnet. Die Funktionen arcsin und arccos werden durch ein Polynom 7. Grades approximiert, der maximale Fehler beträgt $1 \cdot 10^{-5}$. Ist der Betrag des Argumentes größer als 1, so resultiert daraus ein Fehlerstop in 4529. Die Exponentialfunktionen werden ebenfalls durch ein Polynom 7. Grades dargestellt. Der maximale Fehler beläuft sich auf $1 \cdot 10^{-7}$. Der maximale Fehler der Funktionen ln und log beträgt $1 \cdot 10^{-6}$, ist das Argument negativ oder Null, so ergibt sich ein Fehlerstop in 4208.

Zur Erläuterung der Formate von Argument und Ergebnis folgt ein einfaches Programm:

```
.0004000
dim'comp'1100"
10000'x'1000':'a"10000'x'100':'b"10000'x'10':'c"
eexp'a':'a1"eexp'b':'b1"
10exp'a':'a2"10exp'b':'b2"
6'loge'a':'a3"6'loge'b':'b3"
6'log10'a':'a4"6'log10'b':'b4"
asin'a':'a5"asin'b':'b5"
acos'a':'a6"acos'b':'b6"
atan'c':'c1"
sin'c':'c2"cos'c':'c3"
sqrt'b':'b7"
7'print'a"7'print'b"5'print'c"cr"cr"
7'print'a1"7'print'b1"cr"7'print'a2"7'print'b2"cr"
6'print'a3"6'print'b3"cr"6'print'a4"6'print'b4"cr"
5'print'a5"5'print'b5"cr"5'print'a6"5'print'b6"cr"
5'print'c1"cr"7'print'c2"7'print'c3"cr"7'print'b7"
stop'''
i 0322  f 0630
.0000322
1.0000000     .1000000    1.00000        a, b, c
2.7182817    1.1051709                  e-Funktion
9.9999998    1.2589253                  Exponentialfkt. Basis 10
 2.302585      .000000                  ln
  .999999      .000000                  log
 90.00000     5.73917                   asin
   .00000    84.26082                   acos
 45.00000                               atan
 .0174524     .9998476                  sin/cos
1.0000000                               Wurzel
```

Benutzung anderer Unterprogramme

Sollen Unterprogramme benutzt werden, die nicht zum ACT I - System gehören, so ist folgendes zu beachten:

1. Aufruf im Quellenprogramm

Die Anweisung im Quellenprogramm muß folgendermaßen aussehen:
sub'UPK'a"
Darin bedeuten:

sub — das Operationszeichen für den Anruf von Unterprogrammen,

UPK — ein willkürliches Kennzeichen für das betreffende Unterprogramm, das bis zu 5 Zeichen enthalten darf und nicht identisch sein darf mit irgendwelchen Operationszeichen oder mit Symbolen für die Variablen,

a — das Symbol der Variablen, die als Argument des Unterprogramms in den Akkumulator gebracht wird.

Aufgrund dieser Anweisung stellt der Compiler stets folgende Befehlsfolge im Maschinenprogramm her:

B[a] — [a] ist die Adresse für die Größe mit Symbol a.

R 0310 — In 0310 steht für alle Unterprogramme jeweils die Rückkehradresse.

U Lo — Lo ist die erste Adresse des Unterprogramms, die auch gleichzeitig immer die Anfangsadresse sein muß.

Benötigt das Unterprogramm kein Argument, so muß man trotzdem irgendetwas angeben; der Platz für a in der Anweisung muß mit irgendeinem Symbol ausgefüllt sein. Dieses Symbol wird stets in den Akkumulator gebracht, braucht aber dann im Unterprogramm nicht verarbeitet zu werden.

2. Gebrauch mehrerer Argumente

Benötigt ein Unterprogramm mehrere Eingangswerte (Argumente) a, b, c, d, so werden diese nacheinander in der Anweisung aufgeführt.

sub'UPK'a'b'c'd"

Das Maschinenprogramm lautet in diesem Fall:

B[a]
R 0310 — Das Argument a wird wiederum in den
U Lo — Akkumulator gebracht, die xZ-Befehle
xZ[b] — enthalten die Adressen der übrigen
xZ[c] — Argumente, die vom Compiler festgelegt werden.
xZ[d]

Das Unterprogramm muß in diesem Fall so aufgebaut sein, daß es sich die Adressen der xZ-Befehle aus der in 0310 enthaltenen Rückkehr-

adresse beschafft und die Rückkehradresse entsprechend erhöht bzw. die richtige erhöhte Rückkehradresse gleich in den Ausgangssprungbefehl einsetzt. Aus den xZ-Befehlen entnimmt es dann die Adressen der übrigen Argumente zur Verarbeitung.

3.Ergebnis

Eines der Argumente (im allgemeinen das letzte) ist Symbol für das Ergebnis. Das Unterprogramm muß das Abspeichern des Ergebnisses nach der im letzten xZ-Befehl stehenden Adresse selbst besorgen.

4.Ausgang aus dem Unterprogramm

Bei Verwendung eines oder keines Argumentes muß der Ausgangsbefehl i m m e r U 0310 heißen, bei mehreren Argumenten muß die Adresse berechnet werden.

5.Festlegung der Anfangsadresse Lo

Die Anfangsadresse Lo des Unterprogramms wird durch eine Anweisung mit dem Operationszeichen dim festgelegt. Die Anweisung dim'UPK'90" reserviert beispielsweise für das Unterprogramm UPK im Hauptspeicher 90 Plätze. Dieser Anweisung muß unbedingt die Anweisung dim'comp'512" vorausgehen, wenn keines der normalen Unterprogramme des ACT I (festgelegt durch spezielle Operationszeichen wie sin, sqrt, etc.) im Quellenprogramm vorkommt. Kommen solche Unterprogramme aber vor, so heißt die Anweisung dim'comp'576", und sie steht immer als erste Anweisung im Quellenprogramm. Diese Anweisung reserviert 576 Plätze im Hauptspeicher, wobei die Zählung mit Zelle 5763 beginnt und rückwärts läuft.
Damit sind die Spuren 49 - 57 komplett belegt. Die folgende Anweisung dim'UPK'90" reserviert dann die Plätze 4738 - 4863 für das Unterprogramm UPK. In diesem Fall ist die Anfangsadresse für UPK die Adresse 4738:

$$Lo(UPK) = 4738.$$

Ist aber Lo vorgegeben, so sind zwischen dem Unterprogramm UPK und der Spur 49 gerade so viele Plätze zu reservieren, daß UPK richtig liegt.

Beispiel: Lo (UPK) = 3500, UPK benötige 90 Plätze;

Das Quellenprogramm beginnt dann mit

dim'comp'576"	immer notwendige Reservierung,
dim'xyz'806"	Reservierung nötig; aber für beliebigen Zweck,
dim'UPK'90"	Reservierung für UPK von 3500 bis 3625.

Damit UPK in 3500 beginnen kann, ist eine Reservierung von insgesamt 576 + 90 + x Plätzen nötig, wobei sich x berechnet aus

$$x = (5800 - 3500)_{\text{mod } 64} - (576 + 90)$$
$$= 23 \cdot 64 - 666 = 806$$

Umgekehrt errechnet sich die Anfangsadresse aus einer gegebenen Zahl A von reservierten Plätzen zu

Spur (Lo) = 57 - G
Zelle(Lo) = 64 - R

wobei A = 64 · G + R
G = ganze Zahl
Rest R = ganz, kleiner als 64

z.B. A = 666; G = 10; R = 26; Lo = 4738

Kapitel V

Ergänzungen

Speicherbelegung

Es ist noch anzugeben, wo die Operationszeichen gespeichert sind und wie sie verschlüsselt werden.

Die Operationszeichen werden in vier Gruppen eingeteilt: Zeichen bestehend aus 1 oder 2 Bits, 3-Bit-, 4-Bit- und 5-Bit-Zeichen. Jede Gruppe wird durch ein Schlüsselwort W0000000' abgeschlossen. Diese Operationsliste wird auf Spur 36 gespeichert.

Auf Spur 37 stehen Schlüsselworte in den den Operationszeichen der Spur 36 entsprechenden Stellen in dem folgenden Format:

Bit-Nummer (von links)	Inhalt
1 - 3	4 Nullen
4, 5	Vorrang (0, 1, 2 oder 3)
6	Operationsanzeiger
7	linker Operand: 0 = nein 1 = ja
8 - 11	Anzahl der zu erzeugenden Befehle, um eins vermindert
12 - 18	relative Adresse des ersten zu erzeugenden Befehls in Spur 38. Steht in Stelle 12 eine 0, so ist Spur 38 gemeint, steht dort eine 1, so ist Spur 39 gemeint. Die 6 Bits in den Stellen 13 - 18 geben die Zellennummer an, in der der erste Befehl für die betr. Operation steht.
19 - 30	Transfer-Adresse, falls Sprung auf ein Unterprogramm notwandig ist.

Für iter und sub wird in den entsprechenden Zellen der Spur 37 das gleiche gespeichert wie in Spur 36; das Format der Kennworte in 37 ist in diesen beiden Fällen nicht gültig.

Auf den Spuren 38 und 39 stehen:

Die vom ACT I zu erzeugenden tatsächlichen Maschinenbefehle mit Adressen 0000 (außer bei Fest- und Sprungadressen) und einige Bits vor dem Befehlskennzeichen Ø zur Charakterisierung der Art des Operanden. Diese kennzeichnenden Bits sind wie folgt aufgebaut:

Bit-Nummer Vorzeichen, 7 8 9 10	Operand	gesamtes Wort in hexadezimal
0 0 0 0 0	linker Op.	Ø 0 0 0 0
0 1 0 0 0	rechter Op.	1 0 Ø 0 0 0 0
0 1 1 0 0	Zwischen-Op.	1 8 Ø 0 0 0 0
0 1 1 1 1	Festadresse	1 Q Ø T_1 T_2 S_1 S_2
0 1 1 1 0	Sprungadresse	1 J Ø T_1 T_2 S_1 S_2
1 0 0 0 0	Transferadresse	8 0 0 Ø 0 0 0 0

Das Zeichen ∅ steht hier anstelle der jeweiligen Befehlsbuchstaben. Für die speziellen Anweisungen iter und sub wird nichts in die Spuren 38 und 39 geschrieben.

Die Konstanten werden ab Zelle 6200 gespeichert.

Für jeden Index werden drei Speicherstellen benötigt. Wird kein Index definiert, beginnt das Objektprogramm in 0322, werden drei Indizes benutzt, beginnt es bei 0331.

Benutzung von Maschinenbefehlen

Die normalen Maschinenbefehle des LGP-21 können innerhalb eines Quellenprogramms ebenfalls benutzt werden. Sie sind dann mit symbolischen Adressen zu versehen, d.h. anstelle der Adressen stehen die im Programm auch sonst benutzten Symbole. Die in diesem Fall zu benutzenden Operationszeichen und ihre Bedeutung sind in der folgenden Tabelle zusammengestellt.

Operationszeichen	Bedeutung
bring a	Ersetze Akkumulatorinhalt durch Inhalt der dem Symbol a zugeordneten Speicherzelle.
add a	Addiere a zum Akkumulatorinhalt.
subtr a	Subtrahiere a vom Akkumulatorinhalt.
mult a	Multipliziere a mit Akkumulatorinhalt und halte obere Hälfte.
nmult a	Multipliziere a mit Akkumulatorinhalt und halte untere Hälfte.
div a	Dividiere Akkumulatorinhalt durch a.
extrt a	Bilde das logische Produkt aus Akkumulatorinhalt und a.
hold a	Speichere Akkumulatorinhalt in der a zugeordneten Speicherzelle.
clear a	Wie vor, aber anschließend Akkumulator löschen.
stadd a	Schreibe den Adressenteil des Akkumulatorinhalts in die dem Symbol a zugeordnete Speicherzelle.
rdhex a	Lies den hexadezimalen Wert von a in den Akkumulator.
use a	Unbedingter Sprung, als nächster Befehl ist der mit der Adresse von a auszuführen.
trn a	Bedingter Sprung, Sprung zur Adresse von a, wenn der Akkumulator-Inhalt negativ ist. Im anderen Fall wird dieser Befehl übergangen.
ret a	Speichere den um 1 erhöhten Inhalt des Zählregisters im Adressteil des Wortes, das unter der Adresse von a steht. Das Zählregister enthält normalerweise die Adresse des nächsten Befehls. Die Operation ret wird dazu benutzt, die Rückkehradresse in ein Unterprogramm einzusetzen.

In den Befehlen stadd', use', trn' und ret' kann statt eines Symbols auch ein Anweisungskennzeichen, z.B. S17, stehen.

Kapitel VI

Beispiele

Dies Kapitel enthält einige einfache ACT I - Programme. Jedem Programm wird eine kurze Beschreibung seiner Funkion vorangestellt.

Beispiel 1

Bestimmung des kleinsten Elementes einer Zahlenfolge von nichtnegativen Elementen. Als Ein- und Ausgabegerät wird der Flexowriter benutzt. Es ist also die Änderung am Schluß des UPA - Streifens einzulesen. Die Daten werden als Festkommazahlen eingegeben und in Gleitkommazahlen umgewandelt. Gerechnet wird im Gleitkomma. Ausgegeben werden ein Festkommawert, der angibt, an wievielter Stelle das Minimum steht und dahinter das Minimum in Gleitkommaform.

QUELLENPROGRAMM

```
dim'comp'512''
dim'a'200''
index'i'j'k''
s0'read'n''n'-'1';'nl''
1';'i''
s1'read'hilf''2'flo'hilf';'a'i''
iter'i'1'n's1''
1';'i';'j''
s2'i'+'1';'k''a'j'f-'a'k'trn's3''
i'+'1';'j''
s3'iter'i'1'nl's2''
a'j';'min''
cr''cr''0'print'j''tab''fprt'min''
stop''use's0'''
```

Wenn man das Programm compiliert, erhaelt man den folgenden Ausdruck :

```
dim'comp'512''
dim'a'200''
index'i'j'k''
s0'read'n''n'-'1';'nl''
1';'i''
s1'read'hilf''2'flo'hilf';'a'i''
iter'i'1'n'sl''
1';'i';'j''
s2'i'+'1';'k''a'j'f-'a'k'trn's3''
i'+'1';'j''
s3'iter'i'1'nl's2''
a'j';'min''
cr''cr''0'print'j''tab''fprt'min''
stop''use's0'''


i 0331 f 0438

s000  0331
s001  0339
s002  0359
s003  0412


Ausschrift des Objektprogramms (hexadezimal)


v40k0300'
1033j'f0348'300'f0314'4'f0314''20000000'
4''f06g0''''2'q033j'
j0308'f0300'q0310'20314'10318'f0308'j0318'13j00'
f033j'j0318'13j04'f033j'j0318'13j08'f033j'30328'
f3g00'j3j0j'13j0j'w3q00'j3j10'13q00'j3j00'30328'
f3g00'j3j14'13q04'j0330'13j14'30328'f3394'3033j'
f0358'j2qq0'13j00'q3q00'j3j00'w3j0j'w0338'g039j'
13q00'j3j00'j3j04'13j00'q3q00'j3j08'3033j'f0364'
95k46w86'
v27k0400'
12qq0'j0330'3033j'f0370'12qq0'30328'f3500'j3w00'
g0430'13j00'q3q00'j3j04'13j00'q3q00'j3j00'w3j10'
w0338'g03qj'3033j'f0364'12qq0'j3j1j'13g98'80200'
13g98'80200'13q08'j0330'13j04'30328'f3f00'13g94'
80200'13j1j'30328'f3900''f037j''
7387w6g0'

v03k3q00'
4'8''
703k3q0j'
.0000000'
```

Datenbeispiel

```
12'
312'567'458'121'468'517'
418'9734'1855'397'588'735'
```

Eingabe

```
.0000331'       Sprung an den Anfang des Programms
12'
312'567'458'121'468'517'
418'9734'1855'397'588'735'
```

Ausgabe

```
      4.                    .1210000  01
```

Beispiel 2

Bildung des inneren Produkts zweier Vektoren. Als Eingabegeraet dient der Schnelleser, ausgegeben wird ueber Flexowriter.Das Programm arbeitet im Festkomma. Anwendung der Operationen aread' und aprt'.

QUELLENPROGRAMM

```
dim'comp'512''
dim'a'50''dim'b'50''dim't'9''
index'i''
s0'cr''l';'i''
s1'aread't'i''
iter'i'l'8's1''
l';'i''
s2'aprt't'i''
iter'i'l'5's2''
read'n''0'print'n''cr''cr''
l';'i';'j''
s3'read'a'i''0'print'a'i''
when'j'equal'8'trn's4''
j'+'l';'j''use's5''
s4'l';'j''cr''
s5'iter'i'l'n's3''
cr''cr''

l';'i';'j''
s6'read'b'i''0'print'b'i''
when'j'equal'8'trn's7''
j'+'l';'j''use's8''
```

```
s7'l';'j''cr''
s8'iter'i'l'n's6''cr''cr''

l';'i''0';'c''
s9'c'+'a'i'x'b'i';'c''
iter'i'l'n's9''

aprt't'6''aprt't'7''aprt't'8''
0'print'c''
stop''
use's0'''
```

Zum Schluss der Uebersetzung werden die folgenden Adressen ausgeschrieben:

```
i 0325 f 0351

s000  0325
s001  0329
s002  0343
s003  0405
s004  0426
s005  0430
s006  0443
s007  0500
s008  0504
s009  0518
```

Daten

```
Inn'eres 'Pro'dukt 'n ='Loe'sung ':'
12'
5'7'8'3'4'1'0'2'
0'4'9'0'
1'5'2'7'0'0'1'8'
3'0'4'1'
```

Ausgabe

```
Inneres Produkt n =        12.

   5.    7.    8.    3.    4.    1.     .    2.
    .    4.    9.     .

   1.    5.    2.    7.     .     .    1.    8.
   3.     .    4.    1.

Loesung :     129.
```

Beispiel 3

Berechnung des Gauss'schen Fehlerintegrals

$$\phi(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt, \qquad 0 \leq x \leq \infty$$

Naeherungsformel:

$$\phi^*(x) = 1 - \frac{1}{[1 + a_1x + a_2x^2 + a_3x^3 + \dots + a_6x^6]^{16}}$$

a_1 = .0705230784 a_4 = .0001520143

a_2 = .0422820123 a_5 = .0002765672

a_3 = .0092705272 a_6 = .0000430638

Als Ein- und Ausgabegeraet dient der Flexowriter. Das Programm arbeitet im Gleitkomma. Die Daten werden im Festkomma eingegeben. Beim Umwandeln wird das Komma um drei Stellen nach links verschoben.

QUELLENPROGRAMM

```
dim'comp'1400''
dim'a'7''
s0'9'flo','70523'x'1000'+'78'.';'a'1''
9'flo','42282'x'1000'+'12'.';'a'2''
10'flo','92705'x'1000'+'272'.';'a'3''
10'flo','15201'x'100'+'43'.';'a'4''
10'flo','27656'x'100'+'72'.';'a'5''
10'flo','43063'x'10'+'8'.';'a'6''
0'flo'1';'eins''0'flo'16';'sechz''

s1'read'yanf''read'delty''read'yend''3'flo'yend';'yend''
yanf';'y''3'flo'y';'yf''
s2'cr''3'print'y''tab''
,',',',','a'6'fx'yf'f+'a'5'.'fx'yf'f+'a'4'.'fx'yf'f+'a'3'.'
fx'yf'f+'a'2'.'fx'yf'f+'a'1'.'fx'yf'f+'eins';'zw''
eins'f-'eins'f/','eexp','ln'zw'fx'sechz'.'.';'phi''
fprt'phi''
y'+'delty';'y''3'flo'y';'yf''
,'yf'f-'yend'.'trn's2''
stop''use's0'''
```

Zum Schluss der Uebersetzung werden die folgenden Adressen ausgeschrieben :

```
i 0322 f 0633

s000  0322
s001  0436
s002  0459
```

Dateneingabe

```
1000'5'1055'
```

Ausgabe

```
1.000          .8427027  00
1.005          .8447674  00
1.010          .8468114  00
1.015          .8488356  00
1.020          .8508390  00
1.025          .8528230  00
1.030          .8547850  00
1.035          .8567277  00
1.040          .8586514  00
1.045          .8605533  00
1.050          .8624374  00
```

Beispiel 4

Loesung von linearen Gleichungssystemen nach Gauss-Seidel. Als Ein- und Ausgabegeraet dient der Flexowriter.Als Eingabedaten werden verlangt :

1. Zahl der Gleichungen
2. Genauigkeitsschranke
3. Die Matrix A, zeilenweise
4. Die rechte Seite

Das Programm arbeitet im Gleitkomma.

QUELLENPROGRAMM

```
dim'comp'512''
dim'a'901''
dim'b'31''
dim'ex'31''
dim'y'31''
index'i'j'k'p'q''
s0'read'n''finp'tol''n'x'n';'nn''
l';'i''
s1'finp'a'i''
iter'i'l'nn's1''
l';'i''
s2'finp'b'i''
iter'i'l'n's2''
l';'i''
s3'0';'ex'i';'y'i''
iter'i'l'n's3''
s4'l';'i''
s5'b'i';'y'i''
l';'j''
s6'when'i'less'j'trn's7''
when'i'grt'j'trn's7''use's8''
s7',','i'-'l'.'x'n'+'j'.';'p''
y'i'f-'a'p'fx'y'j';'y'i''
s8'iter'j'l'n's6''
,','i'-'l'.'x'n'+'i'.';'q''
y'i'f/'a'q';'y'i''
iter'i'l'n's5''
l';'k''
s9','tol'f-'fabs','y'k'f-'ex'k'.'.'trn's10''
use's12''
s10'l';'i''
s11'y'i';'ex'i''
iter'i'l'n's11''
use's4''
s12'iter'k'l'n's9''
l';'i''cr''
s13'fprt'ex'i''cr''
iter'i'l'n's13''
stop''use's0'''
```

Zum Schluss des Compilierens werden die folgenden Adressen ausgeschrieben :

```
i 0337 f 0629

s000  0337
s001  0349
s002  0362
s003  0411
s004  0424
s005  0426
s006  0434
s007  0441
s008  0503
s009  0535
s010  0553
s011  0555
s012  0604
s013  0614
```

Datenbeispiel - Eingabe

```
.0000337'
3'5000000'-05'
1200000'02'-2000000'01'3000000'01'
-1000000'01'8000000'01'-2000000'01'
-1000000'01'3000000'01'1200000'02'

1800000'02'-3200000'02'6000000'01'
```

Ergebnis

```
.5450038  00          (x1)
-.3572257  01         (x2)
.1438482  01          (x3)
```

Ergänzung: Verbesserung von Objektprogrammen

In dem folgenden Abschnitt wird beschrieben, wie man einen Fehler im Objektprogramm verbessern kann. In einem Objektprogramm sei aufgrund einer falschen Anweisung im Quellenprogramm eine falsche Befehlsfolge entstanden. Diese Befehlsfolge ist in der Ausschrift des Objektprogramms zu suchen und mit Hilfe der Tabelle der Operationen (s.Anhang) zu verbessern.

Beispiel

In Beispiel 1, Seite 29, sei die iter-Anweisung hinter S1 falsch.

Falsche Anweisung: iter' i' 1' n1' S2"

richtige Anweisung: iter' i' 1' n' S1"

falsche Befehlsfolge: (hexadezimal)	13j00'	q3q00'	j3j00'	w3j10'	w0338'	g03qj'
nach Anhang: (dezimal)	B(i)	A(1)	H(i)	S(n1)	S0314	T(S2)
verb.Befehlsfolge:	B(i)	A(1)	H(i)	S(n)	S0314	T(S1)
hexadezimal:	13j00'	q3q00'	j3j00'	w3j0j'	w0338'	g039j'
dezimal:	B6000'	A6200'	H6000'	S6003'	S0314'	T0339'

Da das Objektprogramm spurenweise, immer acht Befehle in einer Zeile, ausgeschrieben wird, kann man einfach durch Abzählen die Adressen der falschen Befehle feststellen. In unserem Beispiel sind das die Adressen 0353 und 0355. Die Verbesserung des Objektprogramms kann mit Hilfe des Eingabeprogramms J1-10.1 mit Startfill vorgenommen oder hexadezimal ausgeführt werden.

a) Verbesserung mit Startfill (vergl. Programmierungshandbuch)

;0000353' S6003'

;0000355' T0339'

Sind mehrere Verbesserungen vorzunehmen, so stellt man sich einen Korrekturstreifen her.

b) Verbesserung von Hand

Es sind die Befehle w3j0j und g039j von Hand zu speichern, siehe Bedienungsanleitung LGP-21, Seite 6, Speichereingabe hexadezimaler Worte von Hand.

Anhang

Tabelle der Operationen

Befehle im Quellenprogramm	Bedeutung	Maschinenbefehle
a';'b'	Setze b gleich a	B (a) H (b)
a'+'b'	Addiere b zu a	B (a) A (b)
a'-'b'	Subtrahiere b von a	B (a) S (b)
a'x'b'	Multipliziere b mit a	B (a) N (b) M 0307 (1 bei q29)
a'/'b'	Dividiere b durch a	B 0308 (1 bei q29) D (b) M (a)
when'a'less'b' trn's10'	Wenn a kleiner als b ist, erfolgt ein Sprung zur Anweisung s10	B (a) S (b) T (s10)
when'a'grt'b' trn's10'	Wenn a größer als b ist, erfolgt ein Sprung zur Anweisung s10	B (b) S (a) T (s10)
when 'a'equal'b' trn's10'	Wenn a gleich b ist, erfolgt Sprung zur Anweisung s10	α B (a) α +1 S (b) α +2 T (α+5) α +3 S 0314 (1 bei q30) α +4 T (s10)
abs'a'	Bilde den Absolutwert von a (Festkomma-Operation)	α B (a) α +1 T (α+3) α +2 U (α+5) α +3 C 0306 α +4 S 0306
Read'a'	Lies "a" als eine ganze Zahl ein und speichere sie bei q 29 nach a	R 0310 U 5900 H (a)
n'print'a'	Drucke "a" mit n Dezimalstellen rechts vom Komma	B (n) H 0312 B (a) R 0310 U 5800

Befehle im Quellenprogramm	Bedeutung	Maschinenbefehle
stop"	Stop	Z 0000
a'f+'b'	Addiere b zu a im Gleitkomma	B (a) H 0312 B (b) R 0310 U 5200
a'f-'b'	Subtrahiere b von a im Gleitkomma	B (a) H 0312 B (b) R 0310 U 5300
a'fx'b'	Multipliziere b mit a im Gleitkomma	B (a) H 0312 B (b) R 0310 U 5400
a'f/'b'	Dividiere b durch a im Gleitkomma	B (a) H 0312 B (b) R 0310 U 5328
finp'a'	Der Wert von a wird als Gleitkommazahl eingelesen	R 0310 U 5502 H (a)
finp'a'i'	Lies die Zahl a und speichere sie im Gleitkommaformat auf dem Speicherplatz, der für das i-te a reserviert wurde	R 0310 U 5502 R 0315 U 0322 H (a)
fprt'a'	Der Wert von a wird als Gleitkommazahl gedruckt	B (a) R 0310 U 5700
fprt'a'i'	Bringe den Wert der i-ten Zelle des Bereiches a und drücke ihn im Gleitkommaformat	R 0315 U 0322 B (a) R 0310 U 5700
n'flo'a'	Wandle die Festkommazahl a mit n Stellen hinter dem Komma um in eine Gleitkommazahl (nicht benutzt während Ein- und Ausgabe)	B (n) H 0312 B (a) R 0310 U 5137
n'unflo'a'	Versetze das Komma der Gleitkommazahl a um n Stellen nach rechts und wandle diese Zahl in Festkommaform um. Die erzeugte Festkommazahl steht bei q = 29	B (n) H 0312 B (a) R 0310 U 5100

Befehle im Quellenprogramm	Bedeutung	Maschinenbefehle
iter'i'1'n's10"	Erhöhe den Index i um 1 und springe zurück zur Anweisung s10; falls aber i größer als n wird, gehe weiter	B (i) A (1) H (i) S (n) S 0314 (1 bei q 30) T (s10)
dim'a'11"	Reserviere 11 Speicherplätze für die Größen a_i (der erste Platz wird für a_o benutzt)	zum Programm kommt nichts hinzu
index'i"	Definiere i als Index 0322 0323 0324	H 0306 B (i) U 0315
sub'tanh'xy'txy"	Springe in das Unterprogramm zur Berechnung des tanh von xy, speichere das Resultat in txy. Für das Unterprogramm tanh muß vorher genügend viel Speicherplatz durch eine dim'sub' Anweisung geschaffen worden sein. Tanh ist auf dem Standard-Unterprogrammstreifen UPB-FK oder UPB-GK nicht enthalten.	α B (xy) α +1 R 0310 α +2 U xxxx α +3 Z (txy)
sub'asec'a'b'c"	Berechne arcsec a/b, speichere das Resultat in c. Bei diesem Unterprogramm sind 2 Eingänge: a und b	α B (a) α +1 R 0310 α +2 U xxxx α +3 Z (b) α +4 Z (c)
ret's10'	Setze die Rücksprungadresse in Anweisung s10 ein. Die Anweisung s10 muß die Form: s10' use' s00" haben	R (s10)
use's10'	Unbedingter Sprung nach s10	U (s10)
cr"	Wagenrücklauf	xB 5938 xP 0200
sin'a'	Berechne den Sinus von a	B (a) R 0310 U 4908
log'a'	Berechne den Brigg'schen Logarithmus der Gleitkommazahl a	B (a) R 0310 U 4954
n'log10'a'	Berechne den Brigg'schen Logarithmus der Festkommazahl a. Das Ergebnis hat n Ziffern links vom Dezimalpunkt	B (n) H 0312 R 0310 U 4958
tab"	Tabulator	xB 5937 xP 0200

Befehle im Quellenprogramm	Bedeutung	Maschinenbefehle
rdhex'a'	Lies ein hexadezimales Symbol	xC 0306 80xI 0000 H[ ]
aread'a'	Lies ein alphanumerisches Symbol	C 0306 I 0000 N 0308(1 bei q 29) H (a)
aprt'a'	Drucke ein alphanumerisches Symbol	B (a) R 0310 U 5145
fabs'a'	Bilde den Absolutbetrag von a (Gleitkommazahl)	B (a) R 0310 T 5632 H[ ]
PS4'	Bedingter Sprungbefehl, abhängig vom Zustand der PS-4-Taste	xZ 0400 U[ ]
PS8'	Bedingter Sprungbefehl (PS-8-Taste)	xZ 0800 U[ ]
PS16'	Bedingter Sprungbefehl (PS-16-Taste)	xZ 1600 U[ ]
PS32'	Bedingter Sprungbefehl (PS-32-Taste)	xZ 3200 U[ ]