JUGEND FORSCHT 1977

L a n d e s w e t t b e w e r b

MAT./ INF.

Thema: Compiler und Interpreter für eine verbesserte BASIC-Version

Teilnehmer: BANHART, John

Die Aufgabe der Arbeit war es, für einen Kleincomputer ein Programm zu entwickeln, das es dem Benützer erlaubt, die Programmiersprache BASIC an dem betreffenden Computer einzusetzen.

BASIC - Versionen, wie sie an großen EDV - Anlagen z.B.: für Terminalbetrieb an Schulen benützt werden, sind für kleine Computer nicht geeignet.

Deshalb wurde durch Kürzung gängiger BASIC - Versionen eine BASIC - Version entwickelt, die einerseits der beschränkten Speicherkapazität eines Kleincomputers Rechnung trägt, andererseits nicht auf das Tisch- oder Taschenrechnerniveau absinkt. Es wurden zudem noch in bescheidenem Umfang Elemente anderer Sprachen aufgenommen, so daß der Ausdruck "verbesserte BASIC-Version" gerechtfertigt ist. Die Beschreibung der verbesserten BASIC-Version wird in den Grundzügen in Abschnitt 2.2 durchgeführt und bildet einen Hauptteil der Arbeit.

Der zweite Teil der Arbeit bestand darin, für den schuleigenen "Eurocomp LGP-30" -Computer ein Übersetzerprogramm für die verbesserte BASIC-Version zu schaffen.

Dabei mußten abermals die Eigenschaften von Kleincomputern berücksichtigt werden: beschränkte Speicherkapazität und oft niedrige Rechengeschwindigkeit. Nach einer Untersuchung verschiedener Übersetzungssysteme stellte es sich heraus, daß es am günstigsten ist, Compilersystem und Interpretiersystem zu kombinieren. Der Compiler ist gewissermaßen ein Vorübersetzer während der Programmeingabe, der die BASIC-Anweisungen in eine halbmaschinensprachliche Zwischenform übersetzt. Der Interpreter tritt während des Programmdurchlaufs in Funktion und führt die vorübersetzten BASIC-Anweisungen aus. Eine ausführliche Erklärung ist in Abschnitt 2.3.3 zu finden.

Die Erkenntnisse, die gewonnen wurden, dürften auch für andere Kleincomputer mit ähnlichen Eigenschaften Gültigkeit besitzen.

Die Beschreibung des Programms beschränkt sich auf die Teile, die nicht nur speziell für den "Eurocomp LGP-30" gelten, sondern von allgemeinem Interesse sind. Außerdem kann in diesem beschränkten Rahmen keine vollständige Beschreibung aller Funktionen erfolgen, sondern es werden nur Kernpunkte herausgegriffen.

B A S I C - SYSTEM FÜR EINEN KLEINCOMPUTER

(Compiler und Interpreter für eine verbesserte BASIC - Version)

JUGEND FORSCHT 1977

JOHN BANHART

KURZE BESCHREIBUNG DER ARBEIT

Die Aufgabe der Arbeit war es , für einen Kleincomputer ein Pro - gramm zu entwickeln , das es dem Benützer erlaubt , die Pro - grammiersprache BASIC an dem betreffenden Computer einzusetzen .

BASIC - Versionen , wie sie an großen EDV - Anlagen z.B: für Terminalbetrieb an Schulen benützt werden , sind für kleine Computer nicht geeignet .
Deshalb wurde durch Kürzung gängiger BASIC - Versionen eine BASIC - Version entwickelt , die einerseits der beschränkten Speicherkapazität eines Kleincomputers Rechnung trägt , anderer - seits nicht auf das Tisch - oder Taschenrechnerniveau absinkt .
Es wurden zudem noch in bescheidenem Umfang Elemente anderer Sprachen aufgenommen , so daß der Ausdruck "verbesserte BASIC - Version" gerechtfertigt ist . Die Beschreibung der verbesserten BASIC - Version wird in den Grundzügen in Abschnitt 2.2 durch - geführt und bildet einen Hauptteil der Arbeit .
Der zweite Teil der Arbeit bestand darin , für den schuleigenen "Eurocomp LGP-30 "-Computer ein Übersetzerprogramm für die ver - besserte BASIC - Version zu schaffen .
Dabei mußten abermals die Eigenschaften von Kleincomputern be - rücksichtigt werden : beschränkte Speicherkapazität und oft niedrige Rechengeschwindigkeit . Nach einer Untersuchung ver - schiedener Übersetzungssysteme stellte es sich heraus , daß es am günstigsten ist , Compilersystem und Interpretiersystem zu kombinieren . Der Compiler ist gewissermaßen ein Vorübersetzer während der Programmeingabe , der die BASIC - Anweisungen in eine halbmaschinensprachliche Zwischenform übersetzt . Der Inter - preter tritt während des Programmdurchlaufs in Funktion und führt die vorübersetzten BASIC - Anweisungen aus . Eine aus - führliche Erklärung ist in Abschnitt 2.3.3 zu finden .
Die Erkenntnisse , die gewonnen wurden , dürften auch für andere Kleincomputer mit ähnlichen Eigenschaften Gültigkeit besitzen .

Die Beschreibung des Programms beschränkt sich auf die Teile , die nicht nur speziell für den "Eurocomp LGP-30" gelten , sondern von allgemeinem Interesse sind . Außerdem kann in diesem beschränkten Rahmen keine vollständige Beschreibung aller Funktionen erfolgen , sondern es werden nur Kernpunkte heraus - gegriffen .

B A S I C - SYSTEM FÜR EINEN KLEINCOMPUTER

(Compiler und Interpreter für eine verbesserte BASIC - Version)

1 EINLEITUNG

Heute werden Programme bei den meisten EDV - Anlagen nicht mehr in der Maschinensprache , sondern in höheren Programmiersprachen abgefasst . Es existieren viele Programmiersprachen , die für spezielle Probleme bestimmt sind (z.B: naturwissenschaftliche Probleme (z.B: Fortran) , kaufmännische Probleme (z.B: Cobol)). Programme in höheren Programmiersprachen können vom Computer nicht unmittelbar (wie die Maschinensprache) verarbeitet werden , sondern müssen vorher von einem speziellen Übersetzerprogramm bearbeitet werden . Die Aufstellung eines solchen Programmes für einen Kleincomputer , in diesem Fall speziell für den schuleigenen " Eurocomp LGP - 30 " , war der eine Teil der Ar - beit . Als Programmiersprache wurde aus folgenden Gründen BASIC gewählt :

- BASIC ist allgemein verwendbar und sehr leicht erlernbar .
- Schon vorhandene Programme in BASIC können ohne große Um - formungen wiederverwendet werden .
- Ein BASIC - System erfordert wegen der strukturellen Ein - fachheit der Sprache relativ wenig Speicherplatz und ist deshalb für Computer mit wenig verfügbarem Speicherplatz besonders geeignet .

Es stellte sich heraus , daß bereits vorhandene BASIC - Versionen (Als Vergleichsbasis wird die , von IBM 1973 an den Schulen eingesetzte BASIC - Version benützt) für Kleincomputer nicht geeignet sind :

- Sie beanspruchen zuviel Speicherplatz
- Sie enthalten Funktionen , die entbehrlich sind (siehe 2.2)
- Einige Funktionen , die in anderen Programmiersprachen ent - halten sind , fehlen .

Deshalb bestand der andere Teil der Aufgabe darin , eine , der beschränkten Speicherkapazität eines Kleincomputers (beim LGP - 30 : 4k Wörter) und den Bedürfnissen der Schule angepasste BASIC - Version zu entwickeln .

In der folgenden Arbeitsbeschreibung kann keine vollständige Beschreibung des BASIC - Systems oder eine ausführliche Definition von BASIC erfolgen , sondern nur ein Überblick über die wichtigsten Ergebnisse .

2 ARBEITSBESCHREIBUNG

2.1 Vorgehen

Die Entwicklung vom ersten Entwurf bis zum fertigen BASIC - System vollzog sich im Wesentlichen in 8 Schritten :

1. Auswahl der Sprache . Aus den , in der Einleitung aufgeführten Gründen wurde BASIC gewählt .
2. Auswahl des Übersetzungssystems . (siehe 2.3.2)
3. Abschätzung des benötigten Speicherplatzes . Hier zeigte es sich , daß schon vorhandene BASIC - Versionen nicht geeignet sind . Kürzungen und Modifikationen wurden vorgenommen . Einige neue Funktionen konnten hinzugefügt werden .
4. Aufstellung der Flussdiagramme für einige Programmteile .
5. Abermalige Speicherplatzabschätzung und weitere gering - fügige Modifikationen . Endgültige Fassung von BASIC .
6. Umsetzung des Programmstamms (siehe 2.3.3) in die Maschinen - sprache und Erprobung am Computer .
7. Umsetzung der einzelnen Funktionen in die Maschinensprache und Erprobung .
8. Enderprobung mit größeren BASIC - Programmen .

2.2 Beschreibung der verbesserten BASIC - Version

2.2.1 Allgemeines

Bei der Entwicklung der neuen BASIC - Version mußten folgende Punkte beachtet werden :

- Es sollten möglichst wenig neue Funktionen eingeführt werden , um den Charakter der Sprache nicht zu sehr zu verändern . Neue Funktionen sollten möglichst durch Erweiterung alter Funktionen erzeugt werden .
- Der Speicherplatzbedarf des Systems durfte ein bestimmtes Maß nicht überschreiten .
- Es sollte trotz dieser Beschränkung ein Maximum an Funktionen zur Verfügung stehen .

2.2.2 Kürzungen

Folgende Funktionen wurden weggelassen :

- Dateienverarbeitung : Mußte entfallen , da der Speicher - platz dazu bei Kleincomputern nicht zu Verfügung steht .
- Matrizenverarbeitung: Wurde zugunsten anderer Funktionen weggelassen.

- DEF - Anweisungen (arithmetische Ausdrücke mit variablem Para - meter)wurden durch die erweiterte Unterprogrammtechnik überflüssig.
- Verschachtelung von Funktionen wurde eingeschränkt .

2.2.3 Verbesserungen

Durch die , in Abschnitt 2.2.2 beschriebenen Maßnahmen wurde so viel Speicherplatz eingespart , daß folgende Funktionen ergänzt werden konnten :

2.2.3.1 Prozeduren (Unterprogramme)

Mit der Anweisung GOSUB XXX wird eine bei XXX anfangende Prozedur aufgerufen d.h: nach XXX verzweigt (Unterprogrammsprung) . Die An - weisung RETURN beendet die Prozedur und es erfolgt eine Ver - zweigung ins Hauptprogramm zurück .
Diese Prozedurtechnik wurde in zwei Punkten verbessert :

a) Prozeduren mit Parameterübergabe
Bei diesen Prozeduren wird die Prozedur auf eine Anzahl von Daten (aktuelle Parameter) angewandt , die beim Prozeduraufruf einer gleichen Anzahl von prozedurinternen (formalen) Parametern zugeordnet werden .

z.B:
```
10    GOSUB 100 (2,3,4)
...
100   PROCEDURE X,Y,Z
110   LET Q=SQR(X+Y+Z)
120   PRINT Q
130   RETURN
```

Die Ausführung des Programmes bewirkt den Druck der Zahl 3

Hier wird die Prozedur "Druck der Quadratwurzel aus der Summe dreier Zahlen 100-130" auf die Parameter "2,3,4" angewandt , die beim Aufruf der Prozedur den , durch die PROCEDURE-Anweisung erklärten Parametern zugeordnet werden .
Durch diese Verbesserung wird die DEF - Anweisung überflüssig , die eine Prozedur mit einem Parameter ist .

b) Kleinprozeduren ohne Parameterübergabe
Kleinprozeduren bestehen aus einer Anweisung und werden mit der neuen Anweisung , USE , aufgerufen .

z.B:
```
10    LET A=B+SQR(C-D)+SIN(E+F)
...
40    USE 10
```

Die USE - Anweisung bewirkt , daß die Anweisung in Zeile 10 zeit - weise als Prozedur interpretiert wird . Der Vorteil besteht da - rin , daß die Anweisung in 10 eine ganz "normale" Anweisung ist , die auch ohne Prozeduraufruf durchlaufen werden kann .

2.2.3.2 Programmverzweigungen

BASIC - Programme bestehen oft zu einem relativ großen Teil aus Programmverzweigungen (GOTO - Anweisungen) , was dazu führt , daß die Programme lang und unübersichtlich werden . Daran läßt sich wenig ändern , da die Verzweigungstechnik zu den Grundelementen von BASIC zahlt .
Es wurden dennoch zwei Verbesserungen durchgeführt , die in manchen Fällen sehr nützlich sind :

a) Indirekte Programmverzweigungen
Bei einer indirekten Programmverzweigung steht an Stelle der Verzweigungsadresse eine Variable . Der Wert der Variablen wird beim Programmdurchlauf als Verzweigungsadresse inter - pretiert . Die Verzweigungsanweisung kann mehrere Male mit Verschiedenen Werten der Variablen durchlaufen werden .
z.B: 10 IF A=B GOTO X mit x=30 bewirkt Verzweigung mach 30
Diese Verbesserung führt manchmal zu erheblichen Verein - fachungen eines Programmes (siehe Anhang II) .

b) Bedingte Verzweigungen mit zwei Adressen
Wenn bei einer normalen IF - Anweisung die Bedingung nicht er - füllt ist , wird die nächste Anweisung ausgeführt . Hier wurde die Möglichkeit geschaffen , in einer bedingten Verzweigungs - anweisung eine Adresse für nicht erfüllte Bedingung anzugeben .
z.B: 100 IF A=B GOTO 150 ELSE 200
Diese Anweisung ersetzt folgendes Anweisungspaar :

```
100  IF A=B GOTO 150
101  GOTO       200
```

2.2.3.3 Sonstige Verbesserungen

a) DATA - Anweisungen
Die DATA - Anweisung wurde so umgewandelt , daß in ihr nicht nur Konstanten (wie bisher) , sondern auch Variablen enthalten sein können . Wie folgendes Beispiel zeigt , bringt diese Ver - besserung erhebliche Platzvorteile mit sich :

```
10  LET U=A              10  DATA A,B,C,D,E
20  LET W=B              20  READ U,W,X,Y,Z
30  LET X=C
40  LET Y=D
50  LET Z=E
```

Beide Programmausschnitte haben die gleiche Funktion!

b) Es wurden noch eine Reihe von Verbesserungen , die die Pro - grammierarbeit erleichtern durchgeführt :
z.B: - längere Variablennamen (bis zu 5 Zeichen)
- beliebig lange Anweisungen (z.B: PRINT - Anweisungen)

2.3 Praktische Realisierung auf einem Kleincomputer

In diesem Abschnitt wird die praktische Realisierung der Ver - besserten BASIC - Version beschrieben . Dabei wird nicht auf programmiertechnische Details eingegangen , die nur für Kenner des verwendeten Computers interessant sind , sondern es wird nur der Grundaufbau beschrieben und an einem Beispiel die Bearbeitung einer Anweisung im Einzelnen demonstriert . Die Erkenntnisse , die am LGP - 30 gewonnen wurden , sind auf alle anderen Kleincomputer mit ähnlicher Kapazität und Rechengeschwindigkeit übertragbar .

2.3.1 Kurze Beschreibung des " Eurocomp LGP - 30 "

Speichermöglichkeiten : interne Speicherung : magnetischer Trommel - speicher mit 4k Wörtern zu 31 Bit . Externe Speicherung : Loch - streifen .
Maschinensprache : Der LGP - 30 hat eine Einadresslogik und eine 31 bit Festkommaarithmetik . Ein Operand ist immer ein "Akkumulator - speicher" der andere ein Speicherwort . Es stehen 16 Elementar - befehle zur Verfügung (4 Grundrechnungsarten , logisches Pro - dukt , Transferoperationen , Verzweigungen , Ein - Ausgabe)
Eingabe von Daten : Tastatur oder Lochstreifenleser
Ausgabe von Daten : Schreibmaschine oder Lochstreifenstanzer

Das Programmieren in der Maschinensprache ist sehr umständlich und zeitraubend . Deshalb bestand der Wunsch nach einem leistungs - fähigen Sprachenübersetzer .

2.3.2 Verschiedene mögliche Systeme

Es gibt zwei Arten von Sprachenübersetzern :

a) Compilersystem
   Alle Anweisungen werden nacheinander in die Maschinensprache des betreffenden Computers übersetzt .
   Nachteil : Nachträgliche Korrekturen sind nicht möglich
   Vorteil : Der Programmdurchlauf geht sehr schnell vor sich, (was für die oft langsamen Kleincomputer wichtig ist) da das Programm aus Maschinenbefehlen besteht .
   Bei manchen Systemen wird das Programm vor jedem Durchlauf vollständig neu compiliert . Dieses System kommt für Klein - computer jedoch nicht in Frage , da es hohe Anforderungen an die Rechengeschwindigkeit stellt .

b) Interpretiersystem

   Alle Anweisungen werden ohne Bearbeitungen gespeichert . Sie

werden lediglich nach ihrer Zeilennummer geordnet . Beim Pro - grammdurchlauf wird jede Anweisung vom Interpreter inter - pretiert d.h: ihre Funktion wird entschlüsselt und ausgeführt .

Nachteil : Der Programmdurchlauf geht sehr langsam vor sich , da umfangreiche Dekodierarbeiten zu leisten sind . In Programmschleifen werden die Anweisungen bei jedem Durchlauf dekodiert , während beim Compiler - system diese Arbeit nur einmal zu leisten ist .

Vorteil : Nachträgliche Korrekturen , Auflisten des Pro - grammes etc. sind möglich .

Es stellte sich heraus , daß für einen Kleincomputer mit be - schränkter Rechengeschwindigkeit und Speicherkapazität eine Kombination beider Systeme günstig ist . So kann sowohl der Technik der Zeilennummern (nur bei Interpretiersystem), als auch der Rechengeschwindigkeit (günstig bei Compilersystem) Rechnung getragen werden .

Die Anweisungen werden vom Compiler in eine verarbeitungs - freundliche Zwischenform umgewandelt und gespeichert . Bei Pro - grammdurchlauf werden sie unter einem verhältnismäßig geringen Rechenaufwand interpretiert (siehe 2,3.3) .

2.3.3 Grundaufbau des Systems

Das BASIC - System besteht aus 4 Hauptprogrammteilen , die in der folgenden Tabelle aufgeführt sind . Die Speicherplatzangaben sind in Prozent des Gesamtspeicherplatzes aufgeführt .

STEUERTEIL (Programm-stamm)	COMPILER	INTERPRETER	BASIC-PROGRAMMSPEICHER Befehlsspeicher 8% Nebenspeicher 9% Datenspeicher 5%
3%	41%	34%	22%
100% = 4k Wörter			

Abb.1

Dieser Grundaufbau dürfte für BASIC - Syteme auf den meisten Kleincomputern gelten .
Die einzelnen Gruppen haben folgende Funktionen :

a) Steuerteil (Programmstamm)
Die Hauptfunktion dieses Programmteils ist die Verarbeitung und Ausführung von Steuerbefehlen vom Operator . Die wichtigsten

sind : "run" : Programmdurchlauf , "program" : Aufruf des Com - pilers zur Eingabe von BASIC - Anweisungen , "list" : Ausdruck einer Liste der BASIC - Anweisungen ua.

b) Compiler

Der Compiler übersetzt Anweisungen in eine halbmaschinensprch - liche Zwischenform , die er im BASIC - Programmspeicher spei - chert (siehe Abb.1) . Der Programmspeicher ist in 3 Teile auf - geteilt . Der Befehlsspeicher umfaßt 320 Speicherworte . Jedes Wort nimmt eine BASIC - Anweisung auf , so daß insgesamt 320 An - weisungen gespeichert werden können . Die Anweisung wird in einem , ihrer Zeilennummer entsprechenden Wort gespeichert , so daß die richtige Reihenfolge der Anweisungen , die bei der Eingabe nicht vorhanden sein muß , hergestellt wird .
Wenn man von einfachen Anweisungen (z.B: END) absieht , ist ein einziges Speicherwort zu klein , um eine komplette Anweisung aufzunehmen . Deshalb wird im Befehlsspeicher nur ein Identi - fizierungscode der Anweisung (In Abb.2a dargestellt durch den Anweisungsnamen z.B: "INPUT") gespeichert . Die restlichen Teile der Anweisung (Daten , Texte etc.) werden im Neben - speicher gespeichert . Die Adresse der Anweisungsteile im Nebenspeicher und die Anzahl der Wörter werden dem Identifizierungs - code im Befehlsspeicher angehängt (im Beispiel: INPUT /3/120) .
Im Datenspeicher ist schließlich Platz für die Speicherplätze für die Variablen und Programmkonstanten .

Beispiel für die Speicherplatzbelegung für die 2 Anweisungen :
1 PRINT (basic-system) und 4 INPUT X,Y,Z

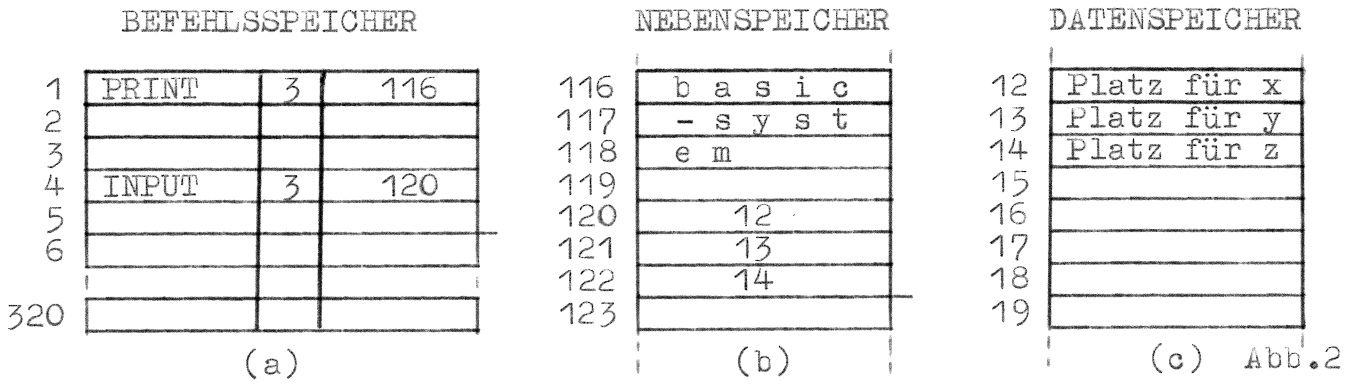


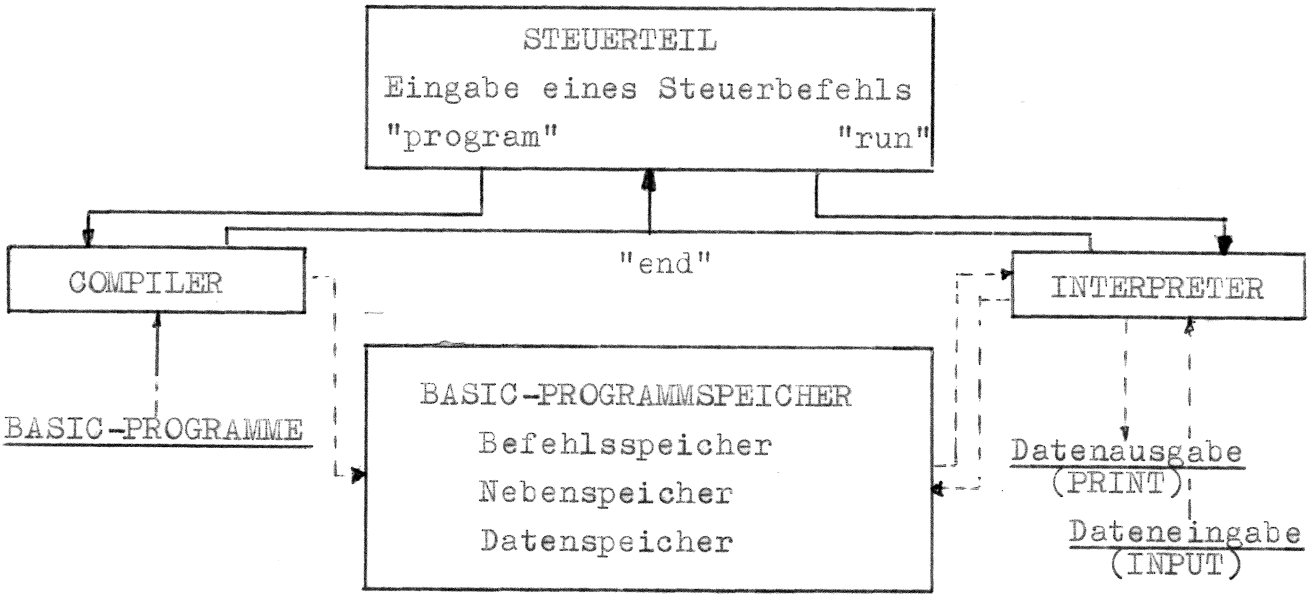
Abb.2

c) Interpreter

Wie schon erwähnt tritt der Interpreter während des Programm - durchlaufs in Funktion . Die in , wie oben beschriebener Form verschlüsselt gespeicherten Anweisungen werden entschlüsselt und aus - geführt . Zum Beispiel werden bei der in Abb.2 aufgeführten

PRINT - Anweisung nach der Identifikation des Identifizierungs - codes ("PRINT"), die im Nebenspeicher gespeicherten 12 Zeichen ("basic-system") ausgedruckt oder ausgestanzt .
Im Abschnitt 2.3.4 wird die Funktion des Interpreters an einem Beispiel ausführlich dargestellt .

Folgende Abbildung zeigt noch einmal deutlich die Zusammenhänge zwischen den 4 Programmteilen . Die gestrichelten Linien der Ab - bildung sind Datenwege .



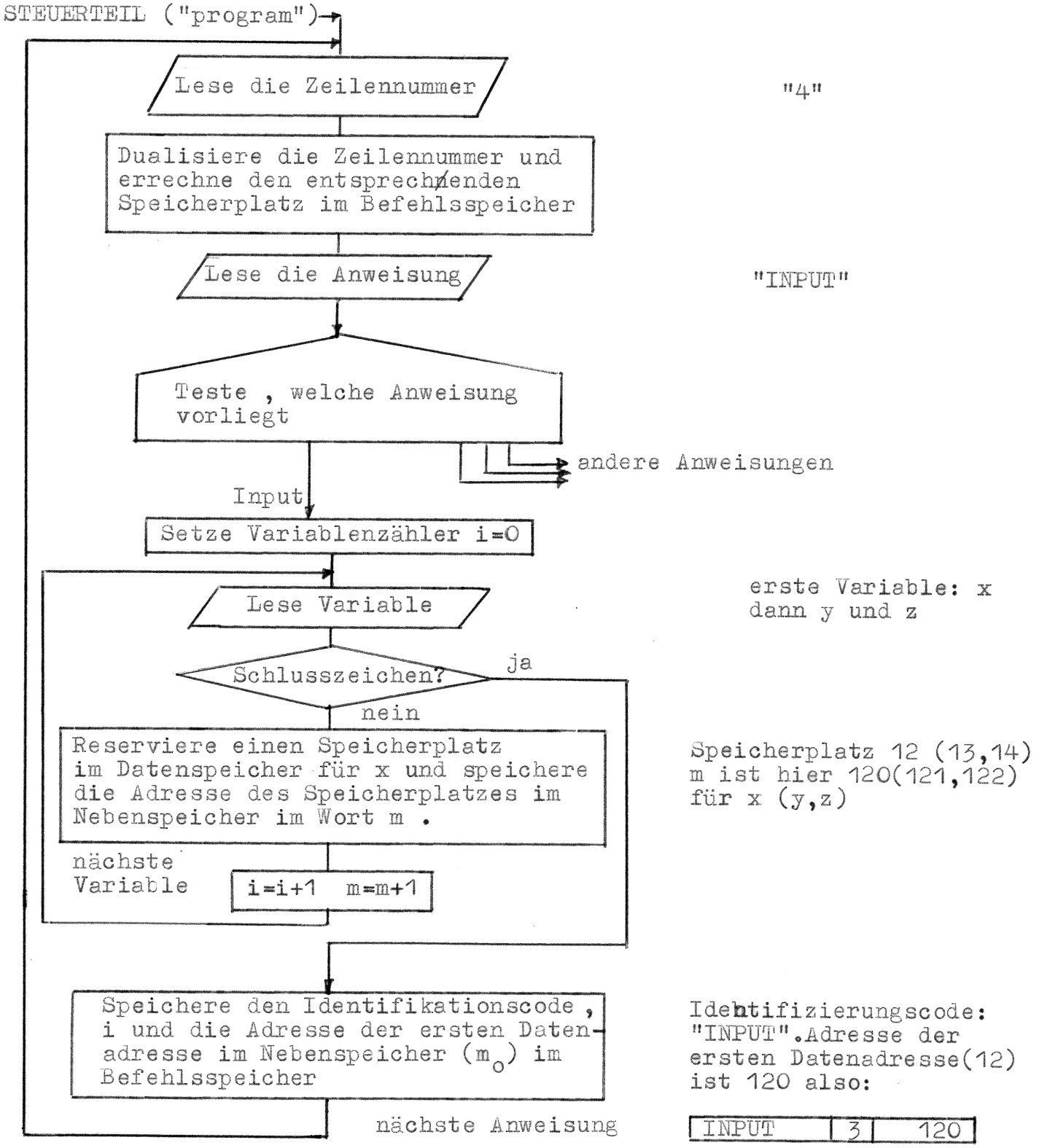
Vorteile dieses Systems : Es ist möglich, nachträglich Änderungen am Programm vorzunehmen , indem man den betreffenden Speicher - platz im Befehlsspeicher löscht oder mit einer neuen Anweisung belegt . Da alle Konstanten bzw. Adressen schon ins Dualsystem bzw. in reale Maschinenadressen umgewandelt sind , geht der Pro - grammdurchlauf (Interpreter) schnell vor sich .

2.3.4 Demonstation der Bearbeitung einer BASIC - Anweisung

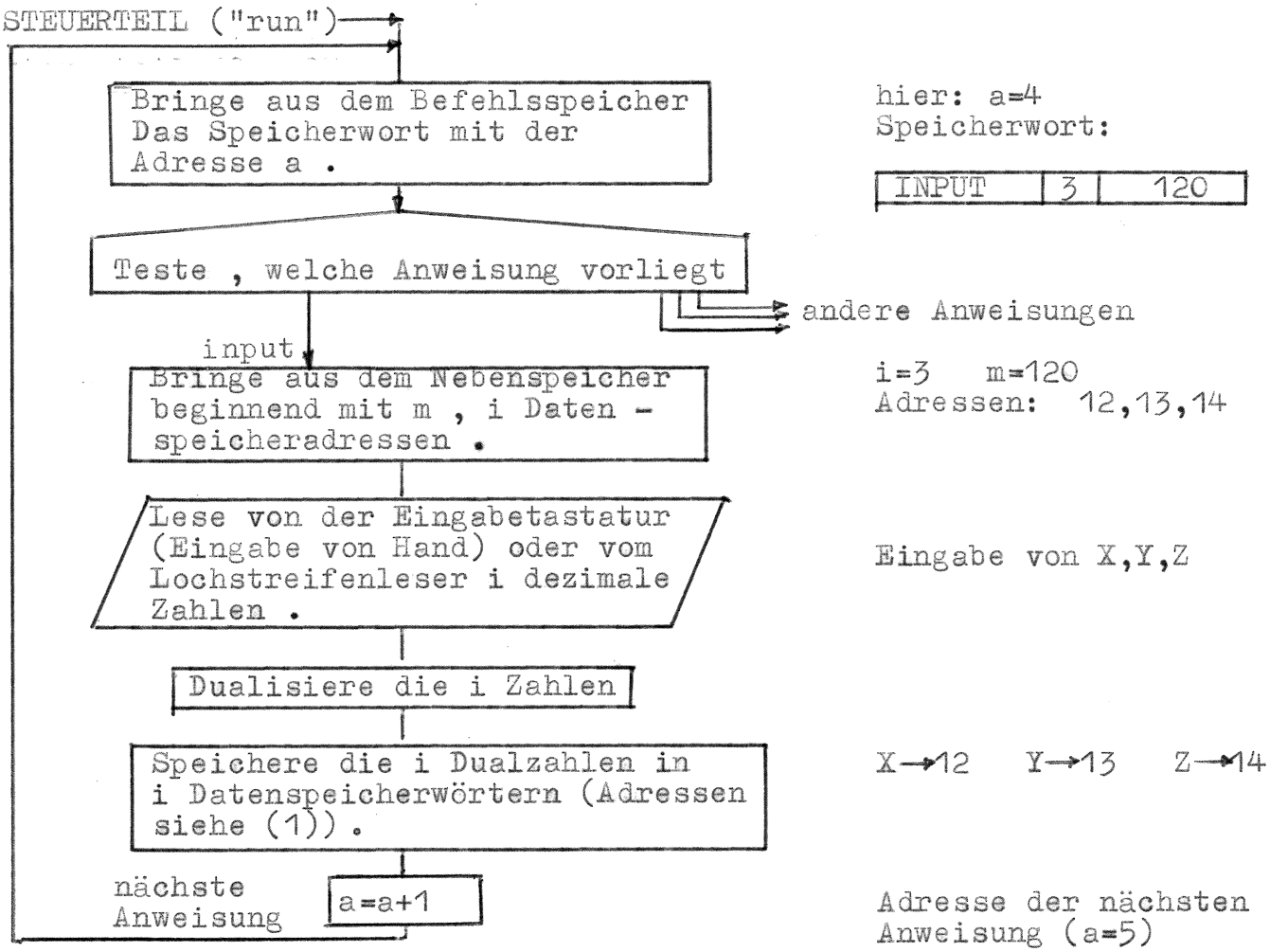
Die folgenden zwei Flussdiagramme stellen Ausschnitte aus dem Compiler (a) und dem Interpreter (b) dar . An ihnen soll die Be - arbeitung einer BASIC - Anweisung an dem Beispiel : 4 INPUT X,Y,Z (siehe Abb.2) demonstriert werden .
Jede Einheit im Flussdiagramm (z.B: "Dualisiere..") besteht in Wirklichkeit aus vielen Einzelschritten , die ihrerseits aus vielen Maschinenbefehlen bestehen . Es ist leider unmöglich , hier auf solche Details einzugehen .
Jeder der insgesamt 22 BASIC - Anweisungen benötigt zwei solcher Programmabschnitte in Compiler und Interpreter , die durch die ,

im Flussdiagramm eingezeichneten Verzweigungen "andere Anweisungen" angedeutet sind .

a) Flussdiagramm des Compilers für INPUT



b) Flussdiagramm des Interpreters für INPUT



3 SCHLUSSBETRACHTUNG

Das entwickelte BASIC - System hat sich in den Monaten seit der Fertigstellung bewährt und übertrifft sowohl an Programmierkomfort als auch an Leistungsfähigkeit die vorhandenen Systeme (assembler - ahnliche , Gleitkommainterpretiersysteme) .
Es wurde damit bewiesen , daß es möglich ist , für einen Klein - computer ein relativ leistungsfähiges BASIC - System mit nicht zu großen Kürzungen von Funktionen zu entwickeln .
Auch die Wahl von BASIC erwies sich als richtig . Eine anspruchs - vollere Programmiersprache hätte nur zu noch größeren Kürzungen geführt und hätte außerdem den Speicherplatz für die Programme weiter verkleinert und hätte im Endeffekt zu schlechtereren Er - gebnissen geführt .
Ein wichtiges Ergebnis ist daher : Bei einem kleinen Computer bringt eine kleine, anspruchslose Programmiersprache bessere Ergebnisse . Aus diesem Satz kann man eine Zukunftsperspektive für BASIC ableiten : Während BASIC an großen Computern verdrängt wird (ist) , wird es sich an kleinen Computern,besonders an Schulen behaupten .

Berechnung von e nach einer Taylorreihe

```
 5    line 2
10    print (berechnung von e nach einer taylorreihe)
15    line 1
20    print (wiviele summenglieder sollen berechnet werden)
25    input n
30    let e=1
35    let f=1
40    for m=1 to n
45    let f=f*m
50    let e=e+1/f
55    next m
60    print e
65    end

ready run

berechnung von e nach einer taylorreihe
wieviele summenglieder sollen berechnet werden
 5
2.71666
ready run
berechnung von e nach einer taylorreihe
wieviele summenglieder sollen berechnet werden
 14
2.71828
ready
```

Dieses Programm druckt zu einer Postleitzahl die entsprechende Stadt

```
 10     line 2
 15     print (postleitzahlenprogramm)
 20     line 1
 25     print (eingabe einer postleitzahl)
 30     input post
 35     let q=post/100+35
 40     goto q
 45     print (berlin)
 50     end
 55     print (hamburg)
 60     end
 65     print (hannover)
 70     end
 75     print (duesseldorf)
 80     end
 85     print (koeln)
 90     end
 95     print (frankfurt)
100     end
105     print (stuttgart)
110     end
115     print (muenchen)
120     end

ready run

postleitzahlenprogramm
eingabe einer postleitzahl
 4000
duesseldorf
ready run

postleitzahlenprogramm
eingabe einer postleitzahl
 7000
stuttgart
ready
```

Programm zur Berechnung der Teiler einer Zahl

```
 5      set 2
10      line 2
15      print (berechnung der teiler einer zahl)
20      input zahl
25      print (die teiler sind )
30      for n=zahl to 1
35      let n1=n/zahl
40      let n2=int(n1)
45      if n1=n2 goto 50 else 55
50      print n
55      next n
60      end
```

```
ready run

berechnung der teiler einer zahl
 70
die teiler sind  70.   35.   14.   10.   7.0   5.0   2.0   1.0
ready run

berechnung der teiler einer zahl
 40
die teiler sind  40.   20.   10.   8.0   5.0   4.0   2.0   1.0
ready
```

Berechnung von π nach der Methode der Eckenverdopplung eines n-ecks

```
 5      line 2
10      print (berechnung von pi nach der archimedischen methode)
15      line 1
20      let n=6
25      let w=1/2
30      print (eckenzahl    innerer umf./2    ausserer umf./2    differenz)
35      line 1
40      let u=sqr(2-w*2)
45      let r=2*sqr((1-w)/(1+w))
50      let w=sqr(2+2*w)/2
55      let ul=n*u/2
60      let rl=n*r/2
65      let diff=rl-ul
70      print n ul rl diff
75      let n=n*2
80      if n max 6144 goto 85 else 35
85      end
```

```
ready run

berechnung von pi nach der archimedischen methode
eckenzahl     innerer umf./2     ausserer umf./2      differenz
6.00000         2.99999          3.46405          .464068
12.0000         3.10581          3.21531          .109544
24.0000         3.13261          3.15960          2.69941 e-02
48.0000         3.13933          3.14603          6.70056 e-03
96.0000         3.14102          3.14268          1.66048 e-03
192.000         3.14142          3.14184          4.19723 e-04
384.000         3.14152          3.14162          1.00153 e-04
768.000         3.14156          3.14157          1.90428 e-05
1536.00         3.14156          3.14154         -2.46690 e-05
3072.00         3.14157          3.14154         -3.93367 e-05
6144.00         3.14158          3.14153         -5.578433 e-05
ready
```

So sieht der Steuerteil in der Maschinensprache aus . Man sieht an diesem Programmabschnitt , wie umständlich das Programmieren mit der Maschinensprache ist .

```
v0220000'
81090'k3w5j'2'80k9j'2'825f4'2'839fj'
2'815g4'2'809gj'2'803j4'2'800jj'
400k0'w01w0'g1wf8'w01w8'g00q8'w01q4'g3080'w01qj'
g3080'w01w4'g3080'w02wj'g02q4'w01q8'g0184'f0174'
80010'40014'
011qf856'v040008j'
13w1j'w00jj'f00gj'7ww0000'10'k1658'70000000'13100'
2'f0000'''g00q0'j0088'k3w1j'f0080'
wwwwj'90098'6009j'q00f0'f00j0'8001j'40020'80024'
40028'j3w48'9014j'w00f4'g00fj'w00fj'g010j'f00fj'
13w48'200f0'w0150'f00k0'11gq4'k25q8'f10f8'3300'
10128'k0888'100f8'k0k24'f0230''''
w000j002'10000'80000'40004'w0008'g04q4'f3080'wwwwj'
k0000'f0154'101k4'k0338'101kj'f0190'101k8'k0338'
29kjw5qf'
v040018j'
10500'k04q4'f0k00'8030j'k3w10'2'80018'4001j'
w0020'g0300'q0028'30488'f0600'201j8'1022j'k0000'
80340'f01f0'f02g8'f02j0'f04q8'1wk3100'800'5j0000'
w000'400'30000'900'k3w38'f27f0''wwwk3184'
13wg8'w0168'f0240'13wk3f00'13w1j'60jq8'f0f0j''
wwwwwww4'k3ww8'101q0'f02w0'3f00'j0208'j3wg8'g29j0'
13ww8'f0208'90400'71qg0'q04qj'f0k44'1022j'k3ww8'
10218'f0280''''30248'f0240'f0000'
17q8j86q'
v01k028j'
''''''''
'''81810'14'2016j'k3w8j'f033j'
'q0028'60010'k3w1j'11583'f2f2j'1023j'k1588'
f012j'30248'f0240'f0264'f0000'
0086wg3j'
```

Die folgenden zwei Programmblätter stellen Programmausschnitte aus dem Systemverteiler (VI) und aus dem Interpreter (VII) dar .

Funktionen der zwei Programmteile:

VI : a) Ausdruck des Wortes "ready" (Zeigt die Bereitschaft , Systembefehle zu empfangen, an) b) Lesen eines Systembefehls c) Entscheidung, welcher Befehl vor - liegt.

VII: a) Suchen des , gerade im Programmdurchlauf benötigten Befehls b) Entscheidung , welcher Befehl vorliegt c) Nach der Ausführung Erhöhung des Programmzählers um 1 d) Interpreter für "END" und "LINE" .

Wenn man sich vor Augen führt , wieviele Maschinenbefehle für diese wenigen und einfachen Funktionen benötigt werden , erkennt man , wie umständlich das Programmieren in der Maschinensprache ist .

Blatt A

Eurocomp GmbH

LGP 30 - Rechenprogramm

Auftrag Nr. _____ Bearbeiter _____ Blatt ___ von ___

Programm Nr. _____ Geprüft _____ Datum _____

Problem Systemverteiler Spur Nr. 00

nicht UT				nicht D	nicht DM	nur N	Speicher	Befehl Opr.	Befehl Adr.	Inhalt der Adresse
50	43	36	29	22	15	8	00	P	1636	Führe einen WR durch
51	44	37	30	23	16	9	01	C	6337	Lösche den ACC
52	45	38	31	24	17	10	02		2	Stopp zur Synchronisation
53	46	39	32	25	18	11	03	P	1339	Drucke „r"
54	47	40	33	26	19	12	04		2	
55	48	41	34	27	20	13	05	P	3741	„e"
56	49	42	35	28	21	14	06		2	
57	50	43	36	29	22	15	07	P	5743	„a"
58	51	44	37	30	23	16	08		2	
59	52	45	38	31	24	17	09	P	2145	„d"
60	53	46	39	32	25	18	10		2	
61	54	47	40	33	26	19	11	P	0947	„y"
62	55	48	41	34	27	20	12		2	
63	56	49	42	35	28	21	13	P	0349	„␣"
0	57	50	43	36	29	22	14		2	
1	58	51	44	37	30	23	15	P	0051	Lese einen Steuer-
2	59	52	45	38	31	24	16	I	0000	befehl ein.
3	60	53	46	39	32	25	17	S	0160	
4	61	54	47	40	33	26	18	T	3142	→ do
5	62	55	48	41	34	27	19	S	0162	
6	63	56	49	42	35	28	20	T	0058	→ get
7	0	57	50	43	36	29	21	S	0157	Sprung zur
8	1	58	51	44	37	30	22	T	2218	→ list Ausführung
9	2	59	52	45	38	31	23	S	0159	des jeweiligen
10	3	60	53	46	39	32	24	T	2220	→ save Systembefehls
11	4	61	54	47	40	33	25	S	0161	(8 gültige Befehle)
12	5	62	55	48	41	34	26	T	2222	→ renum
13	6	63	56	49	42	35	27	S	0263	
14	7	0	57	50	43	36	28	T	0257	→ clear
15	8	1	58	51	44	37	29	S	0158	
16	9	2	59	52	45	38	30	T	0133	→ program
17	10	3	60	53	46	39	31	U	0129	→ manual

Blatt A

Eurocomp GmbH

LGP 30 - Rechenprogramm

Auftrag Nr. _____ Bearbeiter _____ Blatt ___ von ___

Programm Nr. _____ Geprüft _____ Datum _____

Problem Interpreter-Verteiler END+LINE Interpreter Spur Nr. 30

nicht UT				nicht D	nicht DM	nur N	Speicher	Befehl Opr. Adr.	Inhalt der Adresse
50	43	36	29	22	15	8	00	B[ ]	BASIC-Zwischenform-Befehle
51	44	37	30	23	16	9	01	T3014	→ Leerstelle
52	45	38	31	24	17	10	02	Y3003	
53	46	39	32	25	18	11	03	U[ ]	→ Interpreter des betr. Befehles
54	47	40	33	26	19	12	04		
55	48	41	34	27	20	13	05	B3020	Anfang "END" Interpreter
56	49	42	35	28	21	14	06	C4152	
57	50	43	36	29	22	15	07	U0000	→ Sprung zum System-V.
58	51	44	37	30	23	16	08		
59	52	45	38	31	24	17	09		andere Funktionen aus
60	53	46	39	32	25	18	10		Übersichtsgründen weggelassen
61	54	47	40	33	26	19	11		
62	55	48	41	34	27	20	12		
63	56	49	42	35	28	21	13		
0	57	50	43	36	29	22	14	B3000	Erhöhen der Zeilennummer
1	58	51	44	37	30	23	15	A3130	(gespeichert in 3000) um 1@39
2	59	52	45	38	31	24	16	U3021	
3	60	53	46	39	32	25	17		
4	61	54	47	40	33	26	18		
5	62	55	48	41	34	27	19		
6	63	56	49	42	35	28	20		
7	0	57	50	43	36	29	21	H3000	
8	1	58	51	44	37	30	22	-T3005	Abbruchmöglichkeit durch "S"-
9	2	59	52	45	38	31	23	U3000	Taste
10	3	60	53	46	39	32	24		
11	4	61	54	47	40	33	25	E2611	[3J000] Anfang "LINE" I.
12	5	62	55	48	41	34	26	S0014	[2]
13	6	63	56	49	42	35	27	P1663	
14	7	0	57	50	43	36	28	S0926	1@17
15	8	1	58	51	44	37	29	4	Stopp zur Synchronisation
16	9	2	59	52	45	38	30	T3014	→ Ende des "LINE" Befehles
17	10	3	60	53	46	39	31	U3027	nächster WR

Fehlersuche und Fehlerdiagnostik

Ein neues Programm funtioniert in den meisten Fällen gar nicht oder nur teilweise . Deshalb verfügt jedes größere Compiler-oder Interpretiersystem über ein Fehlerdiagrostiksystem . Von diesem wird das eingegebene Programm kontrolliert . Syntaktische und semantische Fehler werden in irgend einer Weise dem Benützer mitgeteilt .
Auch das BASIC-System für den LGP-30 verfügt über ein solches System . Allerdings mussten , um Speicherplatz zu sparen , zwei Maßnahmen durchgeführt werden :

- syntaktische Fehler der einfachsten Art (z.B: Schreib - fehler (prinr statt print)) müssen vom Benützer erkannt werden . Es gibt nämlich so viele Fehlermöglichkeiten dieser Art , daß eine Kontrolle unmöglich ist . Außer - dem werden solche Fehler beim Durchlesen sofort erkannt und behoben .
- Logikfehler der schwersten Art (z.B: bestimmte Schleifen - verschränkungen) werden nicht erkannt . Die Aufdeckung dieser würde komplizierte Rechnungen erfordern und sind deshalb aus Gründen des Speicherplatzes und der Rechen - zeit indiskutabel . Das Fehlen von Fehlermeldungen dieser Art fällt aber nicht ins Gewicht da 1) es nur wenige dieser Fehler gibt und 2) in der Benützeranleitung darauf hingewiesen wird , so daß bei sorgfältigem Programmieren keine Gefahr besteht .

Alle Fehlermeldungen haben folgendes Format : ERROR TS AT XXX

XXX: Zeilennummer des fehlerhaften Befehls (OOO bei Systemfehlern)
T : Gruppe des Fehlers (1,2 Compilerfehler 5,6,7 Interpreterfehler 9 Systemfehler)
S : Fehlernummer

Beispiel: ERROR 17 AT 080

Einige Fehlernummern:

10 Zeilennummer eines Befehls fehlt
13 Der Nebenspeicher ist voll
17 Klammerfehler bei einem arithmetischen Ausdruck
52 Division durch Null
77 Das Programm besitzt keine "END"-Anweisung
99 Undefinierbarer Systemfehler. Empfehlung: System neu einlesen