B. BECKS, R. RINGEL

Kernforschungsanlage Jülich GmbH.
Zentrallabor für Elektronik/
Abt. Nukleare Elektronik und
on-line Datenverarbeitung

BASIC Compiler für 305

Kernforschungsanlage Jülich GmbH., Zentrallabor für Elektronik/
Abteilung Nukleare Elektronik und On-Line Datenverarbeitung
517 Jülich, Postfach 365

- B. BECKS, R. RINGEL -

Eine Anregung der SAK zur Implementierung eines BASIC-Compilers an einem Siemens-Rechner der Reihe 3oo wurde von Herrn Nabert von der GH Paderborn aufgegriffen und im Rahmen von vier Ingenieurarbeiten verwirklicht. Die Motivation von Herrn Nabert, eine solche Arbeit zu betreuen, mag zum einen darin gelegen haben, zu zeigen, daß auch eine Arbeit von so großem Umfang von Ingenieurstudenten als Abschlußarbeit ausgeführt werden kann, und zum anderen in der Überzeugung, daß Dialogsprachen in der Zukunft immer mehr an Bedeutung zunehmen werden.

An dieser Stelle möchten wir besonders darauf hinweisen, daß wir dieses Problem ohne jede Erfahrung analysieren mußten, da selbst die uns zur Verfügung stehende Literatur sich im wesentlichen lediglich mit der Syntaxanalyse arithmetischer Ausdrücke beschäftigt. Die zu Beginn der Arbeit gelösten Probleme könnten heute sicherlich optimaler gelöst werden, zumal die Koordinierung der einzelnen Teilaufgaben gegen Ende der Arbeit durch die gewonnene Übersicht immer besser geworden ist. Besondere Schwierigkeiten bereitete uns auch der stark begrenzte Zeichenvorrat des Systems 300.

Im folgenden soll ein Überblick über die zur Implementierung des Systems benutzten Methoden gegeben werden. Ein dialogfähiger Compiler erhält erst dann seine volle Bedeutung, wenn er in einem Mehrfachzugriffsystem angewendet wird. Aus diesem Grunde wurde der BASIC-Compiler bereits so konzipiert, daß er ohne Änderung in ein Time-Sharing-System eingebaut werden kann. Daher ist bereits für den Single-User-Betrieb ein kleines Betriebssystem erstellt worden, das unter ORG-I die Ein/Ausgaben und die in BASIC geforderten Compiler-unabhängigen Funktionen übernimmt (Listen, Korrigieren, Break usw.). Dieser Programmteil umfaßt 1000 Fw und beinhaltet die für den Ablauf des Compilers notwendigen Programme. Programmteile des Betriebssystems, die nicht für die Compilierung erforderlich sind, werden - wenn nötig - in den Laufbereich eingelesen.

Der minimale Laufbereich wurde auf 4000 Fw begrenzt. Der BASIC-Compiler ist somit in 5000 Fw ablauffähig. Dies hat den Vorteil, daß neben einer Compilierung und der dazu erforderlichen Eingabe, die über Blattschreiber relativ langwierig sein kann, noch andere Programme ablaufen können.

Ein weiterer Vorzug des kleinen Laufbereiches wird erst im Time-Sharing sichtbar. Hier kann der Teil des Compilers, der den Dialog übernimmt, gesondert bereitgestellt werden und nun arbeitsspeicherresident den Dialog mit allen angeschlossenen Teilnehmern führen.

Der kleine Laufbereich macht es allerdings notwendig, alle Listen auf einem Externspeicher zu führen; dadurch gibt es aber auch keine Beschränkungen hinsichtlich Programmlänge (maximaler Kernspeicherbereich) und Variablenanzahl.

Der Compiler (ohne System) setzt sich aus fünf Teilen, den Pässen 1, 2 und 3, dem Binder und dem ARP zusammen.

Der ARP dient einmal als Verbindungsglied zwischen den Pässen. Zeigerstände, die von mehreren Pässen benötigt werden, müssen hier geführt werden. Auch die Buchführung der Arbeitsdateien (Eröffnungsaufrufe) gehört hierzu.

Die Arbeitsdateien werden nur bei der Compilierung gebraucht und anschliessend wieder gelöscht. Die Datei des Quellprogrammes wird beim Abmelden des Benutzers gelöscht, falls dieser sein Programm nicht mit SAVE gesichert hat.

Eine weitere Aufgabe des ARP besteht in der Verbindung des Compilers mit dem System (Ein/Ausgabe, Starten).

Pass 1

Der erste Pass übernimmt den Dialog mit dem Teilnehmer. Organisatorisch gliedert er sich wieder in zwei Teile. Teil 1 (Pass 0) führt die lexikalische Analyse durch. Darunter verstehen wir das Aufbereiten des eingegebenen Statements, also das Entfernen überflüssiger Blanks, das Entfernen der durch ◇(Rhombus) als unerwünscht erklärten Zeichen sowie das Abtrennen der Zeilennummer.

Das komprimierte Statement wird an den Pass 1 übergeben, der die Syntaxanalyse durchführt.

Wird ein Statement als syntaktisch falsch erkannt, wird auf dem Blattschreiber ein #(Irrungszeichen) ausgegeben, im anderen Fall ein *(Stern) gedruckt.

Nach der Eingabe des gesamten Programmes wird vom ARP veranlaßt, daß der Pass 2 eingelesen wird.

Pass 2

Eine vollständige Syntaxanalyse läßt sich im Pass 1 nicht durchführen, da unerlaubte Verschachtelungen von FOR-NEXT, fehlende Zeichennummern bei GOTO-, GOSUB- oder IF-Anweisungen und READ ohne DATA erst erkannt werden können, wenn das ganze Programm sortiert vorliegt, da bei der Sortierung noch die Statements gleicher Zeilennummer behandelt werden müssen. Dabei wird dann das zuerst gelesene Statement eliminiert.

Neben dieser Aufgabe muß im Pass 2 dafür gesorgt werden, daß- unabhängig von der Zeilennummer - die DATA- und DIM-Anweisungen an den Kopf des zu generierenden Programms kommen. Weiter werden die Zeilennummern in den Sprunganweisungen durch eine Zahl ersetzt, die sich aus der Lage des Statements relativ zum Programmanfang ergibt, allerdings ohne Berücksichtigung der DATA- und DIM-Anweisungen.

Pass 3

Dieser Pass hat die Aufgabe, das syntaktisch richtige Programm in den Maschinencode zu übersetzen.

Auch hier ist es wieder notwendig, die Anfangsadressen der einzelnen Statements (im Maschinencode) abzuspeichern. Dies geschieht in Form einer Sprungleiste. Diese wird abgespeichert und nach der Compilierung an das generierte Programm angefügt. Die Sprungleiste enthält absolute Adressen, da auch das Maschinencodeprogramm in absoluter Form erstellt wird. Hierdurch wird das übliche Bereitstellen zu einem Einlesen vereinfacht.

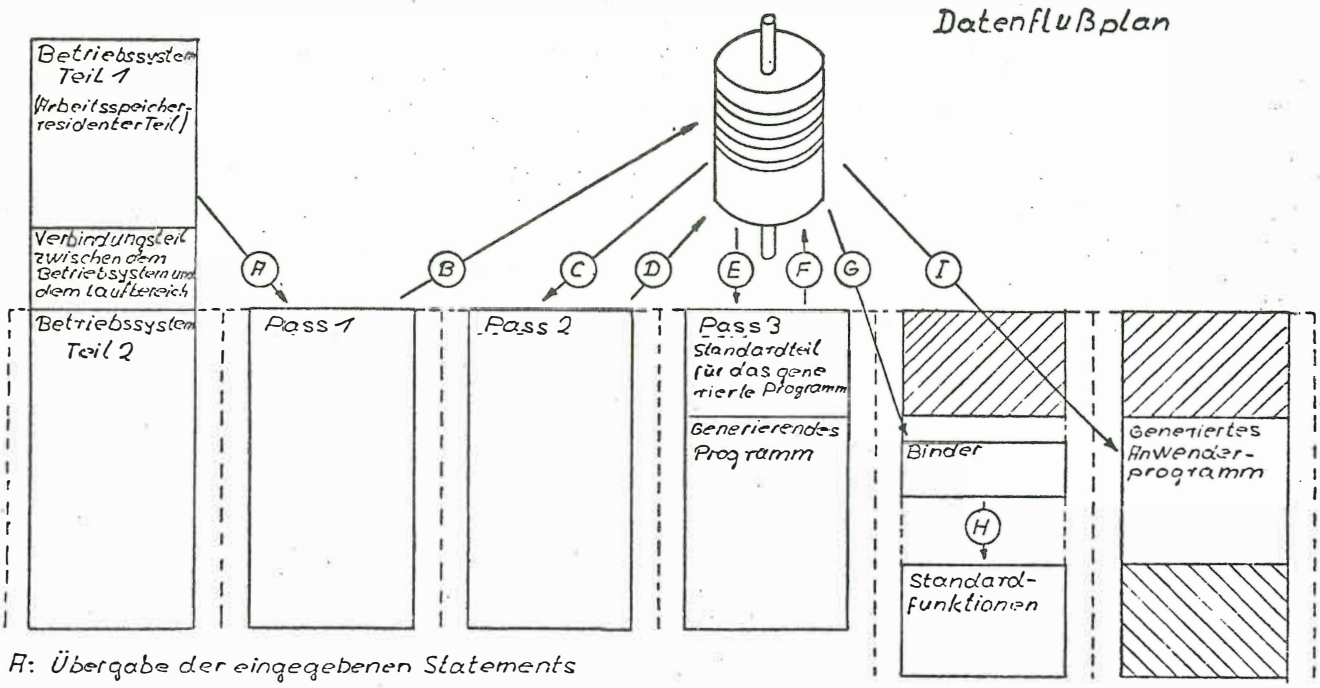
Die hier genannte absolute Adressierung bedeutet keine Einschränkung hinsichtlich der Wahl der Bereitstelladresse des Compilers, da diese Adressen nur mit der durch die Bereitstellung des Compilers festgelegten Anfangsadresse des generierten Programms gebildet werden. Der Start des generierten Programms erfolgt durch einen Sprung aus dem ARP an den Kopf des erstellten Codes. Die Ein/Ausgaben (INPUT, PRINT) werden auch hier über das System abgewickelt.

Pass 4

Dieser Pass hat die Aufgabe, Standardfunktionen und Module aus der Bibliothek an den Objekt-Code anzufügen und dabei diese Funktionen aus der relativen in die absolute Form zu bringen. Eine Verbindung mit dem Objektcode wird über eine Adressenliste im Standardteil des Pass 3 erreicht.

Bibliothek

Die Bibliothek des BASIC-Compilers enthält die in BASIC geforderten Funktionen, die notwendigenKonvertierungsprogramme und die Module für TAB (X) und INPUT.



A: Übergabe der eingegebenen Statements
B: Listen anlegen
C: Listen einlesen
D: Bearbeitete Listen wieder abspeichern
E: Aufbereites Programm und Constanten einlesen
F: Generiertes Programm ablegen
G: Standardfunktionen einlesen
H: ---- " ---- " ---- abspeichern
I: Das generierte Programm einlesen

Vor der Aufstellung der von uns implementierten Teilmenge von BASIC einige Besonderheiten und Abweichungen der allgemeinen BASIC-Syntax.

1. Die Schreibweise für X^N ist ausschließlich X**N.
   Die Operatoren + und - können beliebig kombiniert werden.
   Beispiel: A + -- + - B ergibt A-B.
   Dies gilt auch für den einstelligen Gebrauch (Vorzeichen).
   Beispiel: ---A--B ergibt -A + B.
   Vorzeichenbehaftete Ausdrücke brauchen nicht in Klammern gesetzt zu werden.
   Beispiel: +A/-B*-1+2 ist identisch mit (((+A)/(-B))*(-1))+2

2. Für die Vergleichsoperatoren mußte aus anlagenspezifischen Gründen auf die FORTRAN IV-Schreibweise ausgewichen werden:

   Für = steht .EQ. (equal)
   " < steht .LT. (less than)
   " ≦ steht .LE. (less equal)
   " > steht .GT. (greater than)
   " ≧ steht .GE. (greater equal)
   " ≠ steht .NE. (not equal).

3. Die im PRINT mit "," vorgenommenen Feldeinteilungen sind 16 Zeichen lang. Daraus ergibt sich ein maximaler Ein/Ausgabe String von 8o Zeichen.

4. Anstelle des Semikolons (Strichpunkt) im PRINT tritt der Doppelpunkt (:), da das Semikolon im System 3oo eine Eingabe beendet. Um noch andere Anwenderprogramme neben dem BASIC-Compiler laufen lassen zu können, wurde auf eine Änderung des Endezeichens zu Gunsten dieses Kompromisses verzichtet.

Die Wirkungsweise der BASIC-Anweisungen können in der SIEMENS-Beschreibung des TBS-Dialog-BASIC der 4oo4/46 Ausgabe November 1969, Bestell-Nr. 2-2600-9o4 nachgelesen werden. Auf die Abweichungen von diesem Text wird hier näher eingegangen. Folgende Anweisungen wurden implementiert:

LET — Bei Stringvariablenist eine Zuweisung nur zwischen Stringvariablen möglich.

READ — Stringvariable dürfen nicht indiziert sein.

DATA — Strings müssen durch Apostroph eingeschlossen sein. Die Stringlänge ist beliebig.

PRINT — Anstelle des Semikolons (Strichpunkt) tritt der Doppelpunkt, da Semikolon das Endezeichen einer Zeile (Eingabe) ist.
Die Verwendung des Doppelpunktes bewirkt eine bündige Packung der Ausgabedaten ohne Zwischenraum. Ein Blank kann nur bei positivem Vorzeichen auftreten, da dieses nicht ausgegeben wird. (Dezimale Gleitpunktzahlen werden allerdings immer 16 Zeichen lang ausgegeben, unabhängig von Vorzeichen und Exponentenstellen).
Komma und Doppelpunkt können beliebig kombiniert werden. Die Verwendung des Kommas reserviert ein Feld von 16 Zeichen.

GOTO

IF-THEN — Vergleich von Strings ist nicht möglich

FOR-NEXT — Kontrollen werden nicht durchgeführt. Durch die Art der Abfrage führt aber lediglich STEP 0 zu einer Endlosschleife.

DIM — Unabhängig davon, ob eine Variable ein- oder zweifach indiziert ist, wird ihr, falls kein DIM gegeben wird, nur ein Feld von 11 Plätzen (0-10) reserviert. Feldreservierungen für Strings sind nicht möglich.

END

STOP — Die Anweisung STOP wird als END aufgefaßt.

GOSUB-RETURN — Es können bis zu 14 Unterprogramme ineinander verschachtelt werden. Einbau von Objektmoduln ist nicht möglich.

INPUT — Wird vom Programm "INPUT" gefordert, leuchtet die Anruftaste auf (BSEI). Es wird kein Zeichen dafür ausgegeben (begrenzter Zeichenvorrat). Fehlerhafte Eingaben werden durch # angezeigt. (die komplette Eingabe muß wiederholt werden).
Sind weitere Eingaben erforderlich, wird % ausgegeben. Trennzeichen zwischen den Variablen ist das Komma, Endezeichen ist das Semikolon. Eingabedaten können beliebig lang sein; es werden jedoch nur die ersten 15 Zeichen ausgewertet. Strings müssen in Apostroph eingeschlossen sein.

REM
RESTORE

Bei Anwendung der Funktion TAB kann vorwärts und rückwärts beliebig oft tabuliert werden.

Systemkommandos und ihre Bedeutung:

BASIC;	Mit diesem Kommando meldet sich der Benutzer beim System an.
NEW name;	Es soll ein neues Programm erstellt werden.
OLD name;	Ein mit SAVE auf dem Externspeicher hinterlegtes Programm wird in den Laufbereich eingelesen und kann weiter bearbeitet werden.
EOF;	Beendet eine Programmeingabe
RUN;	Beendet eine Programmeingabe und/oder startet das gerade im Laufbereich stehende Programm.
LIST;	Ausgabe des gesamten Programms.
SAVE;	Eine Kopie des gerade im Laufbereiches stehenden Programms wird auf dem Externspeicher hinterlegt.
UNSAVE;	Die Kopie des gerade im Laufbereich stehenden Programms auf dem Externspeicher wird gelöscht.
COR;	Mit diesem Kommando kommt man vom Kommandozustand in den Eingabemodus. Anschließend kann das Programm im Laufbereich korrigiert werden.
BJE;	Mit diesem Kommando meldet sich der Benutzer beim System ab.
BREAK;	Kann jederzeit gegeben werden. Mit BREAK wird eine laufende Rechnung oder Ausgabe abgebrochen. Wird eine Ausgabe abgebrochen, so wird nach BREAK noch eine Zeile ausgegeben. Anschließend ist das System im Kommandozustand.
RUNSD;	Ausgabe auf Schnelldrucker mit einmaligem Seitenvorschub.) können nur vom
RUNDSDO;	Wie RUNSD jedoch ohne Seitenvorschub.) BBS Ø aus ge-
LISTSD;	Ausgabe des gesamten Programms auf Schnelldrucker.) geben werden.
JOB;	Eingabe eines Quellprogramms von Lochkarte. Das Protokoll wird auf) dem Schnelldrucker ausgegeben. Letzte Karte muß EOF; oder RUN; sein.)
BASE;	Mit diesem Kommando wird das BASIC-System beendet.)

Einige Daten vom Compiler:

Vor jedem Rechenlauf wird das Quellprogramm neu übersetzt.
Arbeitsdatein werden nach jedem Lauf gelöscht.
Aus einem Statement werden durchschnittlich 1o-12 Worte Code.
Das System läuft unter 2 Programmnummern.
Der Compiler ist zu einem ORG mit S-Makros kompatibel,falls dieses eine EAP-Schnittstelle hat.
Mimimaler Kernspeicherbedarf 5000 Worte
Minimaler Externspeicherbedard etwa 48 K Worte
Die Zeit um 1oo Statements zu compilieren und benötigte Standardfunktionen anzubinden, beträgt etwa 3 Sekunden.
Mindeskonfiguration der Version 1:

Extern Geräte :	LKE und/oder LSE
	Trommelspeicher und/oder Plattenspeicher
	Schnelldrucker (nur bei Karteneingabe)
Notwendiges ORG:	ORG I Befehlsvorrat 3o5
Notwendige MAKROS:	BEWA
	BSAU
	BSEI
	ENDE
	EXWA
	LKEI nur bei Lochkarteneingabe
	LSEF
	LSEV
	PSAU
	PSDE
	PSDEA
	PSDL
	PSDS
	PSEI
	PSSF
	PSSS
	PSVI
	SDAT nur bei Lochkarteneingabe
	SDAV " " "
	STRT

NAME: BEWEIS DATUM: 21.12.1972 EINGABEPROTOKOLL

```
*  10  PRINT
*  15  PRINT
*  20  PRINT'    TAB-TEST 3'
*  25  PRINT
*  30  GOSUB 200
*  50  FOR A = 0 TO 70 STEP 5/3
*  70  PRINT TAB(0),'*',TAB(70),'*',TAB(35),'*',
*  60  PRINT TAB(35-A),'*',TAB(35+A+(SGN(35-A)-1)*(A-35)),'*',
*  80  Y=SQR(35**2-(35-A)**2)
*  90  PRINT TAB(35-Y),'*',TAB(35+Y),'*',
*  100 PRINT TAB(Y),'*',TAB(70-Y),'*',
*  95  Y=SQR(35**2-(A-(SGN(A-35)+1)*(A-35))**2)
*  120 IF ABS(A-35).GT.0.05 THEN 135
*  130 GOSUB 200
*  135 IF A.LT. 69.5 THEN 140
*  138 GOSUB200
*  140 PRINT
*  150 NEXT A
*  160 END
*  200 FOR I= 0TO 70
*  210 PRINT TAB(I),'*',
*  220 NEXT I
*  230 RETURN
   RUN
```

TAB-TEST 3