Der Berliner Code.

Beschreibung des an der elektronischen Rechenanlage Z22 im Rechenzentrum Konstanz verwendeten Adressierverfahrens.

1. Einleitung.

Der "Berliner Code" ist ein Verfahren, mit Hilfe sog. "symbolischer Adressierung" das Anfertigen von Rechenprogrammen für die elektronische Rechenmaschine Z 22 zu erleichtern. Er ist im Recheninstitut der T.U. Berlin entwickelt worden von F.R.Güntsch, W. Kuzenko, G.Bruhn und K.Friedrich.

Eine Abänderung der Befehlsliste gegenüber dem sosnt an der Z 22 benutzten und unter dem Namen "Freiburger Code" bekannten Bezeichnungsschema und den ihm zugrunde liegenden Operationen wurde nicht vorgenommen, so daß im "Freiburger Code" geschriebene Programme nach wie vor voll benutzbar sind. Insbesondere lassen sich neben den symbolischen Adressen, über die im Folgenden Näheres berichtet wird, auch feste Adressen der dort bekannten Art weiter verwenden. Der Berliner Code stellt also gegenüber dem Freiburger eine Erweiterung, keine Abänderung dar.

Die Kenntnis des "Freiburger Code" wird im folgenden vorausgesetzt, da über ihn bereits Beschreibungen vorliegen. -

Sinn der "symbolischen Adressierung" ist es, Rechenbefehle (Speicherbefehle, Sprungbefehle usw.) bereits in endgültiger Form mit Adresse schreiben zu können, wenn die endgültige Lage der Speicher auf der Trommel noch nicht bekannt ist. Es besteht so insbesondere die Möglichkeit, in ein fertiges Programm noch einzelne Befehle oder auch größere Teile vor dem Einlesen in die Maschine einzufügen, ohne die Adressen der übrigen Befehle ändern zu müssen. Es wird ferner das sonst übliche und etwas mühselige Abzählen der Speicheradressen vermieden, die hier gar nicht bekannt zu sein brauchen.

Auch beim Benutzen von Unterprogrammen tritt durch symbolische Adressen eine wesentliche Vereinfachung ein, da die Lage der Unterprogramme auf der Trommel beim Programmieren ebenfalls noch nicht bekannt zu sein braucht, sondern beim Einlesen noch frei gewählt werden kann. -

Die symbolische Adressierung wird in zweierlei Schreibweise gebraucht, einerseits in Bezug auf Adressen innerhalb des gerade vorliegenden Programmes, andererseits in Bezug auf Adressen (meistens die Anfangsadresse) weiterer Programme, die von dem anzufertigenden z.B. als Unterprogramme zitiert werden.

2. Adressierung innerhalb eines Programms.

Innerhalb eines Programms läßt sich die Vereinfachung auf dem folgenden Wege verwirklichen. Neben den immer noch benötigten "festen Adressen" der bekannten Art (z.B. für Schnellspeicher) werden für Adressen Abkürzungen eingeführt, die laufend numeriert werden, ohne daß bei diesen Nummern die Einhaltung einer Reihenfolge oder das Benutzen aller Nummern obligatorisch wäre. Diese laufenden Nummern treten, unter Voranstellung eines Punktes, an die Stelle der sonst üblichen festen Adresse. Beispiele: A.1 , T.15 , F.7 . Die Stellen, auf die sich diese Adressen beziehen, werden durch dieselbe Nummer bezeichnet, indem nur die betreffende Zahl vor einem Gleichheitszeichen vor die Bezugsstelle gesetzt wird. Als Beispiel möge die Bildung einer Schleife dienen:

```
:
5=B7
  x
  B8
  +
  QQE.5
:
```

In diesem Beipiel führt der QQE.5 - Sprung an die Stelle zurück, an der 5=B7 steht, so daß (bei Erfüllung der in QQE abgefragten Bedingung) die Schleife nochmals durchlaufen wird.

Während des Einlesens des Programmes in die Maschine werden dann automatisch diese symbolischen Adressen durch die wirklichen Trommeladressen ersetzt.

Selbstverständlich brauchen die symbolischen Adressen nicht zur Bezeichnung von Bezugsstellen im eigentlichen Programm zu dienen, sondern können auch zum Zitieren von Zahlspeichern verwendet werden. In diesem Falle erfolgt die Eingabe der betreffenden Zahl wieder unter Voransetzung der Bezugszahl:

```
:
B.7
:
7=+2,5375o/+12
:
```

Auf der "rechten Seite", d.h. im Adressenteil eines Befehls, darf eine symbolische Adresse beliebig oft verwendet werden. Dies kann z.B. dann erwünscht sein, wenn von mehreren Stellen in denselben Programmteil gesprungen werden soll oder wenn derselbe Parameter an mehreren Stellen benötigt wird. Ein Beispiel dazu:

```
3=B.5
  :
PPE.3
  :
B.5
  :
YE.3
  :
 U.5
  :
5=-1273'
  :
```

Eine mehrfache Verwendung derselben Nummer af der linken Seite innerhalb desselben Unterprogrammes ist natürlich gefährlich. Sie würde bedeuten, daß die Bezugszahl mehrfach definiert wäre, was zu Widersprüchen führen könnte. Sofern in einem solchen Fall nicht, wie weiter unten beschrieben, ein Alarm ausgelöst wird, verwendet die Maschine die jeweils letzte der gegebenen Definitionen der Bezugsstelle.

Da die Unterprogramme eine Programms getrennt eingelesen und von dem hier betrachteten Vorprogramm getrennt verarbeitet werden, dürfen (und sollen) dieselben Zahlen als Bezugszahlen in allen Unter- bzw. Teilprogrammen wieder neu verwendet werden.

Es wurde bereits gesagt, daß die Bezugszahlen innerhalb eines Unterprogramms keine fortlaufende (d.h. lückenlose) Folge von ganzen Zahlen zu sein brauchen. Um Speicherplatz zu sparen, wird jedoch empfohlen, hiervon nicht zu viel Gebrauch zu machen und nach Möglichkeit alle Bezugszahlen zwischen 1 und einem möglichst kleinen 1 zu benutzen. Dies 1 muß, wie später beschrieben, vor

dem Einlesen der Unterprogramme der Maschine übermittelt werden.

Notwendig ist der Hinweis, daß jede Bezugszahl, die als Adresse eines Befehls benutzt wird, auch auf der linken Seite vorkommen, d.h. definiert werden muß. Dies braucht, wie z.B. im folgenden Fall, nicht selbstverständlich zu sein:

```
    :
    :
T.5
    :
    :
B.5
    :
    :
```

In dieser Befehlsfolge wird der in der Rechnung benutzten und durch die Bezugszahl gekennzeichnete Speicher nicht zur Bezeichnung eines Parameters o.ä. verwendet, der bereits vor Beginn der Rechnung mit dem Programm zusammen eingegeben wird, sondern der mit .5 bezeichnete Speicher wird nur während der Rechnung zur Unterbringung eines Zwischenergebnisses vorübergehend benutzt.

Das eben angeführte Beispiel würde von der Maschine nicht richtig eingelesen werden können, da die Zahl .5 nur auf der "rechten" Seite auftritt, aber nirgends "links" aufgeführt ist. In derartigen Fällen muß daher an das Programm noch angehängt werden:

```
    :
    :
5=o'
    :
    :
```

An Stelle von o' kann selbstverständlich eine beliebige belanglose Zahl eingesetzt werden, die bei der Rechnung ja sowieso überschrieben wird.

3. Kontrolle.

Sind Bezugszahlen (wie im eben gegebenen Beispiel) zwar als Adresse benutzt, aber nicht definiert worden, so kann dies durch eine in das Vorbereitungsprogramm eingebaute Kontrolle festgestellt werden. Zu diesem Zweck muß während des Einlesens eine an der Maschine befindliche Taste, die sog. "17-Taste", niedergedrückt werden.

Falls eine Adresse doppelt auftritt, druckt die Maschine beim zweiten Auftreten den Text "ADR k DOPPELT" , wo k die betreffende Bezugszahl ist, und stoppt das Einlesen. Nach Eingabe des ganzen Unterprogrammes wird dann das Vollzähligsein der Definitionen abgefragt. Fehlt eine von diesen, so druckt die Maschine "ADR k FEHLT" , wo k wieder die Bezugszahl ist.

4. Das Anschließen von Unterprogrammen.

Das Zitieren eines anderen Unterprogrammes im Adressenteil eines Befehls (z.B. beim Sprung auf dieses Unterprogramm) wird ähnlich gehandhabt wie das oben beschriebene Zitieren von Bezugsstellen innerhalb desselben Programmes. In diesem Fall wird die Katalognummer des betreffenden Programms, d.h. die Nummer, unter der es in der Kartei erfaßt ist, unter Voransetzen von zwei Punkten an die Stelle der Adresse gesetzt, z.B. F..714 .

Leider ist es auf diese einfache Art nur möglich, den Anfang, d.h. den ersten Speicher eines anderen Programms zu erreichen. Soll ein anderer Speicher des zweiten Programms zitiert werden, so

läßt sich ein Abzählen von Speicheradressen nicht vermeiden. Soll z.B. der n-te Speicher des Programms k erreicht werden, so kann das durch folgende Befehlsfolge erzielt werden:

```
    :
BC.. k
 T  s
Θ  GK  s+ n-1
  o'
    :
```

wo Θ der auszuführende Befehl und s die Adresse eines Schnellspeichers ist, der an der betreffenden Stelle verfügbar, d.h. sonst unbenützt ist. An die Stelle von n-1 muß die um 1 verminderte Zahl n gesetzt werden (ohne Klammern!). Ein Beispiel:

```
    :
BC..763
 T5
BGK5+6
  o'
    :
```

überführt den Inhalt des siebten Speichers von Programm 763 in den Akku. - Falls 763 eines der unten näher beschriebenen "fiktiven Programme"ist, wird hierin BC..763 ersetzt durch B..763 .

Es muß darauf hingewiesen werden, daß diese Befehlsfolge im fertigen Programm steht, daß sie also bei jeder Durchrechnung des betreffenden Programmteils wiederholt wird. Sie erfordert daher evtl. ein vorheriges Sicherstellen und nachtragliches Zurückholen des Akkumulatorinhaltes.

Gelegentlich tritt der Fall auf, daß ein Programm an derselben Stelle wahlweise verschiedene Unterprogramme aufrufen soll. Dies ist z.B. der Fall für Programme zur Berechnung des bestimmten Integrales einer Funktion. Dies Programm soll nach seiner Fertigstellung nicht nur auf eine einzige Funktion als Unterprogramm, sondern auf beliebige Funktionen anwendbar sein. In einem derartigen Fall wird im Integralprogramm ein sog. "fiktives Unterptogramm" zitiert, das auch eine Katalogzahl erhält, aber unter derselben Nummer von Fall zu Fall neu angefertigt wird und nur aus einem einzigen Befehl besteht, nämlich dem Sprungbefehl auf das im vorliegenden Fall gerade benötigte Unterprogramm.

Als Nummer des fiktiven Programmes wird die auf die Nummer des aufrufenden Programmes folgende Zahl genommen. Werden mehrere derartige Unterprogramme für dasselbe Unterprogramm benötigt, werden entsprechend mehr folgende Nummern reserviert. - Eine andere Möglichkeit besteht im letzten Falle darin, unter einer einzigen fiktiven Programmnummer der Reihe nach die Sprünge auf alle benötigten Unterprogramme einzugeben.

Ein Beispiel mit einem fiktiven Programm als Erläuterung.
Wir betrachten das Programm 513 als Hauptprogramm, das mehrfach auf das fiktive Unterprogramm 514 springt. Im vorliegenden Fall soll an dessen Stelle das Unterprogramm 600 treten. Wir haben dann in Programm 513 (wobei der "Vorspann" .18=513 später erläutert wird):

```
      :
      :
   .18=513
      :
      :
   F..514
      :
      :
```

```
.
:
F..514
:
:
F..514
:
:
```

Programm 514 besteht allein aus einem Befehl:

```
.o=514
F..600
```

Das Unterprogramm 600 wird durch diesen "Umweg" über 514 nicht beeinflußt. Insbesondere springt es nach dem Befehl E5 wieder unmittelbar nach 513 zurück, sofern der Schnellspeicher 5 nicht in ihm selbst überschrieben wurde.

Soll später einmal das Programm 513 auf ein anderes Unterprogramm, z.B. auf Nr. 675, angewandt werden, so wird das Programm 514 ausgetauscht gegen

```
.o=514
E..675
```

5. Eingabe eines Programmsystems.

Bei der Eingabe eines Programmsystems muß der Reihe nach auf dem bzw. auf mehreren Lochstreifen stehen (wobei Anführungszeichen nicht zum Lochstreifentext gehören):

1) "T1000T" und anschließen die Angaben für das Druckschema (Tabellenanordnung, Stellenzahl usw.) nach den Regeln des Freiburger Code,

2) "T m T" , wo m die erste Speicheradresse ist, in der der Beginn des Programms abgesetzt werden soll,

3) ". l_{max}" , wobei an die Stelle von l_{max} die größte Bezugszahl tritt, die in einem der nun folgenden Unterprogramme vorkommt.

4) ".l=p" , wobei l die größte Bezugszahl ist, die im jetzt einlaufenden Unterprogramm auftritt, und p die Katalognummer des Unterprogramms.

5) Es folgen die Befehle des Unterprogramms.

6) " .. " Dieser zweifache Punkt dient als Schlußzeichen jedes Unterprogramms.

Bei den Bibliotheksprogrammen stehen 4), 5), 6) bereits fertig auf den Lochstreifen. Die Eingabe von 4), 5), 6) wird für jedes Unterprogramm wiederholt.

7) " ..p " , falls nach der Eingabe unmittelbar mit dem Start des Programms Nr. p begonnen werden soll. Damit ist die Eingabe beendet.

Ein Beispiel soll das Ganze veranschaulichen. Wir nehmen an, daß in Gleitkomma 7-stellig in einer 3-spaltigen Tabelle ausgedruckt werden soll. Weiter nehmen wir an, daß die Unterprogramme 111 (mit den symbolischen Adressen 1 bis 7), 112 (mit 1 bis 15) und 133 (mit 1 bis 12) benutzt werden und ab Speicher Nr. 2o48 eingelesen werden sollen. Der Start soll mit Programm Nr. 111 erfolgen. Dann lautet der vollständige Lochstreifentext:

```
"  T 1o3o T   7' o' o' 3' 2' o' 1'
T 2o48 T  .15
.7=111    (Dann Befehle des Progr. 111)       ..
.15=112   (Dann Befehle des Progr. 112)       ..
.12=1333  (Dann Befehle des Unterprogr. 133)  ..
.. 111  "
```

6. Regeln für das Aufstellen von Unterprogrammen.

Die Verwendung des Berliner Code erhält ihren vollen Wert durch das Vorhandensein einer möglichst umfangreichen Programmbibliothek. Zur Einrichtung dieser Bibliothek, die den Benutzern der Anlage zugute kommen wird, wird deren Mithilfe insofern in Anspruch genommen, als von ihnen erwartet wird, daß sie die von ihnen gerechneten Programme als Unterprogramme möglichst im Berliner Code aufstellen und in jedem Falle nach ihrer Verwendung der Programmbibliothek zur Verfügung stellen. Es muß die weitere Forderung gestellt werden, daß bei der Herstellung der Programme gewisse Regeln der Normung eingehalten werden.

Falls Programme "optimal" programmiert werden müssen, kann u.U. die Verwendung des Berliner Code unzweckmäßig sein, da dann ja ohnehin die Lage der Speicher auf der Trommel festgelegt sein muß. Um auch in diesem Falle die einheitliche Normung der Bibliothek einzuhalten, kann das Programm nach dem Einlesen auf die Magnettrommel mit Hilfe eines speziellen Druckprogrammes im Berliner Code wieder ausgedruckt bzw. ausgelocht werden.

Es sollten jedoch bei jedem Programm unbedingt die folgenden Regeln eingehalten werden:

1) Mathematisch selbständige Programmteile (z.B. Berechnung von Funktionen, Iterationsverfahren usw.), die genügend groß sind (z.B. mehr als 4o - 5o Befehle umfassen), sollten als Unterprogramme gefertigt werden, d.h. durch einen F - Sprungbefehl aufgerufen werden und auf der Trommel getrennt vom Hauptprogramm stehen. Wird der Schnellspeicher 5 im Laufe des Unterprogrammes benutzt, ist er am Anfang des Unterprogrammes zu räumen:

B5
T k

Hier kann k ein Schnellspeicher sein, wenn einer frei ist, oder eine symbolische Adresse. Der Rücksprung am Ende des Unterprogrammes soll durch E k erfolgen, von dort springt die Maschine dann automatisch in das Hauptprogramm zurück.

Sollte aus programmtechnischen Gründen diese Regelung nicht eingehalten werden können, so muß für die Bibliothek ein getrenntes Exemplar des Lochstreifens angefertigt werden, das den geforderten Bedingungen genügt. Dies ist durch Umkopieren der Lochstreifen leicht zu erreichen.

2) Jedes der Bibliothek übergebene Unterprogramm soll durch Sprung auf seinen ersten Speicher gestartet werden können. Liegt der Einlauf des Unterprogrammes an andere Stelle, so muß vor seinem ersten Speicher noch der E - Sprungbefehl auf disen Einlauf eingefügt werden, der dann die Rolle des ersten Speichers übernimmt. Beispiel:

```
                .6=999
Neuer Anfang:     E.6
Alter erster Sp.: 2=B.5
Alter Einlauf:    6=T8
                   :
                   :
                  QQE.2
                   :
                   :
                   E5
```

3) Im Hauptprogramm berechnete Parameter oder Argumente, die das Unterprogramm benötigt, sollen soweit möglich im Schnellspeicher übernommen werden,also vor dem Sprung auf das Unterprogramm dort stehen. Dasselbe gilt für den Rücksprung. Wird eine Funktion einer Variablen für einen Wert des Argumentes berechnet, so sollte vorher das Argument in Schnellspeicher 4 (Akkumulator), am Schluß des Unterprogrammes der Funktionswert in 4 und möglichst auch in 6 stehen. - Müssen viele Parameter übernommen werden (z.B. eine Matrix) oder ist aus anderen Gründen ihre Überhahme im Schnellspeicher nicht ratsam, so bleibt es dem Programmierer überlassen, evtl. nach Rücksprache mit dem Rechenzentrum eine möglichst günstige Lösung zu finden.

4) Jedes Unterprogramm sollte so angefertigt werden, daß es - unter Einhaltung der sonstigen Regeln - möglichst wenig Speicher benötigt. In erster Linie sollen die Schnellspeicher 2 bis 15, dann die übrigen Schnellspeicher und schließlich geeignete Trommelspeicher für Zwischenergebnisse benutzt werden.

5) Für jedes Unterprogramm sollen auf dem Lochstreifen erst diejenigen Speicher gefüllt werden, die während der Rechnung unverändert bleiben (z.B. Befehle, Konstante) und dann erst, durch ein Stück Leerband getrennt, diejenigen, die während der Rechnung beschrieben bzw. geändert werden. Beide Teile sollen an verschiede-

nen Stellen der Trommel stehen können.

Diese Regelung ist nicht obligatorisch, kann aber oft sehr nützlich sein. Ihr Sinn ist es, bei größeren Programmen, die lange Rechenzeit benötigen, die unveränderlichen Teile vor versehentlicher Zerstörung durch Überschreiben während der Rechnung zu schützen. Dies ist dadurch möglich, daß auf der Trommel Blocks von jeweils 512 Speichern abgeschaltet werden können, so daß das dort gespeicherte nur noch zur Ablesung zur Verfügung steht.

6) Eine Zerstückelung von Unterprogrammen in getrennt stehende Teile über 5) hinaus soll nur dann durchgeführt werden, wenn diese Teile wieder selbständige "Unterunterprogramme" sind. Es kann natürlich der Fall eintreten, daß bei großen Programmen, optimaler Programmierung usw. jeder Speicherplatz ausgenutzt werden muß. In diesem Falle sollten aber für die Bibliothek die Einzelteile auf einem einzigen Lochstreifen in benutzbarer Weise zusammenkopiert werden.

7) Für jedes Unterprogramm muß in Zusammenarbeit mit dem Rechenzentrum ein Merkblatt ausgefüllt werden, das alle für spätere Benutzer wichtigen Angaben enthält, z.B. die Zahl der benötigten Speicher, Rechenzeit, mathematische Methode usw. -

Die angeführten Regeln könnten den Eindruck einer starken zusätzlichen Belastung erwecken. Wir weisen jedoch mit Nachdruck darauf hin, daß ihre Befolgung dem Programmierer selbst seine Arbeit wesentlich erleichtert. Dies gilt insbesondere für das Ausprüfen umfangreicher Programme. Gerade in diesem Falle ist es sehr wertvoll, wenn alle Programmteile getrennt geprüft und nach einem einheitlichen Schema behandelt werden können.