Programmgesteuerte Rechenautomaten

H. GÖTZKE

DR.-ING. HORST GÖTZKE

Programmgesteuerte Rechenautomaten

Grundlagen
Aufbau
Arbeitsweise
Anwendungen für Digitalrechner
Datenverarbeitungsanlagen
Analogrechner und
Hybridsysteme

Mit 131 Bildern, 10 Tabellen und 1 Beilage

5., verbesserte Auflage

fv

VEB FACHBUCHVERLAG LEIPZIG

Redaktionsschluß 31. 1. 1970
ES 20 A (20 C 1, 19 B 1)

Satz u. Druck: VOB Fachbuchdruck Naumburg (Saale), IV/26/14, Auftr.-Nr. 29
Veröffentlicht unter der Lizenznummer 114-210/125/70
5,50

Vorwort zur 5. Auflage

Der VII. Parteitag der Sozialistischen Einheitspartei Deutschlands hat der Entwicklung und dem Einsatz der elektronischen Datenverarbeitung gewaltigen Auftrieb gegeben. Dies wurde durch die Weisungen und Programme der Regierung in Auswertung der Beschlüsse, u. a. auch durch das 10. und 11. Plenum, unterstrichen. So gibt es grundsätzliche Festlegungen über die Entwicklung eigener Datenverarbeitungssysteme und dafür notwendiger Programme und Hilfsmittel. Eine Vielzahl an Fachkadern muß ausgebildet werden, und die Lehr- und Studienprogramme der Hoch- und Fachschulen sowie der allgemeinbildenden Schulen haben dem Rechnung getragen.

Die elektronische Datenverarbeitung greift jedoch auch in die Sphäre nahezu jedes einzelnen ein, weil seine Arbeit mittelbar oder unmittelbar durch sie beeinflußt wird. Dazu ist es aber nötig, daß sich jeder moderne Mensch umfangreiches Wissen über die elektronische Datenverarbeitung aneignen muß. Dabei soll unser Buch helfen. Es wendet sich an staatliche Leiter, Funktionäre der Parteien und Massenorganisationen, Techniker, Ökonomen und Verwaltungsangestellte, aber auch an Lehrer, Studenten und Schüler, um ihnen erste, aber zur Beurteilung ausreichende Vorstellungen über Möglichkeiten und Grenzen der Entwicklung und des Einsatzes programmgesteuerter Rechenautomaten bzw. Datenverarbeitungsanlagen zu vermitteln. Dabei wird deutlich, welchen Weg die Rechentechnik der DDR in den letzten Jahren gegangen ist. Auch in der 5. Auflage unseres Buches ging es vor allem um eine anschauliche und verständliche Darstellung. Dennoch stellt die Lektüre einige Anforderungen an den Leser. Deshalb ist es zu empfehlen, Beispiele und Aufgaben nachzurechnen.

Beim Programmieren erfolgt die Arbeit in der Maschinensprache nochmals am Beispiel des Cellatron SER 2, weil immerhin einige hundert Geräte dieses Typs bei uns im Einsatz sind. Außerdem wurde eine Einführung für den Cellatron D 4a aufgenommen. Die moderne Entwicklung von Programmier-

sprachen fand breiteren Raum. Die Anwendung beschränkt sich jedoch lediglich auf die Skizzierung einiger Probleme, da sonst der Rahmen dieses Buches gesprengt worden wäre.

Der Verfasser möchte vor allem den vielen Lesern danken, die Zuschriften zu den vorangegangenen Auflagen eingesandt hatten. Viele ihrer Anregungen werden sie in der vorliegenden Auflage wiederfinden. Danken möchte ich wiederum Herrn Dipl.-Math. Karl-Heinz Müller für seine umfangreichen Hinweise. Besonderer Dank gebührt dem Verlag, der bereitwillig auf viele Wünsche einging.

Horst Götzke

Inhaltsverzeichnis

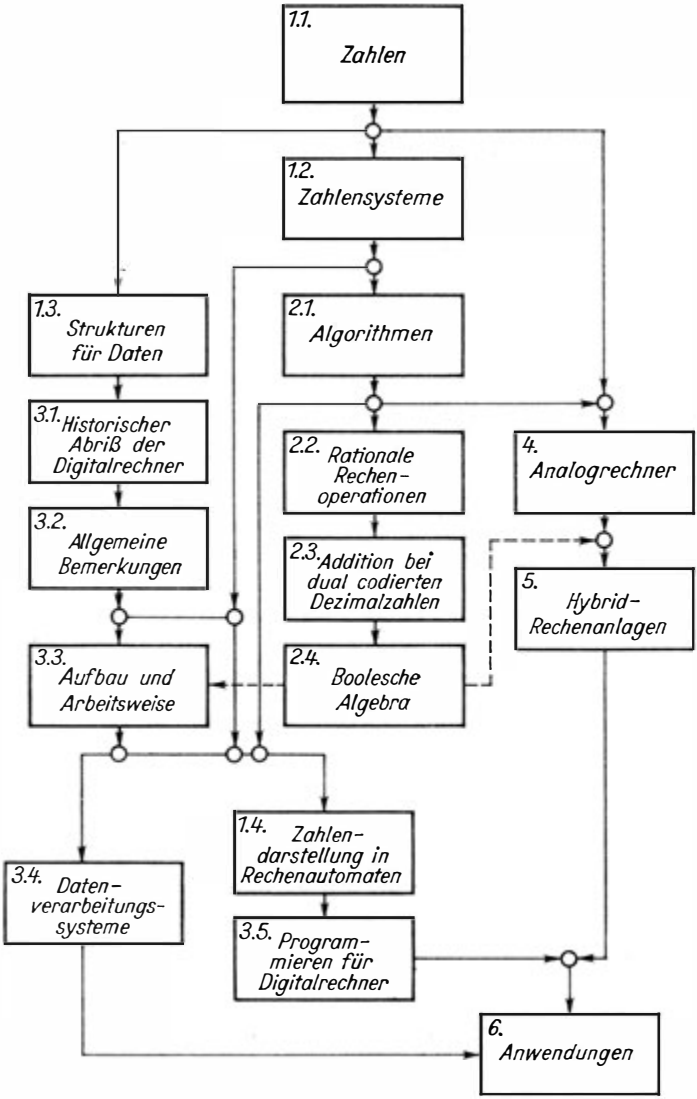
0. Einleitung

Programmgesteuerte Rechenautomaten werden heute überall in Theorie und Praxis benötigt. Es gibt keine wissenschaftliche Disziplin, keinen Zweig der Volkswirtschaft und kaum einen Bereich des gesellschaftlichen Lebens, in denen sie nicht bereits erfolgreich eingesetzt wurden.

Versuchen wir erst einmal die Frage zu ergründen, um welche Problematik es sich eigentlich beim Einsatz programmgesteuerter Rechenautomaten handelt. Es geht letzthin darum, daß die Menschen durch kybernetische Maschinen, und dazu gehören vorrangig die programmgesteuerten Rechenautomaten, auch von einem bestimmten Teil der geistigen Arbeit befreit werden. Bei den körperlichen Arbeiten, insbesondere für die schweren, schmutzigen und besonders gesundheitsschädigenden Arbeiten, bemühen wir uns seit über einem Jahrhundert, sie durch Maschinen und Mechanismen ausführen zu lassen. Anders bei den geistigen Arbeiten. Hier sind wir erst am Anfang einer allerdings sehr vielversprechenden Entwicklung.

Nicht alle geistige Arbeit kann jedoch von Maschinen oder Mechanismen abgenommen werden. Das ist nur für sogenannte formale geistige Arbeit möglich, die aber gerade diejenigen Prozesse umfaßt, die langweilig und ermüdend sind. Sie können routinemäßig erledigt werden und entsprechen dem Schöpfungsdrang der Menschen nur wenig. Andererseits ist dieses Gebiet sehr umfassend. Es reicht von elementaren Rechenprozessen bis beispielsweise zur Übersetzung aus einer lebenden Sprache in eine andere, vom Schachspielen in vereinfachter Form, der Bestimmung von Krankheiten (Diagnostik), dem Übersetzen völlig

unbekannter Sprachen und Symboliken bis zum Komponieren einfacher Grundmelodien, dem Steuern von Maschinen, Lenken von Produktionsabläufen und vielem anderen mehr.
Alle diese Prozesse können durch kybernetische Mechanismen und Maschinen ausgeführt werden. Der Mensch wird dann für die Entwicklung, Konstruktion und den Bau der Geräte, für die Vorbereitung des Arbeitsprozesses selber und zur Überwachung der Arbeit frei. Außerdem obliegen ihm alle Aufgaben, die von den kybernetischen Maschinen nicht bewältigt werden können. Er erhält so die Möglichkeit, sich vorwiegend mit der ihm gemäßen schöpferischen, phantasievollen geistigen wie körperlichen Arbeit zu beschäftigen. Das befähigt ihn zu ungeahnt hohen Leistungen, und er kann durch Anspannung aller seiner schöpferischen Fähigkeiten einen Wohlstand schaffen, wie wir ihn uns zur Zeit nur schwer vorstellen können.
Unser Buch soll dazu beitragen, Kenntnisse auf einem Teilgebiet dieser Entwicklung, der Arbeitsweise und Anwendung programmgesteuerter Rechenautomaten, zu vermitteln.
Die Darlegungen in den einzelnen Abschnitten bedingen einander zum Teil. Sie sind vielfach aber auch „für sich" verständlich, wenn der Leser sich nur für Teilprobleme interessiert. In der umstehenden Skizze sollen diese Zusammenhänge angedeutet werden.


1.1. Zahlen
1.2. Zahlensysteme
1.3. Strukturen für Daten
2.1. Algorithmen
3.1. Historischer Abriß der Digitalrechner
2.2. Rationale Rechen-operationen
4. Analogrechner
3.2. Allgemeine Bemerkungen
2.3. Addition bei dual codierten Dezimalzahlen
5. Hybrid-Rechenanlagen
3.3. Aufbau und Arbeitsweise
2.4. Boolesche Algebra
1.4. Zahlen-darstellung in Rechenautomaten
3.4. Daten-verarbeitungs-systeme
3.5. Programm-mieren für Digitalrechner
6. Anwendungen

1. Zahlen und Zahlensysteme

Zahlen spielen eine bedeutende Rolle in unserem Leben. Wenn wir messen und wägen oder auf andere Weise versuchen, Vorgänge quantitativ zu erfassen, bedienen wir uns der Zahlen. Aber auch in der Gegenüberstellung qualitativer Merkmale werden wir bestrebt sein, die bekannten Fakten zahlenmäßig zu erfassen, zu berechnen und hieraus Schlüsse zu ziehen. Es sei nur an die Bewertung in der Schule erinnert.

So wenden wir Zahlen auf nahezu alle Gebiete des menschlichen Lebens an und erforschen die hierfür geltenden Gesetzmäßigkeiten in einer eigenen Wissenschaft, der Mathematik.

Zahlen spielen verständlicherweise in der maschinellen Rechentechnik ebenfalls die zentrale Rolle. Über Rechenautomaten können wir viel schneller und mit geringerer Fehleranfälligkeit zu Schlüssen kommen, wenn sich die Fakten zahlenmäßig erfassen lassen. Auch gelingt es uns erst mit Hilfe der modernen Rechentechnik, eine solche Anzahl von Zahlen, hier auch Daten genannt, zu verarbeiten, daß viele praktische Probleme nur mittels der Rechenautomaten sinnvoll bewältigt werden können.

Dies soll an einem einfachen Beispiel demonstriert werden: Wir haben uns gewiß schon öfter über eine schlechte Wettervorhersage geärgert. Das Problem ist jedoch nicht so einfach. Es sind zu viele Faktoren, die auf das Wetter Einfluß nehmen. Einigermaßen fundiert läßt sich eine kurzfristige Wettervorhersage durchführen, wenn man diese Einflüsse über ein lineares Differentialgleichungssystem erfaßt, das bei seiner Lösung auf ein System von 300 linearen Gleichungen mit 300 Unbekannten

führt. Die Methode zur Lösung eines solchen Systems ist seit mehreren Jahrhunderten bekannt. Aber – es sind $300^2 = 90000$ Werte zu verarbeiten und $300^3 = 27\,000\,000$ Rechenoperationen auszuführen.
Ein fleißiger Rechner, der mit einer modernen Tischrechenmaschine durchschnittlich 2 Operationen in der Minute bewältigen kann, würde bei normal üblicher Arbeitszeit rund 90 Jahre benötigen, um das Ergebnis zu erhalten. Bis dahin dürfte dies Ergebnis für die Wettervorhersage jedoch nicht mehr von Interesse sein. Verwenden wir jedoch einen modernen programmgesteuerten Rechenautomaten, so liegt das Ergebnis bereits nach wenigen Minuten vor, und die Meteorologen haben die Unterlage für eine weitaus exaktere Wettervorhersage. Allerdings sind in der maschinellen Rechentechnik einige Besonderheiten der Zahlen und des Rechnens zu berücksichtigen.

1.1. Zahlen

Es war wohl die größte geistige Leistung in der Frühgeschichte der Menschheit, die Zahlen begrifflich zu erfassen. Dieser Prozeß erstreckte sich über Jahrhunderte und führte von den benannten Größen zum abstrakten Zahlbegriff.
Vor Erfassen des Zahlbegriffes gab es lediglich Vorstellungen über Mengen und Zuordnungen, wie wir sie auch bei kleinen Kindern beobachten können. Hieraus entwickelten sich erste benannte Größen, z. B. ein Baum, zwei Menschen, drei Hütten usw. Diese benannten Größen umfaßten nur die ersten der natürlichen Zahlen. Ging es über die größten der benannten Werte hinaus (das war in sehr frühen Zeiten schon bei vier Gegenständen erreicht), so benutzte man die Mengenbezeichnung „viel".
Der entscheidende, aber ungeheuer schwierige Schritt war dann, aus einer Vielzahl benannter Größen, z. B. drei Bäumen, drei Steinen, drei Bergen, drei Rehen, drei Hütten, drei Jägern, das über die Benennung hinausgehende gemeinsame Merkmal, nämlich die 3, begrifflich als abstrakte Zahl zu erfassen.
So entstanden im historischen Prozeß die natürlichen Zahlen 1 – 2 – 3 – 4 – 5 –, aus denen sich jedes Zahlensystem logisch entwickeln läßt.

Mit dem Erfassen der abstrakten Zahl haben wir die Möglichkeit, über die Zuordnung das gemeinsame Zahlenmerkmal von einem Gegenstand auf den anderen zu übertragen.
Wollte beispielsweise eine Sippe die Anzahl der Tiere ihrer Herde erfassen und unter Kontrolle behalten, so mußte sie keineswegs etwa die Tiere jeden Tag zählen. Sie ordnete jedem Tier einen Stein zu. Wurde ein Tier geboren, kam ein weiterer Stein dazu. Wurde ein Tier geschlachtet, mußte ein Stein weggenommen werden. Die gesamte Entwicklung des Tierbestandes in der Herde konnte nun auf diese Weise an Steinen verfolgt werden.
Damit sind wir auf ein neues Problem gestoßen, das Schreiben von Zahlen, denn die Steine stellten gewissermaßen eine Liste des Bestandes der Herde dar.
Primitiv könnten wir für jede Eins einen Strich zeichnen, einen Stein in ein bestimmtes Fach legen, einen Knoten in eine Schnur schürzen, wie es die Inka bei ihrer Schrift, dem Quipu, taten, oder eine Kerbe in einen Stock schnitzen, entsprechend den Kerbhölzern bei einigen Südseestämmen.
Damit lassen sich jedoch größere Zahlen nur schwer erfassen, und die gesamte Schreibweise wird äußerst unübersichtlich. Eine erste Abhilfe kann erfolgen, indem wir die Anordnung der Striche, das Schriftbild, übersichtlicher gestalten. Fünf Striche werden so angeordnet, daß vier parallel verlaufen und der fünfte diese vier schräg durchstreicht.
Damit haben wir eine Bündelung in unserem Zahlenbereich vorgenommen. Das führt schließlich zu den Zahlensystemen (Bild 1).
Zahlen lassen sich einmal in der beschriebenen Weise durch Anhäufen von Gegenständen darstellen. Dann kann man diese

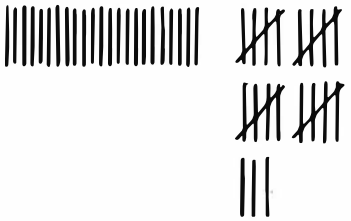


Bild 1. Primitivste Zahlenschreibweise und einfachste Form der Bündelung

Gegenstände den Fingern (digits) zuordnen. Die entsprechenden Rechenhilfsmittel nennt man *Digitalrechner*[1]. Wir können für die Größe einer Zahl aber auch eine physikalische Analogie finden, z. B. die Länge einer Strecke. Rechenhilfsmittel, die mit derart dargestellten Zahlen arbeiten, nennt man *Analogrechner*.

1.2. Zahlensysteme

Die Zahlen werden durch ein Zahlensystem auch in der Darstellungsweise für den praktischen Gebrauch aufbereitet.
An Zahlensysteme und deren Darstellungsweise werden große Anforderungen gestellt:
Die Zahlen müssen sich in einem Zahlensystem schnell und bequem schreiben lassen.
Das Zahlensystem und seine Bezeichnungsweise müssen leicht erlernbar sein.
Es muß, zumindest theoretisch, jede noch so große und auch jede noch so kleine Zahl darstellbar sein.
Man muß mit den Zahlen leicht rechnen können.
Die Erfüllung dieser Forderungen verlangt aber, daß die von uns bereits besprochene Bündelung vorgenommen wird. Bei unserem Beispiel hatten wir uns für eine Bündelung zu fünf entschlossen. Andererseits haben wir zwei Fünferbündel zu zehn zusammengefaßt, um eine möglichst günstige Darstellung zu erhalten. Könnten wir nicht gleich Zehnerbündel herstellen?
In der Tat, theoretisch ist jede positive ganze Zahl mit Ausnahme der Eins als „Bündelzahl" geeignet: 2, 5, 10, 12 oder eine andere.
Aus rein praktischen Erwägungen haben jedoch die Zahlen Fünf und Zehn den Vorzug, weil wir an einer Hand eben fünf Finger und insgesamt zehn Finger zur Verfügung haben. Ordnen wir dann den zu untersuchenden Zahlen die entsprechende Fingerzahl zu, haben wir gleich eine komplette Rechenmaschine. Dabei wollen wir unter einer Rechenmaschine ein Hilfsmittel verstehen, das uns die mechanische Ausführung von Rechnungen ermöglicht. Von Rechenmaschinen können wir also einzelne

[1] *digit* – engl. – Finger, Dezimalziffer

Rechenoperationen, wie Addition, Subtraktion oder Multiplikation, selbständig, ohne zusätzliche Denkprozesse, ausführen lassen. Im betrachteten Beispiel geschieht das durch Krümmen oder Strecken einzelner Finger und primitives Abzählen.
Aus den oben angedeuteten Überlegungen hat sich die „Zehn" als Bündelzahl besonders bewährt.
Was geschieht aber, wenn wir soviel Zehnerbündel haben, daß die Gesamtzahl wiederum unübersichtlich ist?
Dann bündeln wir nochmals, und zwar zehn Zehnerbündel zu einem Hunderterbündel. Danach zehn Hunderterbündel zu einem Tausenderbündel und so weiter.
Jetzt brauchen wir für jede Bündeltype nur eine spezielle Bezeichnung einzuführen, und schon haben wir ein voll ausgebautes Zehnersystem, wie es beispielsweise bereits die alten Ägypter hatten.

1.2.1. Zehnersystem in ägyptischen Hieroglyphen

Die Ägypter bündelten ihre Zahlen nach Zehnern. Sie führten folgende Einzelzeichen ein:

1 einen Strich für die Einheit 1

10 ein Hufeisen für das Zehnerbündel

100 eine Meßlinie für das Hunderterbündel

1 000 eine stilisierte Lotosblume für das Tausenderbündel

10 000 einen Finger für das Zehntausenderbündel

100 000 eine stilisierte Kaulquappe für das Hunderttausenderbündel

1 000 000 eine stilisierte Göttin für das Millionenbündel.

Bild 2. Zahlen in ägyptischen Hieroglyphen

Damit konnten sie jede Zahl schreiben. Sie mußten nur darauf achten, daß niemals mehr Zeichen von einer Bündeltype auftraten als zehn, denn dann mußten sie diese zehn Bündelzeichen herausnehmen und durch das nächsthöhere Bündelzeichen ersetzen.

Bild 3. Darstellung der Zahl 341 027 in ägyptischen Hieroglyphen

Die Stellung der Bündelzeichen, ihre Reihenfolge und Anordnung war dabei völlig uninteressant. Auch störte es nicht, daß keine Null existiert, denn war in einer Zahl eine Bündeltype (beispielsweise Hunderter) nicht vertreten, so wurde sie nicht geschrieben.
Dieses System der Ägypter ist ein voll ausgebautes Zehnersystem. Dennoch ist es unbefriedigend. Für jeden neuen Bündeltyp benötigt man auch ein neues Zeichen. Das erschwert die Darstellung großer Zahlen und deren Lesbarkeit.
Daher muß man für den Aufbau eines modernen Zahlensystems weitere Verallgemeinerungen schaffen. Auch hierfür erst ein historisches Modell.

1.2.2. Positionssystem der Maya

Der mittelamerikanische Indianerstamm der Maya war ein hochentwickeltes Kulturvolk, das bereits im 6. und im 11. bis 13. Jahrhundert seine Blütezeiten durchlebte. Es besaß ein im modernen Sinne voll ausgebautes Zahlensystem, das wir als Positionssystem bezeichnen. Die Maya bündelten zu zwanzig Einheiten, weil bei ihnen die Finger und Zehen zur Zahlenerfassung herangezogen wurden. In ihrer Schreibweise gingen sie jedoch anders vor, als wir es bei den Ägyptern besprochen hatten. Sie unterschieden 20 Zahlzeichen oder Ziffern. Nach

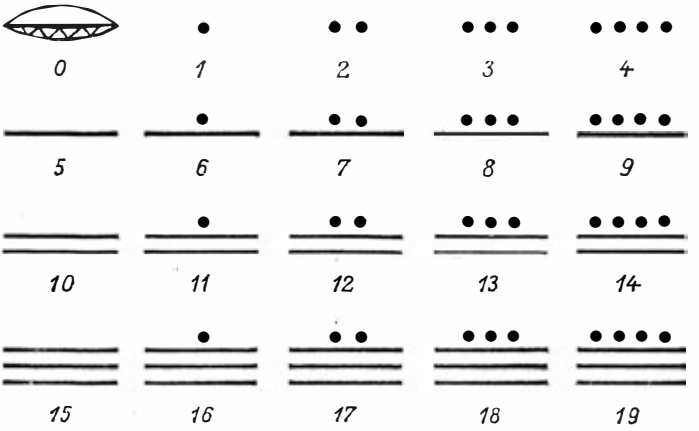


Bild 4. Die zwanzig Ziffern des Positionssystems der Maya

diesen Zeichen erhielt jedes Symbol einen entsprechenden Ziffernwert. Aber auch die Stellung der Ziffer in der Gesamtzahl spielte eine Rolle. Es war eine feste Schreibweise vorgeschrieben, von rechts nach links oder von unten nach oben. So erhielt jede Ziffer zusätzlich zum Ziffernwert noch einen Stellenwert nach folgender Vorschrift:

1. Stelle △	1 =	1
2. Stelle △	20 =	20
3. Stelle △	$18 \cdot 20$ =	360
4. Stelle △	$18 \cdot 20^2$ =	7200
5. Stelle △	$18 \cdot 20^3$ =	144000
6. Stelle △	$18 \cdot 20^4$ =	2880000

In jedem Stellenbereich ergibt sich somit der Wert als Produkt des Ziffernwertes mit dem Stellenwert. Steht beispielsweise in der dritten Stelle eine 10, so liefert das den Wert 3600. Die Gesamtzahl ergibt sich dabei als Summe aller Produkte aus Ziffern- und Stellenwert.

Beispielsweise bedeutet die Zahl

$3 \cdot 7200 \triangleq 21600$

$12 \cdot 360 \triangleq 4320$

$0 \cdot 20 \triangleq 0$

$5 \cdot 1 \triangleq 5$

25925

Bild 5. Die größte bekannte Zahl in der Schreibweise des Positionssystems der Maya (25925)

Die eigenartige Bündelung der 3. Stufe ergab sich deshalb, weil bei den Maya, wie auch bei allen alten Mittelmeervölkern, das Jahr 360 Tage umfaßte. Es war in 18 Monate zu je 20 Tagen eingeteilt. Um diese Jahreseinteilung leicht erfassen zu können, wurde entsprechend vom strengen Zwanzigersystem abgewichen. Das Positionssystem der Maya hat den Vorteil, daß wir mit einer eng begrenzten Zahl von Ziffern auskommen. Danach lassen sich alle noch so großen Zahlen schreiben, ohne daß wir neue Ziffern oder Zahlzeichen hinzufügen müssen.

1.2.3. Positionssysteme

Die Gesetzmäßigkeiten, die wir beim Zahlensystem der Maya erkannt haben, brauchen wir nur zu verallgemeinern, um ein modernes Zahlensystem zu erhalten.
Wir wollen diese Gesetzmäßigkeiten zusammenstellen:

1. Gegeben sei eine ganze positive Zahl, größer als 1, nach der gebündelt wird. Diese Zahl soll Basiszahl heißen und mit B bezeichnet werden.
2. Dann werden $B - 1$ Ziffern und die Null benötigt, was wir wie folgt darstellen können

$$z_i = (0, 1, 2, \ldots, (B - 1))$$

(i soll einen Index andeuten, den wir später zur Zahlenschreibweise benötigen).

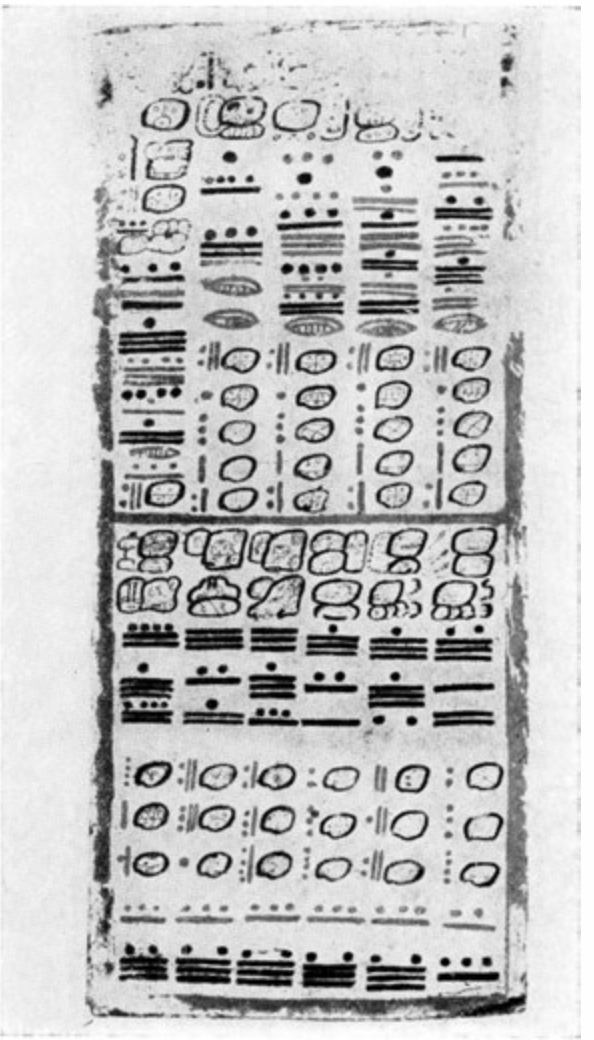


Bild 6. Blatt LI der Dresdener Maya-Handschrift

3. Jede Ziffer erhält entsprechend der Bezeichnung den Ziffernwert.

4. Zahlen werden in fest vorgeschriebener Reihenfolge geschrieben, und zwar

von rechts nach links,

dabei erhält jede Stelle ihren Stellenwert.
Von rechts beginnend ist der Stellenwert der

1. Stelle $= B^0$ $(10^0 = 1)$
2. Stelle $= B^1$ $(10^1 = 10)$
3. Stelle $= B^2$ $(10^2 = 100)$

n-te Stelle $= B^{n-1}$ (10^{n-1})

Die Folge der Ziffern kann nach einer Markierung (Komma) auch zur rechten Seite hin fortgesetzt werden. Dabei erhalten sie die Stellenwerte für

1. Stelle nach dem Komma $= B^{-1}$ $(10^{-1} = 0{,}1)$
2. Stelle nach dem Komma $= B^{-2}$ $(10^{-2} = 0{,}01)$
3. Stelle nach dem Komma $= B^{-3}$ $(10^{-3} = 0{,}001)$

m-te Stelle nach dem Komma $= B^{-m}$ (10^{-m})

(Die Stellenwerte des Zehnersystems wurden zur Übersicht angefügt.)

5. Die Zahl ergibt sich dann wie folgt:
Für jede Stelle wird der Ziffernwert mit dem Stellenwert multipliziert

$$z_i \cdot B^i$$

Es wird über alle Stellen summiert, beginnend bei der höchsten Stelle, deren Ziffer von Null verschieden ist:

$$Z = z_n \cdot B^n + \ldots + z_1 \cdot B^1 + z_0 \cdot B^0 + z_{-1} \cdot B^{-1} + \ldots + \\ + z_{-m} \cdot B^{-m}$$

Dabei können die z_i (für $i = -m, \ldots, -1, 0, +1, \ldots, n$) nur Ziffern sein, also nur die Werte 0, 1, 2, . . . , $B-1$ annehmen.

Der Index i durchläuft hier die Werte $-m, \ldots -1, 0, 1, \ldots, n$. Er stimmt mit dem jeweiligen Exponenten der Basiszahl überein. Wenn wir also im nachfolgenden Abschnitt 6 die Potenzen der Basiszahlen nicht mehr mitschreiben, können wir durch Abzählen des Index vom Komma nach rechts oder links den jeweils zugehörigen Stellenwert rekonstruieren.[1]

6. Für die Schreibweise wird vereinbart, daß die Potenzen von B und das Summationszeichen nicht mitgeschrieben werden:

$$Z = z_n, \ldots, z_1, z_0, z_{-1}, \ldots, z_{-m}$$

Theoretisch gibt es unendlich viele Positionssysteme. Alle natürlichen Zahlen, die größer als 1 sind, können ja als Basiszahl genommen werden. Praktisch jedoch, insbesondere in der maschinellen Rechentechnik, sind vier Systeme von Bedeutung:

das Hexadezimalsystem,
das Dezimalsystem,
das Oktalsystem,
das Dual- oder Binärsystem.

1.2.4. Zahlensysteme der Rechentechnik

Aus den im vorigen Abschnitt zusammengestellten allgemeinen Gesetzmäßigkeiten für Zahlensysteme können wir uns nun leicht die spezifischen Systeme der Rechentechnik erarbeiten. Als Bezugssystem betrachten wir dabei das gebräuchliche Dezimalsystem.

1. Die genannten Zahlensysteme verwenden folgende Basiszahlen:

Hexadezimalsystem	$B = 16$
Dezimalsystem	$B = 10$
Oktalsystem	$B = 8$
Dualsystem	$B = 2$

[1] Wenn wir exakt vorgehen, müssen wir für i den Bereich aller ganzen Zahlen vorgeben: $i = -\infty, \ldots, -m, \ldots, -1, 0, 1, \ldots, n, \ldots, +\infty$. Die endlichen Zahlen ergeben sich dann mit $z_i = 0$ für $i = n + 1, \quad n + 2, \ldots, +\infty$ und $i = -(m + 1), -(m + 2), \ldots, -\infty$

Betrachten wir die Basiszahlen näher, so werden wir feststellen, daß die des Hexadezimal- und des Oktalsystems Potenzen der Basiszahl des Dualsystems sind, denn es gilt ja

$$2^3 = 8 \text{ und } 2^4 = 16.$$

Dies wird sich insbesondere für das Umrechnen aus einem System in ein anderes als günstig erweisen.
Leider fällt das Dezimalsystem diesbezüglich aus dem Rahmen. Hieraus werden einige Schwierigkeiten der Umformung entstehen.

2. Entsprechend den angeführten Basiszahlen werden folgende Ziffern z_i benötigt:

$B = 2 \quad z_i = \{0, L\}$
$B = 8 \quad z_i = \{0, 1, 2, 3, 4, 5, 6, 7\}$
$B = 10 \quad z_i = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$B = 16 \quad z_i = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \dot{2}, \dot{3}, \dot{4}, \dot{5}, \dot{6}, \dot{7}\}$

Die Ziffern $\dot{2}, \dot{3}, \dot{4}, \dot{5}, \dot{6}, \dot{7}$ entsprechen der Verwendungsweise beim Cellatron SER. Sie entstehen dadurch, daß die über den Wert 10 des Dezimalsystems hinausgehenden Ziffern wie folgt zerlegt werden:

$10 = 8 + 2 = P + 2 = \dot{2}$
$11 = 8 + 3 = P + 3 = \dot{3}$
$12 = 8 + 4 = P + 4 = \dot{4}$
$13 = 8 + 5 = P + 5 = \dot{5}$
$14 = 8 + 6 = P + 6 = \dot{6}$
$15 = 8 + 7 = P + 7 = \dot{7}$

P nennt man auch Pseudodezimale.
Für die Arbeit mit dem ZRA 1 hat sich die Schreibweise

$10 = \dot{2} = A$
$11 = \dot{3} = B$
$12 = \dot{4} = C$
$13 = \dot{5} = D$

$14 = \dot{6} = E$
$15 = \dot{7} = F$

eingebürgert.
Im Dualsystem ist es üblich, die duale Eins durch das Zeichen L darzustellen.
Eine Dualziffer bezeichnet man auch als *Bit* (binary digit – engl. – Binärstelle[1]).

3. Den Ziffernwert erhalten wir, auf das Dezimalsystem bezogen, zu:

$B =$ 16	10	8	2	
0	0	0	0	= null Einheiten
1	1	1	L	= eine Einheit
2	2	2		= zwei Einheiten
3	3	3		= drei Einheiten
4	4	4		= vier Einheiten
5	5	5		= fünf Einheiten
6	6	6		= sechs Einheiten
7	7	7		= sieben Einheiten
8	8			= acht Einheiten
9	9			= neun Einheiten
$\dot{2}$				= zehn Einheiten
$\dot{3}$				= elf Einheiten
$\dot{4}$				= zwölf Einheiten
$\dot{5}$				= dreizehn Einheiten
$\dot{6}$				= vierzehn Einheiten
$\dot{7}$				= fünfzehn Einheiten

4. Der Stellenwert ergibt sich beispielsweise für das Dezimalsystem entsprechend den Potenzen der Basiszahl zu

6. Stelle vor dem Komma	$\triangleq 10^5$	= 100000
5. Stelle vor dem Komma	$\triangleq 10^4$	= 10000
4. Stelle vor dem Komma	$\triangleq 10^3$	= 1000
3. Stelle vor dem Komma	$\triangleq 10^2$	= 100
2. Stelle vor dem Komma	$\triangleq 10^1$	= 10

[1] binär: aus zwei Ziffern bestehend

1. Stelle vor dem Komma	$\triangleq 10^0$ =	1
1. Stelle nach dem Komma	$\triangleq 10^{-1}$ =	0,1
2. Stelle nach dem Komma	$\triangleq 10^{-2}$ =	0,01
3. Stelle nach dem Komma	$\triangleq 10^{-3}$ =	0,001
4. Stelle nach dem Komma	$\triangleq 10^{-4}$ =	0,0001

Auch für die anderen Zahlensysteme finden wir dieselbe Zahldarstellung, der jedoch ein anderer Stellenwert, bezogen auf das Dezimalsystem, entspricht:

$B = 16$

6. Stelle vor dem Komma	$\triangleq 16^5$ =	1048576	$\triangleq$ (100000)$_{16}$
5. Stelle vor dem Komma	$\triangleq 16^4$ =	65336	$\triangleq$ (10000)$_{16}$
4. Stelle vor dem Komma	$\triangleq 16^3$ =	4096	$\triangleq$ (1000)$_{16}$
3. Stelle vor dem Komma	$\triangleq 16^2$ =	256	$\triangleq$ (100)$_{16}$
2. Stelle vor dem Komma	$\triangleq 16^1$ =	16	$\triangleq$ (10)$_{16}$
1. Stelle vor dem Komma	$\triangleq 16^0$ =	1	$\triangleq$ (1)$_{16}$
1. Stelle nach dem Komma	$\triangleq 16^{-1}$ =	0,0625	$\triangleq$ (0,1)$_{16}$
2. Stelle nach dem Komma	$\triangleq 16^{-2}$ =	0,00390625	$\triangleq$ (0,01)$_{16}$

$B = 8$

6. Stelle vor dem Komma	$\triangleq 8^5$ =	32768	$\triangleq$ (100000)$_8$
5. Stelle vor dem Komma	$\triangleq 8^4$ =	4096	$\triangleq$ (10000)$_8$
4. Stelle vor dem Komma	$\triangleq 8^3$ =	512	$\triangleq$ (1000)$_8$
3. Stelle vor dem Komma	$\triangleq 8^2$ =	64	$\triangleq$ (100)$_8$
2. Stelle vor dem Komma	$\triangleq 8^1$ =	8	$\triangleq$ (10)$_8$
1. Stelle vor dem Komma	$\triangleq 8^0$ =	1	$\triangleq$ (1)$_8$
1. Stelle nach dem Komma	$\triangleq 8^{-1}$ =	0,125	$\triangleq$ (0,1)$_8$
2. Stelle nach dem Komma	$\triangleq 8^{-2}$ =	0,015625	$\triangleq$ (0,01)$_8$
3. Stelle nach dem Komma	$\triangleq 8^{-3}$ =	0,001953125	$\triangleq$ (0,001)$_8$

$B = 2$

6. Stelle vor dem Komma	$\triangleq 2^5$ =	32	$\triangleq$ LO OOOO
5. Stelle vor dem Komma	$\triangleq 2^4$ =	16	$\triangleq$ L OOOO
4. Stelle vor dem Komma	$\triangleq 2^3$ =	8	$\triangleq$ LOOO
3. Stelle vor dem Komma	$\triangleq 2^2$ =	4	$\triangleq$ LOO
2. Stelle vor dem Komma	$\triangleq 2^1$ =	2	$\triangleq$ LO
1. Stelle vor dem Komma	$\triangleq 2^0$ =	1	$\triangleq$ L
1. Stelle nach dem Komma	$\triangleq 2^{-1}$ =	0,5	$\triangleq$ O,L
2. Stelle nach dem Komma	$\triangleq 2^{-2}$ =	0,25	$\triangleq$ O,OL
3. Stelle nach dem Komma	$\triangleq 2^{-3}$ =	0,125	$\triangleq$ O,OOL
4. Stelle nach dem Komma	$\triangleq 2^{-4}$ =	0,0625	$\triangleq$ O,OOOL

Im Dualsystem werden jeweils vier Dualstellen zu einer Dualtetrade zusammengefaßt.

Da wir die Stellenwerte für das Umformen der Zahlen häufiger benötigen, werden sie auf der Beilage zusammengestellt.

5. Die Zahlen erhalten wir dann durch Multiplizieren der Ziffer mit dem Stellenwert und Summation über alle vorhandenen von Null verschiedenen Ziffern. Also für

$B = 16$ zu
$Z = z_n \cdot 16^n + z_{n-1} \cdot 16^{n-1} + \cdots + z_0 \cdot 16^0 + \cdots + z_{-m} \cdot 16^{-m}$
$B = 10$ zu
$Z = z_n \cdot 10^n + z_{n-1} \cdot 10^{n-1} + \cdots + z_0 \cdot 10^0 + \cdots + z_{-m} \cdot 10^{-m}$
$B = 8$
$Z = z_n \cdot 8^n + z_{n-1} \cdot 8^{n-1} + \cdots + z_0 \cdot 8^0 + \cdots + z_{-m} \cdot 8^{-m}$
$B = 2$
$Z = z_n \cdot 2^n + z_{n-1} \cdot 2^{n-1} + \cdots + z_0 \cdot 2^0 + \cdots + z_{-m} \cdot 2^{-m}$

wobei die z_i nur Werte der für das jeweilige Zahlensystem in 2 definierten Ziffern annehmen dürfen.

6. Für die Zahldarstellung wollen wir vereinbaren, daß wir bei den weniger gebräuchlichen Zahlensystemen die Zahlen in Klammern schreiben und die Basis als Index hinzufügen. So haben alle nachfolgend angeführten Zahlen den gleichen Wert.

$(1200, 11)_{16}$
4608, 06640625
$(11000, 042)_8$
L OOLO OOOO OOOO, OOOL OOOL

Wie aus diesem Vergleich bereits ersichtlich, hat insbesondere der ganzzahlige Anteil der Dualzahlen eine weitaus größere Stellenzahl als der entsprechende Anteil der Dezimalzahlen. Dies ist ein wesentlicher Nachteil der Dualzahlen, denn deren „Länge" vergrößert sich rund auf das $3^1/_3$fache der gleich großen Dezimalzahl. Auch sind Dualzahlen zumindest zu Anfang schwer zu lesen und eben ungewohnt.
In der maschinellen Rechentechnik ist das Dualsystem jedoch von besonderer Bedeutung. Das Rechnen im Dualsystem ist sehr einfach. Es läßt sich, wie wir später nachweisen, durch Schaltelemente mit zwei stabilen Zuständen nachbilden und ausführen. Darauf basiert die Technik moderner digitaler Rechenautomaten.

1.2.5. Zahlenumwandlung

Von den für die Rechentechnik interessierenden Zahlensystemen werden mindestens zwei, das Dezimal- und das Dualsystem, gleichzeitig verwandt.
Hieraus entsteht das Problem, Zahlen aus einem System in ein anderes umzurechnen.
Wollen wir Zahlen aus dem Hexadezimal- oder Oktalsystem in das Dualsystem umwandeln, so ist das sehr einfach. Wir machen uns dabei die Tatsache zunutze, daß zwischen den Basiszahlen die Relation

$$16 = 2^4 \text{ und } 8 = 2^3$$

besteht, auf die bereits hingewiesen wurde.
Entsprechend brauchen wir beim Umformen aus dem Hexadezimalsystem in das Dualsystem lediglich jede Ziffer des ersteren in eine Dualtetrade des letzteren zu verwandeln, wobei gilt:

0 ≙ OOOO	4 ≙ OLOO	8 ≙ LOOO	$\dot{4}$ ≙ LLOO
1 ≙ OOOL	5 ≙ OLOL	9 ≙ LOOL	$\dot{5}$ ≙ LLOL
2 ≙ OOLO	6 ≙ OLLO	$\dot{2}$ ≙ LOLO	$\dot{6}$ ≙ LLLO
3 ≙ OOLL	7 ≙ OLLL	$\dot{3}$ ≙ LOLL	$\dot{7}$ ≙ LLLL

Beispiel:

$$(53)_{16} \triangleq \text{OLOL OOLL} = \text{LOL OOLL}$$

Analog können wir eine Dualzahl in eine Hexadezimalzahl umformen, indem wir jede Dualtetrade einzeln nach obiger Darstellung umformen.

Beispiel:

$$\text{LOLL OOOO LOOL} \triangleq (\dot{3}09)_{16}$$

Für Umwandlungen aus dem Oktal- in das Dualsystem und umgekehrt gilt dasselbe Verfahren, nur daß wir jetzt jede Oktalziffer in eine Dualtriade umwandeln müssen, wobei nun gilt:

0 ≙ OOO	3 ≙ OLL	6 ≙ LLO
1 ≙ OOL	4 ≙ LOO	7 ≙ LLL
2 ≙ OLO	5 ≙ LOL	

Beispiel:

$(123)_8 \triangleq$ OOL OLO OLL = LOL OOLL
LOL OOLL = L OLO OLL $\triangleq (123)_8$

Dieselben Regeln können wir auch für den Bruchanteil verwenden, wobei jetzt allerdings vom Komma nach rechts Dualtetraden respektive Triaden gebildet werden müssen.

Beispiel:

$(0, 2\ 4\ \dot{3})_{16} \triangleq$ O,OOLO OLOO LOLL
$(0, 2\ 4\ 3)_8 \triangleq$ O,OLO LOO OLL = O,OLOL OOOL L
O,LLLO OOOL LOLO $\triangleq (0,\dot{6}1\dot{2})_{16}$
O,LLLO OOOL LOLO = O,LLL OOO OLL OLO $\triangleq$
$(0,7032)_8$

Größere Schwierigkeiten wird uns das Umwandeln aus dem Dezimalsystem in eines der betrachteten Zahlensysteme und dessen Umkehrung bereiten.
Betrachten wir erst das Umrechnen einer Dezimalzahl in eine Dualzahl. Dieser Prozeß wird Konvertieren genannt. Rekonvertieren heißt dann das Umwandeln einer Dualzahl in eine Dezimalzahl. Für diese Rechnungen werden meist die Rechenautomaten selber herangezogen. Dennoch kommt es häufig vor, daß auch der Programmierer oder der Bedienungstechniker eine Zahl konvertieren oder rekonvertieren muß. Dazu verwendet man Tafeln. Da oft ohne Tafel konvertiert werden muß, soll hier eine einfache Vorschrift angeführt werden.
Der ganzzahlige duale Anteil einer Dezimalzahl kann durch eine wiederholte Division des Zahlteiles durch zwei mit Rest ermittelt werden. Der Rest kann dabei ja nur 1 oder 0 sein und liefert jeweils eine Dualstelle, der Reihe nach mit der kleinsten beginnend:

Beispiel:

53 soll konvertiert werden

53:2 = 26 Rest 1
26:2 = 13 Rest 0
13:2 = 6 Rest 1
6:2 = 3 Rest 0

$3:2 = 1$ Rest 1
$1:2 = 0$ Rest 1

somit ist die konvertierte Zahl:

53 △ LLOLOL

Die Probe erhalten wir durch Rekonvertieren. Hierzu verwenden wir die dem Buch beigefügte Potenztabelle. Wir zählen die Exponenten ab und addieren für jede L die entsprechende Dualpotenz.

LLOLOL ergibt:

$$\begin{aligned} 1 \cdot 2^0 &\triangleq 1 \\ + 0 \cdot 2^1 &\triangleq 0 \\ + 1 \cdot 2^2 &\triangleq 4 \\ + 0 \cdot 2^3 &\triangleq 0 \\ + 1 \cdot 2^4 &\triangleq 16 \\ + 1 \cdot 2^5 &\triangleq 32 \\ &\quad 53 \end{aligned}$$

Somit

LLOLOL △ 53.

Sehr einfach ist die Konvertierung von Dezimalbrüchen. Nach dem unten angeführten Schema ist der Dezimalbruch lediglich jeweils mit zwei zu multiplizieren und die in den „Überlauf" einfließende Null oder Eins in Dualziffern zu schreiben. Wir wollen hier jedoch ohne Beweis anmerken, daß nicht jeder endliche Dezimalbruch in einen endlichen Dualbruch konvertiert werden kann.

Beispiel:

$Z = 0{,}171\,875$ ist zu konvertieren

0	171 875 · 2
0	343 750 · 2
0	687 500 · 2
1	375 000 · 2
0	750 000 · 2
1	500 000 · 2
1	000 000 · 2

Die gesuchte Dualzahl ist gleich der Ziffernfolge in der linken Spalte, wenn nach der ersten Null das Komma und alle Ziffern in das Dualsystem gesetzt werden.

$$Z = 0{,}00\text{LOLL}$$

Die linke Spalte ist bei der Multiplikation nicht zu berücksichtigen. Die Probe können wir wiederum durch Rekonvertierung vornehmen. Vergleichen wir die Stellenwerte der Dualzahlen, so ergibt sich:

$$\begin{aligned}
0 \cdot 2^{0} &\triangleq 0\\
+0 \cdot 2^{-1} &\triangleq 0\\
+0 \cdot 2^{-2} &\triangleq 0\\
+1 \cdot 2^{-3} &\triangleq 0{,}125\\
+0 \cdot 2^{-4} &\triangleq 0\\
+1 \cdot 2^{-5} &\triangleq 0{,}03125\\
+1 \cdot 2^{-6} &\triangleq 0{,}015625\\
&\quad\overline{0{,}171875}
\end{aligned}$$

Ist eine Zahl gegeben, die aus einem ganzzahligen Anteil und aus einem Dezimalbruch besteht, beispielsweise $Z = 321{,}34375$, so wird erst der ganzzahlige Anteil konvertiert und anschließend der Dezimalbruchteil.

Beispiel:

$$Z = 321{,}34375$$

$$\begin{aligned}
\text{I. } 321 : 2 &= 160 \text{ Rest } 1\\
160 : 2 &= 80 \text{ Rest } 0\\
80 : 2 &= 40 \text{ Rest } 0\\
40 : 2 &= 20 \text{ Rest } 0\\
20 : 2 &= 10 \text{ Rest } 0\\
10 : 2 &= 5 \text{ Rest } 0\\
5 : 2 &= 2 \text{ Rest } 1\\
2 : 2 &= 1 \text{ Rest } 0\\
1 : 2 &= 0 \text{ Rest } 1
\end{aligned}$$

Der ganzzahlige Anteil ist somit LOLOOOOOL.

II. 0	34375 · 2
0	68750 · 2
1	37500 · 2
0	75000 · 2
1	50000 · 2
1	00000 · 2

Der Bruchanteil ist O, OLOLL.
Die gesamte ganze Zahl ergibt sich als:

$$321{,}34375 = Z = \text{L OLOO OOOL, OLOL L}$$

Zur Probe die Rekonvertierung:

$$\begin{aligned}
1 \cdot 2^8 &\triangleq 256{,}0\\
+0 \cdot 2^7 &\triangleq 0{,}0\\
+1 \cdot 2^6 &\triangleq 64{,}0\\
+0 \cdot 2^5 &\triangleq 0{,}0\\
+0 \cdot 2^4 &\triangleq 0{,}0\\
+0 \cdot 2^3 &\triangleq 0{,}0\\
+0 \cdot 2^2 &\triangleq 0{,}0\\
+0 \cdot 2^1 &\triangleq 0{,}0\\
+1 \cdot 2^0 &\triangleq 1{,}0\\
+0 \cdot 2^{-1} &\triangleq 0{,}0\\
+1 \cdot 2^{-2} &\triangleq 0{,}25\\
+0 \cdot 2^{-3} &\triangleq 0{,}0\\
+1 \cdot 2^{-4} &\triangleq 0{,}0625\\
+1 \cdot 2^{-5} &\triangleq 0{,}03125\\
&\quad 321{,}34375
\end{aligned}$$

Die Umwandlung aus dem Dezimal- in das Hexadezimal- oder Oktalsystem kann nach derselben Vorschrift erfolgen. Das letzte Beispiel soll uns das veranschaulichen.

Beispiel: 321,34375

$$\begin{aligned}
321 : 16 &= 20 \text{ Rest } 1\\
20 : 16 &= 1 \text{ Rest } 4\\
1 : 16 &= 0 \text{ Rest } 1
\end{aligned}$$

ganzzahliger Anteil: $(141)_{16}$

0	34375 · 16
5	50000 · 16
8	00000

Bruchanteil: $(0{,}58)_{16}$
somit
$321{,}34375 \triangleq (141{,}58)_{16}$

Probe:

	1 · 256	$\triangleq$	256
+	4 · 16	$\triangleq$	64
+	1 · 1	$\triangleq$	1
+	5 · 0,0625	$\triangleq$	0,3125
+	8 · 0,00390625	$\triangleq$	0,03125
			321,34375

Entsprechend ergibt sich

321 : 8 = 40 Rest 1
40 : 8 = 5 Rest 0
5 : 8 = 0 Rest 5

ganzzahliger Anteil $(501)_8$

0	34375 · 8
2	75000 · 8
6	00

Bruchanteil $(0{,}26)_8$
somit
$321{,}34375 \triangleq (501{,}26)_8$

Probe:

	5 · 64	$\triangleq$	320
+	0 · 8	$\triangleq$	0
+	1 · 1	$\triangleq$	1
+	2 · 0,125	$\triangleq$	0,25
+	6 · 0,015625	$\triangleq$	0,09375
			321,34375

Meist benötigen wir als Programmierer oder Bediener eines Rechenautomaten nur den ganzzahligen Teil bis zu einer bestimmten Größe (z. B. alle ganzzahligen Werte, die kleiner als 4096 sind). Dann werden Tabellen angewendet.
Betrachten wir als Beispiel die Umwandlung von Oktalzahlen in Dezimalzahlen.

Für $0 \leqq Z \leqq 4096$

gilt der Bereich der Oktalzahlen

$$0 \leqq Z_8 \leqq (10000)_8$$

d. h., wir müssen höchstens vierstellige Oktalzahlen umwandeln. Für die ersten beiden Ziffern verwenden wir folgende Tabelle:

Tabelle 1. Umwandeln vierstelliger Oktalzahlen in Dezimalzahlen

	.0..	.1..	.2..	.3..	.4..	.5..	.6..	.7..
0...	0	64	128	192	256	320	384	448
1...	512	576	640	704	768	832	896	960
2...	1024	1088	1152	1216	1280	1344	1408	1472
3...	1536	1600	1664	1728	1792	1856	1920	1984
4...	2048	2112	2176	2240	2304	2368	2432	2496
5...	2560	2624	2688	2752	2816	2880	2944	3008
6...	3072	3136	3200	3264	3328	3392	3456	3520
7...	3584	3648	3712	3776	3840	3904	3968	4032

Wollen wir nun die Oktalzahl $(5413)_8$ in eine Dezimalzahl umwandeln, so suchen wir uns auf der Tabelle den Wert der Zeile 5... und Spalte .4.., also 2816. Dazu addieren wir $1 \cdot 8 + 3 \cdot 1 = 11$, was wir im Kopf rechnen können. Wir erhalten 2827.

Probe: $2827:8 = 353$ Rest 3
$353:8 = 44$ Rest 1
$44:8 = 5$ Rest 4
$5:8 = 0$ Rest 5

somit $2827 \triangleq (5413)_8$

Mit Hilfe der Tabelle können wir aber auch Dezimalzahlen in Oktalzahlen umwandeln. Betrachten wir das angeführte Beispiel. Gegeben ist 2827. Dann suchen wir uns die nächstkleinere Zahl in der Tabelle und bestimmen danach die ersten beiden Oktalziffern $(54\ ..)_8$. Der Rest 11 wird durch 8 mit Rest geteilt und ergibt so die letzten beiden Ziffern: 11 : 8 = 1 Rest 3. Somit ist die gesuchte Oktalzahl

$$2827 \triangleq (5413)_8$$

Eine ähnliche Tabelle gibt es für Umrechnungen aus und in das Hexadezimalsystem, worauf wir nicht eingehen wollen. Sie liegt in jedem Rechenzentrum vor, das einen ZRA 1 besitzt.[1]

1.3. Strukturen für Daten

Programmgesteuerte Rechenautomaten verarbeiten *Informationen.* Bei den Informationen müssen wir folgende Merkmale unterscheiden:

ihre Bedeutung oder Semantik,
ihre physikalische Erfassung über einen realen physikalischen Informationsträger und
ihre Darstellung, gegebenenfalls als Zeichenfolge eines festgelegten Alphabets.

Die *Semantik einer Information* entspricht der üblichen Bedeutung des Wortes Information in der Umgangssprache. Sie ermöglicht es, Aussagen über abgrenzbare Teilbereiche der Realität zu machen. Hierdurch können wir nach bestimmten Merkmalen klassifizieren, beispielsweise die Bewohner eines Landes in männlich und weiblich, mit einer möglichen weiteren Unterteilung in Jahre als zulässige Merkmale.
Die *physikalische Erfassung der Information* kann recht vielfältig sein. In der digitalen Rechentechnik hat es sich als zweckmäßig gezeigt, solche Träger zu verwenden, die zwei stabile physikalische Zustände aufweisen.

[1] ZRA 1: Zeiss Rechenautomat 1, s. S. 109

Beispielweise:

Impuls	Ausbleiben eines Impulses in einem Leiter
Lochung	Ausbleiben einer Lochung in einem Papierstreifen
lokal polarisiertes Magnetfeld	Ausbleiben des Magnetfeldes auf einer Trägerschicht
positiver Remanenzzustand	negativer Remanenzzustand eines Ferritkernes u. a.

Die *Informationsdarstellung* umfaßt alle Regeln und Vorschriften für den Aufbau des Einzelzeichens, der Zeichenfolgen (Wörter) und der ganzen Informationssätze, Satzfolgen (Kapitel) u. a. Man spricht hier auch vom syntaktischen Aspekt der Information.
Für die digitale Rechentechnik ist man bestrebt, entsprechend dem physikalischen Aspekt alle Zeichen durch einen Dualcode darzustellen.
In Hinsicht auf die verwendeten Zeichen unterscheidet man zwischen

numerischer Arbeitsweise,

bei der man die Ziffern 0, 1, . . . , 9 und eventuelle Sonderzeichen ◇, ◇, ∗, ∗, verwendet und

alpha-numerischer Arbeitsweise.

Hier verwendet man
die Ziffern 0, 1, 2, . . . , 9,
die Buchstaben A, B, C, . . . , Z, gegebenenfalls a, b, c, . . . , z,
Sonderzeichen , , ; , +, —, =, △, <, >, ∗, £, (,), [,] u. a.

Die entsprechend numerisch aufbereiteten Informationen werden Daten genannt.
Werden einzelne Informationssätze verarbeitet, um aus den darin enthaltenen Informationen mit Hilfe mathematischer Regeln neue Informationen für denselben Bereich zu erlangen, so umreißt dies die Rechentechnik im engeren Sinne.

Werden aber Folgen von Informationssätzen verarbeitet, um auch Merkmale mehrerer Bereiche zu vergleichen und zu allgemeineren Schlüssen heranzuziehen, so wird dies mit Datenverarbeitung bezeichnet.

1.3.1. Dualcodierte Dezimalzahlen

Da das Dezimalsystem für die von Rechenautomaten aus betrachtete externe Arbeit gebräuchlich ist, wurde versucht, für die Dezimalziffern Dualcodes einzuführen.
Zur Darstellung der größten Dezimalziffer 9 benötigen wir mindestens vier Bit, denn

$$9 \triangleq \text{LOOL}$$

Daher werden wir erst einmal versuchen, Codierungsschlüssel zu finden, bei denen jede Dezimalziffer in eine Dualtetrade übersetzt wird.
Für die Codierung haben sich in der Rechentechnik drei Schlüssel bewährt.

Direkte dezimal-duale Verschlüsselung

Bei der direkten dezimal-dualen Verschlüsselung, auch Direkt-Code genannt, wird jede Dezimalziffer in die ihrem Ziffernwert entsprechende Dualtetrade verwandelt

$0 \triangleq$ OOOO	$5 \triangleq$ OLOL
$1 \triangleq$ OOOL	$6 \triangleq$ OLLO
$2 \triangleq$ OOLO	$7 \triangleq$ OLLL
$3 \triangleq$ OOLL	$8 \triangleq$ LOOO
$4 \triangleq$ OLOO	$9 \triangleq$ LOOL

Beispiel:

$$320{,}47 \triangleq \text{OOLL OOLO OOOO, OLOO OLLL}$$

Dieser Code hat den Vorteil, daß man die Verschlüsselung leicht lesen kann.

Aiken-Verschlüsselung

Wie wir später noch sehen werden, spielt das sogenannte Neunerkomplement in der maschinellen Rechentechnik eine große Rolle. Unter Neunerkomplement einer Dezimalziffer z wollen

wir dabei diejenige Ziffer z' verstehen, die zur ursprünglichen addiert 9 ergibt.
Also

$$z + z' = 9.$$

Damit das Neunerkomplement leicht gebildet werden kann, hat der Amerikaner *Aiken* folgenden Code eingeführt:

0 ≙ OOOO	9 ≙ LLLL
1 ≙ OOOL	8 ≙ LLLO
2 ≙ OOLO	7 ≙ LLOL
3 ≙ OOLL	6 ≙ LLOO
4 ≙ OLOO	5 ≙ LOLL

Beispiel:

320,47 ≙ OOLL OOLO OOOO,OLOO LLOL

Die Neunerkomplemente sind nebeneinander geschrieben. Die entsprechenden Dualtetraden unterscheiden sich dabei nur darin, daß beim Komplement jeweils für eine O ein L und umgekehrt für ein L eine O gesetzt ist. Dies nennt man Negation. Das Neunerkomplement wird beim Aiken-Code somit durch Negation der Dualtetrade gebildet.

Dreierexzeß-Verschlüsselung

Ein weiterer Code, aus dem das Neunerkomplement durch Negation der Dualtetraden ermittelt werden kann, ist der Dreierexzeß-Code. Er entsteht dadurch, daß zur Dezimalziffer z drei addiert und dann die zugehörige Dualtetrade bestimmt wird.

0 (+3 = 3) ≙ OOLL	9 (+3 = 12) ≙ LLOO
1 (+3 = 4) ≙ OLOO	8 (+3 = 11) ≙ LOLL
2 (+3 = 5) ≙ OLOL	7 (+3 = 10) ≙ LOLO
3 (+3 = 6) ≙ OLLO	6 (+3 = 9) ≙ LOOL
4 (+3 = 7) ≙ OLLL	5 (+3 = 8) ≙ LOOO

Beispiel:

320,47 ≙ OLLO OLOL OOLL, OLLL LOLO

Der Vorteil der Dreierexzeß-Verschlüsselung besteht vor allem darin, daß in jeder Dualtetrade beide Dualziffern O und L vorkommen. Hierdurch können automatische Kontrollen leichter ausgeführt werden.
Es lassen sich noch viele Verschlüsselungssysteme dieser Art aufstellen, da wir aus den Ziffern O und L insgesamt 16 Tetraden bilden können. Praktisch haben aber nur die obenerwähnten Bedeutung, die wir nochmals zusammenfassen wollen:

	Direkt-Code	Aiken-Code	Dreierexzeß-Code
OOOO	0	0	—
OOOL	1	1	—
OOLO	2	2	—
OOLL	3	3	0
OLOO	4	4	1
OLOL	5	—	2
OLLO	6	—	3
OLLL	7	—	4
LOOO	8	—	5
LOOL	9	—	6
LOLO	—	—	7
LOLL	—	5	8
LLOO	—	6	9
LLOL	—	7	—
LLLO	—	8	—
LLLL	—	9	—

Die Dualtetraden, denen in einem Code keine Dezimalziffer entspricht (durch einen Strich markiert), werden *Pseudotetraden* genannt.
Pseudotetraden können wir dazu verwenden, um beispielsweise bei rein numerischer Verschlüsselung einen Code für die *Sonderzeichen* festzulegen.
Für den Zeiss Rechenautomaten 1 (ZRA 1), der Direkt-Code verwendet, gilt folgende Festlegung:

Code	Dualtetrade	Druckzeichen
A	LOLO	✳
B	LOLL	✳
C	LLOO	(Leerzeichen)
D	LLOL	— (Minus)
E	LLLO	◇
F	LLLL	◇

Besondere Schwierigkeiten ergeben sich durch die sehr strengen Forderungen an die Betriebssicherheit. Selbst wenn nur ein Bit unbemerkt verfälscht wird, ist das Gesamtergebnis nicht richtig. Die Fehlerquellen können dabei subjektiv oder objektiv sein. Bereits bei der Programmierung sind Fehler möglich. Eine weitere Quelle liegt in der Bedienung. Beide können wir jedoch weitgehend durch Regeln und Vorschriften, durch Üben und Sammeln von Erfahrungen überwinden.
Fehler entstehen aber auch objektiv durch die Schaltelemente. Der natürliche Verschleiß oder Verwenden schadhafter Elemente kann Störungen der Art hervorrufen, daß Impulsfolgen oder Einzelimpulse zu klein werden. Das nachfolgende Schaltelement wird durch sie dann nicht mehr angeregt, und sie fallen aus. Handelt es sich um Störungen bei Impulsfolgen, werden sie meist schnell erkannt. Die Rechenergebnisse werden hier so stark entstellt, daß Rechenkontrollen den Fehler erfassen.
Schwieriger ist das Fehlererkennen bei Ausfall von Einzelimpulsen. In den meisten Automaten wird in einem Arbeitstakt des Automaten lediglich bei einem dualen L ein Impuls weitergeleitet, während das Ausbleiben des Impulses der dualen O entspricht. Daher wird hier auch gern in übertragenem Sinne vom Ausfall eines Bit gesprochen.
Der Ausfall eines Bit bedeutet also, daß im Zeichen ein duales L gelöscht ist. Haben wir beispielsweise das Zeichen

$$9 \triangleq \text{LOOL},$$

so könnte daraus durch Ausfall des höchsten Bit entstehen:

$$\text{OOOL} \triangleq 1$$

Damit wird aber das Ergebnis in einer Form verfälscht, daß der Fehler meist nicht gleich sichtbar ist.
Hier helfen interne Kontrollen. Sie werden an verschiedenen Stellen im Automaten ausgeführt, beispielsweise vor und nach einer Rechnung, vor und nach einem Transport, beim Lesen und Schreiben. Geprüft wird jeweils ein Zeichen (Ziffer oder Buchstabe) oder eine vorgegebene Anzahl von Zeichen, ein Wort.
Die Prüfung selber erfolgt über ein *Prüfbit.*
Je nach Automatentyp ist für ein Zeichen oder ein Wort eine gerade oder ungerade Anzahl von Dualzeichen L vorgeschrieben. Wir wollen im folgenden annehmen, daß eine ungerade Anzahl von Werten L festgelegt sei. Dann wird das Prüfbit benutzt, um bei der Eingabe in den Automaten immer dieser Forderung zu genügen. Hat ein Zeichen oder ein Wort
eine gerade Anzahl Dualzeichen L, so ist das Prüfbit L,
eine ungerade Anzahl Dualzeichen L, so ist das Prüfbit O.
Die interne Arbeitsweise des Automaten sichert, daß die ungerade Anzahl der Dualzeichen L je Zeichen oder Wort bei der normalen Arbeit immer erhalten bleibt. Fällt durch Störung ein Einzelbit aus, wird die Anzahl der Werte L in Zeichen oder Wort gerade, und bei der nächsten Kontrolle geht der Automat in Stop und zeigt den Fehler an. Dadurch wird der Ausfall eines Bit nahezu völlig ausgeschaltet, denn daß in einem Zeichen oder Wort gleichzeitig zwei Bit ausfallen, ist unwahrscheinlich oder führt zum Ausfall einer Folge von Bit, was durch Rechenkontrollen erkannt wird.
Diese Art der Kontrolle führte dazu, bereits bei der Eingabe hierfür besonders geeignete spezielle Codes zu verwenden. Als Beispiele seien angeführt:

Zwei-aus-fünf-Code

LLOOO ≙ 0	OLOLO ≙ 5
OOOLL ≙ 1	OLLOO ≙ 6
OOLOL ≙ 2	LOOOL ≙ 7
OOLLO ≙ 3	LOOLO ≙ 8
OLOOL ≙ 4	LOLOO ≙ 9

OOOOOOOOOL ≙ 0	OOOOLOOOOO ≙ 5
OOOOOOOOLO ≙ 1	OOOLOOOOOO ≙ 6
OOOOOOOLOO ≙ 2	OOLOOOOOOO ≙ 7
OOOOOOLOOO ≙ 3	OLOOOOOOOO ≙ 8
OOOOOLOOOO ≙ 4	LOOOOOOOOO ≙ 9

Meist wird jedoch ein Code verwendet, bei dem zu den uns bereits bekannten Festlegungen das Prüfbit hinzugefügt wird. Für den Direkt-Code kommt es daher zu folgender Ergänzung:

Pr	2^3	2^2	2^1	2^0	≙ Dualpotenz der Stellen
L	O	O	O	O	≙ 0
O	O	O	O	L	≙ 1
O	O	O	L	O	≙ 2
L	O	O	L	L	≙ 3
O	O	L	O	O	≙ 4
L	O	L	O	L	≙ 5
L	O	L	L	O	≙ 6
O	O	L	L	L	≙ 7
O	L	O	O	O	≙ 8
L	L	O	O	L	≙ 9

Hier hat jedes Zeichen zumindest ein L und stets eine ungerade Anzahl Dualzeichen L.
Nun können wir bereits bei der Eingabe *Prüfbitkontrollen* durchführen und so eine richtige Übernahme der externen Zeichen mit großer Sicherheit erreichen.
Für einen derartigen Code benötigen wir somit mindestens fünf Dualstellen. Bei der Übernahme und im Automaten werden diese Dualwerte durch fünf Kanäle geleitet.
Daher spricht man auch von einem *Fünf-Kanal-Code*. Er findet beim digitalen Kleinrechner Cellatron SER Verwendung (s. S. 205).

1.3.2. Alpha-numerische Codes

Die moderne Rechentechnik und insbesondere die Datenverarbeitung verwenden jedoch alpha-numerische Zeichen, also Ziffern, die Buchstaben des Alphabets und Sonderzeichen. Sie

alle müssen für die Arbeit im Automaten in Folgen von Dualzeichen verwandelt werden. Das geschieht über einen alphanumerischen Code.

Wollen wir nur die zehn Ziffern und eines der normalen Alphabete verwenden (also ohne Umlaute), so müssen wir mindestens $10 + 26 = 36$ Zeichen codieren können.

Untersuchen wir dazu die Möglichkeiten der Zeichendarstellung durch Dualziffern.

Bei Verwendung einer Dualstelle lassen sich 2 Zeichen, bei zwei Dualstellen 4 Zeichen usw. bilden. Allgemein ergibt sich:

1 Dualstelle $\triangleq$ $2 = 2^1$ Zeichen
2 Dualstellen $\triangleq$ $4 = 2^2$ Zeichen
3 Dualstellen $\triangleq$ $8 = 2^3$ Zeichen
4 Dualstellen $\triangleq$ $16 = 2^4$ Zeichen
5 Dualstellen $\triangleq$ $32 = 2^5$ Zeichen
6 Dualstellen $\triangleq$ $64 = 2^6$ Zeichen
7 Dualstellen $\triangleq$ $128 = 2^7$ Zeichen
8 Dualstellen $\triangleq$ $256 = 2^8$ Zeichen

Hieraus ersehen wir, daß mindestens fünfstellige Dualcodes verwendet werden müssen. Das entspricht der Praxis. Es haben sich

Fünf-Kanal-Codes
Sechs-Kanal-Codes
Sieben-Kanal-Codes
und Acht-Kanal-Codes

durchgesetzt. Am häufigsten verwendet man den Fünf- und den Acht-Kanal-Code. Beide finden wir insbesondere im Fernschreibverkehr. So hat der *Fünf-Kanal-Code des Telegrafenalphabets* folgende Form:

OO OOO	Leerzeichen		LO OOO	E	3
OO OOL	T	5	LO OOL	Z	+
OO OLO	Wagenrücklauf		LO OLO	D	Werda
OO OLL	0	9	LO OLL	B	?
OO LOO	Zwischenraum		LO LOO	S	'
OO LOL	H		LO LOL	Y	6
OO LLO	N	,	LO LLO	F	
OO LLL	M	.	LO LLL	X	/

OL OOO	Zeilenvorschub		LL OOO	A	—
OL OOL	L	)	LL OOL	W	2
OL OLO	R	4	LL OLO	J	Kl
OL OLL	G		LL OLL	Ziffern und Zeichen	
OL LOO	I	8	LL LOO	U	7
OL LOL	P	0	LL LOL	Q	1
OL LLO	C	:	LL LLO	K	(
OL LLL	V	=	LL LLL	Buchstaben	

Wir sehen, daß einmal kein Prüfbit vorgesehen ist. Zudem wird durch die Ziffer LLLLL auf Buchstaben und LLOLL auf Ziffern und Zeichen umgestellt, da ja sonst mit den 32 möglichen Zeichen die mindestens 36 Ziffern und Buchstaben nicht dargestellt werden können.

Die Übertragungssicherheit ist bei diesem Code relativ gering. Das wirkt sich beim Übertragen von Texten der menschlichen Sprache nicht sehr stark aus, da wir meist einen Text noch lesen können, wenn ein oder einzelne Buchstaben verfälscht sind.

Anders ist es bei der Datenfernübertragung. Hier kommt es auf jedes Zeichen an. Daher bereitet die Übertragung in diesem Code Schwierigkeiten. Meist wird die Übertragungsgeschwindigkeit reduziert. Sie beträgt maximal 50 Bit/Sekunde oder 50 Baud[1]. Man versucht dabei die Sicherheit durch Mehrfachübertragung zu erhöhen.

Günstiger ist es jedoch, einen anderen Code zu verwenden. Am gebräuchlichsten ist dabei ein Acht-Kanal-Code. Er spielt in der modernen Datenverarbeitung eine so große Rolle, daß eine neue Organisation der Arbeit nach acht Dualstellen umfassenden Zeichen eingeführt wurde.

Das Zeichen zu 8 Bit wird als Byte[2] bezeichnet und kann einzeln

---

[1] 1 Baud (sprich bod) = 1 Bit/Sekunde, Einheit der Übertragungsgeschwindigkeit von Informationen

[2] 1 Byte (sprich bait): kleinstes direkt adressierbares Zeichen. Ein Byte umfaßt meist 8 Bit und kann durch ein alpha-numerisches Zeichen oder bei einigen Automatentypen durch zwei Ziffern belegt werden

angerufen werden. Die 8 Bit eines Byte haben dabei meist folgende Bedeutung (z. B. Robotron 300):

Bit	Wert	Bezeichnung
8	W	Wortmarkenbit
7	V	Überbit V
6	U	Überbit U
5	Pr	Prüfbit
4	2^4	numerisches Bit der Bewertung 8
3	2^3	numerisches Bit der Bewertung 4
2	2^1	numerisches Bit der Bewertung 2
1	2^0	numerisches Bit der Bewertung 1

Mit dem Acht-Bit-Code können wir 256 Zeichen darstellen. Er reicht also aus, um Ziffern, Buchstaben des großen und kleinen Alphabets und viele Sonderzeichen darzustellen.
Die numerischen Bit ermöglichen die Zifferndarstellung entsprechend ihrer Bewertung. Numerische Bit und Überbit benötigen wir zur Darstellung der Buchstaben und Sonderzeichen. Das Prüfbit hat die bereits beschriebene Bedeutung. Mit seiner Hilfe kann immer eine ungerade Anzahl von Dualzeichen L im Byte festgelegt werden.
Interessant ist das *Wortmarkenbit.*
Jedes Zeichen des Codes stellt doch eine Ziffer, einen Buchstaben oder ein Sonderzeichen dar. Aus diesen Zeichen können wir nun *Wörter* zusammenstellen. Zur Markierung des Wortendes verwenden wir das Wortmarkenbit. Aus den Wörtern lassen sich wiederum *Sätze, Gruppen von Sätzen* und *Blöcke* oder *Kapitel* aufbauen. Zur Markierung des jeweiligen Endes benötigen wir spezielle Zeichen, und zwar *Satzmarken, Gruppenmarken und Blockmarken.*
Als Beispiel wollen wir den Acht-Bit-Code anführen, wie er bei der mittleren Datenverarbeitungsanlage Robotron 300 verwandt wird.

8	7	6	5	4	3	2	1	Bit	8	7	6	5	4	3	2	1	Bit
*	O	O	*	O	O	O	O	0	*	O	L	*	O	O	O	O	+
*	O	O	*	O	O	O	L	1	*	O	L	*	O	O	O	L	A
*	O	O	*	O	O	L	O	2	*	O	L	*	O	O	L	O	B
*	O	O	*	O	O	L	L	3	*	O	L	*	O	O	L	L	C

✳	O	O	✳	O	L	O	O	4	✳	O	L	✳	O	L	O	O	D
✳	O	O	✳	O	L	O	L	5	✳	O	L	✳	O	L	O	L	E
✳	O	O	✳	O	L	L	O	6	✳	O	L	✳	O	L	L	O	F
✳	O	O	✳	O	L	L	L	7	✳	O	L	✳	O	L	L	L	G
✳	O	O	✳	L	O	O	O	8	✳	O	L	✳	L	O	O	O	H
✳	O	O	✳	L	O	O	L	9	✳	O	L	✳	L	O	O	L	I
✳	O	O	✳	L	O	L	O	□	✳	O	L	✳	L	O	L	O	~[1]
✳	O	O	✳	L	O	L	L	⫤	✳	O	L	✳	L	O	L	L	.
✳	O	O	✳	L	L	O	O	(	✳	O	L	✳	L	L	O	O	;
✳	O	O	✳	L	L	O	L	:	✳	O	L	✳	L	L	O	L	!
✳	O	O	✳	L	L	L	O	▾	✳	O	L	✳	L	L	L	O	▾▾
✳	O	O	✳	L	L	L	L	]	✳	O	L	✳	L	L	L	L	"

8	7	6	5	4	3	2	1	Bit	8	7	6	5	4	3	2	1	Bit
✳	L	O	✳	O	O	O	O	—	✳	L	L	✳	O	O	O	O	'
✳	L	O	✳	O	O	O	L	J	✳	L	L	✳	O	O	O	L	/
✳	L	O	✳	O	O	L	O	K	✳	L	L	✳	O	O	L	O	S
✳	L	O	✳	O	O	L	L	L	✳	L	L	✳	O	O	L	L	T
✳	L	O	✳	O	L	O	O	M	✳	L	L	✳	O	L	O	O	U
✳	L	O	✳	O	L	O	L	N	✳	L	L	✳	O	L	O	L	V
✳	L	O	✳	O	L	L	O	O	✳	L	L	✳	O	L	L	O	W
✳	L	O	✳	O	L	L	L	P	✳	L	L	✳	O	L	L	L	X
✳	L	O	✳	L	O	O	O	Q	✳	L	L	✳	L	O	O	O	Y
✳	L	O	✳	L	O	O	L	R	✳	L	L	✳	L	O	O	L	Z
✳	L	O	✳	L	O	L	O	≈[2]	✳	L	L	✳	L	O	L	O	≋[3]
✳	L	O	✳	L	O	L	L	)	✳	L	L	✳	L	O	L	L	,
✳	L	O	✳	L	L	O	O	✳	✳	L	L	✳	L	L	O	O	%
✳	L	O	✳	L	L	O	L	=	✳	L	L	✳	L	L	O	L	△
✳	L	O	✳	L	L	L	O	<	✳	L	L	✳	L	L	L	O	>
✳	L	O	✳	L	L	L	L	?	✳	L	L	✳	L	L	L	L	[

[1] Satzmarke
[2] Gruppenmarke
[3] Blockmarke

Das Wortmarkenbit konnte nicht gesetzt werden. Entsprechend ist es dann auch nicht möglich, das Prüfbit zu setzen, da dies unter Einbeziehung des Wortmarkenbit festgelegt wird. Diese Festlegung erfolgt dann bei der Eingabe durch den Automaten. Als Beispiel soll der Satz „ich erhalte 3,75 M" codiert werden, und zwar in die Form, wie er dann im Automaten steht.

LOLLLLLL	"
OOLOLOOL	I
OOLOOOLL	C
LOLOLOOO	H
OOLOOLOL	E
OLOOLOOL	R
OOLLLOOO	H
OOLLOOOL	A
OLOOOOLL	L
OLLLOOLL	T
LOLLOLOL	E
OOOLOOLL	3
OLLOLOLL	,
OOOOOLLL	7
LOOOOLOL	5
OLOLOLOO	M
LOLOLOLL	.
LOLLLLLL	"
LOLLLOLO	Satzmarke

1.4. Zahlendarstellung in Rechenautomaten

Die eigentliche Arbeitseinheit im Automaten ist das Wort. Ein Wort ist eine Folge von Zeichen, die ihrerseits aus Einzelinformationen (jeweils einer Dualstelle, einem Bit entsprechend) bestehen. Wir können dabei zwei Wortarten unterscheiden:

Zahlwörter zur Darstellung der Zahlen und
Befehlswörter zur Darstellung der Befehle.

Zur Wortdarstellung werden je nach Automatentyp eine feste Anzahl von Zeichen – *konstante Wortlänge* – oder eine in bestimmten Grenzen veränderliche Anzahl von Zeichen – *variable Wortlänge* – hinzugezogen. Immer ist die Zahl der Zeichen jedoch nach oben begrenzt.
Bei fester Wortlänge sind

10 bis 12 Dezimalstellen üblich, denen
40 bis 48 Dualstellen oder Bit entsprechen.

Bei variabler Wortlänge können als obere Grenze

10 bis 15 Byte festgelegt werden, denen
80 bis 120 Bit entsprechen.

Diese Begrenzung stellt eine technische Einschränkung dar, die im Widerspruch zur Forderung steht, einen möglichst großen Zahlenbereich bearbeiten zu können.
Zum kompletten Zahlwort gehört zudem:

1. der absolute Betrag der Zahl (die Ziffernfolge ohne Berücksichtigung des Vorzeichens),
2. das Vorzeichen,
3. eventuell Informationen über die Kommastellung,
4. Sonderinformationen, wie die Möglichkeit, eine Zahl besonders zu kennzeichnen (z. B. Q_1; Q_2 und QQ-Zeichen beim ZRA 1) u. a.

Um dennoch einen großen Zahlenbereich mit größtmöglicher Geschwindigkeit verarbeiten zu können, werden zwei Zahlendarstellungen im Rechenautomaten verwendet, die Darstellung als

Festkommazahlen und
Gleitkommazahlen.

Für die nachfolgenden Betrachtungen wollen wir zum besseren Verständnis konstante Wortlänge zu 12 Dezimalstellen annehmen. Bei variabler Wortlänge sind die Betrachtungen analog, nur müßten wir dann als oberste Grenze gegebenenfalls 12 Byte festlegen.

1.4.1. Festkommadarstellung

Bei der Festkommadarstellung wird, wie es der Name bereits sagt, eine feste Kommastellung vorgegeben.
So gibt es Automaten, die nur mit ganzen Zahlen operieren. Das Komma steht dann hinter der letzten Stelle. Hier ist der Zahlenbereich:

$$0 \leqq |Z| \leqq 999\,999\,999\,999,$$

wenn 12 Dezimalstellen angenommen werden.

Andere Automaten rechnen nur mit Zahlen, die kleiner sind als 1. Bei diesen Automaten setzt man das Komma vor die erste Stelle. Der Zahlenbereich ergibt sich dann zu

$$0 \leqq |Z| \leqq 0{,}999\,999\,999\,999,$$

und die kleinste Schrittweite zwischen zwei aufeinanderfolgenden Zahlen ist

$$0{,}000\,000\,000\,001$$

Ergibt sich bei der Festkommadarstellung irgendein Wert, der die oberste Grenze überschreitet, so unterbricht der Automat den Rechenprozeß und zeigt einen „Überlauf" an. Das ist eine wichtige Kontrolle, die vom Automaten selbständig ausgeführt wird. Wäre diese Kontrolle im logischen Aufbau des Automaten nicht vorgesehen, gingen die „übergelaufenen" Stellen verloren, und das Ergebnis würde verfälscht werden.
Die Festkommadarstellung stellt große Anforderungen an den Programmierer. Er muß die gesamte Rechnung wertmäßig verfolgen und erreichen, daß an keiner Stelle des Rechenablaufes der zulässige Zahlenbereich überschritten wird. Das kann er erzwingen, indem er für die Ausgangswerte der Rechnung entsprechende Einschränkungen vorschreibt. Dabei müssen diese Ausgangswerte gegebenenfalls mit einem sogenannten Skalenfaktor multipliziert werden, damit sie in den zulässigen Bereich fallen.
Diese Berechnungen bereiten einige Schwierigkeiten. Daher wird bei modernen Rechenautomaten neben der Festkommadarstellung auch die Gleitkommadarstellung verwandt.

1.4.2. Gleitkommadarstellung

Wollen wir sehr große oder sehr kleine Zahlen darstellen, so benutzen wir eine abkürzende Schreibweise.
Beispielsweise gibt uns die Avogadrosche Zahl Auskunft über die Anzahl der Moleküle im Mol. Ihr Wert ist

$$2{,}6873 \cdot 10^{19}.$$

Als anderes Beispiel wird uns das Plancksche Wirkungsquantum durch den Wert:

$$h = 6{,}625 \cdot 10^{-27} \text{ erg} \cdot \text{s}$$

angegeben.
Wir haben also die sehr große, respektive sehr kleine Zahl dadurch ersetzt, daß wir eine Zehnerpotenz als Faktor herangezogen haben:

$$26\,873\,000\,000\,000\,000\,000 = 2{,}6873 \cdot 10^{19}$$
$$0{,}000\,000\,000\,000\,000\,000\,000\,006\,625 = 6{,}625 \cdot 10^{-27}$$

Diese Möglichkeit der Zahlendarstellung machen wir uns für die Rechentechnik zunutze, indem wir etwas abweichend von der obigen Schreibweise festlegen:
Jede Zahl wird so verschoben, daß sie in den Bereich $0{,}1 \leqq |Z| < 1$ fällt. Diesen Vorgang nennt man *Normieren*.
Entsprechend der Normierung ergibt sich dann der Exponent der Zehnerpotenz.

Beispiel:

$324{,}76$ wird zu $0{,}324\,76 \cdot 10^{3}$
$0{,}0247$ wird zu $0{,}247 \cdot 10^{-1}$

Wie wir sehen, ist die Normierung recht einfach auszuführen. Das Komma wird nach links (positiver Exponent) oder rechts (negativer Exponent) so verschoben, bis die erste Stelle rechts vom Komma von Null verschieden ist, während alle Stellen links vom Komma Null sind.
Aus der Anzahl der Schritte, um die verschoben wird, ergibt sich dann der absolute Betrag des Exponenten.
Eine solche Darstellung der Zahlen durch

$$Z = m \cdot B^{e}$$

nennt man halblogarithmisch. Dabei ist

m die Mantisse,
B die Basis des Zahlensystems,
e der Exponent,
Z die Zahl.

Für die Darstellung im Rechenautomaten können wir die Zahl noch vereinfachen, indem wir die Null vor dem Komma, das Komma selber und die Basis des Zahlensystems nicht schreiben.

Beispiel:

$$
\begin{aligned}
+324{,}76 &= +\ 0{,}32476 \cdot 10^{+3} \triangleq +32476 + 3\\
+0{,}0247 &= +\ 0{,}247\ \ \cdot 10^{-1} \triangleq +24700 - 1\\
-53428 &= -\ 0{,}53428 \cdot 10^{+5} \triangleq -53428 + 5\\
-0{,}0003 &= -\ 0{,}3\ \ \cdot 10^{-3} \triangleq -30000 - 3
\end{aligned}
$$

Hierbei ist es aber ungünstig, daß in jeder Zahl zwei Vorzeichen auftreten, einmal für die Gesamtzahl, zum anderen für den Exponenten. Das können wir jedoch vermeiden.
Der Zahlenbereich im Automaten ist beschränkt. Das gilt dann auch für den Exponenten. Beispielsweise ist der zulässige Exponentenbereich beim ZRA 1

$$-19 \leqq e \leqq +19$$

Dann brauchen wir lediglich eine Transformation des Exponenten durch Addition mit einer Konstanten, die um eine Einheit größer ist als der Betrag der unteren Grenze, vorzunehmen, um eine Vorzeichenrechnung für den Exponenten zu umgehen. Transformieren wir den Exponenten e mit 20:

$$E = e + 20$$

so wird der transponierte Exponent nicht mehr negativ. Er hat dann den zulässigen Bereich

$$1 \leqq E \leqq 39,$$

und die oben angeführten Zahlen erhalten die Form:

$$
\begin{aligned}
&+32476 + 3 \triangleq +32476\ 23\\
&+24700 - 1 \triangleq +24700\ 19\\
&-53428 + 5 \triangleq -53428\ 25\\
&-30000 - 3 \triangleq -30000\ 17
\end{aligned}
$$

Durch die Gleitkommadarstellung wird der zulässige Zahlenbereich eines Automaten wesentlich vergrößert. Nehmen wir das Zahlwort wiederum mit konstanter Länge zu 12 Dezimal-

stellen, so kann man folgende Aufteilung vornehmen, wie sie beim ZRA 1 verwendet wird:

1 Dezimalstelle (4 Bit) für Vorzeichen und Sonderinformationen
2 Dezimalstellen für den Exponenten
9 Dezimalstellen für die Mantisse.

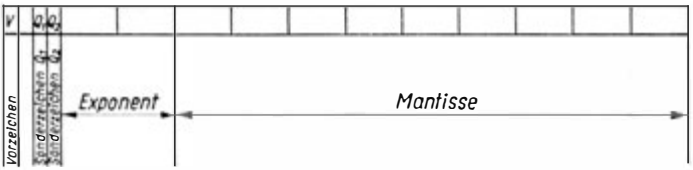


Bild 7. Schematische Darstellung für das Zahlwort des ZRA 1 (Gleitkommadarstellung)

So ergibt sich für den ZRA 1 beispielsweise ein zulässiger Zahlenbereich von

$$10^{-1} \cdot 10^{-19} \leqq |Z| \leqq (1 - 10^{-9})\, 10^{19}$$

oder ausgeschrieben

$$0{,}000\,000\,000\,000\,000\,001 \leqq |Z| \leqq 9\,999\,999\,990\,000\,000\,000.$$

Hierbei muß für die Null eine Sonderfestlegung getroffen werden. Sie wird durch

$$+m10^{E} \text{ mit } m = 0,\ E = 1$$

dargestellt.
Dieser Bereich übersteigt den der Festkommadarstellung wesentlich, für den beim ZRA 1 gilt:

$$-(1 - 10^{-11}) \leqq \pm Z \leqq 1 - 10^{-11}$$

oder

$$-0{,}999\,999\,999\,99 \leqq \pm Z \leqq +0{,}999\,999\,999\,99$$

mit einem Mindestabstand von

$$10^{-11} = 0{,}000\,000\,000\,01$$

Es ist offensichtlich, daß die Programmierer bei der Gleitkommadarstellung wesentlich weniger auf den zulässigen Zahlenbereich zu achten brauchen, da er seltener überschritten wird. Aber auch in diesem Fall stoppt der Automat die Arbeit bei Überlauf.
Für diesen Vorteil nimmt man gern den Nachteil in Kauf, daß die Zahlen vor der Bearbeitung und nach der Berechnung umgeformt werden müssen und daß sich die Rechengeschwindigkeit verringert. Wir wollen jedoch festhalten, daß einige Probleme nur in Festkommadarstellung sinnvoll gerechnet werden können. Das trifft besonders für kommerzielle Rechnungen zu.
Daher haben moderne Automaten meist die Möglichkeit, in beiden Zahlendarstellungen zu rechnen.

1.4.3. Bemerkungen zum Rechnen mit ganzen Zahlen

Viele Probleme lassen sich am zweckmäßigsten in ganzen Zahlen rechnen. Das läßt sich auch mit der Festkommadarstellung erreichen, bei der ein Zahlenbereich

$$0 \leqq |Z| \leqq 0{,}999\,999\,999\,999$$

vorgegeben ist. Im Automaten wird ja lediglich die Ziffernfolge nach dem Komma verwandt. Die kleinste Zahl kann dabei als ganze Zahl angesehen werden

$$0{,}000\,000\,000\,001 \triangleq \ldots\ldots\ldots\ldots 1$$

Beim Rechnen ergeben sich für die Addition und Subtraktion keine Schwierigkeiten, denn es gilt

$$\begin{array}{r} +\ \ldots\ldots\ldots\ldots 1 \\ +\ \ldots\ldots\ldots\ldots 1 \\ \hline \ldots\ldots\ldots\ldots 2 \end{array}$$

da der Automat rechnet

$$\begin{array}{r} 0{,}000\,000\,000\,001 \\ +0{,}000\,000\,000\,001 \\ \hline 0{,}000\,000\,000\,002 \end{array}$$

Anders hingegen bei der Multiplikation und Division. Hier liefert der Automat

$$\ldots\ldots\ldots 1 \times \ldots\ldots\ldots 1 = \ldots\ldots\ldots 0$$

denn es wird gerechnet:

$$0{,}000\,000\,000\,001 \times 0{,}000\,000\,000\,001 =$$
$$= 0{,}000\,000\,000\,000\,000\,000\,000\,001$$

da aber nur 12 Dezimalziffern gewertet werden, ergibt sich Null. Wir können in diesem Falle jedoch noch zu einer stellenrichtigen Multiplikation kommen, wenn der Automat eine Verschiebeoperation hat. Dann rechnen wir:

$0{,}000\,000\,000\,001 \times 0{,}000\,000\,000\,001$ wird verschoben zu $0{,}000\,001 \times 0{,}000\,001$ und liefert $0{,}000\,000\,000\,001$.

Ähnlich können wir durch geschicktes Umformen auch eine ganzzahlige Division erzielen, worauf hier aber nicht eingegangen werden soll.

2. Probleme der formalen geistigen Arbeit

Erinnern wir uns noch einmal an die materielle Produktion. Eine notwendige Begleiterscheinung der industriellen Revolution in der Mitte des vorigen Jahrhunderts war es, daß sich durch Einführung der Maschinen die Technologie wesentlich veränderte. Es mußte untersucht werden, welche Bearbeitungsgänge einander ähnlich waren oder durch andere ersetzt werden konnten, für die man dann spezielle Maschinen entwickelte (Drehmaschinen, Fräsmaschinen u. a.). Ähnliches gilt auch für die in der wissenschaftlich-technischen Revolution von Maschinen auszuführenden geistigen Prozesse.

Das Grundproblem besteht hier darin, den Gesamtprozeß so in Elementarschritte zu zerlegen, daß diese Elementarschritte durch Maschinen oder Geräte ausgeführt werden können. Gelingt dies, kann meist auch der Gesamtprozeß durch Maschinen bearbeitet und gelöst werden.

Es hat sich gezeigt, daß sich alle maschinell zu bearbeitenden geistigen Prozesse auf einige wenige logische Grundoperationen zurückführen lassen, für die es elektromechanische oder elektronische Grundschaltungen gibt, aus denen man dann Automaten aufbauen kann, die das Gesamtproblem lösen.

Die geistigen Prozesse, die eine automatische Bearbeitung zulassen, faßt man vielfach unter dem Begriff formale geistige Arbeit zusammen. Automaten können den Menschen also die formale geistige Arbeit abnehmen.

Das Zurückführen eines Gesamtprozesses auf die formallogischen Grundoperationen erfolgt über einzelne Stufen.

Gehen wir von einem allgemeinen Sachverhalt aus, so müssen

wir als erstes eine Problemstellung erarbeiten. Hierbei kommt es zu Einschränkungen, denn der Sachverhalt ist meist in der allgemeinen Form einer geistigen Bearbeitung nicht zugänglich. Auf ihn wirken zu viel Faktoren und Gesetzmäßigkeiten ein, die wir möglicherweise noch gar nicht erkannt haben oder nicht erfassen können.

Wir kommen zu einer Abstraktion, indem wir uns auf einige wesentliche Seiten des Sachverhaltes orientieren und bewußt oder unbewußt andere Seiten vernachlässigen.

Damit haben wir einen Gewinn erzielt. Durch die Abstraktion erhalten wir eine spezielle Fragestellung, die wir einer weiteren geistigen Bearbeitung unterziehen können.

Nehmen wir einmal die allgemeine Situation, daß wir von der einen Seite eines Flusses auf dessen anderes Ufer wollen. Dafür gibt es viele Möglichkeiten. Wir können eine andere Stelle mit einer Furt suchen. Wir können einen Fährbetrieb einrichten, den Fluß umleiten, eine Brücke bauen oder anderes mehr. Entschließen wir uns dazu, eine Brücke zu bauen, haben wir bereits eine spezielle Situation erhalten.

Brücken gibt es aber in unterschiedlichster Ausführung. Wir prüfen den über die Brücke zu erwartenden Verkehr, den Verkehr auf dem Fluß und das Baugelände. Auch berücksichtigen wir die ökonomischen Möglichkeiten und entschließen uns für einen speziellen Brückentyp – nehmen wir an, eine Bogenbrücke. Damit liegt eine Problemstellung vor, aus der unmittelbar auch spezielle Fragestellungen abgeleitet werden können. So interessieren beispielsweise die geometrischen Formen, die Abmessungen, die Statik, die Ausführungsart, die Ablaufplanung und vieles mehr. Für diese Untersuchungen liefert die Mathematik mit ihren Methoden das Handwerkszeug, um uns über objektive Schlüsse die Daten zu ermitteln, die für die Ausführung des Vorhabens von Interesse sind.

Es ist uns somit gelungen, einen allgemeinen Sachverhalt mit Hilfe der Problemstellung soweit zu präzisieren, daß mathematische Methoden angewandt werden können – ein mathematisches Modell aufgestellt werden kann.

Damit haben wir allgemeine logische Denkvorgänge auf die Auswahl der in den mathematischen Methoden enthaltenen Prozesse und Operationen zurückgeführt. Dies ist ein sehr wichtiger

Schritt auf dem Wege, die Elementarprozesse der formalen geistigen Arbeit zu ermitteln, die von Automaten ausgeführt werden können. Das mathematische Modell bildet ein entscheidendes Zwischenglied zwischen der geistigen Arbeit schlechthin und deren automatischer Bearbeitung mittels Maschinen. Es erzwingt zudem eine straffe formallogische und systematische Behandlung des Problems. Das sind aber letztlich die Kennzeichen wissenschaftlicher Arbeit. Daher wird vielfach der Charakter einer Wissenschaft danach beurteilt, wieweit sie sich mathematischer Methoden als Hilfsmittel bedient. *Karl Marx* vertrat, wie *P. Lafargue* berichtet, hierzu die Ansicht: „*Eine Wissenschaft ist erst dann als voll entwickelt anzusehen, wenn sie dahin gelangt ist, sich der Mathematik bedienen zu können.*" Natürlich müssen durch die mathematische Erfassung weitere Einschränkungen des allgemeinen Sachverhaltes vorgenommen werden. Die Mathematik erfordert eine so spezielle Aufbereitung der Ausgangsdaten, daß bereits hieraus Schwierigkeiten erwachsen, die zu Einschränkungen führen. Auch lassen sich zur Zeit noch keineswegs alle von der gesellschaftlichen Praxis aufgeworfenen Sachverhalte erfassen. Zudem müssen diese jeweils unter Vernachlässigung nebensächlich erscheinender Gesetzmäßigkeiten und Zusammenhänge auf wesentliche Berechnungen präzisiert werden, die dann mit den Methoden einer mathematischen Disziplin bearbeitet werden können.
Doch auch mit der Anwendung der mathematischen Disziplinen in dieser allgemeinen Form haben wir keineswegs die maschinell bearbeitbaren Elementeoperationen der geistigen Arbeit erreicht. Es gelingt uns z. B. noch nicht einmal, ein so einfach aussehendes Integral, wie das der Gaußschen Glockenkurve

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}t^2} \, dt$$

geschlossen zu integrieren.
Wir müssen als nächsten Schritt Algorithmen für die wertmäßige Ermittlung neuer Daten finden. Gelingt es uns beispielsweise, das Ergebnis der statischen Berechnung, die Termine des Ausführungsablaufes, die Mengen der benötigten Materialien und

anderes durch eine Zahl oder eine Zahlenfolge zu kennzeichnen, so können wir mit diesem Ergebnis wiederum in den Sachverhalt zurückwirken, indem wir die vorgenommene Arbeit nach den Berechnungen ausführen – die Brücke also danach bauen.
Klären wir also erst den Begriff des Algorithmus, ehe wir in unseren Betrachtungen fortfahren.

2.1. Algorithmen

Der Algorithmus ist ein zentraler Begriff, der insbesondere in der maschinellen Rechentechnik große Bedeutung hat. Ohne auf seine Theorie eingehen zu wollen, soll er für unsere Arbeit charakterisiert werden.
Ein *Algorithmus* ist eine Rechenvorschrift, die folgenden Anforderungen genügen muß:

1. Sie muß für eine Klasse von Aufgaben gelten,
2. sie muß eindeutig sein,
3. sie muß allgemeinverständlich sein (das soll heißen, daß sie auch von Bearbeitern verstanden und anwendbar sein muß, die die Theorie selber nicht beherrschen) und
4. sie muß bei richtiger Vorgabe der Ausgangswerte und logisch richtiger Anwendung zwangsläufig nach endlich vielen Schritten zum richtigen Ergebnis führen.

Algorithmen treten in unterschiedlichster Form auf. Häufig finden wir sie als arithmetische Formeln zur Lösung numerischer Probleme. Beispielsweise ist für die Lösung der quadratischen Gleichung

$$ax^2 + bx + c = 0$$

der Algorithmus durch die bekannte Formel

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

gegeben. Dabei setzen wir voraus, daß der Bearbeiter die Quadratwurzel aus einer Zahl ziehen kann. Das ist keineswegs selbstverständlich. Ein Rechenautomat kann dieses Problem im allgemeinen nicht unmittelbar lösen. Er kann die Quadratwurzel nur über einen Algorithmus lösen.

Für jeden Radikanden R und für jede Ausgangsnäherung $u_0 \neq 0$ konvergiert der von *Newton* angegebene Algorithmus

$$\frac{1}{2}\left(u_{n-1} + \frac{R}{u_{n-1}}\right) = u_n$$

Wollen wir beispielsweise $\sqrt{2} = 1{,}41421\ldots$ berechnen, so können wir mit $u_0 = 2$ als Ausgangsnäherung beginnen und erhalten nach drei Schritten bereits eine Näherung, die auf vier Stellen nach dem Komma genau ist:

$$\frac{1}{2}\left(2 + \frac{2}{2}\right) = \frac{3}{2} = 1{,}5 = u_1$$

$$\frac{1}{2}\left(\frac{3}{2} + \frac{4}{3}\right) = \frac{17}{12} = 1{,}4166 = u_2$$

$$\frac{1}{2}\left(\frac{17}{12} + \frac{24}{17}\right) = \frac{577}{408} = 1{,}414216 = u_3$$

Der zuletzt angeführte Algorithmus ist für die Rechentechnik besonders interessant. Er demonstriert das Iterationsverfahren. Bei der Iteration ist der Algorithmus so aufgebaut, daß eine Näherungslösung, unter Einhalten eventuell vorgegebener Forderungen, bei jedem Durchlauf verbessert wird. Wir können die Berechnung also mit einer recht groben Ausgangslösung beginnen. Durch mehrmalige Iteration wird die Ausgangslösung so weit verbessert, wie es die verlangte Genauigkeit erfordert.

Ein Algorithmus kann aber auch durch eine Folge verbaler Anweisungen angegeben werden. Der Übung halber wollen wir den Algorithmus zur Konvertierung von Zahlen anführen:

An-weisung:	Instruktion:	dann folgt:
I	Alternative: $Z = 0$? falls ja falls nein	II III
II	Die Folge der Reste, von der letzten Division zur ersten gelesen, bildet die gesuchte konvertierte Zahl	HALT
III	Dividiere die Zahl mit Rest gemäß: $Z : B = Z'$ Rest r	IV
IV	Z' wird zu Z	I

Wir können Algorithmen schließlich durch spezielle algorithmische Sprachen angeben (s. 3.5.2.).
Zur wertmäßigen Ermittlung der Daten gibt es die numerische Mathematik. Sie bedient sich im wesentlichen der vier rationalen Grundoperationen: Addition, Subtraktion, Multiplikation und Division sowie einiger organisatorischer Operationen. Das Problem besteht darin, für die formelmäßige Vorgabe der Lösung einen Algorithmus zur wertmäßigen Lösung zu finden, der sich der obengenannten Operationen bedient.
Damit ist die Zahl der unterschiedlichsten mathematischen Methoden auf einige wenige Operationen reduziert. Es ist uns gelungen, die in so großer Vielzahl verwandten mathematischen Mittel, wie Matrizen, Integrale und Differentialgleichungen, im Grunde genommen auf die vier rationalen Grundoperationen zurückzuführen.
Doch auch dies sind noch nicht die geistigen Elementaroperationen. Um hier jedoch weiter zu kommen, müssen wir uns mit den rationalen Rechenoperationen näher befassen.

2.2. Rationale Rechenoperationen

Die vier Grundrechenarten – *Addition, Subtraktion, Multiplikation* und *Division* – bezeichnet man als *rationale Rechenoperationen.* Der Name ist darauf zurückzuführen, daß diese Operationen ohne Einschränkung auf rationale Zahlen anwendbar sind und daß die Ergebnisse dieser Operationen stets wieder auf rationale Zahlen führen. Rationale Zahlen wiederum sind alle Zahlen, die als gemeiner Bruch der Form

$$z = \frac{p}{q}$$

(p, q ganzzahlig) geschrieben werden können. Dazu gehören also auch die ganzen Zahlen, deren Nenner q den Wert 1 hat und nicht mitgeschrieben wird.
Die rationalen Rechenoperationen sind äußerst einfach. Man kann sie auf die Addition und einige Zusätze zurückführen.

2.2.1. Addition

Wollen wir zwei Zahlen addieren, so zählen wir für gleiche Stellenwerte die Ziffernwerte zusammen.

```
  3 2 1
+ 4 3 7
-------
  7 5 8
```

Dabei ergibt sich jedoch ein Problem dadurch, daß die Addition für einen oder mehrere Stellenwerte ein zweistelliges Ergebnis liefern kann

```
    5   7     6
+   7   2     9
-------------------
  (1)2   9  (1)5
```

Wir erhalten dann eine Spaltensumme s und einen Übertrag $ü$, den wir in Klammern anführen wollen und zur nächsthöheren Stellenwertsumme addieren müssen.

```
                5   7   6
+               7   2   9
-----------------------------
                2   9   5   Spaltensumme
+   1           0   1       Übertrag
-----------------------------
    (0)1    (0)2    (1)0  (0)5
```

Dabei kann nochmals ein Übertrag entstehen, der seinerseits in die nächsthöhere Stellensumme überführt werden muß

```
                5   7   6
+               7   2   9
-----------------------------
                2   9   5   Spaltensumme
+   1           0   1       Übertrag
-----------------------------
        1   2   0   5       Spaltensumme
+   0   0   1   0           Übertrag
-----------------------------
    0   1   3   0   5       Spaltensumme
+   0   0   0   0           Übertrag
-----------------------------
        1   3   0   5       Ergebnis
```

Erst wenn in keiner Spalte ein Übertrag entsteht, ist die Addition abgeschlossen.
Hieraus können wir folgern:

I. Die Addition mehrstelliger Zahlen erfolgt für jeden Stellenwert durch Addition der Ziffernwerte jeder Stelle.

II. Werden zwei Ziffern addiert, entsteht ein zweistelliges Ergebnis, bestehend aus der Spaltensumme s und dem Übertrag $\ddot{u}$, wobei der Übertrag auch den Wert Null annehmen kann.
Diesen Prozeß können wir durch den später zu besprechenden Halbadder technisch realisieren.
Die Werte für die Spaltensumme und für den Übertrag können wir einer *Spaltensummenmatrix* und einer *Übertragsmatrix*[1] entnehmen.

III. Für die volle Addition müssen wir aber drei Eingänge berücksichtigen:

a_k Ziffer a der k-ten Spalte
b_k Ziffer b der k-ten Spalte
$\ddot{u}_{k-1}$ Übertrag aus der $(k - 1)$-ten Spalte.

Als Ergebnis erhält man hier ebenfalls

s_k die Spaltensumme der k-ten Spalte
$\ddot{u}_k$ den Übertrag der k-ten Spalte, der in die $(k + 1)$-te Spalte geleitet werden muß.

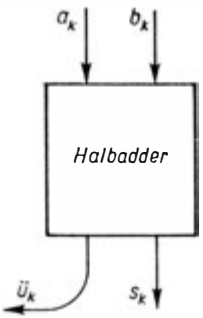


Bild 8. Schema eines Halbadders

[1] Matrix: rechteckige Anordnung von Elementen

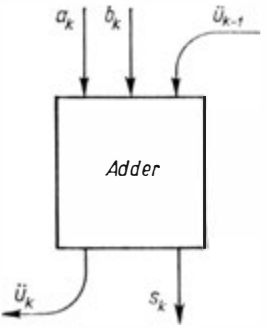


Bild 9. Schema eines Adders

Dieser Prozeß ergibt bei seiner technischen Realisierung dann den Adder.
Damit können wir ein komplettes Rechenwerk aufbauen. Nehmen wir an, daß wir sechsstellige Zahlen verarbeiten wollen, dann sind sechs Adder nötig, um die Addition technisch zu realisieren. Strenggenommen benötigen wir nur 5 Adder und einen Halbadder, da ja in der 1. Stelle noch kein Übertrag zu berücksichtigen ist. Bleibt vorerst noch das Problem, die Spaltensummenmatrix und die Übertragsmatrix aufzustellen.
Führen wir das im Dezimalsystem aus:
Hierzu müssen wir für alle Kombinationen der Ziffer 0, 1, . . ., 9 zu zweien mit Berücksichtigung der Stellung die Summen bilden. Also

$$0 + 0 = (0)0;\ 0 + 1 = (0)1;\ \ldots;\ \text{bis}\ \ldots;\ 9 + 8 = (1)7;\ 9 + 9 = (1)8,$$

wobei der Übertrag wieder durch Klammern angedeutet wird.
Die Spaltensummenergebnisse tragen wir in die Spaltensummenmatrix so ein, daß der erste Summand in der Eingangsspalte und der zweite Summand in der Kopfzeile der Matrix s aufgenommen wird. Am Kreuzungspunkt wird dann das Ergebnis eingetragen. Den Übertrag, in Klammern angeführt, schreiben wir gleichermaßen in die Übertragsmatrix $\ddot{u}$.

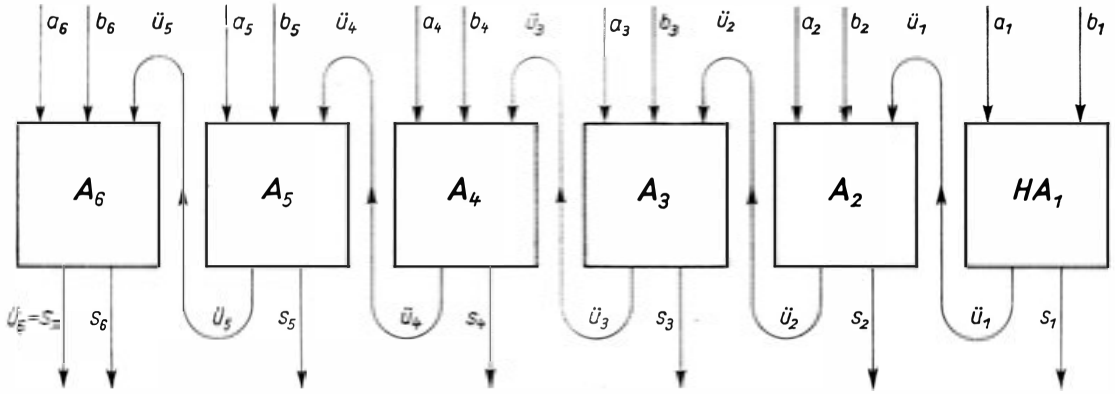


Bild 10. Skizze eines Addierwerkes für sechsstellige Summanden (Parallelverarbeitung)

Damit ergibt sich:

Übertragsmatrix

$\ddot{u}$	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	1	1	1
4	0	0	0	0	0	0	1	1	1	1
5	0	0	0	0	0	1	1	1	1	1
6	0	0	0	0	1	1	1	1	1	1
7	0	0	0	1	1	1	1	1	1	1
8	0	0	1	1	1	1	1	1	1	1
9	0	1	1	1	1	1	1	1	1	1

Spaltensummenmatrix

s	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

Ähnlich lassen sich Matrizen für andere Zahlensysteme herstellen, was dem Leser zur Übung empfohlen wird.
Besonders einfach wird die Situation im Dualsystem. Hier gibt es für die Addition der Ziffer nur die vier Möglichkeiten

O + O = (O)O
O + L = (O)L
L + O = (O)L
L + L = (L)O

und somit die Matrizen

$\ddot{u}$	O	L
O	O	O
L	O	L

s	O	L
O	O	L
L	L	O

Die Addition zweier Dualzahlen ergibt sich damit unter Berücksichtigung des mehrfach auftretenden Übertrages wie folgt:

7	O O O L L L	1. Operand
+ 5	+O O O L O L	2. Operand
12	O O O O L O	Spaltensumme
	+O O L O L	Übertrag

```
  O O L  O O O  Spaltensumme
+O O O  L O     Übertrag
-------------
  O O L  L O O  Spaltensumme
+O O O  O O     Übertrag
-------------
  O O L  L O O
=============
```

Da beim letzten Ergebnis kein Übertrag auftritt, ist dies das Gesamtergebnis.
Praktisch wird jede Spalte unmittelbar unter Berücksichtigung des Übertrages ausgerechnet, also

```
  O O O  L L L
+O O O  L O L
     L  L L
-------------
  O O L  L O O
=============
```

Es wird später unsere Aufgabe sein, die Matrizen für die Addition der Dualziffern technisch zu realisieren.

2.2.2. Subtraktion

In ähnlicher Weise könnten wir die Subtraktion diskutieren und dafür allgemeingültige Gesetzmäßigkeiten und Regeln, also einen Algorithmus, aufstellen. Wir wollen jedoch die Subtraktion so ausführen, daß sie unter Anwendung der Addition bewältigt werden kann.
Das ist relativ einfach durch folgende Überlegung möglich. Wir bilden, wenn a und b zwei beliebige positive Zahlen sind,

$$a - b = a + (0 - b).$$

In diesem Ausdruck wollen wir über die 0 wie folgt verfügen:

$$0 = 10^n - 10^n$$

wobei n so gewählt werden muß, daß 10^n um eine Stelle größer ist als die größte der an der Rechnung beteiligten Zahlen. Bei fester Wortlänge wird n gleich der Anzahl der Ziffern gewählt. Damit ergibt sich

$$\begin{aligned} a - b &= a + (0 - b) \\ &= a + (10^n - 10^n - b) \\ &= a + (10^n - b) - 10^n \end{aligned}$$

Beispiel:

		2	3	7	2	4	6	$n = 6$
—		1	1	3	2	4	3	$10^n = 1\,000\,000$
		2	3	7	2	4	6	
+	1	0	0	0	0	0	0	Zehnerkomplement
	—	1	1	3	2	4	3	
—	1	0	0	0	0	0	0	
		2	3	7	2	4	6	
+		8	8	6	7	5	7	Zehnerkomplement
—	1	0	0	0	0	0	0	
	1	1	2	4	0	0	3	
—	1	0	0	0	0	0	0	
		1	2	4	0	0	3	

Formal können wir $(0 - b)$ bilden, indem wir b von Null subtrahieren.
Diesen Ausdruck $(0 - b)$ wollen wir *B-Komplement*[1] oder, wenn wir uns vorerst auf das Dezimalsystem beschränken, *Zehnerkomplement* nennen.
Beispielsweise ergab sich in der oben durchgeführten Rechnung das Zehnerkomplement für 113 243 zu

1	0	0	0	0	0	0
—	1	1	3	2	4	3
	8	8	6	7	5	7

[1] Exakt wird $B^n - b$ als B-Komplement bezeichnet

und damit die Subtraktion:

	2	3	7	2	4	6	Subtrahend
—	1	1	3	2	4	3	Minuend
	2	3	7	2	4	6	Subtrahend
+	8	8	6	7	5	7	Zehnerkomplement des Minuenden
1	1	2	4	0	0	3	

was offensichtlich richtig ist, wenn wir die „übergelaufene" 1 nicht weiter berücksichtigen. Sie darf auch nicht beachtet werden, denn nach der allgemeinen Ableitung wird sie ja durch die Subtraktion mit 10^n (speziell 10^6) zu Null.
Scheinbar haben wir vorerst nichts gewonnen. Die Subtraktion scheint nur an eine andere Stelle verlegt worden zu sein. Dem ist nicht so, denn die Bildung des Zehnerkomplements kann nach einer sehr einfachen Vorschrift erfolgen:

I. Suche für die Einerstelle die Ziffer, die mit der gegebenen die Summe 10 bildet.

II. Suche für alle anderen Stellen die Ziffern, die mit den angegebenen die Summe 9 bilden.

Damit wird die Subtraktion durch folgenden Algorithmus auf eine Addition zurückgeführt:

1. Bilde für die negative Zahl das Zehnerkomplement.
2. Addiere beide Zahlen.
3. Lasse die überlaufende 1 unberücksichtigt.

Unbefriedigend ist es jedoch, daß für die Bildung des Komplements zwei Regeln berücksichtigt werden müssen. Das können wir vermeiden, indem wir das *Neunerkomplement* bilden.
Zur Ermittlung des Neunerkomplements wird nur die eine Vorschrift herangezogen:
Bestimme für alle Stellen die Ziffer, die, zur gegebenen addiert, 9 ergibt:

$$z_k + z_k' = 9.$$

Dann können wir auch mit diesem Neunerkomplement subtrahieren, wie folgendes Beispiel zeigt:

$$
\begin{array}{rrrrrrr}
 & 2 & 3 & 7 & 2 & 4 & 6 \\
- & 1 & 1 & 3 & 2 & 4 & 3 \\
\hline
 & 2 & 3 & 7 & 2 & 4 & 6 \\
+ & 8 & 8 & 6 & 7 & 5 & 6 \\
\hline
\textcircled{1} & 1 & 2 & 4 & 0 & 0 & 2 \\
 & & & & & & \downarrow \\
+ & & & & & & 1 \\
\hline
 & 1 & 2 & 4 & 0 & 0 & 3
\end{array}
$$

Jetzt muß allerdings der auftretende Überlauf zur letzten Stelle addiert werden.
Dies ergibt sich wiederum aus der allgemeinen Ableitung recht einfach. Wir verfügen in dem Ausdruck $(0 - b)$ jetzt über die Null folgendermaßen:

$$0 = (10^n - 1) - (10^n - 1)$$

Das liefert uns dann die Ableitung

$$
\begin{aligned}
a - b &= a + (0 - b) \\
&= a + [(10^n - 1) - (10^n - 1)] - b \\
&= a + [(10^n - 1) - b] - (10^n - 1) \\
&= a + [(10^n - 1) - b] - 10^n + 1
\end{aligned}
$$

$[(10^n - 1) - b]$ ist jetzt das Neunerkomplement, denn von einer Zahl, die aus genau n Neunerziffern besteht, wird b subtrahiert.

Beispiel:

$$
\begin{array}{rrrrrrr}
 & 2 & 3 & 7 & 2 & 4 & 6 \\
- & 1 & 1 & 3 & 2 & 4 & 3 \\
\hline
 & 2 & 3 & 7 & 2 & 4 & 6 \\
+ & \left[\begin{array}{rrrrrr} 1 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & \\ & & & & & & \\ - & & & & & & 1 \end{array}\right] & & & & & \\
+ & - & 1 & 1 & 3 & 2 & 4 & 3 \\
 & \left[\begin{array}{rrrrrr} 1 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & \\ & & & & & & \\ - & & & & & & 1 \end{array}\right] & & & & & \\
\hline
\end{array}
\qquad
\begin{array}{l}
n = 6 \\
10^6 - 1 = 999999
\end{array}
$$

```
                2 3 7   2 4 6
+ [ [ 1         0 0 0   0 0 0 ] ]
+ [ [ —                     1 ] ]   Neunerkomplement
    —           1 1 3   2 4 3
    —1          0 0 0   0 0 0
    +                       1
  ------------------------------
                2 3 7   2 4 6
+ [ +           9 9 9   9 9 9 ]     Neunerkomplement
    —           1 1 3   2 4 3 ]
    —1          0 0 0   0 0 0
    +                       1
  ------------------------------
                2 3 7   2 4 6
+ [ +           8 8 6   7 5 6 ]     Neunerkomplement
    —1          0 0 0   0 0 0
    +                       1
  ------------------------------
    1           1 2 4   0 0 2
    —1          0 0 0   0 0 0
    +                       1
  ------------------------------
                1 2 4   0 0 2
    +                       1
  ------------------------------
                1 2 4   0 0 3
```

Der Berechnungsalgorithmus für die Subtraktion läßt sich nun wie folgt beschreiben:

I. Überprüfe das Vorzeichen. Ist es +, folgt II. Ist es —, folgt III.

II. Lasse die Zahl unverändert, darauf folgt IV.

III. Bilde das Neunerkomplement, dann folgt IV.

IV. Addiere die Zahlen!

V. Addiere den eventuell auftretenden Überlauf zur letzten Stelle. Dann folgt HALT.

In gleicher Weise können wir im Oktalsystem verfahren. Allerdings entspricht dort dem Zehnerkomplement ein *Achterkomplement* und dem Neunerkomplement ein *Siebenerkomplement.* Erstere nennt man daher auch B-Komplement, wobei B die Basis des Zahlensystems ist und letztere $(B - 1)$-Komplement. Im Dualsystem ist die Benutzung des $(B - 1)$-Komplements besonders vorteilhaft.
Die Vorschrift lautet hier (*Einerkomplement*):
Bestimme für alle Stellen die Dualziffer, die zur gegebenen addiert ein L ergibt

$$d_k + d_k' = \mathrm{L}.$$

Probieren wir die Fälle durch.
Ist $d_k = \mathrm{L}$, so ergibt sich

$$\mathrm{L} + d_k' = \mathrm{L}.$$

Folglich muß $d_k' = 0$ sein.
Ist $d_k = 0$, folgt

$$0 + d_k' = \mathrm{L},$$

also muß $d_k' = \mathrm{L}$ sein.
Das heißt aber doch, daß immer dort, wo in der Originalzahl ein L steht, im Komplement eine O und für die O ein L zu setzen ist. Diese Vorschrift

$$\mathrm{O} = \mathrm{L}^{1}$$
$$\mathrm{L} = \mathrm{O}$$

nennt man Negation. Die Negation wird durch Überstreichen gekennzeichnet, also

$$\overline{\mathrm{O}} \rightarrow \mathrm{L}^{2}$$
$$\overline{\mathrm{L}} \rightarrow \mathrm{O}$$

Damit kommen wir im Dualsystem zu der einfachen Regel für das Bilden des Einerkomplements:

---

[1] Das Zeichen $\rightarrow$ soll hier „wird zu" bedeuten

[2] $\overline{\mathrm{O}}$ gelesen „nicht O"

Das Einerkomplement wird im Dualsystem durch Negation jeder Ziffer gebildet:

$$d_k + \bar{d}_k = \mathrm{L}$$

Die Subtraktion kann dann in folgender Form ausgeführt werden:

	13	+	O	L	L	O	L		
—	6	—	O	O	L	L	O		
	13	+	O	L	L	O	L		
+	93	+	L	L	O	O	L		
①	06	①	O	O	L	L	O		
	1						L		
		+							
	7	+	O	O	L	L	L		

Schließen wir das Vorzeichen nach der Vorschrift

$$+ = \mathrm{O}$$
$$- = \mathrm{L}$$

in die Rechnung ein, so können wir einen Algorithmus für Addition und Subtraktion aufstellen, wie er auf S. 73 dargestellt ist. Diesen Algorithmus wollen wir am Beispiel erproben; wir wollen fünfstellige Dualzahlen benutzen, nehmen also an, daß die Wortlänge des Zahlwortes 5 Bit für die Zahl und 1 Bit für das Vorzeichen, also 6 Bit, beträgt, und nehmen ferner an, daß das Komma hinter der letzten Ziffer steht (also nur ganzzahlige Werte verarbeitet werden).

I.	Fall	+ 13 + 6 = +19	
	1.	$1 \Rightarrow i$	es folgt 2
	2.	+ OLLOL	„ 3
	3.	Vorzeichen +	„ 4
	4.	O OLLOL	„ 6
	6.	Alternative: nein	„ 7
	7.	$i + 1 \Rightarrow 2$	„ 2
	2.	+ OOLLO	„ 3
	3.	Vorzeichen +	„ 4

An-weisung:	Instruktion:	dann folgt:
1	Setze für i eine 1: $1 \Rightarrow i$[1]	2
2	Lies die i-te Zahl	3
3	Alternative: Vorzeichen + Vorzeichen —	 4 5
4	Ist das Vorzeichen +, so setze an diese Stelle 0	6
5	Ist das Vorzeichen —, so setze an diese Stelle L und bilde die Negation der Zahl	6
6	Alternative: War die i-te Zahl die letzte? ja nein	 8 7
7	Bilde $i + 1 \Rightarrow i$	2
8	Addiere die Zahlen über die Vorzeichenstelle hinaus	9
9	Addiere den Überlauf zur letzten Stelle	10
10	Alternative: Ist die Vorzeichenstelle besetzt durch O L	 11 12
11	Vorzeichenstelle ist O, setze dafür +, das Ergebnis ist die gesuchte Zahl	HALT
12	Vorzeichenstelle ist L, setze dafür — und bilde die Negation. Die Negation ist die gesuchte Zahl	 HALT

[1] Das Zeichen ⇒ wird als „ergibt" gelesen. Es wird auf S. 185ff. ausführlich besprochen

4.		O OOLLO	es folgt 6
6.		Alternative: ja	„ 8
8.		O OLLOL	
	+	O OOLLO	
	O	O LOOLL	„ 9
9.	⓪	O LOOLL	
	+	└——→ O	
		O LOOLL	es folgt 10
10.		Alternative O	„ 11
11.	+	LOOLL	„ HALT

Ergebnis: + LOOLL = +19

II. Fall (Wir wollen eine vereinfachte Schreibweise wählen. Dem Leser sei jedoch zur Übung empfohlen, die Ergebnisse mit ausführlicher Beschreibung zu ermitteln, wie im I. Fall angeführt.)

+ 13 — 6 = 7

	+ OLLOL
	— OOLLO
	O OLLOL
	L LLOOL
①	O OOLLO
	└——→ L
	O OOLLL
	+ OOLLL

III. Fall — 13 6 = —7

	— OLLOL
	+ OOLLO
	L LOOLO
	O OOLLO
⓪	L LLOOO
	└——→ O
	L LLOOO
	— OOLLL

IV. Fall $-13 - 6 = -19$

$$
\begin{array}{r}
-\ \mathrm{OLLOL} \\
-\ \mathrm{OOLLO} \\
\hline
\mathrm{L\ LOOLO} \\
\mathrm{L\ LLOOL} \\
\hline
\textcircled{1}\ \mathrm{L\ OLOLL} \\
\quad\ \llcorner\!\!\rightarrow\ \mathrm{L} \\
\hline
\mathrm{L\ OLLOO} \\
-\ \mathrm{LOOLL} \\
\hline\hline
\end{array}
$$

Wir wollen auch noch den Spezialfall betrachten, daß wir als Ergebnis eine Null erhalten:

V. Fall $+13 - 13 = 0$

$$
\begin{array}{r}
+\ \mathrm{OLLOL} \\
-\ \mathrm{OLLOL} \\
\hline
\mathrm{O\ OLLOL} \\
\mathrm{L\ LOOLO} \\
\hline
\textcircled{0}\ \mathrm{L\ LLLLL} \\
\quad\ \llcorner\!\!\rightarrow\ \mathrm{O} \\
\hline
\mathrm{L\ LLLLL} \\
-\ \mathrm{OOOOO} \\
\hline\hline
\end{array}
$$

Es ergibt sich die Besonderheit, daß die Null negativ ist. Diese Tatsache müssen wir beispielsweise beim Programmieren für den Cellatron SER 2[1] berücksichtigen.
Prinzipiell ist es uns aber gelungen, die Subtraktion auf die Addition und die Komplementbildung zurückzuführen. Betrachten wir dies insbesondere im Dualsystem, so vereinfacht sich das Zurückführen der Subtraktion auf die Addition und Negation.

2.2.3. Multiplikation

Es bereitet keine Schwierigkeiten, die Multiplikation auf die Addition zurückzuführen.
Das zeigt uns ein einfaches Beispiel:

[1] Cellatron SER 2: Kleinrechenautomat, s. S. 205

```
      2 1 3 4 7 · 2 3 1
-------------------------
      2 1 3 4 7
+   2 1 3 4 7
+   2 1 3 4 7
+   2 1 3 4 7
+ 2 1 3 4 7
+ 2 1 3 4 7
-------------------------
  4 9 3 1 1 5 7
```

Wir benötigen also die Addition und Linksverschiebung. Besonders einfach wird dies, wenn wir im Dualsystem rechnen. Da nur die Ziffern O oder L auftreten können, wird bei jeder Linksverschiebung entweder bei L der zweite Faktor addiert, oder es wird bei O gleich anschließend die nächste Linksverschiebung ausgeführt.

Beispiel:

```
5 · 9 = 45          O O L O L · O L O O L
-------------------------------------------
              O O L O L
          O O O O O
      O O O O O
  O O L O L
O O O O O
-------------------------------------------
          L O L L O L
```

2.2.4. Division

Auch die Division läßt sich, allerdings erst über die Subtraktion, auf die Addition zurückführen.

Beispiel:

```
  3771120 : 31426 = 120
- 31426
---------
   62852
-  31426
---------
   31426
-  31426
---------
       000
```

Der Divisor wird vom höchstmöglichen Stellenwert des Dividenden so oft subtrahiert, bis der Rest kleiner als der Divisor ist. Dann wird um eine Stelle nach rechts verschoben und das Verfahren bis zur gewünschten Genauigkeit wiederholt. Damit wird die Division zurückgeführt auf die Subtraktion und Rechtsverschiebung und so auf Addition, Komplementbildung und Rechtsverschiebung.
Nunmehr ist es uns gelungen, die rationalen Rechenoperationen auf

Addition,
Komplementbildung und
organisatorische Operationen

zu reduzieren.
Betrachten wir nur das Dualsystem, so vereinfacht sich dies auf duale Addition, Negation und organisatorische Operationen.
Wir brauchen nur noch für diese Operationen technische Aggregate zu finden, um die damit erfaßbaren geistigen Arbeiten zu automatisieren. Dennoch haben wir auch hiermit die geistigen Elementaroperationen nicht erreicht, die durch Maschinen ausgeführt werden können. Selbst diese einfache Rechenweise ist primär für den Automaten zu kompliziert.

2.3. Addition bei dual codierten Dezimalzahlen

Wir wollen eine Besonderheit bei der Addition dual codierter Dezimalzahlen besprechen.
Werden zwei dual codierte Dezimalzahlen addiert – wir wollen uns vorwiegend auf die direkte dezimal-duale Codierung orientieren –, dann entstehen Schwierigkeiten bei der Bildung des Übertrages: Ist der Übertrag O, können die codierten Dualtetraden unmittelbar addiert werden.

	3	OOLL		1	OOOL		3	OOLL
+	6	+OLLO	+	2	+OOLO	+	5	+OLOL
	9	LOOL		3	OOLL		8	LOOO

Tritt jedoch ein Übertrag auf, so wird das Ergebnis verfälscht, indem entweder Dualtetraden entstehen, die bei vorgegebener Codierung keiner Dezimalziffer entsprechen, beispielsweise

3	OOLL	6	OLLO	9	LOOL
+ 7	+OLLL	+ 5	+OLOL	+ 6	+OLLO
10	LOLO	11	LOLL	15	LLLL

wobei offensichtlich kein Übertrag erzeugt wird. Andererseits ergibt sich zwar ein Übertrag, aber die entstandene Spaltensumme ist falsch.

Zum Beispiel:

7	OLLL	9	LOOL	8	LOOO
+ 9	+ LOOL	+ 9	+ LOOL	+ 9	+ LOOL
16	(L)OOOO	18	(L)OOLO	17	(L)OOOL

Die letzten Beispiele zeigen, daß die Verfälschung gerade sechs Einheiten beträgt.
Diese Erscheinung ergibt sich aus der Tatsache, daß im Dezimalsystem bei der Spaltensumme 10 ein Übertrag erfolgt.
Die Darstellung der Zahl in Dualtetraden entspricht aber einem Hexadezimalsystem, und der Übertrag erfolgt, wenn 15 überschritten wird.

Beispiel:

1 + 0 = (0) 1	OOOL + OOOO = (O) OOOL
1 + 1 = (0) 2	OOOL + OOOL = (O) OOLO
1 + 2 = (0) 3	OOOL + OOLO = (O) OOLL
1 + 3 = (0) 4	OOOL + OOLL = (O) OLOO
1 + 4 = (0) 5	OOOL + OLOO = (O) OLOL
1 + 5 = (0) 6	OOOL + OLOL = (O) OLLO
1 + 6 = (0) 7	OOOL + OLLO = (O) OLLL
1 + 7 = (0) 8	OOOL + OLLL = (O) LOOO
1 + 8 = (0) 9	OOOL + LOOO = (O) LOOL
1 + 9 = (1) 0	OOOL + LOOL = (O) LOLO
	OOOL + LOLO = (O) LOLL
	OOOL + LOLL = (O) LLOO

$$
\begin{aligned}
\text{OOOL} + \text{LLOO} &= (\text{O})\,\text{LLOL} \\
\text{OOOL} + \text{LLOL} &= (\text{O})\,\text{LLLO} \\
\text{OOOL} + \text{LLLO} &= (\text{O})\,\text{LLLL} \\
\text{OOOL} + \text{LLLL} &= (\text{L})\,\text{OOOO}
\end{aligned}
$$

Zur Überwindung dieser Schwierigkeit ist eine sogenannte *tetradische Korrektur* auszuführen.
Sie lautet bei der direkten dezimaldualen Codierung:
Übersteigt die Spaltensumme bei der Addition direkt dezimaldual codierter Zahlen den Wert 9, so ist zur Tetrade der Spaltensumme die Korrekturtetrade

$$6 = \text{OLLO}$$

zu addieren.
Damit ergibt sich dann auch der richtige Übertrag.
Korrigieren wir die vorhin betrachteten Beispiele.

Das liefert

OOLL	OLLO	LLOO
+ OLLL	+ OLOL	+ OLLO
LOLO	LOLL	LLLL
+ OLLO	+ OLLO	+ OLLO
(L)OOOO	(L)OOOL	(L)OLOL

OLLL	LOOL	LOOO
+ LOOL	+ LOOL	+ LOOL
(L)OOOO	(L)OOLO	(L)OOOL
+ OLLO	+ OLLO	+ OLLO
(L)OLLO	(L)LOOO	(L)OLLL

was offensichtlich zu richtigen Ergebnissen führt.
Ähnliche Korrekturen müssen auch für die Aiken- und die Dreierexzeß-Verschlüsselung vorgenommen werden.
Für die Addition der Gesamtzahl empfiehlt es sich, spaltenweise die korrigierte Summe zu bilden. Der dabei entstehende Überlauf kann dann unmittelbar in der nächsthöheren Spalte berücksichtigt werden.

Beispiel:

	2	3	7	2	4
+	3	2	6	9	1
	5	6	4	1	5

OOLO	OOLL	OLLL	OOLO	OLOO	1. Summand
OOLL	OOLO	OLLO	LOOL	OOOL	2. Summand
OLOL	OLOL	LLOL	LOLL	OLOL	unkorrigierte Summe
		OLLO	CLLO		Korrekturfaktor
	OOOL	OOOL			Übertrag
OLOL	OLLO	OLOO	OOOL	OLOL	korrigierte Summe

Die Subtraktion kann auch hier auf die Komplementbildung und Addition zurückgeführt werden. Das bereitet bei der direkten Codierung jedoch einige Schwierigkeiten, weil das Komplement berechnet werden muß.
Die Aiken- und Dreierexzeß-Codierung können die Komplementbildung durch Negation bilden, haben dafür aber eine kompliziertere tetradische Korrektur.
Multiplikation und Division sind sinngemäß auszuführen.

2.4. Boolesche Algebra

Zur Ermittlung der Elementaroperationen geistiger Arbeit müssen wir vorübergehend das Rechnen mit Zahlen verlassen. Wir betrachten allgemeine Aussagen, von denen wir lediglich fordern wollen, daß sie „richtig" oder „falsch", „wahr" oder „unwahr" sind.
Behauptet jemand, „es regnet", so wollen wir lediglich zulassen, daß es tatsächlich regnet, dann ist die Behauptung „wahr", oder aber, daß es nicht regnet, dann wurde gelogen, und die Behauptung ist „unwahr". Da der Wahrheitsgehalt unserer Aussagen nur zwei unterschiedliche Werte annehmen kann, sprechen wir auch von einer „zweiwertigen Logik".
George Boole, einem bedeutenden englischen Mathematiker, gebührt das Verdienst, die dabei auftretenden Gesetzmäßigkeiten in der Form aufbereitet zu haben, daß sie Rechenvorschriften

ähneln. Man bezeichnet diese mathematische Disziplin, mit der er die Gesetzmäßigkeiten des logischen Schließens aufzeigte, auch als Boolesche Algebra.

George Boole wurde 1815 in Lincoln geboren, wo sein Vater einen kleinen Kramladen besaß. Durch seine Forschungsergebnisse war Boole bald so bekannt, daß er 1849 bereits einen Lehrstuhl im neu gegründeten Queens College in Irland erhielt.

Das war für die damals in England herrschenden Verhältnisse eine nahezu unvorstellbare Leistung. Die Anstrengungen untergruben jedoch seinen Gesundheitszustand derart, daß er 1864 kurz nach seinem 49. Geburtstag starb.

Boole gilt als einer der „reinsten" Mathematiker. Seine Forschungsergebnisse, und insbesondere die Boolesche Algebra, waren zu seiner Zeit das Vorbild aller „profanen Nutzanwendung" entkleideten Mathematik. Es klingt wie eine Ironie, daß er damit den Grundstein für die modernste angewandte mathematische Disziplin legte, für die maschinelle Rechentechnik.

Die Boolesche Algebra ist die Grundlage der Schaltalgebra, mit deren Hilfe programmgesteuerte Rechenautomaten entworfen und gebaut werden.

Die Boolesche Algebra läßt sich auf drei Operationen aufbauen.

2.4.1. „UND"-Verknüpfung oder Konjunktion

Wir vereinbaren mit unserem Freund:

„(A) wenn es Sonntag ist UND

(B) wenn die Sonne scheint, fahren wir ins Grüne."

Damit haben wir eine logistische Funktion dargestellt.

Die beiden Aussagen, A: wenn es Sonntag ist, B: wenn die Sonne scheint, wurden von uns durch ein „UND" verknüpft.

Diese Verknüpfung „UND" nennen wir Konjunktion. Wir wollen ihren Wahrheitsgehalt untersuchen. Die Aussagen A bzw. B können „wahr" oder „unwahr" sein. Das Ergebnis der Verknüpfung ermitteln wir durch Probieren.

I. A sowie B seien „wahr". Es ist Sonntag, und die Sonne scheint, dann fahren wir, wie vereinbart.

II. A sei „unwahr", es ist also kein Sonntag,
B sei „wahr", die Sonne scheint,
dann wollen wir aber nicht fahren.

III. A sei „wahr", B „unwahr", jetzt ist zwar Sonntag, aber die Sonne scheint nicht,
dann wollen wir auch nicht fahren.

IV. A sei „unwahr" und B sei „unwahr", jetzt ist kein Sonntag, und die Sonne scheint auch nicht,
dann wollen wir verständlicherweise ebenfalls nicht fahren.

Setzen wir das Fahren als „wahr", das Nicht-Fahren-Wollen als „unwahr", so erhalten wir mit Einführung des Zeichens $\wedge$ als „UND" folgende Relationen:

$$\begin{aligned}
&\text{A „wahr"} \wedge \text{B „wahr"} = \text{A} \wedge \text{B „wahr"}\\
&\text{A „unwahr"} \wedge \text{B „wahr"} = \text{A} \wedge \text{B „unwahr"}\\
&\text{A „wahr"} \wedge \text{B „unwahr"} = \text{A} \wedge \text{B „unwahr"}\\
&\text{A „unwahr"} \wedge \text{B „unwahr"} = \text{A} \wedge \text{B „unwahr".}
\end{aligned}$$

Jetzt brauchen wir nur noch allgemeine Symbole einzuführen, um eine Rechenvorschrift ableiten zu können. Für die zweiwertige Logik bietet sich das Dualsystem geradezu als Symbolik an.

Setzen wir „wahr" = L
„unwahr" = O

erhalten wir schließlich:

$$\begin{aligned}
&\text{L} \wedge \text{L} \Rightarrow \text{L}\\
&\text{O} \wedge \text{L} \Rightarrow \text{O}\\
&\text{L} \wedge \text{O} \Rightarrow \text{O}\\
&\text{O} \wedge \text{O} \Rightarrow \text{O}
\end{aligned}$$

(lies: L und L ergibt L)
oder unter Verwendung einer Ergebnismatrix:

$\wedge$	O	L
O	O	O
L	O	L

Ergebnismatrix der Konjunktion

[1] Das Zeichen $\Rightarrow$ wird „ergibt" gelesen. Es wird ausführlich auf S. 185 besprochen

2.4.2. „ODER"-Verknüpfung oder Disjunktion

Beispiel:

Wir waren Zeuge eines Verkehrsunfalls. Der Fahrer des Kraftwagens, der den Unfall verursachte, hat Fahrerflucht begangen, und wir werden nach der Farbe des Wagens gefragt.
Wir sind uns nicht sicher und antworten: „Die Farbe des Wagens war grün ODER blau."
Wir haben zwei Aussagen, A grün, B blau, durch die logistische Funktion „ODER" verknüpft. Dieses „ODER" soll jedoch nicht ausschließend sein, wie in dem Sprichwort: „Er ist mein Freund oder mein Feind."
Die Volkspolizei hat den Täter ermittelt und den Wagen sichergestellt. Jetzt können folgende Fälle eintreten, wenn wir als Verknüpfungszeichen für „ODER" $\vee$ einführen:

I. Der Wagen ist rot, also
A „unwahr"; B „unwahr" $= A \vee B$ „unwahr"

II. Der Wagen ist grün:
A „wahr"; B „unwahr" $= A \vee B$ „wahr"

III. Der Wagen ist blau:
A „unwahr"; B „wahr" $= A \vee B$ „wahr"

IV. Der Wagen ist oben grün, unten blau:
A „wahr"; B „wahr" $= A \vee B$ „wahr"

Führen wir auch hier als Symbol Dualzahlen ein, ergibt sich

$$O \vee O \Rightarrow O$$
$$L \vee O \Rightarrow L$$
$$O \vee L \Rightarrow L$$
$$L \vee L \Rightarrow L$$

oder als Ergebnismatrix der Disjunktion:

$\vee$	O	L
O	O	L
L	L	L

2.4.3. Negation

Wir treffen hier eine uns bereits bekannte logistische Funktion. Es sei die Aussage gegeben,

„es regnet",

dann kommt dieser Aussage der Wahrheitswert „wahr" oder „unwahr" zu.
Die Negation erhalten wir, indem wir der Aussage den Zusatz „nicht" hinzufügen, also behaupten,

„es regnet nicht".

Haben wir vorhin die Wahrheit gesagt, so ist unsere negierte Aussage „unwahr" und umgekehrt.
Die Negation wird durch Überstreichen gekennzeichnet. Die allgemeine Formulierung lautet dann für eine Aussage

$$\overline{\text{„wahr"}} = \text{„unwahr"}$$
$$\overline{\text{„unwahr"}} = \text{„wahr"}$$

oder unter Verwendung von Dualzahlen

$$\overline{\mathrm{L}} = \mathrm{O}$$
$$\overline{\mathrm{O}} = \mathrm{L}$$

Hier tritt das Problem der „Negation der Negation" auf. Im Sprachgebrauch setzen wir für schönes Wetter auch die sprachlich zwar nicht geschickte, aber gleichwertige Redewendung

„es ist kein schlechtes Wetter".

Für unsere allgemeine Vorschrift ergibt sich, daß eine doppelte Negierung den ursprünglichen Wahrheitswert erhält:

$$\overline{\overline{\mathrm{A}}} = \mathrm{A}$$

also $\overline{\overline{\mathrm{O}}} = \mathrm{O}$ und $\overline{\overline{\mathrm{L}}} = \mathrm{L}$.

Prinzipiell gibt es noch mehr logistische Funktionen zweier Veränderlicher. Es lassen sich aber alle auf die besprochene Konjunktion, Disjunktion und Negation zurückführen.

2.4.4. Technische Realisierung der Booleschen Algebra

Die besprochenen Funktionen der Booleschen Algebra lassen sich sehr einfach technisch realisieren.
Betrachten wir dazu als Aussage lediglich das Vorhandensein eines Stromimpulses. Diesem Fall wollen wir den Wahrheitswert L zuschreiben oder dessen Ausbleiben, was dem Wahrheitswert O entsprechen soll. Als Schaltelemente nehmen wir vorerst einfache Schalter, die zwei Stellungen einnehmen können:

Arbeitsstellung	der Impuls wird weitergeleitet (L)
oder	
Ruhestellung	der Impuls wird nicht weitergeleitet (O).

Dann liefert die auf Bild 11 dargestellte Schaltung die Konjunktion, denn es gilt:

1. Schalter 1: O
   Schalter 2: O
   dann wird der Impuls nicht weitergeleitet, also O
2. Schalter 1: L
   Schalter 2: O
   dann wird der Impuls nicht weitergeleitet, also O
3. Schalter 1: O
   Schalter 2: L
   dann wird der Impuls auch nicht weitergeleitet, also O
4. Schalter 1: L
   Schalter 2: L
   dann wird der Impuls weitergeleitet, da der Stromfluß geschlossen ist, also L

Dies liefert als Ergebnismatrix die Konjunktion

$\wedge$	O	L
O	O	O
L	O	L

die wir somit durch Hintereinanderschalten der Schaltelemente erhalten.

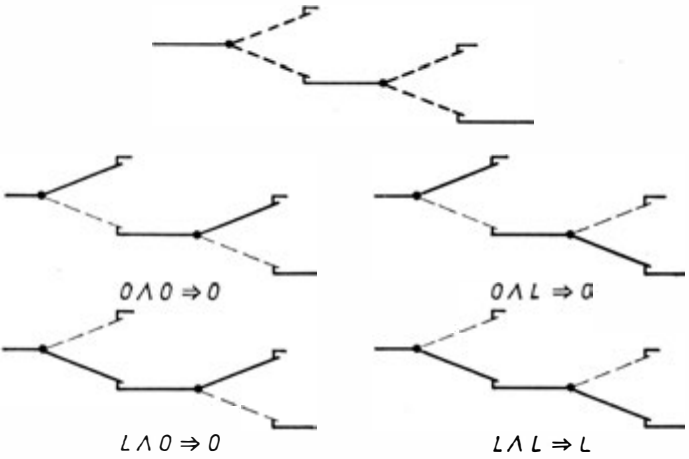


Bild 11. Schaltermodell der Konjunktion

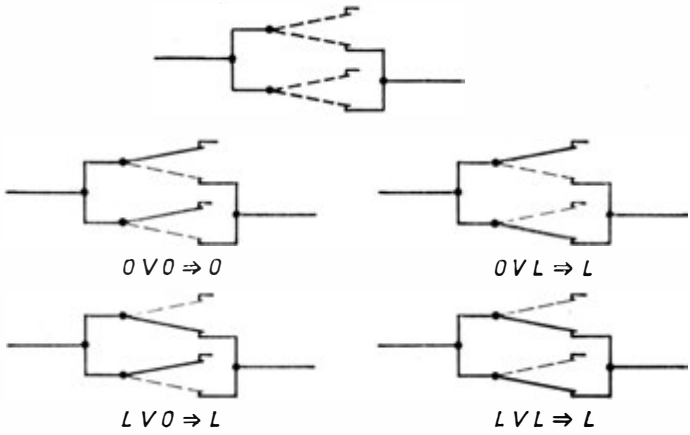


Bild 12. Schaltermodell der Disjunktion

Die Disjunktion erhalten wir durch Parallelschalten zweier Schaltelemente (Bild 12).

Es ergeben sich die Fälle

1. Schalter 1: O
   Schalter 2: O
   Der Impuls wird nicht weitergeleitet, da beide Zweige unterbrochen sind, also O

2. Schalter 1: L
   Schalter 2: O
   Der Impuls wird weitergeleitet, da er über den durch Schalter 1 geschlossenen Zweig fließen kann, also L

3. Schalter 1: O
   Schalter 2: L
   Der Impuls wird weitergeleitet, wobei er über den durch Schalter 2 geschlossenen Zweig fließen kann, also L

4. Schalter 1: L
   Schalter 2: L
   Der Impuls wird weitergeleitet. Beide Zweige sind geschlossen, also L

Fassen wir das Ergebnis in einer Matrix zusammen, ergibt sich die Disjunktion

$\vee$	O	L
O	O	L
L	L	L

Für die Negation ist die Schaltung noch einfacher. Wir brauchen lediglich die Leitung am Ruhekontakt anzuschließen, dann wird das Ergebnis negiert, denn bei Arbeitsstellung wird dann kein Impuls weitergeleitet, also O, während bei Ruhestellung ein Impuls weitergeleitet wird, also L. Bild 13 zeigt das Modell.

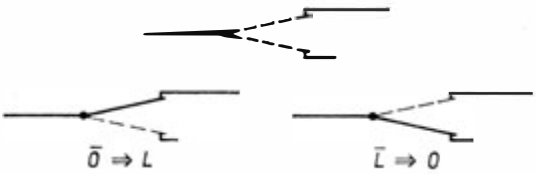


Bild 13. Schaltermodell der Negation

2.4.5. Technische Realisierung der dualen Addition

Die bisher gewonnenen Erkenntnisse müssen wir auf die duale Addition anwenden. Gelingt es uns, die Ergebnismatrizen

$\ddot{u}$	O	L
O	O	O
L	O	L

s	O	L
O	O	L
L	L	O

der dualen Addition durch Funktionen der Booleschen Algebra auszudrücken, können wir die Addition mechanisieren und schließlich den Gesamtkomplex umfangreicher Rechnungen automatisieren.
Für den Übertrag ist das einfach.
Er entspricht der Konjunktion

$\ddot{u}$	O	L
O	O	O
L	O	L

$\wedge$	O	L
O	O	O
L	O	L

und ist durch eine Serienschaltung von zwei Schaltelementen darstellbar.
Schwieriger ist dies für die Spaltensummen. Weder die Disjunktion noch die Konjunktion entsprechen der Ergebnismatrix.
Es gilt aber doch

$$a \wedge \mathfrak{a} \Rightarrow O$$
$$a \vee \mathfrak{a} \Rightarrow L$$

Dann ist

$\overline{a \wedge b}$	O L
O	L L
L	L O

Bilden wir nun

$$(a \vee b) \wedge \overline{(a \wedge b)} \Rightarrow z,$$

so haben wir in der Tat eine mögliche Realisierung der Spaltensummenmatrix, denn es ist für

I. $a = \mathrm{O};\ b = \mathrm{O}$

$$\begin{aligned} z &= (a \vee b) \wedge \overline{(a \wedge b)} \\ &= (\mathrm{O} \vee \mathrm{O}) \wedge \overline{(\mathrm{O} \wedge \mathrm{O})} \\ &= \mathrm{O} \wedge \overline{\mathrm{O}} \\ &= \mathrm{O} \wedge \mathrm{L} \\ &= \mathrm{O} \end{aligned}$$

II. $a = \mathrm{L};\ b = \mathrm{O}$

$$\begin{aligned} z &= (a \vee b) \wedge \overline{(a \wedge b)} \\ &= (\mathrm{L} \vee \mathrm{O}) \wedge \overline{(\mathrm{L} \wedge \mathrm{O})} \\ &= \mathrm{L} \wedge \overline{\mathrm{O}} \\ &= \mathrm{L} \wedge \mathrm{L} \\ &= \mathrm{L} \end{aligned}$$

III. $a = \mathrm{O};\ b = \mathrm{L}$

$$\begin{aligned} z &= (a \vee b) \wedge \overline{(a \wedge b)} \\ &= (\mathrm{O} \vee \mathrm{L}) \wedge \overline{(\mathrm{O} \wedge \mathrm{L})} \\ &= \mathrm{L} \wedge \overline{\mathrm{O}} \\ &= \mathrm{L} \wedge \mathrm{L} \\ &= \mathrm{L} \end{aligned}$$

IV. $a = \mathrm{L};\ b = \mathrm{L}$

$$\begin{aligned} z &= (a \vee b) \wedge \overline{(a \wedge b)} \\ &= (\mathrm{L} \vee \mathrm{L}) \wedge \overline{(\mathrm{L} \wedge \mathrm{L})} \\ &= \mathrm{L} \wedge \overline{\mathrm{L}} \\ &= \mathrm{L} \wedge \mathrm{O} \\ &= \mathrm{O} \end{aligned}$$

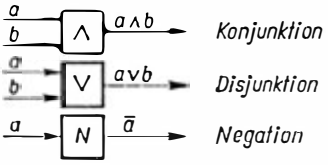
Somit

z	O	L
O	O	L
L	L	O

Wohlgemerkt, das ist eine Möglichkeit, um eine Spaltensumme technisch zu realisieren. Führen wir Symbole nach Bild 14 ein, so erhalten wir durch diese Schaltung einen Automaten, der die Addition zweier Spaltenziffern ohne Berücksichtigung des eventuellen Übertrages aus der nächstniedrigen Spalte ausführt. Diese Schaltung nennt man Halbadder (s. S. 62).
Probieren wir dies au

a	b	$\ddot{u}$	$\overline{\ddot{u}}$	c	s
O	O	O	L	O	O
L	O	O	L	L	L
O	L	O	L	L	L
L	L	L	O	L	O

mit $a \wedge b \Rightarrow \ddot{u}$
$a \vee b \Rightarrow c$
$c \wedge \overline{\ddot{u}} \Rightarrow s$



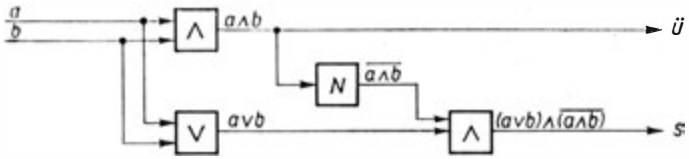


Bild 14. Halbadder nach $(a \vee b) \wedge \overline{(a \wedge b)} \Rightarrow s; a \wedge b \Rightarrow \ddot{u}$

Eine andere Schaltung ergibt sich aus den Formeln:

$$(a \wedge \bar{b}) \vee (\bar{a} \wedge b) \Rightarrow s$$
$$(a \wedge b) \Rightarrow \ddot{u}$$

denn für die vier möglichen Fälle liefert sie:

$a = \mathrm{O}$; $b = \mathrm{O}$
$$(a \wedge \bar{b}) \vee (\bar{a} \wedge b)$$
$$= (\mathrm{O} \wedge \bar{\mathrm{O}}) \vee (\bar{\mathrm{O}} \vee \mathrm{O})$$
$$= (\mathrm{O} \wedge \mathrm{L}) \vee (\mathrm{L} \vee \mathrm{O})$$
$$= \mathrm{O} \vee \mathrm{O}$$
$$= \mathrm{O}$$

$a = \mathrm{L}$; $b = \mathrm{O}$
$$(a \wedge \bar{b}) \vee (\bar{a} \wedge b)$$
$$= (\mathrm{L} \wedge \bar{\mathrm{O}}) \vee (\bar{\mathrm{L}} \wedge \mathrm{O})$$
$$= (\mathrm{L} \wedge \mathrm{L}) \vee (\mathrm{O} \wedge \mathrm{O})$$
$$= \mathrm{L} \vee \mathrm{O}$$
$$= \mathrm{L}$$

$a = \mathrm{O}$; $b = \mathrm{L}$
$$(a \wedge \bar{b}) \vee (\bar{a} \wedge b)$$
$$= (\mathrm{O} \wedge \bar{\mathrm{L}}) \vee (\bar{\mathrm{O}} \wedge \mathrm{L})$$
$$= (\mathrm{O} \wedge \mathrm{O}) \vee (\mathrm{L} \wedge \mathrm{L})$$
$$= \mathrm{O} \vee \mathrm{L}$$
$$= \mathrm{L}$$

$a = \mathrm{L}$; $b = \mathrm{L}$
$$(a \wedge \bar{b}) \vee (\bar{a} \wedge b)$$
$$= (\mathrm{L} \wedge \bar{\mathrm{L}}) \vee (\bar{\mathrm{L}} \wedge \mathrm{L})$$
$$= (\mathrm{L} \wedge \mathrm{O}) \vee (\mathrm{O} \wedge \mathrm{L})$$
$$= \mathrm{O} \vee \mathrm{O}$$
$$= \mathrm{O}$$

Also als Ergebnismatrix

$(a \wedge \bar{b}) \vee (\bar{a} \wedge b)$	O L
O	O L
L	L O

Dem entspricht wieder die Spaltensumme.

Der Übertrag ist, bezogen auf die erste Schaltung, unverändert. Auch hier haben wir eine einfache technische Realisierung des Halbadders (Bild 15)
und in der Tat ergibt sich:

a	b	$\ddot{u}$	$\bar{a}$	$\bar{b}$	$a \wedge \bar{b}$	$\bar{a} \wedge b$	s
O	O	O	L	L	O	O	O
L	O	O	O	L	L	O	L
O	L	O	L	O	O	L	L
L	L	L	O	O	O	O	O

mit $a \wedge b \Rightarrow \ddot{u}$
$(a \wedge \bar{b}) \vee (\bar{a} \wedge b) \Rightarrow s$

Die letzte Schaltfunktion kann man auch leicht durch ein Schaltermodell darstellen (Bild 16).
Die Gleichwertigkeit der beiden angeführten Schaltungen läßt sich unmittelbar durch folgende Umwandlung nachweisen:

$$\begin{aligned}(a \wedge \bar{b}) \vee (\bar{a} \wedge b) &= [a \vee (\bar{a} \wedge b)] \wedge [\bar{b} \vee (\bar{a} \wedge b)] \\ &= (a \vee \bar{a}) \wedge (a \vee b) \wedge (\bar{a} \vee \bar{b}) \wedge \\ &\quad (b \vee \bar{b}) \\ &= (\mathrm{L} \wedge (a \vee b) \wedge (\bar{a} \vee \bar{b}) \wedge \mathrm{L} \\ &= (a \vee b) \wedge (\bar{a} \vee \bar{b}) \\ &= (a \vee b) \wedge \overline{(\bar{a} \wedge b)}\end{aligned}$$

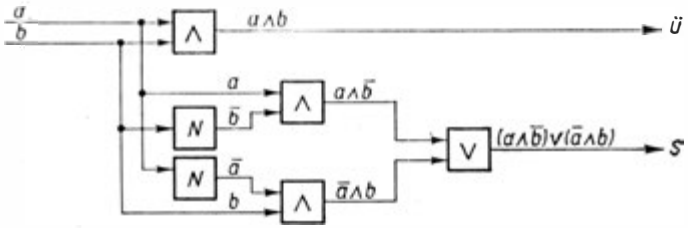


Bild 15. Halbadder nach $(a \wedge \bar{b}) \vee (\bar{a} \wedge b) \Rightarrow s$; $a \wedge b \Rightarrow \ddot{u}$

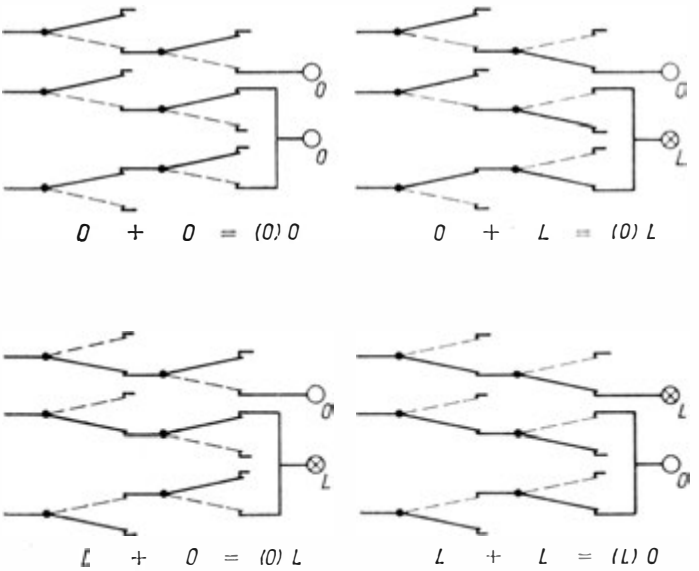


Bild 16. Schaltermodell eines Halbadders (alle drei übereinandergezeichneten Schalter werden gleichzeitig geschaltet)

wenn wir berücksichtigen, daß gilt

$$a \vee \bar{a} = \mathrm{L}$$
$$\mathrm{L} \wedge a = a$$
$$\bar{a} \vee \bar{b} = \overline{a \wedge b}$$

Mit Hilfe der Booleschen Algebra können wir noch viele Formeln zur Schaltung eines Halbadders gewinnen, wobei die entsprechenden Schaltungen für die unterschiedlichen bistabilen Schaltelemente mehr oder weniger vorteilhaft sind.

Für die theoretischen Untersuchungen bei der Entwicklung und Konstruktion von Rechenautomaten gibt es eine eigene wissenschaftliche Disziplin, die *Schaltalgebra*. Ihre Grundlage bildet die Boolesche Algebra.

Eine Aufgabe der Schaltalgebra ist es auch, die für die jeweils verwendeten Schaltelemente günstigste Schaltung zu ermitteln.

An diese werden verschiedene Anforderungen gestellt, wie hohe Betriebssicherheit, hohe Operationsgeschwindigkeit des geschalteten Systems, minimale Anzahl an Schaltelementen und vieles andere. Diese Forderungen widersprechen sich teilweise. Mit Hilfe der Schaltalgebra kann die Einzelschaltung wie auch die Schaltung des gesamten Automaten aufgestellt, untersucht und optimiert werden.

Wie wir uns nunmehr überzeugen konnten, existieren mit den Booleschen Verknüpfungen geistige Elementarprozesse, die durch technische Aggregate ausgeführt werden können. Damit haben wir die uns interessierende Endstufe erreicht und können uns jetzt der Frage zuwenden, wie diese technische Realisierung in Rechenautomaten erfolgt.

3. Digitalrechner

3.1. Historischer Abriß

Nahezu gleichzeitig mit dem Entstehen der Zahlen entwickelten sich auch die ersten Rechenhilfsmittel für das digitale Rechnen. Die Geschichte der Digitalrechner können wir in drei Abschnitte gliedern:

1. Entwicklung der Fingerrechenmaschine,
2. Mechanisierung des Zehnerübertrages und Bau von Tischrechenmaschinen,
3. Einsatz programmgesteuerter Rechenautomaten.

Wie bereits der Name andeutet, waren die Finger das erste Rechenhilfsmittel. Auf dem Prinzip der Fingerrechnung beruhen die ältesten Rechenmaschinen, bei denen gleiche Gegenstände in übersichtlicher Form angeordnet und jeweils ausgezählt werden. Der Prototyp dieser Rechenmaschinen ist der altrömische Abakus.
Bei ihm sind in einer Tafel mehrere Einschnitte angebracht, und zwar für jeden Stellenwert einer. In diese Abschnitte können kreisförmige Knöpfe gelegt und verschoben werden. Der Zehnerübertrag erfolgt „von Hand".
Nach diesem Vorbild sind die *Rechenbretter* aufgebaut, wie sie teils heute noch in der VR China, in Japan, in einigen Republiken der UdSSR und anderen Ländern benutzt werden.
Forschungsergebnisse der letzten Jahre weisen nach, daß die älteste Rechenmaschine mit automatischer Zehnerübertragung

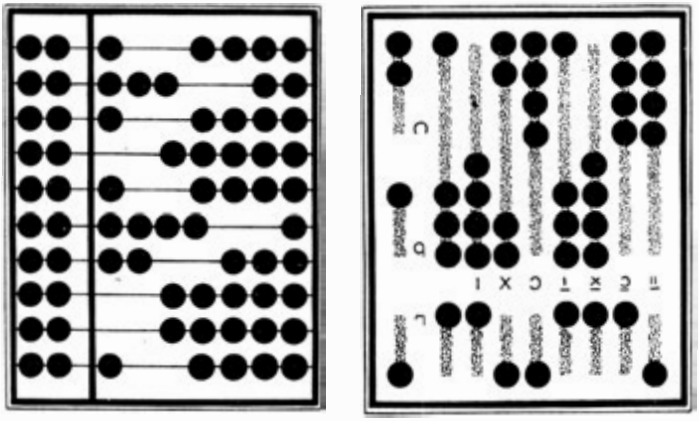


Bild 17. Rechenbrett und römischer Abakus

auf Anregung *Keplers* gebaut wurde. Es fand sich bei Keplers Papieren zu den *Rudolfinischen Tafeln* die Federzeichnung einer Rechenmaschine.

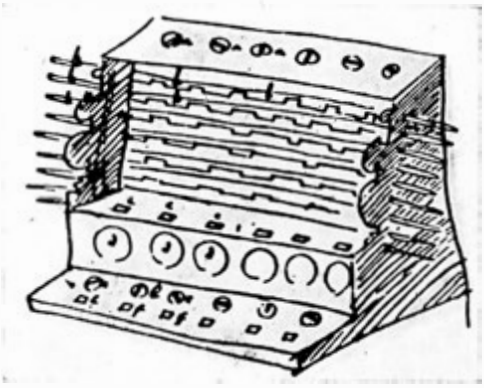


Bild 18. Skizze der ersten Rechenmaschine mit mechanischer Zehnerübertragung

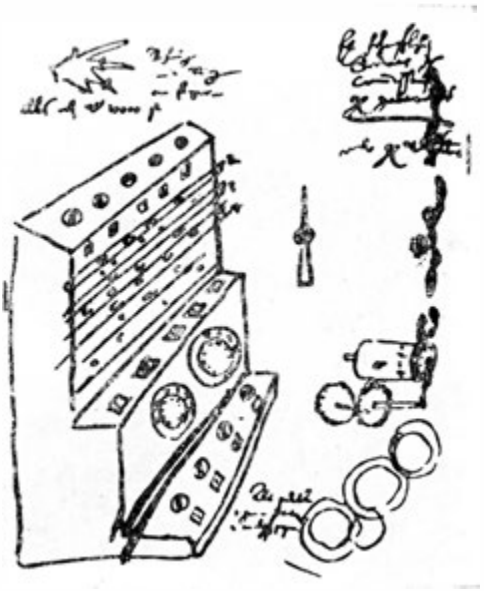


Bild 19. „Konstruktionszeichnung", nach der *Pfister* die erste Rechenmaschine mit mechanischer Zehnerübertragung baute

Sie stammte aus einem Brief des Tübinger Professors für Mathematik und Astronomie *Wilhelm Schickard*, den er am 25. 2. 1624 an Kepler gesandt hatte. Schickard teilte Kepler darin mit, daß er die abgebildete Rechenmaschine nach einem schon vorher fertiggestellten Muster für Kepler habe bauen lassen, daß sie aber durch einen Brand zerstört worden sei. Über das erwähnte Muster gibt ein weiterer Brief Auskunft. Bereits am 20. 9. 1623 schrieb Schickard an Kepler, daß er die von ihm angeregte Rechenmaschine fertiggestellt habe, die auch multiplizieren und dividieren könne und mit einer automatischen Zehnerübertragung versehen sei.

Inzwischen hat sich auch die „Konstruktionszeichnung" in der Stuttgarter Staatsbibliothek gefunden, nach der ein Tübinger

Mechaniker namens *Pfister* diese erste Rechenmaschine gebaut hatte. Vermutlich gab es drei Exemplare dieser Rechenmaschinen, die jedoch alle verlorengingen.

1641 baute der französische Mathematiker *Blaise Pascal* (1623 bis 1662) eine Rechenmaschine für die Addition und Subtraktion mit automatischer Zehnerübertragung (Zweispeziesrechner). 1673 führte der deutsche Mathematiker *Gottfried Wilhelm Leibniz* (1646 bis 1716) in London an der Royal Society einen Vierspeziesrechner vor, der jedoch nicht voll funktionsfähig war. Nach diesen Frühentwicklungen wurde der eigentliche Bau von Rechenmaschinen aber erst einhundert Jahre später nach einem Modell von *Burkhardt* in einer größeren Serie begonnen.

Die Rechenmaschinen sind vielfach verbessert worden und arbeiten jetzt unter Zuhilfenahme von Elektromotoren „vollautomatisch". Ab 1965 werden für Tischrechenmaschinen auch elektronische Schaltelemente verwendet. Weitaus größere Rechengeschwindigkeit und bedeutend geringere Geräusche entwickeln diese Vierspeziesrechner mit und ohne Kontrollstreifendruck. Sie können teilweise auch potenzieren und Quadratwurzel ziehen. Zur besseren Nutzung sind mehrere Summationsregister (sogenannte Speicher) vorgesehen, in denen Zwischen- und Endsummen gebildet werden können. Die Arbeitsweise wird

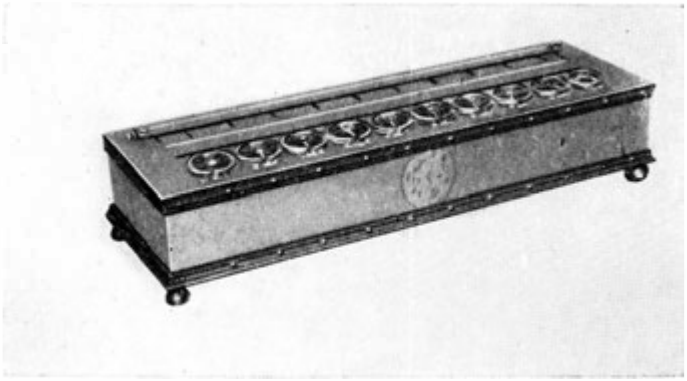


Bild 20. Rechenmaschine von *Pascal*

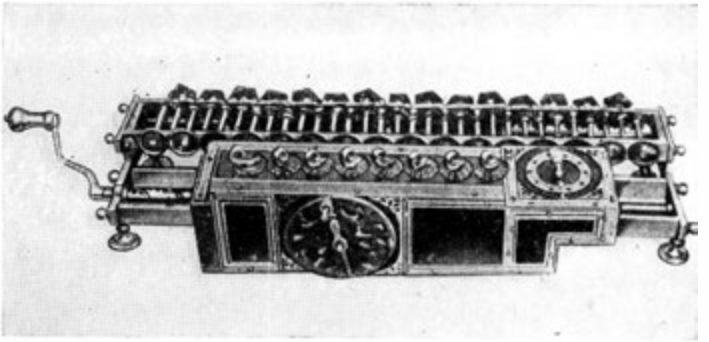


Bild 21. Rechenmaschine von *Leibniz*

Bild 22. Elektronischer Tischrechner Soemtron 220

ebenfalls als vollautomatisch bezeichnet. Bild 22 zeigt den elektronischen Tischrechner Soemtron 220 mit optischer Ergebnisanzeige, Bild 23 den Typ Soemtron 221 mit Kontrollstreifendrucker.

Der Ausdruck vollautomatisch ist jedoch im Sinne der maschinellen Rechentechnik irreführend. Zumindest muß er präzisiert werden. Jede Tischrechenmaschine kann lediglich die arithmetischen Rechenoperationen selber ausführen. Dem entspricht eine Mechanisierung. Von Automatisierung können wir aber erst reden, wenn der Gesamtkomplex einer Berechnung ohne Eingriff durch den Menschen abgearbeitet werden kann.

Bild 23. Elektronischer Tischrechner Soemtron 221

Dazu müssen alle Einzelprozesse nach einem vorher aufzustellenden Programm ablaufen. Man nennt die modernen Rechenautomaten daher auch „programmgesteuert", um das charakteristische Neue zum Ausdruck zu bringen. Ob dabei mechanische, elektromechanische, hydraulische oder elektronische Elemente für die technische Realisierung zur Anwendung kommen, ist von untergeordneter Bedeutung. Allerdings ermöglichen elektronische Elemente die schnellsten Schaltzeiten und werden daher bei modernen Rechenautomaten vorwiegend verwendet.
Für die Programmsteuerung muß der Gesamtprozeß einer Berechnung vorher in Einzelschritte zerlegt werden, die mit den Möglichkeiten des Rechenautomaten in Übereinstimmung zu bringen sind.
Dieser Vorgang – das Programmieren – wird vom Menschen ausgeführt. Ist das Programm aufgestellt, wird es in den Automaten eingegeben, der dann ohne Einwirkung des Menschen den Gesamtkomplex berechnet, wobei er selber nach einprogrammierten Testen Entscheidungen über den weiteren Rechenverlauf treffen kann.
Die Idee, einen programmgesteuerten Rechenautomaten zu konstruieren, hatte der englische Ingenieur *Charles Babbage* (1792 bis 1871). Seinen Überlegungen gingen Arbeiten der Franzosen *Prony* und *Jacquard* voraus. Prony hatte bei Berechnungen von Zahlentafeln das Prinzip des Programmierens angewandt. Er

gliederte den Gesamtprozeß der Bearbeitung auf. Mathematiker hatten die Berechnungsunterlagen (Rechenpläne) aufzustellen, was der Programmierung entsprach. Nach diesem Programm hatten dann Rechner als Hilfskräfte die einzelnen Rechnungen auszuführen. Damit konnte Prony einen Mangel an qualifizierten Mathematikern ausgleichen.

Jacquard hat unabhängig davon eine technische Lösung der Abarbeitung eines Programms für die Steuerung seiner automatischen Webstühle entwickelt. Er steuerte über gelochte Karten (Stahlplatten).

Babbage baute nach dieser Vorstellung 1822 eine Difference Engine (Differenzmaschine), bei der die Rechenoperationen von einer mechanischen Rechenmaschine ausgeführt wurden. Die Steuerung erfolgte über gestanzte Karten. Er arbeitete dann bis zum Ende seines Lebens an einer Analytical Engine – einem

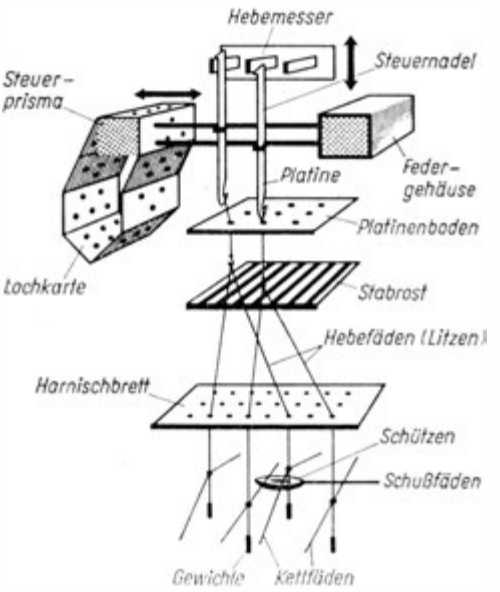


Bild 24. Programmsteuerung des Jacquardschen Webstuhles

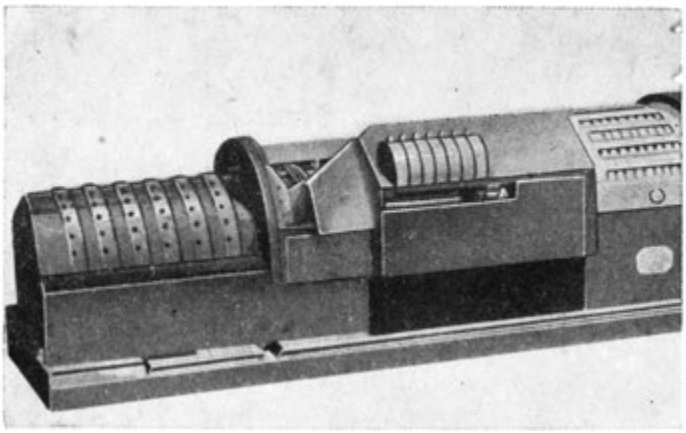


Bild 25. „Difference Engine" von *Babbage*

Rechenautomaten im modernen Sinne. Die technische Unmöglichkeit, ein derartiges Vorhaben allein mit mechanischen Elementen der damaligen Zeit auszuführen, ließ Babbage scheitern, und seine Ideen gerieten ein Jahrhundert in Vergessenheit.

1890 wurde in den USA die 11. Volkszählung durchgeführt. Sie ist deswegen von besonderer Bedeutung, weil die Auswertung der Ergebnisse erstmals maschinell erfolgte. Es kamen dabei *Hollerithmaschinen* zum Einsatz.

Hermann Hollerith (1860 bis 1929), ein Amerikaner, dessen Eltern aus Deutschland auswanderten, nutzte die von der Steuerung der Jacquardschen Webstühle bekannten Lochkarten dazu, die aus der Volkszählung erhaltenen Daten einzulochen, die dann elektromechanisch sortiert und tabelliert werden konnten.

Dies war die Geburtsstunde der *Lochkartentechnik*, die noch heute breite Anwendung findet. Mit *Lochkartenanlagen* kann man nicht nur große Zahlenmengen addieren, subtrahieren und unter Verwendung eines Zusatzgerätes, beispielsweise des in der DDR entwickelten Robotron ASM 18, multiplizieren, sondern auch eine wichtige logische Operation, das Sortieren, ausführen. Die

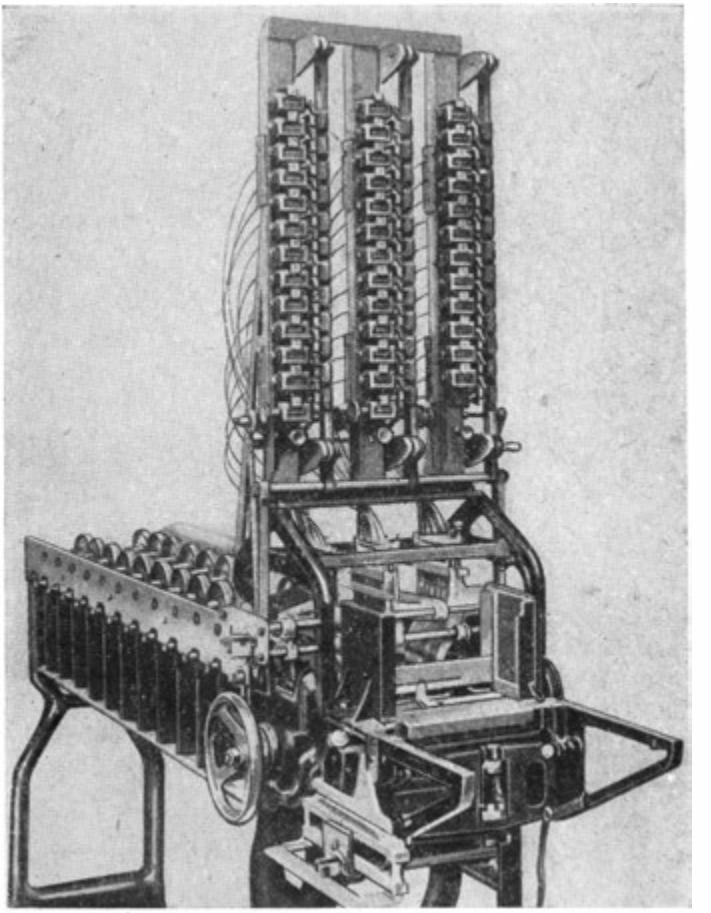


Bild 26. Eine der ersten elektromechanischen Sortiermaschinen nach *Hollerith*

Daten, auch Merkmale, werden nach einem Dezimalcode codiert. Dann können sie auf den Informationsträger, die Lochkarte, übertragen werden, indem an einer bestimmten Stelle einer

Spalte ein Loch gestanzt wird. Die Lage des Loches wird durch die Ziffer bestimmt (Bild 30). Für das Sortieren wird dann eine Spalte abgetastet und die Karte je nach Lage der Spaltenlochung in ein besonderes Fach transportiert.

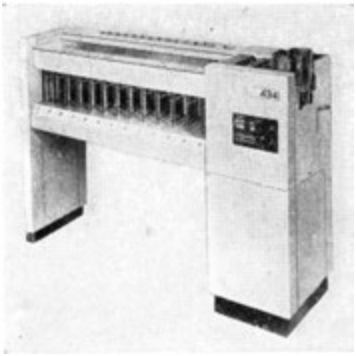


Bild 27. Sortiermaschine Soemtron 434

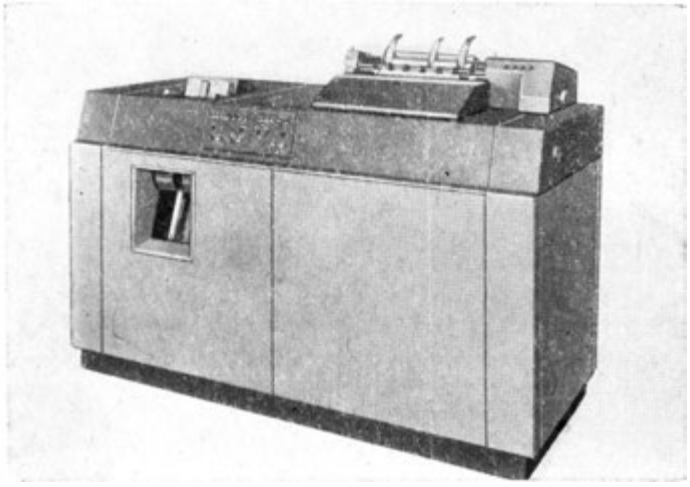


Bild 28a

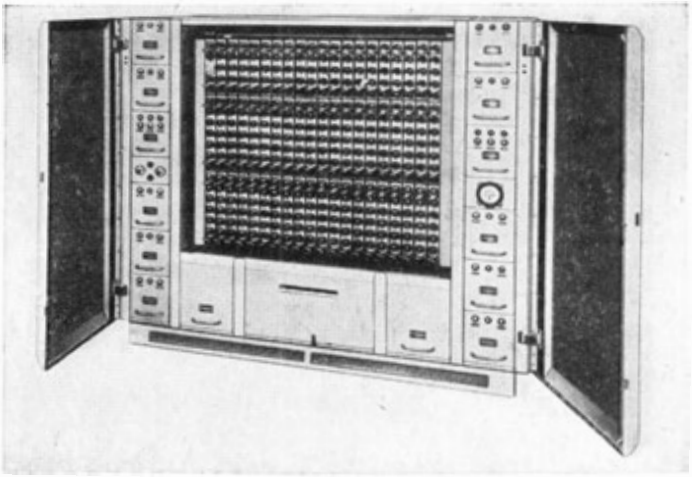


Bild 28b

Bild 28. Tabelliermaschine Soemtron 402 (a), die durch den Rechner Robotron ASM 18 (b) erweitert werden kann

Bild 29
Motorblocksummenlocher
Soemtron 441

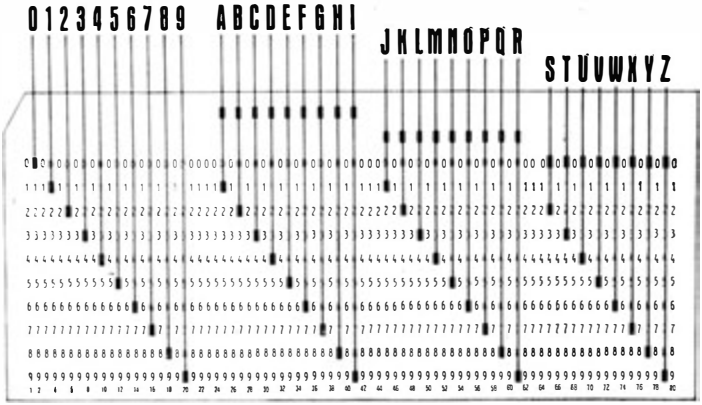


Bild 30. Lochkarte

Lochkartenanlagen ermöglichen eine Zeiteinsparung von etwa 30 %. Sie entsprechen vergleichsweise der Vollmechanisierung und eignen sich besonders für kommerzielle Rechnungen (Bilanzieren, Tabellieren, Lagerhaltungsrechnungen u. a.). Im Gesamtprozeß ihrer Arbeit müssen jedoch immer manuelle Arbeitsgänge eingeschaltet werden, die eine Fehleranfälligkeit und Verzögerung des Ablaufes bewirken. Daher werden Lochkartenanlagen durch moderne programmgesteuerte Datenverarbeitungsanlagen ersetzt, die alle Arbeitsprozesse automatisch mit hoher Geschwindigkeit ausführen können.

Anfang der vierziger Jahre unseres Jahrhunderts entwickelten nahezu gleichzeitig, aber unabhängig voneinander, der amerikanische Mathematiker und Physiker *Howard H. Aiken* und der deutsche Bauingenieur Dr.-Ing. h. c. *Konrad Zuse* je einen programmgesteuerten Rechenautomaten. Beide mit elektromechanischen Elementen – mit Fernsprechrelais. Zuse führte 1941 einem kleinen Kreis den ersten programmgesteuerten Rechenautomaten der Welt vor, der einwandfrei arbeitete. Es war dies sein dritter Entwurf, der **Z 3**. Wenige Monate später arbeitete auch der Relaisrechner von Aiken, der MARK I und kurze Zeit danach der MARK II.

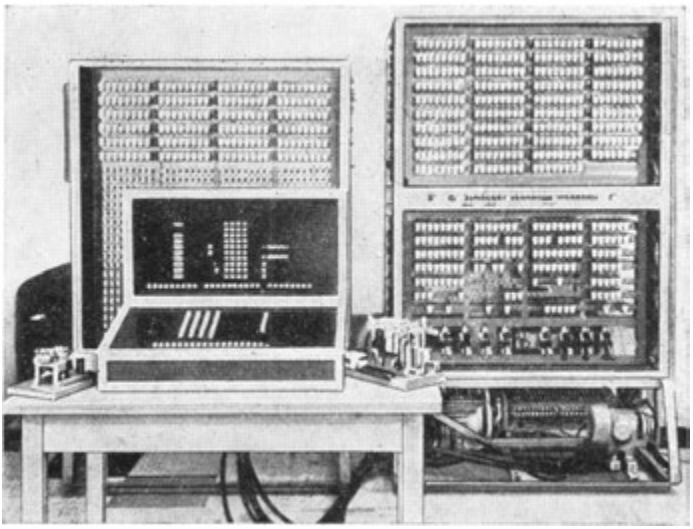


Bild 31 Z 3

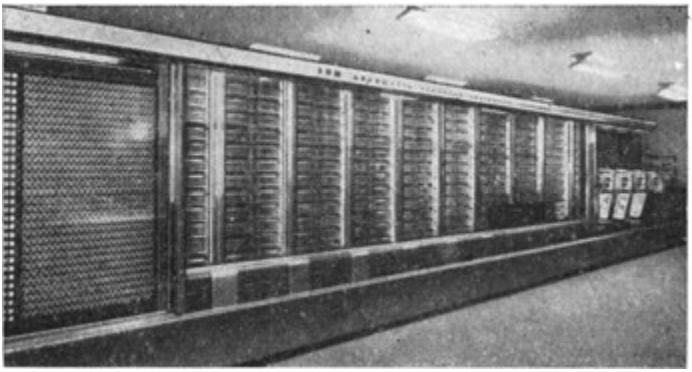


Bild 32. Rechenautomat MARK I von *Howard H. Aiken*

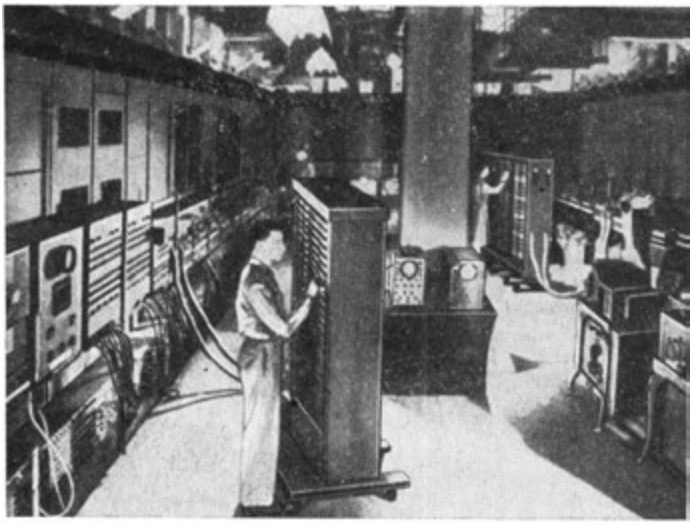


Bild 33. Erster elektronischer Rechenautomat ENIAC

1949 war dann an der Universität Pennsylvania (USA) der erste elektronische Rechenautomat, der ENIAC (Electronic Numerical Integrator and Calculator), fertiggestellt.
Damit begann eine stürmische Entwicklung. In der UdSSR wurden der schnelle Rechner BESM und der URAL I aufgebaut. England, Schweden, die Niederlande, Frankreich, Italien und andere Länder schufen eigene Modelle.
In der DDR setzten die Entwicklung und der Bau von Rechenautomaten etwas später ein.
Der erste arbeitsfähige Automat war der von Prof. Dr. *Kämmerer* und Prof. Dr. *Kortum* entwickelte Relaisrechner OPREMA (Optik-Rechen-Maschine) des VEB Carl Zeiss Jena. Er war 1955 fertiggestellt (Bild 34).
An der Technischen Universität Dresden wurden unter Leitung von Prof. Dr. *Lehmann* am Institut für Maschinelle Rechentechnik vier Geräte entwickelt:

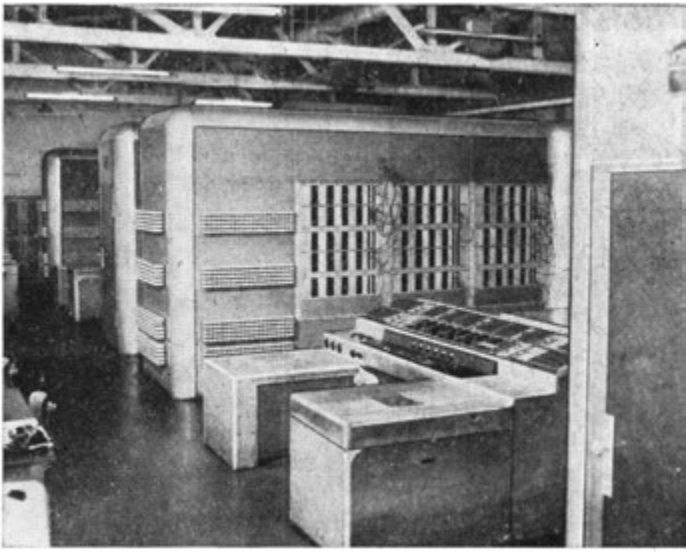


Bild 34. OPREMA

D 1 ein langsamer Digitalrechner mit 100 Op/s
D 2 ein mittelschnelles Gerät mit 1000 Op/s
D 3; D 4 ein digitaler Kleinrechner, der transistoriert als D 4a (D 4 abgerüstet) 1967 in Serienproduktion ging.

Das bewährte Kollektiv unter Prof. Dr. Kämmerer und Prof. Dr. Kortum schuf dann den Zeiss Rechenautomaten 1 (ZRA 1). Der ZRA 1 ist ein langsamer Digitalrechner mit 150 Op/s. Ab 1964 sind fast 30 Geräte dieses Typs in der DDR und in der ČSSR im Einsatz (Bild 37).

Vom VEB Elektronische Rechenmaschinen Karl-Marx-Stadt wurde ein Kleinstrechner entwickelt, der Cellatron SER 2, dessen Serienproduktion in der Firma Büromaschinen-Werke A. G., in Verwaltung, Zella-Mehlis, seit 1963 läuft (Bild 38).

In Karl-Marx-Stadt wurde auch die Konzeption des Robotron 100 und Robotron 300 erarbeitet (Bild 39). War der R 100 zur

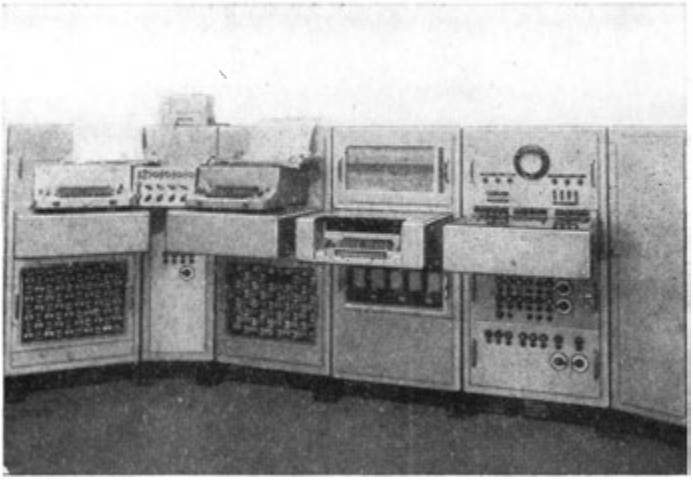


Bild 35. Bedienungspult des D 2

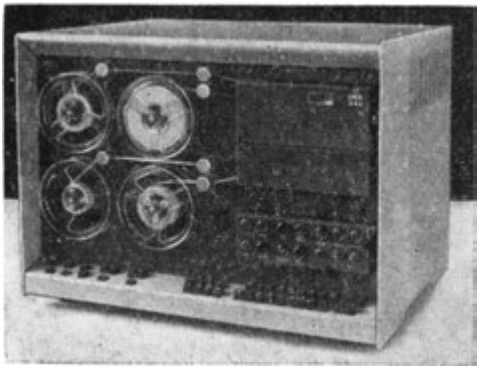


Bild 36. D 4a

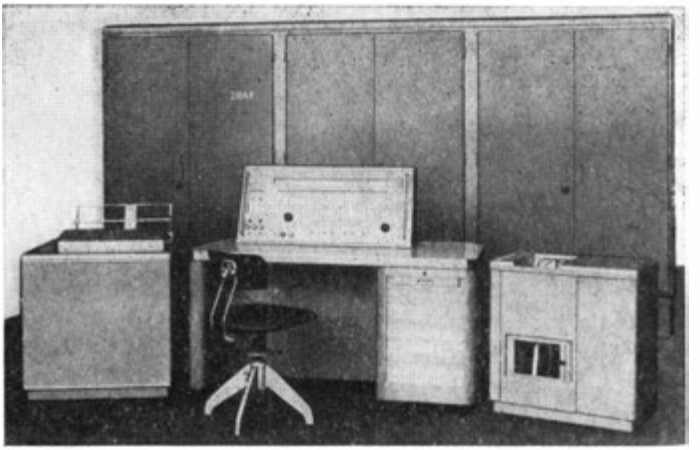


Bild 37. ZRA 1 vom VEB Carl Zeiss Jena

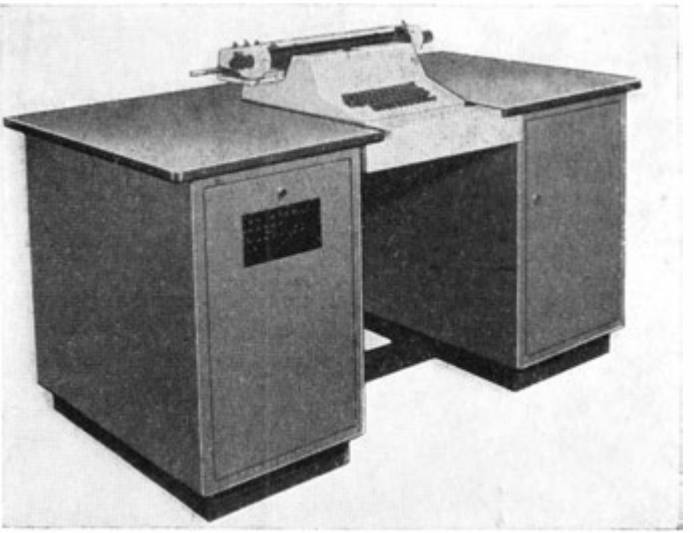


Bild 38. Cellatron SER 2b

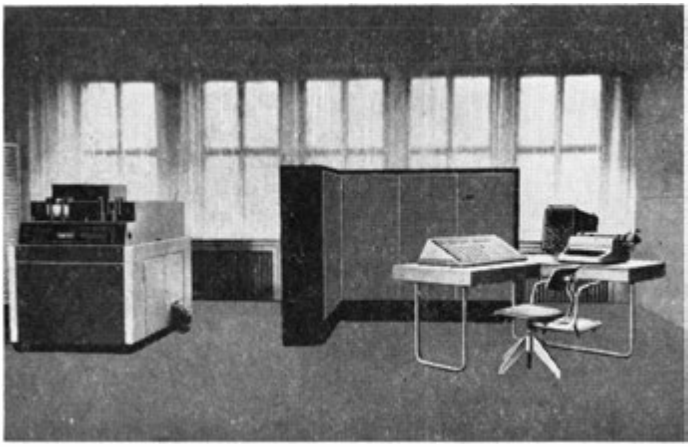


Bild 39. Robotron 100

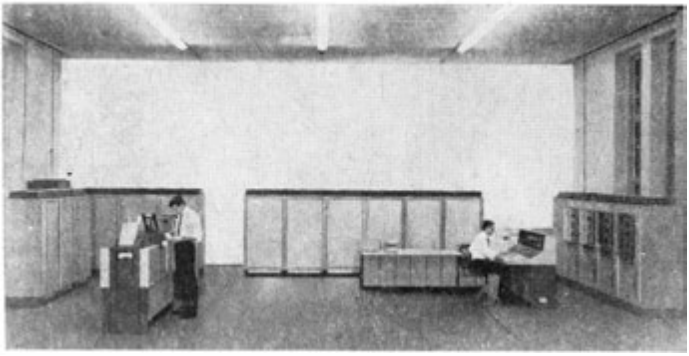


Bild 40. Robotron 300

Komplettierung von Lochkartenstationen gedacht, so ist der R 300 eine moderne mittlere Datenverarbeitungsanlage, die in Moskau auf der „Interorgtechnika 1966" der Welt vorgestellt wurde (Bild 40).

3.2. Allgemeine Bemerkungen

3.2.1. Schaltelemente und Schaltungstechnik

Erst rund zwei Jahrzehnte umfaßt die eigentliche Geschichte der modernen programmgesteuerten Rechenautomaten. Doch in dieser kurzen Zeit hat die Entwicklung und Produktion von Rechenautomaten einen enormen Aufschwung erlebt.

Die *Relaisrechner* (Z3, Z4, MARK I, OPREMA) werden als Vorstufe betrachtet. Elektronische Rechenautomaten unterscheidet man nach drei Generationen, wobei diese Einteilung vor allem bei der Betrachtung typischer Schaltelemente und der Schaltungstechnik sichtbar wird.

Die erste Generation ist durch die Verwendung von *Elektronenröhren* als Schaltelemente charakterisiert. Die Röhren mußten im Automaten durch Leitungen verbunden werden, deren Länge nach Kilometern gemessen werden konnte. Die in sechs- bis siebenstellige Dezimalzahlgröße reichende Zahl der Lötstellen verursachte zudem eine große Fehleranfälligkeit. Mit Röhren als Schaltelemente konnten Rechengeschwindigkeiten erreicht werden, die im Bereich der Millisekunden[1] liegen. Als „Nebenprodukt" erzeugten Rechenautomaten dieser Generation die Wärmeenergie eines kleinen Kraftwerkes, deren Ableitung großen technischen Aufwand erforderte.

Die zweite Generation wurde technisch durch die breite Verwendung von *Transistoren* als Schaltelemente eingeleitet. Zu ihnen gesellten sich Dioden und Ferritkernelemente. Die Schalttechnik ging dabei zu gedruckten Schaltungen mit wenigen Grundtypen auswechselbarer Schaltkarten über. Hierdurch kam es zu einer beträchtlichen Reduzierung des Volumens für den eigentlichen

---

[1] Zusammenstellung der Sekundendezimalgruppen

1 Sekunde	s	10^{0}	s = 1,0	s
1 Millisekunde	ms	10^{-3}	s = 0,001	s
1 Mikrosekunde	µs	10^{-6}	s = 0,000 001	s
1 Nanosekunde	ns	10^{-9}	s = 0,000 000 001	s
1 Picosekunde	ps	10^{-12}	s = 0,000 000 000 001	s
1 Femtosekunde	fs	10^{-15}	s = 0,000 000 000 000 001	s
1 Attosekunde	as	10^{-18}	s = 0,000 000 000 000 000 001	s

Rechner. Die Zuverlässigkeit der Automaten nahm zu, wobei gleichzeitig Rechengeschwindigkeiten im Bereich der Mikrosekunden erreicht werden konnten. Die Störanfälligkeit und damit verbundene Forderung an Klimatisierung und Staubarmut des Raumes wurde nicht mehr durch den Rechner bestimmt, sondern durch die Zusatzgeräte (periphere Geräte).
Programmgesteuerte Rechenautomaten *der dritten Generation* verwenden *Mikrobauelemente*, beispielsweise salzkerngroße Subminiaturtransistoren. Bild 41 bringt einen Größenvergleich der charakteristischen Schaltelemente in den drei Generationen. Entsprechend entwickelte man eine Mikroschalttechnik, bei der

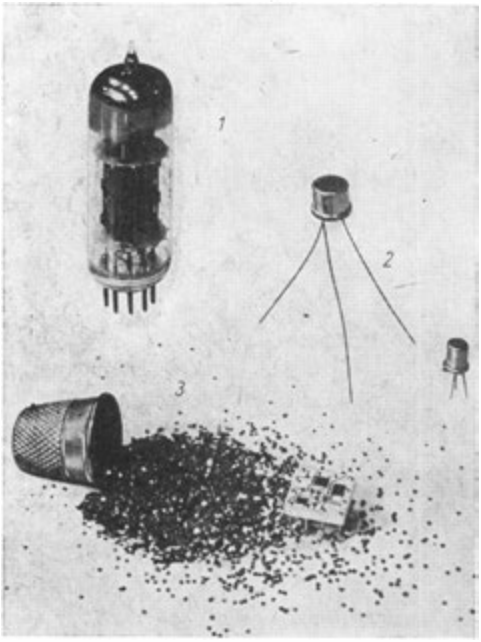


Bild 41. Größenvergleich der charakteristischen Schaltelemente von Rechenautomaten der drei Generationen: (1) Elektronenröhren (2) Transistoren (3) Subminiaturtransistoren

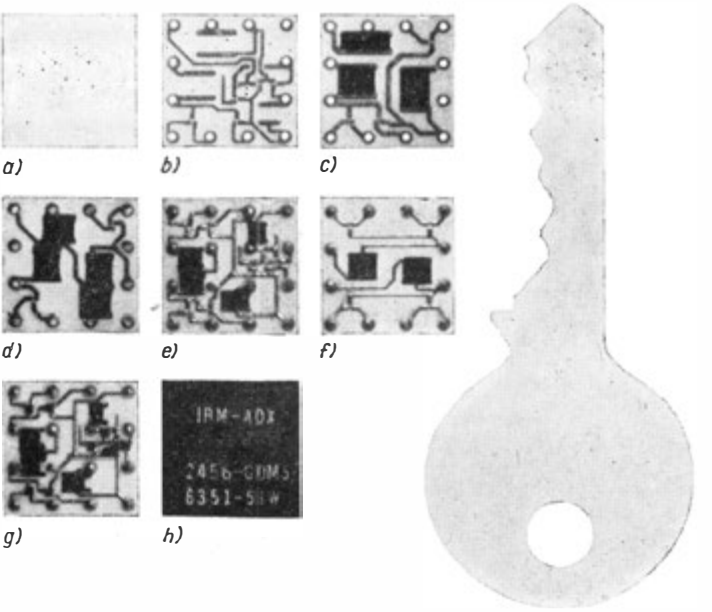


Bild 42. Entstehen eines Mikroschaltelementes
a) Keramische Grundplatte
b) Aufdrucken der Leitungen
c) Anbringen der Widerstände
d) Einsetzen der Stifte
e) Löten der Stifte und Leiter
f) Trimmen der Widerstände
g) Einfügen der Subminiaturtransistoren
h) Überziehen mit einer Schutzschicht und Kennzeichnen

z. B. 12 Mikroschaltungen auf einer Trägerplatte von 1 cm² untergebracht werden können. Den Aufbau und die Herstellung einer Mikroschaltung zeigt Bild 42. Die Mikroschaltungen werden in einer Hülle fest eingeschlossen (Kompaktbaustein) und so störungs- und wartungsfrei zum Einbau in den Automaten vorgelegt. Mit diesem Element lassen sich Rechengeschwindig-

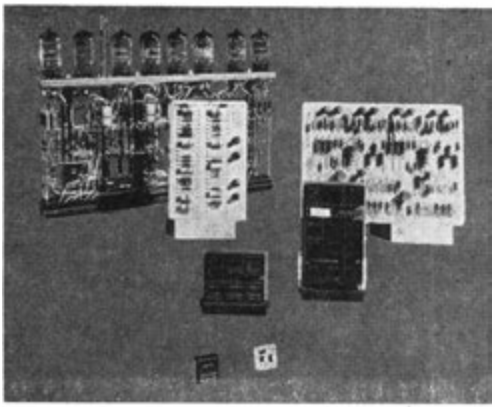


Bild 43. Vergleich der Schalttechnik in den drei Generationen

keiten im Bereich der Nanosekunden erreichen. Gleichzeitig verringern sich das Volumen des Rechners und seine Störanfälligkeit beträchtlich.

Es zeichnet sich bereits eine weitere Verkleinerung der Schaltelemente ab. In der *Monolithtechnik*, auch *Molekularelektronik* genannt, werden die Funktionen bisheriger Schaltelemente durch Zonen eines einkristallinen (monolithischen) Siliziumscheibchens übernommen. Diese Schaltungen werden durch mehrfache Bedampfungs-, Ätz- und Oxydationsverfahren, bei denen Präzisionsschablonen Verwendung finden, hergestellt. Alle Anschlüsse sind mit goldplattierten Anschlußdrähten ver-

Bild 44. Monolithschaltelement im Größenvergleich mit einer Fingerkuppe

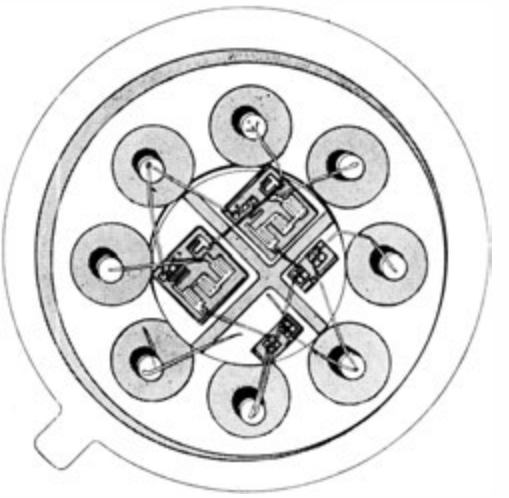


Bild 45. Integrierte Schaltungen der Monolithtechnik

sehen und die Schaltungen in Schutzpackungen hermetisch versiegelt. In der Monolithtechnik können wir 1000 komplette herkömmliche Schaltungen auf einer dünnen Siliziumscheibe von der Größe eines 1-Mark-Stückes unterbringen.
In der Tabelle 2 sind die Packungsdichten, bezogen auf Bit/cm³, zusammengestellt, wobei das menschliche Gehirn als Vergleich mit angeführt ist.

Verwendete Technik	Packungsdichte (Bit/cm³)
Normale Elektronenröhren	0,05
Transistoren	1
Mikroschalttechnik	20
Monolithtechnik	10000
Menschliches Gehirn	1000000000000

Tabelle 2. Packungsdichte von Schaltelementen

Die Zeit ist nicht mehr fern, in der uns digitale Hochleistungsrechner in der Größe einer Zigarrenkiste zur Verfügung stehen werden.

3.2.2. Prinzip der Arbeitsweise

Neben der Betrachtung der Entwicklung von Schaltelementen und Schalttechniken in den drei Generationen gibt es eine Reihe weiterer interessanter Aspekte, die Aufschluß über die Fortschritte in der Rechentechnik geben.
Zuvor müssen wir uns jedoch die prinzipielle Arbeitsweise eines programmgesteuerten Rechenautomaten ansehen.
Wir können die Arbeitsweise eines programmgesteuerten digitalen Rechenautomaten noch in vielen Phasen mit der eines menschlichen Rechners vergleichen.
Nehmen wir an, es soll mit Hilfe einer Tischrechenmaschine, Papier und Bleistift eine Berechnung ausgeführt werden.
Die Arbeitsweise ist dann meist in Form einer übersichtlichen Zusammenstellung der Ausgangswerte oder Ausgangsdaten, gewisser Leerfelder für Zwischen- und Endergebnisse sowie Anmerkungen und Vorschriften auf einem Programmformular fixiert.
Dann vollzieht sich der technologische Arbeitsverlauf wie folgt: Der Rechner sieht die Ausgangsdaten als Zahlen und überträgt sie in die Tischrechenmaschine, indem er die jeder Ziffer entsprechende Taste drückt. Dann löst er nach Vorschrift die Rechenoperation (oder Operationen) aus. Wichtige Zwischenergebnisse überträgt er wiederum als Zahlen in das Formular. Teilweise müssen auch Zwischenergebnisse notiert werden, die nur im Rechenablauf von Interesse sind. Sie werden lediglich auf Notizblättern vorgemerkt, um sie später wieder zu verwenden. Meist muß der Rechner Entscheidungen über den weiteren Rechenverlauf fällen, wenn Zwischenergebnisse verschiedene Rechenwege verlangen (z. B. der Wert der Diskriminante bei quadratischen Gleichungen) und vieles mehr. Ist der Gesamtprozeß richtig aufgestellt, ermittelt der Rechner schließlich das Endergebnis, das er ebenfalls in das Programmformular einträgt. Nach Ausführung einer Kontrollrechnung kann er dann seine Arbeit beenden.

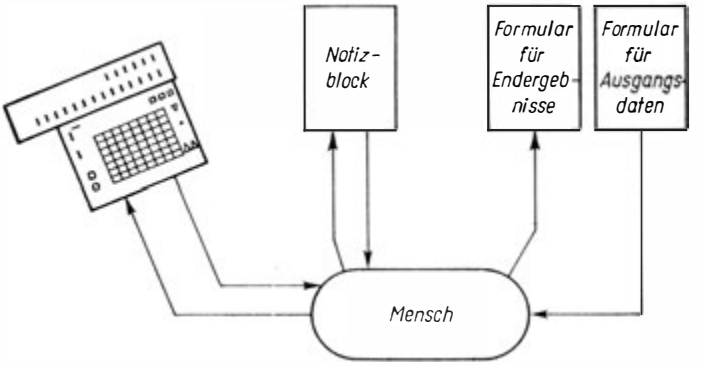


Bild 46. Arbeitsablauf beim Rechenprozeß mit Tischrechenmaschinen

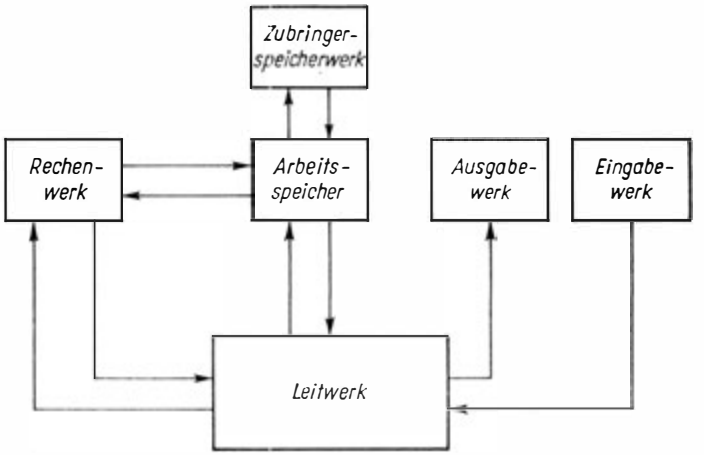


Bild 47. Skizze der wichtigsten Baugruppen eines programmgesteuerten Digitalrechners

Diesem Arbeitsablauf ist der eines programmgesteuerten Rechenautomaten sehr ähnlich. Der Tischrechenmaschine entspricht im Automaten das Rechenwerk, das Programmformular und der Notizblock dem Speicherwerk und der menschliche Rechner schließlich dem Leitwerk. Allerdings müssen wir beim Automaten noch ein Eingabe- und Ausgabewerk einbauen, deren Funktionen der Mensch mit übernommen hat.
Technologisch geben wir die Ausgangsdaten und die Vorschriften für den Rechenverlauf, das Rechenprogramm, in das Eingabewerk. Dieses übersetzt sie in die Sprache der Automaten (elektronische Impulse) und leitet sie in das Speicherwerk, das den Formularen, Notizblöcken und dem Gedächtnis des Menschen entspricht. Hier werden alle Informationen bereitgestellt und nach Vorschrift abgerufen. Das gilt auch für Zwischen- und Endergebnisse, die ebenfalls im Speicherwerk festgehalten werden.
Das Rechenwerk verarbeitet dann die Daten, während das Kommando-, Steuer- oder Leitwerk den Gesamtablauf des Rechenprozesses nach dem Programm lenkt.
Dabei können Verzweigungen im Programm auftreten, bei denen der Automat nach Ermitteln einer Kontrollgröße selber entscheidet, wie weitergerechnet werden muß.
Das Ausgabewerk übernimmt schließlich die Endergebnisse, transformiert sie zurück in Zahlen und gibt sie lesbar aus. Bild 47 skizziert diesen Vorgang, der allerdings der Übersichtlichkeit halber stark vereinfacht ist.

3.2.3. Einsatzgebiete

Alle Rechenautomaten, gleich, auf welchem Gebiet sie eingesetzt werden, sind Maschinen zur Informationsverarbeitung. Doch lassen sich hierbei aus moderner kybernetischer Sicht drei Einsatzsphären unterscheiden.
In ihrer ersten Konzeption waren programmgesteuerte Rechenautomaten als Hilfsmittel für die Wissenschaft und Technik gedacht. Sie sollten die umfangreichen Berechnungen für Probleme der Atom- und Kernphysik, für neue gewaltige technische Konstruktionen, für das Bauwesen u. a. abnehmen. So finden wir zuerst Rechner für sogenannte *wissenschaftlich-technische Berech-*

nungen. Wir könnten diese Aufgaben unter der Sicht der Informationsverarbeitung wie folgt charakterisieren:
Es werden einzelne Datensätze mit Hilfe komplizierter mathematischer Disziplinen und deren Algorithmen zur Ermittlung neuer Informationen für denselben Bereich verarbeitet. Entsprechend müssen für diese Rechnungen Rechen- und Leitwerk stark ausgebaut sein, um möglichst flexibel verwendet werden zu können. An die Ein- und Ausgabe und an die Speicherkapazität werden weniger hohe Anforderungen gestellt. Ein klassischer Vertreter dieser Rechenautomatengruppe ist unser ZRA 1.
Parallel zum Einsatz programmgesteuerter Rechenautomaten für sogenannte wissenschaftlich-technische Berechnungen wurde und wird die klassische -- das soll heißen, auf Hollerith zurückführende – Lochkartentechnik verwendet. Lochkartenstationen sind mit mehreren Maschinensätzen aus

Tabelliermaschinen einschließlich Multiplikationszusatz Robotron ASM 18 (s. Bild 28) zum Summieren, Bilden von Zwischen- und Gruppensummen und Tabellieren,
Sortiermaschinen zum Sortieren der Daten (s. Bild 27),
Motorblocksummenlocher zur Ausgabe der Informationen als gelochte Karten (s. Bild 29),
Kartendopplern,
Magnetlochern und Magnetprüfern

u. a. ausgerüstet. Mit ihrer Hilfe können Rechnungen, wie

Brutto- und Nettolohnrechnungen,
Betriebsabrechnungen,
Lagerhaltungsrechnungen,

vollmechanisiert, aber mit manuellen Zwischenschritten ausgeführt werden. Bald zeigte sich jedoch, daß programmgesteuerte Rechenautomaten den Lochkartenanlagen weit überlegen waren. Ist es mit Lochkartenanlagen möglich, den manuellen Anteil der von ihnen ausgeführten Arbeiten um 30 % zu senken, so kann bei Verwendung von Rechenautomaten 90 % Einsparung erreicht werden. Allerdings benötigt man dazu enorm große Speicher und eine extrem schnelle Ein- und Ausgabe, während an die Flexibilität des Leit- und Rechenwerkes keine so großen Anforderungen gestellt werden.

Hieraus entstanden die Datenverarbeitungsanlagen. Das Gebiet der *Datenverarbeitung* ging dabei unter Nutzung der größeren Möglichkeiten über die klassische Datenverarbeitung hinaus. Nicht nur, daß man die manuellen Zwischenschritte automatisierte, es wurden auch völlig neue Möglichkeiten durch Variantenberechnungen, Planungsrechnung und Optimierung erschlossen.
So können wir die moderne Datenverarbeitung als Disziplin der Informationsverarbeitung wie folgt umreißen:
Es werden Folgen von Informationssätzen verarbeitet, um auch Merkmale mehrerer Bereiche vergleichen und zur Bildung allgemeinerer Schlüsse heranziehen zu können.
Moderne Rechenautomaten sind durch Entwicklung leistungsfähiger Rechner und Zusatzgeräte so ausgerüstet, daß sie beide Aufgabengruppen bearbeiten können. Dabei war und ist man auf dem internationalen Markt bestrebt, ein umfassendes Sortiment von Rechnern anzubieten, um sich den unterschiedlichsten Forderungen der Praxis anpassen zu können. Man unterscheidet:

digitale Kleinrechner,
langsame Digitalrechner,
mittelschnelle Digitalrechner und
Hochleistungsrechner.

Diese Unterteilung erfolgte vorwiegend nach der Rechengeschwindigkeit und mußte nach der technischen Entwicklung mehrfach verändert werden.
In den letzten Jahren hat sich erneut ein Wandel vollzogen. Praktisch stehen nur noch *Kleinrechner* und *Datenverarbeitungssysteme* oder *-familien* im Angebot.
Die Kleinrechner werden in Ingenieurbüros und kleineren Betrieben eingesetzt. Sie sind vorwiegend für wissenschaftlich-technische Berechnungen gedacht. Allerdings sind sie technisch soweit entwickelt, daß sie auch Aufgaben der Datenverarbeitung bewältigen können. Das heißt, die Rechner sind in ihrer Grundausstattung „für sich" arbeitsfähig, haben also Ein- und Ausgabe, Arbeitsspeicher und Leitwerk. Darüber hinaus haben sie aber auch Anschlußkanäle für den Einsatz peripherer Aggregate, wie zusätzlicher Speicher, leistungsfähigerer Ein- und Ausgabegeräte.

Ihre Leistungsfähigkeit ist bei Verringerung des Volumens so gesteigert, daß sie den zuvor erwähnten langsamen und gar mittelschnellen Digitalrechnern gleichgesetzt werden könnten. Zur Charakterisierung sind in Tabelle 3 einige Leistungsdaten auserwählter Automatentypen angeführt.

	D 4a	LGP 21	Friden 6010	
Zahlendarstellung (Konstante Wortlänge)				
dezimal	9	9	16	Stellen
dual	33	32	—	Bit
Rechengeschwindigkeit Add./Subtr. (Festkomma)	1,4	0,3...5,7	1,3	ms
Multipl. (Festkomma)	15	20	50	ms
Division (Festkomma)	20	—	—	ms
Mittl. Zugriffszeit	1,6	20		ms
Speicherkapazität				
Kernspeicher	—	—	20	Wörter
Magnettrommel	4096	4096	—	Zellen

Tabelle 3. Leistungsdaten moderner digitaler Kleinrechner

Digitale Kleinrechner haben sich im praktischen Einsatz bereits als rentabel erwiesen, wenn in ihrem Einsatzbereich mindestens 20 Mitarbeiter theoretisch arbeiten. Ihr Unterhalt ist jedoch relativ teuer. Zudem ist ihr Leistungsvermögen nach oben begrenzt.

Über diese Grenze hinausgehender Bedarf kann durch Datenverarbeitungssysteme oder -familien gedeckt werden.

Das modernste Gebiet ist der Einsatz programmgesteuerter Rechenautomaten – sowohl Rechner für wissenschaftlich-technische Aufgaben als auch Datenverarbeitungsanlagen – als *Prozeßrechner*.

Hier handelt es sich um ein geschlossenes kybernetisches System,

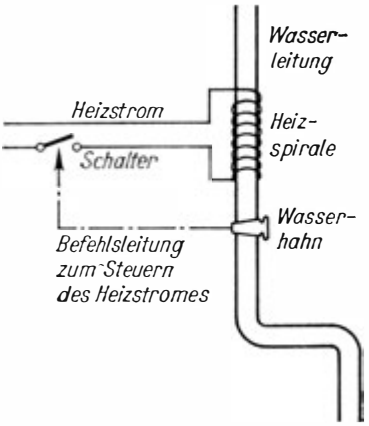


Bild 48. Prinzipskizze eines Durchlauferhitzers als Steuervorgang

und der Regler arbeitet im Echtzeitbetrieb (Real-time-processing[1]).

Klären wir hierzu erst die Begriffe Steuern und Regeln. Sehr einfach erkennen wir den Unterschied am Durchlauferhitzer und Heißwasserspeicher.

Der Durchlauferhitzer demonstriert das „Steuern" eines Prozesses (Bild 48).

Durch das Öffnen des Wasserhahnes wird über die Befehlsleitung der Heizstromkreis geschlossen, wodurch die Spirale das durchlaufende Wasser erwärmt. Schließen wir den Hahn, so wird auch der Stromkreis wieder unterbrochen und das Wasser entsprechend nicht mehr erwärmt. Die Steuerung erfolgt also „von außen" durch die Betätigung des Wasserhahnes.

Der Steuerungsvorgang wirkt von außen auf das System des betrachteten Prozesses.

Anders hingegen beim Heißwasserspeicher. Ein Sinken der Temperatur, die am Temperaturfühler, dem Meßwerterfassungs-

[1] Real-time-processing: Echtzeitverarbeitung, Sofortverarbeitung der Ausgangsdaten.
Fachbegriffe der Rechentechnik sind vielfach vom Englischen her in andere Sprachen eingedrungen

gerät, ermittelt wird, bewirkt über den Regler ein Schließen des Heizstromkreises, bis eine gewünschte Maximaltemperatur erreicht ist. Dabei ist es völlig gleichgültig, ob das Sinken der Temperatur durch Wasserentnahme oder Abkühlen infolge längeren Stehens eintrat.

Der Einsatz von Rechenautomaten zur Prozeßsteuerung (process control) ist sehr vielfältig. Wir finden sie bei Regelung von Produktionsprozessen, Verkehrsanlagen (Kreuzungen u. a.), Flugleiteinrichtungen, militärischen Komplexen usw. Die Regelung erfolgt dabei unter Berücksichtigung und Einbeziehung prozeßexterner Tatbestände.

Prozeßrechner werden nach der Komplexität und dem Grad der Verbundenheit mit den anderen Teilen des Systems unterteilt.

Nach der Komplexität unterscheidet man vier Stufen:

Erfassen kleiner Teilprozesse nach fest vorgegebenem Programm (Betriebsrechner zur Steuerung von Werkzeugmaschinen);

Zusammenwirken mit herkömmlichen Regelsystemen. Der Rechner tritt meist als Regler auf; wobei das Programm teil-

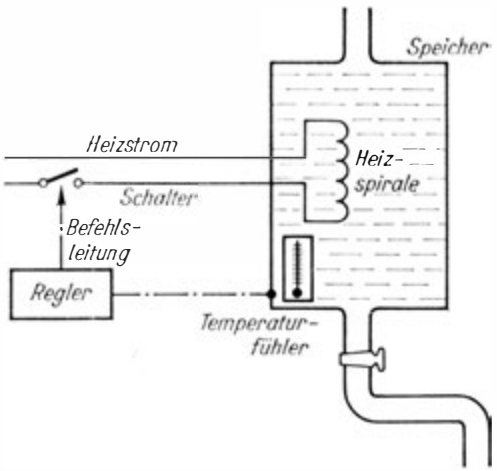


Bild 49. Prinzipskizze eines Heißwasserspeichers als Regelvorgang

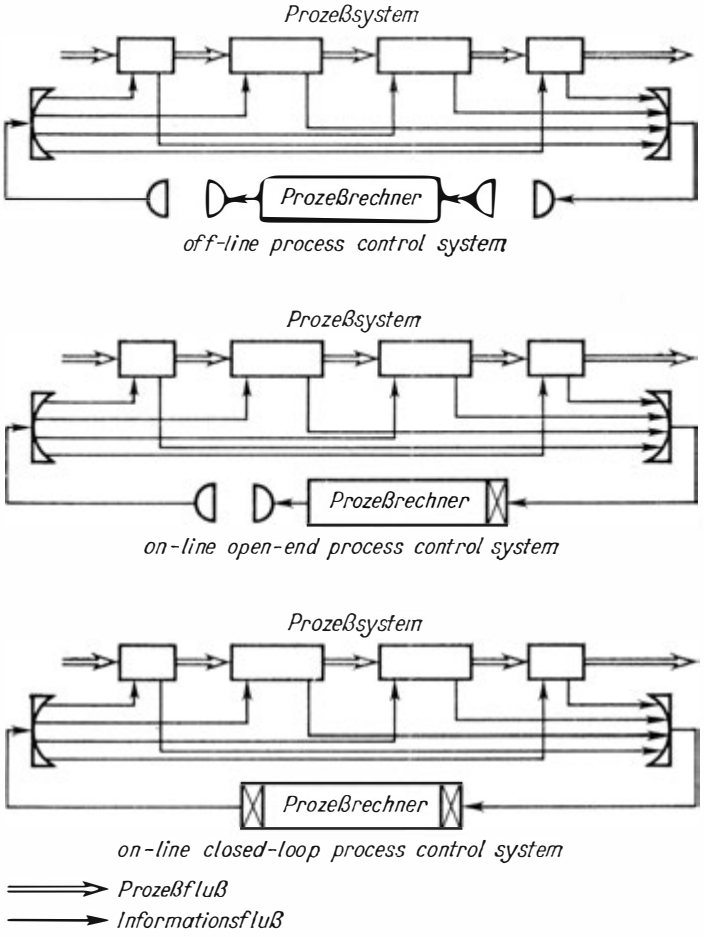


Bild 50. Klassifizierung der Prozeßsteuerung nach dem Verbundenheitsgrad

weise selbständig aufgestellt und geändert wird (multistabile Systeme);
Steuerung mehrerer Anlagen, beispielsweise einer kompletten Taktstraße. Dies stellt bereits hohe Anforderungen an den Rechner;
Regelung komplexer Prozesse unter Berücksichtigung prozeßexterner Tatbestände (Warenhaus unter Einbeziehung der Marktforschung, Bankensystem, Fluggesellschaften u. a.).
Bezüglich des Grades der Verbundenheit können wir folgende Systeme unterscheiden (Bild 50):
offene Prozeßsteuerungssysteme
(off-line process control system).
Hier ist der Rechner nicht unmittelbar mit den sonstigen Elementen des Systems verbunden;
halboffene Prozeßsteuerungssysteme
(on-line open-end process control system).
Nunmehr gehen die Meßwerte direkt als Eingabewerte in den Rechner. Die Ausgabe ist jedoch nicht mit dem Prozeß verbunden;
geschlossene Prozeßsteuerungssysteme
(on-line closed-loop process control system).
Jetzt sind Ein- und Ausgabe des Rechners direkt (on-line) mit den anderen Geräten verbunden, und es ergibt sich ein geschlossener Kreis (closed-loop). Dies ist die höchstentwickelte Form der Prozeßsteuerung. Sie entspricht einem geschlossenen Regelkreis.
Die Prozeßsteuerung verursacht einige zusätzliche technische Probleme, wie automatische Meßwerterfassung (date logging), die Umwandlung von gegebenen Analogwerten in Digitaldaten u. a.

3.2.4. Datenträger

Weitere interessante Aspekte ergeben sich aus der Betrachtung der Maschinenausrüstung (hardware: eigentlich harte Ware im Gegensatz zur Programmausrüstung, der software: weichen Ware) und der Entwicklung der Programmierungstechnik.
Die Informationen können uns in vielfältigster Form für die Eingabe in den Automaten vorgelegt werden. Lassen sie sich

unmittelbar vom vorliegenden Beleg in den Automaten eingeben, so liegen sie auf sogenannten Datenträgern vor.
Die wichtigsten Datenträger sind

Lochkarten,
Lochbänder,
Magnetbänder (Magnetplatten, Magnetkarten),
Magnetschriftbelege,
Klartextbelege.

Der klassische Datenträger ist die *Lochkarte.* Sie wurde bereits im vorigen Jahrhundert durch *Hermann Hollerith* eingeführt. Inzwischen ist sie vielfach verändert worden, jedoch hat sich die Grundform in der 80spaltigen Lochkarte am stärksten durchgesetzt. Wir wollen uns daher auf ihre Betrachtung beschränken. Die 80spaltige Lochkarte hat folgende Maße:

187,3 mm Länge,
82,5 mm Höhe,
0,178 mm Dicke.

Auf ihr sind 80 Spalten in 12 Zeilen angeordnet. Die Zeilen werden in Normallochzone (untere 10 Zeilen) und Überlochzone (obere 2 Zeilen) geteilt.
Die numerischen Werte können entsprechend dem „Eins-aus-zehn"-Code gelocht werden, indem jeweils für das duale L eine rechteckige Lochung in einer Zeile vorgenommen wird (s. Bild 30). In jeder Lochspalte kann also nur ein Zeichen gelocht werden. Wollen wir alpha-numerische Zeichen lochen, müssen wir die beiden Zeilen der Überlochzone hinzuziehen, wie auf Bild 30 dargestellt ist.
Das dort gezeigte Lochungsschema entspricht dem IBM-Code[1], der im folgenden Schema für die alpha-numerische Lochung zusammengefaßt werden kann:

[1] IBM-Code: Lochkartencode der amerikanischen Firma International Business Machines Corporation. Dies ist die führende Firma des westlichen Automatenmarktes. Sie hat 1965 80 % des Marktes beliefert

	0	1	2	3	4	5	6	7	8	9
12		A	B	C	D	E	F	G	H	I
11		J	K	L	M	N	O	P	Q	R
0			S	T	U	V	W	X	Y	Z

Tabelle 4. IBM-Code

Neben dem IBM-Code gibt es noch andere Codes, auf die wir nicht eingehen wollen.
Eine interessante Variante wollen wir jedoch noch betrachten. Es gibt neuere Lochkartensysteme, bei denen die Lochung entfallen kann. Es werden lediglich die Felder der Lochung vorgedruckt und der Wert mit einen Spezialgriffel angekreuzt. Die Arbeitsgeräte „lesen" dann diese Information als Lochung.
Lochkarten haben den großen Vorteil, daß nur eine begrenzte Informationsmenge auf einer Karte enthalten ist. Bei Fehllochungen oder veralteter Information kann die einzelne Karte schnell korrigiert werden. Besonders geeignet ist die Lochkarte für den Sortier- und Mischprozeß, zwei Vorgänge, die in der Datenverarbeitung häufig vorkommen. Die Information kann zudem auch in der gelochten Form leicht und schnell manuell gelesen werden.
Der wesentlichste Nachteil der Lochkarte ist ihre relativ große Masse. Das Einlesen und Ausgeben von Informationen aus oder in die Lochkarte ist sehr zeitaufwendig.
In den letzten Jahren setzt sich der *Lochstreifen* in zunehmendem Maße durch. Papierstreifen unterschiedlicher Länge nehmen die Information als Daten durch Lochkombinationen von 5 bis 8 Löchern in der Zeile auf. Daher spricht man auch von 5-, 6-, 7- und 8-Kanal-Lochstreifen. Die Lochung wird nach einem speziellen Code mit einer Transportlochung vorgenommen. Als Codes eignen sich beispielsweise die in 1.3. beschriebenen. Insbesondere gilt für den Cellatron SER der auf Bild 51 dargestellte Code.
Lochstreifen haben gegenüber der Lochkarte eine weitaus geringere Masse und damit eine größere Ein- und Ausgabegeschwindigkeit. Die Informationen sind auch noch manuell lesbar.

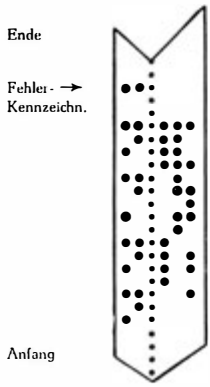


Bild 51. 5-Kanal-Code des Cellatron SER

$1 = 2^0$ $4 = 2^2$
$2 = 2^1$ $8 = 2^3$

Pr Prüflochung (Anzahl der Lochungen muß ungerade sein)
Gelocht wird das interne Wort (Bilder 102 und 103) mit Adressenangabe. Jede Zeile entspricht einer Dualtetrade.
Somit ergeben sich 12 Zeilen für das Wort, 2 Zeilen für die Adresse des Wortes

1	2	●	4	8	Pr	=
		·			●	0
●		·				1
	●	·				2
●	●	·			●	3
		·	●			4
●		·	●		●	5
	●	·	●		●	6
●	●	·	●			7
		·		●		8
●		·		●	●	9
	●	·		●	●	10=P2
●	●	·		●		11=P3
		·	●	●	●	12=P4
●		·	●	●		13=P5
	●	·	●	●		14=P6
●	●	·	●	●	●	15=P7
●	●	· Fehler (stop)				

Nachteilig ist jedoch, daß keine Sortierung vorgenommen werden kann und daß eine nachträgliche Korrektur außerordentlich schwer, praktisch nicht möglich ist.
Immer stärker findet auch das *Magnetband* als Datenträger Verwendung. Es ist den Lochkarten und dem Lochstreifen überlegen, da es auf Grund seiner größeren Informationsdichte eine auch den schnellsten Rechenautomaten entsprechende Ein- und Ausgabegeschwindigkeit erreichen kann. Eine nachträgliche

Korrektur und Aktualisierung der Daten ist leicht möglich, da Magnetbänder gelöscht und beliebig oft wieder beschrieben werden können.

Die Informationen sind beim Magnetband nicht als Lochkombinationen, sondern in Form unterschiedlich magnetisierter Bereiche auf der Bandoberfläche dargestellt, wie wir es vom Tonbandgerät und dessen Schallkonservierung kennen.

Einen Nachteil hat jedoch das Magnetband. Die gespeicherten Informationen können manuell nur mit Hilfe eines komplizierten Gerätes gelesen werden. Das erschwert uns die Kontrolle der Informationen.

Gleiches gilt für die *Magnetkarten* und *Magnetplatten*, die jedoch stärker als Speicher benutzt werden. Beide Datenträger sollen daher bei den Externspeichern Erwähnung finden.

Alle bisher genannten Datenträger haben den Nachteil, daß die Informationen erst aus den Originalbelegen auf den Datenträger übertragen werden müssen. Die Forschungsarbeiten orientieren sich darauf, die Originalbelege in einer Form anzufertigen, daß ihre Informationen unmittelbar in den Automaten eingegeben werden können. Der Originalbeleg soll also zum Datenträger werden. Praktisch haben sich hier zwei Verfahren bewährt, für die NCR[1] auch die erforderlichen Geräte entwickelt hat (Leser, Sortiermaschinen und Drucker).

Das erste Verfahren beruht auf dem Drucken spezieller Typen mit einem Farbstoff, der einen Magnetzusatz enthält. Es gibt verschiedene Drucktypen und damit Schriftsorten. Für die *Magnetschriften* E 13 B und CMC 7 (Bild 52) sind Lese-, Sortier- und Druckgeräte entwickelt. Damit können Belege, die eine Magnetschriftinformation (natürlich an vorgegebener Stelle) enthalten, sowohl manuell als auch direkt in den Automaten gelesen werden.

NCR bietet auch bereits Geräte für eine Klartextauswertung an, allerdings nur für rein numerische Arbeitsweise mit speziellen auf Bild 53 dargestellten Ziffern.

Die eigenartig erscheinende Darstellungsform der Ziffern ergibt

---

[1] NCR: The National Cash Register Co.
(Nationale Registrierkassen GmbH, eine internationale Firma mit 1039 Zweigstellen in 121 Ländern, Sitz USA)

Bild 52. Magnetschriften E 13 B und CMC 7

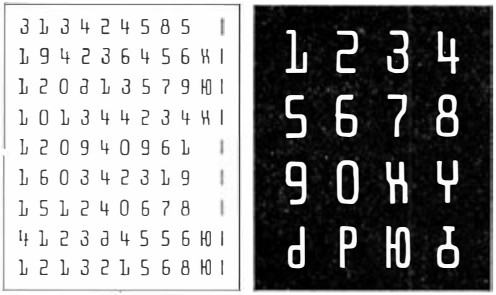


Bild 53. Ziffern für optisch lesbare Informationen

sich aus der Arbeitsweise. Alle Ziffern und Zeichen werden an 10 Stellen abgetastet (Bild 54). Nach der Belegung dieser Felder durch Stellen der Ziffern werden die Dezimalziffern in einen 10 Bit umfassenden Code automatisch übersetzt und weiterverarbeitet.

Darüber hinausgehend wird erforscht, und IBM bietet bereits einen solchen Leser an, wie auch mit normalen unterschiedlichen Lettern gedruckte bzw. geschriebene Informationen gelesen werden können. Man arbeitet an Umwandlern, die gesprochene Worte in automatenverwendbare Informationen umformen. Diese Geräte reichen jedoch bereits in das Gebiet der „lernenden Maschinen". Schließlich sind Versuche erfolgreich abgeschlossen, grafische Darstellungen unmittelbar als Inputinformationen zu verwenden.

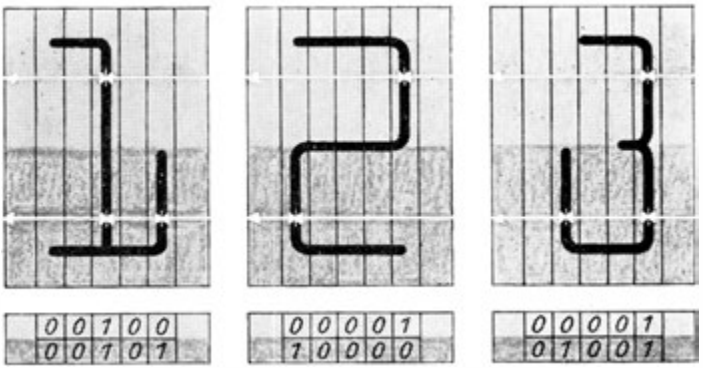


Bild 54. Automatische Codierung optisch lesbarer Zeichen von NCR

3.3. Aufbau und Arbeitsweise eines programmgesteuerten Digitalrechners und seiner Zusatzgeräte

3.3.1. Grundschaltungen

Alle Digitalrechner arbeiten im Dualsystem (von einigen Experimenten, Grundelemente für direktes dezimales Rechnen zu entwickeln, sei abgesehen, da sie keine praktische Bedeutung haben). Dafür benötigen wir bistabile Schaltelemente. Schaltelemente also, die zwei stabile Zustände annehmen können. Der Umschlag muß zudem von einem Zustand in den anderen möglichst schnell ohne „Nebeneinwirkungen" erfolgen. Dem genügen die vorhandenen Schaltelemente jedoch nicht. Daher müssen wir mit ihnen erst eine Grundschaltung aufbauen, die den genannten Forderungen entspricht.

Diese Grundschaltung ist das *Flip-Flop* (engl. – Wippe). Wir können sie uns mechanisch durch eine Wippe veranschaulichen (Bild 55).

Die Wippe verändert ihre Stellung nur, wenn auf das hochstehende Ende eine Kraft wirkt.

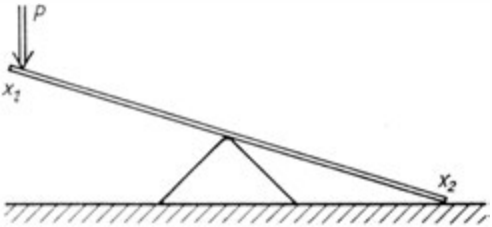


Bild 55. Mechanisches Modell eines Flip-Flop (Wippe)

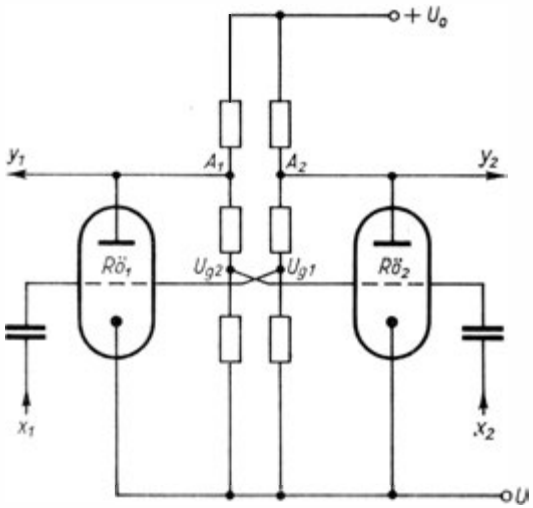


Bild 56. Flip-Flop-Schaltung mit Elektronenröhren

Das elektronische Modell einer Flip-Flop-Schaltung können wir uns mit Hilfe zweier Elektronenröhren (Trioden) aufbauen (Bild 56).
Das System hat zwei stabile Stellungen:

I. Röhre 1 sperrt, und Röhre 2 ist leitend.
Dann schreiben wir den Ausgängen folgende Werte zu:

$$y_1 = \mathrm{L};\ y_2 = \mathrm{O}$$

II. Röhre 1 ist leitend, und Röhre 2 sperrt, was der Festlegung

$$y_1 = \text{O};\ y_2 = \text{L}$$

entsprechen soll.
Der Umschlag erfolgt durch positive Impulse an den Eingängen x_1 oder x_2.
Nehmen wir an, das System befinde sich im Zustand 1, also Röhre 1 sperrt, Röhre 2 leitet, $y_1 = \text{L}$; $y_2 = \text{O}$. Dann bewirken:
ein positiver Impuls in x_2 keine Änderung,
ein positiver Impuls in x_1 einen Umschlag auf
den Zustand II, also Röhre 1 leitend, Röhre 2 sperrend,

$$y_1 = \text{O};\ y_2 = \text{L}.$$

In diesem Zustand II würde ein positiver Impuls in x_1 keinen Umschlag bewirken, während ein solcher in x_2 den Zustand I wieder herstellt.
Auf diese Art ist es auch möglich, Informationen mit Röhren zu speichern. Der Speicherungsprozeß geht sehr schnell vor sich. Allerdings geht die Information bei Ausfall des Stromes verloren. Derartige Flip-Flop-Speicher sind keine permanenten Speicher.
In gleicher Weise kann diese Grundschaltung mit anderen Schaltelementen aufgebaut werden, beispielsweise mit Transistoren.
Aus Flip-Flop-Schaltungen können wir nun die Boolesche Algebra aufbauen und uns so für jedes Schaltelement einen Halbadder konstruieren.

3.3.2. Rechenwerk

Nach der internen Übertragung der Daten unterscheiden wir zwei Grundtypen von Rechenautomaten: Rechner, die im Parallelbetrieb, und Rechner, die im Serienbetrieb arbeiten.
Im *Parallelbetrieb* werden alle Stellen eines Wortes zum gleichen Zeitpunkt (Takt) durch eine entsprechende Anzahl von Leitungen gesandt. Das erfordert einen großen technischen Aufwand, denn es müssen für das gesamte Wort bei konstanter Wortlänge

bzw. für die maximale Größe bei variabler Wortlänge gleiche Leitungen und Schaltelemente vorhanden sein. Dafür erreichen wir bei Parallelbetrieb aber auch eine große Arbeitsgeschwindigkeit.

Serienbetrieb erfordert die Durchgabe des Wortes durch eine Leitung. Der technische Aufwand ist geringer, aber die Arbeitszeit dafür wesentlich größer.

Es gibt daher eine Kombination, den Serienparallelbetrieb. Hierbei werden die Zeichen in Serie, die Bit eines Zeichens jedoch parallel weitergeleitet.

Das Rechenwerk ist derjenige Teil eines Automaten, in dem die arithmetischen Operationen – Addition, Subtraktion, Multiplikation und Division – sowie einige andere Operationen, wie logische Verknüpfungen, Verschiebungen, ausgeführt werden. Wie wir uns im Abschnitt „Rationale Rechenoperationen" bereits erarbeitet haben, lassen sich alle rationalen Rechenoperationen mit Hilfe der Komplementbildung und Verschiebeoperationen auf die Addition zurückführen. Folglich ist ein Addierwerk das Kernstück des Rechenwerkes.

Die Durchführung einer Rechenoperation (beispielsweise der Addition) verlangt im wesentlichen drei Schritte:

1. Bereitstellung der beiden zu verknüpfenden Operanden: „Lesen",
2. Ausführung der eigentlichen Rechenoperation,
3. Lieferung und Aufbewahrung des Ergebnisses: „Schreiben".

Das gilt natürlich auch für die anderen Operationen, z. B. für die Multiplikation. Eine Multiplikation wird aber auf eine mehrfache Addition und Verschiebung zurückgeführt. Das hätte auch ein mehrfaches „Lesen" und „Schreiben" zur Folge. Da dies bei vielen Automaten in den üblichen Speichereinrichtungen zu lange dauern würde, sind spezielle Speicher, die Register, zwischen die Speichereinrichtung und das Leitwerk geschaltet.

Register sind einzellige Speicher, die extrem kurze Zugriffszeiten haben. Sie dienen der unmittelbaren Bereitstellung der Daten oder Befehle zur Abarbeitung. Register werden bei Rechenautomaten sehr vielfältig verwendet. Selbst bei Speichern mit relativ langer Zugriffszeit kann man die Rechengeschwindigkeit durch

Verwendung von Registern recht günstig halten. Das Rechenwerk eines Automaten kann bei Serienbetrieb mit Hilfe von Registern und einem Addierwerk aufgebaut werden. Hierfür gibt es verschiedene Ausführungen:
Eine Möglichkeit besteht darin, daß zwei Operandenregister für die Bereitstellung der Operanden und ein Resultatregister für die Aufbewahrung des vom Adder gelieferten Ergebnisses sorgen (Bild 57).
Eine andere Möglichkeit zeigt Bild 58.
Das eine Operandenregister fällt hier mit dem Resultatregister (Akkumulator) zusammen. Nach erfolgter Rechenoperation gelangt das Resultat in den Akkumulator und wird bis zur nächsten Operation aufbewahrt (akkumuliert). Der von der Opera-

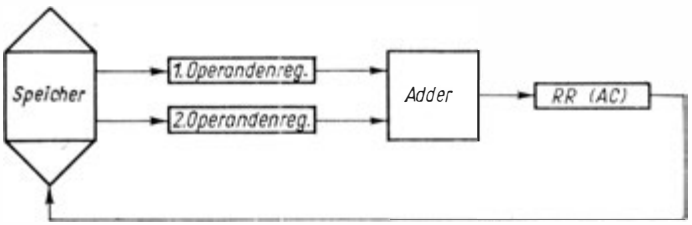


Bild 57. Schema eines Rechenwerkes mit zwei Operandenregistern und einem gesonderten Resultatregister

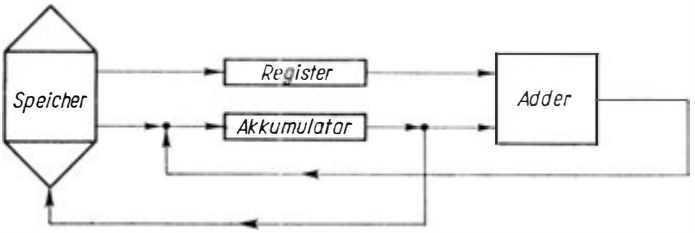


Bild 58. Schema eines Rechenwerkes mit einem Operandenregister und dem Akkumulator als Resultatregister

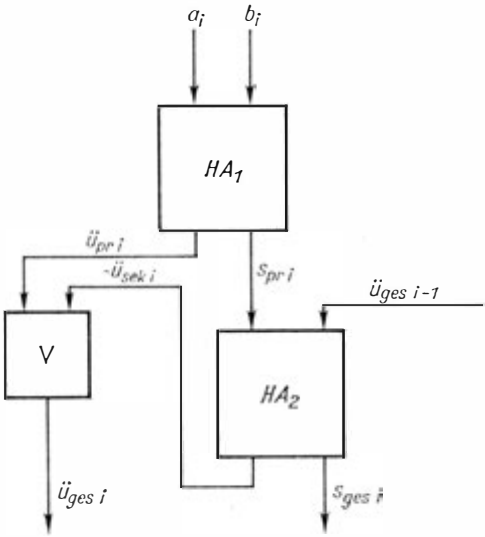


Bild 59. Schema eines Adders aus 2 Halbaddern und einer Disjunktion

tion im Akkumulator bereitgestellte Operand geht damit allerdings verloren und steht dem Rechenwerk nicht mehr zur Verfügung.

Ein Addierwerk für Parallelbetrieb (allerdings mit verkürzter Wortlänge) wurde bereits in Bild 10 dargestellt. Allerdings blieb dort noch offen, wie ein Adder aus den Halbaddern aufgebaut werden kann. Bild 59 gibt hierfür die Erklärung. Die beiden Spaltenziffern a_i und b_i werden in den ersten Halbadder HA_1 geleitet. Sie liefern die primäre Spaltensumme $s_{pr\,i}$ und den Primärübertrag $\ddot{u}_{pr\,i}$. Den Wert $s_{pr\,i}$ leiten wir mit dem Gesamtübertrag $\ddot{u}_{ges\,i-1}$ aus der $(i-1)$-ten Spalte in den zweiten Halbadder HA_2. Der liefert uns die Gesamtspaltensumme $s_{ges\,i}$ und den Sekundärübertrag $\ddot{u}_{sek\,i}$. Primär- und Sekundärübertrag können nun nicht gleichzeitig L sein. Daher liefert eine Disjunktion den Gesamtübertrag $\ddot{u}_{ges\,i}$.

Dies können wir an folgender Tabelle leicht nachrechnen.

a_i	b_i	$s_{pr\ i}$	$\ddot{u}_{ges\ i-1}$	$\ddot{u}_{pr\ i}$	$\ddot{u}_{sek\ i}$	$\ddot{u}_{ges\ i}$	$s_{ges\ i}$
O	O	O	O	O	O	O	O
L	O	L	O	O	O	O	L
O	L	L	O	O	O	O	L
L	L	O	O	L	O	L	O
O	O	O	L	O	O	O	L
L	O	L	L	O	L	L	O
O	L	L	L	O	L	L	O
L	L	O	L	L	O	L	L

Wir haben also drei Eingänge a_i, b_i und $\ddot{u}_{i-1}$ und die Ausgänge s_i, $\ddot{u}_i$ mit den richtigen Werten. Dieser Adder eignet sich besonders für Parallelbetrieb, also für ein Addierwerk, wie auf Bild 10 dargestellt.

Wir wollen im folgenden auch den Aufbau und die Wirkungsweise eines Serienadders kennenlernen.

Die entsprechenden Ziffern der beiden Summanden laufen mit ihrem niedrigsten Stellenwert zuerst in den Adder ein. (In der üblichen Zahlenschreibweise bedeutet das einen Transport von links nach rechts.) Der Einlauf der beiden Operanden ist an einen ganz bestimmten Takt gebunden und geschieht so, daß immer Ziffernpaare gleichen Stellenwertes zugleich einlaufen.

Ein Serienadder läßt sich durch zwei hintereinander folgende Halbadder und ein Element zur Verzögerung um einen Takt realisieren.

In den ersten Halbadder laufen paarweise die Ziffern a_k und b_k beider Summanden ein. Die Ziffern mit dem niedrigsten Stellenwert seien mit a_1, b_1, die mit dem nächsthöheren mit a_2, b_2, die jeweils entstehende Partialsumme mit s_1, s_2 und der zugehörige Übertrag mit $\ddot{u}_1$ und so weiter bezeichnet.

An den Ausgängen des ersten Halbadders stehen nach der Verarbeitung von a_1, b_1 die Partialsumme s_1 sowie der primäre Übertrag $\ddot{u}_1$ zur Verfügung. Der primäre Übertrag $\ddot{u}_1$ muß nun so verzögert werden, daß er zum gleichen Takt mit der nächstfolgenden Partialsumme s_2 in den zweiten Halbadder einläuft. Er wurde mit $\ddot{u}_{1\ verz}$ bezeichnet. Dieser bildet aus s_2 und $\ddot{u}_{1\ verz}$,

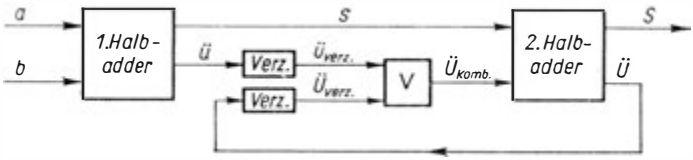


Bild 60. Adder für Serienbetrieb

falls $s_2 = \mathrm{L}$ und $\ddot{u}_{1\ \mathrm{verz}} = \mathrm{L}$, einen sekundären Übertrag $\ddot{U}_1$. Könnte man $\ddot{U}_1$ an den Eingang des zweiten Halbadders zurückführen und dabei so verzögern, daß hier s_3, $\ddot{u}_{2\ \mathrm{verz}}$ und $\ddot{U}_{1\ \mathrm{verz}}$ wirksam sind, würde der Ausgang die Endsumme S liefern. Damit hat der zweite Halbadder drei Eingangsgrößen, so daß er als Adder ausgebildet werden müßte. Das ist jedoch nicht nötig, da $\ddot{U}_{n+1\ \mathrm{verz}}$ und $\ddot{U}_{n\ \mathrm{verz}}$ nicht gleichzeitig L sein können, wie folgende Überlegung zeigt.

Ein primärer Übertrag kann nur dann entstehen, wenn die beiden in den Halbadder einlaufenden Summandenziffern = L sind. Dann ist aber $s = \mathrm{O}$. Ein sekundärer Übertrag könnte sich aber in dem Falle nur dann bilden, wenn ein verzögerter primärer und ein verzögerter sekundärer Übertrag zugleich wirksam werden. Da dies aber die Bildung eines sekundären Übertrages immer in der nächstniedrigen Stelle voraussetzt und bei der niedrigsten Stelle niemals ein sekundärer Übertrag auftritt, kann also kein verzögerter primärer und sekundärer Übertrag gleichzeitig am Eingang des zweiten Adders entstehen. Es kann bei der Addition von drei einziffrigen Dualziffern höchstens ein Übertrag von L auftreten.

Daraus ergibt sich die Möglichkeit, die Eingänge des verzögerten primären und sekundären Einlaufes disjunktiv zu vereinigen, so daß am zweiten Halbadder nur noch s und $\ddot{U}_{\mathrm{komb}} = \ddot{u}_{\mathrm{verz}} \vee \ddot{U}_{\mathrm{verz}}$ wirken (Bild 60).

3.3.3. Leitwerk

Als Kennzeichen der neuen Qualität in der maschinellen Bearbeitung geistiger Prozesse haben wir die Programmsteuerung erkannt. Programmgesteuerte Rechenautomaten können somit

nach der vorher festgelegten Befehlsfolge des Programms selbständig ohne Eingriff des Menschen arbeiten.
Das Leitwerk eines programmgesteuerten Rechenautomaten hat die Aufgabe, Befehle in der vom Programm vorgegebenen Reihenfolge dem Speicher zu entnehmen, zu entschlüsseln und je nach Ergebnis der Entschlüsselung die entsprechende Operation auszulösen.
Ein Befehl besteht aus dem Operations- und dem Adreßteil.
Der Operationsteil gibt an, welche arithmetische, logische oder organisatorische Operation ausgeführt werden soll, und der Adreßteil informiert, woher der Operand oder die Operanden zu entnehmen sind, mit dem (denen) die Operation ausgeführt werden soll.
Die Struktur des Automaten ist vom Adreßteil abhängig. Hiernach unterscheidet man fünf Grundtypen.
Die *Fünfadreßmaschine* hat im Adreßteil fünf Adressen für folgenden Code:

1. Adresse: Adresse des ersten Operanden
2. Adresse: Adresse des zweiten Operanden
3. Adresse: Adresse für das Ergebnis
4. Adresse: Adresse des nächstfolgenden Befehls
5. Adresse: Adresse des nächstfolgenden Befehls in Alternative zur 4. Adresse.

Die *Vieradreßmaschine* gliedert den Adreßteil wie folgt:

1. Adresse: Adresse des ersten Operanden
2. Adresse: Adresse des zweiten Operanden
3. Adresse: Adresse für das Ergebnis
4. Adresse: Adresse für den nächstfolgenden Befehl.

Soll der nächstfolgende Befehl durch eine Alternative erst festgelegt werden, muß ein spezieller Befehl, der bedingte Sprung (durch die Alternative bedingt), vorhanden sein.

Dreiadreßmaschinen haben folgende Verteilung:

1. Adresse: Adresse des ersten Operanden
2. Adresse: Adresse des zweiten Operanden
3. Adresse: Adresse für das Ergebnis.

Die Befehle werden normalerweise in der programmierten Reihenfolge abgearbeitet. Soll diese Reihenfolge im Rechenprozeß verändert werden, sind Befehle für Sprungoperationen erforderlich.

Für die *Zweiadreßmaschine* gilt:

1. Adresse: Adresse des ersten Operanden
2. Adresse: Adresse des zweiten Operanden.

Das Ergebnis wird in einem Resultatregister zwischengespeichert und muß dann mit einem sogenannten Schreibbefehl in den Speicher transportiert werden. Für die Befehlsabarbeitung gilt hier ebenfalls das für die Dreiadreßmaschine Bemerkte.
Der *Einadreßautomat* kann in dieser Adresse lediglich die eines Operanden oder Befehls aufnehmen. Er benötigt für die Bereitstellung der Operanden zwei Register, die durch Lesebefehle mit den Operanden gefüllt werden müssen. Für das Resultat und die Befehlsabarbeitung sind alle Bemerkungen der Zweiadreßmaschine zu übernehmen.
Neben den angeführten Grundtypen gibt es noch mehrere Spezialausführungen, auf die wir nicht näher eingehen wollen.
Praktisch werden Ein- und Dreiadreßmaschinen bevorzugt.
Bei modernen Rechenautomaten gibt es zudem die Möglichkeit, daß der Automat nach dem vorgegebenen Programm selbständig die Adresse verändert.
Nach diesen Bemerkungen wird der Arbeitsablauf eines Leitwerkes für eine Einadreßmaschine verständlich, wie es das vereinfachte Diagramm in Bild 61 veranschaulicht.
Über eine Befehlsaufrufsteuerung (BA) wird nach einer Befehlsadresse aus dem Befehlszähler (BZ) der Befehl aus dem Speicher in das Befehlsregister transportiert. Hierauf erfolgt eine Überprüfung (nach vorherprogrammierten Zeichen), ob eine Adressenänderung vorgenommen werden muß. Ist dies der Fall, so wird die Adressenänderung gegebenenfalls über ein Adressenrechenwerk respektive über den erneuten Aufruf einer Information aus dem Speicher durchgeführt.
Liegt die endgültige Adresse vor, so erfolgt die Operationssteuerung. Abschließend wird der Befehlszähler um eine Einheit erhöht. Die Arbeitsweise im Automaten deutet Bild 62 an.

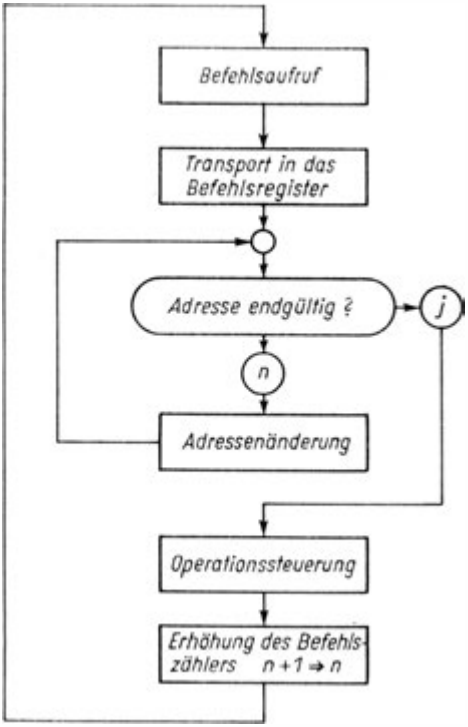


Bild 61. Arbeitsdiagramm für das Leitwerk einer Einadreßmaschine

Mit dem Leitwerk ist meist ein Kommandopult verbunden. Das Kommandopult ist der Arbeitsplatz des Automatenbedieners. Eine Anzahl von Lampen und Lampenfeldern gibt ihm Auskunft über den Betriebszustand des Automaten. Danach kann er bei eventuellen Störungen eingreifen und Fehler oder Störungsquellen beseitigen.
Ist ein Blattschreiber vorhanden, werden über denselben Informationen über den Bearbeitungsstand eines oder mehrerer gleichzeitig ablaufender Programme gegeben.

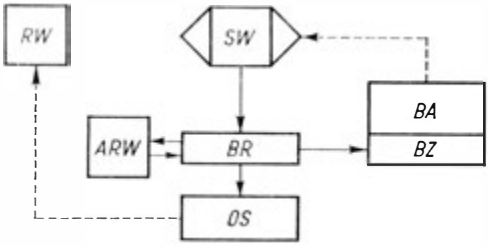


Bild 62. Blockschaltbild zur Demonstration der Arbeit des Leitwerkes einer Einadreßmaschine

RW	Rechenwerk,	BA	Befehlsaufrufsteuerung,
SW	Speicherwerk,	BZ	Befehlszähler,
ARW	Adressenrechenwerk,	OS	Operationssteuerung
BR	Befehlsregister,		

Das erwähnte Lampenfeld mit den Schaltknöpfen, noch stärker jedoch die Schreibmaschine, stellen auch eine einfache (und sehr langsame) Ein- und Ausgabe dar.

3.3.4. Speichereinrichtungen

Speicher sind das Gedächtnis eines Automaten. Alle Angaben, die der Automat während der rechnerischen Bearbeitung eines Problems benötigt, seien es Anfangsdaten, Zwischenergebnisse, Endergebnisse, irgendwelche Konstanten oder Befehle, werden in den Speichereinrichtungen aufbewahrt.
Damit die einzelnen Angaben für eine spätere Weiterverarbeitung wieder auffindbar sind, muß der Speicher adressierbar sein.
Bei fester Wortlänge ist er dazu in Zellen unterteilt, die jeweils ein Wort aufnehmen können (24 bis 50 Bit). Jede Speicherzelle erhält dabei als Kennzeichen eine fortlaufende Nummer, ihre Adresse.
Für variable Wortlänge kann der Wortanfang adressiert werden, da ja jedes Byte adressierbar ist. Das Wortende ergibt sich über die Wortmarke, Gruppen von Daten über Satz-, Gruppen- und Blockmarken. Beim Auffinden einer Angabe braucht also nur

die Adresse des ersten Zeichens bekannt zu sein, dann kann diese Information jederzeit zur Weiterverwendung abgerufen werden.

An die Speichereinrichtungen eines Automaten müssen wir große Anforderungen stellen. Speicher liefern neben der Rechengeschwindigkeit die wichtigste Kenngröße für die Einschätzung eines Automaten.

Ein idealer Speicher sollte:

den Speicherinhalt möglichst lange ohne Verfälschung erhalten (auch wenn durch Störungen zeitweilig der Strom ausfällt; man sagt dann auch, er sollte möglichst energieunabhängig oder permanent sein),

ein großes Fassungsvermögen haben sowie

eine kurze Zugriffszeit (das ist die für das Aufsuchen der Speicherzelle, für das Entnehmen des Wortes und für dessen Transport bis zur weiteren Verwendung benötigte Zeitspanne) zum Inhalt einer beliebigen Speicherzelle,

niedrige Herstellungs- und Unterhaltungskosten,

eine große Betriebssicherheit und

einen möglichst geringen Raumbedarf.

Speicher, die allen diesen Forderungen gleichermaßen genügen, gibt es jedoch nicht. Sie charakterisieren ja auch einen idealen Speicher. In der Praxis widersprechen sich die einzelnen der angeführten Forderungen. So ist beispielsweise die Zugriffszeit der Speicherkapazität direkt proportional. Entgegen unseren Wünschen wächst sie also mit dem Anwachsen der Speicherkapazität. Wollen wir sie dennoch verkürzen, so wirkt sich dies auf die Kosten aus. Letztere steigen zudem bei gleichem Speicherprinzip mit dem Anwachsen der Kapazität. So lassen sich noch viele Widersprüche anführen. Man versucht nun, sie dadurch zu umgehen, daß man für eine Anlage je nach Verwendungszweck verschiedene Speicherarten vorsieht.

Wir unterscheiden nach ihrer Funktion

Pufferspeicher,
Register,
Arbeitsspeicher und
Zubringerspeicher.

Pufferspeicher dienen zur Koppelung zweier Aggregate mit unterschiedlichen Arbeitsgeschwindigkeiten. Wollen wir beispielsweise an einen sehr schnellen Rechner zusätzlich einen langsameren Drucker anschließen, so haben wir zwei Möglichkeiten.

Einmal können wir die Werte auf einen Zwischenträger übergeben, beispielsweise auf ein Magnetband, von dem dann die Daten in den Drucker übertragen werden. Das Magnetband kann dabei den unterschiedlichen Arbeitsgeschwindigkeiten angepaßt werden. Der Drucker arbeitet dann im *off-line-Betrieb* (off-line-processing: Verarbeitung getrennt vom übrigen System). Zum anderen können wir den Drucker direkt an den Rechner anschließen und die unterschiedlichen Arbeitsgeschwindigkeiten durch einen Pufferspeicher ausgleichen. Hier arbeitet der Drucker im *on-line-Betrieb* (on-line-processing: Verarbeitung angeschlossen an das übrige System).

Register haben wir bereits beim Rechenwerk besprochen. Es sind eigentlich Arbeitsspeicher begrenzter Kapazität, für ein Wort oder ein Wortteil, z. B. die Adresse bzw. eine begrenzte Zahl alpha-numerischer Zeichen ausgelegt. Sie sind an bestimmten Stellen fest im Automaten vorgegeben und arbeiten unmittelbar mit anderen Teilaggregaten zusammen oder sind selber ein Teil der Aggregate. Register haben keine numerischen Adressen, sondern meist feste Bezeichnungen. So gibt es beispielsweise:

Operandenregister zur Bereitstellung und Verarbeitung der Operanden,
Akkumulatoren zur Aufnahme des Ergebnisses einer Operation, teilweise auch Resultatregister genannt,
Befehlsregister zur Bereitstellung eines Befehls,
Befehlsaufrufregister zur Abberufung des nächstfolgenden Befehls und andere.

Arbeitsspeicher stehen unmittelbar mit dem Rechenwerk in Verbindung. Aus ihnen können Operanden abgerufen werden. Es muß also das Einzelwort adressierbar sein.

Zubringerspeicher sollen dagegen solche genannt werden, bei denen die Informationen erst über einen Arbeitsspeicher bereit-

gestellt werden können. Sie sind meist nur für eine Gruppe oder einen Block von Daten adressierbar.

Vielfach finden wir auch die Teilung in innere oder interne Speicher, die unmittelbar zum Automaten gehören, und äußere, auch extern oder peripher genannte Speicher, die zusätzlich an den Automaten angeschlossen werden können. Die letzte Benennung ist jedoch nicht eindeutig. Wir wollen daher nach der zuerst angeführten Bezeichnung einteilen. Arbeitsspeicher haben meist eine niedrigere Zugriffszeit als Zubringerspeicher. Dafür sind sie von geringerer Speicherkapazität. Ein geschickter Programmierer wird mit dem Arbeitsspeicher rechnen und möglichst lange Informationsblöcke aus dem Zubringerspeicher überführen, was meist simultan ausgeführt werden kann. Er hält damit den Einfluß der Zugriffszeit, insbesondere zu den Zubringerspeichern, möglichst klein. Die Arbeitsspeicher sind bei allen Automaten als interne Speicher ausgeführt, d. h., sie gehören unmittelbar zum Automaten und sind in ihm fest eingebaut. Bei Zubringerspeichern ist das nicht eindeutig. Sie können Innen- wie auch Außenspeicher sein.

Betrachten wir die technische Ausführung. Im Automaten sind die Informationen „O" und „L" durch elektrische Impulse gegeben. Die Speichereinrichtung muß diese physikalische Darstellungsform so umwandeln, daß sie stabil, adressierbar, mit kurzer Zugriffszeit, kompakt, permanent und mit großem Fassungsvermögen gespeichert werden kann. Hierfür haben sich in der Praxis drei Speicherungsprinzipien auf elektromagnetischer Basis als besonders zweckmäßig erwiesen:

Magnetschichtspeicher,
Magnetkernspeicher und
Dünnschichtspeicher.

Magnetschichtspeicher beruhen auf einem einfachen physikalischen Prinzip. Auf eine nichtmagnetisierbare Trägerschicht wird eine Magnetschicht, beispielsweise aus einer Nickel-Kobalt-Legierung, aufgetragen. Die Elektroimpulse werden über Schreibköpfe in Magnetfelder umgewandelt, die lokal polarisierte Magnetfelder auf die Magnetschicht „schreiben" (Bild 63). Damit ist die Information nahezu unbegrenzt, permeabel gespeichert.

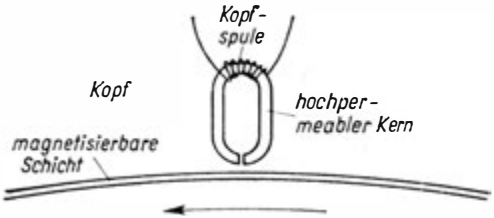


Bild 63. Schreib- oder Lesekopf

Beim „Lesen" vollzieht sich ein umgekehrter Vorgang. Die Magnetfelder induzieren beim Durchgang unter dem Lesekopf einen Stromimpuls in der Spule, der verstärkt wieder im Automaten zur weiteren Verarbeitung verfügbar ist.
Je nach geometrischer Form der Trägerschicht unterscheidet man

Magnettrommel-,
Magnetplatten-,
Magnetband- und
Magnetkartenspeicher,

die mit den entsprechenden Steuereinheiten den kompletten Speicher bilden.
Magnettrommelspeicher haben als Träger einen Zylinder, auf dessen Mantelfläche die Magnetschicht aufgetragen ist. Im Trommelgehäuse sind dann die Schreib- und Leseköpfe angebracht, die jeweils eine Spur des Zylinders beschreiben oder lesen können (Bild 64).
Die Teilung kann nach Zellen oder Zeichenplätzen erfolgen. Diese werden einzeln durch eine Adresse gekennzeichnet. Der Trommelspeicher hat also die Adressiermöglichkeit für das einzelne Wort oder Zeichen. Die Adresse ergibt sich durch Auswahl der Spur oder Bahn als Bahnadreßteil über eine Spurauswahlkoordinate und durch Angabe des Platzes auf der Spur über eine Spurumlaufkoordinate, die den Charakter einer Zeitkoordinate hat. Benutzt man die Zeit als steuernde Koordinate, muß gewährleistet sein, daß alle Vorgänge im Automaten mit dem Umlauf der Trommel synchron laufen. Die Synchronisation

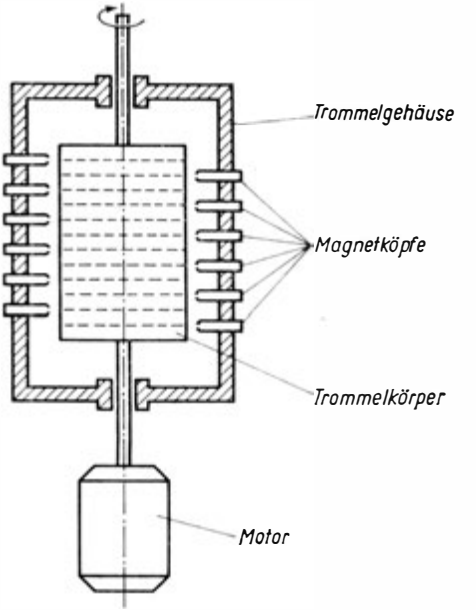


Bild 64. Skizze eines Trommelspeichers

wird dadurch erreicht, daß die gesamte Impulsversorgung des Automaten durch die Trommel gesteuert wird. Zu diesem Zweck befinden sich auf der Trommel spezielle Taktspuren, die den Grundtakt angeben.

Es gibt nun verschiedene Möglichkeiten, aus der Anzahl von Zellen je Spurumlauf eine bestimmte auszuwählen. Grundprinzip bei allen ist, daß die Adresse jeder Zelle mit der gewünschten Adresse verglichen und bei Übereinstimmung, man sagt auch Koinzidenz, ein Signal abgegeben wird, das den Lese- und Schreibvorgang auslöst.

Magnettrommelspeicher sind relativ billig, jedoch durch ihre mechanisch bewegten Teile störanfällig. Sie werden heute für verschiedenste Kapazitäten verwendet. Dabei hängt die Größe

der Kapazität von der Größe der Trommel, der Drehgeschwindigkeit und der Informationsdichte je Spur ab.
Letzteres ist wiederum abhängig von den Eigenschaften der Magnetschicht, vom Aufbau der Magnetköpfe und vom Abstand zwischen Magnetschicht und Kopf. Je kleiner dieser Abstand ist, um so größer ist die Informationsdichte.
Kleine Zugriffszeiten erreichen wir durch hohe Drehzahlen.
Als Beispiel einer kleinen Trommel seien die Daten für die Magnettrommel des ZRA 1 angeführt (Bild 65).

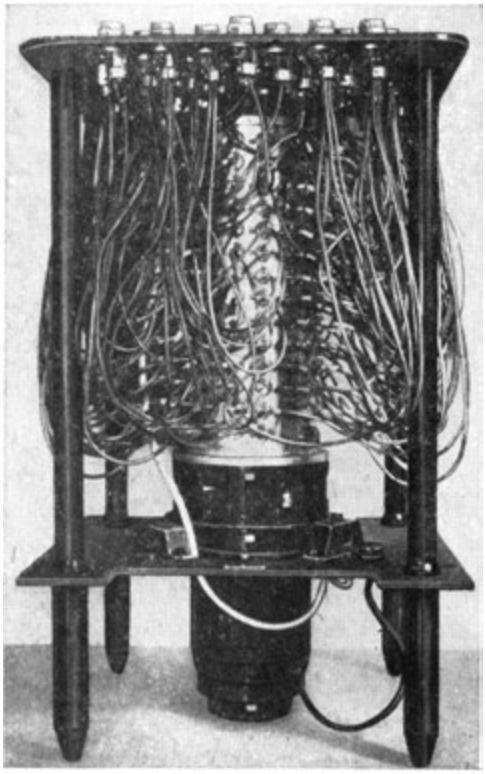


Bild 65. Trommelspeicher des ZRA 1

Speicherkapazität	4096 Worte zu je 48 Bit
Drehzahl	11400 U/min
mittlere Zugriffszeit	2,5 ms
Abstand von Spurmitte zur Spurmitte der Nachbarspur	1,3 mm
Informationsdichte einer Spur	3,3 Inform/mm
Abstand zwischen Magnetkopf und -schicht	30 µm

Große Trommeln, wie sie beispielsweise von der Londoner ICT[1] angeboten werden, haben folgende Werte:

Typ	Kapazität				Wartezeit	Übertragungsgeschwindigkeit	
	Zeichen je	Worte Spur	Spuren	Gesamtkapazität Zeichen		Zeit in	max. Baud
					ms	µs	
1962	1.024	256	128	129.072	10	80	50000
1963	2.048	512	256	324.288	10	40	100000
1964	4.096	1024	512	2.097.152	20	40	100000

Tabelle 5. Vergleichsdaten für Trommelspeicher

An jeden Automaten können bis zu 4 Trommeln angeschlossen werden, die bei den Typen wie folgt angeordnet sind:

1962 1 Gehäuse mit Steuereinheit und 4 Trommeln
1963 2 Gehäuse mit Steuereinheit und je 2 Trommeln
1964 4 Gehäuse mit Steuereinheit und je 1 Trommel.

Magnetplattenspeicher haben an Stelle der Trommel als Träger mehrere auswechselbare oder fest eingebaute Platten. Entsprechend unterscheidet man Wechselplatten- und Großplattenspeicher.

Magnetplattenspeicher sind auch einzeladressierbar. Hier können zudem Gruppen oder Blöcke von Daten abgerufen werden. Sie haben eine große Kapazität bei günstiger mittlerer Zugriffs-

[1] ICT: International Computers and Tabulators Co., London

Bild 66a. Plattenspeicher

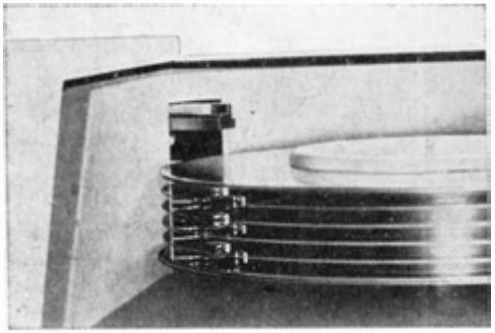


Bild 66b. Darstellung eines Plattenstapels

zeit. Um uns darüber einige Vorstellungen zu machen, seien vergleichsweise die Wechselplattenspeicher

ICT 1953, EAC[1] 4260 und IBM 1311

angeführt.

[1] EAC: Elliott-Automation Computers LTD.

Bild 67. Magnetbandspeicher

	Platten i. Magazin	Kapazität je Platte	Byte je Magazin	Übertragungsgeschwindigk. Zeichen/s	Mittlere Zugriffszeit	Einheit je Steuereinheit
ICT 1953	6	670 K[2]	4.030	66000	87,5 ms	4
EAC 4260	1[1]	600 K		135000		8
IBM 1311	5	20 K	2.020	50000	150,0 ms	4···8

[1] Einzelplatte in staubfreier Schutzkapsel

[2] Für 1024 Byte oder Wörter Speicherkapazität setzt man auch 1 K

Tabelle 6. Vergleichswerte für Wechselplattenspeicher

Betrachten wir auch noch einige Kennwerte des *Großplattenspeichers* ICT 1956:

maximale Kapazität 125···829 K

maximale Einstellzeit 275 ms
Übertragungsgeschwindigkeit 60 000 ··· 100 000 Zeichen/s

Magnetbandspeicher sind weit verbreitet. Es gibt viele technische Ausführungen. Der Träger der Magnetschicht ist ein etwa 300 bis 750 m langes, unterschiedlich breites Magnetband. Die Zeichen sind in einer Einheit meist in mehreren Spuren nebeneinander angeordnet (Bild 68).
Informationen auf Magnetbändern können nur als Datenblock abgerufen werden.
Zum Vergleich seien einige Magnetbandspeicher angeführt:

	Übertragungsgeschwindigkeit	Bandgeschw.	Blockzwischenraum	Kapazität (max.)	Anz. der anschließb. Einheiten
EAC 4268		1,5 m/s	18 mm	9.361 K	8
ICT 1973	max. 60000 Zeich/s	1,9 m/s	19,05 mm	24.000 K	24
ICT 1974	max. 96000 Zeich/s	3,05 m/s	19,05 mm	242.000 K	24

Tabelle 7. Vergleichswerte für Magnetbandspeicher

Magnetkartenspeicher werden von NCR und ICT angeboten. NCR entwickelte den Magnetkartenspeicher CRAM (Card Random Access Memory). Trägermedium sind Karten von der

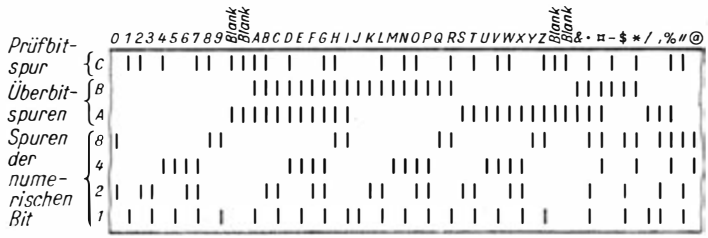


Bild 68. Schematische Darstellung der Magnetfelder zur Zeichendarstellung auf einem Magnetband

Bild 69. Binärmusterkerbung am Kopf der CRAM-Magnetkarte

Größe 80 mm × 350 mm. Sie sind am Kopf mit 8 dualverschlüsselten Kerben versehen. Nach einem bestimmten Binärmuster können wir so eine von 256 Karten auswählen (Bild 69). 256 Karten bilden ein Magazin, das in 30 s ausgewechselt werden kann. Ein Magazin kann 5.500 K bis 16.000 K Zeichen speichern. Mit einer Übertragungsgeschwindigkeit von 1 000 000 Zeichen/s und einer Zugriffszeit von 200 ms können Daten blockweise abgerufen werden.

Die ausgewählte Karte fällt an eine Hohltrommel und wird durch ein Teilvakuum angesaugt. Durch Lese- und Schreibeinrichtungen wird darauf der Lese- oder Schreibvorgang ausgeführt. Wird die Karte nicht mehr benötigt, „hängt" die Fliehkraft sie über die Rückführbahn wieder in das Magazin.

Der Magnetkartenspeicher 1958 des ICT-Systems verwendet Karten der Größe 114 mm × 406 mm und erreicht damit eine Kapazität von 340 Millionen bis 2,7 Milliarden alpha-numerischer Zeichen. Sie können in Blöcken zu 650 Zeichen mit einer Zugriffszeit von 365 ms und einer Übertragungsgeschwindigkeit von 80 000 Zeichen/s gelesen werden.

Magnetkartenspeicher liegen zwischen Magnetplatten- und Magnetbandspeichern. Sie sind sehr rentabel, zumal man beispielsweise mit einer Magnetkarteneinheit CRAM eine Zweiwegsortierung vornehmen kann. Sonst würde man dafür vier Magnetbandeinheiten benötigen.

Magnetkernspeicher verwenden als Informationsträger sogenannte „Rechteckferritkerne". Ferrite sind Eisen-Kohlenstoff-Verbindungen. Rechteckferrite weisen zudem die besondere magnetische Eigenschaft auf, daß sie eine nahezu rechteckige Hysteresisschleife haben.

Erregen wir einen solchen Ferrit in einer Erregungsrichtung (nehmen wir die positive) bis zum Sättigungsgrad und lassen die

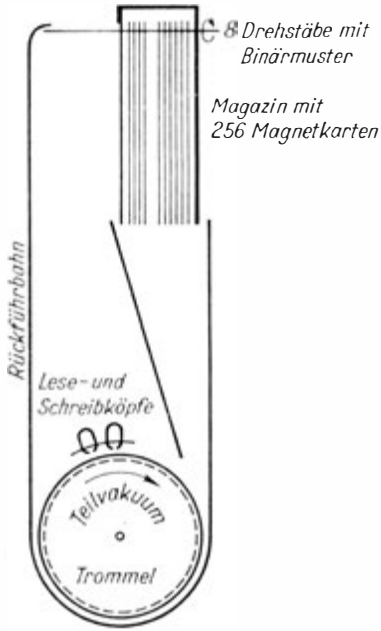


Bild 70. Schema der Arbeitsweise eines Magnetkartenspeichers

magnetische Erregung wieder abklingen bis zum Wert Null, so bleibt im Ferrit eine Restmagnetisierung, die *Remanenz* genannt wird, erhalten. Wir bezeichnen sie mit $+B_r$.
Wollen wir die Remanenz $+B_r$ beseitigen, dann müssen wir den Kern in entgegengesetzter Richtung (also nun negativ) erregen. Dabei ergibt sich aber, daß bei einer bestimmten Feldstärke, der *Koerzitivkraft* $-H_k$, nicht etwa die Nullage der Magnetisierung angenommen wird, sondern die negative Sättigung. Lassen wir jetzt die Erregung abklingen bis zum Wert Null, wird ein negativer Remanenzzustand $-B_r$ eingenommen.
Wiederholen wir diesen Vorgang in positiver Erregungsrichtung, so erfolgt bei der positiven Koerzitivkraft $+H_k$ der Umschlag zum positiven Sättigungsgrad, und nach Abklingen der Erregung wird wiederum der positive Remanenzzustand $+B_r$ eingenommen.

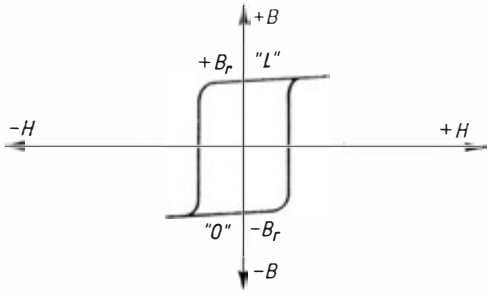


Bild 71. Hysteresisschleife der Rechteckferritkerne

Die dabei entstehende Kurve ist die Hysteresisschleife, wie sie auf Bild 71 dargestellt wird. Sie ist hier nahezu rechteckig, wonach die Ferrite ihren Namen erhalten haben. Die Bedeutung dieses magnetischen Verhaltens liegt darin, daß Ferrite mit rechteckiger Hysteresisschleife durch eine genau bestimmte Koerzitivkraft $\pm H_k$ gezwungen werden, von einem entsprechenden Remanenzzustand in den anderen umzuschlagen. Wird dagegen die Koerzitivkraft nicht erreicht, erfolgt lediglich eine kurzzeitige Verschiebung auf dem waagerechten Teil der Hysteresisschleife ohne Umschlag in den anderen Remanenzzustand.
Rechteckferritkerne erhalten wir, wenn wir aus dem beschriebenen Material kleine Ringe mit 1 bis 3 mm Außendurchmesser anfertigen. Auf Grund der rechteckigen Hysteresisschleife haben die Kerne also zwei stabile Remanenzzustände $+B_r$ und $-B_r$, die sich zur Speicherung der beiden Zahlen O und L des dualen Zahlensystems eignen. Dem positiven Remanenzzustand $+B_r$ ordnen wir den Wert L und dem negativen $-B_r$ den Wert O zu. Damit eine Umschaltung in beide Remanenzzustände erfolgen kann, sind auf den Kern bestimmte Wicklungen aufgebracht. Mittels eines Stromimpulses durch die Empfängerwicklung des Kernes kann er mit der Information O oder L versehen werden. Welche Information der Kern enthält, ist abhängig von der Richtung des vom Stromimpuls erregten Magnetfeldes.
Wir wollen davon ausgehen, daß sich der Kern in seinem stabilen Zustand $-B_r$ = O befindet. Zur Zeit $t = t_1$ schaltet der positive

Impuls den Kern in den anderen Zustand ($+B_r$) um. Der Arbeitspunkt verläuft dabei entlang des gestrichelt gezeichneten Kennlinienteiles im Bild 72.

Der negative Impuls zur Zeit $t = t_2$ schaltet den Kern wieder in den ursprünglichen Zustand ($-B_r$) zurück. Der vollausgezogene Kennlinienteil zeigt den Arbeitspunktverlauf.

Während des jeweiligen Umschaltvorganges wird in der Lesewicklung nach dem Transformatorprinzip ein richtungsabhängiger Spannungsimpuls induziert, der entsprechend ausgewertet werden kann.

Wir können zwei verschiedene Typen von Magnetkernspeichern unterscheiden, einmal die Magnetkernschiebespeicher und zum anderen die Magnetkernmatrixspeicher.

Bei Magnetkernschiebespeichern sind die Magnetkerne durch ein elektrisches Netzwerk so zu einer Kette verbunden, daß ihre Informationen durch Taktimpulse (Treiberimpulse) von Kern zu Kern übertragen (verschoben) werden.

Im Bild 73 ist schematisch der Transport einer Informationsfolge gezeigt.

Nach einer bestimmten Anzahl von Takten, im Beispiel sind es 6, steht die Informationsfolge wieder in ihrer ursprünglichen Form im Speicher.

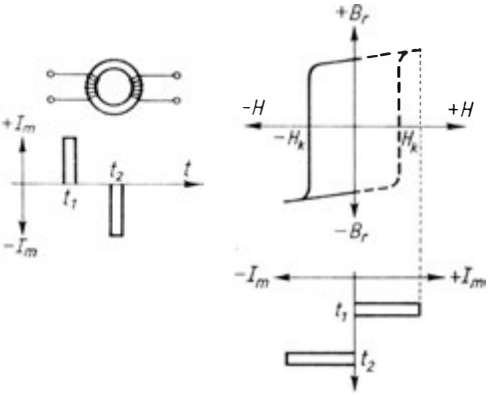
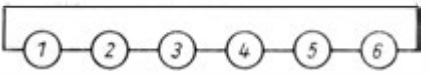


Bild 72. Skizze der Speicherung eines Bit



Ausgangs-stellung	L	0	L	0	0	0
1. Takt	0	L	0	L	0	0
2. Takt	0	0	L	0	L	0
3. Takt	0	0	0	L	0	L
4. Takt	L	0	0	0	L	0
5. Takt	0	L	0	0	0	L
6. Takt	L	0	L	0	0	0

Bild 73. Magnetkernschiebespeicher

Auf die einzelnen Verfahren des Informationstransportes, wie Ein-, Zwei- und Dreitaktverfahren, soll nicht näher eingegangen werden. Wir wollen jedoch vermerken, daß im ZRA 1 Magnetkernschiebespeicher als Register (Schnell- und Umlaufspeicher) Verwendung finden.

Magnetkernmatrixspeicher sind wegen ihrer extrem kleinen Zugriffszeit zu allen Zellen und ihrer hohen Betriebssicherheit ausgezeichnet als Arbeitsspeicher geeignet. Die Zugriffszeit wird im wesentlichen durch die Schaltzeit der Magnetkerne bestimmt. Sie liegt zwischen 1 und 6 µs.

Wie schon die Bezeichnung „Matrix" aussagt, sind die Speicherelemente wie Elemente einer Matrix, also in einem rechteckigen, hier speziell quadratischen Schema, angeordnet. Bild 74 skizziert eine solche Anordnung.

Im Kreuzungspunkt zweier Leitungen, der Zeilenleitung und der Spaltenleitung, befindet sich jeweils ein Magnetkern. Jeder Kern ist somit eindeutig durch seine Koordinaten „Zeile" und „Spalte" gekennzeichnet.

Soll ein Magnetkern in einen anderen Remanenzpunkt gebracht werden, so ist dazu eine Mindestfeldstärke notwendig, die größer als die Koerzitivkraft sein muß. Diese Feldstärke möge durch einen Strom (Magnetisierungsstrom) I_m erreicht werden.

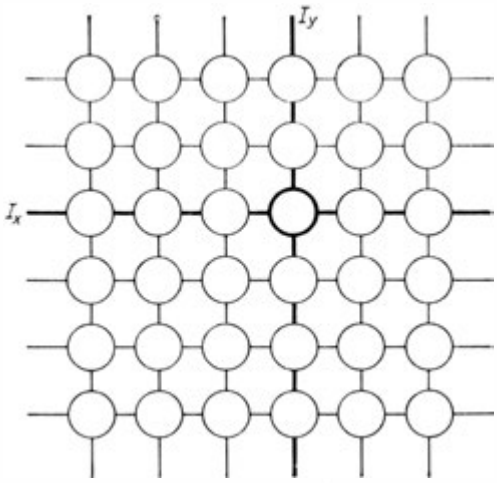


Bild 74. Skizze eines Magnetkernmatrixspeichers

Diesen Strom zerlegen wir in zwei gleich große Teilströme $I_x - I_y = \frac{1}{2} I_m$. Wirkt jetzt ein Teilstrom auf einen Ferrit, kann er keine Änderung des Remanenzzustandes herbeiführen. Nur wenn beide Teilströme, gleichzeitig auf einen Ferritkern wirken, kann er in den anderen Remanenzzustand umschlagen. Nunmehr brauchen wir nur entsprechend der von uns programmierten Zeilen- und Spaltenkoordinate die Teilströme zu leiten, dann wird lediglich der Kern beschrieben, der im Kreuzungspunkt der beiden Koordinaten liegt.

Auf Bild 74 wurde die dritte Zeile und vierte Spalte als Koordinate ausgewählt. Durch Einfließen der Stromimpulse I_{x3} und I_{y4} wird also der Kern (3/4) beschrieben.

Für das Speichern eines L werden dabei die Koordinatenteilströme $+I_{ix}$ und $+I_{ky}$ benötigt.

Zur Durchführung des Lesevorganges ist eine allen Kernen der Matrix gemeinsame Lesleitung vorhanden. Gelesen wird mit Hilfe des negativen Magnetisierungsstromes $-I_m$. Falls sich der

zu lesende Magnetkern im Zustand $+B_r = \mathrm{L}$ befindet, rufen $-I_{tx}$ und $-I_{ky}$ die Umschaltung auf $-B_r = \mathrm{O}$ hervor. Die Induktionsänderung von $+B_r$ nach $-B_r$ induziert in der Leseleitung einen Spannungsimpuls.

Falls der zu lesende Kern bereits die Information O hatte, bewirken $-I_{tx}$ und $-I_{ky}$ lediglich eine kurzzeitige Verschiebung des Arbeitspunktes auf dem waagerechten Teil der BH-Kennlinie – es wird keine Spannung in der Leseleitung induziert.

Der „abgelesene" Spannungsimpuls kann dann nach Verstärkung für weitere Berechnungen verwendet werden.

Im Gesamtaufbau eines Magnetkernspeichers wird von jedem Zeichen nur ein Bit in jede Matrix gespeichert. Es müssen für das Zeichen oder Wort also die entsprechende Zahl Matrizen (acht für das Zeichen wie auf Bild 76 oder 48 für ein 48-Bit-Wort) übereinandergeschichtet werden.

Beim Magnetkernspeicher wird die Information beim Lesen zerstört und muß anschließend wieder aufgebaut werden. Praktisch werden die Informationen zyklisch zerstört und wieder

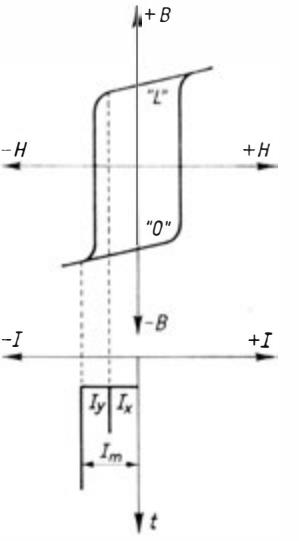


Bild 75. Speicherung bei einer Magnetkernmatrix

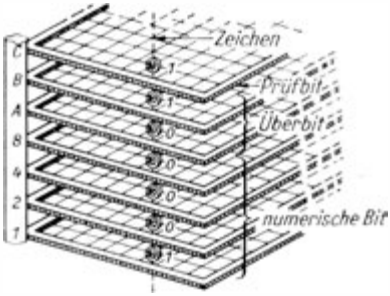


Bild 76. Skizze eines Magnetkernmatrixspeichers für 8-Bit-Zeichen

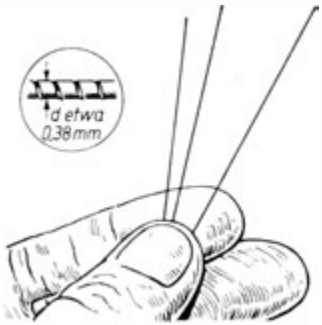


Bild 77. Dünnschicht-Stäbchen-Speicher

aufgebaut. Die zur Regeneration eines Zeichens benötigte Zeit wird auch Zykluszeit genannt.

Dünnschichtspeicher sind Speichereinrichtungen von Automaten der dritten Generation.

Praktisch bewährt hat sich bereits der *Dünnschicht-Stäbchen-Speicher* (Bild 77) in der Zentraleinheit NCR 315 – RMC (Rod Memory Computer: Stäbchen-Speicher-Automat).

Stäbchen von 0,38 mm Durchmesser und 152 mm Länge werden mit einer magnetisierbaren dünnen Schicht versehen, über die wiederum Schreib- und Leseleitungen gelegt werden. Die Zykluszeit (Lese- bzw. Schreibzeit für ein Zeichen) beträgt hier 800 ns. Jedes Stäbchen hat ein Speichervermögen von 40 Bit. Insgesamt können zwischen 40 K und 160 K Byte gespeichert werden.

Bei eigentlichen Dünnschichtspeichern werden Keramikplatten im Vakuum mit einer 2 bis $20 \cdot 10^{-5}$ mm dicken Magnetschicht aus einer Ni-Fe-Legierung bedampft. Dieser Vorgang erfolgt in einem permanenten Magnetfeld, wodurch sich die Ni-Fe-Moleküle als Dipol in eine magnetische Vorzugsrichtung (harte Magnetrichtung) drehen. Die Dipollage in der harten Magnetrichtung entspricht der Speicherung einer O. Wird lokal eine Verdrehung des Dipols in die weiche Magnetrichtung bewirkt, so ist dort ein L gespeichert.

Beschrieben und gelesen werden kann über aufgedampfte Leitungen auf Keramikplatten, die auf die Dünnschichtplatten aufgelegt werden.

Derartige Dünnschichtspeicher haben eine Schaltzeit von etwa 1 ns und eine Packungsdichte, die der Monolithtechnik entspricht (10000 Bit/cm³). Erste Speicher wurden etwa ab 1963 von UNIVAC verwendet.

Auf weitere Speicherprinzipien (z. B. *Laufzeitspeicher*) und Speicherarten (z. B. *Assoziativspeicher*, bei denen nicht durch Adressen, sondern assoziativ durch Informationsvergleich abgerufen wird) kann hier nicht eingegangen werden. Wie bereits bei der technischen Beschreibung angedeutet, eignen sich Dünnschichtspeicher, Kernspeicher und Magnettrommelspeicher durch ihre Einzeladressierbarkeit als Register und Arbeitsspeicher. Sie können aber auch mit den anderen Magnetschichtspeichern als Zubringerspeicher peripher eingesetzt werden.

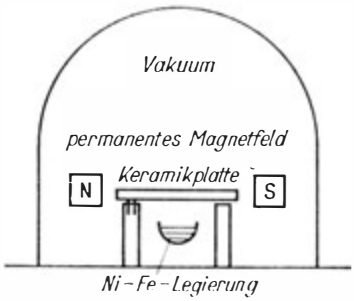


Bild 78. Vorgang beim Bedampfen einer Platte des Dünnschichtspeichers

3.3.5. Ein- und Ausgabegeräte

Ein- und Ausgabegeräte stellen unmittelbar die Verbindung zwischen der Umwelt und dem Automaten her.
Demzufolge benötigen wir für jeden der in 3.2.4. genannten Datenträger und für die manuelle Bedienung Ein- und Ausgabegeräte, also für

manuelle Bedienung,
Lochkarten,
Lochband,
Magnetband,
Magnetschrift und
Klartextbelege.

Wir können dabei direkte und indirekte Ein- und Ausgabegeräte unterscheiden.
Zur direkten Eingabe haben die Automaten meist eine Funktionstastatur, mit deren Hilfe Befehle und Daten manuell in den Automaten eingegeben werden können. Die analoge Ausgabeart ist ein optisch ablesbares Lampenfeld. Ein- und Ausgabegeschwindigkeit sind hier natürlich außerordentlich gering. Daher werden diese Möglichkeiten lediglich für Korrekturzwecke verwandt.
Moderne Automaten haben an Stelle oder zusätzlich zur oben angeführten direkten Ein- und Ausgabe Schreibmaschinen oder Blattschreiber (Schreibgeräte des Fernschreibers), deren Eingabegeschwindigkeit etwa 10 Zeichen/s beträgt, während die Ausgabe eine maximale Geschwindigkeit von 10···15 Zeichen/s erreicht.
Weitaus leistungsfähiger ist die Magnetschrift-Lese- und -Sortiermaschine NCR 407. Dieses Eingabegerät bewältigt 20 komplette Belege in der Sekunde.
Auch der optische Journalstreifenleser (Klartextleser) NCR 420 ist als direktes Eingabegerät sehr leistungsfähig. Er liest 52 Zeilen zu 32 Zeichen in der Sekunde. Allerdings muß der Journalstreifen mit den auf Bild 53 dargestellten Zeichen bedruckt sein.
Den Magnetschrift- und Klartextlesegeräten entsprechen als Ausgabegeräte die Schnelldrucker. Sie geben die Ergebnisdaten

Bild 79. Funktionstastatur und Schreibmaschine des **Cellatron** SER 2

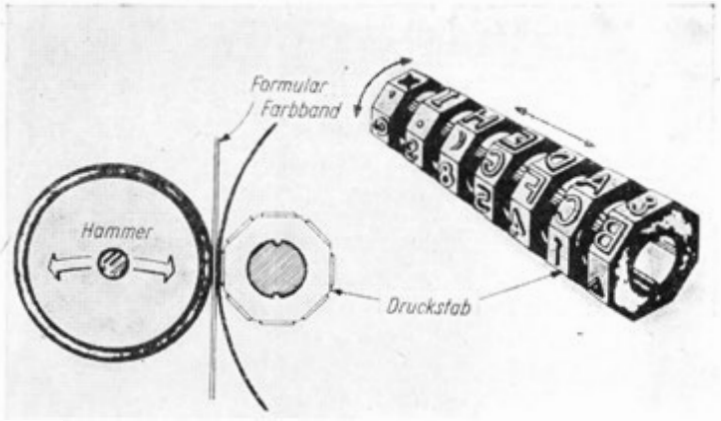


Bild 80. Prinzip eines Schnelldruckers mit Typenrädern

(output data) im Klartext aus. Ihre technische Ausführung ist verschieden.

Bei einer Ausführung sind die Druckzeichen auf dem Umfang eines Typenrades angebracht. Beim Druckvorgang schlägt ein Hammer gegen das Papier, wodurch die Type mit Hilfe eines Farbbandes auf das Papier übertragen wird. Bild 80 demonstriert diesen Vorgang. Nach diesem Verfahren können bis 3300 Druckzeichen in der Sekunde ausgegeben werden.

Der Schnelldrucker IBM 1443 benutzt statt der Typenräder Typenketten (Bild 81). Mit diesen Geräten können 8 Zeilen zu 132 Zeichen in der Sekunde gedruckt werden.

Bild 82 zeigt den für den Robotron 300 vorgesehenen Schnelldrucker, bei dem zwei Bahnen mit 156 Druckstellen parallel bedruckt werden können.

Diese Druckgeschwindigkeit kann durch ein anderes Verfahren wesentlich erhöht werden. Ein System erzeugt die zu druckenden Zeichen auf einem Bildschirm (ähnlich dem Fernsehgerät) und überträgt sie auf eine mit einer Selenschicht belegte Aluminium-

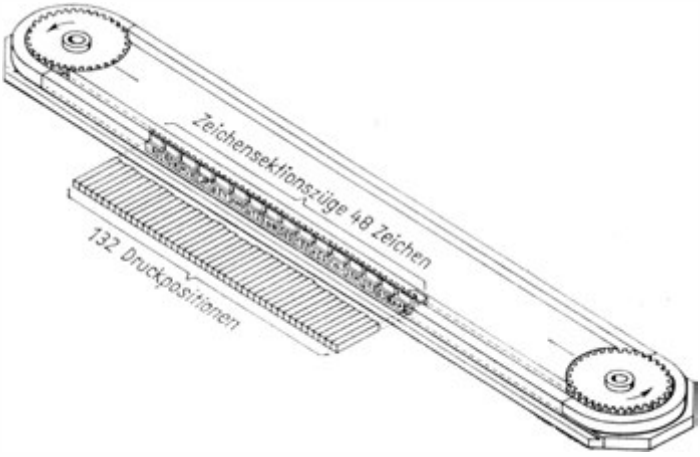


Bild 81. Schema der Arbeit eines Schnelldruckers vom Typ IBM 1443

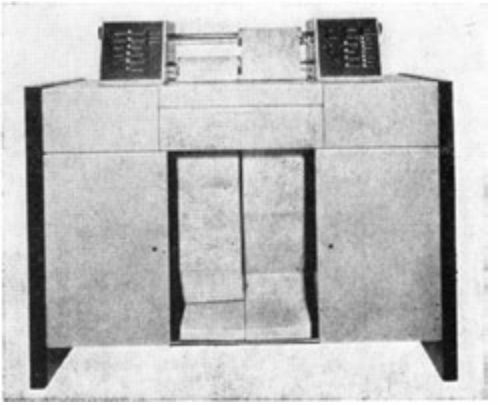


Bild 82. Paralleldrucker des Robotron 300

trommel, die zunächst positiv elektrisch aufgeladen ist. Bei Auftreffen eines Lichtstrahls auf diese Trommeloberfläche verliert die Schicht an der beleuchteten Stelle ihre Ladung. Die Druckzeichen der Bildröhre werden durch ein optisches Umlenkverfahren auf die Trommel gegeben und erzeugen dort „Ladungsbilder", die durch Bestäuben mit einem Spezialpuder sichtbar gemacht werden können. Das Pulver wird auf eine Papierbahn, wie beim Rotationsdruck, übertragen und durch kurze Wärmebehandlung eingebrannt. Das dadurch erhaltene Schriftbild ist sofort dauerhaft als Klartext zu lesen.

Nach diesem Verfahren können in der Größe einer Perlschrift 100000 Zeichen in der Sekunde gedruckt werden (SC 5000 High-Speed Elektronic Printer).

Das modernste direkte Ein- und Ausgabegerät ist der Katodenstrahl-Bildschirm (Bild 83). Derartige Geräte werden von EAC, IBM und anderen angeboten. Die Outputdaten können als Ziffern oder in grafischer Darstellung auf einem Bildschirm erzeugt werden, wie wir es vom Fernsehen kennen. Dabei werden Lesegenauigkeiten von 0,15 mm erreicht.

Interessant ist es, daß für dieses Ausgabegerät eine Lichtquelle als „Schreibstift" konstruiert wurde, der ein Auftragen von

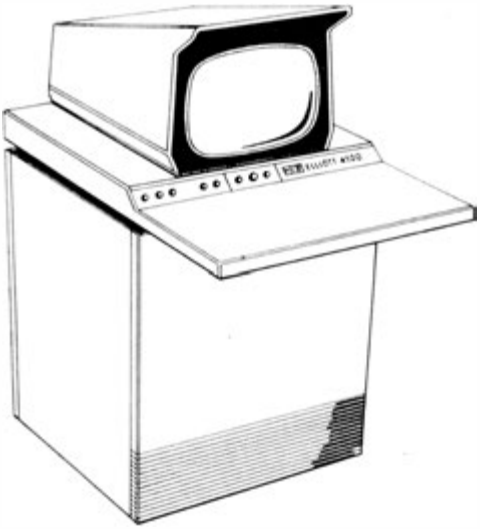


Bild 83. CRT-Katodenstrahl-Bildschirm der Fa. Elliott

Linien oder Zeichen auf den Bildschirm ermöglicht. Diese Zeichen können dann als Zusatzdaten rückgespeichert werden.
Neben diesen Direktgeräten haben wir noch indirekte Ein- und Ausgabegeräte.
Lochkartenleser und Lochkartenstanzer sind für nahezu alle Automatentypen im Angebot. Die Abtastung erfolgt meist fotooptisch, das Stanzen mechanisch. Daraus ergibt sich eine Lesegeschwindigkeit von 100 bis maximal 2000 Karten in der Minute. Die Stanzgeschwindigkeit erreicht Werte von 100 bis 300 Karten in der Minute.
Auch Streifenleser und -stanzer gibt es für jeden Automaten. Das Stanzen erfolgt mechanisch, wobei Geschwindigkeiten von 150 bis maximal 300 Zeichen in der Sekunde erreicht werden.
In der Forschung wird an einem Streifenlocher mittels Funkenentladung gearbeitet. Der zu lochende Streifen läuft über eine Matrize. Oberhalb derselben befindet sich ein Elektrodensystem, das aus zwei Hauptelektroden und einer dazwischen-

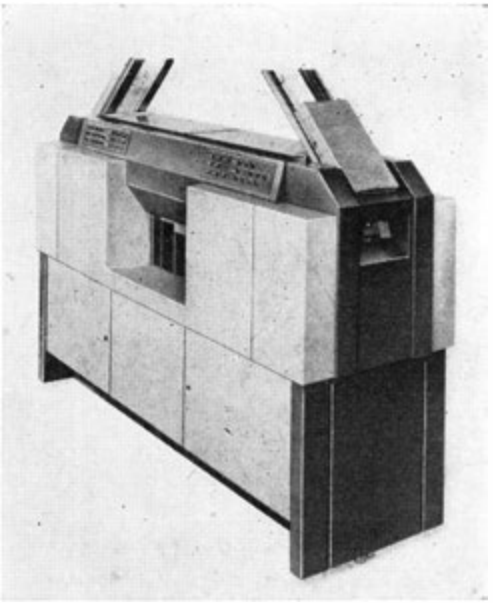
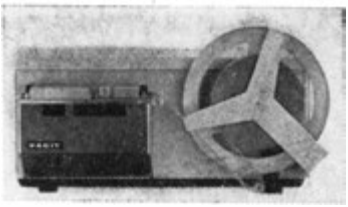


Bild 84. Karten-Lese-Stanz-Einheit des Robotron 300

liegenden Zündelektrode besteht. Über einen Kondensator wird an die Hauptelektroden eine Spannung gelegt, die sich im gewünschten Moment durch die Einwirkung des Zündkontaktes als Funken entlädt. Dadurch entsteht zwischen den Elektroden und dem Papierband kurzzeitig ein Druck von mehreren tausend Atmosphären. Dieser Druck reicht aus, um eine kreisförmige Papierscheibe in das darunterliegende Matrizenloch zu drücken. Dieser Vorgang geht so schnell vor sich, daß die Funkenwärme das Papier nicht einmal ansengen kann. Mit diesem Verfahren kann eine einwandfreie Lochung auch noch bei einem Papiervorschub von 25 m/s erreicht werden.

Das Lesen des Lochstreifens erfolgt mechanisch (geringe Eingabegeschwindigkeit), fotooptisch und kapazitiv. Dadurch lassen sich Lesegeschwindigkeiten von 100 bis 1000 Zeichen/s



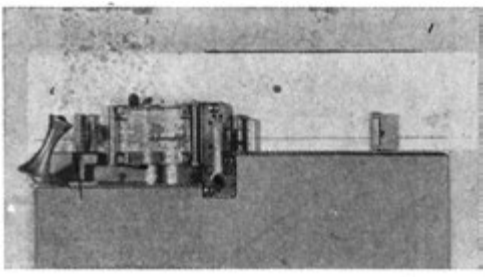


Bild 85. Lochbandleser und -stanzer

(im Extremfall 2000 Zeichen/s) erzielen. Interessant ist das kapazitive Lesen. Es beruht darauf, daß der Papierstreifen durch eine Zahl von Kondensatoren geführt wird. Da Papier und Luft (der Lochung) unterschiedliche Dielektrizitätskonstanten haben, ergeben sich klare Leseimpulse.

Das Magnetband wird meist nur zur Extremspeicherung von Programmen, Konstantenfeldern und Zwischenwerten, die bereits einmal im Automaten waren, benutzt. Hier sind Lese- und Schreibgeschwindigkeiten von 80000 bis 100000 Zeichen/s üblich.

In den letzten Jahren wurden leistungsfähige digitalgesteuerte Zeichengeräte entwickelt und eingesetzt. Mit diesen Geräten ist es möglich, unmittelbar nach den ermittelten Outputdaten grafische Darstellungen anzufertigen. Allerdings müssen dabei die Outputdaten in Befehle für den Zeichner umgerechnet werden.

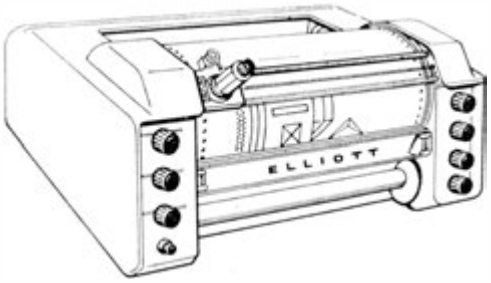


Bild 86. Digitaler Kurvenschreiber Modell Elliott 4290

Die Befehlsmenge umfaßt:

positive Abszisse +X (Trommel vorwärts),
negative Abszisse —X (Trommel rückwärts),
positive Ordinate +Y (Feder links),
negative Ordinate —Y (Feder rechts),
Feder aufsetzen und
Feder anheben.

Die Güte der Zeichnung hängt von der Schrittlänge ab. Sie beträgt 0,25 mm bis 0,12 mm. Die Arbeitsgeschwindigkeit liegt zwischen 200 und 300 Schritten in der Sekunde. Die Zeichnungsgrößen sind unterschiedlich nach Geräteausführung.
Wir können digitale Kurvenschreiber (Bild 86) und Planzeichner unterscheiden. Einmal wird die Zeichnung auf einer Zeichentrommel hergestellt. Die Trommeldrehung entspricht der X-Verschiebung, die Feder wird lediglich in Y-Richtung verschoben.
Planzeichner fertigen die Zeichnung auf einer Planplatte an. Die Feder wird dabei in der X- und Y-Richtung bewegt.

3.3.6. Geräte zur Datenfernübertragung

In zunehmendem Maße erhält die Datenfernübertragung für die moderne Rechentechnik Bedeutung. Die eigentliche Übertragung erfolgt über das Fernschreib- oder Fernsprechnetz oder bereits über Funk.

Der technische Ablauf vollzieht sich in den nachfolgend angeführten Schritten:

Ein Sender gibt den Dualimpuls an einen Modulator.
Hier erfolgt die Signalumsetzung entsprechend dem verwendeten Modulationsverfahren.
Darauf wird das Signal über zwei Drähte übertragen, wobei verschiedene Betriebsarten Verwendung finden.
Ein Demodulator setzt das Signal wieder in einen Dual-

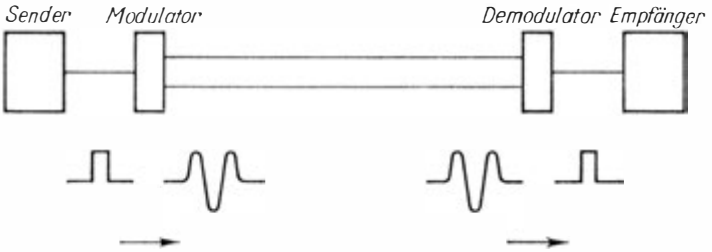


Bild 87. Aufbau eines Datenfernverarbeitungssystems (Zweidraht Simplexbetrieb)

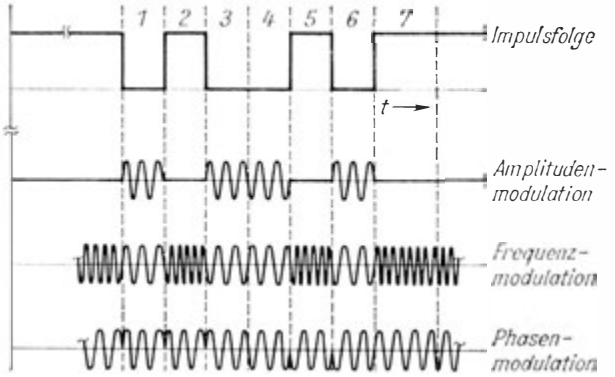


Bild 88. Modulationsverfahren

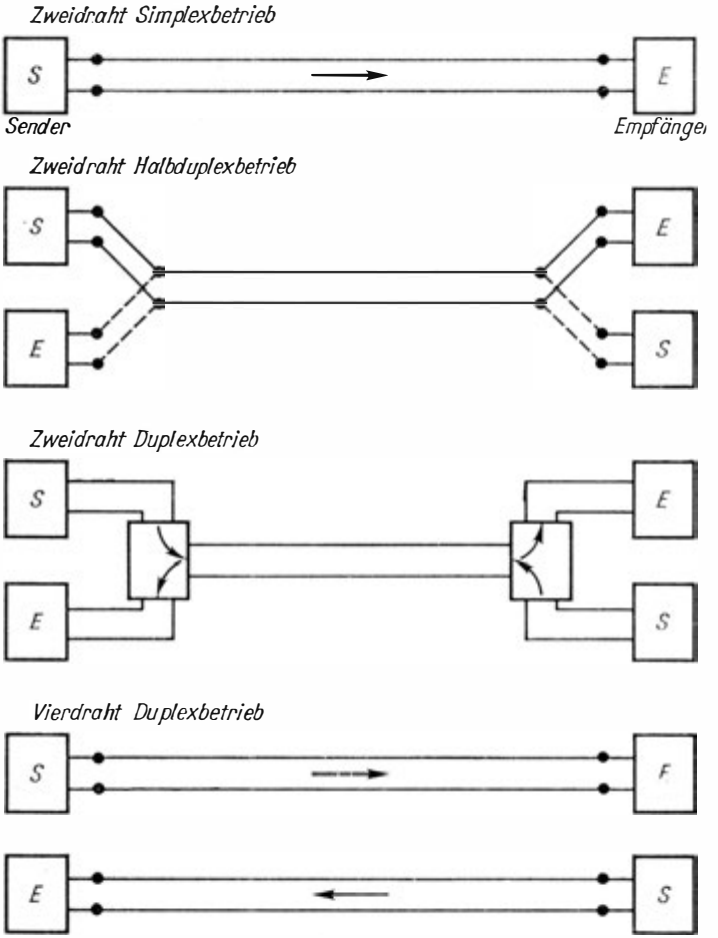


Bild 89. Betriebsarten bei Datenfernübertragung

impuls um, der
vom Empfänger zur weiteren Verarbeitung übernommen wird.

An Modulationsverfahren werden drei verwendet, wie sie auf Bild 88 dargestellt sind.
Dabei können folgende Übertragungsgeschwindigkeiten erreicht werden, bei

Amplitudenmodulation (billigstes Verfahren, aber störanfällig)	10 ··· 100 Baud	~ 10 Zeichen/s
Frequenzmodulation	200 ··· 1200 Baud	~ 150 Zeichen/s
Phasenmodulation	2400 ··· 4800 Baud	~ 510 Zeichen/s

Nach der Betriebsart unterscheidet man (Bild 89)

Zweidraht Simplexbetrieb, bei dem die Daten nur in einer Richtung übertragen werden;
Zweidraht Halbduplexbetrieb, bei dem die Daten wahlweise in einer der beiden Richtungen übertragen werden;
Zweidraht Duplexbetrieb, bei dem die Daten gleichzeitig in beiden Richtungen übertragen werden. Sie dürfen aber nur je einen Teil des Frequenzbereiches benutzen;
Vierdraht Duplexbetrieb mit gleichzeitiger Datenübertragung in beiden Richtungen und im vollen Frequenzbereich.

Bereits beim Halbduplexbetrieb werden Signalumsetzer benötigt, die sowohl modulieren als auch demodulieren können. Sie werden *Modem* genannt.
Von den Firmen IBM, ICT und anderen werden Anschlußeinheiten an den Rechner angeboten, die eine Datenfernübertragung über Fernschreib- und Telegrafenleitungen mit 10 Zeichen/s und Fernsprechleitungen mit 100 Zeichen/s ermöglichen.
Kurzstreckige Datenübertragungen auf Spezialleitungen können dabei noch höhere Übertragungsgeschwindigkeiten erreichen.

3.4. Datenverarbeitungssysteme

Ein arbeitsfähiger Rechenautomat benötigt ein Rechenwerk, Arbeitsspeicher und ein Leitwerk mit Kommandopult, zu dem eine einfache Ein- und Ausgabe (als Tastatur oder Schreibmaschine) gehört. Diese Aggregate werden bei einer Datenverarbeitungsanlage zur Zentraleinheit zusammengefaßt.

Die *Zentraleinheit* ist das Kernstück der Datenverarbeitungsanlage. Sie ist praktisch „für sich" arbeitsfähig. Selten jedoch gibt es ausreichend viel Aufgaben, die allein mit der Zentraleinheit lösbar sind. Daher kann sie durch ein der Aufgabenstellung angepaßtes Sortiment peripherer Geräte, wie Zubringerspeicher, Ein- und Ausgabegeräte, Datenfernübertragungsanlagen, ergänzt werden.

Die Datenverarbeitungssysteme oder -familien umfassen eine Anzahl austauschbarer, unterschiedlich leistungsfähiger Zentraleinheiten, die der Aufgabenstellung und der Entwicklung des Einsatzbereiches angepaßt werden können.

Als Beispiel wollen wir das Sortiment an Zentraleinheiten des ICT-Systems 1900 anführen (Tabelle 8).

Die peripheren Geräte können einmal unabhängig von der Zentraleinheit arbeiten (off-line) und so nur mittelbar über einen Datenträger mit ihr verbunden sein.

Sie können aber über die Datenkanäle auch unmittelbar an die Zentraleinheit angeschlossen werden (on-line). Hierbei ergeben sich durch die unterschiedliche Arbeitsgeschwindigkeit der einzelnen Geräte Schwierigkeiten im Zusammenschluß.

ICT versucht dies, durch unterschiedliche Datenkanäle für die langsam arbeitende Peripherie, wie Lochkartenleser und -stanzer, Lochbandleser und -stanzer, sowie schnelle Datenkanäle für die Speicher u. a. zu erreichen. So lassen sich

je ein langsames peripheres Gerät oder
mehrere Blattschreiber oder
bis 64 optische Anzeigegeräte oder
4 Magnettrommelspeicher oder
4 Wechselplattenspeicher oder
1 Großplattenspeicher oder
4 Magnetkartenspeicher oder
12 Magnetbandeinheiten oder
1 Multiplexer für Datenfernübertragung auf maximal 63 Fernleitungen

an einen entsprechenden Kanal anschließen.

Zudem gibt es bei ICT die *Datenaustauschsteuereinheit* 1995, die einen direkten Informationsaustausch zwischen mehreren ICT-

	1902	1903	1904	1905	1906	1907	1909
Wortzeit µs	6	2	2	2	1.1 ··· 2.1	1.1 ··· 2.1	6
Rechenzeit Gleitkommaarithmetik	nein	nein	nein	ja	nein	ja	ja
Festkomma Add/Subtr	18 µs	7 µs	7 µs	7 µs	2.5 µs	2.5 µs	18 µs
Multipl.	1.5 ms	650 µs	40 µs	40 µs	11.25 µs	11.25 µs	67 µs
Division	2.3 ms	900 µs	44 µs	44 µs	18 µs	18 µs	71 µs
Sprung	13 µs	5 µs	5 µs	5 µs	2.5 µs	2.5 µs	13 µs
Gleitkomma Add/Subtr	1.15 ms	475 µs	111 µs	13 µs	25 µs	2.75 µs	21 µs
Laden	—	—	—	6 µs	6.25 µs	2.5 µs	18 µs
Speichern	—	—	—	8 µs	6.25 µs	2.5 µs	18 µs
Multipl.	5.25 ms	2.25 ms	285 µs	29 µs	60 µs	7.75 µs	37 µs
Division	9.6 ms	3.85 ms	316 µs	51 µs	65 µs	16.75 µs	59 µs
Adreßmodifikation	6 µs	2 µs	2 µs	2 µs	1.25 µs	1.25 µs	6 µs
Skalarprodukt	6.5 ms	2.8 ms	435 µs	60 µs	103.75 µs	20.25 µs	112 µs
Polynomentwicklung	6.4 ms	2.7 ms	403 µs	42 µs	88.75 µs	10.5 µs	58 µs
Kernspeicherworte	4.096	8.192	8.192	8.192	32.768	32.768	16.384
1 Wort = (4 alpha-num.	8.192	16.384	16.384	16.384	bis	bis	32.768
Zeichen) 24 Bit + 1 Paritäts-	16.384	32.768	32.768	32.768	262.144	262.144	
bit							

	1902	1903	1904	1905	1906	1907	1909
Multiprogramming	begrenzt	möglich			möglich		
Hauptprogramme	1	1	4	4	16	16	4
Beiprogramme	(2)	(2)	2	2	3	3	2
gesamt	(3)	(3)	12	12	64	64	12
Datenkanäle							
langsam	—	—	18	18	18	18	18
schnell	—	—	5	5	nach	Bedarf	5
universell	8	8	—	—	—	—	—

Tabelle 8. Leistungsdaten der Zentraleinheiten des ICT-Systems 1900

Zentraleinheiten ermöglicht. Dadurch können an eine hochleistungsfähige Zentraleinheit mehrere Zentraleinheiten mit ihrer Peripherie als Satellitenrechner angeschlossen werden, wodurch sich neue Arbeitsmöglichkeiten in der Datenverarbeitung ergeben.

Bei anderen Datenverarbeitungssystemen hat man gepufferte, standardisierte Schnittstellen (standard interface) entwickelt, durch die ein direkter Anschluß aller Geräte über einfache Steckerverbindungen hergestellt werden kann.

Die Anlagen der Systeme sind dabei so ausgebaut, daß ein Simultanbetrieb möglich ist.

Simultanarbeit heißt, daß gleichzeitig verschiedene Arbeiten der Einzelgeräte ausgeführt werden können, beispielsweise Rechnen, Übertragen von Datenblöcken aus den Zubringerspeichern in die zum Rechnen gerade nicht benötigten Bereiche des Arbeitsspeichers, Ausgabe vom Zubringerspeicher über ein Ausgabegerät und je nach Anlagentyp noch andere Arbeitsprozesse. Dieser Simultanbetrieb muß natürlich vorher im Programm erfaßt sein. Ein geschickter Programmierer kann hier jedoch enorme Zeiteinsparungen erzielen.

Viele Anlagen eines Systems sind zudem für *Multiprogramming* (*time sharing*) ausgelegt. Hierbei können gleichzeitig mehrere Programme mit einer Vorrangbewertung zur Bearbeitung eingegeben werden. Über ein sogenanntes *Monitorprogramm* wird die Arbeit mit den Programmen entsprechend deren Vorrang organisiert. Dabei ist die Zentraleinheit immer in Arbeit. Ergeben sich bei einem Programm durch Datentransport oder andere Prozesse Verzögerungen, wird ein Programm niederer Vorrangstufe abgearbeitet.

Als Beispiel seien nur zwei Programme betrachtet: Tagesbilanz und Betriebsabrechnung, deren Vorrangstufe der aufgeschriebenen Reihenfolge entsprechen soll. Für die Tagesbilanz werden jedoch von einer Stufe an Daten benötigt, die erst ab 15.00 Uhr zur Verfügung stehen. Dann würde bei time sharing die Arbeit am Programm Tagesbilanz unterbrochen, bis die Daten vorliegen. Während der „Wartezeit" arbeitet die Anlage nach dem Programm „Betriebsabrechnung".

Die Arbeitsweise im time sharing oder Multiprogramming ermöglicht eine bedeutende Leistungssteigerung einer Daten-

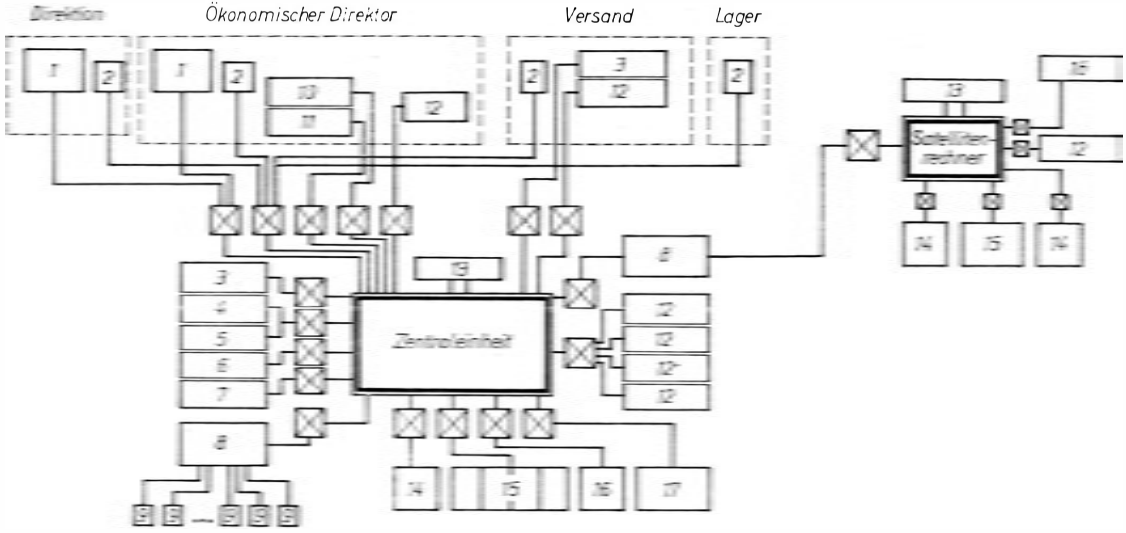


Bild 90. Modell des Einsatzes einer Prozeßdatenverarbeitungsanlage in einem Kaufhaus

1 optische Datenanzeige
2 Blattschreiber
3 Kartenleser und -stanzer
4 Streifenleser
5 Streifenstanzer
6 Klartextleser
7 Magnetschriftleser
8 Datenaustauschsteuereinheit
9 Registrierkassen
10 Lochbandleser
11 Lochbandstanzer
12 Schnelldrucker
13 Kommandopult
14 Plattenspeicher
15 Magnetbandspeicher
16 Großplattenspeicher
17 Magnet-Kartenspeicher
18 Lochbandleser und -stanzer

verarbeitungsanlage. Sie verlangt jedoch zusätzlich den Anschluß eines *Programmzeitmessers*, der die an jedem Programm effektiv gearbeitete Zeit über die Kontrollschreibmaschine des Kommandopultes ausgibt. Das Monitorprogramm veranlaßt zudem die Ausgabe eines Funktionsprotokolls.
Vielfach wird zusätzlich eine Digitaluhr eingesetzt. Das Funktionsprotokoll wird so durch Angabe des Beginn- und Endtermins der Arbeit an jedem Teilprogramm nach Ortszeit ergänzt.
Schließlich ist mit modernen Datenverarbeitungsanlagen Echtzeitbetrieb (real-time-processing) möglich, wie wir ihn bei Prozeßrechnern besprochen haben.
Man spricht dann auch von Prozeßdatenverarbeitungsanlagen, die mit real-time-processing und time sharing als komplexe oder integrierte Systeme der Datenverarbeitung in Kaufhäusern, Versandhäusern, Banken, im Flugwesen u. a. eingesetzt werden. Bild 90 soll dies für einen Modellfall „Kaufhaus mit Versand" zeigen.

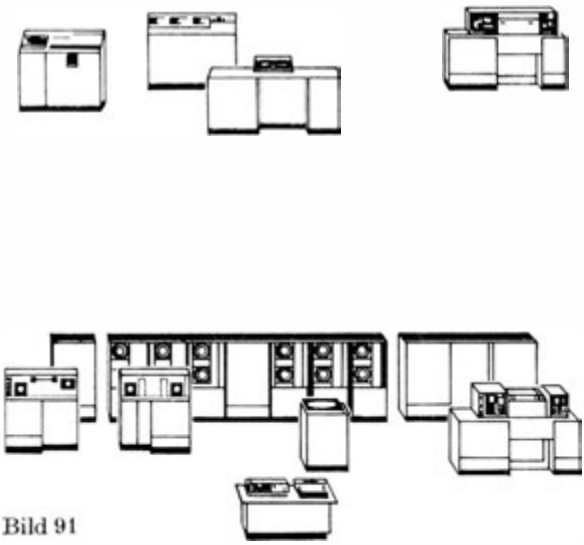


Bild 91

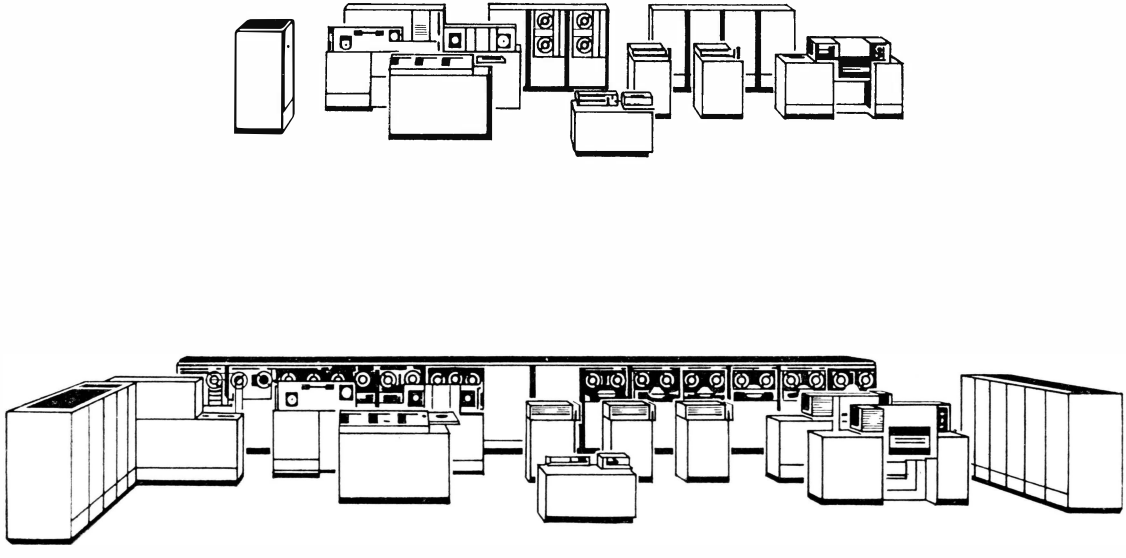


Bild 91. System 4 der Fa. English Electric Leo Marconi in den Ausführungen 4/10, 4/30, 4/50 und 4/70

Anschließend wollen wir einige Datenverarbeitungssysteme oder -familien anführen, wobei es sich lediglich um eine Auswahl handeln kann:

URAL 10	mit URAL 11, 12, 14, 16, 17, 18
IBM 360	mit 360/30, 40, 50, 60, 62, 70, 72, 90
Elliott EAC 4100	mit 4120, 4130
Honeywell H 200	mit H 120, 200, 1200, 2200, 4200, 8200
Siemens 4004	mit Si 4004/15, 25, 45, (65, 75)

Als Beispiel der Ausbaufähigkeit sei das System 4 der English Electric Leo Marconi auf Bild 91 dargestellt.
Alle Geräte sind natürlich voll austauschfähig (kompatibel).

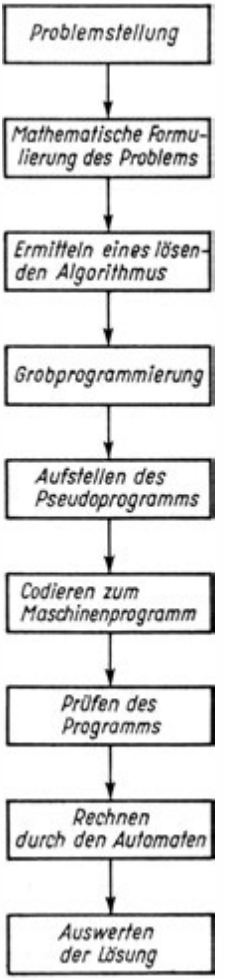
3.5. Programmieren für Digitalrechner

Eine starke Entwicklung können wir auch in der Programmierungstechnik beobachten. Sie reicht von der Programmierung in der Maschinensprache bis zur modernen „software", die als Programmierungshilfe mit der Technik einer Anlage verkauft wird. Vielfach gibt diese software bei Entscheidungen über die Auswahl eines bestimmten Automatentyps letztlich den Ausschlag.
Leider würde es den Rahmen dieses Buches sprengen, wollten wir diese interessante Entwicklung in all ihren Einzelheiten verfolgen. Wir müssen uns auf einige Überlegungen beschränken und wollen uns bei der Auswahl davon leiten lassen, welche Programmierungshilfsmittel für uns Bedeutung haben.
Zum besseren Verständnis werden wir dazu erst einmal den gesamten Bearbeitungsprozeß einer allgemeinen Aufgabenstellung betrachten, wie er unter „klassischen" Bedingungen verläuft.
Gehen wir von einem allgemeinen Sachverhalt aus, so ergeben sich folgende Bearbeitungsschritte:

Aufbereitung mit

Problemstellung,
mathematischer Formulierung,
Ermittlung eines lösenden Algorithmus

Bild 92. Ablaufdiagramm des Lösungsprozesses



Programmierung mit

Grobprogrammierung in einer allgemeinen Form (Strukturdiagramm, Algol usw.),
Aufstellen des Pseudoprogramms,
Codieren zum Maschinenprogramm,
Prüfen des Programms

Rechnung durch den Automaten,
Auswertung des Ergebnisses.

3.5.1. Aufbereiten

Wir wissen bereits, daß ein vorliegender Sachverhalt der Technik, Technologie, Ökonomie oder einer anderen Disziplin nicht unmittelbar rechnerisch bearbeitet werden kann. Wir müssen den Sachverhalt erst so aufbereiten, daß er der numerischen Bearbeitung zugänglich ist.
Dabei muß als erstes der Kern des Sachverhaltes herausgearbeitet werden, der in der Problemstellung seinen Niederschlag findet. Hierzu sind gegebenenfalls umfangreiche Analysen, Recherchen und Berechnungen erforderlich. Die Erarbeitung der Problemstellung kann also durchaus selber ein Bearbeitungsprozeß sein.

Die *Problemstellung* ist die präzise Erfassung der Aufgabe, die sich aus dem allgemeinen Sachverhalt ergibt. Sie wird meist als Abhandlung vorgelegt, die durch Tabellen, statistische Auswertungen und vereinzelte mathematische Formeln untermauert ist. Wünschenswert wäre es dabei, wenn der gesamte zu erfassende Vorgang bereits durch ein Ablaufschema ergänzt würde. Das Ablaufschema kann dabei recht grob gehalten werden, wie beispielsweise Bild 92 für den Bearbeitungsprozeß selbst zeigt. Insgesamt muß es die Problemstellung ermöglichen, den nächsten Arbeitsschritt, die mathematische Formulierung oder das Aufstellen des mathematischen Modells auszuführen. Zudem muß auch die Gesamtorganisation von der Bereitstellung der Eingangsdaten *(input data)* bis zur Anordnung der Ausgangsdaten *(output data)* erkennbar sein.

Aus der Problemstellung ergibt sich die *mathematische Formulierung – das mathematische Modell.* Dabei werden wir vorerst einmal versuchen, unsere Aufgabe in der Problemstellung so zu erfassen, daß wir ein bereits bekanntes mathematisches Modell unmittelbar oder mit geringen Änderungen verwenden können.

Vorhandene Modelle werden auch gern zur Ermittlung einer ersten Näherungslösung angewandt. Damit können wir erreichen, daß möglichst schnell quantitative Aussagen vorliegen, die bereits erste Schlüsse zulassen, daß Erfahrungen gesammelt werden können und daß der gesamte Lösungsprozeß verkürzt wird. Diese erste Näherungslösung wird dann, wenn ein solches Vorgehen überhaupt möglich ist, schrittweise verbessert, bis das Modell alle Forderungen der Praxis erfaßt und komplex angewandt werden kann.

Aus dieser Verfahrensweise wird bereits ersichtlich, daß in den ersten Bearbeitungsschritten der Praktiker den Vorrang hat, wobei er allerdings auch mit dem Mathematiker zusammenarbeiten sollte.

Liegt das mathematische Modell vor, wir werden uns damit noch ausführlich im Abschnitt 6. beschäftigen, gilt es, den *lösenden Algorithmus* zu finden.

Wir erinnern uns, daß mit der mathematischen Formulierung noch keineswegs die Aufbereitung abgeschlossen ist. Beispielsweise ist das Ziehen einer Quadratwurzel kein elementarer

Prozeß. Wie wir im Abschnitt 2.1. gesehen haben, gibt es dafür einen numerischen Algorithmus mit der Vorschrift

$$u_n + 1 = \frac{1}{2}\left(u_n + \frac{R}{u_n}\right)$$

die für jeden Radikanden $O \leqq R$ und jede Ausgangsnäherung $O < u_0$ konvergiert.
Diese Vorschrift nennt man auch formelmäßige Lösung, nach der dann die zahlenmäßige Lösung errechnet werden kann.
Wir müssen somit zwischen der formelmäßigen Bezeichnung oder dem Namen und dem zahlenmäßigen Wert einer Größe unterscheiden.
Der Algorithmus gibt uns an, wie aus Werten, die durch Bezeichnungen ausgegeben sind, neue Werte berechnet werden, die man wiederum durch Bezeichnungen festlegt.
Zur Darstellung verwendet man die *Plangleichung*, bei der wir durch das Zeichen ⇒ (sprich ergibt) unmittelbar ersehen, aus welchen Werten der neue Wert ermittelt werden soll.
Das Ergibtzeichen wird jedoch auch durch folgendes Symbol dargestellt: :=, wobei der Doppelpunkt der Pfeilspitze entspricht.
Plangleichungen werden vielfältig benutzt. Wir wollen uns ihre Anwendung an einigen Beispielen erarbeiten.
Die Verwendung bei Operationen wird uns wenig Schwierigkeiten bereiten. Durch die Plangleichung

$$f(x_1, x_2, x_3, \ldots, x_n) \Rightarrow y \quad \text{oder} \quad y := f(x_1, x_2, x_3, \ldots, x_n)$$

bringen wir zum Ausdruck, daß die Werte der durch $x_1, x_2, \ldots x_n$ bezeichneten Größen entsprechend der Vorschrift f zu neuen Werten verknüpft werden sollen, die wir mit y bezeichnen. Beispielsweise sollen bei

$$a + b \Rightarrow c \quad \text{oder} \quad c := a + b$$

die Werte der Größen a und b durch die Vorschrift Addition verknüpft werden, wobei das Ergebnis die Bezeichnung c erhält. Schwieriger wird uns die Vorstellung der Vorschrift

$$a + b \Rightarrow a \quad \text{oder} \quad a := a + b.$$

Hier sollen die Werte der Größen a und b durch die Addition verknüpft werden, wobei der Wert des Ergebnisses der durch a bezeichneten Größe zugeordnet wird. Diese Vorschrift benötigen wir z. B. für die skalare Multiplikation zweier Vektoren. Sei $\mathfrak{a} = (a_1, a_2, a_3,)$ und $\mathfrak{b} = (b_1, b_2, b_3)$ gegeben, so ist

$$\mathfrak{a} \cdot \mathfrak{b} = (a_1, a_2, a_3) \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3.$$

Dafür schreiben wir, wenn wir noch $s_0 = 0$ als Ausgangssumme setzen

$$s_i = s_{i-1} + a_i \cdot b_i \; (i = 1, 2, 3)$$

oder unter Verwendung der Plangleichung

$$s + a \cdot b \Rightarrow s \text{ oder } s := s + a \cdot b,$$

wobei wir allerdings für die organisatorischen Anweisungen zusätzliche Vereinbarungen treffen müssen.
Diese Darstellung bewirkt mit s als Zwischensumme folgende Berechnung:

0. Schritt $s_0 := 0$
1. Schritt $s_1 := s_0 + a_1 \cdot b_1$
2. Schritt $s_2 := s_1 + a_2 \cdot b_2$
3. Schritt $s_3 := s_2 + a_3 \cdot b_3$

Auch für folgende Festlegungen verwenden wir Plangleichungen:

$$\begin{aligned} 1 &\Rightarrow n & n &:= 1 \\ 0 &\Rightarrow s & s &:= 0 \\ n + 1 &\Rightarrow n & n &:= n + 1 \end{aligned}$$

In den ersten beiden Fällen werden vorgegebene Werte einer Benennung zugeordnet. Damit wird praktisch die Definition des Wertes einer Größe vorgenommen. Im letzteren wird der Wert der Größe n um 1 erhöht und dann wieder n zugeordnet.
Plangleichungen können auch für mehrere Größen verwendet werden, wie es beispielsweise in der Matrizenrechnung üblich ist. Die Ermittlung des lösenden Algorithmus wird im allgemeinen vom Mathematiker vorgenommen, wenn nicht übliche Verfahren Anwendung finden.

Wir werden hierzu noch einige Betrachtungen beim Programmieren anstellen.
Liegt der Lösungsalgorithmus vor, ist die Phase der Aufbereitung abgeschlossen. Diese Phase ist vorerst noch völlig unabhängig vom vorhandenen Automaten. Allerdings sollte man sich hier bereits von Mitarbeitern eines Rechenzentrums beraten lassen, um zu großen Aufwand und gegebenenfalls Doppelarbeit zu vermeiden.

3.5.2. Programmieren

Der Programmierer löst die Aufgabe, alle nun bereits im lösenden Algorithmus vorliegenden Rechnungen und Anweisungen in Anweisungen für einen speziellen Rechenautomaten umzuformen. Die Gesamtheit der Anweisungen für einen Rechenautomaten zur Lösung eines Problems wird als Programm bezeichnet.
Die Kunst, ein Programm zur Lösung eines speziellen Problems mit einem bestimmten Rechenautomaten aufzustellen, besteht darin, den Gesamtprozeß so in einzelne Bestandteile zu zerlegen, daß diese Einzeloperationen mit den in der Befehlsliste des Automaten verzeichneten Operationen übereinstimmen.
Dazu muß das Programm in eine speziell verschlüsselte Sprache übersetzt werden, die der Automat versteht. Man nennt sie die Maschinensprache.
Die Maschinensprache ist für jeden Rechenautomatentyp verschieden. Das direkte Programmieren in der Maschinensprache ist erfahrungsgemäß sehr schwer. Daher erfolgt die Aufstellung des Maschinenprogramms in den bereits angedeuteten Teilschritten:
Grobprogrammierung – Aufstellen des Pseudogramms – Codieren zum Maschinenprogramm – Prüfen des Programms.
Es gilt zudem ein „gutes" Programm herzustellen, das nur wenige Speicherzellen benötigt und mit einer möglichst kurzen Rechenzeit auskommt.
Zum Programmieren brauchen wir Kenntnisse des mathematischen Sachverhaltes für das zu lösende Problem, wie sie bei der Aufbereitung anfallen. Aber wir müssen auch Bescheid wissen über die Eigenheiten des speziellen Automatentyps, für den wir programmieren wollen.

Die Kenntnisse des mathematischen Sachverhaltes stellen wir in einer algorithmischen Form dar. Das kann in verschiedener Weise geschehen. Vorerst wollen wir jedoch sogenannte Strukturdiagramme verwenden.

Die Eigenheit des digitalen Rechenautomaten zeigt sich für den Programmierer in

der Darstellung der Wörter (Zahlen und Befehle),

im Prinzip der Eingabe, Speicherung und Abarbeitung der Wörter sowie im Befehlssystem.

Durch die *Grobprogrammierung* umgehen wir die Schwierigkeit, Programme gleich in der Maschinensprache schreiben zu müssen.

Wir können vorerst noch eine Sprache verwenden, die der menschlichen Ausdrucksweise näher liegt und damit anschaulicher ist. Hier haben die spezifischen Besonderheiten eines speziellen Rechenautomatentyps vorerst nur geringen Einfluß, woraus sich der wesentliche Vorteil ergibt, daß die Programme in den Sprachen der Grobprogrammierung austauschbar und mit geringen Änderungen für viele andere Automatentypen verwendbar sind.

Für die Grobprogrammierung können wir voraussetzen, daß ein Algorithmus vorliegt.

Im Prozeß der Grobprogrammierung wird der Algorithmus mit Einschluß aller organisatorischen und sonstigen Teilprozesse in einer solchen Form dargestellt, daß danach unmittelbar das Pseudogramm aufgestellt werden kann.

Nun zur eigentlichen Sprache der Grobprogrammierung selber. Für sie gibt es viele Ausführungsformen. Um uns hierüber Vorstellungen zu verschaffen, wollen wir vorerst eine Sprache betrachten, die einmal sehr anschaulich ist, weil sie sich grafischer Hilfsmittel bedient und zudem durch ihre Vorzüge für das manuelle Programmieren starke Verbreitung fand. Es handelt sich um die Darstellungsform als *Flußdiagramm*, die von dem 1957 in den USA verstorbenen ungarischen Mathematiker *John v. Neumann* bereits in den ersten Jahren des Einsatzes von Rechenautomaten entwickelt wurde.

Wie bei jeder Sprache müssen wir eine Anzahl Grundzeichen definieren sowie die Regeln, wie aus denselben Worte, Sätze, Ab-

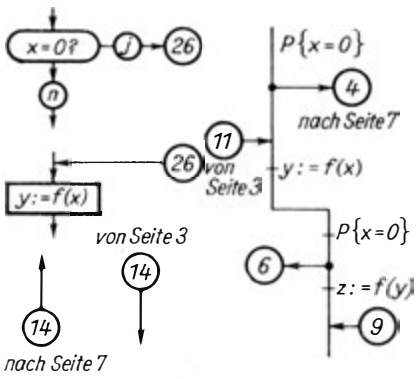
schnitte und Kapitel aufgebaut werden können. Für diese Zeichen und Regeln benötigen wir einmal ihre Anwendungsform – die Syntax – sowie ihre Bedeutung – die Semantik.

Definieren wir das Flußdiagramm selbst:

Ein Flußdiagramm ist eine detaillierte grafische Darstellung eines Programmverlaufes nach einem lösenden Algorithmus. Es stellt die Verbindung zwischen dem mathematischen Modell mit seinen Lösungsalgorithmen und dem Maschinenprogramm dar.
Dabei unterscheiden wir Programmablaufpläne für die Darstellung des eigentlichen Programms und Datenflußpläne zur Erfassung der Gesamtorganisation.
Wir benutzen hier einmal eine algorithmische Formelsprache, wie wir sie in der Aufbereitung bei der Einführung der Plangleichung mit dem Ergibtzeichen kennengelernt haben. Zum anderen deuten wir den Programmverlauf durch eine grafische Darstellung an.
Die algorithmische Formelsprache müßten wir nun ebenfalls eindeutig definieren. Vorerst verlassen wir uns jedoch auf unsere Kenntnisse im Umgang mit mathematischen Formeln.
Für die grafische Darstellung wurde von John v. Neumann die Kästchenmethode vorgeschlagen. Alle Formeln, Anweisungen und sonstigen Festlegungen werden in Kästchen eingeschlossen und der Gesamtfluß durch Flußlinien erfaßt. Diese Art erfordert jedoch einen großen Zeitaufwand. Daher hat sich in den letzten Jahren immer stärker die Leitlinienmethode durchgesetzt. Der Entwurf des DDR-Standards „Sinnbilder für Datenfluß- und Programmablaufpläne" führt die Symbole beider Methoden gleichberechtigt an.
Wir wollen uns auf eine Auswahl von Symbolen beschränken und verweisen auf den genannten Standard (Tabelle 9).
Bereits mit dieser Auswahl können wir komplizierte Programme aufbauen. Jedes Programm beginnt mit dem Organisationskästchen „START" und endet in einem oder mehreren Kästchen „HALT" bzw. „STOP" bei Sonderfällen. Dazwischen liegt das eigentliche Programm, bei dem wir drei Grundformen unterscheiden:

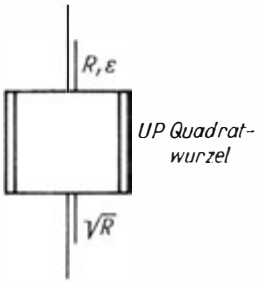
Sinnbilder

Kästchenmethode	Leitlinienmethode	Bedeutung	Erklärung



Feste Konnektoren

Für Unterbrechungen und Fortsetzungen von Programmlinien durch die zeichnerische Darstellung veranlaßt.
Es können mehrere Absprungstellen zu einer Einsprungstelle führen. Zum Konnektor können noch Hinweise zum Auffinden des Anschlusses hinzugefügt werden.



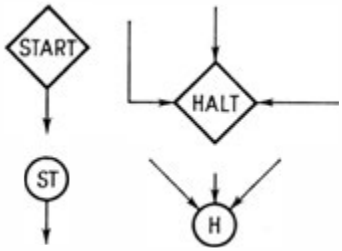
Unterprogramme, Programmteil

Es müssen angegeben werden: Anschlußwerte-Input, Name des UP oder PT, Anschlußwerte-Output

Tabelle 9. Auszug aus dem DDR-Standard Sinnbilder für Datenfluß- und Programmablaufpläne

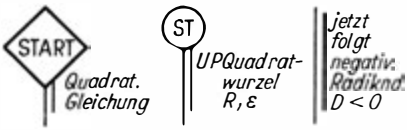
Sinnbilder

Kästchenmethode Leitlinienmethode Bedeutung Erklärung



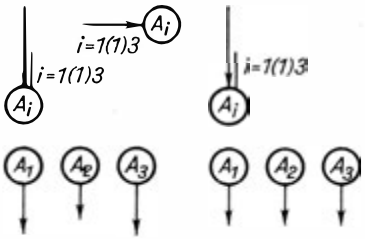
Organisationskästchen für Programmanfang und Programmende, für beide Methoden gleich. Es können
START für Beginn,
HALT für Ende,
STOP für Sonderfall und
ZWISCHENHALT
eingesetzt werden.

Organisationskästchen für Beginn und Ende von Unterprogrammen



Bemerkungen (Comments) und Parameterangaben

Bemerkungen sind grundsätzlich zum Beginn eines Programms anzuführen. Im Programmverlauf nach Bedarf

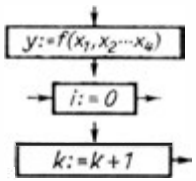
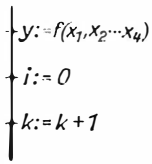
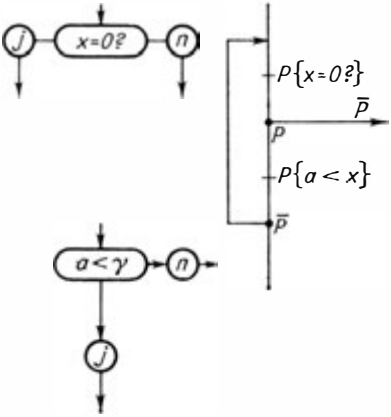


Variable Konnektoren

Für mehrfache Programmverzweigungen nach festgelegter Bedingung. Der Einsprung richtet sich nach dem aktuellen Wert des Parameters

$A_i = A_1 \vee A_2 \vee A_3$

Sinnbilder

Kästchenmethode	Leitlinienmethode	Bedeutung	Erklärung
		Operationskästchen (Plangleichungskästchen) Operationsanweisung	Operationen werden durch Plangleichung ausgelöst (gilt auch für Definitionen von Werten)
		Alternativkästchen; Verzweigungen	Bei der Leitlinienmethode wird eine Verzweigung durch einen Alternativtest P{ ...} eingeleitet. Beide Leitlinien müssen nach der Verzweigung entsprechend dem Test: P Test erfüllt $\overline{P}$ Test nicht erfüllt bezeichnet werden. Ausnahme: Erfolgt lediglich eine Weiterführung, ohne daß auf einem Zweig Operationen ausgeführt werden, dann ist nur der weiterführende Zweig zu kennzeichnen.

Geradeausprogramme,
verzweigte Programme,
zyklische Programme,

die meist nur als Programmteile auftreten.

Das einfachste Programm ist das *Geradeausprogramm*. Es besteht nur aus Operationsschritten (entsprechend auch nur aus Operationskästchen) und hat keine Verzweigungen oder Schleifen. Praktisch kommen reine Geradeausprogramme kaum vor. Sie treten höchstens als Programmteil auf.
Betrachten wir dazu einige Beispiele, die uns auch die Anwendung der Sinnbilder veranschaulichen. Zum besseren Einprägen wird in den Abbildungen Kästchen- und Leitlinienmethode nebeneinander dargestellt.
Beispiel 1: Wir wollen ein lineares Gleichungssystem von zwei Gleichungen mit zwei Unbekannten lösen. Mit Hilfe des bekannten Additionsverfahrens erhalten wir

$$\begin{array}{rcl|r|l} a_{11}x_1 & + a_{12}x_2 & = b_1 & a_{22} & -a_{21} \\ a_{21}x_1 & + a_{22}x_2 & = b_2 & -a_{12} & a_{11} \end{array}$$

$$a_{11}a_{22}x_1 + a_{12}a_{22}x_2 = b_1a_{22}$$
$$-a_{12}a_{21}x_1 - a_{12}a_{22}x_2 = -b_2a_{12}$$
$$(a_{11}a_{22} - a_{12}a_{21})x_1 = b_1a_{22} - b_2a_{12}$$
$$x_1 = \frac{b_1a_{22} - b_2a_{12}}{a_{11}a_{22} - a_{12}a_{21}}$$
$$-a_{11}a_{21}x_1 - a_{12}a_{21}x_2 = -b_1a_{21}$$
$$a_{11}a_{21}x_1 + a_{11}a_{22}x_2 = +b_2a_{11}$$
$$(a_{11}a_{22} - a_{12}a_{21})x_2 = b_2a_{11} - b_1a_{21}$$
$$x_2 = \frac{b_2a_{11} - b_1a_{21}}{a_{11}a_{22} - a_{12}a_{21}}$$

Die hierbei auftretenden Ausdrücke

$$a_{11}a_{22} - a_{12}a_{21}$$
$$b_1\,a_{22} - b_2\,a_{12}$$
$$b_2\,a_{11} - b_1\,a_{21}$$

nennt man Determinanten (2. Ordnung), die nach folgender Vorschrift berechnet werden:

$$\begin{vmatrix} a_{11} a_{12} \\ a_{21} a_{22} \end{vmatrix} = a_{11} a_{22} - a_{12} a_{21} = \Delta$$

$$\begin{vmatrix} b_1 \; a_{12} \\ b_2 \; a_{22} \end{vmatrix} = b_1 \, a_{22} - b_2 \, a_{12} = \Delta_1$$

$$\begin{vmatrix} a_{11} \; b_1 \\ a_{21} \; b_2 \end{vmatrix} = b_2 \, a_{11} - b_1 \, a_{21} = \Delta_2$$

Sichern wir, daß stets gilt $\Delta \neq 0$, so haben wir mit

$$\frac{\Delta_1}{\Delta} = x_1$$

$$\frac{\Delta_2}{\Delta} = x_2$$

das Modell eines Geradeausprogramms, wie Bild 93 darstellt.

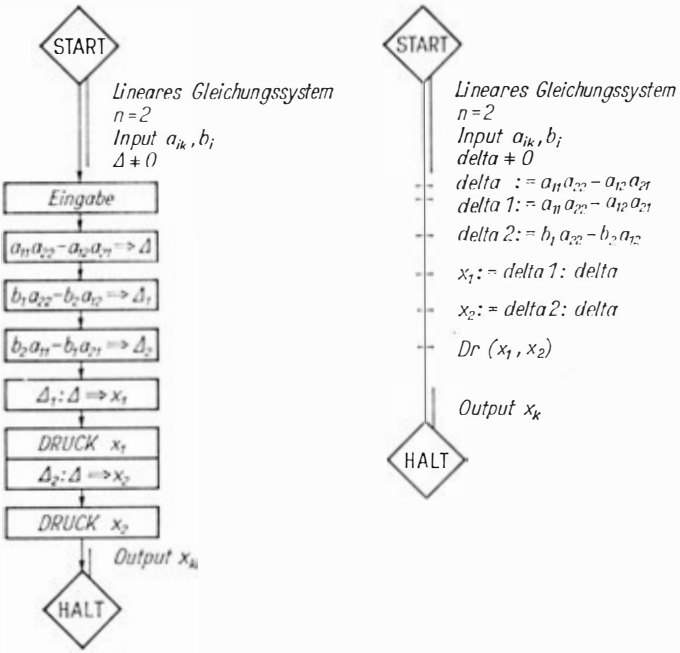


Bild 93. Beispiel eines Geradeausprogramms

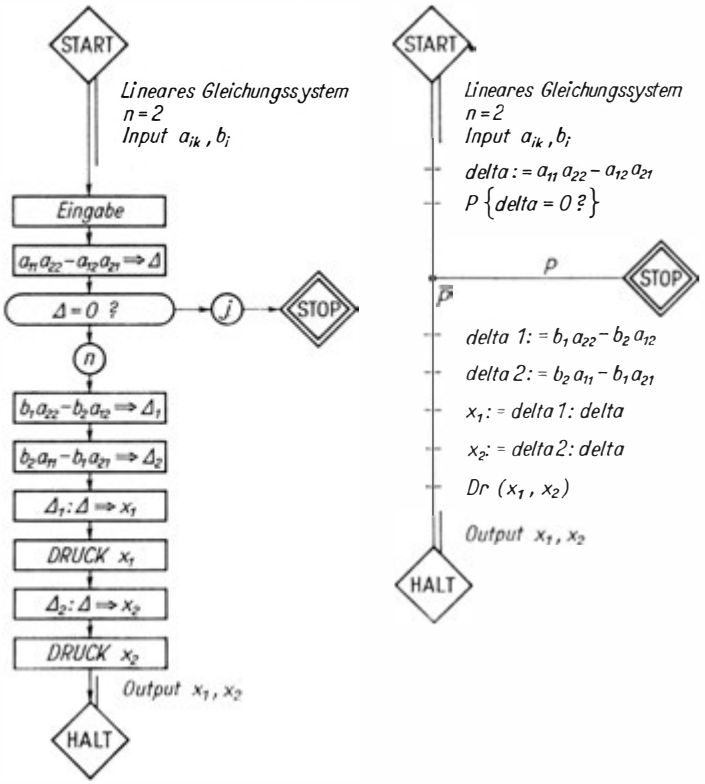


Bild 94. Beispiel eines verzweigten Programms

Sind bei einem Rechenablauf nach Berechnung einer bestimmten Größe (z. B. bei der Berechnung der Wurzeln einer quadratischen Gleichung nach der Ermittlung der Diskriminante) verschiedene Rechenwege möglich, so erhält man nach einer Alternative ein *verzweigtes Programm*.

Beispiel 2: Bleiben wir beim vorhin betrachteten Beispiel. Da wir nicht wissen, ob $\Delta \neq 0$ ist, wollen wir unser Programm erweitern: Wir erhalten dann ein verzweigtes Programm (Bild 94),

denn nach Berechnung von

$$a_{11}a_{22} - a_{12}a_{21} = \Delta$$

fügen wir in den Text eine Alternative ein

$$\Delta = 0?$$

Bei „JA", also $\Delta = 0$, können wir nicht weiterrechnen, da dann $\frac{\Delta_1}{\Delta}$ nicht ausgeführt werden darf. Hier liegt ein Sonderfall vor, den wir durch „STOP" kennzeichnen.
Bei „NEIN", also $\Delta \neq 0$, verläuft die Rechnung wie zuvor. Allerdings handelt es sich hier nur um ein Modellbeispiel, denn praktisch werden lineare Gleichungssysteme nicht nach dieser Methode berechnet.
Betrachten wir nun einige *zyklische Programme*. Wir kennen dabei sogenannte Induktions- und Iterationszyklen, deren Unterscheidung in Spezialfällen Schwierigkeiten bereitet.
Beispiel 3: Bilden wir das Produkt der ersten n Zahlen, so wird das Ergebnis als Fakultät bezeichnet und wie folgt geschrieben:

$$1 \cdot 2 \cdot 3 \cdot 4 \cdot \cdots \cdot (n-1) \cdot n = n!$$

(sprich „n Fakultät").
Speziell ist

$$\begin{aligned}
1! &= 1 = 1\\
2! &= 1 \cdot 2 = 2\\
3! &= 1 \cdot 2 \cdot 3 = 6\\
4! &= 1 \cdot 2 \cdot 3 \cdot 4 = 24\\
5! &= 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120\\
6! &= 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 = 720\\
7! &= 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 = 5040
\end{aligned}$$

Diese Funktion wird in der numerischen Mathematik sehr häufig verwendet. Ihre Berechnung erfolgt über die Plangleichungen

$$\begin{aligned}
1 &\Rightarrow k\\
1 &\Rightarrow i\\
i \cdot k &\Rightarrow i\\
k - n &\Rightarrow p\\
k + 1 &\Rightarrow k
\end{aligned}$$

wobei die Größe p als Prüfgröße eingeführt werden muß. Ist $0 > p$ folgt, daß auch gilt $k < n$. Dann entspricht das Ergebnis noch nicht dem verlangten Wert, und die Berechnung ist nach Ermitteln von $k + 1 \Rightarrow k$ zu wiederholen. Erst bei $0 = p$ gilt $k = n$, und der verlangte Wert liegt vor. Bild 95 zeigt das zugehörige Flußdiagramm.

Bei allen Programmierungsarbeiten können uns Fehler unterlaufen. Daher sollten wir vielfältigste Kontrollen in den gesamten Bearbeitungsprozeß einbauen. Die erste Kontrolle bei

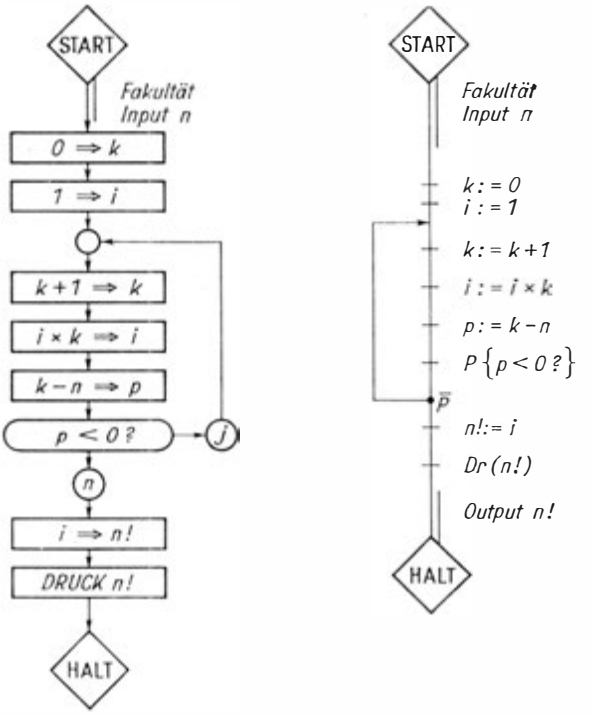


Bild 95. Berechnung der Fakultät über ein zyklisches Programm

der Programmierung sollte am Flußdiagramm oder Flußbild, wie es auch genannt wird, erfolgen.
Zur Durchführung der Kontrolle werden alle im Flußbild angeführten Größen in einer Tabelle erfaßt und darauf streng nach dem Ablauf des Flußbildes für ein nicht triviales, aber übersehbares Beispiel eine manuelle Rechnung durchgeführt. Das oder die Beispiele müssen dabei so gewählt werden, daß alle Teste mit jeder Entscheidung mindestens einmal angesprochen werden. Wir müssen also alle Schleifen und Programmteile mindestens einmal durchlaufen.
Betrachten wir im Bild 96 die Leitliniendarstellung und rechnen das Beispiel für $n = 4$, also $n! = 24$ durch.

Input	k	i	p	P $\{p < 0?\}$	$n!$
4	0	1			
	1	1	—3	ja	
	2	2	—2	ja	
	3	6	—1	ja	
	4	24	0	nein	24

Wie ersichtlich, wurde unser Flußbild richtig aufgestellt. Für ein derart einfaches Beispiel ist dies leicht zu übersehen. Der Vorteil einer solchen Flußbildkontrolle macht sich jedoch erst bei komplizierten Beispielen mit vielen Verzweigungen bemerkbar, wenn dabei der Rechenaufwand auch beträchtlich sein kann. Dieser Aufwand ist meist weitaus geringer als eine Fehlerkorrektur beim Erproben eines Programms.
Die nächsten Beispiele wollen wir als Unterprogramme aufstellen. Hierzu müssen wir Überlegungen zu Unterprogrammen anführen.
Unterprogramme zeigen einige Besonderheiten, bedingt durch den Einbau in ein Hauptprogramm.
Vor allem muß die Bereitstellung der Daten erfolgen, die aus dem Hauptprogramm in das Unterprogramm übernommen werden und umgekehrt. Meist sind sie als nächstfolgende Worte nach dem Absprung einzuspeichern. Bild 96 zeigt ein Beispiel, bei dem 3 Anschlußwerte für das Unterprogramm benötigt wer-

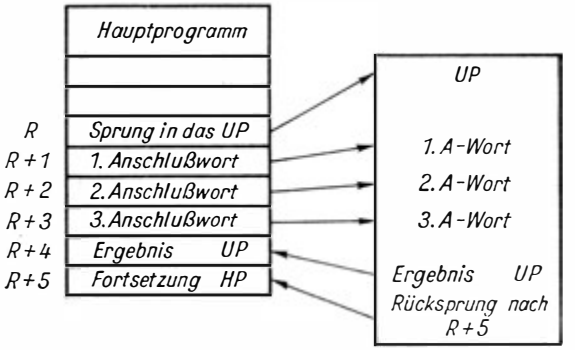


Bild 96. Beispiel für den Einbau eines Unterprogramms, das drei Anschlußwerte benötigt und einen Ergebniswert auswirft

den. Das UP errechnet einen Übergabewert an das HP. Der Rücksprung folgt nach R + 5, wobei R die Absprungadresse sein soll.

Werden größere Datenmengen im UP benötigt, können in die Anschlußworte nach Absprung nur die Leitadressen eingespeichert werden. Bestimmte Operationen dürfen im Unterprogramm nicht ausgeführt werden, beispielsweise Druckoperationen, da man ja nie weiß, wo das UP Verwendung findet. So gibt es noch viele Einschränkungen, auf die hier nicht weiter eingegangen werden kann.

Normalerweise ist ein Unterprogramm „für sich" nicht zu gebrauchen. Man benötigt selbst zum Erproben einen sogenannten Rahmen. Dadurch wird das UP aber auch allgemeiner anwendbar, während die Besonderheiten des Automaten im Rahmen berücksichtigt werden können.

Beispiel 4: Berechnen des skalaren Produktes als Induktionszyklus: Wir wollen auch das skalare Produkt zweier n-dimensionaler Vektoren berechnen. Das erfolgt über den bereits angedeuteten Algorithmus (Seite 186).

$$0 \Rightarrow s$$
$$s + ab \Rightarrow s.$$

Die Zyklenzahl müssen wir in einer Nebenrechnung bestimmen, indem wir (für einen n-dimensionalen Vektor) setzen

$$1 \Rightarrow i.$$

Mit diesem i durchlaufen wir den Zyklus und bilden die Prüfgröße

$$i - n \Rightarrow p,$$

um den Alternativtest

$$p < 0?$$

ausführen zu können. Bei „JA" ist $i < n$, dann muß gebildet werden

$$i + 1 \Rightarrow i$$

und der Zyklus ist erneut zu durchlaufen.
Bei „NEIN" ist $i = n$, und nun haben wir die Rechnung beendet und können den Wert ausgeben.
Führen wir auch hier die Kontrolle für $n = 3$ durch.

Input[1]	S	i	p	P $\{p < 0\}$	Output
$\langle R + 1\rangle$ $\langle R + 2\rangle$ $\langle R + 3\rangle$					$\langle R + 4\rangle$
3 $L^0(a_1)$ $L^0(b_1)$ 0		0			
a_1b_1		1	-2	ja	
$a_1b_1 + a_2b_2$		2	-1	ja	
$a_1b_1 + a_2b_2 + a_3b_3$		3	0	nein	S

Der Einbau des Unterprogramms entspricht also der auf Bild 97 dargestellten Situation. Die Anschlußwerte 2 und 3 sind allerdings Leitadressen $L^0(a_1)$ und $L^0(a_2)$, die tatsächliche Adresse wird dann in jedem Zyklus wie folgt berechnet:

$L^0(a_1) + (1 - 1) \Rightarrow$ Adr (a_1) 1. Durchlauf
$L^0(b_1) + (1 - 1) \Rightarrow$ Adr (b_1) $i = 1$
$L^0(a_1) + (2 - 1) \Rightarrow$ Adr (a_2) 2. Durchlauf

[1] L^0 Symbole, die Adressen darstellen sollen, werden durch den Exponenten 0 markiert

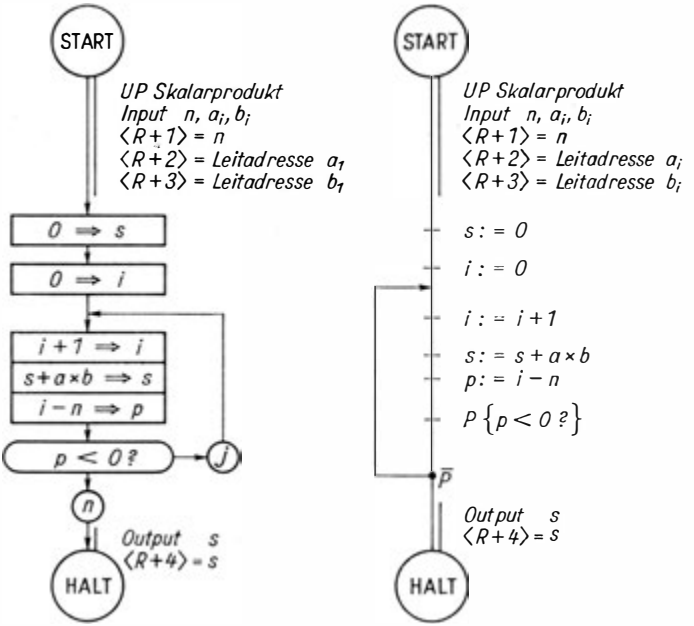


Bild 97. UP Skalarprodukt
Das Zeichen $< >$ soll heißen: Inhalt von

$$L^0(b_1) + (2 - 1) \Rightarrow \text{Adr } (b_2) \quad i = 2$$
$$L^0(a_1) + (3 - 1) \Rightarrow \text{Adr } (a_3) \text{ 3. Durchlauf}$$
$$L^0(b_1) + (3 - 1) \Rightarrow \text{Adr } (b_3) \quad i = 3$$

Als letztes Beispiel wollen wir den schon mehrfach betrachteten Iterationszyklus zur Berechnung einer Quadratwurzel besprechen.

Beispiel 5: Quadratwurzel als Unterprogramm: Das typische Beispiel eines zyklischen Programms ist die Berechnung der Quadratwurzel $\sqrt{R}$, was bekanntlich nach der Plangleichung

$$\frac{1}{2}\left(u + \frac{R}{u}\right) \to u$$

ausgeführt werden kann.

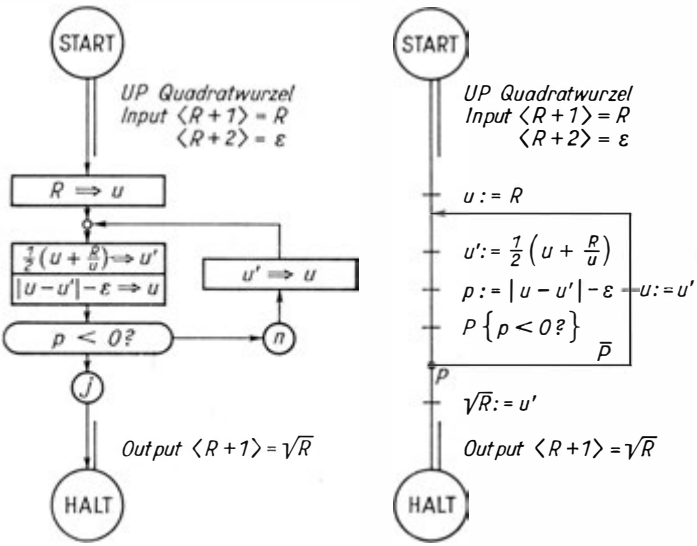


Bild 98. UP Quadratwurzel

Das Verfahren konvergiert, wie bereits gesagt, für jeden Radikanden $0 < R$ und jede Anfangsnäherung $u_0 \neq 0$.
Die Genauigkeit bestimmen wir als absoluten Betrag der Differenz zweier aufeinanderfolgender Näherungen. Dieser Betrag kann kleiner gemacht werden als eine vorher festgelegte Genauigkeitsschranke ε.
Hier liefert die Kontrolle für $R = 2$:

Input		u	u'	p	P{$p < 0$}	Output
⟨R + 1⟩	⟨R + 2⟩					⟨R + 1⟩
2	0,01	2	1,5	0,49	nein	
		1,5	1,4166	0,07	nein	
		1,4166	1,414216	—0,008	ja	1,414

Zum Einbau in das Hauptprogramm sehen wir also, daß im

1. Wort nach dem Sprungbefehl in das UP der Radikand und im 2. Wort die Genauigkeitsforderung eingespeichert sein müssen. Hier entnimmt das UP bei Bedarf diese Werte. Das Ergebnis, die Wurzel, wird wieder in das erste Wort nach dem Sprungbefehl in das UP gespeichert und steht dort für alle Berechnungen des folgenden HP-Teiles zur Verfügung.
Als Rahmen müßten wir die Bereitstellung der aktuellen Parameter R und ε organisieren und können uns gegebenenfalls das Ergebnis zur Kontrolle ausdrucken lassen.
Betrachten wir abschließend noch das Beispiel der quadratischen Gleichung

$$Ax^2 + Bx + C = 0$$

von der wir aber voraussetzen wollen, daß der Koeffizient des quadratischen Gliedes von Null verschieden sein soll, $A \neq 0$.
Die Quadratwurzel ist hier als Unterprogramm eingeschaltet.
Die Lösungen lassen sich wie folgt ermitteln:

Für $D = B^2 - 4AC > 0$ gilt:

$$x_1 = -\frac{B}{2A} + \frac{1}{2A}\sqrt{B^2 - 4AC}, \; x_2 = 0$$

$$x_3 = \frac{C}{Ax_1}, \; x_4 = 0$$

Für $D = B^2 - 4AC < 0$ gilt:

$$x_1 = x_3 = -\frac{B}{2A}$$

$$x_2 = \frac{1}{2A}\sqrt{-(B^2 - 4AC)}$$

$$x_4 = -\frac{1}{2A}\sqrt{-(B^2 - 4AC)}$$

und $z_1 = x_1 + ix_2, \quad z_2 = x_1 + ix_4.$

Für $D = B^2 - 4AC = 0$ gilt schließlich

$$x_1 = x_3 = -\frac{B}{2A}, \; x_2 = x_4 = 0$$

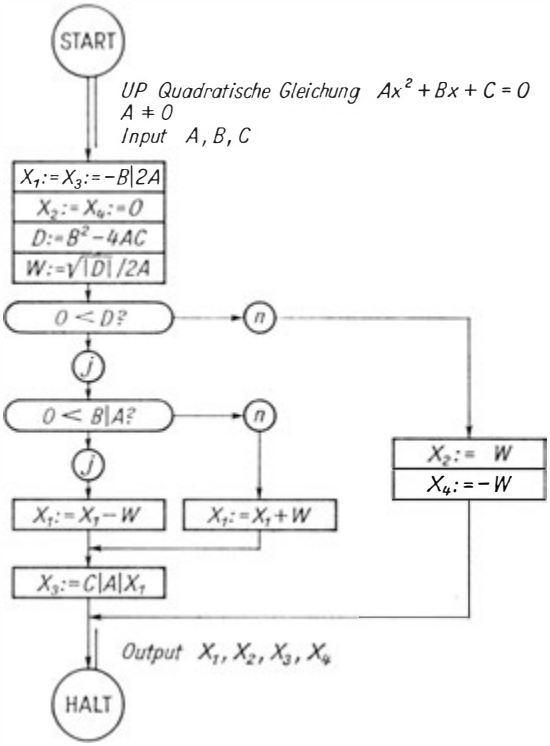


Bild 99. UP Quadratische Gleichung

Die angeführte Errechnung des Wertes x_3 bei reellen Wurzeln erfolgt aus Genauigkeitsgründen. Liegt der Ausdruck $4AC$ nahe bei Null, so ist $\sqrt{0}$ nahezu B, und $\frac{-B+B}{2A}$ könnte sehr ungenau werden. Daher wird der Test $0 < B/A$ eingefügt, und die Berechnung von x_3 erfolgt über die Vietaschen Wurzelsätze. Auf die Darstellung mittels der Leitlinienmethode wollen wir verzichten, da wir im Abschnitt 4 darauf noch einmal zurückkommen. Die Kontrolle schließlich soll dem Leser empfohlen werden.

3.5.3. Cellatron SER 2c

Für das Pseudogramm benötigen wir zusätzliche Kenntnisse über die Besonderheiten des Automaten. Für unsere weitere Programmierungsarbeit wollen wir uns mit dem digitalen Kleinrechner Cellatron SER 2c bekannt machen (Bild 100).
Der Cellatron SER 2c ist ein programmgesteuerter digitaler Kleinstrechner für wissenschaftlich-technische und kommerzielle Rechnungen sowie für Buchungs- und Fakturierarbeiten. Er arbeitet im Serienbetrieb, ist transistoriert und hat folgende technischen Daten:

Addition und Subtraktion ohne Adr	2,5	ms
Addition und Subtraktion mit Adr	50	ms
Multiplikation und Division ohne Adr	140	ms
Multiplikation und Division mit Adr	180	ms
mittlere Zugriffszeit	11	ms

Bild 100. Cellatron SER 2c

Speicherkapazität

Festwertspeicher	127 Wörter
Befehlsspeicher	381 Befehle

Bestückung:

Rechner	
Lochstreifenleser (Zugriffszeit)	50 ms
Lochstreifenstanzer	10 Zeichen/s
Schreibmaschine	10 Zeichen/s

Feste Wortlänge zu 48 Bit

Den strukturellen Aufbau zeigt Bild 101.

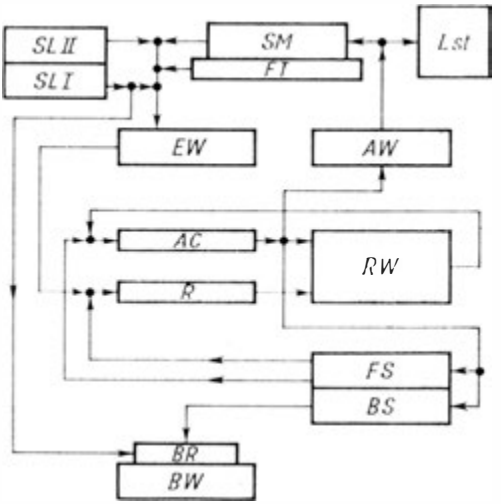


Bild 101. Der strukturelle Aufbau des Cellatron SER 2c

SL I	Streifenleser I	R	Register
SL II	Streifenleser II	EW	Rechenwerk
SM	Schreibmaschine	FS	Festwertspeicher
FT	Funktionstastatur	BS	Befehlsspeicher
EW	Eingabewerk	BR	Befehlsregister
AW	Ausgabewerk	BW	Befehlswerk
AC	Akkumulator	Lst	Lochstreifenstanzer

Beim Cellatron werden Wörter konstanter Wortlänge zu 12 Dualtetraden oder 48 Bit verwendet.
Das Zahlwort besteht aus 12 Zeichen. Dabei müssen wir eine externe und eine interne Zahldarstellung unterscheiden.
Extern umfaßt das Zahlwort 10 geltende Ziffern, ein Komma und das Vorzeichen. Letzteres wird aber nur als – angegeben, wenn die Zahl negativ ist. Dies entspricht der Wertangabe auf dem Eingabeformular:

Bezeichnung	Ziffernfolge und Komma	Vorzeichen
'5' $-a$	96,345 00	—
'3' b	0,883	
'2' 10^7	10 000 000,00	
'7' -1	1,000 000 0	—

Die Kommastellung wird durch eine sog. Kommainformation k festgelegt. Es sind maximal sieben Stellen nach dem Komma möglich.
Entsprechend ist das Bild der Ausgabe, wobei die im Beispiel gezeigten Punkte Leeranschläge der Schreibmaschine darstellen sollen. Es wird also von der ersten geltenden Ziffer bzw. von der Null vor dem Komma an ausgegeben. Die Gesamtzahl darf dabei im Notfall 12 Anschläge nicht übersteigen.

Beispiel (Kommainformationen entsprechen der Eingabe):

Anschlag	12	11	10	9	8	7	6	5	4	3	2	1
$a = -96{,}345$ '5'	.	.	.	9	6	,	3	4	5	0	0	—
$b = 0{,}883$ '3'	.	.	.	.	.	.	0	,	8	8	3	
10^7 '2'	1	0	0	0	0	0	0	0	,	0	0	
-1 '7'	.	.	1	,	0	0	0	0	0	0	0	—

Die interne Zahldarstellung umfaßt 12 Dualtetraden. Die Ziffern werden nach dem Direkt-Code verschlüsselt. Die Pseudotetraden verwendet man, wie beim Hexadezimalsystem beschrieben, als Pseudodezimale. Die Aufteilung der Dualtetraden des internen Zahlwortes ergibt sich wie folgt (Bild 102):

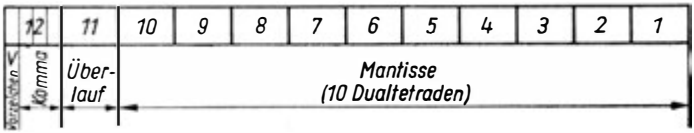


Bild 102. Internes Zahlwort des Cellatron SER 2

10 Tetraden für die Ziffernfolge (Tetrade 1 ··· 10)
1 Tetrade als Überlaufkontrolle (Tetrade 11)
1 Tetrade für das Vorzeichen und die Kommainformation (Tetrade 12). Für das Vorzeichen gilt die Festlegung: (Plus) + = O (Minus) — = L

Damit erhalten die obengenannten Zahlen folgende interne Darstellung:

	Tetrade										
12	11	10	9	8	7	6	5	4	3	2	1
LLOL	OOOO	OOOO	OOOO	OOOO	LOOL	●LLO	OOLL	OLOO	OLOL	OOOO	OOOO
OOLL	OOOO	OOOO	OOOO	OOOO	OOOO	OOOO	OOOO	OOOO	LOOO	LOOO	OOLL
OOLO	OOOO	OOOL	OOOO	OOOO	OOOO	OOOO	OOOO	OOOO	OOOO	OOOO	OOOO
LLLL	OOOO	OOOO	OOOO	OOOL	OOOO	OOOO	OOOO	OOOO	OOOO	OOOO	OOOO

Das Befehlswort umfaßt drei Einzelbefehle, wie auf Bild 103 angeführt.
Wie prinzipiell bei jedem Befehl, umfaßt er auch beim Cellatron Adreß- und Operationsteil.

Bild 103. Befehlswort des Cellatron SER 2

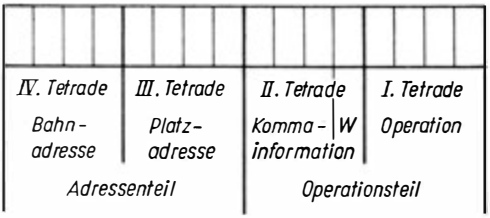


Bild 104. Befehl

Der Adreßteil (III. und VI. Tetrade eines Befehls) gibt die Adresse einer Zelle des Festwertspeichers für Operanden- und Ergebnisspeicherung bzw. des Befehlsspeichers bei Sprungoperationen an (eine Beschreibung der Sprungoperationen folgt beim Befehlssystem).
Der Cellatron SER 2c ist eine Einadreßmaschine. Der Adreßteil umfaßt also eine Adresse.
Die Adresse besteht aus zwei Teilen, der Bahn- und der Platzadresse. Für sie ist folgender Bereich möglich:
Bahnadresse: 0; 1; 2; 3; 4; 5; 6; 7;
Platzadresse: 0; 1; 2; 3; 4; 5; 6; 7; 8; 9; P2; P3; P4; P5; P6; P7;
woraus sich 128 Zellen ergeben! Da aber 0/0 anders vergeben wurde, beträgt die Speicherkapazität nur 127 Wörter.
Der Operationsteil (I. und II. Tetrade eines Befehls) gibt den Befehlstyp an. Dabei unterscheidet man

Transportbefehle,
Operationsbefehle,
Sprungbefehle,
Adressenbefehle und
Regiebefehle.

Die Einzelfestlegungen sind der Befehlsliste des Cellatron SER 2c zu entnehmen.
Von besonderem Interesse ist die II. Tetrade. Sie umfaßt die Kommainformation, den Warteindex und bestimmte Unterscheidungen für Sprungbefehle.
Der erste Bit der zweiten Dualtetrade eines Befehls mit der Wertigkeit $2^0 = 1$ entspricht dem Warteindex. Wird hier ein L gesetzt, so unterbricht der Automat vor Abarbeiten des vor-

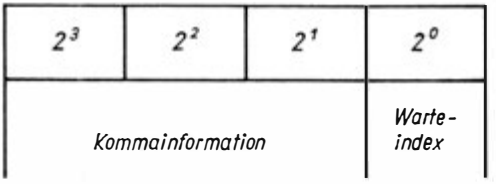


Bild 105. II. Dualtetrade eines Befehls

liegenden Befehls seine Arbeit und zeigt dies optisch auf der Funktionstastatur an. Das entspricht einem HALT oder STOP. Nun kann bei Bedarf vom Bedienenden über die Eingabeeinrichtung von Hand auf den weiteren Ablauf des Programms eingewirkt werden (z. B. durch Eingabe neuer Daten). Nach Drücken der Befehlstaste „START" der Funktionstastatur wird die Abarbeitung des Programms fortgesetzt.
Durch diese Festlegung des Warteindex muß die Kommainformation, auf deren Kommawert die Operanden vor der Operation normalisiert (Herbeiführen der Stellengleichheit) und das Ergebnis nach der Operation gerundet werden, mit doppelter Stellenzahl wie folgt angegeben werden:

für ganze Zahlen	: 0	für 0,0001	: 8
für 0,1	: 2	für 0,00001	: 10 = P2
für 0,01	: 4	für 0,000001	: 12 = P4
für 0,001	: 6	für 0,0000001	: 14 = P6

Die Kommainformationen können also im Code nur geradzahlige Werte annehmen, wobei die Zahlen über 10 unter Verwendung der Pseudodezimalzahlen eingegeben werden müssen.
Wollen wir zusätzlich zur Kommainformation auch einen Warteindex setzen, müssen wir sie um 1 erhöhen:

für ganze Zahlen	: 1	für 0,001	: 9
für 0,1	: 3	für 0,00001	: 11 = P3
für 0,01	: 5	für 0,000001	: 13 = P5
für 0,001	: 7	für 0,0000001	: 15 = P7,

womit die Kommainformationen zu ungeraden Zahlen werden.
Bei den Sprungbefehlen verliert die Kommainformation ihre Bedeutung. Hier wird sie zur Unterscheidung der beiden Sprungarten herangezogen, wie im Befehlssystem dargelegt. Der Warteindex bleibt erhalten.
Unter dem Befehlssystem versteht man ein Verzeichnis aller Befehle, die der Automat ausführen kann.
Diese Befehle werden in der Befehlsliste zusammengestellt. Die Befehlsliste muß dann durch eine knappe Beschreibung der Arbeitsweise des Automaten zur Ausführung der Befehle ergänzt werden. Die Befehlsliste des Cellatron SER 2c fügen wir als Beilage bei.

Der üblichen Bezeichnungsweise nach wollen wir für die Programmierung vereinbaren:

Symbol	Bedeutung	Beispiel
$<\cdots>$	Inhalt der Zelle	$<AC>$ — Inhalt des Akkumulators $<FS/a>$ — Inhalt der Zelle des Festwertspeichers, die die Adresse a hat.
$>\cdots<$	Adresse von $\cdots$	$>x<$ — Adresse des Wertes x
➡	Transport nach $\cdots$	$<FS/a> \rightarrow <AC>$ Der Inhalt der Speicherzelle mit der Adresse a des Festwertspeichers ist in den Akkumulator zu transportieren.
&	eine der vier Grundrechenarten	

3.5.4. Weiterführen des Programmierens

Zum Herstellen des Pseudoprogramms gilt es, ein Programmformular mit folgenden Spalten auszufüllen:

1. Nummer des Befehlswortes von 0/1 bis 7/P7
2. Befehl des Befehlswortes (a-, b- oder c-Befehl)
3. Adreßteil (Bahn- und Platzadresse)
4. Kommainformation und Warteindex
5. Operation
6. Inhalt des Registers
7. Inhalt des Akkumulators
8. Bemerkung.

Die Spalten 1 und 2 liefern die Befehlsadresse, 3 bis 5 geben den Befehl, und 6 bis 8 dienen zur besseren Übersicht.
Neben dem Programmvordruck ist ein Speicherbelegungsplan zu führen, der die Belegung des Festwertspeichers liefert und die Befehlsadresse, von der an die Zelle belegt ist.
Wir wollen das Aufstellen des Programms und des Speicherbelegungsplanes bei der Besprechung der einzelnen Befehlsgruppen an kleinen Beispielen üben.

Transportbefehle lösen einen Transport von Daten innerhalb des Automaten aus. Dieser Transport kann je nach Automatentyp zwischen den Speicherzellen, von einer Speicherzelle in ein Register und umgekehrt sowie zur Ein- und Ausgabeeinrichtung erfolgen.
Beim Cellatron beziehen sich die Transportbefehle auf das Rechenwerk, und wir können Ein- und Ausgabe unterscheiden.
Dabei bedeuten

Eingabe:

$a.k.$E:[1] für $\begin{cases} a \neq 0 & \langle \text{FS}/a \rangle \rightarrow \langle \text{R} \rangle \text{ Lö}^2 \\ a = 0 & \langle \text{SM} \rangle \rightarrow \langle \text{R} \rangle \text{ Lö} \end{cases}$

Für $a \neq 0$ wird der Inhalt der Zelle a des Festwertspeichers in das Register transportiert, dessen alter Inhalt dabei gelöscht wird.
Für $a = 0$ erfolgt eine Eingabe des von der Schreibmaschine angegebenen Wertes.

Lochbandeingabe:

O.O.LE1	⟨SLI⟩ → ⟨R⟩ Lö
O.O.LE2	⟨SLII⟩ → ⟨R⟩ Lö

Der im Streifenleser I respektive II anliegende Wert wird in das Register transportiert.
Bei Eingabeoperationen wird keine Kommainformation gegeben.

Ausgabe:

$a.k.$A für $\begin{cases} a \neq 0 & \langle \text{AC} \rangle \text{ Lö} \rightarrow \langle \text{FS}/a \rangle \\ a = 0 & \langle \text{AC} \rangle \rightarrow \langle \text{SM} \rangle \text{ Druck} \end{cases}$

---

[1] $a.k.$E soll als Abkürzung für einen Cellatronbefehl gesetzt werden
a bedeutet die Adresse (z. B. 00)
k bedeutet die Kommainformation (z. B. 0)
E bedeutet das Symbol des Befehls (hier „Eingabe")
[2] Das Zeichen ⟨⟩ Lö soll bedeuten, daß nach erfolgter Operation der alte Inhalt gelöscht ist

Der Inhalt des Akkumulators wird bei $a \neq 0$ in die Zelle a des Festwertspeichers transportiert, wobei er dann in AC gelöscht ist, bei $a = 0$ folgt eine Ausgabe über die Schreibmaschine ohne Löschen!

Ausgabe Lochstreifen:

O.*k*.AL $\langle AC \rangle \rightarrow \langle L\ ST \rangle$

Der Inhalt des AC wird ohne Löschen in den Lochstreifen gestanzt.

Bei Ausgabebefehlen kann eine Kommainformation k gegeben werden.

Operationsbefehle umfassen neben den arithmetischen Operationen je nach Automatentyp logische Verknüpfungen, Rechts- oder Linksverschiebungen u. a.

Der Cellatron SER 2c hat nur für die arithmetischen Operationen Befehle, dabei führt er die Division nur ganzzahlig, ohne Runden aus. Hier bedeuten:

a.k. & für $\begin{cases} a = 0 & \langle AC \rangle\ \&\ \langle R \rangle \Rightarrow \langle AC \rangle \\ a \neq 0 & \begin{cases} \langle FS/a \rangle \rightarrow \langle AC \rangle \\ \langle AC \rangle\ \&\ \langle R \rangle \Rightarrow \langle AC \rangle \\ \langle AC \rangle \rightarrow \langle FS/a \rangle \text{ Lö} \end{cases} \end{cases}$

Ist $a =$ Null, wird der Inhalt des AC mit dem des R in angegebener Reihenfolge verknüpft, und das Ergebnis steht im AC. Beide Operanden müssen dazu vorher in richtiger Reihenfolge nach AC und R transportiert worden sein.

Für $a \neq 0$ wird ein fest verdrahtetes Teilprogramm mit folgenden Weisungen ausgelöst:

- ✳ Transportiere den Inhalt der Zelle, die in der Adresse angegeben ist, als ersten Operanden in den Akkumulator.
- ✳ Verknüpfe beide Operanden wie beim Ablauf mit Adresse Null.
- ✳ Transportiere den Inhalt des Akkumulators, also das Ergebnis, in die Zelle mit der Adresse a, wobei das Ergebnis aber auch im Akkumulator erhalten bleibt.

Der zweite Operand muß dazu vorher in das Register transportiert werden.

Nun können wir bereits ein kleineres Programm aufstellen. Es sei folgende Ausgangsbelegung gegeben:
⟨0/1⟩ = a, ⟨0/2⟩ = b und ⟨0/3⟩ = c. Diese Werte sollen in die Zellen 1/3, 1/4 und 1/5 transportiert und ausgegeben werden. Alle Werte sollen drei Stellen nach dem Komma haben:

Programmierungsbeispiel 1: Umspeichern mit Druck

1	2	3	4	5	6	7	8
0/1	a	0/4	0	E	0	bel	Löschen R
	b	0/1	6	+	0	a	
	c		6	A	0	a	Druck a
0/2	a	1/3	6	A	0	0	
	b	0/2	6	+	0	b	
	c		6	A	0	b	Druck b
0/3	a	1/4	6	A	0	0	
	b	0/3	6	+	0	c	
	c		6	A	0	c	Druck c
0/4	a	1/5	6	A	0	0	
	b		1				HALT (Wartoindex)

Der Speicherbelegungsplan würde zu diesem Beispiel folgendes Aussehen haben.

Befehl	k	Adressen 0/1	0/2	0/3	0 4 .... 1/3	1/4	1/5
Eingabe	6	a	b	c	0		
0/2 a	6					a	
0/3 a	6						b
0/4 a	6						c

Auf Besonderheiten der Division beim Cellatron SER 2c wollen wir nicht eingehen. Wir empfehlen dazu das Studium der vom Hersteller herausgegebenen Programmieranweisung.

Bei anderen Automatentypen rechnen zu den Operationsbefehlen logische Operationen, Verschiebeoperationen u. a.
Sprungbefehle beziehen sich auf das Programm und dessen Ablauf. Wie wir bereits andeuteten, werden Programme normalerweise in der Reihenfolge der aufgeführten Befehle abgearbeitet. Soll dieser Verlauf unterbrochen werden, so ist ein Sprung zu programmieren. Nach einem Sprungbefehl wird nicht der nächstfolgende Befehl des Programms zur Abarbeitung aufgerufen, sondern der Befehl, dessen Adresse im Sprungbefehl angegeben wird. Man unterscheidet zwei Arten von Sprüngen:

1. den unbedingten Sprung. Er ist immer auszuführen;

2. den bedingten Sprung. Er ist nur auszuführen, wenn eine Bedingung erfüllt ist. Hierbei können verschiedene Bedingungen herangezogen werden, das Vorhandensein oder Fehlen eines Markierungszeichens, das Vorzeichen einer Prüfgröße, das Ergebnis eines Vergleiches zweier Größen u. a.

Beim Cellatron SER 2c gibt es nur einen Test:

Ist das im Akkumulator stehende Ergebnis negativ?

Im Falle „Ja" ist der Sprung auszuführen, im Falle „Nein" wird mit dem nächstfolgenden Befehl des Programms weitergerechnet.
Der Ablauf ist hier folgender:

a.2.S — für $\begin{cases} a \neq 0 & \text{bedingter Sprung zum Befehl, der in Zelle } a \text{ des Befehlsspeichers steht} \\ a = 0 & \text{bedingter Sprung zum Bandprogramm, das in den Streifenleser 1 (SL I) eingelegt sein muß. Liegt dort kein Programm vor, so wirkt der Sprung als HALT.} \end{cases}$

Analog ist der Ablauf beim unbedingten Sprung SU.
Damit sind folgende Sprünge möglich:

Bandprogramm – Trommelprogramm
Trommelprogramm – Trommelprogramm
Trommelprogramm – Bandprogramm

Das Lochband kann dadurch als externer Befehlsspeicher verwendet werden. Innerhalb des Bandprogramms sind jedoch keine Sprungoperationen möglich.

Merke: Es kann nur der a-Befehl eines Befehlswortes angesprungen werden!

Adressenbefehle können wir nur als Modell besprechen, da der Cellatron SER 2 keine automatische Adressenänderung zuläßt. Moderne Rechenanlagen können mit Befehlen rechnen. Damit ist eine automatische Veränderung des Programms während des Rechenablaufes möglich.
Dieses Rechnen mit Befehlen bezieht sich auf das Verändern von Adressen, ist also eine automatische Adressenänderung.
Allgemein unterscheidet man drei Arten der Adressenänderung.

Adressenänderungen erster Art

Betrachten wir einen Automaten mit einer Trommel als internen Speicher. Stellen wir hierfür ein Programm her, so wissen wir nicht, in welchen Zellen es bei der Rechnung stehen wird. Es könnte einmal in den ersten Zellen der Trommel beginnen, aber auch jede andere Stelle kann das Programm aufnehmen, wenn die ersten Zellen bereits besetzt sind.
Daher setzen wir beim Programmieren die Adressen nicht absolut ein, bezogen auf die erste Zelle des Speichers, sondern relativ, bezogen auf den ersten Befehl des Programms. Die absolute Adresse ergibt sich dann durch Addition der relativen Adresse mit einer Konstanten.

$$K + \mathrm{Adr}_{rel} \Rightarrow \mathrm{Adr}_{abs}$$

wobei die Konstante der absoluten Adresse des ersten Programmbefehles entsprechen soll (Bild 106).
Die Konstante ist eine bekannte feste Größe. Sie kann in ein Register gegeben werden und ist dann leicht zu den relativen Adressen zu addieren.
Diese Art der Adressenänderungen, Addition mit einer Konstanten, nennt man von „erster Art".

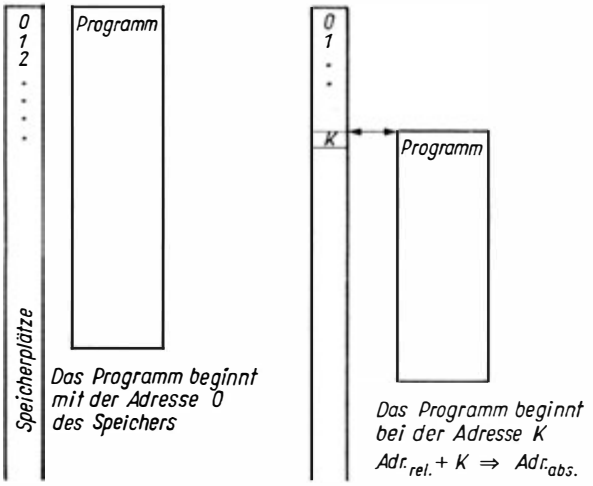


Bild 106. Schematische Darstellung der Adressenänderungen erster Art

Adressenänderungen zweiter Art

Gehen wir wieder davon aus, daß wir ein Programm mit relativen Adressen, beispielsweise zur Berechnung der Quadratwurzel, vorliegen haben. Dann brauchen wir die Quadratwurzelberechnung in einem neuen Programm nicht mehr zu programmieren. Wir können das Wurzelprogramm als Unterprogramm in das Hauptprogramm einfügen.
Die Berechnung der Adressen innerhalb des Nebenprogramms kann durch eine Adressenänderung erster Art ausgeführt werden. Schwieriger ist es, wenn im Unterprogramm Parameter aus dem Hauptprogramm benötigt werden, wenn Zwischenergebnisse in das Hauptprogramm abzugeben sind oder wenn in das Hauptprogramm zurückgeführt werden soll. Hierfür sind Adressenänderungen während des Berechnungsprozesses auszuführen, die „von zweiter Art" genannt werden (Bild 107).

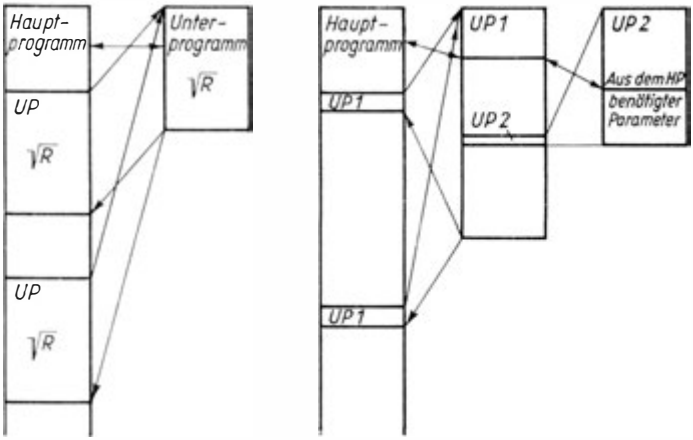


Bild 107. Schematische Darstellung der Adressenänderungen zweiter Art. Werden in einem Unterprogramm Werte benötigt, die im Hauptprogramm adressiert sind, so müssen die Adressen im Unterprogramm umgerechnet werden. Dies kann mehrfach auftreten (Adresse von Adresse)

Adressenänderungen dritter Art

Hierzu wollen wir die Berechnung des Skalarproduktes zweier Vektoren nochmals betrachten. Nach dem auf Seite 200 angeführten Algorithmus mußte nach Definition der Ausgangsgrößen

$$0 \Rightarrow s_0$$
$$1 \Rightarrow i$$

der Zyklus

$$a \cdot b \Rightarrow q$$
$$s + q \Rightarrow s$$
$$i - n \Rightarrow p$$
$$0 = p?$$

n-mal durchlaufen werden, wobei nach jedem Durchlauf eine Substitution des Index i gemäß

$$i + 1 \Rightarrow i$$

vorzunehmen ist. Durch diese Substitution des Index werden die neuen a_i und b_i für den Zyklus bereitgestellt. Das erfolgt durch eine im Berechnungsprozeß vorzunehmende Adressenänderung dritter Art (Änderung von indizierten Variablen).
Die technische Ausführung der Adressenänderungen ist nahezu für jeden Automatentyp verschieden.
Für den Cellatron SER 2c gibt es keine Adressenänderungen und daher auch keine Adressenbefehle. Es besteht lediglich die Möglichkeit, an der Stelle, an der eine Adressenänderung erfolgen soll, einen Warteindex zu programmieren und manuell die Adresse zu ändern.
Zu den *Regiebefehlen* könnte man beim Cellatron SER 2c die Befehle zählen, die zur Bewegung des Wagens der Schreibmaschine dienen. Sie entsprechen den bekannten Möglichkeiten der Schreibmaschine:

Leertaste,
Tabulatorsprung,
Wagenrücklauf und Zeilenschaltung.

Für diese Befehle hat die Kommainformation keine Bedeutung. Sie wird daher lediglich für das Programmieren des Warteindex benutzt. Zusammenfassend können wir nun die Befehlsliste für den Cellatron SER 2c zusammenstellen, die auf der Beilage vermerkt ist.
Nunmehr sind wir in der Lage, das Pseudogramm aufzustellen. Betrachten wir dies am Beispiel.
Programmierungsbeispiel 2: Fakultät (vgl. Bild 96)
Wir wollen das Programm so aufbauen, daß die ganzzahlige Funktion „Fakultät" von $k = 1$ bis $k = n$ mit $n! <$ größte zulässige Zahl, tabelliert wird. Als Ergebnis soll ausgeworfen werden:

1	1
2	2
3	6
4	24
5	120
6	720
7	5040

8	40320
9	362880
10	3628800
:	:

Der Speicherbelegungsplan hat folgende Form:

Befehl	k	Adressen 0/1	0/2	0/3	0/4	0/5	0/6	0/7
Eingabe	0	n	1	0				
0/1 c	0					i		
0/2 a	0				$k-1$			
0/3 a	0						bel	
0/5 b	0				k			
0/7 c	0					2!		

Befehl Adr	Nr.	Adr	Komma-inf.	Op	⟨R⟩	⟨AC⟩	Bemerkungen
0/1	a	0/3	0	E	0	bel	
	b	0/2	0	+	0	1	
	c	0/5	0	A	0	0	$i := 1$
0/2	a	0/4	0	A	0	0	$k-1 := 0$
	b	0/3	0	SU			
	c						
0/3	a	0/6	0	A	bel	0	Säubern des AC
	b	0/2	0	E	1	0	
	c		0	+	1	1	
0/4	a	0/4	0	E	$k-1$	1	
	b		0	+	$k-1$	k	$k := (k-1)+1$
	c		0	T	$k-1$	k	Tabulator
0/5	a		0	A	$k-1$	k	Druck k
	b	0/4	0	A	$k-1$	0	
	c		0	T	$k-1$	0	Tabulatorsprung
0/6	a	0/4	0	E	k	0	$i := i \times k$
	b		0	+	k	k	
	c	0/5	0	E	i	k	$p := k - n$

Befehl Adr	Nr.	Adr	Komma-inf.	Op	⟨R⟩	⟨AC⟩	Bemerkungen
0/7	a		0	x	i	$i \times k$	
	b		0	A	i	$i \times k$	$i := i \times k$
	c	0/5	0	A	i	0	Druck k!
0/8	a	0/4	0	E	k	0	
	b		0	+	k	k	
	c	0/1	0	E	n	k	$p := k - n$
0/9	a		0	—	n	$k-n$	
	b		0	W	n	$k-n$	Wagenrücklauf
	c	0/3	2	S—	n	$k-n$	Sprung bed.
0/P2	a		1				Halt

Die Kommainformation ist dabei ohne Bedeutung, da wir nur ganzzahlig rechnen. Unser Programm ist zudem ein Hauptprogramm, denn in einem Unterprogramm dürften keine Druckoperationen vorkommen.

Programmierungsbeispiel 3: UP Quadratwurzel (vgl. Bild 98).

Im Unterprogramm benötigen wir lediglich die Konstante 0,5. Sie soll sich in der letzten Zelle des Festwertspeichers befinden. Die vorletzte Zelle wird zudem als Keller benutzt.

Die aktuellen Parameter R – Radikand und ε – Genauigkeitsvorgabe müssen mit zwei der Genauigkeitsforderung entsprechenden Korrekturfaktoren für die Division mit dem Hauptprogramm bereitgestellt werden.

Entsprechend sei folgender Speicherbelegungsplan vorgegeben:

Befehl	k	Adressen 7/8	7/9	7/P2	7/P3	7/P4	7/P5	7/P6	7/P7
Ausgang	k	Arb.	Arb.	R	ε	10^{+1}	10^{-1}	Keller	0,5
7/3 a	0							bel	
7/4 a	6		u						
7/5 a	0							bel	
7/9 b	6	u'							
7/P4 a	8							p	
7/P5 a	6		u						

Befehl Adr	Nr.	Adr	Komma-inf.	Op	⟨R⟩	⟨AC⟩	Bemerkungen
7/3	a	7/P6	0	A	bel	0	Säubern des AC
	b	7/P2	0	E	R	0	
	c		6	+	R	R	$u := R$
7/4	a	7/9	6	A	R	0	
	b	7/5	0	SU			Einsprung in den Zyklus
	c						
7/5	a	7/P6	0	A	bel	0	Säubern des AC
	b	7/P2	0	E	R	0	
	c		6	+	R	R	
7/6	a	7/P4	0	E	10^1	R	
	b		6	×	10^1	10^1R	
	c	7/9	0	E	u	10^1R	
7/7	a		0	:	u	$10^1R:u$	$u' := 0{,}5$
	b	7/P5	0	E	10^{-1}	$10^1R:u$	$\left(u + \frac{R}{u}\right)$
	c		6	×	10^{-1}	$R:u$	
7/8	a	7/9	0	E	u	$R:u$	
	b		0	+	u	$u + R:u$	
	c	7/P7	0	E	0,5	$u + R:u$	
7/9	a		6	×	0,5	u'	
	b	7/8	6	A	0,5	0	
	c	7/9	0	E	u	0	
7/P2	a		6	+	u	u	
	b	7/8	0	E	u'	u	$p := (u - u)'$
	c		6	−	u'	$u - u'$	$-\varepsilon$
7/P3	a	7/P3	0	E	ε	$u - u'$	
	b		8	−	ε	$u - u' - \varepsilon$	
	c	7/P6	2	S−	ε	p	
7/P4	a	7/P6	0	A	ε	0	
	b	7/8	0	E	u'	0	$u := u'$
	c		6	+	u'	u'	
7/P5	a	7/9	6	A		0	
	b	7/5	0	SU			Rücksprung zum Anfang des Zyklus
	c						

Befehl Adr	Nr.	Adr	Komma-inf.	Op	<R>	<AC>	Bemerkungen
7/P6	a	7/P6	0	A	bel	0	
	b	7/8	0	E	u'	0	
	c		6	+	u'	u'	
7/P7	a		0	SU			Rücksprung in das HP Adresse angeben!

Das UP wird somit an das Ende des Speichers gegeben. Die Rücksprungadresse ist dabei für jeden Fall festzulegen, desgleichen ε und 1 für die Genauigkeitsfestlegung.
Die Wurzel steht dann einmal in der Zelle 7/8 und im Akkumulator.
In der Fortsetzung des Bearbeitungsprozesses müssen wir das Programm nun in den *Maschinencode* übersetzen, damit es dann gelocht werden kann.
Das Programmierungsbeispiel 2 würde im Maschinencode folgendermaßen aussehen, was wir durch Vergleichen mit den Angaben im Befehlsschlüssel leicht feststellen können:

Bef. Adr	Befehlswort Befehl											
	c				b				a			
	III, 4/III, 3		III, 2	III, 1	II, 4/II, 3		II, 2	II, 1	I, 3/I, 4		I, 2	I, 1
0/1	0	5	0	6	0	2	0	1	0	3	0	5
0/2					0	3	0	7	0	4	0	6
0/3	0	0	0	1	0	2	0	5	0	6	0	6
0/4	0	0	0	P6	0	0	0	1	0	4	0	5
0/5	0	0	0	P6	0	4	0	6	0	0	0	6
0/6	0	5	0	5	0	0	0	1	0	4	0	5
0/7	0	5	0	6	0	0	0	6	0	0	0	3
0/8	0	1	0	5	0	0	0	1	0	4	0	5
0/9	0	3	2	7	0	0	0	P7	0	0	0	2
0/P2									0	0	1	0

3.5.5. Programmiersysteme

Im vorigen Abschnitt haben wir das Aufstellen einfacher Programme in der Maschinensprache verfolgen können. Dabei benutzten wir den Befehlsschlüssel des Cellatron SER 2, der nur 15 Befehle umfaßt und deshalb leicht zu behalten ist. Zudem können die wohl schwierigsten Operationen der Adressenrechnung mit dem Cellatron nicht ausgeführt werden. Auch dadurch wird der Programmierungsprozeß vereinfacht. Allerdings sind dafür die aufgestellten Programme recht starr und können nur begrenzt variiert werden.
Beispielsweise ist das UP Quadratwurzel (Programmierbeispiel 3) nur in den letzten Teil des Trommelspeichers einzugeben und kann auch nur einmal als Unterprogramm verwandt werden, da ja die Rücksprungadresse vorher bestimmt und fest vorgegeben werden muß. (Auf halbautomatische Arbeitsweise mit manueller Adressenänderung können wir hier nicht eingehen.)
Leistungsfähigere Automaten (ZRA 1 oder Importgeräte) sind im Befehlsschlüssel weitaus flexibler, was jedoch den Programmierungsprozeß selber wesentlich kompliziert.
Hieraus wird verständlich, daß man nach internationalen Erfahrungen für umfangreichere Programme den Wert setzt:

> für jeden Befehl eines komplizierten Programms ist eine Stunde Programmierungszeit vorzusehen.

Nehmen wir nun nur die Rechengeschwindigkeit unseres ZRA 1, der 150 Op/s leistet, dann müßte theoretisch ein Programmierer nahezu zwanzig Arbeitstage aufwenden, um den ZRA 1 eine Sekunde zu beschäftigen. Der ZRA 1 ist aber keineswegs ein schneller Automat.
Diese Faustrechnung ist natürlich zugespitzt, dennoch weist sie eindringlich auf das Problem hin, Programmierer auszubilden. Die Forderung wird auch nicht dadurch gemindert, daß alle bereits erarbeiteten Programme für beliebig viele spätere Berechnungen zur Verfügung stehen und in Programmbibliotheken zusammengefaßt werden.
Auch unter Verwendung der Programmbibliothek ist die von uns bisher durchgeführte Programmierungsarbeit unbefriedigend.

Sie erfordert einen zu großen Aufwand.

Sie dauert zu lange.

Sie besteht größtenteils selber aus formaler geistiger Arbeit, insbesondere die Aufstellung des Pseudogramms und das Codieren zum Maschinenprogramm.

Zur Ausführung formaler geistiger Arbeit haben wir doch aber gerade Rechenautomaten konstruiert und gebaut. Hier liegt der Kerngedanke, den Automaten selber zur Aufstellung seines Programms zu benutzen.

Dieser Gedanke ist so alt wie die rund zwanzigjährige Geschichte der programmgesteuerten Rechenautomaten. Bereits der Altmeister Dr. h. c. Konrad Zuse arbeitete 1943 bis 1948 an den Grundlagen und stellte sein „Plankalkül" auf, aus dem wir das Ergibtzeichen übernommen haben.

Heute gibt es für nahezu jeden einigermaßen leistungsfähigen Automaten Programmierungshilfen in Form von Programmiersystemen.

Ein Programmiersystem besteht aus einer Programmiersprache und einem zugehörigen Umwandlungsprogramm.

Umwandlungsprogramme übernehmen die Information aus der Grobprogrammierung und übertragen sie in ein Maschinenprogramm. Dabei können wir zwei Arbeitsweisen der Umwandlung unterscheiden: Die interpretierende Umwandlung übernimmt eine eng begrenzte Informationsmenge, meist einen Satz, übersetzt ihn in die Maschinensprache und führt ihn gleich aus. Übersetzung und Ausführung wechseln also ständig, woraus sich jedoch teilweise recht lange Ausführungszeiten ergeben.

Bei kompilierender Umwandlung hingegen wird das gesamte Programm durch einen Compiler aus der Grobprogrammierung und teilweise sogar aus der algorithmischen Aufbereitung des mathematischen Modells übersetzt und nach Abschluß des Übersetzungsprozesses abgearbeitet.

Welches der beiden Arbeitsverfahren angewandt wird, hängt vom Problemkreis und vor allem vom Automatentyp ab. Das Aufstellen der Umwandlungs- oder Übersetzungsprogramme ist sehr zeitaufwendig. Dieser Aufwand wird um so größer, je allgemeingültiger die Programmiersprache ist.

Die einfachste, wenn auch recht bedeutende Programmierungs-

hilfe ist die Verwendung symbolischer Adressen. Es werden zwar noch alle Befehle einzeln aufgeführt, aber statt einer numerischen Adresse kann ein allgemeines Symbol eingesetzt werden. Für dieses Symbol setzt der Automat dann selbständig die numerische Adresse.
Autocodes sind bereits einfache, allgemeine Programmiersprachen. Sie sind der Maschinensprache, in die übersetzt werden soll, meist noch recht ähnlich. Ihre Instruktionen umfassen kleinere Unterprogramme, die in einem Befehl (Makrobefehl) aufgerufen werden können.
Beispielsweise erfordert ein Einadreßautomat zur kompletten Durchführung einer arithmetischen Operation die Befehle:

1. Einlesen des 1. Operanden aus Zelle a
2. Einlesen des 2. Operanden aus Zelle b
3. Ausführen der arithmetischen Operation
4. Abtransport des Ergebnisses in Zelle c

Diese Befehle könnten beim Autocode zu einem Befehl (Makrobefehl) zusammengefaßt werden:

Führe mit a und b die Operation nach c aus.

Die Struktur vieler Automaten verlangt den Aufbau eines leistungsfähigen Autocodes. Das trifft insbesondere für Kleinrechner zu, bei denen man aus ökonomischen Gründen eine minimale Anzahl von Schaltkreisen verwendet, die zudem mehrfach genutzt werden.
So hat z. B. der D 4a nur ein Register, den Akkumulator und operiert unmittelbar mit dem Arbeitsspeicher, einer Magnettrommel (Bild 108).
Das Befehlssystem ist sehr einfach und umfaßt lediglich einige Mikrobefehle, aus denen dann über einen Autocode Makrobefehle aufgebaut werden müssen.
Betrachten wir dies an einem Beispiel.
Das 33 Bit umfassende Wort des D 4a hat für das Befehlswort nebenstehende Aufteilung.
Der D 4a ist ein Einadreßautomat. Der Speicher umfaßt 128 Bahnen zu jeweils 32 Sektoren also 4096 Zellen zu 32 Bit mit einem Prüfbit.

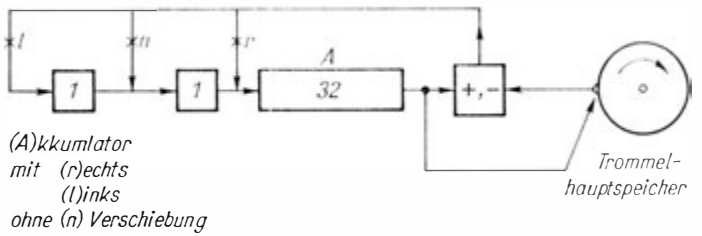
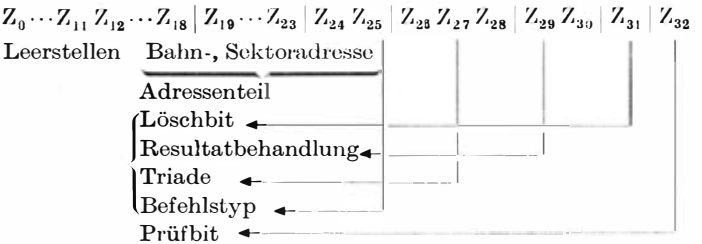


Bild 108. Prinzipskizze der Arbeitsweise von Rechenwerk und Speicher des D 4a



Interessant ist der Operationsteil. Die einzelnen Zeichengruppen haben die in der Tabelle 10 aufgeführte Bedeutung.
Damit können wir insgesamt 256 Mikrobefehle aufbauen, bei denen einige Kombinationen jedoch keine praktische Bedeutung haben. Für das Programmieren in der Maschinensprache selber wäre dieser Befehlsschlüssel jedoch zu schwierig. Beispielsweise können wir über keinen Mikrobefehl verfügen, der die Multiplikation bewirkt. Wir müßten somit bei der algorithmischen Aufbereitung darauf achten, daß wir nur Algorithmen ohne Multiplikation verwenden, oder einen eigenen Multiplikationsalgorithmus aufstellen.
Das kann jedoch durch einen Autocode geschickter gelöst werden. Wir stellen das Programm „Multiplikation" so auf, daß es stets über einen Makrobefehl „Multiplikation" erreichbar ist. Für den D 4a könnte die reine Multiplikation durch folgendes Programm ausgeführt werden:

$Z_{24}Z_{25}$	$Z_{26}Z_{27}Z_2{}^8$	$Z_2{}^9Z_{30}$	Z_{31}	Ziffer	Zeichen	Bedeutung
OO	...	..	.	0...	O	Organisationsbefehle
OO	OOO	..	.	00..	U	Unbedingter Sprung nach Adresse
OO	OOL	..	.	01..	A	Eingabe – Halt
OO	OLO	..	.	02..	Z	Rücksprung oder unbedingtes Halt
OO	OLL	..	.	03..	D	Ausgabe Halt oder Druck
OO	LOO	..	.	04..	N	Sprung bei $\langle AC\rangle < 0$
OO	LOL	..	.	05..	P	Sprung bei $\langle AC\rangle > 0$
OO	LLO	..	.	06..	B	Z-Befehl (unbed. Halt)
OO	LLL	..	.	07..	Ü	Sprung bei Überlauf im Akkumulator
OL	...	..	.	1...	E	Einzelbefehl
LO	...	..	.	2...	G	Gruppenbefehl (32malige Wiederh.)
LL	...	..	.	3...	W	Wiederholungsbefehl (n-malige Ausführung $1 < n \leq 32$)
xx	OOO	..	.	x0..	K	Konjunktion im Akkum.
xx	OOL	..	.	x1..	+	Addition
xx	OLO	..	.	x2..	—	Subtraktion
xx	OLL	..	.	x3..	±	Pus-Minus-Befehl
xx	LOO	..	.	x4..	T	Transport $\langle AC\rangle \rightarrow \langle$Speicher$\rangle$

Z_{24} Z_{25}	Z_{26} Z_{27} Z_{28}	Z_{29} Z_{30}	Z_{31}	Ziffer	Zeichen	Bedeutung
xx	LOL	..	.	x5..	I	Konjunktion im Speicher mit ⟨AC⟩
xx	LLO	..	.	x6..	S	Konjunktion im Speicher mit $\langle\overline{AC}\rangle$
xx	LLL	..	.	x7..	V	Für ⟨AC⟩ > 0 Löschen der angegebenen Zelle
xx	xxx	OO	.	xx0.	H	Resultat nicht verschoben
xx	xxx	OL	.	xx1.	R	Resultat rechts verschoben
xx	xxx	LO	.	xx2.	L	Resultat links verschoben
xx	xxx	LL	.	xx3.	M	Resultat zyklisch links verschoben
xx	xxx	xx	O	xxx0	O	⟨AC⟩ wird nicht gelöscht
xx	xxx	xx	L	xxx1	F	⟨AC⟩ wird vor der Operation gelöscht

Tabelle 10. Befehlsschlüssel des Cellatron D 4a

(x markieren Dualstellen des Befehlswortes, über die bereits verfügt wurde
. kennzeichnet Dualstellen, deren Bedeutung in den folgenden Befehlen erst festgelegt wird)

Programmierungsbeispiel 4: Multiplikation beim D 4a
Voraussetzung:

⟨AC⟩ = 1. Operand
⟨Bb⟩ = 2. Operand (B Spur, b Platzadresse)
⟨AO⟩ = Spur A stehe für die Ausführung der Multiplikation zur Verfügung (AO bezieht sich auf die volle Spur A)

Befehls-nummer	Adr	Befehl Zeichen	Ziffer	Erläuterung
0.	0	G T HO	2400	Gruppenbefehl: Der erste Operand wird aus dem Akkumulator in alle 32 Zellen der Spur A transportiert. Das Resultat bleibt unverändert im Akkumulator, dessen alter Inhalt auch nicht gelöscht wird.
1.	Bb	E + HF	1101	Der erste Operand wird in AC gelöscht. Darauf folgt ohne Resultatsverschiebung: ⟨b⟩ + ⟨AC⟩ ⇒ ⟨AC⟩ Der 2. Operand steht nunmehr im Akkumulator.
2.	AO	G V RO	2710	Gruppenbefehl: Der Inhalt des Akkumulators wird nach jeder Operation um eine Einheit nach rechts verschoben. Die auslaufende Dualziffer wird damit gemäß der Vorschrift + ⇒ O — ⇒ L als Vorzeichen interpretiert, und danach erfolgt die Ausführung des „V-Befehls". Dieser bewirkt, daß in den Zellen der Spur der 1. Operand gelöscht wird, dem eine O des 2. Operanden entspricht.

Befehls-nummer	Adr	Befehl Zeichen	Ziffer	Erläuterung
3.	AO	G + RO	2110	Gruppenbefehl: Es wird ohne vorheriges Löschen des ⟨AC⟩ (denn hier waren ja beim Befehl 2 bereits lauter Werte 0 durch die Linksverschiebung bei der Resultatbehandlung nachgelaufen) mit jeweiliger Linksverschiebung des Resultats über alle Zellen der Spur A summiert. Damit ergeben sich die jeweils höchsten Stellen des Produktes.

Wir wollen uns die Wirkung dieses Programms am Beispiel ansehen. Dazu stellen wir uns einen kleineren Automaten von 4 Sektoren auf einer Spur vor. Die Anzahl der Sektoren muß bei der beschriebenen Arbeitsweise mit der Anzahl der Stellen übereinstimmen. Folglich können wir als Modell auch nur vierstellige Festkommazahlen verknüpfen.
Wir wählen:

O,LLOL × O,LOOL	0,8125 × 0,5625
O,OLLOL	28125
OOLLOL	11250
O,OLLLOLOL	5625
	45000
	0,45703125

Wie wir mit Hilfe der auf der Beilage beigefügten Dualpotenzen feststellen können, stimmen beide Ergebnisse überein. Das angeführte Programm liefert nun die auf Seite 232 dargestellten Operationen.

Wir erhalten somit O,OLLL, was den vier ersten geltenden Ziffern entspricht. Lassen wir zudem die auslaufende Folge OLOL in eine zweite Zelle einlaufen, können wir auch „mit

Stand nach Beendigung des Befehls		Inhalt von A0	A1	A2	A3	AC	(Vorz.)	
Vor 0. Befehl		bel	bel	bel	bel	LLOL	(O)	
0. Befehl	1. Ausführungstakt	LLOL	bel	bel	bel	LLOL	(O)	
	2. Ausführungstakt	LLOL	LLOL	bel	bel	LLOL	(O)	
	3. Ausführungstakt	LLOL	LLOL	LLOL	bel	LLOL	(O)	
	4. Ausführungstakt	LLOL	LLOL	LLOL	LLOL	LLOL	(O)	
1. Befehl		LLOL	LLOL	LLOL	LLOL	LOOL	(O)	
2. Befehl	1. Ausführungstakt	LLOL	LLOL	LLOL	LLOL	OLOO	(L)	
	2. Ausführungstakt	LLOL	OOOO	LLOL	LLOL	OOLO	(O)	
	3. Ausführungstakt	LLOL	OOOO	OOOO	LLOL	OOOL	(O)	
	4. Ausführungstakt	LLOL	OOOO	OOOO	LLOL	OOOO	(L)	
3. Befehl	1. Ausführungstakt	LLOL	OOOO	OOOO	LLOL	OLLO		L
	2. Ausführungstakt	LLOL	OOOO	OOOO	LLOL	OOLL		O
	3. Ausführungstakt	LLOL	OOOO	OOOO	LLOL	OOOL		L
	4. Ausführungstakt	LLOL	OOOO	OOOO	LLOL	OLLL		O

doppelter Genauigkeit" arbeiten oder auf Wunsch die ersten oder letzten geltenden Ziffern auswählen.
Ähnlich können wir nun Makrobefehle für die Division, die Ein- und Ausgabe, Rechenoperationen in Gleitkommadarstellung, Anschluß und Nutzung peripherer Geräte u. a. aufbauen und mit dem Automaten, hier also dem D 4a, als „software" zur Verfügung stellen. Dies erfolgt erfreulicherweise auch bereits vom Herstellerbetrieb und wird als Programminformation zum D 4a dem Kunden und Interessenten übergeben.
Vom Speicher werden dazu eine Anzahl von Zellen blockiert, die das Übersetzerprogramm des Autocodes aufnehmen und uns so eine recht bequem zu handhabende Programmierung ermöglichen. Das Programm ist dabei noch im üblichen Sinne aus Befehlen (Makrobefehlen) zusammengesetzt. An die Programmiersprache werden damit noch keine übermäßigen Forderungen gestellt.

3.5.6. Programmiersprachen

Heute pflegt man drei Komplexe von Programmiersprachen zu unterscheiden:

problemorientierte Sprachen,
maschinenorientierte Sprachen,
Maschinensprachen.

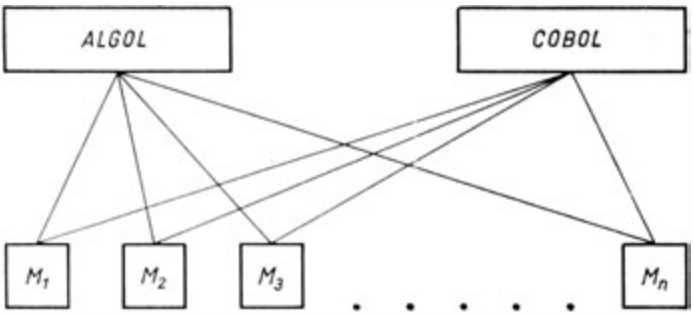
Maschinensprachen haben wir bereits ausführlich kennengelernt. Hierzu gehören die Maschinensprachen des Cellatron SER 2, des D 4a, des ZRA 1 u. a. Vielfach rechnet man aber auch die Autocodes zu den Maschinensprachen.
In den letzten Jahren wurden vor allem problemorientierte Sprachen entwickelt. Sie ermöglichen es, die algorithmische Erfassung des mathematischen Modells unmittelbar in der problemorientierten Sprache zu formulieren. Danach kann dann aus dieser Formulierung vom Automaten mit Hilfe eines Compilers das Maschinenprogramm aufgestellt werden.
Der Problematik umfassender Aufgabenkomplexe entsprechend, haben sich eine Reihe sogenannter operativer Sprachen entwickelt:

FORTRAN Formula Translation System ALGOL 60 Algorithmic Language	für technisch-wissenschaftliche Probleme
COBOL Common Business Oriented Language	für kaufmännische Verfahren
COMIT	für die automatische Übersetzung von Sprachen
AUTOPROMT	zur Steuerung von Werkzeugmaschinen
STRESS	für baustatische Probleme
COCO	für geometrische Probleme des Bauwesens

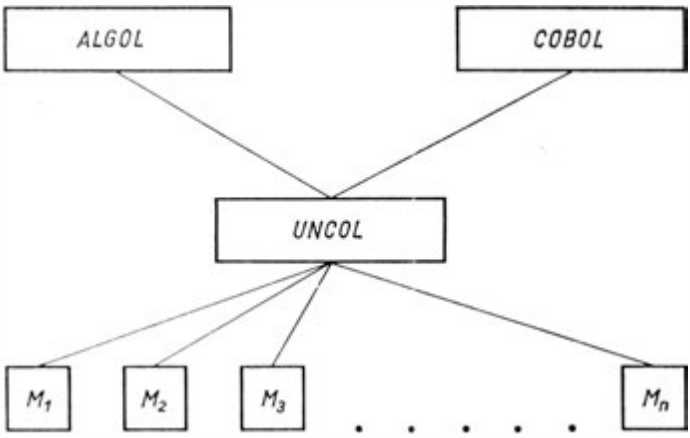
Daneben gibt es noch sogenannte deskriptive Sprachen, deren Entwicklung aber erst in den Anfängen steht.
So wird von einem *Sortierprogramm-Generator* berichtet. In diesem Programm ist die Sortieraufgabe beschrieben, nach der das Sortierprogramm automatisch erzeugt wird.
Ein anderes Programm ist der *Formular-Generator*, mit dessen Hilfe aus vorgegebenen Listen die Informationen entnommen, verarbeitet und ausgegeben werden können, z. B. zum Ausschreiben von Rechnungen.
Es wird auch an einem Generator in deskriptiver Sprache gearbeitet, der Analogrechner-Programme erzeugen kann.
Für die modernen problemorientierten Sprachen benötigten wir sogenannte Compiler, die ein automatisches Aufstellen des Maschinenprogramms nach der Formulierung in der allgemeinen Sprache ermöglichen. Das Aufstellen eines solchen Übersetzungsprogramms (*Compiler*-Programm genannt) ist relativ schwer und benötigt viel Zeit. Daraus könnten wir folgern, daß die vielen problemorientierten Sprachen eher erschwerend als erleichternd wirken. Für jede Sprache müßte ein gesondertes Übersetzungsprogramm angefertigt werden.
Maschinenorientierte Sprachen sollen hier Abhilfe schaffen. Damit die Vorteile der speziellen problemorientierten Sprachen für

die einzelnen Gebiete genutzt werden können, andererseits aber nicht zu viele Übersetzungsprogramme anzufertigen sind, hat man maschinenorientierte Sprachen entwickelt. UNCOL (Universal Computer Oriented Language) ist als einheitliche Zwischensprache vorgeschlagen. Sie ermöglicht eine Übersetzung aus der problemorientierten Sprache, berücksichtigt also die dortigen Besonderheiten. Anschließend kann durch ein zweites Übersetzungsprogramm das Maschinenprogramm hergestellt werden. UNCOL berücksichtigt also auch die Besonderheiten des jeweiligen Rechenautomatentyps.
Diese Möglichkeiten erleichtern die Anwendung problemorientierter Sprachen (Bild 109).
Eine der problemorientierten Sprachen hat sich in den letzten Jahren besonders durchgesetzt – ALGOL 60. Wir wollen sie daher als Beispiel nehmen und auf eine sogenannte Subset-Form reduzieren. Die von uns benutzte Subset-Form soll lediglich die Teile der allgemeinen Form von ALGOL 60 umfassen, die wir zur Darstellung der beiden uns schon bekannten Programme „Fakultät" und „UP Quadratische Gleichung" benötigen.
ALGOL 60 – entstanden aus algorithmic language (algorithmische Sprache) – wurde in der heute vorliegenden Fassung auf einer internationalen Tagung in Paris vom 11. bis 16. Januar 1960 vereinbart. Bei ihr sind Syntax (Grammatik) und Semantik (Bedeutung der Ausdrücke) exakt formuliert, wobei man sich zur Definition der Syntax einer Metasprache bedient. Das ist erforderlich, da ja die neue Sprache mit Symbolen und Worten beschrieben wird, die selber Elemente der neuen Sprache sind. *J. W. Backus* hat hierzu 1959 metalinguistische Formeln vorgeschlagen, die sich allgemein durchgesetzt haben und ihm zu Ehren die Bezeichnung Backussche Normenformen erhielten. Ohne auf die Beschreibung der Backusschen Normalform selber einzugehen,[1] sei lediglich ihr Aufbau erklärt.

[1] Vgl.: Kerner/Ziehlke: Einführung in die algorithmische Sprache ALGOL 60. Leipzig: B. G. Teubner Verlagsgesellschaft 1966, S. 39/40 und Naur, Peter: Revidierter Bericht über die algorithmische Sprache ALGOL 60. Berlin: Akademie-Verlag 1966, S. 10



2n Übersetzungsprogramme erforderlich



n+2 Übersetzungsprogramme erforderlich

Bild 109. Schematische Darstellung des automatischen Programmierens

Wir benötigen die Elemente:

Definitionszeichen	: : =
Oder-Zeichen	|
Klammer	⟨ ⟩

und können damit die sogenannte Backussche Normalform wie folgt aufbauen:

⟨Subjekt⟩ :: = ⟨Prädikat⟩ bzw.
⟨Subjekt⟩ :: =

Mit Hilfe der Backusschen Normalform werden für ALGOL 60 definiert:

Grundbegriffe	(25 Definitionen)
Ausdrücke	(37 Definitionen)
Anweisungen	(30 Definitionen)
Vereinbarungen	(23 Definitionen)

Wir wollen uns hier, wie oben bereits angedeutet, auf eine Auswahl beschränken. Insbesondere wollen wir uns die Verwendung der Backusschen Normalform nur am Beispiel der Grundbegriffe ansehen. Sie umfassen Grundsymbole, Bezeichnungen, Zahlen und Zeichenketten.
Wir betrachten die Grundsymbole. In der Form unseres Subset-ALGOL ist

⟨Grundsymbol⟩ :: = ⟨Buchstabe⟩ | ⟨Ziffer⟩ | ⟨logischer Wert⟩ | ⟨Begrenzer⟩

d. h., Grundsymbol wird definiert als Buchstabe oder Ziffer oder logischer Wert oder Begrenzer. Wie diese Elemente definiert werden, geben die folgenden Backusschen Normalformen an:

⟨Buchstabe⟩ :: = a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z
(also alle kleinen lateinischen Grundbuchstaben ohne Umlaute)

⟨Ziffern⟩ :: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

⟨logischer Wert⟩ :: = true | false
(richtig oder falsch)

⟨Begrenzer⟩ :: = ⟨Operator⟩ | ⟨Trennzeichen⟩ | ⟨Klammer⟩ | ⟨Vereinbarungszeichen⟩ | ⟨Spezifikationszeichen⟩

d. h., Begrenzer sind fünf unterschiedliche Zeichenmengen.

⟨Operator⟩ :: = ⟨arithmetischer Operator⟩ | ⟨Vergleichsoperator⟩ | ⟨logischer Operator⟩ | ⟨Folgeoperator⟩

d. h., auch die Operatoren sind in vier Teilmengen untergliedert:

⟨arithmetischer Operator⟩ :: = + | − | × | / | ↑
⟨Vergleichsoperator⟩ :: = < | ≦ | = | ≧ | > | ≠
⟨logischer Operator⟩ :: = ≡ | ⊃ | ∨ | ∧ | ¬
⟨Folgeoperator⟩ :: = goto | if | then | else | for | do
⟨Trennzeichen⟩ :: = , | . | 10 | : | ; | := | step | until | while | comment
⟨Klammer⟩ :: = ( ) | [ ] | ‘ ’ | begin | end
⟨Vereinbarungszeichen⟩ :: = own | Boolean | integer | real | array | switch | procedure
⟨Spezifikationszeichen⟩ :: = string | label | value.

Diese Grundsymbole sind nahezu vollständig aus der allgemeinen Sprache ALGOL 60 übernommen. Die Bedeutung gleicht meist der bekannten Anwendung in der Arithmetik. Eine besondere Bemerkung erfordern die als Grundsymbol übernommenen Wörter. Sie stammen aus dem Englischen und haben im Deutschen folgende Bedeutung:

array	Feld	label	Marke
begin	Beginn, Anfang	own	eigen
Boolean	Boolesch	procedure	Prozedur
comment	Bemerkung	real	reell
do	tue, führe aus	step	Schritt
else	sonst	string	Zeichenfolge
end	Ende	switch	Verteiler
false	falsch	true	richtig
for	für	then	dann
goto	gehe nach, Sprung n.	until	bis

if	wenn	while	solange
integer	ganzzahlig		
value	Wert		

In der Programmierungssprache ALGOL 60 jedoch haben die Grundsymbole bestimmte Bedeutung für das Programm und die Übersetzung. Sie werden für alle Landessprachen wie vorgegeben übernommen. Um sie eindeutig als ALGOL-Grundsymbole zu charakterisieren und von gleichlautenden Wörtern der Rechenprogramme in englischer Sprache zu unterscheiden, werden sie unterstrichen.

Zu den Grundbegriffen gehören Bezeichnungen als Folgen von Buchstaben und/oder Ziffern, die aber stets mit einem Buchstaben beginnen, Zahlen in der Form ganzer Zahlen oder Zahlen mit beweglichem Komma und Vorzeichen sowie Zeichenketten als Folgen von Grundsymbolen, die in ‘ und ’ eingeschlossen sind.

Ausdrücke umfassen arithmetische, logische und Zielausdrücke. Sie haben in der allgemeinsten Darstellung die Form

if $b = 0$	then $u + v$	else $(p - q) \uparrow 2$	(arithm. A)
if $b < 0$	then $u \supset v$	else $\neg a$	(logisch. A)
if $b > 0$	then 17	else 21	(Zielausdruck)

also wenn ..... dann ..... sonst .....;

Eine besondere Rolle spielen dabei die Funktionen. Sie werden über Unterprogramme, hier Prozeduren genannt, bereitgestellt. Eine Ausnahme bilden die Standardfunktionen, die als „Makrobefehle“ in ALGOL übernommen wurden. Standardfunktionen sind:

abs	(A) absoluter Betrag des Wertes des Ausdruckes A,
sign	(A) Signum des Wertes von A (+1 für $0 <$ A, 0 für A $= 0$ und —1 für A < 0),
sqrt	(A) Quadratwurzel des Wertes A,
sin	(A) Sinus des Wertes von A,

cos (A) Cosinus des Wertes von A,

arctan (A) Hauptwert des Arcustangens des Wertes von A,

ln (A) natürlicher Logarithmus des Wertes von A und

exp (A) Exponentialfunktion des Wertes von A.

Variable können durch Bezeichnungen angegeben werden (z. B. $x1$, omega 3b u. a.). Bei indizierten Variablen werden die Grenzen des Wertebereiches für den oder die Indizes in eckigen Klammern angegeben.
So ist x_i mit $i = 1, 2, 3$ gleichbedeutend mit x [1 :3]
oder a_{ij} mit $i = 1, 2, 3, \ldots, n$ und $j = 1, 2, 3, \ldots, m$
entsprechend mit

a [1 :n, 1 :m].

Bei der Rechnung selber wird der Index ebenfalls mittels eckiger Klammern gekennzeichnet, also entspricht

$$x_3 := \frac{a_3}{a_1 \cdot a_2}$$

in ALGOL

x [3]: = a[3]/a[1]/a[2].

Damit können wir Anweisungen aufbauen, wobei man unterscheidet:
Ergibtanweisung (ähnlich der Plangleichung):

$s := u + v$

Sprunganweisung:

goto marke a 3 (gehe zu, mit Angabe des Zieles)

bedingte Anweisung:

if dis $>$ 0 then sqrt (dis) else
if dis $=$ 0 then sqrt (dis) $=$ 0 else
if dis $<$ o then sqrt (abs(dis))

(falls die Diskriminante positiv ist, ziehe hieraus die Quadratwurzel, sonst, falls die Diskriminante gleich Null ist, ist auch die Quadratwurzel aus der Diskrimi-

nante Null, sonst, falls die Diskriminante negativ ist, ziehe die Quadratwurzel aus dem absoluten Betrag der Diskriminante).

Laufanweisung:

for i: = 1 step 1 until n do

(für i von 1 in 1-er Schritten bis n führe aus:) und

Prozeduranweisungen, auf die wir später eingehen.
Nunmehr können wir schrittweise zum Aufbau eines Rechenprogramms kommen.
Das Kernstück ist die Anweisung. Sie gibt den „Makrobefehl" für den Automaten.
Mehrere Anweisungen können zu einer Verbundanweisung zusammengefaßt werden. Sie werden dann in die Klammern begin und end in der Form

begin A; A; A; ... A end

eingefügt.
Soll beispielsweise das Produkt $f \times k$ gebildet werden und anschließend k sowie das neu errechnete f ausgegeben werden, können wir dies in einer Verbundanweisung

```
begin f: = f × k;
      outinteger (dr, k);
      outinteger (dr, f);
end
```

zusammenfassen.
In ALGOL müssen jedoch alle Größen vor ihrer Verwendung über eine Wertzuweisung vereinbart werden. Wir können dabei unterscheiden Größen vom Typ

Boolean	Wertebereich: true, false
integer	Wertebereich: ganze Zahlen
real	Wertebereich: reelle Zahlen

Sollen ganze Wertefelder vereinbart werden, so können wir das Vereinbarungszeichen array (Feld) verwenden. Dabei sind wiederum zu unterscheiden:

```
Boolean  array
integer  array
real     array
```

Zudem können wir, um Programme leichter lesbar zu machen, Kommentare einflechten. Die Kommentare werden durch comment begonnen und mit ; beendet. Sie können in der Landessprache eingefügt werden, da der Automat sie bei der Übersetzung unberücksichtigt läßt.
Werden Vereinbarungen und Anweisungen durch begin und end umschlossen, so spricht man von einem Block.
Ein Programm ist in ALGOL schließlich ein solcher Block, der auch die Vereinbarungen aller vorkommenden Größen selbst enthält.
Zur Tabellierung des Argumentes k und der Fakultät von k, also $f := k!$, benötigen wir folgendes Programm:

```
begin comment Tabellierung der Argumente
                und Funktionswerte                    } Verein-
                „Fakultät" von 1 bis n;               } barungsteil
      integer   n, k, f;

      ininteger (b, n);                               Einlesen des
                                                      Wertes n vom
                                                      Band b
      f := 1;
      for k := 1 step 1 until n do                    Laufanweisung
      begin f := f × k;
            outinteger (dr, k);                       } Verbund-
            outinteger (dr, f);                       } anweisung
      end;
end Programm;
```

Nach dem Kommentar und der Vereinbarung wird also n aus dem Eingabegerät b eingelesen. Dann wird der Ausgangswert für f gesetzt und schließlich über eine Laufanweisung der Wert der Fakultät errechnet, Argument und Funktionswert über den Drucker dr ausgegeben. Wir müssen dabei beachten, daß sich die Laufanweisung auf die gesamte Verbundanweisung bezieht.

Mit diesen Beispielen konnten wir verständlicherweise die Problematik des Programmierens in ALGOL 60 nur andeuten. Wir müssen uns damit intensiver beschäftigen, um ihre vielfältigsten Möglichkeiten voll zu nutzen. Beispielsweise läßt sich unser Programm „Fakultät" durch Anwenden sogenannter rekursiver Prozeduren wesentlich vereinfachen. Doch können wir hierauf nicht näher eingehen.

ALGOL 60 ist außerordentlich flexibel und so besonders für wissenschaftlich-technische Aufgaben geeignet. Die umfangreiche Datenorganisation kommerzieller Rechnungen bereitet allerdings Schwierigkeiten. Daher wurde für diese Aufgabe die Programmiersprache COBOL entwickelt.

Selbstverständlich erhält das Programmieren in einer problemorientierten Sprache wie ALGOL 60 erst dann seinen eigentlichen Sinn, wenn ein entsprechender Übersetzer (Compiler, speziell für ALGOL 60 ein ALCOR, d. h. ALGOL-Conventer) für den zu nutzenden Automaten vorhanden ist. Die meisten modernen Automaten besitzen Compiler. Auch für den ZRA 1 wurde ein solches Gerät entwickelt.

Hier müssen wir jedoch noch einige Bemerkungen zum Effekt der Übersetzerprogramme bezüglich der Rechenzeit und der Speicherbelegung anschließen. Automatische Übersetzerprogramme haben gegenüber ausgefeilten, manuell aufgestellten Programmen einen Nutzeffekt von 5:1 bis 2:1. Wir müssen also die Häufigkeit der Anwendung eines Programms untersuchen, um gegebenenfalls nach der ALGOL-Darstellung auch noch ein effektiveres, manuell aufgestelltes Programm anfertigen zu lassen.

Mehrfach benutzte Programmteile können zur wiederholten Verwendung als Unterprogramm formuliert werden. Das erfolgt in ALGOL 60 über die Prozedur. Für die Prozeduren gelten demzufolge alle Einschränkungen, die bereits bei dem Unterprogramm erwähnt wurden.

Ohne hier auf weitere Einzelheiten eingehen zu können, sei die Prozedur „quadratische Gleichung" angeführt. Sie soll hier jedoch in einen Rahmen eingebettet werden, damit sie unmittelbar „für sich" verwendet werden kann.

```
prozedur quadratische gleichung
a[1] × x ↑ 2 + a[2] × x + a[3] = 0                    ┐
begin integer i;                                      │ Rahmen
      array a[1:3], x[1:4];                           ┘
      procedure qugl (a, x);                          ┐ Prozedurnamen
      array a, x;                                     │ Prozedurkopf
      comment a[1] ≠ 0;                               ┘
      begin real d, w;                                ┐
            x[1]: = x[3]: = —a[2]/a[1]/2;             │
            x[2]: = x[4]: = 0;                        │
            d: = a[2] × a[2] —4 × a[1] × a[3];        │
            w: = 1/2/a[1] × sqrt (abs (d));           │
            if a > 0 then                             │
            begin x[1]: = x[1] +                      │ Prozedur-
                  + (if a[2]/a[1] > 0 then — 1        │ hauptteil
                  else + 1) × w;                      │
            x [3]: = a [3]/a [1]/x [1];               │
            end else                                  │
            if d < 0 then                             │
            begin x[2]: = w;                          │
                  x[4]: = —w;                         │
                  end;                                │
      end procedure;                                  ┘
      inreal (b, a);                                  ┐
      for i: = 1, 2, 3, do                            │
      outreal (dr, a[i]);                             │ Rahmen
      qugl (a,x);                                     │ Prozeduraufruf
      for i: = 1 step 1 until 4 do                    │
      outreal (dr, x[i]);                             ┘
end programm;.
```

4. Analogrechner

4.1. Historischer Abriß

Rechenhilfsmittel, die auf dem Analogieprinzip beruhen, sind bereits seit längerem bekannt. Das einfachste Gerät dieser Art ist der Rechenschieber.

Mit dem 1876 von Lord *Kelvin (W. Thomson)* aufgestellten Prinzip der Rückkopplung war die Grundidee für den Bau von Analogrechnern vorhanden.

Das Prinzip der Rückkopplung verlangt, daß ein System von Rechenelementen, die den einzelnen Rechenschritten eines mathematischen Systems entsprechen, aufgebaut wird. Dies System wird durch einen Anfangswert angeregt, dabei entsteht ein Ergebnis, das wieder an den Anfang des Systems zurückgekoppelt wird.

Entsprechend der Schaltung, den verwendeten Koeffizienten und Parametern sowie anderern Elementen kann das System konvergieren oder divergieren. Bei Konvergenz spielt sich das System auf einen bestimmten Wert ein (meist Null). Bei Divergenz kann nur ein bestimmter Bereich untersucht werden, ehe das System „übersteuert".

Das Prinzip der Rückkopplung entspricht dem mathematischen Verfahren der Iteration. Aus dem Prinzip können wir bereits ersehen, daß die unabhängige Variable der zu behandelnden mathematischen Aggregate – meist gewöhnliche Differentialgleichungen – stets die Zeit sein muß. Gleichungen, die nicht von der Zeit abhängen, können leicht in eine Zeitfunktion überführt werden.

Lange Zeit wurde die geniale Idee Lord Kelvins kaum beachtet. Das lag einesteils daran, daß Wissenschaft und Forschung noch keinen Bedarf für Rechenmaschinen hatten, zum anderen war es aber auch mit den damals bekannten Bauelementen – die Elektronenröhre war noch nicht erfunden – sehr schwierig, eine befriedigende technische Lösung zu schaffen.

Erst 1914 entwickelte *Udo Knorr* eine mechanische Integrieranlage zum Lösen gewöhnlicher Differentialgleichungen. Diese Anlage wurde von der Deutschen Reichsbahn zur Fahrplanaufstellung benutzt. Bis 1931 sind zwar verschiedene Varianten dieser Anlage fertiggestellt worden, sie zeigten aber keine wesentlichen Verbesserungen. Erst die von *V. Bush*, Cambridge/Mass. (USA), gefertigte mechanische Analogierechenmaschine war eine wesentliche Weiterentwicklung auf diesem Gebiet. Während des zweiten Weltkrieges baute er gemeinsam mit *Caldwell* eine elektromechanische Großanlage mit 18 Integratoren, Tausenden Röhren und Relais und etwa 150 Elektromotoren.

Auch in Deutschland wurde während des Krieges intensiv an diesem Problem gearbeitet, aber ein großer Teil der Ergebnisse ging verloren.

1945 gab *J. M. Jackson* erstmalig die Daten einer vollelektronischen Maschine an, und schon wenige Jahre später wurden in allen Industrieländern hochentwickelte vollelektronische Analogrechner für den universellen Einsatz gebaut. Gegenwärtig geht die Entwicklung von den äußerlich großen, hohe Energie aufnehmenden Röhrengeräten über zu kleinen transistorisierten Geräten, wobei die Verkleinerung keine Einschränkung der mathematischen Möglichkeiten mit sich bringt.

In der DDR wurde an verschiedenen Stellen auf dem Gebiet der Analogrechentechnik gearbeitet. Als erstes Gerät stellte das damalige Wissenschaftlich-technische Büro für Gerätebau, Berlin, den UNIMAR 1, einen mittleren Analogrechner, fertig. Dieser Typ wurde nach Einbau einiger Verbesserungen als UNIMAR 2 in geringer Stückzahl gebaut.

An der Friedrich-Schiller-Universität Jena konnte kurze Zeit später ein mechanischer Analogrechner für die Ausbildung genutzt werden. Er war nach Vorstellungen von Prof. Dr. *Weinel* angefertigt worden.

Die industrielle Fertigung von Analogrechnern ging in der Deutschen Demokratischen Republik von der Hochschule für Elektrotechnik Ilmenau aus. Prof. Dr. *Winkler* entwarf einen elektronischen Analogrechner EAR, der dann in kleiner Serie als EAR 6 im damaligen VEB Rechenmaschinenwerk „Archimedes" Glashütte gefertigt wurde.
Dieser Betrieb stellte darauf seine Produktion von mechanischen Tischrechenmaschinen auf elektronische Analogrechner um und brachte den „endim 2000" als mittleren Analogrechner mit 64 Recheneinheiten auf den Markt.
Der „endim 2000" ist das Standardgerät in der DDR. Es sind rund 30 Geräte dieses Typs im Einsatz. Obwohl dieses Gerät recht gute Leistungskennziffern hat, ist damit der Welthöchststand nicht erreicht.

4.2. Aufbau und Arbeitsweise eines Analogrechners

Moderne Analogrechner bestehen im allgemeinen aus den folgenden Hauptgruppen:

1. Rechenelemente,
2. Programmfeld,
3. Steuergerät,
4. Ausgabe- und Kontrollgerät,
5. Stromversorgung.

Das Zusammenwirken der Hauptgruppen wird im Bild 110 gezeigt. Je nach Art und Umfang der zu behandelnden Probleme ist der in den einzelnen Gruppen getriebene technische Aufwand unterschiedlich. Es gibt Kleinrechner als Tischgeräte mit maximal 20 Rechenelementen und Großanlagen, die ganze Räume füllen und bis etwa 600 Rechenelemente haben. Bild 111 zeigt den mittleren Analogrechner „endim 2000" vom VEB Rechenelektronik Glashütte, der mit 92 Rechenelementen ausgestattet ist, von denen maximal 64 gleichzeitig in Betrieb sein können.
Der gesamte linke Teil des Automaten wird von den Rechenelementen eingenommen. Im oberen Teil der rechten Seite sieht

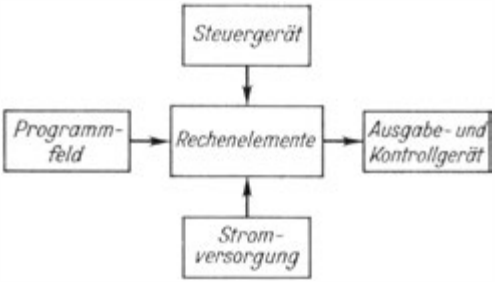


Bild 110. Prinzipieller Aufbau eines Analogrechners

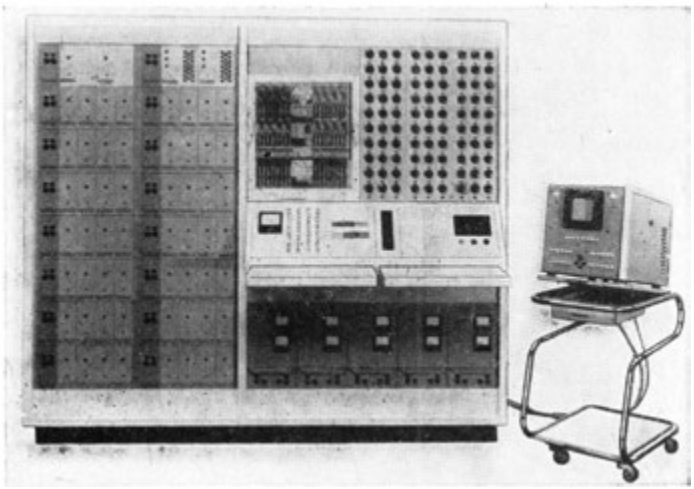


Bild 111. Mittlerer Analogrechner „endim 2000"

man neben dem Programmierfeld mit 2250 Buchsen das Potentiometerfeld mit insgesamt 90 Koeffizientenpotentiometern. Darunter sind das Meßinstrumentenfeld und das Steuerfeld angeordnet, und unter diesen Elementen befindet sich die Stromversorgung. Neben dem Rechner steht ein Spezialoszillograf, der die Funktion des Ausgabe- und Kontrollgerätes übernimmt.

4.2.1. Rechenelemente

Die Rechenelemente bilden das Kernstück jedes Analogrechners. Man geht von dem Prinzip aus, für die auftretenden Rechenoperationen analoge Rechenelemente zu finden. Durch entsprechenden Zusammenschluß dieser Rechenelemente kann dann der mathematische Prozeß nachgebildet und analog gelöst werden. Meist verwendet man die auf Bild 112 dargestellten Rechenelemente.

Die dort verwendeten Symbole weisen zwar auf eine elektronische Lösung der mathematischen Operationen hin, grundsätz-

		Rechenelement	Programmiersymbol	Operation
lineare Rechenelemente	1	Koeffizienten-potentiometer	U_e — a — U_a	$U_a = a\,U_e$ $0 \leqq a \leqq 1$
	2	Vorzeichenumkehr Inverter	U_e — 1 — U_a	$U_a = -U_e$
	3	Summator	U_{e1}, U_{e2}, U_{e3}, U_{en} — a_1, a_2, a_3, a_n — U_a	$U_a = -\sum_{i=1}^{n} a_i\,U_{ei}$
	4	Summenintegrator	U_{e1}, U_{e2}, U_{e3}, U_{en} — a_1, a_2, a_3, a_n — U_a, (U_0)	$U_a = -\sum_{i=1}^{n} a_i \int U_{ei}\,dt + U_0$
nichtlineare Rechenelemente	5	Funktions-multiplikator	U_{e1}, U_{e2} — Π — U_a	$U_a = U_{e1} \cdot U_{e2}$
	6	Funktionsgenerator (Funktionswertgeber)	U_e — $F(\ )$ — U_a	$U_a = -F(U_e)$
nichtlineare Rechenelemente	7	Komparator	U_{e1}, U_{e2} — U_a	Der Kontakt liegt oben, wenn $U_{e1} > U_{e2}$ unten, wenn $U_{e1} < U_{e2}$

Bild 112. Symbole der Rechenelemente eines Analogrechners

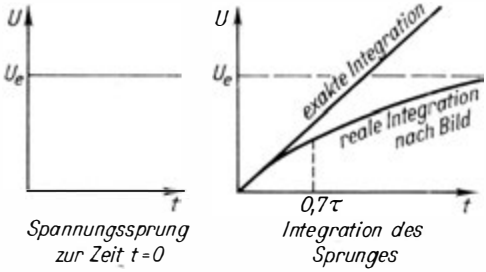


Bild 113. Skizze zum brauchbaren Integrationsbereich

lich sind aber auch mechanische, hydraulische oder andere Elemente möglich.

Vorwiegend werden jedoch elektronische Rechenelemente verwendet. Für sie steht als unabhängige Variable nur die Zeit t zur Verfügung. Als abhängige Variable könnten sowohl die Spannung $U(t)$ als auch der Strom $I(t)$ benutzt werden. Da es aber schaltungstechnisch wesentlich einfacher ist, einen Spannungsverstärker als einen Stromverstärker aufzubauen, wird als abhängige Variable nur $U(t)$ benutzt.

Die Arbeitsweise elektronischer Rechenelemente erfordert verständlicherweise umfassendere Kenntnisse der Elektrotechnik, die zu vermitteln den Rahmen dieses Buches überschreiten

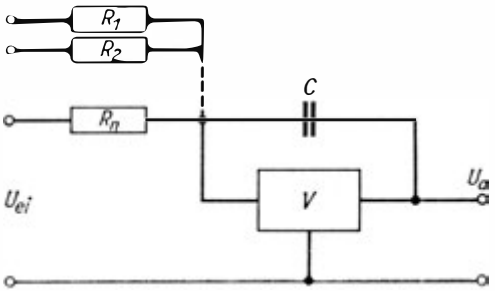


Bild 114. Summenintegrator

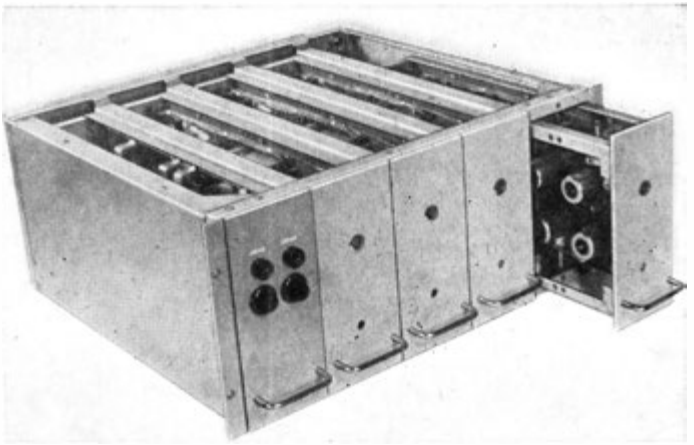


Bild 115. Zentraleinschub des „endim 2000" mit vier Operationsverstärkern

würde. Wir möchten hierzu auf die umfangreiche vorhandene Literatur hinweisen.

Prinzipiell muß für jedes Rechenelement ein Verstärker sehr großer negativer Verstärkung eingesetzt werden ($A \ll 0$). Damit ergeben sich einige Schwierigkeiten in der technischen Realisierung mathematischer Prozesse. Bild 113 zeigt eine exakte und reale Kurve der Integration.

Hieraus können wir ersehen, daß der brauchbare Bereich nur bei $t < \tau$ liegen kann.

Um diesen Bereich auch mit einem entsprechend großen Ausgangssignal benutzen zu können, wird eine Schaltung nach Bild 114 angewendet, die wir als Beispiel anführen möchten. Dort ist ein Summenintegrator dargestellt, der sich von einem einfachen Integrator nur durch mehrere Eingänge unterscheidet. Die Integrationskonstante U_0 erhält der Kondensator vor Beginn der Rechnung in Form einer Anfangsladung.

Auf ähnliche Weise könnte auch die Differentation ermöglicht werden. Während jedoch die Integration auf Fehler und Ungenauigkeiten glättend wirkt, führt die Differentation zum Aufrauhen und damit leicht zur Instabilität der Rechenschaltungen.

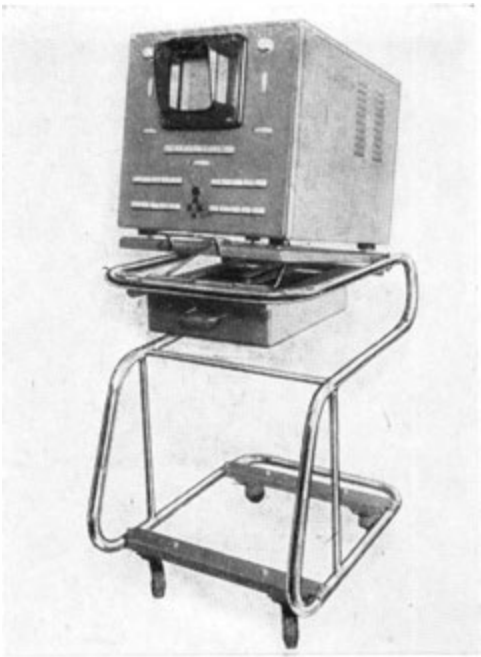


Bild 116. Oszillograf als Kontrollgerät für den „endim 2000"

Die Differentation läßt sich meist durch mathematische Umformungen vermeiden. Da sie technisch auch nur relativ unvollkommen realisierbar ist, werden derartige Rechenelemente nicht hergestellt.

4.2.2. Ergebnisanzeige

Das Endergebnis der Rechnung liegt als Spannungsfunktion $U_a(t)$ vor, die in irgendeiner Form sichtbar gemacht werden muß. Oft interessieren auch noch Zwischenergebnisse, so daß Auswertegeräte mit mehreren Kanälen benötigt werden.
Die einfachste Anzeige geschieht über Ein- oder Zweistrahl-Oszillografen, die vom Rechner nach der Rechenzeit gesteuert werden. Die Lösungsfunktion ist sofort sichtbar und kann zur

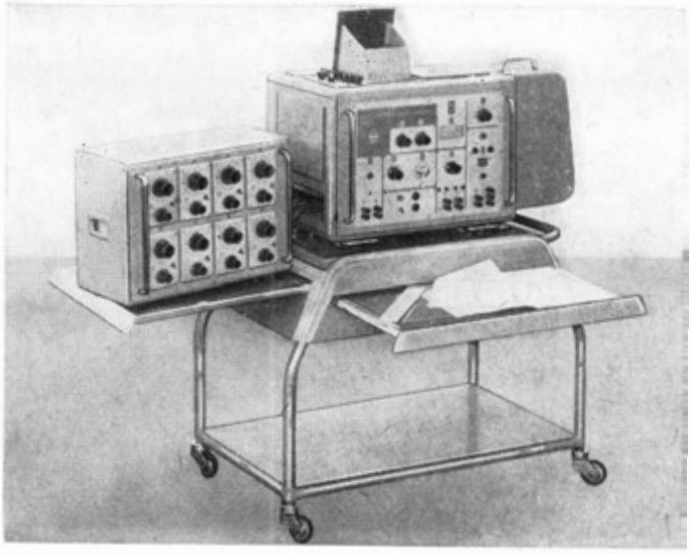


Bild 117. Mehrschleifen-Oszillograf des VEB Meßgerätewerk Zwönitz

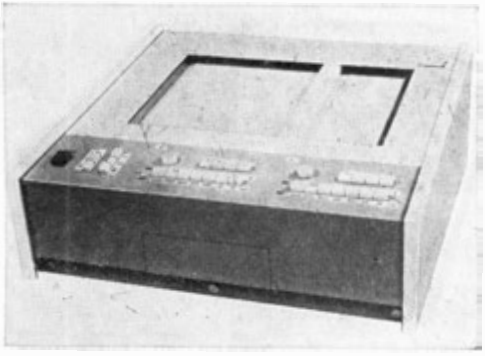


Bild 118. *xy*-Schreiber „endim 2100"

besseren Auswertung fotografiert werden. Vorteilhaft ist die hohe Anzeigegeschwindigkeit, wodurch auch sehr schnell verlaufende Vorgänge aufgezeichnet werden können. Dafür ist die Genauigkeit relativ gering. Oszillografen werden daher meist nur als Kontrollgeräte, nicht aber zur exakten Messung benutzt (Bild 116).

Zur mehrkanaligen Auswertung werden Lichtpunktlinien-Schreiber und Mehrschleifen-Oszillografen (Bild 117) verwendet, die jedoch zum Teil den Nachteil haben, daß das Ergebnis erst nach der Entwicklung des Filmes zur Verfügung steht.

Bei diesen Ausgabegeräten ist nur die Zeit als unabhängige Variable möglich. Die Darstellung der Lösungsfunktion in der Form $g = f(x)$ entfällt hier.

Um aber auch Funktionen der letzten Art aufzeichnen zu können, werden xy-Schreiber benutzt (Bild 118). Das sind Tintenschreiber, die 2 getrennte Eingänge, einen für die x-Richtung und einen für die y-Richtung, haben und damit in beiden Richtungen unabhängig voneinander schreiben können. Die Genauigkeit der xy-Schreiber ist relativ groß, ihr Fehler liegt zwischen 0,1 und 0,5 %.

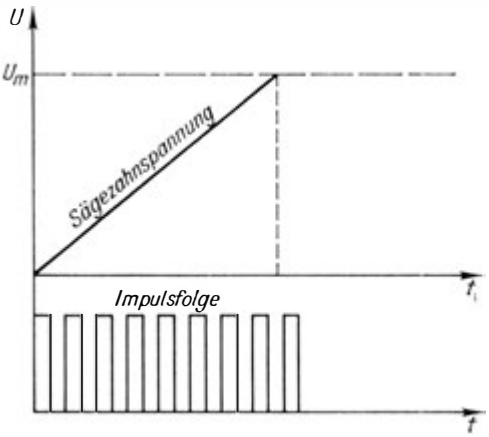


Bild 119. Prinzip des Sägezahnverschlüsselers

Oft benötigt man zur digitalen Weiterverarbeitung des Problems nicht die analogen Werte als Kurven, sondern digitale Werte in Form von Tabellen als diskrete Ordinatenwerte der Kurven. Hierfür wurden Analog-Digitalumwandler entwickelt, von denen es zwei Ausführungsarten gibt, den Sägezahnverschlüsseler und den Stufenverschlüsseler.

Beim Sägezahnverschlüsseler (Bild 119) schwingen zu einem genau definierten Zeitpunkt ein Sägezahngenerator und ein Impulsgenerator an. Erreicht die Sägezahnspannung die Amplitude der Meßspannung U_m, so wird der Impulsgenerator gestoppt. Die Anzahl der bis zu diesem Zeitpunkt erzeugten Impulse ist dann ein Maß für die Größe der zu messenden Spannung.

Beim Stufenverschlüsseler wird die Meßspannung mit einer in diskreten Schritten veränderlichen Kompensationsspannung verglichen.

Sind beide Spannungen gleich groß, wird das Schaltwerk stillgesetzt und die Schalterstellung als Maß für die Größe der Meßspannung benutzt. Der nach den beiden Verfahren gewonnene digitale Wert kann entweder direkt angezeigt (Digitalvoltmeter), durch einen Drucker ausgedruckt oder in einen Lochstreifen gestanzt werden.

Mit Hilfe einiger Zusatzeinrichtungen kann man Analogrechner so steuern, daß sie in äquidistanten Zeitintervallen stoppen, den Druckvorgang einleiten und nach beendetem Druck wieder weiterrechnen. Auch damit ist es möglich, die Ordinatenwerte der Lösungsfunktion numerisch zu erfassen.

Stromversorgung und Steuergerät werden bereits durch ihren Namen hinreichend charakterisiert.

4.3. Programmieren für Analogrechner

Sind die mathematischen Kenntnisse vorhanden, für Analogrechner benötigt man unbedingt die Differential- und Integralrechnung sowie die Kenntnis der Differentialgleichungen, so ist das Programmieren für Analogrechner einfacher auszuführen als vergleichsweise für Digitalrechner.

Mit Analogrechnern werden überwiegend gewöhnliche Differentialgleichungen gelöst. Partielle Differentialgleichungen können

mit üblichen Mitteln nur dann bearbeitet werden, wenn sie sich in ein System gewöhnlicher Differentialgleichungen überführen lassen. Der eigentliche Programmierungsprozeß umfaßt dann folgende Schritte:

das Auflösen der Differentialgleichung nach der n-ten (höchsten) Ableitung,

das Aufstellen der Strukturskizze,

das Anfertigen der Programmierskizze,

das Übertragen des Programms auf das Programmierfeld.

Das Auflösen der Differentialgleichung nach der höchsten Ableitung wird notwendig, weil im Analogrechner keine Differenzierelemente verwendet werden. Die Lösungsfunktion ergibt sich damit durch eine Reihe von Integrationen, wobei die Anzahl der benötigten Integratoren gleich der Ordnung der zu lösenden Differentialgleichung ist. Diese in Kette geschalteten Integratoren liefern an ihrem Ausgang die Lösungsfunktion, wenn die innerhalb der Kette entstehenden Ableitungen und die Lösungsfunktion selber nach der Ausgangsgleichung mit den entsprechenden Koeffizienten multipliziert und summiert wieder auf den Eingang der Kette gegeben werden.

In einer Strukturskizze wird der prinzipielle Aufbau der Rechenschaltung fixiert.

Daraufhin ist zu entscheiden, welche Arbeitsart angewendet werden und mit welcher Geschwindigkeit der Rechenvorgang ablaufen soll. Ist eine Lösungsfunktion zu ermitteln, die durch Veränderung von Koeffizienten erst gesucht werden muß, so empfiehlt es sich, den repetierenden Betrieb mit Rechenzeiten zwischen 10 und 100 ms zu wählen.

Bei genügend kleinen Rechenzeiten erhält man die Lösungsfunktion auf dem Bildschirm des Oszillografen als stehendes Bild und kann die Wirkung der Koeffizientenveränderung besonders gut beobachten.

Diese Rechenweise ist aber nicht sehr genau. Benötigen wir ein möglichst exaktes Ergebnis, ist eine Langzeitrechnung über mehrere Sekunden auszuführen.

Die dabei gewählte Rechenzeit kann ein Bruchteil oder ein Vielfaches des real verlaufenden Prozesses sein.

Dadurch muß ein Maßstab zwischen der Problemzeit und der Rechenzeit des Automaten eingeführt werden, um die Ergebnisse für den real verlaufenden Prozeß aus der Lösungsfunktion ermitteln zu können.

Aber nicht nur für die Zeit (sie entspricht der unabhängigen Variablen), sondern auch für die Amplitude (sie entspricht der abhängigen Variablen) muß eine Maßstabtransformation durchgeführt werden, da die Rechner eine feste Ansteuerungsgrenze haben. Bei Röhrenrechnern liegt sie im allgemeinen bei ± 100 V, und bei Transistorgeräten sind es ± 10 V.

Diese Maßstabtransformation erfordert vor dem Maschinenrechnen eine Abschätzung, um die beim vorliegenden Problem auftretenden Amplituden zu erkennen. Nach der dabei erhaltenen maximalen Amplitude wird dann so transformiert, daß sie den möglichen Ansteuerungsbereich voll ausnutzt. Für kleine Amplituden verringert sich nämlich auch die Genauigkeit, bei größerer Amplitude wird das Gerät übersteuert und verfälscht das Ergebnis.

Nach dem Zeitmaßstab müssen auch die Anfangsbedingungen transformiert werden.

Mit diesen beiden Maßstäben kann die quantitative Programmierung ausgeführt werden, die mit dem Anfertigen der Programmierskizze endet. Die Programmierskizze ist die quantitative Erweiterung der Strukturskizze. Sie enthält die Potentiometer-Einstellungen, die technischen Daten der Integratoren (Zeitkonstante) sowie die Art und die Daten eventuell einzusetzender Funktionswertgeber. Entsprechend der Programmierskizze werden die einzelnen Verbindungen auf dem Programmfeld gesteckt. Moderne Automaten verfügen über auswechselbare Programmfelder. Dadurch können alle Programmierungsarbeiten unabhängig vom Automaten ausgeführt werden und blockieren dort keine Rechenzeiten. Das Programmfeld braucht dann mit dem gesteckten Rahmen nur durch wenige einfache Handgriffe in den Rechner eingesetzt zu werden, und der Rechner ist betriebsbereit.

Das Programmieren wollen wir an einem einfachen Problem von der physikalischen Aufgabenstellung bis zur fertigen Programmierskizze studieren.

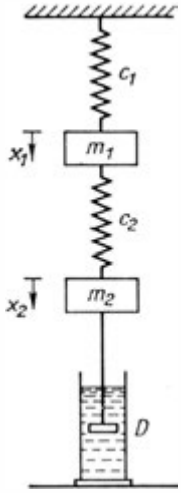


Bild 120. Gekoppeltes Masse-Feder-System mit Dämpfung

Es soll ein gekoppeltes Masse-Feder-System mit Dämpfung (Bild 120) auf sein mechanisches Verhalten untersucht werden. Die Differentialgleichungen für dieses System lauten:

$$\ddot{x}_1 m_1 + c_1 x_1 - c_2(x_2 - x_1) = 0$$

$$\ddot{x}_2 m_2 + D(\dot{x}_2 - \dot{x}_1) + c_2(x_2 - x_1) = 0$$

Die Auflösung beider Differentialgleichungen nach der höchsten Ableitung liefert

$$\ddot{x}_1 = \frac{c_2 x_2 - (c_1 + c_2) x_1}{m_1}$$

$$\ddot{x}_2 = \frac{D \dot{x}_1 - D \dot{x}_2 + c_2 x_1 - c_2 x_2}{m_2}$$

Werden mit

$$a_2 = \frac{c_1}{m_1} \qquad b_1 = b_2 = \frac{c_2}{m_2}$$

$$a_1 = \frac{c_1 + c_2}{m_1} \qquad b_3 = b_4 = \frac{D}{m_2}$$

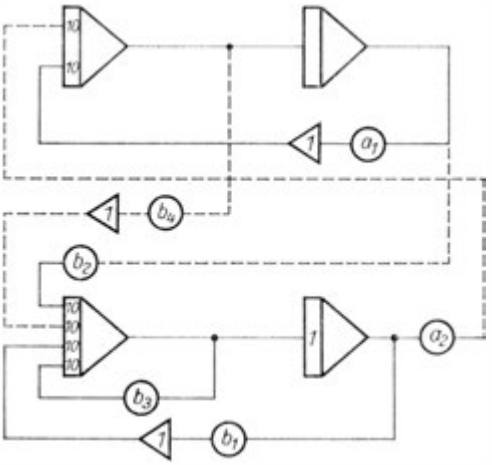
neue Koeffizienten eingeführt, ergibt sich

$$\dot{x}_1 = a_2 x_2 - a_1 x_1$$

$$\dot{x}_2 = b_4 \dot{x}_1 - b_3 \dot{x}_2 + b_2 x_1 - b_1 x_2$$

Nach diesem System können wir die Strukturskizze aufstellen. Dies geschieht erst einmal für jede der Gleichungen. Dabei werden die jeweils anderen Variablen vernachlässigt. In der

Bild 121
Strukturskizze des betrachteten Beispiels



ersten Gleichung erscheinen somit das Glied a_2x_2 und in der zweiten die Glieder $b_4\dot{x}_1$ und b_2x_1 nicht. Erst nach Aufstellen der Skizze für die Einzelgleichungen werden beide Systeme gekoppelt. (Dies entspricht den gestrichelten Linien im Bild 121.) Zur quantitativen Programmierung werden zuerst die Problemvariablen durch Maschinenvariable ersetzt. Soll τ die Maschinenzeit sein und M_t der Zeitmaßstab, so gilt:

$$\tau = M_t \cdot t$$

Entsprechend kann für die Amplitudentransformation gesetzt werden:

$$U_a = M_x \cdot x_a$$

wobei für die Dimension des Amplitudenmaßstabes gilt

$$[M_x] = \left[\frac{\text{Spannung}}{\text{Originaldimension}}\right]$$

Für spezielle Systeme entsprechend Bild 120 mögen sich die Gleichungen mit den Maschinenvariablen wie folgt ergeben:

Fall 1: $m_1 = 2\text{kg}\ m_2 = 2\text{kg}\ c_1 = 6\ \frac{\text{kp}}{\text{cm}}\ c_2 = 8\ \frac{\text{kp}}{\text{cm}}\ D = 16\ \frac{\text{kg}}{\text{s}}$

Fall 2: $m_1 = 2\text{kg}\ \ m_2 = 3\text{kg}\ c_1 = 6\ \frac{\text{kp}}{\text{cm}}\ c_2 = 8\ \frac{\text{kp}}{\text{cm}}\ D = 16\ \frac{\text{kg}}{\text{s}}$

Fall 3: $m_1 = 2\text{kg}\ \ m_2 = 4\text{kg}\ c_1 = 6\ \frac{\text{kp}}{\text{cm}}\ c_2 = 8\ \frac{\text{kp}}{\text{cm}}\ D = 16\ \frac{\text{kg}}{\text{s}}$

$$\dot{U}_{a1} = a_2\, U_{a2} - a_1\, U_{a1} \qquad \dot{U}_{a2} = b_4\, \dot{U}_{a1} - b_3\, \dot{U}_{a2} + b_2\, U_{a1} - b_1\, U_{a2}$$

1) $\dot{U}_{a1} = 3\ U_{a2} - 7\ U_{a1} \qquad \dot{U}_{a2} = 8\ \dot{U}_{a1} - 8\ \dot{U}_{a2} + 4\ U_{a1} - 4\ U_{a2}$

2) $\dot{U}_{a1} = 3\ U_{a2} - 7\ U_{a1} \qquad \dot{U}_{a2} = 5{,}33\ \dot{U}_{a1} - 5{,}33\ U_{a2} + 2{,}67\ U_{a1} - 2{,}67\ U_{a2}$

3) $\dot{U}_{a1} = 3\ U_{a2} - 7\ U_{a1} \qquad \dot{U}_{a2} = 4\ \dot{U}_{a1} - 4\ \dot{U}_{a2} + 2\ U_{a1} - 2\ U_{a2}$

wobei $M_t = 1$ sein soll.

Unbekannt sind nur noch die Zeitkonstanten T der Integratoren, die besonderen, nicht näher erläuterten Beziehungen genügen müssen und die Werte $\frac{1}{10}$ und 1 erhalten sollen. Die Anfangswerte für das Problem seien:

$$x_1(0) = 2;\ \dot{x}_1(0) = 0$$
$$x_2(0) = 0;\ \dot{x}_2(0) = 0$$

Mit dem Maßstab $M_x = 10$ wird dann $U_{a1}(0) = 0{,}2$.

Diese Werte liefert die in Bild 122 dargestellte Programmierskizze. Durch die mit $\frac{1}{\tau} = 10$ bzw. 1 bewerteten Eingänge der Integratoren sind alle Koeffizienten über die Potentiometer einstellbar, deren Arbeitsbereich bekanntlich nur zwischen 0 und 1 liegt.

Die nun aus der Programmierskizze erkennbaren Verbindungen werden auf dem Programmierfeld mit Kurzschlußsteckern oder kurzen Schnüren gesteckt. Die benutzten Potentiometer des Potentiometerfeldes können mit einer im Automaten installierten Brückenschaltung auf den geforderten Wert abgeglichen werden. Auch die Größe der Integrationszeitkonstanten ist auf dem Programmierfeld steckbar. Allerdings stehen für $\frac{1}{\tau}$ meist nur die Werte 0,1, 1 und 10 zur Verfügung.

Damit können nach Einsetzen der gesteckten Programmiertafel in den Rechner an dessen Steuerfeld die nötigen Operationen ausgelöst werden, um das Problem im repetierenden Betrieb oder einmalig zu lösen.

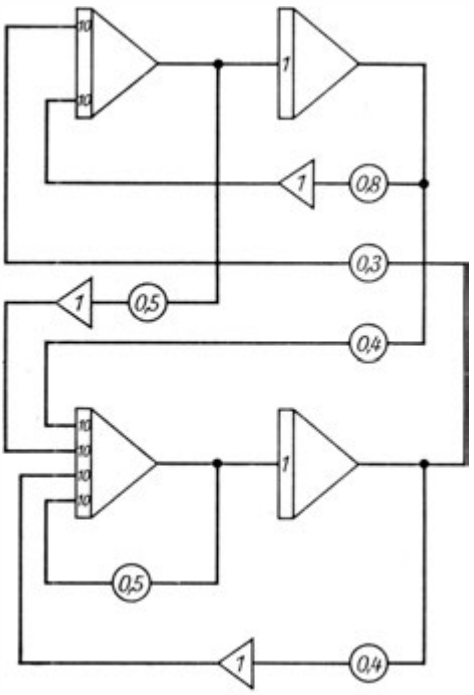


Bild 122. Programmierskizze

Die Lösungsfunktion wird dabei von allen angeschlossenen Ausgabegeräten angezeigt bzw. mitgeschrieben. Da die Meßbereiche der Geräte und die Maßstäbe M_x und M_t bekannt sind, kann das Ergebnis leicht von den Maschinenvariablen wieder auf die Problemvariable zurücktransformiert werden, was eine unmittelbare Auswertung ermöglicht.
Bild 123 zeigt die mit dem xy-Schreiber aufgezeichneten Ergebniskurven für die drei angenommenen Fälle des Beispiels.
Die Veränderung der Koeffizienten läßt auch zu, erst eine optimale Lösung zu ermitteln und dann diese Lösungsfunktion aufzuschreiben.

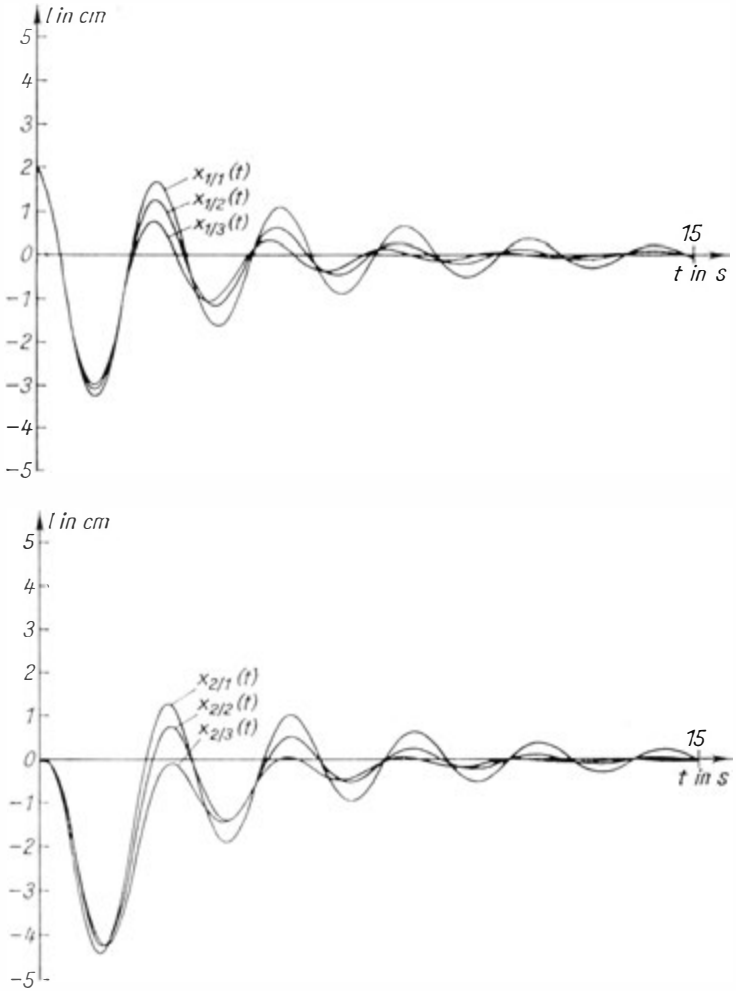


Bild 123. Ergebnisanzeige, auf dem xy-Schreiber ausgegeben. Dabei entsprechen

$x_1/_1\,(t)$ und $x_2/_1\,(t)$ dem **1.** Fall

$x_1/_2\,(t)$ und $x_2/_2\,(t)$ dem **2.** Fall $\quad x_1/_3\,(t)$ und $x_2/_3\,(t)$ dem **3.** Fall

Insbesondere kann auf diese Weise die Lösungsmannigfaltigkeit eingeschränkt werden, um in begrenzteren Bereichen durch Digitalrechner unter beträchtlicher Zeiteinsparung exakte Berechnungen mit jeder erforderlichen Genauigkeit ausführen zu können.

4.4. Simulatoren

Viele Großgeräte und technische Anlagen sind sehr schwer optimal zu dimensionieren, weil es bisher kaum Möglichkeiten gab, das Verhalten dieser Einrichtungen im praktischen Betrieb vor ihrer Herstellung zu untersuchen. Oftmals versagen hier Modelle, die nur das verkleinerte Abbild des Originals sind.
Für Analogrechner wurde dadurch ein vollkommen neues Einsatzgebiet erschlossen. Die benutzten Rechner werden Simulatoren bzw. Modellregelkreise genannt. Simulatoren arbeiten im allgemeinen im Originalzeitbereich, während bei Modellregelkreisen nur dann keine Zeittransformation durchgeführt wird, wenn innerhalb des Modellkreises einzelne Strecken aus Originalgeräten bestehen.
Wird die physikalische Analogie benutzt, um alle Bewegungen, Druckunterschiede oder sonstigen Veränderungen innerhalb einer Einrichtung möglichst naturgetreu, d. h. mit dem gleichen zeitlichen Verhalten und den gleichen Einflüssen aufeinander, durch Elemente der Analogrechner nachzubilden, so hat man das Verhalten dieses Gerätes simuliert.
Simulatoren erlangen immer größere Bedeutung. So werden z. B. die Piloten der großen Verkehrsflugzeuge zum überwiegenden Teil an Simulatoren ausgebildet. Der hohe Preis einer solchen Anlage ist gerechtfertigt, wenn man bedenkt, daß bei dieser Ausbildung weder Mensch noch Material gefährdet sind. Die Ausbildung selbst kann auch wesentlich intensiver erfolgen, denn es ist nicht schwer, dem Flugschüler einen Triebwerkschaden, eine Vereisung der Tragflächen oder Schwierigkeiten bei der Steuerung des Leitwerkes vorzutäuschen. Je nachdem, wie der Pilot in der imitierten Kanzel auf diese Schäden reagiert, wird ihm sein eigener Absturz oder aber die Stabilisierung des Flugzeuges gezeigt, so daß er die Möglichkeit hat, gefährliche

Situationen in der Luft vollkommen gefahrlos auf der Erde meistern zu lernen. Diese Flugsimulatoren haben auch noch den Vorteil, daß das Verhalten der Maschine genau studiert werden kann und Fehler und Mängel der Konstruktion leichter erkennbar sind.

In dem folgenden Beispiel soll gezeigt werden, wie man das Verhalten eines Kraftfahrzeuges simulieren kann. Dabei sollen einige Vereinfachungen vorgenommen werden, damit das Problem besser zu übersehen ist.

Wir nehmen an, das Fahrzeug soll auf einer ebenen Straße fahren und seine Geschwindigkeit nur im Bereich eines Ganges verändern – beim vierten Gang z. B. zwischen 50 und 90 km/h –, das Drehmoment des Motors sei in diesem Bereich nur von der Gaspedalstellung, nicht aber von der Drehzahl abhängig, und der Luftwiderstand steige mit dem Quadrat der Geschwindigkeit. Geht man von der Beschleunigung a aus, so erhält man nach der ersten Integration gemäß

$$v = \int a \, dt$$

die Geschwindigkeit v und nach der zweiten Integration gemäß

$$s = \int v \, dt$$

den zurückgelegten Weg s. Um aber erst einmal a zu erhalten, muß man die Summe aller am Fahrzeug wirkenden Kräfte bilden und dem Newtonschen Gesetz entsprechend durch die Fahrzeugmasse dividieren. Im Bild 124 ist die vereinfachte Schaltung eines Fahrzeugsimulators dargestellt.

Das Potentiometer a_1 ist dabei auf die maximale mögliche Motorkraft, das Potentiometer a_2 auf den Luftwiderstandsbeiwert und a_3 auf die maximal mögliche Bremskraft eingestellt. Mit dem Potentiometer a_4 wird die Fahrzeugmasse berücksichtigt, so daß $a = \frac{F}{m}$ gilt. Nach einmaliger Integration der Beschleunigung entsteht die Geschwindigkeit v, die über einen Funktionswertgeber mit quadratischer Kennlinie und das Potentiometer a_2 die Luftreibungskraft F_2 bestimmt. F_1 ist die wirksame Antriebskraft des Motors und F_3 die wirksame Bremskraft. Nach dem zweiten Integrierelement wird der zurückgelegte Weg s angezeigt.

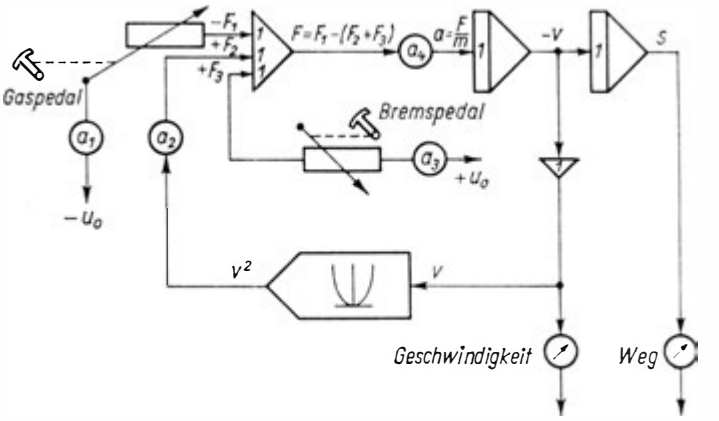


Bild 124. Vereinfachter Fahrzeugsimulator

An diesem Beispiel können wir unschwer das Prinzip der Simulation erkennen. Soll ein Simulator aber wirklich aussagefähig werden, muß seine Schaltung wesentlich mehr Einflüsse erfassen. Sie wird dadurch ungleich komplizierter.

Modellregelkreise werden vorwiegend dazu benutzt, um das Verhalten umfangreicher Regelkreise zu untersuchen und daraus optimale Regelschaltungen abzuleiten. Sie sind also spezielle Simulatoren für Untersuchungen an Regelungssystemen. Modellregelkreise enthalten neben den bereits bekannten Rechenelementen der Analogrechner noch einige spezielle Elemente, z. B. Verzögerer und Differentiatoren. Mit ihrer Hilfe ist es möglich, das regelungstechnische Verhalten der zu regelnden Anlagen noch vor ihrer Fertigstellung im Labor zu untersuchen und entsprechende Schaltungen zu entwickeln. Durch ihren Einsatz läßt sich die Entwicklungszeit für spezielle Regelungssysteme erheblich verkürzen und deren Verhalten einer optimalen Lösung näherbringen.

5. Hybrid-Rechenanlagen

5.1. Allgemeine Bemerkungen

Analogrechner wie Digitalrechner haben Vor- und Nachteile. Bei diesen Rechentypen liegen diese meist einander direkt gegenüber. Vorteile des Analogrechners sind meist Nachteile des Digitalrechners und umgekehrt. So liegt es nahe, einen Rechentyp zu entwickeln, der beider Vorteile vereint und beider Nachteile vermeidet. Dieser Gedanke wird seit vielen Jahren verfolgt. Erst in den letzten Jahren zeigten sich praktisch brauchbare Ergebnisse. Es entstanden die Hybridrechner. Zum Verständnis ihrer Arbeitsweise betrachten wir erst einmal Schwächen und Stärken der digitalen und analogen Rechentechnik.

5.2. Vergleichende Betrachtung der Analog- und Digitalrechentechnik

Stärken der Analogrechner zeigen sich überwiegend in folgenden Fakten:

schnelles Durchrechnen auch komplizierter gewöhnlicher Differentialgleichungen und Differentialgleichungssysteme;

Möglichkeit, über Potentiometerveränderungen auch größere Parameterfelder zu sondieren, um günstige Bereiche zu ermitteln.

Analogrechner verwenden Parallelorganisation, wodurch äußerst kurze Rechenzeiten vorherrschen. Dies ist vor allem für Prozeßrechner im Real-time-Betrieb wichtig.

Programmieren, Erproben der Programme und Programmveränderungen können beim Analogrechner weitaus schneller und mit geringerem Aufwand ausgeführt werden, als es bei der „software"-Entwicklung für Digitalrechner möglich ist.
Bei der Entwicklung und beim Einsatz von Regelungssystemen können Echtglieder oder Echtteile eines Systems vielfach leicht in den Analogrechner mit einbezogen werden. Bei Digitalrechnern müssen wir dazu erst Analog-Digital- und Digital-Analogwandler zwischenschalten.
Schließlich seien noch die weitaus geringeren Kosten für Analogrechner erwähnt, wenn eine Genauigkeit von etwa 1 % ausreicht.
Digitalrechner zeigen demgegenüber Stärken auf folgenden Gebieten:

die mit geringem Aufwand zu erreichende große Genauigkeit;

die Möglichkeit der permanenten Speicherung von Informationen;

die Möglichkeit, im Rechenablauf logische Entscheidungen einzubauen.

Der Digitalrechner hat einen weitaus größeren zulässigen Wertebereich für seine Daten. Das Problem der Skalierung ist praktisch bedeutungslos gegenüber dem Analogrechner mit seiner zulässigen Spannung von ± 100 V für Röhrengeräte und ± 10 V für transistorierte Anlagen.
Mit dem Digitalrechner können Tabellen unmittelbar gelesen und über Interpolationen zur Funktionsdarstellung verwendet werden. Dies bereitet beim Analogrechner bei Funktionen mehrerer Variabler erhebliche Schwierigkeiten.
Schließlich hat der Digitalrechner eine weitaus größere Zuverlässigkeit, was aus folgenden Überlegungen ersichtlich wird:

> Er arbeitet nur mit zwei möglichen Spannungswerten, sein Umfang bleibt unabhängig von der Aufgabenstellung, und die Betriebssicherheit ist über Kontrollprogramme leicht zu überprüfen.

Die hybride Rechentechnik sucht den Einsatz analoger und digitaler Verfahren zur Lösung desselben Problems.

5.3. Entwicklung und Arbeitsweise der Hybridrechner

Bei der Entwicklung der Hybridrechner lassen sich fünf Entwicklungsstufen erkennen:

5.3.1. Ergänzung des Analogrechners mit paralleler Logik

In der ersten Stufe werden einzelne Elemente der Digitaltechnik an den Analogrechner angebaut und über das Steckbrett programmierungstechnisch erfaßt. Es handelt sich dabei um folgende Zusätze:

Elemente der kombinatorischen Logik (UND-Gatter, ODER-Gatter, Negator),
Elemente der sequentiellen Logik (Flip-Flop und Register),
Elemente zur Steuerung eines zeitlichen Programmablaufs (Taktgeber, Zähleinrichtungen),
Elemente, die unmittelbar mit dem Analogrechner gekoppelt werden können (Vorrichtungen zur Steuerung der Betriebsart des Gesamtrechners sowie einzelner Elemente).

Damit lassen sich im sequentiellen Rechnen zwei Rechenarten ermöglichen, die große Vorteile bieten.
Im repetierenden Rechnen kann die Lösung mit hoher Geschwindigkeit analog ermittelt werden, wobei nach jedem Schritt digital ein Parameter verändert wird. Dies liefert über den Bildschirm eine schnelle Übersicht über die Schar der Lösungskurven.
Beim iterativen Rechnen verläuft der Prozeß wie soeben beschrieben, die Parameter werden durch einen Programmzusatz (analog und durch die Logikelemente) jedoch so verändert, daß ein gewünschter Lösungsverlauf möglichst gut angenähert wird. Damit lassen sich automatisch Optimierungen vornehmen und auch Randwertprobleme lösen.
Mit diesem System können aber auch diskontinuierliche Probleme gelöst werden.

5.3.2. Ergänzung des Analogrechners durch Logikelemente und digitale Arithmetik

Das bereits im Abschnitt 5.3.1. beschriebene System wird nun durch Halbadder, Adder und komplette Addierwerke einschließlich der Register ergänzt.
Anwendungsmöglichkeiten ergeben sich vor allem bei Problemen, bei denen infolge eines großen zu durchlaufenden dynamischen Bereichs eine analoge Darstellung nicht mehr ausreicht.
Bereits hier muß es zur analog-digitalen Datenkopplung kommen, wie sie Bild 125 zeigt. Allerdings können in diesem System lediglich Einzelwerte ausgetauscht werden.

5.3.3. Ergänzung durch einen digitalen Speicher

Jetzt können wir bereits von einem Hybridsystem sprechen, da nahezu alle Elemente des Digitalrechners an den Analogrechner angebaut sind. Allerdings ist in dieser Stufe der Digitalteil keineswegs voll ausgebaut und hat somit nicht die Flexibilität eines Digitalrechners.
Dennoch lassen sich nunmehr Probleme wie die Funktionserzeugung durch Tabellenlesen mit linearer Interpolation realisieren.

5.3.4. Kopplung von Analogrechner mit Digitalrechnern für Eingabe-, Ausgabe- und Steuerzwecke

Kopplungssysteme zur Verbindung von Analog- und Digitalrechner haben im wesentlichen Umsetzungs- und Steueraufgaben zu lösen. Der Datenaustausch erfordert das Umsetzen der Daten von analoger in digitale Form sowie die Auswahl aus einer Menge auf den parallelorganisierten Maschinenteil (Analogrechner) mit gleichzeitiger Einfügung dieser Information in den serienorganisierten Maschinenteil (Digitalrechner) und umgekehrt.
Die Steuerung und Koordination des Programmablaufs erfordert zudem

die Steuerung des erwähnten Umsetzungsprozesses, die Koordinierung des Systems zur Herstellung der Synchroni-

sation zwischen den unterschiedlich organisierten Anlagenteilen und den darin ablaufenden Programmen sowie die Steuerung der Betriebszustände des gesamten Systems.

Dies kann über gerätemäßige Vorrichtungen sowie spezielle Monitorprogramme erfolgen.

Die gerätemäßige Bestückung der analog-digitalen Datenkopplung zeigt Bild 125.

Ein Multiplexer schaltet aus einer Anzahl paralleler analoger Eingänge in zeitlicher Aufeinanderfolge jeweils einen aus und verwandelt die Parallelinformation so in eine serienorganisierte Information um.

Der Analog-Digitalumsetzer formt die am Eingang anstehende Gleichspannung in Digitalworte um und läßt die Werte in ein Ausgaberegister auflaufen, von dort kann sie in den Digitalrechner übernommen werden.

Umgekehrt formt ein Digital-Analogumsetzer die serienorganisierten Digitalinformationen in analoge Gleichspannungen um,

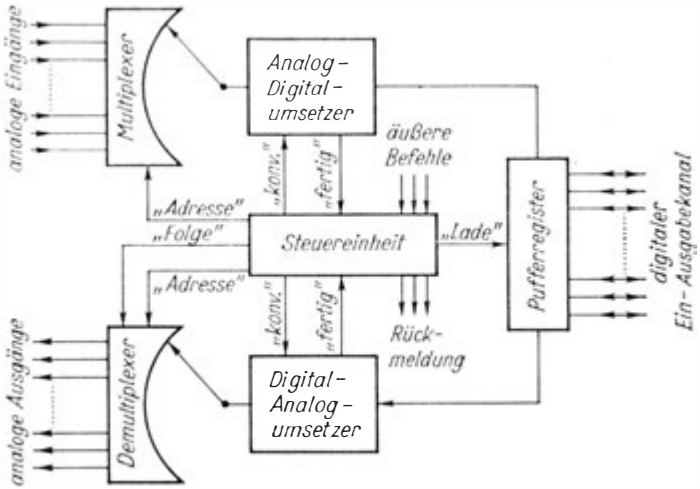


Bild 125. Analog-digitale Datenkopplung

die dann über den Demultiplexer wiederum parallelorganisiert werden. Sie können von dort in den Analogrechner einfließen. Der ganze Prozeß wird von einer Steuereinrichtung ausgelöst, gesteuert und kontrolliert.

Natürlich müssen Analogrechner, Steuereinheit des Übertragungssystems und Digitalrechner insgesamt durch ein Programm und entsprechende Funktionstastaturen gesteuert werden, wobei insbesondere das Synchronlaufen aller Teilsysteme gesichert werden muß. So verwendet man in dem beschriebenen System sowohl Parallelprogrammierung mittels Steckbrettern als auch Serienprogrammierung über eingespeicherte Programme wie bei Digitalrechnern und in Ausnahmefällen auch Steckbrettprogramme wie bei Lochkartenanlagen. Hier ist das Hauptproblem, die zeitliche Koordination zu sichern. Das erfolgt über ein Monitorprogramm und erfordert eine Arbeitsweise des Digitalrechners im time sharing.

Vorwiegend wird in dieser Ausbaustufe der Digitalrechner nur zur Steuerung des Analogrechners benötigt, trotz der hohen Forderungen, die wir an ihn stellen müssen (time sharing).

Von ihm werden dabei hauptsächlich folgende Operationen ausgeführt:

automatisches Einstellen und Verändern von Koeffizientenpotentiometern nach Vorgabe- oder errechneten Werten,

Übernahme von Werten aus sämtlichen anwählbaren Ausgängen des Analogrechners (oder einer Auswahl der Ausgänge), um sie zu drucken oder digital zwischenzuspeichern (Loch- oder Magnetband),

Steuerung der Betriebsarten des Analogrechners,

Zeitfestsetzung für das System und einzelner Elemente desselben.

Zusätzlich können umfangreiche Rechnungen mit äußeren und vom Analogrechner ermittelten Inputdaten vorgenommen werden.

5.3.5. Hybridsysteme

Hybridsysteme sind komplexe Rechner neuen Typs, die alle drei Teilsysteme – Analogrechner, Datenkopplung, Digitalrechner – in sich vereinen. Die drei Teilsysteme sind dabei aufeinander ab-

gestimmt. Insbesondere ist das Datenkopplungssystem nunmehr voll ausgebaut und mehrstufig gepuffert. Auch kann bei Hybridsystemen die Darstellung des mathematischen Modells über alle drei Teilsysteme erfolgen.

Mit ihnen erhalten wir die Möglichkeit eines komplexen Einsatzes vor allem für solche Probleme, die mit reinen Analog- oder Digitalrechnern nicht oder zumindest nur mit ökonomisch nicht vertretbarem Einsatz gelöst werden können. Es seien nur angedeutet:

System gewöhnlicher Differentialgleichungen mit sehr unterschiedlichen Parameterwerten,
Lösung partieller Differentialgleichungen,
komplizierte Simulationsprozesse,
Lösung gewöhnlicher Differentialgleichungen und deren Systeme mit Totzeitgliedern,
statistische Untersuchungen mit wiederholten Trendberechnungen,
Auswerten großer Mengen analoger und digitaler Daten mit Filterung, Integration und anderen Zusatzprozessen u. a.

6. Bemerkungen zur Anwendung programmgesteuerter Rechenautomaten

Nachdem wir uns einen Überblick über programmgesteuerte Rechenautomaten der verschiedensten Arten und Typen sowie deren Arbeitsweise verschafft haben, wollen wir uns einige Vorstellungen über ihre Anwendung erarbeiten.

Kehren wir dazu noch einmal zu dem auf Bild 92 skizzierten Ablaufschema eines Bearbeitungsprozesses zurück. Dort gingen wir davon aus, daß wir vom allgemeinen Sachverhalt bis zur Auswertung der Lösung völlig alleingestellt arbeiten müssen. Eine solche Situation gibt es bei uns jedoch nicht mehr. Die inzwischen gesammelten Erfahrungen und die Existenz mehrerer Rechenzentren erfordern eine andere Bearbeitungsweise für ein beliebig geartetes Problem. Hierbei kommt es zur Zusammenarbeit zwischen den Praktikern der verschiedensten Fachbereiche und den Mitarbeitern der Rechenzentren, wie sie auf Bild 126 angedeutet ist.

Die Mitarbeiter des Fachbereichs erarbeiten stets aus dem allgemeinen Sachverhalt die Problemstellung (s. Seite 182). Allerdings kommt es dabei meist jetzt schon zur konsultativen Mitarbeit durch das Rechenzentrum über dessen Informationsdienst. Die Problemstellung wird so darauf orientiert, nach Möglichkeit vorhandene Modelle und Programme zumindest dazu zu benutzen, um erste Vorstellungen zu ermitteln. Liegt die Problemstellung vor, empfiehlt es sich, eine Konsultation mit den Vertretern des Rechenzentrums durchzuführen, vor allem wenn nur wenig Erfahrungen in der Anwendung von Rechenautomaten vorliegen. Danach wird die mathematische Modellierung der Problemstellung vorgenommen. Hierbei kann es sich ergeben,

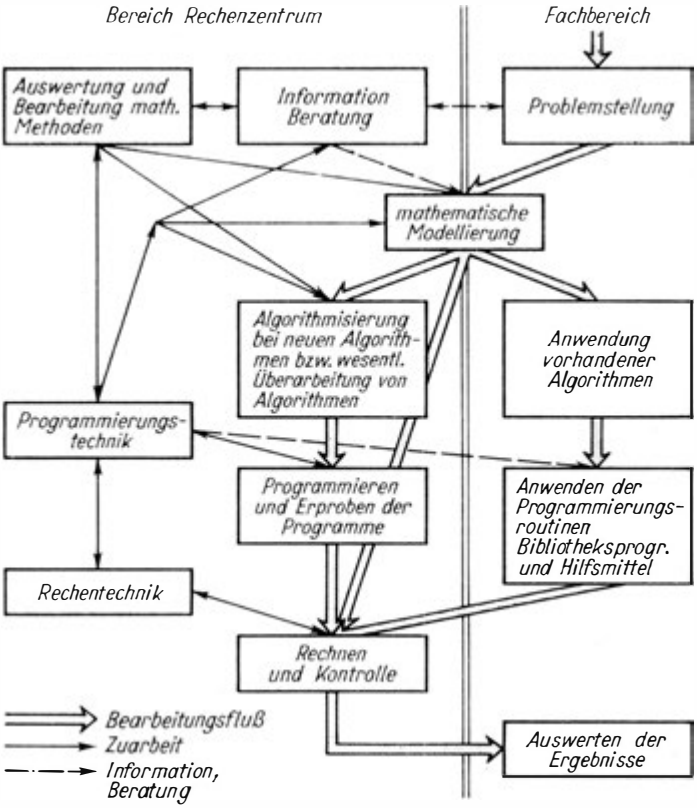


Bild 126. Schema der Bearbeitungsprozesse bei Zusammenarbeit der Fachbereiche mit einem Rechenzentrum

daß vorhandene mathematische Modelle zur Erfassung der Problemstellung voll ausreichen. Dann liegen meist auch bereits die Programme vor, und nach Ermittlung und Aufbereitung der Inputdaten kann die Rechnung unmittelbar beginnen. Dies wird für das Rechenzentrum ein reiner Dienstleistungsauftrag, da lediglich Rechenzeit zur Verfügung gestellt wird.

Ähnlich verhält es sich auch, wenn vorhandene mathematische Modelle mit geringfügigen Änderungen übernommen werden können. Bereiten die Änderungen mathematisch, in programmiertechnischer Hinsicht und in der algorithmischen Erfassung keine prinzipiellen Schwierigkeiten, werden sie vorwiegend durch die Mitarbeiter der Fachbereiche ausgeführt. Von ihnen werden die neuen Programmteile auch erprobt. Das Rechenzentrum tritt wiederum nur als reiner Dienstleistungsbereich auf. Auch wenn zwar keine mathematischen Modelle vorliegen, aber auch keine prinzipiellen Schwierigkeiten bei ihrer Aufstellung, bei der algorithmischen Darstellung und Programmierung auftreten, muß das Rechenzentrum nicht in Anspruch genommen werden.
Anders ist jedoch die Situation bei prinzipiellen Schwierigkeiten in der mathematischen Modellierung, in der algorithmischen Erfassung oder in der Programmierung. Dann soll die Bearbeitung zumindest gemeinsam von Mitarbeitern der Fachbereiche und des Rechenzentrums fortgesetzt werden.
Lediglich im Arbeitsgang „Rechnen" schneiden sich alle Zweige des Bearbeitungsprozesses. Die Auswertung der Ergebnisse wiederum sollte ausschließlich Aufgabe des Fachbereiches sein.
Durch das Rechenzentrum sind die Arbeiten vorwiegend in der Form zu unterstützen, daß dessen Mitarbeiter die erforderliche software zur Verfügung stellen.
Eine erste Form der direkten Hilfe bietet die Programmbibliothek. Alle für das Fachgebiet interessanten Programme werden dort erfaßt und für die Fachbereiche zur Verfügung gestellt.
Hierzu wird durch die Rechenzentren vielfältiges Informationsmaterial herausgegeben, bei dem drei Formen erkennbar sind. Das Programmverzeichnis ist die allgemeinste Form. Hier werden lediglich alle in die Programmbibliothek aufgenommenen Rechenprogramme und Organisationsprojekte erfaßt. Entsprechend den Wirtschaftszweigen werden sie in verschiedene Gruppen gegliedert. Vom Bauwesen werden beispielsweise folgende Gruppen angegeben:

Gruppe 0 Mathematische Programme
Gruppe 3 Bautechnische Programme
Gruppe 5 Bauökonomische Programme
Gruppe 9 Automatentechnische Programme

Über die fehlenden Ziffern ist noch nicht verfügt.
Derartige Programmverzeichnisse werden meist mehrmals im Jahr aktualisiert und sollten in die Hände jedes Interessenten gelangen. Programmkarteien bieten eine etwas weiter reichende Information. Sie enthalten Angaben über Daten, Leistungsfähigkeit und Anwendungsmöglichkeiten der Programme. Nach der Kartei können wir uns die umfassenderen Informationen bestellen.
Programminformationen geben erschöpfend Auskunft über eine „passive" Nutzung eines Programms. Wir wollen dann lediglich Aufgaben nach dem Programm, Programmsystem oder Organisationsprojekt rechnen lassen.
Sie enthalten daher Angaben über

Möglichkeiten und Grenzen des Programms,
Organisation der Inputdaten (gegebenenfalls mit Formblättern für die Eingabe),
Programmablauf und Leistungsfähigkeit,
Organisation der Outputdaten
mit einem übersehbaren, elementar einfachen Beispiel und Angaben über Rechenzeiten und Kosten.

Das sind alle Kenntnisse, die wir für eine Anwendung des Programms benötigen.
Programmdokumentationen geben uns die Möglichkeit der „aktiven" Anwendung eines Programms. Jetzt wollen wir also selber damit rechnen. Dazu enthalten sie

Angaben über Bezeichnung, Umfang, Daten und Hersteller des Programms,
eine Beschreibung des mathematischen Modells,
das Flußbild oder die Darstellung in einer algorithmischen Sprache,
die Befehlsliste (bei Programmen in der Maschinensprache),
Input- und Outputorganisation mit zulässigem Bereich bzw. Grenzwerten der Daten,
die Regieanweisung,
ein oder einige Testbeispiele, so daß eine komplette Prüfung des Programms möglich ist (Durchlauf aller Schleifen und Teste, Kontrollrechnung mit Extremwerten u. a.),

einen Kartensatz, der bereits mit den angegebenen Testbeispielen gelaufen ist.

Es empfiehlt sich also, mit dem nächstgelegenen oder zum Industriezweig gehörenden Rechenzentrum Verbindung aufzunehmen, um sich zumindest das Programmverzeichnis zu beschaffen. Zur Erleichterung haben wir eine Auswahl der Rechenzentren in der DDR, soweit sie nach unserer Kenntnis Auftragsrechnungen ausführen, auf Seite 294 angeführt.

Über die Programmbibliothek hinaus müssen durch die Mitarbeiter der Rechenzentren natürlich auch Programmiersysteme vom Autocode bis zur allgemeinsten problemorientierten Programmiersprache mit den erforderlichen Übersetzungsprogrammen (Compilern) bereitgestellt werden. Das bezieht sich natürlich vorwiegend auf die Programmierhilfen, die von den problemkomplexen und von den technischen Möglichkeiten her wünschenswert oder erforderlich sind.

Beispielsweise hätte es von den technischen Möglichkeiten aus gesehen wenig Sinn, einen COBOL-Compiler für den ZRA 1 zu entwickeln. Ohne die Möglichkeiten des ZRA 1 schmälern zu wollen, würde die begrenzte Speicherkapazität der praktischen Anwendung sehr schnell Grenzen setzen.

Andererseits wird man in einem Rechenzentrum – sagen wir des Bauwesens – kaum ein Programmiersystem für Sprachübersetzer vorrangig einführen, da der Bedarf aus den spezifischen Problemstellungen des Bauwesens zumindest relativ gering sein dürfte.

Über mathematische Modelle ist in den letzten Jahren umfangreiches Material erschienen. Eine einigermaßen umfassende Darlegung der gebräuchlichsten Methoden würde den Rahmen dieses Buches sprengen. Daher möchten wir nur eine Übersicht einfügen. Die fachtechnischen Anwendungen erfordern immer komplexere Modelle. Es wird nicht mehr als ausreichend angesehen, wenn durch ein Modell und mit dem dazugehörenden Programm bzw. Programmsystem die meist recht umfassende und komplizierte technische Berechnung ausgeführt wird. Zusätzlich verlangt man, daß durch das Modell und Programmsystem über Parameterveränderungen eine nach verschiedensten Forderungen optimale Lösung ermittelt wird. Gleichzeitig werden Zusatz-

forderungen gestellt, wie Ermittlung der Massen, Materialien, Kosten u. a.
Mit dem Steigen der technischen Möglichkeiten und der gesammelten Erfahrungen werden diese Forderungen an ein mathematisches Modell und dessen rechentechnische Realisierung weiter steigen, und die Ansprüche werden sich noch mehr erhöhen.
War es beispielsweise vor einigen Jahren noch ausreichend, wenn im Tiefbau zur Stützung der Wände von Baugruben die Rammtiefe der Stützpfähle berechnet wurde, so sind heute die Forderungen an das Programmsystem wie folgt gestiegen:
Aus Angaben über Bodenart, Tiefe der Baugruben und technologischen Informationen soll das Programmsystem aus einem vorgegebenen Sortiment von Pfählen die optimale Type, deren Rammtiefe und Abstand sowie die Kosten und Ausführungszeit berechnen. Diese Forderungen werden derzeit dahin ergänzt, daß auch die Outputdaten für die digitalgesteuerte automatische Zeichnung der Pläne zu ermitteln sind.
In der Ökonomie und Technologie werden vorwiegend zwei Modellklassen unterschieden, statische Modelle und dynamische Modelle. Die statischen Modelle, nicht im Sinne etwa der Baustatik, ermöglichen die Erfassung einer Problemstellung zu einem Zeitpunkt. Werden dabei Bedarf und Aufkommen eines Zeitraumes betrachtet, z. B. der Bedarf eines Jahres, so betrachtet ein statisches Modell diesen Zeitraum ebenfalls ohne zeitliche Veränderung. Benötigen wir exaktere Aussagen, müssen wir den Zeitraum untergliedern, beispielsweise die Jahresplanung durch eine Planung für ein Quartal oder einen Monat ersetzen.
Zu diesen Modellen können wir als bekannteste Methoden und Verfahren nennen:

Verflechtungsbilanzen,
lineare Optimierung,
nichtlineare Optimierung,
Transportoptimierung u. a.

Als Beispiel sei das Modell „Transportoptimierung" angeführt. Die „klassische" Aufgabe, mit der die Mathematik erste bedeutende Erfolge in der Ökonomie errungen hat, ist die Trans-

portoptimierung. Diese Methode läßt sich deswegen so leicht anwenden, weil die erforderlichen ökonomisch-technischen Kennzahlen leicht durch Ausmessen auf einer Karte ermittelt werden können. Bei Anwendung dieser Methode wird davon ausgegangen, daß ein Massenprodukt von mehreren Produzenten in gleicher Qualität erzeugt wird. Es soll an mehrere Verbraucher so transportiert werden, daß der gesamte Transportaufwand zum Minimum wird.

Ein einfaches Modellbeispiel soll uns die Methode veranschaulichen.

Zwei Erzeuger mögen folgende Mengen produzieren: $A_1 = 130$ Einheiten, $A_2 = 70$ Einheiten. Demgegenüber steht der Bedarf von drei Verbrauchern: $V_1 = 100$ Einh., $V_2 = 40$ Einh. und $V_3 = 60$ Einh. Die Entfernungen zwischen den einzelnen Erzeugern und Verbrauchern sind in der folgenden Aufstellung eingetragen, in der auch die erzeugten und benötigten Mengen vermerkt wurden:

	V_1	V_2	V_3	
A_1	7	2	3	130
A_2	4	1	8	70
	100	40	60	200

Meist erfolgt vor der Optimierung eine Belieferung „auf Zuruf". Das soll heißen, daß ein auftretender Bedarf von dem Erzeuger gedeckt wird, der gerade liefern kann.

Damit möge sich folgende Belieferung ergeben haben, wie sie in der nächsten Aufstellung angeführt ist:

	V_1	V_2	V_3	
A_1	50 / 7	30 / 2	5 / 3	130
A_2	50 / 4	10 / 1	10 / 8	70
	100	40	60	200

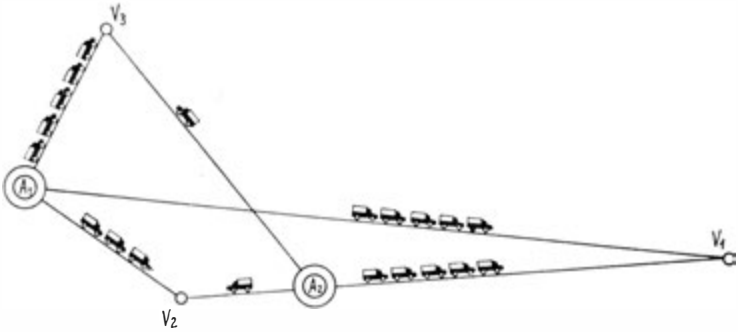


Bild 127. Vor der Transportoptimierung

Diese Verteilung entspricht in etwa der tatsächlich meist vorliegenden Situation. Für die Belieferung ergibt sich ein Aufwand von:

$$50 \cdot 7 + 30 \cdot 2 + 50 \cdot 3 + 50 \cdot 4 + 10 \cdot 1 + 10 \cdot 8 =$$
$$= 350 + 60 + 150 + 200 + 10 + 80 = 850$$
(Einheiten · km),

dem entspricht ein Durchschnittstransport für jede Einheit von 4,25 km.

Durch Anwendung der Transportoptimierung kann eine optimale Belieferung ermittelt werden:

	V_1	V_2	V_3	
A_1	30 / 7	40 / 2	60 / 3	130
A_2	70 / 4	0 / 1	0 / 8	70
	100	40	60	200

Das Gesamstransportaufkommen ist jetzt:

$$30 \cdot 7 + 40 \cdot 2 + 60 \cdot 3 + 70 \cdot 4 + 0 \cdot 1 + 0 \cdot 8 =$$
$$= 210 + 80 + 180 + 280 + 0 + 0 = 750$$
(Einheiten · km),

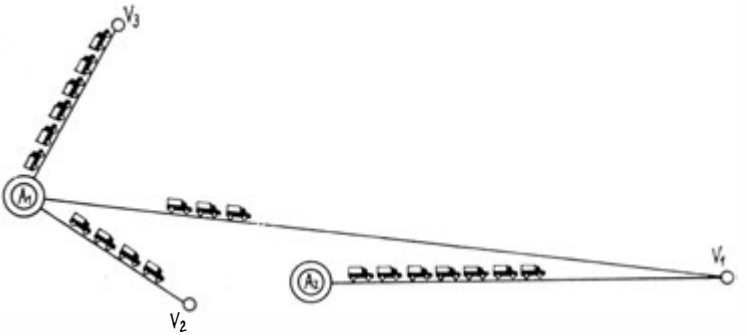


Bild 128. Nach der Transportoptimierung

dem nun ein Durchschnittstransport von 3,75 km für jede Einheit entspricht. Das sind nahezu 12 % Einsparungen. So hoch liegen auch die realen Einsparungen (das Beispiel wurde entsprechend aufgestellt). Internationale Erfahrungen haben ergeben, daß die Einsparungen durch Transportoptimierung um 10 % liegen. Das Gesamtaufkommen der DDR an Gütertransportleistungen betrug 1962 5270 Mio t · km Lastkraftwagen-, 34733 Mio t · km Bahntransporte.

Nehmen wir nur an, daß die Hälfte der transportierten Güter Massengüter sind, die nach dem angeführten Modell der Transportoptimierung erfaßt werden können, und setzen wir noch als Durchschnittswerte 0,26 M · t · km für Lastkraftwagentransporte und 0,064 M · t · km für Bahntransporte, so ergibt sich für die zu optimierende Menge

	Hälfte des Gesamtaufk. Mio t km	Hälfte des Gesamtaufk. Mio M	10 % Einsparung durch Transportoptimierung Mio M
LKW	2635	685	68,5
Bahn	17366	1111	111,1
Mögliche Gesamteinsparung:			179,6 Mio M

Das sind keineswegs utopische Zahlen. Transportoptimierungen, die 1963 im Bauwesen durchgeführt wurden, ergaben folgende Einsparungsmöglichkeiten:

Mauerziegel (Bezirk Leipzig)[1]	17,5 %	220 TM/Jahr
Kies (Bezirk Dresden)[2]	24 %	750 TM/Jahr
Kleinkopfpflaster (Bezirk Dresden LKW)	14,4 %	40 TM/Jahr
Kleinkopfpflaster (DDR Bahntransport)	14,1 %	60 TM/Jahr

Diese Werte sind auch nahezu voll realisierbar, denn von den möglichen 55000 M eines Quartals konnten im Bezirk Leipzig beim Transport von Mauerziegeln 50000 M tatsächlich eingespart werden. Die bei der Transportoptimierung angewandte mathematische Methode läßt sich aber noch für viele andere ökonomische Probleme verwenden, beispielsweise für die Ermittlung einer optimalen Zuordnung.

Dynamische Modelle ermöglichen es vor allem, den zeitlichen Ablauf in die Berechnung einzubeziehen. Mit ihrer Hilfe können wir Vorhaben modellieren, planen und deren Ausführung kontrollieren, wie sie im Verlaufe der Zeit realisiert werden.

Besonders diese Modelle haben in den letzten Jahren an Bedeutung gewonnen. Zu ihnen zählen vor allem die Methoden der Netzplantechnik, wie

Methode des kritischen Weges,
PERT,
RAMPS u. a.

Daneben gibt es für spezielle Probleme das dynamische Programmieren.

Es sei auch ein Beispiel aus der Netzplantechnik angeführt. Es soll ein vereinfachtes Bauvorhaben betrachtet werden. Die Methode beginnt mit der Aufstellung einer Liste aller Einzelabschnitte eines zu planenden Vorhabens. Diese Einzelabschnitte werden Aktivitäten genannt.

---

[1] Gleichlautende Informationen liegen aus den Bezirken Dresden, Erfurt, Gera, Cottbus, Rostock, Berlin u. a. vor

[2] Gleiche Untersuchungen werden in allen Bezirken der DDR durchgeführt

Für unser Beispiel sollen folgende Aktivitäten auftreten:

Herstellen der Fundamente,
Montage der Wandelemente,
Montage der Dachkonstruktion,
Einbringen der Ausrüstung,
Einsetzen der Fenster,
Innenausbau,
Anschließen der Ausrüstung.

Nunmehr gilt es, die Zusammenhänge zwischen den Aktivitäten zu ermitteln. Jede Aktivität erhält dabei zwei Zahlen als Bezeichnung, eine für den Beginn, die andere für das Ende. Beginn und Ende einer Aktivität werden Ereignis genannt.
Beginnen werden wir mit dem Ereignis 0. Da für alle folgenden Aktivitäten die Fundamente vorhanden sein müssen, erhält die Aktivität „Herstellen der Fundamente" die Bezeichnung 0 — 1. Anschließend sollen gleichzeitig die Wandelemente montiert und die Ausrüstung eingebracht werden. Diese Aktivitäten erhalten die Bezeichnung 1 — 2 und 1 — 3. Nach dem Aufstellen der Wandelemente soll wiederum gleichzeitig die Dachkonstruktion ausgeführt und der Innenausbau vorgenommen werden. Damit ergibt sich für die Montage der Dachkonstruktion 2 — 4 und für den Innenausbau 2 — 5. Es folgt das Einsetzen der Fenster, das ebenfalls zum Ereignis 5 abgeschlossen sein soll. Hieraus ergibt sich die Bezeichnung 4 — 5. Nach dem Einbringen der Ausrüstung folgt deren Montage, die als Bezeichnung 3 — 5 erhält. Mit diesen Festlegungen ergibt sich eine geordnete Liste der Aktivitäten

Herstellen der Fundamente	0—1	5
Montage der Wandelemente	1—2	4
Einbringen der Ausrüstung	1—3	5
Montage der Dachkonstruktion	2—4	7
Innenausbau	2—5	6
Anschließen der Ausrüstung	3—5	4
Einsetzen der Fenster	4—5	6

nach der ein Netzplandiagramm konstruiert werden kann (Bild 129). Darauf sind die Zeiten für alle Aktivitäten zu ermitteln. Wir haben sie der Einfachheit halber in die geordnete Liste der

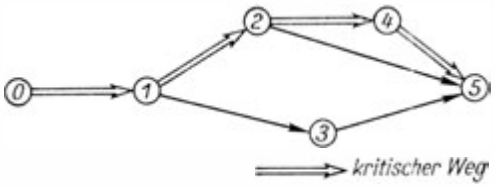


Bild 129. Netzplandiagramm

Aktivitäten eingefügt. Damit können wir alle Wege, die vom Anfangsereignis 0 bis zum Endereignis 5 führen, errechnen und deren Zeitdauer feststellen.
Für unser einfaches Beispiel sind dies die Wege

(1) 0—1—2—4—5 = 22 (Monate)
(2) 0—1—2—5 = 15 (Monate)
(3) 0—1—3—5 = 14 (Monate)

Wie ersichtlich, hat der Weg (1) den größten Zeitwert. Dies ist der kritische Weg, und die darauf liegenden Aktivitäten werden als „kritische Aktivitäten" bezeichnet. Sie bestimmen den Endtermin für den Abschluß des Planungsvorhabens. Alle Verzögerungen oder Verkürzungen in den kritischen Aktivitäten wirken sich auf den Endtermin aus. Da bei umfassenderen Problemen, wie sie in der Praxis nur vorkommen, denn ein so einfaches Modell gibt es praktisch nicht, nur 10 % bis höchstens 20 % aller Aktivitäten kritisch sind, können sich die verantwortlichen Leitkader durch Anwendung dieser Methode auf die wirklich wichtigen Abschnitte ihres Vorhabens konzentrieren und ihre Kräfte sachlich richtig einsetzen.
Nach dieser Methode lassen sich über einige weitere Bearbeitungsstufen Terminkalender aufstellen, volkswirtschaftlich optimale Ausführungszeiten errechnen, gleichförmige Auslastung der Arbeitskräfte oder Maschinen und anderes ermitteln.
Ein Vereinigen statischer und dynamischer Modelle zu einem Gesamtmodell ist zur Zeit noch nicht befriedigend gelöst, obwohl in vielen Forschungseinrichtungen intensiv daran gearbeitet wird.
Vielfach unterscheidet man auch zwischen determinierten und

statistischen Modellen, je nachdem, ob die Koeffizienten, Parameter und Variablen, determinierte oder statistische Größen sind. Statistische Modelle, die also mit Zufallsvariablen und deren Varianz arbeiten, spiegeln meist die objektive Realität besser wider als Modelle über determinierte Größen, denn bei der überwiegenden Mehrzahl der Problemstellungen wirken Gesetzmäßigkeiten nach statistischen Regeln ein. Determinierte Modelle sind dagegen mathematisch und rechentechnisch leichter zu bewältigen. Zudem lassen sich viele Sachverhalte über die Problemstellung mit ausreichender Genauigkeit als determiniertes Modell erfassen. Daher wird den determinierten Modellen meist der Vorrang gegeben.

Die zur Zeit höchste Form der Anwendung mathematischer Methoden für die Planungs- und Leitungstätigkeit ist die Durchführung planstrategischer Spiele. Bei diesen werden reale Wirtschafts- oder Produktionssituationen über ein mathematisches Modell simuliert. Die im Modell enthaltenen Parameter haben meist einen endlichen Variationsbereich, der durch Spielregeln fixiert wird. Zudem wird der Ablauf des Spiels als Wechsel formaler Berechnungsprozesse mit manuellen Entscheidungen der Spieler als persönliche Züge festgelegt.

Betrachten wir dies erst an einem einfachen Beispiel.

Gegeben sei folgende Produktionssituation: Von einem Erzeugnis können zwei gleichwertige Typen in einem Betrieb produziert werden, x_1 Stück vom Typ 1 und x_2 Stück vom Typ 2. Die Belastung des Maschinenparkes erfolge aber durch die Typen unterschiedlich, wobei nur eine Maximalkapazität für jeden Maschinensatz vorhanden sein soll.

Beispielsweise benötigt man beim Maschinensatz 1 mit einer Maximalkapazität von 120 Einheiten für den Typ 1 die doppelte Zeit gegenüber dem Typ 2. Der Maschinensatz 2 mit einer Maximalkapazität von 70 Einheiten werde durch beide Typen gleichermaßen belastet, während beim Maschinensatz 3 mit einer Maximalkapazität von 150 Einheiten Typ 2 eine dreifache Belastung verursacht.

Die staatlichen Organe wollen durch geschickte Preispolitik erwirken, daß eine maximale Anzahl des Erzeugnisses hergestellt und die Maschinensätze 1 und 2 maximal genutzt werden. Dabei

ist es aber volkswirtschaftlich nicht zu vertreten, daß der Betrieb über 1000 Einheiten Gewinn erzielt.
Diese Produktionssituation führt auf das Modell der linearen Optimierung, das von dem sowjetischen Mathematiker *L. W. Kantorowitsch* entwickelt wurde.
Der Betrieb steht in folgender Situation:
Er strebt einen maximalen Gewinn über die Zielfunktion

$$Z = P_1 x_1 + P_2 x_2 \Rightarrow \max$$

an und unterliegt dabei folgenden Einschränkungen
Maschinensatz 1: $2x_1 + x_2 \leqq 120$
Maschinensatz 2: $x_1 + x_2 \leqq 70$
Maschinensatz 3: $x_1 + 3x_2 \leqq 150$
Zudem muß gelten:

$$0 \leqq x_1 \text{ und } 0 \leqq x_2$$

Die Leiter des staatlichen Organs haben die persönlichen Züge, durch „willkürliche" Veränderung der Werte P_1 und P_2 die angestrebten Ziele zu erreichen.
Praktisch unterliegen P_1 und P_2 also bestimmten Spielregeln für manuelle Züge, während die Berechnung der Werte x_1 und x_2 über einen Rechenautomaten erfolgen kann.
Für unsere Situation ergibt sich aus max $\leqq 1000$ die Folgerung, daß gelten muß $P_1 \leqq 15$ und $P_2 \leqq 20$.
Wir wollen die Wertebereiche

$$P_1 = \{5, 10, 15\} \text{ und}$$
$$P_2 = \{5, 10, 15, 20\}$$

zulassen. Das würde 12 Varianten ergeben, deren Ergebniswerte den staatlichen Leitern nicht bekannt sind.
Der Spielverlauf erfolgt nun so, daß P_1 und P_2 nach Erfahrung und Kenntnis vorgegeben werden,

z. B. $P_1 = P_2 = 5$,
das liefert $x_1 = 50$ $x_2 = 20$ $Z = 350$

also einen zu niedrigen Gewinn für den Betrieb. Maschinensatz 1 und 2 sind voll ausgelastet, Maschinensatz 3 hat eine freie Kapazität von 40 Einheiten.

Jetzt folgt ein neuer persönlicher Zug, beispielsweise

$$P_1 = 5, \quad P_2 = 10$$

durch den sich ergibt

$$x_1 = 30,\ x_2 = 40,\ x_1 + x_2 = 70,\ Z = 550$$
$$\text{M1} = 20,\ \text{M2} = \text{M3} = 0.$$

Zwar hat sich der Wert von Z erhöht, aber der Maschinensatz M1 erhält 20 Einheiten freie Kapazität. Daher wird versucht, durch Erhöhen des Anteiles P_1 dies auszugleichen.
Der nächste persönliche Zug sei also

$$P_1 = 10, \quad P_2 = 10$$

das liefert

$$x_1 = 30,\ x_2 = 40,\ x_1 + x_2 = 70,\ Z = 700$$
$$\text{M1} = \text{M2} = 0,\ \text{M3} = 40$$

wobei jedoch mehrere Möglichkeiten bestehen. Wählt das Leitorgan schließlich als nächsten persönlichen Zug

$$P_1 = 15, \quad P_2 = 10$$

ergibt sich

$$x_1 = 50,\ x_2 = 20,\ x_1 + x_2 = 70,\ Z = 950$$
$$\text{M1} = \text{M2} = 0,\ \text{M3} = 40.$$

Dies ist die optimale Lösung, denn würde man P_2 weiter erhöhen, also wählen

$$P_1 = 15, \quad P_2 = 15$$

würde dies liefern

$$x_1 = 30,\ x_2 = 40,\ x_1 + x_2 = 70,\ Z = 1050$$

das ist eine nach den Spielregeln unzulässige Lösung.
Da hier nur wenige Varianten möglich sind, können wir alle durchprobieren, um unser Urteil bezüglich der optimalen Lösung zu erhärten.

Nr	P_1	P_2	x_1	x_2	x_1+x_2	Z	
1	0	5	0	50	50	250	
2	0	10	0	50	50	500	
3	0	15	0	50	50	750	
4	0	20	0	50	50	1000	
5	5	0	60	0	60	300	
6	5	5	30—50	40—20	70	350	mehrere Lö-
7	5	10	30	40	70	550	sungen
8	5	15	0	50	50	750	
9	5	20	0	50	50	1000	
10	10	0	60	0	60	6000	
11	10	5	60	0	60	600	
12	10	10	30—50	40—20	70	700	mehrere Lö-
13	10	15	30	40	70	900	sungen
14	10	20	—	—	—	—	unzulässig,
15	15	0	60	0	60	900	$Z = 1000$
16	15	5	60	0	60	900	
17	15	10	50	20	70	950	
18	15	15	—	—	—	—	unzulässig, $Z = 1000$
19	15	20	—	—	—	—	unzulässig, $Z = 1000$

Es ist offensichtlich, daß die Variante 17 die optimale Lösung liefert.

In planstrategischen Spielen können wir die formale Anwendung mathematischer Modelle mit der Persönlichkeit des Leitenden, mit seinem Intellekt, seiner Erfahrung und seiner Übersicht über Gesetzmäßigkeiten, die im Modell nicht erfaßt werden können (z. B. politische Situationen), vereinen.

In Weiterführung läßt sich im Sinne der mathematischen Spieltheorie eine optimale Strategie ermitteln, die den Leiter in seiner Urteilsfindung unterstützt. Vor allem können planstrategische Spiele zur Schulung und Qualifizierung von Leitkadern verwendet werden. Praktische planstrategische Spiele haben natürlich einen wesentlich größeren Vorrat an Varianten, die nicht systematisch durchprobiert werden können, wie in unserem

Modellfall. Dort kommt den persönlichen Zügen eine viel größere Bedeutung zu.

Für die weiteren Überlegungen bezüglich der Anwendung von Rechenautomaten müssen wir auf die angeführte Literatur verweisen, in der auch Angaben über die Anwendung von Rechenautomaten in solchen Bereichen angegeben sind, die über das reine Rechnen hinausgehen. Erwähnt sei hier lediglich das automatische Übersetzen von Sprachen (maschinelle Linguistik) und das Erkennen von Krankheiten (maschinelle Diagnostik).

Große Bedeutung hat der Einsatz programmgesteuerter Rechenautomaten als Datenverarbeitungsanlagen für Aufgaben der üblichen Lochkartentechnik (Abrechnungswesen, Haushalt, Lagerhaltung u. a.). Hier ist die exakte Erfassung des Informationsflusses sehr wichtig. Dies gilt insbesondere, wenn durch die Rechentechnik oder Datenverarbeitung auch die gesamte Organisation beeinflußt wird.

Der Bearbeiter muß dann vorher ein einwandfreies, wohldurchdachtes Bild des Informationsflusses entwerfen. Es muß zeigen, woher die Inputdaten kommen. Dazu sollte er nach Möglichkeit Primärdaten benutzen, die also im Original anfallen und im Bearbeitungsbereich zur Eingabe nicht erst noch umgewandelt werden müssen (mit Ausnahme des Lochens). Er muß geeignete Kontrollen in den Informationsfluß einbauen und schließlich eine zweckentsprechende Aufbereitung und Verteilung der Outputdaten organisieren.

Nehmen wir ein einfaches Beispiel (Bild 130).

Es soll die Betriebsabrechnung teilautomatisiert werden. Den Input erhalten wir:

1. von der Kaderabteilung, und zwar die Information der Stammkarten aller Mitarbeiter mit Personalnummer, Bruttogehalt, Arbeitszeit u. a.;

2. von der Abteilung Planung ein Kostenträgerverzeichnis, in dem alle Arbeitsgebiete des Bereiches durch einen Kostenträger nach dem Prinzip der Dezimalklassifikation verschlüsselt werden, ferner natürlich die aktuellen Planvorgaben;

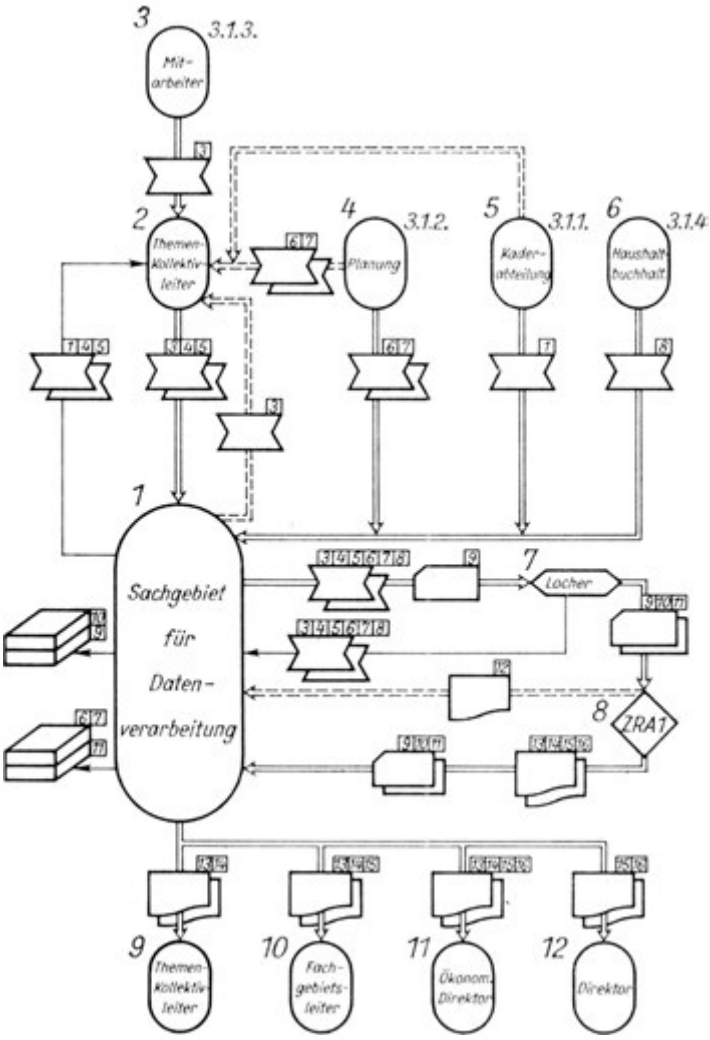


Bild 130. Bild des Informationsflusses für das Beispiel „Betriebsabrechnung“

3. vom Mitarbeiter einen Leistungsbogen mit der Angabe, wieviel Stunden seiner Arbeitszeit er an den für ihn zuständigen Kostenträgerpositionen gearbeitet hat;
4. vom Haushalt die zum Kostenträger direkt zurechenbaren tatsächlichen Kosten nach Kostenträgerverzeichnis.

Diese Inputdaten werden vom jeweiligen Abteilungsleiter kontrolliert und an ein Sachgebiet Datenverarbeitung weitergeleitet. Hier erfolgt eine zweite Kontrolle mit Rückkopplung zum Abteilungsleiter bei Fehlern.
Das Sachgebiet Datenverarbeitung führt und aktualisiert auch die Stammkarteien (Lochkarten), Personalkartei, Planvorgabekartei. Es veranlaßt das Lochen der aktuellen Angaben des Berichtszeitraumes, wie Leistungsbögen und Sachkontenkarten. Danach sichert es das Rechnen und Ermitteln der Outputdaten, z. B. der Zwischensummenbildung nach Abteilungen, Hauptabteilungen und Betrieb sowie Konten, Kontengruppen, Sachgebieten, der Berechnung des prozentuellen Erfüllungsstandes, der kumulativen Aufrechnung mit prozentuellem Gesamterfüllungsstand und anderes mehr.
Schließlich leitet das Sachgebiet Datenverarbeitung die zuständigen Outputdaten an die Empfänger weiter, beispielsweise dem Abteilungsleiter die Abteilungsliste, dem ökonomischen Direktor die Gesamtdarstellung des Soll-Ist-Vergleiches, dem Direktor Schwerpunktkennwerte, die einen Überblick über die Gesamtsituation des Betriebes geben, und diejenigen Kostenträger, bei denen mehr als z. B. 10 % Abweichung von der Planvorgabe vorhanden sind usw.
Als nächstes gilt es nunmehr die erforderlichen Formulare zu entwickeln und den Betrieb auf die neue Arbeitsweise umzustellen.
Besonders vorteilhaft ist es, bei der Aufbereitung der Primärdaten gleichzeitig die Inputdaten für den Automaten zu gewinnen. Das kann beispielsweise über eine Fakturiermaschine mit Lochstreifeneinrichtung erfolgen, wie sie Bild 131 zeigt.
Der Kleinmechanisierung und Bürotechnik ist also größte Aufmerksamkeit zu schenken.
Damit wird es möglich, zur komplexen Datenverarbeitung überzugehen, indem immer umfassendere Komplexe automatisiert

Bild 131. Elektronischer Fakturierautomat mit Lochstreifeneinrichtung Soemtron 381/41

werden, d. h. ohne direkte Einwirkung des Menschen in den Berechnungsprozeß verlaufen.
Das Endziel ist der Aufbau eines integrierten Systems der Datenverarbeitung.
Bei der integrierten Datenverarbeitung kennt man 3 Systemtypen:

die Integration mehrerer miteinander nicht verknüpfter Einzelsysteme (batch-Verarbeitung),
den Aufbau eines eng verflochtenen Komplexsystems (sequentielle Integration) und
die Systemparallelverarbeitung (random System).

Die Einzelmethode wird dadurch charakterisiert, daß eine Vielzahl voneinander unabhängiger Systeme für die Bearbeitung einzelner Probleme durch Automaten erledigt wird.
Die Vorteile des Systems liegen vor allem darin, daß die Einzelsysteme schnell und ohne Rückwirkung aufeinander korrigiert und den veränderten Bedingungen der Praxis angepaßt werden können. Nachteilig ist jedoch, daß diese Methode sehr zeitaufwendig ist und daß insbesondere die Inputdaten mehrfach erfaßt werden müssen.
Das Komplexsystem ermöglicht mit geringen Inputdaten über eine komplexe Verflechtung der einzelnen Bearbeitungsprozesse eine schnelle Bewältigung umfangreicher Aufgabenstellungen. Es hat jedoch den wesentlichen Nachteil, daß sein Aufbau äußerst kompliziert ist und lange Zeit in Anspruch nimmt und daß Änderungen nur schwer und wiederum nur mit großem Zeitaufwand möglich sind.
Die zur Zeit günstigste Methode ist die Systemparallelverarbeitung. Hier werden Einzelsysteme für sich bearbeitet, aber sinnvoll in einen Komplex eingebaut. Diese Methode umfaßt die Vorteile der beiden erwähnten Methoden unter weitgehender Vermeidung der dort genannten Schwierigkeiten. Sie ermöglicht es vor allem, aufbauend auf bestimmten Grundmodellen, eine schrittweise Erweiterung entsprechend den zunehmenden Erfahrungen und der Entwicklung der maschinen- und datentechnischen Basis vorzunehmen.
Die Einführung der Rechentechnik in wissenschaftlich-technische Bereiche und in die Datenverarbeitung hat einschneidende Rückwirkung auf die Arbeit jedes einzelnen und bringt Veränderungen der gesamten Organisation, der Bürotechnik und Leitungstätigkeit mit sich. Jeder muß sich mit den Möglichkeiten und Grenzen des neuen Arbeitsinstruments „programmgesteuerter Rechenautomat" auseinandersetzen. Das bereitet einige Schwierigkeiten und ist für manchen erst unbequem. Diese Umstellung ist jedoch Voraussetzung und Zeichen der persönlichen aktiven Anteilnahme an der Bewältigung der technisch-wissenschaftlichen Revolution.

Auswahl von Rechenzentren der DDR

Deutsche Akademie der Wissenschaften
Institut für Angewandte Mathematik und Mechanik
– Rechenzentrum –
108 Berlin, Mohrenstr. 39, Tel. 200561

Deutsche Akademie der Wissenschaften
Institut für Strukturforschung
Abt. Elektronisches Rechnen
1199 Berlin-Adlershof, Rudower Chaussee, Tel. 642041

Deutsche Akademie der Wissenschaften
Sternwarte Babelsberg
– Rechenzentrum –
1502 Potsdam-Babelsberg, Tel. 78870

Deutsche Akademie der Landwirtschaftswissenschaften zu Berlin
108 Berlin, Krausenstr. 38—39

Deutsche Bauakademie
Institut für Technik und Organisation
Zentrales Rechenzentrum für das Bauwesen
104 Berlin, Hannoversche Straße 30

Deutsches Brennstoffinstitut
– Rechenzentrum –
92 Freiberg/Sa., Bernh.-v.-Cotta-Str. 1

Hochschule für Architektur und Bauwesen Weimar
Institut für Mathematik
53 Weimar, Karl-Marx-Platz 2, Tel. 31 71

Hochschule für Ökonomie
Institut für ökonomische Datenverarbeitung
1157 Berlin-Karlshorst, Hermann-Duncker-Str. 8, Tel. 509171

Humboldt-Universität zu Berlin
II. Mathematisches Institut
– Rechenzentrum –
108 Berlin, Unter den Linden 6, Tel. 58 76 85

Institut für Nachrichtentechnik
der VVB Nachrichten- und Meßtechnik
116 Berlin-Oberschöneweide, Edisonstr. 63, Tel. 63 28 51

Institut Prüffeld für elektrische Hochleistungstechnik
– Rechenzentrum –
113 Berlin-Lichtenberg, Landsberger Chaussee 38a, Tel. 57 60 36

Institut für Schiffbau
25 Rostock, Osthafen, Tel. 63 41

Institut für Verkehrsforschung
108 Berlin, Krausenstr. 17—20, Tel. 58 08 51

Karl-Marx-Universität
Institut für Maschinelle Rechentechnik
701 Leipzig, Liebigstr. 27

Martin-Luther-Universität Halle-Wittenberg
Institut für Numerische Mathematik
401 Halle, Reichardtstr. 9, Tel. 246 44

Technische Hochschule Ilmenau
Institut für Maschinelle Rechentechnik
63 Ilmenau/Thür., Am Ehrenberg

Technische Hochschule – Institut für Mathematik –
90 Karl-Marx-Stadt, Straße der Nationen 62

Technische Hochschule „Otto v. Guericke" Magdeburg
II. Mathematisches Institut
30 Magdeburg, Boleslav-Bierut-Platz 5, Postfach 124

Technische Universität Dresden
Institut für Maschinelle Rechentechnik
8027 Dresden, Zellescher Weg 12—14

Universität Rostock
– Rechenzentrum –
25 Rostock, Universitätsplatz

VEB Atomkraftwerk
Betriebsteil Berlin
110 Berlin-Pankow, Görschstr. 45—46, Tel. 48 02 71

VEB Bergmann-Borsig
1106 Berlin-Wilhelmsruh, Tel. 480821
VEB Carl Zeiss Jena
– Rechenzentrum –
69 Jena, Carl-Zeiss-Str. 1, Tel. 7042
VEB Gasturbinenbau und Energiemaschinenentwicklung
– Rechenzentrum –
83 Pirna, Sonnenstein, Postfach 64, Tel. 791
Zentralinstitut für Automatisierung
– Rechenzentrum –
80 Dresden, Königsbrücker Landstraße, Tel. 593

Literaturverzeichnis

Reihe Automatisierungstechnik. Berlin: VEB Verlag Technik

Bd. 5 Schubert: Digitale Kleinrechner, 3. Aufl. 1966

Bd. 6 Sydow: Elektronische Analogrechner, 2. Aufl. 1966

Bd. 12 Stuchlik: Programmgesteuerte Universalrechner, 2. Aufl. 1966

Bd. 19 Ahner/Bode: Elektronische Datenverarbeitung in der Ökonomie, 1964

Bd. 25 Bär: Einführung in die Schaltalgebra, 2. Aufl. 1966

Bd. 42 Bär: Elektronische Datenverarbeitung – Grundstufe der COBOL-Programmierung, 1966

Bd. 43 Bär: Elektronische Datenverarbeitung – Oberstufe der COBOL-Programmierung, 1966

Bd. 44 Bär: Elektronische Datenverarbeitung – Praxis der COBOL-Programmierung, 1966

Bd. 47 Andersen: ALGOL 60 – eine Sprache für Rechenautomaten, 1967

Bd. 52 Paulin: Kleines Lexikon der Rechentechnik und Datenverarbeitung, 1967

Autorenkollektiv: Mathematik und Wirtschaft, Bde. I bis IV. Berlin: Verlag Die Wirtschaft 1963 bis 1966

Kerner/Zielke: Einführung in die algorithmische Sprache ALGOL. Leipzig: B. G. Teubner Verlagsgesellschaft 1966

Smers: Das maschinelle Lochkartenverfahren. Leipzig: VEB Fachbuchverlag 1969

Sachwort- und Namenverzeichnis

Wir danken allen Betrieben und Instituten, die uns Fotos zur Veröffentlichung überließen.
Bild 6 wurde der Maya-Handschrift der Sächsischen Landesbibliothek Dresden entnommen. Die Handschrift erschien 1962 im Akademie-Verlag Berlin.

B:16	B:8	B:2	+	−	B:2	B:8	B:16
		1	2, 5		— 1		
		2	4, 25		— 2		
	1	3	8, 125		— 3	— 1	
1		4	16, 062 5		— 4		— 1
		5	32, 031 25		— 5		
	2	6	64, 015 625		— 6	— 2	
		7	128, 007 812 5		— 7		
2		8	256, 003 906 25		— 8		— 2
	3	9	512, 001 953 125		— 9	— 3	
		10	1 024, 000 976 562 5		—10		
		11	2 048, 000 488 281 25		—11		
3	4	12	4 096, 000 244 140 625		—12	— 4	— 3
		13	8 192, 000 122 070 312 5		—13		
		14	16 384, 000 061 035 156 25		—14		
	5	15	32 768, 000 030 517 578 125		—15	— 5	
4		16	65 536, 000 015 258 789 062 5		—16		— 4
		17	131 072, 000 007 629 394 531 25		—17		
	6	18	262 144, 000 003 814 697 265 625		—18	— 6	
		19	524 288, 000 001 907 348 632 812 5		—19		
5		20	1 048 576, 000 000 953 674 316 406 25		—20		— 5
	7	21	2 097 152, 000 000 476 837 158 203 125		—21	— 7	
		22	4 194 304, 000 000 238 418 579 101 562 5		—22		
		23	8 388 608, 000 000 119 209 289 550 781 25		—23		
6	8	24	16 777 216, 000 000 059 604 644 775 390 625		—24	— 8	— 6
		25	33 554 432, 000 000 029 802 322 387 695 313		—25		
		26	67 108 864, 000 000 014 901 161 193 847 656		—26		
	9	27	134 217 728, 000 000 007 450 580 596 923 828		—27	— 9	
7		28	268 435 456, 000 000 003 725 290 298 461 914		—28		— 7
		29	536 870 912, 000 000 001 862 645 149 230 957		—29		
	10	30	1 073 751 824, 000 000 000 931 322 574 615 479		—30	—10	
		31	2 147 483 648, 000 000 000 465 661 287 307 739		—31		
8		32	4 294 967 296, 000 000 000 232 830 643 653 870		—32		— 8
	11	33	8 589 934 592, 000 000 000 116 415 321 826 935		—33	—11	
		34	17 179 869 184, 000 000 000 058 207 660 913 467		—34		
		35	34 359 738 368, 000 000 000 029 103 830 456 734		—35		
9	12	36	68 719 476 736, 000 000 000 014 551 915 228 367		—36	—12	— 9
		37	137 438 953 472, 000 000 000 007 275 957 614 183		—37		
		38	274 877 906 944, 000 000 000 003 637 978 807 092		—38		
	13	39	549 755 813 888, 000 000 000 001 818 989 403 546		—39	—13	
10		40	1 099 511 627 776, 000 000 000 000 909 494 701 773		—40		—10

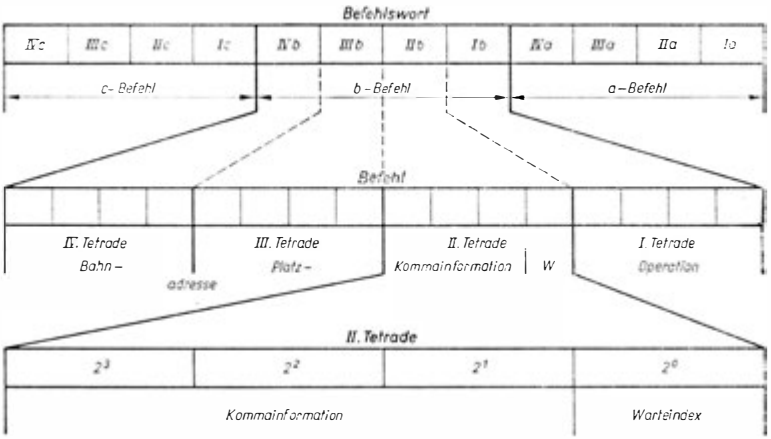
Umwandeln vierstelliger Oktalzahlen in Dezimalzahlen

	.0 .	1 . .	.2 . .	.3 . .	.4 . .	.5 . .	.6 . .	.7 .
0 . . .	0	64	128	192	256	320	384	448
1 . . .	512	576	640	704	768	832	896	960
2 . . .	1024	1088	1152	1216	1280	1344	1408	1472
3 . . .	1536	1600	1664	1728	1792	1856	1920	1984
4 . . .	2048	2112	2176	2240	2304	2368	2432	2496
5 . . .	2560	2624	2688	2752	2816	2880	2944	3008
6 . . .	3072	3136	3200	3264	3328	3392	3456	3520
7 . . .	3584	3648	3712	3776	3840	3904	3968	4032

Befehlsliste des Cellatron SER 2

Operation	Programmierungszeichen	Code dezimal	Code dual	Komma[1]) ohne mit Wortindex	Ablauf m. Adresse Null	Ablauf m. Adresse $a \neq 0$
Addition	+	1	OOOL	$2i$ $2i+1$	⟨AC⟩ + ⟨R⟩ ⇒ ⟨AC⟩	⟨FS/a⟩ → ⟨AC⟩ ⟨AC⟩ + ⟨R⟩ ⇒ ⟨AC⟩ L ⟨AC⟩ → ⟨FS/a⟩
Subtraktion	—	2	OOLO	$2i$ $2i+1$	⟨AC⟩ − ⟨R⟩ ⇒ ⟨AC⟩	⟨FS/a⟩ → ⟨AC⟩ ⟨AC⟩ − ⟨R⟩ ⇒ ⟨AC⟩ L ⟨AC⟩ → ⟨FS/a⟩
Multiplikation	×	3	OOLL	$2i$ $2i+1$	⟨AC⟩ × ⟨R⟩ ⇒ ⟨AC⟩	⟨FS/a⟩ → ⟨AC⟩ ⟨AC⟩ × ⟨R⟩ ⇒ ⟨AC⟩ L ⟨AC⟩ → ⟨FS/a⟩
Division	:	4	OLOO	0 1	⟨AC⟩ : ⟨R⟩ ⇒ ⟨AC⟩[2])	⟨FS/a⟩ → ⟨AC⟩ ⟨AC⟩ : ⟨R⟩ ⇒ ⟨AC⟩ L ⟨AC⟩ → ⟨FS/a⟩
Eingabe	E	5	OLOL	0 1	⟨FT⟩ → ⟨R⟩ Handeingabe	⟨FS/a⟩ → ⟨R⟩
Lochband Eingabe	LB	P4	LLOO	4 5 8 9	⟨SLI⟩ → ⟨R⟩ ⟨SLII⟩ → ⟨R⟩	
Ausgabe	A	6	OLLO	$2i$ $2i+1$	L ⟨AC⟩ → Druck	L ⟨AC⟩ → ⟨FS/a⟩
Ausgabe Lochband	AL	9	LOOL	0 1	⟨AC⟩ → ⟨LsI⟩	
unbedingter Sprung	SU	7	OLLL	0 1	(immer ausführen) Sprung nach Band	(immer ausführen) Sprung nach > BS/a <
Bedingter Sprung	S—	7	OLLL	2 3	(wenn ⟨AC⟩ < 0) Sprung nach Band	(wenn ⟨AC⟩ < 0) Sprung nach > BS/a <
Leertaste	L	P5	LLOL	0 1	Leertaste	
Tabulator	T	P6	LLLO	0 1	Tabulatorsprung	
Wagenrücklauf	W	P7	LLLL	0 1	Wagenrücklauf mit Zeilenschaltung	

[1]) ($i = 0, 1, 2, \ldots, 7$)
[2]) Nur ganzzahlig ohne Runden



POLYTECHNISCHE BIBLIOTHEK

Die Reihe ist durch den Buchhandel zu beziehen

unterrichtet allgemeinverständlich über den Weltstand in allen Zweigen der Technik

Bethke
Verbessern – erfinden
Grundlagen des Erfindungs- und Vorschlagswesens

Conrad
Elektrotechnik – kurz und einprägsam
Wissenswertes vom elektrischen Strom

Conrad/Bolhu
Elektronenröhren
Aufbau – Technologie – Verwendung

Glaser/Kohl
Mikroelektronik
Schaltkreise, Eigenschaften und Technologien

Herrmann
Neue Kraftfahrzeugkunde
Unterlagen für Fahrschule und Fahrpraxis

Reinke
Lüftung – Klimatisierung – Entstaubung
Grundlagen, Verfahren und Geräte

Smers
Das maschinelle Lochkartenverfahren
Allgemeinverständliche Einführung in einen Zweig der Datenverarbeitung

VEB FACHBUCHVERLAG LEIPZIG

Die elektronische Datenverarbeitung greift in die Sphäre jedes einzelnen ein, weil heutzutage die Arbeit mittel- oder unmittelbar durch sie beeinflußt wird. Jeder moderne Mensch möchte sich deshalb ein bestimmtes Wissen über moderne Rechentechnik aneignen – die Notwendigkeit wird mit der 5. Auflage dieses Titels bewiesen.

Der Autor, bekannt von Funk und Fernsehen, vermittelt den Lesern – Ingenieuren, Technikern, Wirtschaftsfunktionären, Fachlehrern, Studenten, Fach- und Oberschülern, Facharbeitern für Datenverarbeitung – ausreichende Vorstellungen über Möglichkeiten und Grenzen der Entwicklung und des Einsatzes programmgesteuerter Rechenautomaten bzw. Datenverarbeitungsanlagen.

Götzke • Programmgesteuerte Rechenautomaten

Programmgesteuerte Rechenautomaten

H. GÖTZKE