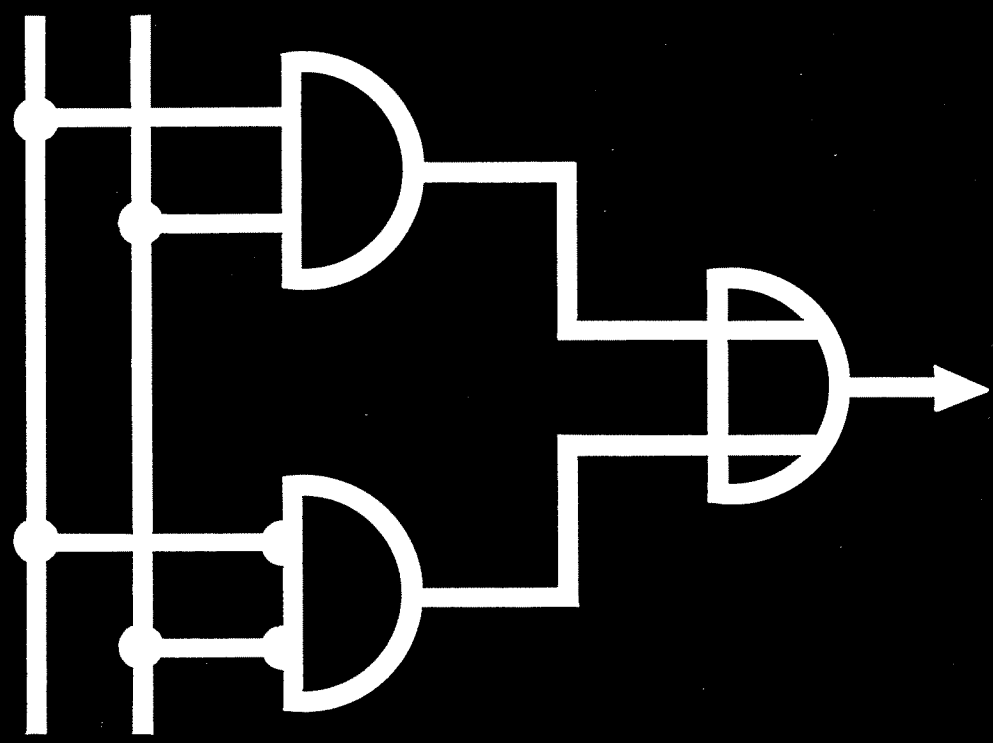
Handbuch der Elektronik



Teil 3
Datenverarbeitung; Technik und Betrieb

Handbuch der Elektronik

Herausgegeben mit Unterstützung des
Bundesministers für das Post- und Fernmeldewesen

Teil 3

Datenverarbeitung; Technik und Betrieb

Herausgeber: Institut zur Entwicklung moderner Unterrichtsmedien e.V.
28 Bremen 1 - Bahnhofstraße 10

Vorwort

Der Elektroniker, der sich mit der Technik der EDV-Anlagen, also mit ihrer Hardware, befassen muß, kommt nicht daran vorbei, sich auch mit der Organisation, dem Aufbau, dem Zusammenspiel der einzelnen Baugruppen, mit den Funktionsabläufen, dem Programmieren usw., kurz mit allem, was mit den Begriffen Software und Orgware ausgedrückt wird, eingehend zu beschäftigen. In diesem dritten Teil des „Handbuchs der Elektronik" ist deshalb der Versuch unternommen worden, neben den Hardware-Beschreibungen auch soviel Software-Kenntnisse zu vermitteln, wie nötig sind, um das Ineinandergreifen einer EDV-Anlage verstehen zu können.

In diesen Band wurde weiter ein Abschnitt aufgenommen, der die theoretischen Grundlagen digitaler Rechenanlagen aus der mathematischen Sicht her beleuchtet. Hier wird der Leser über die Mengenlehre zur Mengenlogik und von da zur Aussagenlogik geführt. Er kann hieraus ohne Mühe erkennen, daß sich die daran anknüpfende Schaltalgebra zwangsläufig aus den vorgenannten Disziplinen entwickelt und daß die Boolesche Algebra, die im Teil 2 des „Handbuchs der Elektronik" (Digitaltechnik) rein auf Verknüpfungsschaltungen bezogen behandelt wurde, ein Teilgebiet der modernen Mathematik darstellt, die heute als Mengenlehre bereits in der Grundschule Einzug gehalten hat.

Wir hoffen, mit dem vorliegenden Teil 3 des „Handbuchs der Elektronik" einen vorläufigen Abschluß des Lehrsystems über das Grundwissen auf dem Gebiet der Elektronik gefunden zu haben. Vorläufig deshalb, weil die Weiterentwicklung elektronischer Geräte und Systeme die Herausgabe eines weiteren Bandes bereits in naher Zukunft notwendig machen könnte. Bis dahin steht dem Leser mit den drei Teilen des vorgenannten Handbuchs und den Repetitorien für die Teile 1 und 2 ein umfassendes Lehrwerk zur Verfügung, das gleichermaßen zum Selbststudium als auch als Nachschlagewerk für die Praxis geeignet ist.

Die Herausgeber

Stand: Herbst 1972

Inhaltsverzeichnis

1. Arbeitsweise und Organisation von Datenverarbeitungsanlagen

1.1. Historische Entwicklung

Die Entwicklung von Zähl- und Rechenmaschinen geht einmal auf die Idee zurück, den Menschen von immer wiederkehrenden Aufgaben zu entlasten und zum anderen, die Fehlerquellen auszuschließen, die hierbei durch menschliche Unzulänglichkeit immer wieder auftreten. Den Gedanken, mechanisch zu rechnen, haben als erste bereits 1623 Wilhelm Schickard und 1642 Blaise Pascal verwirklicht. **W. Schickard** konstruierte eine Maschine für 6stellige Addition und Subtraktion, während die Maschine von **B. Pascal** für die 8stellige Addition und Subtraktion geeignet war. Bereits 1677 konstruierte dann **Gottfried Wilhelm Freiherr von Leibnitz** eine 4-Spezies-Maschine, die die vier Grundrechnungsarten Addieren, Subtrahieren, Multiplizieren und Dividieren ausführen konnte.

Charles Babbage (1792 — 1871) entwickelte rd. 100 Jahre später eine „Differenzmaschine", die das fehlerfreie Berechnen und Drucken von mathematischen Tabellenwerten ermöglichen sollte. 1834 versuchte Babbage, angeregt durch die Technik des Jacquard-Webstuhles (Steuerung durch Serien aneinandergereihter Lochkarten), eine „Analytische Maschine" zu konstruieren. Hierbei handelte es sich um einen Rechenautomaten, der auf rein mechanischer Basis arbeitete. Mit dieser Idee wurden zum erstenmal die Prinzipien von universellen Rechnern festgestellt. Diese Maschine sollte

a) durch ein vorher erarbeitetes Programm gesteuert werden,

b) einen Speicher für Zwischenergebnisse besitzen,

c) die Rechnungen schrittweise unter Verwendung der Zwischenergebnisse durchführen und

d) die Daten in maschinennaher Form ausgeben.

Die für die Realisierung dieser Idee notwendige Mechanik wäre jedoch so umfangreich und kompliziert geworden (z.B. 50 000 Ziffernräder auf 1000 Achsen), daß sie nach dem Stand der damaligen Technik noch nicht verwirklicht werden konnte. Die von Babbage hierfür entwickelten und an sich richtigen Erkenntnisse und Prinzipien mußten deshalb bis etwa 1930 ungenutzt bleiben.

Andere Entwicklungstendenzen führten zur Konstruktion von peripheren Geräten (Schreibmaschinen und deren Kombination mit Lochstreifen) sowie zur Einführung der Lochkarte und der Konstruktion der hierzu benötigten speziellen Geräte. **Falcon** (1728) und **Jacquard** (1801) benutzten die Lochkarte zur Steuerung eines Prozesses (zur Steuerung von Webstühlen). **Hermann Hollerith** hat erstmals 1887 die Lochkarten als Datenträger eingeführt; diese Karten besitzen noch heute das von ihm seinerzeit benutzte Format eines Ein-Dollar-Scheins. Zum Abtasten der Lochkarten verwendete Hollerith metallische Fühlstifte, die bei Erkennen eines Loches einen elektrischen Kontakt herstellen. Mit Hilfe dieses neuen sehr rationellen Verfahrens gelang es Hollerith z.B., die Ergebnisse der 11. amerikanischen Volkszählung im Jahre 1890 in nur einem Sechstel der bis dahin benötigten Auswertezeit vorzulegen. Die Lochkarteneinrichtungen wurden in den nächsten Jahren fortlaufend verbessert. Die Entwicklung von elektro-mechanischen Lochkartenstanzern und Tabelliermaschinen führte 1936 zum Bau der ersten schreibenden Tabelliermaschine und 1939 zur Konstruktion einer Tabelliermaschine, die über eine Stecktafel programmiert werden konnte.

Etwa 100 Jahre nach Babbage wurde die Entwicklung programmgesteuerter Rechner auf verschiedenen Ebenen weiter vorangetrieben. Fast zur gleichen Zeit arbeiteten u.a. Couffignal in Paris, Konrad Zuse in Berlin und Howard Aiken in New York an diesem Problem. Alle in dieser Zeit entwickelten Rechner waren in der Grundkonzeption gleich und waren wie folgt konstruiert:

Dateneingabe,
Datenspeicher,
Programmeingabe,
Steuerwerk,
Datenausgabe und
Rechenwerk.

Bei all diesen Maschinen handelte es sich um Relaisrechner.

Konrad Zuse begann 1935 in Deutschland mit der Konstruktion von programmgesteuerten Rechenanlagen; hierbei handelte es sich zunächst um eine mechanische (Z1) und dann um eine elektromechanische Ausführung (Relaisrechner). Als Programmspeicher diente hierbei jeweils ein Lochstreifen. Im Auftrag der deutschen Versuchsanstalt für Luftfahrt entstand 1941 die „Z3", eine Maschine, die als der erste voll funktionsfähige programmgesteuerte Rechner der Welt anzusehen ist; sie arbeitete bereits nach dem Prinzip der dualen Zahlendarstellung.

1945/46 bauten **Eckert, Mauchly und Goldstine** den ersten Elektronenrechner, der als ENIAC (elektronic numerical integrator and calculator) bezeichnet wurde und mit 18 000 Röhren bestückt war. Etwa zur gleichen Zeit (1946/47) erhielt die Rechnerentwicklung durch den Mathematiker **John von Neumann** neue Impulse. Dadurch, daß sich ein Programm während seines Laufes selbst modifizieren kann, wurde die Einführung des bedingten Sprungbefehls möglich. Daten und Programm wurden somit intern abgespeichert; der Schritt zur Speicherprogrammierung war vollzogen.

Zwischen 1956 und 1958 wurde der Transistor als Schaltelement in den Serienbau von Rechnern eingeführt. Der erste Rechner, der in Europa volltransistorisiert in Serie gebaut wurde, ist die Type „Siemens 2002". Im Jahre 1964/65 wurden integrierte Schaltungen als neue Schaltelemente eingesetzt. Damit konnten Leistungsfähigkeit und Rechengeschwindigkeit gesteigert und die räumlichen Abmessungen verkleinert werden.

Der Entwicklung der „Software" kommt bei der Einführung der Rechner große Bedeutung zu. Mit dem Wachsen der Arbeitsgeschwindigkeiten und der Speichergrößen wird die Programmierarbeit umfangreicher, weil die Anwendungsmöglichkeiten steigen. Zur Erleichterung und Standardisierung sind daher allgemein gültige Programmsprachen entwickelt sowie Vereinbarungen und Normierungen getroffen worden.

Die Rechenanlagen werden häufig nach Art der verwendeten Bauteile und Baugruppen in sogenannte „Generationen" eingeteilt, die durch bestimmte Merkmale gekennzeichnet sind; dies geschieht wie folgt:

1. Generation: Verwendung von Elektronenröhren. Einsatz ab ca. 1946. Programmierung erfolgte ausschließlich im Maschinencode. Rechenleistung ca. 1000 Additionen pro Sekunde.

2. Generation: Gekennzeichnet durch gedruckte Schaltungen mit Halbleiterbauelementen (Transistoren). Einsatz ab ca. 1956. Neben maschinenorientierter auch problemorientierte Programmierung möglich. Rechenleistung ca. 10 000 Additionen pro Sekunde.

3. Generation: Charakterisiert durch die Mikroschalttechnik (integrierte Schaltkreise). Einsatz ab ca. 1965. Programmierung hauptsächlich über problemorientierte Sprachen. Einführung der Betriebssysteme zur optimalen Rechnerauslastung. Rechenleistung ca. 50 000 Additionen pro Sekunde.

Durch den Einsatz hochintegrierter Schaltkreise wird bei der **4. Generation** eine weitere Steigerung der Rechenleistung erwartet (ca. 10 Millionen Additionen pro Sekunde).

Moderne Rechenanlagen sind durch ein Baukastensystem im Aufbau flexibel und daher universell einsetzbar; sie können zur Bearbeitung von maschinell lösbaren Aufgaben aus dem technisch-wissenschaftlichen, dem kommerziellen und dem verwaltungstechnischen Bereich herangezogen werden. Da hierbei nicht nur gerechnet wird, sondern weiter auch Daten verarbeitet werden (vergleichen, sortieren, mischen usw.), ist die heute allgemein gebräuchliche Bezeichnung **„elektronische Datenverarbeitungsanlage"** für die Beschreibung von Rechenanlagen zutreffender. Neben dieser Bezeichnung haben sich auch die Kennzeichnungen „Rechner", „Rechenanlage" und „Computer" eingebürgert; diese Begriffe beschreiben jedoch nicht alle Fähigkeiten des Datenverarbeitungssystems.

1.2. Einteilung der elektronischen Datenverarbeitungsanlagen

Die verschiedenen Datenverarbeitungsanlagen können nach unterschiedlichen Gesichtspunkten gegliedert werden; z.B. nach ihrer Einsatzart, ihrer Zugehörigkeit zu Generationen oder Serien und nach der Art der Darstellung der Daten innerhalb des Rechners. Grundsätzlich unterscheiden sie sich nach der Art der internen Zahlendarstellung in Analog- und Digitalrechner.

1.2.1. Analogrechner

Bei Analogrechnern werden die Zahlen durch physikalische Meßwerte abgebildet. Hierbei entspricht jede Zahl einer bestimmten physikalischen Größe wie z.B. einer Länge, einem Winkel oder der Höhe einer Spannung. Im Gegensatz zu den im nächsten Abschnitt beschriebenen Digitalrechnern, bei denen die zu verarbeitenden Werte durch quantisierte Größen dargestellt werden, arbeiten Analogrechner mit kontinuierlich veränderlichen Größen, die elektrisch oder mechanisch sein können.

Mechanische Rechner sind verhältnismäßig langsam und werden überwiegend in Regel- und Steuerkreisen eingesetzt. Analogrechner, die auf elektrischer Basis arbeiten, eignen sich besonders zur Lösung komplizierter Rechenvorgänge wie z.B. der Integration. Sie sind auch bei komplexen Aufgaben leichter programmierbar als Digitalrechner. Infolge der stetigen und zeitgleichen Abläufe finden Analogrechner als Simulatoren ein ideales Anwendungsgebiet. Nachteilig ist ihre geringe Genauigkeit.

1.2.2. Digitalrechner

In einem Digitalrechner werden die Zahlen durch Ziffernfolgen in einem Zahlensystem dargestellt. Nur die Ziffern dieses Zahlensystems müssen physikalisch realisiert werden. Ein Digitalrechner arbeitet mit diskreten Zuständen. Mit dieser Art der Zahlendarstellung können Zahlen mit beliebig vielen Stellen dargestellt werden. Digitalrechner sind universell verwendbar und ermöglichen bei entsprechendem technischen Aufwand eine beliebige Rechengenauigkeit.

Grundsätzlich könnte das Dezimalsystem zur Zahlendarstellung verwendet werden. Dazu müßten jedoch die 10 Zustände (Ziffern) des Systems physikalisch dargestellt und auch gespeichert werden (Denärprinzip). Dies wäre technisch sehr aufwendig und würde erhebliche physikalische Probleme bei der Speicherung mit sich bringen. Das Speichern von Informationen läßt sich z.Z. am wirtschaftlichsten auf magnetischer Basis realisieren. Hierbei wird die binäre Darstellung (Kombination von nur zwei verschiedenen Zuständen) der Information verwendet. Die binären Zustände „0" und „1" können durch verschiedene Magnetisierungsrichtungen, z.B. von Ferritkernen, dargestellt werden. Ein Darstellungselement mit nur 2 Ausdrucksmöglichkeiten heißt „Bit" (aus dem Englischen: **bi**nary dig**it** ≙ binäre Ziffer).

1.3. Aufbau digitaler Rechenanlagen

Zur Erläuterung des Aufbaus einer elektronischen Datenverarbeitungsanlage soll als Vergleich die Arbeitsweise des Menschen herangezogen werden. Soll z.B. die Addition von zwei Rechnungsbeträgen mit Hilfe einer Tischrechenmaschine erfolgen, so sind zur Lösung dieser Aufgabe eine Folge von Arbeitsschritten auszuführen, die im Normalfall so selbstverständlich sind, daß man sich deren kaum bewußt wird. Die Rechnungen selbst enthalten die einzelnen Rechnungsbeträge und stellen somit die „Datenträger" dar. Was muß nun im einzelnen getan werden?

a) Rechenmaschine auf Nullstellung bringen
b) den ersten Rechnungsbetrag lesen und den Wert des Betrages im Gedächtnis ablegen
c) erste Zahl aus dem Gedächtnis holen und in die Rechenmaschine eintippen
d) Taste „Addieren" drücken
e) zweiten Rechnungsbetrag lesen, im Gedächtnis abspeichern
f) zweite Zahl aus dem Gedächtnis in die Rechenmaschine eintippen
g) Taste „Addieren" drücken
h) Ablesen des Summenwertes und ebenfalls im Gedächtnis abspeichern
i) Summe aus dem Gedächtnis abrufen und auf Papier schreiben

Der Mensch kann diese Funktionen nur erfüllen, weil er durch seine geistigen Fähigkeiten in der Lage ist, den Verarbeitungsvorgang zu steuern. Das Vorgehen beim Bedienen der Rechenmaschine (z.B. „+"-drücken, wenn man addieren will) ist in seinem Gedächtnis gespeichert und kann mit einem Programm eines Rechners verglichen werden. Allgemein formuliert ereignen sich hier

eine **Daten-Aufnahme** (Lesen),

eine **Datenspeicherung** (Speichern),

eine **Datenverarbeitung** (Rechnen, Ordnen, Umformen),

eine **Datenausgabe** (Schreiben) und

eine **Datenübertragung**, wenn das Papier mit dem Summenbetrag an andere Stellen weitergegeben wird.

Diese hier genannten Grundfunktionen umschreiben ein Datenverarbeitungssystem. Da die elektronische Datenverarbeitungsanlage ebenfalls ein Datenverarbeitungssystem ist, muß sie diese genannten Grundfunktionen erfüllen. Damit ergibt sich schematisch der Aufbau einer Datenverarbeitungsanlage aus den 5 Grundbausteinen (Elementen).

1.3.1. Die Bausteine einer Rechenanlage

In diesem Abschnitt werden die Hauptgruppen einer DVA und ihre Funktionen beschrieben. Eine DVA kann in **Eingabe, Ausgabe** und **Zentraleinheit** unterteilt werden. Die Zentraleinheit selbst besteht aus dem **Hauptspeicher** (Arbeits- oder Internspeicher), dem **Steuerwerk** und dem **Rechenwerk.** Die Zentraleinheit ist in der Regel konstruktiv zu einer geschlossenen Baueinheit zusammengefaßt.

1.3.1.1. Eingabe (Lesen)

Wie der Mensch muß auch die Maschine die Informationen lesen und erkennen können, mit denen sie später rechnen soll. Bisher ist es nur möglich gewesen, einen Teil der menschlichen Fähigkeiten, nämlich Ziffern und Buchstaben zu lesen, durch technische Geräte zu ersetzen. Die Gründe hierfür liegen in der Vielfalt der Schriftarten und dem Fehlen von Normierungen. Hierbei sei an die Schwierigkeiten erinnert, die wir selbst beim Lesen von Handschriften haben. Eingabegeräte erfordern zum Datenlesen maschinenlesbare Schrift, festgelegte Datenordnung und genormte Datenträger. So haben z.B. heute verfügbare Geräte zum Lesen von Klarschrift bestimmte Normierungen in der Schriftart und auch in der Art der zu lesenden Belege. Der am weitesten verbreitete maschinenlesbare Datenträger ist die Lochkarte. Die hier verwendete, maschinell lesbare Schriftart ist die Lochschrift. Durch Kombinationen von „Loch" und „kein Loch" sind Ziffern, Buchstaben und Sonderzeichen (Punkt, Komma, Bindestrich usw.) darstellbar. Bevor die Daten also der Maschine zum Lesen angeboten werden können, müssen sie erst in einem zusätzlichen Arbeitsgang in dieser Lochschrift geschrieben werden. Dies geschieht in der Form, daß die Lochkarten mit dem Kartenlocher und die Lochstreifen mittels Streifenlocher oder Fernschreiber gelocht werden. Die Probleme und der Aufbau von Lochschriften werden im Abschnitt 1.3.3. näher erläutert.

1.3.1.2. Ausgabe (Schreiben)

Das maschinelle Drucken von Informationen ist in vielen Variationen möglich. Die Ergebnisse lassen sich so darstellen, daß die Belege keiner

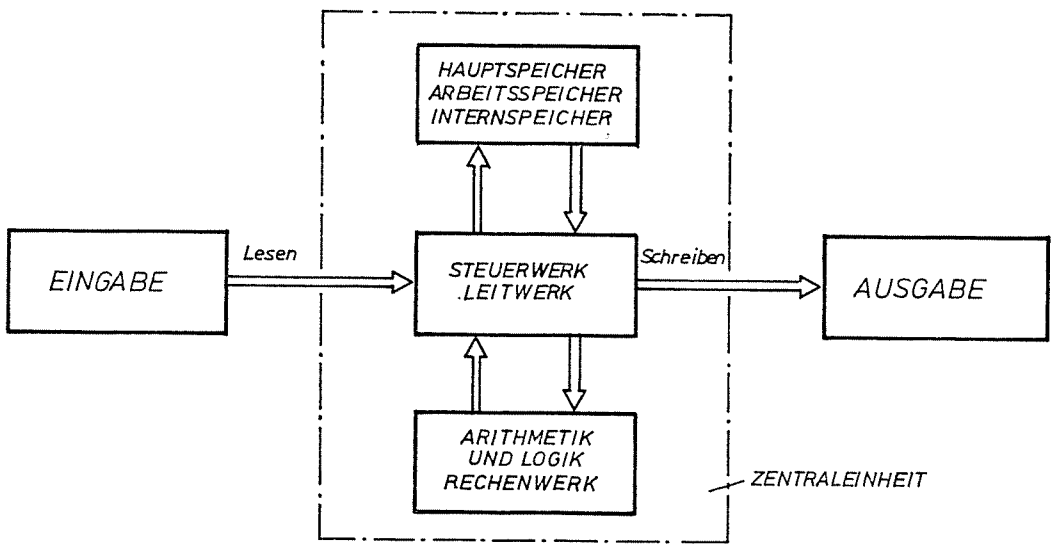


Abb. 1.1 — Blockschema mit den Datenbewegungen zwischen den Elementen einer DVA

weiteren Bearbeitung mehr bedürfen. Das Ausgeben von Daten kann aber auch das Stanzen von Lochkarten oder Lochstreifen bedeuten. Dazu werden die Daten aus dem Speicher über die entsprechenden Geräte und in der bestimmten Lochschrift ausgegeben. Die Ausgabe im Klartext erfolgt meistens über Schnelldrucker, Schreibmaschinen oder Fernschreiber.

1.3.1.3. Arbeitsspeicher

Der Arbeitsspeicher ist das Gedächtnis des Rechners. Alle Informationen, die der Rechner zur Verarbeitung benötigt oder die er bei dem Verarbeitungsvorgang gewinnt, hält er im Arbeitsspeicher fest. Damit eine Information im Speicher wiedergefunden werden kann, muß jeder Speicherplatz durch eine **Adresse** eindeutig gekennzeichnet sein. Man kann sich den Arbeitsspeicher als Regal mit vielen Fächern zum Ablegen von Informationen vorstellen. Jedes Fach hat als Adresse eine Nummer; es wird fortlaufend, mit Null beginnend, numeriert. Unter einer Adresse (einem Fach des Regals) können mehrere Bits abgelegt werden (häufig 8 Bits = 1 Byte). Solche kleinsten adressierbaren Einheiten stellen eine Speicherstelle dar. Die Gesamtzahl der Speicherstellen bestimmt die Kapazität eines Speichers. So hat z.B. ein Speicher mit einem Fassungsvermögen von 1024 Bytes ≙ 1 Kilo Byte = 1024 Fächer für je 8 Bits. Durch die Adressen ist es möglich, genau anzugeben, wo die Daten abgespeichert sind. Die Kenntnis über die Lage der eingelesenen Daten ist für ihre spätere Verarbeitung wichtig. Beim Schreiben in einen Speicher werden Informationen unter den angegebenen Adressen abgespeichert. Beim Lesen von Information aus einem Speicher wird der unter der angegebenen Adresse abgespeicherte Inhalt abgegeben. **Adresse und Inhalt einer Kernspeicherstelle sind also zwei grundverschiedene Dinge.** Die gelesene Information bleibt als solche weiter im Speicher erhalten. Sie kann immer wieder gelesen werden, solange unter der gleichen Adresse keine neue Information eingeschrieben wird. Eine Hauptaufgabe des Arbeitsspeichers — die Speicherung eingegebener oder während der Verarbeitung gewonnener Daten — ist damit beschrieben worden. Die zweite Hauptaufgabe wird im Abschnitt 1.3.1.5. behandelt.

1.3.1.4. Rechenwerk (Arithmetik und Logik)

Das Rechenwerk übernimmt die Funktion der Tischrechenmaschine. Hier erfolgt die eigentliche Verarbeitung der Daten. Daten verarbeiten bedeutet nicht nur Zahlen addieren, subtrahieren, multiplizieren oder dividieren, sondern auch Ordnen der Daten, z.B. Sortieren, Mischen, Auswählen, Vergleichen sowie Umformen und Ändern von Daten (z.B. von einer Verschlüsselung in eine andere). Anhand der logischen Operationen, die ein Rechenwerk neben den arithmetischen durchführen kann, ist es z.B. möglich, die Größe zweier Informationen zu vergleichen. Die Ergebnisse dieser Vergleiche werden durch Setzen von abfragbaren Anzeigen festgehalten (größer, gleich, kleiner).

So wie der Mensch die Zahlen aus dem Gedächtnis in die Tischrechenmaschine eintastet, müssen hier die Werte aus dem Speicher in das Rechenwerk gebracht und nach ausgeführter Verarbeitung als Ergebnis wieder abgespeichert werden. **Operationssteuerungen** ermöglichen die eigenständige Durchführung der Rechenoperationen. Alle arithmetischen Operationen werden auf die Grundrechnungsart „Addieren" zurückgeführt. Der schaltungstechnische Aufbau ist auf die verwendete interne Zahlendarstellung (Dualarithmetik, Dezimalarithmetik) abgestimmt. Einzelheiten hierzu werden in späteren Abschnitten behandelt.

1.3.1.5. Steuerwerk

Das Steuerwerk übernimmt die Aufgabe, die bisher beschriebenen Elemente einer elektronischen Datenverarbeitungsanlage sinnvoll miteinander in Beziehung zu bringen. Greifen wir auf das Beispiel einer Datenverarbeitung durch den Menschen zurück, so kann dieser eine ihm gestellte Aufgabe normalerweise nur dann lösen, wenn eine präzise Anweisung vorliegt, d.h., welche Rechenoperationen mit den verschiedenen Daten in welcher Reihenfolge auszuführen sind. Er muß die Ausführung der Anweisung veranlassen, z.B. das Lesen vom Datenträger, das Eintippen in den Tischrechner, das Aufschreiben der Ergebnisse. Dies sind die Funktionen, die innerhalb der Datenverarbeitungsanlage vom Steuerwerk übernommen werden müssen. **Der Steuerungsteil gibt also Operationsimpulse an die übrigen Anlagenteile.** Dazu muß dem Steuerteil aber auch die Arbeitsfolge bekannt sein. Die Arbeitsanweisungen, die dem Steuerteil zugeführt werden müssen, stehen im Arbeitsspeicher. Innerhalb eines maschinellen Datenverarbeitungssystems fallen damit dem Arbeitsspeicher grundsätzlich zwei Hauptaufgaben zu:

1. **das Speichern von Daten** und
2. **das Speichern der Arbeitsvorschrift.**

Die Arbeitsvorschrift wird allgemein als **Programm** bezeichnet. Ein Programm setzt sich aus vielen Einzelschritten zusammen, die nacheinander ausgeführt werden müssen. Diese Einzelschritte werden **Befehle** genannt. Die Befehle werden nacheinander aus dem Arbeitsspeicher in den Steuerteil übernommen und hier interpretiert (Interpretationsphase). Nach der Umwandlung in Operationsimpulse werden die angesprochenen Elemente der DVA zur Arbeit veranlaßt (Ausführungsphase). Dieser Vorgang wiederholt sich beim Ablauf jedes weiteren Befehls. Im Anschluß an die Ausführungsphase eines Befehls erfolgt die Interpretationsphase der nächsten Instruktion. Der Programmablauf besteht somit aus einem ständigen Wechsel zwischen I-Phase und A-Phase. Während der I-Phase besteht ein Informationsfluß immer in der gleichen Richtung und zwischen den gleichen Elementen (Arbeitsspeicher — Steuerwerk). In der A-Phase werden sowohl die Richtung des Informationsflusses als auch die beteiligten Elemente durch die auszuführende Operation bestimmt.

Die Befehle eines Programms stehen, wie wir gesehen haben, nacheinander im Kernspeicher. Man kann sich das so vorstellen, daß jeweils in einem Fach des Regals eine Befehlsinformation abgelegt ist. Weiß das Steuerwerk vor Beginn des Programmablaufes, in welchem Fach (Adresse) der erste Befehl vorhanden ist, kann das Steuerwerk selbst ermitteln, wo der nächste zu verarbeitende Befehl liegt; in unserem Beispiel nämlich bei Anfangsfach + 1. Diese Adreßerhöhung wird im **Befehlszählregister** des Steuerwerks automatisch vorgenommen, so daß ein Programm nach Festlegen der Anfangsadresse selbsttätig ablaufen kann.

Da das Steuerwerk das wichtigste Element (Intelligenz) einer elektronischen Datenverarbeitungsanlage ist, sollen hier die Aufgaben in Stichpunkten nochmals festgehalten werden. **Das Steuerwerk einer DVA hat also die Aufgabe, die im Arbeitsspeicher bereitliegenden Befehle eines Programmes nacheinander**

a) aus dem Arbeitsspeicher abzuholen (entspricht: Lesen einer gespeicherten Information, Befehl bleibt im Arbeitsspeicher erhalten),

b) zu entschlüsseln (Interpretation),

c) in Steuersignale umzusetzen und

d) mit diesen Signalen den Einsatz der anderen Elemente zu steuern.

Für die Ausführung von Behlen sind natürlich auch Datenbewegungen zwischen den einzelnen Elementen nötig. So hat das Steuerwerk z.B. zu veranlassen, daß Daten

a) in den Arbeitsspeicher kommen und hier gespeichert werden,

b) innerhalb des Arbeitsspeichers von einer Stelle an eine andere übertragen werden (von einem Fach in ein anderes Fach des Regals) und

c) in das Rechenwerk abgegeben oder für die Ausgabe bereitgestellt werden.

Dadurch, daß Programm und Daten im Arbeitsspeicher abgelegt sind, besteht die Möglichkeit, auch das Programm zu ändern, d.h., während des Ablaufs zu modifizieren. **Diese Speicherprogrammierung** (im Gegensatz zu festverdrahteten Programmen, z.B. Waschautomat) **und der selbsttätige Ablauf des Programms sind die entscheidenden Kennzeichen einer elektronischen Datenverarbeitungsanlage.** Rein technisch gesehen besteht zwischen Steuerwerk und Rechenwerk fast keine Trennung. Die Aufteilung dient hier nur der übersichtlichen logischen Darstellung.

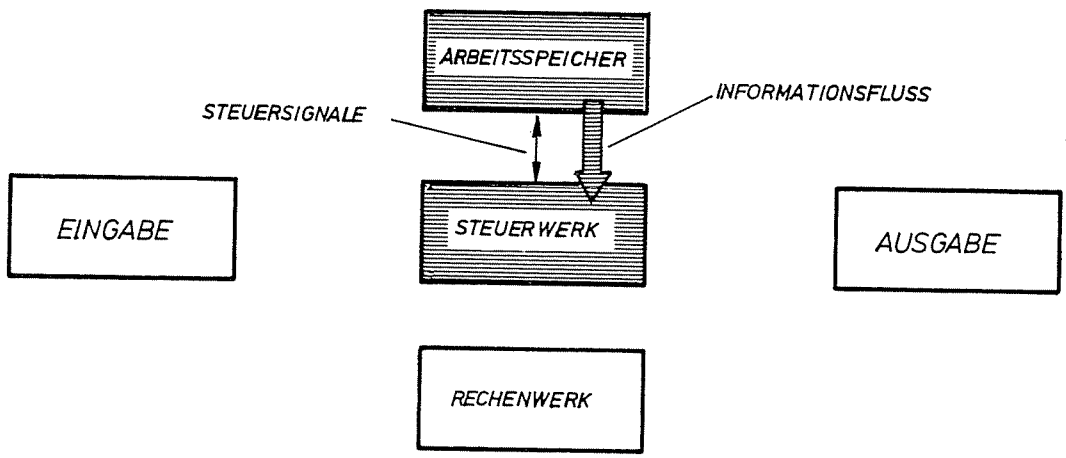


Abb. 1.2 — Befehlsübernahme

1.3.1.6. Beispiele zum Informationsfluß

Informationsfluß bei Befehlsübernahme

In der Interpretationsphase wird ein Befehl aus dem Arbeitsspeicher übernommen und im Steuerwerk analysiert. Informationsfluß besteht nur in Richtung Arbeitsspeicher-Steuerwerk. Für die anschließende Ausführungsphase werden für mehrere Operationen die Informationsflüsse angegeben. Die Befehlsübernahme ist bei allen Operationen gleich.

Informationsfluß bei Eingabe/Ausgabe-Operationen

Bei Eingabevorgängen werden die Daten, gesteuert vom Leitwerk, in den Arbeitsspeicher gebracht und dort abgelegt. Bei Ausgabeoperationen kommen die Daten aus dem Arbeitsspeicher zum Ausgabegerät ebenfalls über das Steuerwerk. Abb. 1.3 zeigt den Informationsfluß während der Ausführungsphase für eine Eingabe- und eine Ausgabeoperation.

Wie bereits erwähnt wurde, sind Eingabegeräte meist Lochkartenlesegeräte und Ausgabegeräte meist Schnelldrucker. Im weitesten Sinne können jedoch auch externe Speicher als Ein- bzw. Ausgabegeräte angesehen werden. Die **Externspeicher** (langsamer Zugriff, billig) werden zur Entlastung des Arbeitsspeichers (schneller Zugriff, teuer) eingesetzt. Sie sind deshalb von besonderer Bedeutung. Die externen Speicher (Magnetbandgeräte, Magnetplattengeräte, Magnettrommelgerät usw.) können sowohl Eingabe- als auch Ausgabefunktionen übernehmen. Sie lassen sich „lesen" und „beschreiben".

Beispiel: Zum maschinellen Doppeln des Inhalts von Magnetbändern muß die Information zunächst in den Arbeitsspeicher eingelesen werden; erst dann kann sie mit einem Ausgabebefehl aus dem Arbeitsspeicher auf das neue Band geschrieben werden.

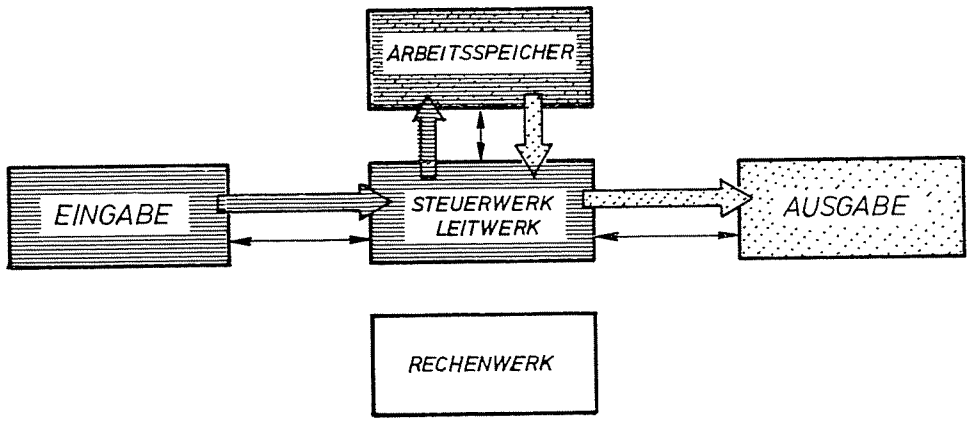


Abb. 1.3 — Informationsfluß bei E/A-Operationen

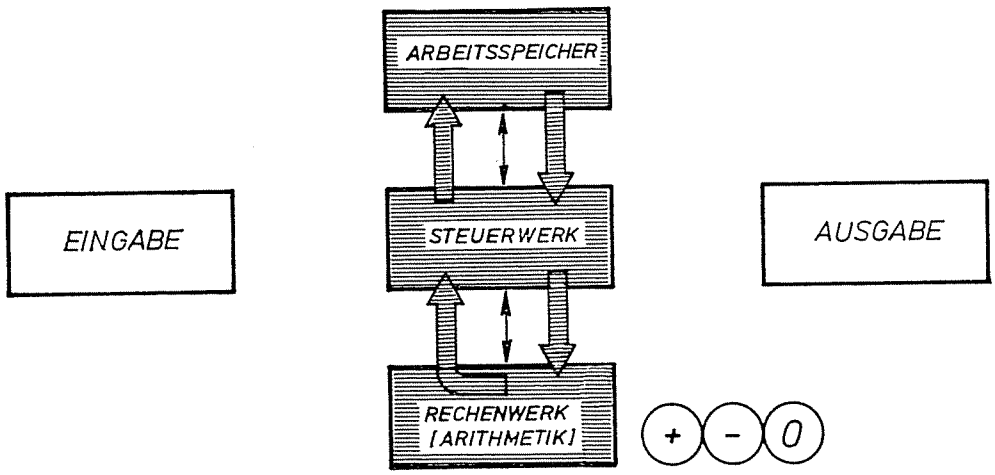


Abb. 1.4 — Informationsfluß bei Rechenoperationen

Informationsfluß bei arithmetischen Operationen

Bei der Durchführung einer arithmetischen Operation (vgl. hierzu Abb. 1.4) veranlaßt das Steuerwerk zuerst den Datentransport der Operanden zum Rechenwerk. Das Rechenwerk führt dann auf Anweisung des Steuerwerkes die gewünschte Rechenoperation aus. Das Ergebnis wird über das Steuerwerk wieder im Arbeitsspeicher abgelegt. Neben dem Ergebnis werden häufig bei allen arithmetischen Operationen auch Anzeigen gebildet, die angeben, ob ein Ergebniswert positiv, negativ oder Null ist. Diese Anzeigen werden im Rechenwerk gesetzt; sie sind abfragbar und können den Ausgangspunkt für nachfolgende logische Entscheidungen bilden. Es ist hierbei zu beachten, daß die Anzeigen nur für die unmittelbar davor ausgeführte Rechenoperation gelten, weil die Anzeigen durch jede evtl. folgende neue arithmetische Operation verändert werden können.

Informationsfluß bei logischen Operationen

Die Logik im Rechenwerk kann im wesentlichen folgende Funktionen ausführen:

a) Testen, ob eine Zahl größer, kleiner oder gleich Null ist,

b) Vergleichen von zwei Informationen und

c) logische Verknüpfung von zwei Informationen.

Wie Abb. 1.5 zeigt, liefern diese Operationen im allgemeinen kein Ergebnis im Kernspeicher (außer c). Sie setzen nur die entsprechenden abfragbaren Anzeigen im Rechenwerk. Das Steuerwerk stellt dem Rechenwerk die oder den Operanden zur Verfügung und leitet die logische Operation ein. Im Rechenwerk werden die Anzeigen größer, kleiner oder gleich Null gesetzt.

Informationsfluß bei Übertragungsoperationen

Übertragungsoperationen (vgl. hierzu Abb. 1.6) werden durchgeführt, um Datenfelder von einer Stelle des Arbeitsspeichers zu einer anderen Stelle zu übertragen. Es kann vorkommen, daß Daten für bestimmte Vorgänge an mehreren Stellen des Arbeitsspeichers benötigt werden oder bestimmte Felder auf definierte Ausgangswerte zu setzen sind. Der Inhalt einer angegebenen (adressierten) Speicherstelle wird zunächst in das Steuerwerk übernommen und von hier unter einer neuen Adresse wieder in den Arbeitsspeicher zurückgeschrieben. Die ursprüngliche Information bleibt erhalten, da sie ja nur „gelesen" wurde.

Sprungoperationen

Mit der Einführung der Sprungbefehle (John v. Neumann) wurde ein entscheidender Schritt zur Weiterentwicklung getan. Im Regelfall sind die Befehle im Arbeitsspeicher hintereinander abgelegt und während des Ablaufs werden sie in dieser Reihenfolge ausgeführt. Das entspricht einem linearen Programmablauf. Soll nun hiervon abgewichen werden, so können Sprungbefehle eingesetzt werden, die sich im allgemeinen die bei Rechenoperationen gesetzten Anzeigen zunutze machen. Diese Tatsache ermöglicht es, abhängig von der Art der Anzeige, Teile des linearen Ablaufes auszulassen (zu überspringen) und mit Befehlen, die an einer anderen Stelle des Arbeitsspeichers stehen, fortzufahren. Sprungbefehle müssen also auch Angaben über den nächsten auszuführenden Befehl mit sich führen. Wird in einer solchen Verzweigungsinstruktion die Adresse für den nachfolgend auszuführenden Befehl zurückgesetzt, so können

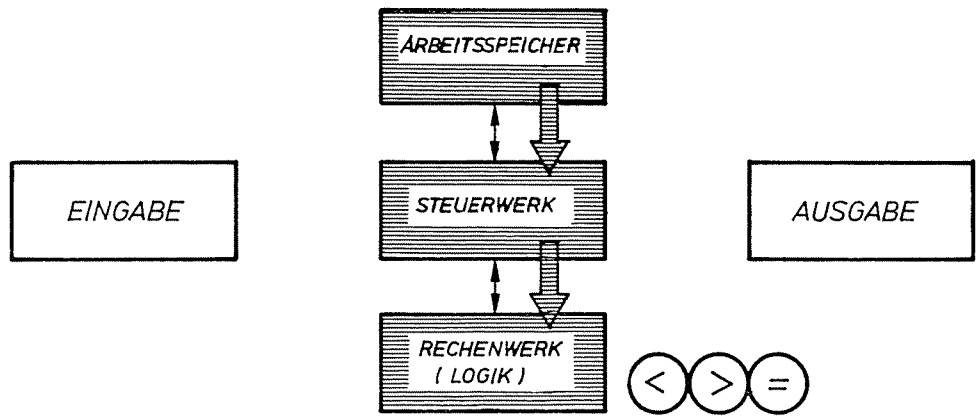


Abb. 1.5 — Informationsfluß bei logischen Operationen

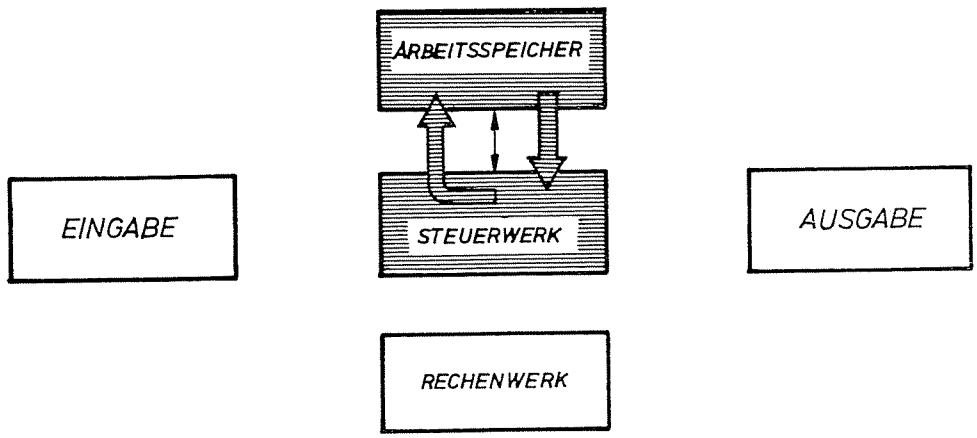


Abb. 1.6 — Informationsfluß bei Übertragungsoperationen

bestimmte Programmteile mehrmals durchlaufen werden, bis z.B. eine bestimmte Bedingung erfüllt ist (sog. Programmschleifen). Von bedingten Sprüngen spricht man, wenn von dem linearen Ablauf nur dann abgewichen werden soll, wenn die Bedingungsanzeigen den geforderten Zustand aufweisen. Unbedingte Sprunganweisungen nehmen auf die Bedingungsanzeigen keine Rücksicht. Der lineare Ablauf wird — für Rücksprünge und Sprünge zum Programmanfang — auf jeden Fall verlassen.

1.3.2. Grundsätzlicher Befehlsaufbau

Für die maschinelle Lösung eines Problems muß dieses zunächst in eine Reihe von Einzelschritten aufgelöst werden. Jeder einzelne Arbeitsschritt muß der Maschine in Form eines Befehls (Instruktion) eingegeben werden. Die Gesamtheit der zur Lösung eines Problems benötigten Instruktionen nennt man ein Programm. Das Programm ist zusammen mit den Daten (Eingabewerte, Konstanten) im Arbeitsspeicher einer DVA abgespeichert. Deshalb wollen wir nochmals herausstellen, daß Daten passive Informationen, Befehle aktive Informationen darstellen. Da das Programm im Arbeitsspeicher steht, können evtl. auch Befehle durch die Anlage selbst verändert werden (Befehlsmodifikation). Auf den selbsttätigen Ablauf eines Programms unter Kontrolle des Steuerwerks wurde bereits hingewiesen. Der geschilderte Funktionsablauf im Steuerwerk läßt erkennen, aus welchen Teilen sich ein Befehl zusammensetzen muß. Ein Befehlswort umfaßt die vollständige Beschreibung eines Arbeitsschrittes und besteht aus zwei Teilen, dem **Operationsteil** und dem **Adreßteil.**

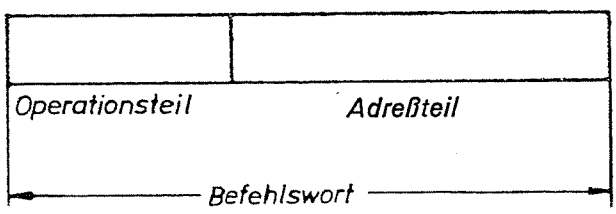


Abb. 1.7 — Aufbau eines Befehlswortes

Der Operationsteil bestimmt, was der Rechner tun soll, d.h., welche Operation eine Anlage ausführen soll. Der Adreßteil enthält die Adresse des Operanden, den der Rechner bei dieser Operation benutzen soll. Er enthält sozusagen nur die Fachnummer innerhalb des Regals, in der sich die zu bearbeitende Information befindet, beschreibt aber nicht den Inhalt des Faches.

Ein Befehl muß eine dem Rechner verständliche Anweisung enthalten. Für die Darstellung muß also eine dem Rechner angepaßte Sprache verwendet werden, die sogenannte Maschinensprache. Die Struktur der Maschinensprache wird durch die schaltungstechnische Realisierung der Rechner-Hardware bestimmt. Für jede von der Maschine ausführbare Operation ist eine durch die Schaltungstechnik bestimmte Verschlüsselung vergeben. Zur Codierung werden heute meistens binäre Darstellungsweisen verwendet, weil damit die beste Eingliederung in die logischen Schaltungen erreicht wird. Diese Verschlüsselung ist einfach und betriebssicher. Der für jeden speziellen Typ von Operationen gültige Code (Operationscode) wird für jeden einzelnen Rechner schon vom Konstrukteur festgelegt. Abb. 1.7 zeigt den Aufbau eines Befehlswortes.

Da die Adressen innerhalb des Arbeitsspeichers fortlaufend vergeben sind (Nummer der Fächer), handelt es sich bei der Adressenangabe um eine reine Zahl. Für die Verschlüsselung der Adresse muß die interne Zahlendarstellung für die jeweilige Maschine verwendet werden. In der Regel sind dies Dualzahlen (vgl. hierzu Abschn. 1.3.3.). Technisch gesehen ist der Arbeitsspeicher ein Kernspeicher, d.h., als Speicherelemente dienen kleine Ferritkerne, die durch zwei verschiedene Magnetisierungszustände in der Lage sind, zwei getrennte Informations-

zustände (binär) anzunehmen. Deshalb muß für die interne Codierung meist die binäre Darstellungsweise gewählt werden. Der Adreßteil enthält also als Zahl die Adressen (Lage) der Operanden.

1.3.2.1. Einteilung der Befehle

Entsprechend den Bausteinen einer DVA lassen sich die Befehle in 4 Gruppen einteilen:

1. **Ein-Ausgabeoperation:** Lesen, Schreiben über E/A-Geräte
2. **Übertragungsoperation:** Verschieben von Daten im Speicher
3. **Rechenoperationen:**
   a) arithmetische Operationen (Addieren, Subtrahieren, Multiplizieren, Dividieren)
   b) logische Operationen (Vergleichen, Testen, UND, ODER)
4. **Verzweigungsbefehle:**
   a) unbedingte Sprungbefehle (verzweigen immer)
   b) bedingte Sprungbefehle (verzweigen nur, wenn eine bestimmte Bedingung erfüllt wird)

1.3.2.2. Adreßprinzipien

In einem Befehlswort können im Adreßteil auch mehrere Adressen stehen (vgl. hierzu Abb. 1.7). Nach der Anzahl dieser Adressen läßt sich eine Klassifizierung der elektronischen Datenverarbeitungsanlagen nach Ein-Adreß-, Zwei-Adreß- und Drei-Adreßmaschinen vornehmen. Obwohl selten angewendet, soll zum besseren Verständnis auch das Drei-Adreß-Prinzip besprochen werden. Wir wollen seine Funktionsweise anhand eines Beispiels erklären, und zwar soll die Aufgabe a + b = c gelöst werden. Unter der Adresse 1 ist der Summand a und unter der Adresse 2 der Summand b gespeichert. Im Operationscode steht der Schlüssel für einen Additionsbefehl. Bei der Ausführung werden dem Rechenwerk die unter ihrer Adresse gefundenen Werte für die Summanden a und b übergeben. Das Rechenwerk führt die Addition aus. Das Steuerwerk veranlaßt die Abspeicherung des bereitgestellten Ergebnisses unter der Adresse 3 im Arbeitsspeicher. Für das Ergebnis und die beiden Summanden ist jeweils eine Adresse und damit jeweils ein Speicherplatz vorhanden.

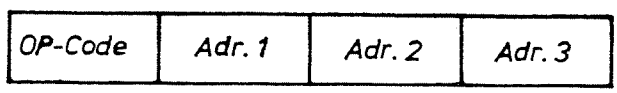


Abb. 1.8 — Drei-Adreß-Prinzip

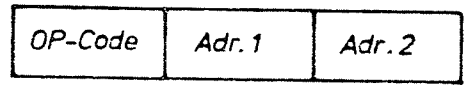


Abb. 1.9 — Zwei-Adreß-Prinzip

Bei Zwei-Adreßmaschinen kann für das Ergebnis keine eigene Adresse angegeben werden. Das Ergebnis wird im allgemeinen dort abgespeichert, wo der erste Operand ausgelesen wurde. Der ursprüngliche Inhalt dieser Speicherstelle (1. Operand) wird überschrieben und geht verloren (wird die Information im weiteren Programmablauf noch benötigt, muß sie vorher zusätzlich in eine andere Speicherstelle übertragen werden).

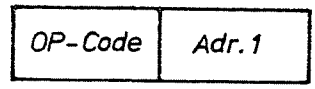


Abb. 1.10 — Ein-Adreß-Prinzip

Bei einer Ein-Adreßmaschine kann immer nur die Adresse eines Operanden angegeben werden. Für die Lösung der Aufgabe sind also mehrere Befehle nötig. Gleichzeitig muß auch ein allgemeiner Zwischenspeicher (ein Register im Rechenwerk) bereitgestellt werden, mit dessen Inhalt die Addition durchzuführen ist. Zur Lösung müßten folgende Befehle durchgeführt werden:

Bringe den Summanden a (unter Adr. 1) in den Zwischenspeicher (Akkumulator).

Addiere den Summanden b (unter Adr. 2) auf den Inhalt des Zwischenspeichers.

Bringe das Ergebnis aus dem Zwischenspeicher in die Speicherstelle für die Summe c (unter Adr. 3).

Der Zwischenspeicher selbst braucht nicht angegeben zu werden, da er bei Ausführung der entsprechenden Befehle automatisch angesprochen wird. Zusammenfassend ist festzustellen: **Mehradreß-Befehle sind leistungsfähiger als Einadreß-Befehle, benötigen jedoch zur Abspeicherung mehr Platz. Programme mit Mehr-Adreß-Befehlen sind einfacher und übersichtlicher.**

1.3.2.3. Wort- und Stellenmaschinen

Bei einer **Wortmaschine** ist die Länge der Operanden immer gleich. Der Arbeitsspeicher ist praktisch in Informationseinheiten konstanter Länge (z.B. 16, 32 oder 40 Bits) aufgeteilt. Diese fest vorgegebenen Speicherbits entsprechen

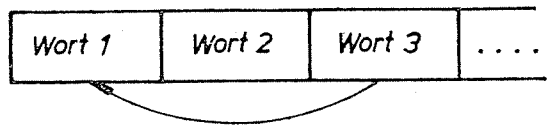


Abb. 1.11 — Wortprinzip

einem Maschinenwort, das gleichzeitig über parallele Schaltkreise verarbeitet werden kann. Jede Informationseinheit, hier ein Wort, hat eine Adresse. Somit wird bei der Wortmaschine das Wort zur kleinsten adressierbaren Einheit. Durch die Parallelübertragung kann z.B. ein Wort bitparallel von einem Platz auf einen andern gebracht werden. Zwar ist hierbei der technische Aufwand größer, jedoch sind Wortmaschinen schneller als Stellenmaschinen, weil nur ein Verarbeitungszyklus zur Übertragung genügt. Nachteilig bleibt hier die schlechte Ausnutzung des Speichers, wenn Maschinenwort und Daten unterschiedlich groß sind. Ist z.B. ein Wort 32 Bits lang und werden für die Darstellung eines Datenelements nur 8 Bits benötigt, bleibt der Rest ungenutzt.

Stellenmaschinen können — rein technisch betrachtet — über eine bestimmte Anzahl von Schaltkreisen nur eine begrenzte Information verarbeiten. Im extremsten Fall wäre dies ein Speicherbit. Jedoch sind meist mehrere Bits zu einer Speicherstelle (Byte) zusammengefaßt. Die kleinste adressierbare Speicherstelle ist somit das Byte, das aus 8 Bits besteht. Stellenmaschinen verarbeiten in einem Zyklus jeweils nur eine Stelle. Dadurch wird die Verarbeitung an sich langsamer (weil mehrere Verarbeitungszyklen notwendig sind), jedoch läßt sich damit eine optimale Anpassung der Speicherstellen an die Dateninformation erreichen, wenn unterschiedliche Längen bei den Daten auftreten. Hierbei entsteht ein weiteres Problem, wie nämlich zusammenhängende Speicherbereiche zu kennzeichnen sind. Möglichkeiten bestehen durch Wortmarken, Begrenzungsbits und — wie bei fast allen modernen Rechenmaschinen — durch Längenangaben. Zur Markierung dient dann die Anfangsadresse und die Längenangabe für einen Operanden. So hat z.B. eine Übertragungsinstruktion für eine Zwei-Adreßmaschine folgendes Aussehen:

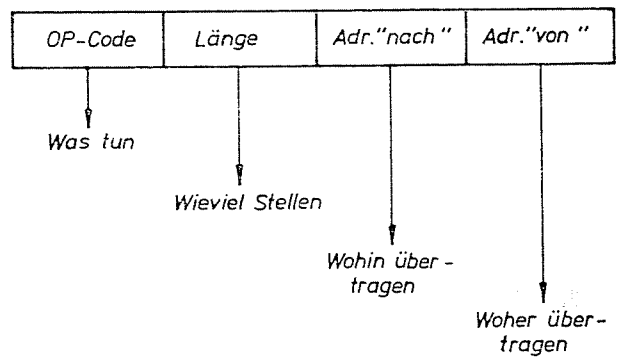


Abb. 1.12 — Befehl mit Längenangabe

Bei modernen Rechnern sind immer variable und feste Längen nebeneinander möglich; sie sind auch Mehr-Adreß-Rechner, d.h., sie führen abhängig von der auszuführenden Operation eine oder mehrere Adressen im Befehlswort. Für jeden Rechnertyp steht eine bestimmte Anzahl von Operationen zur Verfügung (bei kleinen Anlagen etwa 30; bei großen Anlagen mehr als 100). Die ausführbaren Operationen entsprechen dem Befehlsvorrat einer Maschine. Er umfaßt also alle Befehle, die der Rechner versteht (in Maschinensprache) und zu deren Ausführung (durch die Konstruktion bedingt) er in der Lage ist. Da man bei der Anwendung eines Befehls dessen Aufbau kennen muß, sind die Befehlsworte in einer Befehlsliste zusammengestellt. Die Befehlsliste beschreibt also für jeden gültigen Befehl eines Rechnertyps, wie der Operationsteil und wie die Struktur des Adreßteils aufgebaut sein muß.

1.3.2.4. Sprung- und Sequenzmaschinen

Der Vollständigkeit halber sei hier noch die Einteilungsmöglichkeit in Sprung- und Sequenzmaschinen erwähnt. Bei Sprungmaschinen ist praktisch jeder auftretende Befehl ein Sprungbefehl, da die Befehle im Kernspeicher nicht zusammenhängend abgelegt sind. Sequenzmaschinen verarbeiten die Befehle in der Reihenfolge, wie sie hintereinander im Kernspeicher abgelegt sind. Dieser lineare Ablauf kann nur durch spezielle Sprungbefehle unterbrochen werden (vgl. hierzu Abschn. 1.3.1.6. unter Sprungoperationen). Moderne Rechner sind Sequenzmaschinen.

1.3.3. Informationsdarstellung

In diesem Abschnitt wird ein Überblick über die Möglichkeiten der Informationsdarstellung in einem Rechner gegeben. Auf eine ins einzelne gehende Behandlung von Verschlüsselungsmethoden wird verzichtet. Unter Berücksichtigung der bisherigen Ausführungen ergibt sich für die zusammenfassende Übersicht zur Informationsdarstellung folgendes Bild:

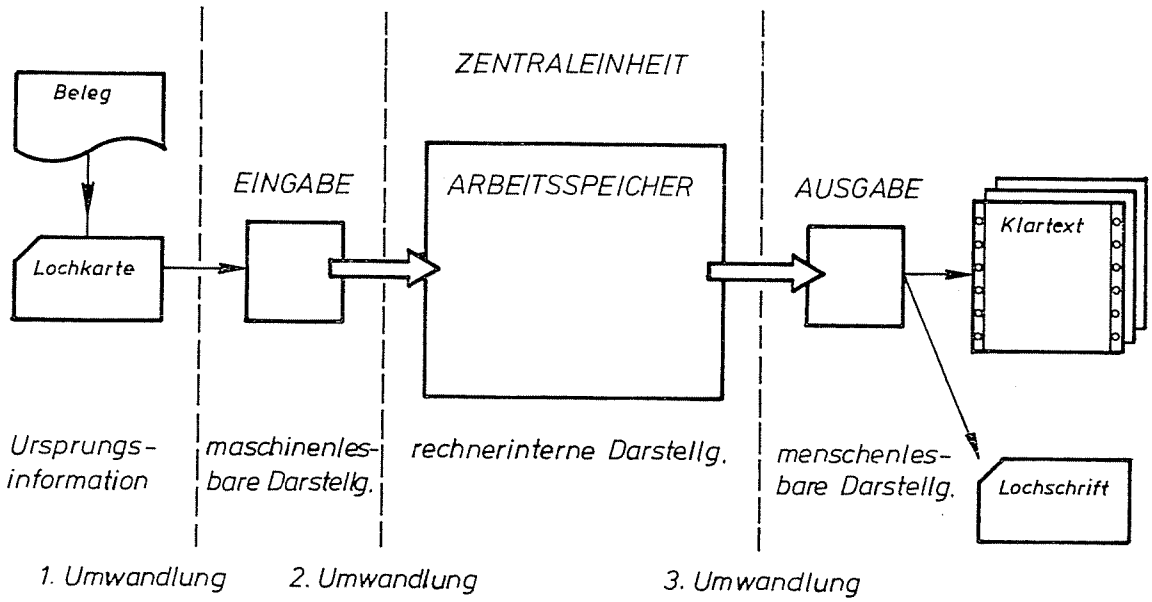


Abb. 1.13 — Schematische Darstellung der Datenverschlüsselung

Zunächst müssen die Daten von den Urbelegen auf einen maschinell lesbaren Datenträger gebracht werden. Dazu werden sie beispielsweise mit Hilfe eines Kartenlochers und bestimmter Lochanweisungen auf Lochkarten abgelocht; ihre Darstellungsform wird somit umgewandelt **(1. Umwandlung).** Dies geschieht in der Regel über Geräte, die der Mensch selbst bedienen muß (Locherin). Erst dann liegen die Daten in einer Form vor, die das „Lesen" über die Eingabegeräte möglich macht. Die 1. Umwandlung kann entfallen, wenn Eingabegeräte eingesetzt werden, die die Belege direkt lesen können (z.B. optischer Belegleser, Magnetschriftleser u.a.).

Bei der Verarbeitung der Daten innerhalb der Zentraleinheit wird wieder eine spezielle, die sog. rechnerinterne Datendarstellung, gefordert. Die Daten werden einer **2. Umwandlung** unterworfen, die schon „hardware-mäßig" (durch feste Verdrahtung) in den Eingabegeräten durchgeführt wird. Damit sind die Informationen speicherbar und sie können verarbeitet werden. Für die interne Verschlüsselung der Daten muß man zwei Gruppen von Informationen unterscheiden:

a) die Daten als zu verarbeitende passive Information und

b) die Befehle als aktive Information, die eine Verarbeitung der Daten bewirken sollen (vgl. hierzu Abschn. 1.3.1.5.).

Die **3. Umwandlung** der Daten ist für die Ausgabe notwendig. Die Daten müssen aus der internen Darstellung in eine für den Menschen lesbare Form gebracht werden. Diese 3. Umwandlung wird in den Ausgabegeräten vorgenommen; deshalb müssen die Daten schon intern den Möglichkeiten des Ausgabegerätes angepaßt werden. Sollen die Ergebnisse z.B. einer Maschine und nicht dem Menschen zur weiteren Verarbeitung angeboten werden, kann man natürlich auch Ausgabegeräte verwenden, die maschinell lesbare Datenträger erzeugen (z.B. Lochkarte, Lochstreifen, Bänder usw., Sonderfall:Klartext für optische Leser).

1.3.3.1. Die Lochkarte

Die Lochkarte ist der am häufigsten verwendete maschinell lesbare Datenträger. Die hierfür verwendete Lochschrift ist eine maschinell lesbare Schriftart, in der sich durch Kombinationen von „Loch" und „kein Loch" Ziffern, Buchstaben und Sonderzeichen darstellen lassen.

Eine Lochkarte hat die Abmessungen von 18,7 cm x 8,3 cm und besteht aus nicht leitendem Karton (0,2 mm stark). Die Einteilung einer Lochkarte ist genormt und besteht aus 80 Spalten und 12 Zeilen (vgl. hierzu Abb. 1.14). Die Zeilen sind nach Ziffern benannt (12, 11, 0—9). Die Normallochzone bilden die Zeilen 0—9. Die Überlochzone besteht aus den Zeilen 12 und 11. Der Eckenabschnitt ist ein Hilfsmittel zur Erkennung von falsch eingelegten Lochkarten innerhalb eines Stapels.

Eine Ziffer ist innerhalb einer Spalte durch ein Loch in der entsprechenden Zeile darstellbar. Buchstaben werden durch zwei Löcher in einer Spalte verschlüsselt. Dieses Lochpaar setzt sich aus einem Loch in den Zeilen 1—9 und einem

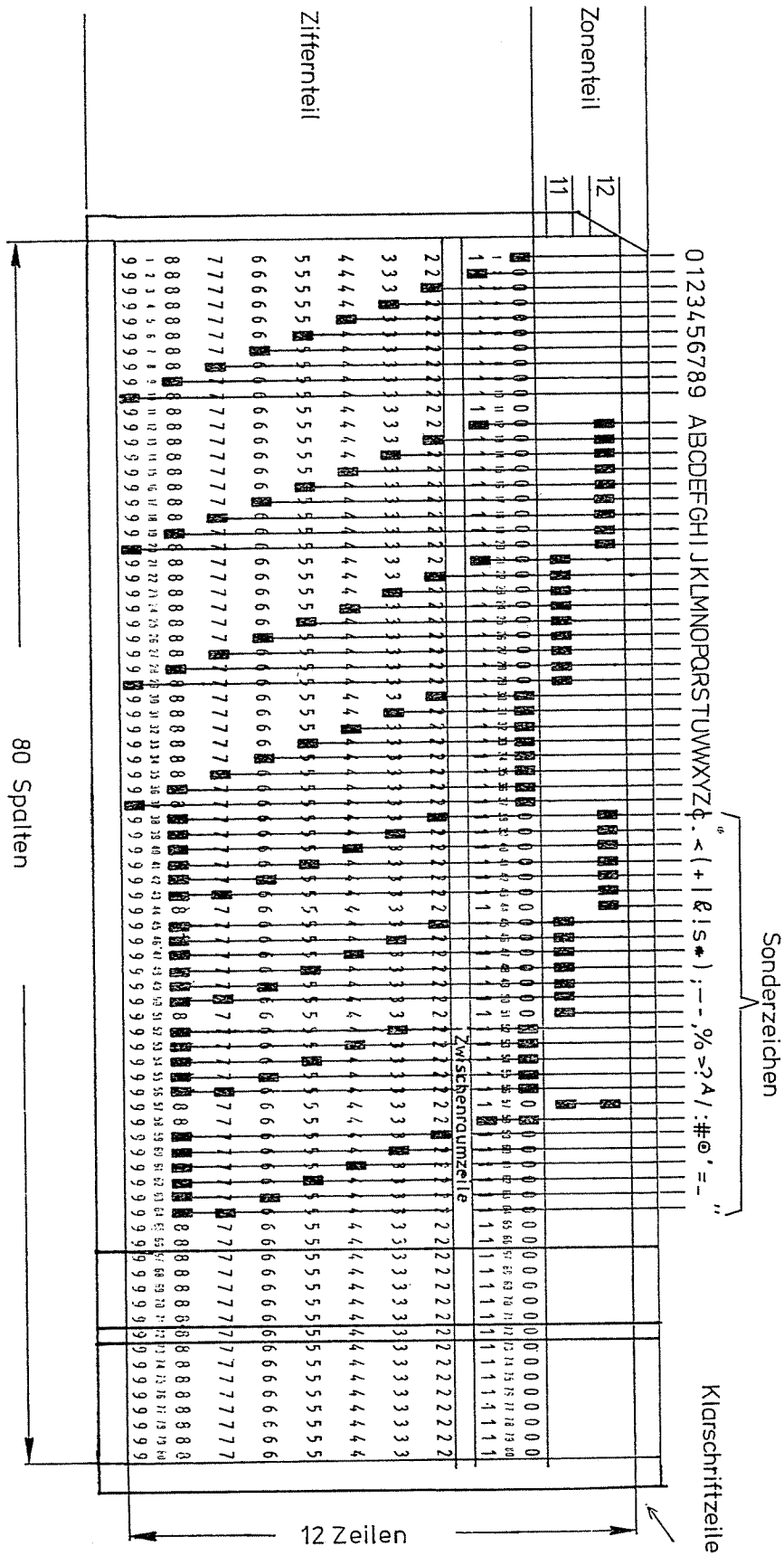


Abb. 1.14 — Zeichenvorrat einer Lochschrift (Lochstellenkombinationen)

ZONENTEIL	ZIFFERNTEIL										
	KEINE	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
KEINE	LEER-ZEICH.	0	1	2	3	4	5	6	7	8	9

Abb. 1.15 — Lochkartencode

Loch in den Zeilen 12, 11 und 0 zusammen. Dabei ergeben sich ein Zifferteil (1—9) und ein Zonenteil (12, 11, 0).

Wie aus Abb. 1.15 ersichtlich, ergeben sich somit folgende Gruppen:

	Zonenlochung	Ziffernlochung
Buchstaben A — I	Zeile 12	1 — 9
Buchstaben J — R	Zeile 11	1 — 9
Buchstaben S — Z	Zeile 0	2 — 9
Ziffern 0 — 9	keine	0 — 9

Sonderzeichen sind durch zwei oder mehrere Lochungen in einer Spalte gekennzeichnet.

Für die Vorzeichendarstellung innerhalb der Lochschrift wurde folgende Vereinbarung getroffen. Mehrstellige Zahlen benötigen mehrere nebeneinanderliegende Spalten (≙ Lochfeld). Bei Zahlenangaben erhält die Spalte mit der niedrigsten Stelle ein zusätzliches Überloch. Dabei steht das 11er Überloch für das Minuszeichen; das 12er Überloch für eine positive Kennung kann auch weggelassen werden. Lochkarten werden mit dem Kartenlocher nach den Vorschriften der Lochanweisung abgelocht. Die Lochanweisung gibt an, welche Daten aus dem Ursprungsbeleg in welche Spalten der Lochkarte zu stanzen sind. Meistens haben die Locher eine zusätzliche Schreibeinrichtung, mit der alle Lochungen spaltengerecht am oberen Rand der Lochkarte (Klarschriftzeile) im Klartext mitgeschrieben werden. Damit sind die gelochten Daten auch für den Menschen leichter lesbar. Entstehen Lochkarten bei einem Ausgabevorgang (Kartenstanzeinrichtung), kann die Klarschriftzeile nachträglich durch einen Übersetzungsvorgang (Lochschriftübersetzer) gewonnen werden.

Es gibt verschiedene Arten von Lochkarten. **Ziffernlochkarten** haben nur die 10 Lochzeilen aufgedruckt. **Verbundlochkarten** sind mit erklärendem Text versehen. Sie eignen sich auch zur Karteiführung. **Zeichenlochkarten** haben Zeichenfelder, in denen durch Strichmarkierungen Eintragungen von Hand vorgenommen werden können. Die Markierungen werden in einem gesonderten Arbeitsgang auf speziellen Geräten in Lochungen auf derselben Karte umgesetzt. Daneben gibt es noch weitere spezielle Lochkartenarten, auf die hier nicht näher eingegangen werden soll.

1.3.3.2. Rechnerinterne Darstellung

In der Zentraleinheit sind neben den Daten auch Befehle darzustellen. Daten können aus Zahlen sowie Text- und Steuerinformationen bestehen. Steuerinformationen sind Zeichen, die zur Steuerung von peripheren Geräten und zur gegenseitigen Abgrenzung von Informationen dienen (z.B. Buchstaben-, Ziffernumschaltung, Satzende, Blockende, Wagenrücklauf, Zeile usw.). Andererseits ist es innerhalb der Zentraleinheit nur möglich, Informationen durch eine Kombination von nur zwei Zuständen (binär) darzustellen. Ein Darstellungselement mit nur zwei Ausdrucksmöglichkeiten heißt binäres Element oder auch Bit. Die beiden möglichen Zustände dieses Elements werden mit „0" und „1" bezeichnet. (Um Verwechslungen mit den Dezimalziffern 0 und 1 zu vermeiden, können die möglichen Zustände auch mit den Buchstaben „O" und „L" ausgedrückt werden.) Sollen mehr als zwei Bedeutungen beschrieben werden, sind mehrere binäre Elemente zu kombinieren. Mit zwei Binärelementen lassen sich z.B. 4 Bedeutungen verschlüsseln, nämlich 00, 01, 10 und 11. Die Zuordnung einer einzelnen Bedeutung an eine bestimmte Kombination nennt man Codierung. Die Zuordnungsvorschrift für einen bestimmten Zeichenvorrat ist der Code. Die Frage, wieviel binäre Elemente man braucht, um alle

Zeichen eines Repertoires codieren zu können, wird durch den Logarithmus dualis (ld) der Anzahl der Zeichen beantwortet. (Zur Darstellung von 100 Zeichen braucht man ld 100 = 6,64 ≈ 7 binäre Elemente.) Trotzdem wird die Anzahl der Elemente eines Codes oft aus Sicherheitsgründen über das notwendige Maß hinaus vergrößert. Die zusätzlichen Elemente, die nicht direkt zur Erkennung eines Zeichens notwendig sind, werden zur Fehlerkennung herangezogen und allgemein als Redundanz (Weitschweifigkeit) bezeichnet.

Die Darstellung von Zahlen

Zur binären Darstellung von Zahlen werden heute überwiegend zwei Verfahren verwendet. Dies sind die Darstellung im Dualsystem und die Darstellung im Dezimalsystem mit binärer Zifferncodierung. Die Darstellung von Zahlen erfolgt in Stellenwert- oder Positionssystemen, die auf einer bestimmten ganzzahligen Basis (> 1) aufgebaut sind. Stellenwertsysteme sind dadurch charakterisiert, daß der Wert ihrer Ziffern abhängig von ihrer Stellung innerhalb der Zahl ist. Die Stellenwertsysteme werden meist nach ihrer Basis benannt, z.B. Dezimalsystem (Basis 10), Dualsystem (Basis 2), Oktalsystem (Basis 8). In Abb. 1.16 werden zwei Stellenwertsysteme gegenübergestellt. Der Ziffernwert ist gleich der angeschriebenen, für das System gültigen Ziffer. Stufenzahl ist die ganzzahlige Potenz der Basis. Der Stellenwert ergibt sich aus dem Produkt Ziffernwert x Stufenzahl. So viele Einheiten einer Stufe, wie die Basis angibt, bilden eine Einheit der nächsten Stufe. Im Dezimalsystem ergeben z.B. 10 Einheiten (Basis 10) einer Stufe eine Einheit der nächsthöheren Stufe.

Dualsystem

Eine Dualzahl ist eine Zahlendarstellung mit der Basis 2, bei der jede Ziffer durch Binärzeichen dargestellt wird, wobei die Ziffernstellen nach ihrer Wertigkeit, also nach Potenzen von 2, geordnet sind.

a) Dezimalsystem

Basis: 10
Gültige Ziffern: 0, 1, ... 9
Schreibweise: z.B.: 2 1 1

$2 \cdot 10^2 + 1 * 10^1 + 1 \times 10^0$

Stellenwert	=	Ziffernwert	*	Stufenzahl
200	=	2	*	10^2
10	=	1	*	10^1
1	=	1	*	10^0
211				

Zahlenwert: 211

b) Dualsystem

Basis: 2
Gültige Ziffern: 0, 1
Schreibweise: z.B.: 1 0 1

$1 * 2^2 + 0 * 2^1 + 1 * 2^0$

Stellenwert	=	Ziffernwert	*	Stufenzahl
4	=	1	*	2^2
0	=	0	*	2^1
1	=	1	*	2^0
5				

Zahlenwert: 5

Abb. 1.16 — Stellenwertsysteme

Jede im Dezimalsystem auftretende Zahl kann im Dualsystem (oder auch anderen) dargestellt werden. Im Dualsystem ist der Stellenwert durch Potenzen von 2, also 2^0, 2^1, 2^2, 2^3 usw. (entspricht im Dezimalsystem 10^0, 10^1, 10^2 usw.) gegeben. Die Stufenzahl im Dualsystem wird auch Bitwertigkeit genannt.

0	0	0	0	0	0	0	0	1	1	1	0	1	= Dualzahl
2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	= Bitwertigkeit
–	–	–	–	–	–	–	–	16	8	4	2	1	

Abb. 1.17 — Umrechnung dual-dezimal

Somit ergibt sich nach Abb. 1.17 der Dezimalwert zu: 16 + 8 + 4 + 1 = 29.

Für die Umrechnung dezimal-dual wird im allgemeinen das Verfahren der fortgesetzten Division angewendet (vgl. hierzu Abb. 1.18).

dezimal ⟶ dual

					Ziffernwert * Stufenzahl
$(29)_{10}$	: 2 =	14	Rest	1	⟶ 2^0
14	: 2 =	7	Rest	0	2^1
7	: 2 =	3	Rest	1	2^2
3	: 2 =	1	Rest	1	2^3
1	: 2 =	0	Rest	1	2^4

$(1\,1\,1\,0\,1)_2$

Abb. 1.18 — Umrechnung dezimal-dual

Die Rechenregeln für die 4 Grundrechnungsarten entsprechen auch im Dualsystem denen des Dezimalsystems.

Beispiel einer Addition:

dezimal	dual
26	11010
+ 2	+ 010
28	11100

Daraus ergeben sich folgende Rechenregeln:

0 + 0 = 0;	Übertrag 0
0 + 1 = 1;	„ 0
1 + 0 = 1;	„ 0
1 + 1 = 0;	„ 1

Beispiel einer Subtraktion:

Die Subtraktion wird auf eine Addition zurückgeführt, und zwar wird eine Addition mit dem Komplement zur nächsthöheren Stufenzahl vorgenommen. Ein Komplement gewinnt man durch Ergänzung der Zahl bis zu ihrer nächsthöheren Potenz.

10er-Komplement

```
  7      7          (3 + 7 = 10)      111
 -3     +7                           -011
  4    1 4                            100
```

2er-Komplement

```
 111
+101     (011 + 101 = 1000)
1 100
```

nächsthöhere duale Stufenzahl

nächste Stufenzahl

```
 975     975
 -39    +961     (39 + 961 = 1000)
 936    1 936
```

Es ist zu beachten, daß die Ergänzung immer auf den Minuend zu beziehen ist. Bei der komplementären Addition muß der letzte Übertrag verhindert werden. Für die Bildung des Komplements gilt die Regel: Alle Ziffern von rechts bis zur ersten „1" einschließlich übernehmen und den Rest invertieren (umdrehen).

Beispiel einer Multiplikation:

```
16 × 6      10000 × 110
96          10000
             10000
              00000
            1100000
```

Als Rechenregeln ergeben sich hier: 0 × Zahl = 0; 1 × Zahl = Zahl

Der Vorgang der Multiplikation kann auch auf eine mehrfache Addition zurückgeführt werden. Am vorherigen Beispiel demonstriert ergibt sich:

```
10000 × 110
      |||└→0 × 10000    00000      Stellenverschiebung
      ||└──→1 × 10000   10000.
Zwischenergebnis:       100000     Addition
      |└──────→1 × 10000 10000..   Stellenverschiebung
Ergebnis:               1100000    Addition
```

Beispiel einer Division:

```
 24 : 5 = 4 Rest 4      11000 : 101 = 100
-20                    +011          Rest 100
  4                     001
                      1  00
```

Addition des Komplements des Divisors

Ebenso wie bei der Multiplikation läßt sich nachweisen, daß auch die Division durch eine fortgesetzte Addition des Komplements des Divisors darstellbar ist. Bei der Realisierung von Dual-Rechenwerken in Datenverarbeitungsanlagen genügt im allgemeinen neben einem Addierwerk je eine Vorrichtung zur Komplementbildung und zur Stellenverschiebung.

Dezimalsysteme mit binärer Zifferndarstellung

Bei Dezimalsystemen mit binärer Zifferndarstellung werden die Ziffern einer Dezimalzahl einzeln binär codiert, wobei jedoch die Struktur der Dezimalzahl erhalten bleibt. Dieses System wird auch als **binär codiertes Dezimalsystem (BCD)** bezeichnet.

Beispiel:

3	2	5	Dezimalzahl
0011	0010	0101	BCD-Zahl
		$2^3 2^2 2^1 2^0$	
		Tetrade	

In diesem Beispiel sind also die Ziffern des Dezimalsystems als Dualzahlen dargestellt. Es gibt eine Reihe von Möglichkeiten zur binären

dezimal	dual
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
17	10001
—	—
—	—
—	—

Abb. 1.19 — Zahlendarstellung

Verschlüsselung von Dezimalzahlen in Tetradenform (Aikencode, Drei-Exzeß-Code usw.), die hier nicht näher erläutert werden. Am weitesten verbreitet ist der BCD-Code, der für die Verschlüsselung der Ziffern eindeutige Wertig-

keiten (8, 4, 2, 1) verwendet. Zur Verschlüsselung im Dualsystem sind pro Dezimalziffer mindestens 4 Bitstellen erforderlich, da die höchste Dezimalziffer ja 9 sein kann (9 ≙ 1001). Für die Rechenregeln gilt, daß die Bits jeder Ziffer den Regeln des Dualsystems gehorchen, während zur Behandlung der Überträge zwischen den einzelnen Ziffern die Regeln des Dezimalsystems gelten.

Beispiel:

```
   128
 + 143
 -----
   271

0001    0010    1000
0001    0100    0011
-------------------
0010    0110    1011
        0001    1010   Korrektur
-------------------
0010    0111    0001
  2       7       1
```

Ergibt die Summe zweier BCD-Ziffern mehr als 1010 = 10, so muß ein Übertrag zur niedersten Stelle der nächsten Dezimalstelle erfolgen. Hat ein Übertrag stattgefunden, muß die erzeugende Dezimalstelle um 1010 reduziert werden.

Hieraus ist schon zu ersehen, daß der Rechenaufwand im BCD-System größer als im Dualsystem ist. Auf die Subtraktion, Multiplikation und Division wird nicht weiter eingegangen, da hier nur die grundsätzlichen Unterschiede der Darstellungsformen aufgezeigt werden sollen. Bei der Realisierung von sogenannten Dezimalrechenwerken hängt es von der Codierung der Dezimalziffern ab, inwieweit sich die Grundrechnungsarten auf die Addition zurückführen lassen. Oft werden eigene Schaltnetze für die Division, Subtraktion und Multiplikation eingesetzt.

Sind bei einer Aufgabe innerhalb einer DVA viele Berechnungen nötig, so wird man die Rechenoperationen natürlich im Dualsystem durchführen. Dies hat aber zur Folge, daß die Zahlenwerte zweimal umgeschlüsselt werden müssen, da die Ein- und Ausgabewerte ja grundsätzlich in dezimaler Form vorliegen. Bei vielen Arbeiten, vor allem im kommerziellen Bereich, ist die Zahl der Rechenoperationen jedoch gering; hierbei wäre die Umschlüsselungszeit dann größer als die eingesparte Rechenzeit. Deshalb ist bei modernen Rechenanlagen zusätzlich eine **Dezimalarithmetik** vorhanden, die in sinnvoller Weise (über das BCD-System) das für den Menschen geeignete Dezimalsystem und das vom internen Aufbau der Maschine notwendige Binärsystem miteinander verbindet.

Darstellung von Textinformation

Mit den bisher erwähnten Darstellungsformen lassen sich nur numerische Informationen darstellen. Zur Verschlüsselung von alphabetischen Daten, Sonderzeichen und Steuerzeichen wird ein erweiterter Code notwendig. Der alphanumerische Zeichenvorrat umfaßt 64 einzelne Elemente (26 Großbuchstaben, 10 Ziffern, 28 Sonderzeichen). Diese 64 Möglichkeiten lassen sich mit einem 6stelligen Binärcode ($2^6 = 64$) realisieren, wenn man dazu übergeht, die Textinformation ähnlich wie in der Lochschrift anzugeben. In der Lochschrift ist es so, daß sich die Buchstaben aus einem Ziffernteil und einem Zonenteil zusammensetzen (vgl. hierzu Abschn. 1.3.3.1.). Der Ziffernteil kann hier in der Art der BCD-Verschlüsselung angegeben, der Zonenteil aber muß in zwei zusätzlichen Bitstellen untergebracht werden.

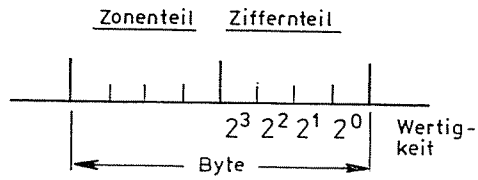
Beispiel:

Zonenteil	Ziffernteil (2^3 2^2 2^1 2^0)	Wertigkeit
00	0000	0
00	0001	1
00	1001	9
01	0001	A
01	0010	B
10	0001	I
10	0010	K
11	0010	S
11	0011	T

Lochkartencode	
Zonenlochung	Ziffernlochung
—	0
—	1
—	9
12	1
12	2
11	1
11	2
0	2
0	3

Die meisten modernen Rechenanlagen haben intern eine Bytestruktur, d.h., die kleinste adressierbare Einheit innerhalb des Kernspeichers ist eine Einheit von 8 Bits oder 1 Byte. In solchen Anlagen wird für die Verschlüsselung von alphanumerischer Information ein 8stelliger Code verwendet. Damit lassen sich $2^8 = 256$ Kombinationen realisieren. Es besteht also die Möglichkeit, neben Großbuchstaben auch Kleinbuchstaben und mehrere Steuerzeichen aufzubauen. Einer der gebräuchlichsten 8-Bit-Codes für die rechnerinterne Darstellung ist der EBCDI-Code (Extended - binary - coded - decimal - interchange-Code). Wie der Name schon sagt, handelt es sich hier um einen erweiterten BCD-Code, der nach dem gleichen Prinzip wie im

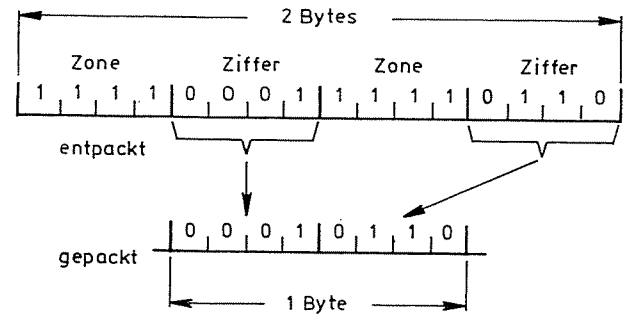
vorstehenden Beispiel aufgebaut ist. Nur stehen jetzt für den Zonenteil ebenfalls 4 Bitstellen zur Verfügung.



1111 0000 ≙ 0
1111 0001 ≙ 1
1111 1001 ≙ 9
1100 0001 ≙ A
1100 0010 ≙ B
1101 0001 ≙ I
1110 0010 ≙ S

Der EBCDI-Code ist nicht der Code der Datenträger. Bei der Eingabe von Lochkarten wird zum Beispiel die Lochschrift in den EBCDI-Code umgewandelt. Bei der Ausgabe wird er dann im Ausgabegerät wieder in Lochschrift oder Klartext umgewandelt. Nach der 2. Umwandlung stehen alle Zeichen im EBCDI-Code im Arbeitsspeicher. Sollen die Daten in einem anderen Code weiterverarbeitet werden (z.B. dual, dezimal), muß erst eine entsprechende Umschlüsselung erfolgen. Dafür stehen jedoch innerhalb des Befehlsvorrates einer Maschine geeignete Operationen zur Verfügung. Abb. 1.20 zeigt den gesamten Zeichenvorrat des EBCDI-Codes.

Aus Abb. 1.20 geht hervor, daß die Dezimalziffern auch mit 8 Bits verschlüsselt sind (wobei der Zonenteil 4 Bits belegt und der Zifferntteil entspr. dem BCD-Code ebenfalls 4 Bits beansprucht). Da hierbei der Zonenteil immer



gleich ist (um sie von den anderen Gruppen zu unterscheiden), genügen bei Ziffern zur eindeutigen Aussage nur die 4 Bits des Zifferntteils, d.h., man kann bei Ziffern den Zonenteil weglassen und nur die Zifferntteile anein-

Zifferntteil

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	NUL				PF	HT	LC	DEL								
1	0001					RES	NL	BS	IL								
2	0010					BYP	LF	EOB	PRE			SM					
3	0011					PN	RS	UC	EOT								
4	0100	SPACE										¢	.	<	(	+	|
5	0101	&										!	$	*	)	;	¬
6	0110	—	/									^	,	%	—	>	?
7	0111											:	#	@	'	=	"
8	1000		a	b	c	d	e	f	g	h	i						
9	1001		j	k	l	m	n	o	p	q	r						
A	1010			s	t	u	v	w	x	y	z						
B	1011																
C	1100		A	B	C	D	E	F	G	H	I						
D	1101		J	K	L	M	N	O	P	Q	R						
E	1110			S	T	U	V	W	X	Y	Z						
F	1111	0	1	2	3	4	5	6	7	8	9						□

Zonenteil

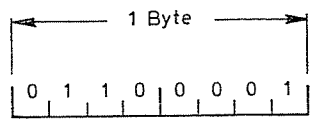
Abb. 1.20 — EBCDI-Code

Steuerzeichen

NuL	Nil (Füllzeichen)	RES	Sonderfolgenende	BYP	Sonderfolgenanfang	PN	Stanzer ein
PF	Stanzer aus	NL	Zeilenvorschub mit Wagenrücklauf	LF	Zeilenvorschub	RS	Leser stop
HT	Horizontal-Tabulator			EOB	Blockende	UC	Großbuchstaben
LC	Kleinbuchstaben	BS	Rückwärtsschritt	PRE	Bedeutungsänderung der beiden Folgezeichen	EOT	Ende der Übertragung
DEL	Löschen	IL	Leerlauf			SM	Betriebsartenänderung
						SPA	Zwischenraum

anderreihen. Diese Art der Zahlendarstellung entspricht dem BCD-System und heißt **gepackte Darstellung** (im Gegensatz zur entpackten Form).

Die gepackte Darstellung ist also für Zahlen notwendig, die in der Dezimalarithmetik verarbeitet werden sollen. Aus den bisherigen Ausführungen über die Darstellung der Informationen im Kernspeicher läßt sich bei dem im folgenden Beispiel angeschriebenen Bitmuster eines Bytes eine verschiedenartige Interpretation abgeben.



Wird das obenstehende Bitmuster als alphanumerisches Zeichen interpretiert, dann entspricht es dem Zeichen "/",denn nach der Codetabelle des EBCDI-Codes (Abb. 1.20) ergeben ein Zonenteil 0110 und ein Zifernteil 0001 die Zuordnung zum Zeichen "/". Betrachtet man dieses Bitmuster als gepackte Darstellung, dann würden hier zwei Zifernteile codiert sein; nämlich "0110" (dezimal "6") und "0001" (dezimal "1").

Unterstellt man andererseits, daß dieses Bitmuster innerhalb eines Bytes eine Dualzahl verschlüsselt, dann würde sich unter Zugrundelegung der Stellenwertigkeiten für die einzelnen Bitpositionen folgende Dezimalzahl ergeben:

Bitmuster ist 0 1 1 0 0 0 0 1

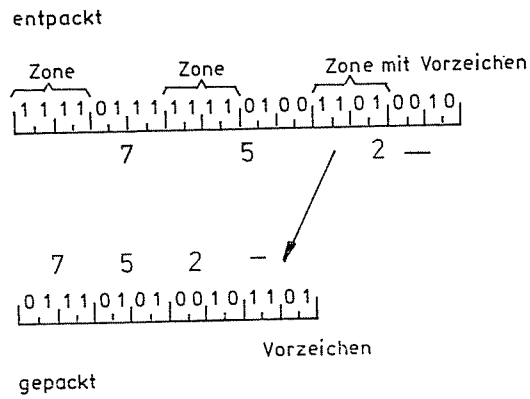
Wertigkeit ist $^{7\,6\,5\,4}$ 2 2 2 2 $^{3\,2\,1\,0}$ 2 2 2 2

Damit ergibt sich die Dezimalzahl aus $1 \times 2^6 + 1 \times 2^5 + 1 \times 2^0 = 97$.

Diese Darstellung ist die Voraussetzung für das Arbeiten mit der Dualarithmetik. Die hier aufgezeigten Mehrdeutigkeiten des Bitmusters resultieren daraus, daß man es sozusagen mit verschiedenen Schablonen (Codes) betrachtet. Das ist eine wichtige Voraussetzung für die Datenverarbeitung. Es wurde schon gesagt, daß Datenverarbeitung nicht nur den Umgang mit numerischer Information, sondern auch das Sortieren, Mischen, Ändern usw. von alphanumerischen Daten bedeutet. Z.B. kann man durch den Vergleich der EBCDIC-Verschlüsselung eines alphabetischen Zeichens mit der eines anderen Zeichens feststellen, welches Zeichen mathematisch gesehen größer ist. Das sind die Voraussetzungen, um einen Sortiervorgang für Anschriften und sonstige alphabetische Informationen einzuleiten. Für den Vergleichsvorgang muß man das Bitmuster der EBCDIC-Verschlüsselung als duale Information auffassen. Der Aufbau des EBCDI-Codes ist so geschickt gewählt, daß sich das Alphabet — was die Wertigkeit der Bitstellen betrifft — in aufsteigender Reihenfolge abbildet (vgl. hierzu Abb. 1.20). Das sind die grundlegenden Forderungen für die Sortierbarkeit und den Vergleich solcher Daten.

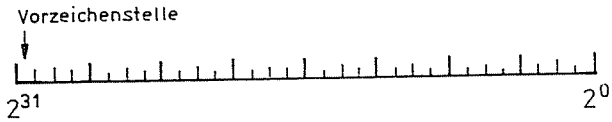
Das Vorzeichen wird im EBCDI-Code im Zonenteil der niedrigsten Ziffer vermerkt (analog zur Lochkarte — Überloch in niedrigstwertiger Stelle). Bei der gepackten Darstellung steht das Vorzeichen immer getrennt in der rechten Hälfte des letzten Bytes und wird immer in dem am weitesten rechts stehenden Halbbyte mitgeführt.

Beispiel:



Im Dualsystem wird das Vorzeichen an der am weitesten links stehenden Bitstelle innerhalb der Dualzahlen angenommen. Dabei entspricht die "0" = "+" und "1" = "—".

Beispiel: In einem Byte läßt sich dual nur die Zahl 255 darstellen. Durch Aneinanderreihung von mehreren Bytes lassen sich größere Zahlenwerte dual darstellen. Häufig sind eine feste Anzahl von Stellen pro Zahl vereinbart (32 Stellen = Länge eines Registers = 4 × 1 Byte). Es ist mathematisch beweisbar, daß die (vgl. unter Dualsystem S. 22) aufgestellten Rechenregeln auch für die Einbeziehung des Vorzeichens, mit genannter Definition, gelten.



Darstellung der Befehle

Der grundsätzliche Befehlsaufbau sowie die Verschlüsselung der Adressenangaben wurden bereits im Abschn. 1.3.2. behandelt. Anhand eines Maschinenbefehls einer DVA (Siemens 4004) soll nochmals auf die Darstellung hingewiesen werden. Hier handelt es sich um einen Additionsbefehl mit zwei Adressenangaben.

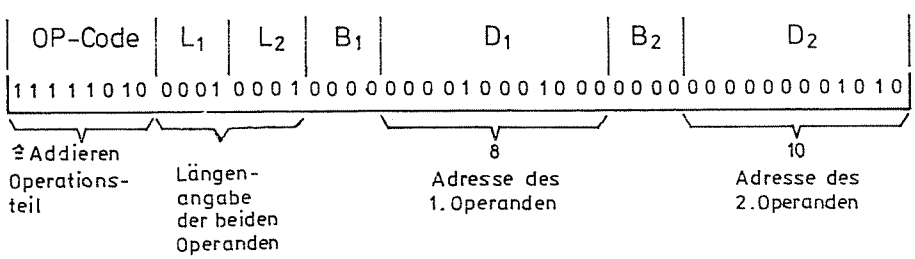


Abb. 1.21 — Maschinenbefehl

Für den Operationscode stehen 8 Bits = 1 Byte zur Verfügung. Theoretisch könnten also 256 Befehle verschlüsselt werden. L_1 enthält die Längenangabe für den 1. Operanden, L_2 die für den 2. Operanden. Dafür stehen jeweils 4 Bits zur Verfügung; die maximale Längenangabe kann also 16 sein, denn die Angabe erfolgt im Dualsystem. Für die Adresse stehen zunächst nur 12 Bits bereit. Die Adressenangabe enthält die Nummer (Fachnummer im Regal) des 1. Bytes des Operanden. Da die Nummernangaben ebenfalls im Dualzahlensystem erfolgen, können mit 12 Bits aber nur 4096 Adressen angesprochen werden (2^{12} = 4096). Arbeitsspeicher haben aber bis zu 512 Kilo Byte und mehr an adressierbarem Volumen. Aus diesem Grunde wird die Adressierung in eine Basisadresse (B_1, B_2) und eine Distanzadresse (D_1, D_2) zerlegt. Die 4 Bits unter B_1 und B_2 geben die Nummer eines Registers an, in dem die zugehörige Basisadresse zu finden ist. Es können 16 Register angesprochen werden. Innerhalb des Registers stehen 32 Bits zur Verfügung, so daß mit der Summe aus Basis- und Distanzadresse jeder Punkt eines Kernspeichers erreicht werden kann (2^{32} = 4 294 967 296).

Dieses Verfahren der Adressierung wird auch **indirekte Adressierung** genannt. Sie bringt den Vorteil, daß der Adreßteil eines Befehls kurz gehalten werden kann. Der zweite Vorteil ergibt sich dadurch, daß schon allein durch Änderungen in der Basisadresse (im Register) Verschiebungen von Daten und Programm aufgefangen werden können, ohne den Adreßteil der Befehle ändern zu müssen. Die echte Adresse wird erst im Steuerwerk aus dem Inhalt des Basisadreßregisters und der Nummer der Distanzadresse gebildet. Bei einer **direkten Adressierung** kann die Adresse aus dem Adreßteil des Befehls direkt übernommen werden. Bei der **indizierten Adressierung** wird die Adresse im Steuerwerk mit Angaben von Indexregistern gebildet. Der in Abb. 1.21 dargestellte Maschinenbefehl kann also unter der Annahme, daß im Basisregister 0 die Adresse 90 steht, wie folgt interpretiert werden:

Addiere den Inhalt der Speicherstellen 100 (Basis = 90; Distanz = 10) und 101 (1. Byte des 2. Operanden auf Stelle 100, Länge 2 — ergibt Stellen 100 und 101 —) zu dem Inhalt der Speicherstellen 98 und 99 (1 Byte → B_1 = 90, D_1 = 8, L_1 = 1) und speichere die Summe ebenfalls in die Speicherstellen 98 und 99.

(Anmerkung: Länge von 1 entspricht einem $L_{1,2}$ = 000, Länge von 16 einem $L_{1,2}$ = 1111 (15); hier ist L_1 = 1, also Länge 2). Der Befehl selbst belegt 6 Byte im Arbeitsspeicher.

Gültigkeitsprüfung

In einer DVA werden Daten zwischen den Elementen und innerhalb der Elemente übertragen. Zur Erhöhung der Übertragungssicherheit wurde ein **Prüfbit** als redundantes Signal eingeführt. Dieses Prüfbit wird durch die Anlage selbst erzeugt und entsteht bei der Übernahme von Daten in den Arbeitsspeicher, d.h., speichertechnisch gesehen besteht ein Byte aus 9 Bits (für den Programmierer sind allerdings nur 8 Bits zugänglich). Dieses Prüfbit wird beim Speichern der Information zugesetzt und beim Lesen überprüft. Es ergänzt die Anzahl der „1" innerhalb des Bytes auf einen ungeraden Wert. Dieses sogenannte **Paritycheck-Verfahren** gestattet die Erkennung von Fehlern (bei jeder Übertragung werden die „1" gezählt und verglichen) und wird am häufigsten in elektronischen Datenverarbeitungsanlagen angewendet. Hierbei handelt es sich nur um die Sicherheit innerhalb der Anlage.

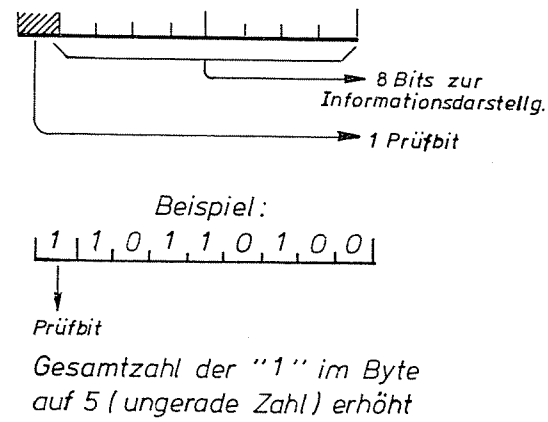


Abb. 1.22 — Elektronisches Speicherelement

Bei den Codeumwandlungen der Ein-Ausgabegeräte wird eine Kontrolle auf volle Übereinstimmung der Zeichen nicht durch die Methode des redundanten Zusatzes, sondern durch **Rückkopplung** durchgeführt. Innerhalb der Schaltnetzwerke werden die decodierten Zeichen wieder in den ursprünglichen Code umgewandelt und durch Vergleich überprüft. Bei der Ein-Ausgabe erfolgt eine **Vollprüfung** der Zeichen, bei der inneren Verarbeitung dagegen eine **Gültigkeitsprüfung.**

Das Sedezimalsystem

Für das Testen von Programmen ist oftmals ein Kernspeicherauszug notwendig. Das Ausdrukken in binärer Darstellung wäre sehr unübersichtlich und schlecht. Für die Ausgabe solcher Auszüge wird deshalb meist eine Art Kurzschrift, nämlich das **Sedezimal- oder Hexadezimalsystem** verwendet. Dabei werden immer 4 Bits zusammengefaßt und jeder der 16(2^4) Vier-

Bit-Anordnungen wird eine Ziffer zugeordnet. Damit ergibt sich ein System mit 16 Ziffern; also ein Zahlensystem zur Basis 16. Für die 16 Ziffern des Sedezimalsystems werden die Dezimalziffern 0—9 und die Großbuchstaben A—F benutzt; d.h., in dieser Verwendung handelt es sich nicht mehr um Buchstaben, sondern um Ziffern des Sedezimalsystems. Obwohl das Sedezimalsystem normalerweise für interne Rechenzwecke nicht genutzt wird, sollen die Regeln kurz zusammengefaßt werden.

Basis: 16
Gültige Ziffern: 0, 1, ... 9, A, B, C, D, E, F,
Schreibweise: z.B.: 0 A 5

$0 * 16^2 + A * 16^1 + 5 * 16^0$

Stellenwert	=	Ziffernwert	*	Stufenzahl
0	=	0	*	256 (16^2)
160	=	$10_{(A)}$	*	16 (16^1)
5	=	5	*	1 (16^0)
Zahlenwert: 165				

Abb. 1.23 — Sedezimalsystem

dezimal	dual	sedezimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Abb. 1.24 — Darstellungsform

Die Rechenregeln gelten im Sedezimalsystem ebenso wie im Dual- und Dezimalsystem. Hier soll deshalb auf die Umrechnung hingewiesen werden:

dual (0001 1101) 2
sedezimal (1 D) 16

dezimal ⇄ sedezimal

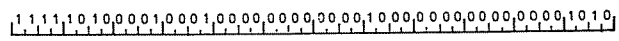
$(29)_{10} : 16 = 1$ Rest D
$1 : 16 = 0$ Rest 1

$(1\ D)_{16}$

Ziffernwert * Stufenzahl = Stellenwert
$D \rightarrow 16_0 = 13$
$16_1 = 16$
$(29)_{10}$

Die Angabe von Bitmustern wird in der 3. Generation der Rechenanlage im sedezimalen Code vorgenommen. Neben der besseren Verständlichkeit wird dadurch, daß eine Vierergruppe durch eine Sedezimalziffer ersetzt wird, bei der Darstellung an Länge der Zahl gespart (1 : 4).

Beispiel: Der in Abb. 1.21 gezeigte Maschinenbefehl wird nach der Aufteilung in Vierergruppen wie folgt umgewandelt:



F A 1 1 0 0 0 8 0 0 0 A

Bitmuster sedezimal abgekürzt

Gleitkommadarstellung

Bei den bisher behandelten Zahlendarstellungen ist die Rechnung mit der Kommastelle nicht berücksichtigt worden. Die Zahlen werden als ganze, ungebrochene Zahlen ohne Komma behandelt. Bei der Addition von Zahlen verschiedener Größenordnungen ist es Aufgabe des Programmierers, die Operanden so gegeneinander zu verschieben, daß durch die Addition entsprechender Stellenwerte eine korrekte Summe gebildet wird.

Das Rechnen, das selbsttätig auf die Stellung des Kommas keine Rücksicht nimmt, bezeichnet man auch als **Festkommaarithmetik.** Dabei wird für jede Zahl eine bestimmte Anzahl von Stellen festgesetzt, die nicht überschritten werden kann. Um aber auf einfache Weise die Stellung des Punktes zu erkennen und dabei auch Brüche und sehr große ganze Zahlen in der gleichen Form ausdrücken zu können, mußte eine neue Art der Zahlendarstellung eingeführt werden — die **Gleitkommarechnung.** Das ist eine halblogarithmische Zahlendarstellung mit einer Mantisse und einem Exponenten. In der allgemeinen Form wird eine Zahl angegeben mit: $A^n * B_1$, wobei für A ein fester Wert gilt, der z.B. 2, 16 oder 10 sein kann. Die 10 wird bei intern dezimal arbeitenden Maschinen verwendet; sie ist für die Erläuterungen am einfachsten und gilt für die nachstehenden Ausführungen. Eine Zahl in der Form 427,93456 läßt sich auch wie folgt schreiben: 0,42793456 x 10^3 oder 0,042793456 x 10^4. Durch die Angabe des Dezimalbruches und des Zehnerexponenten ist eine Zahl beliebiger Grö-

ßenordnung eindeutig definiert, wobei der Dezimalbruch als Mantisse und der Zehnerexponent auch als Charakteristik bezeichnet wird. Um immer von gleichen Voraussetzungen ausgehen zu können, wird eine **normalisierte Gleitkommadarstellung** notwendig, bei der die Mantisse immer mit absolutem Betrag $\geqq 0{,}1$ . . . . . sein muß.

Beispiel: $0{,}042793456 \times 10^4$ nicht normalisiert
$0{,}42793456 \times 10^3$ normalisierte Darstellung

Die Mantisse wird um die Anzahl der führenden Nullen nach links verschoben und der Exponent entsprechend berichtigt. Wenn diese Voraussetzungen gegeben sind, braucht die Abspeicherung nur in einer verkürzten Form zu erfolgen.

4 2 7 9 3 4 5 6	0 3
Mantisse	Charakteristik

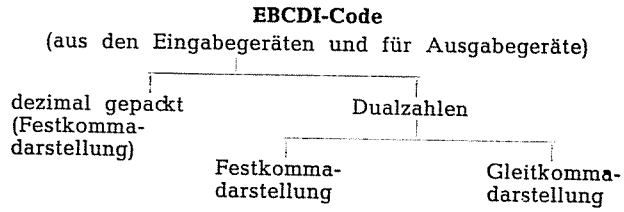
Charakteristik = der ganzzahlige Exponent der Zahl 10

Die Mantisse und der Exponent können je ein spezielles Vorzeichen annehmen.

Beim Arbeiten mit Gleitkommazahlen wird die Stellung des Kommas in einer Zahl automatisch berücksichtigt. Vor einer Addition muß z.B. durch Verschiebungen der Mantisse der Exponent angeglichen werden. Erst danach können die Mantissen addiert werden. Die Multiplikation von zwei Gleitkommazahlen wird durch Multiplikation der Mantissen und Addition der Exponenten erreicht.

Die Beispiele zeigen, daß die Länge der Mantisse einen entscheidenden Einfluß auf die Genauigkeit hat. Durch die automatische Normalisierung können die rechts nachkommenden Nullen Genauigkeiten vortäuschen, die gar nicht vorhanden sind.

Bei den modernen Rechnern der 3. Generation kann man zusammenfassend von folgenden Möglichkeiten der Zahlendarstellung ausgehen:



Für die Umwandlung der Daten in die verschiedensten Darstellungen (auch untereinander) stehen besondere **Konvertierungsbefehle** zur Verfügung. Die Gleitkommaarithmetik kann sowohl über programmierte Routinen als auch durch eine festverdrahtete Logik realisiert werden.

Am Beispiel der Siemens 4004 wird für die Gleitkommaarithmetik folgende Anleitung gegeben:

Darstellung: $(\text{Wert})_{16} = (\text{Mantisse})_{16} * {}_{16}(\text{Exponent})_{10}$

(Bei normalisierter Darstellung ist der Zahlenvorrat $16^{-65} \leqq \text{Wert} < 16^{63}$. Der Zahlenwert für die Festkommarechnung beträgt $-2^{31} \leqq \text{Wert} < 2^{31} - 1$).

Diese Zahlendarstellung wird meist beim Programmieren mit problemorientierten Program-

Beispiel: $0{,}2400 \cdot 10^3$
$0{,}3400 \cdot 10^4$

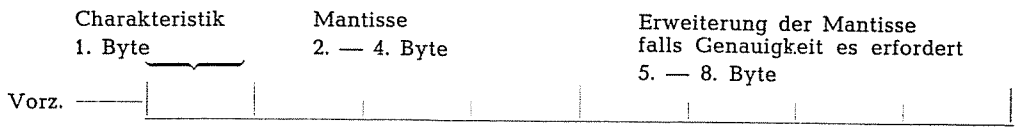
Addition

	2400	03	
	3400	04	Angleichung der Exponenten
	0240	04	
+	3400	04	
	3640	04	

$100{,}7 = 0{,}1007 \cdot 10^3$
$6{,}384 = 0{,}6384 \cdot 10^1$

Subtraktion

	1007	03	
	6384	01	Angleichung der Exponenten
	1007	03	
−	0063	03	
	0944	03	Ergebnis muß normalisiert werden
	9440	02	



miersprachen eingesetzt. Für das Ausschreiben von Zahlen in Gleitkommadarstellung werden verschiedene Formate bereitgehalten.

1.3.4. Grundlagen der Programmierung

Hier soll der Weg von der Instruktion zum Programm beschrieben werden. Kenntnisse über die Funktion und Arbeitsweise der Zentraleinheit und der Peripherie sowie die interne Datendarstellung und den Befehlsaufbau wurden in den vorhergehenden Abschnitten vermittelt. Ebenso wurden die grundsätzlichen Arten von Befehlen angedeutet. Die Vermittlung des genauen Befehlsspektrums ist eigentlich Sache eines maschinenorientierten Programmierkursus für eine spezielle Anlage. In diesem Rahmen soll versucht werden, eine allgemein gültige Aussage zu machen.

Ohne Anweisungen des Benutzers ist eine elektronische Datenverarbeitungsanlage nicht arbeitsfähig; jedoch bringt sie gewisse Fähigkeiten mit, die technisch realisiert sind. Der Benutzer muß diese kennen, um mit der Anlage arbeiten zu können. Das Programm beschreibt ein darauf abgestimmtes Lösungsverfahren. Die Gesamtheit der zur Lösung eines Problems notwendigen Instruktionen bildet ein Programm. Die Instruktionen (Befehle) ergeben sich aus dem Befehlsvorrat für die eingesetzte Datenverarbeitungsanlage.

Zur Durchführung der Lösung einer Aufgabe müssen eine Datenverarbeitungsanlage und ein Programm vorhanden sein. Im allgemeinen werden diese beiden Komponenten auch als **Hardware** und **Software** bezeichnet. Hardware umfaßt sämtliche technischen Teile einer DVA (Schränke, Gestelle, Mechanik, Verdrahtung, Bauelemente). Software umschreibt alles, was gedanklich erarbeitet und festgehalten wurde (Unterlagen, Verfahrensbeschreibungen, Programme, Programmbeschreibungen).

1.3.4.1. Weg von der Stellung der Aufgabe bis zum Programmablauf

Bei einem Problem, das mit Hilfe einer Datenverarbeitungsanlage bearbeitet werden soll, geht man immer von einer gegebenen Ausgangssituation aus und versucht, ein bestimmtes Ziel zu erreichen. Programmieren ist nun nichts anderes als das Überlegen, wie man von der Ausgangssituation zu dem angestrebten Ziel kommt und dann das schriftliche Festhalten dieses Weges. Das Programm beschreibt diesen Weg in allen Einzelheiten unter Berücksichtigung aller Möglichkeiten und Bedingungen. Dabei stehen für eine Datenverarbeitungsanlage eben nur eine gewisse Menge an Befehlen zur Verfügung, die in der Befehlsliste katalogmäßig zusammengestellt sind. Die Programmierung ist die entscheidende Arbeit für die Lösung einer Aufgabe durch eine Datenverarbeitungsanlage. Der Weg von der Aufgabenstellung bis zum fertigen Programm muß sorgfältig vorbereitet werden. Im allgemeinen läßt sich für den Lösungsweg folgende Einteilung festlegen:

1. **Formulierung des Problems** (Aufgabenstellung)
2. **Analyse des Problems** (mathematische Formulierung des Problems zeigt Weg der Verarbeitung, Festlegen der Ein-Ausgabedaten, Speicherbereiche angeben, Schnittstellen zu vorhandenen oder anderen Problemen und Programmen, Datenflußplan erstellen)
3. **Programmerstellung** (logische Auflösung des Problems in Einzelschritte → Programmablaufplan, Primärprogramme codieren, Maschinenprogramm erzeugen, Dokumentation)
4. **Ausprüfung des Programms — Programmtest**
5. **Programmablauf — Ausführung des Programms.**

Diese einzelnen Schritte müssen **nacheinander** ausgeführt werden, wenn eine DVA wirkungsvoll arbeiten soll. Jedoch sind die Grenzen zwischen den einzelnen Schritten oftmals fließend. Die Problemanalyse nimmt bei kommerziellen Problemen die wichtigste Stellung ein, weil es sich in diesem Bereich meist um integrierte Datenverarbeitungssysteme handelt. (Diese Problematik wird im Abschnitt 1.3.6. gesondert behandelt.).

Wir wollen jedoch zunächst an einem einfachen Beispiel (Berechnen des Inhalts einer Kugel) die angesprochenen Schritte in detaillierter Form vorstellen. Dieses Beispiel stammt aus dem technisch-wissenschaftlichen Bereich, wo in der Regel ein Mann alle Schritte nacheinander ausführt, während sich im kommerziellen Bereich verschiedene Kräftegruppen mit der Analyse (Systemanalytiker) und der Programmierung befassen.

Formulierung des Problems

Zunächst wird die Aufgabe in der Umgangssprache genau umschrieben und schriftlich festgelegt. Dabei sind alle Einzelheiten und Bedingungen festzuhalten. Dieser Schritt ist besonders bei der maschinellen Bearbeitung von Aufgaben von großer Bedeutung. Da es sich meist um Einzelprobleme aus einer Gesamtlösung handelt, sind klare Abgrenzungen und Definitionen not-

wendig. Bei größeren Problemstellungen — z.B. Inventur — sind alle notwendigen Abmachungen wie Einzelpreis des Artikels, Stückzahl, Summe aller Gesamtwerte pro Artikel, Wert des Warenlagers usw. anzugeben.

In unserem Beispiel soll für beliebig viele Durchmesserangaben von Kugeln das zugehörige Volumen ermittelt werden. Die Eingabe erfolgt per Lochkarten; jeweils eine Lochkarte enthält eine Durchmesserangabe. Die Ausgabe der Werte soll auf einem Schnelldrucker erfolgen und in einer Zeile den Durchmesser und das zugehörige Volumen enthalten.

Problemanalyse

Nach der vollständigen Festlegung der Aufgabe muß herausgefunden werden, ob dieses Problem überhaupt mit Hilfe der elektronischen Datenverarbeitung lösbar ist. Dazu muß die formulierte Aufgabe in einer mathematischen Formel ausdrückbar sein. Das ist naturgemäß für technisch-wissenschaftliche Aufgabenstellungen von besonderer Bedeutung. Weiter muß untersucht werden, ob die gewählten mathematischen Formeln sich auch für das numerische Rechnen mit einer Datenverarbeitungsanlage eignen. Bei komplexen Problemen muß die Aufgabe evtl. durch Idealisierungen und Vernachlässigungen vereinfacht werden, damit sie überhaupt einer mathematischen Behandlung zugänglich wird. Bei der Wahl des numerischen Lösungsverfahrens sollte auch die Art der zu benutzenden Anlage mit in Betracht gezogen werden, weil dadurch z.B. die Verwendung von gewissen mathematischen Näherungsverfahren notwendig wird.

Bei kommerziellen Problemen müssen bei diesem Schritt meist die Schnittstellenbedingungen zu dem integrierten Gesamtsystem angegeben werden, so z.B. Datenaufrufe, Mehrfachzugriff zu Informationen, gemeinsame Formulare, bereits vorhandene Unterroutinen, Aufrufe von bestehenden Programmen usw.

Für das gewählte Beispiel gilt für die Berechnung des Volumens die Formel $V = \pi/6 * d^3$. Besondere Schwierigkeiten ergeben sich nicht; die Aufgabe ist maschinell lösbar.

Programmerstellung

Bei der Programmerstellung sind mehrere Arbeitsgänge zu erledigen. Zunächst muß festgelegt werden, was als Endergebnis gewünscht wird und wie es ausgegeben werden soll. Im einzelnen heißt das:

Was soll als Ergebnis ausgedruckt werden?

Wo soll das Ergebnis erscheinen?

Wie soll das Ergebnis angeordnet sein (Format)?

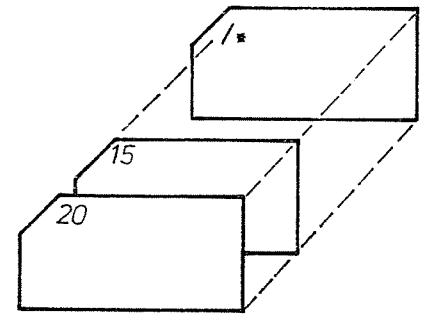
Die gleichen Überlegungen gelten für die Eingabedaten. Es wird zusammengetragen, welche Eingabedaten nötig sind und in welcher maschinenlesbaren Form sie vorliegen sollen. Hat man größere Arbeiten mit vielen Daten, ist es sinnvoll, die Speicherbereiche für Daten zu bestimmen. Damit wird festgelegt

a) wo die Daten „hingelesen" werden sollen (wo die Eingabeoperation die gelesenen Daten im Arbeitsspeicher ablegt),

b) wo die Daten (evtl. auf welchen Speichergeräten) während der Verarbeitung zwischengespeichert werden und

c) wo im Arbeitsspeicher die Ergebnisse für die Ausgabeoperation zur Verfügung stehen.

Für das gewählte Beispiel wird angenommen, daß sich die Eingabewerte auf Lochkarten befinden. Für jede neue Durchmesserangabe wird eine eigene Lochkarte erstellt. Der Zahlenwert ist in den Spalten 1 und 2 als ganze Zahl



Eingabekarten

Abb. 1.25 — Eingabekarten

abgelocht. Das Programm zur Berechnung des Volumens soll nur ganzzahlige Werte verarbeiten. Da eine unbestimmte Zahl von Karten verarbeitet wird, muß als Kennung, daß die letzte Karte eingelesen wurde, eine besondere Karte mit einer vereinbarten Lochung (in Abb. 1.25: /*) folgen. Die Ausgabe auf dem Schnelldrucker soll so aussehen, daß nach einer Überschriftzeile jeweils Durchmesser und Volumen zu schreiben sind:

VOLUMENBERECHNUNG FUER EINE KUGEL

DER INHALT EINER KUGEL MIT xx CM DURCHMESSER IST xxxxxx CCM.

Der nächste Schritt ist die Aufstellung von Datenfluß- und Programmablaufplänen. Der Programmablaufplan ist die graphische Darstellung der Behandlungsweise, mit der die Daten inner-

halb eines Maschinensystems verarbeitet werden. Der **Programmablaufplan** zeigt, wie eine Arbeit getan werden muß. Der **Datenflußplan** dagegen zeigt, welche Arbeiten getan werden müssen. Der Datenflußplan ist die Darstellung des Gesamtsystems, in dem die Ausgangsdaten in die gewünschten Ergebnisse (Zieldaten) umgewandelt werden. Zur Darstellung dieser Pläne dienen vereinheitlichte Symbole, auf die im nächsten Abschnitt näher eingegangen wird.

Bevor der Programmablaufplan gezeichnet werden kann, muß das Problem in Einzelschritte zerlegt sein, die von der Datenverarbeitungsanlage ausgeführt werden können. Um diese Detaillierung zu demonstrieren, folgt eine Zerlegung an einem zunächst von der maschinellen Datenverarbeitung nicht berührten Gebiet.

Stellen Sie sich vor, ein Schüler, der nur die vier Grundrechnungsarten beherrscht, soll als Hausaufgabe den Inhalt einer Kugel von 20 cm Durchmesser berechnen. Da er das nicht kann, ruft er Sie zu Hilfe. Sie wissen oder schauen nach, daß die Formel für den Inhalt einer Kugel $V = 4/3 * \pi \cdot r^3 = 1/6 * \pi * d^3$ ist. Sie werden nun nicht versuchen, die logischen Fähigkeiten des Schülers aufzubessern, sondern die bereits vorhandenen ausnutzen. D.h., die Aufgabe muß den logischen Fähigkeiten des Schülers angepaßt werden, etwa wie folgt:

a) dividiere 3,14 durch 6 ($\triangleq \pi/6$),

b) schreibe das Ergebnis der Division unter a) auf (Speichern),

c) schreibe die Zahl 20 auf (entspricht dem Einlesen des Durchmessers),

d) multipliziere die unter c) aufgeschriebene Zahl mit sich selbst ($\triangleq d \cdot d = d^2$),

e) schreibe das Ergebnis der Operation unter d) auf,

f) multipliziere die Zahlen unter c) und e) miteinander ($\triangleq d^3 = d^2 \cdot d$),

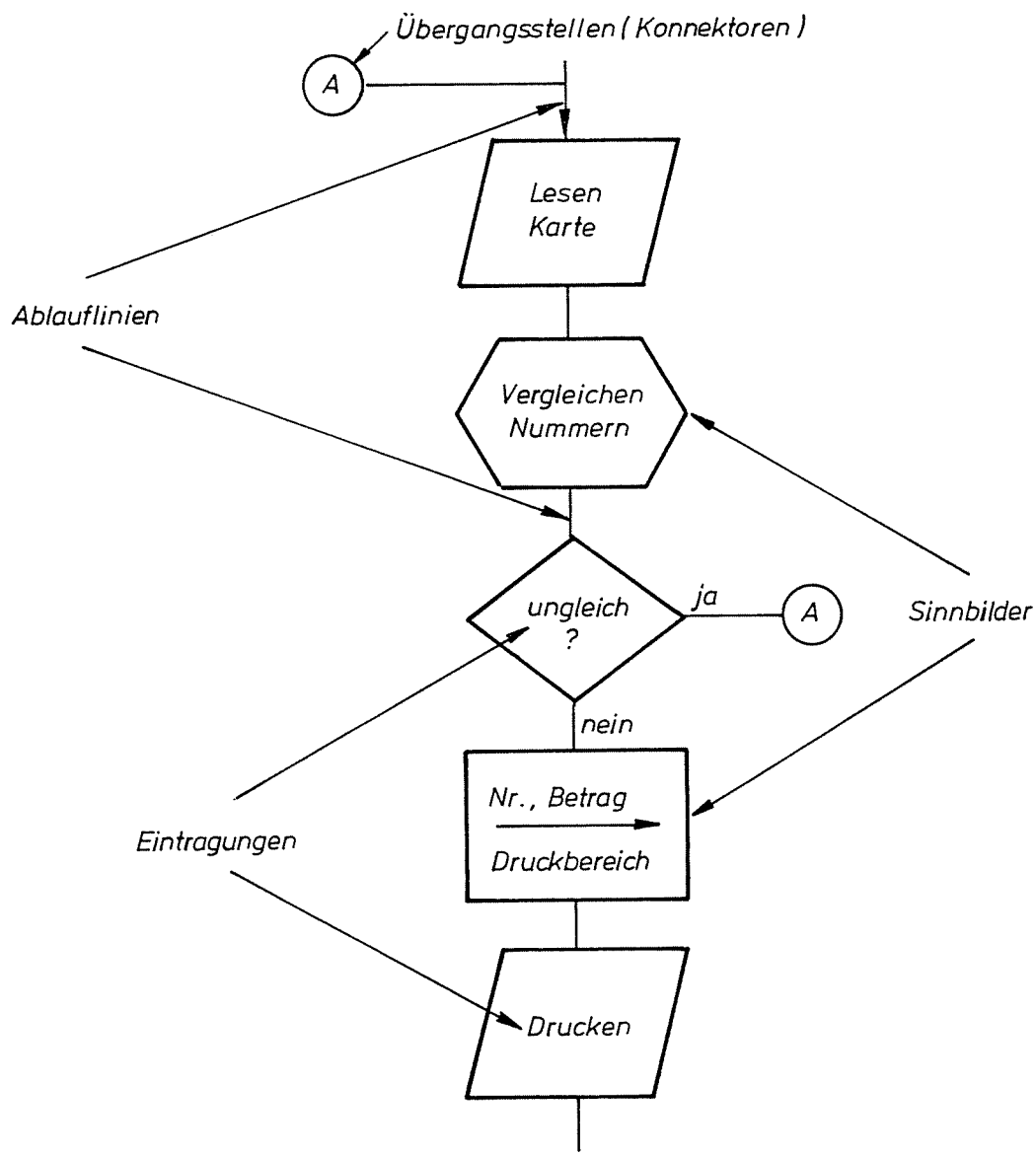


Abb. 1.26 — Terminologie des Programmablaufplans

g) schreibe das Ergebnis der Multiplikation unter f) auf,

h) multipliziere die Zahlen unter g) und b) miteinander ($\triangleq \pi/6 * d^3$) und

i) schreibe das Ergebnis der Multiplikation unter h) auf.

Die Zahl unter i) entspricht dem Volumen einer Kugel von 20 cm Durchmesser.

Es ist unschwer zu erkennen, daß der Schüler hier eine DVA symbolisiert. Die elektronische Datenverarbeitungsanlage kann nur entsprechend ihren logischen Fähigkeiten rechnen, und sie weiß auch nicht, was sie rechnet. Die Interpretation der Zahlenwerte und Ergebnisse bleibt dem Programm und damit dem Programmierer überlassen. So gesehen bedeutet Programmieren das Auflösen eines Problems in Einzelschritte, zu deren Durchführung eine gegebene Datenverarbeitungsanlage aufgrund ihres Befehlvorrats in der Lage ist. Das Lösungsverfahren muß den

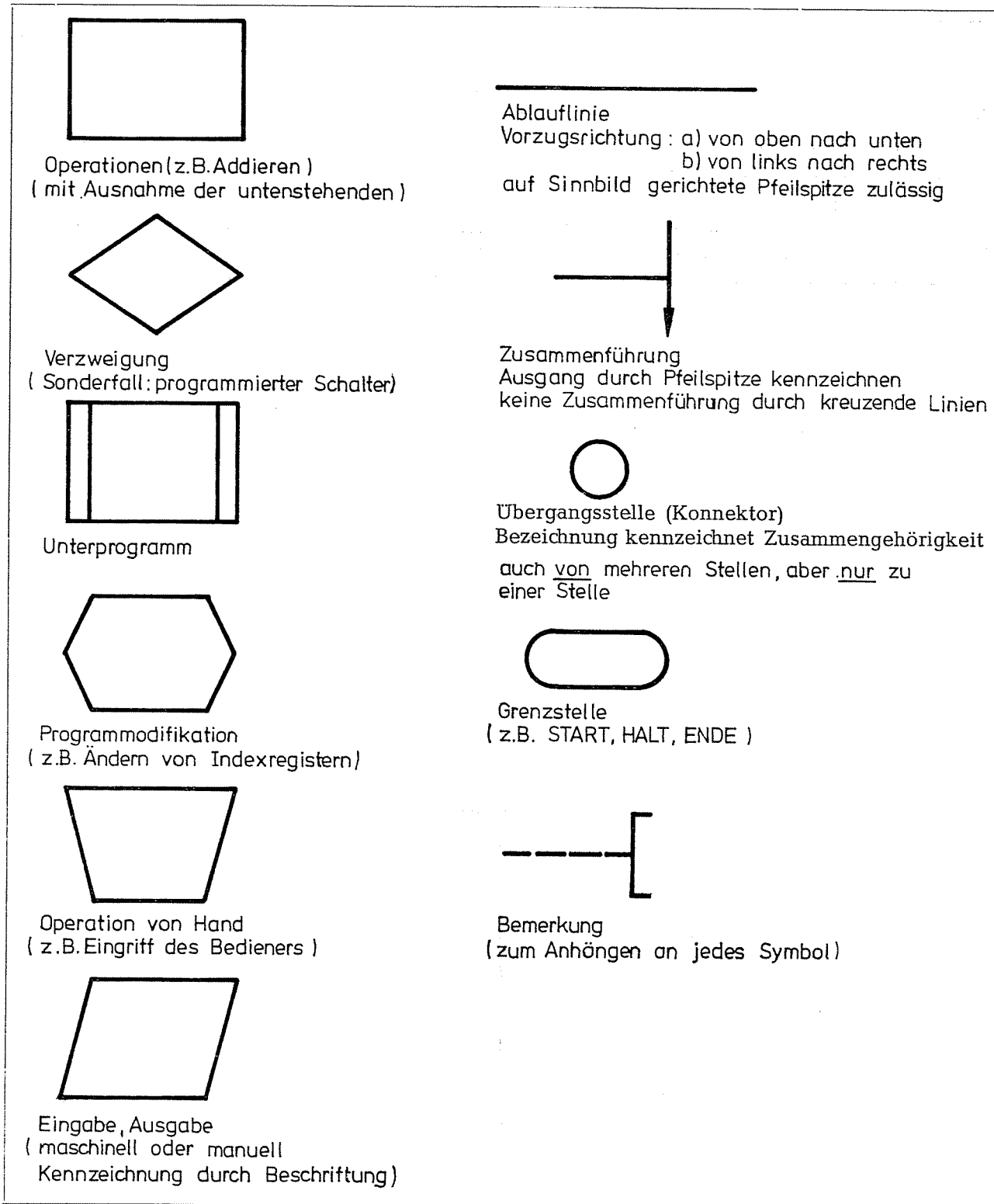


Abb. 1.27 — Symbole für Programmablaufpläne

logischen Fähigkeiten der Maschine angepaßt werden; es sind immer numerische Lösungsverfahren. In dem hier gezeigten Beispiel übernehmen Sie praktisch die Funktion eines Übersetzers, der die Aufgabenstellung aus der Logik der Fachsprache in die Logik der Maschinensprache überträgt. Dazu muß man selbstverständlich die Logik beider Sprachen beherrschen.

Bevor für das gewählte Beispiel der Datenflußplan und der Programmablaufplan niedergeschrieben werden können, muß zunächst die Symbolik für diese Darstellung erläutert werden.

Symbole für Programmablaufpläne

(Vgl. hierzu Abb. 1.26, 1.27)

Die Symbole für Programmablauf- und Datenflußpläne sind vereinheitlicht und in DIN 66001 zusammengestellt. Es gibt im wesentlichen Symbole für Operationen, für die Ein- und Ausgabe und ein Symbol „Ablauflinie". Wegen ihrer Übersichtlichkeit sind Programmablaufpläne ein allgemein anerkanntes Darstellungsmittel. Für das Zeichnen stehen Schablonen mit allen Symbolen zur Verfügung. Zur Beschreibung von umfangreichen Programmen werden oft mehrere Stufen von Ablaufplänen erstellt, wobei die letzte Stufe in allen Einzelheiten zu erstellen ist. Die Symbole eines Programmablaufplans können durch Eintragungen (Operationsangaben) ergänzt werden. Anstelle von längeren Texten können dazu auch Kurzeintragungen verwendet werden (vgl. hierzu Abb. 1.29). Das Symbol „Bemerkung" kann weiteren Text enthalten bzw. darauf hinweisen (vgl. hierzu Abb. 1.27).

Als Beispiel für die Anwendung des Ablaufplans soll eine Summe gebildet werden, die aus Beträgen — in Lochkarten enthalten — entsteht.

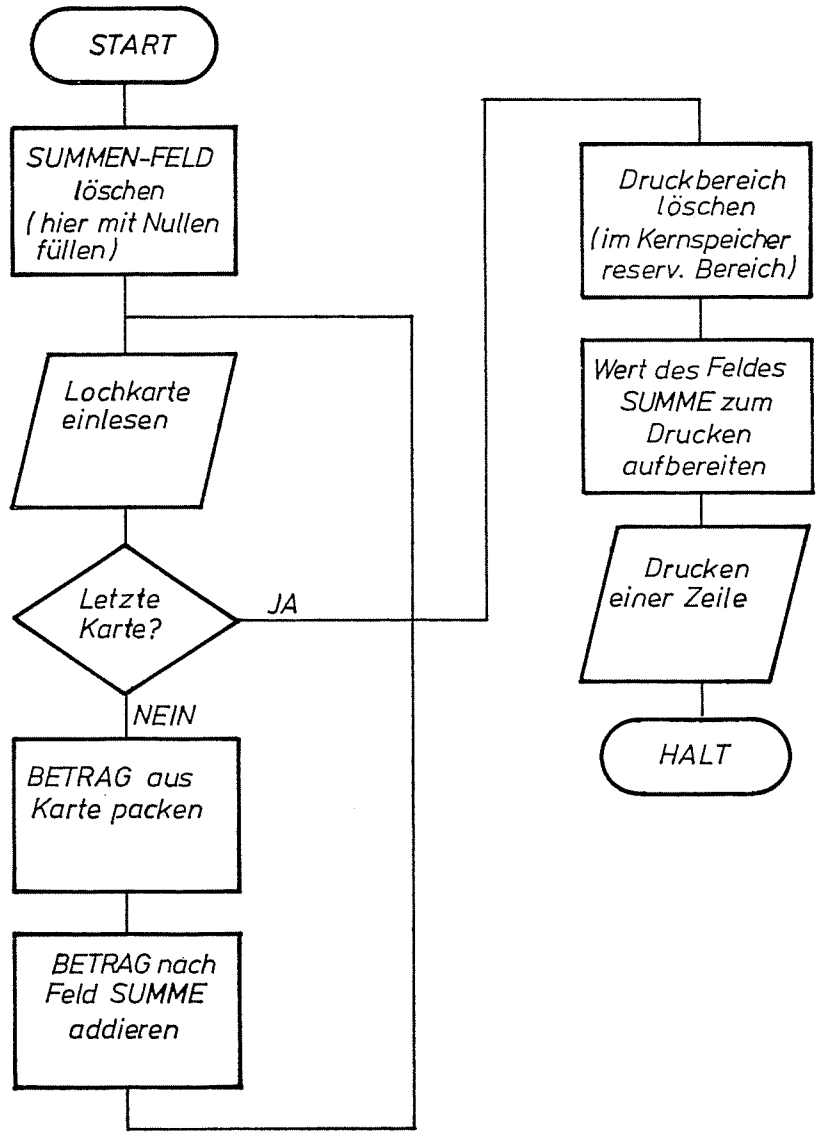


Abb. 1.28 — Beispiel für einen Programmablaufplan

Dabei muß der eingelesene EBCDI-Code in gepackte Darstellung gebracht werden, damit gerechnet werden kann. Für die Ausgabe auf einen Schnelldrucker muß wieder entpackt werden. Die Summe ist nach Erkennen der letzten Lochkarte über den Schnelldrucker auszugeben.

Die Programmodifikation in Abb. 1.27 steht für eine arithmetische, logische oder Übertragungsoperation. Mit der Programmodifikation werden Entscheidungen vorbereitet und Instruktionen oder Konstanten geändert. Sie selbst ist jedoch keine Entscheidung und erstellt keine Ausgabedaten. Das erste Symbol in Abb. 1.27 steht für Verarbeitungsfunktionen; dies sind arithmetische und/oder Übertragungsoperationen, die dem Erstellen von Ausgabedaten dienen. An Kurzeintragungen in Symbolen sind möglich:

Zeichen	Bedeutung
⟷	„Vergleiche"
⟶	„übertrage . . . . nach . . . . "
=	„ergibt sich aus" bzw. „wird"
+	„addiere" bzw. „plus"
—	„subtrahiere" bzw. „minus"
* (für ·)	„multipliziere mit . . . . " bzw. „mal"
/ (für :)	„dividiere durch . . . . "

Abb. 1.29 — Kurzeintragungen

Symbole für Datenflußpläne

Es gibt hier im wesentlichen Symbole für das Bearbeiten und für die Datenträger sowie das Symbol „Flußlinie". Die Symbole für die Datenträger bezeichnen sowohl den Datenträger als auch — in Verbindung mit einer Flußlinie — das Eingeben, Umspeichern oder Ausgeben der Daten (vgl. hierzu Abb. 1.31).

Als Beispiel für die Darstellung eines Datenflußplans dient eine Auftragsabwicklung. Die Aufträge werden abgelocht und dann auf ein Magnetband übernommen, das nach Kunden-Nr. und Artikel-Nr. sortiert wird. Danach wird im Zusammenhang mit dem Kundenstammband die Rechnungsschreibung vorgenommen. Dabei werden auch Daten über den Umsatz, den Bestand und die Disposition gewonnen (vgl. hierzu Abb. 1.32).

Der Programmablaufplan wird aufgrund der Zerlegung in Einzelschritte aufgestellt, wobei auch auf die Zahlendarstellung Rücksicht genommen werden muß. Weiter ist zu ersehen, daß ein Programm aus Instruktionen und Konstanten besteht. Für die Berechnung muß der Wert $\pi/6$ gebildet werden. Dies ist ein Wert, der für die Lösung der Aufgabe benötigt wird, aber nicht über die Eingabe kommt. Solche Konstanten werden mit den Befehlen gleichzeitig in den Kernspeicher gebracht (in den einzelnen Programmiersprachen gibt es für die Definition solcher konstanten Werte spezielle Anweisungen). Wir nehmen an, daß über die Dualarithmetik zu rechnen ist. Dazu muß der Eingabewert für den Durchmesser entsprechend umgeschlüsselt werden. Bei der Ausgabe muß die Rückverwandlung durchgeführt werden. Das Berechnen des Ausdrucks $\pi/6$. wird nur einmal vorgenommen, da er immer gleich bleibt. Mit einem Einlesebefehl kann nur der Inhalt einer Lochkarte gelesen werden. Sollen mehrere Lochkarten eingelesen werden, müssen weitere Lesebefehle gegeben werden. Im Programmablaufplan wird gezeigt, daß das Programm nach dem Drucken wieder zum Einlesen verzweigt (unbedingte Verzweigung). Hinter den Eingabekarten muß noch eine Karte folgen, die das Ende der Daten anzeigt (vgl. hierzu Abb. 1.25). Wird nach dem Einlesen dieser Karte das eindeutige Endekriterium erkannt, wird der Programmlauf beendet (vgl. hierzu Abb. 1.33).

Der Programmablaufplan wurde zum besseren Verständnis nach dem logischen Fortgang aufgebaut, ohne auf den Befehlsvorrat eines bestimmten Anlagetyps Rücksicht zu nehmen.

Für die Musteraufgabe zur Berechnung des Inhalts einer Kugel würde der Datenflußplan folgendes Aussehen haben:

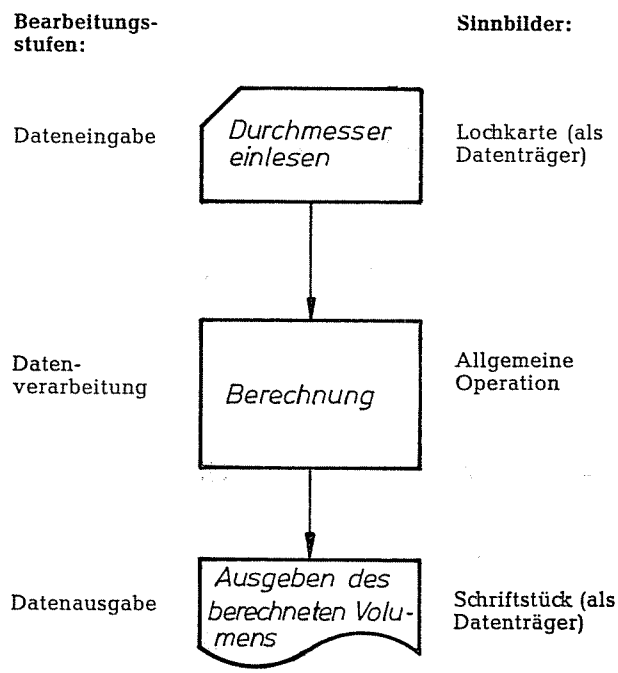


Abb. 1.30 — Datenflußplan zum Beispiel „Berechnen des Inhalts einer Kugel"

Programmiersprachen

Mit dem Programmablaufplan ist der logische Aufbau des Programms, die eigentliche Programmierarbeit, abgeschlossen. Dieser logische Ablauf muß jetzt in eine Form gebracht werden, die von der Datenverarbeitungsanlage verstanden wird. Es muß ein Maschinenprogramm entstehen. Dieser Schritt entspricht den Punkten: **Primärprogramm schreiben** und **Maschinenprogramm erzeugen.**

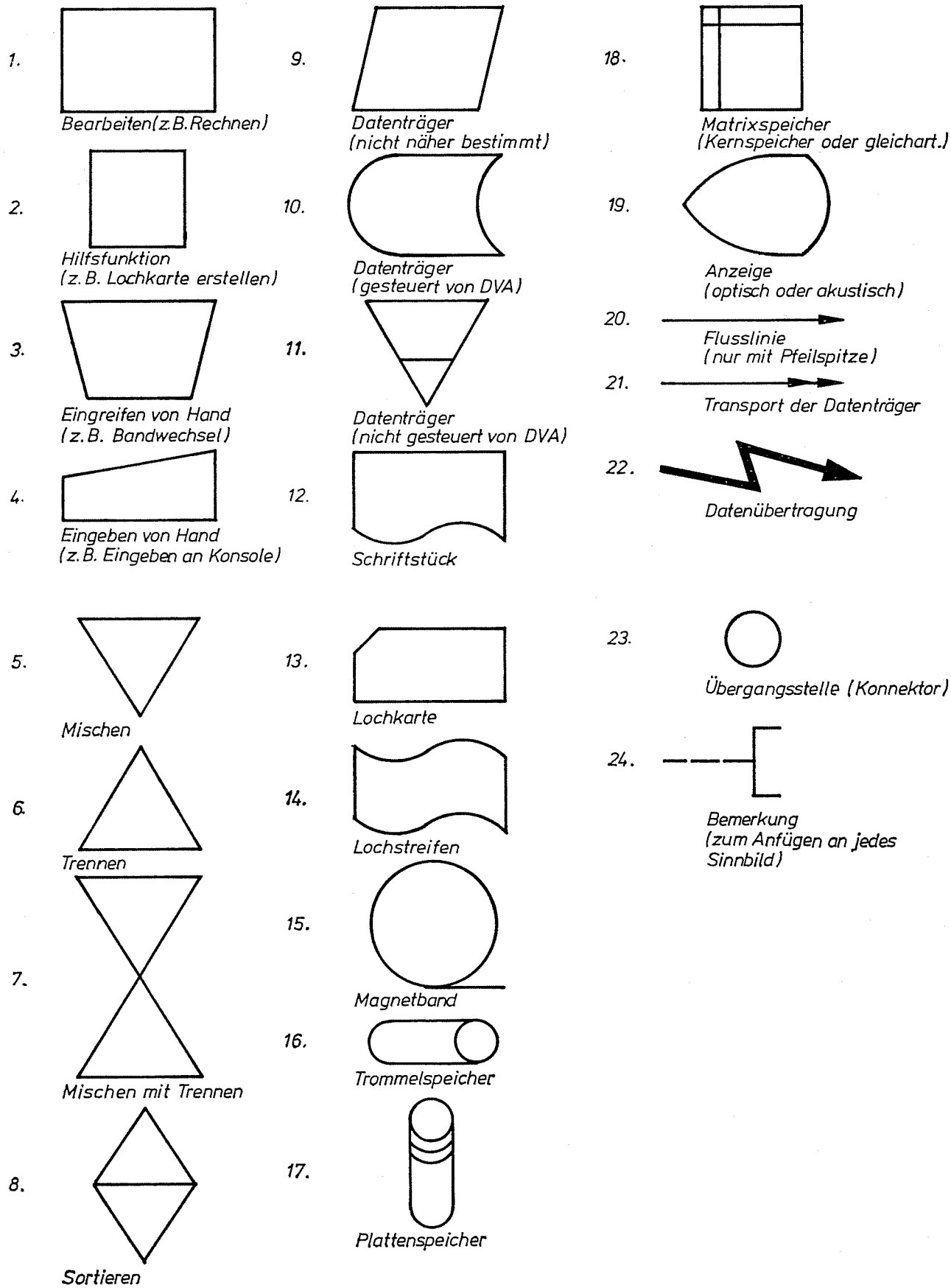


Abb. 1.31 — Symbole für Datenflußpläne

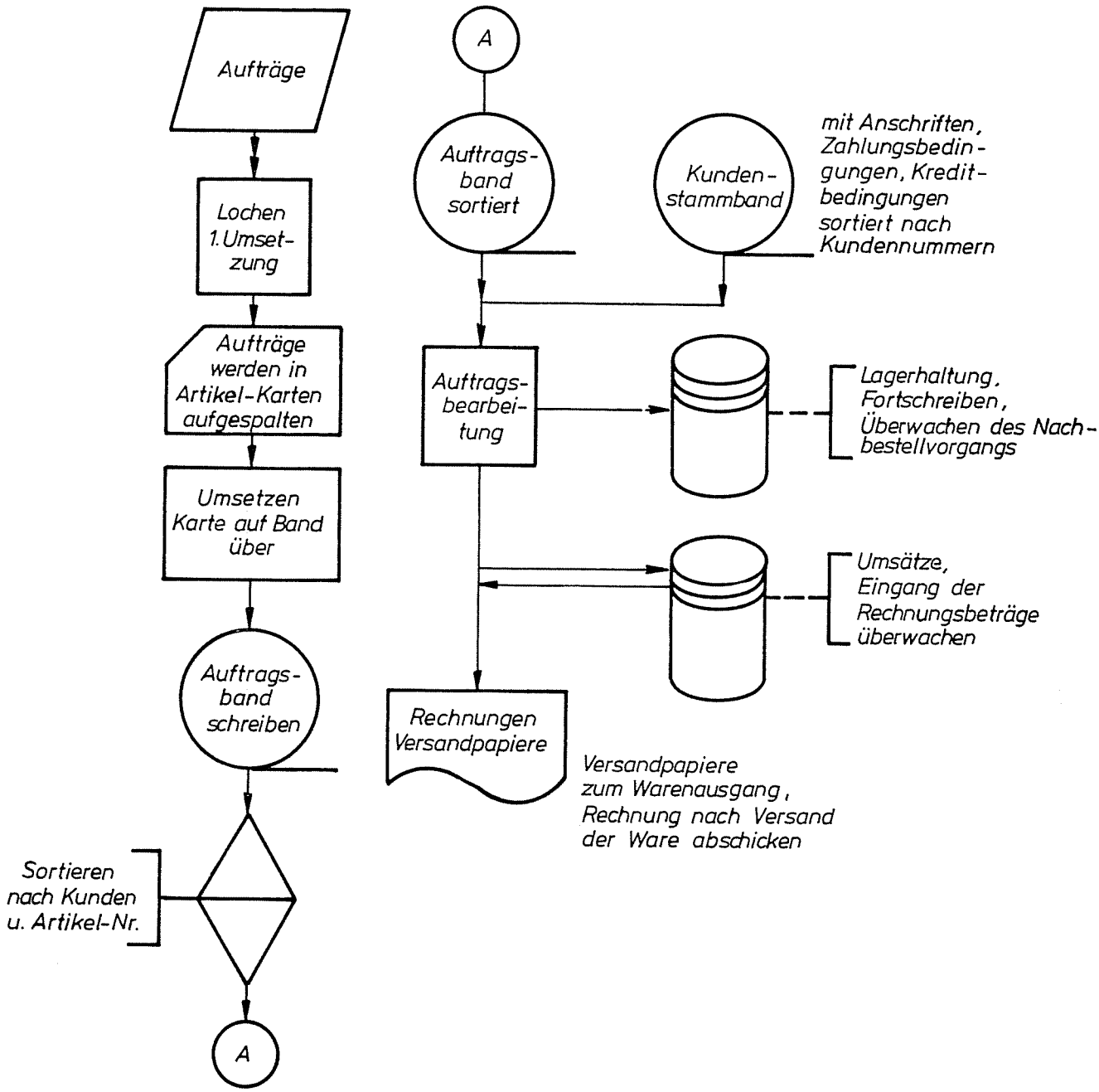


Abb. 1.32 — Beispiel eines Datenflußplanes

Zum Verständnis dieser Punkte muß jedoch das Wesentliche über die Programmiersprachen erläutert werden.

Maschinensprache: Zur Verständigung mit der Maschine ist eine spezielle Befehlscodierung definiert worden. Hier handelt es sich meist um eine binäre Codierung, die sozusagen ein Abbild der Maschinenlogik ist. Wenn man weiß, auf welchen Speicherplätzen die zu verarbeitenden Daten zur Verfügung stehen, kann man die Adressen direkt dual angeben und in das Befehlswort einsetzen (wo die Daten im Kernspeicher abgelegt sind, weiß man aus der Adressenangabe für den Einlesebefehl, dem ja gesagt werden muß, wohin die Daten zu transportieren sind). Fügt man noch den binären Befehlscode hinzu, ist der Befehl komplett. Entsprechend dem in Abb. 1.21 dargestellten Befehl sieht ein maschineninternes Befehlswort so aus, wie in Abb. 1.34 dargestellt.

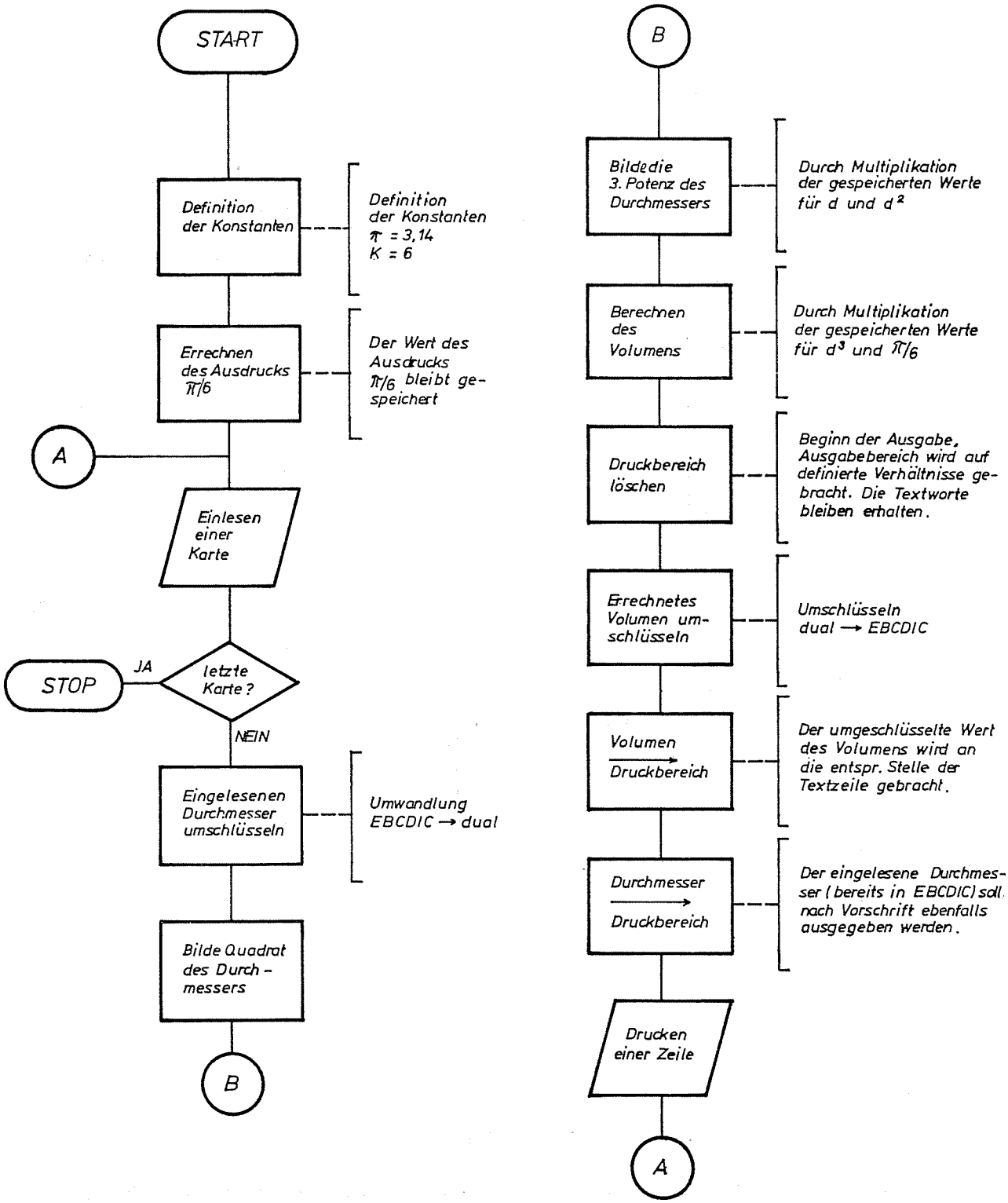


Abb. 1.33 — Programmablaufplan zum Beispiel „Berechnen des Inhalts einer Kugel"

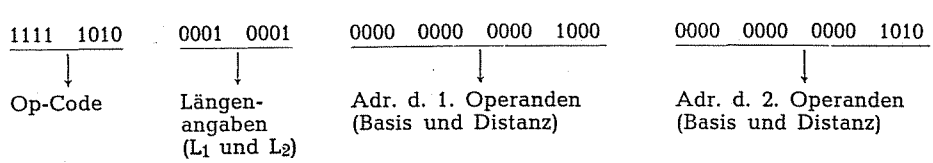


Abb. 1.34 — Maschinenbefehl

Es handelt sich hier um einen Addierbefehl, der den Inhalt von Speicherplatz 10 (wenn Basisadresse gleich Null ist) zu dem Inhalt von Speicherstelle 8 addiert und die Summe in Speicherplatz 8 ablegt. Die Längenangabe für beide Operanden ist 2. Bezogen auf die Musteraufgabe müssen die logischen Einzelschritte des Programmablaufplans durch einen Befehl des Maschinenrepertoires umschrieben werden. Schreibt man alle Befehle in der Sprache, die eine Maschine verstehen kann, codiert man in Maschinensprache. Da die Maschinensprache ein Abbild der internen Logik ist, gibt es für alle Maschinen mit unterschiedlicher Logik eine spezielle Maschinensprache. Die elektronischen Datenverarbeitungsanlagen der 1. Generation wurden durchweg in Maschinensprache programmiert. Da man sich alle Befehlsverschlüsselungen und die Lage der Daten merken muß, ist diese Art der Programmierung jedoch langwierig, schwierig und nur selten fehlerfrei niederzuschreiben.

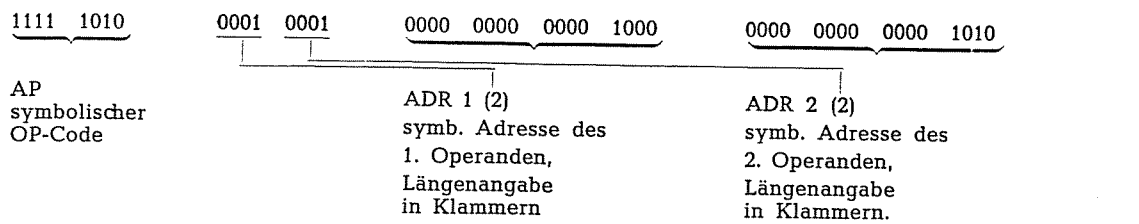
Maschinenorientierte Programmiersprachen:

Eine Erleichterung ergibt sich aber schon dadurch, daß man z.B. für den echten Operationsschlüssel eine symbolische Bezeichnung einsetzt. Diese symbolischen Operationsschlüssel bestehen meist aus mnemotechnischen Abkürzungen (Mnemotechnik: Verfahren zur Unterstützung des menschlichen Gedächtnisses). So steht z.B. AD für Addiere, S für Subtrahiere, M für Multipliziere usw. Etwa in der gleichen Art lassen sich auch die Speicheradressen durch symbolische Adressenangaben ersetzen. Damit entfällt für den Programmierer die Berechnung der direkten (absoluten) Adresse. Ein Befehl in der symbolischen Programmiersprache hat den gleichen Aufbau wie in der Maschinensprache. Für den in Abb. 1.34 dargestellten Maschinenbefehl kann symbolisch angegeben werden:

se zugeordnet werden. Die Einzelschritte eines Programmablaufplans können auch als symbolisch codierte Befehle niedergeschrieben werden. Damit wird die Programmerstellung wesentlich erleichtert; es ergeben sich weniger Fehlermöglichkeiten. Zur Ausführung in der Maschine muß aber das Programm wieder in der Maschinensprache zur Verfügung stehen. Für die gewonnene Programmierungserleichterung ist ein zusätzlicher Übersetzungsvorgang durchzuführen. Da die symbolische Programmierung direkt am Maschinenbefehl orientiert ist (Abb. 1.35), gehen diese beiden Sprachen durch eine einfache Umbenennung ineinander über (maschinenorientierte Programmiersprache). Für den Umbenennungs- oder Übersetzungsvorgang werden die elektronische Datenverarbeitungsanlage und ein Programm benutzt. Das Programm, das einen solchen Umbenennungsvorgang vornimmt, wird allgemein **Assembler** genannt. Häufig werden auch maschinenorientierte symbolische Programmiersprachen als Assembler bezeichnet. In diesem Fall tragen die Sprache und das zugehörige Übersetzungsprogramm den gleichen Namen. Wir fassen die Aufgaben des Assemblers noch einmal zusammen:

1. **Ersetzen der symbolischen Operationsschlüssel durch echte Operationsschlüssel** und
2. **Ersetzen der symbolischen Adressen durch absolute oder relative Maschinenadressen.**

Da jedem symbolischen Befehl ein Maschinenbefehl entspricht, handelt es sich hier um eine 1:1-Zuordnung. Man spricht von einem 1:1-Übersetzer. Jedoch können häufig benötigte Befehlsfolgen innerhalb einer symbolischen Programmiersprache standardisiert werden, d.h., sie werden beim Programmieren nicht niedergeschrieben, sondern an dieser Stelle wird ein Makro-



Hinweis: die symbolische Längenangabe 2 wird im Maschinenbefehl in L_1 = 0001 umgewandelt.

Abb. 1.35 — Symbolische Programmierung

Die Vergabe des Namens für symbolische Adressen steht in gewissem Umfang jedem frei. Er muß jedoch einmal definiert sein, d.h., einmal muß der symbolischen Angabe die echte Adres-

aufruf in das Programm eingesetzt. Erst beim Übersetzungsvorgang wird im Programm die Befehlsfolge der Makrodefinition in das Programm eingefügt und mit übersetzt. Der Makro-

aufruf entspricht hier keinem Äquivalent in der Maschinensprache, sondern einer vorher festgelegten Folge von Maschinenbefehlen.

Problemorientierte Programmiersprachen: Auch das Programmieren mit maschinenorientierten Programmiersprachen ist aufwendig, weil immer noch die Zerlegung in maschinengerechte Einzelschritte notwendig ist. Dagegen sind problemorientierte Programmiersprachen von der Logik einzelner Maschinentypen unabhängig. Damit ist es dem Programmierer möglich, Aufgaben in einer der Logik der Fachsprache nahestehenden Form zu definieren (problemorientiert). Die Anpassung an die Maschinenlogik (Zerlegung in ausführbare Einzelschritte) wird jetzt maschinell durch den Übersetzungsvorgang vorgenommen. Die Ausdrucksmittel von problemorientierten Programmiersprachen sollen es ermöglichen, Programme in einer bequem lesbaren, natürlichen Form niederzuschreiben. Auf das gewählte Beispiel angewendet, lautet z.B. eine problemorientierte Programmierung:

```
.
.
.
D = 20
.
.
.
.
V = 3,14/6 * (D ** 3)
.
.
.
.
```

Abb. 1.36 — Problemorientierte Programmierung

Man sieht deutlich, daß das Problem hiermit in enger Anlehnung an die mathematische Formelsprache programmiert werden kann. Die problemnahen Programmiersprachen verlangen eine eindeutige und verständliche Definition (Standardisierung). Die angegebenen Möglichkeiten (Anweisungen) müssen durch entsprechende Operationen interpretierbar sein.

Zum Ablauf muß die Befehlsfolge im Maschinencode vorliegen. D.h., man muß die in der Logik der Fachsprache formulierte (programmierte) Aufgabe in die Logik der Maschine überführen. Hier werden wieder die elektronische Datenverarbeitungsanlage sowie ein Programm (Übersetzer) benutzt, das zunächst die Anweisung in maschinenausführbare Schritte zerlegt und die zur Ausführung dieser Schritte benötigten echten Maschinenbefehle erzeugt. Ein solcher Übersetzer wird im allgemeinen als **Compiler** bezeichnet. Da einer Anweisung im Programm eine Folge von mehreren generierten Maschinenbefehlen entspricht, wird dieser Vorgang auch 1 : n - Übersetzung genannt. Die in Abb. 1.36 gezeigte Anweisung wird entsprechend der geschilderten Einzelschritte vom Compiler zerlegt. Da ein Compiler für sämtliche Arten von Anweisungen nur einen bestimmten Rahmenablauf zugrunde legen kann, erzeugt er nicht die optimale Befehlsfolge, wie sie in der Assemblersprache möglich wäre. Compilierte Programme belegen im allgemeinen mehr Speicherplatz und führen damit zu größeren Programmlaufzeiten. Dagegen läßt sich aber ein Programm in problemorientierter Sprache leichter schreiben, lesen, verstehen, testen und leichter erlernen. Es werden keine Kenntnisse über die maschineninterne Logik benötigt. Problemorientierte Sprachen lassen sich somit — sofern ein entsprechendes Programmiersystem vorhanden ist — für die elektronischen Datenverarbeitungsanlagen verschiedener Hersteller einsetzen.

Die bekanntesten problemorientierten Programmiersprachen sind:

FORTRAN (FORmular TRANslation); die FORTRAN-Sprache ist eine leistungsfähige und flexible Programmiersprache, die sich eng an die Symbolik der Mathematik anlehnt. Sie dient hauptsächlich zur Lösung von technisch-wissenschaftlichen Problemen. Die FORTRAN-Sprache wurde schon in den Jahren 1954/55 entwickelt und ab 1956 für die Benutzer zur Verfügung gestellt. Inzwischen ist sie, merklich verbessert, die am weitesten verbreitete Programmiersprache geworden. FORTRAN wird auch in der kommerziellen Datenverarbeitung eingesetzt.

ALGOL (ALGOrithmic Language); sie dient ausschließlich zur Beschreibung und Programmierung in der naturwissenschaftlich-technischen Anwendung (auch für Verfahren der numerischen Mathematik). Sie richtet sich weitestgehend nach den Symbolen der Mathematik. Diese Programmiersprache wird zum größten Teil im Universitätsbereich angewandt (auch bei wissenschaftlichen Veröffentlichungen).

COBOL (COmmon Business Oriented Language); die COBOL-Sprache eignet sich besonders zum Programmieren kommerzieller Aufgaben. Die Entwicklung dieser Sprache geht auf eine Zusammenarbeit von Benutzern und Herstellern elektronischer Datenverarbeitungsanlagen zurück und ist speziell auf kaufmännische und buchhalterische Datenverarbeitung zugeschnitten. Insbesondere besteht große Freizügigkeit in der Art der Datendarstellung und in der Behandlung großer Datenmengen.

PL 1 (Programming Language 1); sie ist eine neuentwickelte problemorientierte Programmiersprache, die sowohl für die Lösung kommerzieller als auch wissenschaftlich-technischer Probleme geeignet ist. Sie vereinigt in sich Elemente von FORTRAN und COBOL und erlaubt auch die Ausnutzung aller Möglichkeiten einer DVA (wie Sofortverarbeitung etc.).

In diesem Zusammenhang müssen auch die Generatorprogramme aufgeführt werden. Der Generator erzeugt aufgrund von Angaben über die Problemstellung, über Art, Struktur und Lage der Daten sowie über die Anlagenkonfiguration ein Programm, das die gestellte Aufgabe (meistens Sortieren, Mischen, Listenschreiben) mit geringem Programmieraufwand löst. Ein Generator ist auch ein Übersetzer und damit ein Programm, das ein anderes Programm dadurch erzeugt, daß die Angaben in skeletthaft vorhandene Programmteile eingefügt werden oder in dem Maschinenbefehle gemäß vorhandener Modellbefehle erstellt werden.

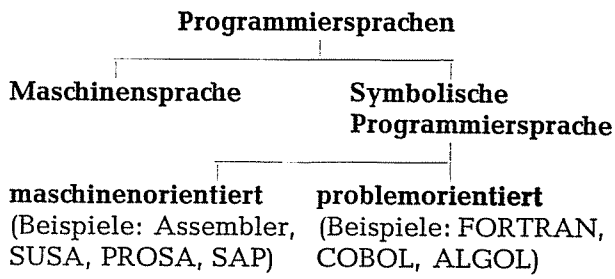


Abb. 1.37 — Übersicht über die Programmiersprachen

Im allgemeinen wird das in einer Programmiersprache niedergeschriebene Problem als Quellenprogramm oder Primärprogramm bezeichnet, während das aus dem Übersetzungsvorgang gewonnene, in der Maschine ablauffähige Programm als Maschinenprogramm oder Objektprogramm bekannt ist.

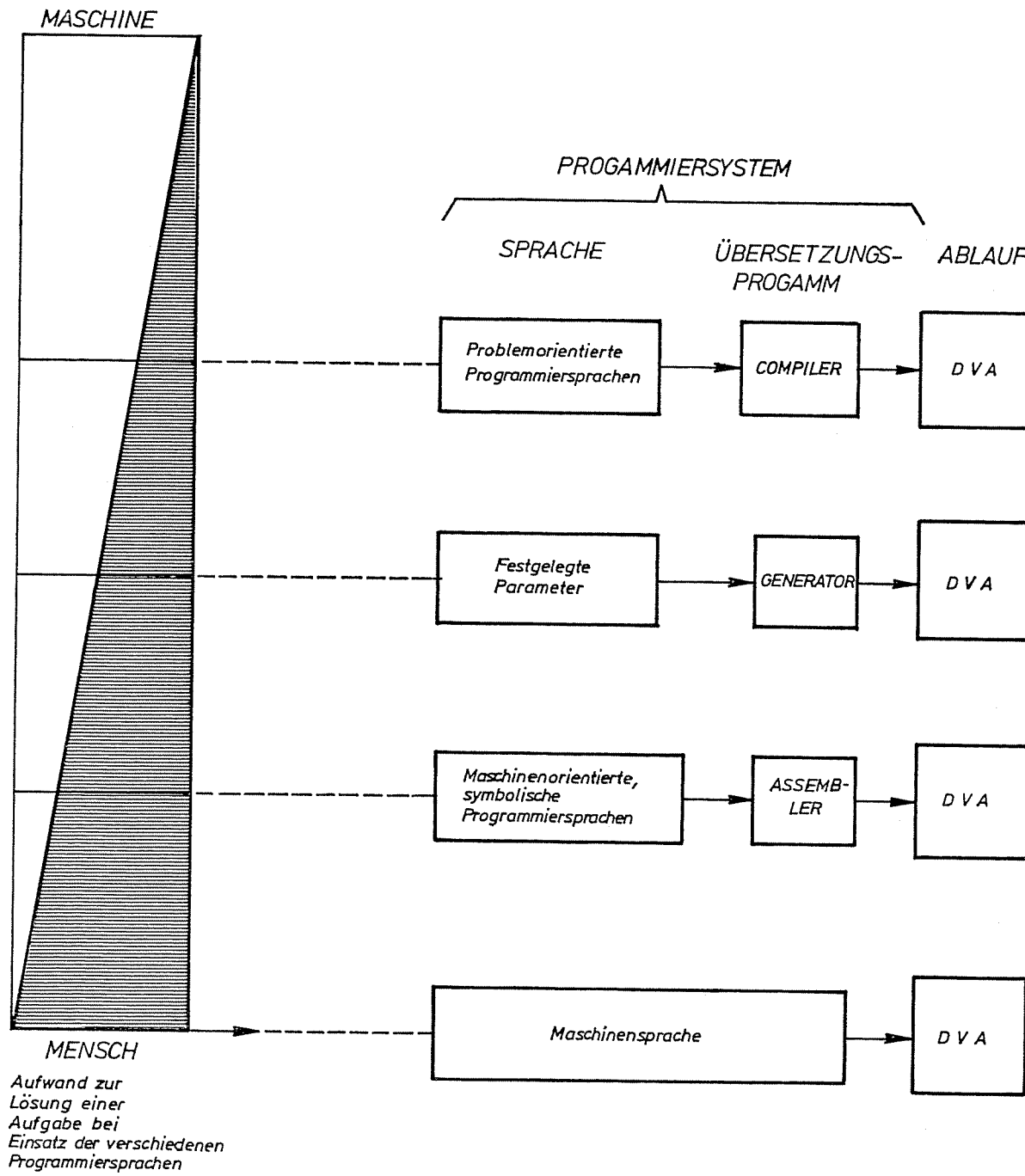


Abb. 1.40 — Möglichkeiten zur Programmerstellung

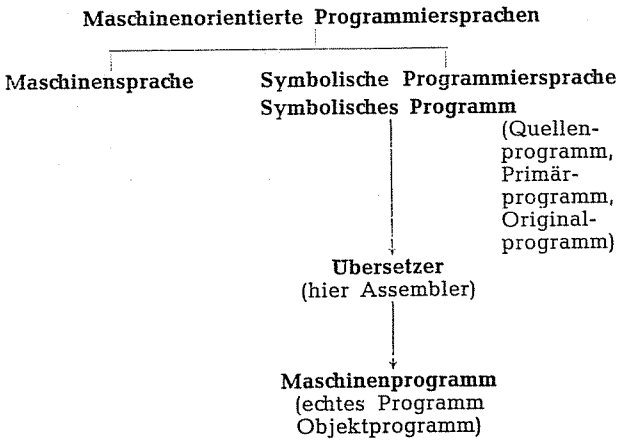


Abb. 1.38 — Übersicht über die maschinenorientierten Programmiersprachen

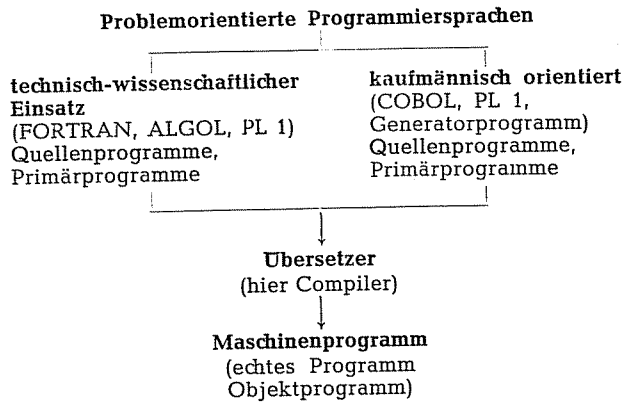


Abb. 1.39 — Übersicht über die problemorientierten Programmiersprachen

Nachdem nun die Ausführungen zu den Programmiersprachen abgeschlossen sind, fahren wir mit der Beschreibung der Programmerstellung fort. Ein Primärprogramm erstellen heißt, die einzelnen Schritte des Programmablaufplans in ein geschriebenes Programm — als Folge von Anweisungen in eine bestimmte Programmiersprache — umsetzen. Bei der Formulierung der Arbeitsschritte zur Lösung des Problems bringen höhere Programmiersprachen entscheidende Erleichterungen. Es wurde bereits erwähnt, daß der Schritt vom Primärprogramm zur Maschinensprache automatisch durch den Einsatz von Übersetzungsprogrammen (Assembler, Compiler) durchgeführt werden kann. Damit die Anweisungen des Primärprogramms von der Datenverarbeitungsanlage eingelesen werden können, müssen die Anweisungen zunächst abgelocht werden, damit sie maschinell lesbar sind. Es wird immer eine Anweisung in eine Lochkarte gestanzt. Die Anweisungen selbst werden auf spezielle Formulare geschrieben, damit sie leichter abgelocht werden können. Die Primärkarten bilden die Eingabedaten für das Übersetzungsprogramm. Nach dem Ablauf der Übersetzung werden die Maschinenbefehle des Programms auf Lochkarten ausgegeben (oder auf externen Speichern in Bibliotheken abgelegt). Der Kartensatz mit dem Objektprogramm kann aufbewahrt und zu einem Ablauf in den Kernspeicher eingelesen (geladen) werden. Der Übersetzer protokolliert neben dem Primärprogramm auch die Befehle der Maschinensprache (meist in sedezimaler Form) und weist auf formale Fehler in den Primäranweisungen hin. Vor Beginn des Übersetzungsvorgangs muß erst das Übersetzungsprogramm selbst in den Arbeitsspeicher gebracht werden; Übersetzungsprogramme liegen in Maschinenform meist auf externen Speichermedien vor (vgl. hierzu Abb. 1.41).

Nach der Übersetzung muß das gewonnene Objektprogramm zur Ausführung erst geladen werden, und zwar wird das vorher eingelesene Übersetzungsprogramm zerstört und überschrieben. Dieses Programm erfordert dann seine eigenen Eingabedaten. Die Übersetzungsprogramme werden vom Hersteller einer Datenverarbeitungsanlage erstellt und gehören zum Software-Lieferumfang. Für das Erlernen dieser Sprachen gibt es bei den Herstellerfirmen sogenannte Kundenlehrgänge. Im allgemeinen werden die Komponenten Sprache und Übersetzungsprogramm mit dem Begriff „Programmier-System" umschrieben. Problemorientierte Programmiersprachen sind im wesentlichen herstellerunabhängig und damit auch nicht an spezielle Anlagen gebunden. Sie werden in der modernen Generation der Datenverarbeitungsanlagen bevorzugt eingesetzt. Die Nachteile einer längeren Laufzeit und größeren Speicherbelegung werden durch die Vorteile einer leichteren Erlernbarkeit und einfacheren Anwendung aufgehoben.

Programmtest

Schon während des Übersetzungsvorgangs finden die ersten Programmprüfungen statt. Die Übersetzungsprogramme sind in der Lage, formale Fehler innerhalb der Codierung zu erkennen (z.B. Lochfehler, nicht definierte Adressen, nicht zugelassene Zeichen, Fehlen von immer notwendigen Anweisungen usw.). Der schwierige Teil des Ausprüfens beginnt erst mit dem logischen Test eines Programms. Es ist selten, daß ein Programm auf Anhieb von Anfang bis Ende durchläuft. Neben dem Programmieren sind zur Programmprüfung Probebeispiele (Testdaten) zu entwerfen, die so beschaffen sind, daß sie ein genau bekanntes Ergebnis liefern und außerdem alle evtl. in der Praxis vorkommenden Fälle berücksichtigen. Manchmal ist es auch sinnvoll,

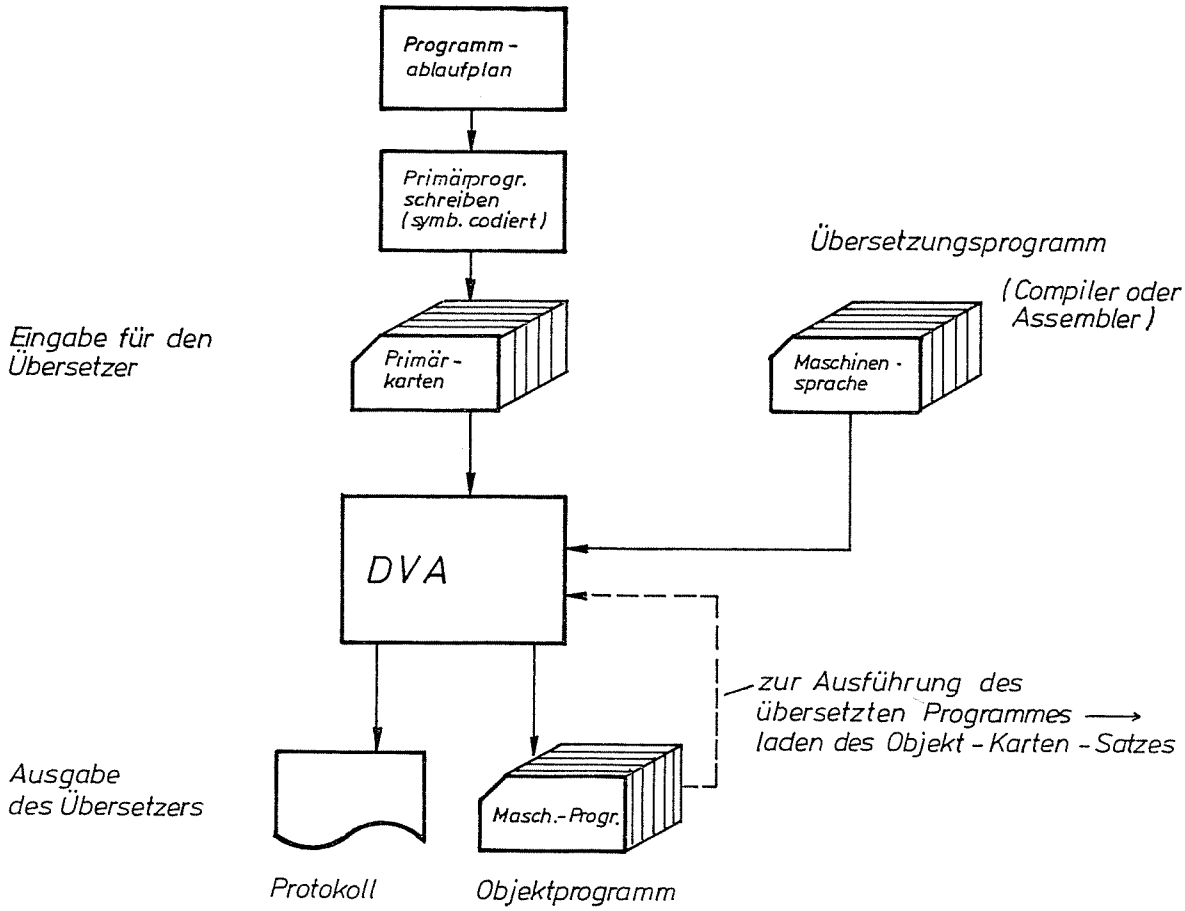


Abb. 1.41 — Schematische Darstellung des Übersetzungsvorgangs

Programme in kleineren, überschaubaren Teilen auszutesten. Um aber logische Fehler in ihren Ursachen erkennen zu können, muß der Programmierer wissen, was im Arbeitsspeicher geschehen ist. Dazu kann er sich den Speicherinhalt in sedezimaler Form ausdrucken lassen und dann analysieren. Wurde im Probefall festgestellt, daß das Programm falsche Ergebnisse liefert oder an undefinierten Stellen stehenbleibt (z.B. kein Aussprung aus Schleife), so steht neben dem Speicherabzug noch ein **„Tracing"-Programm** zur Verfügung. Tracing-Programme interpretieren das zu prüfende Programm, indem sie für jede Anweisung innerhalb des Programms bestimmte Angaben ausschreiben, die die Fehlersuche erleichtern.

Programmablauf

Ist ein Programm vollständig ausgetestet, kann es für den Einsatz im Datenverarbeitungsbetrieb freigegeben werden. Ein einmal fertiges, fehlerfreies Programm kann immer wieder, auch mit anderen Eingabedaten, verwendet werden. Bei Ablauf eines fertigen Programms ist der Programmierer nicht mehr anwesend. Die Handhabung des Programms und der Daten übernimmt der Operateur, der die Anlage bedient (Ablauf vorbereiten, Ablauf durchführen, Ergebnisse dem entsprechenden Anwender zuschikken usw.). Hierfür müssen ihm eine Bedienungsanweisung und eine Programmbeschreibung zur Verfügung stehen. Jedes Programm muß dokumentiert werden. Dies ist noch Aufgabe des Programmierers. Für jedes Programm müssen detaillierte Beschreibungen vorhanden sein. Im allgemeinen sollten sie folgende Teile enthalten:

1. eine Bedienungsleitung mit Angaben über Schalterstellungen, Verhalten bei Fehlern, Beschreibung der Eingabedaten und ihrer Formate, benutzte Einheiten,
2. eine allgemeine Beschreibung des Programms mit Namen, Autor, Aufgabenstellung, Zielsetzung (mathematische Methode, Formelplan, Genauigkeit, Fehlerrechnung, Konvergenzbedingungen), Datenflußplan, Programmablaufplan, Testdaten und Testergebnisse, Speicherbedarf, Rechenzeit, Art der verwendeten Programmiersprache, Fehlerausdrucke des Programms selbst und

3. spezielle Angaben für einen anderen Benutzer, der evtl. Änderungen einbauen möchte (Aufruf des Programms, Steuerung durch Parameter, Einbaumöglichkeiten in Betriebssysteme usw.).

Eine sorgfältige und gewissenhaft geführte Dokumentation kann das Entwickeln von neuen Programmen einschränken, weil für die Lösung eines Problems evtl. schon fertiggestellte Programme mit kleinen Änderungen verwendet werden können oder als Unterprogramme zu integrieren sind.

Da aber die Anwendungsgebiete nicht stabil bleiben, wird eine Anpassung an neue Bedingungen immer notwendig sein. Eine Programmpflege ist deshalb auch nach Eingliederung in den Datenverarbeitungsbetrieb notwendig.

Zusammenfassend wird für eine Programmentwicklung unter Berücksichtigung des Aufwandes Mensch/Maschine folgendes Bild gegeben:

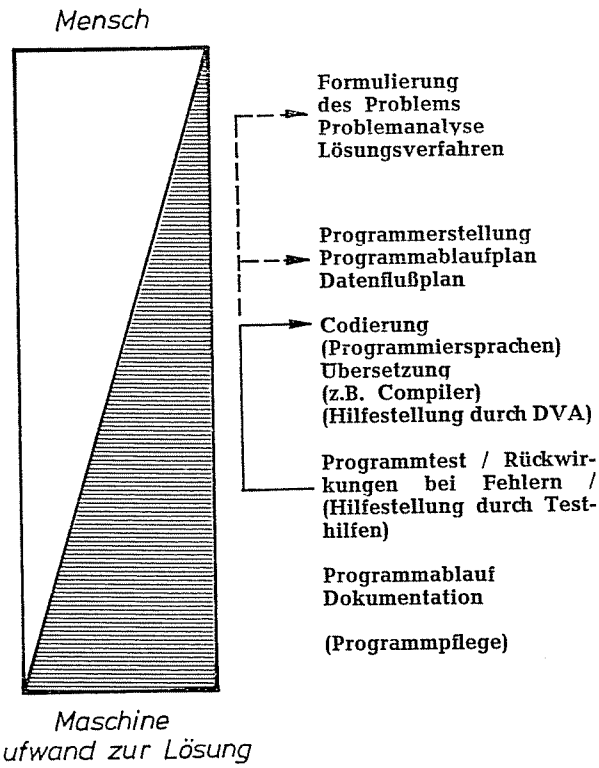


Abb. 1.42 — Weg von der Aufgabenstellung bis zum Programm

1.3.5. Betriebsweisen von Datenverarbeitungsanlagen

Zunächst war die Benutzung einer elektronischen Datenverarbeitungsanlage eigentlich nur auf einzelne Anwender mit speziellen Einzelaufgaben abgestellt. Es konnte immer nur ein Arbeitsvorgang ablaufen, d.h., ein Programm wurde nach dem anderen abgearbeitet. Ein Hauptnachteil dieser Einzelprogrammverarbeitung war die schlechte Ausnutzung der Peripherie und der Zentraleinheit. Rechenintensive Programme belegen die Ein- und Ausgabegeräte nur kurzzeitig; andererseits wird aber bei Programmen, die die Peripherie auslasten, der Rechnerkern nur wenig angesprochen. Zusätzlich ergab sich noch, daß jede Aufgabe von dem Operateur an dem Rechner neu aufbereitet und eingegeben werden mußte. Als Gegenmaßnahmen wurden zunächst Normierungen der Programme und Bedienungsanweisungen sowie die Einführung von einfachen Betriebssystemen (Monitor) vorgenommen. Dadurch war es möglich, einen Teil der bisher vom Operateur wahrgenommenen Funktionen auf die Maschine selbst zu übertragen. Trotz dieser Fortschritte war die optimale Ausnutzung der Peripherie noch nicht gegeben. In einem großen Rechenzentrum sind viele Programme vorhanden, die unterschiedliche Anforderungen an die Betriebsmittel (Ein- und Ausgabegeräte, Speicherplatz) stellen. Zur besseren Ausnutzung mußte ein erweitertes Betriebssystem entwickelt werden, das mehrere Programme gleichzeitig laden kann. Hierbei wird nun eines dieser Programme gestartet und das Betriebssystem übernimmt die Aufgabe zu untersuchen, ob eines der anderen Programme, die im Speicher stehen, die noch verfügbaren Betriebsmittel nutzen kann. Den Aufgaben, die sofort erledigt werden müssen und solchen, die durch Wartezeiten nicht benachteiligt werden dürfen, kann durch eine Vorrangsteuerung Rechnung getragen werden (Prioritätenvergabe). Die Überwachung und Behandlung mehrerer Programme wird als Mehrprogrammbetrieb (Multiprogramming) bezeichnet und durch speziell hierfür entwickelte Betriebssysteme realisiert. Ein Nachteil ist die Kompliziertheit des Betriebssystems, das um so schwieriger wird, je besser das Zusammenspiel des Gesamtsystems sein soll. Mit der Entwicklung des Mehrprogrammbetriebes einher ging die Einführung der Datenfernübertragung, d.h., der Anschluß von Datenübertragungseinrichtungen an Datenverarbeitungsanlagen. Die Endplätze sind als Datensende- und Datenempfangsstationen über Übertragungsleitungen an eine entfernte zentrale Datenverarbeitungsanlage angeschlossen.

Erst die Kombination von Mehrprogrammbetrieb und Datenfernverarbeitung führt zu einer neuen Stufe der Betriebssysteme, zum **Dialogsystem.** Im Gegensatz zum Mehrprogrammbetrieb erfolgt hier die Steuerung der Benutzer-

Aufgaben durch ein gegebenes Zeitraster; jeder Endplatz erhält einen echten Anteil am System. Die Dialogsysteme (Time-sharing-Systeme) bieten den Vorteil, daß der Benutzer direkten Kontakt mit der Anlage hat. Die Überlegungszeit des Menschen ist größer als die Reaktionszeit des Teilnehmer-Rechensystems, so daß beim Endplatzbenutzer der Eindruck entsteht, er sei der einzige Teilnehmer am System. Dialogsysteme haben meist noch genügend Kapazität, um auch Programme abzuarbeiten, die nicht im Dialogbetrieb arbeiten (Hintergrundprogramme). Nach dieser globalen Beschreibung der Entwicklungstendenzen wird im folgenden auf einzelne Formen des Rechnerbetriebs eingegangen.

1.3.5.1. Spezielle Ein-Ausgabekanäle

Das Geschwindigkeitsgefälle in der Verarbeitung zwischen den Einrichtungen der Peripherie ($\sim$ msek) und der Zentraleinheit ($\sim \mu$sek) ist sehr groß. Da aber eine Information, die eingelesen werden muß, erst verarbeitet werden kann, wenn sie vollständig im Speicher vorliegt, ergibt sich ein ungünstiges Verhältnis zwischen Warte- und Arbeitszeiten einer Zentraleinheit. Ein erster Schritt zur Beseitigung dieses Mißverhältnisses war erreicht, als der Ein-Ausgabe-Verkehr über eigenständige Ein-Ausgabekanäle abgewickelt wurde. Diese E/A-Kanäle arbeiten selbständig. Sie werden zum Beispiel für eine Eingabe vom System angestoßen und erst dann wieder abgefragt, wenn sie dem System das Ende des Auftrages anzeigen. Prinzipiell arbeiten die E/A-Geräte hierbei mit ihrer eigenen Arbeitsgeschwindigkeit mit einem Pufferspeicher (Kanal) zusammen. Erst wenn der Pufferspeicher die entsprechende Anzahl Zeichen aufgenommen oder abgegeben hat, erfolgt ein Datentransport zwischen Puffer und Arbeitsspeicher. Während der E/A-Kanal eine Arbeit ausführt, kann das Programm schon weiterarbeiten (bis z.B. die Eingabedaten selbst verarbeitet werden müssen). Da jedoch mit einem Programm keine weitergehende Ausnutzung der Speicher- und Rechenkapazität zu erzielen ist, ging man nach Weiterentwicklung der E/A-Kanaltechnik zum Multiprogramming über.

1.3.5.2. Multiprogramming

Bei den modernen Rechenanlagen werden zur Übertragung von Daten zwischen E/A-Geräten und der Zentraleinheit Selektor- und Multiplexkanäle verwendet. Selektorkanäle bedienen immer nur ein schnelles Ein-/Ausgabegerät, während Multiplexkanäle mehrere langsame Ein-/Ausgabegeräte versorgen können. Diese Kanäle arbeiten nach Anstoß durch die Zentraleinheit selbständig. Diese Tatsache macht man sich beim Multiprogramming zunutze, indem man mehrere Programme „simultan" arbeiten läßt. Dabei wird unter Multiprogramming das Ausnutzen von Wartezeiten (bei E/A-Vorgängen) eines Programms durch ein anderes Programm verstanden. Entstehen Pausen durch E/A-Vorgänge, arbeitet die Zentraleinheit in der Zwischenzeit andere Programme ab. Für die Durchführung des Multiprogramming müssen gewisse Voraussetzungen erfüllt werden:

1. **Relative Adressierung.** Beim Multiprogramming muß es möglich sein, mehrere Programme im Arbeitsspeicher zu haben (Arbeitsspeicher muß groß sein). Da man zunächst nicht weiß, auf welche Arbeitsspeicherplätze ein Programm geladen wird, darf man die Programmadressen nicht absolut (direkt) sondern muß sie verschiebbar (relativ) angeben. Bei der Übersetzung von Programmen vergibt der Übersetzer die Adressen so, als würde das Programm immer an der Speicherstelle mit der Adresse Null beginnen. (Adressierung relativ zu Null, die erste Speicherstelle des Arbeitsspeichers hat immer die Adresse Null.) Beim Laden des Programms muß vom Betriebssystem auf alle Adressen des Programms ein konstanter Wert addiert werden (verschiebbares Laden). Abhängig von der Wahl des Wertes der Konstanten kann ein Programm an beliebige Stellen im Arbeitsspeicher gebracht werden. Bei dem unter 1.3.3.2. beschriebenen indirekten Adressierungsverfahren (Basis + Distanz) genügt hier schon eine entsprechende Änderung des Inhaltes des verwendeten Basisregisters.

2. **Vergabe von Programmprioritäten.** Den Programmen wird schon beim Laden eine Priorität zugeordnet. Zunächst wird das Programm mit der höchsten Priorität abgearbeitet. Treten Unterbrechungen — hier E/A-Wartezeiten — auf, wird das Programm mit der nächst niederen Priorität ausgeführt. Ist jedoch in der Zwischenzeit ein Programm mit höherer Priorität arbeitsfähig geworden (z.B. E/A-Vorgang beendet), wird natürlich mit der höheren Priorität weitergearbeitet.

Der Übergang zwischen den einzelnen Programmen muß von einem **Organisationsprogramm** gesteuert werden. Dies ist ein Programm, das

vom Hersteller der Datenverarbeitungsanlage geliefert wird und ein Teil der später erläuterten Betriebssysteme ist. Dieses Organisationsprogramm muß alle Ein-/Ausgabeübertragungen überwachen, indem es alle Ein-/Ausgabebefehle selbst erteilt. (Ein vom Benutzer geschriebener Lese- oder Schreibbefehl wird von der Maschine in einen freiwilligen Sprung in das Überwachungsprogramm übersetzt). Bei jeder Beendigung eines Ein-/Ausgabevorgangs wird die Kontrolle durch eine automatische Unterbrechung dem Überwachungsprogramm überreicht. Eine DVA arbeitet nach dem „one-at-a-time"-Prinzip. Es kann immer nur ein Programm laufen und damit auch die notwendigen Register belegen. Die Register müssen sowohl vom Überwachungsprogramm als auch von dem Benutzerprogramm belegt werden. Damit aber jedes unterbrochene Programm an definierter Stelle wieder anfangen kann, muß bei jeder Unterbrechung und bei jedem freiwilligen Sprung in das Organisationsprogramm das Abspeichern der Registerinhalte des unterbrochenen Programms erfolgen und umgekehrt (wenn Überwachungsprogramm fertig ist und ein normales Programm weiterlaufen kann). Da Datenverarbeitungsanlagen über mehrere Register verfügen, nehmen diese dauernden Umspeicherungen relativ viel Zeit in Anspruch (Koordinierungsaufwand).

An folgendem Beispiel soll die Abarbeitung von drei Programmen im Multiprogramming-Betrieb gezeigt werden:

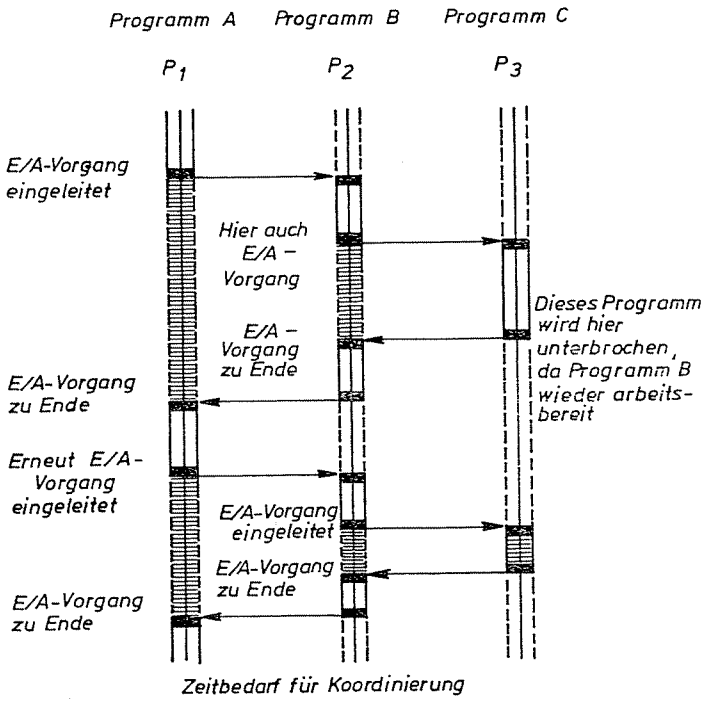


Abb. 1.43 — Schema zum Multiprogramming

Im Arbeitsspeicher sind 3 Programme verschiedener Prioritäten vorhanden (P_1 = höchste Priorität). Prog A läuft bis E/A-Vorgang auftritt. Die entstehende Wartezeit nutzt Prog B solange, bis hier auch ein E/A-Befehl auftritt. Die hier entstehende Wartezeit nutzt Prog C, das hier keinen E/A-Vorgang einleitet. Programm C hat aber die niedrigste Priorität; es wird unterbrochen, sobald ein Programm mit höherer Priorität arbeitsbereit wird (hier Prog B, E/A-Vorgang beendet). Programm B wird fortgesetzt, bis Ein-Ausgabe-Wartezeit von Programm A beendet ist. Das Programm mit der niederen Priorität wird unterbrochen (Prog B) und Prog A läuft weiter. Die im Programmlauf gekennzeichneten Strecken stellen den Zeitbedarf der „Haushaltsroutinen" (Umspeichern, Register etc.) dar, der für die Koordinierung durch das Organisationsprogramm benötigt wird. Die Anzahl der simultan arbeitenden Programme ist abhängig von der Größe des Arbeitsspeichers und den Fähigkeiten des Organisationsprogramms. Zusammenfassend kann gesagt werden: das Organisationsprogramm hält alle zur Zeit laufenden E/A-Operationen fest. Einzelne Programme können also nicht E/A-Vorgänge ansprechen; sie müssen ihre Ein- und Ausgaben beim Organisationsprogramm anmelden, das den E/A-Vorgang anstößt und dessen Beendigung überwacht. Muß z.B. ein Programm auf Eingabedaten warten, stellt das Organisationsprogramm den augenblicklichen Stand des Programms sicher und fährt mit der nächst niederen Priorität fort.

Mit der Einführung des Multiprogramming war es auch möglich, **Real-Time-Betrieb** (Echtzeitverarbeitung) durchzuführen. Echtzeitverarbeitung bedeutet, daß innerhalb einer Datenverarbeitungsanlage ständig Programme betriebsbereit sind, die die Verarbeitungsergebnisse in einer vorgegebenen Zeit zur Verfügung stellen. (Sonderform: Prozeßrechner, wenn die Istwerte eines Regelkreises den Eingabedaten und die Stellwerte den Ausgabedaten entsprechen.) Echtzeitprogramme erhalten im Simultanbetrieb die höchste Priorität und sind damit immer arbeitsbereit.

1.3.5.3. Time-Sharing-Betrieb

In dem Simultanbetrieb der bisher geschilderten Art ergibt sich ein Problem, wenn z.B. in der höchsten Priorität ein Programm mit wenig oder keinen E/A-Vorgängen abläuft. Für andere Programme nutzbare Wartezeiten entstehen somit nur sehr selten oder überhaupt nicht. Sind z.B. mehrere Programme (40—100) so zu koordinieren, läßt sich kaum eine brauchbare Zuordnung finden. Um diesen Nachteil zu vermeiden, geht man dazu über, daß man z.B. auf die Prioritäten verzichtet und jedem Programm einen gewissen Zeitanteil der Rechenzeit zuordnet. Ist diese Zeit verbraucht, wird das Programm unterbrochen und am Ende einer Warteschlange eingereiht. Jedes Programm kommt sozusagen gleichmäßig an die Reihe. Dieses Verfahren ist auch als Time-Sharing bekannt. Datenverarbeitungsanlagen, die dies ermöglichen, heißen auch Teilnehmer-Rechensysteme oder Dialogsysteme. Solche Systeme sind sehr flexibel, weil man den externen

Benutzern nicht die ganze Periode zuzuordnen braucht, sondern zusätzlich die üblichen Arbeiten eines Rechenzentrums (auch im Multiprogramming) durchführen kann. Andererseits können aber auch einzelnen Teilnehmern verschiedene Zeitintervalle und Prioritäten zugeordnet werden. Eingesetzt wird das Time-Sharing-Verfahren für den Dialogverkehr (Problemeingabe über Datenstation → Antwort im selben Augenblick ebenfalls über Datenstation) vieler Benutzer mit der Datenverarbeitungsanlage. Die Zuteilungszeit für jeden Teilnehmer bezieht sich nur auf die Zuordnung des Rechnerkerns; eingeleitete E/A-Vorgänge werden aber auch während dieser Zeit bis zur nächsten Zuordnung ausgeführt. Dadurch entsteht beim Teilnehmer der Eindruck, daß er die Datenverarbeitungsanlage für sich allein beansprucht.

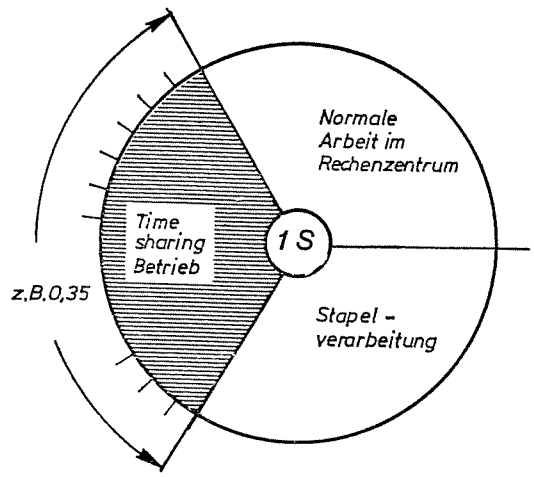


Abb. 1.44 — Verteilung der Zeitscheibe

Je mehr Programme jedoch im Arbeitsspeicher vorhanden sein sollen, um so mehr Kernspeicherplatz muß zur Verfügung gestellt werden. Andererseits lassen aber technische und Kostengründe oft die Realisierung von Arbeitsspeichern mit mehr als 512 kByte nicht zu. Die Erweiterung des Arbeitsspeichers gelingt in einer wirtschaftlich vertretbaren Form, indem man augenblicklich nicht benutzte Programmteile auf schnelle externe Speichermedien (Trommel) auslagert.

Beispiel: In einem Arbeitsspeicher ist nur ein Teil eines Programms untergebracht; weiterer Platz ist nicht vorhanden. Wird nun der nicht im Arbeitsspeicher befindliche Teil des Programms benötigt, muß vom Betriebssystem folgender Vorgang eingeleitet werden: Zu einem bestimmten Zeitpunkt arbeitet immer nur ein Programm. Deshalb können von einem Programm, das noch länger warten muß, Teile auf externe Speicher ausgelagert und dafür die gerade benötigten Teile des anderen Programms in den Arbeitsspeicher übernommen werden. Dazu denkt man sich den Arbeitsspeicher in Teile von 4096 Bytes = 1 Seite unterteilt. Dies ist die kleinste Einheit, die zwischen externen Speichern und dem Arbeitsspeicher übertragbar ist. Dieser stark vereinfacht geschilderte Vorgang heißt „Seitenwechsel" oder „paging". Hier wird theoretisch mehr physikalischer Arbeitsspeicherraum für die gleichzeitige Abwicklung bereitgestellt als überhaupt vorhanden. Dieser theoretisch beliebig große Arbeitsspeicher wird auch als „virtueller Speicher" bezeichnet. Der Inhalt des gesamten virtuellen Speichers befindet sich auf einem externen Speicher. In den physikalischen Arbeitsspeicher werden also nur die Seiten des virtuellen Speichers übertragen, die die gerade benötigten Befehle oder Operanden des laufenden Programms enthalten. Die Durchführung des Seitenwechsels steuert das Betriebssystem, das auch weiß, wo welche Seiten gespeichert wurden und die Umrechnung der virtuellen Adresse in echte Arbeitsspeicheradressen bewirkt. Der Anwender braucht sich darum nicht zu kümmern. (Bei der Datenverarbeitungsanlage des Siemens-System 4004/46 stehen z.B. im virtuellen Speicher 512 Seiten zu 4096 Bytes = 2 Millionen Bytes zur Verfügung, obwohl der Arbeitsspeicher nur 262 kBytes groß ist).

1.3.5.4. Weitere Betriebsarten

Multiprocessing umschreibt einen echten „Simultanbetrieb". Hier werden Programme echt gleichzeitig verarbeitet. In einem Multiprocessing-System sind mehrere Zentraleinheiten oder auch mehrere Rechenwerke zu einem System zusammengefaßt.

Im Gegensatz zur Real-Time-Verarbeitung steht die sogenannte Stapelverarbeitung, auch **„batch-processing"** genannt, wo zunächst alle das Programm betreffenden Daten gesammelt und dann in einem Schub (einem Stapel) verarbeitet werden (vgl. hierzu Abb. 1.45). Real-Zeit-Probleme hängen meist mit Datenfernverarbeitung (Datenübertragung) zusammen, da den Datenendstationen nach den Anfragen sofort die Antworten übermittelt werden.

Auf den Real-Time-Berieb wird im Abschnitt 2.3.4. näher eingegangen.

1.3.5.5. Betriebssysteme

Es ist bereits erwähnt worden, daß die Steuer- und Organisationsprogramme für den Multiprogramming- und Time-Sharing-Betrieb (Teilnehmerbetriebssystem) von der Herstellerfirma der Datenverarbeitungsanlage bereitgestellt werden. Diese Programme sind Bestandteile des sogenannten Betriebssystems. Das Betriebssystem, auch Systemsoftware genannt, bietet dem Benutzer Erleichterungen bei der Programmierung und bei der Bedienung sowie eine bessere Nutzung der Hardware, indem es Voraussetzungen für die Simultanarbeit, für die Datenübertragung etc. schafft. Das Organisationsprogramm muß mit seinem Ablaufteil immer im Kernspeicher vorhanden sein. Die weniger häufig anzusprechenden Teile der Systemprogramme sind auf einem externen Speicher abgelegt und werden von dort nur bei Bedarf

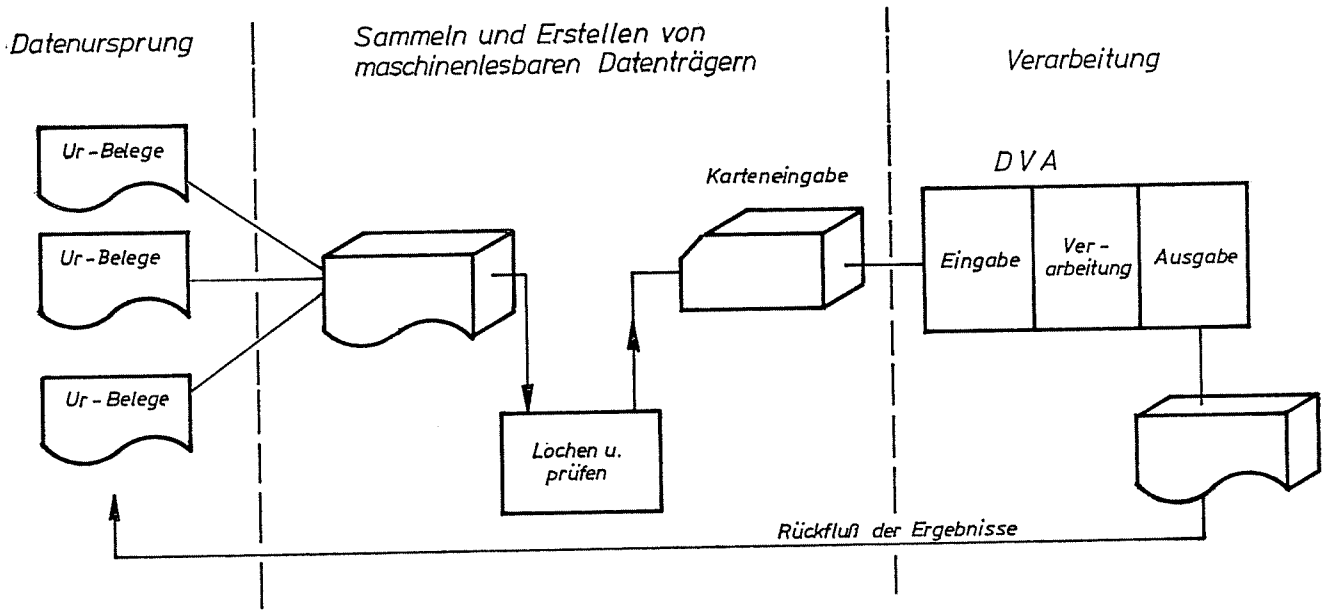


Abb. 1.45 — Schematische Darstellung des Stapelbetriebes

in den Arbeitsspeicher übernommen. Man nennt diese externen Speicher deshalb Systemträger oder System-Residenz; einige Betriebssysteme sind nach der System-Residenz benannt. Man spricht von einem Bandbetriebssystem, wenn der Systemträger ein Magnetband ist und von einem Platten-Betriebssystem, wenn die System-Residenz eine Magnetplatte ist.

Die Auswahl des Betriebssystems muß für jede Anlage stets nach Aufgabenstellung, Anlagenausstattung und Arbeitsspeicherkapazität erfolgen. Grundsätzlich gliedert sich ein Betriebssystem in **Organisationsprogramme, Sprachen-** (Übersetzer) und **Dienstprogramme.**

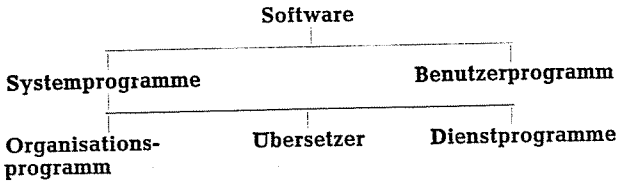


Abb. 1.46 — Gliederung der Software

Die **Organisationsprogramme** (auch Supervisor) beschäftigen sich mit Kontroll- und Steuerungsaufgaben, so daß man sie im wesentlichen als eine Erweiterung des Steuerwerks ansehen kann. Der wichtigste Teil des Organisationsprogramms ist der **Ablaufteil** (Executive), der immer im Arbeitsspeicher resident sein muß.

Der Ablaufteil hat weiter folgende wichtige Programmteile:

den Programm-Steuerungsteil,
den Ein-Ausgabeteil und
den Bedienungsteil.

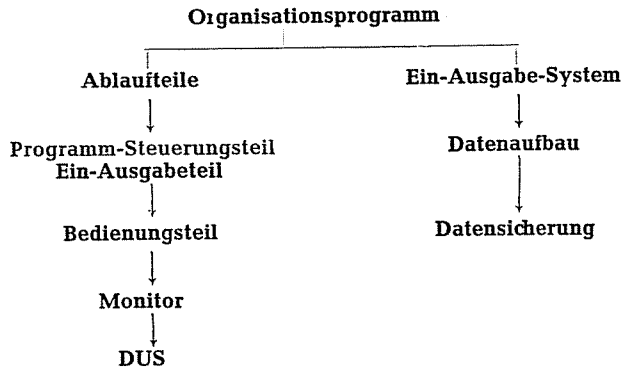


Abb. 1.47 — Unterteilung „Organisationsprogramm"

Der Programm-Steuerungsteil übernimmt das Aufsuchen des Programms in der Bibliothek, das Laden des Programms und seine Überwachung. Der Ein-Ausgabeteil übernimmt die Weitergabe der Befehle an die E/A-Geräte und von diesen Geräten, den Aufbau und das Abarbeiten von Warteschlangen an den Kanälen sowie die Fehlererkennung bei E/A-Operationen. Der Bedienungsteil übernimmt den Dialog zwischen dem System, den Programmen und dem Operateur über den Bedienungsblattschreiber (Auswerten und Entgegennehmen von Antworten und Aufrufen). Der Monitor regelt den Ablauf von nacheinander folgenden Programmen. Ein gewünschter Ablauf kann mittels Steuerkarten vorbereitet werden. Wenn Datenfernverarbeitung durchgeführt werden soll, muß der Ablaufteil noch um ein Datenübertragungssystem erweitert werden. Dieses Programmteil steht ebenfalls ständig im Arbeitsspeicher und übernimmt den Aufruf der Endstation, Annahme und Puffern von Meldungen, Codewandlungen, Fehlerbehandlung und Unterbrechungsbehandlung. Das Datenübertragungssystem bildet die Schnittstelle zwischen der Fernverarbeitungsperipherie und dem Benutzerprogramm, das immer höchste Priorität haben muß.

Das **Ein-Ausgabe-System** (EAS) des Organisationsprogramms verwaltet die Daten (Dateien). Auf den externen Speichermedien sind verschiedene Speicherungsformen möglich. Das EAS stellt die Verbindungen her vom System zu den

extern gespeicherten Daten über spezielle Zugriffsmethoden. Bei Lochkarten-, Drucker- und Magnetbanddateien treten keine großen Organisationsprobleme auf. Hier kann nur seriell (starr fortlaufend), z.B. eine Karte nach der anderen oder eine Zeile nach der anderen, verarbeitet werden. Das EAS übernimmt den Dateiaufbau (ungeblockt, geblockte Sätze, feste Länge der Sätze, variable Länge der Sätze) und die Datensicherung. Hierzu werden die Dateien mit Etiketten versehen. Solche Kennsätze verhindern die irrtümliche Verwendung und das Überschreiben von Dateien.

Bei Großspeicherdateien (Magnetplatte, Magnettrommel, Magnetkarte) sind verschiedene Speicherungsformen möglich. Neben der seriellen (starr fortlaufend) kann auch die sequentielle (logisch fortlaufend) Abspeicherung gewählt werden. Da die Großspeicher den direkten Zugriff (random access) auf Daten ermöglichen (vgl. hierzu auch Abschn. 2.3.4.), kann man auch gestreut abspeichern. Beim Aufbau der Datei braucht keine Reihenfolge beachtet zu werden. Der Speicherplatz wird durch direkte oder indirekte Adressierung errechnet. Bei direkter Adressierung besteht ein Zusammenhang zwischen Ordnungsbegriff und Speicheradresse (Ordnungsbegriff → Speicherplatz und auch Speicherplatz → Ordnungsbegriff). Bei der indirekten Adressierung besteht dieser Zusammenhang insofern nicht, als eine Rückrechnung vom Speicherplatz zum Ordnungsbegriff nicht möglich ist. Eine Kombination ist bei der gestreuten und logischen Speicherung möglich, wenn für den direkten Zugriff mehrere Indexleisten, nach verschiedenen Ordnungsbegriffen sortiert, die Adresse der Datensätze enthalten (index-sequentiell). Die Struktur der Daten und das Ändern bzw. Hinzufügen von Datenelementen bringen erheblichen Organisationsaufwand. Dazu bietet das EAS eine weitgehende Unterstützung an (z.B. Aufbau und Änderungen der Dateien und Indextabellen, Lesen und Schreiben in Dateien, Zuweisen von Ersatzspuren, Fehlererkennung).

Als weiterer Hauptbestandteil des Betriebssystems sind die **Übersetzerprogramme** zu nennen. Am Beispiel des Platten-Betriebssystems für eine Siemens 4004/45 können dies sein: Assembler, FORTRAN, ALGOL, COBOL und Listenprogramm-Generator. Diese Übersetzer sind zur Überführung der in symbolischen Programmiersrpachen abgefaßten Primärprogramme in ausführbare Maschinensprache erforderlich.

Bei den modernen Datenverarbeitungsanlagen wird die Programmierung in Bausteinen durchgeführt (Modultechnik). Größere Aufgaben werden in Teilprobleme aufgegliedert, so daß mehrere Programmierer eingesetzt werden können.Jeder Programmteil kann getrennt übersetzt und formal ausgetestet werden. Dabei kann jeder Teil auch in einer anderen Programmiersprache abgefaßt werden. Die Übersetzungsprogramme erzeugen alle Module mit gleichen Aufbau. Aus der Zahl der Module wird mit Hilfe eines Binderprogramms ein ablauffähiges Maschinenprogramm erzeugt. Über den Binderlauf hat man Einfluß auf die Struktur des Maschinenprogramms. Für mehrere Programmteile kann z.B. ein gemeinsamer Speicherbereich definiert werden.

Der dritte wesentliche Bestandteil des Betriebssystems sind die **Dienstprogramme.** Hierunter fallen die Bibliotheksverwaltung, Umsetzprogramme, Überwachungs- und Diagnoseprogramme als Testhilfen und der Sortier-Misch-Generator. Aus der vorhin angedeuteten Programmiertechnik geht hervor, daß man die Programmbibliothek einteilen kann nach Primärprogrammen, nach Modul- und Makrobibliothek und nach ablauffähigen Programmen (Phasen). Die Bibliotheksverwaltungsprogramme helfen beim Neuaufnehmen, Löschen und Ändern in den verschiedenen Bibliotheken. Die Bibliothek selbst wird auf der System-Residenz als Speichermedium realisiert. Die Umsetzung von Daten von einem Datenträger auf einen anderen ist eine in jedem Rechenzentrum häufig auftretende Tätigkeit. Dafür stellt das Betriebssystem eine Vielzahl von Umsetzprogrammen zur Verfügung (z.B. Karte → Band, Band → Platte usw.).

Für das Sortieren (Sätze einer Datei werden geordnet) und das Mischen (mehrere sortierte Dateien werden zu einer einzigen zusammengefaßt) ist innerhalb des Betriebssystems ein besonderer Sortier-Misch-Generator bereitgestellt. Der Benutzer braucht nur mit Hilfe besonderer Steuerkarten die Daten über den Sortierbegriff und die Sortierfolge anzugeben. Der Sortiervorgang wird unter bestmöglicher Nutzung der Anlagenkapazität ausgeführt.

Komfortable Betriebssysteme lassen sich nur bei großen Anlagen einsetzen, weil die ständig residenten Programmteile infolge der umfangreichen Aufgaben viel Speicherplatz belegen (≈ 10 KByte). Für kleinere Anlagen können, wegen des hohen Speicherbedarfs, nur Monitor- oder Basis-Betriebssysteme gewählt werden. Sie haben keine Überwachungs- und Kontrollsysteme für die Simultanarbeit und gestatten nur die automatische Durchführung von Arbeiten in einer vorgegebenen Reihenfolge. Sie helfen Rüstzeiten zu verkürzen und nehmen dem Operateur das umständliche Laden und Starten der Programme ab.

1.3.6. Einsatzplanung im kommerziellen Bereich

Rechner werden auf wissenschaftlich-technischem Gebiet und für die kommerzielle Anwendung eingesetzt. Bei wissenschaftlichen Vorgängen findet meist nur reiner Programmbetrieb statt, d.h., eine Vielzahl von Programmen müssen übersetzt, geladen und abgearbeitet werden. In der kommerziellen Anwendung steht oft die Automatisierung von Verwaltungsarbeit im Vordergrund. Die Einplanung einer Datenverarbeitungsanlage in eine bestehende Organisation erfordert umfangreiche Vorarbeit. In der Regel lassen sich diese Arbeiten nach ihrem chronologischen Ablauf wie folgt aufteilen:

1. **Ist-Aufnahme (Untersuchung)** Bereichsabgrenzung, Sachverhalte
2. **Analysekritik (Diagnose)** Vergleich; welche Daten wohin
3. **Soll-Vorschlag (Therapie)** verbessern, vereinfachen
4. **Erproben, Einführen (Behandlung)** vorbereiten, einführen, überwachen.

Damit lassen sich folgende Arbeitsphasen definieren:

1. **Projektierungsphase:** mit Ist-Aufnahme bis Sollvorschlag,
2. **Einsatzvorbereitungsphase:** mit Datenflußplan, Programmablaufplan, Programmierung, Test und Dokumentation
3. **Übernahmephase:** mit Umstellungsarbeiten, Organisationsanweisungen.

In der Projektierungsphase müssen die Aufgabenstellung und die Zielsetzung eindeutig festgelegt werden. Grundlage bildet die sorgfältige und umfassende Analyse der auf die elektronische Datenverarbeitung zu übernehmenden Aufgaben. Die Ist-Aufnahme ist mit den betreffenden Stellen und Sachbearbeitern gemeinsam durchzuführen und schriftlich festzuhalten. Neben der Festlegung des bisherigen Datenflusses ist auch die Sammlung und das Sichten aller Belege, Listen, Formulare und Formblätter erforderlich. Sind die konventionellen Arbeitsgänge analysiert, kann eine kritische Durchleuchtung einsetzen, die z.B. Angaben über die zu verarbeitenden Daten, die nötigen Informationen, den Informationsgehalt von Unterlagen und eine Zusammenfassung von Aufgaben und Daten bringen soll.

Erst die genaue Kenntnis des Ist-Zustandes ermöglicht Vorschläge für den Soll-Ablauf, der zu konzipieren ist. Der Systemanalytiker, der die Ist-Aufnahme und den Soll-Vorschlag ausarbeitet, muß sehr genau über die Einsatzmöglichkeiten einer Datenverarbeitungsanlage Bescheid wissen, damit die gewonnenen Erkenntnisse in ein optimales Organisationssystem mit elektronischer Verarbeitung umgewandelt werden können. Hierbei muß bei der Personalauswahl die Betriebserfahrung höher eingestuft werden als die Kenntnis über die Datenverarbeitung. Es muß Gruppenarbeit (teamwork) einsetzen, weil einzelne Sachbearbeiter durch die komplexen Probleme überfordert würden. Optimale organisatorische Lösungen ergeben sich meist erst nach Diskussionen innerhalb der Gruppe und zwischen den Projektgruppen (die jeweils nur für ein Arbeitsgebiet eingesetzt sind).

Wichtigste Punkte, die bei der Aufstellung von Sollvorschlägen beachtet werden müssen:

a) für den sinnvollen Einsatz einer Datenverarbeitungsanlage sollte eine umfassende Neuorganisation im Rahmen einer Gesamtplanung vorgesehen werden,

b) es sollte nicht versucht werden, den herkömmlichen Ablauf auf die Datenverarbeitungsanlage zu übernehmen (Kartei = Datei) und

c) beim Entwurf des Sollvorschlags ist auf eine schrittweise Einführung Rücksicht zu nehmen.

In der Einsatzvorbereitungsphase werden die Aufgaben des Sollvorschlages in Programme umgesetzt. Dabei soll zwischen den einzelnen Phasen keine längere Unterbrechung vorkommen, weil durch Änderungen in der Ausgangssituation eine neue Ist-Aufnahme notwendig werden könnte. Man unterscheidet in der Einsatzvorbereitungsphase folgende Stufen:

Datenfluß- und Programmablaufplan,

Programmierung,

Test,

Dokumentaion mit Programmbeschreibung

und

Bedienungsanleitung.

Unter anderen Gesichtspunkten sind diese Stufen bereits im Abschn. 1.3.4.1. ausführlich beschrieben. Vor der Programmierung sind zusätzlich gewisse Rahmenvorschriften auszuarbeiten, die Programmaufbau-, Sicherheits- und Kontrollmaßnahmen generell beschreiben und regeln; hierbei ist auch die zu verwendende Programmiersprache festzulegen. Im allgemeinen wird man für Programme, die keinen zeitlich optimalen Ablauf haben müssen, den Einsatz von problemorientierten Programmiersprachen (COBOL, FORTRAN) vorziehen, weil der Programmier- und Testaufwand klein bleibt.

Es muß gewährleistet sein, daß die Arbeiten der technischen Planung, die Installation des Rechenzentrums und die Festlegung des erforderlichen Personalbedarfs sowie dessen Ausbildung gleichzeitig vorangetrieben werden, damit keine unnötigen Verzögerungen eintreten. Die Übernahme muß sorgfältig vorbereitet sein. Meistens wird nur die Übernahme von Teilbereichen erfolgen, bis nach und nach das Gesamtsystem aufgebaut werden kann. Die Übernahme beginnt praktisch mit der Übergabe der Programme von

der Programmiergruppe an das Rechenzentrum. Die während der Einsatzzeit der Programme notwendigen Änderungen und Anpassungen werden jetzt in der Regel zentral durchgeführt (Programmpflege). Bei der Durchführung einer Einsatzvorbereitung für komplexe, integrierte Datenverarbeitungssysteme ist für den gezielten Ablauf aller Tätigkeiten innerhalb der einzelnen Phasen oft auch eine zeitplanmäßige Überwachung notwendig (Netzplantechnik).

1.4. Größenklassifizierung von Datenverarbeitungsanlagen

Das große Angebot an Computern ist eigentlich nur noch überschaubar, wenn eine Klassifizierung nach Größen gefunden wird. Jedoch ist es schwierig, eine einheitliche Einstufung durchzuführen, da es kein entscheidendes Abgrenzungsmerkmal gibt. Vielmehr muß man aus einer Reihe von Merkmalen die Einstufung einer Anlage ableiten. Dabei ergibt sich von selbst, daß die Grenzen zwischen den einzelnen Stufen fließend sein müssen. An Merkmalen, die für die Beurteilung von elektronischen Datenverarbeitungsanlagen wichtig sind, gelten:

1. **Arbeitsgeschwindigkeit**

   Im wesentlichen wird die Arbeitsgeschwindigkeit einer Zentraleinheit durch die Zykluszeit des Arbeitsspeichers und die Geschwindigkeit des Rechenwerkes bestimmt. Ein Arbeitsgang in einem Kernspeicher — ein Zyklus — umfaßt immer einen Lese- und Schreibvorgang. Die Zeit, die ein Speicher für einen vollständigen Arbeitsgang benötigt, ist die Zykluszeit. Je kürzer die Zykluszeit, um so schneller kann eine Information in den Speicher geschrieben oder ausgelesen werden. Die Schnelligkeit des Rechenwerks ist abhängig von seinem technischen Aufbau.

2. **Leistungsfähigkeit**

   Die Leistungsfähigkeit einer Zentraleinheit wird maßgebend bestimmt durch den Befehlsvorrat sowie die Leistungsfähigkeit der Befehle und der Ein- und Ausgabetechnik. Hierzu gehört auch die Kernspeichergröße, die entscheidend ist für die Möglichkeiten zur Simultanverarbeitung.

3. **Verarbeitungsmöglichkeiten**

   Sind Multiprogramming, Datenfernverarbeitung und Time-Sharing-Betrieb vorhanden? Damit wird ein weiteres Merkmal angesprochen, weil für diese Betriebsarten eine angepaßte Peripherie vorhanden sein muß.

4. **Möglichkeiten des Anschlusses von peripheren Geräten**

   Da für die rationelle Bearbeitung einer Aufgabe (im kaufmännischen Bereich) die anschließbare Peripherie (z.B. Belegverarbeitung) von ausschlaggebender Bedeutung ist, müssen die Anschlußmöglichkeiten zur Beurteilung herangezogen werden.

5. **Software**

   Als weiteres Kriterium ist die Software mit System- und Anwender-Software zu sehen. Die programmierte Lösung eines Problems wird um so vorteilhafter, je mehr fertige Programmkomplexe vorliegen. Die möglichen Anwendungsformen einer Datenverarbeitungsanlage resultieren aus der Güte der zur Verfügung stehenden System-Software (Betriebssystem).

6. **Kauf- bzw. Mietpreis**

   Als letztes Abgrenzungsmerkmal muß der Miet- oder Kaufpreis genannt werden. Im allgemeinen überwiegt bei größeren Rechnern das Mietsystem. In der Regel entspricht der Kaufpreis etwa dem 50fachen der Monatsmiete. Bisher sind hier Hard- und Software im Preis inbegriffen. Vereinzelt wird jedoch auch die getrennte Preisangabe praktiziert (Unbundling).

Für die Kassifizierung von elektronischen Datenverarbeitungsanlagen wird in der Literatur folgende Einteilung vorgenommen:

a) **elektronische Tischrechenmaschinen**

b) **Kleinstcomputer, elektronische Abrechnungsautomaten**

c) **Kleincomputer**

d) **mittlere elektronische Rechenanlagen**

e) **Großrechenanlagen**

Unter dem Begriff der „mittleren Datentechnik" werden in etwa die Kleinst- und Kleincomputer sowie Abrechnungsautomaten zusammengefaßt.

Abb. 1.48 enthält eine Tabelle mit den wichtigsten Merkmalen zur Klassifizierung.

	Merkmale zur Klassifizierung					
Elektronische Tischrechner	Prinzipiell arbeiten alle Klassen nach den gleichen Grundsätzen. Die Aufteilung in die Elemente: Ein-/Ausgabe, Steuerwerk, Rechenwerk, Speicherwerk gilt generell	Zusammenfassung durch Überbegriff: Mittlere Datentechnik (MDT)	**Rechengeschwindigkeit:** ∼ ms 4 Grundrechnungsarten In der Regel kein variables Programm	**Eingabe:** Tastatur **Ausgabe:** Anzeigen (Displays) in der Regel keine weitere Peripherie	Kaufpreis: ca. 15 000,— DM	
Kleinstcomputer			**Rechengeschwindigkeit:** im Bereich ms - µs Alle Rechenarten Variable Programme Speicherkapazität: ≈ 16 k Stellen	**Eingabe:** Tastatur als Grundausstattung jedoch Anschlußmöglichkeiten für die automatische Eingabe (z.B. Lochstreifen, Lochkarten, Magnetkonten)	Kaufpreis: bis zu 100 000,— DM Mietpreis: bis zu 2 000,— DM pro Monat	
Kleincomputer			**Rechengeschwindigkeit:** ∼ µs Zusätzlich logische Entscheidungen und Verzweigungen evtl. System-Software einsetzbar Speicherkapazität: bis 32 k Stellen	**Ausgabe:** Drucker, Schreibmaschinen jedoch Anschlußmöglichkeiten für die Erstellung von maschinenlesbaren Datenträgern (z.B. Lochstreifen, Lochkarten, Magnetkarten)	Abschlüsse überwiegend auf Mietbasis	Mietpreis: ca. 2000,— bis 12 000,— DM pro Monat
mittlere elektronische Datenverarbeitungsanlage			**Rechengeschwindigkeit:** µs umfangreicher Befehlsvorrat Multiprogramming möglich Speicherpapazität: bis ∼ 64 k Speicherzykluszeit: µs - ms	**Ein-/Ausgabe:** Anschlußmöglichkeiten aller üblichen Peripheriegeräte		Mietpreis: ca. 10 000,— bis 40 000,— DM pro Monat
elektronische Großrechenanlage			**Rechengeschwindigkeit:** bis ns leistungsfähiger Befehlsvorrat Multiprogramming, Time-Sharing, Datenfernverarbeitung usw. Speicherkapazität: ab 64 k Speicherzykluszeit: ∼ ns	**Ein-/Ausgabe:** übliche Peripherie, Sprachausgabe, Bildschirme, Zeichengeräte		Mietpreis: ab 40 000,— DM pro Monat

Abb. 1.48 — Größenklassifizierung von elektronischen Datenverarbeitungsanlagen

2. Aufbau einer EDV-Anlage

2.1. Das Leitwerk einer programmgesteuerten digitalen Datenverarbeitungsanlage

2.1.1. Grundlegende Eigenschaften des Leitwerkes

2.1.1.1. Aufgaben des Leitwerkes

Zur Zentraleinheit einer EDV-Anlage gehören drei Funktionseinheiten: das Leitwerk (auch Steuerwerk oder Kommandowerk genannt), der Zentralspeicher und das Rechenwerk. Über Busleitungssysteme sind alle drei Funktionseinheiten miteinander verbunden. Das Leitwerk ist das eigentliche Kernstück einer EDV-Anlage. Sämtliche Abläufe in der Zentraleinheit und ihr Zusammenspiel werden vom Leitwerk gesteuert. Nach DIN 44300 hat die Funktionseinheit „Leitwerk" etwa folgende Bedeutung: **Das Leitwerk steuert die Reihenfolge, in der die Befehle eines Programms ausgeführt werden; es entschlüsselt die Befehle, wobei diese gegebenenfalls modifiziert werden, und es gibt die zur Ausführung notwendigen digitalen Signale ab.** Um also den Aufbau und die Aufgaben eines Leitwerkes zu verstehen, muß der Arbeitsablauf in der Maschine bei der Verarbeitung von Daten nach einem Programm genau bekannt sein. An dem Modellbild einer Zentraleinheit (Abb. 2.1) soll der Arbeitsablauf erklärt werden.

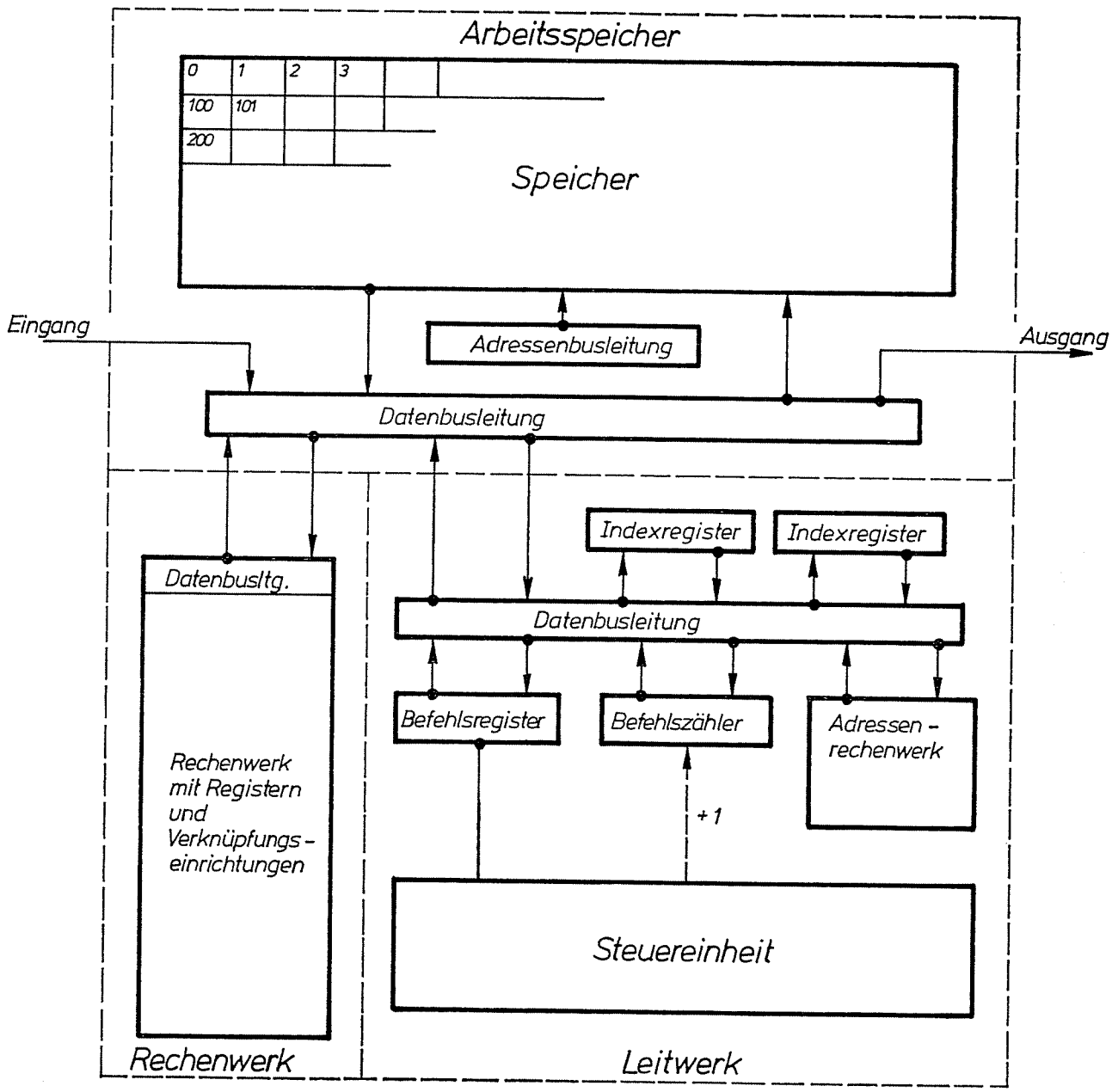


Abb. 2.1 — Modellbild einer Zentraleinheit

Im Zentralspeicher stehen die zu verarbeitenden Daten und das Programm, welches genau angibt, wie diese Daten zu verarbeiten sind. Das Programm ist eine Aneinanderreihung von Befehlen, die von der Maschine nacheinander auszuführen sind. Diese Befehlskette kann nur durch besondere Befehle unterbrochen werden, z.B. durch Sprungbefehle. Die Befehle der Befehlskette befinden sich normalerweise in der notwendigen Reihenfolge ihrer Verarbeitung in Speicherplätzen mit aufeinanderfolgenden Adressen. Steht z.B. der elfte Befehl des Programms in dem Speicherplatz mit der Adresse 212, so ist der zwölfte Befehl in dem Speicherplatz mit der Adresse 213 abgespeichert.

Das Leitwerk muß nun die Befehle in der richtigen Reihenfolge aus dem Speicher holen, sie für die Dauer der Bearbeitung aufbewahren, ihre Bedeutung entschlüsseln und ihre Ausführung veranlassen und überwachen. Es besitzt hierzu eine Reihe von Speichern, Register genannt. Sie können entweder ein ganzes Befehlswort oder Teile davon aufnehmen. Die beiden wichtigsten sind das **Befehlsregister** und der **Befehlszähler.** Aufgabe des Befehlsregisters ist es, den aus dem Speicher über Datenbusleitungen zum Leitwerk transportierten Befehl während seiner Ausführung aufzubewahren.

Unter einer Datenbusleitung versteht man Vielfachleitungen, in die Informationsquellen digitale Daten einspeisen und über die mehrere Adressaten die Daten erhalten können.

Der Inhalt des Befehlsregisters wird zum Teil der Steuereinheit übermittelt, z.B. der Operationsteil und gegebenenfalls auch eine Angabe über die Länge der zu verarbeitenden Daten. Der Adreßteil des Befehls dagegen kann über die Adressenbusleitung wieder dem Speicher zugeleitet werden, denn er enthält eine Angabe, wo die Operanden sich befinden, mit denen gearbeitet werden soll. Nachfolgend soll der Arbeitsablauf einer Einadreßmaschine beschrieben werden; der eine Operand einer durchzuführenden Rechenoperation befindet sich in einem Register des Rechenwerkes, z.B. im Akkumulator. Um die Befehle in der durch das Programm vorgeschriebenen Reihenfolge aus dem Speicher holen zu können, muß die Adresse des jeweils nächsten Befehls bekannt sein. Diese Adresse entnimmt das Leitwerk dem Befehlszähler. Da nun die Befehle im Speicher an Speicherplätzen mit aufeinanderfolgenden Adressen abgespeichert sind, braucht nur am Beginn eines Programms der Befehlszähler mit der richtigen Adresse geladen zu werden. Nach Abarbeitung eines Befehls wird der Inhalt des Befehlszählers um eins erhöht und enthält damit automatisch die Adresse des nächsten Befehls. Anders ist dies bei auszuführenden Sprungbefehlen. Hier findet, durch die Steuereinheit veranlaßt, ein Transport der Sprungadresse aus dem Befehlsregister zum Befehlszähler statt. Die Adresse des nächsten auszuführenden Befehls ist damit ebenfalls wieder im Befehlszähler zu finden. **Beim Ausführen eines Befehls durchläuft das Leitwerk eine Reihe von Befehlsphasen.**

2.1.1.2. Befehlsphasen einer Einadreßmaschine

Wir gehen davon aus, daß gerade ein Befehl abgearbeitet ist; im Befehlsregister befindet sich also der alte, jetzt ungültige Befehl, im Befehlszähler befindet sich die Adresse des neuen Befehls. Die einzelnen Befehlsphasen werden von der Steuereinheit nacheinander ausgeführt.

1. Phase: Die Adresse des neuen Befehls wird aus dem Befehlszähler dem Speicher übermittelt.

2. Phase: Der Zentralspeicher ermittelt den adressierten Speicherplatz. Der Inhalt wird gelesen und zum Befehlsregister transportiert. Nach der Phase 2 befindet sich der neue Befehl also in dem Befehlsregister.

3. Phase: In der Befehlsphase 3 werden eventuell notwendige Adressenmodifikationen durchgeführt. Ob eine solche Adressenänderung notwendig ist und auf welche Weise sie durchgeführt werden soll, erkennt die Steuereinheit an dem Aufbau des Befehls. Derartige Modifikationen sind aus vielen Gründen fast immer erforderlich. **Man unterscheidet bei einer Adressenmodifikation zwischen einer Indexmodifikation und einer Adressensubstitution.**

Bei einer **Indexmodifikation** ergibt sich die endgültige Adresse durch Addition der im Befehl angegebenen Adresse und dem Inhalt eines sogenannten Indexregisters. Meistens sind mehrere Indexregister vorhanden; der Aufbau des Befehls läßt dann erkennen, welches Indexregister zur Adressenänderung herangezogen werden soll. Im Modellbild einer Zentraleinheit (Abb. 2.1) sind zwei derartige Indexregister angedeutet. Häufig besitzt das Leitwerk für diese auszuführende Addition ein besonderes Adressenrechenwerk. Die durch die Indexmodifikation ermittelte Adresse wird dann als endgültige Adresse dem Speicher übermittelt.

Bei einer **Adressensubstitution** handelt es sich um den Austausch des Adreßteils eines Befehls.

Die endgültige Adresse, auch effektive Adresse genannt, ist als Inhalt eines Speicherplatzes im Speicher vorhanden. Im Befehl wird also bei einer Adressensubstitution nur die Adresse des Speicherplatzes angegeben, in dem sich die endgültige Adresse als Inhalt befindet.

Mehrere Adressenmodifikationen können nacheinander erfolgen, also z.B. nach einer Adressensubstitution eine Indexmodifikation. Das Leitwerk prüft also nach jeder durchgeführten Adressenrechnung, ob eine weitere Modifikation notwendig ist.

4. Phase: In der Befehlsphase 4 wird durch die Steuereinheit geprüft, ob der auszuführende Befehl ein Sprungbefehl ist. Ist dies der Fall, so wird weiter geprüft, ob es sich um einen unbedingten oder einen bedingten Sprungbefehl handelt. Bei einem bedingten Sprungbefehl muß zusätzlich geprüft werden, ob die Sprungbedingung erfüllt ist. Ist die Sprungbedingung erfüllt oder liegt ein unbedingter Sprungbefehl vor, so soll die Befehlskette an dieser Stelle verlassen werden und an anderer Stelle wieder neu anfangen. Die Anfangsadresse der neuen Befehlskette, d.h. die Adresse, wohin gesprungen werden soll und wo sich der nächste Befehl befindet, ist die im Adreßteil des Befehls angegebene Adresse. Es ist also erforderlich, daß diese Adresse in den Befehlszähler überführt wird. Der alte Inhalt des Befehlszählers wird überschrieben und geht verloren. Der Sprungbefehl ist jetzt abgearbeitet. Es kann wieder mit der Befehlsphase 1 begonnen werden. Ist bei einem bedingten Sprungbefehl die Sprungbedingung nicht erfüllt, so soll die Befehlskette nicht verlassen werden. Es wird deshalb sofort zur Befehlsphase 7 übergegangen und der Aufruf des nächsten Befehls vorbereitet. Zeigt sich in der Befehlsphase 4, daß kein Sprungbefehl vorliegt, so wird unmittelbar die Befehlsphase 5 erreicht.

5. Phase: Die Steuereinheit prüft hier, ob ein Stoppbefehl vorliegt, das heißt ein Befehl, der den automatischen Verarbeitungsablauf unterbricht und anhält. Ist dies der Fall, so arbeitet die Maschine im Halt, andernfalls wird zur Befehlsphase 6 übergegangen.

6. Phase: In dieser Befehlsphase wird die eigentliche Operation, die im Befehl angegeben ist, durchgeführt. Es werden Daten transportiert, z.B. vom Zentralspeicher zum Rechenwerk, vom Eingabewerk zum Zentralspeicher, und es werden Daten im Rechenwerk miteinander verknüpft. Ist die Operation beendet, so folgt die Befehlsphase 7.

7. Phase: In dieser Phase wird der Inhalt des Befehlszählers um 1 erhöht. Er enthält dann die Adresse des nächsten Befehls. Die Befehle der Befehlskette stehen — wie schon erwähnt — hintereinander mit aufeinanderfolgenden Adressen im Zentralspeicher.

Auf die Befehlsphase 7 folgen dann wieder die Phasen 1 und 2 mit dem Aufruf des nächsten Befehls.

In Abb. 2.2 sind die einzelnen Befehlsphasen einer Einadreßmaschine bei der Ausführung eines Befehls noch einmal dargestellt.

2.1.1.3. Zusammenfassung der Aufgaben des Leitwerkes und die zur Lösung notwendigen Hauptbestandteile

Es sollen noch einmal die Aufgaben des Leitwerkes zusammengefaßt und die zur Lösung notwendigen Bestandteile angegeben werden. **Das Leitwerk muß die Befehle in der notwendigen Reihenfolge aus dem Speicher holen. Die Adresse des jeweils nächsten Befehls entnimmt es dem Befehlszähler. Der Befehl muß für die Dauer der Operation aufbewahrt werden. Hierzu dient das Befehlsregister. Die Steuereinheit muß den Befehl aufrufen, ihn entschlüsseln und die zur Ausführung notwendigen digitalen Signale erzeugen. Sind im Adreßteil des Befehls Angaben gemacht über notwendige Adressenmodifikationen, so sind diese vor der eigentlichen Operation auszuführen. Hierzu dienen Indexregister und ein eventuell vorhandenes Adressenrechenwerk.**

2.1.2. Aufbau des Leitwerkes

In einer elektronischen Datenverarbeitungsanlage werden die Daten (Zahlen, Buchstaben und Zeichen) sowie die Befehle durch Bitkombinationen dargestellt, d.h. durch Kombinationen von Binärzeichen, die einer Codierung unterliegen. Diese Bitkombinationen sind je nach Maschine unterschiedlich lang und auch der verwendete Code ist nicht immer der gleiche. Man unterscheidet zwischen **dual- und dezimalorganisierten elektronischen Rechenanlagen.** Weiter unterscheidet man zwischen **Wortmaschinen und Stellenmaschinen.** Bei Wortmaschinen ist die Anzahl der Bits eines Wortes immer gleich, bei Stellenmaschinen kann die Wortlänge variieren. Es wird dann Anfang und Ende des Wortes adressiert oder nur der Anfang des Wortes und die Länge des Wortes, also die Zahl der Bits.

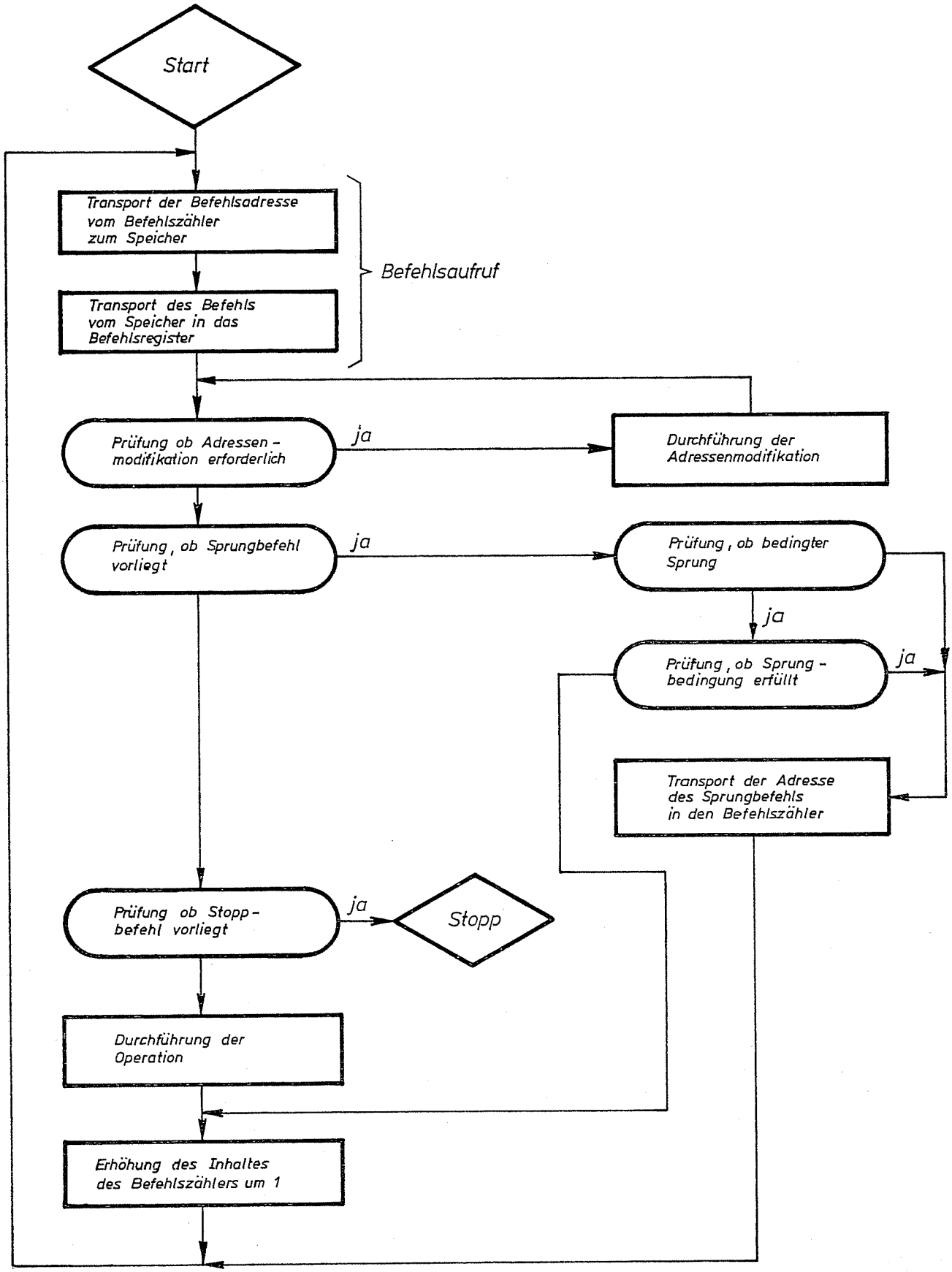


Abb. 2.2 — Befehlsphasen einer Einadreßmaschine

Sollen nun Daten und Befehle in einer EDV-Anlage transportiert und verknüpft werden, so müssen die entsprechenden Bitkombinationen transportiert und verknüpft werden.

Wird die entsprechende Bitkombination eines Wortes in ihrer Gesamtheit auf einmal transportiert oder verarbeitet, so spricht man von Parallelbetrieb. Für jedes Bit des Wortes muß also eine Transportleitung vorgesehen werden bzw. muß für die Verknüpfung das entsprechende Verknüpfungsglied vorhanden sein. **Wird jedoch jedes Bit eines Wortes einzeln und zeitlich nacheinander verarbeitet und transportiert, so spricht man von Serienbetrieb.** Diese Verarbeitung erfordert sehr viel weniger Geräteaufwand als die Parallelverarbeitung, dauert jedoch länger. Es gibt EDV-Anlagen, bei denen beide Betriebsarten gemischt werden, z.B. bei dezimalorganisierten Rechenanlagen. Hier werden die zur Darstellung **einer** Dezimalziffer notwendigen 4 Bits parallel übertragen und verarbeitet, die einzelnen Dezimalstellen jedoch seriell. Man bezeichnet diese Betriebsart dann **„Serienparallelbetrieb".**

Der Aufbau des Leitwerkes ist von den unterschiedlichen Codes und Betriebsarten abhängig. Er ist außerdem sehr komplex und umfangreich, obwohl nur wenige Elementarschaltungen wie Verknüpfungsschaltungen und Flipflops verwendet werden. Nachfolgend werden einige Möglichkeiten aufgezeigt, wie die dem Leitwerk gestellten Aufgaben gelöst werden können.

2.1.2.1. Aufbau von Registern

In einem Register ist für jedes zu speichernde Bit ein Speicherelement vorhanden, hier ein Flipflop. Abb.2.3 zeigt die symbolische Darstellung eines 24-Bit-Registers für parallele Bitübertragung. Durch den Übernahmetakt wird erreicht, daß die Information, die am Paralleleingang des Registers ansteht, in die Flipflops übernommen wird und am Ausgang erscheint. Sie wird gespeichert, bis durch einen neuen Übernahmetakt eine neue Information eingelesen wird. Abb. 2.4 zeigt einen Ausschnitt aus einem solchen Register. Es ist hier aus D-Flipflops aufgebaut. Ein D-Flipflop zeigt nach dem Einwirken eines Taktes immer die Information am Ausgang, die vor Einwirkung

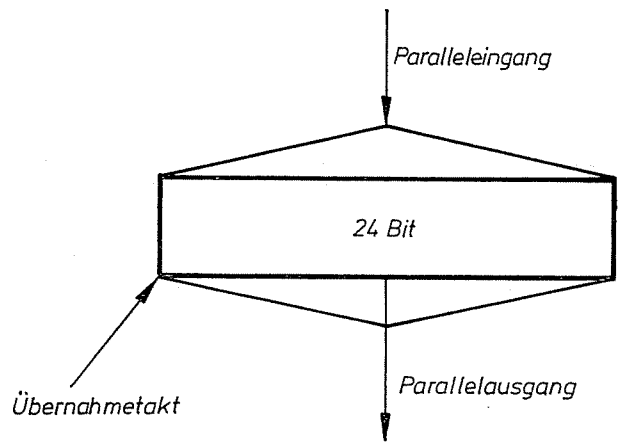


Abb. 2.3 — Symbolische Darstellung eines 24-Bit-Registers

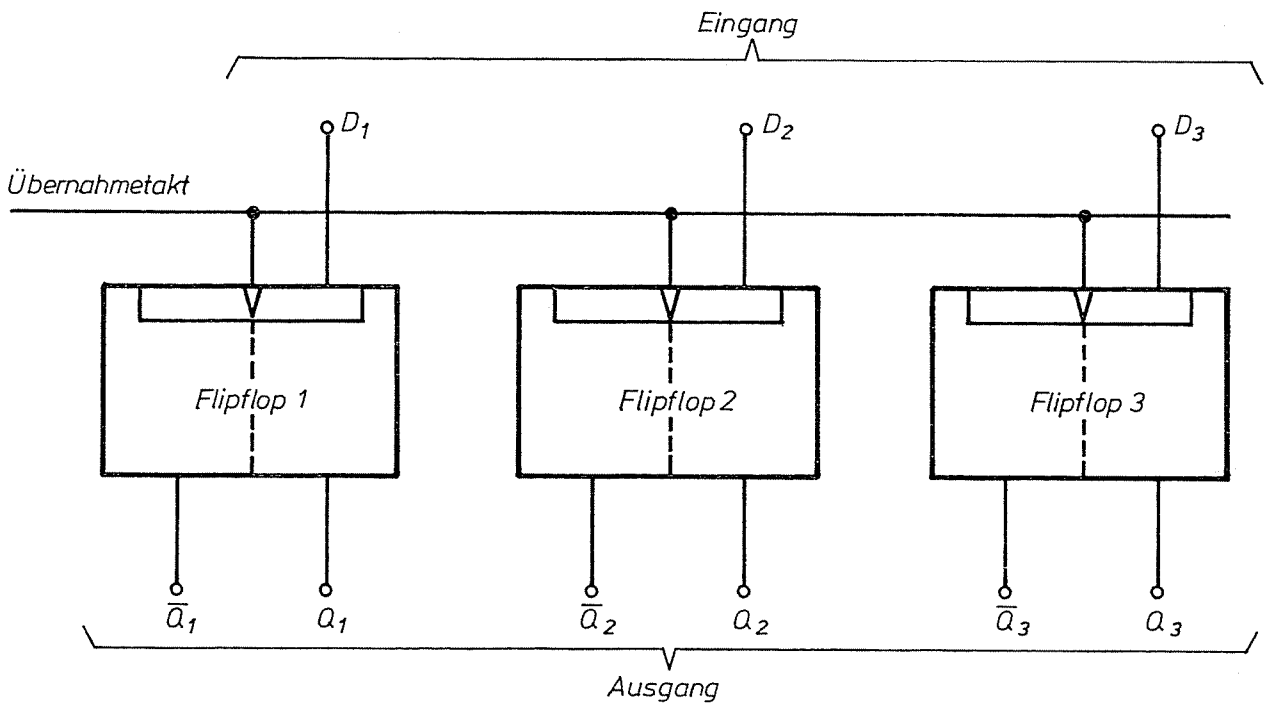


Abb. 2.4 — Ausschnitt aus einem Register mit paralleler Bitübertragung

des Taktes am sogenannten D-Eingang angelegen hat. Bei zeitlich aufeinanderfolgenden Takten erscheint also am Ausgang die Eingangsinformation, verzögert um einen Takt. Aus dieser Tatsache leitet sich der Name D-Flipflop, Delay-Flipflop ab. (Delay bedeutet Verzögerung.) An den D-Eingängen liegt die zu übernehmende Information. Zur Übernahme der Information in das Register wird auf der allen Flipflops gemeinsamen Taktleitung der Übernahmetakt angelegt und die Information eingelesen; die vorher im Register vorhandene Information braucht hierzu nicht gelöscht zu werden. An den Ausgängen der Flipflops kann die Information abgenommen werden.

Die Eingabe, Ausgabe und Speicherung von seriell dargestellten Daten erfordert ein Schieberegister. Hier sind die einzelnen Flipflops so miteinander verbunden, daß die Information jedes einzelnen Speicherelements durch einen Schiebetakt auf das benachbarte Flipflop übertragen wird. Die Information wird gleichsam beim Einlesen in das Register eingeschoben und beim Auslesen wieder herausgeschoben. Bei einem 24-Bit-Register benötigt man 24 Schiebetakte zum Einlesen und ebenfalls 24 Schiebetakte zum Auslesen. Soll die Information beim Auslesen nicht im auszulesenden Register verloren gehen, so wird der Ausgang des Registers auf den Eingang zurückgekoppelt. Ist die Information eingelesen und damit gespeichert, so kann sie auch parallel an den Ausgängen der Registerelemente abgenommen werden. Das Schieberegister ermöglicht also auch eine Serien-parallel-Umsetzung, wie sie z.B. zur Entschlüsselung eines Befehls notwendig wird.

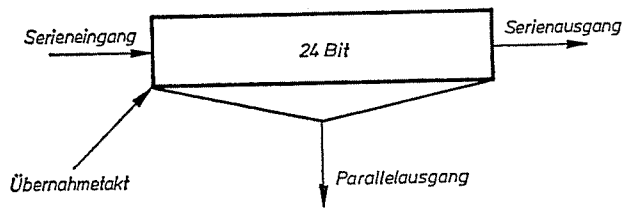


Abb. 2.5 — Symbolische Darstellung eines Registers

Abb. 2.5 zeigt die symbolische Darstellung eines Schieberegisters, Abb. 2.6 einen Ausschnitt aus dem Signalflußplan eines Schieberegisters. Es ist ebenfalls aus D-Flipflops aufgebaut. Beim Auslesen ist die Rückkopplung wirksam, beim Einlesen wird sie abgetrennt.

Um sowohl Serien- als auch Parallelübertragung zu ermöglichen, kann ein Register so geschaltet werden, daß es durch Anlegen einer entsprechenden Steuerinformation sowohl über den Paralleleingang als auch über den Serieneingang gesetzt werden kann. Dies gilt auch für die Ausgabe.

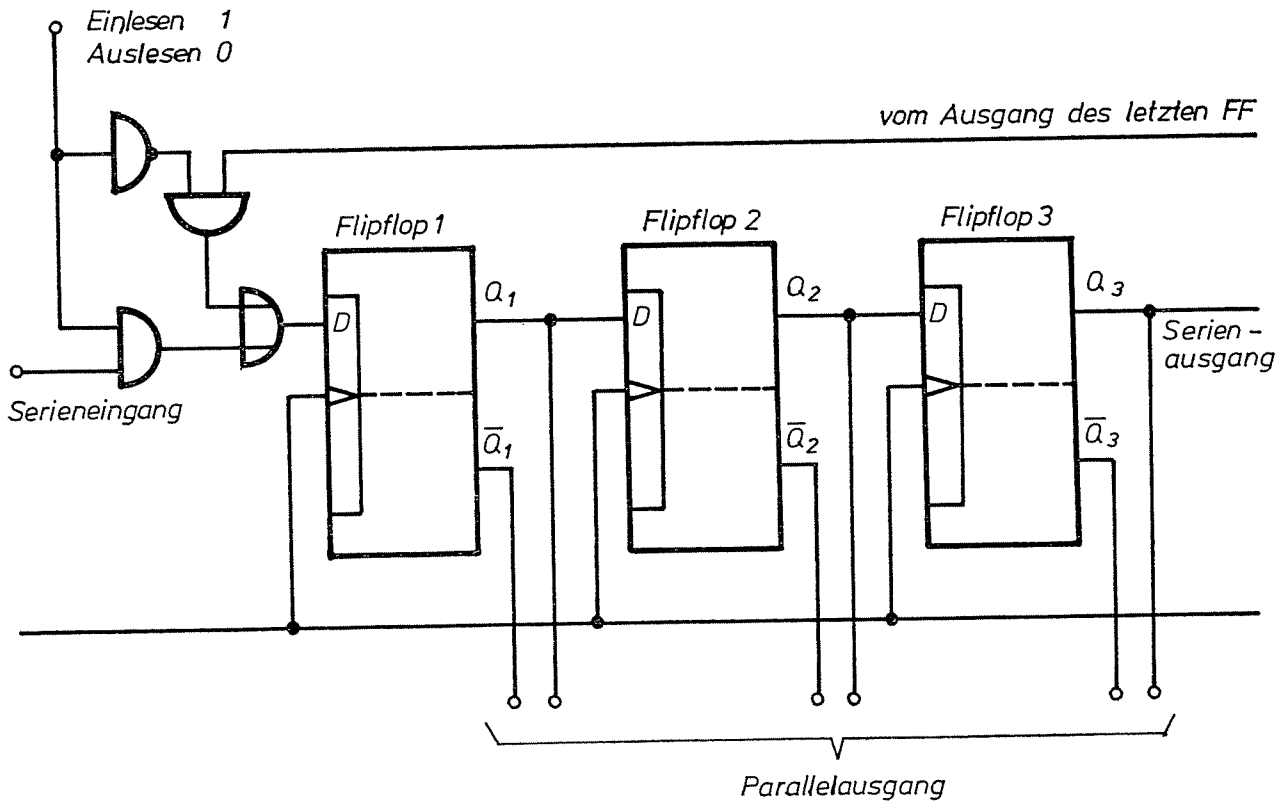


Abb. 2.6 — Ausschnitt aus dem Signalflußplan eines Schieberegisters

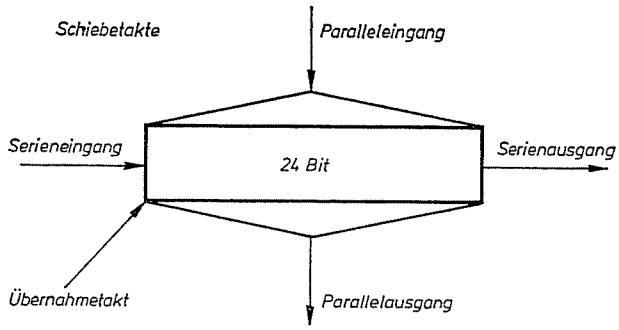


Abb. 2.7 — Symbolische Darstellung eines Registers

Die symbolische Darstellung eines solchen Registers zeigt Abb. 2.7; einen Ausschnitt aus dem Signalflußplan Abb. 2.8. Liegt 1 am Steuereingang an, so wird über den Paralleleingang eingelesen; bei 0 am Steuereingang wird auf ein Schieberegister umgeschaltet. Es werden immer die jeweils richtigen UND-Tore freigegeben.

Die Aufgabe des **Befehlszählers** kann durch einen **Zähler mit Voreinstellung** übernommen werden. Durch die Voreinstellung wird die erste Adresse der Befehlskette in den Zähler übernommen. Zählimpulse, ausgelöst etwa in der Befehlsphase 7, erhöhen den Zählerstand nach jedem Befehl um 1. Der Befehlszähler enthält dann die Adresse des nächsten Befehls. Bei Sprungbefehlen wird der alte Zählerstand gelöscht und durch die Voreinstellung der neue Zählerstand — die neue Befehlsadesse — eingeschrieben.

Das Leitwerk enthält bei vielen Maschinen ein **Adressenrechenwerk,** das die notwendigen Adressenumrechnungen ermöglicht. Dies kann dann sogar unabhängig von einer gerade laufenden Rechenoperation erfolgen. Im Abschnitt 2.2. werden der Aufbau und die Wirkungsweise von Rechenwerken beschrieben, so daß an dieser Stelle auf eine Beschreibung des Adressenrechenwerkes verzichtet werden kann.

2.1.2.2. Aufbau der Steuereinheit

Die Steuereinheit hat die Aufgabe, die einzelnen Befehle zu entschlüsseln und die zur Ausführung notwendigen Signale zu erzeugen. Durch diese Steuersignale werden die einzelnen Teilfunktionen der gesamten Anlage in bestimmter zeitlicher Reihenfolge ausgelöst. Gemeint ist hier z.B. das Öffnen von Torschaltungen, das Erzeugen von Taktfolgen für das Übertragen von Bitkombinationen oder das Auslösen von Triggerimpulsen für das Arbeiten einzelner Funktionseinheiten. Die Steuereinheit muß also feststellen, welche einzelnen, oft in einigen Schritten nacheinander vorzunehmenden Tätigkeiten durch den Befehl aufgerufen werden und diese der Reihe nach auslösen. Damit erhält man im Prinzip eine Ablaufsteuerung. **Die Steuereinheit ist damit ein Automat, der eine Reihe von Zuständen durchläuft, in denen er einzelne Befehle nach außen abgibt.** Durch welche Zustände er läuft, hängt sowohl von den auszuführenden Befehlen als auch von zusätzlichen Bedingungen ab, wie z.B. von dem Ergebnis einer Sprungbe-

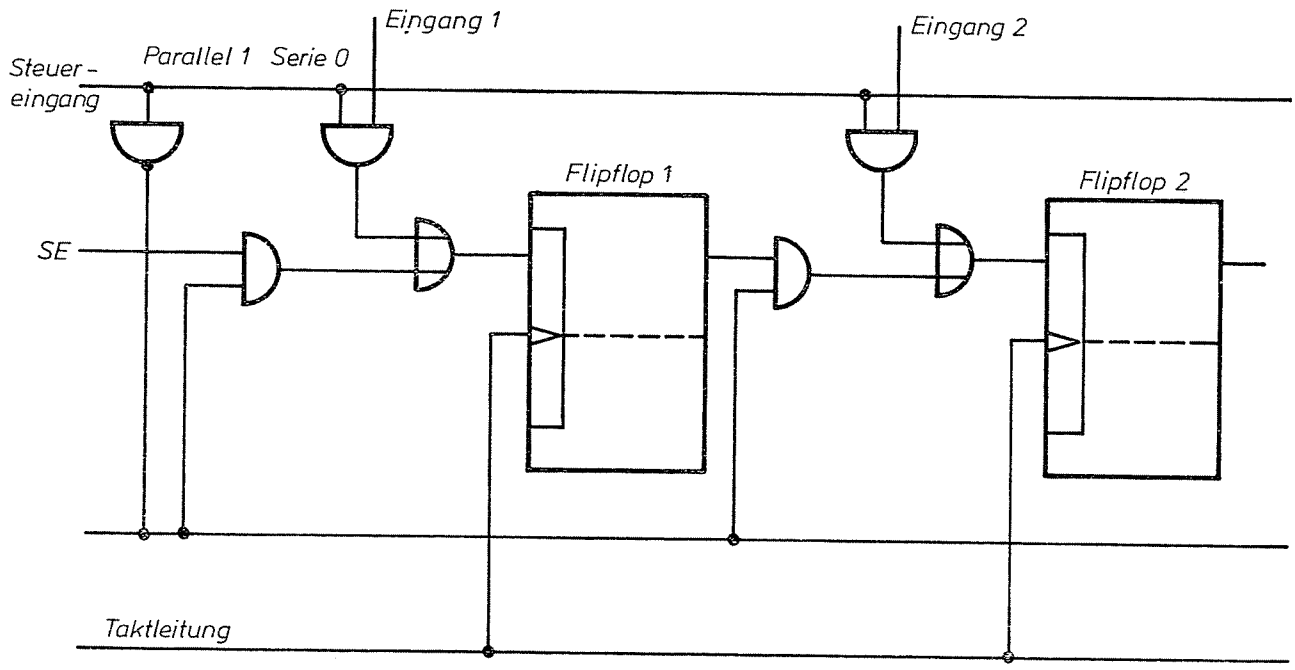


Abb. 2.8 — Ausschnitt aus dem Signalflußplan eine Registers mit paralleler und serieller Bitübertragung

dingung. **Oft sind Ablaufsteuerungen hierarchisch gegliedert.** Es werden dann gewisse Steuertätigkeiten an untergeordnete Steuerwerke delegiert. Der Aufbau der Steuereinheit wird dadurch übersichtlicher, einfacher und wirtschaftlicher.

Eine allgemeine Theorie der Steuerwerke, die es gestattet, alle Steuerwerke systematisch zu betrachten, gibt es nicht. In der Praxis ist der Entwurf von Steuerwerken weitgehend Erfahrungssache; deshalb kann der Aufbau der Steuereinheit nicht allgemein angegeben werden. Einen Ansatz zum systematischen Aufbau von Steuerwerken stellen das **Schaltwerkskonzept** und das **Mikroprogrammsteuerwerk** dar.

Über ein Schaltwerk ist folgendes auszusagen: Es kann mehrere innere Schaltzustände annehfen, die mit $Z_0, Z_1, \ldots, Z_n$ bezeichnet werden. Es empfängt von außen Eingangssignale $E_1 \ldots E_m$ und gibt nach außen Ausgangssignale $A_1 \ldots A_v$ ab. Die Ausgangssignale sind durch die Eingangssignale **und** durch den inneren Schaltzustand bestimmt, in dem sich das Schaltwerk befindet. Der innere Zustand des Schaltwerkes, in den das Schaltwerk nach Einwirken eines Taktes geht, ist durch die Eingangssignale **und** durch den inneren Zustand bestimmt, in dem sich das Schaltwerk vor Eintreffen des Taktes befindet. Die Eingangssignale setzen sich aus dem Operationsteil des gerade laufenden Befehls und den Entscheidungszeichen zusammen. Über Entscheidungszeichen erhält dabei die Steuereinheit ständig Informationen von außen, z.B. vom Rechenwerk oder von den Eingabe- und Ausgabegeräten. Die Ausgangssignale gehen nach außen an die verschiedenen Einheiten des Rechenautomaten. Es soll vorausgesetzt werden, daß das Schaltwerk synchron arbeitet, hierunter versteht man folgendes: Das Schaltwerk braucht immer nur zu gewissen Zeitpunkten gleichen Abstandes betrachtet zu werden. Werden z.B. einflankengesteuerte Flipflops zur Erzielung der einzelnen inneren Zustände verwendet, so ist der zu betrachtende Abtastzeitpunkt die steuernde Flanke des Taktimpulses. Die Abtastzeitpunkte sollen mit $t_n, t_{n+1} \ldots$ bezeichnet werden. Es wird weiter angenommen, daß die verwendeten Verknüpfungsschaltungen keine Signalverzögerungen verursachen. Die Signale an den Eingängen der Flipflops zum Zeitpunkt t_n wirken sich dagegen erst zum Zeitpunkt t_{n+1} aus, also genau verzögert um eine Taktzeit. Unter diesen Voraussetzungen kann ein Schaltwerk durch zwei

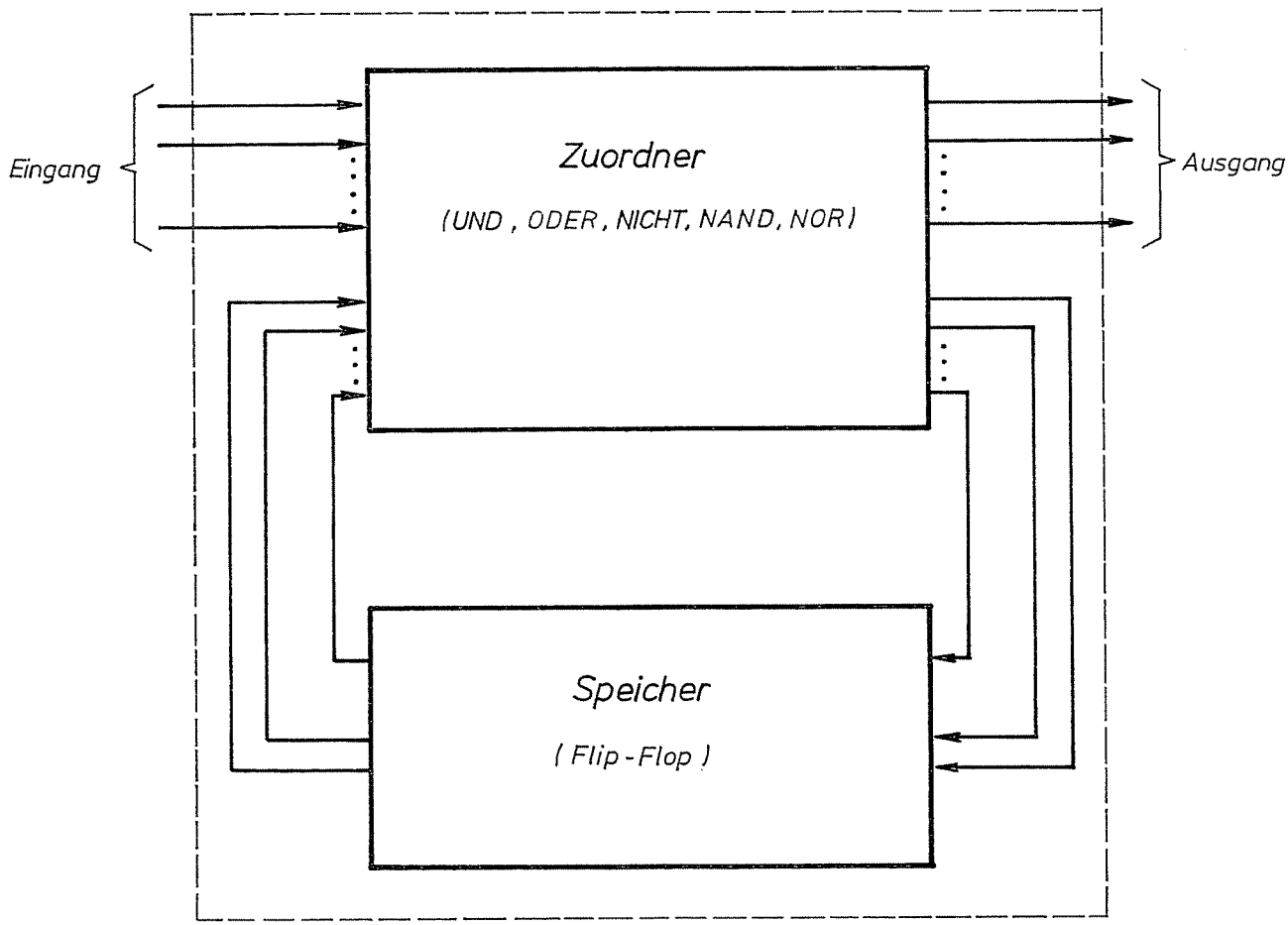


Abb. 2.9 — Prinzip eines Schaltwerkes

mathematische Funktionen der Schaltalgebra beschrieben werden.

$$A(t_n) = f\left[E(t_n)\ ,\ Z(t_n)\right]$$

Die Signale an den Ausgängen des Schaltwerkes zum Zeitpunkt t_n sind also abhängig von den Signalen am Eingang des Schaltwerkes zum Zeitpunkt t_n und von dem inneren Zustand des Schaltwerkes zum Zeitpunkt t_n.

$$Z(t_{n+1}) = \varphi\left[E(t_n)\ ,\ Z(t_n)\right]$$

Der Zustand des Schaltwerkes zum Zeitpunkt t_{n+1}, also nach Eintreffen eines neuen Taktes, ist abhängig von den Eingangssignalen zum Zeitpunkt t_n und von dem Zustand des Schaltwerkes zum Zeitpunkt t_n.

Diese Formeln besagen, daß die Ausgangssignale und der Folgezustand des Schaltwerkes von den Eingangssignalen und von der Vorgeschichte des Schaltwerkes abhängen. Ein Schaltwerk hat also ein Erinnerungsvermögen. Es muß deshalb **Speicher** besitzen, **die durch ihren Inhalt den Zustand des Schaltwerkes repräsentieren. Ein Zuordner,** d.h. ein Schaltnetz aus Verknüpfungsschaltungen, **bewirkt die gewünschte Abhängigkeit zwischen den Ausgangssignalen und dem Folgezustand des Schaltwerkes sowie den Eingangssignalen und dem vorhandenen Zustand des Schaltwerkes.** Das Schaltwerk besteht also aus zwei Teilen: dem Speicher, der mit seinem Inhalt den Zustand des Schaltwerkes angibt und dem Zuordner, der die gewünschte Abhängigkeit bewirkt. Abb. 2.9 zeigt das Prinzip eines Schaltwerkes.

Bei binären Speicherelementen, z.B. bei Flipflops, ist die Mindestanzahl von Speicherelementen durch die Zahl der notwendigen inneren Zustände des Schaltwerkes bestimmt. Besteht der Speicher aus k Flipflops, so können maximal 2^k verschiedene Zustände realisiert werden. Soll der Steuerautomat z.B. 9 innere Zustände durchlaufen, so sind mindestens 4 Flipflops erforderlich. Vier Flipflops ermöglichen insgesamt 16 verschiedene Zustände, drei Flipflops jedoch nur 8 Zustände. **Die Mindestanzahl der Speicherelemente kann selbstverständlich überschritten werden. Der Aufwand an Flipflops wird dann größer, aber der notwendige Aufwand für die Verknüpfungsschaltung nimmt ab.**

Das Verhalten eines Schaltwerkes läßt sich anschaulich durch einen sogenannten **„Graphen"** darstellen. Das ist eine graphische Darstellung des Schaltverhaltens, in der innere Zustände durch kleine Kreise und Übergänge von einem Zustand in einen anderen durch Pfeile dargestellt werden. Die den Übergang bestimmenden Eingangssignale und die abgegebenen Ausgangssignale werden außerdem mit in den Graph eingetragen.

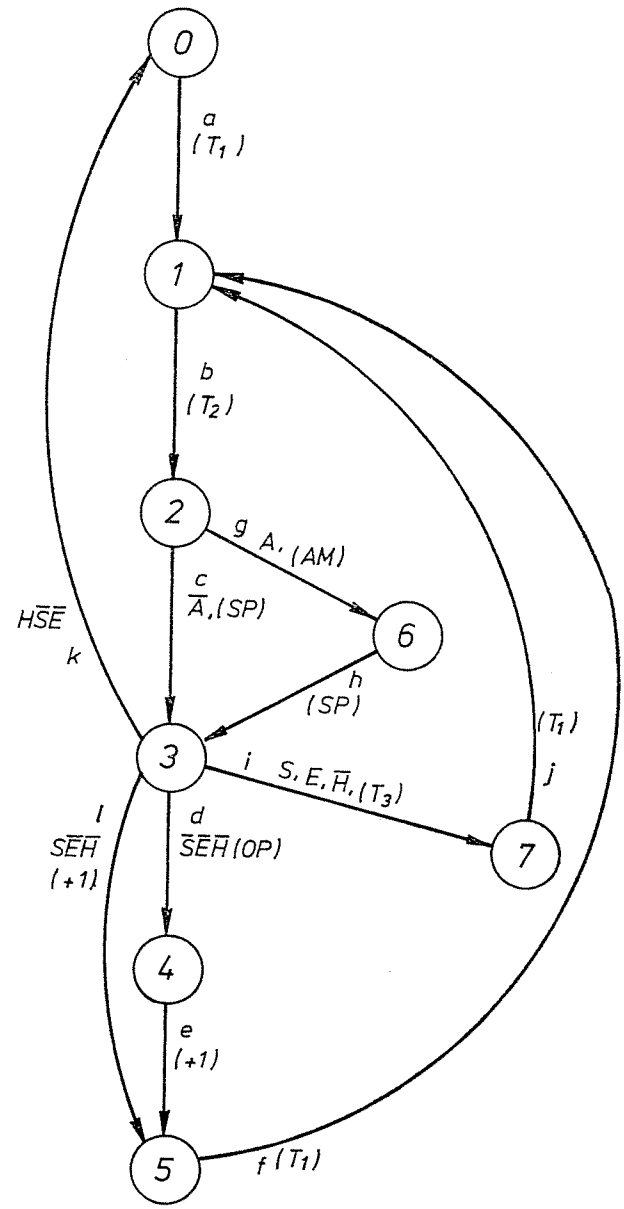


Abb. 2.10 — Graph für ein Schaltwerk zur Erzeugung der einzelnen Befehlsphasen

Als Beispiel für ein Schaltwerk soll eine Ablaufsteuerung beschrieben werden, die z.B. den Ablauf der einzelnen Befehlsphasen eines Befehls ermöglicht. Es gilt der in Abb. 2.10 angegebene Graph, der sich aus Abb. 2.2 ergibt.

Weiter soll folgende Voraussetzung gelten: Ein Taktflipflop T bestimmt durch seine Lage, ob das Steuerwerk von einem Zustand in einen anderen übergehen soll, z.B. erst dann, wenn die gewünschte und eingeleitete Operation beendet ist. Es bestimmt außerdem, ob die Ausgangs-

signale abgegeben werden. Die hierzu notwendigen Steuermittel werden aus Gründen der Übersichtlichkeit nicht beschrieben und angegeben.

Die Zahlen in dem Graphen (Abb. 2.10) bezeichnen die einzelnen Zustände, die kleinen Buchstaben die Übergänge, die großen Buchstaben die Eingangssignale und die großen, eingeklammerten Buchstaben die abzugebenden Ausgangssignale. Der Ruhestand des Schaltwerkes ist der Zustand 0. Wird jetzt das T-Flipflop durch einen Startimpuls gesetzt, so geht das Schaltwerk mit dem nächsten Taktimpuls in den Zustand 1 über. Es wird der Transport der Adresse des nächsten Befehls aus dem Befehlszähler zum Speicher veranlaßt (Übergang a, Ausgangssignal T_1). Ist der Transport ausgeführt, so geht das Schaltwerk in den Zustand 2 über. Der Transport des Befehls vom Speicher in das Befehlsregister wird eingeleitet (Übergang b, Ausgangssignal T_2). Vom Zustand 2 aus gibt es zwei Übergangsmöglichkeiten. Zeigt die Entschlüsselung des Befehls, daß keine Adressenmodifikation notwendig wird (Eingangssignal $\overline{A}$), so wird in den Zustand 3 übergegangen (Übergang c, Ausgangssignal für die Prüfung einer Sprungbedingung SP). Liegt dagegen eine Aufforderung zur Adressenmodifikation vor (Eingangssignal A), so wird als nächster Zustand der Zustand 6 erreicht (Übergang g, Ausgangssignal AM). Während des Zustandes 3 wird geprüft, ob ein Sprungbefehl vorliegt und ob eine eventuell verlangte Bedingung erfüllt ist. In diesem Zustand soll außerdem entschieden werden, ob ein Stoppbefehl vorliegt und die Maschine damit in den Ruhestand 0 übergeht. Liegt weder ein Sprungbefehl noch ein Stoppbefehl vor, Eingangssignal $\overline{S}$ und $\overline{H}$ und $\overline{E}$, so geht das Schaltwerk in den Zustand 4 über. Hier wird die verlangte Operation durchgeführt (Übergang d, Ausgangssignal OP). Liegt ein Stoppbefehl vor (Eingangssignal H), so folgt auf den Zustand 3 der Zustand 0 (Übergang k, kein Ausgangssignal). Bei einem unbedingten Sprungbefehl oder einem bedingten Sprungbefehl, bei dem die Sprungbedingung erfüllt ist, folgt auf den Zustand 3 der Zustand 7. Hier wird die Sprungadresse in den Befehlszähler transportiert (Übergang i, Ausgangssignal T_3, Eingangssignal S, E, $\overline{H}$). Auf Zustand 3 folgt Zustand 5, wenn die Sprungbedingung nicht erfüllt ist. Im Zustand 5 wird der Befehlszähler um 1 erhöht (Eingangssignal S, $\overline{E}$, $\overline{H}$; Übergang 1, Ausgangssignal +1. Auf den Zustand 4 folgt grundsätzlich der Zustand 5 (Übergang e, Ausgangssignal +1). Zustand 5 wird abgelöst durch den Zustand 1 (Übergang f, Ausgangssignal T_1). Auf den Zustand 6 folgt der Zustand 3 (Übergang h, Ausgangssignal SP). Vom Zustand 7 aus gelangt das Schaltwerk in den Zustand 1 (Übergang j, Ausgangssignal T_1).

Alle notwendigen Eingangs- und Ausgangssignale sind nachfolgend noch einmal zusammengestellt:

A	=	Adressenmodifikation erforderlich
S	=	Sprungbefehl liegt vor
H	=	Stoppbefehl liegt vor
E	=	Sprungbedingung erfüllt
T_1	=	Transport Befehlsadresse zum Speicher
T_2	=	Transport Befehl zum Befehlsregister
T_3	=	Transport Befehlsadresse zum Befehlszähler
SP	=	Aufforderung zur Prüfung, ob Stopp- oder Sprungbefehl vorliegt und ob die gestellte Sprungbedingung erfüllt ist
AM	=	Aufforderung zur Adressenmodifikation
OP	=	Aufforderung zur Ausführung der gewünschten Operation
+1	=	Erhöhung des Inhaltes des Befehlszählers um 1

Das Schaltwerk soll also insgesamt 8 innere Zustände durchlaufen können. Es sind damit mindestens 3 Flipflops notwendig, um dieses Schaltwerk zu realisieren. Diese 3 Flipflops sollen mit X, Y und Z bezeichnet werden. Welche der 8 Kombinationen der drei Flipflops den einzelnen inneren Zuständen zugeordnet wird, ist hier rein willkürlich, und zwar ist die Zuordnung nach den Dualzahlen vorgenommen worden. Die Wahrheitstabelle des Schaltwerkes (Tabelle 2.1) zeigt den Gesamtzusammenhang für die 3 Flipflops und den erforderlichen Zuordner. Der Strich an einigen Stellen der Tabelle soll bedeuten, daß es hier gleichgültig ist, ob 0 oder 1 vorliegt.

Bei der Betrachtung des Graphen und der Tabelle 2.1 ist zu beachten, daß das Schaltwerk z.B. in dem Zustand 1 den verlangten Transport T_1 nicht selbst durchführt. Dies wird vielmehr durch ein untergeordnetes Schaltwerk bewirkt. Es wird lediglich die nächste auszuführende Arbeit vorbereitet. Beim Übergang von dem Zustand 1 in den Zustand 2 wird also das Ausgangssignal T_2 vorhanden sein, das die im Zustand 2 **auszuführende Arbeit veranlaßt.** Dies gilt gleichermaßen auch für alle anderen Zustände und Ausgangssignale. Es müssen also immer der Übergang und die in diesem Zeitpunkt vorhandenen Signale betrachtet werden.

Übergang	Zustände zum Zeitpunkt t_n				Eingangssignale				Zustände zum Zeitpunkt $t_{(n+1)}$				Ausgangssignale zum Zeitpunkt t_n						
		X	Y	Z	A	S	E	H		X	Y	Z	T_1	T_2	T_3	SP	AM	OP	+1
a	0	0	0	0	—	—	—	—	1	0	0	1	1	0	0	0	0	0	0
b	1	0	0	1	—	—	—	—	2	0	1	0	0	1	0	0	0	0	0
c	2	0	1	0	0	—	—	—	3	0	1	1	0	0	0	1	0	0	0
d	3	0	1	1	—	0	0	0	4	1	0	0	0	0	0	0	0	1	0
e	4	1	0	0	—	—	—	—	5	1	0	1	0	0	0	0	0	0	1
f	5	1	0	1	—	—	—	—	1	0	0	1	1	0	0	0	0	0	0
g	2	0	1	0	1	—	—	—	6	1	1	0	0	0	0	0	1	0	0
h	6	1	1	0	—	—	—	—	3	0	1	1	0	0	0	1	0	0	0
i	3	0	1	1	—	1	1	0	7	1	1	1	0	0	1	0	0	0	0
j	7	1	1	1	—	—	—	—	1	0	0	1	1	0	0	0	0	0	0
k	3	0	1	1	—	0	0	1	0	0	0	0	0	0	0	0	0	0	0
l	3	0	1	1	—	1	0	0	5	1	0	1	0	0	0	0	0	0	1

Tabelle 2.1 — Wahrheitstabelle für das Schaltwerk

Der Aufbau des Zuordners läßt sich aus der Tabelle 2.1 entnehmen. Es ist allerdings hierzu notwendig, den Flipfloptyp festzulegen. In diesem Beispiel sollen JK-Masterslave-Flipflops verwendet werden. Tabelle 2.2 zeigt die vier Arbeitsfälle des Flipflops und die hierzu notwendige Beschaltung der J- und K-Eingänge.

	Q (t_n)	Q (t_{n+1})	J	K
Fall I	0	0	0	—
Fall II	0	1	1	—
Fall III	1	0	—	1
Fall IV	1	1	—	0

Tabelle 2.2 — Vorbereitungstabelle für JK-Masterslave-Flipflops

Die Striche in der Tabelle sagen aus, daß hier beliebig 1 oder 0 anliegen kann. Die für die Beschaltung der J- und K-Eingänge notwendigen Schaltfunktionen lassen sich aus der Wahrheitstabelle des Schaltwerkes angeben, ebenfalls die für die Ausgangssignale notwendigen Verknüpfungen.

Für den J-Eingang des Flipflops X ergibt sich z.B. folgender Zusammenhang:

> J muß 1 werden bei den Übergängen d, g, i und l. J muß 0 werden bei den Übergängen a, b, c und k.

Für den K-Eingang ergibt sich:

> K muß 1 werden bei den Übergängen f, h und j.
>
> K muß 0 sein bei dem Übergang e.

In der Schreibweise der Schaltalgebra angegeben und vereinfacht ergeben sich folgende Schaltfunktionen:

$$J_x = YZ\bar{S} \vee Y\bar{Z}A \qquad T_1 = \bar{X}\bar{Y} \vee XYZ$$
$$K_x = Y \vee Z \qquad T_2 = \bar{X}\bar{Y}Z$$
$$J_y = \bar{X}Z \qquad T_3 = \bar{X}YZE$$
$$K_y = ZX \vee Z\bar{S} \qquad SP = \bar{X}Y\bar{Z}\bar{A} \vee XY\bar{Z}$$
$$J_z = X \vee \bar{Y} \vee A \qquad OP = \bar{X}YZ\bar{S}\bar{E}\bar{H}$$
$$K_x = \bar{X}\bar{Y} \vee \bar{X}\bar{S} \qquad 1+ = \bar{X}YSE$$

Die hier angegebenen Schaltfunktionen sind nicht die einzig möglichen, sie stellen nur eine von vielen Lösungen dar. Die Richtigkeit der Schaltfunktionen läßt sich durch entsprechendes Einsetzen der Werte in die Tabelle nachprüfen.

Abb. 2.11 zeigt den Signalflußplan dieses Steuerautomaten. Nicht eingezeichnet ist die Erzeugung der Anfangszustände, die dadurch notwendig wird, daß beim Einschalten einer Anlage die Stellung der Flipflops vom Zufall bestimmt ist. Das Taktflipflop T ist in seiner Funktion ebenfalls nur angedeutet. Die für die drei Flipflops gemeinsame Taktleitung ist nur dann freigegeben, wenn das Taktflipflop T gesetzt ist. Dies erfolgt durch ein Startzeichen oder durch die Meldung, daß die durch das Steuerwerk veranlaßte Arbeit ausgeführt wurde. Das Steuerwerk gibt auch nur dann Steuersignale nach außen ab, wenn das Flipflop T sich in Arbeitsstellung befindet, wenn also eine neue Arbeit ausgelöst werden soll.

Nachfolgend wird der Arbeitsablauf des Steuerwerkes bei einem Sprungbefehl ohne Adressenmodifikation beschrieben. Die Sprungbedingung sei erfüllt. Es liegen also folgende Eingangssignale vor:

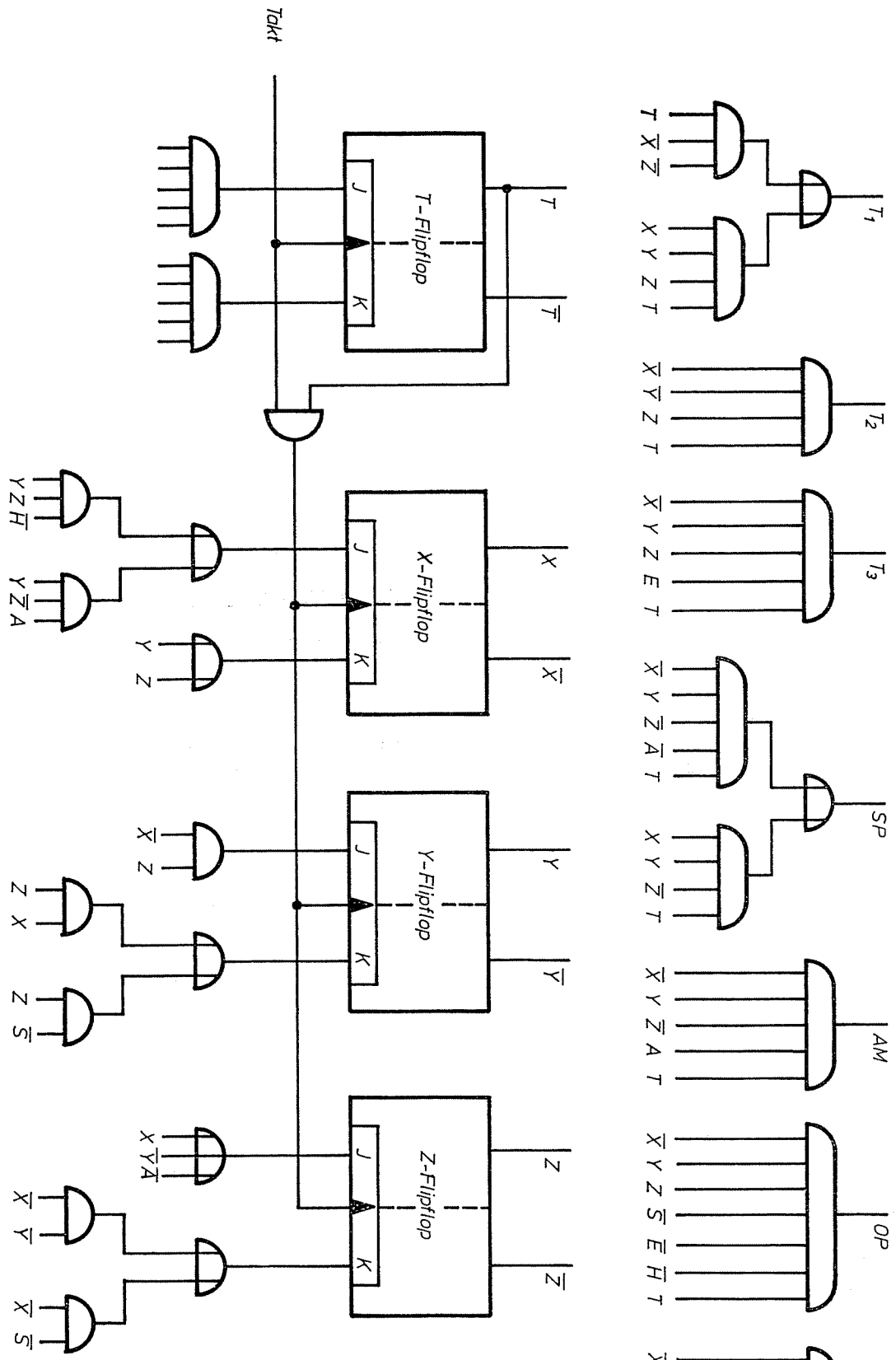


Abb. 2.11 — Steuerwerk zur Erzeugung der Befehlsphasen

Keine Adressenmodifikation	$\bar{A}$
Sprungbefehl liegt vor	S
Sprungbedingung erfüllt	E
Kein Stoppbefehl	$\bar{H}$

Das Schaltwerk soll sich im Ruhezustand **(X = 0 Y = 0 Z = 0),** d.h. im Zustand 0 befinden. Es werden keine Signale nach außen abgegeben, da das Taktflipflop nicht gesetzt ist. Wird jetzt durch einen Startimpuls das Flipflop T gesetzt, so werden die Taktleitung und das Ausgangssignal T_1 freigegeben.

Bei der Stellung 0 des Schaltwerkes und den vorliegenden Eingangssignalen wird nur das Flipflop Z auf den J- und dem K-Eingang vorbereitet; es wird mit dem nächsten Takt gesetzt; der Zustand 1 ist erreicht **(X = 0 Y = 0 Z = 1).**

Das Taktflipflop T wird wieder zurückgestellt. Ist der Transport der Befehlsadresse zum Speicher erfolgt und auch der neue Befehl gelesen, so wird das T-Flipflop erneut gesetzt. Es wird dann das Signal T_2 abgegeben. Im Zustand 1 ist der J-Eingang des Flipflops Y und der K-Eingang des Flipflops Z vorbereitet. Y wird mit dem nächsten Takt gesetzt, Z wird zurückgesetzt; das Schaltwerk befindet sich im Zustand 2 **(X = 0 Y = 1 Z = 0).**

Erst nach erfolgtem Transport des Befehls zum Befehlsregister wird das T-Flipflop erneut gesetzt. Das Steuersignal SP wird nach außen abgegeben. Im Zustand 2 und durch das Eingangssignal $\bar{A}$ ist der K-Eingang des Flipflops X und der J-Eingang des Flipflops Z vorbereitet. Das Z-Flipflop wird gesetzt; es ist der Zustand 3 erreicht **(X = 0 Y = 1 Z = 1).**

Ist die im Zustand 3 verlangte Prüfung erfolgt (Sprungbefehl, Sprungbedingung erfüllt, kein Stoppbefehl), so wird das Signal T_3 abgegeben. Es ist außerdem der J- und der K-Eingang des Flipflops X, der J-Eingang des Flipflops Y und des Flipflops Z vorbereitet. Das Schaltwerk geht mit dem nächsten Takt in den Zustand 7 **(X = 1 Y = 1 Z = 1).**

Ist die Adresse des Sprungbefehls zum Befehlszähler übermittelt, so wird erneut das Flipflop T gesetzt. Das Steuersignal T_1 wird abgegeben. Vorbereitet sind die J- und K-Eingänge des Flipflops X, der K-Eingang des Flipflops Y und der J-Eingang des Flipflops Z. Die Flipflops X und Y werden mit dem Eintreffen des nächsten Taktes zurückgesetzt; es ist wieder der Zustand 1 erreicht **(X = 0 Y = 0 Z = 1).**

Der weitere Ablauf ist jetzt davon abhängig, welcher Befehl neu aus dem Speicher abgerufen wird. Es kann nun sein, daß die angegebene Lösung des Steuerwerkes nicht einen minimalen Schaltungsaufwand ergibt. Wären z.B. anstatt der Mindestanzahl von 3 Flipflops 8 Flipflops gewählt worden, von denen nur ein einziges in den einzelnen Zuständen jeweils 1 zeigt, so wäre der Aufwand an Flipflops zwar größer geworden, der Aufwand an Verknüpfungsschaltungen hätte jedoch abgenommen. Ein solches Schaltwerk ist übersichtlicher und auch leichter zu entstören. Man erhält bei einer derartigen Lösung einen Ringzähler als Steuerkette, wobei, durch die Taktimpulse gesteuert, eine 1 durch den Ring läuft und die einzelnen Steuerphasen markiert. Abb. 2.12 zeigt einen Ausschnitt aus einem Ringzähler zur Erzeugung der einzelnen Befehlsphasen in stark vereinfachter Form. Verwendung finden D-Flipflops. Das Schaltwerk zur Erzeugung der Befehlsphasen ist nur ein Teil der Steuereinheit. Die Gesamtheit der Schaltwerke für alle Teilaufgaben einer Steuerung ergibt erst die gesamte Steuereinheit.

Eine besonders gebräuchliche Form einer Operationssteuerung ist das Mikroprogrammsteuerwerk. Die Eingangssignale für das Mikroprogrammsteuerwerk sind hier ebenfalls der Operationsteil des auszuführenden Befehls und Entscheidungszeichen von außen. **Der Operationsteil des Befehls kann hier als Adressierung eines festverdrahteten Programms angesehen werden. Durch ihn wird eine Reihe von Mikrooperationen ausgelöst.** Mikrooperationen sind z.B. Addition von Akkumulatorinhalt und Operandenregisterinhalt, Rechts- und Linksverschiebungen des Bitmusters im Akkumulator, Transporte von Daten usw. **Aus diesen Mikrooperationen lassen sich ganze Programme zusammenstellen, die durch einen einzigen Befehl abgerufen werden.**

Von Interesse ist die Frage, wieviel Mikrobefehle, d.h. wieviel Elementaroperationen wünschenswert sind. Gibt es viele verschiedene Mikrooperationen, so wird das gewünschte Mikroprogramm kurz, die Maschinenbefehle können schnell aufeinander folgen. Der technische Aufwand steigt stark an. Gibt es nur wenige Mikrooperationen, so bedeutet dies einen geringeren Schaltungsaufwand. Der gewünschte Maschinenbefehl ist aber unter Umständen nur durch ein umfangreiches und langes Mikroprogramm erreichbar. Dies führt dann zu langsamen Maschinenbefehlsfolgen. Schnelle EDV-Anlagen besitzen eine größere Zahl von Mikrooperationen. Die Schnelligkeit muß aber durch größeren technischen Aufwand bezahlt werden.

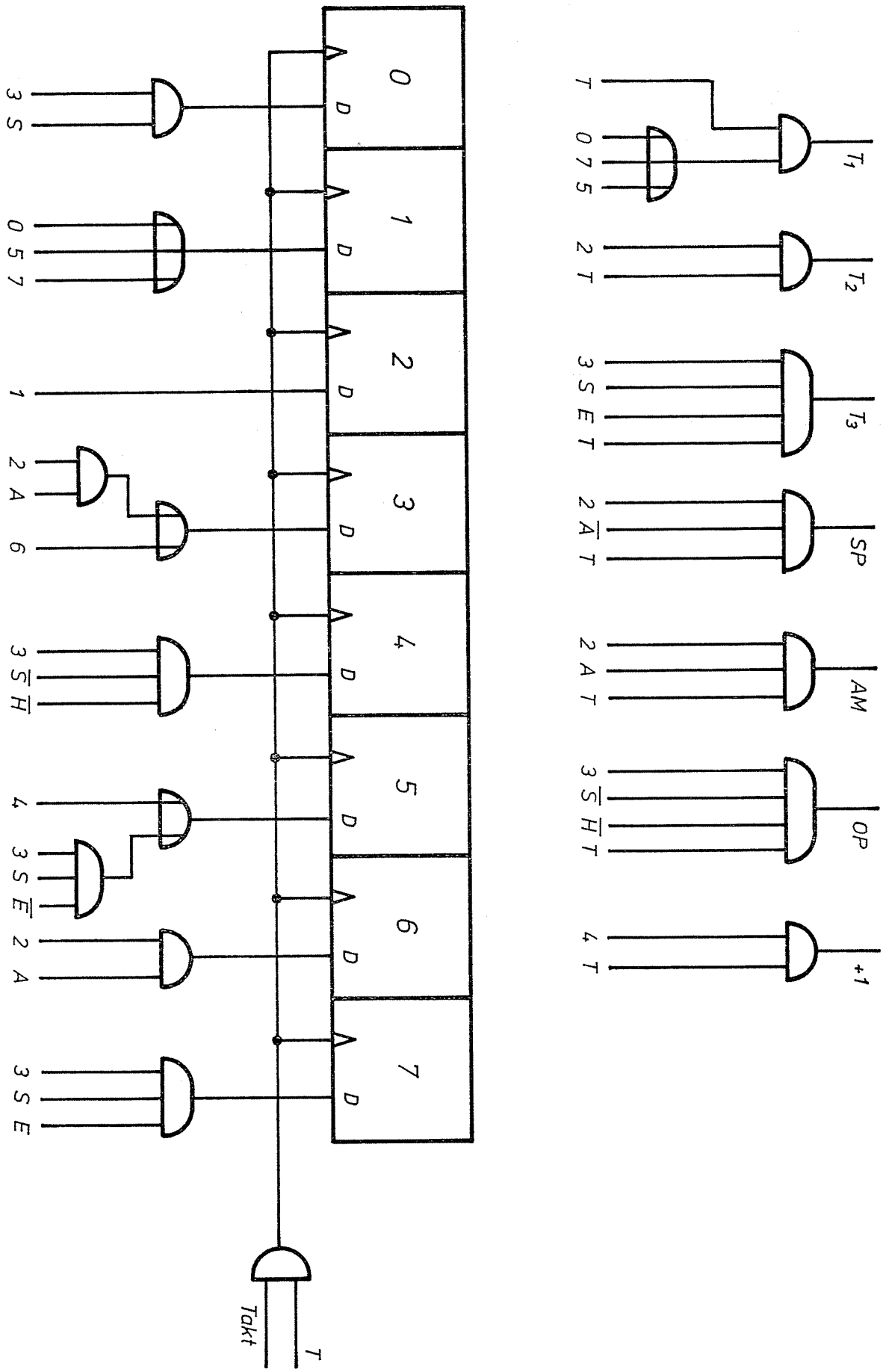


Abb. 2.12 — Steuerwerk zur Erzeugung der Befehlsphasen

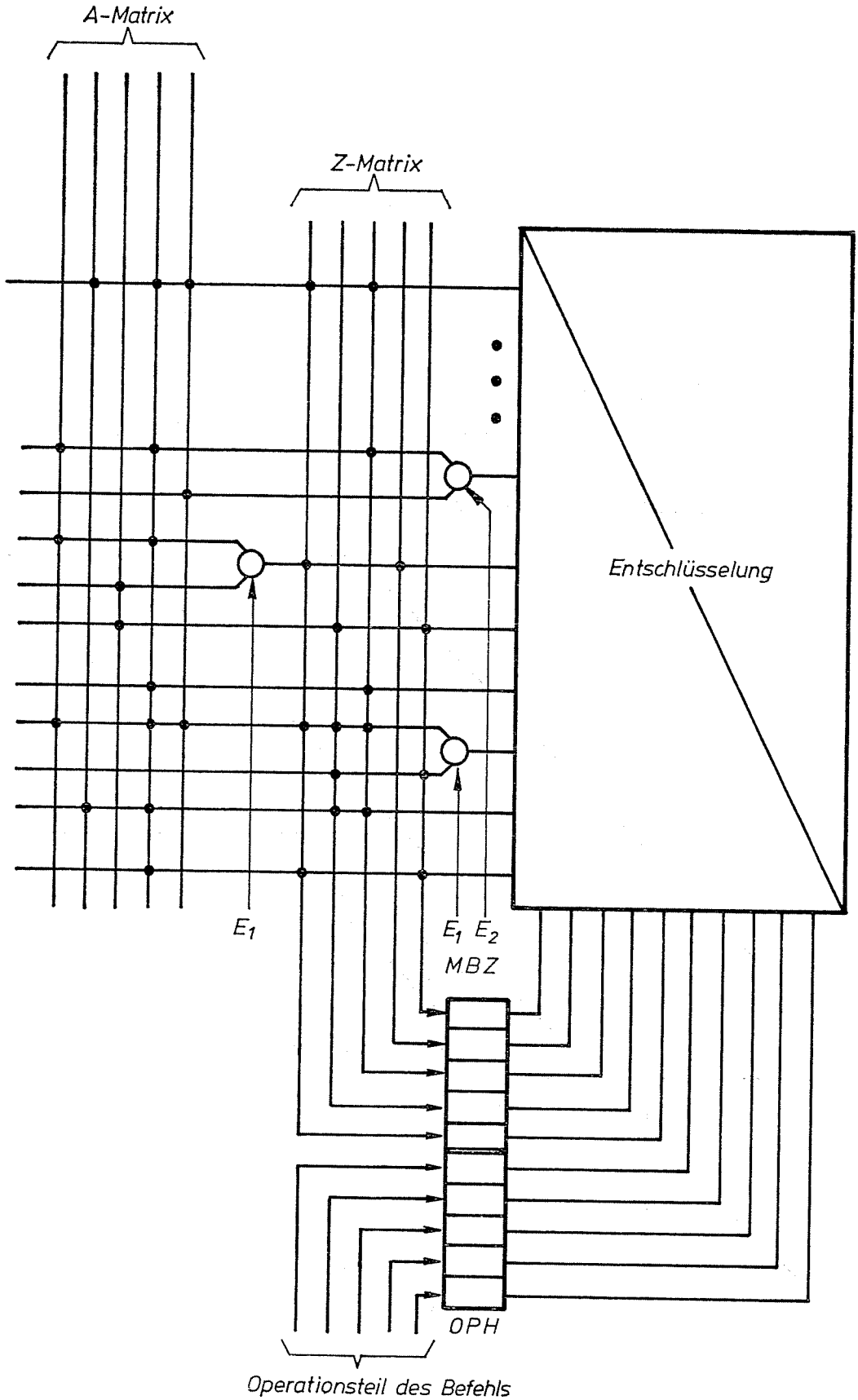


Abb. 2.13 — Prinzipbild eines Mikroprogrammsteuerwerkes

Wie ist nun ein Mikroprogrammsteuerwerk aufgebaut? Auch hier ist wieder ein Speicher vorhanden. **Dieser Speicher besteht aus zwei Teilen. Der eine Teil wird Operationshalter genannt, der andere Teil Mikrobefehlszähler. Der Inhalt des Operationshalters kennzeichnet den auszuführenden Befehl. Der Inhalt des Mikrobefehlszählers kennzeichnet die gerade durchgeführte Mikrooperation.** Abb. 2.13 zeigt das Prinzip eines solchen Mikroprogrammsteuerwerkes.

Die Information des gesamten Speichers, also die Angabe über die gewünschte Mikrooperation, wird einer Entschlüsselungsmatrix zugeführt. Es wird hier so entschlüsselt, daß immer nur eine der Ausgangsleitungen 1 führt, alle anderen 0, damit ist die auszuführende Mikrooperation angegeben. Die Ausgangsleitungen dieser Entschlüsselungsmatrix münden in eine Ausgangsmatrix (A-Matrix) und in eine Zustandsmatrix (Z-Matrix). Die Ausgangsmatrix gibt die zur Ausführung der Mikrooperation notwendigen digitalen Signale ab. Die Zustandsmatrix stellt die Information, die beim nächsten Takt in den Mikrobefehlszähler eingelesen werden soll, bereit. Es wird also schon wieder der nächste Mikroschritt vorbereitet. Kommt der nächste Takt, so ergibt die in den Mikrobefehlszähler eingelesene Information zusammen mit dem Operationsteil des Befehls die nächste Mikrooperation.

Die Signale der Z-Matrix und der A-Matrix werden aber nicht allein von dem Ausgang der Entschlüsselung bestimmt, sondern sind z.T. von Entscheidungszeichen von außen abhängig. Dies ist in Abb. 2.13 angedeutet, wo sich einige der entschlüsselten Ausgänge abhängig von einem Entscheidungszeichen E_i verzweigen, also abhängig davon, ob eine 0 oder eine 1 vorliegt.

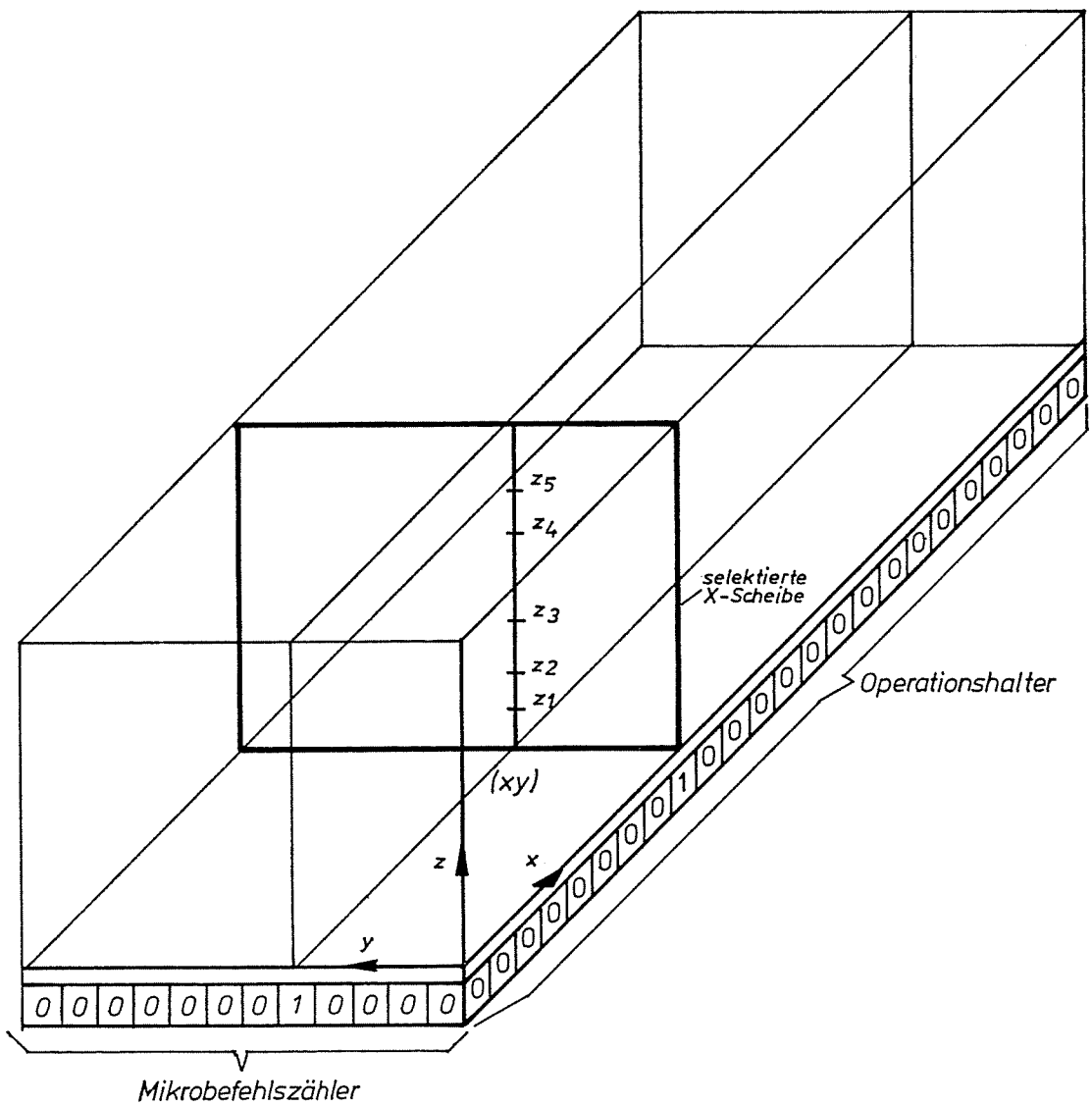


Abb. 2.14 — Mikroprogrammsteuerwerk

Zu Beginn eines jeden Mikroprogramms steht der Mikrobefehlszähler in der Stellung 0. Es wird der Operationsteil eines Befehls in den Operationshalter transportiert, ein Mikroprogramm wird ausgewählt. Mit jedem Takt nimmt der Mikrobefehlszähler dann eine neue Stellung ein, durchläuft das Mikroprogramm, bis er wieder in der Ruhestellung ist. Es kann ein neuer Befehl folgen.

Bei einem Mikroprogrammsteuerwerk kann jeder Befehl als eine Art Unterprogramm aufgefaßt werden. Die Operationssteuerung wird übersichtlich und es besteht auch häufig die Möglichkeit, ein solches Unterprogramm, also einen Maschinenbefehl, durch den Benutzer einer EDV-Anlage selbst zu erzeugen bzw. zu verändern. Diese Vorteile — Übersichtlichkeit und leichte Änderbarkeit — treten bei der Anordnung in der Abb. 2.14 noch deutlicher hervor.

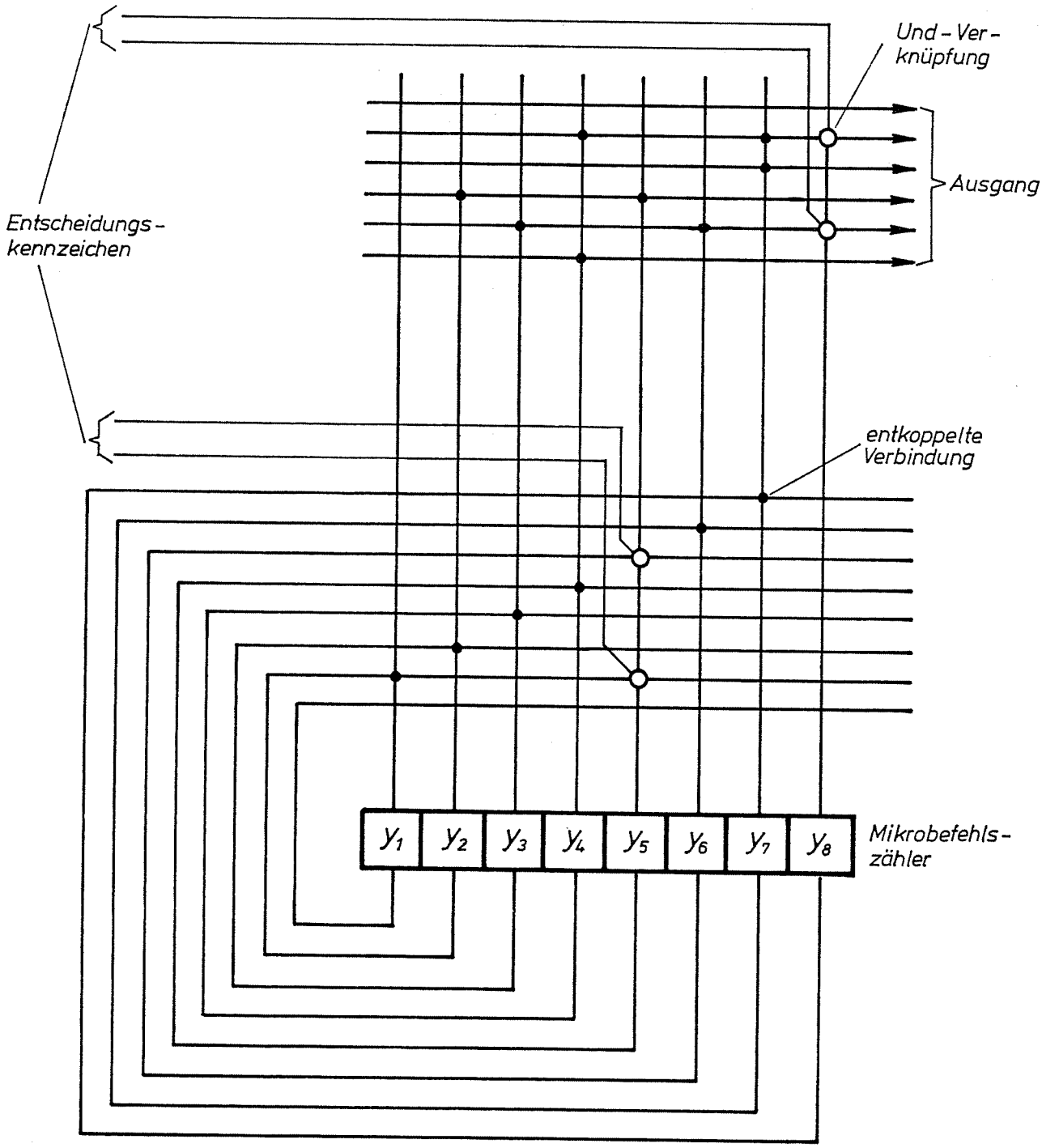


Abb. 2.15 — Schnittebene x aus dem Mikroprogrammsteuerwerk

Der Operationshalter und der Mikrobefehlszähler werden hier als gesonderte Speicher betrachtet und jeweils einzeln entschlüsselt. Abhängig vom Inhalt des Operationshalters und vom Inhalt des Mikrobefehlszählers sollen über eine A-Matrix und eine Z-Matrix Ausgangssteuersignale und neue Eingangssignale für den Mikrobefehlszähler erzeugt werden. Man kann sich die beiden Register — Operationshalter und Mikrobefehlszähler — mit ihrer Entschlüsselung in die x- und y-Richtung als eine räumliche Gitterkonstruktion denken. Die zu erzeugenden Ausgangssignale liegen dann in der z-Ebene des Raumgitters. Abb. 2.14 zeigt ein solches Mikroprogrammsteuerwerk. Durch den Inhalt des Operationshalters, also durch den auszuführenden Befehl, wird nur eine bestimmte x-Koordinate aufgesucht, d.h. es liegt nur eine 1 an **einem** x-Punkt, alle anderen führen 0. Es wird also aus dem räumlichen Gitter eine bestimmte x-Scheibe ausgesucht. Der Inhalt des Mikrobefehlszählers ergibt nach der Entschlüsselung ebenfalls nur **eine** 1 an eine der y-Koordinaten. Im Schnittpunkt der x- und y-Koordination befinden sich UND-Schaltungen an den Punkten der z-Richtung, die jetzt Signale nach außen abgeben solsen. Über ODER-Schaltungen können die Signale der einzelnen z-Koordination gegebenenfalls wieder zusammengefaßt werden. Die UND-Schaltungen im Schnittpunkt der x- und y-Koordinaten können außerdem von Entscheidungszeichen abhängig gemacht werden. Pro Taktzeit des Mikroprogrammsteuerwerkes wird also jeweils eine Senkrechte des räumlichen Gitters angesprochen.

Abb. 2.15 zeigt eine Schnittebene aus dem Mikroprogrammsteuerwerk. Diese Schnittebene entspricht einer bestimmten Operation eines Befehls, z.B. eines Multiplikationsbefehls. Die A-Matrix ergibt die nach außen abzugebenden Signale, die Z-Matrix ergibt die neue Information zum Einstellen des Mikrobefehlszählers. Für eine solche x-Ebene kann nun eine steckbare Baugruppe vorgesehen werden. Ebenfalls können die Z-Matrix und die A-Matrix auch durch den Inhalt eines Festwertspeichers realisiert werden. Der x-y-Schnittpunkt bestimmt dann die Adresse des zu lesenden Speicherplatzes eines solchen Festwertspeichers.

2.2. Rechenwerk

2.2.1. Allgemeines

Das Rechenwerk einer EDV-Anlage ist die Funktionseinheit, die die Durchführung der erforderlichen arithmetischen und logischen Operationen ermöglichen. Es werden hier Zahlen addiert, subtrahiert, multipliziert oder dividiert sowie UND-, ODER- oder andere logische Verknüpfungen durchgeführt. Es werden abfragbare Anzeigen, z.B. Flipflops, gesetzt, die erkennen lassen, ob eine Zahl negativ, positiv oder Null ist. Hieraus lassen sich dann Bedingungen für Sprungbefehle ableiten. Die zu verarbeitenden Daten, die Operanden, werden aufgrund von Befehlen durch das Leitwerk in das Rechwerk transportiert. Ebenfalls teilt das Leitwerk dem Rechenwerk in Form eines Steuersignals mit, welche Operation durchzuführen ist und erwirkt den eventuell notwendigen Rücktransport des entstandenen Ergebnisses. Die Ausführung der gewünschten Operation geschieht überwiegend durch das Rechenwerk selbsttätig.

2.2.2. Grundsätzliche Einteilung der Rechenwerke

Rechenwerke lassen sich nach verschiedenen Gesichtspunkten einteilen. Man unterscheidet z.B. zwischen Dualrechenwerken und Dezimalrechenwerken. Bei **Dualrechenwerken** werden binärcodierte Dualzahlen verarbeitet. Es wird also zur Zahlendarstellung das Dualzahlensystem verwendet. Bei **Dezimalrechenwerken** wird dagegen das Dezimalzahlensystem beibehalten, nur die einzelnen Dezimalziffern werden binärcodiert dargestellt. Da die Rechengesetze vom verwendeten Zahlensystem abhängen, z.B. die Übertragsbildung bei der Addition oder die Komplementbildung, unterscheiden sich Dezimalrechenwerke von Dualrechenwerken.

Hinsichtlich der Ausführungsform eines Rechenwerkes ist außerdem zwischen Parallel-, Serien- und Serienparallelrechenwerken zu unterscheiden. Bei Serienrechenwerken werden die einzelnen Bitstellen einzeln und zeitlich nacheinander, bei Parallelrechenwerken dagegen alle Bitstellen gleichzeitig verarbeitet. Bei Serienparallelrechenwerken werden die einzelnen Bits eines Wortes zu Gruppen zusammengefaßt und die Bitstellen einer Gruppe jeweils gleichzeitig, die einzelnen Gruppen eines Wortes zeitlich nacheinander verarbeitet. Dezimalrechner arbeiten überwiegend nach dem Serienparallelprinzip, wobei die binärcodierten Dezimalziffern jeweils einer solchen Gruppe entsprechen.

Neben den verschiedenen Ausführungsformen existieren noch verschiedene Ausbaustufen für ein Rechenwerk. Im einfachsten Falle ermöglicht eine Rechentabelle aus UND- und ODER-Verknüpfungen die gewünschte Addition. Eine höhere Ausbaustufe besitzen Rechenwerke, die die Elementaroperationen wie Addition, Subtraktion und Stellenverschiebung durchführen können. Höhere Operationen wie Multiplikation oder Division werden durch ein Unterprogramm abgewickelt.

Die höchste Ausbaustufe haben Rechenwerke, bei denen auch die Multiplikation und die Division festverdrahtet sind. Es gibt sogar Rechenwerke, die noch höhere Rechenoperationen nach einem festvorgegebenen Programm ablaufen lassen, z.B .das Wurzelziehen. Durch einen einzigen Maschinenbefehl kann dann diese Rechenoperation abgerufen werden.

Alle Rechenoperationen lassen sich auf eine Folge von stellenrichtigen Additionen zurückführen; dadurch wird der Aufwand eines Rechenwerkes in erträglichen Grenzen gehalten.

2.2.3. Arithmetische Voraussetzungen

2.2.3.1. Grundrechnungsarten im Dualzahlensystem

In einer EDV-Anlage haben Zahlensysteme praktische Verwendung gefunden, die auf einer Radixschreibweise, auch Stellenschreibweise genannt, beruhen. Charakteristisch für ein solches Zahlensystem ist die verwendete Basis (Grundzahl, Radix). Potenzen dieser Basis bestimmen die **Stellenwerte** der einzelnen Ziffern einer Zahl. Beim üblichen dezimalen Zahlensystem ist die Grundzahl 10, beim Dualzahlensystem ist die Grundzahl 2. Aus dem Dualzahlensystem läßt sich das oktale System mit der Basis 8 und sedezimale (hexadezimale) System mit der Basis 16 ableiten. Das Dualzahlensystem und das Rechnen mit Dualzahlen ist im „Handbuch der Elektronik; Teil 2, Digitaltechnik" ausführlich beschrieben. Nachfolgend werden die Regeln für die Konvertierung und die Grundrechnungsarten kurz erklärt und an einigen Beispielen erläutert.

Konvertierung im Dualzahlensystem

Dualzahlen und Dezimalzahlen lassen sich ohne jede Einschränkung ineinander umrechnen. Diesen Übergang von einem Zahlensystem in das andere nennt man **Konvertierung.** Das eine Zahlensystem ist hierbei das **Quellsystem,** das andere das **Zielsystem.** Die eigentliche Konvertierung ist ein Rechenvorgang, der im Quellsystem oder im Zielsystem durchgeführt werden kann. Wir wenden ein Konvertierungsverfahren an, mit dem es möglich ist, ganze und gebrochene Zahlen in das Dualzahlensystem und zurück zu konvertieren. Die eigentliche Rechnung läuft hierbei im Dezimalzahlensystem ab.

Fall 1:

Umwandlung ganzer Zahlen vom Dezimalzahlensystem in das Dualzahlensystem

Die Konvertierung geschieht durch wiederholtes Dividieren der umzuwandelnden Zahl durch 2. Die sich hierbei ergebenden Reste, 0 bzw. 1, ergeben dann, in der entsprechenden Reihenfolge gelesen, die Dualzahl.

Beispiel: Es soll die Zahl 62 umgewandelt werden.

62 : 2 = 31	Rest 0	↑	niedrigste Stelle der Dualzahlen
31 : 2 = 15	Rest 1		
15 : 2 = 7	Rest 1		
7 : 2 = 3	Rest 1		
3 : 2 = 1	Rest 1		
1 : 2 = 0	Rest 1		höchste Stelle der Dualzahlen

62 ≙ 111110
dezimal dual

Fall 2:

Umwandlung gebrochener Zahlen vom Dezimalzahlensystem in das Dualzahlensystem

Die gebrochene, als Dezimalbruch geschriebene Zahl, wird jetzt wiederholt mit 2 multipliziert. Die sich durch die Multiplikation ergebende Stelle vor dem Komma liefert jeweils eine Stelle der gesuchten Dualzahl. Mit dem Ergebnis nach dem Komma wird dann die nächste Multiplikation durchgeführt. Es ist nun möglich, daß die Rechnung nicht aufgeht, daß sich z.B. ein periodischer Bruch ergibt. Die Rechnung muß dann nach der gewünschten Stellenzahl abgebrochen werden. Im nachfolgenden Beispiel sind 10 Stellen nach dem Komma ermittelt worden.

Beispiel: Es soll die Zahl 0,62 umgewandelt werden

0,62 × 2 = 1,24	liefert 1		erste Stelle nach dem Komma
0,24 × 2 = 0,48	liefert 0		
0,48 × 2 = 0,96	liefert 0		
0,96 × 2 = 1,92	liefert 1		
0,92 × 2 = 1,84	liefert 1		
0,84 × 2 = 1,68	liefert 1		
0,68 × 2 = 1,36	liefert 1		
0,36 × 2 = 0,72	liefert 0		
0,72 × 2 = 1,44	liefert 1		
0,44 × 2 = 0,88	liefert 0	↓	

0,62 = 0,1001111010
dezimal dual

Fall 3:

Umwandlung einer ganzen Zahl aus dem Dualzahlensystem in das Dezimalzahlensystem

Bei dieser Umwandlung werden die einzelnen Zwischenergebnisse der Rechnung mit 2 multipliziert und die Stelle der umzuwandelnden Dualzahl addiert. Man beginnt immer mit Null, d.h. das erste Zwischenergebnis ist Null.

Beispiel: Es soll die Zahl 111110 (dual) umgewandelt werden.

	Dualzahl
0 × 2 + 1	= 1
1 × 2 + 1	= 3
3 × 2 + 1	= 7
7 × 2 + 1	= 15
15 × 2 + 1	= 31
31 × 2 + 0	= 62

111110 = 62
dual dezimal

Fall 4:

Umwandlung einer gebrochenen Dualzahl in eine Dezimalzahl

Hier wird zum Zwischenergebnis der Rechnung die Stelle der gegebenen Dualzahl addiert und anschließend durch 2 geteilt. Auch hier beginnt man mit Null.

Beispiel: Es soll 0,1001111010 (dual) in eine Dezimalzahl umgewandelt werden.

0	+	0 = 0	: 2 = 0
0	+	1 = 1	: 2 = 0,5
0,5	+	0 = 0,5	: 2 = 0,25
0,25	+	1 = 1,25	: 2 = 0,625
0,625	+	1 = 1,625	: 2 = 0,8125
0,8125	+	1 = 1,8125	: 2 = 0,90625
0,90625	+	1 = 1,90625	: 2 = 0,953125
0,953125	+	0 = 0,953125	: 2 = 0,4765625
0,4765625	+	0 = 0,4765625	: 2 = 0,23828125
0,23828125	+	1 = 1,23828125	: 2 = 0,619160625

0,1001111010 = 0,619160625
dual dezimal

Addition im Dualzahlensystem

Die Addition zweier Zahlen wird im Dezimal- und auch im Dualzahlensystem in eine Reihe von einzelnen Additionen aufgelöst. Es werden jeweils die Ziffern mit gleicher Stellenzahl addiert, wobei ein eventueller Übertrag der vorhergehenden Stelle mit zu berücksichtigen ist. Bei digitalen Rechenwerken werden nun fast immer nur zwei Zahlen zur gleichen Zeit addiert, d.h. zwei Ziffern pro Stelle. Da die beiden Ziffern nur die beiden Werte 0 und 1 annehmen können, ergeben sich für die Addition folgende Rechenregeln:

a) Addition zweier Dualziffern ohne Berücksichtigung eines Übertrages der vorhergehenden Stelle

Ziffer 1		Ziffer 2		Summe	Übertrag
0	+	0	=	0	0
0	+	1	=	1	0
1	+	0	=	1	0
1	+	1	=	0	1

Die erste Stelle einer Dualzahl kann nach dieser Rechenregel addiert werden. Schon bei der nächsten Stelle muß ein möglicherweise auftretender Übertrag der vorhergehenden Stelle mit berücksichtigt werden.

b) Addition zweier Dualziffern mit Berücksichtigung des Übertrages der vorhergehenden Stelle

Ziffer 1	Ziffer 2	Übertrag ü	Summe	Übertrag U
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

ü = der zu berücksichtigende Übertrag der vorhergehenden Stelle,

U = der Übertrag, der an die nächste Stelle weitergegeben wird.

Beispiel für die Anwendung dieser Rechenregeln:

Dualzahlen		Dezimalzahlen zur Kontrolle
101110010	Ziffer 1	370
+ 10011011	Ziffer 2	155
111110010	Übertrag	010
1000001101		525

Subtraktion im Dualzahlensystem

Ähnlich der Addition werden bei der Subtraktion immer Ziffern gleichen Stellenwertes voneinander subtrahiert. Als Ergebnis entsteht die Differenz. Es muß aber auch berücksichtigt werden, daß sich Ziffern mit niedrigem Stellenwert zur Differenzbildung unter Umständen etwas borgen, entlehnen und daß auch zur eigenen Differenzbildung von der nächsthöheren Stelle eine Entlehnung vorgenommen werden muß. Hieraus leitet sich jetzt die Rechenregel für die Subtraktion zweier Dualzahlen ab. Bedenken Sie, daß bei der Entlehnung einer 1 von der nächsten höheren Stelle praktisch zwei Einsen

an die untere Stelle entliehen werden; die nächsthöhere Stelle besitzt immer einen genau doppelt so großen Stellenwert wie die Stelle, an die entliehen wird.

Minuend	Subtrahend	Entlehnung e von rechts	Differenz	Entlehnung E nach links
0	0	0	0	0
1	0	0	1	0
0	1	0	1	1
1	1	0	0	0
0	0	1	1	1
1	0	1	0	0
0	1	1	0	1
1	1	1	1	1

Beispiel für die Anwendung dieser Rechenregel:

Es wird von der niedrigsten zur höchsten Stelle hin subtrahiert.

	Dualzahlen		Dezimalzahlen zur Kontrolle
	101110010	Minuend	370
—	10011011	Subtrahend	— 155
	010011111	Entlehnung	001
	011010111	Differenz	215

Vergleicht man die Regeln für die Addition und Subtraktion miteinander, so sieht man, daß die Regeln zwar ähnlich sind, jedoch nicht völlig übereinstimmen. Logische Netzwerke, Addierer genannt, die die Additionsregeln erfüllen, können deshalb bei einer unmittelbaren Differenzbildung nicht benutzt werden. Es müßten hierzu extra Subtrahierschaltungen vorhanden sein. Deshalb ist es besser, die Subtraktion durch Verwendung des Komplementverfahrens auf eine Addition zurückzuführen. Dann können zur Addition und Subtraktion die gleichen technischen Einrichtungen benutzt werden.

Subtraktion durch Addition des Komplementes

Das B-Komplement: Bei der Subtraktion durch das Komplementverfahren benutzt man eine geeignete Hilfsgröße K. Diese Hilfsgröße K ist eine positive Zahl und ihr Betrag kann z.B. so gewählt werden, daß die Subtraktion der Zahl K durch eine einfache Streichung der höchsten Stelle des Ergebnisses erfolgen kann. Dies ist der Fall, wenn bei Zahlen mit maximal n Stellen $K = B^n$ gewählt wird. B ist hierbei die Basis des verwendeten Zahlensystems. Bei maximal 3 Stellen im Dezimalzahlensystem würde K dann zu $K = 10^3 = 1000$. Bei maximal 5 Stellen im Dualzahlensystem würde K zu $B^n = 2^5 = 32 \triangleq 100000_{\text{dual}}$.

Die Differenz einer Zahl y zu dieser Zahl K nennt man das B-Komplement der Zahl y, geschrieben $\bar{y}_B$.

Bitte verwechseln Sie nicht $\bar{y}_B$ mit der Negation von zweiwertigen Variablen in der Schaltalgebra. $\bar{y}_B$ ist hier eine Zahl, nämlich die Differenz von K — y.

Die Subtraktion x — y läßt sich durch Einführung des B-Komplementes in eine Addition überführen.

$$x - y = x + \underbrace{(K - y)}_{\bar{y}_B} - K = x + \bar{y}_B - K$$

Die Subtraktion von K ist jedoch nicht notwendig, da für K z.B. im Rechenregister kein Platz ist und K als Überlauf verloren geht. Die Subtraktion durch die Addition des B-Komplementes einer Zahl soll nun an einigen Beispielen erläutert werden. Wir betrachten zunächst nur Beispiele, die als Ergebnis keine negativen Zahlen ergeben.

Beispiel im Dezimalzahlensystem mit 3stelligen Zahlen

K wird zu $10^3 = 1000$.

Aufgabe: 653 — 251 Das B-Komplement von 251 ist 1000 — 251 = 749

```
  653          653
         →
— 251        + 749
 ____          111
              1402  Ergebnis 402
              ↑——— geht als Übertrag verloren
```

Beispiel im Dualzahlensystem mit 5stelligen Zahlen

K wird deshalb zu $2^5 = 32$ ⟶ entspricht 100000 dual.

Aufgabe: 21 — 15, entspricht 10101 — 1111. Das Komplement von 15 zu 32 ist 32 — 15 = 17 entspricht 10001 dual

Dezimalzahlen zur Kontrolle

```
  21      21
— 15  +   17  — 32
          00
          38  — 32  = 6
```

Dualzahlen

```
  10101   →    10101
— 01111      + 10001
               10001   Übertrag
              100110   Ergebnis 00110
              ↑——— geht als Überlauf verloren
```

Für die Herstellung des B-Komplementes im Dualzahlensystem gibt es eine Regel, durch die das Komplement unmittelbar gefunden werden kann.

Sie lautet: Man erhält das B-Komplement einer Dualzahl, indem man alle Stellen negiert, die links der niedrigsten von Null verschiedenen Stelle stehen.

Beispiel: y = 10010 (←), $\bar{y}_2$ = 01110

y = 1100100 (←), $\bar{y}_2$ = 0011100

Für maschinelle Verarbeitung ist diese Regel nicht gut geeignet. Eine einfache Regel hierfür erhält man jedoch, wenn man als Hilfsgröße nicht B^n wählt, sondern den Wert $B^n—1$. Man erhält dann das sogenannte (B—1)-Komplement.

Das (B—1)-Komplement: Wird das Komplement zur Hilfsgröße $K = B^n—1$ gebildet, so erhält man das (B—1)-Komplement.

Beim Dezimalzahlensystem mit maximal 3stelligen Zahlen erhält man also als Hilfsgröße K den Wert: $K = B^n—1 = 1000 — 1 = 999$.

Beispiele:

y	= 253	= 762
$\overline{y_9}$	= 746	= 237
	999	999

Das Komplement ergibt sich durch Ergänzung aller Stellen auf 9, also eine einfache Regel. Beim Dualzahlensystem wird die Hilfsgröße K den folgenden Wert annehmen:

$$K = B^n—1 = 2^n—1$$

Bei maximal 5 Stellen ergibt sich für K der Wert 31

$K = 2^5 — 1 = 32 — 1 = 31$ entspricht 11111_{dual}

Man erhält nun das (B—1)-Komplement einer Dualzahl durch die Negierung jeder Stelle der Dualzahl.

Beispiele:

y	= 0100101	= 11010011
$\overline{y_1}$	= 1011010	= 00101100
	1111111	11111111

Diese Regel ist sehr einfach und auch maschinell leicht durchführbar.

Wird nun mit diesem Komplement die Subtraktion durchgeführt, so wird eine nachträgliche Addition einer 1 erforderlich, da sich nach der Addition des Komplementes $(B^n—1)—y$ und der Subtraktion von B^n durch den Überlauf folgender Wert ergibt:

$$x + \underbrace{\left[(B^n—1) — y\right]}_{\overline{y}_{B-1}} — B^n = x — y — 1$$

Durch eine nachträgliche Addition von 1 erhält man dann schließlich das gewünschte Ergebnis.

Beispiel im Dezimalzahlensystem:

Aufgabe: 653 — 251

Das (B-1)-Komplement von 251 ist 748

653	653	
— 251	+ 748	
	1401	
	+ ⟶1	zusätzliche Addition von 1 durch
	402	Addition des Überlaufes
		Ergebnis 402

Beispiel im Dualzahlensystem:

Aufgabe: 10101 — 01111 Das (B-1)-Komplement von 01111 ist 10000

10101	10101	
— 01111	+ 10000	
	100101	
	⟶1	zusätzliche Addition von 1
	00110	durch Addition des Überlaufes
		Ergebnis 00110

Mit dem Umweg über das (B—1)-Komplement läßt sich nun auch leicht das B-Komplement bilden, und zwar bildet man zuerst das (B—1)-Komplement und erhält dann durch eine Addition von 1 das B-Komplement.

Negative Zahlen

Soll eine Subtraktion unbeschränkt durchführbar sein, so müssen auch negative Zahlen zugelassen und darstellbar sein. Normalerweise werden negative Zahlen durch ein Vorzeichen von den positiven Zahlen unterschieden. Auch im Digitalrechner ist eine solche Darstellung möglich. **Das Vorzeichen + wird hier durch eine 0 und das Vorzeichen — durch eine 1 repräsentiert.** Wenn im Rechner alle Zahlen gleiche Stellenzahl haben (fehlende Stellen werden mit Null aufgefüllt), ergibt sich folgende Darstellung bei einer festen Stellenzahl von 12 Bits.

+ wird durch 0 angegeben	→ 0'000101110010	= + 370
— wird durch 1 angegeben	→ 1'000010011011	= — 155
	Vorzeichen	

Für eine Multiplikation oder Division ist diese Zahlendarstellung gut geeignet, für eine Addition oder Subtraktion ergeben sich jedoch Schwierigkeiten. Bei unterschiedlichem Vorzeizeichen, z.B. bei (—7) + (+3), wird zuerst festgestellt, daß eigentlich keine Addition vorliegt, sondern eine Subtraktion, denn (—7) + (+3) = (+3) — (+7). Nun ist aber 7 größer als 3, und es muß also von 7 die 3 abgezogen werden, wobei das Ergebnis negativ wird. All diese Überlegungen erfordern aber eine komplizierte Technik. Es gibt nun eine andere Möglichkeit, negative Zahlen darzustellen, nämlich unter Verwendung sogenannter „konegativer" Zahlen. Konegative Zahlen ergeben sich aus folgender Überlegung: Betrachtet man die Subtraktionsaufgabe $x — y = z$ als die Addition der negativen Größe —y und schreibt unter Verwendung vorzeichenbehafteter Zahlen, so ergibt sich:

$$(+x) + (—y) = +z$$

Vergleicht man dies mit der Komplementform

$$x - y = x + y_B = z$$

(+ verlorengehender Übertrag),

so entspricht —y dem Wert $\bar{y}_B$. Das bedeutet, daß man negative Zahlen durch ihr Komplement darstellen kann. **Durch ihr Komplement dargestellte negative Zahlen heißen konegative Zahlen. Es kann das B-Komplement und auch das (B—1)-Komplement verwendet werden.** Auch hier wird durch eine zusätzliche Vorzeichenstelle angegeben, ob es sich um eine positive (Vorzeichenbit 0) oder um eine konegative Zahl (Vorzeichenbit 1) handelt.

Beispiel für die Zahl — 13 im Dualzahlensystem

a) 1'001101 Vorzeichen und Betrag
b) 1'110011 Vorzeichen und B-Komplement
c) 1'110010 Vorzeichen und (B-1)-Komplement

Bei der Addition und Subtraktion bringt die Verwendung konegativer Zahlen Vorteile, da für positive und negative Zahlen gleiche Rechengesetze gelten und keine Vorzeichenberechnungen erfolgen müssen. **Ein Nachteil der Vorzeichendarstellung durch 0 und 1 ergibt sich, wenn z.B. bei einer Addition der zulässige Zahlenbereich überschritten wird. Der Übertrag der ersten Stelle wird dann zum Vorzeichenbit zugeschlagen und das Vorzeichen wechselt. Dieser Fehler würde nicht erkannt werden.**

Beispiel: 3 Stellen und 1 Vorzeichenstelle

5 + 5 = 10

```
  0'101
  0'101
 ------
  1'010   Das Ergebnis wird
  ↑       als konegative
          Zahl gedeutet
```

Diese Schwierigkeit wird durch Einführung einer zusätzlichen Bitstelle, einer sogenannten Schutzstelle, vermieden. Diese Schutzstelle fängt den unerlaubten Übertrag auf und läßt erkennen, daß der Zahlenbereich überschritten wurde. Allgemein gilt, daß Vorzeichenstelle und Schutzstelle gleichen Wert haben müssen, also beide Null bei positiven Zahlen und beide Eins bei konegativen Zahlen. Ein beim Addieren bzw. Subtrahieren auftretender Übertrag ist auch bei konegativen Zahlen nach dem (B—1)-Komplement grundsätzlich der ersten Stelle aufzuaddieren.

Die Tabelle 2.3 zeigt Ihnen die ersten 7 negativen und positiven Zahlen im Dualzahlensystem unter Verwendung der verschiedenen Darstellungsmöglichkeiten. Ein Nachteil des (B-1)-Komplementes ist die Tatsache, daß es eine positive und eine negative Null gibt, die aber vom Rechner als völlig gleich angesehen werden muß.

Dualzahl	Vorzeichen und Betrag	B-Komplement	(B-1)-Komplement
00 000	+ 0	0	+ 0
00 001		+ 1	
00 010		+ 2	
00 011		+ 3	
00 100		+ 4	
00 101		+ 5	
00 110		+ 6	
00 111		+ 7	
11 000	— 0	— 8	— 7
11 001	— 1	— 7	— 6
11 010	— 2	— 6	— 5
11 011	— 3	— 5	— 4
11 100	— 4	— 4	— 3
11 101	— 5	— 3	— 2
11 110	— 6	— 2	— 1
11 111	— 7	— 1	— 0

↑↑ Schutzstelle / Vorzeichen

Tabelle 2.3 — Tabelle der ersten 7 positiven und negativen Zahlen

Es folgen einige Beispiele für die Addition und Subtraktion mit konegativen Zahlen. Die Subtraktion erfolgt durch Addition des B-Komplementes bzw. des (B-1)-Komplementes. Ist das Ergebnis negativ, also eine 1 an der ersten Stelle, so findet man die Dezimalzahl durch Komplementierung des Ergebnisses.

a) Konegative Zahlen durch das B-Komplement

dezimal	dual		
+ 3	00 011		
+ — 2	+ 11 110		
+ 1	1 00 001		
	↑ geht verloren		

dezimal	dual		
— 4	11 100	→ 11 100	
— — 5	— 11 011	+ 00 101	B—K
+ 1		1 00 001	
		↑ geht verloren	

dezimal	dual		
+ 4	00 100	→ 00 100	
— + 5	— 00 101	+ 11 011	B—K
— 1		11 111	

dezimal	dual	
+ 3	00 011	
+ + 6	00 110	
+ 9	01 001	
	↑ Bereichsüberschreitung	

dezimal	dual	
— 3	11 101	
+ — 6	+ 11 010	Komplement
— 9	110 111	
	↑	
	geht verloren	Bereichsüberschreitung

b) Konegative Zahlen durch das (B-1)-Komplement

dezimal	dual	
+ 3	00 011	
+ — 2	+ 11 101	
+ 1	1 00 000	
	1	zusätzliche Addition des Übertrages
	00 001	

dezimal	dual	
— 4	11 011	
+ — 2	+ 11 101	
— 6	111 000	
	1	zusätzliche Addition des Übertrages
	11 001	

dezimal	dual		
— 4	11 011	→ 11 011	
— — 5	— 11 010	+ 00 101	B—K
+ 1		100 000	
		1	zusätzliche Addition des Übertrages
		00 001	

dezimal	dual		
+ 4	00 100	→ 00 100	
— + 5	— 00 101	+ 11 010	B—K
— 1		11 110	
			keine zusätzliche Addition, da kein Übertrag

dezimal	dual	
— 3	11 100	
+ — 6	+ 11 001	
— 9	101 101	
	1	zusätzliche Addition des
	01 110	Übertrages
	↑	
	Bereichsüberschreitung	

Multiplikation im Dualzahlensystem

Auch die Multiplikation verläuft im Dualzahlensystem im üblichen dezimalen Rechenschema. Das hierbei notwendige „Einmaleins" ist sehr einfach, da ja nur die Ziffern 1 und 0 vorkommen.

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

Der Multiplikand einer Multiplikationsaufgabe wird nun nach den nebenstehenden Rechenregeln mit jeweils einer Multiplikatorstelle verknüpft, und die Teilergebnisse werden stellenrichtig miteinander addiert.

Beispiel für eine Multiplikation:

11011 × 101 = entspricht 27 × 5 = 135

a) 11011 × 1 = 11011
b) 11011 × 0 = + 00000
011011
c) 11011 × 1 = + 11011
10000111 = 135

Als Ergebnis entsteht eine Zahl mit einer Stellenzahl, die der Summe der Stellenzahlen der beiden Faktoren entspricht. Das muß bei der technischen Realisierung berücksichtigt werden.

Die Division wird später im Zusammenhang mit der technischen Lösung des Rechenwerkes beschrieben.

2.2.3.2. Binärcodierte Dezimalzahlen

Die Verwendung des dualen Zahlensystems in Digitalrechnern hat neben vielen Vorteilen auch den Nachteil, daß alle Zahlen zweimal konvertiert werden müssen; aus dem Dezimalsystem in das Dualsystem und wieder zurück. Dieser Rechenvorgang kann das Rechenwerk erheblich zeitlich in Anspruch nehmen. Da dieser Rechenvorgang im Dualzahlensystem erfolgt, müssen alle Dezimalzahlen vor dieser Konvertierung in binärcodierte Dezimalzahlen umgewandelt werden. Es stellt sich die Frage, ob man nicht schon mit diesen binärcodierten Zahlen direkt rechnen kann. Dies ist tatsächlich möglich. Da aber die vier Bitstellen, die zur Darstellung der 10 Ziffern benötigt werden, insgesamt 16 Vierergruppen, Tetraden genannt, bilden können, müssen 6 Tetraden ungenutzt bleiben. Sie entsprechen keiner Dezimalziffer und heißen deshalb Pseudotetraden. Entsteht z.B. bei einer Addition eine solche Pseudotetrade, so muß sie erkannt und korrigiert werden. Zu diesem Zweck muß ein **Code** eingesetzt werden, der sinnvoll in einem Rechenwerk angewendet werden kann.

Dualwert	Tetraden	Aiken-Code	Stiebitz-Code
0	0 0 0 0	0	
1	0 0 0 1	1	
2	0 0 1 0	2	
3	0 0 1 1	3	0
4	0 1 0 0	4	1
5	0 1 0 1		2
6	0 1 1 0		3
7	0 1 1 1		4
8	1 0 0 0		5
9	1 0 0 1		6
10	1 0 1 0		7
11	1 0 1 1	5	8
12	1 1 0 0	6	9
13	1 1 0 1	7	
14	1 1 1 0	8	
15	1 1 1 1	9	

Tabelle 2.4 — Übliche Binärverschlüsselungen im Rechenwerk

Praktische Bedeutung haben der Stiebitz-Code (Dreiexzeß) und der Aiken-Code erlangt. Beide Codes erfüllen die Bedingung der einfachen Korrektur und der Komplementbildung.

Die Tabelle 2.4 zeigt den Aiken-Code und den Stiebitz-Code. Sie können erkennen, daß die Ziffernzuordnung symmetrisch zur Mittellinie ist und daß sich Tetraden, deren Dezimalziffern sich auf Neun ergänzen, auf 1111 ergänzen. Es entsteht also wie beim Dezimalzahlensystem ein Übertrag, wenn die Dezimalsumme größer ist als Neun.

Als Beispiel für die Rechnung mit binärcodierten Dezimalziffern werden nachfolgend Addition und Subtraktion im Stiebitz-Code erklärt. Die einzelnen Bitstellen der Tetraden werden nach den Regeln der Dualzahlen addiert. Nach der Addition findet dann eine Korrektur des Ergebnisses statt. Die Korrekturvorschrift ergibt sich aus folgender Überlegung: Jede Tetrade hat einen um 3 höheren Dualwert als die ihr zugeordnete Dezimalzahl. Bei einer Addition zweier Tetraden entsteht also eine Dezimalzahl, die einen um 6 höheren Dualwert hat. Das Ergebnis muß also um den Dualwert 3 vermindert werden, da das Ergebnis auch nur einen um 3 höheren Dualwert haben darf. Ist nun aber das Dezimalergebnis über 10, so ist das Dualergebnis um 6 größer, also über 16. Da aber ein Übertrag verloren geht — er wird an die nächste Stelle abgegeben — wird praktisch 16 abgezogen. Es werden also 10 für die nächste Stelle und 6 für die eigene Stelle abgezogen. Dies ist aber 3 zuviel. Beim Auftreten eines Übertrages muß deshalb eine 3 dazu addiert werden.

Die Korrekturregel lautet also: Entsteht ein Übertrag bei der Addition, so ist der Dualwert von 3, also 0011 zur Ergebnistetrade hinzuzuaddieren. Entsteht kein Übertrag, so ist der Dualwert 3 zu subtrahieren. Die Subtraktion erfolgt über die Addition des B-Komplementes (1101) und Unterdrückung des evtl. auftretenden Übertrages.

Dies soll nun an einigen Beispielen deutlich gemacht werden.

Beispiel:

```
  56                    1000   1001
+ 27                  + 0101   1010
  83                       1
      kein Übertrag,     1110  10011   auftretender
      also Add.        + 1101    0011   Übertrag,
      von 1101         ------  ------   deshalb
                       11011     0110   Addition v. 0011
                       ↑
                       geht als Übertrag verloren
```

Durch Einführung einer Vorzeichenstelle und einer Schutzstelle kann auch für negative Zahlen die Darstellung über das Komplement gewählt werden. Komplementiert wird durch Negation jeder einzelnen Stelle.

$$+\ 53 = 00\ 1000\ 0110$$

$$-\ 53 = 11\ 0111\ 1001$$

Beispiel:

```
   27      00 0101  1010
-(-13)   - 11 1011  1001
  40
```

Lösen durch Addition des Komplementes

```
   00 0101  1010
+  00 0100  0110
          1←|
   00 1010 10000
      1101  0011
-------------------
   00 10111 0011   entspricht + 40
   ↑
   geht als Übertrag verloren
```

Beispiel:

```
  -13     11 1011 1001
+ -29     11 1010 0011
  -42   111 0101 1100
           0011 1101
        -------------
        11 1000 1001
                   1
        -------------
        11 1000 1010   entspricht -42
```

Die Regeln für die Addition und Subtraktion für positive und negative Zahlen und für die Korrektur sind also einfach und durch eine Maschine beherrschbar. Die Multiplikation und die Division von binärcodierten Dezimalzahlen werden im Zusammenhang mit der technischen Lösung des Rechenwerkes beschrieben (vgl. hierzu Abschn. 2.2.4.2. und 2.2.4.3.).

2.2.4. Aufbau des Rechenwerkes

2.2.4.1. Addition und Subtraktion

Addierer

Wie wir bereits wissen, lassen sich alle Grundrechnungsarten auf die stellenrichtige Addition von Dualzahlen zurückführen. Wesentlicher Bestandteil eines Rechenwerkes sind deshalb Verknüpfungsschaltungen, die es gestatten, zwei Dualziffern nach den Additionsgesetzen des Dualzahlensystems zu verknüpfen. Diese Verknüpfungsschaltungen werden Addierer genannt. Ein Addierer besitzt drei Eingänge, zwei Eingänge für die beiden zu addierenden Dualziffern und einen weiteren Eingang, um einen

eventuell auftretenden Übertrag der vorhergehenden Stelle mit zu berücksichtigen. Als Ergebnis liefert der Addierer zwei Ausgangswerte: die Summe S und den Übertrag Ü für die nächsthöhere Stelle (Abb. 2.16).

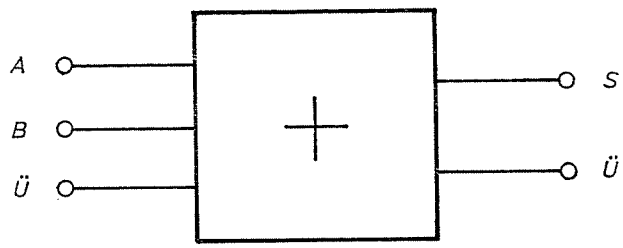


Abb. 2.16 — Volladdierer

Die logische Zuordnung der drei binären Eingangsvariablen A, B und ü zu den zwei Ausgangsgrößen S und Ü ist in der Tabelle 2.5 noch einmal angegeben.

A	B	ü	S	Ü
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabelle 2.5 — Logiktafel für Volladdierer

Diese logische Zuordnung zwischen Eingang und Ausgang läßt sich nun auf verschiedene Weise realisieren. Die in der Praxis ausgeführte Lösung hängt unter anderem vom verwendeten Bausteinsystem ab. Eine elegante Lösung mit geringem Aufwand ergibt sich, wenn die Addition in zwei Teiladditionen zerlegt wird. Es wird zuerst eine Addition der beiden Dualziffern vorgenommen und dann in einer nachfolgenden Addition der Übertrag ü zum Ergebnis der ersten Addition hinzugefügt. Der Addierer besteht dann aus zwei Teilen, die hintereinander geschaltet werden. Die Überträge Ü der einzelnen Teilschaltungen werden hier durch eine NAND-Schaltung zusammengefaßt.

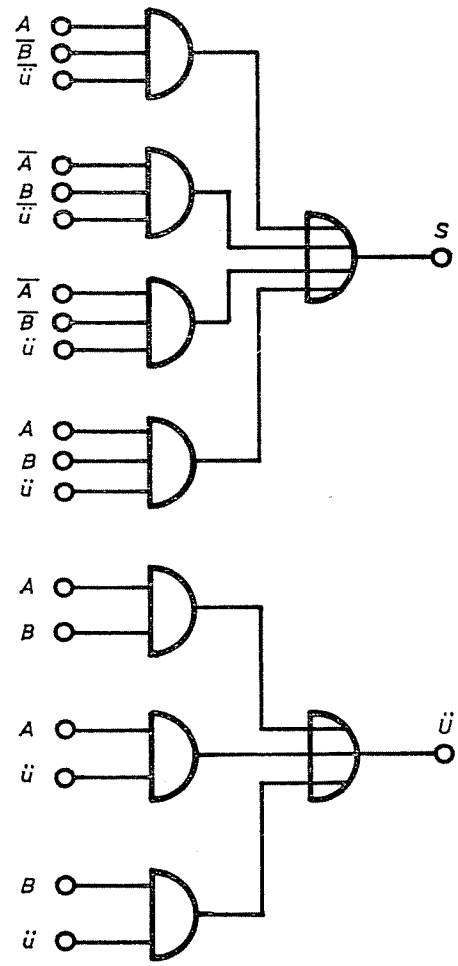


Abb. 2.18 — Volladdierer

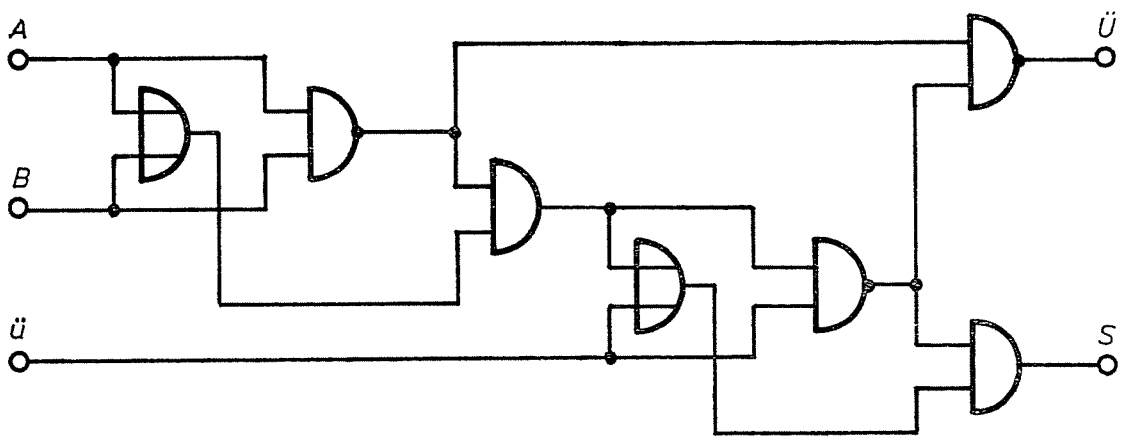


Abb. 2.17 — Schaltung eines Volladdierers

Zwei weitere Lösungen zeigen die Abbildungen 2.18 und 2.19. Beachten Sie, daß bei Abb. 2.18 außer den eigentlichen Eingangsvariablen A, B und ü auch noch deren Negationen zur Verfügung stehen müssen. Dies ist aber nicht unbedingt ein Nachteil, da ja die Dualzahlen in Flipflops gespeichert werden und deshalb auch negiert abgefragt werden können.

Nimmt man nun an, daß der Aufwand einer Logikschaltung proportional der Anzahl der Eingänge und der Transistoren ist, die aufgebracht werden müssen, so ist die Schaltung nach Abb. 2.17 am günstigsten. Es werden nur 14 Eingänge

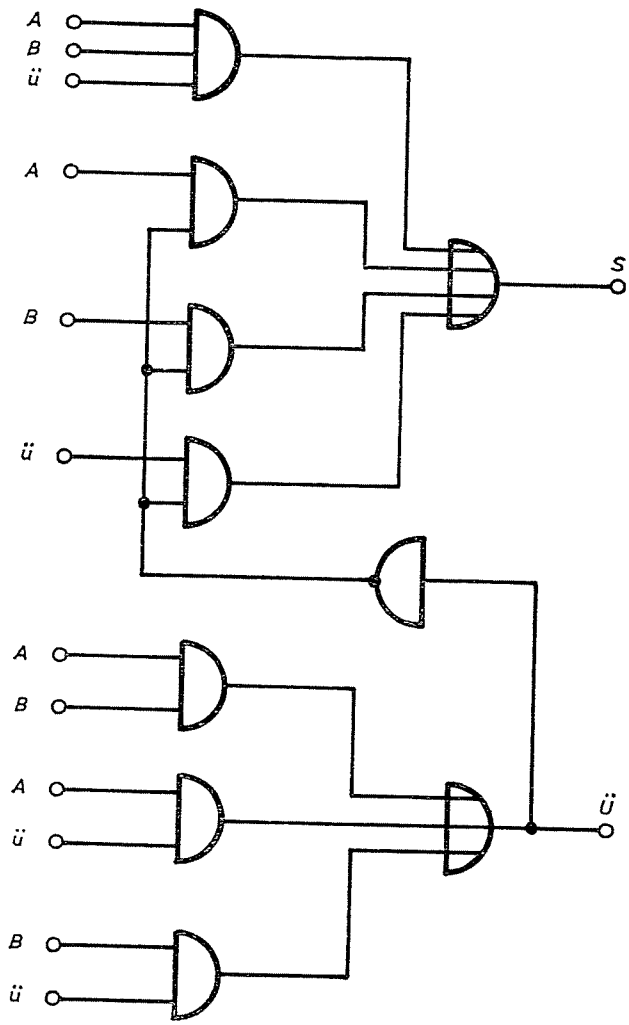


Abb. 2.19 — Volladdierer

(Dioden) und drei Transistoren benötigt, hingegen bei den beiden anderen Schaltungen bis zu 25 Dioden. Geht man jedoch von der Arbeitsgeschwindigkeit aus, so müssen bei Abb. 2.17 vier, bei Abb. 2.18 dagegen nur zwei Verknüpfungsbausteine durchlaufen werden. Durch die integrierte Schaltungstechnik werden heute schon Addierer auf den Markt gebracht, bei denen der Anwender nur noch die äußere Beschaltung der Bausteine durchzuführen hat. Die innere Schaltung deckt sich zum Teil mit der Schaltung der Abb. 2.19.

Bei der Addition von zwei binärcodierten Dezimalziffern werden vier Addierer parallel geschaltet und die vier Bitstellen der Dezimalziffern nach den Regeln der Dualzahlen auf einmal miteinander verknüpft. Das hierbei entstehende Ergebnis muß eventuell noch korrigiert werden. Als Beispiel soll wieder der Stiebitz-Code gewählt werden. Abb. 2.20 zeigt den Aufbau eines solchen Dreiexzeß-Tetraden-Addierers. Abhängig vom entstehenden Übertrag in der vierten Stufe der vier ersten Addierer wird durch drei weitere nachgeschaltete Addierer und einen Negator entweder 1101 (Ü=0) oder 0011 (Ü=1) zum Ergebnis der ersten Addition hinzugefügt.

Serienaddierwerk

Bei einem Serienaddierwerk werden die einzelnen Ziffern der beiden zu addierenden Zahlen zeitlich nacheinander einem einzigen Addierer zugeführt und dort verknüpft. Es werden immer Ziffern mit gleichem Stellenwert addiert und zwar mit der niedrigsten Stelle voran. Ein Serienaddierer besitzt deshalb also nur einen einzigen Addierer. Neben Addierern sind Register weitere Hauptbestandteile eines Rechenwerkes, also auch eines Serienaddierers. Sie speichern die zu verknüpfenden Zahlen für die Dauer der Rechenoperation und nehmen auch das Ergebnis auf. Zwei solcher Speicherregister sind in einem Serienaddierwerk vorhanden. Das eine Register ist für die Zahl bestimmt, zu der etwas hinzuaddiert bzw. von der etwas subtrahiert werden soll. Es nimmt auch nach der Operation das Ergebnis auf und wird allgemein Akkumulator genannt. Das zweite Register enthält die Zahl, die hinzuaddiert bzw. subtrahiert werden soll. Dieses zweite Register soll hier Register 2 genannt werden. Das Register 2 kann aus Kostengründen auch von einem Register des Speicherwerkes vertreten werden. Beide Register, Akkumulator und Register 2, sind Schieberegister, d.h., die Information wird seriell ein- und ausgelesen. Der beim Addierer auftretende Übertrag Ü muß nun für eine Taktzeit des Addierwerkes gespeichert werden, da er ja zur Addition der jeweils nächsten Stelle benötigt wird, die jedoch eine Taktzeit später erfolgt. Hierzu benutzt man ein Flipflop. Die am Eingang des Flipflops Ü anliegende Information 0 oder 1 wird durch einen Taktimpuls in das Flipflop eingele-

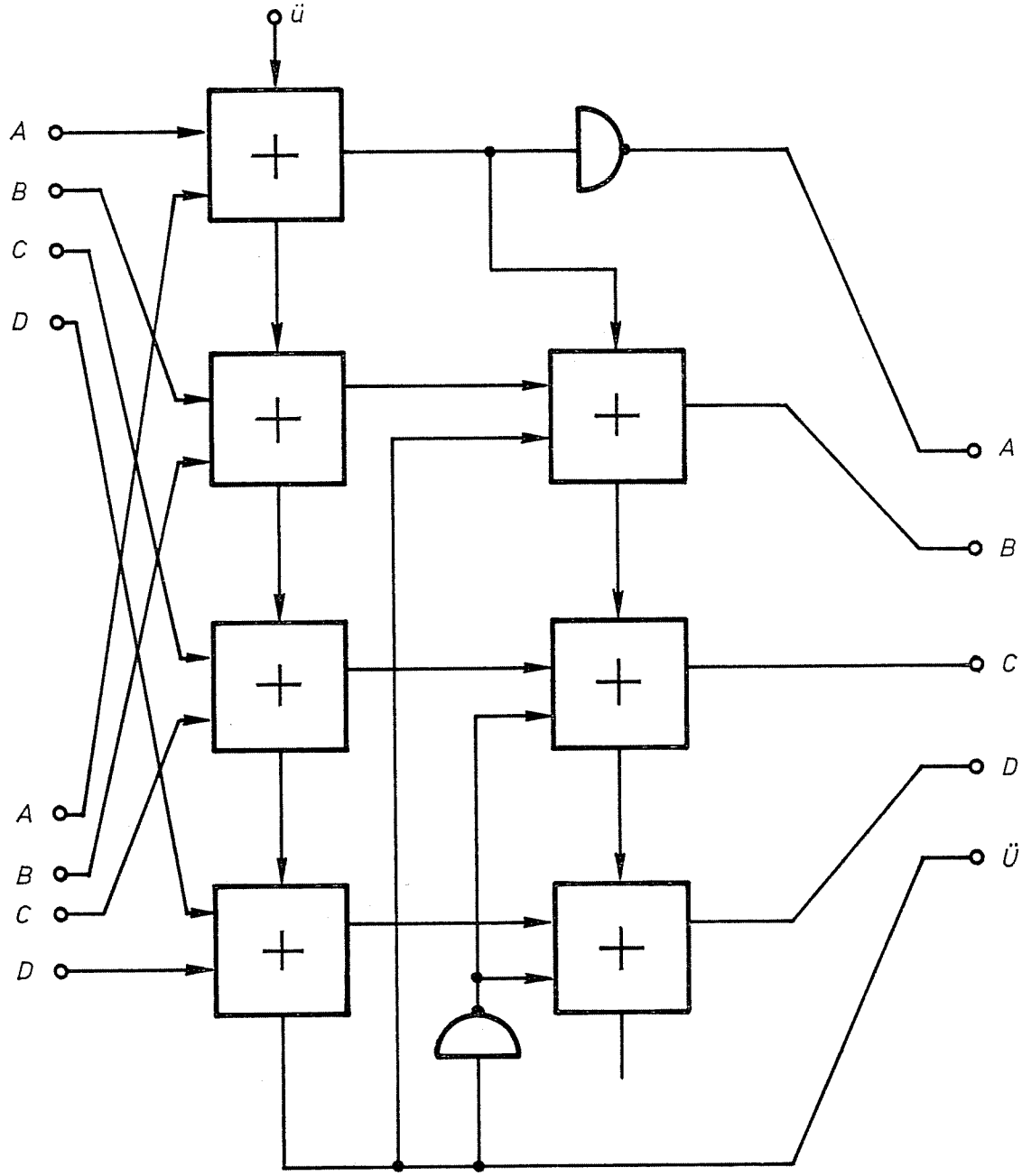


Abb. 2.20 — Dreiexzeß-Tetraden-Addierer zur Addition von zwei binärcodierten Dezimalziffern

sen und erscheint daher nach dem Takt am Ausgang. Die Information am Eingang erscheint also genau um eine Taktzeit verzögert am Ausgang; man spricht deshalb auch von einer eingeschalteten Verzögerung um eine Taktzeit. Abb. 2.21 zeigt das Prinzipbild eines Serienaddierwerkes für 6stellige Dualzahlen.

Eine Addition geht nun unter der Annahme, daß beide Zahlen schon in den entsprechenden Registern stehen, folgendermaßen vor sich: Am Steuereingang 1 und 2 liegt während des ganzen Additionsvorganges „1-Signal". Es ist damit der Eingang des Akkumulators mit dem Ausgang des Addierers verbunden. Ebenfalls liegt der Ausgang des Registers 2 unnegiert an dem zweiten Eingang des Addierers. Durch einen Steuerimpuls wird das Flipflop Ü vor der eigentlichen Addition auf Null gesetzt. Die Addition geschieht durch sechs Takte auf die beiden Schieberegister und auf das Übertragsflipflop Ü. Vor dem ersten Takt liegen die beiden ersten Ziffern der beiden Zahlen an dem Addierer. Am Ausgang erscheinen die sich ergebende Summe und der Übertrag. Durch den ersten Taktimpuls wird die Summe in die erste Stelle des Akkumulators eingelesen, während alle Ziffern der beiden

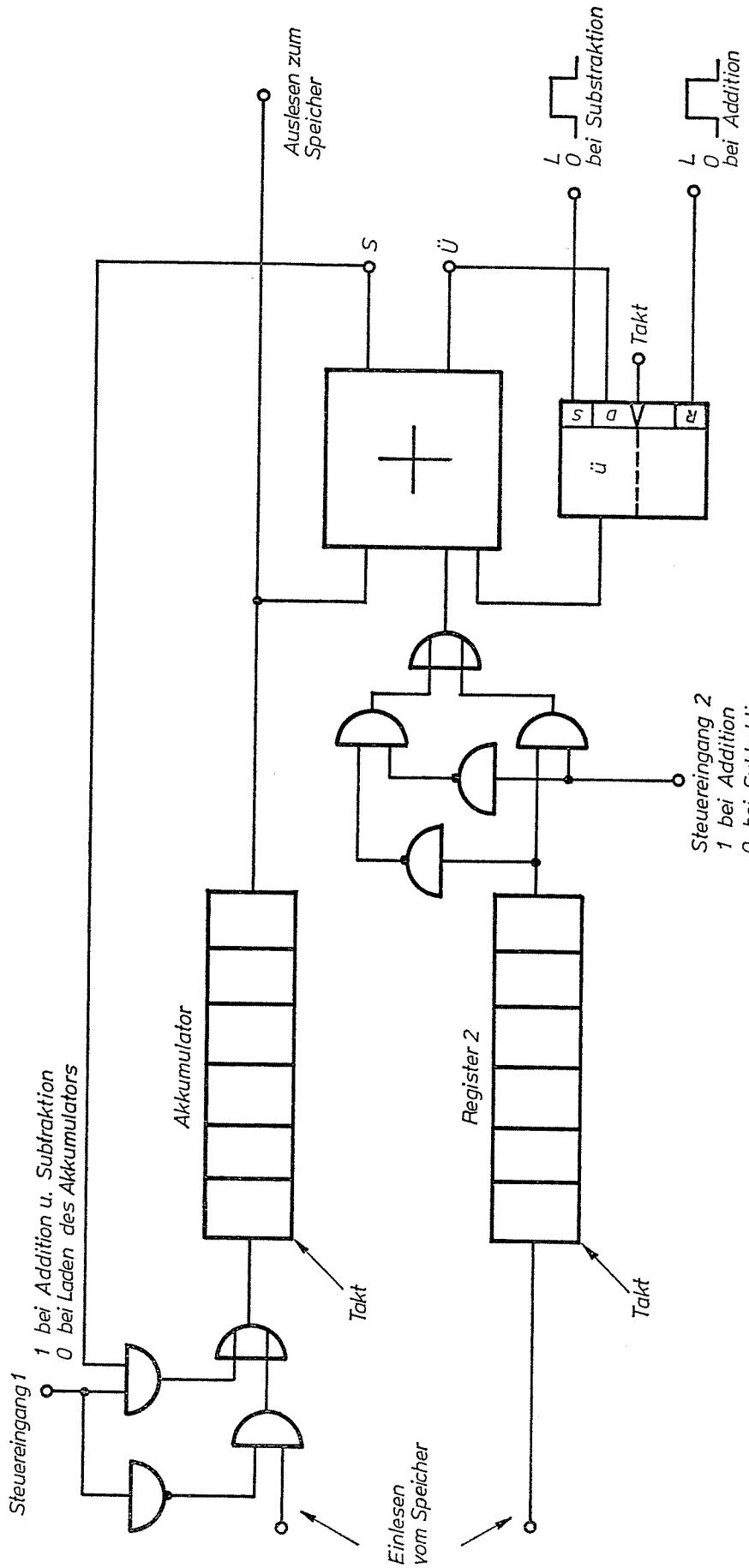


Abb. 2.21 — Serienaddierwerk für duale Zahlen

Zahlen jeweils um eine Stelle nach rechts weitergegeben werden. Der zuvor entstandene Übertrag ist ebenfalls durch den Taktimpuls in das Flipflop Ü eingelesen worden und steht zur nächsten Addition bereit. Nach insgesamt sechs Schiebetakten sind beide Zahlen addiert worden. Das Akkumulatorregister ist mit dem Inhalt der Addition gefüllt, das Register 2 ist ohne Inhalt. Ein noch im Übertragsflipflop Ü vorhandener Übertrag geht verloren.

Die Subtraktion zweier Zahlen ist abhängig von der im Rechner verwendeten Darstellung der negativen Zahlen. Wenn wir davon ausgehen, daß das B-Komplement verwendet wird, ergibt sich eine einfache Operation. Am Steuereingang 1 liegt dann „1-Signal", am Steuereingang 2 „0-Signal". Das Register 2 gibt dann seinen Inhalt negiert an den Addierer ab. Vor der eigentlichen Subtraktion wird das Übertragsflipflop Ü durch einen Impuls auf 1 gesetzt, dies entspricht einer Addition von 1 zur ersten Stelle. Addition einer 1 und Negierung aller Ziffern ergibt jedoch das B-Komplement. Nach sechs folgenden Schiebetakten ist dann die Subtraktion beendet.

Soll der Akkumulator geladen werden, so wird der Steuereingang 1 auf „0-Signal" gelegt. Die Rückführung des Akkumulators über den Addierer ist dann unterbrochen und der Einleseeingang des Akkumulators freigegeben. Es kann jetzt eine neue Zahl in den Akkumulator eingegeben werden.

Paralleladdierwerk

Bei einem Paralleladdierwerk werden alle Ziffern auf einmal addiert. Es müssen deshalb soviel Addierer vorhanden sein, wie die Zahlen Stellen haben.

Zwei Register bewahren die Zahlen für die Dauer der Operation auf. Die einzelnen Addierer sind so miteinander verbunden, daß der in einem Addierer entstehende Übertrag Ü der einlaufende Übertrag ü der nächsthöheren Stelle wird. Der letzte Addierer gibt seinen Übertrag Ü an den ersten Addierer zurück, so daß eine geschlossene Kette von Addierern entsteht.

Die Ausgänge des Akkumulators führen auf die Eingänge der Addierer. Die Ausgänge des Registers 2 können direkt oder negiert an den Addierer abgegeben werden. Dies ist abhängig von der Art des Steuersignals 1. Der Summenausgang des Addierers führt zu den Eingängen der Flipflops des Akkumulatorregisters. Abb. 2.22 zeigt einen Ausschnitt aus einem solchen Paralleladdierwerk.

Unter der Voraussetzung, daß die erste Zahl im Akkumulator steht und die zweite Zahl dem Register 2 auf den Eingängen angeboten wird, läuft eine Addition wie folgt ab: Durch einen Taktimpuls wird die zweite Zahl in das Register 2 eingelesen. Am Steuereingang 1 liegt „1-Signal", so daß nach der Einstellzeit der Flipflops des Registers 2 beide Zahlen am Eingang der Addierer anliegen. Die Summe der jeweils beiden Ziffern stellt sich fast augenblicklich ein. Anders sieht es mit dem Übertrag aus. Es kann nämlich sein, daß ein Übertrag in der ersten Stelle entsteht, bis zur letzten Stelle durchläuft und wieder bei der ersten Stelle zurückkommt.

Hierzu ein Beispiel:

1 1 1 1 1 1 1	Zahl 1
0 0 0 0 0 0 1	Zahl 2
1 1 1 1 1 1 1	Übertrag
0 0 0 0 0 0 0	
1	Addition des Übertrages
0 0 0 0 0 0 1	

Müssen viele Stellen addiert werden (bis zu 60 Dualstellen), so kann es eine erhebliche Zeit dauern, bis sich das endgültige Ergebnis einstellt. Die Additionszeit wird deshalb fast ausschließlich durch die Laufzeit des Übertrags bestimmt. Es gibt eine Reihe von Verfahren, sowohl elektrischen als auch logischen Aufbaus, um die Zeit der Übertragsfortpflanzung zu vermindern. Diese Verfahren berühren jedoch nicht die prinzipielle Wirkungsweise des Paralleladdierwerkes; deshalb wird auf eine nähere Beschreibung dieser Verfahren verzichtet. Hat sich also das Ergebnis unter Berücksichtigung des durchlaufenden Übertrags eingestellt, liegt das Ergebnis auch am Eingang des Akkumulatorregisters. Durch einen Taktimpuls wird dann das Ergebnis in den Akkumulator eingelesen.

Bei einer geforderten Subtraktion wird durch Anlegen von „0-Signal" auf den Steuereingang 1 erreicht, daß die negierten Ausgänge des Registers 2 auf die Addierer geschaltet werden. Dies entspricht der Bildung des (B-1)-Komplementes. Die nachträgliche Addition von 1 geschieht automatisch durch die Abgabe des Übertrags der höchsten Stelle auf den Addierer der ersten Ziffer. Das Ergebnis der Subtraktion wird dann durch einen Taktimpuls in das Akkumulatorregister eingelesen und steht anschließend nach außen zur Verfügung.

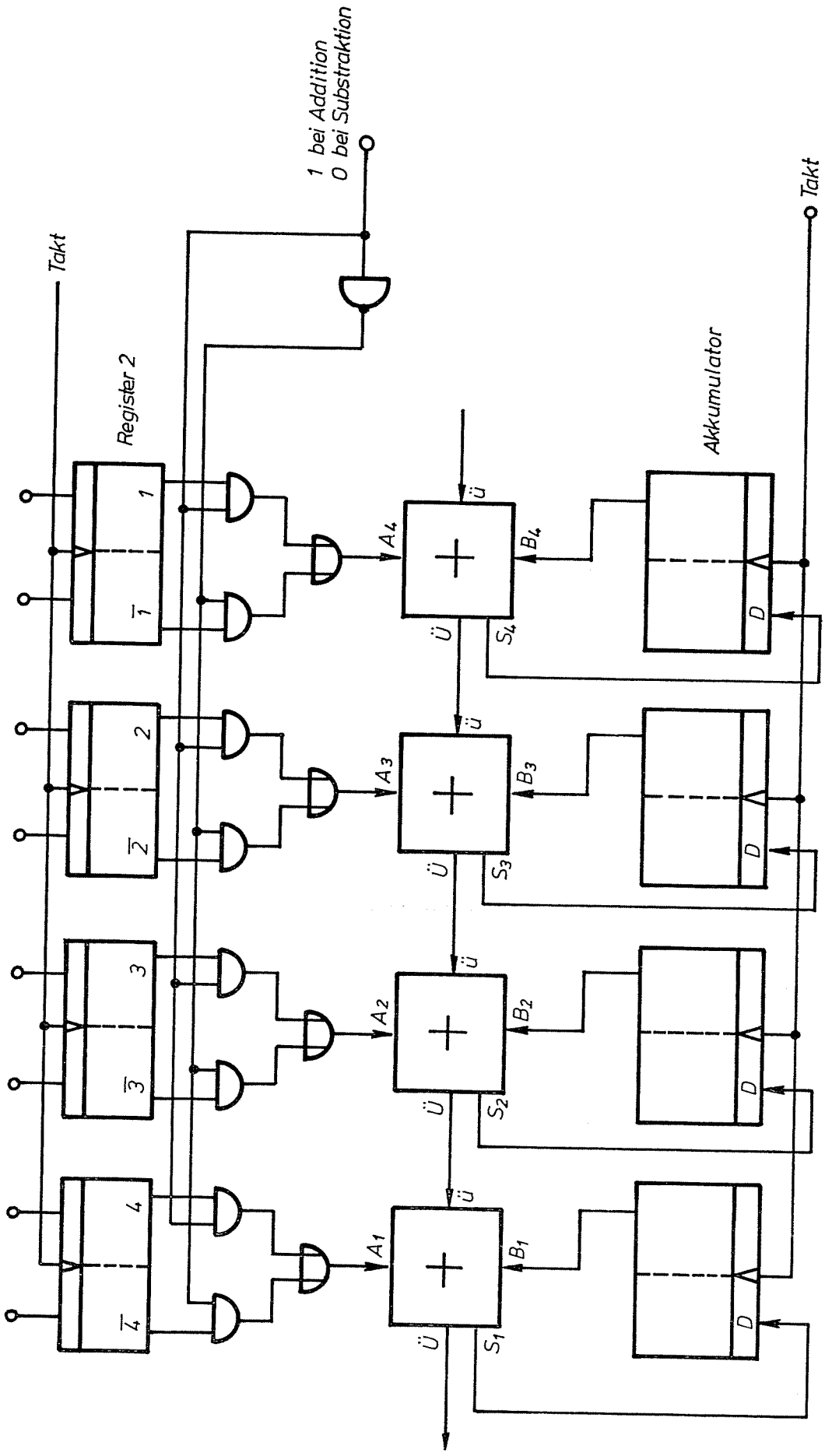


Abb. 2.22 — Paralleladdierwerk für duale Zahlen

Serienparalleladdierwerk

Einen Kompromiß zwischen notwendigem Aufwand und erreichbarer Geschwindigkeit bildet das Serienparalleladdierwerk. Hier werden die zu verarbeitenden Zahlen in Zifferngruppen aufgeteilt, und die Ziffern einer Gruppe werden parallel verarbeitet. Die Gruppen bestehen z.B. aus 8 Bits = 1 Byte oder — wie bei einem Dezimaladdierwerk — aus den vier Bits einer Tetrade zum Verschlüsseln einer Dezimalziffer. Als Beispiel für ein solches Rechenwerk soll eine mögliche Konfiguration eines Dezimalrechenwerkes angedeutet werden.

In Abb. 2.23 sind nur die Datenwege eingezeichnet, alle Steuerleitungen fehlen. Die Datenleitungen verbinden die einzelnen Einheiten mit der gemeinsamen Datenbusleitung. Die Kreise in diesen Leitungen deuten Schaltstellen an; UND-Schaltungen, die durch das Leitwerk geschaltet werden können. Schaltstellen, die durch einen Kreis angedeutet sind, sind für eine Taktzeit offen, Schaltstellen mit zwei Kreisen sind für zwei Taktzeiten geöffnet.

Das Rechenwerk soll einen Halbleiterspeicher mit 12 x 4 Bits besitzen. Jeweils 4 Bits werden durch eine Adresse angesprochen. Denkt man sich die Speicherkapazität von 48 Bits auf zwei Register aufgeteilt (Akkumulator und Register 2), so erhält jedes Register 24 Bits. Diese liegen in 6 Speicherzellen mit aufeinanderfolgenden Adressen. Das Adreßregister enthält die Adresse für diesen Speicher, wobei der Inhalt über die Datenbusleitung verändert werden kann.

Eine Addition geschieht folgendermaßen: Vom Speicher her werden die beiden Zahlen, die verknüpft werden sollen, in den Halbleiterspeicher eingelesen. Es werden hierzu die entsprechenden Schaltstellen vom Leitwerk geöffnet, zuerst die vom Speicher auf die Datenbusleitung, dann die Schaltstelle zum Halbleiterspeicher. Weiter wird nach jedem Transport einer Tetrade die Adresse im Adreßregister um 1 erhöht. Sind beide Zahlen im Halbleiterspeicher, beginnt die eigentliche Addition.

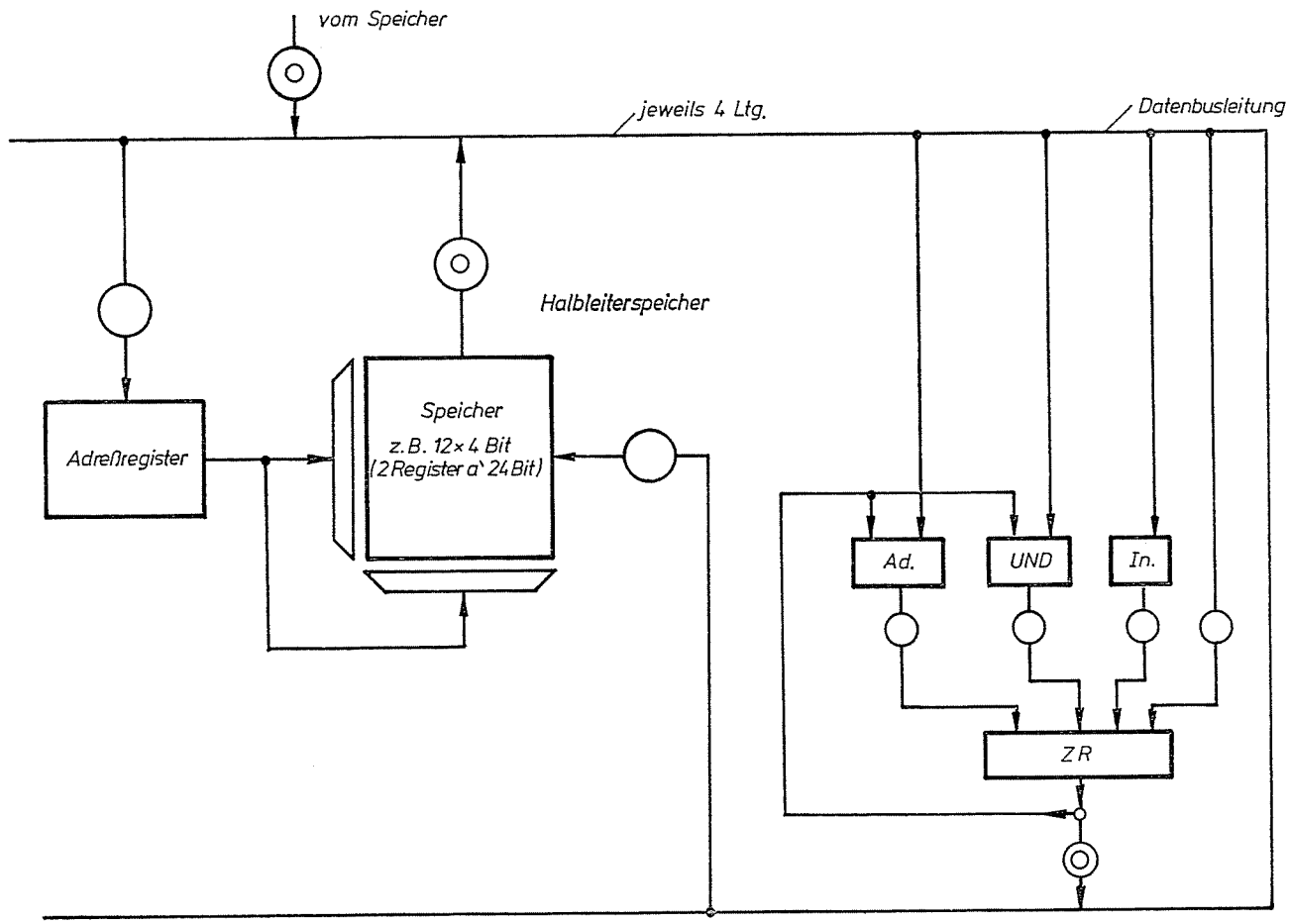


Abb. 2.23 — Serienparalleladdierwerk für Dezimalzahlen

Die erste Tetrade der zweiten Zahl wird in das Zwischenregister ZR eingelesen. Dann wird die erste Tetrade der ersten Zahl auf die Datenbusleitung geschaltet. An der Addierschaltung stehen somit beide zu addierenden Tetraden und es wird die Summe gebildet, natürlich mit der entsprechenden Korrektur. Gleichzeitig wird z.B. auch die UND-Verknüpfung gebildet, hier angedeutet für eine mögliche logische Verknüpfung. Wird jetzt die Verbindung zwischen der Addierschaltung und dem Zwischenregister ZR hergestellt, so wird mit dem nächsten Takt das Ergebnis in das Zwischenregister ZR eingelesen. Dieses Ergebnis wird dann wieder auf die Datenbusleitung eingespeist und in den Halbleiterspeicher übernommen. Das wiederholt sich bei jeder zu verarbeitenden Tetrade. Selbstverständlich wird durch entsprechende Schaltmittel dafür gesorgt, daß der Übertrag stellenrichtig addiert wird. Bei einer Subtraktion wird die Tetrade des Subtrahenden negiert über den Inverter I_n in das Zwischenregister ZR eingelesen, anschließend addiert. Das Leitwerk sorgt für den richtigen Ablauf der Addition und Subtraktion durch das notwendige Schließen der Schaltstellen des Rechenwerkes.

2.2.4.2. Multiplikation

Duale Parallelmultiplikation

Im Gegensatz zur Addition hat bei einer Multiplikation das Ergebnis soviel Stellen, wie die Faktoren zusammen Stellen haben. Das Ergebnis füllt also zwei Register normaler Wortlänge. Zur Multiplikation wird deshalb das Rechenwerk mit drei Registern ausgestattet; außer dem Akkumulator und dem Register 2 noch mit einem Register 3. Das Register 3 wird auch häufig Akkumulatorverlängerung genannt, da es mit dem Akkumulator gekoppelt ein einziges großes Register doppelter Wortlänge ergibt. Das Prinzipbild eines Parallelmultiplikationswerkes für Dualzahlen sowie den Ablauf einer solchen Multiplikation zeigt die Abb. 2.24.

Bei Beginn der Rechnung steht der eine Faktor, der Multiplikand, im Register 2; der andere Faktor, der Multiplikator, steht im Register 3, der Akkumulator ist leer. Abhängig von der niedrigsten Stelle des Multiplikators im Register 3 (durch die UND-Verknüpfungen) wird der Inhalt des Registers 2 entweder zum Addierer durchgeschaltet (dies entspricht einer Multiplikation mit 1) oder gesperrt (dies entspricht der Multiplikation mit 0). Nach der Addition und

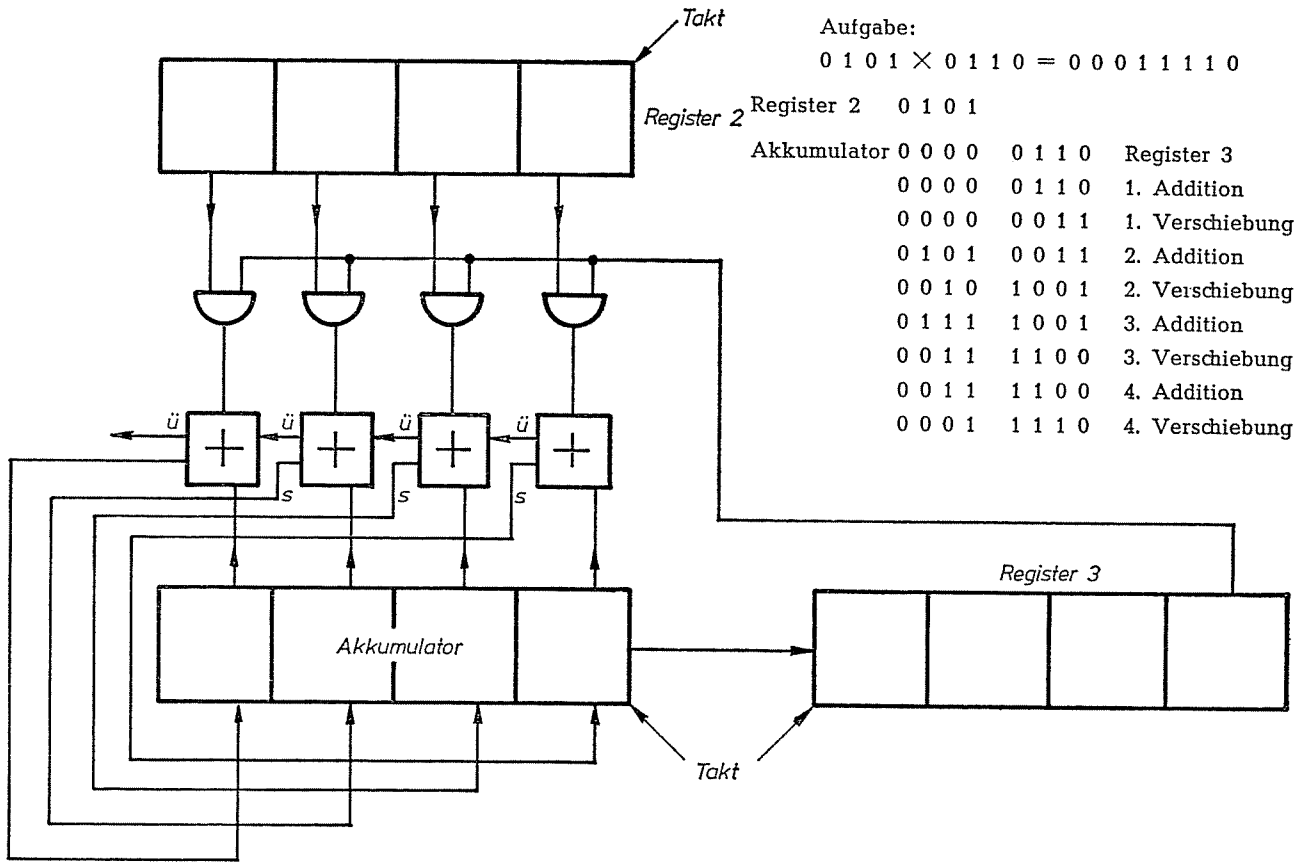


Abb. 2.24 — Prinzipbild eines Parallelmultiplikationswerkes

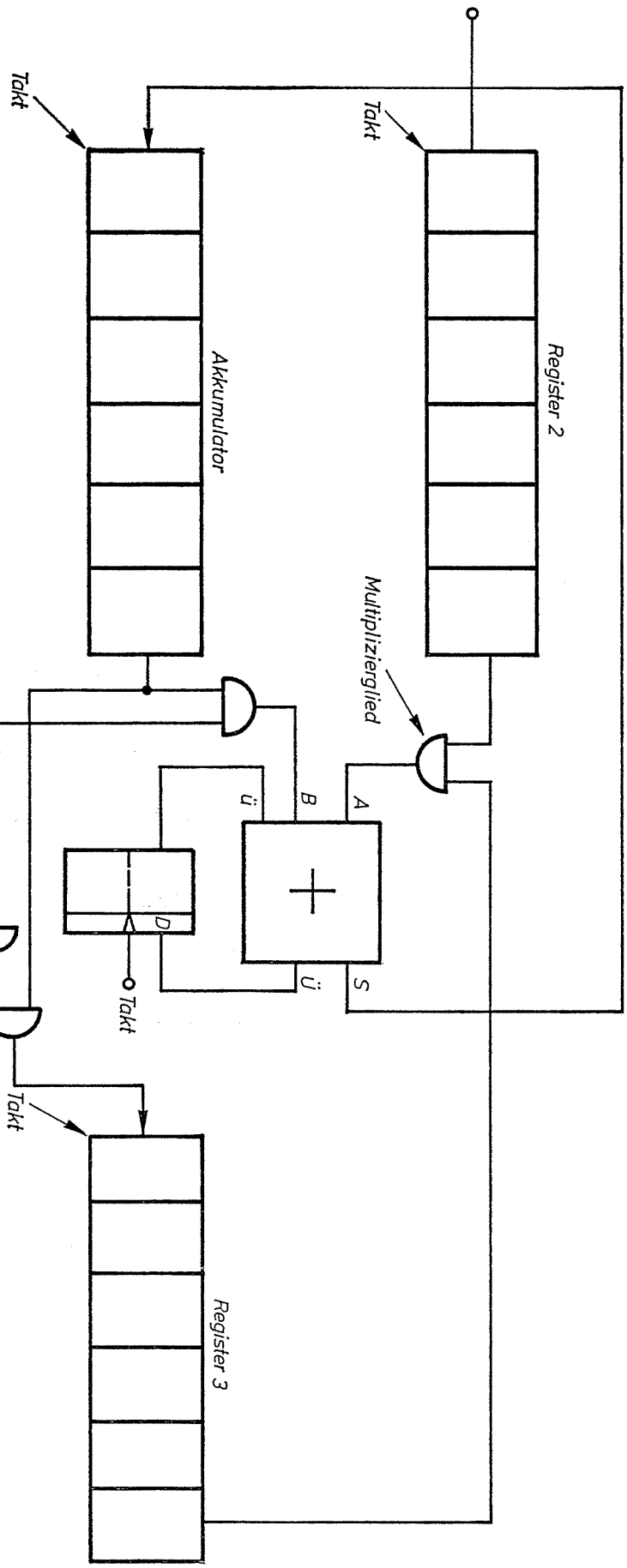


Abb. 2.25 — Prinzipbild eines Serienmultiplikationswerkes

Übernahme in den Akkumulator durch einen entsprechenden Takt wird durch einen Schiebeimpuls für Akkumulator und Register 3 deren Inhalt um eine Stelle nach rechts verschoben. Dadurch wird die nächste stellenrichtige Addition vorbereitet. Die letzte Stelle des Multiplikators im Register 3 geht dabei verloren, sie ist ja auch schon verarbeitet. Wieder erfolgt, abhängig vom Inhalt der letzten Stelle des Registers 3, die jetzt stellenrichtige Addition des Multiplikanden. Nach insgesamt vier Additionen und vier Verschiebungen steht das Ergebnis der Multiplikation im Akkumulator und im Register 3. Der Multiplikator ist verlorengegangen. **Eine Parallelmultiplikation setzt sich also aus einer Addition und einer Verschiebung je Stelle zusammen.**

Duale Serienmultiplikation

Ein Serienmultiplizierwerk baut sich auf ein Serienaddierwerk auf. Zum Akkumulator wird auch hier ein Register 3 (Akkumulatorverlängerung) hinzugefügt. Es enthält zu Beginn der Rechnung den Multiplikator. Der Multiplikand steht im Register 2, der Akkumulator ist leer. Abb. 2.25 zeigt das Prinzip eines solchen Multiplikationswerkes. Abhängig vom Inhalt der letzten Stelle des Registers 3 wird der Inhalt des Registers 2 zum Inhalt des Akkumulators hinzuaddiert. Hierzu erhalten Akkumulator, Register 2 und Übertragsflipflop U Taktimpulse; bei sechs Dualstellen also 6 Takte. Die letzte Stelle des Multiplikators ist damit verarbeitet. Durch einen zusätzlichen, auf Akkumulator und Register 3 wirksamen Schiebeimpuls und Umschalten auf den Steuerleitungen wird der Akkumulator vom Addierer getrennt und mit dem Register 3 gekoppelt. Der gesamte Inhalt wird dadurch um eine Stelle nach rechts verschoben. Es folgt dann wieder die Addition des Registers 2.

Nach insgesamt 6 x 6 Schiebetakten für Akkumulator und Register 2 und 6 Schiebeimpulsen für Akkumulator und Register 3 steht das Ergebnis im Akkumulator und Register 3.

Dezimalmultiplikation

Bei einer dezimalen Multiplikation gilt wieder das Ihnen bekannte „Einmaleins". Die Faktoren können Werte zwischen 0 und 9 annehmen, das Produkt sogar Werte zwischen 1 und 81. Während bei der Dualmultiplikation eine UND-Schaltung für die Multiplikation zweier Dualziffern ausreicht, muß hier eine richtige Einmaleinstafel geschaffen werden. Einen Ausschnitt aus einer solchen Tafel zeigt die Abb. 2.26.

Der Tetraden-Code wird durch eine Entschlüsselungsmatrix entschlüsselt, so daß jeweils nur eine Zeilen- und eine Spaltenleitung „1-Signal" führt. In den Kreuzungspunkten sitzen UND-Schaltungen, die dann ansprechen und das zweiziffrige Produkt bestimmen, das nach außen gegeben werden soll. Eine Multiplikationstafel hat also zwei Eingänge für jeweils 1 Tetrade, entsprechend Ziffer 1 x Ziffer 2 und zwei Ausgänge, einen für den Produkteiner und einen für den Produktzehner. Die Multiplikation wird ähnlich wie in einem dualen Serienmultiplizierwerk abgewickelt. Die UND-Schaltung wird hier durch die Einmaleinstafel und einen nachgeschalteten Addierer ersetzt. In die Produktzehnerleitung muß eine Verzögerung (Flipflop) eingeschaltet werden, um die stellenrichtige Addition der Zehnerwerte zu ermöglichen.

Etwas geringeren Aufwand erhält man, wenn die Multiplikation auf eine wiederholte Addition zurückgeführt wird. Eine Multiplikationstafel wird dann nicht mehr benötigt.

Die Zahl der Additionen (bis maximal 9) läßt sich außerdem verringen, wenn man ständig einige Vielfache des Multiplikanden erzeugt. Die Herstellung der Vielfachen ist verhältnismäßig einfach und bei geschickter Ausnutzung kommt man mit 2 Additionen pro Multiplikation von zwei Ziffern aus. Die Abb. 2.28 zeigt das Prinzip hierfür. Es ist außerdem von Vorteil, daß diese Vielfachen auch für die Division verwendet werden können.

2.2.4.3. Division

Bei Rechenwerken, bei denen die Division nicht durch ein Programm (Software) abgewickelt wird sondern festverdrahtet ist, ist der Rechengang ähnlich dem Verfahren, wie Sie von Hand dividieren. Der Divisor wird so oft von dem Dividenden abgezogen, bis der Rest kleiner ist als der Divisor. Die Anzahl der möglichen Subtraktionen ergibt die erste Stelle des Quotienten. Anschließend wird das Verfahren mit dem um eine Stelle verschobenen Rest wiederholt. Dies geht so weiter, bis alle Stellen des Dividenden erfaßt sind und das Ergebnis ermittelt ist. Während man nun im Dualzahlensystem mit einer Subtraktion immer jeweils eine Stelle des Quotienten ermittelt hat, sind im Dezimalzahlensystem bis zu 9 Subtraktionen möglich. Eine Ersparnis bedeutet auch hier die Anwendung

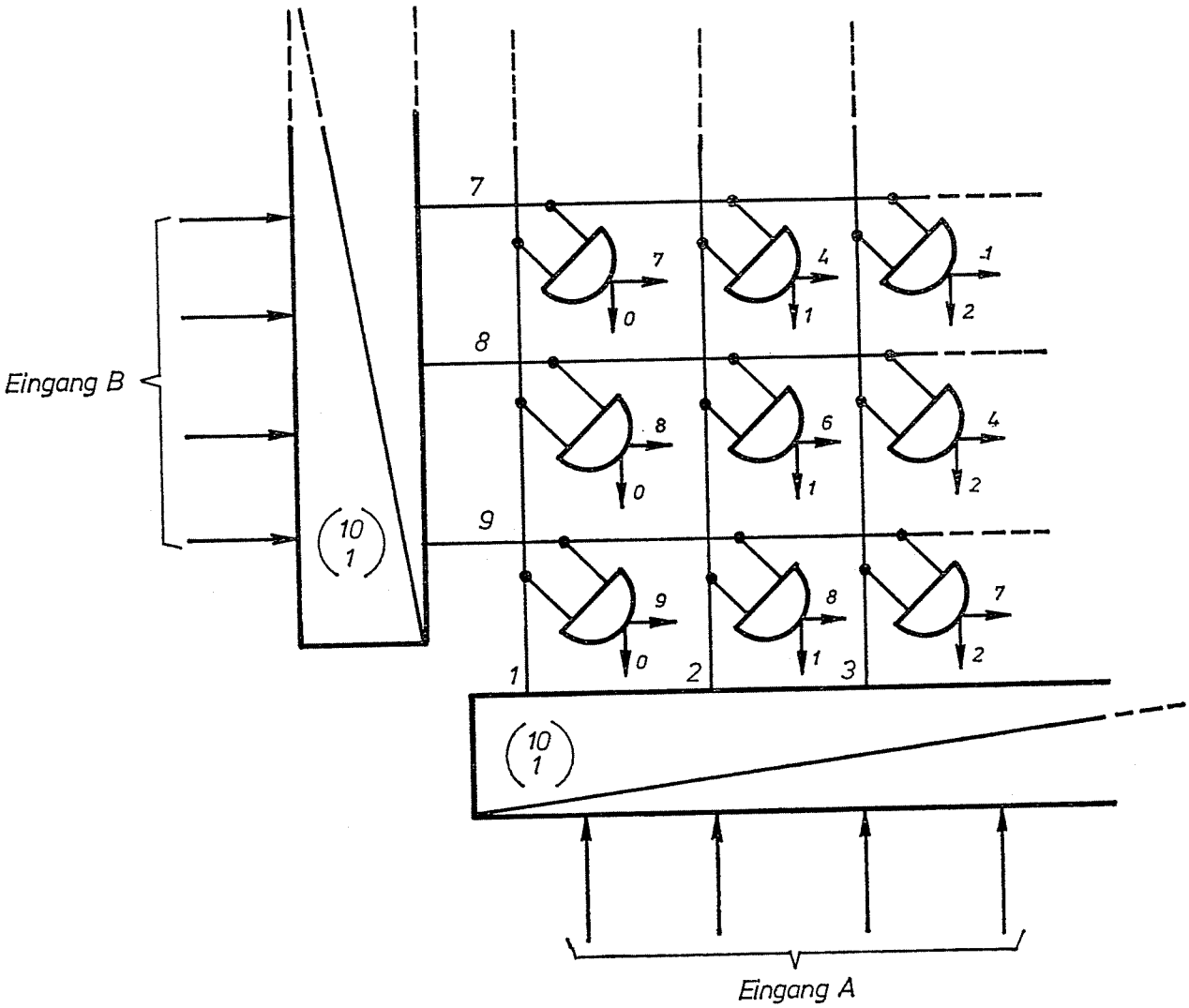


Abb. 2.26 — Multiplikationstafel für dezimale Multiplikation

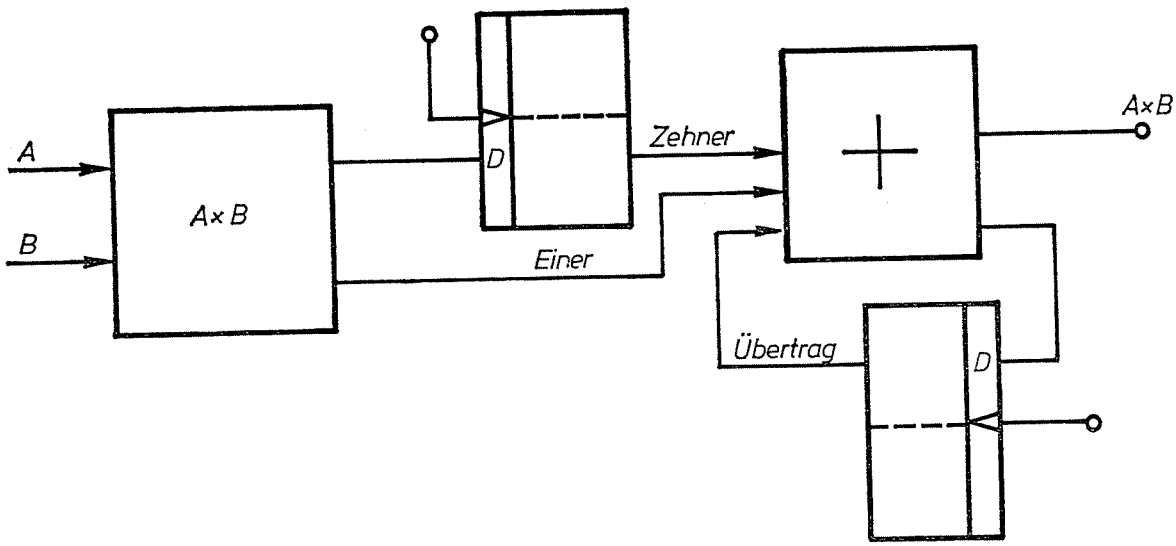


Abb. 2.27 — Stellenrichtige Addition der Einer und Zehner

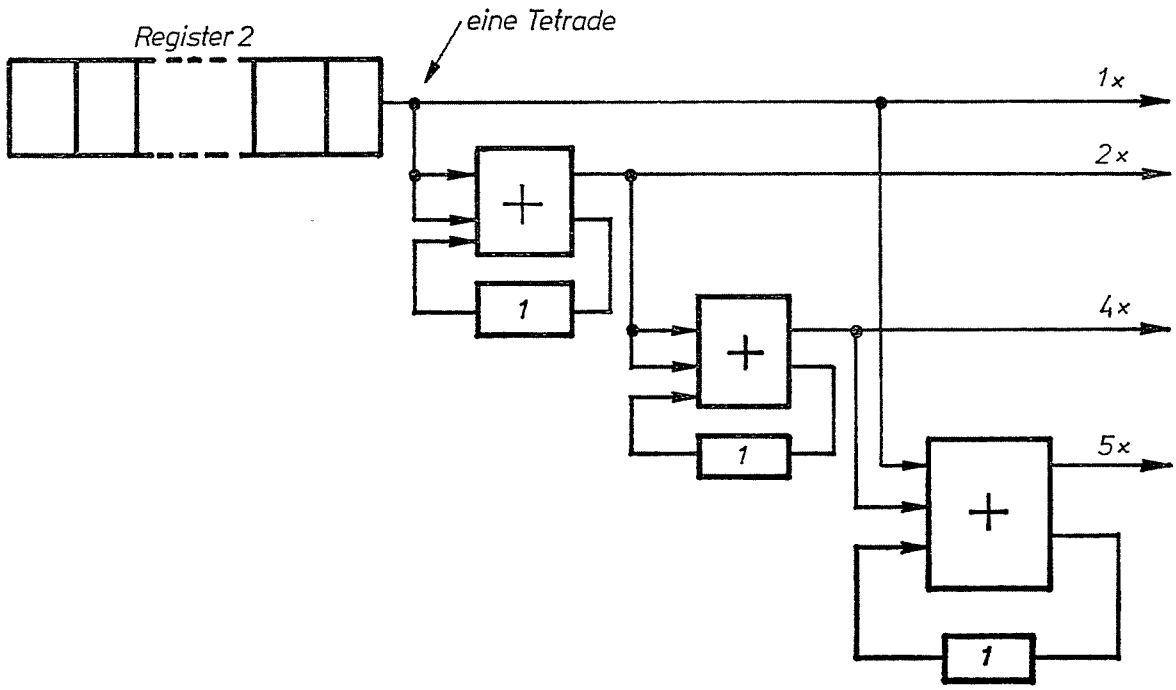


Abb. 2.28 — Erzeugen der Vielfachen

von Vielfachen des Divisors. An einem Beispiel wollen wir die Division im Dualzahlensystem erklären. Es liegt folgende Aufgabe vor:

Dividend Divisor

11 00 : 0010 =

entspricht 12 : 2 = 6

Es ist nun von der ersten Stelle des Dividenden der Divisor abzuziehen. Dies geschieht durch Addition des B-Komplementes und Unterdrükkung des Übertrages. Glückt die Subtraktion, so ist der Rest eine positive Zahl oder Null. Es wird dann eine 1 für den Quotienten gesetzt. Glückt die Subtraktion nicht, so ist der Rest negativ, die Quotientenstelle wird 0.

In diesem Fall müßte die Subtraktion rückgängig gemacht und mit dem um eine Stelle nach links verschobenen Rest erneut versucht werden. Dies ist zeitaufwendig und läßt sich verkürzen. Nehmen wir an, vorher war die Zahl R vorhanden, dann ergibt sich nach mißglückter Subtraktion R—y, wobei y dem Divisor entspricht. Es muß also wieder y addiert werden und es entsteht wieder R. Dieser Wert wird dann um eine Stelle nach links verschoben. Dies bedeutet aber im Dualzahlensystem eine Multiplikation mit 2, es entsteht also 2R. Wird jetzt die Subtraktion erneut versucht, so ergibt sich als Ergebnis 2R—y. Dies läßt sich aber aus R—y erreichen durch eine Linksverschiebung (2(R-y)) und Addition von y

$$2(R-y) + y = 2R-y$$

Hierdurch wird eine Addition eingespart. Ist der Rest also negativ, so erfolgt eine Linksverschiebung und eine Addition des Divisors.

Ist der Rest dagegen Null, so ist nach Setzen einer Eins in die Quotientenstelle weder eine Addition noch eine Subtraktion notwendig. Es folgt dann sofort die nächste Linksverschiebung.

Es gibt also drei Teiloperationen:

1. Verschieben des Dividenden um eine Stelle nach links.

2. a) Der Rest ist größer als Null, es wird das B-Komplement des Divisors addiert.

   b) Der Rest ist Null, dann wird sofort zu Teiloperation 3 übergegangen.

   c) Der Rest ist kleiner als Null, also negativ, dann wird der Divisor addiert.

3. a) Ist der neue, sich durch Teiloperation 2 a) oder 2 c) ergebende Rest positiv oder Null, so wird in die Quotientenstelle eine 1 eingeschrieben.

   b) Ist der Rest negativ, wird eine Null in die Quotientenstelle eingeschrieben, ebenfalls bei Teiloperation 2 b).

Mit diesen Regeln können wir nun die gestellte Aufgabe lösen.

```
                    Dividend      Divisor   Quotient
                 0 0 0 0 1 1 0 0 : 0 0 1 0 = 0 1 1 0
               +   1 1 1 0
negativ            1 1 1 1
                     0 0 1 0
Übertrag
und positiv        1 0 0 0 1
                       1 1 1 0
Übertrag und Null  1 0 0 0 0
ebenfalls Null       0 0 0 0
```

In einem Rechenwerk sind für die Division ebenfalls 3 Register vorhanden. Am Anfang der Rechnung befindet sich der Divisor im Register 2, der Dividend im Register 3 und der Akkumulator ist leer. Akkumulator und Register 3 sind als Schieberegister miteinander gekoppelt. Am Ende der Rechnung befindet sich im Register 3 der Quotient und im Akkumulator der Rest. Die gestellte Aufgabe ist nachfolgend noch einmal in registergerechter Schreibweise dargestellt.

```
| 0  0  1  0 |        Register 2

| 0  0  0  0 | | 1  1  0  0 |
  Akkumulator      Register 3

0 0 0 0 1 1 0 0     Anfangsstellung
0 0 0 1 1 0 0 0     1. Verschiebung
1 1 1 1 1 0 0 [0]   Addition des Komplementes
1 1 1 0 0 0 [0] 0   2. Verschiebung
0 0 0 1 0 0 [0 1]   Addition des Divisors
0 0 1 0 0 [0 1] 0   3. Verschiebung
0 0 0 0 0 [0 1 1]   Addition des Komplementes
0 0 0 0 [0 1 1 0]   Auffüllen mit Null
```

Dieser Ablauf ist bei Parallel- und Serienrechenwerken der gleiche, wobei eben bei Serienrechenwerken die Steuerung komplizierter ist. Ähnlich ist sie auch bei Dezimalrechenwerken.

2.2.5. Zusammenfassung

Die meisten Rechenwerke enthalten 3 Register, den Akkumulator, das Register 2 und das Register 3; sie nehmen die zu verarbeitenden Operanden auf. Weiter haben wir Operationsbildner wie Addierer, Einmaleinstafel oder Vervielfacher.

Operationssteuerungen beeinflussen den Ablauf der Rechnung und schalten die notwendigen Schaltmittel. Sie stellen gegebenenfalls notwendige Verbindungen her und entscheiden aufgrund des vorliegenden Befehls, ob die Zahl oder ihr Komplement addiert werden soll. Bei der Multiplikation wird durch die Operationssteuerung außerdem das Vorzeichen der Rechnung ermittelt.

Jede Rechenoperation kann in folgende Schritte aufgeteilt werden:

1. **Einspeichern der Operanden in das richtige Register; dabei kann die einlaufende Zahl eventuell komplementiert werden.**
2. **Berechnung des Vorzeichens bei der Multiplikation und Division.**
3. **Ablauf der eigentlichen Rechnung.**
4. **Meldung an das Leitwerk, daß die Rechnung beendet ist.**

Man erkennt, daß sich jeder Rechenvorgang aus kleinen Teiloperationen zusammensetzt; es läuft ein kleines Mikroprogramm ab, das meistens festverdrahtet ist.

2.3. Speicher

2.3.1. Arbeitsspeicher

2.3.1.1. Forderungen an Arbeitsspeicher

Im rein logischen Konzept einer EDV-Anlage kommt dem Arbeitsspeicher eigentlich keine große Bedeutung zu; sein Vorhandensein wird vorausgesetzt, von Interesse sind lediglich seine Kapazität und seine Organisation. Der Speicher als Teil der Zentraleinheit spielt also im Vergleich zum Rechenwerk nur eine untergeordnete Rolle. In der praktischen Konzeption und Ausführung eines Computers liegen die Gewichte ganz anders. Heute sind Schaltkreistechniken mit Schaltzeiten um 1 ns — wie z.B. ECL — kein Luxus mehr; d.h., Taktzeiten von 20—30 ns ließen sich ohne Probleme realisieren. Dagegen steht aber die unverhältnismäßig lange Zugriffszeit — das ist die Zeit, die zum Finden und Auslesen einer Information benötigt wird — von mindestens 200—300 ns zum Kernspeicher. Was nützt also die schnellste Schaltkreistechnik, wenn die Zentraleinheit 10 Taktzeichen und mehr auf das Zubringen von Informationen warten muß. Daher versucht jeder Hersteller von Computern, Speichermedien — das ist das zur Speicherung verwendete Material — einzusetzen, die in erster Linie eine geringe Zugriffszeit ermöglichen. Die Speichermedien sollen außerdem eine große Kapazität bei kleinerem Raum-

volumen zu möglichst geringem Preis zulassen und sie sollten möglichst direkt durch IC ansteuerbar sein.

Die ersten beiden Punkte — wesentlich höhere Geschwindigkeit und hohe Kapazität — sind die beiden wichtigsten. **Die Einschränkung, daß dabei keine zu hohen Kosten pro gespeichertes Bit anfallen dürfen, ist leicht an folgendem Beispiel erkennbar: wenn die Kosten pro Bit nur um 10 Pfennig steigen, so verursacht das bei einer EDV-Anlage mit einem 64 K großen Speicher einen Preisanstieg von DM 58.982,40 (für 8 Bits + 1 Parity-Bit pro Wort)!**

Man erkennt, daß die Größe des Arbeitsspeichers ganz bedeutend in den Preis der EDV-Anlage eingeht und außerdem, daß seine Geschwindigkeit auf die Arbeitsgeschwindigkeit der Anlage einen sehr wesentlichen Einfluß hat. Der dritte Punkt, Kompatibilität mit den IC der Zentraleinheit, ist ganz sicher für die logische Konzeption nicht entscheidend. Er geht aber in Geschwindigkeit und Preis der Anlage mit ein. Denn jedes Anpassen der IC-Signale an ein anderes Spannungsniveau erfordert neben dem Aufwand an meist aus diskreten Elementen aufgebauten Schaltungen auch zusätzliche Geschwindigkeitseinbuße.

Der ideale Arbeitsspeicher sollte also wie folgt aussehen:

1. **Zugriffszeit im Größenbereich der Taktzeit,**
2. **beliebige Größe ohne zu hohe Kosten** und
3. **möglichst gleiche, mindestens aber kompatible Schaltkreistechnik zu der der Zentraleinheit.**

An wichtigen Begriffen wäre zu klären:

Adresse: Jede Speicherzelle hat eine genau definierte Lage, die in ihrer Adresse angegeben wird. Man muß also zum Wiederfinden einer im Speicher abgelegten Information deren Adresse kennen. Im allgemeinen bezieht sich die Adresse auf ein Wort, also nicht auf ein Bit.

Adreßvolumen: Maximal mögliche Zahl von Adressen.

Organisation: Unter Speicherorganisation ist das Ordnungsprinzip im Speicheraufbau zu verstehen. Vor allem wird unter diesem Begriff definiert, wieviel Bits zu einem Wort zusammengefaßt sind und ob zum Beispiel bei Aufruf einer Adresse mehrere Worte automatisch ausgelesen werden.

Lesen: Die Übernahme einer Information aus dem Speicher.

Schreiben: Das Ablegen einer Information im Speicher.

Zugriffszeit: Die Zeit, die vom Aufruf der Adresse bis zum Erscheinen des ausgelesenen Signals am Ausgang des Speichers vergeht.

Zykluszeit: Bei verschiedenen Speichermedien wird die Information beim Lesen zerstört. Soll sie erhalten bleiben, so muß sie nach dem Lesen neu eingeschrieben werden. Erst danach kann der Speicher zu erneuter Arbeit herangezogen werden. Die Zykluszeit ist also die Zeit, die vom Aufruf der Adresse bis zum Ende des Wiedereinschreibvorganges vergeht.

2.3.1.2. Speichermedien

Ferritkerne

Die Geschichte des Arbeitsspeichers ist die des Magnetkernspeichers, den Forrester und Rajchmann im Jahr 1951 erfanden. Die Speicherelemente sind Ringkerne aus Ferritmaterial, einer Mischung aus Eisenoxid mit Oxiden von Mangan, Magnesium und Zink. Durch Pressen und Sintern werden runde Kerne erzeugt, die in den ersten Jahren Außendurchmesser von 2 mm hatten und mit denen eine Zykluszeit von 10 μs erreichbar war. Für sehr schnelle Speicher werden heute Kerne mit 0,3 mm Außendurchmesser eingesetzt; sie lassen Zykluszeiten von 0,5 μs zu. Die Speicherfähigkeit eines derartigen Kernes beruht auf den ferrimagnetischen Eigenschaften des Materials. Es behält, wenn es mit entsprechend hoher Feldstärke beaufschlagt wurde, die Richtung des erregenden Magnetfeldes bei, auch wenn dieses wieder verschwindet. Ein Ringkern läßt sich in zwei Richtungen — links- oder rechtsherum — magnetisieren, d.h., er kann die beiden digitalen Zustände „0" oder „1" einnehmen. Mit anderen Worten: er speichert genau ein Bit. Ausführlich ist der Ferritringkern im Handbuch der Elektronik, Teil 2, Digitaltechnik, beschrieben. Hier sollen nur die wichtigsten Begriffe noch einmal erläutert werden.

Den Zusammenhang zwischen magnetischer Feldstärke H und Induktion B beschreibt die **Hystereseschleife.** Für Speicherzwecke sollte sie möglichst Rechteckcharakter (daher auch der Begriff Rechteckferrite) zeigen. In Abb. 2.29 ist eine typische Hystereseschleife dargestellt.

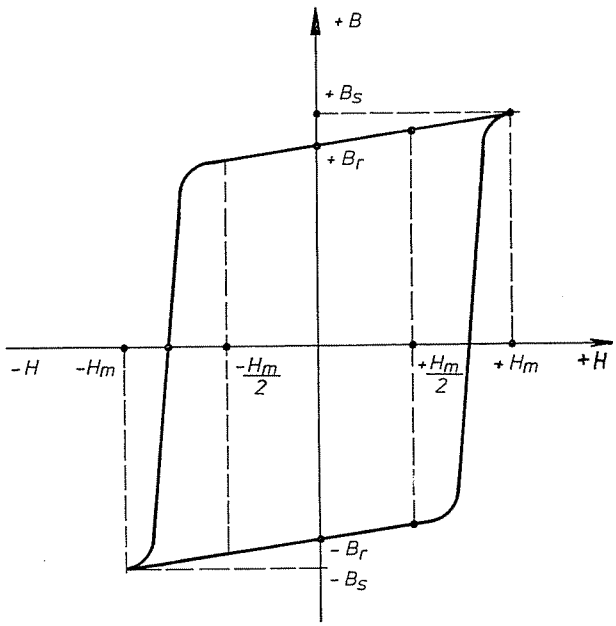


Abb. 2.29 — Hystereseschleife eines Rechteckferriten

Wenn die magnetische Feldstärke von 0 zum Wert + H_m steigt, nimmt die Induktion B den Wert + B_s ein. Lag sie vorher im Punkt — B_r, so ändert sie sich stark erst nach Überschreiten eines bestimmten Betrages von H, geht dann aber sehr schnell in den positiven Bereich über. In diesem Moment dreht sich die Magnetisierungsrichtung um. Wenn die Feldstärke H auf den Wert 0 zurückkehrt — der Strom in der Erregerspule abgeschaltet wird —, bleibt die Magnetisierungsrichtung des Kernes erhalten, die Induktion geht lediglich auf den Wert der positiven Remanenz + B_r zurück. Der Kern hat also das kurzzeitige Erregen mit + H_m gespeichert. Soll der Kern magnetisch wieder umgeklappt werden, also die Induktion — B_r annehmen, so muß eine negative Feldstärke auf ihn einwirken. Sie muß mindestens den Wert der Koercitivfeldstärke — H_c überschreiten, dann nimmt der Kern die Lage — B_s, bzw. nach Abschalten des Erregerfeldes — B_r ein. Der positiven Remanenz + B_r ordnet man den Wert der digitalen „1", — B_r den Wert „0" zu.

Der Betrag der Feldstärke H_m, der zum Umklappen der Magnetisierungsrichtung notwendig ist, hängt eng mit den äußeren Abmessungen des Kernes zusammen. Je kleiner der Kern ist, um so geringer die Feldstärke und damit der Ansteuerstrom und um so schneller ist der Umklappvorgang selbst, also die Schaltgeschwindigkeit. Zur Auswahl eines ganz bestimmten Kernes unter den Hunderttausenden eines Speichers wird das **Koinzidenzprinzip** angewendet (Abb. 2.30). **Hierbei sind durch jeden Kern zwei Drähte gefädelt, die jeweils den halben Strom führen dürfen, der zum Umklappen notwendig ist.**

Erst wenn **beide Drähte gleichzeitig** mit $\frac{I}{2}$ erregt werden, reicht die erzielte Feldstärke aus, den Kern in die andere Remanenzlage zu bringen. Vom Kern muß daher gefordert werden, daß die Feldstärke $\frac{H_m}{2}$ auf gar keinen Fall eine Änderung der Remanenzlage hervorrufen darf. Die Drähte bilden ein Koordinatennetz, in deren Kreuzungspunkten die Kerne sitzen. Die Lage jedes einzelnen Kernes ist also durch die beiden

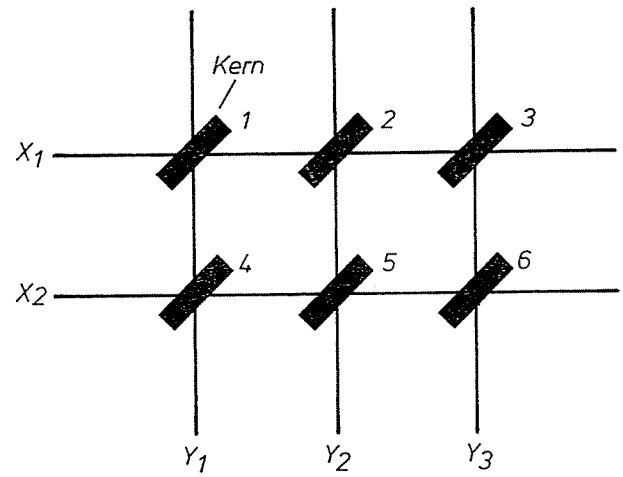


Abb. 2.30 — Koinzidenzprinzip

Koordinaten x und y eindeutig definiert. Soll z.B. der Kern 5 in Abb. 2.30 ausgewählt werden, müssen die Drähte x_2 und y_2 Strom führen. Nur durch diese Anordnung der Speicherzellen kann der Aufwand für das Finden einer bestimmten Information in wirtschaftlich vertretbaren Grenzen gehalten werden.

Eine im Ferritkern abgespeicherte Information läßt sich nur dadurch lesen, daß der Kern in Richtung der negativen Remanenz magnetisiert wird. **Hatte er eine „1" gespeichert, so klappt er jetzt auf den Wert —B_r um, bei einer „0" ändert er seinen Zustand nicht.** Das Umklappen des magnetischen Feldes im Kern erzeugt in einem ihn umgebenden elektrischen Leiter einen Spannungsimpuls. Dieser Impuls ist eine eindeutige Aussage dafür, daß eine „1" gespeichert war. Beim Lesen einer „0" wird zwar auch der Teil von —B_r bis —B_s der Hystereseschleife durchlaufen, also auch ein Spannungsimpuls induziert; aber dieser ist im Verhältnis zu dem der gelesenen „1" wesentlich geringer, eine eindeutige Aussage mithin möglich. Der Leseimpuls wird auf allen durch den Kern führenden Drähten erzeugt, also auch auf den beiden Drähten der x- und y-Koordinaten (auch Adressendrähte genannt). Wenn man deren Verhalten gleich nach dem Ansteuern beobachtet, kann eindeutig auf den Informationsinhalt der Kerne geschlossen werden. Das setzt aber eine sehr aufwendige Elektronik voraus, denn das Lesesignal ist bedeutend kleiner als Ansteuer- und Störsignale. Dieses Problem ist nur zu lösen, indem der Leseverstärker ausschließlich in der Zeit aufgesteuert wird, in der das Lesesignal zu erwarten ist. Der Aufwand an Elektronik läßt sich stark

reduzieren, wenn zusätzlich zu den Adreßdrähten noch ein sogenannter **Lesedraht** durch den Kern gefädelt wird. Wie Abb. 2.31 zeigt, kann dieser Draht **durch sämtliche Kerne einer Matrix** laufen, da während eines Lesevorganges immer nur ein Kern ausgewählt ist.

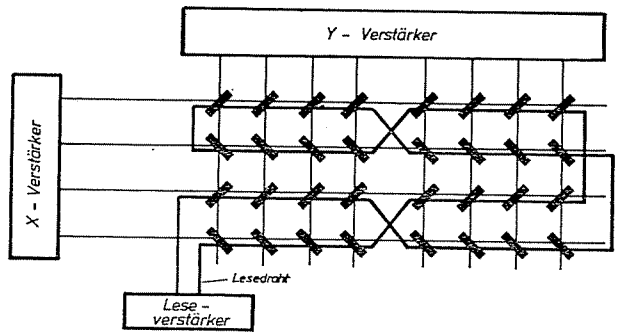


Abb. 2.31 — Verlauf des Lesedrahtes

Durch die in der Abbildung gezeigte Art der Fädelung lassen sich Störimpulse weitgehend vermeiden. Der Lesedraht verursacht natürlich zusätzliche Kosten. Es ist eine reine Wirtschaftlichkeitsfrage, wann die Kosten für den Lesedraht höher sind als der Elektronikaufwand bei nur zwei Drähten. Grundsätzlich läßt sich sagen, daß der Lesedraht in großen Ferritkernspeichern zu teuer wird. Ebenso muß er bei sehr kleinen Kernen (meistens schon bei 0,5-mm-Kernen) entfallen, weil es nicht mehr möglich ist, mehr als zwei Drähte durch das winzige Loch im Kern zu fädeln.

Arbeitsspeicher sind im allgemeinen wortorganisiert. Wenn z.B. jedes Wort aus 4 Bits besteht, läßt sich der Speicher am günstigsten durch Nebeneinanderstellen von 4 Matrizen aufbauen. **Zu einem Wort gehören dann alle Kerne gleichnamiger x- und y-Koordinaten der einzelnen Matrizen;** d.h., es können jeweils vier x- und vier y-Drähte hintereinandergeschaltet und durch gemeinsame Treiberverstärker angesteuert werden.

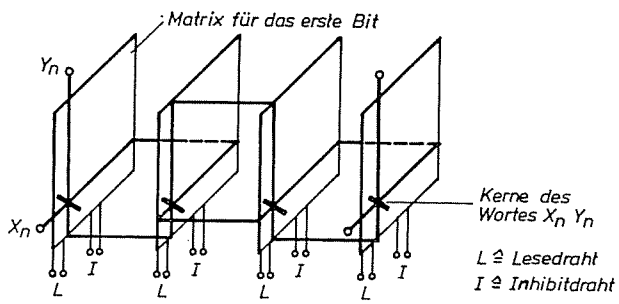


Abb. 2.32 — Aufbau eines Speichers mit 4 Bits je Wort

Abb. 2.32 zeigt den Aufbau eines derartigen Speichers. Der Vorteil, Treiberverstärker für die Adressierung sparen zu können, muß in diesem Fall mit einem Nachteil erkauft werden: da sämtliche Bits eines Wortes gemeinsam angesteuert werden, können zwar alle Bits durch den Lesevorgang in die negative Remanenz gebracht werden, aber sie könnten auch nur alle gemeinsam beim Schreiben den Zustand „1" annehmen. Dies läßt sich mit einem weiteren Draht, dem sogenannten **Blockier- oder Inhibitdraht,** vermeiden. Er wird ähnlich wie der Lesedraht durch alle Kerne einer Matrix gefädelt. Soll z.B. das zweite Bit des Wortes beim Schreiben auf „0" bleiben, so wird durch den Inhibitdraht der zweiten Matrix ein Strom geschickt, der einem halben Erregerstrom entspricht, aber in umgekehrter Richtung. Die in diesem Kern erzeugte Feldstärke ergibt sich zu

$$\frac{H_m}{2} + \frac{H_m}{2} - \frac{H_m}{2} = \frac{H_m}{2}$$

also nicht genug, um den Kern aus der Nullage zu bewegen. Der Inhibitdraht muß natürlich pro Matrix ansteuerbar sein. Verzichtet man auf ihn, müssen die Adreßdrähte in jeder Matrix separat betrieben werden können. Auch hier zeigt sich, daß die Entscheidung für oder gegen Inhibitdraht aus rein wirtschaftlichen Gesichtspunkten zu fällen ist. Für die verschiedenen Verfahren haben sich folgende Begriffe durchgesetzt: **Speicher mit 3 Ebenen oder Dimensionen und vier Drähten nennt man 3-D-Speicher, solche mit nur drei Drähten** (meist fehlt der Lesedraht) **2½-D-Speicher,** aber auch die meisten reinen Zweidrahtspeicher gehören dem 2½-D-Prinzip an, wenn sie wortorientiert aufgebaut sind.

Als 2-D-Organisation, also zweidimensional, wären noch die sogenannten Linearspeicher zu erwähnen, die zwar die kürzesten Zykluszeiten gestatten, aber den doppelten bis dreifachen Aufwand erfordern. Sie finden wegen der hohen Kosten nur ausnahmsweise als Arbeitsspeicher Verwendung und dürften künftig in jedem Fall durch preiswertere und schnellere Halbleiterspeicher völlig verdrängt werden. In Linearspeichern sind die einzelnen Bits eines Wortes hintereinander in einer Matrix angeordnet, also nicht über verschiedene Matrizen verteilt. Diese Speicher sind ausführlich, wie insgesamt die Ferritkernspeicher, im Handbuch der Elektronik, Teil 2, Digitaltechnik, beschrieben worden.

Speicher mit dünnen magnetischen Schichten

Da die Schaltgeschwindigkeit eines Ferritringkernes wesentlich von seiner räumlichen Größe abhängt, stößt man bei der Erhöhung der Geschwindigkeit durch Verkleinern des Kernes an herstellungstechnische Grenzen. Der Innendurchmesser wird irgendwann so klein, daß sich praktisch keine Drähte mehr durch das winzige Loch fädeln lassen. Was liegt näher, als das

Magnetmaterial direkt auf einen Draht aufzubringen? So bestechend einfach dieser Gedanke ist, bis zu seiner technischen Realisierung waren viele Probleme zu bewältigen. Inzwischen ist der Herstellungsprozeß so weit ausgereift, daß Magnetdrahtspeicher nach vorsichtiger Schätzung in den nächsten Jahren die Ferritkernspeicher voll ersetzen können. Das Speichermedium besteht aus einem 80—125 μm dicken **Kupferberylliumdraht,** um den herum eine 1 μm dicke **Permalloy-Schicht** aufgebracht ist. Das vollständige Speicherelement wird erst dadurch realisiert, daß um diesen Draht herum in kurzen Abständen (vgl. Abb. 2.33) Wortleitungen geführt werden.

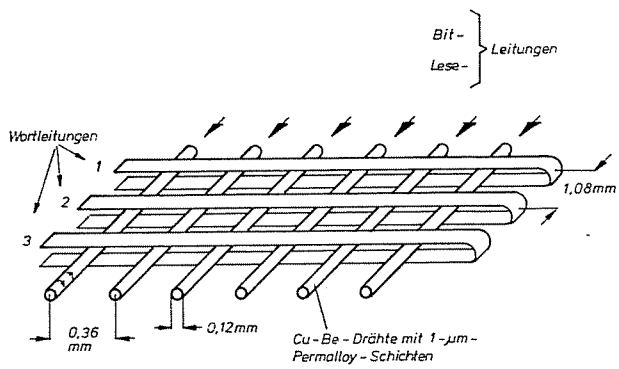


Abb. 2.33 — Magnetdrahtspeicher

Die Information ist durch **die Lage der Magnetisierungsrichtung,** entweder im Uhrzeigersinn oder entgegengesetzt, auf dem Draht gespeichert. In Längsrichtung des Drahtes ist die Magnetisierung **nur sehr schwer zu verändern,** so daß sich das benachbarte Speicherelement — unterhalb der nächsten Wortleitung — nicht beeinflussen läßt. Wie kann nun überhaupt die Magnetisierungsrichtung geändert, also eine Information eingeschrieben werden? Der Vorgang soll anhand der Abb. 2.34 veranschaulicht werden.

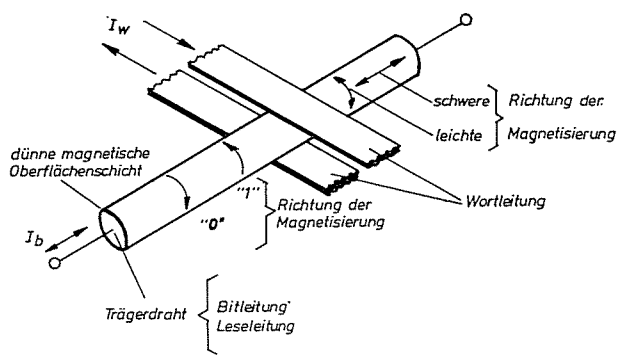


Abb. 2.34 — Einzelne Speicherelemente eines Magnetschichtdrahtspeichers

Ein Strom I_w durch den Wortleiter dreht das Magnetfeld der Schicht nahezu senkrecht aus der eingestellten Lage heraus. Wenn jetzt zusätzlich durch den Draht ein Bitstrom fließt, d.h. **Koinzidenz zwischen Wort- und Bitstrom** besteht, wird das Feld der Schicht entweder links- oder rechtsherum gedreht, je nach Polarität des Bitstromes. Beim Auslesen einer Information genügt ein Wortstrom in der Wortleitung. Das sich aus der eingestellten Lage drehende Feld induziert in dem Draht eine Spannung, deren Polarität von der Richtung des Feldes, mithin der eingeschriebenen Information abhängt. Nach Abschalten des Wortstromes kehrt das magnetische Feld in die Ausgangslage zurück, die **Information wurde durch das Lesen nicht zerstört.** Der Magnetschichtdrahtspeicher ermöglicht Zugriffszeiten für Lesen um 100 ns; Zykluszeiten für kombiniertes Lesen und Schreiben liegen bei 250 ns. Das ist zwar immer noch keine Revolution der Speichertechnik, aber doch ein beträchtlicher Gewinn gegenüber Ferritkernspeichern.

Um noch kürzere Ummagnetisierungszeiten als die von Magnetschichtdrähten zu realisieren, müssen die speicherfähigen Magnete noch weiter verkleinert werden. Dies ist aus herstellungstechnischen Gründen nur bei **ebenen Flächen als Unterlage** möglich. Hier lassen sich so glatte Oberflächen erzeugen, daß die Magnetflecken auf die Abmessungen von etwa 0,1 x 0,4 mm und eine Dicke von nur 0,04 bis 0,1 μm reduziert werden können. Die Magnetflecken bestehen aus Permalloy; sie werden im Hochvakuum auf die silberverspiegelten Platten aufgedampft. Den Aufbau dieser **Dünnschicht- oder Dünnfilmspeicher** genannten Anordnung zeigt Abb. 2.35. Über den Permalloyflecken liegen Lese-, Wort- und Bitleitungen. Die Richtung des Magnetfeldes gibt den Wert der abgespeicherten Information an. Die Ansteuerung einer einzelnen Zelle erfolgt in Koinzidenz der Signale auf Wort- und Bitleitungen.

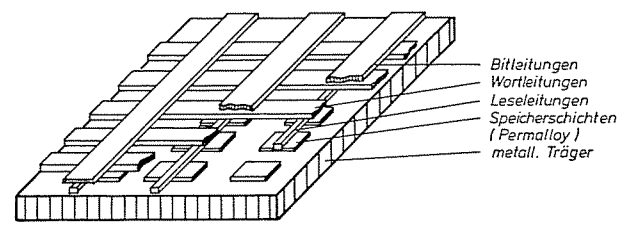


Abb. 2.35 — Dünnfilmspeicher

Dünnfilmspeicher sind bisher von verschiedenen Herstellern in kleineren Rahmen verwendet worden. Mit ihren Zykluszeiten bis zu 50 ns bieten sie viel Anreiz zum Einsatz als z.B. Notiz-

blockspeicher. In diesem Fall ist der Teil des Arbeitsspeichers, der sehr häufig benutzt wird, aus dem Gesamtkomplex herausgelöst, um mit schnelleren Speichermedien eine höhere Gesamtarbeitsgeschwindigkeit zu erreichen.

Zu den beiden Speicherverfahren mit dünnen Schichten ist festzustellen, daß sie rein vom Speichermedium her dem Ferritkern überlegen sind und daß hierdurch alle prinzipiellen, technologischen und herstellungstechnischen Probleme gelöst sind. Ihr Großeinsatz in den kommenden Jahren kann aber an einem nicht unwesentlichen Faktor scheitern: Große Teile der Ansteuerelektronik für Ferritkernspeicher sind als preiswerte IC auf dem Markt, bei den letztbesprochenen Speichermedien lassen sich diese jedoch nicht oder nur ausnahmsweise verwenden. Ihre Ansteuerelektronik muß deshalb aus diskreten Elementen teuer aufgebaut werden. So wird der Ferritkern über dem Umweg der Ansteuerelektronik auch in naher Zukunft noch konkurrenzfähig bleiben; sein Ende leiten Halbleiterspeicher ein.

Halbleiterspeicher

Nach vielen Anläufen in der Vergangenheit hat erst jetzt **der Halbleiterspeicher den Durchbruch zum voll einsatzfähigen Arbeitsspeicher** geschafft. Die Schwierigkeiten sind in drei Nachteilen zu suchen: Bisher lag der Preis einer Speicherzelle immer noch merklich höher als der anderer Speichermedien, zum anderen bereitet die Wärmeabfuhr der eng zusammengedrängten Bauelemente Probleme und zum dritten geht die Information des Speichers bei Ausfall der Stromversorgung verloren. Da aber der Halbleiterpreis ständig sinkt, ist der Zeitpunkt vorhersehbar, zu dem elektronische Speicherelemente billiger als magnetische sein werden. Zur Zeit sind wegen der Einsparung in der Ansteuerelektronik Speicher mit einigen 10 000 Bits schon preiswerter als magnetische. Das Problem der Wärmeabfuhr wird sich auch weitgehend erledigen, wenn anstelle von bipolaren endgültig unipolare Halbleiter eingesetzt werden, und zwar in Form von COSMOS-Schaltungen. (COSMOS oder CMOS sind komplementär symmetrische (p- und n-Kanal) MOS-Schaltungen.) Bipolare Flipflops benötigen etwa 10 mW Leistung, ein COSMOS-Flipflop nur 1 nW. Die Leistungsaufnahme eines 1-Millionen-Bit-Speichers mit bipolaren Elementen würde 10 kW betragen, die eines COSMOS-Speichers nur 1 mW!

Wie kann nun ein digitales Signal elektrisch gespeichert werden? Im theoretisch einfachsten Fall stellt jeder Kondensator ein Speicherelement dar. Leider aber verliert ein Kondensator in der Praxis seine Ladung wegen des endlich hohen Isolationswiderstandes. Normalerweise kommen Kondensatoren höchstens für wenige Mikro- oder Millisekunden als Speicher in Frage. Eine Ausnahme bilden **Kondensatoren in unipolaren IC,** d.h. genau gesagt, bei MOS-Feldeffekttransistoren. Weil innerhalb eines IC Isolationswiderstände von $10^{15}\ \Omega$ erzeugt werden können, läßt sich hier die Ladung eines Kondensators lange aufrechterhalten (in Versuchsentwicklungen bis zu 14 Tage). Diese extrem lange Speicherfähigkeit wird sicher erst in ferner Zukunft genutzt werden können. Schaltungen, die heute auf dem Markt sind, entladen sich schon nach wenigen Millisekunden. **Nach dieser Zeit muß der speichernde Kondensator wieder aufgeladen werden, sonst ginge die Information verloren.** Die Funktion eines solchen Speichers mit wahlfreiem Zugriff **(RAM, Random Access Memory)** soll am Beispiel des Speicherelementes 1103 der Firma Intel erläutert werden. Die Grundlage dieser Technik sind Feldeffekttransistoren des P-Kanal-Enhancement-Typs mit einer Gate-Elektrode aus polykristallinem Silizium.

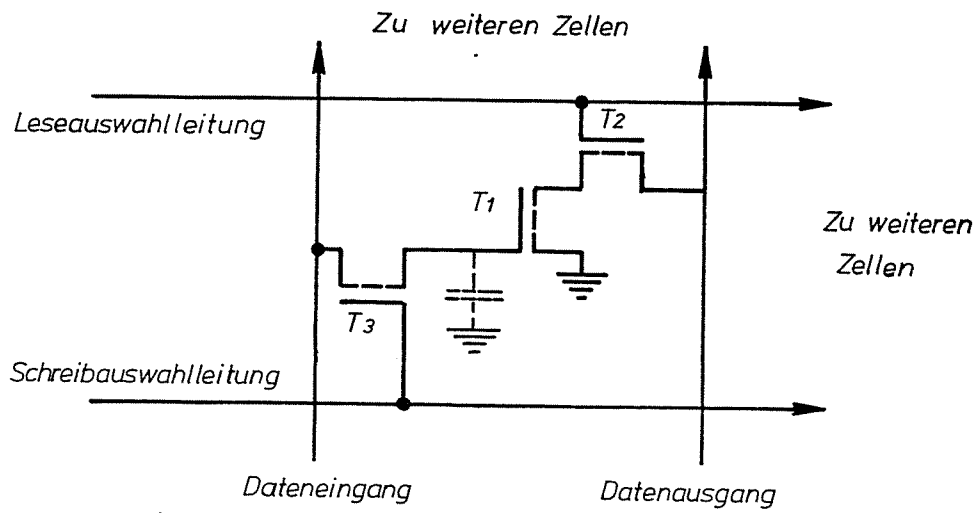


Abb. 2.36 — Speicherzelle des 1103

Diese „Silicon-Gate-FET" zeichnen sich gegenüber Aluminium-Gate-FET durch eine niedrige Schwellspannung und eine wesentlich höhere Schaltgeschwindigkeit aus. Da sie in IC außerdem sehr dicht zusammengedrängt werden können, bringt Intel auf einem Chip von 2,9 x 3,5 mm 1024 Bits unter. Eine Speicherzelle besteht lediglich, wie Abb. 2.36 zeigt, aus 3 Transistoren. Den eigentlichen **Speicherkondensator bildet — wie in dieser Technik allgemein üblich — die Gatekapazität eines Transistors,** hier die von T_1 (in der Abbildung gestrichelt gezeichnet).

Ist die Gatekapazität so weit aufgeladen, daß T_1 leitet, speichert die Zelle eine „1", ist T_1 dagegen gesperrt, entspricht dieser Zustand der "0".

Soll nun eine Information eingeschrieben werden, so wird für eine zu speichernde "1" negative Spannung an den Dateneingang gelegt (bei einer "0" dagegen Massepotential) und in Koinzidenz dazu die Schreibauswahlleitung negativ angesteuert. Jetzt ist T_3 leitend, die Gatekapazität von T_1 nimmt eine negative Ladung an. Damit ist die Information aufgenommen, die Schreibauswahlleitung muß noch auf Massepotential zurückgeschaltet werden, um T_3 zu sperren und die Information endgültig zu speichern.

Zum Auslesen wird zunächst die parasitäre Leitungs-Kapazität des Datenausganges negativ aufgeladen. Anschließend steuert man die Leseauswahlleitung negativ an, d.h. T_2 ist leitend. Wenn nun T_1 auch leitend ist, kann sich die parasitäre Kapazität des Datenausganges über den Weg T_2/T_1 gegen Masse entladen; ist dagegen T_1 gesperrt, bleibt die Ladung dieser Kapazität erhalten. Ob aber T_1 gesperrt oder leitend ist, bestimmt die abgespeicherte Information. Aus dem Verhalten des Datenausganges im Anschluß an das Ansteuern der Leseauswahlleitung kann also eindeutig auf den Speicherinhalt geschlossen werden. War eine "1" gespeichert, zeigt der Datenausgang Nullpotential, bei der gespeicherten "0" zeigt er negatives Potential.

Intel garantiert eine Speicherzeit der Zelle von 2 ms. Danach kann sich die Gatekapazität von T_1 soweit entladen haben, daß keine zutreffende Aussage über die zuvor abgespeicherte Information gewonnen werden kann. Daher muß alle 2 ms ein Erneuerungszyklus erfolgen, das ist ein etwas vereinfachter Lese-Schreibzyklus.

Abb. 2.37 zeigt den Aufbau eines Chips. Die Speicherzellen sind in einer Matrix von 32 x 32 angeordnet. Die an gemeinsamen Auswahllei-

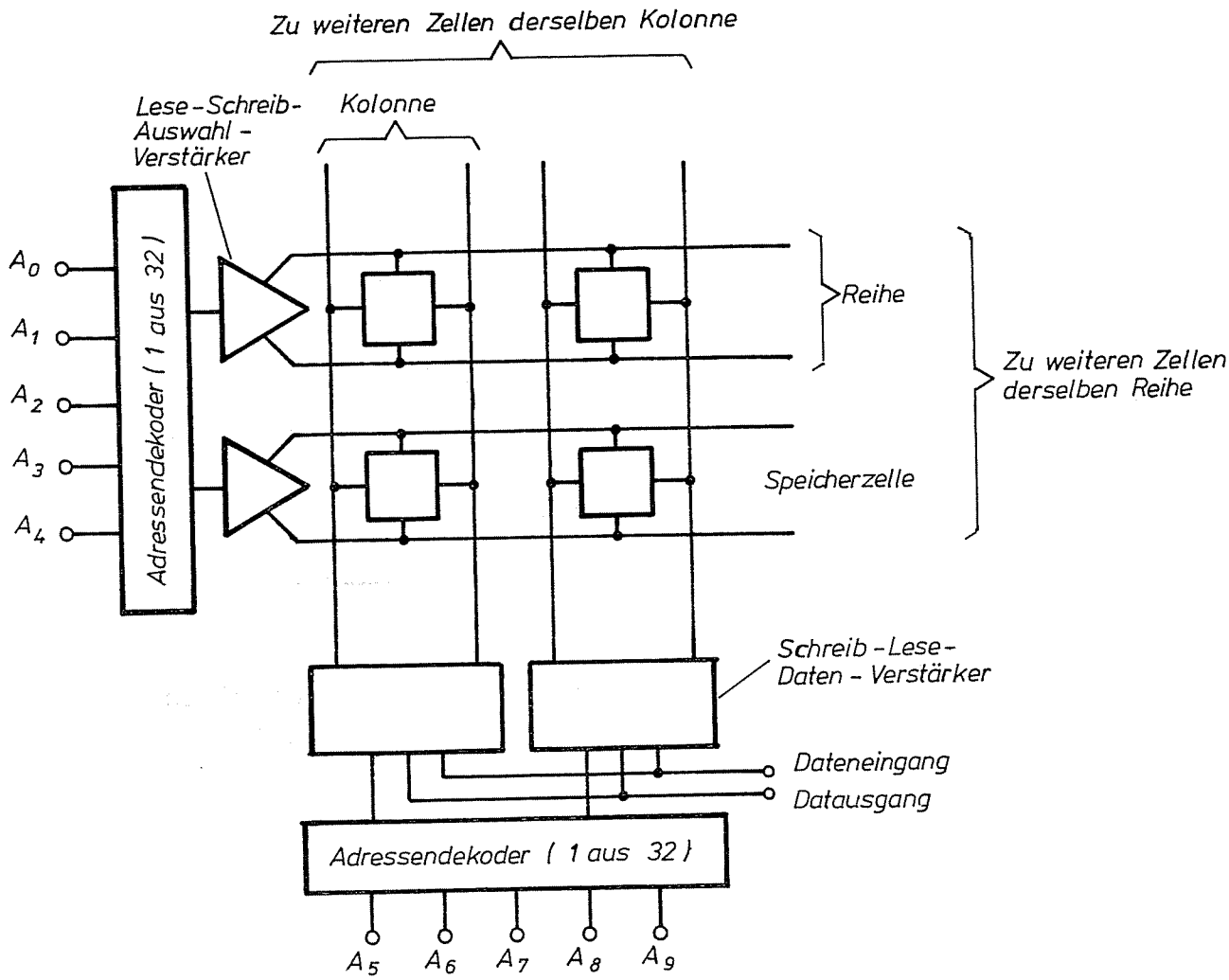


Abb. 2.37 — Organisation des 1103

tungen liegenden Zellen sind als Reihe, die an gemeinsamen Datenleitungen als Kolonne bezeichnet. Eine einzelne Speicherzelle läßt sich also durch das Ansteuern der betreffenden Reihe und Kolonne adressieren. Auf dem Chip sind neben den Speicherzellen sämtliche Verstärker für die Lese-/Schreibauswahl, die Datenein- und -ausgänge und die Adressendecoder untergebracht. Die Zugriffszeit für Lesen beträgt immerhin 300 ns, die Zykluszeit für einen Lese-/Schreibvorgang 580 ns. Die mittlere Verlustleistung pro Chip liegt bei 250 mW. Die Geschwindigkeit dieses Speichers fällt demnach in den Bereich von schnellen Ferritkernspeichern.

Die **hohen Zykluszeiten** von Halbleiterspeichern auf der Basis unipolarer IC sind auf die verhältnismäßig **großen parasitären Kapazitäten der Feldeffekttransistoren** zurückführen. Der Effekt, den man einerseits zum Abspeichern der Information ausnutzt, erzeugt andererseits das Problem der langen Schaltzeiten. Die Zykluszeiten lassen sich allerdings auch bei MOS-Technologie verkürzen, wenn anstelle dynamischer Schaltprinzipien statische Speicherschaltungen verwendet werden. Zu diesem Zweck bietet sich das **Flipflop** an; **es ist die universell eingesetzte Schaltung, sowohl bei unipolaren als auch bei bipolaren Speichern.**

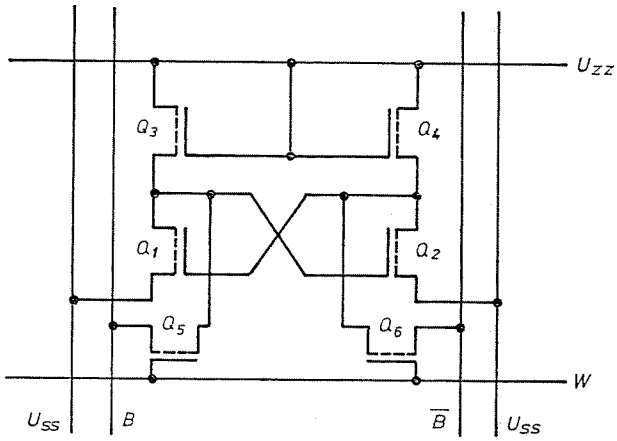


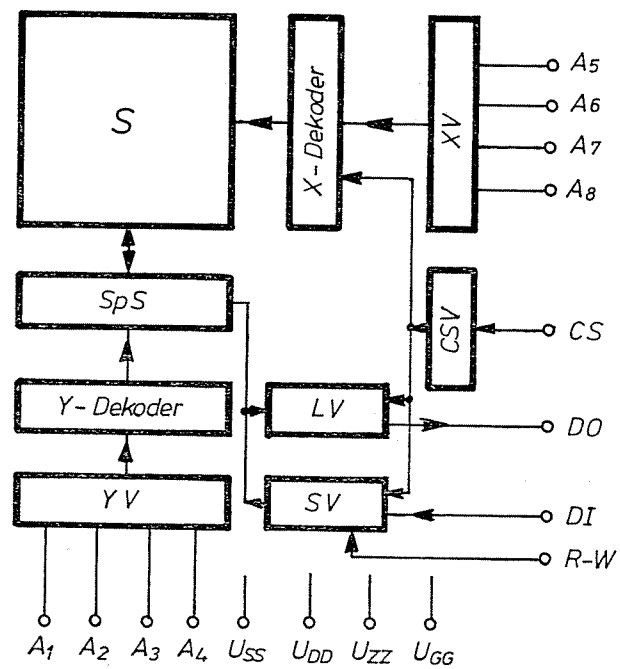
Abb. 2.38 — Statische Speicherzelle

Abb. 2.38 zeigt eine Speicherzelle des Siemens MOS-Speicher GDQ 106. Die Versorgungsspannung liegt an der Leitung U_{zz} und das „0"-Potential an U_{ss}. Das eigentliche Flipflop ist mit den beiden Schalttransistoren Q_1 und Q_2 und den als Arbeitswiderstände geschalteten Transistoren Q_3 und Q_4 aufgebaut. Die Funktion dieses aus zwei kreuzgekoppelten Invertern bestehenden Flipflops ist klar ersichtlich.

Ist z.B. Q_1 durchgeschaltet, so liegt "0"-Potential (zusätzlich der Restspannung) am Gate von Q_2, damit ist dieser gesperrt. Die beiden Transistoren Q_5 und Q_6 fungieren als Steuer- und Koppeltransistoren. Wenn die Wortleitung W von "0"- auf "1"-Potential angehoben wird, können Q_5 und Q_6 auf die Bitleitungen B und $\overline{B}$ durchschalten. Zum Schreiben müssen B und $\overline{B}$ komplementäre Potentiale führen, d.h., wenn $\overline{B}$ "1"-Potential aufweist, muß B auf "0" liegen (und umgekehrt). Die "0" von B kann über Q_5 auf das Gate von Q_2 durchgreifen; war Q_2 zuvor leitend, d.h. lag hohes Potential an, wird er nun gesperrt, das Flipflop kippt also. War Q_2 jedoch vorher schon gesperrt, ändert sich sein Zustand auch jetzt nicht.

Das Schreiben besteht demnach aus zwei Vorgängen: Ansteuern der Transistoren Q_5 und Q_6 über die Wortleitung und Setzen des Flipflops über die Bitleitungen. Es zeigt sich jetzt eindeutig, daß die Bitleitungen komplementäre Signale führen müssen. Zum Lesen des eingeschriebenen Zustandes werden im Leseverstärker beide Bitleitungen hochohmig an die Versorgungsspannung geschaltet, also nicht mehr komplementär betrieben. Wenn nun mit "1"-Signal an der Wortleitung Q_5 und Q_6 leitend gesteuert werden, zieht derjenige Transistor seine Bitleitung auf 0-Potential, dessen Flipflophälfte durchgeschaltet hat. Im zuvor betrachteten Fall, daß Q_1 leitend geschrieben wurde, wird dessen niedriges Potential über Q_5 auf die Bitleitung B durchgreifen können. Damit ermöglichen die Bitleitungen eine eindeutige Aussage, in welcher Lage das Flipflop steht.

Der Aufbau einer derartigen Speicherzelle ist — für ein Flipflop — äußerst einfach; sämtliche Schaltelemente bestehen aus Transistoren. Deshalb können auf einem Chip auch verhältnismäßig viel Speicherzellen untergebracht werden. Der GDQ 106 hat 256 Bits. Chips mit 1024 Bits (z.B. Texas Instruments TMS 1103 NC) sind problemlos herstellbar. **Aber der Grundaufbau aller statischen MOS-Speicher ist die in Abb. 2.38**



S	256 Speicherzellen	YV	Y-Verstärker
SpS	Spaltenverstärker	R-W	Lese-Schreib-Signal
CSV	CS-Verstärker	CS	Sperrsignal (Chip Select)
LV	Leseverstärker	DI	Dateneingang
XV	Schreibverstärker	DO	Datenausgang
SV	X-Verstärker	A1-A8	Adressen

Abb. 2.39 — Schaltung des GDQ 109

gezeigte Zelle. Adreßdecodierer und Schreibleseverstärker hängen hingegen sehr von der internen Organisation ab. Abb. 2.39 zeigt die komplette Schaltung des GDQ 109, der in 256 Worten zu je 1 Bit organisiert.

Die Adressen werden codiert als X- und Y-Adresse über die Eingänge A 1 — A 8 eingegeben, laufen über X-Verstärker (XV) und Y-Verstärker (YV) auf die X- und Y-Decoder. Am X-Decoder liegen 16 Wortleitungen, die auf die 256 Speicherzellen aufgeteilt sind. Der Y-Decoder wählt einen von 16 Spaltenschaltern (SpS) aus. An den Spaltenschaltern liegen die Bitleitungen. Die Lage einer Speicherzelle ist also durch eine von 16 Wortleitungen und einen von 16 Spaltenschaltern mit seinen zwei Bitleitungen definiert. Soll eine Information eingeschrieben werden, so wird der Eingang R-W auf "0" geschaltet und die Information über den Schreibverstärker SV und die Spaltenschalter SpS auf die Bitleitungen gelegt. Zum Lesen schaltet man R-W auf "1"; damit ist der Schreibverstärker gesperrt und die Information der Bitleitung liegt über Spaltenschalter und Leseverstärker LV am Datenausgang DO. Über den Eingang CS und den entsprechenden Verstärker läßt sich das gesamte Chip sperren. Diese Möglichkeit ist wichtig, wenn die Anzahl der Worte erweitert werden soll. An den Eingängen U_{SS} liegt Masse (0 V), an U_{DD} — 13 V, an U_{ZZ} — 16 V und an U_{GG} — 27 V. Die Spannung U_{ZZ} versorgt ausschließlich Speicherzellen; auf ihre Stabilität und vor allem ihr Vorhandensein muß besonders geachtet werden, da mit ihrem Ausfall der Speicherinhalt verlorengeht.

Die Zugriffszeit des GDQ 106 liegt bei 700 ns, sein Leistungsverbrauch beträgt 360 mW. Speicher dieser Art mit Zugriffszeiten um 300 bis 400 ns sind bereits auf dem Markt.

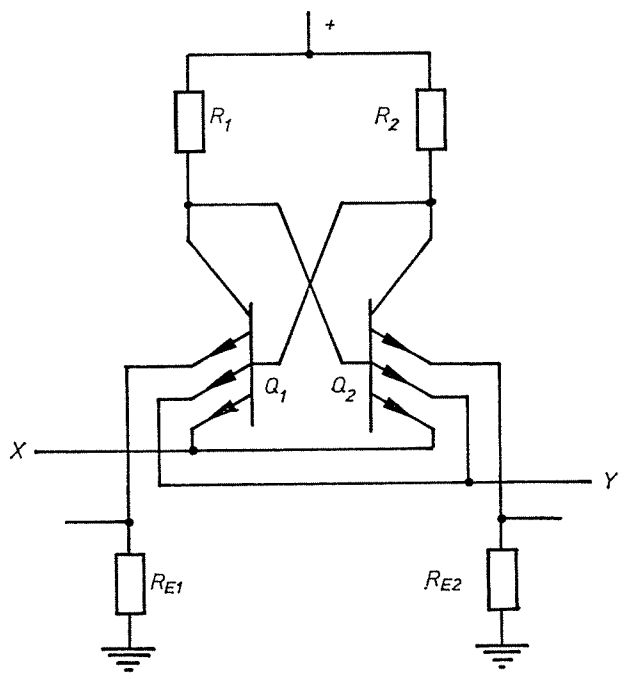


Abb. 2.40 — Bipolare Speicherzelle

Dem statischen MOS-Speicher an Geschwindigkeit weit überlegen sind bipolare Schaltkreise. Diese, meist als MSI oder LSI direkt aus der TTL-Technik hervorgegangen, sind nur als statische Speicher möglich. **Ihre typischen Zugriffszeiten liegen bei 30—50 ns, der Leistungsverbrauch für ein 16-Bit-Chip um 300 mW** (also fast soviel wie das vorher beschriebene 256-Bit-Chip GDQ 106). Als Beispiel soll hier die in Abb. 2.40 gezeigte Zelle IM 5502 der Firma Intersil beschrieben werden. (Die Schaltungen aller Hersteller von bipolaren Speichern sind sehr ähnlich, zum größten Teil sogar völlig gleich.)

Die eigentliche Speicherzelle besteht auch hier wieder aus einem bis auf ein Minimum vereinfachten Flipflop. Die bei einem üblichen Flipflop sehr aufwendige Ansteuerschaltung konnte durch die Verwendung von Multi-Emitter-Transistoren umgangen werden.

Wichtig für die Funktion sind die beiden Widerstände R_{E1} und R_{E2}. Über sie werden die Transistoren Q_1 und Q_2 — bei unbeschalteten Emittern — als Emitterfolger betrieben. Wenn z.B. Q_1 durchgeschaltet hat und damit Q_2 gesperrt ist, wird über R_{E1} ein Strom fließen, der einen entsprechenden Spannungsabfall verursacht, während an R_{E2} keine Spannung abfallen kann. Legt man jetzt einen der Emitter von Q_2 direkt auf "0"-Potential, so wird Q_2 durchschalten, weil seine Basis in jedem Fall positives Potential erhält. Damit liegt aber auch am Kollektor von Q_2 "0"-Potential, d.h. Q_1 wird gesperrt. Über diesen Weg kann also die Lage des Flipflops verändert werden. Von den drei Emittern der Transistoren dienen zwei der Adressierung (je einer für X- und Y-Adresse) und der dritte dem Schreiben und Lesen. Solange die Zelle nicht angesprochen ist, liegen die X- und Y-Leitungen auf "0"-Potential, werden also die Transistoren Q_1 und Q_2 in Emitterschaltung betrieben. Bei Adressierung der Zelle schalten die zuständigen X- und Y-Leitungen in Koinzidenz auf "1"-Potential. Jetzt findet nur der dritte Emitter über R_{E1} (in der betrachteten Lage des Flipflops) Erdpotential.

Wenn man nun das Potential an den Leitungen RW_0 bzw. RW_1 mißt, kann man eindeutig die Lage des Flipflops bestimmen. Soll andererseits das Flipflop gekippt werden, so muß die Leitung RW 1 auf „0" gelegt werden. Über die RW-Leitung kann also sowohl gelesen als auch geschrieben werden.

Abb. 2.41 zeigt die Speicherzelle mit den für alle Flipflops gemeinsam verwendeten Schreib- und Leseverstärkern. Diese Verstärker sind doppelt vorhanden, sowohl für Q_1 als auch für Q_2. Es ist festgelegt, daß bei durchgeschaltetem Q_1 die Zelle eine „0" gespeichert hat, bei durchgeschaltetem Q_2 eine „1". Hiernach sind die Verstärker mit „0" bzw. „1" bezeichnet.

Zum Schreiben einer "1" wird nach Adressieren der Zelle der Eingang W_1 auf "1", d.h. auf + 5 V gelegt. Damit schaltet der Schreibverstärker durch, Q_6 legt die Leitung RW_1 auf Erdpotential, das Flipflop kippt in die Stellung "1". Hatte es zuvor schon diese Stellung inne, ändert es sich nicht. Voraussetzung für den Schreibvorgang ist natürlich, daß immer nur ein Schreibverstärker angesteuert wird. Beim Lesevorgang bleiben beide Schreibverstärkereingänge auf „0". Wenn die Zelle eine „1" ge-

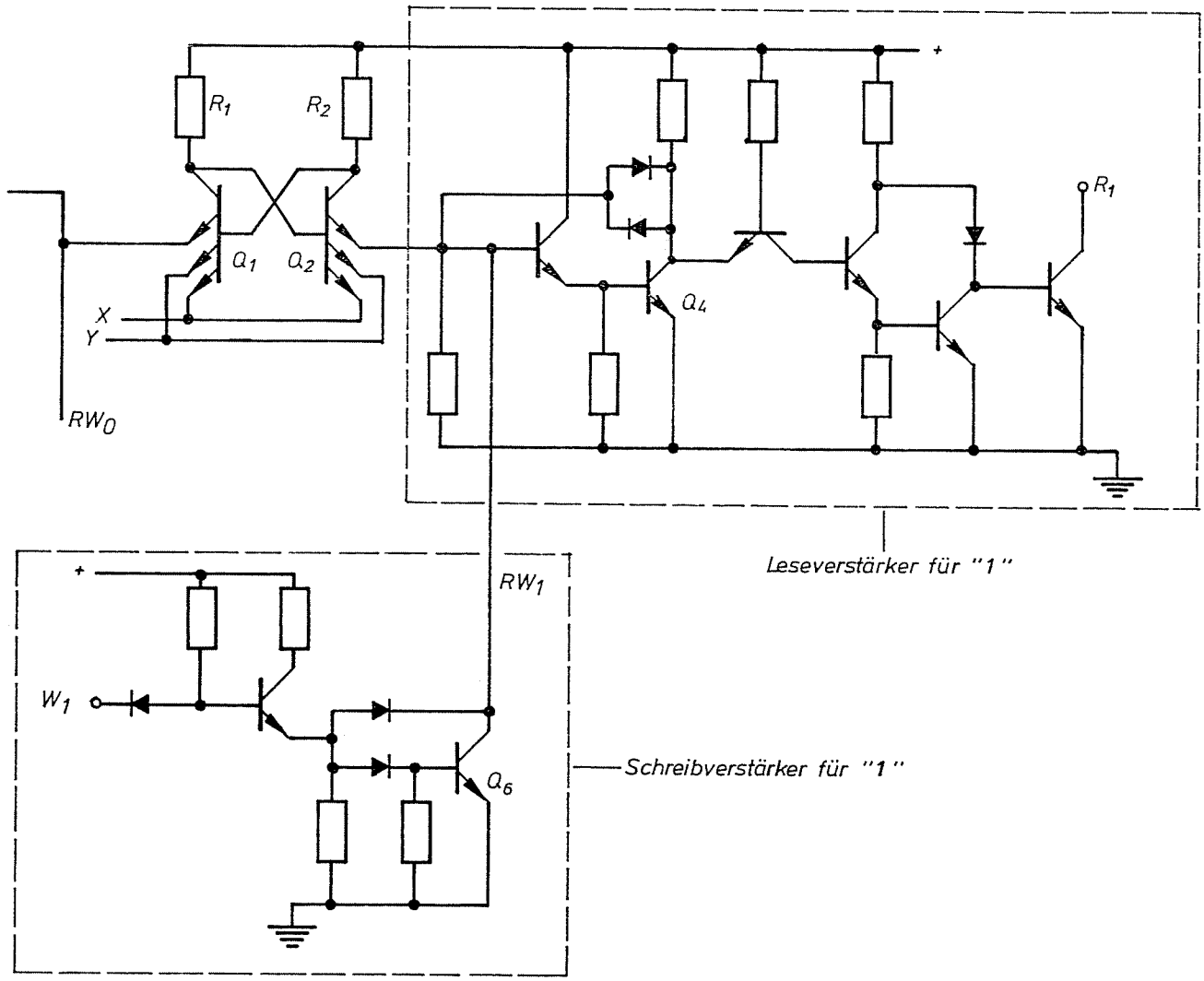
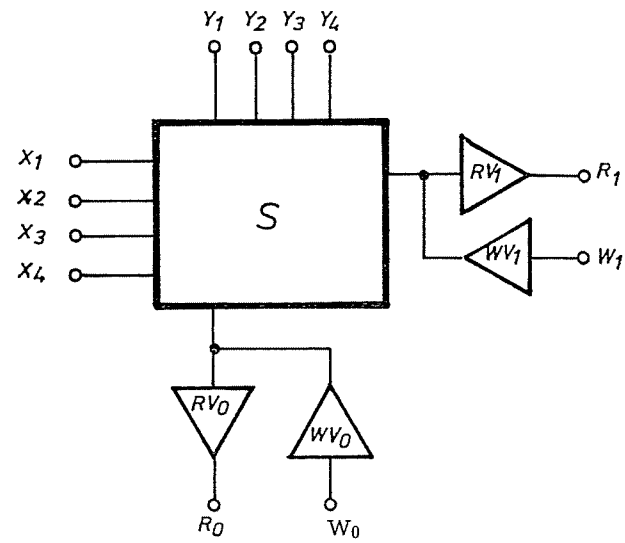


Abb. 2.41 — Bipolare Speicherzelle mit Schreib- und Leseverstärker

speichert hat, erhält die Basis von Q_4 positives Potential über R_2 und Q_2; der Leseverstärker kann durchsteuern. sein Ausgangstransistor schaltet von "1" auf "0" um. Das Signal erscheint also invertiert am Leseverstärker.



S 16 Speicherstellen
RV_1 Leseverstärker „1"
WV_1 Schreibverstärker „1"
RV_0 Leseverstärker „0"
WV_0 Schreibverstärker „0"
X_1-X_4 X-Adressen
Y_1-Y_4 Y-Adressen

Abb. 2.42 — Organisation des IM 5502

In Abb. 2.42 ist die komplette Schaltung des IM 5502 dargestellt. Der Speicher ist in 16 Worte zu je ein Bit organisiert. Die Adresseneingänge sind x_1 — x_4 und y_1 — y_4. Die Eingänge W_1 und W_0 dienen zum Schreiben, die Lesesignale erscheinen an R_1 bzw. R_0. Die Zugriffszeit liegt bei 20 ns, der Leistungsverbrauch bei 300 mW.

Zusammenfassend kann über Halbleiterspeicher noch einmal gesagt werden, daß ihnen die Zukunft gehört. Verschiedene Hersteller bieten schon heute große Datenverarbeitungsanlagen mit ausschließlich Halbleiterspeichern an, z.B. IBM mit der 370/145, die bis zu 512 KBytes große Arbeitsspeicher besitzt. Häufig findet man auch einen Kompromiß zwischen Ferritkern- und Halbleiterspeicher. Hier wird ein Pufferspeicher auf Halbleiterbasis verwendet, in dem jeweils ganze Datenblöcke aus dem Ferritkernspeicher zwischengespeichert werden, um die mittlere Zugriffszeit zu verkürzen. Dieses Verfahren dürfte sich vermutlich nur in der Übergangsphase zum reinen Halbleiterspeicher lohnen. Bei Drucklegung dieses Buches läßt sich

noch nicht sagen, welches Prinzip des Halbleiterspeichers endgültig zum Durchbruch kommen wird. Dynamische MOS-Speicher mit über 16 000 Bits pro Chip erscheinen auf dem Markt. Der Nachteil aller MOS-Prinzipien ist ihre verhältnismäßig hohe Zugriffszeit. Der in diesem Punkt weit überlegene bipolare Speicher bringt dagegen aber nur einen Teil der Informationen auf einem Chip unter und verbraucht wesentlich mehr Leistung pro Bit.

2.3.2. Festwertspeicher

2.3.2.1. Einsatz von Festwertspeichern

Einen Speicher, dessen Inhalt unveränderlich festgelegt ist und der nur gelesen werden kann, nennt man Festwertspeicher (englisch Read-Only-Memory, ROM). Derartige Speicher finden sich im täglichen Leben sehr häufig, z.B. ist der Programmablauf von Fahrstühlen oder der eines elektromechanischen Wählers fest abgespeichert, in diesen Fällen nicht in eigens gebauten Speicherzellen, sondern ganz schlicht in den Steuerrelais mit ihrer Verdrahtung. Bei derartig einfachen Ablaufsteuerungen würde es sich nicht lohnen, übliche Speichermedien mit ihrer Leseelektronik zu verwenden. Aber z.B. bei der komplizierteren Waschmaschinensteuerung stellt eine echte Festwertspeicherung schon eine wirtschaftliche Lösung dar. Wahrscheinlich werden bald elektronische Festwertspeicher gegenüber dem elektromechanischen Programmschalter konkurrenzfähig sein.

In sehr vielen elektronischen Datenverarbeitungsanlagen werden ebenfalls Festwertspeicher verwendet. Während in den ersten EDV-Generationen der Ablauf im Steuerwerk durch die Verdrahtung festgelegt war, also z.B. ein Additionsbefehl ganz andere Logikbausteine durchlief als ein Schiebebefehl, stößt bei den neueren Computergenerationen **jeder Befehl einen oder mehrere Zyklen in einem ROM an.** In diesem Festwertspeicher ist für jeden Befehl das Programm abgespeichert, nach dem das Steuerwerk zu arbeiten hat. (Dieses Programm ist das sogenannte **Mikroprogramm.)** Damit läßt sich gegenüber der alten Lösung einerseits sehr viel Aufwand im Steuerwerk einsparen, andererseits — und das ist der Hauptvorteil — kann der Befehlsvorrat der Maschine durch Erweitern oder Ändern des Festwertspeichers bedeutend leichter verändert werden. Damit ist aber auch gesagt, daß **ein Festwertspeicher nicht in jedem Fall so fest sein sollte, daß sein Informationsinhalt überhaupt nicht zu ändern ist.** Außerdem ist noch zu verlangen, daß das Einspeichern einer Information nicht allzu aufwendig sein sollte und daß — speziell bei der Anwendung im Computer — sehr kurze Zykluszeiten notwendig sind. Neben dem Einsatz als Mikroprogrammspeicher finden ROM zunehmend Verwendung als Tabellentafeln (z.B. mit logarithmischen oder trigonometrischen Werten) und für Codier- oder Decodierzwecke (z.B. als Zeichengeneratoren).

2.3.2.2. Prinzipien von Festwertspeichern

Festwertspeicher mit Dioden oder Widerständen

Baut man eine ganz einfache Matrix aus x- und y-Drähten auf und schaltet in deren Kreuzungspunkte z.B. Widerstände in Abhängigkeit von gewünschten Informationen, dann ist damit ein sehr einfacher Festwertspeicher verwirklicht. Wenn in Abb. 2.43 das unter der Adresse x_1 abgespeicherte Wort durch Anlegen von Spannung an die Leitung x_1 aufgerufen wird, so erscheint die Spannung nur an den Leseleitungen, die durch Widerstände mit der x_1-Leitung verbunden sind. Es würde also für x_1 die Information 1001 ausgelesen, und zwar so oft dieses Wort aufgerufen wird. Ein Verändern der Information wäre nur durch Aus- bzw. Umlöten

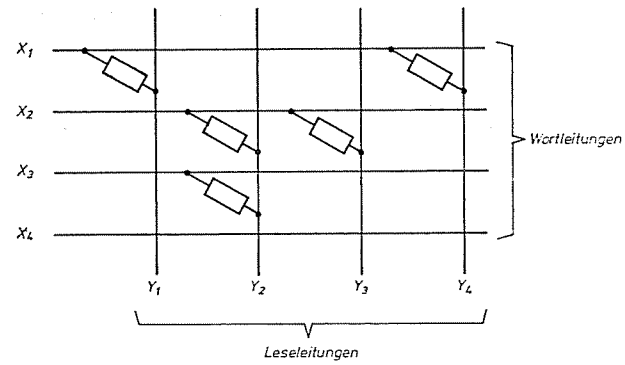


Abb. 2.43 — Festwertspeicher mit Widerständen

der Widerstände möglich. An Widerständen fällt Spannung ab. Ist dieser Spannungsabfall störend, so können die Widerstände durch Dioden ersetzt werden (Abb. 2.44). Das Lesen der Information geschieht wie vorher, lediglich die Polarität der Spannung ist zu beachten. Ein zusätzlicher Vorteil dieser Schaltung liegt in der Entkopplung der einzelnen Speicherzellen.

Nachteil dieser oder ähnlicher Festwertspeicher ist ihr doch sehr arbeitsintensiver Aufbau, da

für jedes einzelne Bit Handgriffe notwendig sind. Außerdem ist der Platzbedarf erheblich, selbst bei dem üblichen Aufbau auf gedruckten Schaltungen.

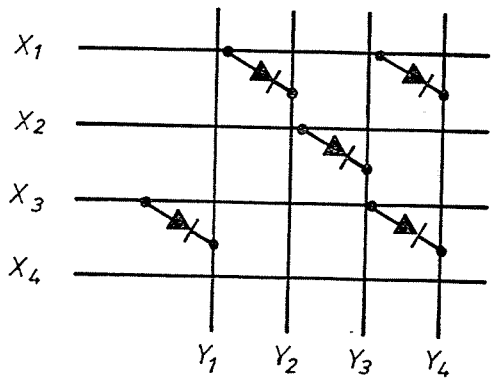


Abb. 2.44 — Festwertspeicher mit Dioden

Festwertspeicher mit Ferritkernen

Eine andere Möglichkeit der Festwertspeicherung sind verhältnismäßig große Ferritringkerne. Durch sie werden Drähte — je nach Informationsgehalt — hindurchgefädelt oder aber vorbeigeführt. Abb. 2.45 zeigt das Prinzip. Durch die vier Kerne führen je eine Leseleitung und eine gemeinsame Vorstromleitung. Die anderen Drähte sind Erregerleitungen. Für jedes Wort ist eine Erregerleitung nötig; soll eine „1" gespeichert werden, läuft sie durch den Kern, bei einer „0" ist sie an ihm vorbei verlegt. Die Information ist also in der Führung dieser Leitungen abgespeichert. Die Vorstromleitung führt

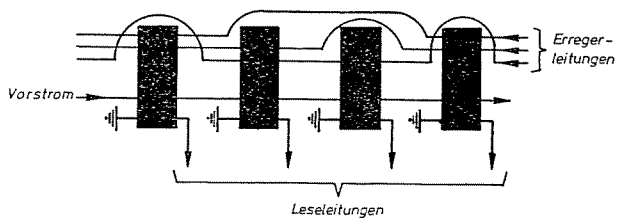


Abb. 2.45 — Festwertspeicher mit Ferritkernen

dauernd Strom und hält die Kerne im „0"-Zustand. Wird aber in eine der Erregerleitungen ein hinreichend großer Strom eingeprägt, so können nur die Kerne kippen, durch die sie hindurchführt. Der Ummagnetisierungsvorgang induziert in die Leseleitung einen entsprechenden Spannungsimpuls.

Dieses Prinzip des Festwertspeichers hat den sehr großen Nachteil, daß soviel Erregerdrähte durch die Kerne gefädelt werden müssen, wie der Speicher Worte fassen soll. Da die Kerne nicht beliebig vergrößert werden können, bleibt nur der Ausweg, Gruppen mit jeweils weitgehend eigener Leseelektronik zu bilden. Außerdem verlangt das Fädeln Sorgfalt und Aufmerksamkeit. Denn ein Ändern ist zwar möglich, wird aber problematisch, wenn nicht nur neue Drähte nachgezogen (und die alten einfach außer Betrieb gesetzt), sondern die alten auch entfernt werden müssen.

Festwertspeicher mit induktiver Kopplung

Alle bisher besprochenen ROM sind schwierig zu verändern; ihr Aufbau insgesamt erfordert Aufwand an Volumen; das Festlegen der Informationen ist arbeitsintensiv. Die Firma Siemens AG hat einen Festwertspeicher entwickelt, dessen Bau und Programmierung weitgehend automatisierbar sind und der verhältnismäßig leicht und schnell zu ändern ist. Er wird im künftigen elektronischen Wählsystem EWS der Deutschen Bundespost als Mikroprogrammspeicher verwendet, weiter kann er als Tabellenspeicher z.B. für Teilnehmerberechtigungen oder für Umwerter eingesetzt werden. Das Prinzip zeigt Abb. 2.46. Die Wort- und Leseleitungen sind als geätzte Leiterbahnen auf doppelseitig kaschierter Kunststoffolie aufgebracht. An den Kreuzungspunkten sind sie — der besseren Kopplung wegen — ein kurzes Stück parallel geführt. Auf diese Stellen können kleine Ferritplättchen gelegt werden. Zum Auslesen eines Wortes wird die zugehörige Wortleitung über einen elektronischen Schalter an Spannung gelegt; es fließt ein Strom, der durch den Abschlußwiderstand Rw begrenzt ist. Betrachtet man die Wortleitung W_1 und die Leseschleife L_1, so werden über die Kapazitäten der Koppelpunkte A_{11} und B_{11} (die gleich groß sind) zwei gleiche, aber gegenphasige Spannungen eingekoppelt. Der Differenzverstärker am Ende der Leseleitung unterdrückt diese beiden Spannungen; an seinem Ausgang erscheint kein Signal. Wird jedoch auf die Stelle A_{11} ein Ferritplättchen gelegt, so entsteht beim Aufruf der Wortleiung W_1 in der Leseschleife L_1 eine zusäzliche Induktionsspannung. Zwischen den Enden der Leseschleife tritt eine Spannung auf, die der Differenzverstärker verstärkt weitergibt. In unserem Beispiel hat sie beim Einschalten des Stromes eine positive Flanke. Legt man das Ferritplättchen jedoch auf die Stelle B_{11}, so wird eine negative Flanke entstehen. Diese beiden Signale definieren die digitalen Informationen „1" und „0", sie werden also die Lage des Ferritplättchens bestimmen. Die Ferritplättchen und die beiden Leitungsabschnitte bilden einen Übertrager mit sehr geringer Induktivität und damit sehr kleiner Zeitkonstante. Die Zykluszeit des Speichers hängt deshalb nur in geringem Maße vom verwendeten Ferritmaterial ab (mit dem praktisch immer Flankenanstiegszeiten von 1 ns möglich sind), sondern fast

ausschließlich von der Schaltkreistechnik. Zykluszeiten unter 100 ns sind somit keine Schwierigkeit.

Die Ferritplättchen werden auf eine Kunststoffträgerplatte aufgeklebt, die wiederum über die Platte mit den Wort- und Leseleitungen geschoben wird. Das Aufkleben der richtigen Information an der richtigen Stelle ist insofern sehr vereinfacht, als alle Abmessungen der Leiterbahnen und der Ferritplättchen auf Lochkartenmaße bezogen sind.

Wenn also ganz normale Lochkarten den Informationen entsprechend gelocht sind, so können diese als Schablonen für das Aufkleben der Kerne — ein Ferritplättchen paßt genau in ein Lochkartenloch — verwendet werden. Da die Lochkarten in jedem üblichen Kartenstanzer per Programm abgelocht werden können, ist das Erstellen einer Lochkarten-Schablone keine Schwierigkeit. Pro abzuspeicherndes Bit sind zwei Lochpositionen nötig; insgesamt lassen sich pro Lochkarte also 480 Bits unterbringen.

Will man den Informationsgehalt des Speichers ändern, so können natürlich die Ferritplättchen in ihrer Position umgesetzt werden. Normalerweise aber wird man eine Trägerplatte komplett neu bestücken und diese anstelle der alten einschieben.

Monolithische Festwertspeicher

Auch bei den ROM geht der Trend zur integrierten Lösung. Wie sehr gerade dieses Spezialgebiet im Fluß ist, zeigen die fast täglichen Neuankündigungen aller Hersteller. Bisher hat sich kein einheitliches Schaltungsprinzip durchsetzen können. **Allgemein sind Schaltungen üblich, die direkt mit TTL- bzw. DTL-Schaltkreisen kompatibel sind. Grundsätzlich läßt sich zwischen nicht programmierbaren und programmierbaren Festwertspeichern — letztere PROM genannt — unterscheiden.** Unter PROM sind Festwertspeicher zu verstehen, die vom Anwender (in den meisten Fällen nur einmal) programmiert werden können. Mit den nicht programmierbaren ROM begann die Entwicklung. Es handelt sich auch hier um eine Leiterbahnmatrix, in deren Kreuzungspunkte Koppelelemente geschaltet sind. Das elektrische Vorhandensein des Koppelelementes — meistens Transistoren oder Dioden — entscheidet über den Informationsgehalt des Speichers. Für den Anwender wichtig ist die Technik der Herstellung. Aus wirtschaftlichen Gründen werden sämtliche Produktionsschritte, die für alle ROM eines Typs gleich sind, durchgeführt und die vorgefertigten Produkte auf Lager gelegt. Erst wenn der Absender die abzuspeichernden Informationsmuster dem Hersteller angibt, wird die letzte Maske angefertigt: diese Maske bestimmt, in welchen Matrixpunkten Wort- und Leselei-

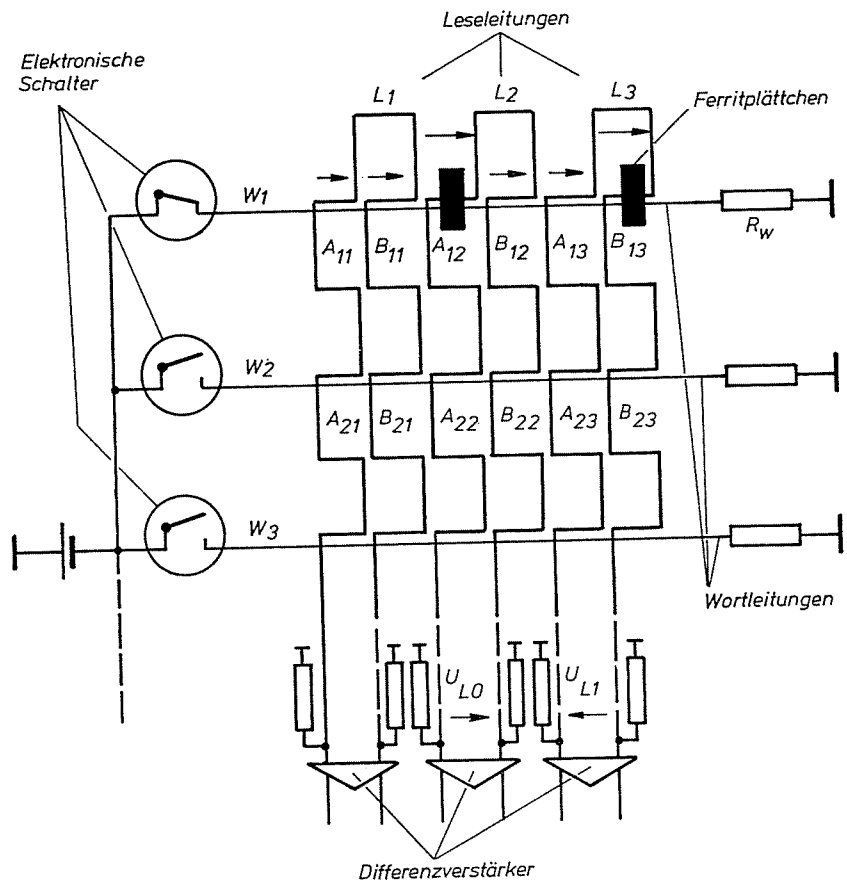


Abb. 2.46 — Festwertspeicher mit induktiver Kopplung

tungen elektrisch verbunden sind. Nach Aufbringen dieser lezten Verbindungsleiterbahnen wird der Chip gekapselt, die Informationen liegen jetzt unveränderbar endgültig fest. **Ein Ändern des Speicherinhaltes — z.B. beim Erweitern des Befehlsvorrates des Rechners — ist nur dadurch möglich, daß man neue Masken für neue Chips herstellt; die alten Chips sind dann wertlos geworden.**

Monolithische ROM sind sowohl auf bipolarer als auch auf unipolarer Basis möglich und üblich. Ebenso werden Lösungen mit Dick- oder Dünnfilmschaltkreisen angeboten. Weil meist sehr viele Informationen auf einem Chip untergebracht sind (4096 Bits sind heute kein Problem), müssen die Adressendecoder ebenfalls auf das Chip integriert werden, andernfalls wäre die Zahl der Anschlüsse viel zu hoch. Für die Organisation des Speichers sind Anzahl und Zusammenschaltung der Adressendecoder maßgebend. Häufige Kombinationen sind Worte zu 4 oder zu 8 Bits; sehr flexible Anordnungen lassen natürlich Speicher mit Worten zu 1 Bit zu. Einen kompletten Festwertspeicher zeigt Abb. 2.47. Es handelt sich um das 2048 Bit — ROM IM 7603 der Firma Intersil.

Der Speicher ist in 256 Worten zu je 8 Bits organisiert, die Leseleitungen liegen — über Trennverstärker — an den Ausgängen B_1 bis B_8. Die Wortadresse wird über die Leitungen A_1 bis A_8 codiert als X- und Y-Adresse eingegeben. Der Speicher ist auf MOS-Technologie aufgebaut; als Koppelpunkt dient ein MOS-FET, dessen Drain-Source-Strecke bei Programmierung einer „1" überbrückt wird. Die Zugriffszeit beträgt 350 ns. Der IM 7603 ist gedacht als Zeichnungsgenerator für z.B. Datensichtgeräte oder Lumineszensdioden-Anzeigegeräte, als Code-Umsetzer, für Mikroprogrammierung usw. Für verschiedene Standard-Anwendungen liegt er fertig programmiert auf Lager; bei kundenspezifischer Programmierung wird die Maske für den letzten Metallisierungsschritt entsprechend geätzt.

Der Nachteil, daß die Programmierung immer ein spezieller Herstellungsschritt ist, führte zur Entwicklung von Festwertspeichern, **die der Anwender selbst programmieren kann.** Das

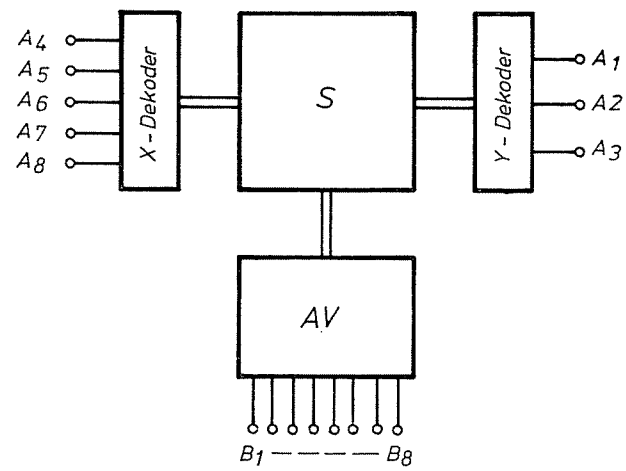


Abb. 2.47 — IM 7603 — ROM (Intersil)

Prinzip dieser Technik ist verhältnismäßig einfach. Bei der Herstellung werden sämtliche Matrixpunkte elektrisch leitfähig miteinander verbunden. Zum Programmieren legt man an die Wort- und Leseleitung, die das entsprechende Bit addressiert, einen so hohen Strom, daß das Koppelelement oder ein Teil dessen abschmilzt. Beim Lesen wird dieses veränderte Element die gegenteilige Information zu der eines „normalen" Elementes zeigen. Ein typisches Beispiel dieser Art zeigt Abb. 2.48. Hier besteht die eigentliche Zelle aus einer kurzen Nickel-Chrom-Bahn, die bei sämtlichen Koppelpunkten der Matrix Wort- und Leseleitung miteinander verbindet. Somit zeigen alle Zellen nach der Her-

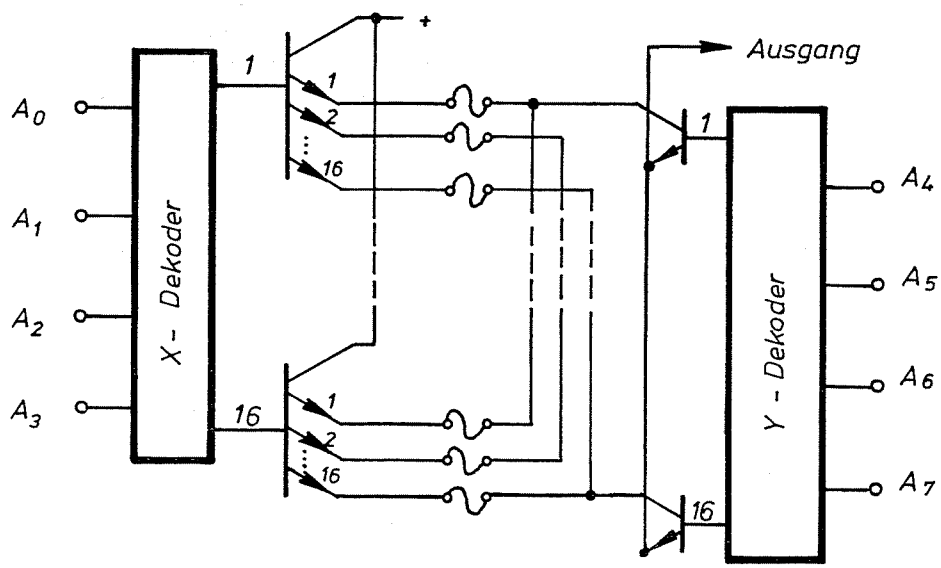


Abb. 2.48 — PROM — 1256 (Harris)

stellung eine „1". Zum Programmieren wird über die der betreffenden Zelle zugehörige Wort- und Leseleitung ein so hoher Stromimpuls gegeben, daß die Nickel-Chrom-Bahn abschmilzt. Die Zelle führt jetzt die Information „0". Die Ansteuer- und Leseelektronik des 1256 ist in bipolarer Technik ausgeführt, daher erreicht er 50 ns Zugriffszeit. Der Speicher ist in 256 Worten zu je 1 Bit organisiert.

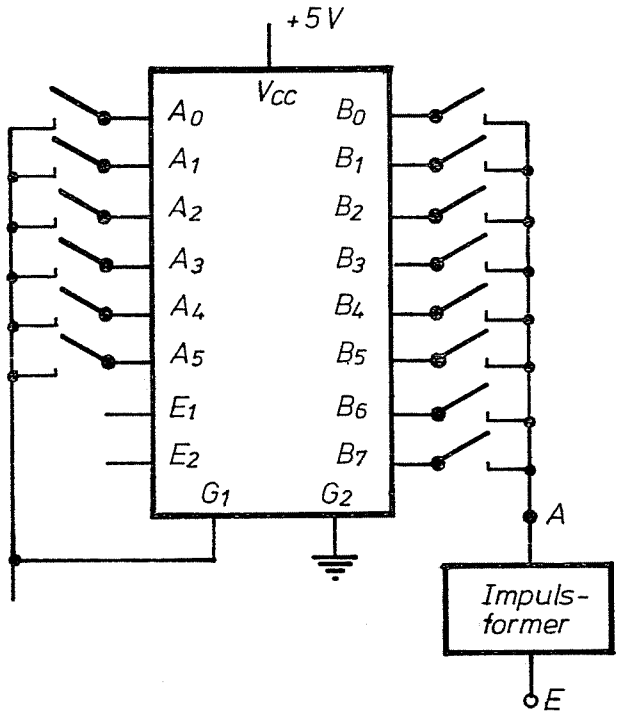


Abb. 2.49 a — PROM 0512

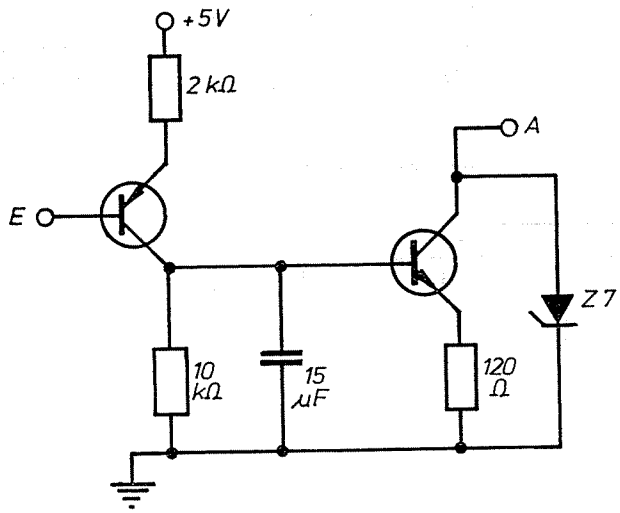


Abb. 2.49 b — PROM 0512

Die Firma Harris bietet noch eine andere Version eines PROM an — Typ 0512 —, bei der ebenfalls Verbindungsstellen zwischen Wort- und Leseleitung abgeschmolzen werden. Abb. 2.49 a zeigt die Sockelschaltung, Abb. 2.49 b den Impulsformer zum Programmieren. Die Schaltung dieses Impulsformers, dem ein üblicher Impulsgenerator vorgeschaltet werden muß, ist so einfach, daß sie in jedem Labor nachgebaut werden kann; d.h., jeder Anwender kann ohne Aufwand selbst programmieren.

Zu erwähnen sind noch zwei weitere Entwicklungen. Für Zwecke, bei denen der Inhalt von Festwertspeichern mehrmals geändert werden soll, bietet die Firma Intel folgendes Verfahren an: In den Koppelpunkten der Matrix liegen Feldeffekttransistoren, wobei die Drain-Source-Strecke das eigentliche Verbindungselement bildet. Wenn das Gate in ein sehr hochohmiges Gebiet eingebettet ist — wie das bei MOS-FET der Fall ist — genügt ein einmaliger Ladestoß auf das Gate, um den Transistor dauernd durchzuschalten. Dieser Ladestoß wird als einmaliger Durchbruchspannungsimpuls auf das Gate gebracht; dieses kann sich wegen der absoluten Isolationsfähigkeit der Umgebung nicht mehr entladen. Soll nun der Informationsinhalt des Speichers geändert werden, legt man das mit einem Fenster versehene Bauelement in energiereiche Strahlung (UV-Strahlung genügt bereits); die Gates der Koppeltransistoren werden entladen und können jetzt neu programmiert werden.

Als weiteres, neu programmierbares Festwertspeichersystem sind die sogenannten **Ovonics** zu nennen. Hier handelt es sich um Glaszusammensetzungen, die eine ähnliche Kennlinie wie Vierschichtdioden aufweisen. Diese, von der amerikanischen Firma Ovshinsky entwickelten Elemente sind ebenfalls sehr klein; als Speicherelement verwendet können sie entweder hochohmig (durch Überschreiten einer bestimmten Spannung) oder niederohmig (durch Stromüberschreiten) geschrieben werden. Die Ovonic-Speicher dürfen beliebig oft gelesen, aber nur bis zu 100mal neu beschrieben werden.

2.4. Geräte der Peripherie

2.4.1. Allgemeines

Im Rahmen dieser Ausarbeitung sollen die peripheren Geräte nicht in allen Einzelheiten ihres Aufbaus und ihrer Arbeitsweise geschildert werden. Hier sollen vielmehr in einem Überblick die wichtigsten Begriffe der Peripherie und die gebräuchlichsten peripheren Geräte in einer kurzgehaltenen Beschreibung zusammengefaßt

werden. In den vorangegangenen Abschnitten wurde die Funktion einer Datenverarbeitungsanlage (DVA) erklärt. Dabei wurde im wesentlichen auf die Zentraleinheit mit dem Leitwerk, dem Rechenwerk und dem Arbeitsspeicher eingegangen. Zu einer DVA gehören jedoch weiter als wichtige Bestandteile Einrichtungen, über die die Daten der verschiedensten Informationsträger (z.B. Lochkarten, Lochstreifen, Magnetbänder, Magnetkarten u.a.) eingegeben und die errechneten Ergebnisse auf geeignete Speichermedien wieder ausgegeben werden können. Für diese Zwecke stehen einer DVA eine Anzahl unterschiedlicher **Ein-/Ausgabegeräte** zur Verfügung. Diese Geräte werden zusammen mit den Externspeichern (z.B. Magnetbandgeräte) unter dem Oberbegriff **„Periphere Geräte"** zusammengefaßt. Oft findet man auch die Bezeichnung „Anschlußgeräte". **Die in ihrer Gesamtheit an einer DVA angeschlossenen Geräte nennt man Peripherie.**

2.4.2. Gliederung der peripheren Geräte

Werden in der elektronischen Datenverarbeitung Geräte in direkter Verbindung bzw. über eine Gerätesteuerung mit der Zentraleinheit betrieben, spricht man von **On-line-Betrieb.** Im Gegensatz dazu findet z.B. die Umwandlung eines Urbelegs (z.B. handschriftliche Aufzeichnung) in eine maschinenlesbare Form mit Hilfe

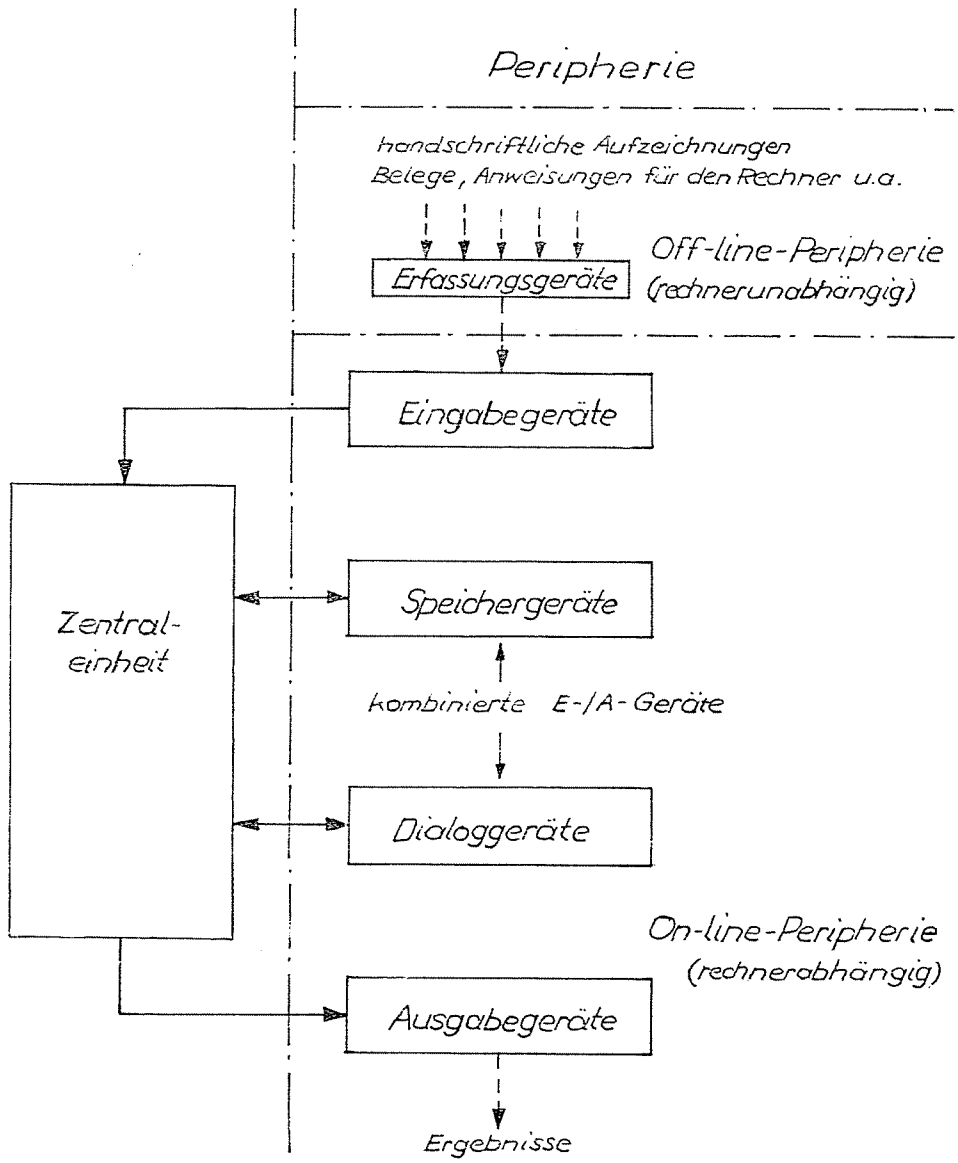


Abb. 2.50 — Übersicht über die Peripherie einer DVA

von Erfassungsgeräten (z.B. Karten- oder Streifenlocher), die nicht direkt mit der Zentraleinheit verbunden sind, im **Off-line-Betrieb** statt. Periphere Geräte, die nur in Abhängigkeit von der Zentraleinheit, also rechnerabhängig betrieben werden können, rechnet man zur **On-line-Peripherie.** Ist ein peripheres Gerät unabhängig von der Zentraleinheit arbeitsfähig, gehört es zur **Off-line-Peripherie.**

2.4.3. On-line-Peripherie

2.4.3.1. Anschluß der peripheren Geräte an die Zentraleinheit

Würde man die Ein-/Ausgabe mit ihren relativ langsamen Arbeitsgeschwindigkeiten direkt auf die Zentraleinheit arbeiten lassen, so wäre der Rechner bei jeder Ein-/Ausgabeoperation während dieser Zeit für andere Aufgaben blockiert. Aus diesem Grunde ist eine DVA mit einer Ein-/Ausgabesteuerung, auch Kanalsteuerung genannt, ausgerüstet. Zusammen mit den Ein-/Ausgabekanälen bildet sie eine vom Rechenablauf unabhängige Einheit. Die vom Steuerwerk eingeleiteten Ein-/Ausgabeoperationen werden von der Ein-/Ausgabesteuerung übernommen und von ihr selbständig zu Ende geführt. Die Ein-/Ausgabesteuerung und der jeweilige Kanal übernehmen die Koordination des Datenaustausches zwischen der Zentraleinheit und der On-line-Peripherie. Weitere Aufgaben der Ein-/Ausgabesteuerung sind

a) Auswahl eines entsprechenden peripheren Gerätes,

b) Überwachung eines oder mehrerer peripherer Geräte,

c) Starten von Operationsbefehlen,

d) Abgabe von Ein-/Ausgabe-Erledigungsmeldungen (Setzen von Programmunterbrechungen, die das Betriebssystem z.B. bei Multiprogramming auswertet),

e) Absetzen von Fehlermeldungen an die Zentraleinheit und

f) Weitergabe der gespeicherten externen Daten an den Arbeitsspeicher und umgekehrt usw.

Damit die peripheren Geräte mit der Ein-/Ausgabesteuerung zusammenarbeiten können, ist auch ihnen eine Steuerung, die Gerätesteuerung, zugeordnet. Sie befindet sich jeweils zwischen dem Kanal der Zentraleinheit und einem oder mehreren peripheren Geräten. Ihre Aufgabe besteht unter anderem darin, den Informa-

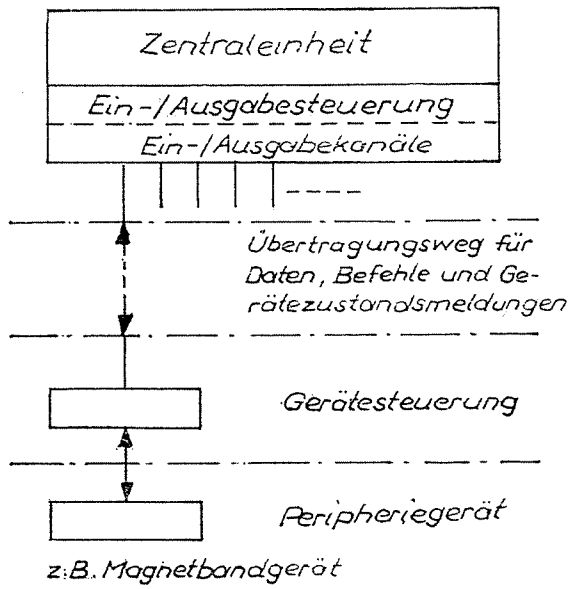


Abb. 2.51 — Grundsätzliche Darstellung eines Geräteanschlusses

tionsfluß der angeschlossenen Geräte unmittelbar zu steuern. Abb. 2.51 zeigt das Prinzip eines Geräteanschlusses, das für die gesamte On-line-Peripherie gilt. Die Aufgaben einer Gerätesteuerung können wie folgt zusammengefaßt werden:

1. Geschwindigkeitsumsetzung

Da die peripheren Geräte aufgrund ihrer mechanischen Arbeitsweise bedeutend langsamer arbeiten als die Zentraleinheit, ist eine Geschwindigkeitsumsetzung erforderlich. Hierfür sind entsprechende Pufferspeicher vorhanden, die mit der Arbeitsgeschwindigkeit der peripheren Geräte aufgefüllt werden. Die gespeicherten Daten werden dann mit der Geschwindigkeit der Zentraleinheit an die Ein-/Ausgabesteuerung abgesetzt. Entsprechend wird in umgekehrter Richtung verfahren.

2. Codeumwandlung

Die verschiedenen Codes der peripheren Geräte müssen in den Maschinencode der Zentraleinheit umgewandelt werden.

3. Formatumsetzung

Bei der Eingabe muß die Datenanordnung auf den externen Datenträgern entsprechend der in der Zentraleinheit erforderlichen Anordnung umgesetzt werden. Bei der Ausgabe sind die Daten in umgekehrter Richtung umzusetzen.

4. Impulsumsetzung

Die von der Zentraleinheit eintreffenden Steuerimpulse werden in Arbeitsimpulse umgesetzt, d.h., die Energie wird soweit angehoben, daß Arbeitsvorgänge der peripheren Geräte ausgelöst und gesteuert werden können.

Die Datenübertragungsleitung zwischen der Zentraleinheit und den Gerätesteuerungen nennt man Kanal. Er stellt die Verbindung zwischen der geräteorientierten Steuereinrichtung und der rechnerorientierten Ein-/Ausgabesteuerung dar. Man unterscheidet in der Datenübertragungstechnik zwei Arten von Kanälen, den Selektor- und den Multiplexkanal.

Selektorkanal: Der Selektorkanal eignet sich für den Anschluß peripherer Geräte mit hoher Übertragungsgeschwindigkeit (z.B. Magnetbandgeräte). Über ihn kann jeweils nur ein peripheres Gerät mit der Zentraleinheit korrespondieren, gleichgültig wie viele Geräte an dem Kanal angeschlossen sind. Damit wird klar, daß eine Simultanarbeit (Gleichzeitigkeitsverkehr) mehrerer an einen Selektorkanal angeschlossener Geräte nicht möglich ist.

Multiplexkanal: Wie der Selektorkanal arbeitet auch der Multiplexkanal zusammen mit der Ein-/Ausgabesteuerung unabhängig von der Zentraleinheit. Er ist in der Lage, den Informationsfluß in beiden Richtungen zeitlich ineinander verschachtelt zu übertragen. Auf diese Weise können an einen Multiplexkanal mehrere langsame periphere Geräte (z.B. Lochstreifen- und Lochkartenlesegeräte, Schnelldrucker u.a.) angeschlossen werden und simultan arbeiten.

Die **Schnittstelle,** auch Standardanschluß genannt, stellt den Übergang zwischen Kanal und Gerätesteuerung dar. Durch sie sind die Übergabebedingungen festgelegt und die Verbindungselemente (z.B. Stecker) genormt. Dadurch findet man an jeder Schnittstelle die gleichen Bedingungen vor, die die Voraussetzungen darstellen für eine freizügige Austauschbarkeit peripherer Geräte.

2.4.4. Gliederung der On-line-Peripherie

Wie aus Abb. 2.50 zu entnehmen ist, werden die rechnerabhängigen Geräte unterteilt in

1. **Eingabegeräte**
2. **Ausgabegeräte**
3. **Dialoggeräte und**
4. **Speichergeräte.**

2.4.4.1. Eingabegeräte

Abgesehen von den Speicher- und Dialoggeräten, die kombinierte Ein-/Ausgabegeräte darstellen und deshalb ebenfalls in der Lage sind, Daten eingeben zu können, werden nur die Lochstreifen- und Lochkartenlesegeräte sowie die Belegleser (Magnetschriftleser, Markierungsleser, Klarschriftleser u.a.) als Eingabegeräte bezeichnet. Sie dienen zum Lesen der auf den verschiedensten Datenträgern angebotenen Informationen und zum Umsetzen in die von der Zentraleinheit verwendeten Form der Informationsdarstellung. Die umgesetzten Daten werden über die Gerätesteuerung an die Zentraleinheit weitergegeben.

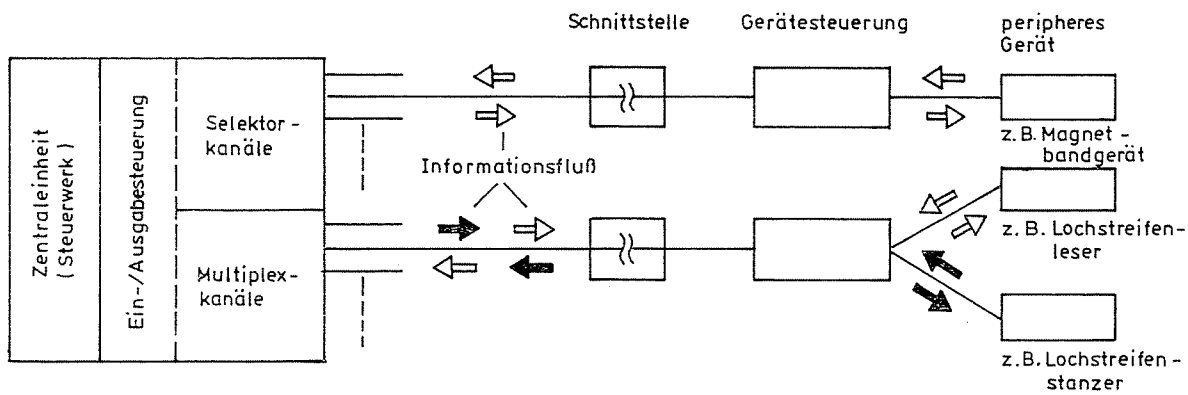


Abb. 2.52 — Grundsätzliche Darstellung des Selektor- bzw. Multiplex-Kanalprinzips

Lochstreifenleser

Der Lochstreifenleser ist so universell gestaltet, daß er Streifen mit 5, 6, 7 und 8 Lochspuren lesen kann. Je nach Modell erreicht er eine Lesegeschwindigkeit von 400 bis 2000 Zeichen (Lochkombinationen) pro Sekunde. Er besteht im wesentlichen aus

a) dem Transportmechanismus,

b) der Leseeinheit (Lesekopf) und

c) der Steuerung.

Der Streifenvorschub während des Lesevorgangs und das Zurückspulen des Streifens ist Aufgabe des Transportmechanismus. Der Lesekopf liest die einzelnen Zeichen in der Regel auf fotoelektrischem Wege. Die Steuerung übernimmt die Geschwindigkeits-, Format-, Code- und Impulsumsetzung (siehe Abschnitt 2.4.3.1.). Der Lochstreifenleser ist in der Lage, den Streifen auf Parityfehler zu überprüfen. Es können wahlweise geradzahlige oder ungeradzahlige Paritykontrollen durchgeführt werden. Die maximale Streifenlänge je Spule beträgt 300 m. Damit ergibt sich, bei einem Zeichenabstand von 10 Zeichen je Zoll oder 2,54 mm, ein Fassungsvermögen von 120 000 Zeichen je Spule. Der Streifen kann vorbereitend gelocht und mit Maximalgeschwindigkeit abgesetzt werden. Eine Korrektur falsch gelochter Zeichen ist nur mit großem Aufwand möglich und muß als Nachteil angesehen werden.

Abb. 2.53 zeigt eine kombinierte Lochstreifenlese und -stanzeinrichtung. Links befindet sich der Lochstreifenstanzer, rechts der Leser. Die gemeinsame Steuerung ist ebenfalls in der Geräteeinheit enthalten.

Lochkartenleser

Die Beschaffenheit und die Codierung der vom Lochkartenleser zu verarbeitenden Lochkarten kann als bekannt vorausgesetzt werden. Seine wichtigsten Bestandteile sind

a) die Kartenzuführungseinrichtung,

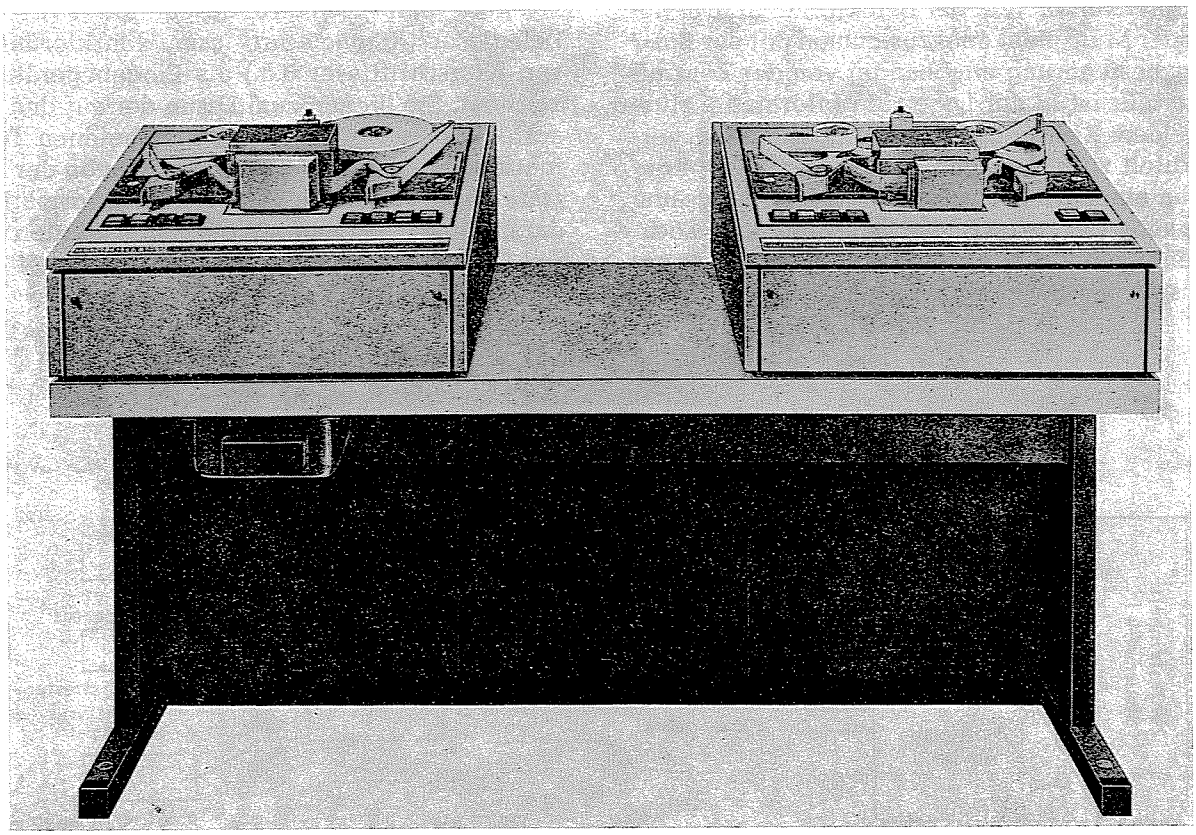
b) die Transporteinrichtung,

c) die Leseeinrichtung,

d) die Ablagefächer und

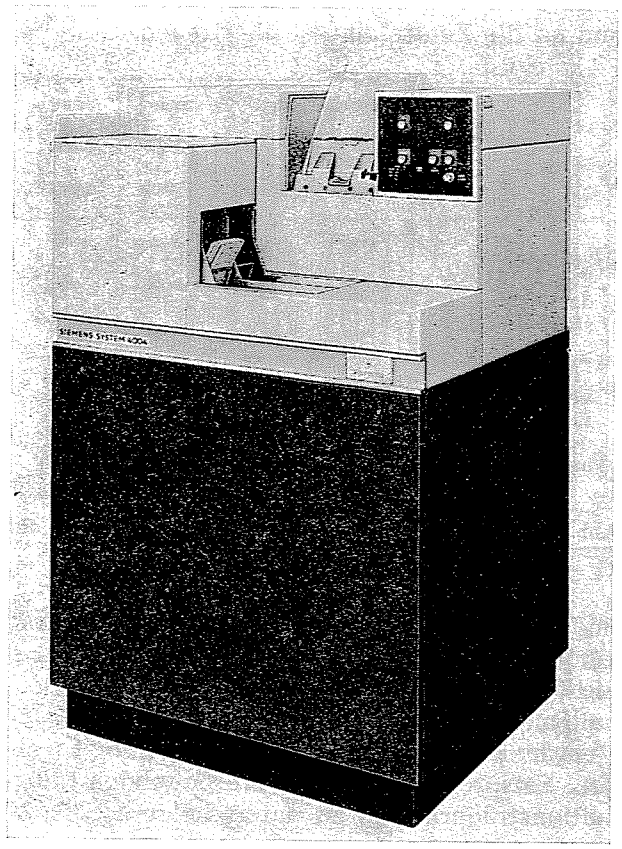
e) die Steuerung.

Der Kartenstapel wird in den Kartenzuführungsschacht eingelegt. Nacheinander werden die Karten einzeln von der Transporteinrichtung abgeholt, an der Leseeinrichtung vorbeigeführt und abschließend in eins der Ablagefächer gelegt. Jede Karte wird in der Leseeinrichtung einzeln und zwar spaltenweise — von Spalte 1 bis 80 — meist fotoelektrisch abgetastet. Um die Sicherheit des Lesevorgangs zu erhöhen, sind fast alle Geräte mit zwei Lesestationen ausgestattet, wobei die zweite als Kontrollstation dient. Über einen Vergleicher werden die gelesenen Daten miteinander verglichen und bei unzulässigen Lochkombinationen als fehlerhaft erkannt. Die beanstandeten Karten werden mit Hilfe des Ablagefachaus-



(Werkfoto Siemens)

Abb. 2.53 — Lochstreifenlese und -stanzeinrichtung



(Werkfoto Siemens)

Abb. 2.54 — Lochkartenlesegerät

wählers in das Fehlerfach gesteuert und dort abgelegt. Die Weitergabe der fehlerhaften Information an die Zentraleinheit wird unterdrückt. In der Leseeinrichtung erfolgt gleichzeitig die Umsetzung des Lochkartencodes — jede Spalte enthält 12 Lochmöglichkeiten — in den Maschinencode (z.B. EBCDI-Code) der Zentraleinheit. Die umgesetzten und als richtig erkannten Zeichen werden an den Arbeitsspeicher der Zentraleinheit weitergegeben.

Die Lesegeschwindigkeit der gebräuchlichsten Lochkartenlesegeräte liegt etwa bei 40 000 bis 86 000 Karten je Stunde. Dies entspricht einer Datenübertragungsrate von etwa 2000 Zeichen je Sekunde. Die Lochkarte hat gegenüber dem Lochstreifen den Vorteil, daß sie sich leicht ändern läßt und sortierfähig ist.

Belegleser

Von Seiten der Anwender besteht seit jeher der Wunsch, menschenlesbare Belege direkt, fehlerfrei und ohne größere Verzögerung eingeben zu können. Man möchte das nachträgliche Umsetzen der numerischen oder alphanumerischen Angaben auf maschinell lesbare Datenträger (wie Lochstreifen, Lochkarte, Magnetkarte, Magnetband usw.) vermeiden, um damit eine zusätzliche Fehlerquelle zu beseitigen und die dafür benötigte Zeit einsparen zu können. Damit wird die Forderung nach einer automatisch lesbaren Schrift erhoben. Zu diesem Zweck wurden die einzelnen Schriftzeichen stilisiert und standardisiert. Die Arbeiten der ISO (International Organization for Standardization) führten zur Festlegung der in der Bundesrepublik am weitesten verbreiteten **OCR-A-Schrift** (OCR-A = Optical Character Recognition, Form A). Sie ist genormt und besteht aus 10 Ziffernzeichen und 4 Hilfszeichen; eine Erweiterung auf 26 Buchstaben und 7 Sonderzeichen ist geplant. Ihre Darstellungsart, bezüglich einer universellen Lesbarkeit, stellt den bisher brauchbarsten Kompromiß dar.

Entsprechend dem maschinellen Auswerte- bzw. Abtastverfahren werden die automatisch erkennbaren Schriften unterteilt in optisch, magnetisch und elektrisch lesbare Schriften. Das

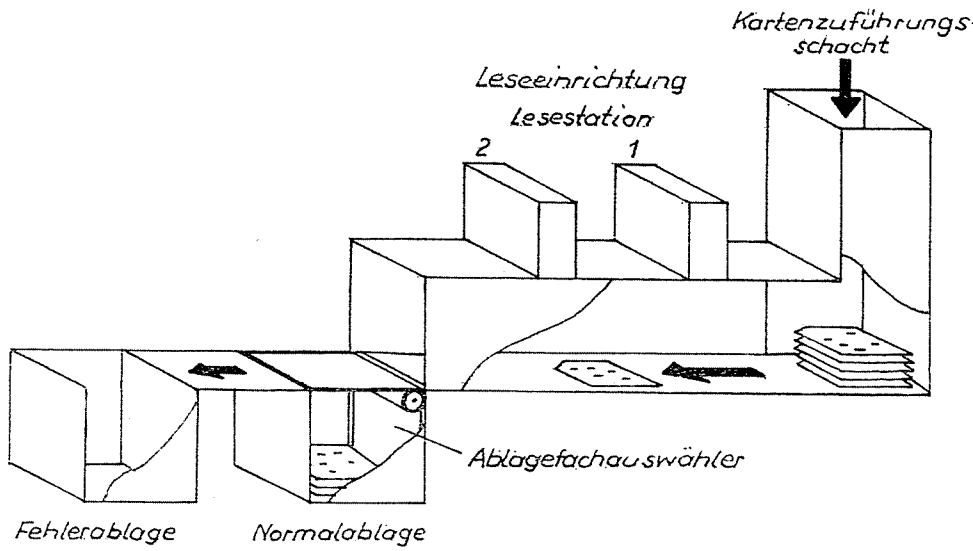


Abb. 2.55 — Grundsätzliche Darstellung der Funktion eines Lochkartenlesers

Prinzip der elektrischen Entzifferung — direkte Berührung der Datenträger mit den Abfühlbürsten — konnte sich wegen der großen Störanfälligkeit nicht durchsetzen. Bevor auf die OCR-A-Schrift etwas näher eingegangen wird, soll eine magnetisch lesbare Analogschrift an einem Beispiel erläutert werden. Die in Amerika häufig angewandte E-13-B-Schrift gehört zu dieser Gruppe. Ihre maschinelle Entzifferung beruht auf dem Prinzip der Übereinstimmung zweier Impulse. Die charakteristische Impulsform entsteht durch die Auswertung des sich laufend ändernden Widerstandswertes innerhalb der abgetasteten Zeichenbreite, wobei die einzelnen Zeichen vor dem Abtastvorgang magnetisiert werden.

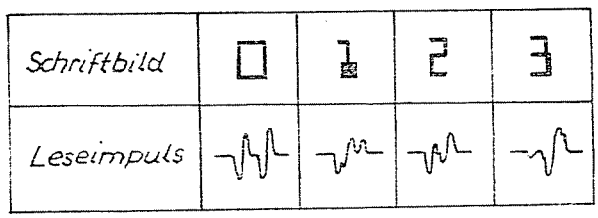


Abb. 2.56 — Die E-13-B-Schrift als Beispiel einer magnetisch lesbaren Analogschrift

Die ISO-Schrift OCR-A ist eine optisch lesbare Digitalschrift. Sie wird mit Hilfe von Fotozellen entziffert. Die einzelnen Zeichenfelder sind rasterartig aufgeteilt. Beim Lesevorgang wird dieses Raster auf seinen „Schwarz-Weiß-Inhalt" fotoelektrisch abgetastet. Die ermittelten Werte werden auf Übereinstimmung mit definierten Sollgrößen geprüft, danach digital umgesetzt und an die Zentraleinheit übertragen.

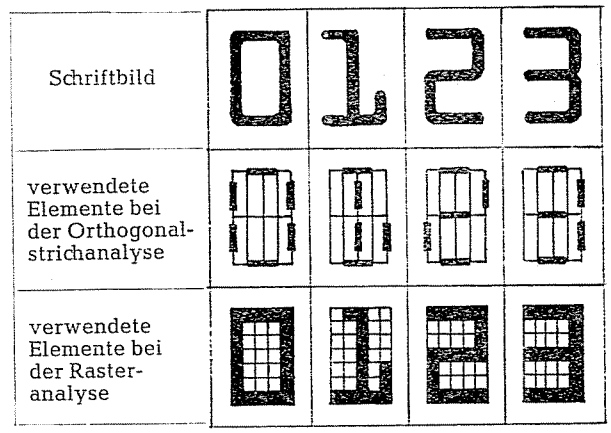
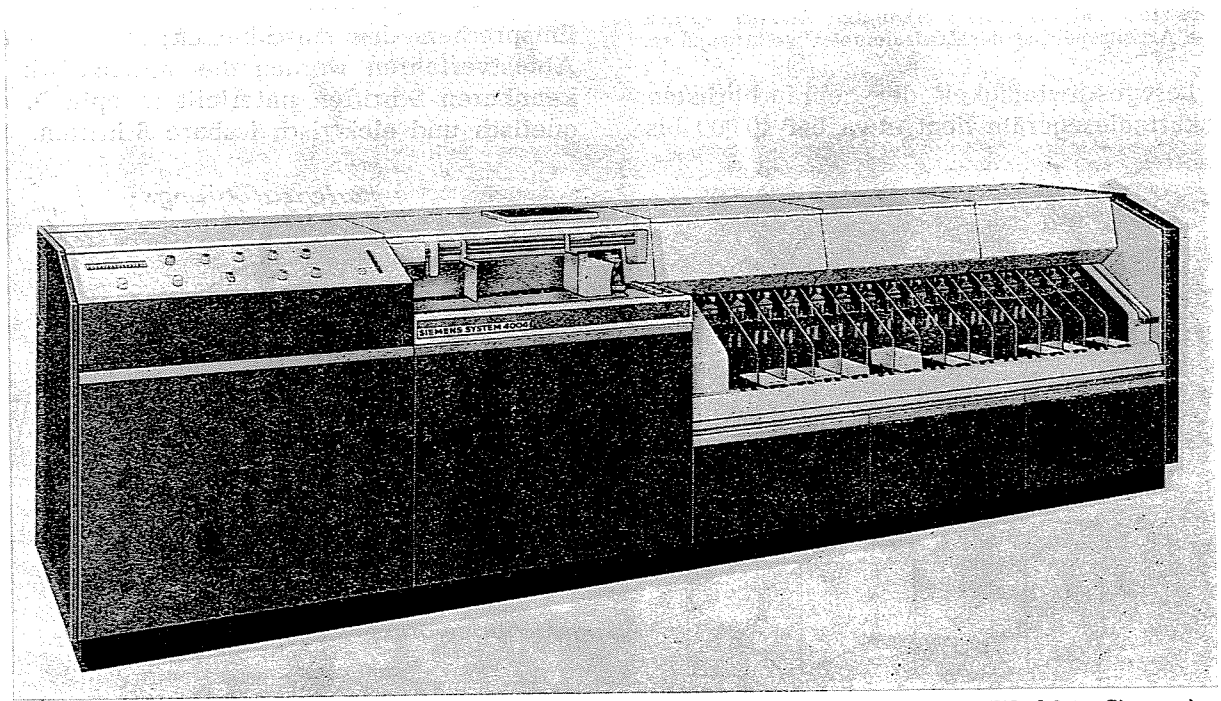


Abb. 2.57 — Darstellung der OCR-A-Schrift nach der Orthogonalstrich- und Rasteranalyse

Im Gegensatz zur magnetisch lesbaren Schrift kann die optisch erkennbare bei Verschmutzung nicht mehr einwandfrei entziffert werden. Deshalb müssen diese Datenträger besonders vor Verschmieren und Verschmutzen geschützt werden. Entsprechend den zu verarbeitenden Informationsträgern sind zu unterscheiden:

a) der **Markierungsleser,** der bestimmte Markierungen indentifizieren kann, die vorher an dafür festgelegte Stellen anzubringen sind. Seine Lesegeschwindigkeit beträgt etwa 3 bis 7 Belege pro Sekunde.



(Werkfoto Siemens)

Abb. 2.58 — Klarschrift- bzw. Belegleser

b) der **Magnetschriftleser,** der mit Hilfe von Magnetköpfen Schriftzeichen entziffert, die vorher mit einem ferrithaltigen Farbstoff aufgetragen werden müssen. Seine Lesegeschwindigkeit liegt zwischen 8 und 25 Belegen pro Sekunde.

c) der **Klarschriftleser,** der wegen seiner großen Verbreitung näher erläutert werden soll.

Die wichtigsten Teile des Klarschriftlesers sind:

a) das Eingabefach,

b) die Lesestation,

c) die Ablagefächer und

d) die Steuerung.

In das Eingabefach können bis zu 3600 Belege eingelegt werden. Die Lesestation liest Datenträger, die mit numerischen Zeichen der OCR-Schrift bedruckt sind. In einem Durchlauf wird jeweils eine Schriftreihe je Beleg gelesen. Es ist jedoch auch möglich, eine zweite Lesestation zum Abtasten einer zweiten Zeile je Durchlauf oder als Markierungsleseeinrichtung zum Erkennen von Handschriftmarkierungen einzusetzen. Das Gerät ist sowohl als Klarschriftleser als auch als kombinierter Belegleser und -sortierer auf dem Markt. Beide Geräte können im Off- und On-line-Betrieb arbeiten. Im On-line-Betrieb werden die gelesenen Zeichen an die Zentraleinheit übertragen, wobei die Zentraleinheit evtl. auch die Ablage der Belege steuert. Die Lesegeschwindigkeiten liegen, je nach Modell, bei etwa 14 bis 25 Belegen pro Sekunde.

2.4.4.2. Ausgabegeräte

Während die Eingabegeräte zum Erfassen, Umsetzen und zur Weitergabe der einer DVA angebotenen Informationen dienen, ist es Aufgabe der Ausgabegeräte, die vom Arbeitsspeicher eintreffenden Daten in die entsprechenden Codes umzusetzen und irgendeiner Form auf einen geeigneten Datenträger zu bringen. Die gebräuchlichsten Ausgabegeräte sind

a) der Lochstreifenstanzer,

b) der Lochkartenstanzer,

c) der Schnelldrucker und

d) der Plotter (Zeichengerät).

Lochstreifenstanzer

Diese Geräte stanzen Lochstreifen mit 5, 6, 7 und 8 Spuren der verschiedensten Breiten. Über eine Aufwickelvorrichtung können bis zu 300 m Lochstreifen aufgespult werden. Die wichtigsten Teile des Lochstreifenstanzers sind

1. die Transporteinrichtung,
2. die Stanzeinrichtung (Stanzkopf) und
3. die Steuerung.

Die Transporteinrichtung hat die Aufgabe, den Vorschub des Lochstreifens und das Aufwickeln während der Verarbeitung vorzunehmen. Die Stanzeinrichtung stanzt die umgeschlüsselten Daten zeichenweise, d.h., bei jedem Stanzvorgang wird die gesamte Lochkombination in den Lochstreifen gestanzt. Dabei wird eine Parityprüfung über alle 8 Spuren durchgeführt. Die Steuerung setzt die zur Ausgabe vorgesehenen rechnerinternen Daten in den jeweiligen Code des verwendeten Lochstreifens um. Weitere Aufgaben der Steuerung sind in Abschnitt 2.4.3.1. näher beschrieben. Die Stanzleistung bewegt sich, je nach Modell, zwischen 100 und 150 Zeichen pro Sekunde. In Abb. 2.53 ist ein Lochstreifenstanzer in Verbindung mit einem Lochstreifenleser dargestellt.

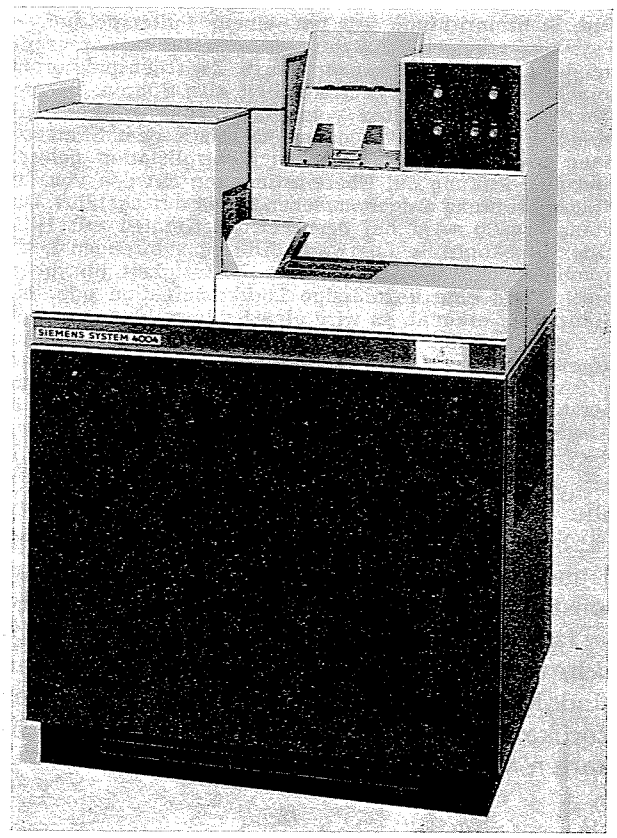
Lochkartenstanzer

Der Lochkartenstanzer dient dazu, die von der Zentraleinheit eintreffenden Daten umzusetzen und auf Lochkarten zu-übertragen. Er wird hauptsächlich dann verwendet, wenn die ausgegebenen Daten zu einem späteren Zeitpunkt maschinell weiterverarbeitet werden sollen. Seine wesentlichen Bestandteile sind

a) die Transporteinrichtung,

b) die Stanzeinrichtung und

c) die Steuerung.



(Werkfoto Siemens)

Abb. 2.59 — Lochkartenstanzer

Die Transporteinrichtung entnimmt dem Kartenmagazin automatisch Karte für Karte und transportiert sie zur Stanzeinrichtung. Danach werden sie an der Prüfstation vorbeigeführt und abschließend über den Ablagefachaus-

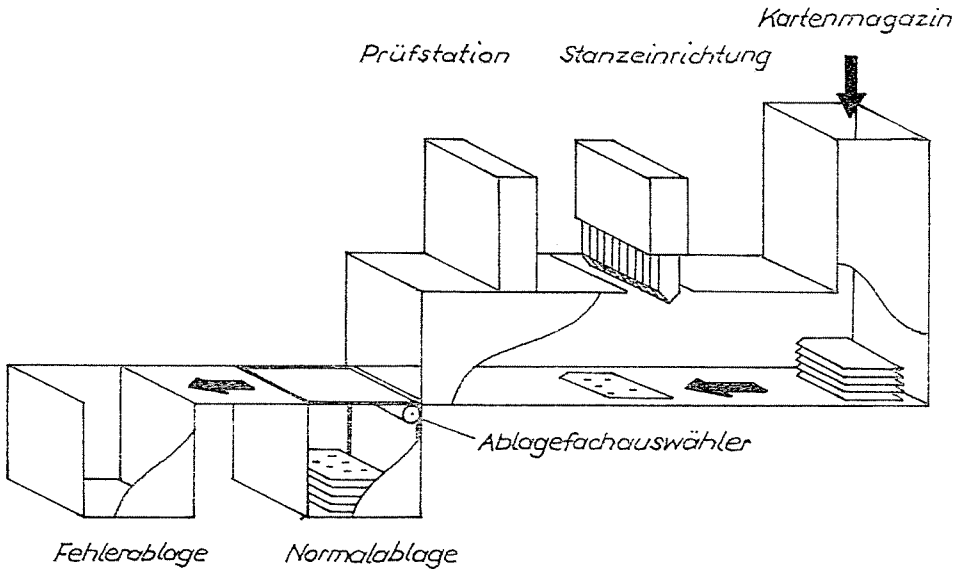


Abb. 2.60 — Grundsätzliche Darstellung der Funktion eines Lochkartenstanzers

wähler entweder im Normal- oder Fehlerfach abgelegt. Die gebräuchlichsten Lochkartenstanzer stanzen die 80-spaltigen Lochkarten **zeilenweise.** Zu diesem Zweck besitzen diese Stanzer 80 Stanzstempel. Verschiedene Geräte stanzen die Lochkarten auch mehrzeilig bzw. mehrspaltig oder alle möglichen Stanzstellen gleichzeitig. Die Stanzgeschwindigkeit eines Gerätes verhält sich entsprechend dem angewandten Stanzverfahren.

Die Stanzeinrichtung ist mit einem Pufferspeicher — Fassungsvermögen 80 Bytes — ausgestattet, der es ermöglicht, den Stanzvorgang durch Zwischenspeicherung unabhängig von der Zentraleinheit auszuführen. Bei der Übertragung der Daten vom Puffer zum Drucker wird eine Parityprüfung vorgenommen. Nach dem Stanzvorgang wird jede Karte von der Kontrollstation gelesen und eine Prüfung auf Übereinstimmung mit den von der Stanzeinrichtung aufgenommenen Daten durchgeführt. Der Stanzvorgang wird bei den meisten Geräten mit Hilfe einer Echokontrolle überwacht. Die zu stanzenden Lochkombinationen werden auf ihre Zulässigkeit hin überprüft. Wird eine unzulässige Lochkombination oder ein Stanzfehler erkannt, so wird eine Fehlermeldung erzeugt. Bevor die Daten in die Lochkarten gestanzt werden können, müssen sie vom rechnerorientierten Code in den entsprechenden Lochkartencode — definierte Lochkombinationen — umgesetzt werden. Diese und weitere, in Abschnitt 2.4.3.1. beschriebenen Aufgaben, werden von der Steuerung übernommen.

Die Stanzgeschwindigkeit beträgt, je nach Gerät bzw. verwendetem Stanzverfahren, zwischen 1 und 10 Karten pro Sekunde. Ist der Lochkartenstanzer mit einem „Binärzusatz" ausgestattet, so ist ein binäres Stanzen von Lochkarten möglich, wobei alle Lochkombinationen zugelassen sind. Mit einem weiteren Zusatzgerät kann der Lochkartenstanzer auch zum Lesen von Lochkarten verwendet werden.

Schnelldrucker

Entsprechend den verwendeten Druck- bzw. Schreibsystemen können die Geräte zur Klarschriftausgabe wie folgt gegliedert werden:

1. mechanische Drucker

a) Seriendrucker

Nadel- oder Drahtschriftdrucker (z.B. Lochschriftübersetzer)

Typenhebeldrucker (elektrische Büromaschinen)

Blatt- oder Streifenschreiber der Telegrafie (Fernschreiber)

Bedienungsblattschreiber einer DVA u.a.

b) Zeilendrucker

Kettendrucker

Trommeldrucker

Schnelldrucker

2. nichtmechanische Drucker bzw. Schreiber

Druck- bzw. Schreibvorgang geschieht auf physikalischem oder elektrochemischem, thermischem, elektrofotografischem, magnetischem oder elektrostatischem Wege (z.B. elektronisch gesteuerter Tintenschreiber u.a.)

Die wohl bekanntesten Seriendrucker sind der **Fernschreiber** und die **elektrische Schreibmaschine.** Die auf dem Typenhebel angebrachte Drucktype wird schlagartig gegen das vor einer Druckwalze befindliche Papier geführt und unter Verwendung eines Farbbandes zum Abdruck gebracht. Die Schreibgeschwindigkeiten liegen, je nach dem gewählten Gerät, zwischen 7 und 30 Zeichen pro Sekunde.

Der **Nadelschriftdrucker** druckt die einzelnen Zeichen mit Hilfe von Schreibnadeln, die jeweils

so angeordnet sind, daß aus einem Rechteck von 5 mal 7 Nadeln ein Zeichen gebildet werden kann. Die in Druckstellung gebrachten Nadeln stoßen gegen ein Farbband und bringen damit das Schreibzeichen auf dem dahinter liegenden Papier zum Abdruck. Die Schreibgeschwindigkeit liegt, je nach Modell, zwischen 8 und 16 Zeichen pro Sekunde.

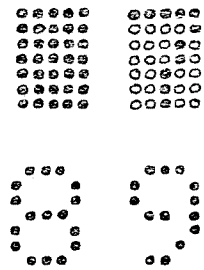


Abb. 2.61 — Prinzipielle Darstellung des Nadelrasters und der sich daraus bildenden Nadelrasterzeichen

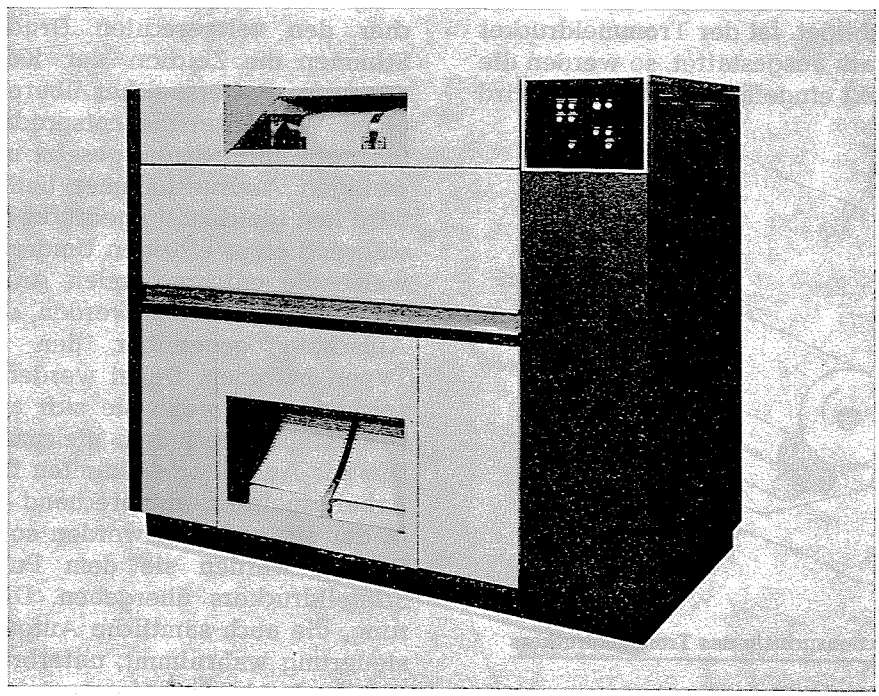
Die **nichtmechanischen Drucker** haben bisher, trotz des Vorteils einer lautlosen Arbeitsweise, noch nicht die Bedeutung erlangt wie die mechanischen Drucker. Auf eine Erläuterung dieser Geräte soll deshalb verzichtet werden. In der Datenverarbeitung werden heute fast ausschließlich mechanische Zeilendrucker verwendet. Der Schnelldrucker ist ein solches, speziell für DVA konstruiertes Datenausgabegerät. Mit ihm werden die von der Zentraleinheit eintreffenden Daten in Klarschrift übersetzt. Seine wesentlichen Bestandteile sind

a) die Druckeinheit und

b) die Druckersteuerung.

Der Schnelldrucker stellt einen Hochleistungsdrucker dar, der entsprechend dem verwendeten Typenträger als Ketten- oder Trommeldrukker zur Anwendung kommt. Beim **Kettendrucker** sind alle Buchstaben, Ziffern und Sonderzeichen auf einer Kette angeordnet, die den eigentlichen Typenträger bildet. Vor der Typenkette befindet sich ein Farbtuch und dahinter das Papier. Die Kette wird parallel an einer Reihe von Andruckhämmern vorbeigeführt, wobei bei einem Typenabdruck der entsprechende Andruckhammer von hinten gegen Papier, Farbtuch und Type schlägt. Es kommt immer dann zu einem Abdruck, wenn sich die abzudruckende Type in einem bestimmten Augenblick vor der Druckstelle befindet. Auf die Steuerung des Druckvorganges wird bei der Erläuterung des Trommeldruckers näher eingegangen. Die Anzahl der Andruckhämmer richtet sich nach der Zahl der Druckstellen, die in einer Zeile abgebildet werden können. Zur Erhöhung der Schreibleistung sind die abdruckbaren Zeichen mehrmals, bis zu 5 Gruppen zusammengefaßt, auf der Typenkette enthalten.

Der Trommeldrucker, auch Typenwalzendrucker genannt, verwendet als Typenträger eine Walze, deren Breite der jeweiligen Zeilenlänge entspricht. Die Typenwalze setzt sich aus so vielen Typenrädern zusammen wie Schreibstellen in-



(Werkfoto Siemens)

Abb. 2.62 — Schnelldrucker

nerhalb einer Zeile (bis zu 160) vorhanden sind. Auf dem Umfang eines jeden Typenrades sind in den meisten Fällen 64 Zeichen (26 Großbuchstaben, 10 Ziffern, 28 Sonderzeichen) angeordnet. Alle übrigen 192 Zeichenkombinationen des EBCDI-Codes können nicht abgedruckt werden; an der entsprechenden Druckstelle wird ein Zwischenraum gebildet. Ist der Trommeldrucker mit einem Farbwerk ausgestattet, so werden die Schreibtypen direkt eingefärbt, andernfalls wird

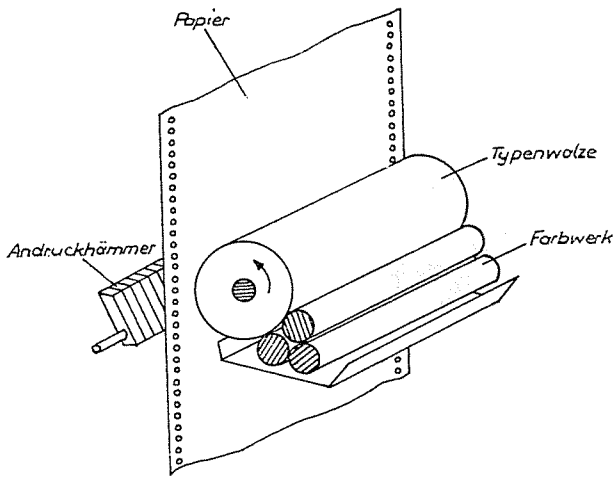


Abb. 2.64 — Funktionsprinzip des Trommeldruckers

zum Abdruck ein Farbtuch verwendet, das sich zwischen Papierbahn und Typenwalze befindet. Auch der Trommeldrucker besitzt so viele Andruckhämmer wie Schreibstellen in einer Zeile vorgesehen sind. Der Druck einer kompletten Zeile dauert so lange, wie die Typenwalze für

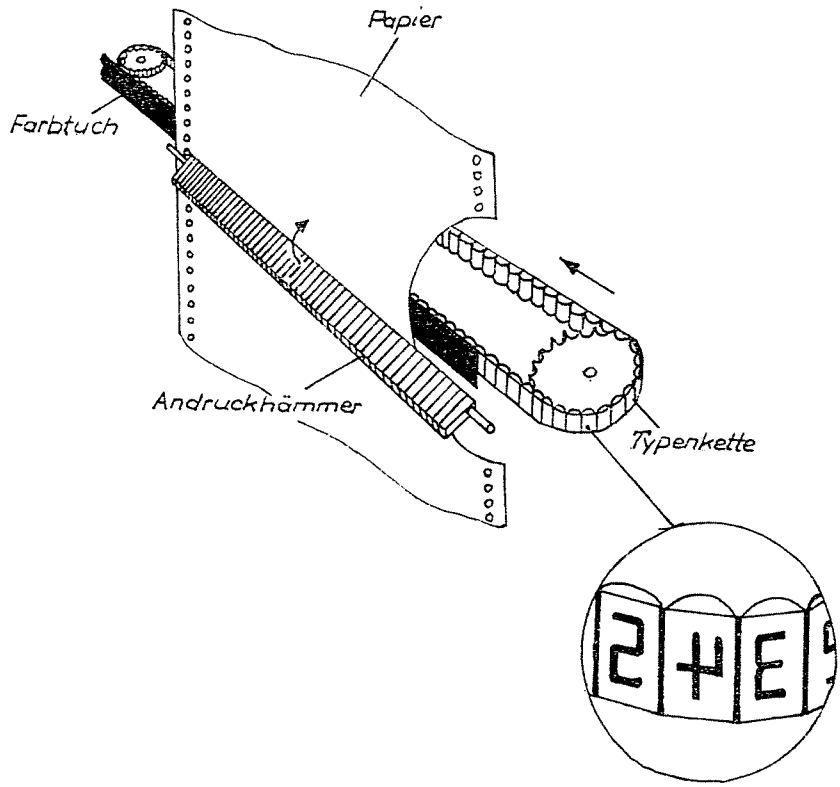


Abb. 2.63 — Funktionsprinzip des Kettendruckers

eine Umdrehung benötigt. Gleichzeitig mit der sich ständig drehenden Typenwalze rotiert eine Kontrollwalze, mit deren Hilfe die Druckersteuerung im Zusammenwirken mit dem Pufferspeicher den zeitgerechten Druckablauf steuert. Stimmen die Zeichen der Kontrollwalze mit denen im Pufferspeicher überein, werden über einen Verstärker die entsprechenden Andruckmagnete unter Strom gesetzt und somit die jeweiligen Andruckhämmer betätigt. Eine Zeile wird fast gleichzeitig gedruckt und zwar so, daß während einer einzigen Umdrehung der Typenwalze die entsprechenden Andruckhämmer so rechtzeitig ausgelöst werden, daß sie sich beim Aufschlag gegenüber den abzudruckenden Typen befinden. Dabei werden gleiche Schriftzeichen, auch wenn sie sich an verschiedenen Druckstellen befinden, gleichzeitig abgedruckt. Die zur Ausgabe bestimmten Daten werden im Arbeitsspeicher, entsprechend der Reihenfolge, in der sie gedruckt werden sollen, aufbereitet. Danach werden sie dem Pufferspeicher des Schnelldruckers übergeben. Die Druckersteuerung, die auch sämtliche Aufgaben der Gerätesteuerung wahrnimmt, unterbricht die Übertragung, sobald der Puffer — dessen Fassungsvermögen eine Druckzeile beträgt — gefüllt ist. Wird ein unzulässiges Zeichen in den Puffer-

speicher eingegeben, so wird ebenfalls eine Fehlermeldung erzeugt. Der eigentliche Druckvorgang wird unabhängig von der Zentraleinheit ausgeführt. Mit dem Schnelldrucker können Papierbahnen von 50 bis 552 mm Breite bedruckt

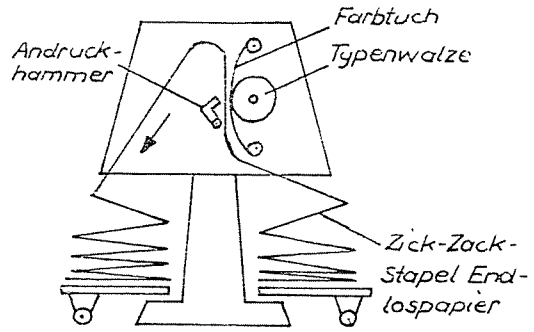


Abb. 2.65 — Prinzip des Papierdurchlaufs

werden. Der Druck erfolgt auf Endlospapier- oder Endlosvordruckbahnen, die in Zickzacklagen gefalzt sind. Die bedruckten Bögen legen sich von selbst, bedingt durch die an den Trennungslinien angebrachten Perforationen, in Zickzackstapeln zusammen. Mit Hilfe der Druckersteuerung ist es möglich, bei zweibahnigen Druckern beide Papierbahnen gleichzeitig, jedoch unabhängig voneinander zu bedrucken. Um den notwendigen Papiervorschub bewerkstelligen zu können, müssen der Druckeinheit bestimmte Befehle erteilt werden. Diese Steuerdaten werden dem Pufferspeicher entweder durch das Programm oder über einen vorgefertigten Steuer- bzw. Vorschublochstreifen übergeben. Der Steuerlochstreifen ist ein zusammengeklebter Streifen, der in eine im Schnelldrucker dafür vorgesehene Abtasteinrichtung eingelegt wird. Je nach gewünschtem Vorschub wird ein spezieller Steuerlochstreifen angefertigt und bei Bedarf ausgewechselt. Der Vorschub kann entweder nur durch das Programm oder über den Steuerlochstreifen oder über beide vorgenommen werden. Im Gegensatz zu den nichtmechanischen Druckern können mit dem Schnelldrukker bis zu 5 Durchschläge angefertigt werden. Die Typenkette sowie die Typenwalze sind leicht herauszunehmen und gegen einen anderen Typenträger — mit verschiedenem Zeichenvorrat — auszutauschen. Die Schreibgeschwindigkeiten betragen, je nach verwendetem Modell und genutztem Zeichenvorrat, beim Kettendrucker bis zu 61 000 und beim Trommeldrucker bis zu 90 000 Zeilen pro Stunde.

Plotter

Schon seit langem sind rechnergesteuerte Zeichengeräte zur Herstellung äußerst genauer und differenzierter Zeichnungen im Gebrauch. Die Wirkungsweise eines solchen graphischen Ausgabegerätes soll anhand eines Plotters kurz erläutert werden.

Entsprechend der Konstruktion und des angewandten Zeichenverfahrens lassen sich die verschiedenartigen Plotter unterteilen in **mechanische Plotter** (mechanische Flachzeichner, mechanische Trommelzeichner) und **elektronische Plotter.**

Der mechanische Plotter als **Flachzeichner** verwendet die X-Y-Koordinaten zur Darstellung der abzubildenden Zeichen. Sowohl die X- als auch die Y-Achse werden durch einen eigenen Motor getrennt voneinander angetrieben. Zur Steuerung müssen die einzelnen Koordinatenwerte festgelegt und in einem Rechner abgespeichert sein. Für jedes darzustellende Zeichen muß demnach eine Art Unterprogramm erstellt werden. Ein geeignetes Ablaufprogramm fügt die einzelnen Unterprogramme in der richtigen Reihenfolge zusammen und steuert so, unter Mitwirkung der Gerätesteuerung im On-line-Betrieb, das Zeichengerät. Die Steuerung ist auch im Off-line-Betrieb, über externe Datenträger, möglich. Zur Darstellung der Zeichnungen können Transparent- oder Zeichenpapier, Zeichenkarton, geeignete Folien usw. verwendet werden. Mit einem Flachzeichner kann man unter gewissen Voraussetzungen auch gravieren. Beim **Trommelzeichner** wird das Zeichenpapier über eine Trommel geführt. Das hat zwar einerseits den Vorteil, daß dieses Gerät ohne Unterbrechung eine Vielzahl von Zeichnungen anfertigen kann; andererseits jedoch ist es nicht möglich, bestimmte Zeichnungsmaterialien (z.B. Zeichenkarton) zu verwenden. Das Schreibgerät bewegt sich beim Trommelzeichner nur in horizontaler Richtung; die vertikalen Bewegungen ergeben sich durch die Trommeldrehung. Zum Abdruck des Linienverlaufs werden bei beiden Geräten überwiegend Kugelschreiberminen oder Tuschefedern benutzt.

Der elektronische Plotter zeichnet auf Mikrofilm. Die Zeichnung ist deshalb nicht sofort sichtbar; sie muß erst entwickelt werden. Ein Katodenstrahl belichtet den Film und hält so den abzubildenden Linienzug fest. Seine Steuerung geschieht ähnlich wie beim Flachzeichner. Statt der dort verwendeten mechanischen Antriebssysteme werden bei ihm für die X-Y-Koordinaten elektronische Ablenksysteme verwendet.

2.4.4.3. Dialoggeräte

Datenendgeräte, die einen Informationsaustausch zwischen Benutzer und DVA ermöglichen, nennt man Dialoggeräte. Der Datenaustausch zwischen Datenendgerät und DVA erfolgt wechselseitig. Eine Anfrage wird vom Rechner derart beantwortet, daß daraus Erkenntnisse gewonnen und Rückschlüsse gezogen werden können. Während des Arbeitsprozesses kann es erforderlich werden, Fragen an den Rechner zu richten, die dieser in bestimmter Weise zu beantworten hat. Das gleiche geschieht in umgekehrter Richtung; Meldungen des Rechners erfordern eine Beantwortung durch den Bediener. Folgende Datenendgeräte gestatten einen Dialogverkehr zwischen dem Benutzer und einer DVA

a) Fernschreiber,

b) Bedienungsblattschreiber und

c) Datensichtstation.

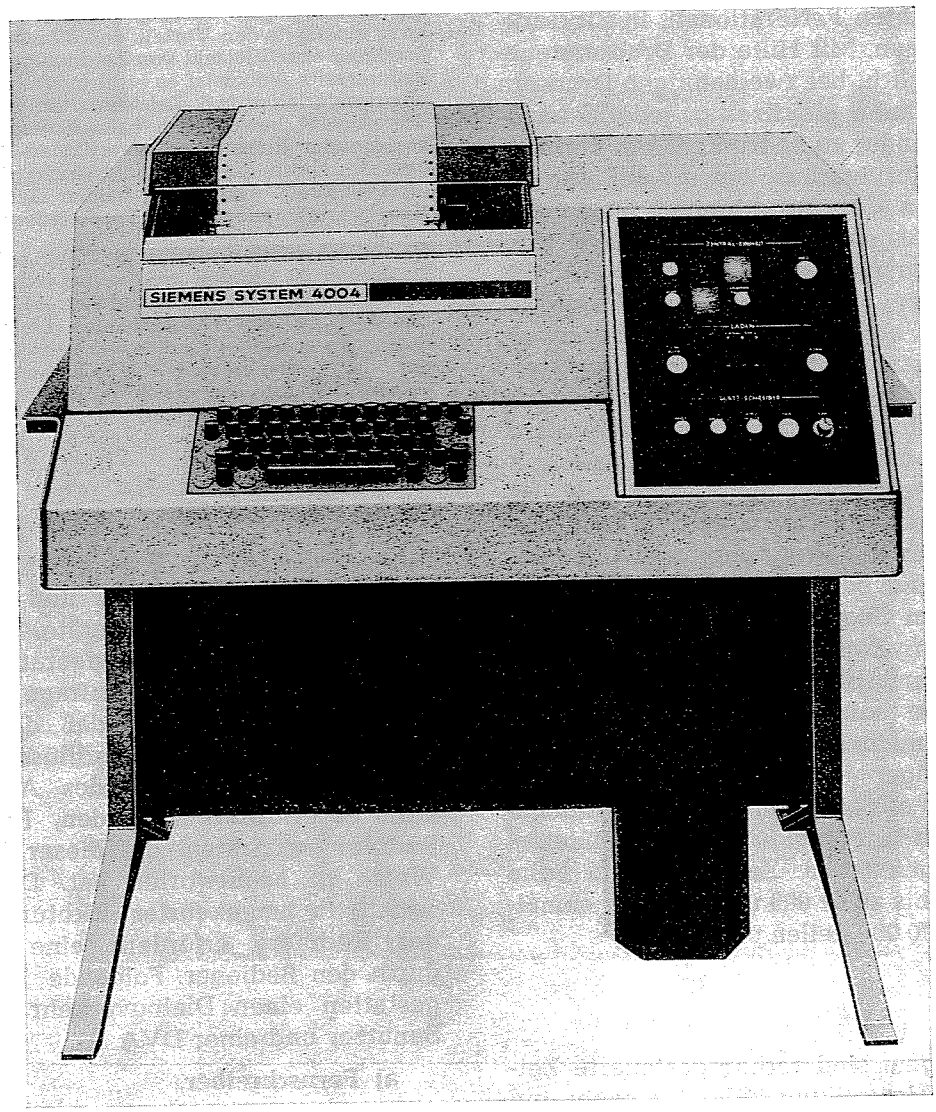
Fernschreiber

Der Fernschreiber eignet sich besonders zur Ein- und Ausgabe von geringen Datenmengen. Er wird in der Praxis auch zu Kontroll- und Steuerzwecken verwendet. Einerseits gibt der Rechner programmierte Informationen an ihn ab; andererseits nimmt er Anweisungen, z.B. Aufruf eines Programms, vom Rechner entgegen. Die wesentlichen Bestandteile eines Fernschreibers sind:

das Tastenwerk,
der Sender,
der Empfänger,
der Drucker und
der Antriebsmotor.

Der Fernschreiber ist über eine Steuereinheit, die sich unmittelbar beim Rechner befindet, mit der DVA verbunden. Die Steuereinheit hat die Aufgabe, sowohl eine Codeumsetzung als auch eine Geschwindigkeitsanpassung vorzunehmen.

Die Tastatur eines Fernschreibers ist in ihrem Aufbau ähnlich der einer gewöhnlichen Schreibmaschine und wird wie diese bedient. Beim Sendevorgang wird eine Taste angeschlagen, die das Schließen bzw. Öffnen von Sendekontakten bewirkt. Je nach Stellung dieser Kontakte werden die von einem Impulsgeber abgegebenen Impulse entweder als „Strom" bzw. „Kein Strom" zum Empfänger übertragen. Der geschriebene Text kann auf dem eigenen Fernschreiber mitgelesen und als Protokoll verwendet werden. Der Fernschreiber arbeitet nach dem Start-Stop-Prinzip, d.h. Sende- und Empfangsstation werden synchron betrieben, wobei immer der erste Sendeimpuls als Start- und der letzte als Stopimpuls ausgewertet werden. Jedes Fernschreibzeichen besteht aus 7 Impulsen, dem 20 ms langen Startimpuls, den 5 je 20 ms langen Informationsimpulsen und dem 30 ms langen Stopimpuls. Der Fernschreiber kann zusätzlich mit einem Lochstreifensender ausgestattet sein. Dieses Gerät erlaubt das Absetzen eines auf einem Handlocher vorher gefertigten Lochstreifens mit der maximalen Schreibgeschwindigkeit von etwa 10 Zeichen pro Sekunde. Ein weiteres Zusatzgerät ist der Empfangslocher, mit dem die an einem Fernschreiber ankommende Information auf einen Lochstreifen übertragen werden kann.



(Werkfoto Siemens)

Abb. 2.66 — Bedienungsblattschreiber

Bedienungsblattschreiber

Der Bedienungsblattschreiber (vgl. hierzu Abb. 2.66) ist im Grunde ein speziell an die Arbeitsweise einer DVA angepaßter Fernschreiber. Im wesentlichen besteht er aus dem Drucker, der Tastatur und dem Bedienungsfeld. Die beim Fernschreiber vorhandene Steuerung ist hier in die Zentraleinheit integriert, da der Bedienungsblattschreiber eine für den Betrieb eines Rechners unbedingt notwendige Einrichtung darstellt. Die Arbeitsweise des Ein-Ausgabeblattschreibers entspricht der des Fernschreibers, wobei der Text fortlaufend mit Großbuchstaben geschrieben wird. Seine Schreibgeschwindigkeit beträgt, je nach Modell, bis zu 20 Zeichen pro Sekunde. Als selbständige Einheit dient der Bedienungsblattschreiber zur Überwachung und Bedienung der Zentraleinheit sowie zum Austausch von Daten zwischen dem Rechner und dem Operateur (Bediener einer DVA). Mit den im Bedienungsfeld enthaltenen Lampentasten, Drucktasten, Anzeigelampen, Schaltern und akustischen Signalgebern ist der Operateur in der Lage, die Zentraleinheit manuell zu steuern und jederzeit den augenblicklichen Arbeitszustand des Rechners zu überblicken.

Die wichtigsten, über den Bedienungsblattschreiber abzuwickelnden Arbeiten sind

a) Starten bzw. Stoppen eines Programms,
b) Eingeben von Steuerdaten zur Fehlerberichtigung,
c) Zuordnen von Geräten,
d) Papier- oder Spulenwechsel vornehmen,
e) Eingeben von Bedienungsanweisungen und
f) Beantworten und Auswerten von Meldungen.

Alle vom Operateur eingegebenen und von der DVA an den Bedienungsblattschreiber ausgegebenen Daten werden protokolliert, so daß ein lückenloser Nachweis über den Datenaustausch zwischen Operateur und Rechner vorliegt.

Datensichtstation

Die Datensichtstation, auch Bildschirmgerät oder data display genannt, ist ein in direkter Verbindung mit der Zentraleinheit arbeitendes Dialoggerät zum Austausch alphanumerischer Zeichen. Sie ermöglicht dem Benutzer, in kürzester Zeit selbst umfangreiche Informationen visuell zu erfassen. Die Datensichtstation besteht im wesentlichen aus

a) der Anzeigeeinheit (Bildschirm),
b) der Tastatur und
c) der Steuereinheit.

Die Anzeigeeinheit verwendet zur Abbildung der Daten den Bildschirm einer Katodenstrahlröhre. Bei den meisten Geräten ist zur Steuerung des Elektronenstrahls ein Festwertspeicher vorhanden, der sämtliche Steuerinformationen enthält, die zur Darstellung des gesamten Zeichenvolumens notwendig sind. Der Festwertspeicher ist Bestandteil eines Zeichengenerators, der entweder nach dem Raster-, dem Profilstrahl- oder nach dem Griffelverfahren arbeitet.

Beim **Rasterverfahren** wird ähnlich wie beim Aufzeichnungsverfahren eines Fernsehbildes vorgegangen. Der Elektronenstrahl wird entweder in X- oder Y-Richtung abgelenkt und erzeugt dabei sichtbare Bildpunkte, die je nach Anordnung die entsprechenden Zeichen ergeben.

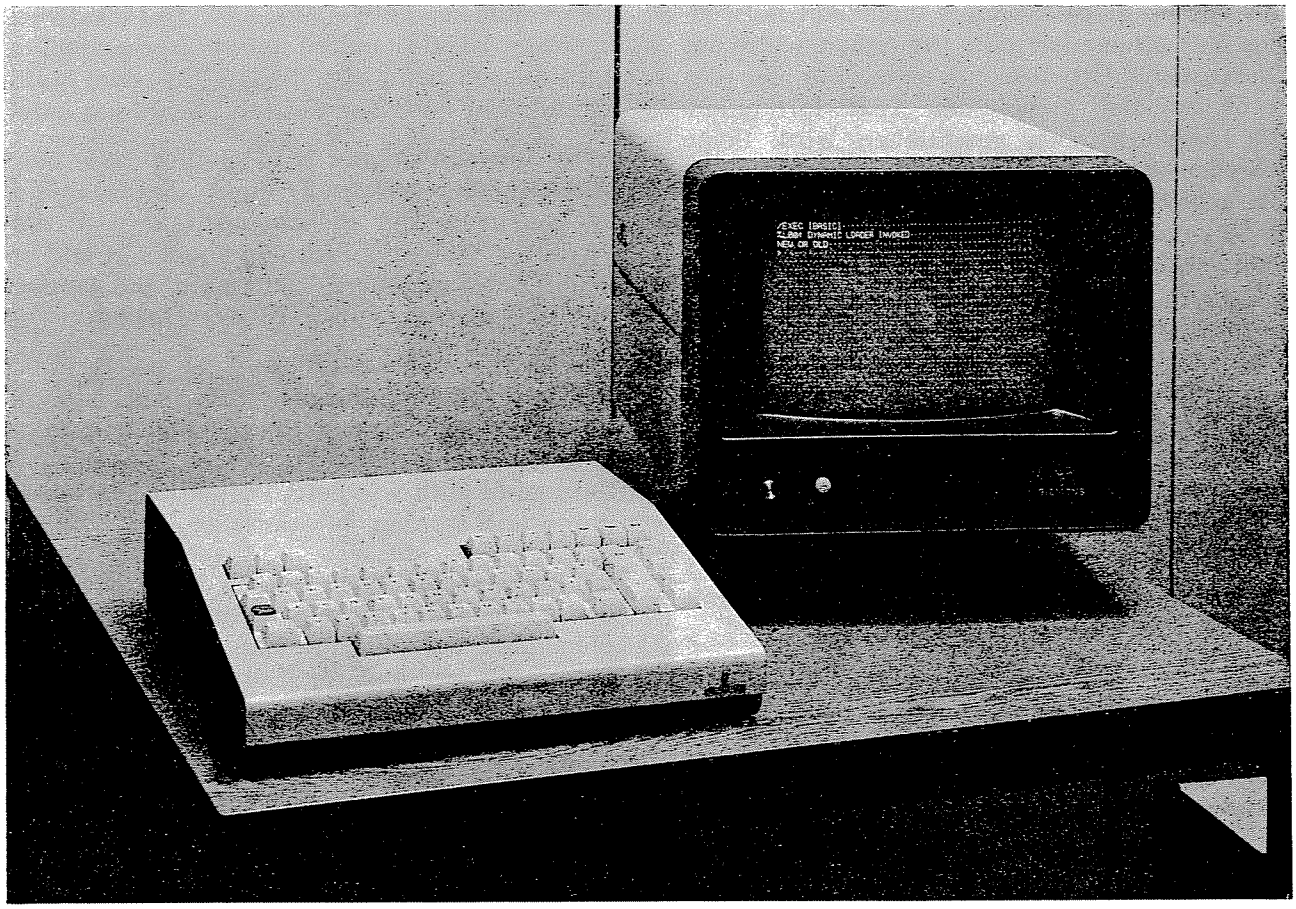
Bei der **Profilstrahltechnik** ist für jedes darstellbare Zeichen in einer Spezialelektronenröhre eine Schablone enthalten, wobei die einzelnen Zeichen ähnlich wie beim Projektionsverfahren dargestellt werden.

Bei der **Griffelmethode** werden die darzustellenden Zeichen aus einzelnen Strich- und Bogenelementen zusammengesetzt. Diese Methode hat gegenüber anderen Darstellungsverfahren den Vorteil, daß sie weniger Steuerdaten benötigt, da für jedes Strich -bzw. Bogenelement nur jeweils die Anfangs- und Endkoordinaten notwendig sind. Codezeichen der Zentraleinheit dienen zur Adressierung der im Festwertspeicher des Zeichengenerators enthaltenen Steuerbefehle. Zu den bisher geschilderten Datenaufzeichnungsverfahren, bei denen die einzelnen Zeichen aus vorher gespeicherten Formelementen zusammengesetzt werden, kommt noch ein weiteres hinzu. Bei diesem Verfahren wird der Elektronenstrahl von einem in der Zentraleinheit laufenden Programm geführt. Datensichtstationen dieser Art werden besonders bei der Lösung von Konstruktionsproblemen eingesetzt.

In einem eigenen Laufzeitspeicher werden Ein- bzw. Ausgabedaten gespeichert, so daß die Zentraleinheit nur während des Datenaustausches belegt ist. Gleichzeitig werden die gespeicherten Daten, entsprechend ihrer vorgesehenen Anordnung auf dem Bildschirm, formatgerecht aufbereitet. Darüber hinaus dient der Laufzeitspeicher der Bildwiederholung (etwa 65mal pro Sekunde), die eine flimmerfreie Darstellung der Zeichen gewährleistet.

Für die Dateneingabe ist die Datensichtstation mit einer Tastatur ausgestattet. Sie entspricht einer normalen Schreibmaschinentastatur, die durch eine Reihe von Funktionstasten erweitert wurde. Mit ihr wird der Eingabetext eingetastet, wobei die Daten vom Laufzeitspeicher der Gerätesteuerung aufgenommen und gleichzeitig mit dem Eintasten auf dem Bildschirm sichtbar werden. Damit kann man den geschriebenen Text überprüfen und gegebenenfalls korrigieren, wobei falsche Zeichen einzeln gelöscht oder überschrieben werden können. Ist eine Verbindung mit der Zentraleinheit hergestellt, so wird der eingespeicherte Text mit einer Geschwindigkeit von 1200 bis 4800 Bits pro Sekunde übertragen. Die Datensichtstation kann zusätzlich mit einem Datenschreiber ausgerüstet werden. Durch Betätigen einer dafür vorgesehenen Taste wird der im Laufzeitspeicher enthaltene bzw. auf dem Bildschirm abgebildete Text mit einer Geschwindigkeit von 20 Zeichen pro Sekunde ausgedruckt.

Spezielle Datensichtstationen, die mit einer besonderen Steuereinheit ausgerüstet sein müssen, ermöglichen die Dateneingabe mit Hilfe eines „Lichtgriffels". Bei diesem Verfahren erfolgt die Eingabe durch direktes „Beschreiben" des Bildschirmes mit einem Griffel. Er ist mit der Steuereinheit verbunden und in den meisten Fällen mit einer Fotodiode ausgerüstet. Überstreicht man mit dem



(Werkfoto Siemens)

Abb. 2.67 — Datensichtstation mit dazugehörender Tastatur

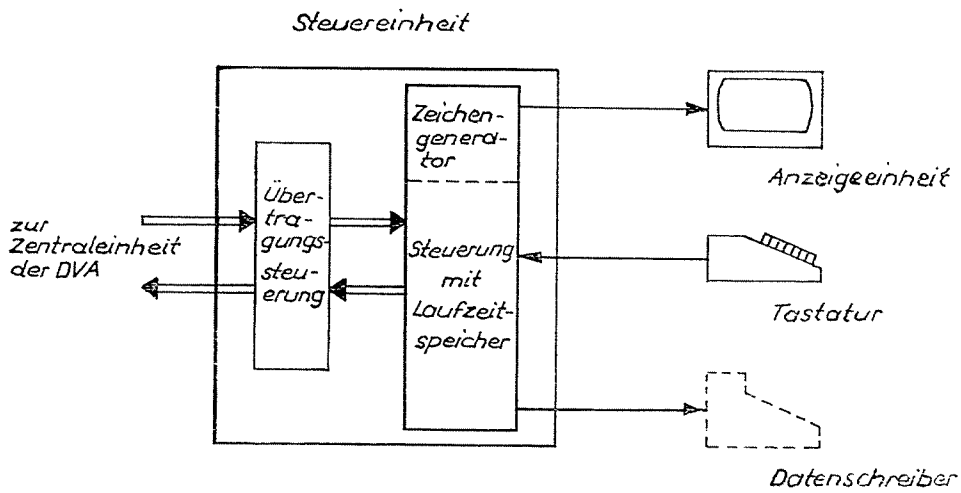


Abb. 2.68 — Blockschaltbild einer Datensichtstation

Griffel den Bildschirm, so wird die Fotodiode entsprechend den Hell-Dunkelwerten erregt. Die Diode sendet nun ihrerseits analoge elektrische Signale an die Steuereinheit. Die eintreffenden Impulse werden ausgewertet und die jeweilige Position des Lichtgriffels auf dem Schirm ermittelt. Damit kann der Lichtgriffel dazu benutzt werden, bestimmte Operationen auszuführen, z.B. können aus einer Fülle von angezeigten Daten ganz bestimmte Gruppen gekennzeichnet und anschließend getrennt zur Anzeige gebracht werden.

Die Datensichtstation ist wie kein anderes Dialoggerät geeignet, über beliebige Entfernungen hinweg mit einer DVA zu korrespondieren. Wegen ihrer einfachen Handhabung und ihrer Eignung für den Einsatz in den unterschiedlichsten Anwendungsgebieten wird sie in Zukunft immer mehr an Bedeutung gewinnen.

2.4.4.4. Speichergeräte

Als Speicher im weitesten Sinn bezeichnet man alle vorkommenden Datenträger. Sie sind befähigt, Informationen zeitlich unbegrenzt aufzubewahren und nach Bedarf wieder zur Verfügung zu stellen. Je nach ihrer Funktion innerhalb einer DVA unterscheidet man zwischen Intern- oder Primär- und Extern- oder Sekundärspeichern.

Die **Internspeicher** sind in der Zentraleinheit als Arbeitsspeicher oder Register eingesetzt; auf ihre Funktion soll hier nicht näher eingegangen werden. Die **Externspeicher** gehören zur Peripherie einer DVA. Mit ihrer Hilfe wird die Zentraleinheit entlastet. Sie dienen vor allem zur Aufnahme großer Datenmengen, die auf Abruf bereitgehalten werden. Selten benötigte Daten und umfangreiche Dateien werden nicht im Arbeitsspeicher sondern in den billigeren Externspeichern aufbewahrt. Externspeicher können ganze Programme oder Teile von ihnen, die nicht mehr unmittelbar benötigt werden, zwischenzeitlich abspeichern. Ebenso werden besonders umfangreiche, für einen Programmablauf in der Zentraleinheit benötigte Datengruppen auf dem als Zwischenspeicher eingesetzten Externspeicher bereitgehalten. Darüber hinaus wird die gesamte Software (Betriebssystem mit seinen Organisations-, Dienst-, Test- und Übersetzungsprogrammen) in einem Zwischenspeicher gelagert.

Die bedeutendsten Externspeicher arbeiten nach dem Prinzip des Magnetschichtspeichers. Hierbei werden die zu speichernden Daten auf ein magnetisierbares Material, das in den meisten Fällen aus einer dünnen Eisenoxidschicht besteht, aufgetragen. Als Träger werden Materialien verwendet, die den unterschiedlichsten mechanischen Beanspruchungen gewachsen sind. Der Datenträger wird an den Schreib- bzw. Leseköpfen vorbeigeführt, wobei die Information eingeschrieben oder ausgelesen wird. Beim Schreibvorgang erzeugt der Informationsstrom, in Abhängigkeit der zu speichernden Daten, in dem Schreibkopf ein magnetisches Feld, das seinerseits in der magnetisierbaren Schicht des darunter vorbeigleitenden Datenträgers kleine Magnetzellen aufbaut. Beeinflußt durch den Informationsstrom erhalten die einzelnen Magnetzellen unterschiedliche Polarität. Daraus ergibt sich, daß die Magnetzellen bei entsprechender Vereinbarung entweder die Information "0" oder "1" enthalten Beim Lesevorgang tastet der Lesekopf den darunter vorbeilaufenden Datenträger ab. Die magnetischen Kraftfelder der einzelnen Magnetzellen induzieren nun ihrerseits im Lesekopf Spannungsimpulse wechselnder Richtung, die nach entsprechender Umsetzung den gespeicherten Informationsinhalt wiedergeben.

Das gebräuchlichste Aufzeichnungsverfahren ist unter dem Namen **Wechselschrift** oder auch NRZ- bzw. non-return-to-zero-Schrift bekannt. Hierbei wird jedes mit dem Wert „1" zu übertragende Bit durch einen Magnetisierungswechsel gekennzeichnet. Beim Eintreffen einer „Ø" wird die vorhandene Polarität beibehalten. Weitere Aufzeichnungsverfahren sind die Wechseltakt- und die Richtungstaktschrift.

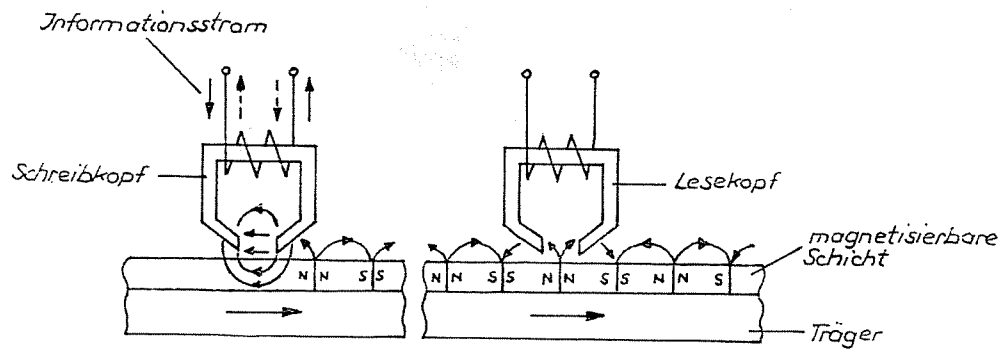


Abb. 2.69 — Grundsätzliche Darstellung des Schreib- bzw. Lesevorganges eines Magnetschichtspeichers

Die einzelnen Speicher werden nach der Art ihres Zugriffs in solche mit direktem — **Direktzugriffsspeicher** — und in solche mit seriellem Zugriff eingeteilt. So verfügen alle adressierbaren Speicher (z.B. Arbeitsspeicher) über einen direkten Zugriff. Zu den Direktzugriffsspeichern zählen

a) der Magnetplattenspeicher,
b) der Magnettrommelspeicher,
c) der Magnetkartenspeicher und
d) der Magnetstreifenspeicher.

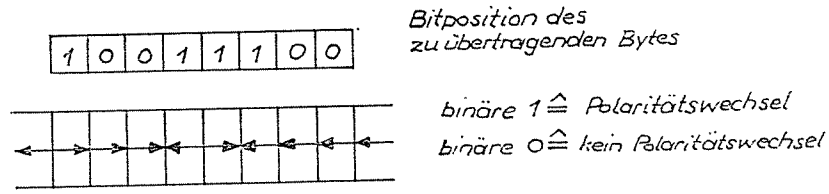
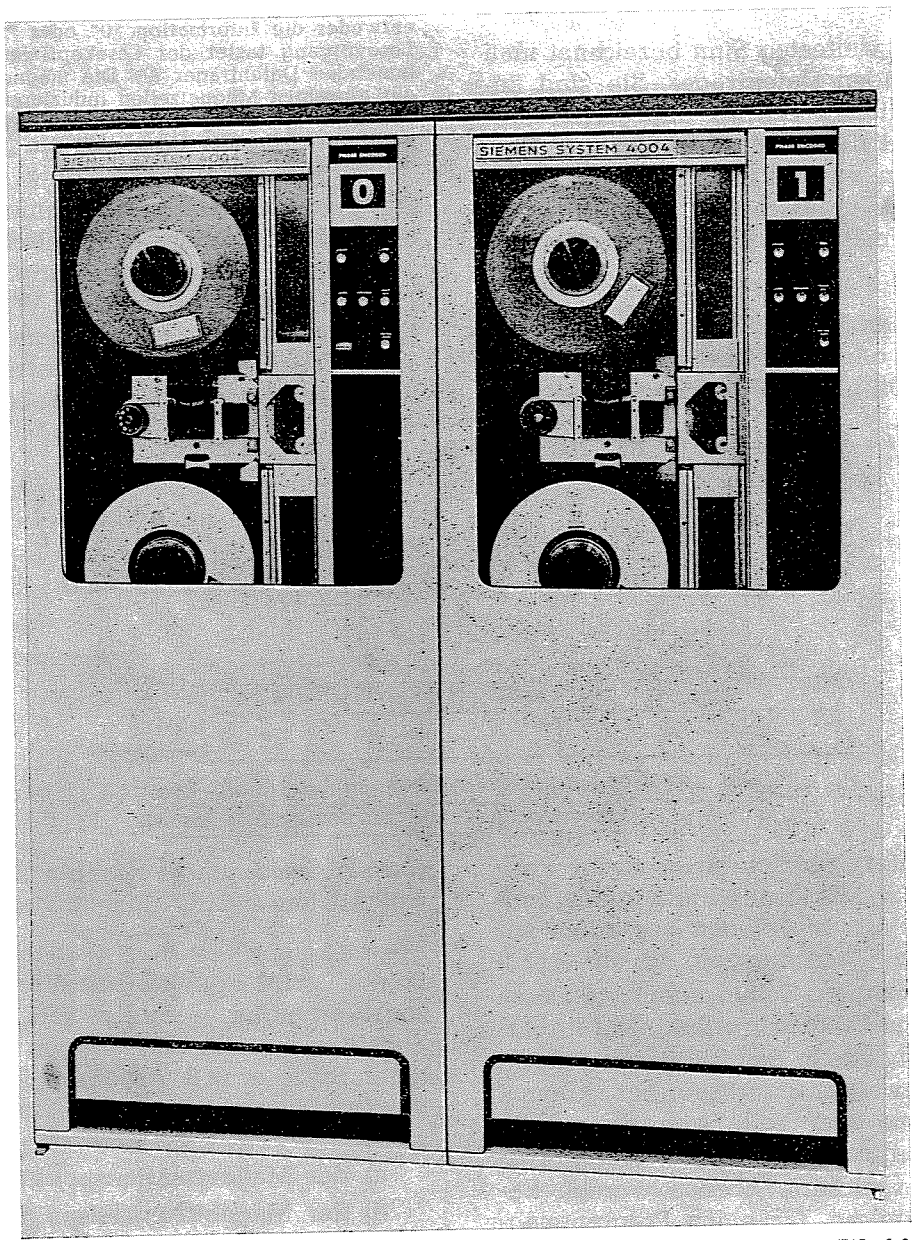


Abb. 2.70 — Prinzip der Wechselschrift-Aufzeichnung

Sie ermöglichen den direkten Zugriff, unabhängig von der Reihenfolge der Datenaufzeichnung, da eine gezielte Auswahl der entsprechenden Spur oder des Satzes getroffen werden kann. Dieses Verfahren ist unter dem Namen wahlfreier Zugriff oder ran-dom-access bekannt. **Im Gegensatz dazu wird beim Magnetbandspeicher ein serielles bzw. sequentielles Zugriffsverfahren angewandt.** Darunter ist zu verstehen, daß die gespeicherten Daten nur in einer vorgegebenen Reihenfolge verarbeitet werden können, d.h., der Zugriff erfolgt in der Reihenfolge der Datenaufzeichnung. Weitere bestimmende Merkmale für den Einsatz eines Speichers sind

a) seine Kapazität,
b) sein Preis pro Bit und
c) seine Zugriffszeit.



(Werkfoto Siemens)

Abb. 2.71 — Magnetbandzwillingsgerät

Magnetbandspeicher

Da es sich bei dem Magnetbandspeicher um ein **Speichergerät mit seriellem Zugriff** handelt, das keinen schnellen Zugriff zu Daten innerhalb eines Bandes erlaubt, wird der Magnetbandspeicher vorzugsweise dort verwendet, wo fortlaufend geordnete Datenmengen verarbeitet werden sollen. Er besteht im wesentlichen aus

dem Magnetband,

der Magnetbandeinheit und

der Magnetbandgerätesteuerung.

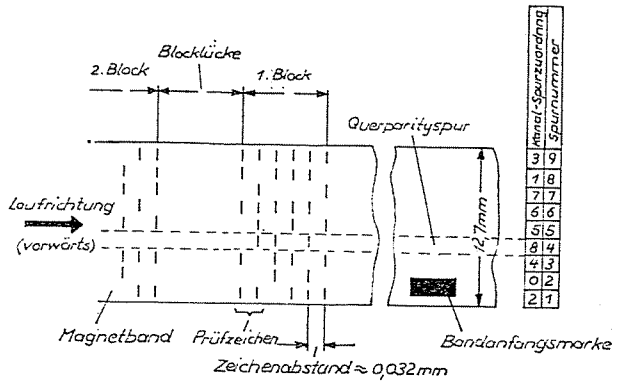


Abb. 2.72 — Prinzipielle Darstellung der Datenaufzeichnung bei einem Magnetband mit 9 Spuren

Als Datenträger wird ein Magnetband verwendet, das einem Tonband für Musikaufzeichnungen ähnelt. Es besteht aus einer Kunststoffolie (12,7 mm breit und 0,05 mm dick), das einseitig mit einer magnetisierbaren Schicht versehen ist. Auf eine genormte Spule von 27 cm Durchmesser können etwa 730 m Magnetband aufgewickelt werden. Man unterscheidet, je nach verwendetem Gerät, Bänder mit 7 oder 9 parallelen Spuren. Am häufigsten werden Bänder mit 9 Spuren verwendet, bei denen die einzelnen Bits eines Zeichens nebeneinander angeordnet sind und mit 8 Informationsbits und einem Paritybit eine Informationseinheit bilden. Beim Schreiben oder Lesen eines jeden Zeichens wird eine Paritätskontrolle durchgeführt, wobei die „Einsen" eines Zeichens summiert und entweder zu einer geraden oder ungeraden Quersumme ergänzt werden. Die Aufzeichnung eines Zeichens erfolgt gleichzeitig, wobei die einzelnen Bits parallel, also quer zur Bewegungsrichtung, aufgezeichnet werden. Die Schreibdichte beträgt bei 9-Spur-Geräten 320 Zeichen pro cm.

Auf dem Magnetband stehen die Zeichen, entgegen der EBCDI-Code-Darstellung, in der angegebenen Kanal-Spurzuordnung.

Von den 730 m Bandlänge stehen für die eigentliche Datenaufzeichnung nur etwa 710 m zur Verfügung; 20 m werden für das Einlegen des Bandes und die Pufferschleifen benötigt. Damit die Magnetbandeinheit den beschriebenen Bereich eines Bandes erkennen kann, befinden sich eine Bandanfangs- und eine Bandendemarke auf jedem Magnetband. Um es optimal auszunutzen, wird die Aufzeichnung in Blöcken von 2000 bis 4000 Zeichen angestrebt; Blöcke mit weniger als 12 Zeichen sind untersagt, da es zu Mißdeutungen im Fehlerfall kommen kann. Je nach Bandausnutzung bewegt sich die darauf unterzubringende Datenmenge zwischen 5 und $22 \cdot 10^6$ Zeichen. Die Übertragungsgeschwindigkeit eines 9-Spur-Gerätes hängt von der verwendeten Bandgeschwindigkeit ab, die 0,95, 1,9 oder 3,8 m pro Sekunde betragen kann. Daraus ergeben sich die Übertragungsraten von 30 000, 60 000 oder 120 000 Zeichen pro Sekunde.

Die Magnetbandeinheit enthält einen Transportmechanismus, der eine exakte Führung und Bewegung des Bandes gewährleistet. Jede Spule wird von einem eigenen Spezialmotor angetrieben; dadurch werden Bandgeschwindigkeiten bis zu mehreren Metern pro Sekunde ermöglicht. Es ist hierbei unbedingt erforderlich, daß das Band mit konstanter Geschwindigkeit am Magnetkopfsystem vorbeiläuft. Die Arbeitsweise des Magnetbandgerätes erfolgt nach dem Start-Stop-Prinzip, wobei das Band nach jedem Schreib- bzw. Lesevorgang gestopt und danach wieder in Bewegung gesetzt wird. Die dadurch entstehenden Blocklücken müssen möglichst klein gehalten werden. Unter Verwendung eines Spezialkupplungssystems in Verbindung mit einem unabhängigen Bandsteuermechanismus werden kleine Start-Stop-Zeiten erreicht. Der Nachteil, der sich ergibt, weil die Bandspulen verhältnismäßig träge reagieren, wird durch zwei Pufferschleifen ausgeglichen. Ein Steuermechanismus tastet jeweils die Größe der Pufferschleifen ab, wertet die ermittelten Meßdaten aus und regelt so die Bewegung der Spulen.

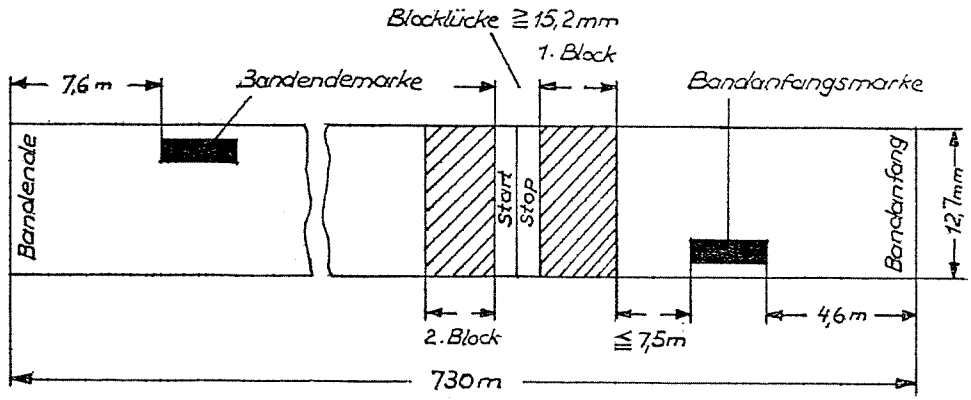


Abb. 2.73 — Struktureller Aufbau eines Magnetbandes

Das Magnetkopfsystem ist ebenfalls Bestandteil der Magnetbandeinheit. Es besteht aus getrennten Lese-, Schreib- und Löschköpfen und ist in der Lage, alle parallellaufenden Spuren gleichzeitig abzuarbeiten. Die Köpfe sind so angeordnet, daß die gespeicherten Daten nach dem Schrei-

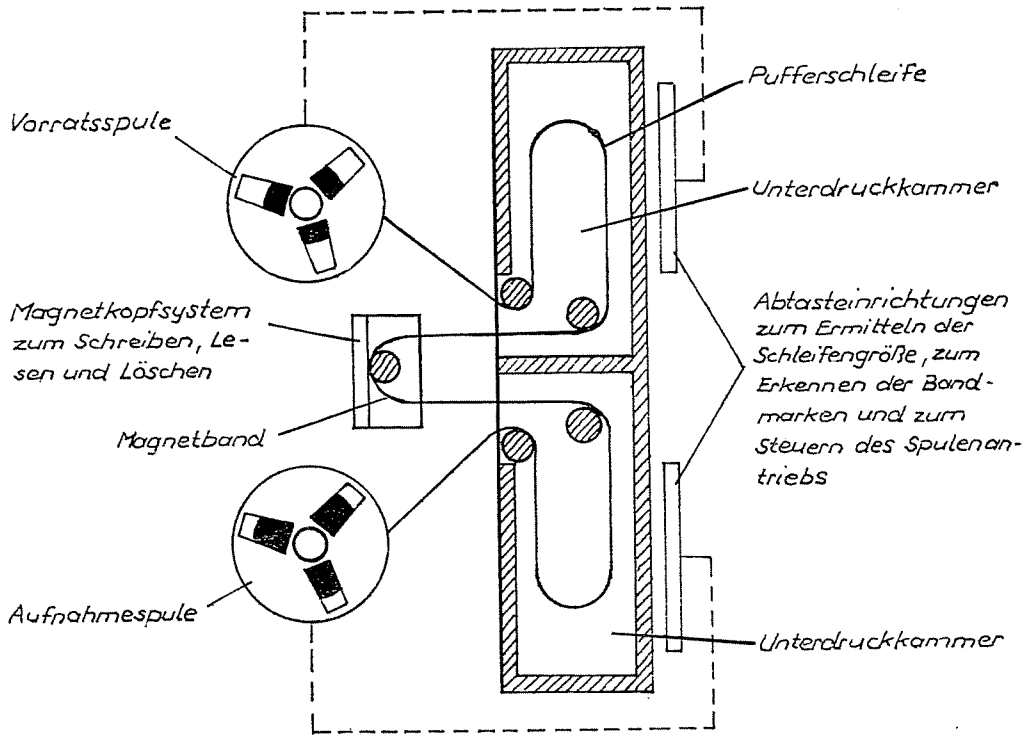


Abb. 2.74 — Schematische Darstellung des Bandantriebs

ben gelesen und somit kontrolliert werden können. Der auf einem beschriebenen Band stehende Text muß beim Überschreiben vorher gelöscht werden. Zu diesem Zweck befindet sich vor dem Schreibkopf ein Löschkopf, der das Löschen des Bandes übernimmt.

Die Magnetbandgerätesteuerung nimmt die von der Zentraleinheit eintreffenden Daten auf und setzt sie in den für das Magnetbandgerät erforderlichen Code um, wobei sie gleichzeitig den gesamten Befehlsablauf überwacht und kontrolliert. Sie führt Paritätskontrollen während des Schreib- und Lesevorganges durch und überträgt beim Erkennen eines Fehlers eine Fehlermeldung an die Zentraleinheit. Die Regelung des Bandlaufes und das Überwachen der Bandmarken ist ebenfalls Aufgabe der Magnetbandgerätesteuerung. Um den hohen Aufwand in Grenzen zu halten, ist eine derartige Gerätesteuerung für den Anschluß mehrerer Magnetbandspeicher konstruiert. Ist ein Magnetbandgerät mit zwei Laufwerken ausgestattet, die gemeinsam in einem Gehäuse untergebracht sind, spricht man von einem Magnetbandzwillingsgerät (vgl. hierzu Abb. 2.71).

Magnetplattenspeicher

Der Magnetplattenspeicher gehört wie die Bandgeräte zu den Magnetschichtspeichern. Die Tendenz der letzten Jahre geht dahin, als Externspeicher vorzugsweise den Magnetplattenspeicher einzusetzen. Im Gegensatz zu den Nachteilen des Magnetbandes, das durch die ständige direkte Berührung mit den Schreib-Leseköpfen einem starken Verschleiß unterworfen ist, unterliegt die Magnetplatte keiner mechanischen Beanspruchung. Der Magnetplattenspeicher besteht im wesentlichen aus

(Werkfoto Siemens)

Abb. 2.75 — Magnetplattenspeicher

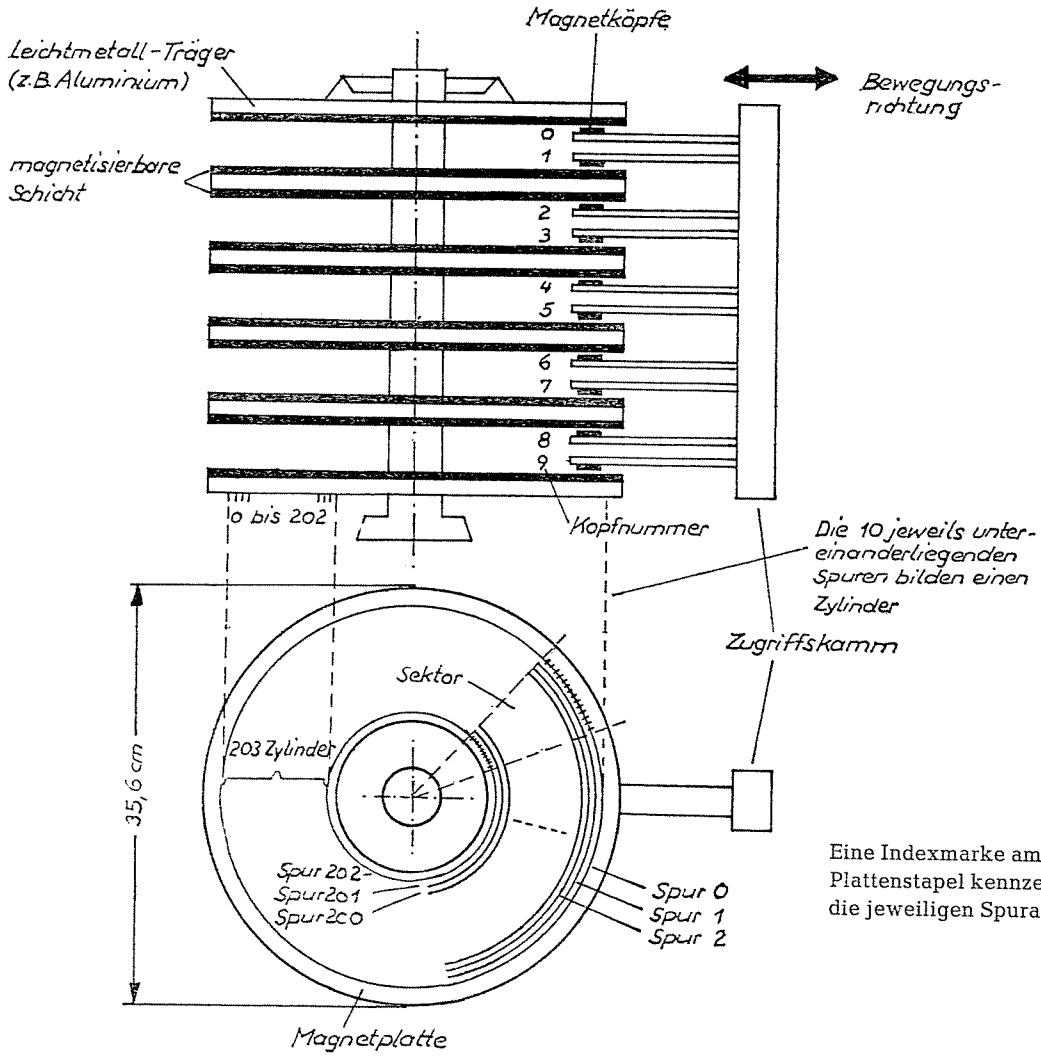
a) dem Plattenstapel,

b) der Positionierungseinrichtung mit den Magnetköpfen und

c) der elektronischen Steuerung.

Als Datenträger wird eine, auf Ober- und Unterseite mit einer magnetisierbaren Schicht versehene Leichtmetallplatte von etwa 36 cm Durchmesser verwendet. In den meisten Fällen werden die einzelnen Platten zu Plattenstapeln zusammengefaßt, wobei der Stapel, je nach verwendetem Gerät, waagerecht oder senkrecht angeordnet sein kann. Er hat in seiner gebräuchlichsten Form ein Gewicht von etwa 4,5 kg und läßt sich leicht auswechseln. An einem aus 6 Platten bestehenden Plattenstapel soll die Funktionsweise erläutert werden.

Es werden heute bereits Magnetplattenstapel mit einem Speichervolumen von mehr als $50 \cdot 10^6$ Bytes eingesetzt. Der Plattenstapel rotiert mit einer gleichbleibenden Geschwindigkeit von 2400 Umdrehungen in der Minute. Da die Radien der auf einer Platte befindlichen Kreise (Spuren) unterschiedlich sind, die Übertragungsrate aber zu jedem Zeitpunkt konstant sein muß, kann die Dichte der aufgezeichneten Daten innerhalb der einzelnen Spuren nicht gleichmäßig sein. Die Aufzeichnungsdichte verengt sich mit abnehmendem Radius. Während die Bits eines Zeichens beim Magnetband in 7 bzw. 9 parallelen



Eine Indexmarke am Plattenstapel kennzeichnet die jeweiligen Spuranfänge.

Abb. 2.76 — Prinzipielle Darstellung des Magnetplattenspeichers

Die jeweils äußeren Plattenseiten eines Stapels werden, wegen zu befürchtender Beschädigungen, nicht als Speichermedium verwendet. Damit ergeben sich bei einem Stapel von 6 Platten 10 nutzbare Plattenseiten. Die Oberflächen der einzelnen Magnetplatten sind in konzentrische Kreise — Spuren genannt — eingeteilt. Auf jeder der Platten befinden sich 203 solcher Spuren. Berücksichtigt man, daß die Spuren 200 bis 202 einer jeden Plattenseite nur als Ersatzspuren zur Verfügung stehen, so verfügt dieser Plattenstapel über 2000 zur Aufnahme von Daten verwendbare Spuren. Jede einzelne Spur ist in der Lage, 3625 Bytes zu speichern; das ergibt, bei einem Block je Spur, eine Speicherkapazität von $7{,}25 \cdot 10^6$ Bytes.

Spuren gleichzeitig aufgezeichnet werden, geschieht die Aufzeichnung eines Zeichens bei der Magnetplatte innerhalb einer Spur bitseriell, d.h., die einzelnen Bits eines Zeichens werden nacheinander abgespeichert. Die Übertragung der Daten zur Zentraleinheit wird wie beim Magnetbandgerät in Blöcken vorgenommen. Für das Einschreiben und Auslesen der Daten ist jeder Plattenseite ein Magnetkopf zugeordnet,

der im Gegensatz zum Magnetbandgerät, wo für den Schreib- und Lesevorgang je ein Magnetkopf pro Spur vorhanden ist, sowohl zum Lesen als auch zum Schreiben benutzt wird. Er ist meist als flache Scheibe ausgebildet und an einem Arm befestigt. Wie schon erwähnt, unterliegt die Magnetplatte keiner mechanischen Beanspruchung. Der Magnetkopf wird von einem Luftpolster (etwa 7,5 μ) getragen, das sich zwischen ihm und der rotierenden Platte ausbildet, womit ein gleichbleibender Abstand sichergestellt wird. Um die 203 Spuren einer Platte erreichen zu können, muß der entsprechende Magnetkopf mehr oder weniger in den Plattenstapel hineingeschoben werden. Diesen Vorgang, der programmgesteuert abläuft, nennt man **Positionierung.** Soll eine weitere Spur aufgesucht werden, so ist eine Umpositionierung vorzunehmen, die immer eine Verlängerung der Zugriffszeit mit sich bringt. Um möglichst kleine Zugriffszeiten zu erzielen, wird beim Plattenspeicher der sogenannte **„Kammzugriff"** angewandt. Hierbei werden sämtliche Magnetkopfarme starr miteinander verbunden, die damit einen Kamm bilden. Somit können alle Magnetköpfe gleichzeitig zwischen den Platten bewegt werden. Die untereinanderliegenden Spuren aller Plattenseiten werden als Zylinder aufgefaßt, wobei sich alle Magnetköpfe immer über den gleichen Spuren, d.h. auf dem gleichen Zylinder befinden. Damit ist man in der Lage, durch Ansprechen der einzelnen Magnetköpfe, ohne sie bewegen zu müssen, innerhalb des Zylinders eine von 10 Spuren anzusprechen und zu den auf ihr befindlichen Daten zugreifen zu können. Daraus wird ersichtlich, daß die Datenmengen so organisiert werden müssen, daß sie innerhalb eines Zylinders abgespeichert werden können; die Zugriffszeit wird damit auf ein Minimum herabgesetzt. Dieses Verfahren, das eine verhältnismäßig einfache Adressierung der Daten gestattet, sowie der relativ kurze Zugriff machen den Magnetplattenspeicher zu einem der am häufigsten verwendeten Direktzugriffsspeicher. Allerdings sei erwähnt, daß von einem direkten Zugriff nur bedingt gesprochen werden kann. So erfolgt ein Zugriff in zwei Stufen, wobei nach Auswahl des entsprechenden Magnetkopfes dieser positioniert, d.h. zu einer bestimmten Spur bewegt werden muß. Es kann jedoch erst geschrieben oder gelesen werden, wenn sich die gesuchte Stelle, innerhalb der aufgesuchten Spur, unter dem Magnetkopf befindet. Diese Such- bzw. Positionierungsvorgänge laufen alle programmgesteuert ab. Die durchschnittliche Zugriffszeit eines Magnetplattenspeichers beläuft sich auf etwa 87,5 ms und setzt sich zusammen aus:

a) einem Zeitanteil, der benötigt wird, wenn ein Wechsel von Spur 1 auf Spur 200 vorgenommen werden muß; er beträgt im Mittel 75 ms und

b) einer Wartezeitspanne von etwa 12,5 ms, die durch eine volle Umdrehung der Platte entstehen kann.

Seine Schreib- bzw. Lesegeschwindigkeit (Datenfluß) beträgt 156 000 Bytes pro Sekunde. Dem Magnetplattenspeicher wird eine Gerätesteuerung vorgeschaltet, deren Aufgaben im Abschn. 2.4.3.1. näher beschrieben wurden.

Magnettrommelspeicher

Auch der Magnettrommelspeicher gehört zu den Magnetschichtspeichern, von denen er allerdings, wegen seiner einfachen mechanischen Konstruktion, zu den weniger aufwendigen Speichergeräten zählt. Der Kernspeicher hat den Magnettrommelspeicher aus der Zentraleinheit verdrängt; er wird heute ausschließlich als Externspeicher verwendet. Stellvertretend für eine Vielzahl von Trommelspeichern sollen am Beispiel des hier behandelten Magnettrommelspeichers die funktionellen Zusammenhänge und die Arbeitsweise derartiger Speicher erläutert werden. Seine bedeutendsten Bestandteile sind

a) die Magnettrommel,

b) die Magnetkopfeinheiten und

c) die elektronische Steuerung.

Als Datenträger wird eine Leichtmetalltrommel verwendet, auf deren Oberfläche eine dünne magnetisierbare Schicht — meist ein Überzug einer Kobalt-Nickel-Legierung — aufgebracht ist. Die Trommel dreht sich mit hoher konstanter Geschwindigkeit und kann sowohl senkrecht als auch waagerecht angeordnet sein. Sie ist, im Gegensatz zum Plattenstapel des Magnetplattenspeichers, nicht austauschbar. Die Oberfläche der Trommel ist in Spuren, deren Abstand voneinander etwa 0,25 mm beträgt, eingeteilt, auf denen die Daten aufgezeichnet werden. Je nach Modell ist für jede einzelne oder für mehrere Spuren gemeinsam ein Schreib-Lesekopf vorhanden, wobei diejenigen Spuren, die sich bei einer Einstellung unterhalb

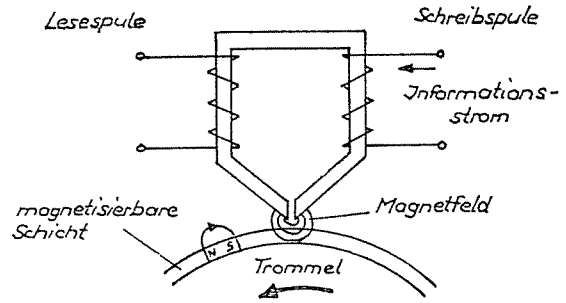


Abb. 2.77 — Grundsätzliche Darstellung des Schreib- bzw. Lesevorganges eines Magnettrommelspeichers

der Schreib-Leseköpfe befinden, in Gruppen zusammengefaßt (in diesem Falle 8 Spuren) als Zylinder bezeichnet werden. Wie die Oberfläche der Magnetplatte ist auch die magnetisierbare Schicht der Magnettrommel keiner mechanischen Beanspruchung ausgesetzt. Auch hier werden die Magnetköpfe von einem durch die Rotationsgeschwindigkeit erzeugten Luftpolster getragen. Das Schreiben bzw. Lesen von Informationen geschieht nach dem Prinzip aller Magnetschichtspeicher, nur wird hier ein kombinierter Magnet-Schreib-Lesekopf verwendet.

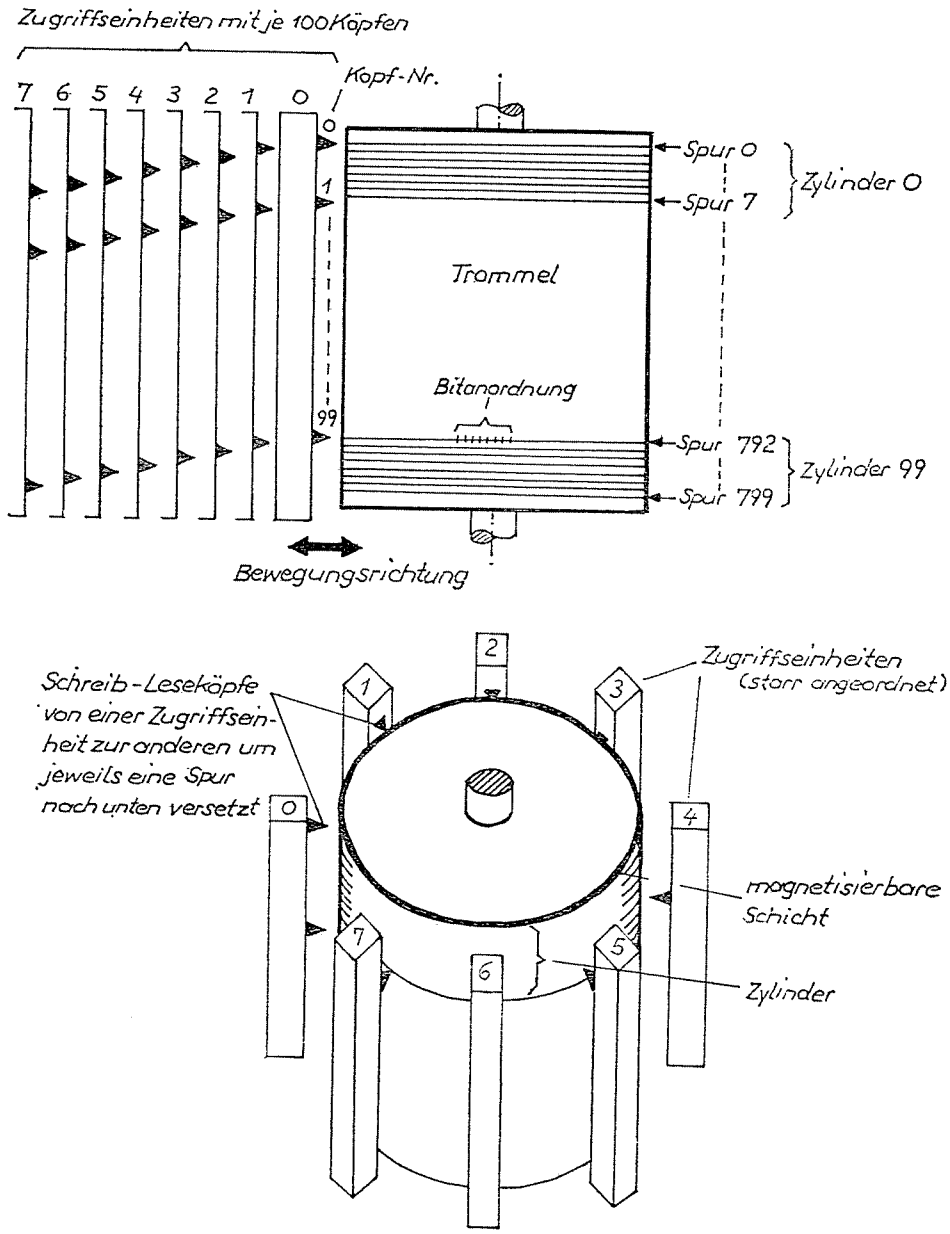


Abb. 2.78 — Prinzip des Magnettrommelspeichers

Beeinflußt durch die Richtung des Informationsstromes wird ein sich ständig wechselndes magnetisches Feld erzeugt, das innerhalb der entsprechenden Spur auf der Trommeloberfläche winzige Magnetzellen aufbaut. Die Richtung des Informationsstromes ist entscheidend für die sich bildende Polarität der Magnetzellen. Dementsprechend erhält jede Zelle den Informationsinhalt "1" oder "0". Ähnlich verläuft der Vorgang beim Lesen der Daten. Hierbei wird in der Lesespule, hervorgerufen durch das magnetische Feld der unter dem Schreib-Lesekopf vorbeilaufenden Magnetzellen, ein Strom induziert, der entsprechend seiner Richtung den Informationsinhalt der Speicherzellen wiedergibt. Der Lesevorgang verändert den Zustand der einzelnen Magnetzellen nicht, so daß die Daten erhalten bleiben. Das Löschen einer einmal eingeschriebenen Information erfolgt erst dann, wenn sie durch eine erneute Eingabe überschrieben wird.

Die Trommeloberfläche des Gerätes nach Abb. 2.78 ist in 800 Spuren eingeteilt. Um die Trommel verteilt sind festmontiert 8 Magnetkopfträger angeordnet, die als Zugriffseinheiten bezeichnet werden. Bedingt durch den sehr kleinen Spurabstand ist es nicht möglich, alle Schreib-Leseköpfe auf einer Zugriffseinheit unterzubringen. Demzufolge wurden jeder Zugriffseinheit 100 Schreib-Leseköpfe zugeordnet, die jeweils um 8 Spuren versetzt angeordnet sind. Unter Anwendung des Wendeltreppenprinzips sind 8mal 100 Schreib-Leseköpfe der 8 Zugriffseinheiten in der Lage, jeweils die 8 Spuren einer der 100 Zylinder anzusprechen, wobei die Spuren 1 bis 8 des 1. Zylinders von den jeweils ersten Schreib-Leseköpfen aller 8 Zugriffseinheiten erreicht werden. Wie beim Magnetplattenspeicher geschieht die Abspeicherung der einzelnen Zeichen bitseriell, wobei die Auswahl einer Spur durch die Adressierung eines bestimmten Schreib-Lesekopfes vorgenommen und mit Hilfe eines Programms von der Zentraleinheit gesteuert wird. Auch hier werden die Daten blockweise an den Arbeitsspeicher abgegeben.

Die Kapazität eines Magnettrommelspeichers hängt weitgehend vom Bau der Trommel selbst ab und zwar stellt sie einen Kompromiß dar zwischen der Trommelgröße, der zulässigen Umlaufgeschwindigkeit und der daraus resultierenden Zugriffszeit. Jede der vorhandenen 800 Spuren ist in der Lage, 5160 Bytes aufzunehmen. Damit ergibt sich ein Speichervolumen von insgesamt $4{,}13 \cdot 10^6$ Bytes. Da für jede Spur ein Schreib-Lesekopf vorgesehen ist und diese starr zugeordnet sind, entfällt die Positionierungszeit, die beim Magnetplattenspeicher die Zugriffszeit negativ beeinflußt. Für die Ermittlung der Zugriffszeit des Magnettrommelspeichers fällt demnach nur noch die Umdrehungsgeschwindigkeit der Trommel maßgebend ins Gewicht. Bei 3000 Umdrehungen in der Minute ergibt sich eine mittlere Zugriffszeit von etwa 10 ms, die sich aus einem Zeitanteil für die Adressierung des entsprechenden Kopfes und einer mittleren Wartezeit auf den Spuranfang zusammensetzt. Will man die Zugriffszeit weiter herabsetzen, so läßt sich dies durch Steigerung der Rotationsgeschwindigkeit erreichen. Die dabei auftretenden Fliehkräfte zwingen jedoch den Konstrukteur, die Trommel entsprechend kleiner zu gestalten. Andererseits ist es möglich, bei Vernachlässigung der Zugriffszeit und unter Verwendung kleiner Umdrehungsgeschwindigkeiten den Trommeldurchmesser derart zu vergrößern, daß sehr hohe Speicherkapazitäten erzielt werden. Die Übertragungsrate des Magnettrommelspeichers beträgt etwa 275 000 Bytes pro Sekunde. Er wird wegen seines **direkten Zugriffs** und der **kleinen Zugriffszeit vorzugsweise** dort verwendet, wo Daten ständig zur Verfügung stehen müssen und einem schnellen Zugriff unterliegen.

Magnetkartenspeicher

Wie alle bisher beschriebenen Großspeicher gehört auch der Magnetkartenspeicher zu den Magnetschichtspeichern. Er arbeitet in wahlfreiem Zugriff und stellt im Prinzip einen Trommelspeicher dar, dessen Trommelmantel austauschbar ist. Er besteht im wesentlichen aus

a) den Magnetkartenmagazinen,

b) der Transporteinrichtung,

c) der Schreib-Leseeinrichtung und

d) der elektronischen Steuerung.

Als Speichermedium werden Magnetkarten aus flexiblem Kunststoffmaterial verwendet. Sie sind auf einer Seite mit einer ferromagnetischen Schicht belegt, auf der die Daten gespeichert werden. Die Magnetkarten, mit einer Länge von 40,6 cm und einer Breite von 11,4 cm, werden einseitig beschrieben und weisen am Rand individuelle Kerbungen auf, die der automatischen Auswahl der einzelnen Karten aus den Magazinen dienen. Die Kerbungen jeder einzelnen Karte sind in ihrer Zahl und Form verschieden, so daß dadurch die Auswahl einer bestimmten Karte möglich wird.

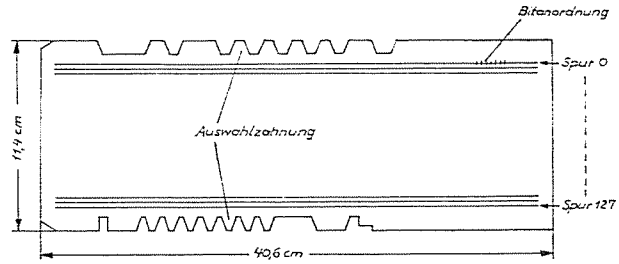


Abb. 2.80 — Magnetkarte

Die Magnetkarte ist mit 128 Spuren versehen, die getrennt angesprochen werden. Die Datenaufzeichnung erfolgt bitseriell, d.h., die abzuspeichernden Zeichen werden nacheinander auf einer Spur Bit für Bit aufgezeichnet. Der Magnetkartenspeicher faßt insgesamt 2048 Karten, die in 8 Magazinen mit jeweils 256 Karten untergebracht sind. Die einzelnen Magazine sind leicht auszuwechseln und untereinander austauschbar. Geht man von der Annahme aus, daß jede Spur nur aus einem Block besteht und bringt man die erforderliche Organisationsinformationen in Abzug, so können je Spur 2048 Bytes untergebracht werden. Damit ergibt sich — bei 2048 Magnetkarten mit je 128 Spuren und einem Datenfassungsvermögen von 2048 Bytes je Spur — eine nutzbare Speicherkapazität von 536 870 912 Bytes. Die 128 Spuren einer Karte werden jeweils in Gruppen zu 8 Spuren in Zylinder eingeteilt, so daß jede Karte 16 Zylinder enthält.

Die folgende Zusammenstellung soll einen Überblick über die Kapazität der einzelnen Speichereinheiten eines Magnetkartenspeichers geben.

Speichereinheit	Kapazität in Bytes
Spuren pro Karte	2048
Zylinder (8 Spuren)	$8 \cdot 2048 = 16384$
Karte (16 Zylinder)	$16 \cdot 16384 = 262144$
Magazin (256 Karten)	$2 \cdot 56 \cdot 262144 = 67108864$
Magnetkartenspeicher (8 Magazine)	$8 \cdot 67108864 = 536870912$

Die Magnetkarten haben nur eine begrenzte Lebensdauer, so werden allgemein 10 000 Zugriffe bei 20 Trommelumdrehungen je Zugriff als durchschnittlicher Grenzwert

festgelegt. Ein extra dafür vorgesehener Schreib-Lesekopf dient zur Kartenzugriffszählung, wobei eine Meldung abgegeben wird, sobald der Grenzwert erreicht ist. Die Karte wird dann automatisch ausgesondert. Die Transporteinrichtung befördert die ausgewählte Magnetkarte vom Kartenmagazin über einen Zuführungskanal zur Schreib-Leseeinrichtung und sorgt dafür, daß die Karte nach erfolgtem Datenaustausch in das Magazin zurückgebracht wird. Dabei ist es nicht notwendig, die Karte einzusortieren; es genügt, wenn sie an letzter Stelle im Magazin abgelegt wird.

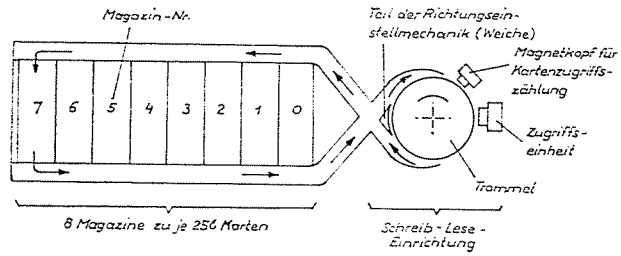


Abb. 2.81 — Schematische Darstellung des Magnetkartenspeichers

Der Datenaustausch wird von 8 Schreib-Leseköpfen vorgenommen, die zu einer Zugriffseinheit zusammengefaßt sind. Die eigentliche Schreib-Leseeinrichtung besteht aus einer sich drehenden Trommel, der Zugriffseinheit und einer Richtungseinstellmechanik für die Steuerung des Kartentransportes. Beim Lese- bzw. Schreibvorgang werden die Karten, sobald sie an der Trommel angelangt sind, von einem Haltemechanismus auf die Trommeloberfläche gepreßt und unter den Schreib-Leseköpfen vorbeigeführt. Da eine Karte in 128 Spuren eingeteilt ist und nur 8 Schreib-Lesekopfpaare vorhanden sind, muß die Zugriffseinheit in 16 verschiedene Einstellungen — entsprechend den 16 Zylindern — positioniert werden können.

Für die Spuradresse sind folgende Daten erforderlich und vom Programm anzugeben

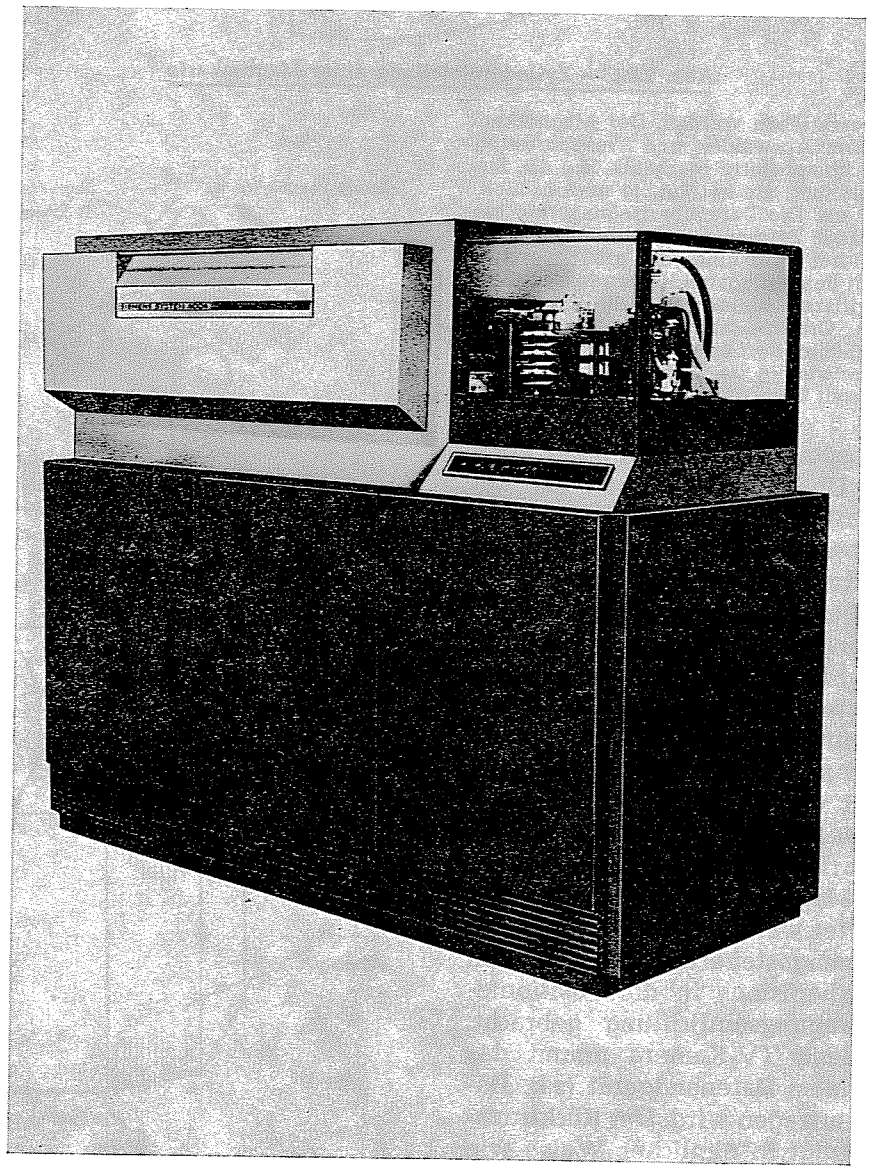
a) Nummer des Magazins,

b) Nummer der Karte innerhalb des Magazins,

c) Nummer des Zylinders und

d) Nummer des Schreib-Lesekopfes (Spurnummer).

Bis zu 8 Magnetkartenspeicher können an eine Gerätesteuerung angeschlossen werden, deren Aufgaben im Ab-



(Werkfoto Siemens)

Abb. 2.79 — Magnetkartenspeicher

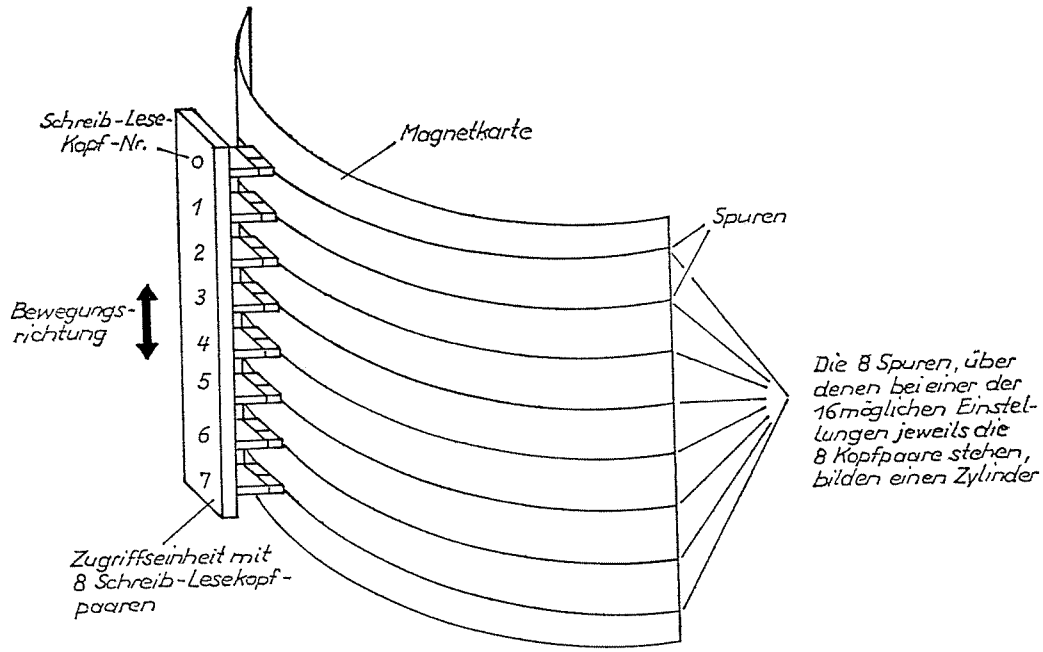


Abb. 2.82 — Zylindereinteilung einer Magnetkarte

satz 2.4.3.1. bereits beschrieben wurden. Der Magnetkartenspeicher verfügt zum Auswählen und Transportieren der Karten über eine komplizierte Mechanik, die die Zugriffszeit negativ beeinflußt; sie beträgt, je nach Modell, zwischen 470 und 530 ms. Die Datenübertragungsrate bewegt sich bei etwa 70 000 Zeichen pro Sekunde. So stellt der Magnetkartenspeicher zwar eines der größten externen Speichergeräte mit den geringsten Kosten je Speicherstelle dar, seine Zugriffszeit allerdings ist die ungünstigste der betrachteten Großspeichergeräte. Aus diesem Grunde wird er vornehmlich dort eingesetzt, wo zwar eine umfangreiche Datei notwendig ist, andererseits aber auf eine schnelle Zugriffszeit kein großer Wert gelegt wird.

Der Magnetkartenspeicher kann mit einer Positionierungs- und Übertragungseinheit als Zusatzgerät ausgestattet werden. Damit ist er in der Lage, textliche und bildliche Information zu speichern. Hierunter werden vor allem Zeichnungen, Fotografien, Tabellen und sonstige bildliche Darstellungen verstanden. Als Datenträger wird eine Magnetkarte verwendet, auf der etwa 1200 Mikrobilder, mit einer Größe von 3,5 mm x 5,0 mm, untergebracht werden können. Ein Kartenspeicher mit 8 Magazinen könnte den Informationsinhalt einer Bibliothek mit etwa 100 000 bis 200 000 Bänden aufnehmen. Durch eine Umschalteeinrichtung kann von dem bereits bekannten Magnetbetrieb auf optischen Betrieb umgeschaltet werden. Die von der Zentraleinheit programmgesteuerte Karte wird von dem Transportmechanismus in die Positionierungs- und Übertragungseinrichtung gebracht. Eine dort vorhandene TV-Kamera nimmt das Bild auf, das zu einem Datenendgerät (z.B. Datensichtstation) übertragen wird. Der Rücktransport der Karte erfolgt in ähnlicher Weise wie beim Magnetbetrieb. Auf diese Art könnten z.B. alle technischen Unterlagen eines Großbetrie-

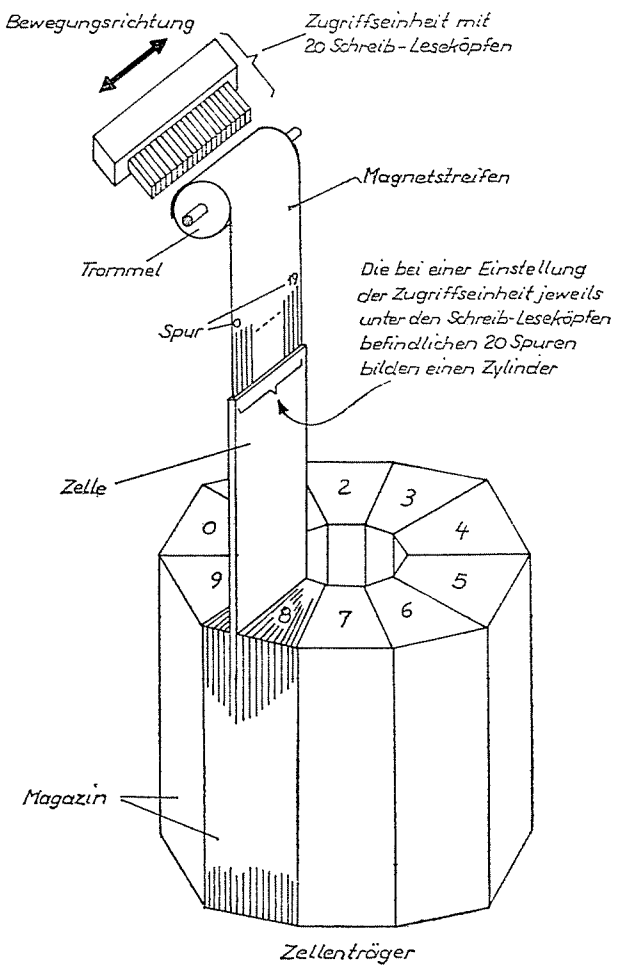


Abb. 2.83 — Schematische Darstellung eines Magnetstreifenspeichers

bes zentral abgespeichert und mittels Datensichtstationen vom jeweiligen Arbeitsplatz aus im Dialogverkehr angefordert werden.

Magnetstreifenspeicher

Als letzter Großspeicher sei noch der Magnetsireifenspeicher (vgl. hierzu Abb. 2.83) erwähnt. Er arbeitet nach dem Prinzip des Magnetkartenspeichers und ist ein **Speichergerät mit direktem Zugriff.** Der verwendete Datenträger besteht aus einer 2,5 Zoll breiten und 13 Zoll langen streifenförmigen Folie, deren eine Seite mit einer magnetisierbaren Schicht versehen ist. In einem trommelartigen Zellenträger befinden sich 10 Magazine, von denen jedes 20 Zellen enthält, wobei jeder Zelle 10 Magnetstreifen zugeordnet sind. Der Magnetstreifen ist in 100 Spuren unterteilt, die in 5 Gruppen mit je 20 Spuren zu Zylindern zusammengefaßt sind. Bei einem Fassungsvermögen von 2000 Bytes je Spur besitzt ein Streifenspeicher mit 10 Magazinen ein Speichervolumen von $400 \cdot 10^6$ Bytes. Die Datenübertragungsrate beträgt ca. 55 000 Zeichen pro Sekunde.

Durch Drehen des Zellenträgers wird das entsprechende Magazin unter eine rotierende Trommel gebracht. Die Auswahlmechanik entnimmt dem betreffenden Magazin den gewünschten Magnetstreifen und transportiert ihn auf die Trommel. Dort wird er von einer Haltevorrichtung auf die Trommeloberfläche gepreßt und unter einer Zugriffseinheit von 20 Schreib-Leseköpfen vorbeigeführt. Die Zugriffseinheit ist in achsialer Richtung zur Trommel beweglich angeordnet und kann in 5 Schreib-Lesestellungen positioniert werden. Nach erfolgtem Datenaustausch muß der Streifen, im Gegensatz zur Magnetkarte, an seinen vorherigen Platz zurückgebracht und dort eingeordnet werden.

Die Zugriffszeit setzt sich aus den einzelnen Zeitanteilen zusammen, die für die Auswahl eines Streifens und für die Positionierung der Zugriffseinheit erforderlich sind.

2.4.5. Off-line-Peripherie

Alle Geräte, die ohne Mitwirkung der Zentraleinheit einer DVA betrieben werden können, rechnet man zur Off-line-Peripherie Das wesentliche Merkmal des Off-line-Betriebes besteht darin, daß die Übertragung der Daten zwischen rechnerunabhängigen Geräten vorgenommen

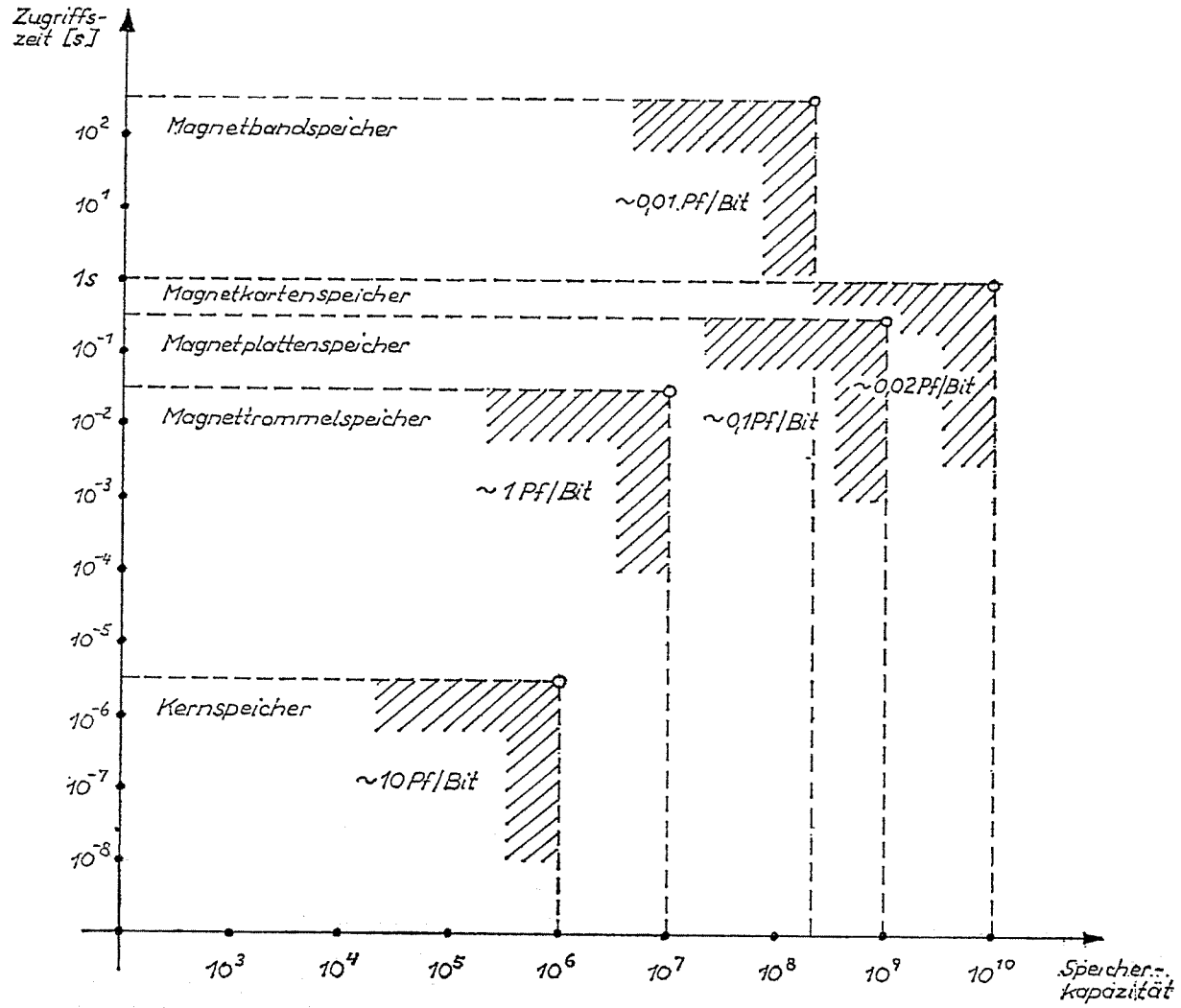


Abb. 2.84 — Überblick über Zugriffszeit, Kapazität und Preis pro Bit der gebräuchlichsten Speicher

wird. Sie werden vor allem zur Erfassung von Daten und deren Aufbereitung in eine maschinenlesbare Form sowie zur Umsetzung und Übertragung von Informationen von einem Datenträger auf einen anderen eingesetzt. Durch ihre Verwendung wird die Zentraleinheit entlastet und somit die Wirtschaftlichkeit einer DVA erhöht. Die gebräuchlichsten Geräte der Off-line-Peripherie werden nachfolgend kurz beschrieben.

2.4.5.1. Kartenlocher

Der Kartenlocher ist ein Datenerfassungsgerät, mit dessen Hilfe Informationen manuell auf Lochkarten übertragen werden können. Er kann, je nach Verwendungszweck, entweder als numerischer oder als alphanumerischer Locher eingesetzt werden. Er besteht im wesentlichen aus

a) der Tastatur,

b) der Transportmechanik und

c) der Loch- und Abfühlstation.

Die Tastatur ist in Aufbau und Bedienungsweise mit der einer Schreibmaschine vergleichbar. Werden mit einem Kartenlocher ausschließlich Zahlen gelocht, so erhält er eine reine **Zahlentastatur.** Seine Arbeitsgeschwindigkeit kann dadurch wesentlich gesteigert werden. Sollen außer Zahlen auch Buchstaben und Sonderzeichen gelocht werden, wird eine **alphanumerische Tastatur** verwendet. Bestandteil beider Tastaturarten sind eine Reihe von Funktionstasten, mit denen interne Steuervorgänge veranlaßt werden können.

Der Transportmechanismus entnimmt dem Kartenmagazin — Fassungsvermögen etwa 500 Karten — die ungelochte Karte und befördert sie zur Lochstation. Ist der Lochvorgang abgeschlossen, sorgt er für eine einwandfreie Ablage der Karte. In der Lochstation werden, veranlaßt durch das Anschlagen der Tastatur, die Daten in die entsprechenden Lochkombinationen umgesetzt und in die Karte gestanzt. Mit der Abfühlstation ist der Kartenlocher in der Lage, die Lochstellen einer bereits gelochten Karte abzutasten und auf eine zweite Karte zu duplizieren. Die Stanzgeschwindigkeit des Kartenlochers beträgt etwa 20 Spalten pro Sekunde. Um bestimmte Arbeitsvorgänge (z.B. Vorschub, Buchstaben-Ziffern-Umschaltung usw.) steuern zu können, werden — für jedes Kartenformat getrennt — Programmkarten verwendet. Die Programmkarte wird auf eine Programmtrommel gespannt, deren Rotationsgeschwindigkeit mit der Durchlaufgeschwindigkeit der Karten in der Loch- und Abfühlstation übereinstimmt, so daß vereinfacht von einem programmgesteuerten Arbeitsablauf gesprochen werden kann.

Häufig ist der Kartenlocher mit einer Zusatzeinrichtung ausgestattet, die es ermöglicht, auf dem oberen Rand der Karte in Klarschrift das in der darunter liegenden Spalte gelochte Zeichen abzubilden. Eine weitere Zusatzeinrichtung gestattet das Übertragen der Daten von einem Lochstreifen auf Lochkarten, wobei eine Lochstreifensteuerung die notwendigen Steuervorgänge übernimmt.

2.4.5.2. Kartenprüfer

Um Lochfehler weitgehend auszuschließen, werden die mit einem Kartenlocher hergestellten Lochkarten mit Hilfe eines Kartenprüfers einer Prüfung unterzogen. Er ist ähnlich wie ein Kartenlocher aufgebaut und unterscheidet sich von ihm nur dadurch, daß an Stelle der beim Kartenlocher vorhandenen Stanzstempel Abtaststifte vorhanden sind. Die Bedienungsperson tastet nochmals die bereits in der Lochkarte vorhandenen Daten ein, wobei der Kartenprüfer kontrolliert, ob die vom Orginalbeleg eingetasteten Daten mit denen in der Lochkarte übereinstimmen. Stellt der Kartenprüfer eine Unstimmigkeit fest, so leuchtet eine Fehlersignallampe auf. Die fehlerhafte Karte wird auf Veranlassung der Bedienungsperson ausgesondert. Die Prüfgeschwindigkeit beträgt etwa 25 Spalten pro Sekunde.

2.4.5.3. Streifenlocher

Der am weitesten verbreitete Streifenlocher, bekannt unter der Bezeichnung „Lochstreifenlocher", stellt ein Zusatzgerät des Fernschreibers dar. Mit Hilfe der Fernschreibtastatur werden die Daten in den Lochstreifen eingestanzt, wobei gleichzeitig ein Protokoll der eingetasteten Daten mitgeschrieben wird. Ein weiteres Zusatzgerät des Fernschreibers ist das Lochstreifenlesegerät, mit dessen Hilfe die Lochkombinationen bereits gelochter Lochstreifen abgetastet und über den Lochstreifenlocher auf einen zweiten Lochstreifen übertragen werden können. Alle dabei umgesetzten Daten werden ebenfalls protokolliert. Streifenlocher können, je nach Verwendungszweck, zusammen mit den verschiedensten Büro- und Buchungsmaschinen oder als einfacher Handlocher eingesetzt werden.

2.4.5.4. Magnetbandschreiber

Mit Hilfe des Magnetbandschreibers können Daten direkt auf ein Magnetband geschrieben werden. Der Magnetbandschreiber ist in seiner Konzeption und in seiner Arbeitsweise mit dem Kartenlocher vergleichbar. Er ist mit einem Pufferspeicher ausgerüstet, der die eingetasteten Informationen aufnimmt, sie zu einem Block zusammenstellt und diesen nach Drücken einer

Funktionstaste auf das Magnetband überträgt. Die in den Puffer eingeschriebenen Daten werden einer Paritykontrolle unterzogen, wobei als falsch erkannte Zeichen vor dem Beschreiben des Magnetbandes korrigiert werden können. Der Magnetbandschreiber kann auch zum Prüfen bereits beschriebener Bänder verwendet werden, wobei die Daten blockweise vom Puffer übernommen werden. Danach werden die von der Bedienungsperson eingetasteten Daten zeichenweise mit denen im Pufferspeicher verglichen. Nach der Korrektur eventueller Fehler wird der im Puffer stehende Block auf das Band zurückgeschrieben.

2.4.5.5. Lochschriftübersetzer

Mit Hilfe des Lochschriftübersetzers können die in Lochkombinationen auf Ausgabe-Lochkarten enthaltenen Daten in Klartext übersetzt und auf der gleichen Karte abgedruckt werden. Dadurch wird der Anwendungsbereich der Lochkarten erweitert, die damit als Karteikarten, Kontenkarten oder sonstige Belege verwendet werden können. Ist der Lochschriftübersetzer mit einer entsprechenden Zusatzeinrichtung ausgestattet, so besteht die Möglichkeit, unabhängig von den auf einer Lochkarte befindlichen Lochkombinationen zusätzliche Angaben aufzudrucken. Die zu erreichende Schreibgeschwindigkeit beläuft sich etwa auf 40 Spalten pro Sekunde.

2.4.5.6. Lochkartendoppler

Der Lochkartendoppler dient zum automatischen Vervielfältigen bereits mit Daten versehener Lochkarten. Dabei können alle oder nur ein Teil der in den Originalkarten enthaltenen Daten auf eine beliebige Anzahl von Leerkarten übertragen werden. Die Originalkarten werden in das Magazin der Abfühleinheit und die neu zu lochenden Karten in das Magazin der Stanzeinheit eingelegt. Anhand eines Vergleichers werden die Daten der Originalkarten mit denen der neu gelochten Karten auf Übereinstimmung geprüft. Eine automatische Fehleranzeige signalisiert die festgestellten Unstimmigkeiten. Für die Doppelung von zwei 80spaltigen Lochkarten benötigt der Lochkartendoppler etwa eine Sekunde.

2.4.5.7. Tabelliermaschine

Die Tabelliermaschine erfüllt in der Lochkartentechnik die Aufgaben eines Druckers. Sie wird vornehmlich zum Ausfüllen von Vordrucken und zum Auflisten von Programm- sowie Ein- und Ausgabekartenstapeln verwendet, wobei eine Prüfung auf Lochfehler erleichtert wird. Die Tabelliermaschine kann mehrere Rechenwerke besitzen, die zum Addieren oder Subtrahieren bearbeiteter Lochkarten dienen. An einer Schalttafel werden die oft recht komplizierten Arbeitsprogramme gesteckt, die für den automatischen Ablauf der verschiedenen Arbeitsvorgänge erforderlich sind.

2.4.6. Datenfernverarbeitung *)

In den vorangegangenen Kapiteln wurde bei der Betrachtung der Peripherie einer DVA davon ausgegangen, daß die zur Verarbeitung vorgesehenen Informationen an Ort und Stelle zur Verfügung stehen und daß die während der Datenverarbeitung entstehenden Ergebnisse in unmittelbarer Nähe des Rechners weiterverarbeitet oder ausgewertet werden. Ein derartiges Arbeitsverfahren ist jedoch nicht immer möglich. Häufig befinden sich die Geräte der Datenerfassung und die DVA an geographisch verschiedenen Orten. Damit erhebt sich die Frage: Wie gelangen die Daten schnell und sicher zur Verarbeitung und wie erhält man ohne Zeitverzögerungen Zugriff zu den Verarbeitungsergebnissen? Die Beantwortung dieser Frage führt zur Datenfernverarbeitung. Sie ermöglicht es, Daten vom Ort ihres Entstehens über beliebige Entfernungen hinweg zum Ort der Verarbeitung zu übertragen und umgekehrt, die Verarbeitungsergebnisse dorthin zu übermitteln, wo sie benötigt werden. Die eigentliche Datenübertragung kann entweder über Datenträger mit Hilfe geeigneter Transportmittel oder aber durch die Übertragung elektrischer Signale geschehen.

2.4.6.1. Datenfernverarbeitung im On-line-Betrieb

Werden die übertragenen Daten direkt in die Zentraleinheit einer DVA eingegeben und nach erfolgter Verarbeitung von ihr gesendet, so spricht man von einer Datenfernverarbeitung im On-line-Betrieb. Dieses Arbeitsverfahren wird immer dann angewandt, wenn für die Bearbeitung eines Vorganges nur geringe Zeit zur Verfügung steht, d.h. wenn im **Realzeit- bzw. Real-time-Verfahren** gearbeitet wird.

*) Weitere Ausführungen hierzu finden Sie im Band „Dateldienste der DBP; Datenfernverarbeitung", der vom Verlag der Deutschen Postgewerkschaft, 6 Frankfurt 71, Rhonestraße 2 herausgegeben wurde.

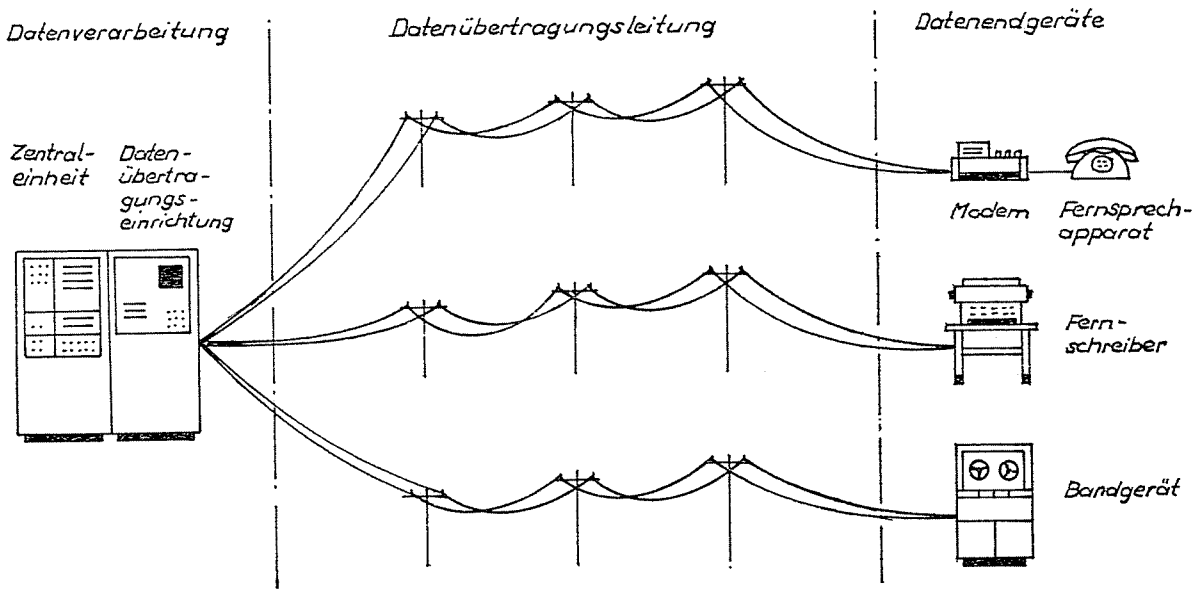


Abb. 2.85 — Prinzipielle Darstellung einer Datenfernverarbeitung im On-line-Betrieb

2.4.6.2. Datenfernverarbeitung im Off-line-Betrieb

Werden die Daten zur Verarbeitungsstelle übertragen, dort zwischengespeichert und auf einem Datenträger in einen maschinenlesbaren Code umgesetzt, wobei die eigentliche Verarbeitung zu einem späteren Zeitpunkt, unabhängig von der Datenübertragung, vorgenommen wird, dann spricht man von einer Datenfernverarbeitung im Off-line-Betrieb. Dieses Verfahren kommt dann zur Anwendung, wenn für die Bearbeitung der Daten genügend Zeit vorhanden ist.

Übertragungsgeschwindigkeit der verwendeten Datenendgeräte sind für die Datenübertragung unterschiedliche Übertragungsleitungen und Datenübertragungseinrichtungen (z. B. Modems) zu wählen. So stehen an Übertragungsleitungen Telegrafen- und Fernsprechstandleitungen sowie das Telex-, Datex- und öffentliche Fernsprechnetz zur Verfügung. Man unterscheidet grundsätzlich zwei Arten von Übertragungswegen, die Wählverbindung und die Standverbindung. Da gewählte Verbindungen weit mehr störungsanfällig sind als festgeschaltete und darüber hinaus eine Standverbindung stets die gleichen Verhältnisse aufweist, kann sie mit

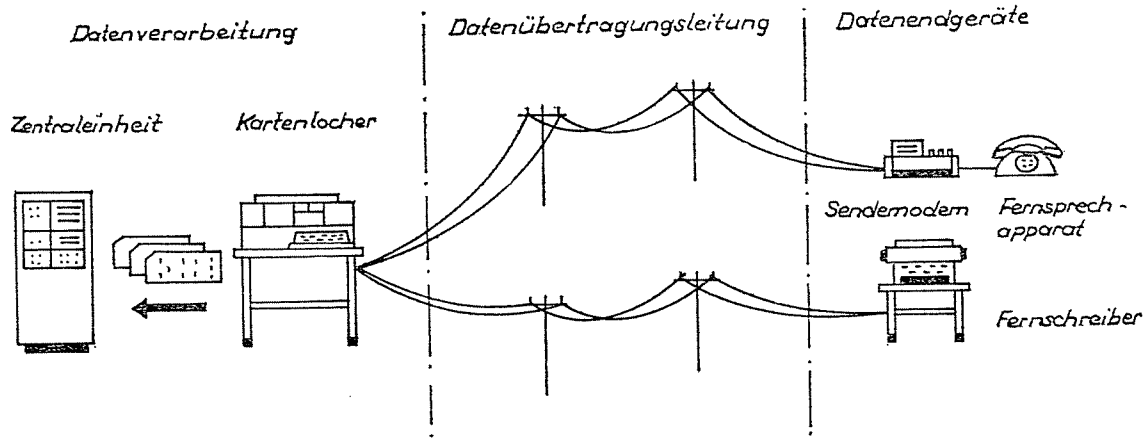


Abb. 2.86 — Prinzipielle Darstellung einer Datenfernverarbeitung im Off-line-Betrieb

2.4.6.3. Datenübertragung

Innerhalb eines Datenfernverarbeitungssystems stellt die Datenübertragung das Bindeglied zwischen den Ein- / Ausgabegeräten und den Einrichtungen der Datenverarbeitung dar. Je nach

einer höheren Übertragungsgeschwindigkeit betrieben werden als eine Wählverbindung. Dementsprechend verhalten sich die Übertragungsgeschwindigkeiten, die in der folgenden Übersicht zusammengestellt sind.

Übertragungsweg	max. Übertragungsgeschwindigkeit in Bit/s
Fernsprechwählverbindung	1200
Fernsprechstandverbindung	4800
Telegrafenwählverbindung (Telex)	50
Telegrafenwählverbindung (Datex)	200
Telegrafenstandverbindung	200
Breitband-Datenverbindung	etwa 150 000

gabe von Daten alle von der DVA ankommenden Meldungen auszuwerten und erforderlichenfalls zur Anzeige zu bringen oder notwendig werdende Maßnahmen automatisch einzuleiten. Wie aus Abb. 2.87 ersichtlich ist, sind an beiden Enden der Datenübertragungsleitung unterschiedliche Datenübertragungseinrichtungen eingesetzt. So wird beim Anschluß eines Fernschreibers als Datenübertragungseinrichtung ein Fernschaltgerät und bei der DVA eine Anschlußeinheit verwendet. Die Aufgabe der Steuerung ist es, die Verbindung zwischen der eigentlichen Datenübertragungseinrichtung und

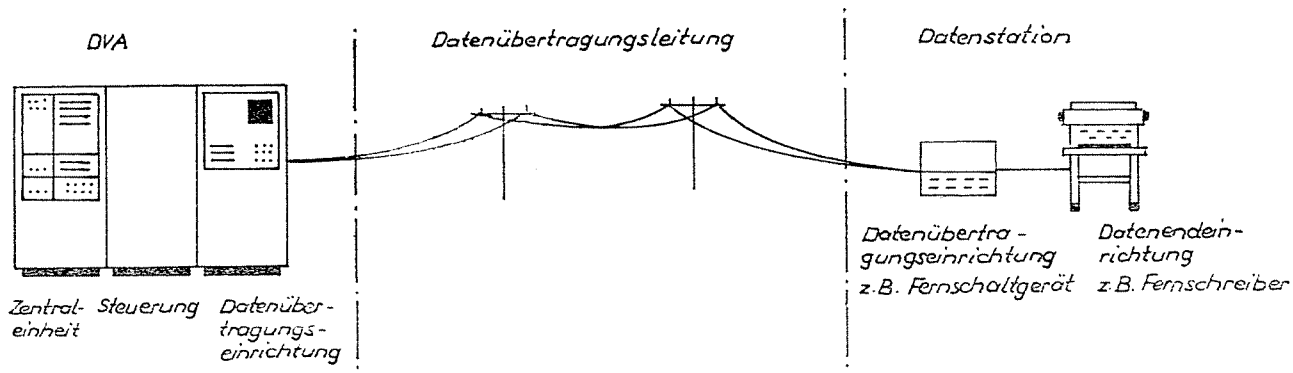
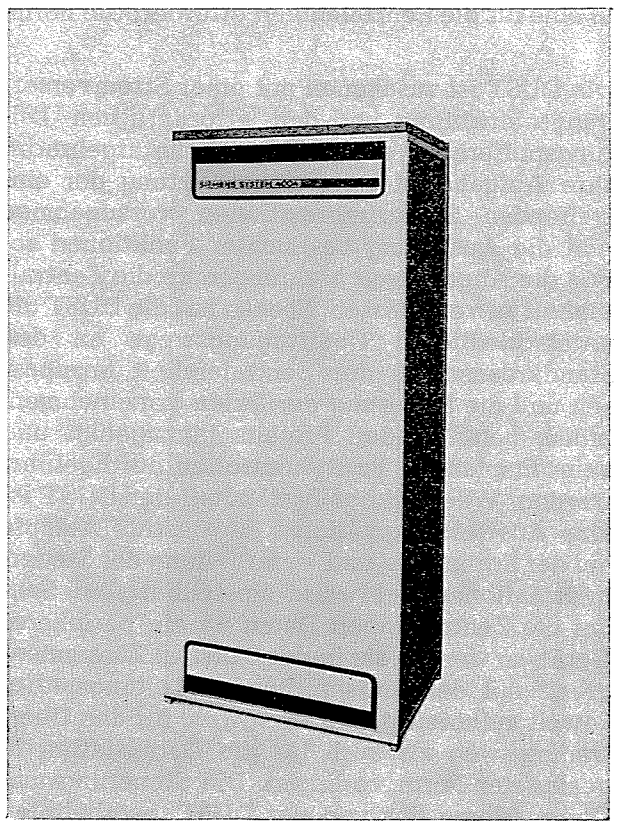


Abb. 2.87 — Prinzipielle Geräteanordnung einer Datenfernverarbeitung bei Verwendung eines Fernschreibers

Am Ein- bzw. Ausgang einer jeden Übertragungsleitung sind Datenübertragungseinrichtungen, z.B. Modems (Abkürzung für **Mo**dulation und **Dem**odulation) vorhanden, die als Verbindungsglieder zwischen der Übertragungsleitung und den auf der Sende- bzw. Empfangsseite angeschlossenen Geräten eingesetzt sind.

Die Datenübertragungseinrichtung der Sendeseite hat die Aufgabe, die zu übertragenden Daten umzusetzen, d.h. in eine für die Übertragung geeignete Frequenzlage zu bringen. Umgekehrt hat sie auf der Empfangsseite dafür zu sorgen, daß aus den eintreffenden Informationssignalen die ursprünglichen Daten zurückgewonnen werden können. Darüber hinaus ist es Aufgabe der Datenübertragungseinrichtungen, von der Datenübertragungsleitung bzw. von den Datenendeinrichtungen oder der DVA eintreffende Schaltkennzeichen auszuwerten und weiterzuleiten. Die Arbeitsweise der unterschiedlichen Datenübertragungseinrichtungen richtet sich nach der für den jeweiligen Einsatzfall erforderlichen Übertragungsgeschwindigkeit. Die Datenendeinrichtung hat die Aufgabe, dafür zu sorgen, daß eine reibungslose Kommunikation mit der Zentraleinheit möglich ist. Wird von ihr aus eine Verbindung aufgebaut, übernimmt meistenteils eine in der Datenendeinrichtung vorhandene Steuerung die automatische Datenübertragungsprozedur. Ebenso ist es Aufgabe der Datenendeinrichtung, neben der Ein- und Ausder Zentraleinheit herzustellen. In der Regel ist für jede Leitung ein Puffer und für alle Leitungen gemeinsam eine Steuerung vorhanden, die



(Werkfoto Siemens)

Abb. 2.88 — Datenübertragungssteuerung

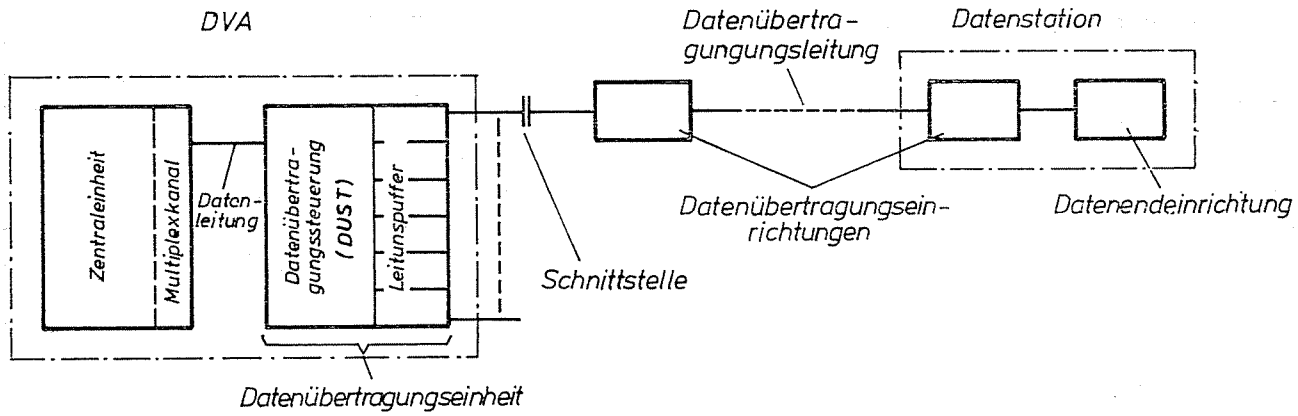


Abb. 2.89 — Prinzipielle Einordnung der DUST bei einem System für Datenfernübertragung

über einen Multiplexkanal an die Zentraleinheit der DVA angeschlossen ist. Stellvertretend für eine Reihe von zwar unterschiedlichen, in ihrer Funktionsweise jedoch ähnlichen Steuerungen, soll hier die der Fa. Siemens näher erläutert werden.

Die Datenübertragungssteuerung (DUST) ist über einen Multiplexkanal mit der Zentraleinheit verbunden. Die vom Leitungspuffer aufgenommenen Steuerzeichen und sonstigen Informationen werden an die DUST weitergegeben. Hier erfolgt eine Umsetzung der Datensignale in eine für die Zentraleinheit erforderliche Form.

Die DUST ist zusammen mit einer Stromversorgungseinrichtung und den erforderlichen Leitungspuffern in einem Schrank untergebracht. Ihre Aufgabe ist es, die Auswertung der eintreffenden Steuerinformationen vorzunehmen und die Auslösung bestimmter Funktionen sowie die Abgabe von Meldungen an die Zentraleinheit zu veranlassen. Ebenso hat die DUST die unterschiedlichen Übertragungswege an den Standardanschluß der Zentraleinheit anzupassen und die Sicherung der Daten vorzunehmen, indem beim Senden Prüfbits hinzugefügt und beim Empfangen Paritykontrollen durchgeführt werden. Wesentlicher Bestandteil der DUST ist eine Auswertelogik, deren gewünschte Abläufe bei der Erkennung und Verarbeitung der Steuerinformationen wahlweise einstellbar sind. Sendet die Zentraleinheit Daten an ein peripheres Gerät, so werden sie in der DUST in Zusammenarbeit mit dem entsprechenden Leitungspuffer soweit aufbereitet, daß sie sowohl für den Übertragungsweg als auch für die Datenstation in geeigneter Form vorliegen. Die Steuerung und Überwachung der von der DUST bearbeiteten Vorgänge liegt bei der Zentraleinheit und läuft auf Veranlassung eines erweiterten Organisationsprogramms ab, das speziell für die Anforderungen einer direkten Datenfernverarbeitung geschaffen wurde.

2.4.6.4. Betriebsarten der Datenfernverarbeitung

Grundsätzlich lassen sich in der Datenfernverarbeitung zwei Betriebsarten unterscheiden, der **Dialogbetrieb** und der **Stapelbetrieb.** Der Dialogbetrieb stellt eine Datenfernverarbeitung im Realzeitverfahren dar. Dabei werden die von einer Datenstation an die Zentraleinheit einer DVA gesandten Daten sofort verarbeitet und das Verarbeitungsergebnis umgehend der Datenstation übermittelt. **Unter den Dialogbetrieb fällt auch das Time-sharing-Verfahren,** das einer Vielzahl von Benutzern ermöglicht, gleichzeitig im Dialog mit einer DVA verkehren zu können, wobei die DVA den Dialog so gestaltet, daß beim Benutzer der Eindruck entsteht, die DVA stünde ausschließlich ihm zur Verfügung. Der Dialogbetrieb eignet sich vor allem für Auskunftssysteme aller Art. Auch die Arbeitsabläufe der Buchungsverfahren von Banken, Verkehrsgesellschaften und ähnlich gelagerter Betriebe lassen sich mit dem geschilderten Verfahren zeitsparend und wirtschaftlich abwickeln. Der **Stapelbetrieb** wird überall dort angewandt, wo eine Datenfernverarbeitung zwar erforderlich, ein sofortiges Verarbeiten der Daten aber nicht notwendig ist. Die Daten werden mit lokalen Datenerfassungsgeräten auf maschinenlesbaren Datenträgern erfaßt und bis zu einer gewissen Anzahl gesammelt. Die Verarbeitung der von einer Datenstation bei der Zentraleinheit eintreffenden Daten wird erst vorgenommen, wenn die Eingabe einer größeren Datenmenge stattgefunden hat. Entsprechend wird mit den Ergebnisdaten verfahren, die erst dann der

Datenstation zugeleitet werden, wenn die Verarbeitung einer größeren Datenmenge abgeschlossen ist. Für beide Arten der Datenfernverarbeitung werden unterschiedliche Datenendgeräte verwendet, die den Gegebenheiten beider Betriebsarten angepaßt sein müssen.

2.4.6.5. Periphere Geräte der Datenfernverarbeitung

Die Ein-/Ausgabegeräte der Datenfernverarbeitung mit ihren Gerätesteuerungen können über Datenübertragungseinrichtungen an die Standardanschlüsse des Multiplexkanals herangeführt werden. Solche Einrichtungen können sein: Lochstreifengeräte, Lochkartengeräte, Drucker, Auskunfts- oder spezielle Buchungsstationen, Fernschreiber und Datensichtstationen. Die z.Z. am häufigsten eingesetzten Dateneinrichtungen für den On-line-Datenfernverkehr sind der Fernschreiber und die Datensichtstation. Die Funktionsweise der wesentlichsten peripheren Geräte wurde bereits in den vorangegangenen Abschnitten erläutert. Es sollen an dieser Stelle nur noch der Fernsprechapparat und der Datenfernsprechapparat in ihrer künftig möglichen Eigenschaft als Datenstation erwähnt werden.

Fernsprechapparat als Datenendgerät in Nebenstellenanlagen

Mit der Einführung des Tastwahlapparates wurden die Voraussetzungen geschaffen, auch den Fernsprechapparat als Datenendgerät einsetzen zu können. Die erforderlichen Steuerfunktionen übernimmt hierbei eine elektronische Nebenstellenanlage.

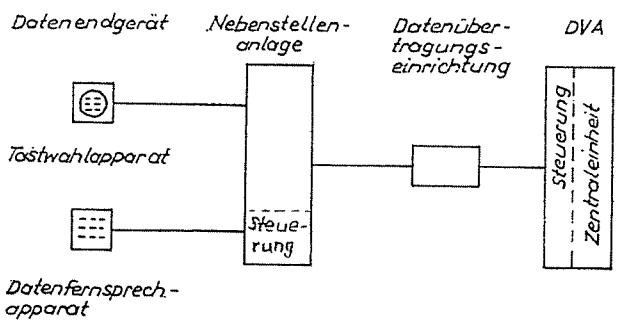


Abb. 2.90 — Prinzipielle Darstellung einer Datenverbindung zwischen einer Nebenstellenanlage und einer DVA

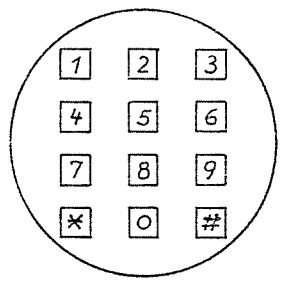


Abb. 2.91 — Beispiel einer Fernsprechtastatur

Die innerhalb der Nebenstellenanlage für einen Datenaustausch berechtigten Teilnehmer sind in der Lage, mit Hilfe der Fernsprechtastatur durch Eintasten einer bestimmten Rufnummer mit der DVA in Verbindung zu treten. Die Zentraleinheit antwortet mit einer akustischen Quittungsgabe. Eine Reihe von codierten Ziffernfolgen ermöglichen dem Fernsprechteilnehmer, einen Dialogverkehr mit der DVA zu führen. Er ist befähigt, bestimmte Programme aufzurufen, wobei ihm die DVA durch Absenden festgelegter Töne oder neuerdings in künstlicher Sprache die Dateneingabe beantwortet. Damit wird der Benutzerkreis eines Dialogverkehrs mit einer DVA derart erweitert, daß dem Fernsprechapparat als Dialoggerät eine große Zukunft eingeräumt werden muß.

Datenfernsprechapparat

Der Datenfernsprechapparat (zunächst nur im Versuchsstadium) ist ein speziell für die Belange einer Datenstation weiterentwickelter Fernsprechapparat. Mit seinem Tastenfeld und den Zusätzen für eine automatische Dateneinund -ausgabe wird der Fernsprechapparat zu einer „intelligenten" Dialogstation. Auch bei ihm wird die Steuerung von elektronischen Nebenstellenanlagen übernommen (vgl. hierzu Abb. 2.92).

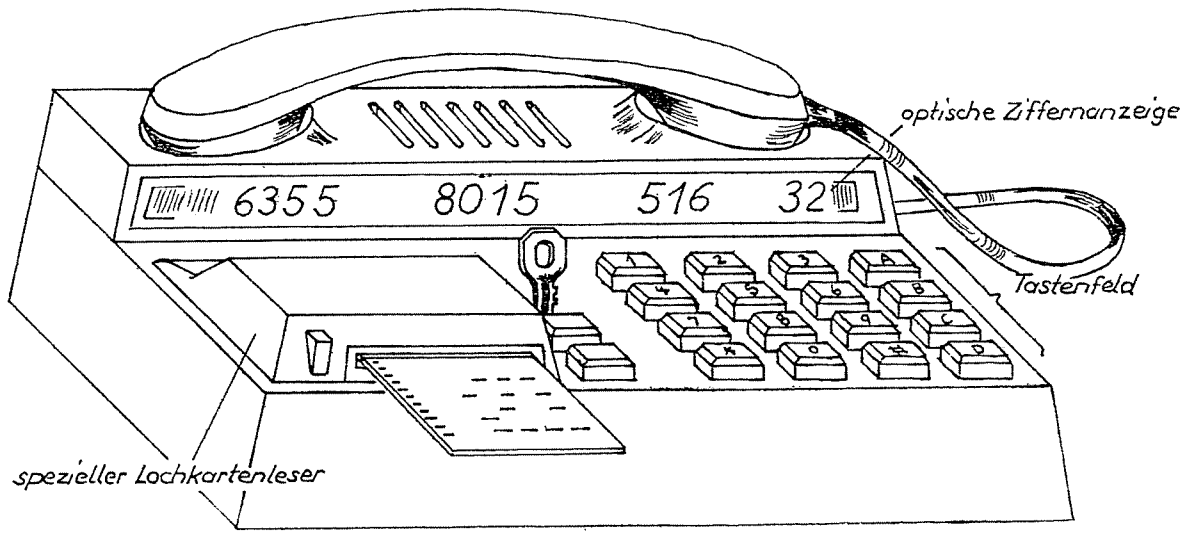
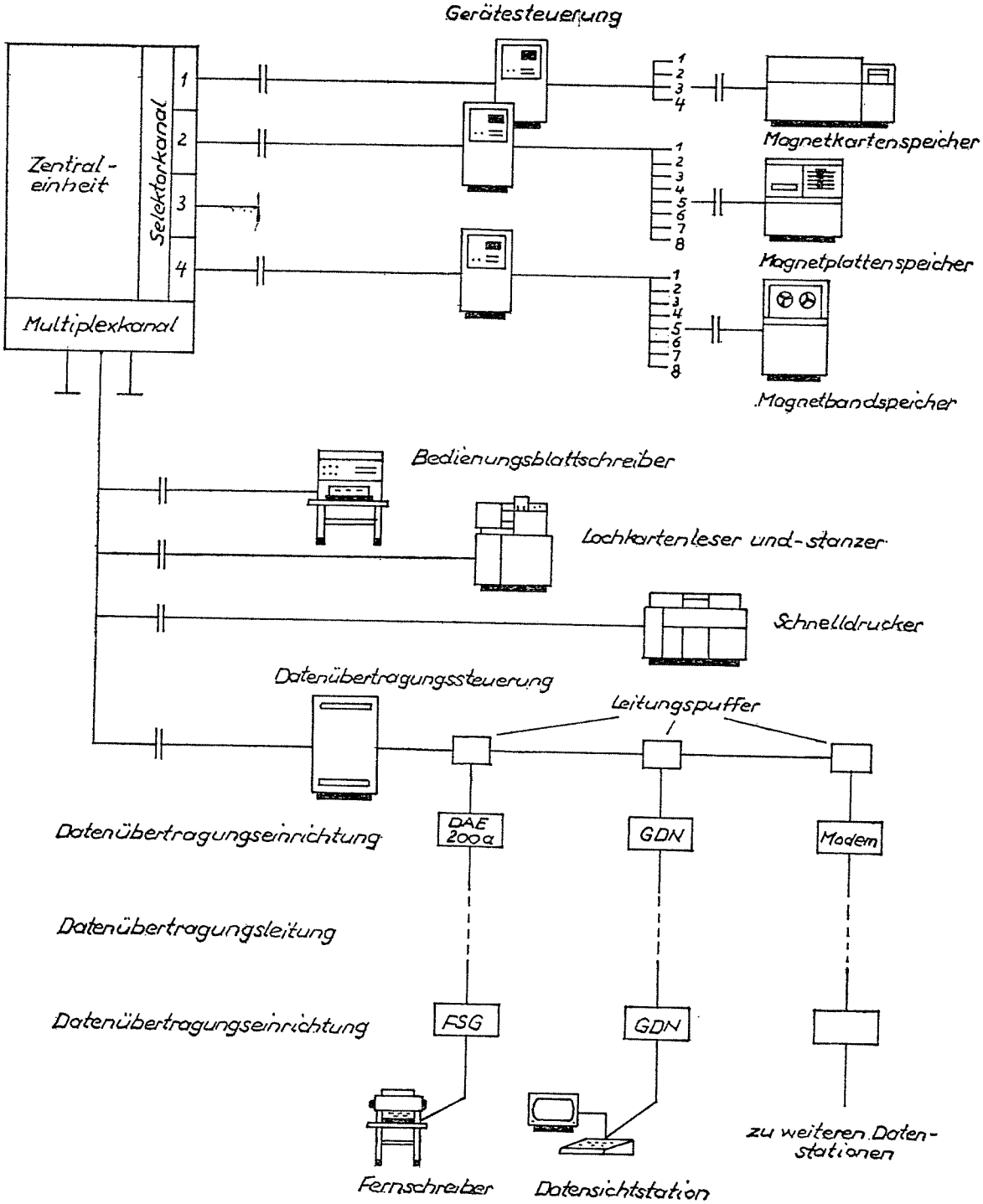


Abb. 2.92 — Datenfernsprechapparat



DAE = Datenanschlußeinheit
GDN = Gleichstrom-Datenübertragung für Niederpegel
FSG = Fernschaltgerät

Abb. 2.93 — Beispiel einer DVA-Anlagenkonfiguration (On-line-Peripherie)

3. Erweiterung der theoretischen Grundlagen

3.1. Einführung in die Mengenlehre

Viele Probleme der Datenverarbeitung lassen sich anschaulich mit den einfachen Strukturen der Mengenlehre darstellen, begreifen und lösen. Mindestkenntnisse der Mengenlehre sind deshalb für jeden Elektroniker äußerst nützlich, in vielen Anwendungsfällen sogar notwendig. Die Mengenlehre stellt eine universell verwendbare mathematische Theorie dar, die zudem noch leicht faßlich ist. Der Vorteil der Mengenlehre ist, daß nur allgemeine Objekte beliebiger Gesamtheiten betrachtet werden. Zunächst lassen wir nur ganz einfache Sätze über die **Objekte = Elemente** der beliebigen **Gesamtheiten = Mengen** zu. Das Weglassen aller speziellen Eigenschaften der Elemente einer Menge gestattet es, das Wesen der Mathematik schlechthin zu erkennen. Tiefliegende Gemeinsamkeiten verschiedener mathematischer Theorien werden sichtbar. Die Mengenlehre wurde um die Jahrhundertwende von dem deutschen Mathematiker Georg Cantor begründet. Ihre Bedeutung für die Grundlagenforschung und Naturwissenschaft ist heute unumstritten. Dem Elektroniker hilft die Mengenlehre bei der Betrachtung folgender Probleme

a) Synthese und Analyse von Schaltnetzen und Schaltwerken,

b) Codierungsprobleme,

c) Informationstheorie,

d) Programmiersprachen und vieles andere mehr.

3.1.1. Der Cantorsche Mengenbegriff

Den innerhalb der Mengenlehre benutzten Ausdruck „Menge" dürfen wir uns als umgangssprachliche Menge vorstellen. Cantor wollte unter einer Menge folgendes verstanden wissen: **„Unter einer Menge M verstehen wir die Zusammenfassung irgendwelcher Objekte unserer Anschauung oder unseres Denkens zu einem Ganzen. Die Objekte heißen Elemente m der Menge".** Für diesen Sachverhalt haben sich verschiedene Schreibweisen eingebürgert. Allen Schreibweisen gemeinsam ist, daß die Elemente der Menge in geschweiften Klammern eingeschlossen werden. Das sieht dann so aus:

$$M = \{m\} \qquad (1)\ ^{*})$$

In obiger Formel ist M die betrachtete Menge, der Buchstabe m repräsentiert dabei die **Elemente** der Menge. Hierbei ist m als Stellvertreter verschiedener Elemente anzusehen, über den noch ergänzende Aussagen gemacht werden müssen. Es gibt drei Formen, um diese ergänzenden Aussagen über die in einer Menge enthaltenen Elemente zu formulieren. Im folgenden Abschnitt 3.1.2. finden Sie Beispiele für diese drei Darstellungsformen.

3.1.2. Darstellungen von Mengen

Die Eigenschaften der Mengenelemente können zunächst **verbal** — d.h. also mit Worten — beschrieben werden. Dazu einige Beispiele:

Wir betrachten die Menge **N** aller natürlichen Zahlen. Natürliche Zahlen sind die Zahlen 1; 2; 3; usw.

Wir schreiben: **N** = Menge aller natürlichen Zahlen oder **Q** = Menge aller Bruchzahlen. **Q** wäre also die Menge aller Bruchzahlen, positive und negative.

Für bestimmte Zahlenmengen haben sich einheitliche Buchstabensymbole eingebürgert; diese Buchstaben heben wir durch Fettdruck hervor:

N = Menge aller natürlichen Zahlen

Z = Menge aller ganzen Zahlen, also 0; ±1; ±2 usw.

Q = Menge aller Bruchzahlen

R = Menge aller reellen Zahlen, also alle ganzen Zahlen, Bruchzahlen und irrationalen Zahlen

Natürlich ist die verbale Darstellung von Mengen nicht auf „Zahlenmengen" beschränkt. Was zu einer Menge zählen soll, bleibt dem Konstrukteur der Menge überlassen. Als Anregung für Mengenkonstruktionen hier wahllos drei Beispiele:

*) Die Formeln im Abschnitt 3 sind lfd. numeriert, da im weiteren Verlauf auf verschiedene Grundformeln verwiesen werden muß.

M = Menge aller Städte der Bundesrepublik Deutschland mit mehr als 1 Million registrierten Einwohnern

S = Menge aller am 1. 11. 71 in Hamburg angemeldeten privaten Personenkraftwagen

P = Menge aller aktiven Beamten der Deutschen Bundespost am 1. 12. 1971

Eine weitere Möglichkeit zur Darstellung von Mengen besteht in der **Aufzählung der Elemente,** die zu der betrachteten Menge gehören sollen:

$$M = \{ 1, 3, 5, 12 \} \qquad (2)$$

Die Menge M enthält alle Elemente, die in der geschweiften Klammer aufgezählt sind. Z.B. würde die Menge **P** aller Primzahlen, die größer als 0 und kleiner als 10 sind, so aussehen:

$$\mathbf{P} = \{ 1, 3, 5, 7 \} \qquad (3)$$

Die Menge B der ersten 3 Buchstaben des Alphabets schreiben wir dann so: B = {a, b, c}

Die Menge D aller Buchstaben des Wortes ‚Diode' sähe so aus: D = {i, o, d, D, e}

Sie erkennen hieran eine bedeutsame Tatsache: Die Anordnung der Elemente innerhalb der geschweiften Klammern ist zunächst unwesentlich. Erst wenn wir uns mit „geordneten" oder „wohlgeordneten Mengen" zu beschäftigen haben, ist die Anordnung der Elemente von Bedeutung.

Schließlich lassen sich Mengen durch **Aussagen** umschreiben:

$$M = \{ x \mid A(x) \} \qquad (4)$$

In M ist x dabei der Stellvertreter für ein Element, das durch die Aussage A(x) näher definiert wird. Für die Menge **N** aller natürlichen Zahlen würden wir dann schreiben: **N** = { x | x ist eine natürliche Zahl }.

Unsere Primzahlenmenge (3) wäre damit auch so darstellbar: **P** = { x | x ist Primzahl und $0 < x < 10$ }.

(Der Ausdruck in der Klammer ist zu lesen: x ist Primzahl, und x ist größer als 0 und kleiner als 10).

Gerade die Darstellung von Mengen in der Aussageform sollten wir uns merken. Innerhalb von Programmiersprachen fordern wir ständig eine widerspruchsfreie und knappe Umschreibung für bestimmte Anweisungen und Operationen. Die 3 folgenden Beispiele mögen der Übung dienen.

Beispiel 1

Die Menge M = { 2, 4, 6, 8, 10, 12 } ist in der Aussageform niederzuschreiben.

Es fällt auf, daß wir es mit allen **geraden Zahlen** zwischen 2 und 12 zu tun haben. Die Lösung lautet mithin:

$M = \{ x \mid x \text{ ist eine gerade Zahl und } 2 \leqq x \leqq 12 \}$

Beispiel 2

$$\mathbf{N} = \{ -1, -2, -3, \ldots -(n+1), -[(n+1)+1] \}$$

Wir stellen fest, daß hier alle negativen ganzen Zahlen mit Ausnahme der 0 gemeint sind. Die gleiche Menge würde in der Aussageform so aussehen:

$\mathbf{N} = \{ x \mid x \text{ ist eine negative ganze Zahl und } x \neq 0 \}$

Beispiel 3

K = { Karo-As, Herz-As }

Beide Elemente aus K sind rote Asse eines französischen Kartenblattes, so daß wir schreiben können:

K = { x | x ist ein As und x ist rot }

Wir halten also fest:

Mengen können beschrieben werden

a) verbal

b) durch Aufzählung der Elemente

c) durch Aussagen

3.1.3. Venn-Diagramme

Ein anschauliches Hilfsmittel zur Darstellung von Mengen stellt die grafische Methode dar. Nach dem Mathematiker John Venn tragen derartige Flächenfiguren den Namen **‚Venn-Diagramme'.** Abb. 3.1. zeigt 3 Möglichkeiten, um Mengen als Flächenfiguren darzustellen.

a) Zur ‚Punktmenge' M gehören alle Punkte innerhalb des Kreises.

b) Die Menge enthält die Elemente a, b, c, d; diese liegen innerhalb des ‚Mengenkörpers' M.

c) Innerhalb der „Universalmenge" Ω finden wir eine **Teilmenge** M dargestellt.

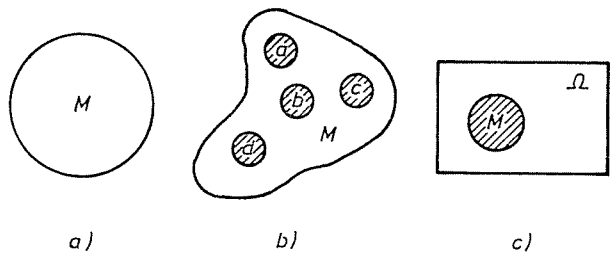


Abb. 3.1 — Venn-Diagramme

Venn-Diagramme eignen sich besonders für die Auswertung von Mengenbeziehungen.

3.1.4. Die Elementbeziehung

Wir betrachten jetzt die Elemente einer Menge. Die Tatsache, daß ein bestimmtes Element x in einer Menge M enthalten ist, schreiben wir

$$x \in M \tag{5}$$

und sprechen: **„x ist Element von M".** Ist das Element x in einer Menge N nicht zu finden, gilt:

$$x \notin M \tag{6}$$

Wir sprechen: **„x ist nicht Element von M".**

Beispiel 4

Es liegen die 3 Mengen

$$A = \{ 1, 2, 3, 3, 5, 7 \}$$

$$B = \{ x \mid x \text{ ist eine gerade Zahl und } 0 < x < 10 \}$$

$$C = \{ 1, \{ 2, 3, 5 \}, 7, 4 \}$$

vor. Es sind die Elementbeziehungen für das Element ‚2' für alle 3 Mengen aufzustellen.

Nach dem bisher Gesagten gilt ohne weitere Erläuterung:

$2 \in A$ sowie

$2 \in B$, denn B in anderer Form lautet:

$B = \{ 2, 4, 6, 8 \}$.

Die Verneinung der Elementbeziehung $2 \notin C$ bedarf einer Erläuterung:

Aus der Mengenkonstruktion

$$C = \{ 1, \{ 2, 3, 5 \}, 7, 4 \}$$

geht hervor, daß die Menge $\{ 2, 3, 5 \}$ ihrerseits **Element** der Menge C ist. Das ist eine äußerst zweckmäßige Verabredung: **Jede Menge kann Element einer anderen Menge sein!**

Für das Element 2 gilt aber:

$$2 \in \{ 2, 3, 5 \}.$$

Daraus folgt aber nicht, daß 2 auch Element von C ist, also $2 \notin C$.

3.1.5. Die Teilmengenbeziehung

Wir betrachten Abb. 3.2.

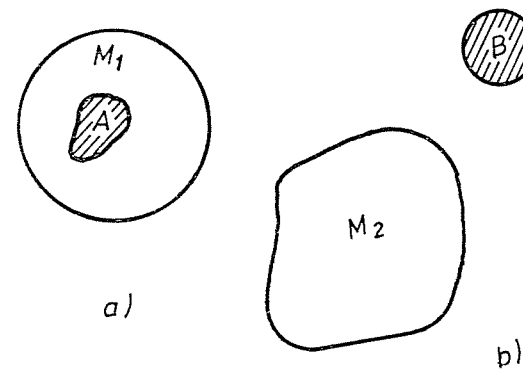


Abb. 3.2 — Teilmengenaussagen

a) A ist Teilmenge von M_1: $A \subset M_1$

b) B ist **nicht** Teilmenge von M_2: $B \not\subset M_2$

Rein überlegungsmäßig können wir doch innerhalb einer bestimmten Menge einige Elemente mit bestimmten Eigenschaften herausgreifen und als neue Menge begreifen. Weil diese neue Menge aber ein Teil der ursprünglichen Menge darstellt, nennen wir sie **Teilmenge oder Untermenge** der umfassenden Menge. So ließen sich z.B. alle blonden Schüler einer Schulklasse als Teilmenge der gesamten Klasse begreifen. Alle Hamburger Bürger bilden in diesem Sinne eine Teilmenge aller Bundesbürger.

Mathematisch definieren wir die Teilmenge so:

„A heißt Teilmenge von M, wenn jedes Element von A auch Element von B ist".

Verwenden wir unsere bekannten Symbole, so läßt sich schreiben:

$$A \subset M \text{ genau dann, wenn für } \textbf{jedes } x \in A \text{ folgt } x \in M \tag{7}$$

Wesentlich ist, daß Teilmengen aus nur **einem** Element bestehen können. Betrachten wir die Menge M, die 3 Elemente a, b, c enthält:

$$M = \{ a, b, c \}.$$

Wir finden wesentliche Zusammenhänge:

$a \in M$	$b \in M$	$c \in M$
$\{a\} \subset M$	$\{b\} \subset M$	$\{c\} \subset M$

Weiterhin gilt:

$$\{a, b\} \subset M;\ \{a, c\} \subset M;\ \{b, c\} \subset M$$

Neben den angeführten 6 sogenannten **„echten Teilmengen"**

$$\{a\},\ \{b\},\ \{c\},\ \{a, b\},\ \{a, c\},\ \{b, c\}$$

zählen wir auch die Menge $M = \{a, b, c\}$ zu den Teilmengen.

Aus Gründen der Zweckmäßigkeit enthält daneben **jede Menge** als Teilmenge auch eine **leere Menge.** Diese Menge hat überhaupt keine Elemente, wir geben ihr das Symbol $\emptyset$.

In Kurzform können wir schreiben:

$$\emptyset = \{ x \mid x \neq x) \} \qquad (8)$$

In Worten: „Die leere Menge enthält die Elemente, die nicht sich selbst gleich sind".

Die Menge selbst und die leere Menge sind sogenannte **„unechte Teilmengen"** einer **jeden** Menge:

$\emptyset \subset M$ gilt demnach für **alle** M!

Bezeichnen wir die Anzahl der Elemente einer Menge mit n, lassen sich grundsätzlich

$$z = 2^n \qquad (9)$$

Teilmengen konstruieren. Bei $n = 3$ sind demnach $2^3 = 8$ Teilmengen denkbar.

Die beiden Symbole „$\in$" für die **„Mengenmitgliedschaft"** (Elementbeziehung) und „$\subset$" für das **„Enthaltensein"** (Teilmengenbeziehung) müssen jedoch wohlunterschieden werden.

In unserer Menge $M = \{a, b, c\}$ ist a zwar Element von M, also $a \in M$, aber keineswegs auch Teilmenge von M! Ebenso ist $\{a\}$ eine echte Teilmenge $\{a\} \subset M$, aber keineswegs Element von M: $\{a\} \notin M$.

In Anlehnung an die obige Schreibweise führen wir für das **„Nichtenthaltensein"** das Symbol $\not\subset$ ein:

$$a \in M \quad \text{aber} \quad a \not\subset M.$$

Wir definieren:

$$A \not\subset M \text{ genau dann, wenn es ein } x \in A \text{ gibt, für das gilt } x \notin M \qquad (10)$$

Über die Teilmengenbeziehung läßt sich jetzt auch die **Gleichheit** zweier Mengen feststellen:

$$A = B \text{ genau dann, wenn } A \subset B \text{ und } B \subset A \qquad (11)$$

Die Mengengleichheit ist der Sonderfall einer **beliebigen Beziehung,** die sich zwischen Mengen herstellen läßt. Im nächsten Abschnitt wollen wir eine weitere Beziehung kennenlernen: die sogenannte „Äquivalenzrelation". Der Ausdruck **Relation** ist die mathematisch exakte Bezeichnung für Beziehungen zwischen Mengen und ihren Elementen. Weil in der Booleschen Algebra neben der Universalmenge Ω nur noch die leere Menge $\emptyset$ als Konstante benutzt wird, sollen noch 3 Beispiele für leere Mengen angeführt werden.

Beispiel 5

Gesucht ist die Menge aller natürlichen Zahlen, die kleiner als 1 und größer als 0 sind. Diese Menge ist leer:

$$\emptyset = \{ x \mid x \in N \text{ und } 0 < x < 1 \}$$

Beispiel 6

M sei die Menge aller **geraden** Primzahlen. Auch diese Menge ist leer:

$$\emptyset = \{ x \mid x \in \text{Primzahlen und } x \text{ ist eine gerade Zahl} \}$$

Beispiel 7

Für eine Menge M gilt $M \subset \emptyset$. Was ist M für eine Menge? Nach dem bisher Gesagten muß M die **leere Menge** sein:

$$M = \emptyset$$

3.1.6. Die Äquivalenzrelation

Um die Äquivalenzrelation begreifen zu können, führen wir den Begriff der **Mengenmächtigkeit** ein. Bei endlichen Mengen verstehen wir

unter der Mächtigkeit einer Menge die **Anzahl der in ihr enthaltenen Elemente.**

So hätte z.B. die Menge M = { 1, 9, 2, 3, 2, 5 } die Mächtigkeit 5. Das Element „2" ist natürlich sich selbst gleich und darf demnach nur **einmal** berücksichtigt werden!

Wir definieren:

> Zwei Mengen A und B sollen einander **äquivalent** sein, wenn sie von gleicher Mächtigkeit sind. (12)

Symbolisch drücken wir die **Äquivalenzrelation** durch das Zeichen „∼" aus:

$$A \sim B \qquad (13)$$

Die Äquivalenz zweier Mengen A und B läßt sich sehr einfach durch einen Vorgang feststellen, den wir als **Abbildung** bezeichnen. Dieser Ausdruck „geistert" durch die gesamte „neue Mathematik", wir wollen ihn uns deshalb gut einprägen. Wir nehmen ein einfaches Beispiel: Auf einer Party sind 5 Herren und 5 Damen anwesend. Wird zum Tanz gebeten, entstehen demnach 5 Tanzpaare. Weil in diesem speziellen Fall bei der Zuordnung der beiden Mengen eine Elementeerschöpfung auftritt, sind die beiden Mengen gleichmächtig und demnach einander **äquivalent.** Den Zustand der Zuordnung von Elementen einer Menge A zu Elementen einer Menge B bezeichnen wir als **Abbildung.** Abb. 3.3 zeigt die eben behandelte spezielle Abbildung zweier gleichmächtiger Mengen an einem „Graph".

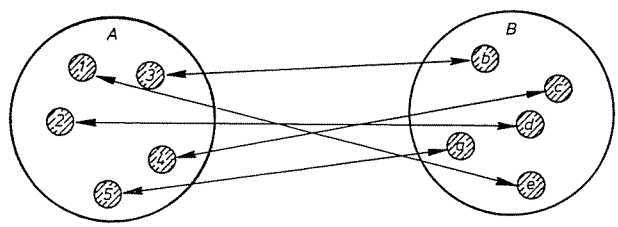


Abb. 3.3 — Die umkehrbar eindeutige oder ein-eindeutige Abbildung zweier Mengen A und B

Tritt bei der Abbildung zweier Mengen eine Elementeerschöpfung auf, heißt die Abbildung **umkehrbar eindeutig** oder **ein-eindeutig.**

Schließen wir die Elementepaare in eckige Klammern ein, also [x; y] = Elementepaar, dann besteht die Abbildung F aus der Menge der Elementepaare. Auf Abb. 3.3 bezogen:

$$F = \{[1;e], [2;d], [3;b], [4;c], [5;a]\} \qquad (14)$$

In der Mathematik bezeichnen wir eine derartige Zuordnungsvorschrift zur Bildung von Elementepaaren als **Gleichung** oder unter bestimmten Voraussetzungen auch als **Funktion.** Weichen die Mächtigkeiten zweier Mengen voneinander ab, kann die Abbildung nicht mehr umkehrbar eindeutig sein. Wir schwächen ab und sprechen von einer **eindeutigen Abbildung,** wenn **jedes** Element aus A **einmal** in der Abbildung auftritt. Abb. 3.4 zeigt eine eindeutige Abbildung am Beispiel eines einfachen Codes.

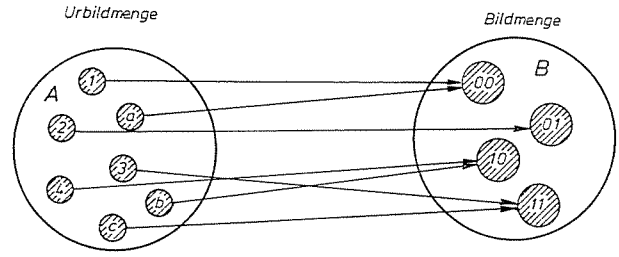


Abb. 3.4 — Eindeutige Abbildung der Menge A auf die Menge B

Der bekannte Fernschreibcode stellt z.B. eine eindeutige Abbildung nach Abb. 3.4 dar. Bedingt durch die Ziffern- und Buchstabenumschaltung werden Elemente der sogenannten **Bildmenge** B zweimal benutzt. Daß diese Abbildung nicht umkehrbar ist, leuchtet unmittelbar ein: Die Zuordnung gilt nur in Pfeilrichtung; wir können von einem bestimmten **Bildelement** y ε B nicht auf ein bestimmtes **Urbildelement** x ε A schließen.

3.1.7. Elementare Mengenalgebra

Ähnlich wie wir mit Zahlen rechnen können, lassen sich Mengen miteinander **verknüpfen.** Die drei elementaren Operationen mit Mengen sind die Bildung von

Mengendurchschnitt,

Mengenvereinigung und

Komplementärmenge.

3.1.7.1. Durchschnitt von Mengen

Der Durchschnitt von Mengen darf nicht verwechselt werden mit dem umgangssprachlichen Durchschnitt irgendwelcher Größen. Der Mengendurchschnitt ist das Ergebnis einer eindeutig definierten mathematischen Operation mit Mengen. Als mathematischen Operator für den Durchschnitt zweier Mengen verwenden wir das

Symbol „ ∩ “, so daß für die **Schnittmenge** zweier Gesamtheiten A und B der folgende konstruktive Aufbau entsteht:

$$A \cap B = D \tag{15}$$

Wir sprechen: **„A geschnitten mit B ist die Schnittmenge D“.**

Der Mengendurchschnitt wird uns sofort klar, wenn wir zwei konkrete Mengen A und B betrachten:

$$A = \{ 2, 4, 5, 6, 7, 8, 10 \}$$
$$B = \{ 1, 4, 7, 10 \}$$

Als **Durchschnitt** D definieren wir: Die Menge der Elemente, die einer Menge A und **zugleich** einer Menge B angehören, heißt der Durchschnitt beider Mengen.

In Kurzform gilt demnach:

$$A \cap B = \{ x \mid x \in A \text{ und } x \in B \} \tag{16}$$

Der Durchschnitt der eben genannten konkreten Mengen A und B wäre demnach:

$$A \cap B = \{ 4, 7, 10 \}.$$

Die Entstehung des Durchschnittes läßt sich anschaulich mit Hilfe von Venn-Diagrammen verdeutlichen. Abb. 3.5 zeigt 2 Beispiele.

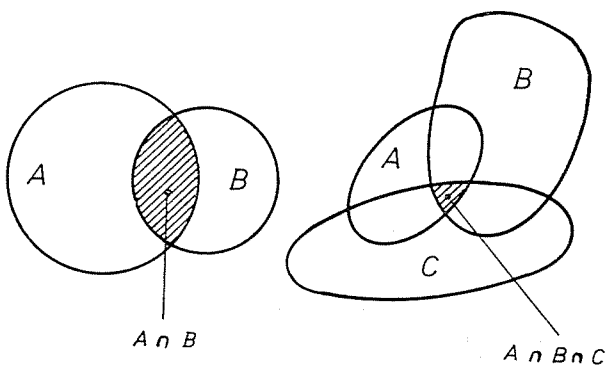


Abb. 3.5 — Durchschnitt von Mengen

Einleuchtend ist, daß der Durchschnitt **elementefremder** Mengen leer ist. Derartige Mengen heißen **disjunkte Mengen.**

Beispiel 8

Wir suchen die Schnittmengen A ∩ B, A ∩ C, B ∩ C sowie A ∩ B ∩ C der drei Mengen:

$$A = \{ 1, 2, 3, 4 \}$$
$$B = \{ 4, 5, 6 \}$$
$$C = \{ 6, 5 \}$$

Die Lösungen sind leicht gefunden:

$$A \cap B = \{ 4 \}$$
$$A \cap C = \{ \ \} = \emptyset.$$

A und C sind also disjunkte Mengen, sie verfügen über **kein gemeinsames** Element.

$$B \cap C = \{ 5, 6 \}$$

Wegen $A \cap C = \emptyset$ ist natürlich auch
$A \cap B \cap C = \emptyset$.

Bedeutsam ist, daß die Durchschnittverknüpfung kommutativ und assoziativ ist:

a) Kommutativgesetz (Vertauschungsregel)

$$A \cap B = B \cap A \tag{17}$$

Ob A mit B oder B mit A geschnitten wird, ist demnach gleichgültig. Wir erinnern uns an die formal ähnlichen arithmetischen Festlegungen:

$$a \cdot b = b \cdot a$$

b) Assoziativgesetz (Abtrennungsregel)

$$A \cap (B \cap C) = (A \cap B) \cap C \tag{18}$$

Dieses Gesetz sagt aus, daß die Reihenfolge der Schnittbildung bedeutungslos ist. Klammern innerhalb von Mengendurchschnitten dürfen also nach Belieben gesetzt oder fortgelassen werden. Auch dieses Gesetz kennen wir aus der elementaren Arithmetik: $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.

Weiterhin leuchten die beiden folgenden Beziehungen unmittelbar ein:

$$A \cap A = A$$
$$A \cap \emptyset = \emptyset \tag{19}$$

Diese beiden Formeln sagen nichts anderes aus, als daß

a) der Durchschnitt einer Menge mit sich selbst die Menge selbst ergibt und

b) der Durchschnitt **jeder** Menge mit der leeren Menge wiederum die leere Menge ergibt.

3.1.7.2. Vereinigung von Mengen

Die Vereinigung A ∪ B zweier Mengen A und B enthält alle Elemente, die zu A **oder** zu B gehören.

Wir schreiben

$$A \cup B = V \tag{20}$$

und sprechen: **„A vereinigt mit B ergibt die Vereinigungsmenge V".**

In Kurzform gilt für die Vereinigung:

$$A \cup B = \{ x \mid x \in A \text{ oder } x \in B \} \tag{21}$$

Das Venn-Diagramm in Abb. 3.6 verdeutlicht den Sachverhalt.

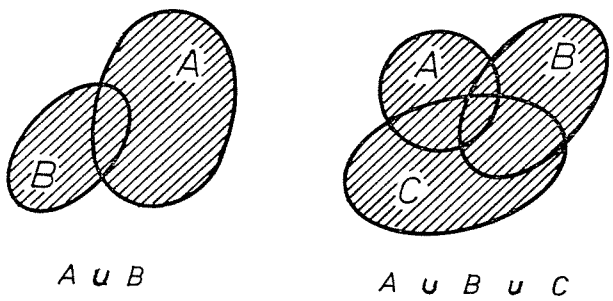


Abb. 3.6 — Vereinigung von Mengen

Beispiel 9

Gesucht werden die Vereinigungsmengen A ∪ B, A ∪ C, B ∪ C sowie A ∪ B ∪ C der drei Mengen:

$$A = \{ 1, 2, 3, 4 \}$$
$$B = \{ 4, 5, 6 \}$$
$$C = \{ 5, 6 \}$$

Wir finden als Lösung:

$$A \cup B = \{ 1, 2, 3, 4, 5, 6 \}$$
$$A \cup C = \{ 1, 2, 3, 4, 5, 6 \}$$
$$B \cup C = \{ 4, 5, 6 \}$$
$$A \cup B \cup C = \{ 1, 2, 3, 4, 5, 6 \}$$

Auch die Mengenvereinigung ist sowohl **kommutativ** als auch **assoziativ:**

$$\begin{aligned} A \cup B &= B \cup A \\ (A \cup B) \cup C &= A \cup (B \cup C) \end{aligned} \tag{22}$$

Auch hier erinnern wir uns an die arithmetischen Festlegungen für die Addition von Kardinalzahlen:

$$a + b = b + a$$
$$(a + b) + c = a + (b + c)$$

Unmittelbar einleuchtend sind die beiden folgenden Formeln:

$$\begin{aligned} A \cup \emptyset &= A \\ A \cup A &= A \end{aligned} \tag{23}$$

Aus der Vereinigungsverknüpfung zwischen Mengen läßt sich die Addition von Kardinalzahlen erklären. Wir betrachten die beiden **disjunkten** Mengen:

A mit der Mächtigkeit a = 4 und

B mit der Mächtigkeit b = 5.

Die Vereinigungsmenge A ∪ B = C verfügt dann über genau a + b = 4 + 5 = 9 Elemente.

Für die einfache Addition sind somit nur **disjunkte,** d.h. elementefremde Mengen, tauglich. Der Durchschnitt beider Mengen muß leer sein!

Zwischen der Durchschnittmenge A ∩ B sowie der Vereinigungsmenge A ∪ B gilt stets die **Teilmengenrelation:**

$$(A \cap B) \subset (A \cup B) \tag{24}$$

Abb. 3.7 verdeutlicht diesen Zusammenhang.

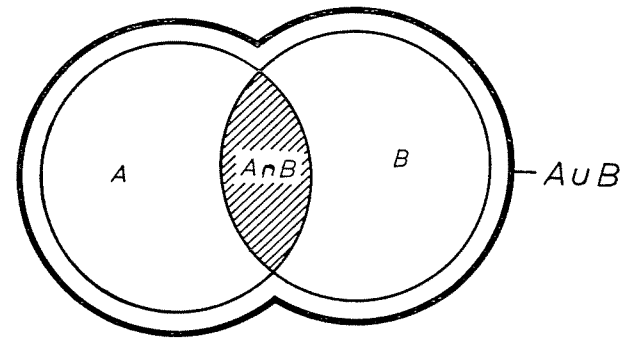


Abb. 3.7 — Es gilt stets: (A ∩ B) ⊂ (A ∪ B)

Diese in Abb. 3.7 gezeigte Aussage ist äußerst wichtig in Hinblick auf die Rechenregeln der Aussagenlogik und Schaltalgebra. Dort erscheint die Vereinigung als sogenannte „Inklusiv-ODER-Beziehung". Die aus dem Mengendurchschnitt heraus erklärbare „UND-Verknüpfung" ist stets innerhalb der „Inklusiv-ODER-Verknüpfung" enthalten.

3.1.7.3. Komplementärmengen

Um mit Mengen nach mengenalgebraischen Gesetzen rechnen zu können, lassen wir von vornherein nur eine bestimmte **Universalmenge Ω** zu. Ω soll dabei **alle** Elemente enthalten, die innerhalb der Theorie vorkommen können. Wir stellen uns die Universalmenge der Theorie als

rechteckförmiges Venn-Diagramm vor. Wie Abb. 3.8 es zeigt, liegen dann alle denkbaren Mengen A, B, C, ..., Z innerhalb des Mengenkörpers Ω, so daß folgt:

Für **alle** M gilt:

$M \subset \Omega$ gilt genau dann, wenn $M \cap \Omega = M$.

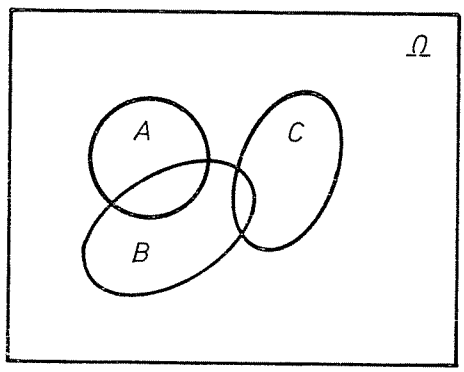


Abb. 3.8 — Alle Mengen der Theorie sind Teilmengen der Universalmenge Ω

Aus Abb. 3.8 geht weiter hervor, daß die Vereinigung einer beliebigen Menge M mit der Universalmenge Ω stets die Universalmenge Ω ergibt:

$$M \cup \Omega = \Omega \tag{25}$$

Um die Definition der **Komplementär- oder Ergänzungsmenge** verstehen zu können, betrachten wir die Menge $A \subset \Omega$ in Abb. 3.9.

Wir definieren: **„Die Menge der Elemente innerhalb von Ω, die nicht zu A gehört, heißt die Komplementärmenge zu A".** Komplementärmengen kennzeichnen wir durch Überstreichung, also $\overline{A}$.

In Kurzform lautet die Definition für $\overline{A}$:

$$\overline{A} = \{ x \mid x \in \Omega \text{ und } x \notin A \} = \{ x \mid x \notin A \} \tag{26}$$

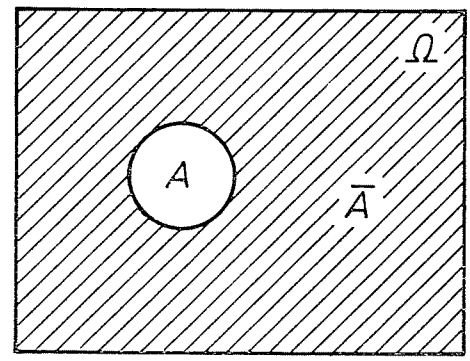


Abb. 3.9 — Die zu A komplementäre Menge $\overline{A}$ im Venn-Diagramm

Beispiel 10

Wir suchen einen mengenalgebraischen Ausdruck für die in Abb. 3.10 dargestellte sogenannte **Differenzmenge** $D = A \setminus B$.

Lies: „D ist die Differenzmenge **A ohne B**".

Die **Differenzmenge** ist wie folgt definiert:

$$A \setminus B = \{ x \mid x \in A \text{ und } x \notin B \} \tag{27}$$

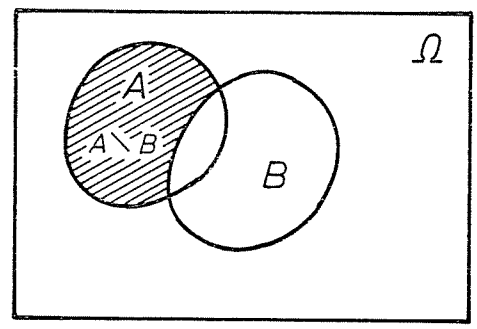
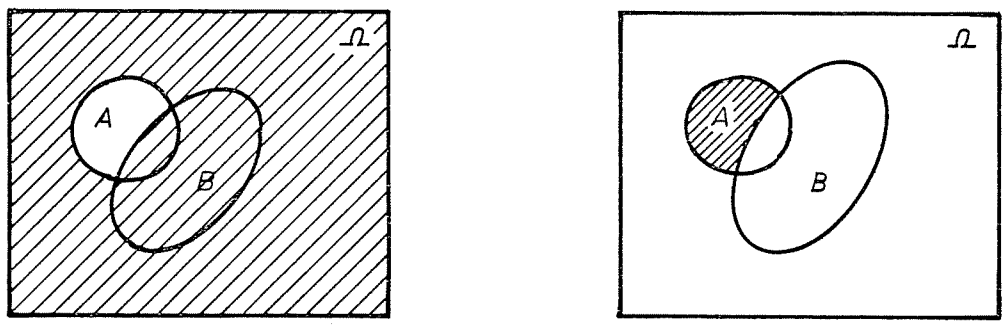


Abb. 3.10 — Die Differenzmenge A\B im Venn-Diagramm

Wir finden als erste Lösung:

$$A \setminus B = A \cap \overline{B} \tag{28}$$



a) b)

Abb. 3.11 — Beweis am Venn-Diagramm, daß $A \setminus B = A \cap \overline{B} = \overline{\overline{A} \cup B}$

a) Vereinigung von $\overline{A}$ mit B ergibt $\overline{A} \cup B$. b) Das Komplement von $\overline{A} \cup B$ ergibt $A \setminus B$.

Daneben existiert noch eine zweite Lösung, die den gleichen Sachverhalt erfaßt. Wir betrachten zu diesem Zwecke Abb. 3.11 a), die die Beziehung $\overline{A} \cup B$ veranschaulicht.

Aus Abb. 3.11 a) geht hervor, daß die Vereinigung $\overline{A} \cup B$ alle Elemente umfaßt, die nicht in $A \setminus B$ enthalten sind. Gemäß unserer Definition (26) muß dann das Komplement

$$\overline{\overline{A} \cup B} = A \setminus B \quad \text{sein.}$$

An diesem einfachen Beispiel erkennen Sie bereits folgendes: Ein gegebener mengenalgebraischer Sachverhalt läßt sich stets durch mindestens 2 Formeln erfassen. Die Hintergründe liegen in dem noch zu besprechenden **Dualitätsprinzip,** welches seinen Niederschlag in den **De Morganschen Gesetzen** findet.

3.1.7.4. Das Dualitätsprinzip

Das Dualitätsprinzip sagt aus, daß es zu jeder mengenalgebraischen Gleichung eine **duale Gleichung** gibt, die unter Verwendung der gleichen Mengen einen äquivalenten Sachverhalt erfaßt. Wir betrachten dazu Abb. 3.12.

Daß Abb. 3.12 a) im Prinzip den gleichen Sachverhalt wie Abb. 3.12 b) erfaßt, leuchtet unmittelbar ein: beide Bilder sind einander gleichwertig.

Beispiel 11

Für den in Abb. 3.13 dargestellten Sachverhalt ist zunächst eine Gleichung zu finden. Im zweiten Schritt ist die duale Beziehung zu bilden.

Der mengenalgebraische Ausdruck für das schraffierte Gebiet in Abb. 3.13 muß lauten: $X = (A \cup B) \cap \overline{C}$. Die duale Beziehung heißt dann: $\overline{X} = (\overline{A} \cap \overline{B}) \cup C$.

Aus diesem einfachen Beispiel geht der Formalismus für das Aufstellen dualer Beziehungen unmittelbar hervor: Wenn in einer mengenalgebraischen Gleichung **alle** Mengen durch ihr Komplement ersetzt werden und gleichzeitig das Verknüpfungszeichen „$\cap$" durch „$\cup$" ersetzt wird und umgekehrt, dann entsteht eine **duale Beziehung.** Als duale Gleichung bezeichnen wir eine neue Gleichung, deren Richtigkeit genau wie in der ursprünglichen Gleichung von den **gleichen** Mengen abhängt.

Das Dualitätsprinzip ist äußerst nützlich, da es auch in der Schaltalgebra auftaucht. Dort erleichtern wir uns häufig durch Anwendung des

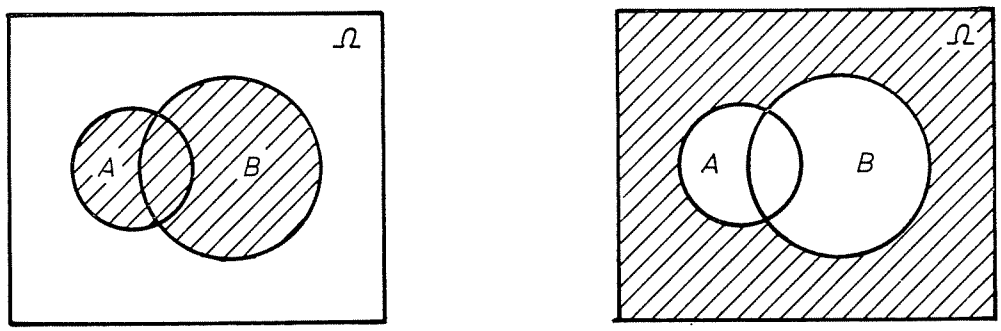


Abb. 3.12 — Das Dualitätsprinzip am Venn-Diagramm

a) die Vereinigung $A \cup B = V$, b) die zur Vereinigung $A \cup B = V$ duale Gleichung $\overline{V} = \overline{A} \cap \overline{B}$.

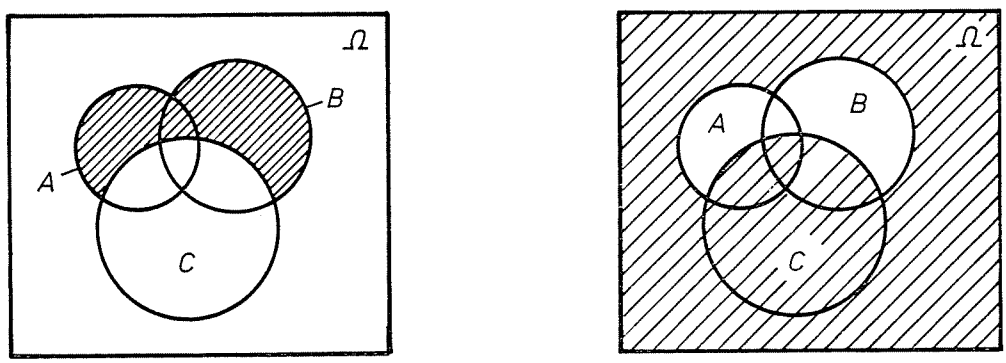


Abb. 3.13 — Mengenverknüpfungen am Venn-Diagramm

Dualitätsprinzips die rechnerische Vereinfachung von Schaltfunktionen.

3.1.7.5. Die Gesetze von De Morgan

Die De Morganschen Gesetze befassen sich mit der Komplementierung ganzer mengenalgebraischer Ausdrücke. Diese Gesetze sind unmittelbar aus dem Dualitätsprinzip ableitbar. Wir betrachten Abb. 3.14, in der der mengenalgebraische Ausdruck $\overline{A \cap B}$ als schraffierte Fläche dargestellt ist.

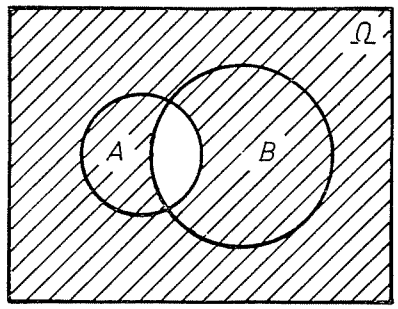


Abb. 3.14 — Das Gesetz von De Morgan $\overline{A \cap B} = \overline{A} \cup \overline{B}$ am Venn-Diagramm

Daß die in Abb. 3.14 schraffierte Fläche ebensogut durch den Ausdruck $\overline{A} \cup \overline{B}$ erfaßt wird, leuchtet nach dem bisher Gesagten unmittelbar ein. Das **erste Gesetz von De Morgan** lautet somit:

$$\overline{A \cap B} = \overline{A} \cup \overline{B} \tag{29}$$

Beispiel 12

Anhand des Venn-Diagramms ist das **zweite Gesetz von De Morgan**

$$\overline{A \cup B} = \overline{A} \cap \overline{B} \tag{30}$$

zu beweisen.

Die Lösung muß so aussehen:

Die beiden De Morganschen Identitäten stellen mit den anschließend folgenden Distributivgesetzen die wichtigsten Regeln für die rechnerische Vereinfachung von Schaltfunktionen dar. In Worten können wir die beiden De Morganschen Gesetze wie folgt erfassen:

a) Die Komplementärmenge eines Mengendurchschnittes ist gleich der Vereinigung der Komplementärmengen der einzelnen Mengen.

b) Die Komplementärmenge einer Mengenvereinigung ist gleich dem Durchschnitt der Komplementärmengen der betrachteten Mengen.

3.1.7.6. Das erste Distributivgesetz

Zu dem aus der Arithmetik bekannten Distributivgesetz (Verteilungsregel) $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ gibt es ein Analogon in der Mengenalgebra. Dem Durchschnittoperator „$\cap$" kommt hierbei quasi die Bedeutung des Multiplikationszeichens „$\cdot$" zu:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \tag{31}$$

Daß dieses Gesetz richtig ist, beweisen wir am Venn-Diagramm. Wir bilden zunächst die beiden Durchschnitte $A \cap B$ sowie $A \cap C$, um sie dann zu vereinigen. Abb. 3.16 zeigt das Verfahren.

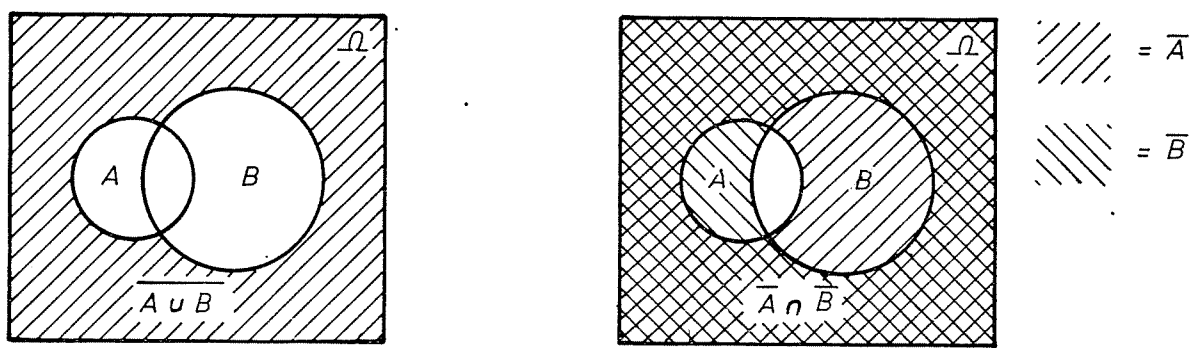


Abb. 3.15 — Zweites Gesetz von De Morgan: $\overline{A \cup B} = \overline{A} \cap \overline{B}$ am Venn Diagramm

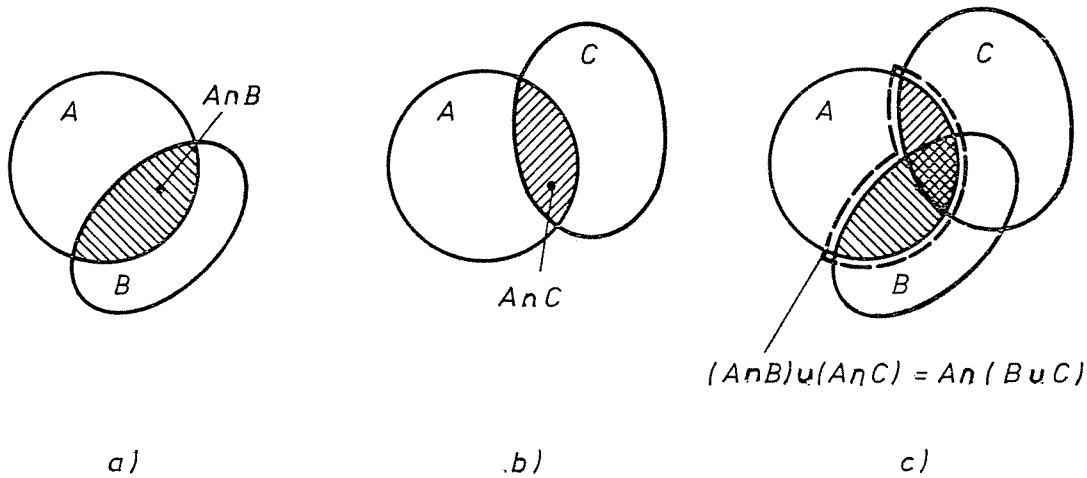


Abb. 3.16 — Das erste Distributivgesetz im Venn-Diagramm

a) Bildung des Durchschnitts $A \cap B$
b) Bildung des Durchschnitts $A \cap C$
c) Vereinigung beider Durchschnitte: $(A \cap B) \cup (A \cap C)$

Wir erkennen, daß in Abb. 3.16 c) die schraffierte Fläche tatsächlich dem Ausdruck $A \cap (B \cup C)$ entspricht.

Beispiel 13

Der mengenalgebraische Ausdruck $A \cap (A \cup B)$ ist zeichnerisch mit Venn-Diagrammen zu vereinfachen.

Die Lösung ist leicht gefunden, wie Abb. 3.17 es zeigt. Wir bilden zunächst die Vereinigung $A \cup B$, um dann den Durchschnitt mit A zu bilden.

Die soeben gefundene Beziehung

$$A \cap (A \cup B) = A \tag{32}$$

heißt wegen der Absorption von B das erste **Absorptionsgesetz.** Dieses Gesetz läßt sich auch rechnerisch beweisen, indem wir auf den Ausdruck das erste Distributivgesetz (31) anwenden:

$$A \cap (A \cup B) = (A \cap A) \cup (A \cap B)$$

Wegen $A \cap A = A$ (19) entsteht:

$$A \cap (A \cup B) = A \cup (A \cap B).$$

Aufgrund der Teilmengenaussage (24)

$$(A \cap B) \subset (A \cup B)$$

muß die Schnittmenge $A \cap B$ stets innerhalb von A liegen! Die Anweisung lautet mithin, eine echte Teilmenge $(A \cap B) \subset A$ mit der Menge A zu vereinigen: Das ergibt aber die Menge selbst.

3.1.7.7. Das zweite Distributivgesetz

Das zweite Distributivgesetz der Mengenalgebra lautet:

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \tag{33}$$

Für dieses Gesetz finden wir kein Analogon in der normalen Algebra, denn dort stellt die Formel

$$a + (b \cdot c) \neq (a + b) \cdot (a + c)$$

eine Ungleichung dar. Den Beweis für die Richtigkeit des zweiten Distributivgesetzes der Mengenalgebra liefert Abb. 3.18 am Venn-Diagramm. Wir bilden zunächst die beiden Vereinigungsmengen $A \cup B$ sowie $A \cup C$, um sie dann zu schneiden.

Abb. 3.17 — Das Absorptionsgesetz $A \cap (A \cup B) = A$ am Venn-Diagramm

a) Bildung der Vereinigungsmenge $A \cup B$ b) Die Schnittmenge $A \cap (A \cup B)$ entspricht eindeutig A

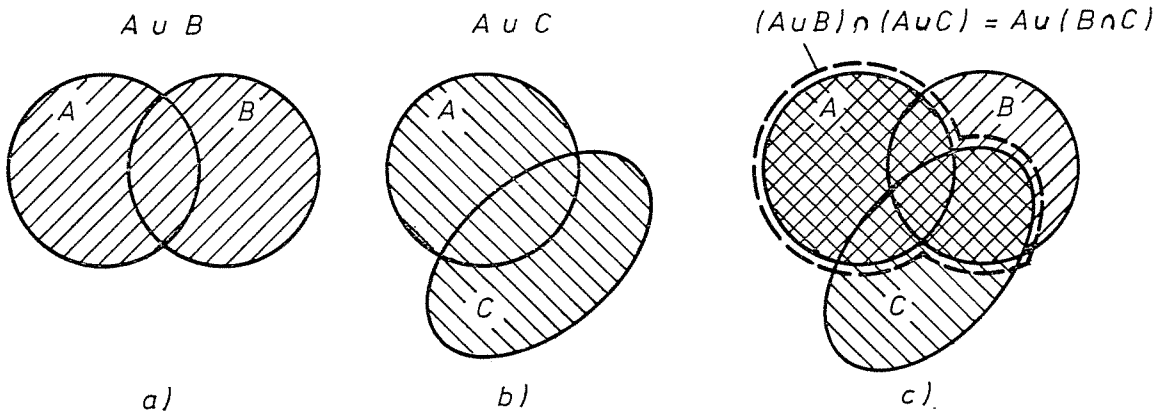


Abb. 3.18 — Das zweite Distributivgesetz im Venn-Diagramm

a) Bildung der Vereinigungsmenge A ∪ B
b) Bildung der Vereinigungsmenge A ∪ C
c) Der Durchschnitt (A ∪ B) ∩ (A ∪ C) ist gleich A ∪ (B ∩ C)

Beispiel 14

Das zweite Absorptionsgesetz A ∪ (A ∩ B) = A ist mit Hilfe von Venn-Diagrammen zu beweisen.

Wir gehen so vor, daß wir zunächst den Durchschnitt (A ∩ B) bilden. Wegen (A ∩ B) ⊂ A liegt dieser stets innerhalb von A, so daß die Vereinigung mit A wiederum A ergibt!

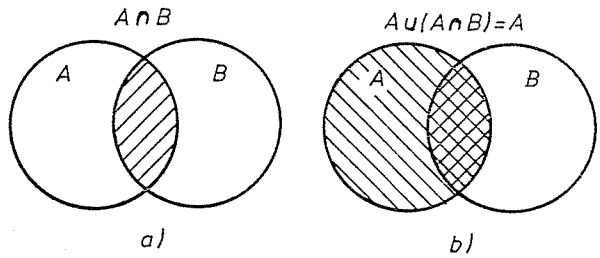


Abb. 3.19 — Das Absorptionsgeseitz A ∪ (A ∩ B) = A am Venn-Diagramm

a) Bildung des Durchschnitts A ∩ B
b) Vereinigung von A ∩ B mit A ergibt A

Das mit Abb. 3.19 bewiesene Gesetz

$$A \cup (A \cap B) = A \qquad (34)$$

heißt das zweite **Absorptionsgesetz** der Mengenalgebra.

3.1.7.8. Zusammenstellung der Theoreme der Mengenalgebra

In den vorangegangenen Abschnitten haben wir gesehen, wie aus den drei Grundverknüpfungen „Durchschnitt", „Vereinigung" und „Mengenkomplement" neue Beziehungen gewonnen werden können. Derartige zusammengesetzte mengenalgebraische Ausdrücke bezeichnen wir als **Theoreme.** In diesem Abschnitt stellen wir zur einfacheren Handhabung alle für die Mengenalgebra wichtigen Gesetze zusammen. Entsprechend dem Dualitätsprinzip werden die Rechenregeln jeweils zu Paaren zusammengefaßt.

1. Grundgleichungen mit der Universalmenge und der leeren Menge

a) Bildung des Mengenkomplements

$$\overline{\emptyset} = \Omega \qquad \overline{\Omega} = \emptyset$$

b) Durchschnitt

$$\emptyset \cap \emptyset = \emptyset$$
$$\emptyset \cap \Omega = \emptyset$$
$$\Omega \cap \emptyset = \emptyset$$
$$\Omega \cap \Omega = \Omega$$

c) Vereinigung

$$\Omega \cup \Omega = \Omega$$
$$\Omega \cup \emptyset = \Omega$$
$$\emptyset \cup \Omega = \Omega$$
$$\emptyset \cup \emptyset = \emptyset$$

2. Grundgleichungen mit einer Menge A

a) Zweifaches Komplement

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

b) Durchschnitt

$$A \cap \emptyset = \emptyset$$
$$A \cap \Omega = A$$
$$\bar{A} \cap \emptyset = \emptyset$$
$$\bar{A} \cap \Omega = \bar{A}$$

c) Vereinigung

$$\bar{A} \cup \Omega = \Omega$$
$$\bar{A} \cup \emptyset = \bar{A}$$
$$A \cup \Omega = \Omega$$
$$A \cup \emptyset = A$$

d) Tautologiegesetze

$$A \cap A = A \qquad \bar{A} \cup \bar{A} = \bar{A}$$
$$\bar{A} \cap \bar{A} = \bar{A} \qquad A \cup A = A$$

e) Komplementbeziehungen

$$A \cap \bar{A} = \emptyset \qquad \bar{A} \cup A = \Omega$$
$$\bar{A} \cap A = \emptyset \qquad A \cup \bar{A} = \Omega$$

3. Grundgleichungen mit 2 Mengen

a) Kommutative Gesetze

$$A \cap B = B \cap A \qquad A \cup B = B \cup A$$

b) De Morgansche Theoreme

$$\overline{A \cap B} = \bar{A} \cup \bar{B} \qquad \overline{A \cup B} = \bar{A} \cap \bar{B}$$
$$A \cap B = \overline{\bar{A} \cup \bar{B}} \qquad A \cup B = \overline{\bar{A} \cap \bar{B}}$$

4. Grundgleichungen mit 3 Mengen

a) Assoziative Gesetze

$$(A \cap B) \cap C = A \cap (B \cap C) = A \cap B \cap C$$

$$(A \cup B) \cup C = A \cup (B \cup C) = A \cup B \cup C$$

b) Distributive Gesetze

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$
$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

5. Absorptionsgesetze

$$A \cap (A \cup B) = A$$
$$A \cap (\bar{A} \cup B) = A \cap B$$

$$A \cup (A \cap B) = A$$
$$A \cup (\bar{A} \cap B) = A \cup B$$

3.2. Grundzüge der Aussagenlogik

Die Aussagenlogik stellt einen sogenannten mathematischen **Kalkül** dar, um aus beliebigen Anfangsaussagen unter Anwendung der definierten Verknüpfungsgesetze immer neue zusammengesetzte Aussagen (Theoreme) zu gewinnen. Die zugelassenen mathematischen Regeln stellen die **Kalkülregeln** des logischen Systems dar. Unter allen bekannten logischen Systemen stellt die Aussagenlogik den einfachsten Kalkül dar.

Die Elemente der Aussagenlogik sind **Aussagen** in Form einer sprachlichen Mitteilung. Als Zeichenfahrzeuge führen wir für Aussagen Buchstabensymbole ein, z.B.:

A: Der Mond ist unbewohnt.

B: Transistoren sind elektronische Bauelemente.

In Verbindung mit Verknüpfungszeichen, deren Bedeutung eindeutig und widerspruchsfrei festgelegt ist, bietet die Verwendung von Zeichenfahrzeugen die Möglichkeit zu formalisieren. Ein derartiges formal-logisches System schuf 1854 zum erstenmal der Engländer George Boole, weshalb die mathematische Logik auch zu den übergeordneten **Booleschen Verbänden** zählt. Zu einem Booleschen Verband gehören heute die wichtigen Kalküle: Mengenalgebra, mathematische Logik, Schaltalgebra und Ereignisalgebra.

Allen Kalkülen gemeinsam ist, daß sie zweiwertig oder **bivalent** sind. Der zweiwertige Charakter bezieht sich hierbei auf die Objekte, mit denen innerhalb eines Kalküls gerechnet wird. In der bereits behandelten Mengenalgebra ist der bivalente Charakter durch die Symbole ε und $\notin$ bzw. $\subset$ und $\not\subset$ gegeben. Über ein Element a oder eine Teilmenge M können schließlich nur die **zwei** Aussagen gemacht werden:

Entweder ist $a \varepsilon M$ bzw. $M \subset \Omega$

oder $a \notin M$ bzw. $M \not\subset \Omega$

„Halbenthaltensein" oder „Halbmengenmitgliedschaft" ist ausgeschlossen.

In der Aussagenlogik ist die Zweiwertigkeit dadurch gegeben, daß wir den betrachteten Aussagen nur **zwei** Wahrheitswerte zubilligen. Wir betrachten die beiden Aussagen A und B:

A: Der Mensch ist ein Säugetier.

B: Fünf plus Zwei ist gleich Vierzehn.

Elementare Verabredungen erlauben uns zu sagen, daß die Aussage A **wahr** ist, B dagegen ist **falsch.** Aus Zweckmäßigkeitsgründen führen wir für die Zustände „Wahr" und „Falsch" sogenannte **Wahrheitswerte** ein:

Wahre Aussage = Wahrheitswert 1 Falsche Aussage = Wahrheitswert 0	(35)

„Halbwahrheiten" und „Halbunwahrheiten", wie sie manchmal im Sprachgebrauch vorkommen, schließt die mathematische Logik jedoch von vornherein aus. Als Grundlage aller logischen Betrachtungen merken wir uns deshalb die beiden unmittelbar einleuchtenden Prinzipien:

1. **Widerspruchsprinzip:**

   Keine Aussage ist zugleich unwahr und wahr.

2. **Prinzip vom ausgeschlossenen Dritten:**

   Jede Aussage muß entweder wahr oder unwahr sein.

Die Wortkombination: „Der Transistor läuft 100 Meter in 9 Sekunden bergauf" ist also in logischer Hinsicht **keine** Aussage.

3.2.1. Verknüpfungen der Aussagenlogik

Die Verknüpfungen der Aussagenlogik lassen sich umkehrbar eindeutig auf die Gesetzmäßigkeiten der Mengenalgebra abbilden (Abb. 3.3.). Ebenso wie wir in der Mengenalgebra mit den drei Grundbeziehungen „Durchschnitt", „Vereinigung" und „Komplementbildung" auskommen, genügen in der Aussagenlogik die drei Verknüpfungen **„Konjunktion", „Disjunktion"** und **„Negation".** Aus Gründen der Zweckmäßigkeit operieren wir daneben noch mit der **„Implikation"** und der **„Äquivalenz".** Die genannten Verknüpfungen werden in den folgenden Abschnitten näher erläutert. Rein formal stellen die Verknüpfungen der Aussagenlogik die gleichen mathematischen Anweisungen dar wie die Verknüpfungen der Mengenalgebra. Der Zusammenhang zwischen Mengenalgebra und Aussagenlogik kommt in der folgenden Tabelle 3.1 unmittelbar zum Ausdruck. Beachten Sie bitte die Ähnlichkeit der Verknüpfungszeichen.

	Mengenalgebra	**Aussagenalgebra**
Definierte Elemente	Mengen	Aussagen
Bivalenter Charakter	ε $\notin$	Wahr = 1 Falsch = 0
1. Verknüpfung Bezeichnung	$\bar{A}$ Komplementärmenge	$\bar{A}$ bzw. $\neg A$ Negation
2. Verknüpfung Bezeichnung	$A \cup B$ Vereinigung	$A \vee B$ Disjunktion
3. Verknüpfung Bezeichnung	$A \cap B$ Durchschnitt	$A \wedge B$ Konjunktion
4. Verknüpfung Bezeichnung	$\bar{A} \cup B$ (Siehe Abb. 3.11 a)	$A \Longrightarrow B$ bzw. $A \supset B$ Implikation
5. Verknüpfung Bezeichnung	$(A \cap B) \cup (\bar{A} \cap \bar{B})$ Äquivalenz	$A \Longleftrightarrow B$ Äquivalenz
1. Konstante	Universalmenge Ω	Tautologie (Immer wahr)
2. Konstante	Leere Menge $\emptyset$	Kontradiktion (Nie wahr)

Tabelle 3.1

Korrespondenzen zwischen Mengen- und Aussagenalgebra

3.2.1.1. Die Negation

Die Negation $\bar{A}$ bzw. $\neg A$ einer Aussage A ist genau dann wahr, wenn die Aussage A selbst falsch ist. Umgekehrt ist die Negation $\bar{A}$ genau dann falsch, wenn die Aussage A richtig ist. Wir betrachten dazu ein Beispiel:

Die Aussage A möge lauten:

A: $2 + 2 = 7$

Diese Behauptung ist offensichtlich falsch, ihr kommt also der Wahrheitswert 0 zu. Verneinen wir jedoch diese Aussage, dann erhalten wir eine richtige Behauptung:

$\bar{A}$: $2 + 2 \neq 7$

Wir sprechen: „2 und 2 ist **nicht** 7".

Somit entspricht die Negation in der mathematischen Logik der **sprachlichen Verneinung** einer Behauptung. Der Zusammenhang zwischen einer Aussage A und der Negation $\bar{A}$ dieser Aussage läßt sich anschaulich in Form einer sogenannten **Wahrheitstabelle** darstellen. Zu diesem Zwecke tragen wir in die Spalte unter der Aussage A die beiden möglichen Wahrheitswerte 0 und 1 ein. Die zweite Spalte für die Negation $\bar{A}$ enthält dann die Komplemente zu den Wahrheitswerten von A.

A	$\bar{A}$
0	1
1	0

Tabelle 3.2

Wahrheitstafel für die Negation

Aus Tabelle 3.2 entnehmen wir somit für das Rechnen mit Konstanten:

$$\boxed{\begin{aligned}\bar{1} &= 0\\ \bar{0} &= 1\end{aligned}} \qquad (36)$$

Besonders wichtig ist das „Prinzip der doppelten Verneinung". Wird eine vorliegende Aussage zweimal negiert, dann ist diese Aussage wiederum wahr. Das folgende Beispiel mag dieses **Prinzip der doppelten Verneinung** verdeutlichen.

Wir betrachten die Aussage

B: $2 + 2 = 4$

Diese Aussage, zweifach verneint, ergibt wiederum eine wahre Aussage, nämlich: „Die Behauptung, daß $2 + 2$ **nicht** 4 ist, ist **nicht** wahr". Für mehrfache Negationen gilt somit:

$$\boxed{\begin{aligned}\bar{\bar{1}} &= 1\\ \bar{\bar{0}} &= 0\end{aligned}} \qquad (37)$$

3.2.1.2. Die Disjunktion

Wenn wir im Sprachgebrauch Aussagen miteinander verknüpfen, benutzen wir häufig das Wort **Oder.** Die Disjunktion der mathematischen Logik kommt der Bedeutung dieses Wortes „Oder" sehr nahe, so daß wir auch von der „ODER-Verknüpfung" sprechen. Hierbei ist jedoch Vorsicht am Platze, wie sich gleich zeigen wird. Wenn wir im Sprachgebrauch das Wort „Oder" benutzen, meinen wir sehr häufig „Entweder-Oder". Diese Mehrdeutigkeit des Wortes „Oder" schaltet die mathematische Logik aus. Die Disjunktion ist so erklärt, daß der Wahrheitswert der beiden verknüpften Aussagen immer dann 1 wird, wenn die erste Aussage wahr ist **oder** die zweite Aussage wahr ist **oder** aber wenn **beide** Aussagen wahr sind.

Die Wahrheitstafel für die Disjunktion zeigt Tabelle 3.3.

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Tabelle 3.3

Wahrheitstafel der Disjunktion

Als Verknüpfungssymbol wählen wir das Zeichen „$\vee$", das aus dem lateinischen ‚vel = oder' abgeleitet wurde. Wir schreiben also:

$$\boxed{F = A \vee B} \qquad (38)$$

und sprechen: „F ist gleich A **oder** B".

Beispiel 15

Die beiden Aussagen

A: 2 ist kleiner als 5 und

B: Wasser ist naß

sind disjunktiv miteinander zu verknüpfen. Dabei ist so vorzugehen, daß alle in Tabelle 3.3 gezeigten Wahrheitskombinationen erfaßt werden. Bei der Lösung verfahren wir so, daß wir überall dort, wo eine 0 in der Tabelle steht, die Aussage negieren, d.h. verneinen. Wir erhalten damit die vier möglichen disjunktiven Verknüpfungen:

1. 2 ist **nicht** kleiner 7 **oder** Wasser ist **nicht** naß ist eine falsche Aussage.

   Formelmäßig: $\overline{A} \vee \overline{B} = 0$

2. 2 ist **nicht** kleiner 7 **oder** Wasser ist naß ist eine wahre Aussage.

   Formelmäßig: $\overline{A} \vee B = 1$

3. 2 ist kleiner 7 **oder** Wasser ist **nicht** naß ist eine wahre Aussage.

   Formelmäßig: $A \vee \overline{B} = 1$

4. 2 ist kleiner 7 **oder** Wasser ist naß ist eine wahre Aussage.

   Formelmäßig: $A \vee B = 1$

Weil in der Disjunktion entsprechend der eben gefundenen Lösung die zusammengesetzte Aussage auch dann wahr ist, wenn sowohl A als auch B gleichzeitig wahr sind, sprechen wir auch von der **Inklusiven-ODER-Verknüpfung.** Die vorab bereits angesprochene Wortwahl „Entweder-Oder" schließt die letztgenannte vierte Bedingung als falsche zusammengesetzte Aussage aus. Die zusammengesetzte Aussage wird in diesem Falle falsch. Eine derartig verschärfte ODER-Verknüpfung bezeichnen wir in der mathematischen Logik auch als **Exklusives-ODER.** Tabelle 3.4 zeigt die Wahrheitskombinationen für das Exklusive-ODER.

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Tabelle 3.4

Wahrheitstafel für das Exklusive-Oder

3.2.1.3. Die Konjunktion

Die Benutzung des Bindewortes „Und" in der Umgangssprache kommt der Bedeutung der Konjunktion in der mathematischen Logik sehr nahe. Wir sprechen deshalb auch von der **UND-Verknüpfung.** Als Verknüpfungszeichen wählen wir neben anderen Möglichkeiten das am häufigsten anzutreffende Zeichen „$\wedge$". Tabelle 3.5 erklärt die Konjunktion.

A	B	A $\wedge$ B
0	0	0
0	1	0
1	0	0
1	1	1

Tabelle 3.5

Wahrheitstafel für die Konjunktion

Die Konjunktion liefert also immer dann den Wahrheitswert 1, wenn alle in ihr vorkommenden Aussagen ebenfalls wahr sind. Sobald auch nur eine der verwendeten Aussagen den Wahrheitswert 0 hat, wird die Konjunktion ebenfalls 0.

Beispiel 16

Die beiden Aussagen

A: Das Gras ist grün sowie

B: Die Katze ist ein Tier

sind gemäß Tabelle 3.5 konjunktiv miteinander zu verknüpfen.

Die Lösung lautet:

1. Das Gras ist **nicht** grün **und** die Katze **kein** Tier ist eine falsche Aussage.

   Formelmäßig: $\overline{A} \wedge \overline{B} = 0$

2. Das Gras ist **nicht** grün **und** die Katze ein Tier ist eine falsche Aussage.

   Formelmäßig: $\overline{A} \wedge B = 0$

3. Das Gras ist grün **und** die Katze **kein** Tier ist eine falsche Aussage.

   Formelmäßig: $A \wedge \overline{B} = 0$

4. Das Gras ist grün **und** die Katze ein Tier ist eine wahre Aussage.

   Formelmäßig: $A \wedge B = 1$

Wie unzureichend die Sprache ist, mag das folgende Beispiel verdeutlichen. Es zeigt, daß im Sprachgebrauch häufig die Bedeutung des Wortes ‚Und' mit der Bedeutung des Wortes ‚Oder' verwechselt wird. Sicherlich kennen Sie das im Straßenverkehr übliche Hinweisschild: „Bei ROT und GELB halten". Entsprechend Tabelle 3.5 für die Konjunktion müßte demnach nur gehalten werden, wenn die Ampel ROT **und** GELB zugleich ist. Bei ROT allein

sowie bei GELB allein hingegen darf gefahren werden. Gerade das aber kann die Ampel nicht meinen wollen. Es müßte also richtiger heißen: „Bei ROT **oder** GELB halten".

Tabelle 3.3 für die Disjunktion zeigt, daß jetzt nur noch gefahren werden darf, wenn die Ampel nicht leuchtet oder aber wenn sie auf GRUN steht.

3.2.1.4. Die Implikation

Für die Implikation der Aussagenlogik schreiben wir

$$A \Longrightarrow B \tag{39}$$

und sprechen: „A impliziert B" oder auch: „Wenn A, dann B".

Wie Tabelle 3.6 zeigt, wird die Implikation $A \Longrightarrow B$ in nur einem einzigen Fall falsch, nämlich dann, wenn die Aussage A wahr ist und die Aussage B unwahr ist.

A	B	$A \Longrightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

Tabelle 3.6

Wahrheitstafel für die Implikation $A \Longrightarrow B$

Wir nennen in diesem Fall die Aussage A die **Voraussetzung** und die Aussage B die **Konsequenz.** Aus dieser Verabredung folgt, daß die Implikation nicht kommutativ ist; das will heißen, daß die beiden Aussagen A und B nicht vertauscht werden dürfen. So liefert z.B. die Implikation $A \Longleftarrow B$ ein anderes Ergebnis, wie Tabelle 3.7 es zeigt.

A	B	$A \Longleftarrow B$
0	0	1
0	1	0
1	0	1
1	1	1

Tabelle 3.7

Wahrheitstafel für die Implikation $A \Longleftarrow B$

Bei der Implikation muß die Suche nach geeigneten Redewendungen scheitern. Alle Hilfsmittel der Deutung, anstelle von „A impliziert B" zu sagen:

„Wenn A, so B"
oder „Aus A folgt B"
oder „Wenn A, dann B"
oder „A bringt B mit sich"

sind im Prinzip nur unbefriedigende Behelfe.

Die Implikation macht somit sehr deutlich, daß die Wörter der Umgangssprache nicht die legitime Deutung der logischen Symbole sind. Wir müssen uns stets vor Augen halten, daß die Fachsprache der mathematischen Logik und die Umgangssprache zwei völlig verschiedene Sprachen sind. Wir müssen die Implikation deshalb als das hinnehmen, was sie sein soll: Eine mathematisch äußerst zweckmäßige **Definition.** Nützliche Definitionen führen wir immer dann ein, wenn sich in einer Rechnung dadurch Vereinfachungen erzielen lassen. Genau das erlaubt die Implikation. Ein Blick auf die Tabellen 3.6 und 3.7 zeigt, daß wir allerdings die Implikation definitorisch ersetzen können. Die Tatsache, daß in Tabelle 3.6 nur in der dritten Zeile der Wahrheitswert 0 für die Implikation $A \Longrightarrow B$ erscheint, läßt sich auch über die Disjunktion erfassen:

$$A \Longrightarrow B = \overline{A} \vee B \tag{40}$$

Entsprechend gilt für die Implikation $A \Longleftarrow B$ nach Tabelle 3.7:

$$A \Longleftarrow B = A \vee \overline{B} \tag{41}$$

3.2.1.5. Die Äquivalenz

Als fünfte Verknüpfung der Aussagenlogik kennen wir noch die Äquivalenz. Wie Tabelle 3.8 zeigt, wird die Äquivalenz genau dann wahr, wenn A und B den **gleichen** Wahrheitswert haben. Für die Äquivalenz schreiben wir symbolisch

$$A \Longleftrightarrow B \tag{42}$$

und sprechen: „A äquivalent B" oder auch: „A genau dann, wenn B".

A	B	A⟺B
0	0	1
0	1	0
1	0	0
1	1	1

Tabelle 3.8

Wahrheitstafel für die Äquivalenz

Ebenso wie die Implikation ist die Äquivalenz eine zweckmäßige Definition. Wollten wir hier nach Redewendungen suchen, so würden wir keine passenden Kombinationen finden. Im Sprachgebrauch vermögen wir äquivalente Aussagen nicht von gleichen Aussagen zu unterscheiden. Daß wir allerdings die Äquivalenz über die drei Grundverknüpfungen: Negation, Disjunktion und Konjunktion definitorisch erfassen können, mag das folgende Beispiel verdeutlichen.

Beispiel 17

Die beiden Implikationen $A \rightarrow B$ und $A \leftarrow B$ sind konjunktiv miteinander zu verknüpfen. Daß das Ergebnis die Äquivalenz sein muß, ist anhand einer logischen Tabelle zu beweisen. Tabelle 3.9 zeigt die Lösung.

A	B	A→B	A←B	(A→B) ∧ (A←B)	A⟷B
0	0	1	1	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	1	1	1	1

Tabelle 3.9

Beweis für die Übereinstimmung

$$(A \rightarrow B) \wedge (A \leftarrow B) = A \longleftrightarrow B$$

In der Tat ergibt die konjunktive Verknüpfung der beiden Implikationen $A \Longrightarrow B$ sowie $A \Longleftarrow B$ unsere soeben definierte Äquivalenzrelation. Wir können demnach schreiben:

$$(A \Longrightarrow B) \wedge (A \Longleftarrow B) = A \Longleftrightarrow B \qquad (43)$$

Damit wird auch der Doppelpfeil als Verknüpfungssymbol sinnvoll. Für die Implikationen $A \Longrightarrow B$ sowie $A \Longleftarrow B$ können wir aber die Formeln (40) sowie (41) benutzen. Wir erhalten dann:

$$A \Longleftrightarrow B = (\bar{A} \vee B) \wedge (A \vee \bar{B}) \qquad (44)$$

Dieses Beispiel zeigt sehr schön, wie wir von Anfangsaussagen durch systematische Anwendung der Kalkülregeln zu immer neuen zusammengesetzten Aussagen gelangen können. Schließlich läßt sich die Äquivalenzrelation gemäß Tabelle 3.8 auch noch anders umschreiben. Die Tatsache, daß die Äquivalenz immer dann den Wahrheitswert 1 liefert, wenn die Aussagen beide gleich sind, läßt sich wie folgt umschreiben:

$$A \Longleftrightarrow B = (\bar{A} \wedge \bar{B}) \vee (A \wedge B) \qquad (45)$$

Durch Einsetzen der Formel (44) für $A \Longleftrightarrow B$ erhalten wir den folgenden Zusammenhang:

$$A \Longleftrightarrow B = (\bar{A} \vee B) \wedge (A \vee \bar{B}) = (\bar{A} \wedge \bar{B}) \vee (A \wedge B) \qquad (46)$$

3.2.2. Wichtige Gesetze der Aussagenlogik

Wie wir eingangs sagten, lassen sich die Regeln der Aussagenlogik umkehrbar eindeutig auf die Regeln der Mengenalgebra abbilden. Bei dieser Abbildung ersetzen wir die folgenden Zeichen:

das Durchschnittssymbol ‚∩' durch das Symbol ‚∧'
das Vereinigungssymbol ‚∪' durch das Symbol ‚∨'
die Universalmenge Ω durch den Wahrheitswert 1
die leere Menge ∅ durch den Wahrheitswert 0

Jetzt können praktisch alle im Abschnitt „Mengenalgebra" definierten Rechengesetze übernommen werden.

3.2.2.1. Kommutative Gesetze

Ebenso wie in der Mengenalgebra der Durchschnitt und die Vereinigung kommutativ sind, sind in der Aussagenlogik auch die Konjunktion und die Disjunktion kommutativ:

$$\begin{aligned} A \wedge B &= B \wedge A \\ A \vee B &= B \vee A \end{aligned} \qquad (47)$$

Diese Gesetze sind unmittelbar einleuchtend, wir wollen deshalb auf eine strenge Beweisführung verzichten.

A	B	C	$A \wedge B$	$A \wedge C$	$B \vee C$	$A \wedge (B \vee C)$	$(A \wedge B) \vee (A \wedge C)$
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	1	1	1

Tabelle 3.10

Beweis des ersten Distributivgesetzes: $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$

3.2.2.2. Distributive Gesetze

Die beiden distributiven Gesetze beschäftigen sich mit der Zusammenfassung von aussagenalgebraischen Ausdrücken durch Klammern; sie lauten:

$$\begin{aligned} A \wedge (B \vee C) &= (A \wedge B) \vee (A \wedge C) \\ A \vee (B \wedge C) &= (A \vee B) \wedge (A \vee C) \end{aligned} \qquad (48)$$

Am zweckmäßigsten beweisen wir diese beiden Gesetze durch Tabellen (vgl. Tabelle 3.10).

Der Beweis des zweiten Distributivgesetzes sei dem Leser überlassen. Es empfiehlt sich, hierfür das in Tabelle 3.10 gezeigte Schema zu benutzen.

3.2.2.3. Duale Aussagen

Wie in jedem Booleschen Verband läßt sich ein bestehender Sachverhalt durch zwei Aussagen formulieren. Orientierten wir uns bisher an dem Wahrheitswert 1 der zusammengesetzten Aussagen, so müssen wir jetzt umdenken. Liegt ein aussagenalgebraischer Ausdruck in Form einer logischen Tabelle vor, dann suchen wir die Zeilen heraus, für die die zusammengesetzte Aussage den Wahrheitswert 0 einnimmt. Als Beispiel wählen wir die Wahrheitstabelle für die Disjunktion:

A	B	$A \vee B = F$
0	0	0
0	1	1
1	0	1
1	1	1

Wahrheitstabelle für die Disjunktion

Die duale Aussage zu $F = A \vee B$ lautet:

$$\begin{aligned} F &= A \vee B \\ \bar{F} &= \bar{A} \wedge \bar{B} \end{aligned} \quad \text{duale Aussagen} \qquad (49)$$

Daß die duale Aussage $\bar{F} = \bar{A} \wedge \bar{B}$ den gleichen Sachverhalt wie die ursprüngliche Aussage $F = A \vee B$ erfaßt, ist leicht nachprüfbar. Tabelle 3.11 zeigt die Zusammenhänge.

A	B	$\bar{A}$	$\bar{B}$	$F = A \vee B$	$\bar{F} = \bar{A} \wedge \bar{B}$
0	0	1	1	0	1
0	1	1	0	1	0
1	0	0	1	1	0
1	1	0	0	1	0

Tabelle 3.11

Die zu $F = A \vee B$ duale Aussage lautet $\bar{F} = \bar{A} \wedge \bar{B}$

Die gleichen Überlegungen lassen uns die duale Aussage für die Konjunktion finden:

$$\begin{aligned} F &= A \wedge B \\ \overline{F} &= \overline{A} \wedge \overline{B} \end{aligned} \qquad (50)$$

Der tabellarische Beweis für diese Behauptung sei dem Leser überlassen. Es soll an dieser Stelle lediglich ein sprachliches Beispiel angeführt werden.

Die konjunktive Aussage möge lauten:

„Am Montag **und** am Donnerstag gehen wir zum Schwimmen".

Die duale Aussage lautet dann nach (50):

„Wenn **nicht** Montag ist **oder** wenn **nicht** Donnerstag ist, gehen wir **nicht** zum Schwimmen".

Die beiden Aussagen beschreiben offensichtlich den gleichen Sachverhalt: Nämlich es wird nur am Montag **und** am Donnerstag geschwommen. Der Formalismus zum Aufstellen dualer Beziehungen geht unmittelbar aus (49) und (50) hervor: Wenn in einer Gleichung oder algebraischen Identität eines Booleschen Verbandes alle Aussagen negiert werden und gleichzeitig das Verknüpfungszeichen „$\wedge$" durch „$\vee$" ersetzt wird und umgekehrt, dann entsteht eine duale Aussage. Als **duale Aussage** bezeichnen wir eine neue Gleichung, deren Richtigkeit genau wie in der ursprünglichen Gleichung von denselben Aussagen abhängt.

Natürlich gilt dieser Formalismus nicht nur für 2 Anfangsaussagen, sondern ganz allgemein für beliebig viele:

$$\begin{aligned} A \wedge B \wedge C \wedge \ldots \wedge X &= Z \\ \overline{A} \vee \overline{B} \vee \overline{C} \vee \ldots \vee \overline{X} &= \overline{Z} \end{aligned} \qquad (51)$$

Für disjunktiv verknüpfte Aussagen gilt demnach:

$$\begin{aligned} A \vee B \vee C \vee \ldots \vee X &= Z \\ \overline{A} \wedge \overline{B} \wedge \overline{C} \wedge \ldots \wedge \overline{X} &= \overline{Z} \end{aligned} \qquad (52)$$

Weiter gilt das Dualitätsprinzip auch für zusammengesetzte Aussagen. Wir können deshalb schreiben:

$$\begin{aligned} (A \wedge B) \vee (C \wedge D) &= F \\ (\overline{A} \vee \overline{B}) \wedge (\overline{C} \vee \overline{D}) &= \overline{F} \end{aligned} \qquad (53)$$

3.2.2.4. Die Gesetze von De Morgan

Aus dem Dualitätsprinzip heraus lassen sich die besonders für die Vereinfachung von aussagenalgebraischen Ausdrücken wichtigen De Morganschen Gesetze herleiten.

Wir betrachten zu diesem Zwecke die beiden Formeln:

$$\begin{aligned} A \wedge B &= F \\ \overline{A} \wedge \overline{B} &= \overline{F} \end{aligned}$$

Negieren wir beide Seiten der konjunktiven Grundgleichung $F = A \wedge B$, so entsteht:

$$\overline{A \wedge B} = \overline{F}$$

Da $\overline{F}$ immer sich selbst gleich ist, können offenbar die beiden Ausdrücke für $\overline{F}$ gleichgesetzt werden:

$$\overline{A} \vee \overline{B} = \overline{A \wedge B} \qquad (54)$$

Dieser Zusammenhang heißt das **Erste De Morgansche Gesetz.** Die Richtigkeit dieses Gesetzes beweisen wir wiederum durch eine Tabelle.

A	B	$\overline{A}$	$\overline{B}$	$A \wedge B$	$\overline{A \wedge B}$	$\overline{A} \vee \overline{B}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

Tabelle 3.12

Tabellarischer Beweis für das De Morgansche Gesetz

$$\overline{A} \vee \overline{B} = \overline{A \wedge B}$$

Das Zweite De Morgansche Gesetz finden wir über die beiden dualen Aussagen:

$$\begin{aligned} F &= A \vee B \\ \overline{F} &= \overline{A} \wedge \overline{B} \end{aligned}$$

Wir negieren die Grundgleichung $F = A \vee B$ und setzen mit der dualen Aussage $\overline{F} = \overline{A} \wedge \overline{B}$ gleich. Als Ergebnis erhalten wir das **Zweite De Morgansche Gesetz:**

$$\overline{A} \wedge \overline{B} = \overline{A \vee B} \qquad (55)$$

Nach dem in Tabelle 3.12 gezeigten Schema mag der Leser für Übungszwecke die Richtigkeit von (55) selbst beweisen.

Eine nochmalige Verneinung der beiden De Morganschen Gesetze führt auf die äußerst wichtigen Identitäten:

$$\boxed{\begin{aligned}\overline{\overline{A \vee B}} &= A \vee B = \overline{\overline{A} \wedge \overline{B}} \\ \overline{\overline{A \wedge B}} &= A \wedge B = \overline{\overline{A} \vee \overline{B}}\end{aligned}} \qquad (56)$$

Neben den Distributivgesetzen stellen die De Morganschen Gesetze die wichtigsten Regeln für die Vereinfachung von komplizierten aussagenalgebraischen Ausdrücken dar. Der Leser sollte sich aus diesem Grunde besonders diese Gesetze einprägen.

3.2.3. Zusammenstellung der Theoreme der Aussagenlogik

Zur einfacheren Handhabung seien an dieser Stelle noch einmal alle wichtigen Theoreme der Aussagenlogik angeführt. Entsprechend dem Dualitätsprinzip werden die Rechenregeln jeweils zu Paaren zusammengefaßt.

1. Grundgleichungen mit den Wahrheitswerten 0 und 1

a) Negation

$$\boxed{\overline{0} = 1} \qquad \boxed{\overline{1} = 0}$$

b) Konjunktion

$$\boxed{\begin{aligned}0 \wedge 0 &= 0 \\ 0 \wedge 1 &= 0 \\ 1 \wedge 0 &= 0 \\ 1 \wedge 1 &= 1\end{aligned}}$$

c) Disjunktion

$$\boxed{\begin{aligned}1 \vee 1 &= 1 \\ 1 \vee 0 &= 1 \\ 0 \vee 1 &= 1 \\ 0 \vee 0 &= 0\end{aligned}}$$

2. Grundgleichungen mit einer Aussage

a) Prinzip der doppelten Verneinung

$$\boxed{\overline{\overline{A}} = A}$$

b) Konjunktive Verknüpfung

$$\boxed{\begin{aligned}A \wedge 0 &= 0 \\ A \wedge 1 &= A \\ \overline{A} \wedge 0 &= 0 \\ \overline{A} \wedge 1 &= \overline{A}\end{aligned}}$$

c) Disjunktive Verknüpfung

$$\boxed{\begin{aligned}\overline{A} \vee 1 &= 1 \\ \overline{A} \vee 0 &= \overline{A} \\ A \vee 1 &= 1 \\ A \vee 0 &= A\end{aligned}}$$

d) Tautologiegesetze

$$\boxed{\begin{aligned}A \wedge A &= A \\ \overline{A} \wedge \overline{A} &= \overline{A}\end{aligned}} \qquad \boxed{\begin{aligned}\overline{A} \vee \overline{A} &= \overline{A} \\ A \vee A &= A\end{aligned}}$$

e) Satz vom ausgeschlossenen Widerspruch

$$\boxed{\begin{aligned}A \wedge \overline{A} &= 0 \\ \overline{A} \wedge A &= 0\end{aligned}} \qquad \boxed{\begin{aligned}\overline{A} \vee A &= 1 \\ A \vee \overline{A} &= 1\end{aligned}}$$

3. Grundgleichungen mit 2 Aussagen

a) Kommutative Gesetze

$$\boxed{A \wedge B = B \wedge A} \qquad \boxed{A \vee B = B \vee A}$$

b) De Morgansche Theoreme

$$\boxed{\begin{aligned}\overline{A \wedge B} &= \overline{A} \vee \overline{B} \\ A \wedge B &= \overline{\overline{A} \vee \overline{B}}\end{aligned}} \qquad \boxed{\begin{aligned}\overline{A \vee B} &= \overline{A} \wedge \overline{B} \\ A \vee B &= \overline{\overline{A} \wedge \overline{B}}\end{aligned}}$$

4. Grundgleichungen mit 3 Aussagen

a) Assoziative Gesetze

$$\boxed{(A \wedge B) \wedge C = A \wedge (B \wedge C)}$$

$$\boxed{(A \vee B) \vee C = A \vee (B \vee C)}$$

b) Distributive Gesetze

$$\boxed{\begin{aligned}A \wedge (B \vee C) &= (A \wedge B) \vee (A \wedge C) \\ A \vee (B \wedge C) &= (A \vee B) \vee (A \wedge C)\end{aligned}}$$

5. Absorptionsgesetze

$$\boxed{\begin{aligned}A \wedge (A \vee B) &= A \\ A \wedge (\overline{A} \vee B) &= A \wedge B\end{aligned}}$$

$$\boxed{\begin{aligned}A \vee (A \wedge B) &= A \\ A \vee (\overline{A} \wedge B) &= A \vee B\end{aligned}}$$

3.3. Das Wesen der Schaltalgebra

Wegen ihres zweiwertigen Charakters läßt sich die Schaltalgebra ebensogut wie die Aussagenlogik auf die Mengenalgebra umkehrbar eindeutig abbilden. Während jedoch Mengenalgebra und Aussagenlogik unabhängig von irgendwelchen technischen Anwendungen existieren, erlaubt die Kenntnis der Schaltalgebra die Lösung umfangreicher technischer Probleme, sogenannter „digitaler Schaltkreise". Unsere heutigen digitalen Schaltkreise sind dadurch gekennzeichnet, daß in ihnen nur mit Schaltern gearbeitet wird. Während der zweiwertige Charakter in der Aussagenlogik durch die Wahrheitswerte „wahr" und „falsch" gegeben ist, liegt er in der Schaltalgebra in Form eines geschlossenen oder geöffneten Kontaktes vor. Ein Schalter kann demnach nur zwei Betriebszustände einnehmen; wir sagen kurz: „Der Schalter hat einen **binären** Charakter". Der binäre Charakter ergibt sich zwangsläufig dadurch, daß an einem geöffneten Kontakt die Betriebsspannung zu messen ist, während am geschlossenen Kontakt keine Spannung abfällt. Es werden also nur einfachste Auswahlentscheidungen zwischen dem Vorhandensein bzw. Nichtvorhandensein einer physikalischen Größe getroffen. Diese Auswahlentscheidungen bezeichnen wir in der Schaltalgebra als **Binärentscheidungen.** Eine Binärentscheidung wird getroffen zwischen 2 **Binärwerten.** Die Binärwerte bezeichnen wir mit 0 und 1, häufig auch mit 0 und L.

Wir ordnen nun diese Binärwerte unseren bistabilen Schaltelementen zu: **Die Tatsache, daß ein Schalter geschlossen ist, erfaßt der Binärwert 1; der offene Kontakt hingegen wird charakterisiert durch den Binärwert 0.**

Wenn wir so verfahren, lassen sich alle in der Aussagenlogik behandelten Verknüpfungen einfach durch Kontaktnetzwerke nachbilden. Wir werden sehen, daß die Konjunktion dann der 0 Reihenschaltung von Arbeitskontakten entspricht, die Disjunktion der Parallelschaltung von Arbeitskontakten. Die Negation schließlich ist gegeben durch die Umwandlung eines offenen Kontaktes in einen geschlossenen und umgekehrt. Tabelle 3.13 stellt die wichtigsten Begriffe der Schaltalgebra denen der Aussagenlogik gegenüber.

	Schaltalgebra	**Aussagenlogik**
Objekte	bistabile Schalter	Aussagen
Binärcharakter	Schalter geschlossen Schalter geöffnet	Wahr Falsch
1. Verknüpfung Bezeichnung	$A \cdot B$ Reihenschaltung von Arbeitskontakten	$A \wedge B$ Konjunktion
2. Verknüpfung Bezeichnung	$A + B$ Parallelschaltung von Arbeitskontakten	$A \vee B$ Disjunktion
3. Verknüpfung Bezeichnung	$\bar{A}$ Ruhekontakt	$\bar{A}$ Negation
1. Konstante	Leerlauf — ständig offener Kontakt	Kontradiktion (Immer falsch)
2. Konstante	Kurzschluß — ständig geschlossener Kontakt	Tautologie (Immer wahr)

Tabelle 3.13

Korrespondenzbeziehungen zwischen Schaltalgebra und Aussagenlogik

Wir hatten bereits bei den Ausführungen über die Mengenalgebra und Aussagenlogik gesehen, daß mit Mengen und Aussagen sogar gerechnet werden kann. Auch die Schaltalgebra stellt einen derartigen **Kalkül** zur Berechnung von Schaltnetzwerken dar. Da die Schaltalgebra auf den Anwender zugeschnitten ist, werden allerdings die Verknüpfungszeichen an die arithmetischen Rechenoperationen angepaßt. Bei der Berechnung umfangreicher Kontaktnetzwerke interessiert den Anwender weniger, ob er es mit Reihen- oder Parallelschaltungen zu tun hat, sondern er will problemlos rechnen.

Weil die meisten schaltalgebraischen Verknüpfungen die gleichen arithmetischen Anweisungen darstellen wie in der normalen Algebra, verwenden wir in der Schaltalgebra auch die Zeichen „ · " und „ + ". So wird aus dem „∧"-Zeichen für die Konjunktion das **schaltalgebraische Produkt,** während aus dem „∨"-Zeichen für die Disjunktion die **schaltalgebraische Summe** wird (vgl. hierzu Tabelle 3.1).

Es gilt demnach die Abbildung:

A ∧ B	⟺	A · B
Konjunktion	⟺	Produkt der Schaltalgebra
A ∨ B	⟺	A + B
Disjunktion	⟺	Summe der Schaltalgebra
$\bar{A}$	⟺	$\bar{A}$
Negation	⟺	Komplementbildung, Inversion

Nach diesen einführenden Betrachtungen können wir sagen, daß die Schaltalgebra sich auf alle Systeme anwenden läßt, die aus Bausteinen mit nur 2 möglichen stabilen Zuständen aufgebaut sind. Ob ein System aus Schaltern, Relaiskontakten, pneumatischen Schaltern, elektronischen Schaltern oder Magnetkernen aufgebaut ist, soll uns ab jetzt nicht mehr interessieren. Wir lösen uns aus diesem Grunde vollends von der technischen Realisierung derartiger Digitalsysteme und beschränken uns auf die Darstellung der **Schaltfunktionen** durch Symbole. Was Schaltfunktionen sind, wollen wir im nächsten Abschnitt kennenlernen.

3.3.1. Schaltfunktionen

Jedes technische System, das nur mit den beiden Binärwerten 0 und 1 arbeitet, bezeichnen wir als **binäres Digitalsystem.** Derartige Systeme haben in der Regel mehrere Eingänge, in die Binärwerte eingegeben werden, und einen oder mehrere Ausgänge. An den Ausgängen lassen sich nun entsprechend einer vorgegebenen Programmierung wiederum nur unsere Binärwerte 0 und 1 abnehmen. Wie die Kombinationen der Eingangsbinärwerte an den Ausgang weitergegeben oder nicht weitergegeben werden, hängt von dem Schaltnetzwerk selbst ab. Es läßt sich jedoch allgemein sagen: Die binären Ausgangskombinationen hängen auf jeden Fall von den Eingangskombinationen ab. Die folgende Abb. 3.20 zeigt Ihnen ein derartiges binäres Digitalsystem mit 2 Eingängen und 2 Ausgängen.

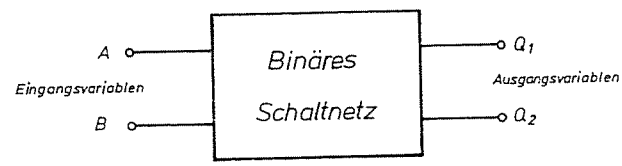


Abb. 3.20 — Binäres Schaltnetz

Die Tatsache, daß die Ausgangskombinationen von den Eingangskombinationen abhängen, beschreiben wir so: „Zwischen den Ausgangskombinationen und den Eingangskombinationen besteht ein **funktioneller** Zusammenhang.

Allgemein halten wir in der Mathematik einen derartigen Sachverhalt durch die Schreibweise

$$y = f(x) \tag{57}$$

fest. Wir meinen damit, daß eine bestimmte Variable y von einer vorgegebenen Variablen x abhängt. Der Ausdruck **Variablen x, y** deutet lediglich an, daß wir es hier mit Stellvertretern für beliebige Werte zu tun haben. Weil y von x abhängig ist, bezeichnen wir y als **abhängige** und x als **unabhängige Variable.**

Bezogen auf unser Digitalsystem in Abb. 3.20 können wir jetzt sagen, daß jeder Ausgang Q_1/Q_2 eine **Funktion** der Eingänge A/B ist. Wir schreiben:

$$\begin{aligned} Q_1 &= f(A, B) \\ Q_2 &= f(A, B) \end{aligned} \tag{58}$$

Wir sprechen: „Die Ausgangsvariable Q_1 ist eine Funktion der Eingangsvariablen A und B. Q_2 ist eine Funktion der Eingangsvariablen A und B".

Hierbei wollen wir uns stets vor Augen halten, daß alle vorkommenden Variablen — also Eingangs- und Ausgangsvariablen — nur der beiden Binärwerte 0 und 1 fähig sind.

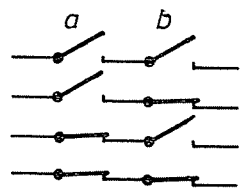
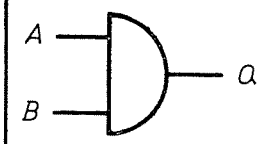
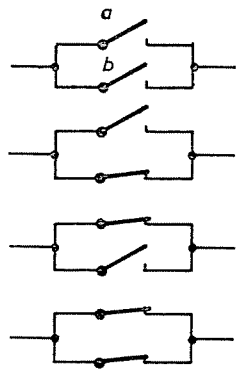
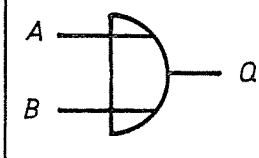
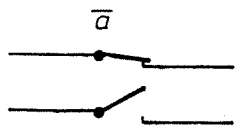
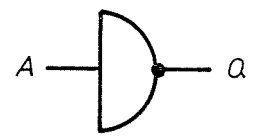
Logische Tabelle	Kontaktstellungen	Symbol	Schaltfunktion Bezeichnung
A B Q 0 0 0 0 1 0 1 0 0 1 1 1			$Q = A \cdot B$ UND
A B Q 0 0 0 0 1 1 1 0 1 1 1 1			$Q = A + B$ ODER
A Q 0 1 1 0			$Q = \overline{A}$ NICHT INVERSION

Abb. 3.21 — Die drei Grundfunktionen UND, ODER, NICHT

3.3.1.1. Grundfunktionen

Es zeigt sich, daß alle noch so komplizierten Digitalschaltungen nach Abb. 3.20 sich aus drei elementaren Grundfunktionen zusammensetzen lassen. Diese 3 Grundfunktionen sind die Funktionen UND, ODER, NICHT. Abb. 3.21 zeigt diese 3 Grundfunktionen mit ihren Schaltsymbolen nach DIN 40700, Blatt 14.

In der ersten Spalte der Abb. 3.21 finden Sie die logischen Tabellen für die 3 Grundfunktionen UND, ODER, NICHT. Abb. 3.21 zeigt prägen der Schaltsymbole in Spalte 3 zu erleichtern, sehen Sie in der zweiten Spalte die äquivalenten Kontaktschaltungen. Die Schaltfunktionen und deren Bezeichnung finden Sie in der vierten Spalte.

3.3.1.2. NAND und NOR als Universalfunktionen

Unsere heutigen Digitalsysteme in integrierter Schaltkreistechnik sind dadurch gekennzeichnet, daß fast ausschließlich mit den sogenannten NAND- bzw. NOR-Bausteinen gearbeitet wird. Diese Bausteine bringen sowohl dem Hersteller als auch dem Anwender Vorteile. Sie verfügen jeweils über aktive Transistoren, so daß die Binärsignale in jedem Baustein regeneriert werden. Es zeigt sich nämlich, daß mit nur 2 Funktionen im Prinzip alle noch so komplizierten digitalen Schaltkreissysteme realisiert werden können. Und zwar benötigen wir entweder die

UND-Funktion mit der NICHT-Funktion

oder die

ODER-Funktion mit der NICHT-Funktion.

Werden UND-Funktion und NICHT-Funktion in einem Baustein zusammengezogen, dann sprechen wir von der NICHT-UND- oder englisch von der NAND-Funktion (NOT-**AND** = NAND).

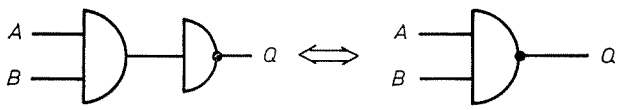
Dementsprechend bedeutet das Wort ‚NOR', daß wir es mit einem NICHT-ODER-Baustein

(NOT-OR = NOR) zu tun haben. Wenn es uns gelingt, mit nur der NAND- oder mit nur der NOR-Funktion die 3 Grundfunktionen UND, ODER, NICHT nachzubilden, können wir im Prinzip mit nur einer dieser beiden Funktionen auskommen. Das uns dieses gelingt, zeigen die folgenden Ausführungen.

Abb. 3.22 zeigt Ihnen zunächst einen NAND-Baustein. Sie erkennen, daß dieser NAND-Baustein zusammengesetzt ist aus der UND-Funktion mit anschließender Inversion. Im Schaltsymbol kommt die Inversion dadurch zum Ausdruck, daß wir an die Peripherie des Halbkreises einen schwarzen Negationspunkt zeichnen.

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

a)



b)

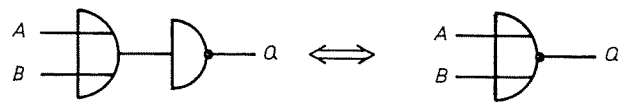
Abb. 3.22 — Die NAND-Funktion

a) Funktionstabelle
b) zusammengesetztes und vereinfachtes Schaltsymbol

In gleicher Weise kennzeichnen wir die NOR-Funktion. Abb. 3.23 zeigt die NOR-Funktion mit dem Schaltsymbol und der Funktionstabelle.

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

a)



b)

Abb. 3.23 — Die NOR-Funktion

a) Funktionstabelle
b) zusammengesetztes und vereinfachtes Schaltsymbol

Wir stellen uns jetzt vor, daß wir nur NAND-Glieder mit jeweils 2 Eingängen zur Verfügung haben. Mit Hilfe dieser NAND-Gatter wollen wir die 3 Grundfunktionen UND, ODER, NICHT nachbilden. Dabei helfen uns die bereits aus der Aussagenlogik und Mengenalgebra bekannten Gesetze.

Um die Inversion durchzuführen, benötigen wir das Tautologiegesetz $A \cdot A = A$. Wir verbinden die beiden Eingänge des NAND-Gatters und erhalten bereits so die gewünschte Negation. Abb. 3.24 a) zeigt die Verfahrensweise.

Abb. 3.24 b) zeigt Ihnen, wie die UND-Funktion durch NAND-Gatter zu realisieren ist. Wir benötigen ein NAND-Gatter für $\overline{A \cdot B}$ und ein weiteres Gatter für die anschließende Inversion

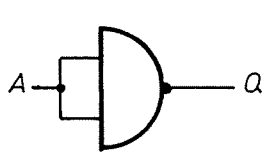
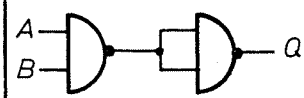
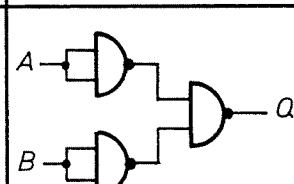
Nachbildung	NICHT durch NAND	UND durch NAND	ODER durch NAND
Schaltbild			
Funktion	$Q = \overline{A \cdot A} = \overline{A}$	$Q = \overline{\overline{A \cdot B}} = A \cdot B$	$Q = \overline{\overline{A} \cdot \overline{B}} = A + B$
Gesetz	Tautologie	Doppelte Negation	De Morgan
	a)	b)	c)

Abb. 3.24 — Nachbildung der Grundfunktionen durch NAND

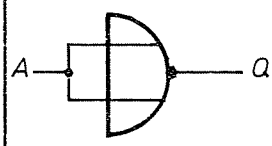
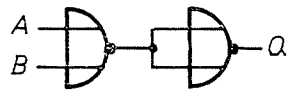
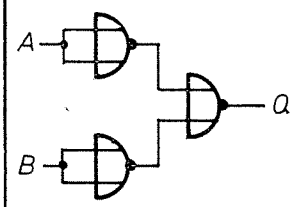
Nachbildung	NICHT durch NOR	ODER durch NOR	UND durch NOR
Schaltbild			
Funktion	$Q = \overline{A + A} = \overline{A}$	$Q = \overline{\overline{A + B}} = A + B$	$Q = \overline{\overline{A} + \overline{B}} = A \cdot B$
Gesetz	Tautologie	Doppelte Negation	De Morgan
	a)	b)	c)

Abb. 3.25 — Nachbildung der Grundfunktionen durch NOR

$\overline{\overline{A \cdot B}} = A \cdot B$. Hier wurde das „Prinzip der doppelten Verneinung" angewendet.

Zur Realisierung der ODER-Funktion über NAND-Gatter benötigen wir das De Morgansche Gesetz:

$$A + B = \overline{\overline{A} \cdot \overline{B}}.$$

Sie sehen, daß der Aufwand bereits 3 NAND-Gatter beträgt. Desgleichen können wir die Grundfunktionen UND, ODER, NICHT auch durch NOR-Gatter nachbilden. Die einfache Komplementbildung ist ohne weiteres verständlich; sie erkennen sie in Abb. 3.25 a).

Die ODER-Funktion erreichen wir nach dem „Prinzip der doppelten Verneinung". Wir brauchen zu diesem Zwecke lediglich die NOR-Funktion $\overline{A + B}$ zu komplementieren: $\overline{\overline{A + B}} = A + B$. Abb. 3.25 b) zeigt das Verfahren.

Bei der Nachbildung der UND-Funktion durch NOR hilft uns das De Morgansche Gesetz: $A \cdot B = \overline{\overline{A} + \overline{B}}$. Der Aufwand beträgt 3 Gatter vom Typ NOR, wie Abb. 3.25 c) es zeigt.

Wenn Systeme lediglich mit NAND-Bausteinen realisiert werden, sprechen wir kurz von der „NAND-Technik". Dementsprechend heißen Systeme, die nur mit NOR-Gattern arbeiten, auch Systeme in „NOR-Technik". Aus diesem Grunde nennen wir NAND und NOR die **Universalfunktionen** der digitalen Schaltkreissysteme.

3.3.2. Rangordnung der Verknüpfungszeichen

Genau wie in der normalen Algebra kommen den Verknüpfungszeichen schaltalgebraischer Funktionen ganz bestimmte Prioritäten zu. Ebenso wie in der Algebra das „ · "-Zeichen mehr bindet als das „ + "-Zeichen, genießt die Operation UND in der Schaltalgebra Vorrangstellung vor der Operation ODER. Weiter kommt einem Negationsstrich $\overline{A + B}$ über mehrere Variablen die gleiche Bedeutung zu wie einer Klammer in der normalen Algebra: $\overline{A + B} = \overline{(A + B)}$.

Aus dieser Kenntnis heraus können wir uns also die Klammern um ein schaltalgebraisches Produkt sowie um eine Negation ersparen:

$$\boxed{\begin{aligned} \overline{(A \cdot B)} &= \overline{A} \cdot \overline{B} = \overline{A}\,\overline{B} \\ \overline{(A + B)} &= \overline{A} + \overline{B} \end{aligned}} \qquad (59)$$

Weiter darf der Punkt für die „Multiplikation" auch weggelassen werden.

3.3.3. Die wichtigsten Rechenregeln der Schaltalgebra

In diesem Abschnitt wollen wir sämtliche Rechenregeln der Schaltalgebra nochmals zusammenhängend darstellen. Daß ebenso wie in der Mengenalgebra und in der Aussagenlogik die Assoziativgesetze, die Distributivgesetze und die Kommutativgesetze gelten, dürfte selbst-

verständlich sein. Weiter gelten nach wie vor das Dualitätsprinzip sowie die De Morganschen Gesetze. Entsprechend dem Dualitätsprinzip werden die Rechenregeln zu Paaren zusammengefaßt.

1. Grundgleichungen mit den Binärwerten 0 und 1

a) Inversion

$$\bar{0} = 1 \qquad \bar{1} = 0$$

b) UND-Funktion

$$0 \cdot 0 = 0$$
$$0 \cdot 1 = 0$$
$$1 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

c) ODER-Funktion

$$1 + 1 = 1$$
$$1 + 0 = 1$$
$$0 + 1 = 1$$
$$0 + 0 = 0$$

2. Grundgleichungen mit einer Variablen

a) Prinzip der doppelten Verneinung

$$\overline{\bar{A}} = A$$

b) UND-Funktion

$$A \cdot 0 = 0$$
$$A \cdot 1 = A$$
$$\bar{A} \cdot 0 = 0$$
$$\bar{A} \cdot 1 = \bar{A}$$

c) ODER-Funktion

$$\bar{A} + 1 = 1$$
$$\bar{A} + 0 = \bar{A}$$
$$A + 1 = 1$$
$$A + 0 = A$$

d) Tautologiegesetze

$$A \cdot A = A$$
$$\bar{A} \cdot \bar{A} = \bar{A}$$

$$\bar{A} + \bar{A} = \bar{A}$$
$$A + A = A$$

e) Satz vom ausgeschlossenen Widerspruch

$$A \cdot \bar{A} = 0$$
$$\bar{A} \cdot A = 0$$

$$\bar{A} + A = 1$$
$$A + \bar{A} = 1$$

3. Grundgleichungen mit 2 Variablen

a) Kommutative Gesetze

$$A \cdot B = B \cdot A \qquad A + B = B + A$$

b) De Morgansche Theoreme

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$
$$A \cdot B = \overline{\bar{A} + \bar{B}}$$

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$
$$A + B = \overline{\bar{A} \cdot \bar{B}}$$

4. Grundgleichungen mit 3 Variablen

a) Assoziative Gesetze

$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

$$(A + B) + C = A + (B + C)$$

b) Distributive Gesetze

$$A \cdot (B + C) = AB + AC$$

$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

5. Absorptionsgesetze

$$A \cdot (A + B) = A$$
$$A \cdot (\bar{A} + B) = A \cdot B$$

$$A + (A \cdot B) = A$$
$$A + (\bar{A} \cdot B) = A + B$$

3.4. Die Boolesche Algebra

Während die Mengenalgebra um die Jahrhundertwende von dem Hallenser Mathematiker Georg Cantor begründet wurde und die Schaltalgebra zurückgeht auf die Erkenntnisse, die Shannon 1938 als Erster formulierte, entwickelte bereits zu Beginn des 19ten Jahrhunderts der englische Mathematiker Georg Boole einen abstrakten Kalkül, der losgelöst ist von allen gegenständlichen Verwendungen. Georg Boole, der in der Zeit von 1815 bis 1864 lebte, schuf einen abstrakten mathematischen Kalkül, der heute als **Boolesche Algebra** bezeichnet wird. Die Boolesche Algebra ist eine Algebra, bei der die Variablen nur zwei Werte 0 und 1 annehmen können. Während jedoch in der Mengenalgebra mit Mengen operiert wird und in der Aussagenalgebra mit Aussagen, kommt den Objekten 0 und 1 der Booleschen Algebra keinerlei Bedeutung zu. Die Frage nach der sprachlichen Bedeutung der Booleschen Variablen 0

und 1 ist also sinnlos. Die Boolesche Algebra verzichtet auf jede technische Anwendung und beweist ihre Gesetze streng mathematisch. In der Booleschen Algebra sind nur 3 Verknüpfungen definiert, nämlich die Komplementierung, die Addition und die Multiplikation.

Auf die Beweisführung für die Booleschen Verknüpfungsregeln kann verzichtet werden, da ihnen im Grunde genommen die gleiche Bedeutung zukommt wie in der Aussagenlogik und in der Mengenalgebra. Weil die Boolesche Algebra die Vorgängerin aller anderen Algebren mit nur 2 Objekten ist, bezeichnen wir derartige Algebren mit dem Oberbegriff **Boolesche Verbände** oder **Boolesche Algebren.**

In der folgenden Tabelle 3.14 stellen wir abschließend alle Booleschen Algebren einander gegenüber. Der Vollständigkeit halber haben wir auch die hier nicht behandelte Ereignisalgebra mit hineingenommen. Die Ereignisalgebra ist besonders von Bedeutung in der Wahrscheinlichkeitsrechnung, in der mathematischen Statistik und in der Informationstheorie. In der letzten Zeile der Tabelle finden Sie noch die Hauptanwendungsgebiete für die einzelnen Algebren.

Boolesche Algebra

	Boolesche Algebra	Schaltalgebra	Mengenalgebra	Ereignisalgebra	Aussagenalgebra
Objekte	ohne Bedeutung	Bistabile Schaltelemente	Mengen	Ereignisse	Aussagen
Binärer Charakter	0 — 1	geschlossen — offen	ε ∉	Ereignis tritt ein Ereignis tritt nicht ein	Wahr Falsch
1. Verknüpfung Bezeichnung	A · B Boolesches Produkt	A · B Reihenschaltung	A ∩ B Durchschnitt	A ⊓ B UND-Ereignis	A ∧ B Konjunktion
2. Verknüpfung Bezeichnung	A + B Boolesche Summe	A + B Parallelschaltung	A ∪ B Vereinigung	A ⊔ B ODER-Ereignis	A ∨ B Disjunktion
3. Verknüpfung Bezeichnung	$\bar{A}$ Boolesche Negation	$\bar{A}$ Ruhekontakt	$\bar{A}$ Komplementärmenge	$\bar{E}$ (E') Gegenereignis	$\bar{A}$ (¬ A) Negation
1. Konstante	0	Leerlauf, ständig offener Kontakt	Leere Menge ø	Unmögliches Ereignis	Kontradiktion Niemals wahr
2. Konstante	1	Kurzschluß, ständig geschlossener Kontakt	Universalmenge Ω	Sicheres Ereignis	Tautologie Immer wahr
Anwendungs- gebiete	Reine Mathematik	Logische Schaltungen EDV	Matematische Grundlagenforschung Schulmathematik	Wahrscheinlichkeits- rechnung, Informa- tionstheorie, Statistik	Logik und Logistik

Tabelle 3.14