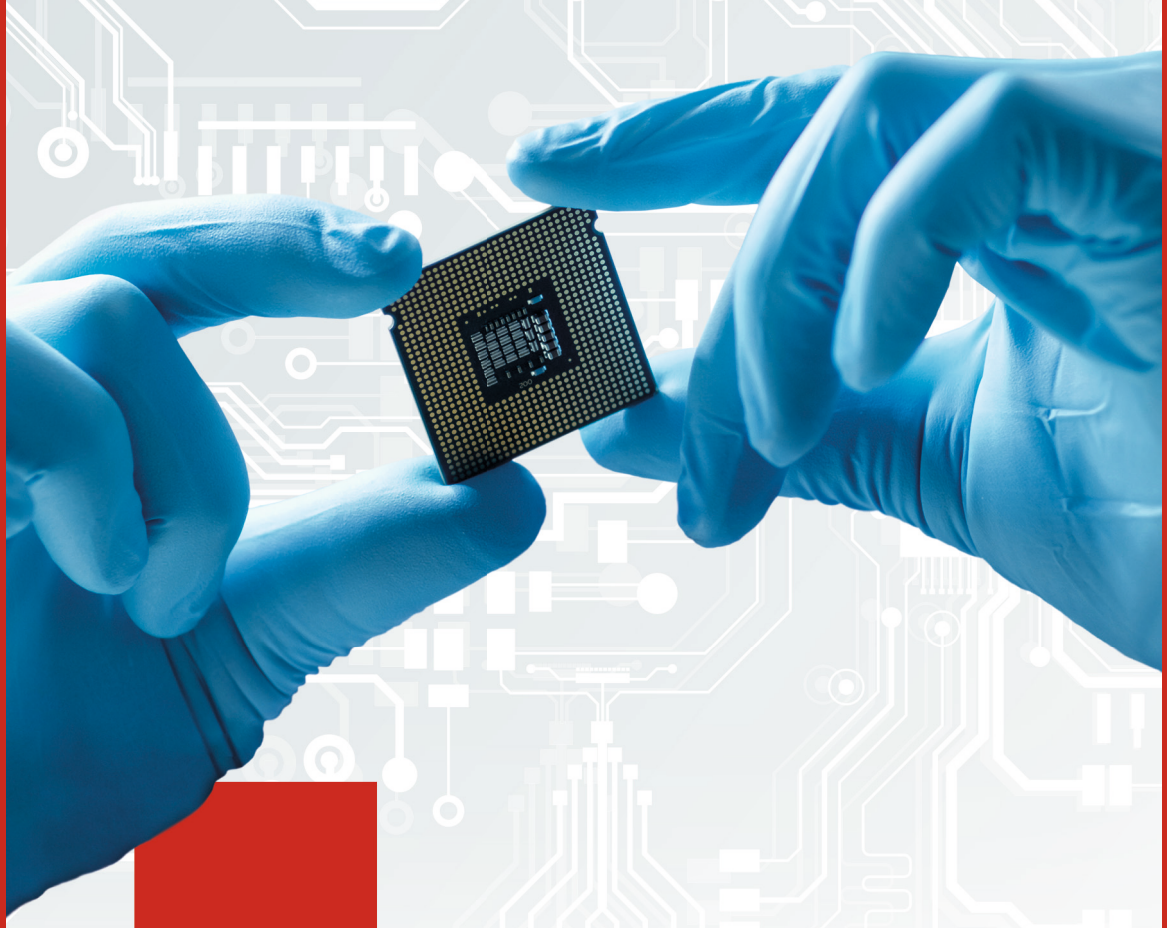


7. Auflage

Dirk W. **HOFFMANN**

GRUNDLAGEN DER TECHNISCHEN INFORMATIK

Im Internet: **Lösungen zu den Übungsaufgaben**

HANSER

Hoffmann

Grundlagen der Technischen Informatik

Dirk W. Hoffmann

Grundlagen der Technischen Informatik

7., aktualisierte Auflage

HANSER

Prof. Dr. Dirk W. Hoffmann
Fakultät für Informatik und Wirtschaftsinformatik,
Hochschule Karlsruhe – Technik und Wirtschaft

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Lektorat: Brigitte Bauer-Schiewek
Satz: Dirk W. Hoffmann
Titelmotiv: © shutterstock.com/Olga Zhurba und Gorodenkoff
Covergestaltung: Max Kostopoulos
Coverkonzept: Marc Müller-Bremer, www.rebranding.de, München
Druck und Bindung: Beltz Grafische Betriebe, Bad Langensalza
Printed in Germany

Print-ISBN: 978-3-446-47779-7
E-Book-ISBN: 978-3-446-47813-8

Vorwort

Die Computertechnik hat in wenigen Jahrzehnten eine Entwicklung vollzogen, die in ihrer Geschwindigkeit und Intensität einzigartig ist. Setzten sich die ersten Computer noch aus vergleichsweise wenigen Schaltkreisen zusammen, so verrichten in jedem modernen Arbeitsplatzrechner, Tablet-PC oder Smartphone Abermillionen von Transistoren ihren Dienst und führen in jeder Sekunde Milliarden von Berechnungen aus. Doch so rasant die Entwicklung der letzten Jahrzehnte auch war: Vergleichen wir die Maschinen der Pionierzeit mit unseren modernen Rechenboliden, so lassen sich eine Reihe von Grundprinzipien identifizieren, die sich im Laufe der Zeit zwar weiterentwickelt, aber im Kern nicht verändert haben. Diese Grundprinzipien, zusammen mit ihren modernen Ausprägungen, formen das Gebiet der technischen Informatik und sind Gegenstand des vorliegenden Buchs.

Geschrieben habe ich das Buch für Bachelor-Studenten der Fachrichtungen Informatik, Elektrotechnik, Informationstechnik und verwandter Studiengänge. Inhaltlich habe ich mich dabei an den typischen Lehrinhalten orientiert, die im Grundstudium an Hochschulen und Universitäten vermittelt werden. Neben dem Grundlagenwissen aus den Gebieten der Halbleitertechnik, der Zahlendarstellung und der booleschen Algebra werden die Entwurfsprinzipien kombinatorischer und sequenzieller Hardware-Komponenten bis hin zur Beschreibung moderner Prozessor- und Speicherarchitekturen vermittelt. Damit spannt das Buch den Bogen von den mathematischen Grundlagen digitaler Schaltelemente bis hin zu den ausgefeilten Hardware-Optimierungen moderner Hochleistungscomputer.

Es ist mir ein besonderes Anliegen, den Stoff anwendungsorientiert und didaktisch ansprechend zu vermitteln. Damit das Buch sowohl vorlesungsbegleitend als auch zum Selbststudium eingesetzt werden kann, werden die Lehrinhalte aller Kapitel durch zahlreiche Übungsaufgaben komplementiert. Des Weiteren habe ich etliche Anwendungsbezüge mit aufgenommen, um eine enge Verzahnung zwischen Theorie und Praxis zu erreichen.

Seit dem Erscheinen der letzten Auflage habe ich wieder zahlreiche Zuschriften erhalten, über die ich mich sehr gefreut habe. Namentlich bedanken möchte ich mich bei Herrn Prof. Dr. Michael Wiehl für wertvolle Hinweise im Bereich der MOS-Schaltungstechnik. Inzwischen erscheinen die *Grundlagen der technischen Informatik* in der siebten Auflage, und ich bin weiterhin jedem aufmerksamen Leser für Hinweise zu Verbesserungsmöglichkeiten oder Fehlern dankbar.

Karlsruhe, im April 2023 Dirk W. Hoffmann

Symbolwegweiser


Definition


Satz, Lemma, Korollar


Leichte Übungsaufgabe


Mittelschwere Übungsaufgabe


Schwere Übungsaufgabe

Lösungen zu den Übungsaufgaben

In wenigen Schritten erhalten Sie die Lösungen zu den Übungsaufgaben:

1. Gehen Sie auf die Seite www.dirkwhoffmann.de/TI
2. Geben Sie den neben der Aufgabe abgedruckten Webcode ein
3. Die Musterlösung wird als PDF-Dokument angezeigt

Inhaltsverzeichnis

1 Einführung

„The first microprocessor only had 22 hundred transistors. We are looking at something a million times that complex in the next generations – a billion transistors. What that gives us in the way of flexibility to design products is phenomenal.“

Gordon E. Moore, Intel Corporation

1.1 Was ist technische Informatik?

Blicken wir auf die Entwicklung der letzten hundert Jahre zurück, so hat keine andere technische Innovation unser Leben mehr verändert als die Erfindung des Computers, wie wir ihn heute kennen. Die Geschwindigkeit, mit der die digitale Revolution immer größere Bereiche unseres täglichen Lebens erobert und umgestaltet hat, ist nicht nur in der Retrospektive atemberaubend. Die Auswirkungen sind heute tief bis in unser kulturelles und gesellschaftliches Leben zu spüren. Ob wir wollen oder nicht: Wir stehen heute an der Schwelle des *ubiquitären Computerzeitalters* und haben sie in manchen Bereichen auch schon überschritten. Mit der Fortsetzung der kontinuierlich voranschreitenden Miniaturisierung und der zunehmenden Vernetzung verschiedenster Geräte ist der Computer von morgen allgegenwärtig und in vielen Fällen nicht einmal mehr als solcher zu erkennen.

Hand in Hand mit der sich rasant entwickelnden Computertechnik wuchs gleichermaßen die Bedeutung der Informatik, die sich in kürzester Zeit von einer Nischendisziplin zu einer eigenständigen Wissenschaft entwickeln konnte (vgl. Abbildung 1.1). Eine ihrer Kernsäulen ist die *technische Informatik*, die sich grob gesprochen mit dem *Entwurf, der logischen Struktur und der technischen Realisierung von Computer-Hardware* beschäftigt.

Ausgehend von der elementaren Hardware-Komponente des *Logikgatters* beschäftigt sich die technische Informatik mit der Konstruktion

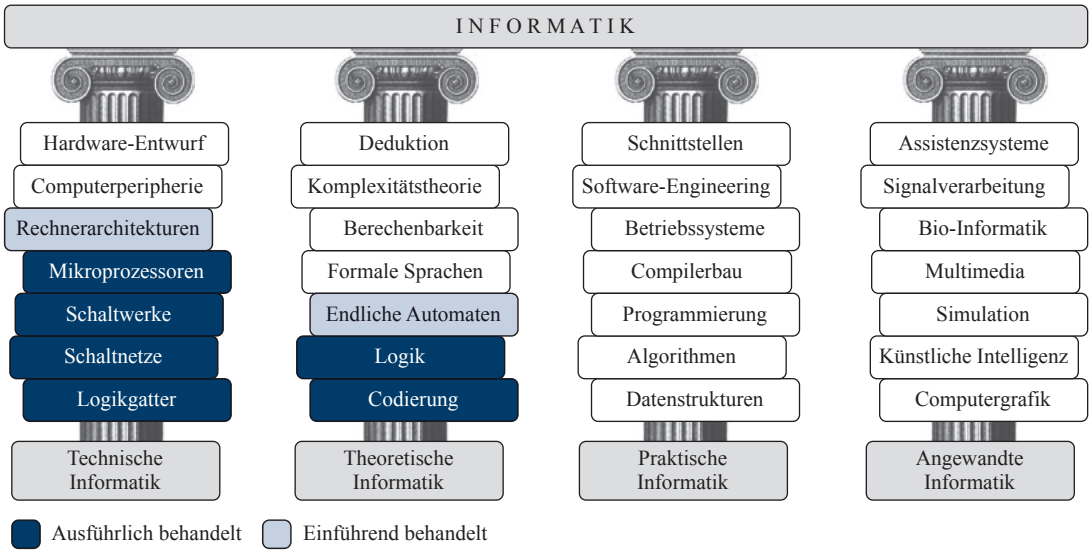


Abbildung 1.1: Die vier Säulen der Informatik

komplexer Digitalschaltungen. Hierzu gehören einfache *Schaltnetze* genauso wie komplexe, mit *Speicherelementen* angereicherte *Schaltwerke*. Durch das wechselseitige Zusammenspiel von Millionen von Schaltelementen sind wir heute in der Lage, Systeme mit einer Komplexität zu konstruieren, die noch vor nicht allzu langer Zeit als unmöglich erachtet wurde. Nichtsdestotrotz lässt sich selbst das komplexeste System stets auf die gleichen Grundprinzipien zurückführen. Die folgenden Kapitel werden diese in ein helleres Licht rücken und den Leser praxisnah in die technische Funktionsweise moderner Computersysteme einführen.

Die technische Informatik ist eng mit der theoretischen Informatik verzahnt. Viele der dort entwickelten Konzepte aus den Bereichen der Codierungstheorie, Logik und der endlichen Automaten dienen uns als das mathematische Fundament zur Beschreibung von Computer-Hardware. In entsprechender Weise werden wir uns auch in diesem Buch mit etlichen Teilaspekten dieser Disziplin beschäftigen. Doch bevor wir vollends in die Welt der Bits und Bytes eintauchen, wollen wir einen kurzen Streifzug durch die junge, aber bewegte Geschichte der Computertechnik wagen und uns mit der Frage beschäftigen, wohin die Reise in den nächsten Jahren führen wird.

1.2 Vom Abakus zum Supercomputer

Die ersten mechanischen Rechenhilfen

Wir beginnen unseren Streifzug durch die Geschichte im elften Jahrhundert vor Christus. Etwa zu dieser Zeit wird in China mit dem *Suan pan* die erste mechanische Rechenhilfe entwickelt – der sogenannte *Abakus*. Obwohl das auf den ersten Blick primitiv anmutende Rechenbrett nicht viel mit der heutigen Computertechnik verbindet, stellt der Abakus einen bedeutenden Schritt in Richtung des maschinellen Rechnens dar und ist mit seiner redundanten Zifferndarstellung ein willkommener Einstieg in die Thematik der Zahlensysteme.

Abbildung 1.2 zeigt das Bild eines chinesischen Abakus, wie er noch heute auf vielen fernöstlichen Warenmärkten die uns vertraute elektronische Kasse ersetzt. Aufgebaut ist der Suan pan aus einer Reihe von Stäben, auf denen jeweils 7 bewegliche Kugeln in zwei unterschiedlichen Segmenten aufgefädelt sind. Das obere Segment – der *Himmel* – enthält jeweils 2 und das untere Segment – die *Erde* – die restlichen fünf Kugeln. Zur Zahlendarstellung verwendet der Abakus das uns geläufige arabische System. Jeder Stab repräsentiert eine einzelne Ziffer, deren Wert sich aus der Stellung und den Wertigkeiten der einzelnen Kugeln bestimmt. In die Berechnung des Ziffernwerts gehen ausschließlich diejenigen Kugeln ein, die nach *innen*, d. h. in Richtung der mittleren Querstrebe, geschoben wurden. Jede Kugel aus dem oberen Segment erhöht den Ziffernwert dabei um 5 und jede Kugel aus dem unteren Segment um 1.

Abbildung 1.3 demonstriert die Darstellung der Zahl 10. Wie das Beispiel zeigt, ist die Zahlendarstellung im Gegensatz zum klassischen Dezimalsystem nicht eindeutig. Der Grund hierfür liegt in der Anzahl und der Wertigkeit der Kugeln, mit denen sich nicht nur die klassischen Dezimalziffern 0 bis 9, sondern auch die Werte 10 bis 15 darstellen lassen.

Der Aufbau des Abakus hat sich im Laufe der Zeit und durch den Einfluss verschiedener Kulturkreise in unterschiedliche Richtungen weiterentwickelt. So besitzt der japanische *Soroban* im Gegensatz zum chinesischen Suan pan nur noch 5 statt 7 Kugeln und der russische *Stschoty* kennt z. B. überhaupt keine Aufteilung in Himmel und Erde mehr.

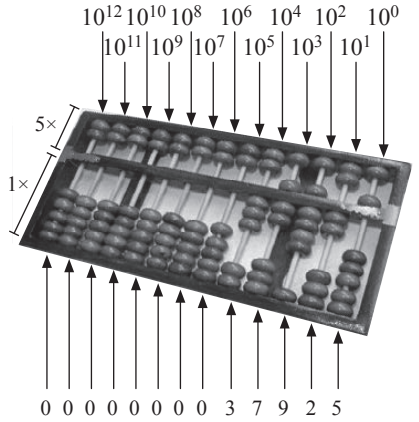
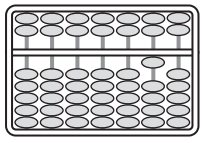
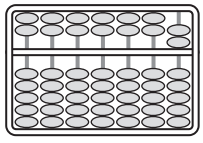


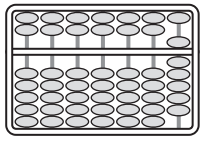
Abbildung 1.2: Der *Suan pan* (chinesischer Abakus)



$10 = 1 \cdot 10$



$10 = 2 \cdot 5$



$10 = 1 \cdot 5 + 5 \cdot 1$

Abbildung 1.3: Die Zahl 10 kann mit Hilfe des Abakus auf drei verschiedene Weisen dargestellt werden.

„Dasselbe, was Du auf rechnerischem Weg gemacht hast, habe ich kürzlich mechanisch versucht und eine aus 11 vollständigen und 6 verstümmelten Rädchen bestehende Maschine gebaut, welche gegebene Zahlen im Augenblick automatisch zusammenrechnet: addiert, subtrahiert, multipliziert und dividiert. Du würdest hell auflachen, wenn Du da wärest und sehen könntest, wie sie, so oft es über einen Zehner oder Hunderter weggeht, die Stellen zur Linken ganz von selbst erhöht oder ihnen beim Subtrahieren etwas wegnimmt.“

Wilhelm Schickard (1592 - 1635)

Abbildung 1.4: Schickard in einem Brief vom 20.9.1623 an Johannes Kepler

Abbildung 1.5: Die Schickard’sche Rechenuhr

Die Schickard’sche Rechenuhr

Mit dem Abakus lassen sich Zahlen *darstellen* und mit etwas Übung selbst komplexe arithmetische Operationen durchführen. Gleichwohl erfolgen alle Berechnungen stets manuell. Weit mehr als tausend Jahre vergingen, bis der deutsche Astronom, Mathematiker und Theologe Wilhelm Schickard das erste mechanische Rechenwerk ersann, mit dessen Hilfe sich zwei Zahlen zum einen vollautomatisch addieren und subtrahieren und zum anderen halbautomatisch multiplizieren und dividieren ließen. Schickard, der am 22.4.1592 im schwäbischen Herrenberg geboren wurde und 1619 einem Ruf an die Universität Tübingen folgte, verband eine lebenslange Freundschaft mit dem kaiserlichen Mathematiker und Astronomen Johannes Kepler, der für seine umfangreichen Berechnungen zur Planetenbewegung bis dato auf Papier und Bleistift angewiesen war.

Die Schickard’sche Rechenuhr besteht aus drei Teilen, die übereinander angeordnet sind (siehe Abbildung 1.5). Der obere Teil entspricht dem *Multiplikationswerk*, der mittlere dem *Additionswerk* und der untere einem *Speicher*, der die Funktion einer Merkhilfe für Zwischenergebnisse übernimmt. Die Funktionsweise des Multiplikationswerks entspricht im Wesentlichen dem Prinzip der *Napierstäbchen*, die auf den schottischen Mathematiker Lord John Napier of Merchiston zurückgehen [32]. Das Multiplikationswerk und das Additionswerk verfügen über keinerlei mechanische Verbindung. Die verschiedenen bei der Produktberechnung entstehenden Teilsummen mussten deshalb manuell abgelesen und per Hand in das Addierwerk übertragen werden. Aus diesem Grund war die Multiplikation mit Hilfe der Schickard’sche Rechenuhr nur halbautomatisch möglich.

Von der Schickard’schen Rechenuhr wurden nur zwei Exemplare überhaupt gebaut, die in den Wirren des Dreißigjährigen Kriegs für immer verloren gingen. Anhand von Skizzen und Aufzeichnungen aus den Nachlässen von Schickard und Kepler konnte die Rechenuhr jedoch rekonstruiert und die Funktionstüchtigkeit in mehreren Nachbauten nachträglich unter Beweis gestellt werden.

Es war die Pest, die das Leben von Wilhelm Schickard und seiner Familie im sechzehnten Jahr des Dreißigjährigen Kriegs ein tragisches Ende nehmen ließ. Zuerst rafft der schwarze Tod im Jahre 1634 seine Frau und seine drei Töchter dahin. Ein Jahr später, am 24. Oktober 1635, stirbt auch Wilhelm Schickard – zwei Tage, bevor sein neunjähriger Sohn ebenfalls der Seuche erliegt.

Die Rechenmaschinen des Charles Babbage

Weitere Meilensteine im Bereich des maschinellen Rechnens stellen die Rechenmaschinen des britischen Mathematikers und Ökonomen Charles Babbage dar, der am 26. Dezember 1791 in London das Licht der Welt erblickte [45]. Bereits mit 22 Jahren wird Babbage Mitglied in der Royal Society und nimmt 1823 unter Förderung der britischen Regierung die Arbeiten an der *Differenzenmaschine* auf. Im Gegensatz zur Schickard'schen Rechenuhr, die für die automatische bzw. halbautomatische Durchführung von primitiven arithmetischen Operationen konzipiert war, sollte die Differenzenmaschine in der Lage sein, ganze Wertetafeln komplexer Polynome selbstständig zu erzeugen.

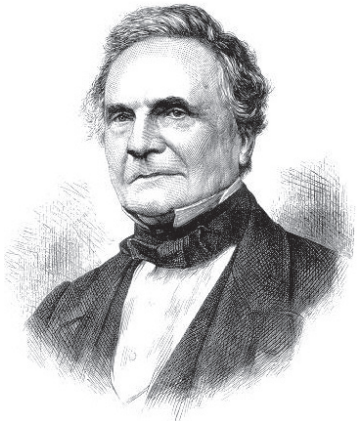
Das Prinzip der Maschine beruhte auf der Newton'schen *Differenzenmethode*, mit der solche Berechnungen auf trickreiche Weise unter der ausschließlichen Verwendung von Additionen und Subtraktionen durchgeführt werden können. Um zum Beispiel das Polynom

$$y = x^2 - 2x + 3$$

mit Hilfe der Differenzenmethode an den Stützstellen $x = 0,1,2,3,\ldots$ auszuwerten, gehen wir in drei Schritten vor:

- Im ersten Schritt stellen wir die *initiale Differenzentabelle* auf. Für Polynome n-ten Grades tragen wir zunächst die ersten $n+1$ manuell berechneten Funktionswerte $y(0)$ bis $y(n)$ ein. Die nächsten Spalten werden durch sukzessive Differenzberechnung erzeugt. Dabei enthält die zweite Spalte die Differenzen der Elemente der ersten Spalte, die dritte Spalte die Differenzen der Elemente der zweiten Spalte und so fort. Insgesamt entsteht auf diese Weise die in Abbildung 1.7 (oben) dargestellte Tabelle.
- In der dritten Spalte erhalten wir als Differenz zweiter Ordnung den Wert 2. Egal um wie viele Stützstellen Sie die Tabelle nach unten ergänzen – die Elemente der dritten Spalte sind für unser Beispiel stets gleich. Aus funktionsanalytischer Sicht ist dieses Ergebnis auch nicht weiter verwunderlich, schließlich berechnen wir durch die n-malige Differenzenbildung nichts anderes als die diskrete Ableitung n-ter Ordnung und diese ist für Polynome n-ten Grades stets konstant. Unseren Überlegungen folgend können wir somit die dritte Spalte, wie in Abbildung 1.7 (Mitte) dargestellt, im zweiten Schritt beliebig nach unten erweitern.
- Im dritten Schritt füllen wir fehlende Einträge der Differenzentabelle von rechts nach links auf und können die gesuchten Funktionswerte

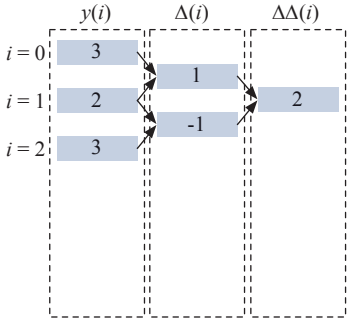
„One evening I was sitting in the rooms of the Analytical Society, at Cambridge, my head leaning forward on the table in a kind of dreamy mood, with a table of logarithms laying open before me. Another member, coming into the room, and seeing me half asleep, called out, 'Well Babbage, what are you dreaming about?' to which I replied, 'I am thinking that all these tables (pointing to the logarithms) might be calculated by machinery.' [95]



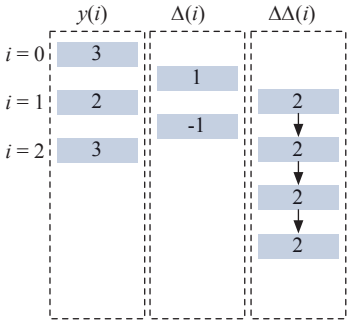
Charles Babbage (1791 – 1871)

Abbildung 1.6: Charles Babbage

- Die initiale Differenzentabelle:



- Fortsetzen der letzte Spalte:



- Ableiten weiterer Stützstellen:

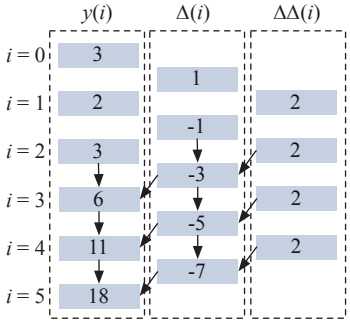


Abbildung 1.7: Stützstellenberechnung mit Hilfe der Differenzenmethode am Beispiel des Polynoms $y = x^2 - 2x + 3$

schließlich in der ersten Spalte ablesen. Die auf diese Weise vervollständigte Differenzentabelle ist für unser Beispiel in Abbildung 1.7 (unten) dargestellt.

Leider wird die Differenzenmaschine nie gebaut. Zum einen gestaltet sich die Fertigung vieler der geschätzten 25.000 Bauteile schwieriger als erwartet, zum anderen bringt Babbage stets neue Ideen und Verbesserungsvorschläge ein, die das Projekt zusätzlich verlangsamen. Als die Kosten schließlich vollends aus dem Ruder laufen, zieht sich die britische Regierung 1842 aus dem Projekt zurück.

In der Folgezeit ersann Babbage neben einer deutlich verbesserten *Differenzenmaschine 2* auch die *analytische Maschine*, die in ihrer Komplexität alle seine bisherigen Entwürfe nochmals weit übertrifft. Obwohl es sich bei der analytische Maschine um eine vollständig mechanische Konstruktion handelt, finden sich in deren Entwurf viele der klassischen Elemente wieder, die auch heute noch die Grundstrukturen moderner Computer prägen. So verfügt die analytische Maschine über einen Speicher und ein getrenntes Rechenwerk (*Mill*). Zur Ausgabe der Ergebnisse sah Babbage einen speziell konstruierten Drucker vor. Programmiert werden sollte die Maschine mit Hilfe von *Lochkarten* – ein Konzept, das zu dieser Zeit bereits bekannt war und erstmals 1805 von dem französischen Buchbinder Joseph-Marie Jacquard zur Automatisierung der Webtechnik eingesetzt wurde. Die analytische Maschine war so allgemein konzipiert, dass sie in der Lage gewesen wäre, selbst komplexe Kontrollstrukturen wie Schleifen, Unterprogrammaufrufe und bedingte Sprünge abzubilden. Leider war der Entwurf nicht nur konzeptionell, sondern auch in seiner Komplexität der damaligen Zeit weit voraus und die analytische Maschine wurde ebenfalls nie vollendet.

Trotz seiner unbestrittenen Erfolge in verschiedenen Gebieten der Wissenschaft war Babbage am Ende seines Lebens tief von dem Scheitern seiner drei Großprojekte gezeichnet. Mit der britischen Regierung, die jeder weiteren finanziellen Förderung eine Absage erteilte, ging er noch zu Lebzeiten hart ins Gericht. Am 18. Oktober 1871 starb Babbage in seinem Haus in London als enttäuschter und verbitterter Mann im Alter von 79 Jahren. 150 Jahre nach ihrer Erfindung schaffte zumindest die Differenzenmaschine 2 dann doch noch den Sprung vom Reißbrett in die Realität. Am Londoner Science Museum wurde die Maschine nach Originalplänen rekonstruiert. 1991 wurden die Arbeiten an der funktionsfähigen Maschine beendet – pünktlich zum 200. Geburtstag ihres Erfinders.

Die elektrische Revolution

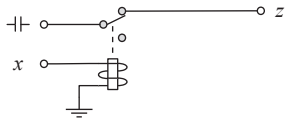
Die Verwendung elektromechanischer Bauteile stellt einen Quantensprung in der Computergeschichte dar. Mit der elektrischen Spannung als Informationsträger war auf einen Schlag die Konstruktion von Rechenmaschinen möglich, die in ihrer Komplexität und Zuverlässigkeit weit über das hinausgingen, was rein mechanisch arbeitende Apparaturen je zu leisten im Stande gewesen wären. Die ersten Rechenmaschinen dieser Art wurden Anfang der Vierzigerjahre gebaut und verwendeten zunächst *elektromechanische Relais* zur Daten- und Kontrollflusssteuerung. Das Relais, das schon kurz nach seiner Erfindung im Jahre 1835 durch Joseph Henry mit der Telegraphie das Kommunikationszeitalter einläutete, löste damit rund hundert Jahre später eine weitere technologische Revolution aus.

Zeitgleich mit dem Relais, das nur die zwei Zustände *offen* und *geschlossen* unterscheidet, hält das *Binärsystem* endgültig Einzug in die Computertechnik und bildet bis heute die Grundlage aller maßgebenden Computerarchitekturen. Abbildung 1.8 zeigt, wie sich die logischen Grundoperationen mit Hilfe konventioneller elektromechanischer Relais konstruieren lassen. In Kapitel 4 werden wir auf die verschiedenen Logikoperationen im Detail eingehen und in den darauf folgenden Kapiteln zeigen, wie sich unter alleiniger Verwendung der Elementarverknüpfungen NOT, AND und OR Hardware-Schaltungen mit nahezu beliebiger Komplexität in die Praxis umsetzen lassen.

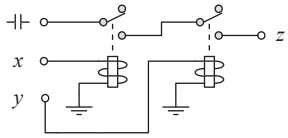
Die legendäre Z3 des Konrad Zuse

Konrad Zuse konstruierte 1941 mit der Z3 den ersten voll funktionsfähigen Rechner dieser neuen Generation [96]. Im Gegensatz zu der noch weitgehend mechanisch arbeitenden Z1 und dem Versuchsmodell Z2, die aufgrund technischer Mängel beide nur unzuverlässig ihren Dienst verrichteten, verwendete Zuse in der Z3 ausschließlich elektromechanische Relais. Die Maschine war aus ca. 2000 Einzel-Relais aufgebaut und konnte mit einer Taktfrequenz von 5 bis 10 Hertz betrieben werden. Insgesamt kam die Z3 auf ein Gesamtgewicht von ca. 1000 kg und besaß eine Leistungsaufnahme von 4000 Watt. Nicht nur aus der Sicht der Ingenieurkunst war die Z3 bis dato einzigartig – sie war vor allem auch aus konzeptioneller Sicht ein Meisterwerk. Viele Konzepte, die sich heute in modernen Computerarchitekturen wiederfinden, waren bereits in der Z3 vorhanden, wenngleich auch in deutlich primitiverer Form:

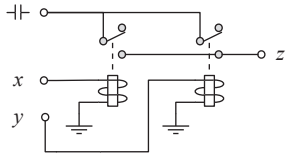
- Die NOT-Verknüpfung:



- Die AND-Verknüpfung:



- Die OR-Verknüpfung:



- Der Relais-basierte Zustandsspeicher:

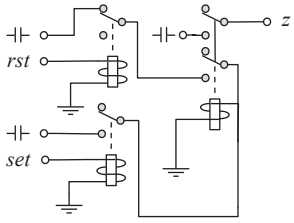


Abbildung 1.8: Die Basiskomponenten eines Relais-Rechners

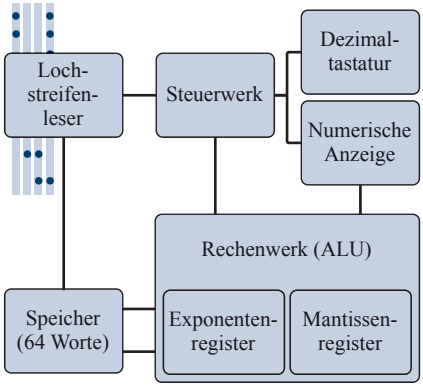


Abbildung 1.9: Blockschaltbild der Z3

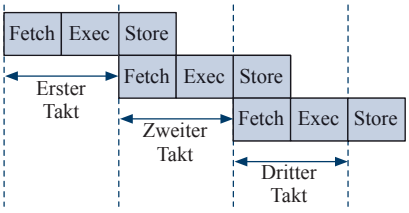


Abbildung 1.10: Die Pipeline-Architektur der Z3

Befehl	Bedeutung	Codierung
Ein- und Ausgabe		
`Lu`	Eingabe	01110000
`Ld`	Ausgabe	01111000
Speichertransfer		
`Pr` *z*	Lesen	11*Adresse*
`Ps` *z*	Schreiben	10*Adresse*
Arithmetik		
`Lm`	Multiplizieren	01001000
`Li`	Dividieren	01010000
`Lw`	Wurzelziehen	01011000
$\texttt{Ls}_1$	Addieren	01100000
$\texttt{Ls}_2$	Subtrahieren	01101000

Tabelle 1.1: Der Befehlssatz der Z3

- Wie das Blockschaltbild in Abbildung 1.9 demonstriert, sind der *Speicher* und der *Prozessor* – bestehend aus *Steuer-* und *Rechenwerk* – klar voneinander getrennt. Diese Zweiteilung findet sich auch heute noch in den allermeisten Computerarchitekturen wieder.

- Intern verwendet die Z3 das *Gleitkommaformat* zur Darstellung rationaler Zahlen. Über die numerische Tastatur eingegebene Operanden wurden automatisch in das interne Gleitkommaformat übersetzt und die Rechenergebnisse für die Anzeige ebenfalls vollautomatisch in das Dezimalsystem zurückkonvertiert. Damit war die Erfindung von Zuse auch hier anderen Maschinen weit voraus, die mit dem *Festkommaformat* eine wesentlich primitivere Art der Darstellung einsetzten. In Kapitel 3 werden wir uns intensiv mit den Unterschieden des Gleit- und Festkommaformats beschäftigen.

- Das Prinzip der *Mikroprogrammierung*, das auch heute die Grundlage vieler moderner Mikroprozessoren bildet, findet sich bereits in der Z3 verwirklicht. Im Zentrum des Steuerwerks befindet sich hierzu der sogenannte *Mikrosequenzer*, mit dessen Hilfe sich komplexe Rechenoperationen in mehrere elementare Berechnungsschritte zerlegen und nacheinander ausführen ließen. Im Zusammenhang mit der CISC- und RISC-Architektur werden wir in Kapitel 12 auf dieses Prinzip der Ablaufsteuerung zurückkommen.

- Zur Geschwindigkeitssteigerung führt die Z3 den Befehlsstrom *überlappend* aus, d. h. während des Zurückschreibens des Ergebniswerts ist der Rechner in der Lage, bereits die nächste Instruktion einzulesen. Wie in Abbildung 1.10 gezeigt, folgte die Z3 damit einer dreistufigen *Pipeline-Architektur*, die mit einem einstufigen Versatz ausgeführt wurde. In Kapitel 12 werden wir uns auch mit diesem Architekturprinzip genauer beschäftigen.

- Auch das verwendete Rechenwerk musste zu seiner Zeit keinen Vergleich scheuen. So setzte die Z3 zur Beschleunigung der Addition die *Carry-look-ahead-Technik* ein, die auch heute noch die Grundlage vieler Addierwerke bildet. In Kapitel 7 werden wir dieses Additionsprinzip im Detail kennen lernen.

Programmiert wurde die Z3 mit Hilfe eines 8-spurigen *Lochstreifens*. Jede Zeile codiert einen einzigen Befehl, zusammen mit den dazugehörigen Operanden. Wie die Befehlsübersicht in Tabelle 1.1 zeigt, lassen sich die insgesamt neun Befehle in die Kategorien *Ein- und Ausgabe*, *Speichertransfer* und *Arithmetik* unterteilen.

Mit Hilfe der Befehle **`Lu`** bzw. **`Ld`** wird eine Dezimalzahl über die numerische Tastatur eingelesen bzw. auf der numerischen Anzeige ausgegeben. Beide Befehle stoppen die Ausführung der Maschine, bis der Benutzer manuell die Fortsetzung veranlasst. Die Kommunikation mit dem Hauptspeicher wird über die Befehle **`Pr`** bzw. **`Ps`** gesteuert. Die anzusprechende Speicheradresse wird hierzu in den 8-Bit-Opcode der Befehle hineincodiert. Mit den 6 zur Verfügung stehenden Adressbits lassen sich insgesamt $2^6 = 64$ Speicherworte adressieren. Die restlichen Befehle dienen zur Durchführung der vier arithmetischen Grundrechenarten sowie der Berechnung der Quadratwurzel.

Betrachten wir den Befehlssatz der Z3 genauer, so fällt auf, dass mit den vorhandenen Befehlen weder eine Verzweigung noch eine Schleife programmiert werden kann. Der Programmierung sind hierdurch enge Grenzen gesetzt und die Z3 stellt damit keinen Universalrechner im eigentlichen Sinne dar.

Konrad Zuse (1910 – 1995)

Abbildung 1.11: Bronze-Statue in Hünfeld zum Gedenken an Konrad Zuse

Die Harvard Mark I

Fast zeitgleich mit dem Bau der Z3 wurde an den IBM Laboratories in New York unter der Regie von Howard H. Aiken der größte jemals fertiggestellte elektromechanische Rechner gebaut – der *Automatic Sequence Controlled Calculator* (ASCC) [17]. In einer feierlichen Zeremonie wurde die gigantische Maschine am 7. August 1944 an die Harvard University übergeben und offiziell in Betrieb genommen. Die Komplexität der Konstruktion, die fortan den Namen *Harvard Mark I* trug, lässt sich anhand weniger Kenndaten bereits erahnen. So bestand die gesamte Apparatur aus ca. 765.000 Einzelkomponenten, die über etliche Kilometer Kabel miteinander verbunden waren. Insgesamt füllte die Konstruktion mit 6 Metern Länge und über 2 Metern Höhe eine Werkhalle vollständig aus.

Die Harvard Mark I enthielt 72 *Akkumulatoren*, die jeder für sich eine dezimalcodierte Zahl mit einer Genauigkeit von 23 Ziffern speichern konnten. Neben einem weiteren Kurzzeit- sowie einem Konstantenspeicher verfügte der Rechner über separate Multiplikations- und Divisionseinheiten. Gesteuert wurde die Harvard Mark I über 24-spurige Lochstreifen, deren allgemeiner Aufbau in Abbildung 1.12 dargestellt ist. Jede Zeile codiert einen einzelnen Steuerbefehl, der sich aus drei Komponenten zusammensetzt. So definieren die Einstanzungen der ersten 8 Spuren die Quelladresse und die Einstanzungen der mittleren 8 Spuren die Zieladresse einer Operation. Die restlichen 8 Spuren definieren den Opcode des auszuführenden Befehls. Die Harvard Mark I konnte die so

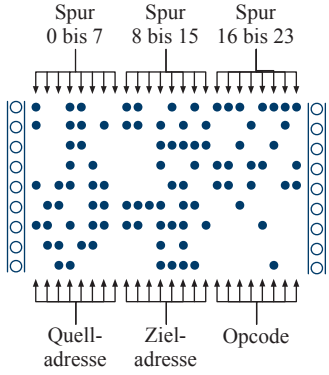


Abbildung 1.12: Aufbau des Lochstreifens zur Steuerung der Harvard Mark I

Dass wir heute im Software- und Hardware-Bereich Fehler gemeinhin als *Bugs* bezeichnen, wird in vielen Quellen in Zusammenhang mit der Harvard Mark I gebracht. Gerüchten zufolge geriet eine Motte in eines der zahlreichen elektrischen Relais des Rechners und brachte auf diese Weise die gesamte Maschine zum Erliegen. Die Motte wurde entfernt und der Begriff *Debugging* war geboren.
In der Tat gehört die Geschichte zu den klassischen Mythen des Computerzeitalters. Die besagte Motte gab es wirklich, allerdings stammte sie nicht aus der Harvard Mark I, sondern von ihrem Nachfolger, der Harvard Mark II. Der Begriff des *Debuggings* wurde jedoch bereits viel früher geprägt, wie z. B. der folgende Ausschnitt aus einem Brief von Thomas Edison vom 18. November 1878 an Theodore Puskas verrät:

„It has been just so in all my inventions. The first step is an intuition – and comes with a burst, then difficulties arise. This thing gives out and then that – 'Bugs' – as such little faults and difficulties are called – show themselves and months of anxious watching, study and labor are requisite before commercial success – or failure – is certainly reached“ [47]

Nichtsdestotrotz war die Motte der Harvard Mark II wahrscheinlich der erste richtige *Bug* der Computergeschichte. Howard Aiken selbst bezeichnete sie als den *„first actual case of bug being found“*.

codierten Befehle ausschließlich sequenziell abarbeiten – Befehle für Schleifen, Verzweigungen und Unterprogrammaufrufe sah die Rechnerarchitektur noch nicht vor. Damit ließen sich Endlosschleifen nur auf physikalischem Wege durch das Zusammenfügen des Lochstreifens zu einem endlosen Band realisieren.

Das bereits in der Harvard Mark I umgesetzte Konzept, Code und Daten in getrennten Speichern vorzuhalten, hat im Bereich moderner Computerarchitekturen in den letzten Jahren eine wahre Renaissance erlebt. So arbeiten z. B. moderne digitale Signalprozessoren fast ausschließlich nach diesem Prinzip, da sich im Vergleich zur klassischen Von-Neumann-Architektur deutlich höhere Übertragungsraten zwischen der Zentraleinheit und dem Speicher erreichen lassen. In Anlehnung an ihre historischen Wurzeln sprechen wir in diesem Zusammenhang heute immer noch von der *Harvard-Architektur*.

Die Harvard Mark I wurde erst 1959 außer Betrieb gestellt und anschließend zerlegt. Einige Komponenten sind heute noch im *Cabot Science Scenter* der Harvard University zu sehen, der Rest befindet sich im Besitz der IBM Laboratories in New York und des Smithsonian-Instituts in Washington, D.C.

Die ENIAC

Die Frage, ob wir in der Z3 und der in etwa zeitgleich entwickelten Harvard Mark I die ersten richtigen *Computer* vor uns sehen dürfen, ist immer wieder Gegenstand hitziger Diskussionen und in Fachkreisen umstritten. Einige Experten sehen in der freien Programmierbarkeit und der konzeptionellen Untergliederung der Maschinen in Speicher, Rechenwerk und Steuerwerk die wesentlichen Grundzüge eines Computers verwirklicht. Andere Fachleute sind der Meinung, dass die *Universalität* des zu Grunde liegenden Berechnungsmodells das wesentliche Charakteristikum des Computers darstellt. Dieser Interpretation folgend, gelten sowohl die Z3 als auch die Harvard Mark I zwar als die ersten universellen Rechenmaschinen, nicht jedoch als Computer im modernen Sinne.

Den Bau der ersten voll funktionsfähigen Rechenmaschine, die nahezu allen Definitionen des modernen Computer-Begriffs standhält und daher von vielen Experten als der erste wirkliche Computer der Welt angesehen wird, datieren wir auf das Jahr 1946 – das Jahr, in dem die ENIAC[1] der Öffentlichkeit vorgestellt wurde [36, 86]. Der Rechnerko-

[1] Electronic Numerical Integrator And Computer

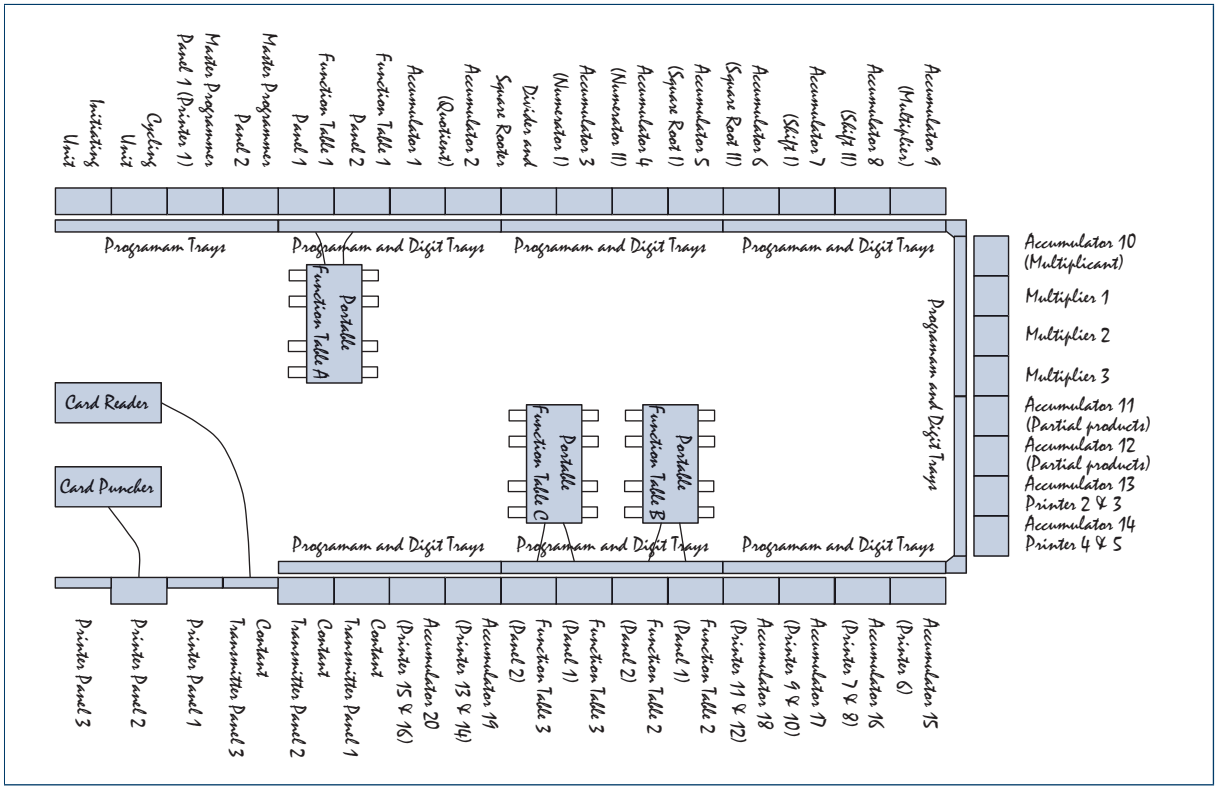


Abbildung 1.13: Der „Floorplan“ der ENIAC

loss wurde an der Moore School of Electrical Engineering der University of Pennsylvania unter der Leitung von J. Presper Eckert und John W. Mauchly gebaut und beeindruckte schon aufgrund seiner schieren Größe. Wie in Abbildung 1.13 dargestellt, bestand die ENIAC aus insgesamt 30 Einheiten, die U-förmig über den gesamten Raum verteilt angeordnet waren. Die gesamte Konstruktion kam auf ein Gesamtgewicht von knapp 30 Tonnen (Abbildungen 1.15 bis 1.17).

Der Entschluss, einen Rechner dieses Ausmaßes überhaupt zu bauen, wurde aus der Not geboren. Die Intensität, mit der der Zweite Weltkrieg 1939 über die Welt hereinbrach, setzte die U.S.-Armee unter enormen Zeitdruck, sich auf den immer wahrscheinlicher werdenden Kriegseintritt vorzubereiten. Unter anderem experimentierten die U.S.-Streitkräfte mit großkalibrigen ballistischen Geschützen, die anhand von Trajektorien-Tabellen auf ihr Ziel ausgerichtet wurden. Das Aufstellen der Tabellen erforderte das Lösen von Differenzialgleichungen

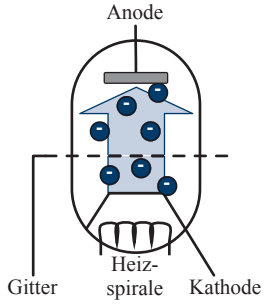


Abbildung 1.14: Die Trioden-Röhre als Schaltelement. Der durch die Heizspirale verursachte Stromfluss von der Kathode zur Anode kann durch das Anlegen einer Spannung an das Metallgitter blockiert oder verstärkt werden – ein einfaches Schaltelement entsteht.

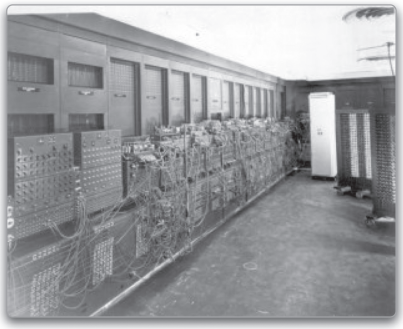


Abbildung 1.15: Teilansicht der ENIAC (U.S.-Armee-Foto)

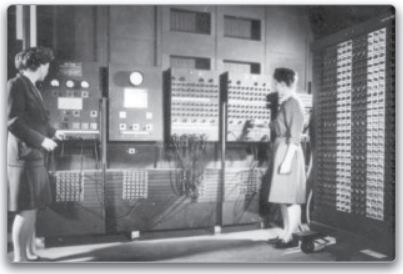


Abbildung 1.16: Das *Main Control Panel* der ENIAC (U.S.-Armee-Foto)

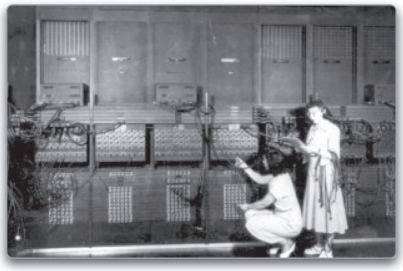


Abbildung 1.17: Programmiert wurde die ENIAC durch „Kabelstecken“ (U.S.-Armee-Foto)

zweiter Ordnung und damit Fähigkeiten außerhalb des Leistungsspektrums damaliger Rechenmaschinen. Die ENIAC, mit deren Hilfe die automatisierte Berechnung der Trajektorien-Tabellen möglich werden sollte, war in zweierlei Hinsicht revolutionär:

- Im Gegensatz zur Z3 oder der Harvard Mark I war es möglich, Verzweigungen und Schleifen zu programmieren. Damit unterscheidet sich das formale Berechnungsmodell der ENIAC deutlich von ihren Vorgängern und ist insbesondere mit dem heutiger Computer vergleichbar, auch wenn die Programmierung des Kolosses im Vergleich zur heutigen Technik kaum unterschiedlicher sein könnte.
- In der ENIAC wurde auf die Verwendung elektromechanischer Relais verzichtet und die Schaltlogik stattdessen mit Hilfe von *Vakuumröhren* implementiert. Zwar ist die ENIAC nicht der erste Röhrenrechner der Welt, aber der erste Großrechner, der die Vakuumröhre in Form der *Triode* konsequent als Schaltelement verwendete. Im Vergleich zum Relais zeichnet sich die Röhre durch eine um den Faktor 1000 bis 2000 gesteigerte Schaltgeschwindigkeit aus, so dass die ENIAC mit einer für damalige Verhältnisse beeindruckenden Taktfrequenz von 100 Kilohertz betrieben werden konnte. Eine einzige Multiplikation berechnete die ENIAC in nur knapp 3 Millisekunden.

Auf der negativen Seite brachte die ENIAC konstruktionsbedingt auch gravierende Nachteile mit sich. Ein wesentliches Element der Röhrentriode ist, wie in Abbildung 1.14 skizziert, die Heizspirale. Wird sie zum Glühen gebracht, so beginnt die Kathode Elektronen zu emittieren, die von der Anode angezogen und aufgefangen werden. Konstruktionsbedingt sind Vakuumröhren damit äußerst stromhungrige Bauelemente. Die Leistungsaufnahme der ca. 18.000 in der ENIAC verbauten Vakuumröhren summierte sich auf sagenhafte 174.000 Watt. Des Weiteren besitzen Vakuumröhren im Vergleich zu Relais eine sehr begrenzte Lebensdauer und das langwierige Auffinden und Austauschen defekter Röhren gehörte zur täglichen Arbeit eines ENIAC-Ingenieurs.

Der wohl gravierendste Nachteil betrifft jedoch die Handhabung der Maschine, die nicht auf eine flexible Programmierung ausgelegt war. Der Datenpfad wurde mit Hilfe von Steckverbindungen fest verdrahtet und ein Programm für die ENIAC war damit nichts anderes als ein Verbindungsplan, der die benötigten Steckverbindungen beschrieb. Hierdurch war ein flexibler Wechsel zwischen verschiedenen Programmen von vorneherein ausgeschlossen.

Die Limitierungen der ENIAC inspirierten den in Budapest geborenen und später in die USA immigrierten Wissenschaftler John von Neumann zu einer bahnbrechenden Idee [34]. Sein Konzept sah vor, Programme und Daten beide in einem einzigen Speicher abzulegen und damit nicht mehr länger als getrennte Entitäten zu behandeln: Die Idee der *Von-Neumann-Architektur* war geboren (vgl. Abbildung 1.18). In Kapitel 11 werden wir dieses Prinzip, an dem sich auch die meisten modernen Computerarchitekturen heute noch orientieren, im Detail diskutieren.

Trotz aller Limitierungen und Schwierigkeiten war die ENIAC ein eindrucksvoller Beweis für die Funktionstüchtigkeit der Röhrentechnik. In den Folgejahren wurden weitere Großcomputer gebaut, die dem alternden Koloss in ihrer technischen Konzeption und Handhabung in immer größerem Maße überlegen waren. Die ebenfalls legendäre, an der Manchester University entwickelte *Manchester Mark I* und der *Electronic Delay Storage Automatic Calculator* (*EDSAC*) der Cambridge University sind zwei Beispiele aus dieser Zeit. Wie die ENIAC basierten auch diese Rechner auf der Röhrentechnologie und begründen zusammen die Computer der *ersten Generation*.

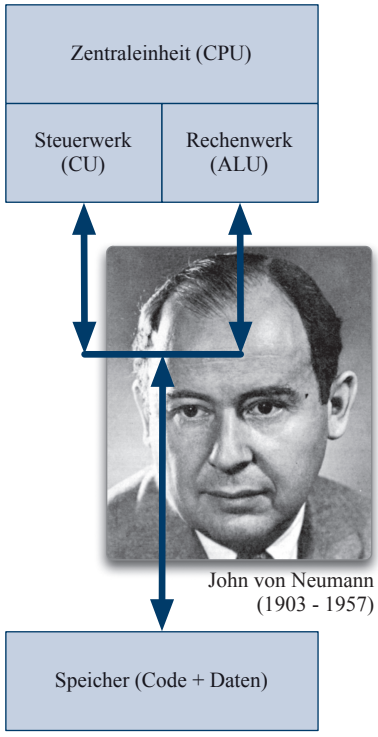


Abbildung 1.18: John von Neumann begründete die grundlegende Rechnerarchitektur, die auch heute noch die Basis moderner Computer bildet.

Der Siegeszug des Transistors

Der hohe Stromverbrauch und die begrenzte Lebensdauer der Vakuumröhre wurden mehr und mehr zum limitierenden Faktor bei der Entwicklung immer ausgefeilterer Rechnerkonstruktionen und für Forscher aus aller Welt zur treibenden Kraft bei der Suche nach neuen Technologien. Am 1. Juli 1948 verkündeten die Bell Laboratories in New York offiziell den Durchbruch und stellten den ersten praxistauglichen *Transistor* vor, der die bis dato dominierende Vakuumröhre vollständig ersetzen sollte:

> *„An amazingly simple device, capable of performing efficiently nearly all the functions of an ordinary vacuum tube, was demonstrated for the first time yesterday at Bell Telephone Laboratories where it was invented. Known as the Transistor, the device works on an entirely new physical principle discovered by the Laboratories in the course of fundamental research into the electrical properties of solids. Although the device is still in the laboratory stage, Bell scientists and engineers expect it may have far-reaching significance in electronics and electrical communication.“*
>
> *Bell Laboratories, Press Release [9]*

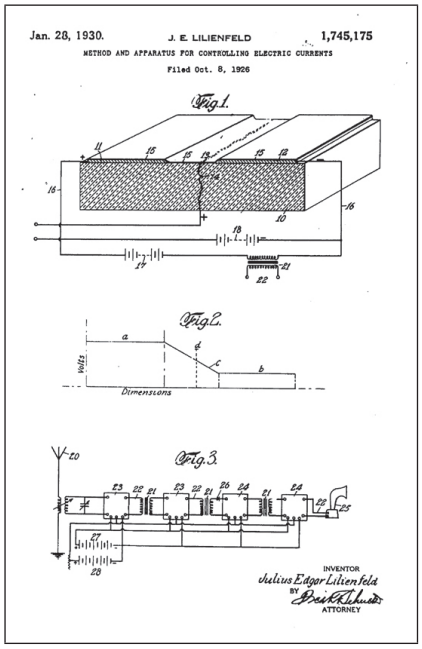


Abbildung 1.19: Julius Lilienfeld patentiert als Erster die Idee des Feldeffekt-Transistors. Die Abbildung zeigt einen Auszug aus der zweiten Patentschrift aus dem Jahre 1928 [56, 57].

An der Entwicklung des ersten technisch verwertbaren, voll funktionsfähigen Transistors waren die Ingenieure William Shockley, John Bardeen und Walter Brattain maßgeblich beteiligt und erhielten am 1. Dezember 1956 als Anerkennung ihrer bahnbrechenden wissenschaftlichen Leistung den Nobelpreis [6, 80]. Das Ergebnis der Arbeiten von Shockley, Bardeen und Brattain gilt heute als der erste technisch verwertbare und voll funktionsfähige Transistor – die Idee desselben war jedoch schon weit früher bekannt. So wurde das Prinzip des Transistors bereits 1928 von Julius Edgar Lilienfeld zum Patent angemeldet (Abbildung 1.19).

Der erste Computer auf Transistor-Basis wurde an der Manchester University Anfang der Fünfzigerjahre gebaut. Dem 1953 fertiggestellten Prototyp folgte eine deutlich erweiterte Variante, die zwei Jahre später in Betrieb genommen werden konnte. Aufgrund der noch sehr jungen Transistortechnik war der Rechner den modernen Röhren-Computern von damals sowohl in seiner Leistungsfähigkeit, aber auch in seiner Zuverlässigkeit deutlich unterlegen. Sahen einige Experten den Transistor gegenüber der Röhre zu Anfang im Nachteil, so konnte die neue Technologie den beginnenden Wettstreit jedoch alsbald für sich entscheiden. Mit der zunehmenden Verfeinerung der Technik stand mit dem Transistor ein Schaltelement zur Verfügung, das der Röhrentriode in puncto Leistungsaufnahme, Geschwindigkeit und Zuverlässigkeit haushoch überlegen war.

Der Übergang von der Röhre zum Transistor läutete das Zeitalter der Computer der *zweiten Generation* ein und war von weiteren wichtigen Entwicklungen begleitet. Steuer- und Rechenwerke wurden deutlich komplexer und bis dato fortschrittliche Konzepte wie die Verwendung indizierbarer Register oder der Einsatz von Gleitkomma-Hardware gehörten schnell zum Stand der Technik. Speicherseitig hielt der *Ferritkernspeicher* Einzug, der Daten im Kilobyte-Bereich speichern konnte, allerdings in mühevoller Handarbeit gefertigt werden musste. Auch die ersten Programmiersprachen datieren auf diese Zeit. 1957 wurde der erste FORTRAN-Compiler ausgeliefert und 1960 die Programmiersprache COBOL verabschiedet. Beide Sprachen sind heute immer noch im Einsatz – wenngleich sich deren Verwendungszweck nur noch auf wenige Spezialanwendungen konzentriert.

Der Siegeszug des Transistors begann im Jahre 1958, als es dem für Texas Instruments arbeitenden Ingenieur Jack Kilby gelang, den ersten integrierten Schaltkreis (engl. *Integrated Circuit*, kurz IC) herzustellen. Konnten Transistoren bis zu diesem Zeitpunkt ausschließlich als diskretes Bauelement verbaut werden, war es mit Kilbys Technik nunmehr möglich, mehrere Transistoren auf einem kleinen Stück Silizium zu in-

tegrieren. Kilbys Entdeckung, für die er im Jahr 2000 – fünf Jahre vor seinem Tod am 20.5.2005 – mit dem Nobelpreis geehrt wurde, war vielleicht der entscheidendste Durchbruch der Computergeschichte und der Auslöser einer bis heute anhaltenden Leistungsexplosion, die sich zur damaligen Zeit niemand jemals hätte erträumen können.

Bereits ein Jahr nach der Entwicklung des integrierten Schaltkreises gelang es dem Halbleiterhersteller Fairchild Semiconductor, mit der *Planartechnik* die Grundlage der Massenfertigung zu schaffen. Im Folgejahr brachte Fairchild nahezu zeitgleich mit Texas Instruments den ersten kommerziellen IC auf den Markt. Weitere drei Jahre später begann Fairchild 1963 mit der Produktion des Modells 907, der bereits zwei vollständige Logikgatter in einem einzigen Chip vereinte und damit die SSI-Technologie begründete. SSI ist die Abkürzung für *Small-Scale Integration* und bezeichnet eine von mehreren Komplexitätsklassen, in die sich integrierte Schaltklassen einordnen lassen.

Tabelle 1.2 fasst die verschiedenen Komplexitätsklassen zusammen, denen sich integrierte Schaltkreise zuordnen lassen. Zwischen den einzelnen Klassen existiert keine scharfe Trennung und die numerischen Angaben zur Gatteranzahl sind als Orientierungswerte zu verstehen. So unterscheiden sich auch die in der Literatur angegebenen Gattergrenzen mitunter erheblich und insbesondere die Abgrenzung im Bereich der VLSI- und ULSI-Klassen wurde im Laufe der Zeit mit zunehmender Chip-Komplexität immer wieder nach oben korrigiert oder durch neu eingeführte Klassen (SLSI, XLSI, GSI) ergänzt.

Der *Micromosaic*, ebenfalls von Fairchild Semiconductor auf den Markt gebracht, integrierte 1967 bereits 150 Logikgatter und ist aus historischer Sicht in zweierlei Hinsicht interessant. Zum einen gehörte er mit der Anzahl integrierter Gatter zu den ersten ICs der MSI-Technik (*Medium-Scale Integration*), zum anderen ist er der Vorläufer programmierbarer Logikbausteine, die wir in Kapitel 7 genauer betrachten werden.

Für die Herstellung des Micromosaics wurde ein regelmäßig aufgebauter Chip verwendet und die Transistorverbindungen erst später durch das Einbringen einer kundenspezifischen Belichtungsmaske erzeugt. Auf diese Weise war es erstmals möglich, eine speziell auf einen Kunden zugeschnittene Schaltung mit vergleichsweise geringem Aufwand und Kosten zu produzieren.

Eines der ersten vollständigen Computersysteme, das sich die integrierte Schaltkreistechnologie zu Nutze machte, war das legendäre *System/360*, das am 7. April 1964 der Öffentlichkeit vorgestellt wur-

Small-Scale Integration
■ kurz SSI ■ weniger als 100 Gatter
Medium-Scale Integration
■ kurz MSI ■ 100 bis 1000 Gatter
Large-Scale Integration
■ kurz LSI ■ 1000 bis 10.000 Gatter
Very-Large-Scale Integration
■ kurz VLSI ■ 10.000 bis 100.000 Gatter
Ultra-Large-Scale Integration
■ kurz ULSI ■ 100.000 bis 1.000.000 Gatter
Super-Large-Scale Integration
■ kurz SLSI ■ 1.000.000 bis 10.000.000 Gatter
Extra-Large-Scale Integration
■ kurz ELSI oder XLSI ■ 10.000.000 – 100.000.000 Gatter
Giga-Scale Integration
■ kurz GSI ■ mehr als 100.000.000 Gatter

Tabelle 1.2: Klassifizierung integrierter Schaltkreise

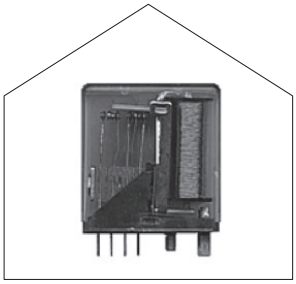
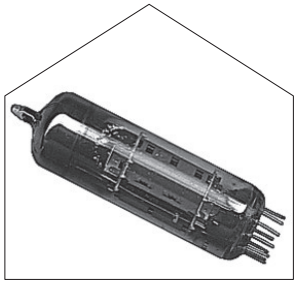
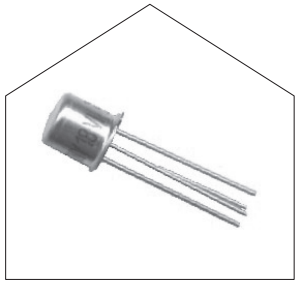
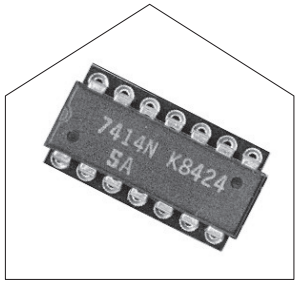
Erste Generation		Zweite Generation	Dritte und vierte Generation
1940 – 1954		1955 – 1964	ab 1965
Relais	Vakuumröhre	Transistor	Integrierter Schaltkreis
Schaltzeit: 10^{-1} s	Schaltzeit: 10^{-4} s	Schaltzeit: 10^{-6} s	Schaltzeit: $< 10^{-9}$ s
			

Tabelle 1.3: Die Entwicklung der Schaltelemente in der Übersicht

de [74]. Mit dieser Modellreihe begründete die Firma IBM die Ära der *Mainframe-Computer* und läutete gleichermaßen das Ende der Pionierzeit der Computertechnik ein. Der Aufwand, den IBM in die neue Idee investierte, war gigantisch. Zu Hochzeiten arbeiteten rund 50.000 Mitarbeiter an dem Projekt, dessen Gesamtkosten auf über 5 Milliarden US-Dollar geschätzt werden. Das System/360 brachte einige technische Neuerungen mit sich, die wir noch heute in modernen Computersystemen verwenden. So nutzt die S/360 beispielsweise das *Zweierkomplement* zur Darstellung negativer Zahlen und führt das Prinzip der byteweisen Speicheradressierung ein. Auf beide Konzepte kommen wir in Kapitel 3 im Detail zurück. Neben der Entwicklung von Großrechnern hat die zunehmende Verfügbarkeit leistungsfähigerer und billigerer SSI-, MSI- und LSI-Komponenten auch den Bau immer kleinerer und kostengünstigerer Computer ermöglicht. Die um 1970 gebauten PDP-Rechner (PDP = Programmable Data Processor) der Digital Equipment Corporation (DEC) sind Beispiele solcher *Minicomputer*.

Nach dem elektromagnetischen Relais, der Röhre, und dem Transistor in Form eines diskreten Bauelements ist der *integrierte Schaltkreis* bereits der vierte revolutionäre Wechsel der Basistechnologie, den die Computertechnik in ihrer vergleichsweisen jungen Geschichte erfahren hat. Genau wie die Relais- und Röhrentechnologie stellvertretend für die Computer der ersten Generation und die diskrete Transistortechnik für die Computer der zweiten Generation steht, so werden die frühen Rechner auf Basis integrierter Schaltkreise als Computer der dritten Ge-

neration bezeichnet. Bis zum heutigen Tag arbeiten Computer nach diesem Prinzip, wenngleich sich hochintegrierte Schaltkreise der letzten Generationen bezüglich ihrer Leistungsdaten kaum noch mit den ersten integrierten Schaltkreisen vergleichen lassen. Tabelle 1.3 stellt die Basistechnologien der verschiedenen Computergenerationen gegenüber.

Auch die Computertechnik der dritten Generation wurde durch einhergehende Entwicklungen weiter geprägt. Der Ferritkernspeicher hatte ausgedient und wurde nach und nach durch integrierte Speicherchips ersetzt. Diese waren deutlich kleiner und schneller als ihre Vorgänger und ermöglichten hierdurch drastisch höhere Speicherkapazitäten zu geringeren Kosten. Mehr und mehr Lochkartenleser mussten interaktiven *Text-Terminals* weichen und so ging auch die Ära der Lochkarte beständig ihrem Ende zu. Auch die ersten Betriebssysteme entstanden auf Computern dieser Generation. Bereits mit Hilfe der ersten Betriebssysteme war es möglich, verschiedene Programme gleichzeitig auszuführen und zu überwachen. Die Produktivität der Software-Entwicklung nahm hierdurch erst richtig an Fahrt auf.

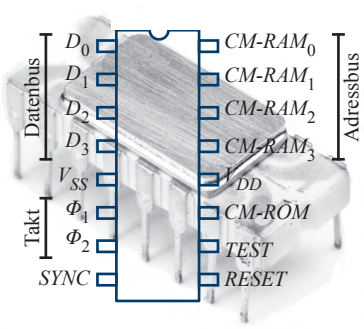


Abbildung 1.20: Der 4004-Mikroprozessor

Der erste Mikroprozessor

1968 verließen Robert Noyce, Gordon Moore und kurze Zeit später Andrew Grove die Firma Fairchild Semiconductor und gründeten die Integrated Electronics Corporation (Intel) im kalifornischen Santa Clara [46]. Bereits zwei Jahre später stellte Intel mit dem *1103* den ersten DRAM-Speicherbaustein vor. Die eigentliche Sternstunde schlug für Intel jedoch im Jahre 1971, als es dem Ingenieur Federico Faggin als Erstem gelang, alle Komponenten eines Prozessors gemeinsam auf einem einzigen Stück Silizium zu integrieren. Bereits kurze Zeit später brachte Intel mit dem *Intel 4004* den ersten Mikroprozessor der Welt auf den Markt und leitete damit den Beginn der vierten und bis dato letzten Computergeneration ein.

Der in einer Strukturbreite von 10 µm gefertigte Prozessor bestand aus 2250 Transistoren und lief mit einer Taktfrequenz von 740 kHz. Wie in Abbildung 1.20 gezeigt, wird das Taktsignal über zwei separate Leitungen zugeführt. Eine steigende Taktflanke im Signal ϕ_2 bewirkte den eigentlichen Zustandswechsel, während das zeitversetzt angelegte Signal ϕ_1 zur internen Steuerung eingesetzt wurde. Die beiden Pins V_{SS} und V_{DD} dienten der Stromaufnahme und versorgten den Prozessor mit einer Spannung von 5 V bzw. −10 V. Intern arbeitete der Prozessor, wie in Abbildung 1.21 skizziert, mit einer Bitbreite von gerade einmal 4 Bit und verfügte neben 16 Datenregistern über 4 Stapelregister zur Spei-

Sprungbefehle	
`JUN`	Direkter Sprung
`JIN`	Indirekter Sprung
`JCN`	Bedingte Verzweigung
`JMS`	Unterprogrammaufruf
`BBL`	Unterprogrammende
Ladebefehle	
`FIM`	Register laden (direkt)
`FIN`	Register laden (indirekt)
`LD`	Register → Akkumulator
`XCH`	Register ↔ Akkumulator
`LDM`	Wert → Akkumulator
Arithmetikbefehle	
`INC`	Inkrementieren
`ISZ`	`INC` + bedingter Sprung
`ADD`	Addition
`SUB`	Subtraktion
Sonstige Befehle	
`NOP`	Keine Operation

Tabelle 1.4: Die Grundoperationen des Intel 4004-Prozessors

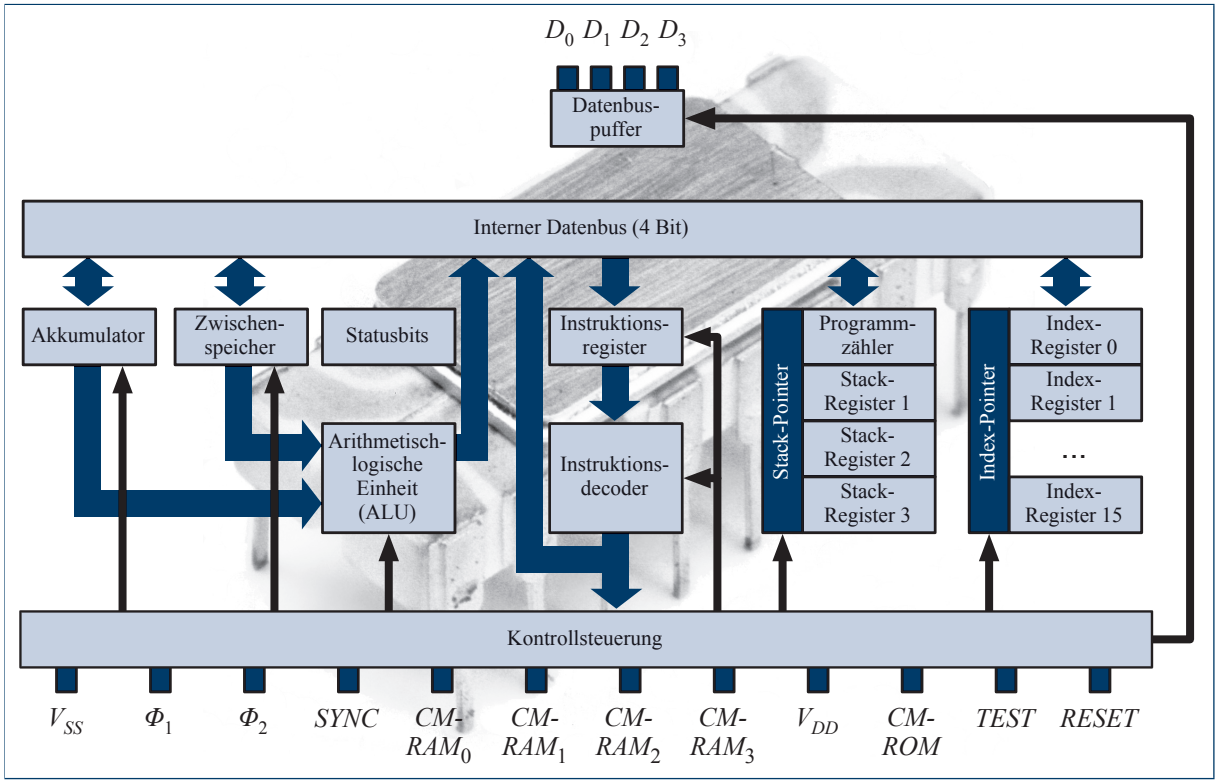


Abbildung 1.21: Blockschaltbild des 4004-Mikroprozessors

cherung des Programmzählers und der Rücksprungadressen von Unterprogrammaufrufen.

Um die mindestens 8 Bit langen Befehlswörter mit Hilfe des 4 Bit breiten Bus des 4004 trotzdem in einem einzigen Takt einlesen zu können, arbeitete der Prozessor im sogenannten *Multiplexing*-Modus. Das bedeutet, dass der Bus mit doppelter Taktfrequenz betrieben wurde, so dass die ersten 4 Bit in der ersten Hälfte und die restlichen 4 Bit in der zweiten Hälfte einer Taktperiode eingelesen werden konnten. Obwohl die Multiplexing-Logik die interne Architektur des 4004 deutlich verkomplizierte, vereinfachte sie erheblich die Produktion. Der Prozessor passte vollständig in ein 16-Pin-Gehäuse.

Tabelle 1.4 vermittelt einen detaillierteren Eindruck über die Grundbefehle des 4004-Prozessors. Sehen wir von der NOP-Instruktion ab, die ausschließlich dazu dient, die Befehlsausführung um einen einzi-