Tabelle aller Operatoren in C

Priorität	Operatoren	Bezeichnung
1	`()  []  ->  .`	Klammer, Vektor, Strukturverweise

2[1]	`!  ~  ++  --`
	`-  (Typ)  *`
	`&  sizeof`
		Negation, Einerkomplement, Increment, Decrement, Vorzeichen, Castoperator, Indirektion- Operator, Adressoperator, Objektgröße

3	`*  /  %`	Multiplikation, Division, Modulo (Divisionsrest)
4	`+  -`	Addition, Subtraktion
5	`<<  >>`	Shift links, Shift rechts
6	`<  <=  >  >=`	kleiner, kleiner gleich, größer, größer gleich
7	`==  !=`	gleich, ungleich
8	`&`	bitweises UND
9	`^`	bitweises EXODER
10	`|`	bitweises ODER
11	`&&`	logisches UND
12	`||`	logisches ODER
13[1]	`? :`	Konditional-Operator

14[1]	`=  +=  -=`
	`*=  /=  %=`
	`<<=  >>=  &=`
	`|=  ^=`
		Zuweisung, zusammengesetzte Zuweisungsoperatoren

15	`,`	Kommaoperator

Bei gleichrangigen Operatoren erfolgt die Auswertung von links nach rechts

[1] Bei Operatoren der Rangstufe 2, 13 und 14 erfolgt die Auswertung von rechts nach links


Oldenbourg

Oldenbourg Lehrbücher für Ingenieure

Herausgegeben von
Prof. Dr.-Ing. Helmut Geupel

Die fachliche und didaktische Qualität der Ingenieurausbildung wird zunehmend an internationalen Maßstäben gemessen. Dieser Herausforderung hat sich ein Autorenteam zusammen mit dem Oldenbourg Wissenschaftsverlag gestellt und die Buchreihe „Oldenbourg Lehrbücher für Ingenieure" geschaffen. Zentrales Anliegen dabei ist, dem Studenten mit anschaulich geschriebenen Texten für das jeweilige Fach ein grundlegendes Verständnis zu vermitteln.

Der Praxiseinsatz der ersten Bücher dieser Reihe hat gezeigt, dass es in der Tat den Studenten damit leichter fällt, sich in das neue Stoffgebiet einzufinden, dass sogar Teile davon selbständig angeeignet werden können. Nicht zuletzt wird dadurch die Eigeninitiative der Studenten trainiert – eine Fähigkeit, die Voraussetzung für die ständige Weiterbildung im Berufsleben ist – und mehr Raum für das Verarbeiten des Lehrstoffes im Unterricht gewonnen.

In der Reihe Oldenbourg Lehrbücher für Ingenieure sind bereits erschienen:

Bruno Assmann	Technische Mechanik, Band 1: Statik
Bruno Assmann	Technische Mechanik, Band 2: Festigkeitslehre
Bruno Assmann	Technische Mechanik, Band 3: Kinematik und Kinetik
Bruno Assmann	Aufgaben zur Kinematik und Kinetik
Bruno Assmann	Aufgaben zur Festigkeitslehre
Axel Böttcher, Franz Kneißl	Informatik für Ingenieure. Grundlagen und Programmierung in C
Joachim Erven, Dietrich Schwägerl	Mathematik für Ingenieure
Jürgen Gobrecht	Werkstofftechnik – Metalle
Hubert Hinzen	Maschinenelemente 1
Hubert Hinzen	Maschinenelemente 2
Fritz Tröster	Steuerungs- und Regelungstechnik für Ingenieure
Norbert Weichert, Michael Wülker	Messtechnik und Messdatenerfassung
Herbert Windisch	Thermodynamik

Informatik für Ingenieure

Grundlagen und Programmierung in C

von
Prof. Dr. Axel Böttcher,
Fachhochschule München
Prof. Dr. Franz Kneißl,
Fachhochschule Regensburg

2., überarbeitete Auflage

Oldenbourg Verlag München Wien

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Böttcher, Axel:
Informatik für Ingenieure : Grundlagen und Programmierung in C / von
Axel Böttcher ; Franz Kneißl. – 2., überarb. Aufl.. – München ;
Wien : Oldenbourg, 2002
(Oldenbourg-Lehrbücher für Ingenieure)
ISBN 3-486-25812-5

Rosenheimer Straße 145, D-81671 München
Telefon: (089) 45051-0
www.oldenbourg-verlag.de

Lektorat: Sabine Ohlms
Herstellung: Rainer Hartl
Umschlagkonzeption: Kraxenberger Kommunikationshaus, München
Gedruckt auf säure- und chlorfreiem Papier
Druck: R. Oldenbourg Graphische Betriebe Druckerei GmbH

Vorwort

Der Anteil der Software-Erstellung an der Ingenieurstätigkeit hat sich in den letzten Jahren dramatisch erhöht. Heute besteht die Wertschöpfung bei vielen technischen Produkten zum großen Teil aus der Entwicklung von Software – typischerweise in der Programmiersprache **C** oder objektorientierten Sprachen wie **C++** und **JAVA**.

Weniger dramatisch geändert hat sich in den meisten Studiengängen der Anteil der Informatik an der Gesamtstundenzahl. Um Stoff für aktuelle Themen und Anforderungen unterzubringen, müssen bisherige bewährte Inhalte reduziert werden.

In vielen Fachbereichen bedeutet dies nicht nur die Reduktion von Grundlagen wie z. B. „Geschichte der Datenverarbeitung" – auch Inhalte, die didaktisch besonders erprobt waren, fallen der Anpassung zum Opfer. Hierunter ist z. B. der Erstunterricht in einer für den Einstieg günstigen Programmiersprache wie **PASCAL** zu rechnen.

Der schwierigere Einstieg etwa über die Sprache **C** wird in den ersten Semestern durch ein weiteres Problem verstärkt: die Inhomogenität der Vorkenntnisse, die die Studierenden aus ihrer schulischen und beruflichen Ausbildung mitbringen. Wir haben uns daher entschlossen, ein Buch auf den Markt zu bringen, das dieser Problematik begegnet.

Inhaltlich geschieht dies durch die Konzentration auf aktuelle Themen, deren Kenntnis heute von jedem Ingenieur verlangt wird. Dazu gehört vor allem die Programmierung in **C** – nicht nur wegen der weiten Verbreitung – auch weil späterer Unterricht in Programmiersprachen wie **C++** oder **JAVA** darauf aufbauen kann. Grundbegriffe der Computertechnik Zahlendarstellung und Codes werden als nötiges Hintergrundwissen für die Programmierung vermittelt. Anwendungsorientierte Algorithmen und Beispiele bilden das nötige Anschauungsmaterial.

Durch eine stark ausgearbeitete Aufbereitung wollen wir den Einstieg erleichtern. Vor allem ist damit die durchgängige Strukturierung in kleine, überschaubare Lernschritte gemeint, sorgfältige Auswahl der Darstellungsmittel wie z. B. durchgängige Verwendung von Syntaxdiagrammen, viele Fragen und Aufgaben und nicht zuletzt viele Tips zum professionellen Arbeiten und zur Vermeidung typischer Fehler.

Unterstützend legen wir dem Buch eine CD bei, die die Voraussetzungen bietet, um alle Beispiele und Aufgaben aus dem Buch nachzuvollziehen. Enthalten ist der Borland®- **C** bzw. **C++** Compiler 5.5 sowie ein passender Freeware-Editor. Damit steht dem Leser eine Entwicklungsumgebung, zur Verfügung, mit der sich alle Aufgaben und Beispiele aus dem Buch programmieren lassen (einschließlich der Grafik-Anwendungen) – für Anfänger ein ideales Startset.

Entstanden ist das Buch auf der Basis von Vorlesungen und Praktika am Fachbereich Elektrotechnik der Fachhochschule Regensburg, die dort in den ersten zwei Semestern gehalten

werden. Es vermittelt das gemeinsame Basiswissen der Ingenieurinformatik, das in allen technischen Fächern benötigt wird. Damit zielt es in gleicher Weise auf die Informatikausbildung anderer Fachbereiche bzw. überhaupt auf Anfänger, die in C und Informatik für Ingenieure einsteigen wollen.

Wo die Studienordnung über die Inhalte des Buches hinaus Platz für Informatik bietet, ist es für den Einstieg vorgesehen, dem später weitere Vorlesungen und Praktika folgen sollen. Zum Beispiel ist es als Basis gedacht, auf der Themen wie objektorientierte Programmierung in C++ oder JAVA, Einsatz von Klassenbibliotheken, Software Engineering, betriebssystemnahe Themen, Datenbanken oder Internet-Themen aufbauen können.

Für die Unterstützung während der Arbeiten sind wir zu besonderem Dank verpflichtet

- unseren Familien, denen wir auch dieses Buch widmen. Sie haben es während der ganzen Zeit geduldig ertragen, dass uns das Buchprojekt stark absorbiert hat,
- Herrn Prof. Dr. Oechslein, der das Projekt unterstützt hat,
- Herrn Prof. Dr. Geupel, der das Projekt verlagsseitig betreut hat,
- dem Lektorat des Oldenbourg-Verlags, das unsere Ideen unterstützte.

Zur Zweiten Auflage

Gegenüber der ersten ist die zweite Auflage vollständig durchgesehen und fehlerbereinigt. Wir danken an dieser Stelle für die vielen Rückmeldungen von unseren Lesern und Rezensenten.

Die CD wurde überarbeitet. Statt der Cygnus® – Umgebung ist die für Anfänger etwas einfacher zu handhabende Entwicklungsumgebung mit dem Borland C++ Compiler 5.5 aufgenommen worden sowie ein passender Freeware- Editor für Programme, der Zeilennummern und Syntax-Highlighting von Schlüsselworten beherrscht.

Die Grafikpakete auf der CD wurden vereinheitlicht und um Funktionen erweitert, die von Lesern der 1. Auflage gewünscht wurden.

Axel Böttcher, Franz Kneißl

Inhalt

Vorwort V

1 Grundbegriffe der Computertechnik 1

1.1 Einführung ... 1

1.2 Anwendungsprogramme ... 3

1.3 Betriebssysteme ... 5

1.4 Hardware ... 7

1.5 Prozessoren, Busse und Speicher ... 8

1.5.1 Das Bussystem ... 8

1.5.2 Der Prozessor ... 10

1.5.3 Der Speicher ... 10

1.5.4 Peripheriegeräte ... 12

1.6 Die Befehlsebene ... 12

1.7 Die Logikebene ... 17

2 Zahlendarstellung 21

2.1 Zahlensysteme für ganze Zahlen ... 21

2.2 Rechnen mit Potenzen ... 22

2.3 Umwandlung zwischen Zahlensystemen ... 23

2.3.1 Zielverfahren: Multiplikationsmethode ... 24

2.3.2 Quellverfahren: Divisionsmethode ... 24

2.4 Rechnen im Dualsystem ... 25

2.5 Rechnerinterne Darstellung von ganzen Zahlen ... 27

2.5.1 Das Eins-Komplement ... 28

2.5.2 Das Zwei-Komplement ... 30

2.6 Darstellung und Umwandlung gebrochener Zahlen ... 32

2.6.1 Zielverfahren: Divisionsmethode ... 33

2.6.2 Quellverfahren: Multiplikationsmethode ... 33

2.7 Rechnerinterne Darstellung gebrochener Zahlen ... 34

2.8 Fragen ... 35
2.9 Aufgaben ... 35

3 Zeichencodes 37
3.1 7 Bit ASCII ... 38
3.2 8 Bit ISO 8859 ... 40
3.3 16 Bit Unicode ... 42
3.4 Fragen ... 43

4 Einführung in das Programmieren in C 45
4.1 Zur Geschichte von C ... 45
4.2 Erste Schritte ... 47
4.3 Syntaxdiagramme ... 49
4.4 Praxis des Programmierens ... 50
4.5 Aufgaben ... 52

5 Grundelemente, Variablen, Konstanten, Datentypen 53
5.1 Übersicht ... 53
5.2 Programmstruktur ... 54
5.3 Lexikalische Grundvoraussetzungen ... 55
5.3.1 Zeichensätze ... 55
5.3.2 Formatfreie Schreibweise ... 56
5.3.3 Bezeichner ... 57
5.3.4 Einschränkungen ... 58
5.4 Variablen und Konstanten ... 59
5.4.1 Variablen und Konstanten zur Compilezeit, Deklaration ... 59
5.4.2 Variablen und Konstanten zur Ladezeit ... 60
5.4.3 Variablen und Konstanten zur Laufzeit ... 61
5.4.4 Verschiedene Konstanten-Begriffe ... 62
5.5 Elementare Datentypen ... 63
5.5.1 Ganzzahlige Datentypen ... 64
5.5.2 Gleitpunkttypen ... 68
5.5.3 Beispielprogramm ... 69
5.5.4 Benutzerdefinierte Typen ... 71
5.6 Fragen ... 74
5.7 Aufgaben ... 75

6 Formatierte Ein- und Ausgabe 77

6.1 Formatierte Ausgabe ... 77
6.1.1 Die Formatelemente für formatierte Ausgabe ... 78
6.1.2 Beispiele ... 80
6.1.3 Fehlerquellen ... 81
6.2 Formatierte Eingabe ... 81
6.2.1 Beispiel zur formatierten Eingabe ... 82
6.2.2 Besonderheiten und Fehlerquellen ... 82
6.3 Aufgaben ... 84

7 Operatoren und Ausdrücke 85

7.1 Ein erstes Beispiel ... 85
7.2 Arithmetische Ausdrücke ... 88
7.3 Der Zuweisungsoperator ... 89
7.4 Zusammengesetzte Operatoren ... 90
7.5 Unitäre arithmetische Operatoren ... 90
7.6 Der Kommaoperator ... 91
7.7 Wahrheitswerte und logische Ausdrücke ... 91
7.8 Der konditionale Operator ... 93
7.9 Aufgaben ... 94

8 Logische und bitweise Operatoren 97

8.1 Logische Verknüpfungen ... 97
8.2 Bitweise Operatoren ... 98
8.3 Fragen ... 103
8.4 Aufgaben ... 103

9 Standardbibliothek 105

9.1 Ein/ Ausgabe ... 106
9.2 Datei-Ein/ Ausgabe ... 106
9.3 Grenzwerte ... 108
9.4 Mathematik ... 109
9.5 Zufallszahlen ... 110
9.6 Zeichenbehandlung ... 111
9.7 Zeichenketten ... 112
9.8 Konvertierung Intern-/ Extern-Darstellung ... 112

9.9 Speicherverwaltung ... 113
9.10 Starten/ Beenden ... 114
9.11 Nicht-Standardfunktionen ... 114
9.12 Aufgaben ... 115

10 Kontrollstrukturen 117

10.1 Bedingte Verzweigung ... 118
10.2 Auswahl (Fallunterscheidung) ... 121
10.3 Laufschleifen (Wiederholungsanweisungen) ... 122
10.3.1 Die `while`-Anweisung ... 123
10.3.2 Die `do-while`-Anweisung ... 125
10.3.3 Anwendung: Bestimmung von Nullstellen einer Funktion ... 125
10.3.4 Die `for`-Anweisung ... 128
10.4 Sprunganweisungen ... 129
10.4.1 Die `continue`- und `break`-Anweisungen ... 129
10.4.2 Die `goto`-Anweisung ... 130
10.4.3 Die `return`-Anweisung ... 130
10.5 Aufgaben ... 131

11 Präprozessor 133

11.1 Die `#include`-Direktive ... 133
11.2 Symbolische Konstanten ... 134
11.3 Vordefinierte Symbole ... 135
11.4 Makros ... 136
11.5 Bedingte Compilierung ... 137
11.6 Beispielprogramm: Testversion Newton-Raphson ... 139

12 Algorithmen: Reaktive Programme, Automaten 141

12.1 Endliche Automaten ... 143
12.2 Direkte Implementierung von Automaten ... 145
12.3 Beispielprogramm: Verkaufsautomat ... 146
12.4 Erkennende Automaten ... 148
12.5 Aktionen in Automaten-Programmen ... 150
12.6 Fragen ... 151
12.7 Aufgaben ... 152
12.7.1 DFÜ-Protokolle ... 152
12.7.2 Filter für Escape-Sequenzen in HTML Dateien ... 154

13 Vektoren 157

13.1 Abgeleitete Typen in C, Übersicht ... 157

13.2 Eindimensionale Vektoren ... 158
13.2.1 Deklarationssyntax ... 158
13.2.2 Zugriff auf Ganz- und Komponenten-Variable ... 159

13.3 Zur Deklarations-Syntax in C ... 161

13.4 Mehrdimensionale Vektoren ... 162

13.5 Fragen ... 164

13.6 Aufgaben ... 165

14 Algorithmen: Sortierverfahren, Zufallszahlen 167

14.1 Sortieren ... 167
14.1.1 Bubblesort ... 168
14.1.2 Sortieren durch Auswahl ... 168
14.1.3 Bucket Sort ... 169

14.2 Zufallszahlen ... 170
14.2.1 Ein Simulator ... 171
14.2.2 Zufallszahlen mit bestimmten Eigenschaften ... 173

14.3 Fragen ... 176

14.4 Aufgaben ... 177

15 Algorithmen: Lineare Gleichungssysteme 181

15.1 Die Gauß-Elimination ... 181

15.2 Algorithmus ... 183

15.3 Programm ... 183

15.4 Aufgaben ... 184

16 Pointer 185

16.1 Übersicht ... 185

16.2 Programmierung mit Pointern ... 185
16.2.1 Funktionsprinzip ... 185
16.2.2 Syntax ... 187
16.2.3 Zugriff auf Pointer oder Bezugsvariable ... 187
16.2.4 Pointer ohne Bezugsvariable, Nullpointer, void* ... 189

16.3 Pointer und Vektoren ... 190
16.3.1 Vektoren und Pointer-Arithmetik ... 190
16.3.2 Vektorzugriff in Pointerschreibweise ... 191
16.3.3 Vektoren von Pointern, Pointer auf Vektoren ... 192

16.4 Dynamische Variable mit `malloc` und `free` ... 195

16.5 Auswahl-Sort durch Zeigervertauschung ... 197
16.6 Pointer und `const` ... 198
16.7 Fragen ... 199
16.8 Aufgaben ... 201

17 Unterprogramme 203
17.1 Syntax ... 204
17.2 Der Parametermechanismus ... 207
17.3 Referenzparameter ... 209
17.4 Lokale, globale und statische Variablen ... 212
17.5 Funktionsdeklarationen, Modularisierung und Headerdateien ... 215
17.6 Fragen ... 219
17.7 Aufgaben ... 219

18 Algorithmen: Grafikausgabe 221
18.1 Programmpaket für Grafikausgaben ... 221
18.2 Kurven zeichnen ... 224
18.3 Programmiertechniken: Funktion als Argument ... 225
18.4 Aufgabe ... 226
18.5 Koordinatentransformationen ... 227
18.6 Aufgabe ... 229
18.7 Professionelle Programmiertechniken am Beispiel Koordinatentransformation . 229
18.7.1 Namensgebung und Bezeichner ... 230
18.7.2 Programmstruktur ... 232
18.7.3 Beispielprogramm: Koordinatentransformation ... 232
18.8 Aufgaben ... 234

19 Dateien 235
19.1 Der Datentyp `FILE` ... 236
19.2 Formatierte Ein-/Ausgabe ... 237
19.2.1 Formatierte Ausgabe mit `fprintf()` ... 237
19.2.2 Formatierte Eingabe mit `fscanf()` ... 238
19.2.3 Weitere Funktionen für das formatierte Einlesen von Datei ... 238
19.3 Standarddateien ... 239
19.4 Binäre Ein-/Ausgabe ... 239
19.5 Aufgaben ... 241

20 Structs und komplexe Datenstrukturen 243
20.1 Strukturen mit `struct` ... 243
20.2 Zeiger auf Strukturen ... 246
20.3 Anwendungsbeispiel: Komplexe Zahlen ... 246
20.4 Listen ... 247
20.5 Exkurs: Rekursive Funktionen ... 251
20.6 Aufgaben ... 252

21 Algorithmen: Graphentheorie 253
21.1 Problemstellung ... 253
21.2 Darstellung von Graphen durch Matrizen ... 256
21.3 Der Algorithmus von Dijkstra ... 257
21.4 Fragen ... 262
21.5 Aufgaben ... 262

22 Algorithmen: Interpretative Implementierung von Automaten 263
22.1 Programmiertechniken: Pointer auf Funktionen ... 263
22.2 Schema zur Umsetzung in Programme ... 264
22.3 Beispielprogramm ... 266
22.4 Fragen ... 268
22.5 Aufgaben ... 269
22.5.1 Erzeugung eines Paritätsbits ... 269
22.5.2 Escape-Sequenz-Filter interpretativ ... 269

23 Fortgeschrittene Themen 271
23.1 Typumwandlungen ... 271
23.1.1 Anlässe von Typumwandlungen ... 271
23.1.2 Art der Typumwandlungen ... 273
23.2 Union-Typen ... 276
23.3 Argumente und Rückgabewert von `main(...)` ... 279

24 Literatur 281
Index 283

1 Grundbegriffe der Computertechnik

1.1 Einführung

In den vergangenen Jahrzehnten hat die Computertechnik eine rasante Entwicklung durchgemacht.

- Die Hardware hat sich hinsichtlich Geschwindigkeit, Kapazität und Komplexität alle 1,5 Jahre nahezu verdoppelt.
- Betriebssysteme unterstützen jeden PC-Besitzer mit Eigenschaften, die es früher nur bei Großrechnern oder teuren Workstations gab.
- In der Anwendungssoftware entstanden Produkte, die von jedermann intuitiv zu bedienen sind und die nützliche Funktionen für jedermann bieten.
- Die Verbreitung des Internet ist explosionsartig gestiegen. Computer in Firmen sind ohne Vernetzung gar nicht mehr vorstellbar.

Tabelle 1.1-1 zeigt Meilensteine und Entwicklungen, auf die in den folgenden Kapiteln Bezug genommen wird.

Auf Grund dieser Entwicklungen sind Computer auf breiter Basis in jeden Lebensbereich und in alle Winkel der Welt vorgedrungen. Der stärkste Motor dieser Entwicklung war natürlich der PC – der Computer für jeden – der die massenhafte Verbreitung getragen hat[1]. Wachsende Stückzahlen ermöglichten immer komplexere Prozessoren zu immer geringeren Kosten. Damit einher ging ein starker Druck zur Standardisierung. Heute stellen nur noch wenige Firmen komplexe Prozessoren her und auch die Anzahl der Anbieter von Chipsätzen, die um die Prozessoren herum benötigt werden, ist überschaubar geworden.

Der weitaus überwiegende Teil der Ingenieure, die heute mit Computern zu tun haben, entwirft daher keine neuen Computersysteme, sondern ist mit der Erstellung von Software beschäftigt. Aus diesem Grund wird in den folgenden Kapiteln der Hardware-Teil relativ kurz behandelt.

---

[1] Deswegen und wegen des wahrscheinlich hohen Anteils der PC-Besitzer unter den Lesern sind im vorliegenden Kapitel die Entwicklungen und Meilensteine vorwiegend aus der PC-Perspektive dargestellt.

Tabelle 1.1-1: Zeittafel mit verschiedenen Meilensteinen und Entwicklungen der letzten 30 Jahre

	Anwendungs-software	Betriebssysteme	Hardware	Internet
1970	1969 Programmier-sprachen bilden die Schnittstelle des Anwenders zum Computer 1980 XEROX Smalltalk 80	Großrechnersysteme 1969 UNIX V1 1980 XEROX Mesa 1980 CP/M	1970 niedriger Integ-rationsgrad 1970 VLSI Technik verfügbar 1975 Intel 8080	1969 ARPA Net 1972 eMail
1980	PC-Anwendungen - Tabellenkalkulat. - Textverarbeitung - Finanzverwaltung - Datenbank - Taschenrechner - Spiele etc.	1981 MS-DOS 1990 Windows 3.0	1980 Intel 8086 1980 0,1 MIPS 1981 IBM-PC mit 8086 und MS DOS 1985 Intel 80286 1987 Intel 80386 1990 Intel 80486	1980 TCP/ IP
1990	Integrierte Anwendungen Grafische Oberflächen Internet-Anwendungen Multimedia	1994 Linux 1.0 1994 OS/2 1994 Windows NT 1995 Windows 95 1998 Windows 98	1990 10 MIPS 1993 Intel Pentium 1995 Pentium Pro 1997 Pentium II 1999 Pentium III	1991 WWW, HTML 1992 Mosaic, Net-scape, Explorer 1995 Java 1999 Jini
2000	Componentware	2000 Windows 2000	2000 100 MIPS	

Für den Ingenieur von besonderer Bedeutung ist die breite Durchdringung von technischen Produkten durch Computer. Unser heutiger technologischer Standard ist ohne Einsatz von Computertechnik gar nicht denkbar. Als Beispiele seien nur einige wenige Bereiche genannt:

- Fahrzeugtechnik: In einem Fahrzeug werden alle sicherheitsrelevanten Funktionen durch embedded controls gesteuert. Dazu gehören die Airbagsteuerung, Antiblockiersysteme oder elektronische Differentialsperren. Ferner werden bald elektronische Navigationssysteme zur Standardausstattung für Kraftfahrzeuge gehören.
- Produktionstechnik
- In Flugzeugen ist das sog. "Fly-by-wire" Stand der Technik. Dabei erkennen intelligente Sensoren die Aktionen des Piloten, übertragen die Informationen auf elektronischem Weg zu den gesteuerten Systemen und steuern die betroffenen Komponenten.
- Moderne Umweltmesssysteme stützen sich auf Computertechnik zur Sammlung und Auswertung der Messdaten von Umweltsensoren.
- Telekommunikation: umfangreiche Programme steuern die Mobilfunknetze und Handies und alle anderen Telekommunikationssysteme, die uns viele Annehmlichkeiten bringen.

Mit dieser Entwicklung hat sich der Anteil der Software-Erstellung an der Ingenieurstätigkeit dramatisch erhöht. Heute besteht etwa die Wertschöpfung bei Telekommunikationsprodukten zum überwiegenden Teil aus der Entwicklung von Software.

In der Entwicklung von technischen Anwendungen haben die Programmiersprache **C** und objektorientierte Sprachen wie **C++** oder Java heute weite Verbreitung. Nachdem **C++** und Java auf **C** aufbauen, ist heute die Programmierung in der Sprache **C** unabdingbares Basiswissen, das sich jeder angehende Ingenieur aneignen muss.

Die Programmiersprache **C** macht daher den größten Teil dieses Buches aus. Zuvor werden in den Kapiteln „Grundbegriffe der Computertechnik", „Zahlendarstellung" und „Zeichencodes" Grundkenntnisse vermittelt, die als Hintergrundwissen für die Programmierung notwendig sind.

1.2 Anwendungsprogramme

> „Anwendungssoftware (application software), auch Applikationssoftware genannt, ist Software, die Aufgaben des Anwenders mit Hilfe eines Computersystems löst." [Balzert 1996]

Die umwälzenden Veränderungen auf dem Gebiet der Anwendungssoftware sind vor allem auf geänderte Vorstellungen davon zurückzuführen, was ein Anwender ist.

Bis Ende der siebziger Jahre waren dies Institutionen oder Organisationen, die sich zur Wahrnehmung ihrer Aufgaben elektronischer Datenverarbeitungsanlagen bedienten. Dort wurden Spezialisten beschäftigt, die mit der DV-Anlage umgehen konnten. Die wichtigste Schnittstelle zum Computer waren damals Programmiersprachen.

Wenn damals z. B. das Problem $\sqrt{sin(1.7)}$ zu lösen war, schrieb der Programmierer ein Programm, übersetzte und startete es und bekam das Ergebnis ausgedruckt. Für die Bedienschritte war das Eintippen einer Reihe von kryptisch aussehenden Kommandozeilen nötig, wie man sie später auch noch während der Ära des DOS-PC kannte.

Bewegung kam durch die Arbeiten am XEROX Palo Alto Research Center in die Szene. Dort wurde Ende der siebziger Jahre erforscht, wie man Computer allgemein zugänglich machen kann. Eine Fragestellung dabei war, wie man die Bedienung gestalten muss, um sie für Jedermann handhabbar zu machen.

Das Ergebnis waren die Einführung von Dingen, die heute jedem selbstverständlich sind: Einsatz von Grafik- statt Text-orientierten Bildschirmen, fensterorientierte grafische Bedienoberflächen, Maus, das WYSIWYG-Prinzip („What You See Is What You Get") usw[2].

XEROX hat aus diesen bahnbrechenden Arbeiten selbst nicht den kommerziellen Erfolg machen können, den später andere Firmen (z. B. Apple oder Microsoft) damit hatten.

Ein großer Teil des Umsatzes mit Computern wird heute mit privaten Anwendern gemacht – die Utopie vom Computer für Jedermann ist damit weitgehend verwirklicht.

---

[2] Ein gutes Bild über die damalige Aufbruchsstimmung kann man sich durch die Darstellung im Sonderheft zu Smalltalk 80 der Zeitschrift Byte, Jahrgang 1980, machen.

Ein entscheidender Schritt auf diesem Weg war die Entwicklung einer bis dahin völlig neuen Art von Software, die sich um die Bedürfnisse von privaten Kunden bemüht.

Wenn heute ein PC-Besitzer wissen will, wieviel $\sqrt{sin(1.7)}$ ist, dann startet er einfach die Taschenrechner-Anwendung auf seinem Desktop und tippt `1.7 sin √` , wonach `0,9958236844203` erscheint.

Die wichtigsten Kategorien von PC-Anwendungen finden sich in Tabelle 1.1-1.

In den neunziger Jahren haben sich neue Trends abgezeichnet:

- Integrierte Anwendungen
- Componentware
- Client/ Server-Anwendungen
- Multimedia

„**Integrierte Anwendungen**" bedeutet, dass mehrere Anwendungen zusammenarbeiten. Man setzt heute z. B. voraus, dass Ergebnisse der Tabellenkalkulation mit der Maus in ein Textdokument gezogen werden können.

„**Componentware**" bedeutet, dass nicht nur kleine Funktionen aus einer Standardbibliothek (vgl. Kapitel „Standardbibliothek") verfügbar sind, sondern dass komplette Anwendungen in das eigene Programm eingebunden werden können. So kann man z. B. in ein kleines Programm zur Auswertung von Daten, das man selbst schreibt, eine komplette Datenbank als Bestandteil einbinden.

Client/ Server-Anwendungen laufen nur zum Teil auf dem Computer des Benutzers (Client). Durch Kommunikation über das Netz tauscht Client Information mit der Server-Seite aus und kann so Dienstleistungen des Servers auf dem Computer des Benutzers erbringen.

Multimedia-Anwendungen beziehen Geräusche, synthetische oder echte Musik, Zeichnungen, Photos, berechnete Szenen und Videos in die Anwendung mit ein. Der Erlebnis-Charakter der PC-Welt wird damit dem Film und Fernsehen noch ähnlicher – und der erreichbare Kreis von potentiellen Benutzern noch größer.

Für Benutzer von Computern bringen die obengenannten Errungenschaften eine kolossale Steigerung des Handhabungs-Komforts. Nicht so gut sieht die Sache für den Programmierer aus. Er benutzt zwar integrierte Entwicklungsumgebungen, wo er fast unbemerkt zwischen unzähligen Tools – vom Texteditor bis zum Debugger – grafisch fensterorientiert navigiert, aber sein wesentliches Kommunikationsmedium ist eine Programmiersprache. Insofern erinnert seine Tätigkeit an die Ära vor dem PC.

Programme, die grafische Bedienoberflächen realisieren, sind wesentlich komplexer, als Programme im Textmodus. Letztere sind daher für den Einstieg in die Programmierung didaktisch zu bevorzugen. Für Einsteiger in die Programmierung kommt daher noch eine kleine Enttäuschung hinzu: die Übungsprogramme arbeiten „nur" mit Text-Ein/ -Ausgabe.

In diesem Buch ist zwar nicht Platz für richtige Windows-Applikationen, aber wir wollen zumindest eine Brücke zur Welt der Grafik schlagen. In Kapitel „Algorithmen: Grafikausgabe", werden wir wenigstens das Zeichnen mit Grafik-Primitiven kennen lernen und in verschiedenen Übungen und Aufgaben davon Gebrauch machen. Zu diesem Zweck gibt es als Material zum Buch ein Grafik-Paket für die Borland- und die Microsoft-Entwicklungsumgebungen.

1.3 Betriebssysteme

Ein Benutzer greift nicht direkt auf seinen Computer zu, sondern mit Hilfe eines Anwendungsprogramms. Ähnlich greift das Anwendungsprogramm nicht direkt auf die Hardware zu, sondern mit Hilfe des Betriebssystems.

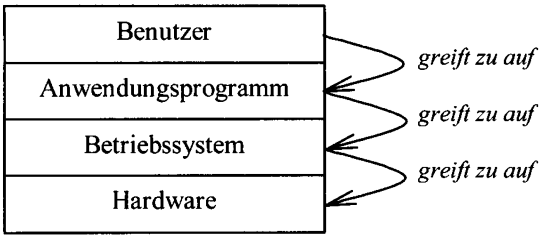


Abbildung 1.3-1: Schichtenmodell: Benutzer/Anwendung/ Betriebssystem/Hardware

Programme können die Dienste, die sie vom Betriebssystem benötigen, über sogenannte „system calls" aufrufen. Die verfügbaren system calls sind für das jeweilige Betriebssystem vorgegeben und als dessen API (Application Programming Interface) definiert.

Betriebssysteme leisten die folgenden Dienste

- Systemhochlauf
  - Hardware-Test
  - „Bootstrap" Laden des Betriebssystems
  - Registrierung der Hardware- und Ein-/Ausgabe-Geräte-Konfiguration
- Kommandoausführung
  - Kommandozeilen-Interpretation (z. B. DOS-Box, Shell)
  - Ausführung der Programme, die die Kommando-Funktionen realisieren
  - Grafische Bedienoberflächen mit Desktop, Fenstern, Menüs, Dialogen etc.
- Programmabwicklung
  - Start/ Beenden von Programmen
  - Zuteilung des Prozessors an laufende Programme
- Betriebsmittelverwaltung
  - Arbeitsspeicher
  - Dateisystem
  - Zugang zu angeschlossenen Ein-/Ausgabe-Geräten
  - Zugang zum Internet/ lokalen Netz
- Hardwareanpassung
  - Kommunikation mit den physikalischen Ein-/Ausgabe-Geräten
  - Bearbeitung von Unterbrechungsanforderungen von Geräten

Es wird unterschieden, ob Betriebssysteme Umgebungen (Einstellungen, Verzeichnisse, Berechtigungen etc.) für einen oder für mehrere Benutzer verwalten können. Entsprechend wird das System als multi user- (z. B. LINUX) oder single user- (z. B. DOS) System bezeichnet.

Hinsichtlich der Programmabwicklung unterscheidet man multi tasking und single tasking-Betriebssysteme. Die Eigenschaft gibt an, ob das Betriebssystem mehrere Programme simultan laufen lassen kann (z. B. Windows NT, LINUX) oder nur eines (z. B. DOS). Ein in Ausführung befindliches Programm wird auch als „task" oder „Prozess" bezeichnet.

In jedem Fall wird jedem Anwendungsprogramm eine Umgebung geboten, als ob es alleine auf dem Computer liefe[3]. Bei nur einer Task entspricht die Umgebung der physikalischen Maschine. Wenn mehrere Tasks gleichzeitig aktiv sind, spricht man von einer „virtuellen Maschine", die für jede Task bereitgestellt wird. Die Betriebsmittel der physikalischen Maschine müssen – unmerklich für die Tasks – vom Betriebssystem reihum abwechselnd allen Tasks zugeteilt werden.

Anfang der siebziger Jahre hatte jeder große Computerhersteller (damals gab es noch mehr als heute) sein eigenes Betriebssystem. Meist waren dies multi tasking-Großrechnersysteme, später auch Multi-User fähig.

Mit dem Aufkommen von Mikroprozessoren (vgl. Kapitel 1.4) begann ein neuer Zweig in der Evolution der Betriebssysteme. Die neuen Computer waren anfangs nur mit sehr schwacher Hardware ausgestattet und nur für jeweils einen Benutzer gedacht. Daher entwickelte man einfache single tasking/ Single user Betriebssysteme.

1980 hatte die Firma Digital Research mit CP/ M die Nase bei den Mikroprozessor-Betriebssystemen vorne. Trotzdem erhielt Microsoft den Zuschlag für das IBM-PC-Betriebssystem und so startete die PC-Ära 1981 mit MS-DOS – einem single tasking/ single user System mit textbasierter Bedienung[4].

1981 hatte die Avantgarde bei XEROX mit ihrem Mesa-System bereits gezeigt, wie ein Betriebssystem aussehen musste, das fensterorientierte grafische Bedienung besonders unterstützte – es war natürlich mit Multitasking ausgestattet. Trotzdem dauerte es noch etwa 15 Jahre, bis sich auf dem PC-Massenmarkt mit Windows NT, OS/2 oder LINUX „echte" multi tasking Systeme durchsetzten.

Als Vorstufe schaffte 1990 Microsoft Windows 3.0 den Durchbruch. Diese Version hatte erst sehr eingeschränkte Multitasking-Fähigkeiten. Aber immerhin wurde die textbasierte Oberfläche von DOS durch fensterorientierte grafische Bedienung abgelöst.

Die größte Verbreitung haben heute 3 Linien von PC-Betriebssystemen:

- Windows 9x als populäre Zwischenformen auf dem Weg von Windows 3.x zu Windows NT
- Windows NT als Vertreter des PC-Zweigs der Multitasking/Multiuser-Betriebssysteme
- LINUX als später Nachfolger[5] der Großrechnersysteme – inzwischen sind dort natürlich ebenfalls fensterorientierte grafische Bedienoberflächen verfügbar.

---

[3] bis auf eine etwaige Verlangsamung, wenn der Rechner mehrere Programme gleichzeitig abwickelt

[4] Es geht die Legende, daß ein Digital Research Repräsentant namens Gary Kildall wegen seinem Fliegerei-Hobby Gespräche mit IBM-Vertretern verpaßte und so Bill Gates zu dessen Chance verhalf. Wie wir wissen, hat dieser sie sehr gut genutzt ...

[5] Neuimplementierung durch Linus Torvalds

1.4 Hardware

Die bisher geschilderte Entwicklung wäre gar nicht möglich gewesen, wenn nicht die Hardware ca. alle 2 Jahre ihre Geschwindigkeit und Kapazität verdoppelt hätte. Von 0,1 MIPS[6] 1980 ist man im Jahr 2000 bis zu 100 MIPS fortgeschritten.

Diese Steigerung bei fallenden Preisen wurde durch den Einstieg in eine Technik ermöglicht, die um 1970 die nötige Reife erlangt hatte: VLSI („Very Large Scale Integration"), also die Technik, große Mengen von Schaltkreisen auf einen einzigen Silizium-Chip aufzubringen. Der Urvater 8080 der Intel-Familie war einer der frühen Prozessoren auf VLSI-Basis, die in großen Stückzahlen hergestellt wurden.

Die Zeittafel in Tabelle 1.1-1 zeigt die Intel-Linie. Von Generation zu Generation „wanderten" immer mehr Bausteine aus den Chipsätzen, die auf der Hauptplatine den Prozessor umgeben, auf das Prozessor-Chip selbst.

Damit sind die Prozessoren sehr komplex geworden. Bei mehr als 10 Millionen Transistoren ist es nicht mehr möglich, einen Prozessor komplett auf einem Schaltplan darzustellen.

Um Prozessoren zu betrachten, muss man sich daher auf verschiedenen Abstraktionsniveaus bewegen. Abbildung 1.4-1 zeigt fünf gebräuchliche Betrachtungsebenen.

Auf jeder Ebene werden bestimmte Elemente dargestellt, und wie diese miteinander in Verbindung stehen. Dabei wird von den Details abstrahiert, die auf niedrigeren Ebenen gezeigt werden. Anders ausgedrückt: jede Ebene stellt dar, wie die Elemente der darüberliegenden „von innen" aussehen bzw. funktionieren.

Prozessoren, Busse, Speicher	Die Ebene stellt dar, welche Hauptelemente das System besitzt, und wie diese miteinander verbunden sind
Befehle	Die Ebene stellt den Satz von Befehlen (Maschinenbefehle) dar, den der Prozessor beherrscht
Register-Transfer	Die Ebene stellt die Register[7] dar, wie diese mit ihrer Umgebung verbunden sind und welche Verknüpfungen vorgesehen sind, um den Befehlssatz zu realisieren.
Logik	Die Ebene stellt Gatter[8] und Flipflops[9] dar und wie diese verbunden sind, um Register, Verknüpfungen und Transfers zu realisieren.
Hardware-Realisierung	Die Ebene stellt Bauteile (z. B. Transistoren, Kondensatoren etc.) dar und wie diese verbunden sind, um Gatter oder Flipflops zu realisieren

Abbildung 1.4-1: Betrachtungsebenen

[6] MIPS = Millionen Maschinen-Instruktionen pro Sekunde

[7] Ein n Bit breites Register besteht aus n Flipflops, die jeweils ein Bit speichern können. Die Flipflops sind so mit ihrer Umgebung verschaltet, dass vorgesehene Transfers oder Verknüpfungen möglich sind.

[8] Schaltungen zum Verknüpfen von Bits

[9] Schaltungen zum Speichern von Bits

1.5 Prozessoren, Busse und Speicher

In diesem Abschnitt werfen wir einen kurzen Blick auf die einzelnen Komponenten eines Computers. Wir beschränken uns dabei vornehmlich auf die Darstellung der weit verbreiteten Personal Computer.

1.5.1 Das Bussystem

Zentraler Bestandteil eines Computers ist das Bussystem. Es verbindet alle Komponenten des Computers untereinander und ermöglicht den Transport von Informationen zwischen ihnen. Ferner können über das Bussystem Verbindungen zur Außenwelt hergestellt werden. Ein Bus besteht primär aus Leitungen zur Übertragung elektrischer Signale zwischen den angeschlossenen Komponenten. Die elektrischen Signale repräsentieren die zu übertragende Information. Busleitungen, die zum Datenaustausch dienen, heißen Datenleitungen. Sind mehr als zwei Komponenten angeschlossen – was die Regel ist – so verfügt ein Bus auch über einen Mechanismus, der es der sendenden Komponente erlaubt, eine andere als Empfänger anzugeben. Jede angeschlossene Komponente erhält dazu eine Adresse. Die Adresse der jeweils angesprochenen Komponente wird über die sog. Adressleitungen des Busses übertragen. Daneben verfügt jeder Bus auch über Steuerleitungen, über die Kontrollinformation ausgetauscht wird. Diese kann etwa dem Empfänger der Information mitteilen, was er damit zu tun hat. Typische Geräte, die an Busse angeschlossen werden, sind Steuergeräte (Controller) für Tastatur und Maus, Festplatten und Diskettenlaufwerke sowie Grafik- und Netzwerkkarten,

Heute übliche Bussysteme sind international standardisiert. Damit haben viele verschiedene Hersteller die Möglichkeit, Komponenten zu liefern, die am jeweiligen Bussystem betrieben werden können. Komponenten gleicher Funktionalität können beliebig gegeneinander ausgetauscht werden. Für jede Anwendung existiert eine breite Palette von Angeboten, die auf unterschiedliche Bedürfnisse und Erfordernisse zugeschnitten sind. So gibt es kostengünstige Angebote, die speziell den Massenmarkt der Heimanwender bedienen bis hin zu solchen Angeboten, die auf spezielle professionelle Anwendungen zugeschnitten sind und sich preislich in der Größenordnung eines Kleinwagens bewegen.

Abbildung 1.5-1 gibt stark vereinfacht einen Überblick über die heute gängigen Bussysteme. Die wesentlichen Bustypen sind:

- X-Bus
  Dies ist der älteste der noch verwendeten Busse. Seinen Namen verdankt er dem IBM PC/XT, dem Urvater unserer heutigen PCs. Mit ihm werden alle historischen Komponenten, wie Uhr, Tastatur und Maus angesteuert.
- ISA-Bus (Industry Standard Architecture)
  Der ISA-Bus ist ein langsamer Bus, der hauptsächlich noch aus Kompatibilitätsgründen für ältere Einsteckkarten verwendet wid.
- PCI-Bus (Peripheral Component Interconnect)
  Dies ist der derzeit übliche Standard-Bus, der über 32 Adress- und 32 Datenleitungen verfügt und hohe Übertragungsgeschwindigkeiten ermöglicht.

- SCSI-Bus (Small Computer Systems Interface)
  Der SCSI-Bus wird insbesondere für Geräte für professionelle Anwendungen eingesetzt.
- USB (Universal Serial Bus)
  Der USB ist als Allzweck- Bus für ein weites Spektrum von Externgeräten gedacht, die bisher typischerweise über die serielle oder parallel- Schnittstelle angeschlossen waren.

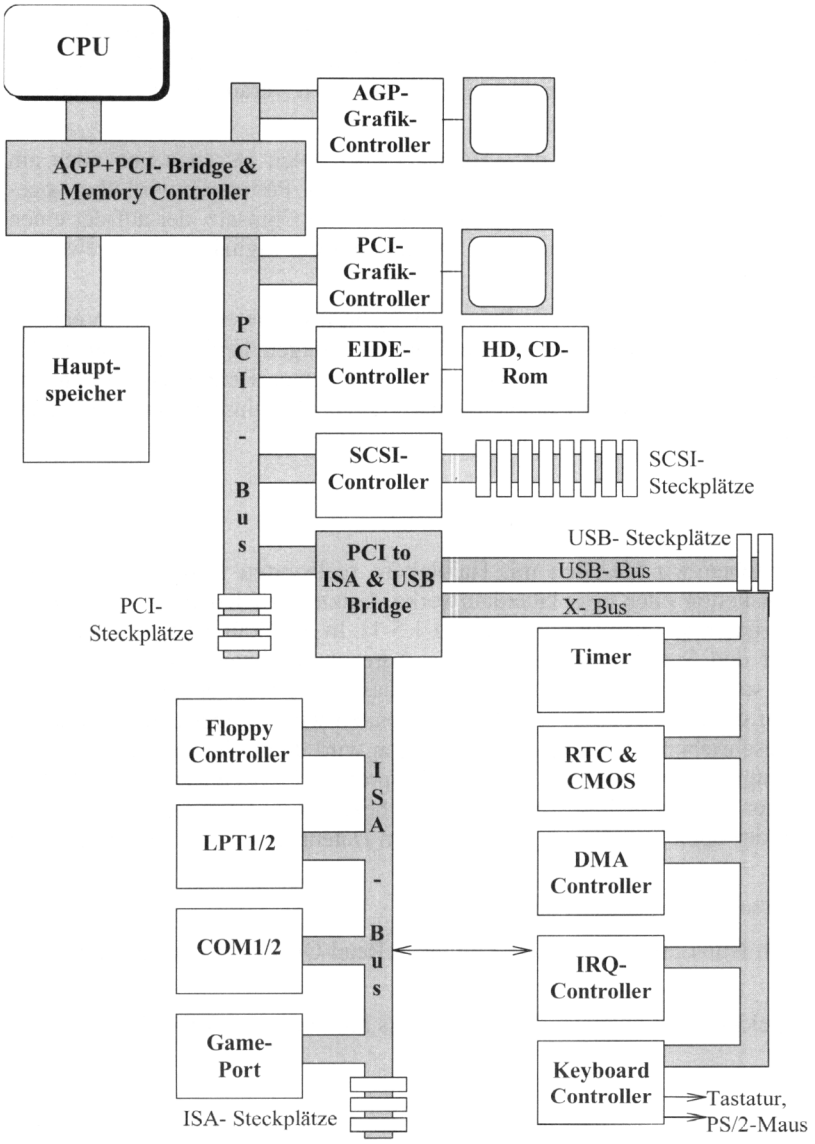


Abbildung 1.5-1: PC mit den gängigen Bussystemen und der damit verbundenen Peripherie

1.5.2 Der Prozessor

Kernstück jedes Rechners ist der Prozessor[10] (Central Processing Unit, kurz CPU). Er ist das zentrale Organ zur Manipulation von Daten. Der Teil des Bussystems, der direkt mit dem Prozessor verbunden ist, heißt Prozessorbus. Der Prozessorbus ist auf die Verbindung des Prozessors mit dem Speicher zugeschnitten. Die Anzahl der Adress- und Datenleitungen bei modernen Prozessorbussen beträgt stets ein Vielfaches von acht. Üblich sind heute bis zu 64 Leitungen.

Aus dem Hauptspeicher liest der Prozessor Instruktionen ein. Er interpretiert diese Instruktionen und führt sie aus. Die Ausführung von Instruktionen bedeutet meist, Daten zu manipulieren. Dazu sind die entsprechenden Daten einzulesen, zu verändern und zurückzuschreiben.

Der Prozessor kann Daten sowohl aus dem Hauptspeicher holen als auch von allen am Bussystem angeschlossenen Komponenten. Die Verbindung zur Peripherie wird über sogenannte „Bridges" hergestellt. Die AGP- und PCI-bridge ist ein Chipsatz, der auf der einen Seite wie Speicher angesprochen wird und auf der anderen die Signale für das jeweilige Bussystem zur Verfügung stellt.

Ein Prozessor verfügt ferner über spezielle Steuerleitungen, über die er zur Unterbrechung seiner Arbeit aufgefordert werden kann. Diese Unterbrechungen heißen Interrupts. Als Folge eines Interrupts unterbricht der Prozessor seine momentane Befehlsausführung, um etwa eine wichtigere Aufgabe zu übernehmen und nach deren Abschluss zur ursprünglichen Aufgabe zurückzukehren.

1.5.3 Der Speicher

Unter Speicher verstehen wir hier den mit Halbleitern realisierten Hauptspeicher eines Computers, nicht Disketten oder Festplattenlaufwerke. Letztere zählen in unserer kurzen Übersicht zu den Peripheriegeräten (vgl. Abbildung 1.5-1). In Abbildung 1.5-2 ist das Prinzip des Zugriffs auf den Speicher dargestellt. Die Adresse des Speicherwortes, auf das zugegriffen werden soll wird über den Adressbus in das Adressregister geschrieben (A). Soll ein Datum unter dieser Adresse gespeichert werden, so wird es über den Datenbus in das Datenregister geschrieben (D). Über Steuerleitungen wird dem Speicher mitgeteilt, ob Lese- oder Schreibzugriff gewünscht wird (R/W). Entsprechend wird entweder das Datum aus dem Datenregister an die durch das Adressregister bezeichnete Speicherzelle geschrieben, oder der Inhalt der Speicherzelle ausgelesen und im Datenregister zur Abholung bereit gestellt.

Es werden zwei Klassen von Speichern unterschieden.

- Festspeicher auch Nur-Lese-Speicher oder englisch Read-Only Memory, abgekürzt mit ROM.
- Schreib-Lese-Speicher oder englisch Random Access Memory, abgekürzt RAM.

[10] Ein Rechner kann auch mehrere Prozessoren umfassen, Wir beschränken unsere Darstellung auf Einprozessorsysteme.

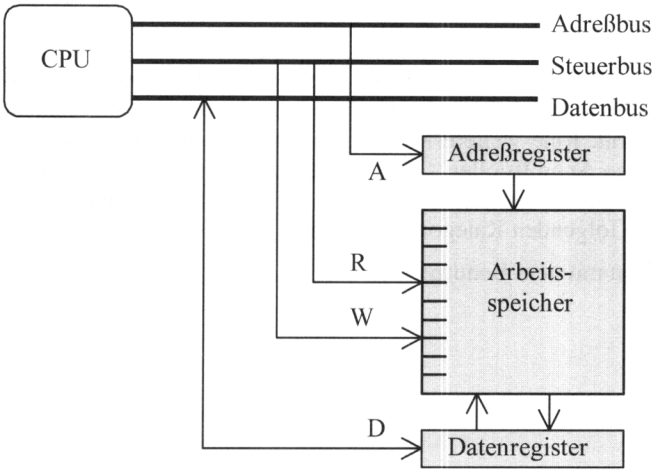


Abbildung 1.5-2: Prinzipdarstellung des Zugriffs auf den Speicher

Random Access bedeutet übersetzt „wahlfreier Zugriff", was hier soviel bedeutet, als dass auf den Speicher jederzeit beliebig lesend oder schreibend zugegriffen werden kann. Bei den ROM-Speichern gibt es verschiedene Arten, darunter auch solche, die sich löschen und wieder beschreiben lassen. Bei diesen Speichern ist das (Wieder-) Beschreiben ein aufwendiger Vorgang und es kann nicht zu jeder Zeit beliebig gelesen oder geschrieben werden. Tabelle 1.5-1 gibt einen Überblick über die verschiedenen Arten von Speichern.

Tabelle 1.5-1: Verschiedene Arten von Speichern

Speicherklasse	Speichertyp	Name und Beschreibung
Schreib-/Lesespeicher	SRAM	Statisches RAM. Diese Speicher sind sehr schnell, erfordern aber den Einsatz von sechs Transistoren je gespeichertem Bit.
	DRAM	Dynamisches RAM. Dynamische RAMs lassen sich mit weniger Aufwand realisieren, als SRAMs (ein Transistor pro Bit), sind aber langsamer. Der Speicherinhalt wird in einem Kondensator gespeichert, der seine Ladung mit der Zeit verliert und daher ständig wieder aufgeladen werden muss (Refresh).
Nur-Lesespeicher (Festspeicher)	MROM	Maskenprogrammierbares ROM. Es wird mit dem Herstellungsprozess programmiert. Der Speicherinhalt lässt sich nicht mehr ändern.
	PROM	Programmierbares ROM. Diese Speicher sind vom Anwender einmalig mit einem speziellen Programmiergerät programmierbar. Der Speicherinhalt lässt sich danach nicht mehr ändern.
	EPROM	Erasable PROM. Diese Speicher sind vom Anwender mit einem speziellen Programmiergerät programmierbar. Der Speicherinhalt lässt sich durch Bestrahlung mit einer UV-Lampe wieder löschen und neu beschreiben. Bausteine, deren Inhalt sich durch Anlegen von elektrischen Signalen löschen lässt, heissen Flash- EPROM. Wenn sich nicht nur der komplette Inhalt, sondern selektiv einzelne Adressen löschen lassen, spricht man von EEPROM (Electrically Eraseable ~).

1.5.4 Peripheriegeräte

Ein Computer verfügt über verschiedene Peripheriegeräte, die mit der Außenwelt in Kontakt stehen. Wir haben sie in Abbildung 1.5-1 bereits als die an das Bussystem angeschlossenen Komponenten kennengelernt. Eine weitere Möglichkeit ist der indirekte Anschluss über eine der dafür vorgesehenen Schnittstellen (Tastaturschnittstelle, Mausschnittstelle, Parallel-Schnittstelle (LPTn), serielle Schnittstelle (COMn) und USB- Schnittstelle). Man kann Peripheriegeräte grob in die folgenden Kategorien einteilen.

- Peripherie zur Kommunikation mit dem Benutzer
  - Grafikkarte
  - Maus, Tastatur
  - Soundkarte
  - Gameport
- Peripherie zur Daten- Ein/ Ausgabe
  - Drucker
  - Scanner
  - Digitalkamera
- Speicherperipherie
  - Festplatten
  - Diskettenantriebe
  - CD-, DVD- Antriebe und -Brenner
  - Wechselplattenantriebe
  - Datenstreamer
- Peripherie zur Kommunikation mit anderen Computern oder Informationstechnologie-Geräten
  - Modem, ISDN- Karten
  - Netzwerkkarte

1.6 Die Befehlsebene

Wir werden nun die Befehlsebene näher betrachten. Wir werden uns also damit beschäftigen, wie ein Prozessor Befehle abarbeitet. Die Befehle, die ein Prozessor abarbeitet sind immer Ergebnis eines Programmiervorgangs. Wir werden in diesem Buch zwar nicht behandeln, wie man einen Computer direkt auf seiner Befehlsebene programmiert. Vielmehr werden wir uns eines Anwendungsprogramms bedienen (der sog. Compiler), um die in späteren Kapiteln des Buches entwickelten C-Programme in Prozessorbefehle umzusetzen und ablaufen zu lassen. Aber auch wenn wir uns mit einer höheren Programmiersprache beschäftigen, ist ein gewisses Verständnis für die Abläufe auf Ebene der Prozessorbefehle

erforderlich. Für tiefere Einblicke sei auf die entsprechende Fachliteratur verwiesen, etwa [Hennessy 1990].

Die Aufgabe des Prozessors besteht darin, ständig Befehle aus dem Speicher zu holen und diese auszuführen. Dieser Vorgang wiederholt sich ohne Unterbrechung bis zum Abschalten des Computers. Was dabei im Einzelnen abläuft werden wir nun näher betrachten.

Ein elementarer Befehl kann etwa darin bestehen, zwei Zahlen aus dem Speicher zu lesen, zu addieren und das Ergebnis in den Speicher zu schreiben. Ein wesentliches Merkmal eines Computers ist, dass die zu bearbeitenden Daten im selben Speicher stehen können, wie die auszuführenden Befehle. Es gibt auch kein äußerliches Unterscheidungsmerkmal für Befehle und Daten. Allein durch den Programmablauf ist festgelegt, welche Speicherinhalte in den Prozessor geholt und als Befehle interpretiert werden.

In Abbildung 1.6-1 ist das Modell eines einfachen Prozessors und seines Arbeitsspeichers dargestellt. An diesem Modell wollen wir die einzelnen Komponenten eines Prozessors und seine Arbeitsweise verdeutlichen.

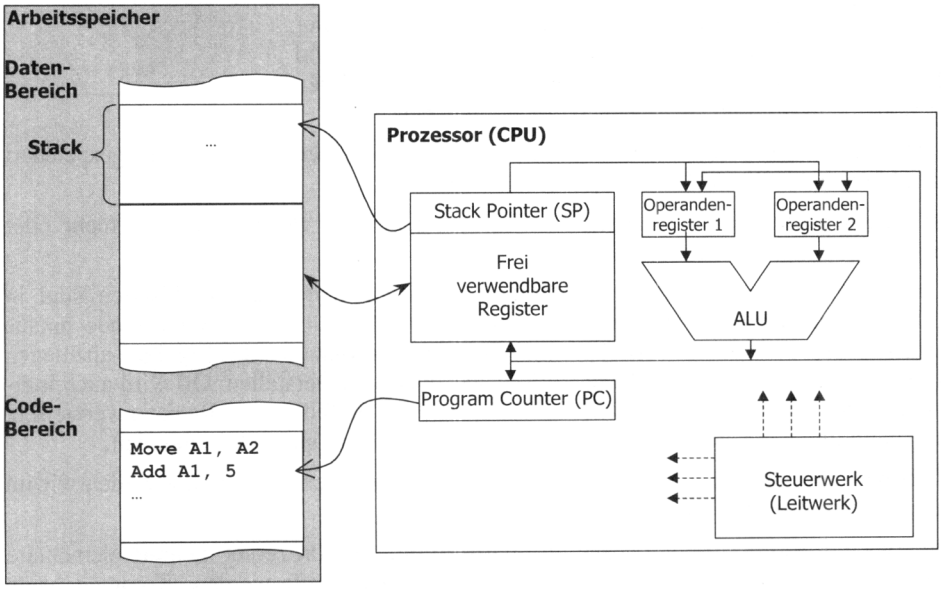


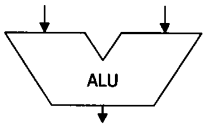
Abbildung 1.6-1: Modell eines einfachen Prozessors

Jeder Prozessor verfügt über mehrere Register, in denen er Operanden oder Ergebnisse zwischenspeichern, oder auch oft benutzte Daten oder Speicheradressen direkt vor Ort halten kann. Register bieten Speicherplatz für n Bits, die zur Weiterverarbeitung zur Verfügung stehen, oder zu anderen Teilen der CPU bzw. über den Bus nach außen transportiert werden können. Der Zugriff auf Register ist deutlich schneller, als der Zugriff auf Informationen im Hauptspeicher, da kein Transfer über den Prozessorbus erforderlich ist. Ein wesentliches Register ist der Befehlszähler (englisch: program counter, abgekürzt PC). Der Befehlszähler enthält immer die Adresse des nächsten zu bearbeitenden Befehls. Er ist in Abbildung 1.6-1 mit einem Zeiger dargestellt, der in den Bereich des Speichers zeigt, in

dem die Befehle stehen, der sog. Code-Bereich. Nach Abarbeitung eines Befehls wird der Befehlszähler auf den nächsten Befehl im Speicher positioniert. Ein weiterer Bereich des Speichers wird als sog. Kellerspeicher (auch Stapelspeicher, englisch: Stack) genutzt. Der Stack dient hauptsächlich dazu, bei Unterprogrammaufrufen die Adresse für den Rücksprung aus dem Unterprogramm zu sichern und Parameter an das Unterprogramm zu übergeben. Wir werden darauf im Kapitel über Unterprogramme detaillierter eingehen. Auf den Stapelspeicher wird stets nur nach ganz bestimmten Vorschriften zugegriffen. Die auf dem Stapelspeicher gespeicherten Daten müssen in umgekehrter Reihenfolge wieder entnommen werden, in der sie dort abgelegt wurden. Das zuletzt abgelegte Datum muss also zuerst wieder entnommen werden. Ein spezielles Prozessorregister – der Stack-Pointer – zeigt auf die Stelle im Stapelspeicher, an die aktuell geschrieben werden darf.

Ein großer Teil des Prozessors dient der Verarbeitung von Daten. Dieser „Datenpfad" genannte Teil ist in der Abbildung auf eine Einheit zur Durchführung arithmetischer und logischer Operationen beschränkt:

Diese Komponente heißt Arithmetisch-Logische Einheit (englisch: Arithmetic and Logical Unit, abgekürzt ALU). Sie ist hier so dargestellt, dass sie bis zu zwei Operanden verknüpfen und ein Ergebnis liefern kann. Sie kann die üblichen arithmetischen Operationen durchführen, aber auch Werte vergleichen.



Der ALU vorgelagert sind zwei Register, in denen die Operanden zwischengespeichert werden können, bis die ALU ihre Operation ausführt.

Meist stehen noch weitere Register zur Verfügung, die zur Programmierung mehr oder weniger frei benutzt werden können.

Wir wollen uns nun der Befehlsausführung zuwenden. Typische Befehlsarten sind in Tabelle 1.6-1 angegeben. Arithmetische Befehle enthalten über den Befehlscode hinaus Angaben darüber, wo die zu verknüpfenden Daten stehen bzw. wo das Ergebnis abzulegen ist. In Frage kommen für diese Angaben Register oder Speicherstellen. Oft wird auch indirekt adressiert, das bedeutet, dass die Angaben bei den Befehlen auf Register verweisen, welche dann ihrerseits die Speicheradressen der eigentlichen Operanden enthalten.

Ein Beispiel über das Zusammenwirken von Befehlen in einem Programm werden wir im Abschnitt über Operatoren und Ausdrücke sehen.

Wenn etwa ein Additionsbefehl ausgeführt wird, so führt der Prozessor mehrere elementare Operationen aus: zuerst müssen die Operanden herangeschafft werden. Kommen beide Operanden aus dem Speicher, so müssen sie nacheinander geholt werden, da über den Prozessorbus zu einem Zeitpunkt lediglich ein Datum transportiert werden kann. Nun stehen die Operanden in den Operandenregistern bereit. Danach wird die eigentliche Addition ausgeführt. Das Ergebnis steht dann in einem speziellen Register[11] und muss von diesem ggf. noch an das eigentliche Ziel transportiert werden.

[11] Dieses Register hieß früher, als Prozessoren noch über ganz wenige Register verfügten, Akkumulator-Register, kurz Akku.

Tabelle 1.6-1: Verschiedene Arten von Befehlen

Befehlsart	Beschreibung
Befehle zum Transfer von Daten	Zum übertragen eines Datums zwischen zwei Stellen im Speicher (ohne es zu verändern), oder zwischen Speicher und Register bzw. zwischen zwei Registern.
Arithmetische Befehle	Diese Befehle führen arithmetische Operationen, wie Addition, Subtraktion oder Multiplikation aus.
Vergleichsbefehle	Vergleichen zwei Werte miteinander. Das Ergebnis wird in einem speziellen, sog. Flag-Register gespeichert, so dass es danach weiter verwendet werden kann, etwa für bedingte Sprungbefehle.
Befehle zur Beeinflussung des Programmablaufs (Sprungbefehle)	Bewirken Fortsetzung des Programms nicht mit dem unmittelbar nächsten Befehl im Speicher, sondern an einer im Befehl angegebenen Speicheradresse. Diese Befehle bewirken damit ein Überschreiben des Programmzählers mit einem neuen Wert. Der Sprung kann dabei unbedingt erfolgen, oder von einer Bedingung abhängen (etwa Ergebnis der vorangegangenen Operation, z. B. einer logischen Vergleichsoperation).
Spezialbefehle	Hierzu zählen Stackoperationen, Ein-/Ausgabebefehle oder Befehle zum Auslösen von Programmunterbrechungen bzw. zum Anhalten des Prozessors.

Zur Ausführung eines Befehls sind also innerhalb des Prozessors viele elementare Operationen zu koordinieren. Dafür sorgt ein Teil des Prozessors, der Steuerwerk heißt. Das Steuerwerk ist über spezielle Leitungen, die Steuerleitungen, mit allen anderen Komponenten der CPU verbunden und steuert diese entsprechend an, wie in Abbildung 1.6-2 angedeutet.

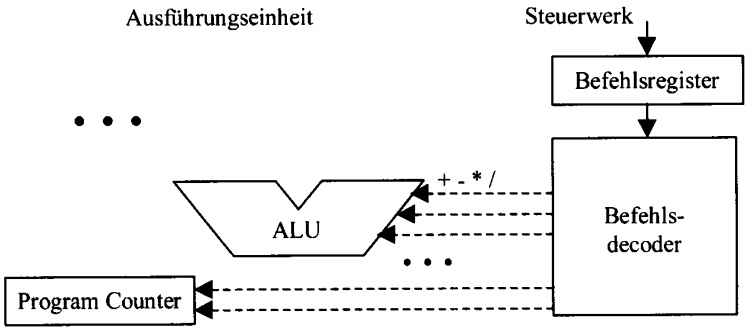


Abbildung 1.6-2: Das Steuerwerk beeinflusst alle Komponenten der Ausführungseinheit über Steuerleitungen.

Das Steuerwerk ist verantwortlich für die Durchführung des Befehlszyklus, jene Schleife in der permanent Befehle aus dem Speicher geholt und ausgeführt werden. Der Befehlszyklus umfasst folgende Phasen:

1. Holphase
   In dieser Phase wird der nächste auszuführende Befehl aus dem Speicher in das Befehlsregister des Steuerwerks gebracht.

2. Dekodierphase:
   Diese Phase dient dem Entschlüsseln und interpretieren des Befehls. Der Teil des Befehls, der über die Art der Operation Auskunft gibt, wird von dem Teil getrennt, der die Quelle der Operanden beschreibt.
3. Ausführungsphase
   Mit Hilfe logischer Schaltungen werden Operanden geholt, Verknüpfungen durchgeführt und die Ergebnisse gespeichert. Auch wird, falls erforderlich, der Befehlszähler beeinflusst. Er wird im Normalfall erhöht, so dass er auf den nächsten Befehl zeigt. Bei einem Sprungbefehl wird er ggf. mit der im Sprungbefehl angegebenen Zieladresse überschrieben.

Diese Sicht auf die Befehlsebene eines Prozessors aus der Perspektive der einzelnen Prozessorregister heißt auch Register-Transfer-Ebene. In Abbildung 1.6-3 ist ein 8086-Prozessor auf Register-Transfer-Ebene gezeigt. Der 8086 ist zwar ein recht alter Prozessor, aber auch in modernen Pentium-Prozessoren sind die gezeigten Elemente immer noch vorhanden. Sie wurden mittlerweile durch ausgeklügelte Komponenten ergänzt, die den Prozessor schneller und effizienter machen. Eine Diskussion aller Einheiten eines Pentium-Prozessors würde den Rahmen dieses einleitenden Kapitels sprengen.

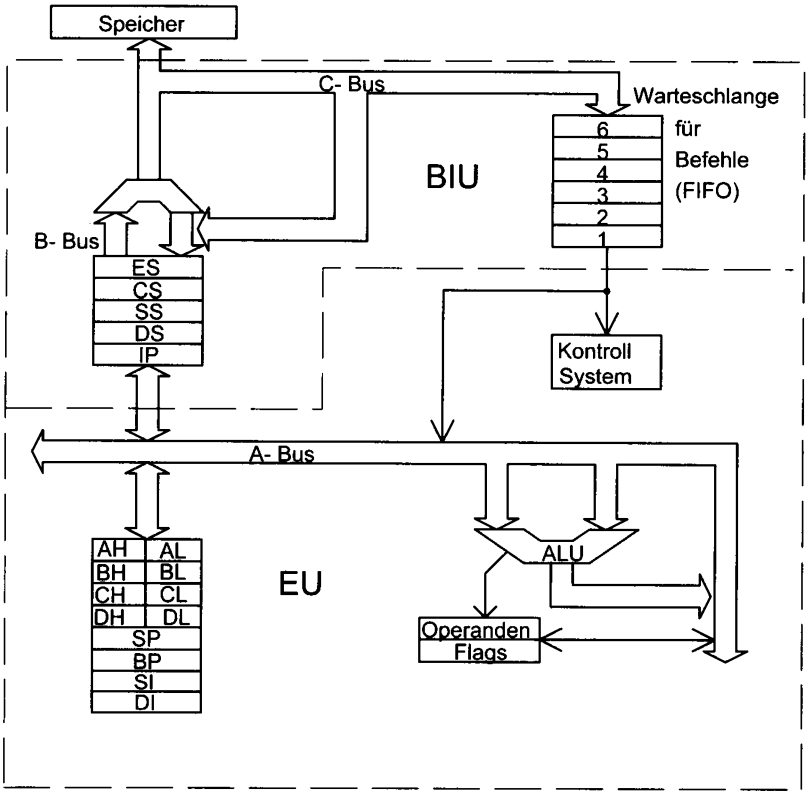


Abbildung 1.6-3: Struktur des 8086-Prozessors (Register-Transfer-Ebene)