Übersicht über den Inhalt des Buches

Im Mittelpunkt der Informatik stehen der Algorithmus, die von ihm benötigten Datenstrukturen und die verwendeten Rechnerarchitekturen. Dabei ist unter Algorithmus ein Schema zu verstehen, nach dem eine gestellte Aufgabe systematisch abgearbeitet werden kann. Es gibt verschiedene Möglichkeiten, einen Algorithmus zu bearbeiten. Mit Hilfe eines Beispiels soll erklärt werden, wie ein Algorithmus für ein Rechnersystem erstellt wird, und welche Informatikwerkzeuge dazu notwendig sind.

Es ist die Aufgabe eines Prüfsystems, verschiedene Meßdaten einzulesen, sie zu dokumentieren und anschließend als eine lineare Funktion im kartesischen Koordinatensystem darzustellen. Hierfür werden die Meßpunkte von einem Meßgerät abgelesen, nach einem vorgegebenen Schema in die Regressionsgleichung eingesetzt und abgearbeitet. Als Ergebnis liegt die Gleichung einer Geraden vor. Diese Gerade wird dann in eine Graphik eingetragen.

Die vorgegebene Aufgabe läßt sich grundsätzlich mit konventionellen Mitteln lösen, sie ist aber bei vielen Meßpunkten sehr zeitaufwendig. Eine wesentliche Verbesserung der Verarbeitungsgeschwindigkeit läßt sich unter Einhaltung der Meß- und Rechenvorschriften durch ein Rechnersystem erzielen. Da ein solches System seine Aufgabe in diskreten Schritten abarbeitet, darf das Schema nur Operationen enthalten, die die Maschine durchführen kann wie z. B. Einlesen von Meßdaten, Multiplizieren, Dividieren, Addieren, Subtrahieren und Plotten der Meßdaten. Der Algorithmus für das Einlesen der Daten, die Durchführung der linearen Regression und die graphische Darstellung muß also in Grundoperationen überführt werden, und diese sind schrittweise auszuführen.

Wird ein Algorithmus für die Abarbeitung im Rechner entworfen, so muß er zunächst einmal rechnergerecht vorbereitet werden. Es ist vorteilhaft, ihn auf Papier niederzuschreiben oder ihn interaktiv mit Hilfe einer graphischen Oberfläche zu entwerfen. Die Eingabe des Algorithmus in den Rechner muß über eine dem Menschen leicht verständliche Programmiersprache geschehen. Diese Sprache versteht aber wiederum der Rechner nicht, sie muß für ihn mit Hilfe eines Übersetzerprogrammes in rechnergerechte Form übersetzt werden. In der Regel haben Rechner verschiedener Hersteller unterschiedliche Befehlsvorräte. Aus diesem Grund muß das Übersetzerprogramm mit der vorliegenden Rechnerarchitektur kompatibel sein.

Das Ablaufschema eines Algorithmus läßt sich besonders übersichtlich durch Graphiken, wie z. B. Struktogramme, darstellen. Diese Graphiken sind somit ein wesentlicher Bestandteil der Programmierumgebung und sollten zur übersichtlichen Darstellung von Algorithmen benutzt werden.

Der Algorithmus wird als Programm auf einem Speichermedium abgelegt, zu seiner Abarbeitung muß er aber im internen Speicher, dem Arbeitsspeicher, vorliegen. Ebenfalls ist für die Meßdaten Speicherplatz vorzusehen. Diese Meßdaten werden in einer eigens dafür geschaffenen Datenbasis gespeichert. Mit ihrer Hilfe ist es möglich, schnell auf die Daten zuzugreifen, geordnete Rechneroperationen durchzuführen, die Daten zu sortieren, neue Daten einzufügen usw.

Das hier verwendete Meßsystem benutzt einen oder mehrere Prozessoren und eine Meß- und Datenperipherie. Für diese Peripherie, bestehend aus dem Meßgerät und der Datenübertragungsstrecke zur graphischen Oberfläche, müssen Programme vorhanden sein, die den Ablauf im Rechnersystem, die Datenkommunikation, die Ein- und Ausgabe von Daten und den Zugriff zum Speicher steuern.

Der relativ einfache Algorithmus zur Lösung der Meßaufgabe erfordert das Vorhandensein einer großen Anzahl verschiedener Informatikwerkzeuge, wie z. B. das Meßgerät, den Rechner, die graphische Oberfläche, die Datenübertragungsstrecken, die Programmiersprache, den Übersetzer, das Betriebssystem, die Datenbank usw. In diesem Buch werden die Grundlagen der wichtigsten Informatikwerkzeuge in den einzelnen Kapiteln erklärt und es wird versucht, in der „Übersicht über den Inhalt des Buches“ den Zusammenhang der Werkzeuge verständlich zu machen.

Kapitel 1 erläutert zunächst die wichtigsten Grundbegriffe der Informatik. Anschließend wird die Bedeutung der Datenverarbeitung in der Technik besprochen, und es wird gezeigt, daß die Informatik in den letzten drei Jahrzehnten ein integraler Bestandteil einer soliden Ausbildung für Ingenieure und Naturwissenschaftler wurde. Wie jede Wissenschaft ist auch die Informatik aus einfachen Anfängen entstanden. Zunächst mußten die Zahlensysteme und Rechenoperationen erfunden werden und danach war es möglich, Rechenmaschinen zu konzipieren, mit denen aufwendige Berechnungen wesentlich vereinfacht werden konnten. Die ersten Rechenhilfen bestanden zunächst aus einfachen und später aus sehr komplizierten mechanischen Rechenwerken, die ganz oder teilweise mit der Hand gesteuert wurden. Einfache mechanische Speicher entstanden vor etwa 200 Jahren. Charles Babbage konzipierte um 1833 den ersten Rechner, der aus einem Eingabewerk, Rechenwerk, Steuerwerk, Speicher und Ausgabewerk bestand. Eine Rechnerarchitektur nach diesem Prinzip ließ sich aber erst mit der Erfindung des elektronischen Rechners und des magnetischen Speichers praktisch verwirklichen. Zu den Pionieren der heutigen Rechenmaschine gehören H. H. Aiken, I. P. Eckert, J. W. Mauchly und K. Zuse. Mit der Entwicklung des Transistors und seiner Einbettung in hochintegrierte Schaltkreise war es möglich, die modernen Hochleistungsrechner zu entwerfen und zu fertigen. Die Entwicklung des Rechners und seiner Softwaresysteme wird in verschiedene Generationen eingeteilt. Es wird ein Versuch gemacht, diese Evolution kurz zu erklären und die Art verwendeter Rechner zu beschreiben. Am Ende dieses Kapitels wird dann noch die Bedeutung der Software besprochen.

Kapitel 2 vermittelt die zum Verständnis des Buches notwendigen Theoriekenntnisse, wobei die eingeführten Begriffe durch praxisbezogene Beispiele ergänzt werden. Ausgangspunkte unserer Überlegungen sind die drei Schlüsselbegriffe der Informatik: Information, Nachricht und Daten. Aus diesen Grundbegriffen werden zuerst Verfahren zur Kodierung und rechnerinternen Darstellungen von Dualzahlen abgeleitet. Das erste Unterkapitel wird abgeschlossen mit der Shannonschen Informationstheorie und der Hakenschen Semantikinterpretation von Information durch ihren Effekt.

Algorithmen sind Lösungsverfahren, die eine Reihe vorgegebener Bedingungen, wie Exaktheit und Terminierung, erfüllen. Funktionen sind berechenbar, wenn sie als Algorithmen dargestellt werden können. Neben der Berechenbarkeit ist die Komple-

xität von Algorithmen in der Informatik von besonderer Bedeutung, denn sie definiert den Aufwand, der notwendig ist, um einen Algorithmus auf einem Rechner durchzuführen.

Durch die Aussagen- und Prädikatenlogik wird nicht nur die formale Grundlage der Künstlichen Intelligenz aufgebaut, sondern auch die Boolesche Algebra begründet, die zur Bildung von Normalformen geführt hat. Diese Formen bilden die Basis für die Entwicklung von Schaltnetzen und Schaltkreisen.

Die Automatentheorie liefert eine formale Beschreibung digitaler Schaltungen, bei welchen die Ausgangsinformation und der innere Zustand im nachfolgenden Zeitpunkt als Funktion der Eingangsinformation und der inneren Zustände ermittelt werden können. Für die Informatik ist besonders die Theorie der endlichen Automaten von Interesse. Ein Beispiel für die Anwendung endlicher Automaten ist der Compilerbau, oder allgemeiner: die Analyse formaler Sprachen. Bei dieser Analyse wird der Automat durch die Grammatik der formalen Sprache (Programmiersprache) vorgegeben, und die Eingabedaten veranlassen den Übersetzer, den Zustand in Abhängigkeit der Eingabe zu ändern, d. h. den Syntaxbaum zu durchlaufen. Die Funktionsweise eines Compilers wird in Kapitel 5 erläutert. Für die Beschreibung von Schaltwerken werden vorzugsweise die Automatenmodelle von Mealy und Moore verwendet.

Grammatiken bilden ferner die angemessene Darstellungsform, um nicht nur die verschiedenen Sprachklassen formaler Sprachen zu definieren (Chomsky-Hierarchie), sondern sie legen auch fest, welche Sprache durch welchen Automaten erkannt werden kann.

Graphen und Bäume sind sehr gut dafür geeignet, strukturierte Objekte zu beschreiben und darauf spezielle Suchverfahren wie Graphensuche zu implementieren. Sie bilden daher auch die Grundlage für die Definition von Datenstrukturen (Kapitel 3). Bäume werden auch zur effizienten Speicherung von Dateien verwendet. Derartige Datenstrukturen können sowohl als gerichtete als auch ungerichtete Graphen implementiert werden, je nachdem, welche Zugriffskriterien von Bedeutung sind. Kapitel 4 stellt hierfür geeignete Sortieralgorithmen und Suchalgorithmen vor.

Zur Klasse der gerichteten, bipartiten, d. h. mit zwei verschiedenen Knotentypen versehenen Graphen, gehören die Petri-Netze. Sie bieten Modellierungsmöglichkeiten zur Darstellung nebenläufiger Prozesse sowie auch von dynamischen Komponenten, wie sie z. B. bei der Steuerung von Produktionsanlagen oder Verkehrssystemen auftreten. Trotz dieses großen Anwendungspotentials hat sich die von Petri 1962 gegründete Theorie jedoch noch nicht in breitem Umfang durchgesetzt.

Kapitel 3 führt den Leser in den Begriff der Datenstrukturen und ihren internen Darstellungen in Rechnern ein. Neben Algorithmen gehören Datenstrukturen zu den Grundbegriffen der Informatik. Mit ihrer Hilfe werden Daten nicht in beliebiger Ansammlung gespeichert und verarbeitet, sondern sie werden so strukturiert, daß sie von den sie bearbeitenden Algorithmen möglichst zeit- und speichersparend verwendet werden.

Naturgemäß besteht ein enger Zusammenhang zwischen dem verwendeten Algorithmus und der dafür am geeignetsten Datenstruktur. Diese Kopplung wird durch

Datentypen herbeigeführt, die Datenstrukturen und darauf erlaubte Operationen zu einer Einheit zusammenfügen. Aus der Sicht des objektorientierten Programmierens ist ein Datentyp allerdings kein Objekt, denn als solches muß es mit anderen Objekten Nachrichten austauschen und sowohl Variable als auch Operationen vererben können.

In diesem Kapitel konzentrieren wir uns auf die elementaren Datenstrukturen, die immer wieder auftauchen. Dies sind: Felder, Verbunde (Records) und Listen. Nichtlineare Listen werden dazu verwendet, Bäume und Graphen zu implementieren, wobei sie üblicherweise rekursiv definiert werden. Suchbäume sind speziell dafür entwickelt worden, Dateien nach bestimmten Schlüsseln zu durchsuchen. Sind die Dateien auf externen Speichern abgelegt, dann werden für die Schlüsselsuche spezielle Suchbäume, die als B-Bäume bezeichnet werden, eingesetzt.

Kapitel 4 behandelt Algorithmen und Prozesse. Ein Algorithmus ist der Grundbaustein der Datenverarbeitung, denn ohne ihn läßt sich kein Programm erstellen, und ohne dieses arbeitet ein Rechner nicht. Prinzipiell kann zwar ein Algorithmus zur Lösung einer Problemklasse unabhängig von der Programmiersprache, in die er umgesetzt wird, und dem verwendeten Rechner, auf dem das Programm später laufen soll, definiert werden, doch kommt es in der Praxis zu wesentlichen Laufzeitunterschieden, je nachdem, welche Programmiersprache und welches Rechnersystem eingesetzt wird.

Prozesse sind aktive Funktionsträger, die den zeitlichen Ablauf von Programmen steuern und dazu benutzt werden, um verschiedene Programmabläufe zu synchronisieren. Dabei kommt es wesentlich darauf an, ob die Prozesse nur sequentiell oder parallel abgearbeitet werden können. Dies wird durch die verwendete Programmiersprache und vorhandene Rechnerarchitektur bestimmt. Als sequentielle Standard-Programmiersprache wird in diesem Kapitel OBERON, die Nachfolgesprache von PASCAL, verwendet. Die Parallelisierung sequentieller Algorithmen wird in Anlehnung an die FORTRAN-90-Notation vorgestellt. Rechnerarchitekturen werden in Kapitel 7 beschrieben.

In der Datenverarbeitung gibt es eine Menge von Grundalgorithmen, die immer wieder benötigt werden und daher bereits implementiert wurden, damit sie nicht jedesmal neu erdacht werden müssen. So gibt es Standardalgorithmen zum Sortieren von Zahlen, zur Durchführung arithmetischer Operationen, zum gezielten Aufsuchen von Elementen in Dateien etc. Einige dieser Standardalgorithmen werden in diesem Kapitel besprochen. Ferner werden als eine neuere und erfolgversprechende Form nichtdeterministischer Optimierungsverfahren genetische Algorithmen vorgestellt.

Kapitel 5 beschäftigt sich mit dem Thema Programmiersprachen und deren Umsetzung in rechnerinterne Befehle. Für den entworfenen Algorithmus zur Beschreibung eines Prozesses erstellt der Programmierer ein ablauffähiges Programm. Da die Kenntnis einer höheren Programmiersprache vorausgesetzt wird, richtet sich das Augenmerk des Kapitels zunächst auf die Assemblersprache. Es wird die Struktur des Assemblerbefehls analysiert und dann der Ablauf eines Programmes als Maschinencode im Rechner erklärt. Die zur effizienten Programmierung verwendeten Adressierungsarten werden im Detail besprochen und Begriffe wie Makro und Unterprogram-

me werden vorgestellt. Zur Eigenart eines Rechners gehört auch sein Vorrat an Assemblerbefehlen. So gibt es bei einfachen Rechnern vielleicht nur 80 und bei mächtigen Rechnern bis 500 Befehle. Von diesen werden die wichtigsten Befehlsarten und deren Wirkung erläutert. Das nächste Ereignis im Entstehen eines Programmes ist dessen Übersetzung in Maschinencode. Dies wird bei einer höheren Sprache unter Zuhilfenahme eines Compilers und bei einer niedrigen Sprache mit einem Assembler erreicht. Mit dem Übertragen des fertigen Codes in den Rechner durch den Lader ist dann das Programm ablauffähig. Nach der Diskussion der Grundzüge der Assemblersprachen werden höhere Programmiersprachen besprochen, und zwar werden deren wesentliche Struktur und die wichtigsten Fähigkeiten mit Hilfe von C erläutert. Eine konsequente Weiterentwicklung der höheren Sprachen führt zur Konzeption logischer, funktionaler und objektorientierter Programmiersprachen. Dieses Kapitel wird mit einer Funktionsbeschreibung der Systemprogramme Compiler, Assembler und Lader abgeschlossen.

Kapitel 6 gibt eine Einführung in moderne Entwicklungshilfen für Software. Die Erfahrung hat gezeigt, daß die Erstellung von Software die aufwendigste und zeitraubendste Aufgabe einer Rechnerinstallation ist. Aus diesem Grund sollte der Entwickler von Software mit modernen Softwareengineering-Methoden vertraut sein. Der Lebensweg der Software wird typisch in die Phasen Istanalyse, Anforderungsbeschreibung, Grobentwurf, Feinentwurf, Kodierung, Test, Integration und Wartung eingeteilt. Es ist wichtig, daß Softwarefehler möglichst schon in den ersten Entwicklungsphasen gefunden werden. Je weiter sich ein Fehler in anschließenden Phasen fortsetzt, desto teurer ist es, ihn zu beseitigen. Für die Erstellung von möglichst fehlerfreier Software wurde eine große Anzahl von Entwicklungshilfen entworfen. Qualitative Eigenschaften der Software beziehen sich auf deren Funktionalität, Konstruktion, Benutzbarkeit, Zuverlässigkeit und Leistungsfähigkeit. Ein guter Softwareentwickler wird versuchen, diese Qualitäten in seine Software einzubauen. Jedoch werden in der Praxis verschiedene dieser Anforderungen sich gegenseitig widersprechen, oder sie würden bei strenger Einhaltung zu ineffizienter Software führen. Es muß daher ein Kompromiß bei der Auswahl von Anforderungen für die Softwarequalität gefunden werden. Bei der Diskussion in diesem Kapitel werden als besonders wichtige Eigenschaften guter Software ihre Modularität und Strukturiertheit hervorgehoben, anschließend wird dann ein Überblick über die gebräuchlichsten Entwicklungshilfen für Software gegeben.

Kapitel 7 behandelt die Architektur von einzelnen Rechnern, von Rechnernetzen und den Aufbau digitaler Speicherhierarchien. Die Rechnerarchitektur, welche von Neumann in den 40er Jahren konzipiert hat, bildet auch heute noch die Grundlage der meisten Rechner. Die anfänglichen Ausführungen dieses Kapitels befassen sich mit der Struktur und dem Von-Neumann-Operationsprinzip eines Rechners. Das Kapitel endet mit der Beschreibung von Rechnernetzen. Diese Netze bilden die Basis für verteilte Rechnersysteme, mit deren Hilfe heute eine Fülle neuer Telekommunikationsdienste wie Telekauf oder Telebanking über Internet angeboten werden. Diese Dienste werden in Kapitel 8 angesprochen.

Dazwischen wird ein Bogen gespannt, der nicht nur die Grundstrukturen heutiger Prozessoren beleuchtet, sondern auch ausführlich ihr erweitertes Funktionsspektrum

schildert. Zu diesem erweiterten Funktionsumfang zählen vor allem die Superskalarität und die Multiprozessorfähigkeit, wobei die zuletzt genannte Funktion auch die Datenkonsistenz von Daten, die parallel in verschiedenen Speichermedien verarbeitet werden, garantiert. Danach wird das Prinzip der Mikroprogrammierung vorgestellt, und mit diesem Hintergrund ausgestattet, werden die prägnanten Eigenschaften bzw. Unterschiede von CISC- und RISC-Prozessoren vorgestellt.

Rechner müssen mit Speicher ausgestattet sein, um Programme und die dazugehörenden Daten aufnehmen zu können. Die wichtigsten allgemeinen Kenngrößen von Speichern sind ihre Zugriffszeiten und Kapazitäten. Beide Speichermerkmale stehen sich aber diametral gegenüber, weshalb der Speicher hierarchisch angeordnet wird. Vom Prozessor aus betrachtet wächst in dieser Hierarchie die Zugriffszeit und die Kapazität des Speichers an.

Auf der untersten Stufe dieser Speicherhierarchie (Primärebene) sind der Cache-Speicher und der Arbeitsspeicher angesiedelt. Danach folgen Plattenspeicher (Sekundärebene) und Bandspeicher (Tertiärebene), wobei Cache-Speicher Pufferspeicher sind, die sich zwischen Prozessor und Arbeitsspeicher befinden und eingeführt wurden, um den deutlichen Unterschied in der Zykluszeit zwischen Prozessor und Arbeitsspeicher auszugleichen. Wegen ihrer zunehmenden Bedeutung für moderne Prozessoren wird der Cache-Speicher sowohl von seinem Aufbau als auch den verwendeten Auswahl- und Verdrängungsstrategien her ausführlicher beschrieben. Dabei werden als prägnante Organisationsformen von Cache-Speichern vor allem voll- und teilassoziative Caches vorgestellt. Es folgen technische Beschreibungen digitaler Speicher. Dazu zählen Halbleiterspeicher, Magnetblasenspeicher, optische Speicher und magneto-optische Speicher.

Moderne Prozessoren bieten bereits auf der Hardwareebene eine Reihe von Unterstützungen für Betriebssystemfunktionen an, damit diese möglichst schnell und zuverlässig ausgeführt werden können. Zu diesen Aufgabenbereichen, welche diese Unterstützung erfahren, zählen vor allem die Prozeß- und Speicherverwaltung. Sie werden hardwaremäßig primär unterstützt durch das Unterbrechungssystem, die Speicherverwaltungseinheit und durch die Integration von Cache-Speicher auf dem Prozessorchip. Betriebssystembezogene Details dieser beiden Verwaltungen werden im nächsten Kapitel vorgestellt.

Kapitel 8 ist den Betriebssystemen und den verteilten Rechnersystemen gewidmet. Betriebssysteme gehören zum übergeordneten Begriff der Systemsoftware, wozu auch die in Kapitel 5 beschriebenen Module wie Compiler, Lader und Binder gehören. Ein Betriebssystem entlastet den Anwender davon, sich selbst um die Steuerung und Verwaltung eines Rechners kümmern zu müssen, denn dazu müßte er spezielle Systemprogramme, wie z. B. zur Prozeßverwaltung, schreiben, damit diese sowohl die Anwenderprogramme zum geordneten Ablauf bringen als auch die einzelnen Funktionen der Hardware, z. B. durch Gerätetreiber, ansteuern. Somit nimmt ein Betriebssystem dem Nutzer eine Fülle von Arbeiten ab, wie das Planen von Programmabläufen, die Optimierung der Prozessorauslastung und das Erkennen und Beheben von Fehlern. Als Folge hiervon braucht der Anwender keine speziellen Kenntnisse mehr von geräte- und schnittstellentechnischen Details der Rechner sowie der realen als auch

logischen Datenübertragung zwischen dem Rechner und der Peripherie zu besitzen. Das Betriebssystem entlastet ihn von Detailfragen der Maschinensteuerung und Maschinenverwaltung und verringert somit die Komplexität der zu lösenden Aufgabe ganz deutlich, bzw. ermöglicht sie erst.

Zu den klassischen Aufgabenbereichen von Betriebssystemen, die in diesem Buch beschrieben werden, gehören die Verwaltung und der Betrieb von Prozessen und ihren Interaktionen, von Speichern und von Dateien. Die Beschreibung dieser Themenbereiche folgt allerdings nicht dem üblichen Schema, sondern geht von einem globalen Systemansatz aus und überträgt allgemein gültige Verwaltungsprinzipien auf Betriebssysteme. Besonders zu nennen ist hierbei die Einteilung der Betriebsmittel in reale, virtuelle und logische Betriebsmittel und die Ableitung eines Operationsprinzips für Betriebssysteme, das vor allem aus den Mustern abgeleitet wird, die zwischen Prozessen bei ihren Interaktionen ablaufen. Dabei liefern Prozesse das Schlüsselkonzept für die Organisation eines Betriebssystems.

Eine weitere wichtige Charakterisierung von Betriebssystemen liegt darin, ob sie zentral oder verteilt sind. Hierbei wird ein System verteilt genannt, wenn seine Steuerung dezentral ist und seine Komponenten sich an räumlich verschiedenen Stellen befinden und so miteinander verbunden werden, daß unabhängig von der Verteilung die Funktionalität des Gesamtsystems garantiert wird. Alle oben angesprochenen Themenfelder wie Prozeß- und Dateiverwaltung werden jeweils aus einer zentralen und verteilten Sicht ausführlich beschrieben.

Kapitel 9 behandelt als eine besondere Rechnerart den Echtzeitrechner. Dieser ist für die Leitung und Steuerung von technischen Prozessen von Bedeutung. Technische Prozesse können sehr vielseitig sein, so versteht man darunter Prozesse zur Stromerzeugung, zur Fertigung von Rohmaterialien und Maschinen, Meßsysteme zur Überwachung von Patienten, Regelsysteme für Flugzeuge usw. Rechner für die Steuerung solcher Prozesse haben dieselben Grundkomponenten wie kommerzielle und wissenschaftliche Systeme, und die in den Kapiteln 1–8 diskutierten Grundlagen können auch für sie verwendet werden. Um einen Prozeß leiten zu können, braucht der Rechner jedoch noch zusätzliche Einrichtungen, wie z. B. einen Datenkanal zur Prozeßperipherie, eine Echtzeituhr und ein Unterbrechungswerk. Ein Echtzeitrechner muß über seine Peripherie mit dem Prozeß und auch dem Bediener Daten austauschen können. Für die Prozeßkommunikation gibt es Digital- und Analog-Ein-/Ausgabemoduln, sowie Zählwerke und Grenzwertanzeiger. Die Kommunikation mit dem Bediener geschieht über konventionelle Peripheriegeräte. Eine besondere Stellung unter den Echtzeitsystemen nimmt der Mikrorechner ein. Er kann im Gegensatz zu kommerziellen Prozeßrechnern nach dem Baukastenprinzip modular an die Anwendung angepaßt werden. Will man einen größeren oder komplexeren Prozeß leiten, so wird die Steuer- oder Regelaufgabe hierarchisch aufgeteilt und modular auf Einzelrechnern implementiert. Diese Einzelrechner werden dann hierarchisch über ein Bussystem in verschiedenen Ebenen zusammengeschlossen.