Informations- und Kommunikationstechnik

für Betriebswirte
und Wirtschaftsinformatiker

Von

Dipl.-Ing. Dr. rer. pol. Lutz J. Heinrich
o. Universitätsprofessor für Betriebswirtschaftslehre
und Wirtschaftsinformatik an der Universität Linz

Dipl.-Ing. Dr. techn. Franz Lehner
o. Universitätsprofessor für Wirtschaftsinformatik
an der Wissenschaftlichen Hochschule für Unternehmensführung Koblenz

und

Mag. Dr. rer. soc. oec. Friedrich Roithmayr
o. Universitätsprofessor für Wirtschaftsinformatik
an der Universität Innsbruck

4., verbesserte Auflage
mit 176 Abbildungen

R. Oldenbourg Verlag München Wien

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Heinrich, Lutz J.:
Informations- und Kommunikationstechnik für Betriebswirte und Wirtschaftsinformatiker / von Lutz J. Heinrich, Franz Lehner und Friedrich Roithmayr. – 4., verb. Aufl. – München ; Wien : Oldenbourg, 1994
ISBN 3-486-22830-7

NE: Lehner, Franz:; Roithmayr, Friedrich:

Gesamtherstellung: Grafik + Druck, München

ISBN 3-486-22830-7

Inhaltsverzeichnis

Vorwort

Vorwort zur vierten Auflage

Die 3. Auflage dieses Lehrbuchs, die 1993 erschien und die gegenüber der 2. Auflage vollständig überarbeitet und erweitert war, fand ein so reges Interesse, daß die 4. Auflage schon ein Jahr später erscheinen kann. Angesichts dieses kurzen Zeitraums bestand keine Notwendigkeit zu grundlegenden inhaltlichen Änderungen, sodaß sich die Autoren darauf beschränken konnten, formale Fehler und sprachliche Ungenauigkeiten zu beseitigen. An einigen Stellen wurden die Literaturhinweise aktualisiert. Besitzer der 3. Auflage können diese daher im Lehrbetrieb praktisch gleichwertig neben der 4. Auflage verwenden.

Dieses Lehrbuch ist im Zusammenhang mit den anderen Veröffentlichungen der Reihe "Wirtschaftsinformatik" im Oldenbourg Verlag zu sehen, und zwar "Wirtschaftsinformatik - Einführung und Grundlegung", "Systemplanung" (zwei Bände) und "Informationsmanagement". Es enthält den Stoff, dessen Kenntnis Voraussetzung für das Verständnis von "Systemplanung" und "Informationsmanagement" ist. Die Reihe wird durch das "Wirtschaftsinformatik-Lexikon" und das "Wirtschaftsinformatik-Wörterbuch" sowie das "Übungsbuch Wirtschaftsinformatik" ergänzt.

L. J. Heinrich
F. Lehner
F. Roithmayr

Aus dem Vorwort zur dritten Auflage

Sowohl die im Herbst 1988 erschienene 1. Auflage als auch die im Frühjahr 1990 erschienene 2. Auflage dieses Lehrbuchs haben bei den Studierenden an Universitäten und Fachhochschulen eine gute Aufnahme gefunden. Vier Jahre nach Herausgabe der 1. Auflage kann daher bereits die 3. Auflage vorgelegt werden. Während sich die 2. Auflage auf die Beseitigung einiger Fehler konzentrierte, so daß die 1. Auflage für die Studierenden verwendbar blieb, erfolgte mit der 3. Auflage eine gründliche Überarbeitung des gesamten Bestands, wenn auch in unterschiedlichem Umfang. So hat sich z.B. im Kapitel Schutztechnik nur wenig geändert, während das Kapitel Programmiersystem stark verändert wurde. Insgesamt gesehen haben sich die Autoren bemüht, neuere Entwicklungen der Informations- und Kommunikationstechnik - soweit sie deren Kenntnis für Betriebswirte und Wirtschaftsinformatiker für erforderlich halten - zu berücksichtigen. Aus diesen Gründen können die bisherigen Auflagen ohne Informationsverlust nicht mehr im Lehrbetrieb verwendet werden.

Die inhaltliche Kontinuität insgesamt wurde bewußt bewahrt, indem der Stoffumfang und die Gliederung des Stoffes in Kapitel und in Lerneinheiten beibehalten wurden. Die Autoren sind weiterhin der Auffassung, daß "ihre Gliederung", die sich von der üblicherweise verwendeten Gliederung wesentlich unterscheidet, deutliche Vorteile hat - und daß sie sich bewährt hat. Sie wird in dieser Auflage

noch besser verdeutlicht, indem sie in der Einführung visualisiert und mit den entsprechenden Hervorhebungen in allen nachfolgenden Kapiteln immer wieder verwendet wird. Damit wird auch der Zusammenhang zwischen den Teiltechnologien besser als bisher verdeutlicht.

Aus dem Vorwort zur ersten Auflage

Der Titel "Informations- und Kommunikationstechnik für Betriebswirte und Wirtschaftsinformatiker" macht erstens deutlich, daß der Gegenstand dieses Lehrbuchs primär die **Technik** ist und nicht die Verfahren, Prinzipien, Methoden usw. der Anwendung der Technik (also nicht Technologie); zweitens, daß nicht nur die **Daten**verarbeitungstechnik Gegenstand ist, sondern die **Informations- und Kommunikationstechnik** (also z.B. auch die Technik zur Sprachverarbeitung) und drittens, daß der Adressatenkreis des Lehrbuchs Betriebswirte und Wirtschaftsinformatiker sind. Es enthält also den technisch orientierten Grundlagenstoff der Wirtschaftsinformatik.

Trotz einer großen Anzahl an Lehr- und Fachbüchern, die im Titel den Begriff "Datenverarbeitung" oder "Wirtschaftsinformatik" verwenden, haben die Autoren bis zum Zeitpunkt der Abfassung dieses Vorworts keine Quelle mit diesem technisch orientierten Grundlagenstoff der Wirtschaftsinformatik gefunden, die sie uneingeschränkt als Lehrtext für Betriebswirte und Wirtschaftsinformatiker empfehlen könnten. Die verfügbaren Lehrbücher versuchen entweder, alle oder zumindest mehrere Teilgebiete der Wirtschaftsinformatik abzudecken; sie sind dann einerseits sehr umfangreich, andererseits bringen sie den technisch orientierten Grundlagenstoff nicht umfassend genug. Oder sie beschränken sich auf einen Teil der Informations- und Kommunikationstechnik (meist auf die Datenverarbeitung), den sie außerdem nicht so abhandeln, wie dies nach Auffassung der Autoren für Betriebswirte und Wirtschaftsinformatiker erforderlich ist.

Die Bezeichnung dieser Veröffentlichung als Lehrbuch soll ihren primären Zweck, als Unterlage für einschlägige Lehrveranstaltungen und zum Selbststudium für Studierende der Betriebswirtschaftslehre und der Wirtschaftsinformatik zu dienen, verdeutlichen. Darüber hinaus wird diese Veröffentlichung auch für viele Praktiker, die sich in das Gesamtgebiet der Informations- und Kommunikationstechnik einarbeiten wollen, von Nutzen sein.

Alphabetisches Verzeichnis der Lerneinheiten

EINFÜ - Einführung

Lernziele

Sie kennen den Charakter des vorliegenden Lehrbuchs. Sie wissen, wie dieses Lehrbuch gegliedert ist. Sie können die Sichtweise der Betriebswirtschaftslehre und der Wirtschaftsinformatik auf die Informations- und Kommunikationstechnik erläutern. Sie können die Wirtschaftsinformatik in Teilgebiete und Teildisziplinen gliedern und erkennen aus dieser Gliederung, welche Bedeutung die Informations- und Kommunikationstechnik für Studierende der Betriebswirtschaftslehre und der Wirtschaftsinformatik hat.

Definitionen und Abkürzungen

Aufgabe (task) = die aus dem Leistungsprogramm einer Organisation abgeleitete Teilleistung einer ihrer Struktureinheiten bzw. der in diesen tätigen Aufgabenträger.

Betriebswirtschaftslehre (Business Administration) = eine Realwissenschaft, deren Gegenstandsbereich das wirtschaftliche Handeln in Betriebswirtschaften ist.

Bild (picture) = eine geordnete Menge von Bildpunkten auf einer Fläche.

Daten (data) = die Zeichen oder kontinuierlichen Funktionen, die aufgrund von bekannten oder unterstellten Abmachungen und vorrangig zum Zweck der Verarbeitung Information darstellen.

Information (information) = die handlungsbestimmende Kenntnis über historische, gegenwärtige und zukünftige Zustände der und Vorgänge in der Wirklichkeit.

Informationsinfrastruktur (information infrastructure) = die Einrichtungen, Mittel und Maßnahmen zur "Produktion" von Information und zur Kommunikation in einer Organisation.

Informations- und Kommunikationssystem (information and communications system) = ein Mensch-Aufgabe-Technik-System zur "Produktion" von Information und zur Kommunikation; Teil der Informationsinfrastruktur.

Informationsfunktion (information function) = die Gesamtheit der Aufgaben einer Organisation, die sich mit Information und Kommunikation als wirtschaftliches Gut befassen.

Informationsmanagement (information management) = das Leitungshandeln in einer Organisation in bezug auf den Produktionsfaktor Information und in bezug auf Kommunikation.

Kommunikation (communications) = der Austausch von Information mit dem Zweck, das Handeln in bezug auf definierte Ziele optimal zu gestalten.

Sprache (speech) = ein System von Zeichen, das dem Menschen zum Ausdrücken von Gedanken, Gefühlen, Willensregungen usw. dient.

Systemplanung (systems planning) = das vorausschauende, systematische Durchdenken und Formulieren von Zielen, Verhaltensweisen und Handlungsalternativen, die Auswahl optimaler Alternativen sowie die Festlegung von Anweisungen zur Realisierung optimaler Alternativen in bezug auf Informations- und Kommunikationssysteme als Planungsobjekt.

Text (text) = eine logisch zusammengehörige Folge von Wörtern und Sätzen.

Wirtschaftsinformatik (economic informatics) = die sozial- und wirtschaftswissenschaftliche Disziplin, deren Gegenstandsbereich die Informationsfunktion von Organisationen ist.

Charakter des Lehrtextes

Das vorliegende Buch ist ein Lehrbuch. Die Autoren betonen dies deshalb, um den Eindruck zu vermeiden, daß in diesem Buch neue Erkenntnisse der Wirtschaftsinformatik vermittelt werden. Alles, was in diesem Buch beschrieben wird, ist bereits in vielen Lehrbüchern, Fachbüchern und Forschungsberichten dokumentiert worden; es ist "Stand der Technik". Die Leistung der Autoren besteht im wesentlichen darin, aus der bestehenden Informationsflut für die Studierenden der Betriebswirtschaftslehre und der Wirtschaftsinformatik das herausgezogen und als Lehrtext dargestellt zu haben, was sie als das unbedingt notwendige **Grundlagenwissen** über die Informations- und Kommunikationstechnik ansehen und was sie als **Eingangsvoraussetzung** für ihre Lehrveranstaltungen im zweiten Studienabschnitt bzw. im Hauptstudium verlangen. Dieses Buch dokumentiert also das Mindestwissen eines Studierenden der Betriebswirtschaftslehre oder der Wirtschaftsinformatik über die Informations- und Kommunikationstechnik; es vermittelt Grundlagenwissen für die Wirtschaftsinformatik, nicht Wirtschaftsinformatik-Wissen.

Gliederung des Lehrtextes

Das Buch ist in Lerneinheiten gegliedert, die zu **Kapiteln** zusammengefaßt sind, und zwar:

- Grundlagen der Informations- und Kommunikationstechnik,
- Eingabe- und Ausgabetechnik,
- Speichertechnik,
- Verarbeitungstechnik,
- Programmiersystem,

- Netz- und Transporttechnik,
- Transportdienste,
- Schutztechnik.

Jede Lerneinheit ist in folgende **Abschnitte** gegliedert:

- Lernziele,
- Definitionen und Abkürzungen,
- Stoffinhalt der Lerneinheit,
- Demonstrationsbeispiel,
- Kontrollfragen,
- Quellenliteratur,
- Vertiefungsliteratur.

Die **Lernziele** geben an, was der Leser nach Durcharbeitung der Lerneinheit kennen bzw. können sollte. Die **Definitionen und Abkürzungen** erläutern alle Begriffe, die in der Lerneinheit verwendet werden und deren Kenntnis für das Verständnis des Lernstoffs unbedingt erforderlich ist. Ein Vergleich der Definitionen in verschiedenen Lerneinheiten kann Unterschiede der Definitionen zu der gleichen Bezeichnung zeigen. Die Unterschiede beruhen auf dem unterschiedlichen Kontext, in dem ein Begriff in den Lerneinheiten verwendet wird ("Stelle" kann im Kontext A etwas anderes bezeichnen als im Kontext B). Der Stoffinhalt jeder Lerneinheit ist in mehrere kurze **Teilabschnitte** gegliedert; meist beginnt die Darstellung mit einem **Überblick**. Das **Demonstrationsbeispiel** erläutert den Stoffinhalt beispielhaft; die **Kontrollfragen** dienen zur Überprüfung der Erreichung der Lernziele.

Bei den Literaturangaben wird zwischen **Quellenliteratur** und **Vertiefungsliteratur** unterschieden. Quellenliteratur ist die Literatur, die für die Ausarbeitung der Lerneinheiten herangezogen wurde. Dabei bestand die Aufgabe der Autoren darin, den Stoff aus umfangreichen, in der Regel mehrere hundert Seiten umfassenden Quellen so zu selektieren und zu komprimieren, daß eine Darstellung auf durchschnittlich acht Seiten je Lerneinheit möglich wurde. Dem Leser wird empfohlen, die Quellenliteratur heranzuziehen, wenn die Darstellung als zu selektiert oder zu komprimiert empfunden wird, wenn also Verständnisschwierigkeiten entstehen sollten. Mit der Vertiefungsliteratur werden Quellen angegeben, deren Stoff deutlich über den der Quellenliteratur hinausgeht. Die Vertiefungsliteratur soll also einem über den Stoff dieses Buches hinausführenden Literaturstudium dienen.

Umfang der Technik

Dieser Lehrtext befaßt sich einerseits nicht nur mit der **Daten**technik und der mit ihrer Hilfe durchführbaren **Daten**verarbeitung, sondern bezieht in gleicher Weise die Technik ein, mit deren Unterstützung die Informationsarten **Bild**, **Sprache** und **Text** verarbeitet werden, also Bildverarbeitung, Sprachverarbeitung und Textverarbeitung. Andererseits befaßt sich dieser Lehrtext nicht nur

mit der **Verarbeitung** der Informationsarten Bild, Daten, Sprache und Text (einschließlich ihrer Eingabe und Ausgabe), sondern auch mit der Technik, welche die **Übertragung** unterstützt (im allgemeinen als "Datenübertragung" bezeichnet). Dabei wird sowohl die Hardware als auch die Software behandelt. Ergänzend zur Hardware und Software der Verarbeitung und der Übertragung von Bild, Daten, Sprache und Text wird ausführlich auf die **Schutztechnik** eingegangen, da die Nutzung der Informations- und Kommunikationstechnik in Organisationen ohne eine ausgefeilte Schutztechnik heute nicht mehr denkbar ist.

In diesem Lehrbuch nur am Rande und ergänzend zur Technik werden einige Arbeits-, Entwicklungs-, Produktions- und Implementierungsverfahren der Technik (z.B. die Systemplanung und das Informationsmanagement), die in ihrer Gesamtheit gemeinsam mit der Technik die "Informations- und Kommunikations**technologie**" ausmachen, behandelt. Deshalb wird im Titel des Buches bewußt der Terminus "Informations- und Kommunikations**technik**" verwendet. Die "Verfahren der Technik" werden in anderen Lehrbüchern der Autoren behandelt (vgl. die im Vorwort und in den Abschnitten Quellenliteratur angegebenen Lehrbücher).

Sichtweisen auf die Technik

Bei der Auswahl des Lernstoffs und bei seiner Darstellung wurde versucht, die Sichtweise der **Betriebswirtschaftslehre** und die Sichtweise der **Wirtschaftsinformatik** auf die Informations- und Kommunikationstechnik in den Vordergrund zu rücken. Beide Sichtweisen sind nach Auffassung der Autoren dadurch gekennzeichnet, daß Informations- und Kommunikationstechnik Hilfsmittel für die Gestaltung von Informations- und Kommunikationssystemen ist, nicht also selbst zum Gegenstandsbereich der Betriebswirtschaftslehre und der Wirtschaftsinformatik gehört. Dies gilt zweifellos noch mehr für die Betriebswirtschaftslehre, die Technik im wesentlichen nur so weit interessiert, wie dies zur Benutzung von Informations- und Kommunikationssystemen erforderlich ist. Die Wirtschaftsinformatik erfordert oft ein vergleichsweise tieferes Verständnis für die Technik, auch wenn ihre Gestaltung nicht zu ihrem Gegenstandsbereich gehört.

Studierende der Betriebswirtschaftslehre stehen der Technik also im allgemeinen ferner als Studierende der Wirtschaftsinformatik. Im Zusammenhang mit dem Stoffinhalt der Lerneinheiten und der angegebenen Quellen- und Vertiefungsliteratur heißt das für Studierende der Wirtschaftsinformatik, daß sie die Literatur auch benutzen sollten, während der Stoffinhalt der Lerneinheiten für Studierende der Betriebswirtschaftslehre im allgemeinen ausreicht.

Teilgebiete der Wirtschaftsinformatik

Die Gliederung der Wirtschaftsinformatik als Wissenschaft in Teilgebiete kann mit einer Orientierung an den Phänomenen Mensch, Aufgabe sowie Informa-

tions- und Kommunikationstechnik (mit anderen Worten: am Phänomen "Informations- und Kommunikationssystem") vorgenommen und - davon ausgehend - um die Teilgebiete Systemplanung und Informationsmanagement ergänzt werden.

- Teilgebiet 1: Der **Mensch** als Komponente von Informations- und Kommunikationssystemen mit seinen Beziehungen zur Aufgabe und zur Informations- und Kommunikationstechnik.
- Teilgebiet 2: Die **Aufgabe** als Komponente von Informations- und Kommunikationssystemen mit ihren Beziehungen zum Menschen und zur Informations- und Kommunikationstechnik.
- Teilgebiet 3: Die **Informations- und Kommunikationstechnik** als Komponente von Informations- und Kommunikationssystemen mit ihren Beziehungen zum Menschen und zur Aufgabe.
- Teilgebiet 4: Die **Systemplanung** als die Menge der Methoden und Werkzeuge zur Erklärung und Gestaltung von Informations- und Kommunikationssystemen.
- Teilgebiet 5: Das **Informationsmanagement** als das gesamte Leitungshandeln in einer Organisation bezüglich ihrer Informations- und Kommunikationsfunktion und der für diese entwickelten und implementierten Informationsinfrastruktur.

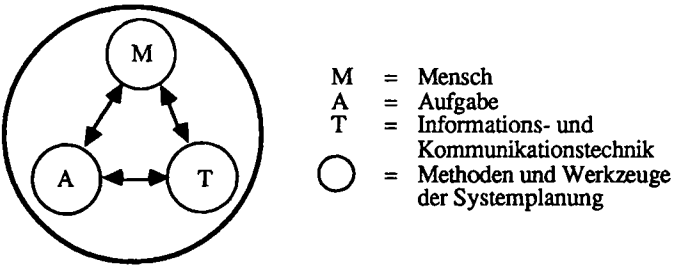


Abb. EINFÜ-1: Grundstruktur des Informations- und Kommunikationssystems

Abbildung EINFÜ-1 zeigt die Grundstruktur des Informations- und Kommunikationssystems mit den Phänomenen Mensch, Aufgabe und Technik, den zwischen ihnen bestehenden **Beziehungen** und der sie zusammenfügenden Systemplanung; das Teilgebiet Informationsmanagement ist wegen seiner unternehmensweiten Dimension daraus nicht erkennbar. "Informations- und Kommunikationstechnik" an sich ist kein Teilgebiet der Wirtschaftsinformatik; sie hat den Charakter eines Grundlagengebiets für die Wirtschaftsinformatik, den sie im Zusammenhang mit der Erklärung und der Gestaltung von Informations- und Kommunikationssystemen gewinnt. Grundlagenwissen über die Informations- und Kommunikationstechnik ist eine wichtige Voraussetzung für das Verständnis der Wirtschaftsinformatik.

Eine andere Systematik gliedert die Wirtschaftsinformatik in Teildisziplinen, und zwar in Allgemeine Wirtschaftsinformatik und Besondere Wirtschaftsinformatiken. Die **Allgemeine Wirtschaftsinformatik** befaßt sich mit den Phänome-

nen der Informationsfunktion, die aufgabenunabhängig sind. Ihre Erklärungen und Gestaltungshilfsmittel sind in jeder Art von Organisation (z.B. sowohl in einer Betriebswirtschaft als auch in einer Öffentlichen Verwaltung) zu beobachten; ihr Erkenntnisobjekt ist die Informationsfunktion von Organisationen schlechthin, völlig unabhängig von der Art der Organisation. Die **Besonderen Wirtschaftsinformatiken** befassen sich mit den Besonderheiten der Informationsfunktion in bestimmten Klassen von Organisationen, so z.B. die Betriebsinformatik mit den Besonderheiten der Informationsfunktion in Betriebswirtschaften. Diese Systematik ist aus Abbildung EINFÜ-2 ersichtlich. Der Lernstoff des Lehrbuchs ist im wesentlichen Grundlagenstoff der Allgemeinen Wirtschaftinformatik.

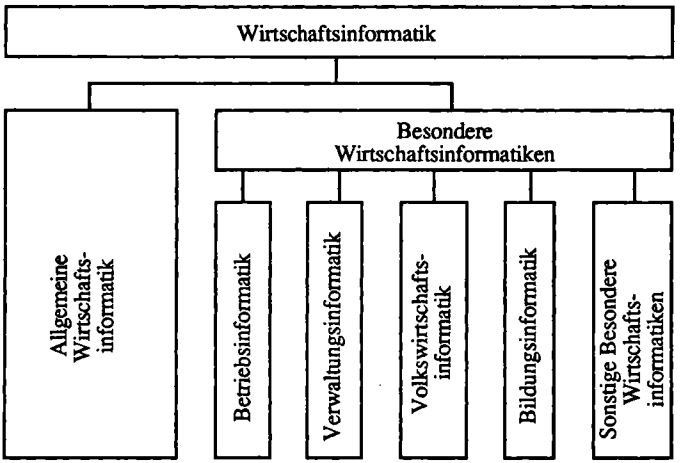


Abb. EINFÜ-2: Teildisziplinen der Wirtschaftsinformatik

Kontrollfragen

1. Welche Funktion haben die Lernziele und die Kontrollfragen?
2. Erläutern Sie die Funktion von Quellenliteratur und Vertiefungsliteratur.
3. Wodurch sind die Sichtweisen der Betriebswirtschaftslehre und der Wirtschaftsinformatik auf die Informations- und Kommunikationstechnik gekennzeichnet?
4. Nennen Sie die fünf Teilgebiete der Wirtschaftsinformatik.
5. Welche Bedeutung hat "Informations- und Kommunikationstechnik an sich" für die Wirtschaftsinformatik?

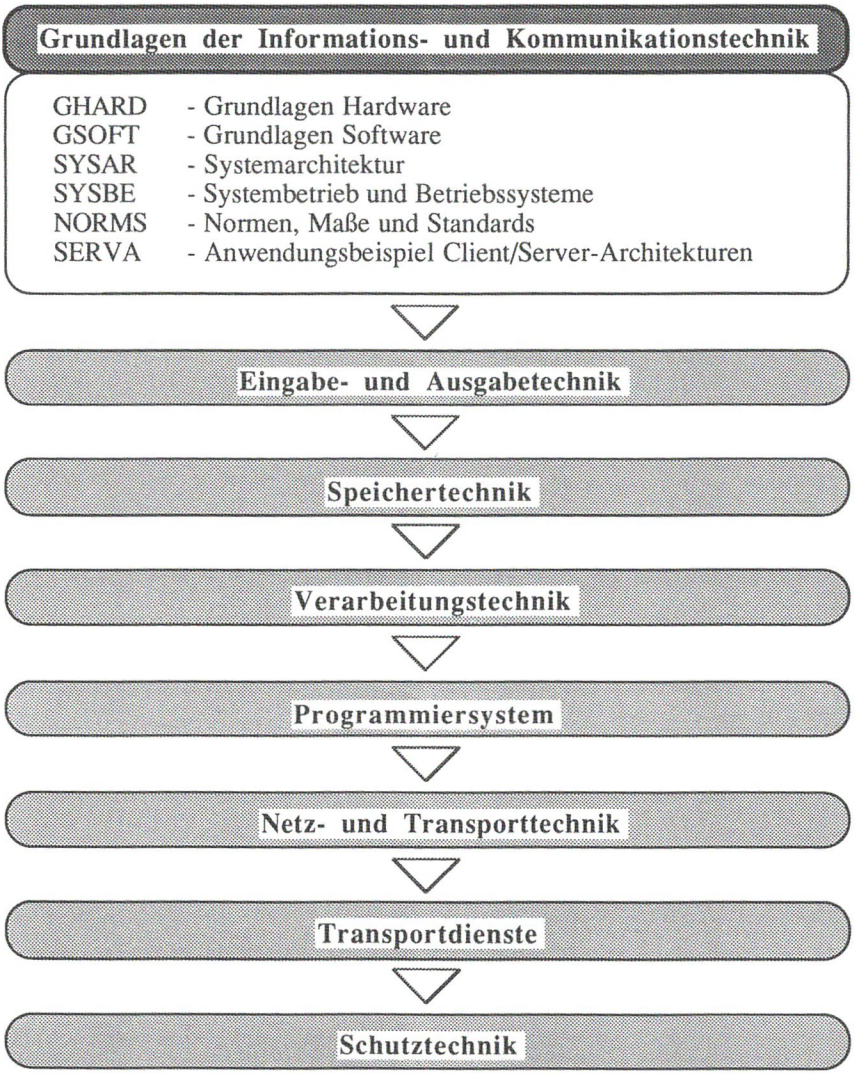
Quellenliteratur

Heinrich, L. J. und Roithmayr, F.: Wirtschaftsinformatik-Lexikon. 4. A., Oldenbourg Verlag, München/Wien 1992, insbes. Kapitel "Einleitung"

Vertiefungsliteratur

Heinrich, L. J.: Informationsmanagement - Planung, Überwachung und Steuerung der Informationsinfrastruktur. 4. A., Oldenbourg Verlag, München/Wien 1992

Heinrich, L. J.: Systemplanung - Planung und Realisierung von Informatik-Projekten (2 Bände). 6. A. (Bd. 1) bzw. 5. A. (Bd. 2), Oldenbourg Verlag, München/Wien 1994


Grundlagen der Informations- und Kommunikationstechnik
GHARD - Grundlagen Hardware
GSOFT - Grundlagen Software
SYSAR - Systemarchitektur
SYSBE - Systembetrieb und Betriebssysteme
NORMS - Normen, Maße und Standards
SERVA - Anwendungsbeispiel Client/Server-Architekturen
Eingabe- und Ausgabetechnik
Speichertechnik
Verarbeitungstechnik
Programmiersystem
Netz- und Transporttechnik
Transportdienste
Schutztechnik

GHARD - Grundlagen Hardware

Lernziele

Sie können den konstruktiven Aufbau eines Computers beschreiben. Sie kennen die wichtigsten Hardwarebauelemente und verstehen deren Zusammenspiel. Sie können eingesetzte Herstellungstechnologien nennen und deren Leistungsfähigkeit und Grenzen sowie die zukünftige Entwicklung abschätzen. Sie können eine Klassifikation der gebräuchlichsten Rechnerarten vornehmen und kennen die Haupteinsatzgebiete.

Definitionen und Abkürzungen

Bus (bus) = ein aus mehreren funktionsmäßig zusammengehörigen Signalleitungen bestehendes System, mit dem ein sequentieller Austausch von Signalen, Daten und Nachrichten zwischen den Funktionseinheiten eines Datenverarbeitungssystems möglich ist.

CPU = Abkürzung für Central Processing Unit.

Einzelplatzsystem (single user system) = ein Datenverarbeitungssystem, mit dem, bedingt durch die Hardware- und/oder Softwareausstattung, nur ein Benutzer arbeiten kann.

Integrierte Schaltung (integrated circuit) = eine Zusammenfassung mehrerer elektronischer Bauelemente, die auf einer Leiterplatte aufgebracht sind; unterschieden werden Standardchips (Prozessorbausteine, Speicherbausteine und Logikschaltungen), die für einen breiten Markt produziert werden, und kundenspezifische Schaltungen für spezielle Anwendergruppen. Synonym: Chip.

Kanal (channel) = eine Funktionseinheit, welche die Daten von der Zentraleinheit zur Peripherie und umgekehrt überträgt.

Kompatibilität (compatibility) = die Eigenschaft eines Datenverarbeitungssystems, ohne Anpassungsarbeiten oder Änderungen mit anderen Systemen zusammenarbeiten zu können.

Leiterplatte (board) = eine Vorrichtung zur Aufnahme elektrischer Baugruppen (Moduln) sowie deren Verbindung untereinander.

Mehrbenutzersystem (multiuser system) = ein Datenverarbeitungssystem, mit dem mehrere Benutzer gleichzeitig, aber voneinander unabhängig arbeiten können.

Mikroprozessor (microprocessor) = der Prozessor eines Mikrocomputers.

MIPS = Abkürzung für Million Instructions per Second; eine Maßeinheit für die Geschwindigkeit von Prozessoren. Synonym: MOPS (Million Operations per Second).

Multitasking (multitasking) = die Fähigkeit eines Betriebssystems, mehrere Aufgaben (engl.: task) in einem Datenverarbeitungssystem gleichzeitig bearbeiten zu können.

PCM = Abkürzung für Plug Compatible Manufacturer; ein Hersteller von Hardware, die steckerkompatibel zur Hardware des Marktführers IBM ist.

Peripherie (peripheral equipement) = eine Sammelbezeichnung für alle an die Zentraleinheit angeschlossenen Eingabegeräte, Ausgabegeräte und Speicher.

Prozessor (processor) = eine Funktionseinheit innerhalb des Datenverarbeitungssystems, die das Steuerwerk und das Rechenwerk umfaßt.

Vektorrechner (array processor) = ein Datenverarbeitungssystem, das die parallele Ausführung von Befehlen durch die Hardwarestruktur und den Befehlssatz unterstützt.

Überblick

Hardware bezeichnet die Gesamtheit aller physikalischen Bestandteile eines Datenverarbeitungssystems. Die Entwicklung der Hardwaretechnologie läßt sich an den **Rechnergenerationen** ablesen:

- In der **ersten Generation** verwendete man Röhrentransistoren.
- Die **zweite Generation** war durch die Einführung von Transistorschaltkreisen gekennzeichnet.
- Die **dritte Generation** beginnt mit dem Einsatz integrierter Schaltkreise und ist durch den Einsatz von SSI- und MSI-Schaltungen gekennzeichnet. Der Integrationsgrad beträgt bei SSI (Small Scale of Integration) 2 bis 64 Schaltelemente pro Chip, bei MSI (Medium Scale of Integration) 64 bis 2000 Schaltelemente pro Chip.
- In der **vierten Generation**, der die meisten derzeit eingesetzten Computer zuzurechnen sind, werden LSI-Schaltungen und VLSI-Schaltungen verwendet. Der Integrationsgrad beträgt bei LSI (Large Scale of Integration) 2000 bis 64000 Schaltelemente pro Chip, bei VLSI (Very Large Scale of Integration) 64000 bis 1.000.000 Schaltelemente pro Chip.
- Die Rechner der **fünften Generation** sollen durch ihre Architektur speziell für Expertensystem-Anwendungen umfangreiche Unterstützung bieten (vgl. dazu die Lerneinheiten SYSAR und WISSE). Angestrebt wird ferner eine weitere Steigerung der Integrationsdichte bei Schaltelementen auf über eine Million Schaltelemente pro Chip (ULSI, Ultra Large Scale of Integration).

Die einzelnen Entwicklungsstufen der Hardware sind nicht nur durch die zunehmende **Integrationsdichte** bei den Hardwarebausteinen gekennzeichnet, sondern spiegeln sich auch in Merkmalen wie Schaltzeiten, Leistungsaufnahme und Funktionalität wider. Mit der dritten Generation begann z.B. der Timesharing-Betrieb, der die Leistungsfähigkeit vor allem durch die Einsatzform enorm verbesserte (vgl. Lerneinheit SYSBE). Das Potential liegt also neben technischen Verbesserungen vor allem auch in der Organisation und Nutzung der Hardwarekomponenten, d.h. der Systemarchitektur (vgl. Lerneinheit SYSAR). Die Chipgröße bzw. die zunehmende Miniaturisierung kann als direkte Folge der skizzierten technischen Entwicklung angesehen werden, da die Reduktion von Schaltzeiten am einfachsten durch eine "Verkürzung" der Leitungswege realisiert wird. Parallel zu dieser Entwicklung stieg der Preis pro Chip nur langsam. Dies führte zu einem Preisverfall bei der Hardware und zu einer Durchdringung aller Bereiche der Wirtschaft und Verwaltung mit dem Computer.

Klassifikation von Computern

Die Begriffe Computer, Rechner und Datenverarbeitungssystem werden synonym verwendet. Die hier verwendete Klassifikation orientiert sich primär am **Einsatzbereich** von Computern, der indirekt auch Auskunft über ihre Leistungsfähigkeit gibt. Auf die Angabe der Rechnerleistung in Form von technischen Leistungsmerkmalen wie Speicherkapazität, Rechnerleistung, Zykluszeit, Übertragungsraten usw. wurde verzichtet. Der Grund ist in der permanenten Veränderung in diesem Bereich zu sehen, die eine längerfristig gültige Klassifikation schwierig macht. Außerdem ist die Rechnerleistung keine eindimensionale Größe. Sie hängt vielmehr von verschiedenen Faktoren wie Taktfrequenz des Prozessors, Verfügbarkeit von Pufferspeichern, Einsatz von Zusatzprozessoren für Spezialaufgaben (Ein-/Ausgabe, mathematische Funktionen) und vom Typ der auszuführenden Befehle ab. Der Wert kann de facto sowohl über als auch unter den von Herstellern angegebenen Werten liegen.

Nach dem Einsatzbereich werden Zentralrechner, Abteilungsrechner, Arbeitsplatzrechner und Spezialrechner unterschieden. Die Abgrenzung zwischen den einzelnen Klassen ist nicht immer eindeutig möglich. Die vielfältigen Vernetzungsmöglichkeiten (z.B. Anbindung von PCs und Workstations als Terminal am Zentralrechner) und die Zunahme der Rechnerleistung bei kleineren Computern bewirken einen fließenden Übergang der Grenzen (vgl. Lerneinheit NETAN).

Zentralrechner: Aufgrund ihres primären Einsatzes in Rechenzentren und Großunternehmen werden sie auch als **Großrechner** bezeichnet. Sie ermöglichen den zentralen Anschluß einer sehr großen Anzahl von Bildschirmarbeitsplätzen und Peripheriegeräten. Im allgemeinen erfordern sie Klimaanlagen und spezielles Bedienungspersonal. Zentralrechner werden traditionellerweise auch als Universalrechner bezeichnet, obwohl diese Bezeichnung heute auch für Mini- und Mikrorechner zutreffend ist.

Abteilungsrechner: Sie werden auch als **Minicomputer** bezeichnet; sie dienen gewöhnlich der gemeinsamen Nutzung weniger Anwendungssysteme in Großunternehmen (z.B. Buchhaltung und Lohnverrechnung) oder als Ersatz für den zentralen Großrechner in Zweigstellen sowie in kleinen bis mittleren Unternehmen. Es handelt sich im allgemeinen um multitaskingfähige Mehrplatzsysteme (vgl. Lerneinheit SYSBE), die überwiegend für kommerzielle Aufgaben eingesetzt werden.

Arbeitsplatzrechner: Die Nutzung solcher Rechner erfolgt meist nur durch einen oder wenige Anwender. Oft handelt es sich um Einplatzsysteme, die aber immer häufiger über die Möglichkeit des Multitasking verfügen. Ein Verbund mehrerer Arbeitsplatzrechner in einem Netz oder die Mitverwendung als Terminal für einen Mini- oder Großrechner sind durchaus üblich. Durch den modularen Hardwareaufbau und standardisierte Schnittstellen kann eine sehr umfangreiche Peripherie angeschlossen werden. Zur Klasse der Arbeitsplatzrechner zählen **Personalcomputer** (auch als PC oder Mikrocomputer bezeichnet) und **Workstations**. Im Unterschied zu herkömmlichen PCs weisen Workstations eine wesentlich höhere Prozessorleistung und Funktionalität auf (integrierte hochauflösende, graphische Benutzeroberfläche und integrierte Netzfähigkeit, Multi-Tasking). Als spezielle Form des PCs verdienen noch sogenannte **Laptops** Erwähnung. Dabei handelt es sich um eine tragbare Kleinausführung etwa im Format eines Aktenordners. Sie verfügen über die gleiche Funktionalität und Leistungsfähigkeit wie herkömmliche PCs. Man bezeichnet diese tragbaren PCs, die sowohl unabhängig (betrieben mit einer Batterie oder einem Ladegerät) als auch im Verbund mit anderen PCs (z.B. in einem Lokalen Netz) einsetzbar sind, auch als Portables oder Notebooks.

Spezialrechner: In diese Klasse fallen alle Computer, die nicht universell einsetzbar sind und deren Leistungen daher kaum miteinander verglichen werden können. Die Systemarchitektur solcher Rechner ist - im Gegensatz zu Universalrechnern - meist auf das zu unterstützende Anwendungsgebiet abgestimmt (vgl. Lerneinheit SYSAR). Beispiele sind Hybridrechner, Prozeßrechner, Superrechner und Parallelrechner. **Hybridrechner** verfügen sowohl über ein digitales als auch über ein analoges Rechenwerk und werden für die Simulation komplexer Systeme eingesetzt. **Prozeßrechner** dienen zur Überwachung und Steuerung industrieller oder sonstiger physikalisch/chemischer Prozesse und arbeiten typischerweise im Echtzeitbetrieb. Sie werden daher häufig auch als Leitrechner bezeichnet. **Superrechner** kommen vor allem in der Wissenschaft, im militärischen Bereich und in der Raumfahrt zum Einsatz. Weltweit sind nur wenige derartige Systeme installiert, an die höchste Anforderungen bezüglich Zuverlässigkeit und Geschwindigkeit gestellt werden. **Parallelrechner** verfügen über eine Hardwarestruktur, welche die rechnerinterne Verarbeitung zum Teil parallel durchführt. Damit wird bei bestimmten Aufgabenklassen eine wesentlich kürzere Bearbeitungszeit bewirkt. Zur Klasse der Parallelrechner, die heute eine immer größere Bedeutung gewinnen, zählen Multiprozessor-Systeme sowie Vektorrechner und Array-Rechner. Superrechner sind meist als Parallelrechner konzipiert (vgl. auch Lerneinheit SYSAR).

Funktionaler Aufbau des Computers

Computer setzen sich aus Funktionseinheiten zusammen, die bezüglich ihrer Aufgabenstellung klar abgegrenzt werden können. Prinzipiell müssen Funktionseinheiten zur **Eingabe, Verarbeitung und Speicherung** sowie **Ausgabe** von Daten vorhanden sein. Das **EVA-Prinzip** (EVA = Eingabe/Verarbeitung/Ausgabe) ist bis heute das beherrschende Paradigma für die Realisierung des gesamten Computers, aber auch einzelner Hardwarebausteine, geblieben. Zwischen den und innerhalb der einzelnen Funktionseinheiten wird durch Signalleitungen, die man als Bus-System bezeichnet, eine physikalische Verbindung hergestellt. Das logische Zusammenspiel wird durch die Systemarchitektur festgelegt (vgl. Lerneinheit SYSAR).

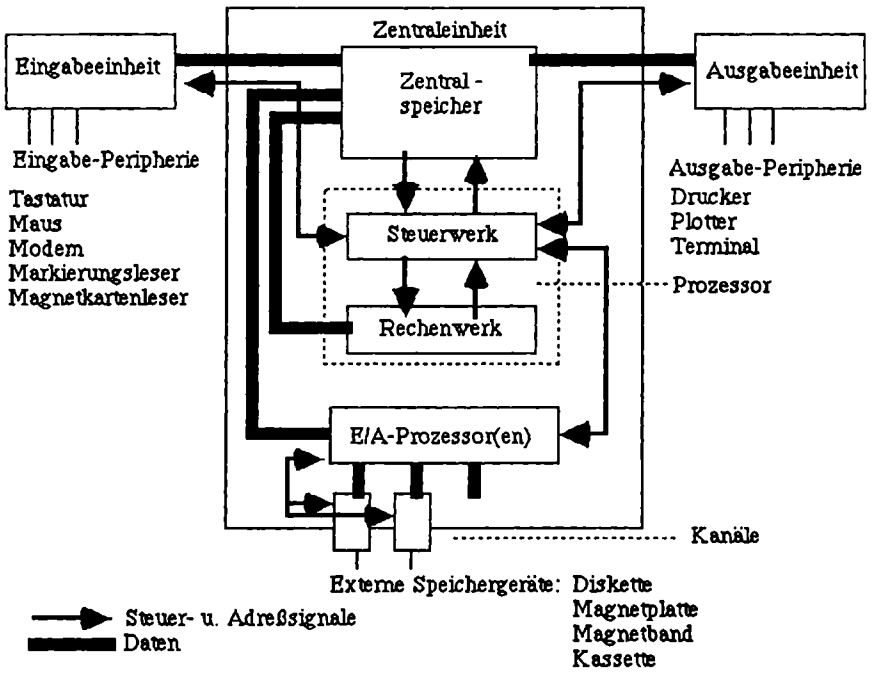


Abb. GHARD-1: Funktionaler Aufbau eines Computersystems

Abbildung GHARD-1 zeigt den funktionalen Aufbau eines Computersystems. Jede der dargestellten Funktionseinheiten verfügt selbst wieder über eine Detailstruktur mit einer internen Verarbeitungs- und Steuerlogik. Die **Eingabeeinheiten** und die **Ausgabeeinheiten** werden im Kapitel "Eingabe- und Ausgabetechnik" näher behandelt (siehe Lerneinheit EATEC). Die **Zentraleinheit** besteht aus dem Prozessor und dem Zentralspeicher; sie wird im nächsten Abschnitt genauer erläutert.

Die Auslegung der Signalleitungen des Bus-Systems wurde bei früheren Rechnergenerationen wegen der hohen Hardware-Preise nach dem **Minimalitätsprinzip** (Prinzip des minimalen Hardware-Aufwands) vorgenommen. Diese Restriktionen gibt es heute nicht mehr. Im Rahmen der Systemarchitektur wird versucht, durch eine optimale Gestaltung der internen "Verkehrswege" die Leistung zu beeinflussen. Die Bus-Architektur größerer Datenverarbeitungssysteme unterscheidet sich dabei funktional nicht von jener eines Mikrorechners. Um die Kommunikation zu regeln, muß der Zustand des Bus-Systems stets bekannt sein und kontrolliert werden. Diese Verwaltungsaufgaben übernimmt das Steuerwerk oder ein Spezialprozessor, der Busverwalter (engl.: arbiter) genannt wird. Bei den übertragenen Signalen können drei Signalarten unterschieden werden, nämlich Steuersignale, Adreßsignale und Daten.

- **Steuersignale** dienen zur Steuerung von gleich- oder untergeordneten Funktionseinheiten. Als Grundprinzip für Prozessorunterbrechungen (Interrupts) findet man die **Taktsteuerung** (synchrone Unterbrechung) und die **Ereignissteuerung** (asynchrone Unterbrechung).
- **Adreßsignale** dienen zur Adressierung von Registern oder Speicherstellen im Arbeitsspeicher.
- **Daten** werden zum Zweck der Verarbeitung, der Ein-/Ausgabe oder der Speicherung zwischen Arbeitsspeicher und Register übertragen.

Prozessor

Prozessor ist die zusammenfassende Bezeichnung für das Steuerwerk und das Rechenwerk. Bei einem Mikrorechner ist der Prozessor in einem Chip integriert, bei einem Minirechner sind es meist mehrere Chips. Bei Großrechnern besteht der Prozessor gewöhnlich aus einer größeren Anzahl von Schaltungen.

Das **Steuerwerk** sorgt für die zeitlich und funktional abgestimmte Ausführung der Befehle (vgl. Lerneinheit DATEN), die durch das Betriebssystem oder die Anwendungsprogramme vorgegeben sind, sowie für die Zuführung der Befehle und Daten aus dem Zentralspeicher ins Rechenwerk (vgl. Lerneinheit SYSAR). Das Steuerwerk besteht aus logischen Schaltungen und Registern. Die wichtigsten Register sind der Befehlszähler und das Programmstatuswort. Das **Rechenwerk** führt die arithmetisch/logischen Befehle aus. Es besteht im wesentlichen aus Registern und binären Schaltnetzen und verknüpft die Daten, die vom Steuerwerk durch die Adressen bezeichnet werden.

In Abhängigkeit von der Systemarchitektur (vgl. Lerneinheit SYSAR) werden weitere Spezial-Prozessoren eingesetzt. **Eingabe- und Ausgabeprozessoren** in Verbindung mit den Kanälen dienen zur Steuerung der peripheren Geräte. Dazu gehört die Abwicklung des Datenverkehrs, der Ausgleich der unterschiedlichen Verarbeitungsgeschwindigkeit dieser Geräte im Verhältnis zur CPU und die Behandlung der Unterbrechungssignale. Beim PC werden diese Aufgaben häufig vom Prozessor selbst wahrgenommen. **Service-Prozessoren** können zur autonomen Überwachung der Funktionsfähigkeit aller Einheiten eingesetzt wer-

den. **Daten-Prozessoren** übernehmen die Verwaltung des Zentralspeichers.

Zentralspeicher und Speicherzugriff

Der **Zentralspeicher** dient zur Aufnahme der Daten und Programme für die Betriebsdauer eines Computers. Zur Abgrenzung gegenüber dem **externen Speicher**, welcher der längerfristigen Aufnahme von Daten und Programmen dient (vgl. Lerneinheit SPMED), wird er auch als **interner Speicher** bezeichnet. Weitere Bezeichnungen für den Zentralspeicher sind Hauptspeicher und Arbeitsspeicher.

Nach den technischen Eigenschaften können beim Zentralspeicher folgende Speichertypen unterschieden werden:

ROM-Speicher (Read Only Memory): Damit bezeichnet man alle nicht flüchtigen Speicher, d.h. Speicher, bei denen der Speicherinhalt durch den Wegfall der Versorgungsspannung nicht verloren geht. ROMs werden meist zur Aufnahme jener Programme des Betriebssystems eingesetzt, die für den Bootstrap-Vorgang verwendet werden, also für das automatische Laden des Betriebssystems nach dem Einschalten des Computers. Der Speicherinhalt kann vom Benutzer nicht verändert werden. Der Vorgang der Speicherbelegung erfolgt beim ROM-Speicher mit Hilfe spezieller Geräte und wird - etwas verwirrend - als **Programmierung** bezeichnet. ROM-Speicher, die nur einmal "programmiert" werden können, bezeichnet man auch als PROM (Programmable Read Only Memory). Wiederverwendbare ROM-Speicher werden als EPROM (Erasable Programmable Read Only Memory) bezeichnet, wenn die Löschung durch Bestrahlung mit UV-Licht erfolgt, und als EEPROM (Electrically Erasable Programmable Read Only Memory), wenn elektrische Impulse verwendet werden.

RAM-Speicher (Random Access Memory): Damit wird jener Teil des Speichers bezeichnet, der für Software und für Daten zur Verfügung steht. Dieser Speicher ist flüchtig. Gegen einen ungewollten Datenverlust muß durch zusätzliche technische Einrichtungen (vgl. Lerneinheit HAWAS) oder Maßnahmen im Betriebssystem Vorkehrung getroffen werden. Nach der Bauweise unterscheidet man statische und dynamische RAM-Speicher. Bei dynamischen RAM-Speichern muß der Speicherinhalt nach einem Lese-Vorgang neu eingeschrieben werden, bei einem statischen RAM-Speicher wird der Speicherinhalt durch das Lesen nicht gelöscht.

Von der Funktion her ist der Zentralspeicher als **Speicherhierarchie** angelegt. Die Daten werden schrittweise aus "langsamen" Speichereinheiten zur Befehlsausführung in schnelle Registerspeicher übertragen, die direkt mit dem Rechenwerk verknüpft sind. Die wichtigsten Ebenen dieser Speicherhierarchie sind Arbeitsspeicher, Pufferspeicher und Register.

Der **Arbeitsspeicher** (engl.: working storage) setzt sich aus ROM und RAM zusammen. Aus dem RAM entnimmt der Prozessor schrittweise die **Befehle** eines Programms sowie die in den Befehlen adressierten **Daten.** Der **Pufferspeicher** (engl.: buffer memory) ist ein schneller Speicher, der Daten, die von einer Funktionseinheit zu einer anderen übertragen werden, vorübergehend aufnimmt. Seine Aufgabe ist der Geschwindigkeitsausgleich zwischen den langsamen Speichern und der schnellen Zentraleinheit; damit wird eine bessere Auslastung der Zentraleinheit erreicht. Eine ebenfalls übliche Bezeichnung für Pufferspeicher ist Cache-Speicher. **Register** (engl.: register) sind Bestandteile des Prozessors; sie haben eine geringe Speicherkapazität, die meist ein Wort beträgt. Mit "Wort" wird die maximale Anzahl von Zeichen bezeichnet, die mit den elementaren Befehlen des Computers gleichzeitig verarbeitet oder übertragen werden können. Die heute übliche Einteilung von Mikroprozessoren in 8-Bit-, 16-Bit- und 32-Bit-Prozessoren orientiert sich an diesem Sachverhalt. Register sind sehr schnelle Speicher, die der Arbeitsgeschwindigkeit der CPU angepaßt sind. Sie nehmen Daten und Ergebnisse für die kurze Zeit der Befehlsdurchführung auf.

Der **Speicherzugriff** erfolgt mit Hilfe eines Vorgangs, der als **Adressierung** bezeichnet wird. Der Zentralspeicher ist **direkt adressierbar**, das heißt, jede Speicherstelle hat eine eigene Adresse, die zum Auslesen und Speichern der Daten verwendet wird. Die Adressen der Speicherzellen sind fortlaufende Nummern, wobei bei Null zu zählen begonnen wird. Die Adresse wird aus verschiedenen Gründen bei den Befehlen eines Programms (vgl. Lerneinheit PROSP) nicht in der endgültigen Form angegeben, sodaß die Adresse vor dem eigentlichen Zugriff (Adressierung) auf die Daten durch spezielle Schaltungen des Steuerwerks erst ermittelt werden muß. Diesen Vorgang bezeichnet man als **Adreßdecodierung.** Gebräuchlich sind folgende Adressierungsarten, die in Befehlen mit mehr als einem Operanden auch gemischt vorkommen können:

- **Direkte oder absolute Adressierung**; eine Adreßdecodierung ist nicht erforderlich, die Adresse ist im Befehl enthalten.
- **Unmittelbare Adressierung**; die Daten, die vom Befehl verarbeitet werden sollen, sind bereits Teil des Befehls selbst.
- **Relative Adressierung**; die Adresse ergibt sich durch Addition des Inhalts des Programmzählers mit dem Verschiebungsfaktor, der im Befehl angeführt ist.
- **Registeradressierung**; die Adresse bezeichnet keine Speicherzelle im Arbeitsspeicher, sondern ein Register, in dem die Daten für die Befehlsausführung enthalten sind.
- **Indirekte Adressierung**; die eigentliche Adresse ist in jener Speicherzelle, deren Adresse im spezifizierten Register angegeben ist, zu finden.
- **Indizierte Adressierung**; die Adresse wird durch Addition des im Register angegebenen Index zur Adresse, die im Befehl vorhanden ist, ermittelt (Einsatz: Zugriff auf Tabellen).
- **Basisadressierung**; die Adresse wird durch Addieren des im Befehl angegebenen Verschiebungsfaktors zu der im Register spezifizierten (Basis)Adresse gebildet.

Schaltkreisfamilien

Für die Herstellung von Prozessoren, wie auch für die von Speichern und Ein-/Ausgabe-Schaltungen, werden verschiedene **Fertigungstechnologien** verwendet, die zu Schaltkreisfamilien zusammengefaßt werden. Hohe Packungsdichte und hohe Verarbeitungsgeschwindigkeit sind bei der Herstellung von Chips konkurrierende Ziele. Die Verwendung von Hardwareteilen aus unterschiedlichen Familien in einem Computersystem ist üblich. Es besteht daher ein entsprechender Bedarf an **Anpassungsschaltungen** für den Ausgleich unterschiedlicher Bausteineigenschaften. Die nachfolgend aufgezählten Schaltkreisfamilien sind heute gebräuchlich. Da sich das Typenspektrum ständig erweitert, wird dabei keine Vollständigkeit angestrebt, sondern versucht, die Konsequenzen der Verwendung einer bestimmten Technologie aufzuzeigen. Nicht näher eingegangen wird auf die verwendeten Basismaterialien (am häufigsten sind dies Germanium, Silizium und Gallium-Arsenid) sowie auf den Herstellungsprozeß der Chips.

- **STD-TTL** (Standard-Transistor-Transistor-Logik): Sie hat bis heute die weiteste Verbreitung gefunden und zeichnet sich vor allem durch hohe Schaltgeschwindigkeit aus.
- **Schottky-TTL**: Durch die Einführung der sogenannten Schottky-Diode zwischen Kollektor und Basis des Transistors wird gegenüber STD-TTL eine Steigerung der Schaltgeschwindigkeit erreicht.
- **ECL** (Emitter Coupled Logic): Der Vorläufer der Schottky-Dioden-Technik mit dem gleichen Effekt der Erhöhung der Schaltgeschwindigkeit.
- **LTTL** (Low power TTL): Standard-TTL-Schaltungen mit geringerer Leistungsaufnahme bei gleichzeitiger Verlängerung der Schaltzeiten.
- **LSTTL** (Low power Schottky-TTL): Trotz geringer Leistungsaufnahme werden die gleichen Schaltzeiten wie bei STD-TTL erzielt.
- **MOS** (Metal Oxid Semiconductor): Wegen der besonders hohen Packungsdichte werden vor allem Halbleiterspeicher mit großer Kapazität hergestellt. Die Leistungsaufnahme ist sehr gering. Unterschieden wird zwischen N-Kanal-MOS (NMOS) und P-Kanal-MOS (PMOS).
- **CMOS** (Complementary Metal Oxid Semiconductor): Der Platzbedarf und die Integrationsdichte sind etwa so groß wie bei STD-TTL. Die Vorteile dieser Schaltung liegen in der variablen Versorgungsspannung.
- **I2L** (Integrated Injection Logic): Die Einfachheit des verwendeten Prinzips der Stromeinkopplung in einen Transistorschalter ermöglicht eine sehr hohe Packungsdichte. Möglich sind große Schaltgeschwindigkeiten bei hohen Strömen und geringe Schaltgeschwindigkeiten bei niedrigen Strömen.

Demonstrationsbeispiel

Am Beispiel von TARGON von Nixdorf werden der funktionelle Aufbau und die Bus-Struktur eines Datenverarbeitungssystems gezeigt (vgl. Abbildung GHARD-2). TARGON ist ein Minirechner mit einer RISC-Architektur (vgl. Lerneinheit SYSAR) und einer UNIX-Oberfläche (vgl. Lerneinheit SYSBE). Die Hardwarekomponenten sind in der Fast-Schottky-TTL-Technologie ausgeführt. Aus Grün-

den der Fehlertoleranz sind alle Datenwege doppelt ausgelegt. Die Basisarchitektur ist durch eine durchsatzsteigernde Verteilung von Funktionen auf dedizierte Prozessoren gekennzeichnet. Die Aufgaben der Zentraleinheit werden vom Applikationsprozessor wahrgenommen. I/O-Prozessoren steuern die peripheren Geräte. Der Service-Prozessor ermöglicht die Ferndiagnose. Bei allen Prozessoren handelt es sich um 32-Bit-Mikroprozessoren.

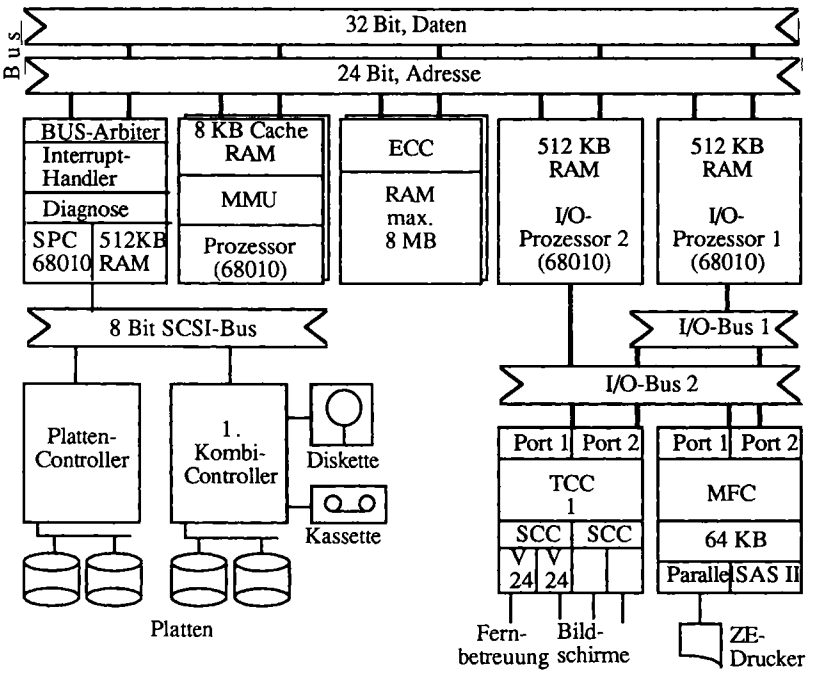


Abb. GHARD-2: Aufbau und Busstruktur des Systems TARGON

Kontrollfragen

1. Geben Sie eine Klassifikation der wichtigsten Rechnerarten.
2. Aus welchen Funktionseinheiten besteht ein Datenverarbeitungssystem?
3. Erläutern Sie den Aufbau und die Aufgaben der Zentraleinheit.
4. Beschreiben Sie die verschiedenen Speicherarten und ihre Funktion.
5. Was versteht man unter Adressierung; welche Adressierungsarten gibt es?

Quellenliteratur

Bauer, F. und Goos, G.: Informatik. 2 Bände, 4. A., Springer Verlag, Berlin et al. 1991

Goldschlager, L. und Lister, A.: Informatik. 3. A., Verlag C. Hanser, München/Wien 1990

Kurbel, K.: Das technologische Umfeld der Informationsverarbeitung. Arbeitsbericht Nr. 2, Institut für Wirtschaftsinformatik, Universität Münster 1991

Rechenberg, P.: Was ist Informatik? Verlag C. Hanser, München/Wien 1991

Vertiefungsliteratur

Chroust, G.: Mikroprogrammierung und Rechnerentwurf. Oldenbourg Verlag, München/Wien 1989

Oberschelp, W. und Vossen, G.: Rechneraufbau und Rechnerstrukturen. 2. A., Oldenbourg Verlag, München/Wien 1987

Schweizer, G. et al.: Mikrorechner - Architektur und Programmierung. Verlag Vieweg, Braunschweig/Wiesbaden 1986

GSOFT - Grundlagen Software

Lernziele

Sie können den Begriff und die Bedeutung der Software erklären und kennen ihre Erscheinungsformen. Sie können eine Unterscheidung zwischen den verschiedenen Softwarearten treffen. Sie verstehen den Zusammenhang zwischen Software und Algorithmus. Sie kennen Darstellungstechniken für Algorithmen. Sie verstehen den Aufbau von Datenverarbeitungssystemen als Software-Hardware-Hierarchie.

Definitionen und Abkürzungen

Anwender (user) = eine Person oder Personengruppe, die Informations- und Kommunikationstechnik bei der Bewältigung ihrer Aufgaben verwendet.

Betriebssystem (operating system) = die Gesamtheit aller Programme, die den Betrieb des Datenverarbeitungssystems ermöglichen, ohne auf eine bestimmte Anwendung Rücksicht zu nehmen.

CNC = Abkürzung für Computerized Numeric Control; siehe Numeric Control.

Datenbank (data base) = eine strukturierte Sammlung von Daten, die durch ein Datenbankmanagementsystem verwaltet und für einen zeitlich langen Gebrauch verfügbar gehalten wird.

Interpreter (interpreter) = ein Programm, das Anweisungen in einer bestimmten Programmiersprache auf einem Datenverarbeitungssystem ausführt, ohne daß diese Anweisungen vorher in die Maschinensprache übersetzt werden müssen.

Menü (menu) = eine ablauforganisatorisch sinnvoll geordnete Menge von Kommandos, die auf einem Ausgabegerät angezeigt wird und die es dem Anwender erlaubt, die nächste Aktion im Arbeitsablauf durchzuführen.

Metasprache (meta language) = eine Sprache zur Beschreibung einer (Programmier-)Sprache.

Multitasking (multitasking) = die Fähigkeit eines Betriebssystems, mehrere Aufgaben (engl. task) in einem Computer gleichzeitig zu bearbeiten.

Numeric Control (numeric control) = die Steuerung von Maschinen, zumeist von Werkzeugmaschinen, durch ein Programm; abgekürzt: NC.

Objektprogramm (object program) = ein in einer Maschinensprache vorliegendes und meist aus einer Programmiersprache mit einem Übersetzer übersetztes Programm. Synonym: Maschinenprogramm.

Programmiersprache (programming language) = eine zum Abfassen von Programmen geschaffene künstliche Sprache.

Quellprogramm (source program) = ein nicht in Maschinensprache abgefaßtes Programm. Synonyme: Quelltext, Quellcode.

Übersetzer (compiler) = ein Programm, das in einer Programmiersprache A abgefaßte Anweisungen (Quellprogramm) ohne Veränderung der Arbeitsvorschriften in die Anweisungen einer Programmiersprache B umwandelt (übersetzt).

Überblick

Neben den **Daten** bilden die **Algorithmen** den zweiten Grundpfeiler der Informations- und Kommunikationstechnik. Die Grundlage für dieses Verständnis ist eine **duale Weltsicht**. Demnach wird die Wirklichkeit in **Objekte** und **Aktionen** (an diesen Objekten), Dinge und Handlungen, passive und aktive Elemente, Statik und Dynamik usw. eingeteilt. Sowohl unser Denken als auch unsere Sprache sind von diesen Kategorien geprägt. Eine direkte Entsprechung findet sich aber auch in der Welt des Computers. Objekte und Dinge werden in Form von Datenobjekten bzw. Daten abgebildet, Aktionen und Handlungen werden als Algorithmen dargestellt. Diese Entsprechung ist die Erklärung für die universelle Einsetzbarkeit des Computers.

Der Algorithmus wird hier in enger Verbindung mit der Hardware gesehen, was auch in der sogenannten Software-Hardware-Hierarchie zum Ausdruck kommt, die in dieser Lerneinheit noch genauer erläutert wird. Programme oder Software sind lediglich verschiedene Manifestationen von Algorithmen, sie unterscheiden sich nicht substantiell, sondern nur in ihrer Darstellungsform voneinander und können ineinander übergeführt werden. Zum Verständnis und zur Abgrenzung der Begriffe ist es zweckmäßig, die verschiedenen Erscheinungsformen der Software und den Weg vom Algorithmus bis zum ausführbaren Programm näher zu betrachten (vgl. auch Lerneinheit PROSY).

Der erste Schritt besteht in der Formulierung der Handlungsanweisungen bzw. der Lösungsvorschrift für die Aufgabe, die mit dem Computer unterstützt werden soll. Das Ergebnis wird **Algorithmus** bezeichnet, der in dieser Form aber im allgemeinen nicht auf dem Computer ausführbar ist. Der Algorithmus wird im nächsten Schritt in ein Programm umgewandelt.

Ein **Programm** ist eine zur Lösung einer Aufgabe vollständige Anweisung zusammen mit allen dazu erforderlichen Vereinbarungen. Ein Programm setzt sich aus Befehlen einer Programmiersprache zusammen. Programme werden zunächst meist als **Quellprogramm**, d.h. in einer für den Menschen lesbaren Form erstellt. Die Transformation des Algorithmus in ein Quellprogramm wird als Programmierung bezeichnet, wenn sie manuell erfolgt. Zum Teil existieren

aber auch schon Werkzeuge, die diese Transformation automatisch durchführen oder zumindest unterstützen. In diesem Fall spricht man von Code-Generierung (vgl. Lerneinheit CASEE). Quellprogramme können im allgemeinen nicht direkt auf dem Computer ausgeführt werden (ausgenommen es wird ein Interpreter eingesetzt, vgl. Lerneinheit PROSP und Lerneinheit TOOLS).

Im dritten Schritt wird das Quellprogramm in das sogenannte **Objektprogramm** oder **Maschinenprogramm** transformiert, d.h. in jene Form, die auf dem Computer ausführbar ist. Man bezeichnet diese Transformation als Übersetzung (vgl. Lerneinheit TOOLS).

Software ist die zusammenfassende Bezeichnung für ein Programm in allen seinen Erscheinungsformen und Bestandteilen. Insbesondere zählt dazu auch die Dokumentation, die zum Verständnis des Algorithmus, seiner programmiertechnischen Umsetzung und als Bedienerhilfe bei der Programmausführung erforderlich sein kann. Ein besonders wichtiger Teil der Software ist nicht zuletzt auch die Bedienungsoberfläche, d.h. jener Teil der Software, mit dem der Mensch bei der Anwendung in Kontakt kommt (vgl. Lerneinheit SCHNI); dazu zählen Bildschirmmasken, Hilfe-Funktionen und Menüs sowie alle Befehle, die zur Benutzung des Datenverarbeitungssystems direkt eingegeben werden.

Software wird aber auch als übergreifende und zusammenfassende Bezeichnung für den gesamten "immateriellen" Teil eines Datenverarbeitungssystems verwendet. Sie umfaßt also die Gesamtheit aller **Programme**. Dazu zählen neben dem Bestand an Anwendungsprogrammen auch alle Programmierhilfen, Systemprogramme und insbesondere das Betriebssystem. Ein wesentlicher Teil der Komponenten eines Datenverarbeitungssystems ist also Software. Ein weiteres Charakteristikum ist, daß sowohl die Entwicklung als auch der Einsatz von Software ebenfalls wieder mit Hilfe von Software erfolgt.

Die Situation, die heute bei der Bereitstellung von Software besteht (vgl. auch Lerneinheiten PROSY und ENDAN), kann durch drei prinzipiell verschiedene Lösungsansätze beschrieben werden:

- Entwicklung von Individualsoftware durch Spezialisten (Programmierer);
- Einsatz von Standardsoftware (Beschaffung fertiger Software auf dem Software-Markt);
- Verlagerung von Teilen der Softwareentwicklung in die Fachabteilung (individuelle Datenverarbeitung, Einsatz von Endbenutzerwerkzeugen).

Der dargestellte Prozeß der Softwareentwicklung sowie die verschiedenen Erscheinungsformen und Zwischenstadien orientieren sich an den prozeduralen Programmiersprachen, bei denen der Algorithmus explizit beschrieben werden muß. Man nennt diese Sprachen daher auch algorithmische Sprachen. Bei Programmiersprachen der vierten und fünften Generation ist diese Grundstruktur oft nicht mehr erkennbar und die Entwicklung eines Programms erfordert vom Programmierer z.T. eine völlig neue Denkweise (vgl. Lerneinheit PROSP). Soft-

ware, Software-Entwicklungsumgebung und Programmiersprache lassen sich oft nicht mehr klar voneinander abgrenzen. Auch Prototyping, CASE-Systeme und Hypertext-Systeme tragen zur Veränderung der "klassischen" Art der Softwareentwicklung bei. Das Angebot an Wegen wird vielfältiger, Ziel und Endergebnis ist aber unverändert ein auf dem Computer ausführbares Programm. Weitere Einzelheiten zum Prozeß der Softwareentwicklung und zu den Werkzeugen, die dabei Verwendung finden, werden in den Lerneinheiten PROSY, TOOLS und CASEE dargestellt.

Klassifikation von Software

Allgemein wird zwischen **systemorientierter Software** (Systemsoftware) und **problemorientierter Software** (Anwendungssoftware) unterschieden. Abbildung GSOFT-1 zeigt diese Klassifikation im Überblick.

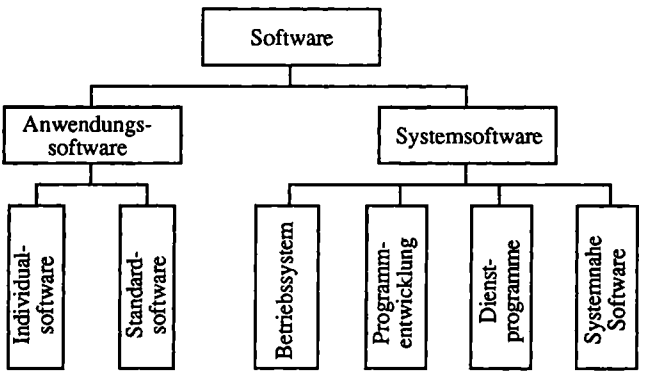


Abb. GSOFT-1: Klassifikation von Software

Die **Systemsoftware** umfaßt alle Programme, die zum Betrieb des Datenverarbeitungssystems notwendig sind oder diesen unterstützen. Der wichtigste Bestandteil ist dabei das **Betriebssystem**, dessen Aufgaben und Funktionsweise in der Lerneinheit SYSBE näher erläutert werden. Zur **Programmentwicklung** werden z.B. Übersetzer, Lader, Binder, Debugger und Endbenutzerwerkzeuge verwendet (vgl. Lerneinheiten TOOLS, CASEE und ENDAN). Als Beispiel für **Dienstprogramme** können Sortierprogramme und Diagnoseprogramme genannt werden. Sortierprogramme ordnen Datensätze nach Kriterien, die vom Benutzer festgelegt werden. Meist besteht bei Sortierprogrammen auch die Möglichkeit, Daten nach bestimmten Merkmalen zu selektieren. Diagnoseprogramme werden zur Feststellung von Softwarefehlern und Hardwarefehlern verwendet. Unter **systemnaher Software** werden schließlich jene Programme zusammengefaßt, die keiner der genannten Klassen eindeutig zugeordnet werden können. Dazu gehören Netz-Software (vgl. Lerneinheit FINET), Datenbanksysteme (vgl. Lerneinheit DAMOD), Data Dictionaries, Accounting- oder Abrechnungssoftware und Sicherungssoftware (vgl. Lerneinheit SCHAN).

Die **Anwendungssoftware** umfaßt die Programme, die zur Unterstützung oder Durchführung betrieblicher Aufgaben, aber auch im technisch/wissenschaftlichen Bereich eingesetzt werden. Bei der Anwendungssoftware wird weiter zwischen Standardsoftware und Individualsoftware unterschieden. **Standardsoftware** wird für den Gebrauch bei mehreren Anwendern entwickelt. **Individualsoftware** wird vom Hersteller gewöhnlich nur für einen bestimmten Anwender erstellt.

Eine andere Möglichkeit der Klassifikation von Anwendungssoftware ist die Unterscheidung nach **Haupteinsatzgebieten**. Man unterscheidet u.a. Branchensoftware (z.B. Software für Banken, Software für Speditionen, Software für Reisebüros), branchenneutrale Software (z.B. Software für die Finanzbuchhaltung, Software für die Lohn- und Gehaltsabrechnung) und anwendungs- bzw. funktionsbezogene Software (z.B. Software für die Lagerwirtschaft, Software für die Produktionswirtschaft, technisch/wissenschaftliche Software).

Eine Sonderstellung nimmt noch die **individuelle Datenverarbeitung** (IDV) ein. Primär versteht man darunter die Entwicklung von Software durch Endbenutzer oder Mitarbeiter der Fachabteilung mittels leistungsfähiger "Endbenutzerwerkzeuge" (vgl. Lerneinheit ENDAN). Die Entwicklung von Software ist aber nicht das einzige Merkmal der IDV. Zumindest gleichberechtigt ist die freie Entscheidung des Benutzers über Art und Umfang des Softwareeinsatzes. Unter diesem Gesichtspunkt zählt z.T. auch Standardsoftware zur IDV. Nicht immer ist eine eindeutige Abgrenzung zwischen konventioneller Datenverarbeitung und individueller Datenverarbeitung möglich. Standardsoftware wird immer dann der konventionellen Datenverarbeitung zuzurechnen sein, wenn Massendaten verarbeitet werden und der Benutzer bei der Aufgabendurchführung nicht selbst über den Einsatz eines Programms entscheidet. In diesem Sinn kann z.B. Standardsoftware für Lagerverwaltung oder Kostenrechnung nicht zur IDV gezählt werden. Beispiele für die IDV unter Einsatz von Standardsoftware sind: Programme zur Bilanzanalyse, Programme zur Erstellung von Finanzplänen, Controlling Software sowie Programme zur Terminverfolgung und Projektmanagementsoftware.

Algorithmen

Ein Computer kann nur solche Aufgaben unterstützen, die durch Befehle oder Operationen beschrieben werden können, welche auf dem Computer ausführbar sind. Die Beschreibung des Vorgehens, wie eine bestimmte Aufgabe auszuführen ist, bezeichnet man als **Algorithmus**. Die Ausführung des Algorithmus, d.h. den Vorgang selbst, bezeichnet man als **Prozeß**. Der Algorithmus-Begriff ist keine Erfindung der Informatik, vielmehr kommt er auch in anderen Bereichen sowie in vielfältigen Erscheinungsformen vor (z.B. in Form von Rezepten, Montageanleitungen, Gebrauchsanweisungen und Notenblättern). Es ist unerheblich, ob die Darstellung des Algorithmus mehr in natürlicher Sprache erfolgt oder ob die einzelnen Verfahrensschritte formal beschrieben werden. Wichtig ist hingegen,

daß die Beschreibung der Schritte **eindeutig** ist und daß die Schritte **ausführbar** sind. Außerdem muß die Anzahl der Schritte **endlich** sein, da der Algorithmus sonst nie ein **konkretes Ergebnis** (Determiniertheit) erbringen könnte.

Der Algorithmus ist die Voraussetzung dafür, daß zur Durchführung einer Aufgabe ein **Programm** in einer bestimmten **Programmiersprache** formuliert werden kann. Der Algorithmus ist das Verfahren zur Aufgabendurchführung, dargestellt in einer "neutralen" Sprache. Ein Algorithmus besteht aus elementaren Aktionen (Befehlen, Anweisungen, Operationen) und aus Regeln über die Reihenfolge der Durchführung dieser Aktionen, den sogenannten Ablaufstrukturen. Der Algorithmus ist sowohl unabhängig von der Programmiersprache, in der das Programm geschrieben werden soll, als auch vom Computer, auf dem das Programm später ausgeführt wird. Das bedeutet nicht, daß die Programmiersprache und der Computer bei der Formulierung des Algorithmus unwichtig sind. Sie ermöglichen z.B. die raschere, billigere und zuverlässigere Ausführung von Programmen. Sie haben Einfluß darauf, wie einfach oder wie kompliziert eine Aufgabe als Algorithmus ausgedrückt wird, d.h. letztlich auf die menschliche Anstrengung, die bei der Programmierung aufzuwenden ist.

Die **Entwicklung eines Algorithmus** ist eine kreative, geistige Tätigkeit, für welche die genauen Vorgehensschritte nicht angegeben werden können. Die Tätigkeit kann zwar vorstrukturiert und z.T. methodisch unterstützt werden, sie ist aber selbst nicht algorithmisierbar. Daraus leitet sich die Frage ab, welche Tätigkeiten oder Aufgaben als Algorithmus darstellbar sind. Diese Frage wird in der Informatik unter dem Titel "**Berechenbarkeit**" untersucht. Da es für viele Aufgaben mehrere mögliche Algorithmen gibt, sind auch die **Eigenschaften von Algorithmen** von Interesse. Wichtige Eigenschaften sind die Korrektheit sowie die Dauer der Ausführung und der Speicherbedarf, die man auch unter dem Begriff Komplexität von Algorithmen zusammenfaßt. Die Untersuchung solcher Eigenschaften ermöglicht den Vergleich von Alternativen, aber auch die Überprüfung von Algorithmen, z.B. ob sie mit der verfügbaren Computerleistung realisierbar sind. Für manche Aufgaben waren zwar seit langer Zeit Verfahren bekannt, sie konnten aber erst mit zunehmender Leistungsfähigkeit der Computer ausgeführt werden.

Wesentlich für die **Notation von Algorithmen** sind die Verständlichkeit und die Eindeutigkeit sowie eine möglichst automatische Transformation in die formale Darstellung durch eine Programmiersprache. Einen bedeutenden Fortschritt in diesem Zusammenhang stellt das 1966 von Böhm und Jacobini aufgestellte **Strukturtheorem** dar. Die Kernaussage dieses Theorems ist, daß sich jeder Algorithmus durch **drei Grundbausteine** oder **Darstellungselemente**, nämlich Sequenz, Entscheidung (Verzweigung) und Wiederholung (Iteration), darstellen läßt. Die einzelnen Verarbeitungsschritte, die aus diesen Grundbausteinen zusammengesetzt werden, werden meistens hintereinander ausgeführt (**sequentielle Ablaufstruktur**). Manche Aufgaben erlauben oder erfordern dagegen eine gleichzeitige Ausführung (**parallele Ablaufstruktur**). Von **kollateraler Ablaufstruktur** spricht man, wenn in einem Algorithmus beide Formen vorkom-

men. Algorithmen haben im allgemeinen einen eindeutig vorgeschriebenen Ablauf (deterministische Algorithmen). Dies muß jedoch nicht unbedingt so sein. Deterministische Algorithmen sind erstmals in Verbindung mit der Expertensystem-Technologie in breiterem Umfang erfolgreich eingesetzt worden (vgl. Lerneinheit WISSE). Nicht-deterministische Algorithmen sind nur dann interessant, wenn sie zu einem eindeutig bestimmten Ergebnis führen, das heißt auch nicht-deterministische Algorithmen müssen determiniert sein.

Heute existiert ein breites Spektrum an **Darstellungstechniken** und Ausdrucksmitteln für Algorithmen. Es reicht von **graphischen Darstellungsformen** (z.B. Ablaufdiagramm, Flußdiagramm, Struktogramm) bis zu **verbalen Darstellungsformen** (z.B. Pseudocode). Die Unterscheidung ist auch nach dem Grad der Formalisierung möglich, der von der **natürlichsprachlichen Beschreibung** über die **halbformale Darstellung** mit sogenannten Algorithmen-Beschreibungssprachen bis zur **formalen Darstellung** in einer Programmiersprache reicht. Graphische Darstellungstechniken zeichnen sich durch Flexibilität und Anschaulichkeit aus. Sie sind bei einer entsprechenden Gliederung auch übersichtlich, jedoch relativ weit von Programmiersprachen entfernt. Der Vorteil formaler Darstellungsformen ist, daß sie mit geringem Aufwand in eine Programmiersprache transformiert werden können. Dafür sind sie weniger anschaulich und erfordern üblicherweise bei der Darstellung einen höheren Aufwand sowie entsprechende Kenntnisse und Übung in der Anwendung. In CASE-Systemen werden meist verschiedene Darstellungstechniken angeboten, um den Prozeß der Software-Entwicklung möglichst gut zu unterstützen (vgl. Lerneinheit NORMS und CASEE).

Software/Hardware-Hierarchie

Die Programmausführung erfolgt durch den Computer. Für Programme wird in diesem Zusammenhang häufig auch der Begriff Software verwendet, um sie gegenüber der Hardware klar abzugrenzen. Der Benutzer kommt primär mit der Benutzeroberfläche der Anwendungssoftware in Berührung. Die Anwendungssoftware ist in einer bestimmten Programmiersprache geschrieben; sie baut auf dem Vorhandensein eines Übersetzers auf. Bei der Programmausführung werden die Dienste des Betriebssystems in Anspruch genommen. Das Betriebssystem wiederum baut auf dem Prozessor, dem Speicher und den Eingabe- und Ausgabeeinheiten auf. Aus dieser Darstellung geht hervor, daß Computersysteme als Hierarchie von Software verstanden werden können (vgl. Abbildung GSOFT-2). Die Hierarchie könnte nach unten weiter fortgesetzt werden, da die einzelnen Hardware-Komponenten selbst wieder aus einfacheren Bauteilen zusammengesetzt sind.

Die Grenze zwischen Hardware und Software ist allerdings nicht so eindeutig festgelegt, wie man dies aufgrund der Abbildung GSOFT-2 annehmen könnte. Vielmehr erfüllt jede Komponente eines Computers bestimmte Funktionen (z.B. Übersetzung eines Quellprogramms in ein ausführbares Objektprogramm). Man-

che dieser Funktionen werden traditionellerweise softwaremäßig realisiert, andere durch Hardware. Diese Zuordnung ist aber nicht fest vorgegeben, d.h. im Prinzip kann jede Funktion sowohl durch Hardware als auch durch Software realisiert werden. Die Grenzen zwischen Hardware und Software sind fließend und die Zuordnung bestimmter Funktionen wird sich mit dem technologischen Fortschritt immer wieder wandeln. Die Wahl der Realisierungsform wird von Größen wie Kosten für die Entwicklung, Leistung der Komponente usw. beeinflußt.

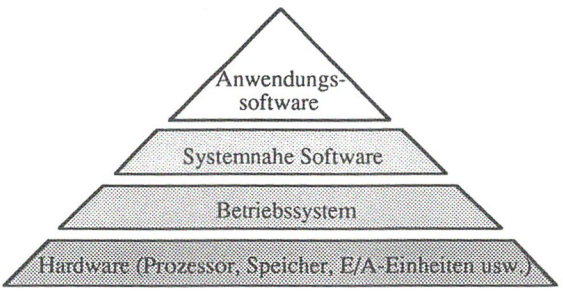


Abb. GSOFT-2: Software-Hardware-Hierarchie

Die enge Verbindung zwischen Hardware und Software drückt sich auch darin aus, daß Computer nur selten als Gerät oder Maschine bezeichnet werden (was den Hardware-Charakter betonen würde), sondern viel häufiger als **System** oder Computersystem. Eine weitere Entsprechung findet sich im sogenannten **EVA-Prinzip** (Eingabe-Verarbeitung-Ausgabe, d.h. das gesuchte Ergebnis wird aus einem gegebenen Ausgangszustand ermittelt), welches die Grundlage für die Realisierung aller gängigen Hardwarebausteine bildet (vgl. Lerneinheit GHARD). Aus der Anwendung des gleichen Prinzips beim Entwurf von Algorithmen kann ihre grundsätzliche Ersetzbarkeit durch Hardware abgeleitet werden.

Der Vorgang der bewußten Verlagerung einer Funktion von einer höheren Ebene auf eine niedrigere, hardwarenahe Ebene wird als **Migration** bezeichnet. Eine wichtige Rolle spielt dabei häufig die Firmware (Mikroprogramme). **Firmware** bezeichnet "Bauteile" eines Computers, welche die Eigenschaften von Hardware und Software vereinen. Anders ausgedrückt ist Firmware eine besonders flexible Form der hardwaremäßigen Realisierung, die durch ihre Programmierbarkeit auf einer Ebene zwischen Hardware und Software einzuordnen ist.

Demonstrationsbeispiel

Gezeigt werden zwei Erscheinungsformen eines Programms, nämlich die Benutzeroberfläche und das Quellprogramm, und der Unterschied zur traditionellen Software-Entwicklung wird demonstriert. Als Demonstrationsobjekt wurde ein Anwendungsprogramm ausgewählt, mit dessen Hilfe Lesbarkeits-Indizes für Texte ermittelt werden können. Das Programm kann z.B. dazu eingesetzt wer-

den, die Softwaredokumentation anhand bestimmter stilistischer Merkmale auf ihre Qualität zu überprüfen. Der **Algorithmus** wird durch die Eingabe, die Verarbeitung und die Ausgabe bestimmt (EVA-Prinzip). Als **Eingabe** dienen Texte bzw. Textstichproben. Die **Verarbeitungsschritte** sind durch die Berechnungsvorschriften der Lesbarkeitsformeln vorgegeben, die von der Sprachwissenschaft entwickelt worden sind. Dazu gehört auch die Ermittlung der Parameter (z.B. Anzahl der Wörter, Anzahl der Sätze), die in diesen Formeln als Berechnungsgrundlage dienen. Die **Ausgabe** ergibt sich in Form der Rechenergebnisse (z.B. Lesbarkeitsindex von Flesch, Fog-Index).

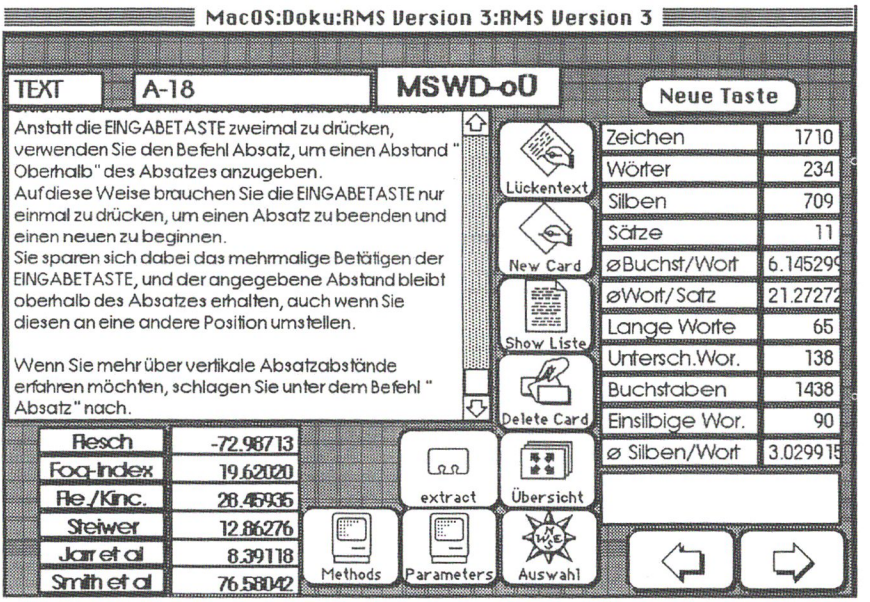


Abb. GSOFT-3: Benutzeroberfläche eines Anwendungsprogramms

Abbildung GSOFT-3 zeigt die **Benutzeroberfläche** des Programms. Diese ist mit Fenstertechnik und Pop-up-Menüs gestaltet. Im Eingabefenster sieht man den Ausschnitt eines Textes, der analysiert wird. Ebenfalls zu sehen sind die Ergebnisse, nämlich die ermittelten Parameter und die errechneten Lesbarkeits-Indizes. Die verfügbaren Programmfunktionen können durch Anklicken der "Buttons" mit dem mausgesteuerten Cursor ausgeführt werden. Abbildung GSOFT-4 zeigt einen Ausschnitt aus dem **Quellprogramm**, das sich "hinter" dem Button "Parameters" verbirgt, und dessen Aufgabe in der Ermittlung und Anzeige der Parameter des analysierten Textes besteht.

Das Programm wurde mit Hypercard entwickelt, einem prototyping-orientierten, integrierten Entwicklungswerkzeug, das auf Apple Macintosh-Computern zur Verfügung steht. Dadurch ergeben sich Unterschiede gegenüber der Entwicklung eines Programms in einer herkömmlichen Programmiersprache. Algorithmen im

traditionellen Sinn müssen nur für die Programmfunktionen "hinter" den Buttons entwickelt werden. Dabei stellt die Programmiersprache sehr mächtige Sprachelemente zur Verfügung, sodaß z.B. der Vorgang des Zählens der Wörter und Sätze im Algorithmus nicht mehr in seinen Einzelschritten dargestellt werden muß (vgl. Abbildung GSOFT-4). Die Benutzeroberfläche (d.h. die Bildschirmmasken und ihre logischen Verbindungen) kann unabhängig von der Formulierung der Algorithmen bzw. der Programmierung gestaltet und verändert werden. Das Programm selbst ist in die Entwicklungsumgebung eingebunden und wird durch einen Interpreter ausgeführt.

```
-- Anzahl buchstaben
get number of characters of field "TEXT"
put it into field "tb"
-- Anzahl Wörter
get number of words of field "TEXT"
put it into field "tw"
-- Anzahl Sätze
-- Die einzelnen Sätze müssen unbedingt durch ".;!?" und return
-- voneinander getrennt sein !!!!
put 0 into it
repeat with i= 1 to the number of lines in field "TEXT"
  put line i of field "TEXT" into Satz
  if number of words in Satz > 0 then
    if last char of Satz is in ".;?!" then
      put it + 1 into it
    else
      if char (length(Satz) - 1) of Satz is in ".;?!" then
        put it + 1 into it
      end if
    end if
  end if
end repeat
put it into field "ts"
-- average words pro Satz
put (field "tw"/ field "ts") into field "ow_s"
--
show field "Anderes Feld"
show background button "OKAY"
```

Abb. GSOFT-4: Teil des Quellprogramms zum Ermitteln der Parameter

Kontrollfragen

1. Welche Softwarearten werden unterschieden?
2. Welche Bedeutung hat der Algorithmus bei der Softwareentwicklung?
3. Erläutern Sie Gemeinsamkeiten und Unterschiede zwischen Programmen und Algorithmen.
4. Beschreiben Sie die wesentlichen Eigenschaften von Algorithmen.
5. Was versteht man unter Software-Hardware-Hierarchie?

Quellenliteratur

Bauer, F. und Goos, G.: Informatik. 2 Bände, 4. A., Springer Verlag, Berlin et al. 1991

Goldschlager, L. und Lister, A.: Informatik. 3. A., Verlag C. Hanser, München/Wien 1990

Heinrich, L. und Burgholzer P.: Systemplanung - Die Planung von Informations- und Kommunikationssystemen. Oldenbourg Verlag, München/Wien, Band 1, 5. A. (1991) bzw. Band 2, 4. A. (1990)

Rechenberg, P.: Was ist Informatik? Verlag C. Hanser, München/Wien 1991

Vertiefungsliteratur

Balzert, H.: Die Entwicklung von Softwaresystemen. 2. A., B.I.-Wissenschaftsverlag, Mannheim et al. 1984

Bolkart, W.: Programmiersprachen der vierten und fünften Generation. Verlag McGraw Hill, Hamburg 1987

Knuth, D.: The Art of Computer Programming. 3 Bde., 2. A., Verlag Addison Wesley, Reading/Mass. 1982

Meyer, B.: Objektorientierte Softwareentwicklung. Verlag C. Hanser, München/Wien 1990

Ottmann, T. und Widmayer, P.: Algorithmen und Datenstrukturen. B.I. Wissenschaftsverlag, Mannheim et al. 1990

SYSAR - Systemarchitektur

Lernziele

Sie können die Aufgaben der Systemarchitektur erklären und verstehen den Einfluß von Architekturprinzipien auf die Leistung eines Computers. Sie können das Von-Neumann-Prinzip und seine Schwächen erläutern. Sie kennen den Entwicklungsstand der heute verbreiteten Systemarchitektur bei gängigen Rechnersystemen sowie Architekturalternativen und Entwicklungstendenzen. Sie können den Entwurfsprozeß der Systemarchitektur und die dafür verwendeten Hilfsmittel nennen.

Definitionen und Abkürzungen

Befehlssatz (instruction set) = die Menge aller Befehle, die von einem Datenverarbeitungssystem ausgeführt werden können.

Fehlertolerantes System (fault tolerant system) = ein System, das auch mit einer begrenzten Anzahl fehlerhafter Subsysteme seine spezifizierte Funktion erfüllt und fähig ist, Fehler selbst zu erkennen und zu lokalisieren.

Firmware (firmware) = die Menge aller Mikroprogramme eines Rechners.

Kompatibilität (compatibility) = die Eigenschaft eines Datenverarbeitungssystems, ohne Anpassungsarbeiten oder Änderungen mit anderen Systemen zusammenarbeiten zu können.

Mainframe (mainframe) = ein leistungsfähiges Datenverarbeitungssystem, dessen Hauptaufgabe die Bereitstellung zentraler Computerleistung für eine große Anzahl von Benutzern mit unterschiedlichen Anforderungen ist.

Mehrprozessorsystem (multiprocessor system) = ein Datenverarbeitungssystem, bei dem ein Zentralspeicher von mehreren Prozessoren benutzt wird.

Mehrrechnersystem (multicomputer system) = eine gemeinsame Funktionseinheit (häufig das Betriebssystem), welche die Zusammenarbeit mehrerer Datenverarbeitungssysteme, die über eigene Hardwareressourcen verfügen, steuert.

Mikroprogramm (microprogram) = eine Folge von Befehlen zur Realisierung der Befehle konventioneller Maschinensprachen.

Operationsprinzip (operation principle) = das funktionelle Verhalten der Systemarchitektur, das durch Festlegen einer Informationsstruktur (Datenwege) und einer Kontrollstruktur (Steuerlogik) gegeben ist.

Rechner (computer) = siehe Datenverarbeitungssystem.

Rechnerfamilie (computer family) = eine Menge von Computern mit gleicher Architektur oder gleicher Benutzeroberfläche, aber mit verschiedener Leistung; Software, die für einen Rechner aus dieser Menge entwickelt wurde, ist gewöhnlich auf allen Rechnern der Rechnerfamilie ablauffähig.

Überblick

Die Systemarchitektur verbindet die physikalischen Elemente der Datenverarbeitungsanlage (Hardware) durch geeignete Anordnung (physisch und logisch) so miteinander, daß das jeweilige Einsatzziel optimal realisiert wird. Das Zusammenspiel der Komponenten wird durch das Betriebssystem überwacht (vgl. Lerneinheit SYSBE). Die Systemarchitektur hat sich mit der technischen Entwicklung der Hardware wesentlich verändert. Zunächst waren Röhren und Transistoren die Grundbausteine für die Systemarchitektur. Später wurden mehrere solcher Elemente als Register oder Schaltwerk als Grundeinheiten verwendet. Heute betrachtet man Prozessoren, Verbindungseinrichtungen (Bus-System) und Speicher bei den Strukturüberlegungen als Gestaltungselemente.

Die Grundaufgabe eines Datenverarbeitungssystems ist die Speicherung, Verarbeitung und Darstellung von Daten. Ein Rechner, der auf jedem Gebiet einsetzbar sein soll, wird **Universalrechner** genannt. Rechner, die für eine spezielle Aufgabe konzipiert werden, heißen **Spezialrechner** ("dedizierte Systeme"). Dementsprechend wird zwischen Universalrechner-Architekturen und Spezialrechner-Architekturen unterschieden. Neben der Erfüllung des vorgegebenen Zwecks wird von einem architektonischen Konzept verlangt, daß es bestimmte Anforderungen optimal erfüllt. Solche Anforderungen können Leistungs-, Sicherheits- und Erweiterbarkeitsanforderungen sein.

Die Beschreibung der Struktur und des funktionalen Verhaltens von Datenverarbeitungssystemen erfordert die Verwendung formaler Hilfsmittel. Formale Hilfsmittel sind Automatentheorie, Schaltwerktheorie, Petrinetze, Berechnungsschemata und Rechner-Entwurfssprachen. Die **Automatentheorie** und die **Schaltwerktheorie** stellen Modelle zur Darstellung der Struktur und des funktionalen Verhaltens von Computern zur Verfügung. **Petrinetze** erlauben die Beschreibung und die Simulation dynamischer Prozesse. **Berechnungsschemata** dienen zur Darstellung von Transformationsprozessen. Mit ihrer Hilfe können die Struktur von Datenverarbeitungssystemen und die darauf auszuführenden Programme beschrieben werden. **Rechner-Entwurfssprachen** erlauben eine Beschreibung von Datenverarbeitungssystemen und deren Einzelelementen mit den Mitteln von Programmiersprachen. Sie sind ein wichtiges Bindeglied zwischen dem Entwurfs- und dem Produktionsprozeß eines Datenverarbeitungssystems.

Entwurf von Computern

Die **Systemarchitektur** entsteht als Teilergebnis beim Entwurf eines Computers. Sie wird, von den angestrebten Zielen ausgehend, **top-down** entwickelt. Auf der obersten Ebene wird zunächst mit Hilfe von Grundbausteinen wie Prozessoren, Speicher, Bus-System usw. die Gesamtarchitektur festgelegt. Diese Gesamtarchitektur wird in den weiteren Entwurfsschritten verfeinert und die verwendeten Komponenten im Detail spezifiziert. Ein Beispiel für den Detailentwurf ist die Festlegung der Prozessor-Architektur. Bei der Entwicklung von Datenverarbeitungssystemen müssen über die funktionalen Leistungsanforderungen hinaus auch wirtschaftliche Randbedingungen beachtet werden. Zu erwähnen sind hier vor allem die Entwicklungsdauer und die Entwicklungskosten. Daher werden Computer rechnerunterstützt entworfen (z.B. Einsatz von CAD-Systemen, Verwendung spezieller Programmiersprachen).

Der Entwurf eines Computers erfolgt in drei Schritten: Entwurf der Systemarchitektur, Logikentwurf und physikalischer Entwurf. Der **Entwurf der Systemarchitektur** umfaßt die Definition der Aufgabenstellung (Benutzeranforderung), die Beschreibung des Einsatzgebietes und der gewünschten Systemeigenschaften sowie der Einbettung in eine bestimmte Umgebung. Ergebnisse sind die Festlegung der Hardware- und der Firmwaremoduln, der modulare Aufbau samt den Verbindungswegen (Datenwege und Steuerlogik), die Partitionierung von Funktionen auf Chips und der Registertransfer-Entwurf. Diese Ergebnisse werden beim **Logikentwurf** durch logische Gatter (Schaltungen) ersetzt und schließlich beim **physikalischen Entwurf** durch Hardware-Bausteine in einer bestimmten Technologie (vgl. Lerneinheit GHARD) realisiert.

Von-Neumann-Architektur

Die Von-Neumann-Architektur ist seit ihrer Einführung durch den Mathematiker John von Neumann im Jahr 1949 noch immer die Grundlage für die meisten heute verwendeten Computer. Abbildung SYSAR-1 zeigt die Grundstruktur eines Rechners mit einer Von-Neumann-Architektur. Sie besteht aus der CPU, der Eingabe- und Ausgabeeinheit, dem Speicher sowie dem Bus, der diese Komponenten verbindet. Angewendet wird das **Prinzip des minimalen Hardware-Aufwands**, da keine der genannten Komponenten zur Erfüllung der Aufgaben eines Datenverarbeitungssystems weggelassen werden kann.

Die Daten und die Programmbefehle sind für die Dauer der Programmausführung im gleichen Speicher (Zentralspeicher) abgelegt (Prinzip des minimalen Speicheraufwands). Bei der Einführung der Von-Neumann-Architektur war das **Prinzip des minimalen Speicheraufwands** ein großer Fortschritt, weil Programme im Speicher dynamisch geändert werden konnten. Dies führt in der Folge meist zu einem Engpaß beim Speicherzugriff, sodaß die Speicher-Schnittstelle bei dieser Architektur scherzhaft auch **Von-Neumann-Flaschenhals** genannt wird. Eine reine Von-Neumann-Architektur war solange gerechtfertigt,

wie die Hardwarekosten hoch waren. Dies trifft jedoch heute nicht mehr zu. Das Prinzip des minimalen Hardware-Aufwands läßt allerdings keine Vervielfachung der Hardware-Betriebsmittel zu, die z.B. eine verstärkte Parallelisierung oder eine höhere Ausfallstoleranz bewirken könnten. Eine Reihe weiterer offener Probleme ergibt sich beim Einsatz höherer Programmiersprachen (vgl. Lerneinheit PROSP):

- Es gibt nur einen gemeinsamen Speicher für Programme und Daten, in höheren Programmiersprachen wird aber zwischen Befehlen und Daten getrennt.
- Höhere Sprachkonstrukte (Blöcke, Prozeduren, Moduln, Prozesse) werden nicht unterstützt.
- Der linearen Speicherorganisation stehen mehrdimensionale Datenstrukturen (z.B. Felder, Listen, Bäume) in höheren Programmiersprachen gegenüber.
- Daten in höheren Programmiersprachen sind oft strukturiert (z.B. Sätze) und haben eine feste Bedeutung; abhängig von dieser Bedeutung sind nur bestimmte Operationen zulässig.

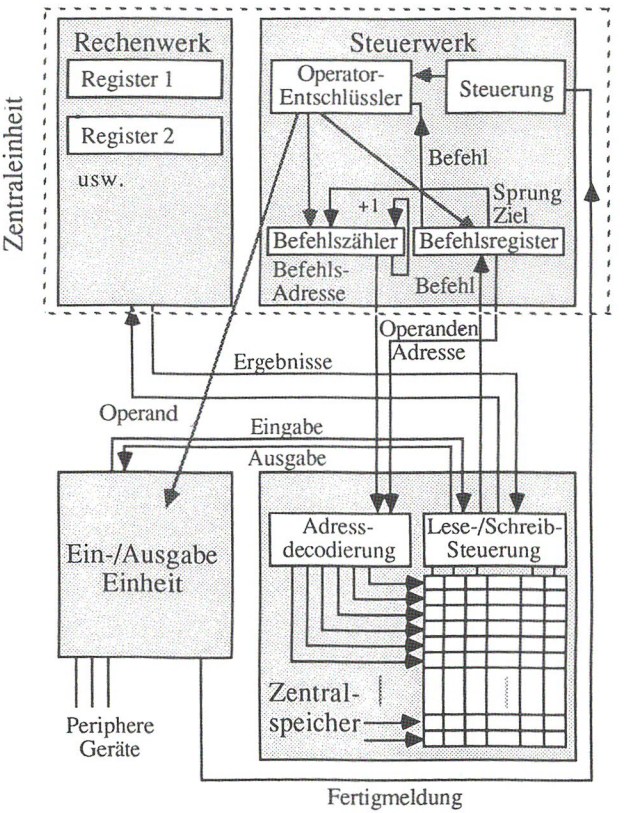


Abb. SYSAR-1: Struktur eines Von-Neumann-Rechners (Quelle: nach Giloi)

Für die Diskrepanz zwischen den Konzepten höherer Programmiersprachen und der Von-Neumann-Architektur wurde der Begriff **semantische Lücke** geprägt. Da die semantische Lücke durch Software geschlossen werden muß, ergeben sich längere Programmlaufzeiten und umfangreichere Programme. Noch schwerwiegender sind die Konsequenzen beim Einsatz objektorientierter und funktionaler Programmiersprachen (vgl. Lerneinheit PROSP). Die Laufzeiten bei der Programmausführung auf Computern mit herkömmlichen Systemarchitekturen sind hier wenig zufriedenstellend.

Befehlsausführung

Die Festlegung der Funktionsweise eines Rechnersystems erfolgt letztlich durch die Vorgabe einer Befehlsfolge, die als **Programm** im Arbeitsspeicher steht und vom Prozessor abgearbeitet wird. Die Art der möglichen Operationen eines Prozessors ist durch den **Befehlssatz** festgelegt. Der Umfang des Befehlssatzes ist daher ein wesentliches Merkmal der Systemarchitektur. Für die Realisierung des Befehlssatzes gibt es unterschiedliche Möglichkeiten. Die einzelnen Befehle können durch Hardware, durch Firmware oder durch Software realisiert werden. Die hardwaremäßige Realisierung stellt dabei die leistungsfähigste, aber gleichzeitig auch die teuerste Lösung dar. Folgende Befehlsklassen können unterschieden werden (vgl. auch Lerneinheit PROSP):

- Befehle zum Transport von Daten,
- Befehle zur arithmetischen und logischen Verknüpfung von Daten,
- Befehle für Programmverzweigungen.

Die Befehlsausführung erfolgt nach einem sich wiederholenden Ablauf, dem **Befehlszyklus**. Das Verstehen des Befehlszyklus ist die Grundvoraussetzung für das Verstehen der Arbeitsweise eines Computers. Bei der Von-Neumann-Architektur wird der Befehlszyklus auch als **Von-Neumann-Zyklus** bezeichnet. Dabei wird iterativ, bis zum Ausschalten des Computers, folgende Schrittfolge ausgeführt:

- Transportieren des nächsten auszuführenden Befehls, auf den der Befehlszähler verweist, ins Befehlsregister;
- Erhöhen des Befehlszählers (Verweis auf den nächsten Befehl);
- Adreßcodierung und Transportieren der Operanden vom Zentralspeicher ins Rechenwerk;
- Ausführen des Befehls;
- Transportieren des Resultats vom Rechenwerk in den Zentralspeicher;
- weiter bei Schritt 1.

Für die Ausführung eines Befehls werden Daten, Adreß- und Steuerinformationen benötigt. Die Übertragung dieser Informationen zwischen den einzelnen Komponenten des Prozessors besorgt das **Bus-System**. Die Bus-Breite, also die Anzahl der binären Zeichen, die gleichzeitig übertragen werden können, ist ein

Leistungsmerkmal für Datenverarbeitungssysteme, das besonders im PC-Bereich zur Klassifikation herangezogen wird. Man spricht in diesem Zusammenhang von 8-Bit-, 16-Bit- und 32-Bit-Rechnern. Die Ausführungszeit für verschiedene Befehle ist gewöhnlich unterschiedlich lang. Die Ausführung eines Befehls zur Multiplikation zweier Zahlen dauert z.B. länger als eine Addition. Die Dauer des Befehlszyklus kann daher immer nur als Durchschnittswert angegeben werden.

Architekturalternativen und Entwicklungstendenzen

Die Entwicklung auf dem Gebiet der Systemarchitekturen ist in den letzten Jahren so vielfältig gewesen, daß es bisher keine brauchbare Klassifizierung gibt. Der bekannteste Ansatz geht von einer Unterscheidung in Befehlsströme und Datenströme aus:

- Das Datenverarbeitungssystem bearbeitet zu einem gegebenen Zeitpunkt einen oder mehr als einen Befehl (SI, single instruction vs. MI, multiple instruction).
- Das Datenverarbeitungssystem bearbeitet zu einem gegebenen Zeitpunkt einen oder mehr als einen Datenwert (SD, single data vs. MD, multiple data).

Die Unterscheidung von SISD-, SIMD-Architekturen usw. ist jedoch wenig aussagefähig. Die Vorschriften, die das Zusammenwirken der verwendeten Hardwarekomponenten regeln, werden damit nicht erfaßt. Man bezeichnet diese Vorschriften als **Operationsprinzip**. Hardwarestruktur und Operationsprinzip definieren gemeinsam die Systemarchitektur.

Bei den Operationsprinzipien unterscheidet man das Von-Neumann-Prinzip, den impliziten Parallelismus und den expliziten Parallelismus.

Beim **Von-Neumann-Prinzip** werden die Befehle durch einen Rechner mit einer Von-Neumann-Architektur sequentiell ausgeführt. Beim **impliziten Parallelismus** nutzt man die Möglichkeiten, die in konventionellen Programmen gegeben sind. Solche parallel ausführbaren Aktivitäten können z.B. konkurrente Prozesse (Taskebene) oder Benutzerprogramme (Jobs) sein. Beim **expliziten Parallelismus** werden Programm- und Datenstrukturen so standardisiert, daß eine Parallelarbeit vorgezeichnet ist (z.B. standardisierte Programmstruktur und arrayförmige Datenstrukturen).

Bezüglich der vorherrschenden **Hardwarestruktur** unterscheidet man Einprozessor-Systeme, homogene und inhomogene Multiprozessor-Systeme sowie Polyprozessor-Systeme.

Zu den **Einprozessor-Systemen** zählen alle Rechner mit der klassischen Von-Neumann-Architektur, sowie sonstige Rechner, die mit nur einem Prozessor ausgestattet sind. Bei **homogenen Multiprozessor-Systemen** sind alle Prozessoren hardwaremäßig gleich, können sich dabei aber in ihrer Rolle im System unterscheiden (symmetrische bzw. asymmetrische Systeme). Die in **inhomogenen**

Multiprozessor-Systemen verwendeten Prozessoren unterscheiden sich in der Hardware. **Polyprozessor-Systeme** sind Systeme, die im Gegensatz zu den bisher beschriebenen Systemen keine zentrale Systemaufsicht haben ("Systeme mit verteilter Kontrolle", vgl. Lerneinheit SERVA).

Ein besonderes Augenmerk bei der Diskussion um Architekturalternativen gilt der Von-Neumann-Architektur und der Überwindung ihrer Schwächen. Viele der Systemarchitekturen, die heute verwendet werden, arbeiten nämlich noch immer mit Systemkomponenten (z.B. Prozessoren) auf der Basis des Von-Neumann-Prinzips. Die Abweichungen vom "klassischen" Von-Neumann-Rechner finden sich meist in der übergeordneten Kontrollstruktur. Stark vereinfacht lassen sich dabei folgende Ansätze unterscheiden:

Parallelisierung bei der Befehlsausführung: Die Realisierung auf der Prozessorebene erfolgt meist mittels **Pipelining** (Fließband-Prinzip). Man versteht darunter die eindimensionale Anordnung von Funktionseinheiten innerhalb des Prozessors, die eine überlappte Befehlsausführung ermöglicht und zu einer Leistungssteigerung führt. In besonderer Weise wird die parallele Befehlsausführung in Vektorrechnern genutzt, bei denen allerdings eine geordnete und gleichartige Datenmenge (z.B. Vektoren, Matrizen) vorausgesetzt wird. Abbildung SYSAR-2 zeigt die überlappte Befehlsausführung beim Pipelining.

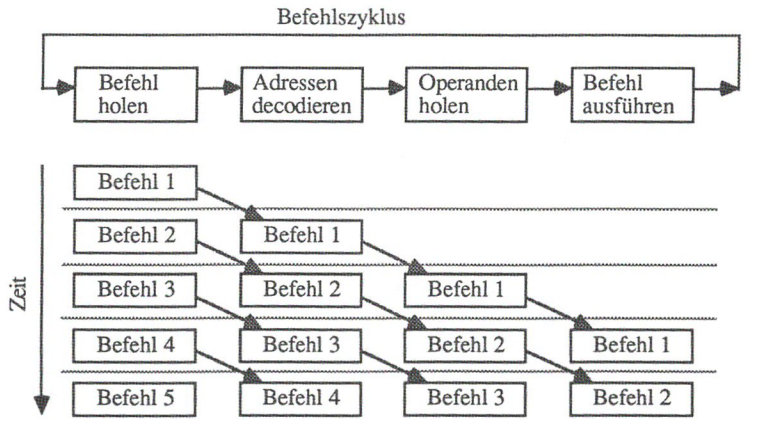


Abb. SYSAR-2: Befehlsausführung beim Pipelining (Quelle: Rechenberg)

Dezentralisierung der Kontrollstruktur: Die Idee, die Abarbeitung der Befehle implizit aus dem Datenfluß abzuleiten, führte zur Datenflußarchitektur. Die Programme werden hier durch gerichtete Graphen beschrieben. Eine wachsende Bedeutung kommt diesen Überlegungen bei den Rechnern der 5. Generation zu (vgl. Lerneinheit WISSE). Von ganz anderen Überlegungen geht man dagegen bei Polyprozessor-Architekturen aus: Der Kontrollfluß wird auf der Prozeßebene durch Verteilung auf entsprechende Prozessoren dezentralisiert. Als Beispiel seien Transputer-Systeme genannt. Neue Erkenntnisse sind auch aus dem

Bereich der neuronalen Netze zu erwarten. Ein Beispiel dafür ist die in Boston entwickelte Connection-Maschine mit derzeit 64000 Prozessoren. Die "Kunst" wird darin bestehen, eine Aufgabe so zu zerlegen, daß diese große Anzahl von Prozessoren gleichmäßig ausgelastet ist.

Verbesserung beim Speicherzugriff: Der Engpaß des Von-Neumann-Flaschenhalses wird durch eine Trennung von Daten und Programm überwunden. Der Vorteil solcher Datentyp-Architekturen liegt in der Verbesserung der Sicherheit und einer Vereinfachung der Befehle. Auch die Speicherzugriffsgeschwindigkeit wird durch den Einsatz von Pufferspeichern (Cache-Speicher) weiter gesteigert. Derzeit können bei herkömmlichen Rechnern nur 25%, bei RISC-Architekturen etwa 60% der Prozessorgeschwindigkeit genutzt werden (der Prozessor könnte also schnellere Speichereinheiten bedienen).

Anpassung der Hardwarestruktur an die Problemstruktur: Bestimmte Aufgaben, die einer hinreichend großen Problemklasse zugeordnet werden können, führen zur Entwicklung von Spezialrechnern. Typische Beispiele sind die Bildverarbeitung (vgl. Lerneinheit BILDV) und die Spracherkennung (vgl. Lerneinheit SPRAC).

Die einfache Architektur des Von-Neumann-Rechners ist trotz der bekannten Schwächen flexibel genug, um noch immer als Basis für alle heutigen Universalrechner zu dienen. Zwei Faktoren sind bei der Einführung neuer Architekturen besonders zu berücksichtigen:

- das Streben nach verbessertem Kosten/Nutzenverhältnis als antreibende Kraft (manche Anwendung wird überhaupt erst bei einer entsprechenden Leistungsfähigkeit des Computers interessant) und
- der Zwang zur Aufrechterhaltung der Software- und oft auch der Hardware-Kompatibilität als hemmende Kraft (Rechnerfamilien).

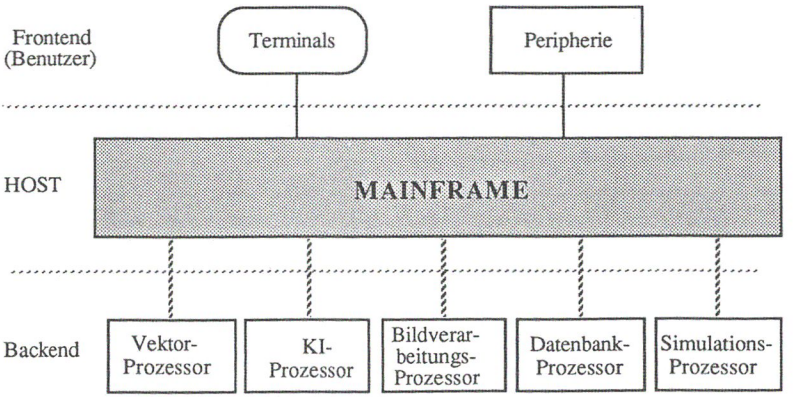


Abb. SYSAR-3: Systemarchitektur für Spezialrechner (Quelle: nach Schwärtzel)

Das Problem der Kompatibilität existiert auch für Rechner mit Spezialarchitekturen. Sie werden daher vielfach, wie in Abbildung SYSAR-3 dargestellt, als Backends für herkömmliche Zentralrechner eingesetzt, sodaß die Benutzeroberfläche eine Von-Neumann-Architektur bleibt. Ein wichtiger Einfluß ergibt sich auch aus der zunehmenden Vernetzung von Rechnern. Dies führt unmittelbar dazu, daß Aufgaben der Systemarchitektur z.T. auf eine andere Ebene verlagert werden (vgl. hierzu die Lerneinheiten SERVA und NETAN).

Demonstrationsbeispiel

Der Befehlssatz eines Computers und seine Realisierung im Befehlszyklus sind ein wesentliches Merkmal der Systemarchitektur. Seit dem Beginn der 80er Jahre sind dabei zwei gegenläufige Tendenzen zu beobachten, die unter den Begriffen CISC (Complex Instruction Set Computer) und RISC (Reduced Instruction Set Computer) zusammengefaßt werden. RISC stellt die neuere Tendenz dar und bezeichnet eine Systemarchitektur auf der Basis stark vereinfachter, auf das Notwendigste reduzierter Befehlssätze. Sie ist die Abkehr von einer bisher ausgeprägten Tendenz zu immer umfangreicheren und in ihrer Komplexität den höheren Programmiersprachen angenäherten Befehlssätzen. Folgende Merkmale kennzeichnen eine RISC-Architektur:

- Ein-Zyklus-Befehle, d.h. alle Befehle haben die gleiche Zykluszeit und das gleiche Befehlsformat und werden unmittelbar von der Hardware ausgeführt;
- Verzicht auf Firmware zur Realisierung der Befehle;
- wenige Befehle und wenige Arten der Speicheradressierung;
- Verlagerung des Aufwands in den Compiler;
- Reduzierung komplexer zugunsten regulärer Hardwarestrukturen.

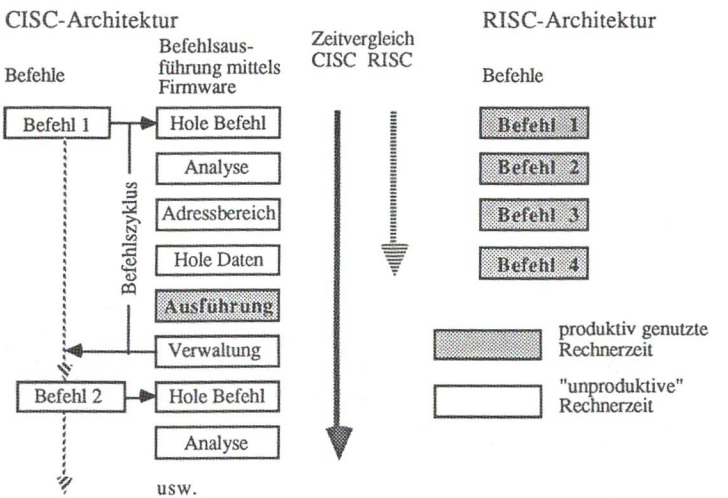


Abb. SYSAR-4: Zeitvergleich RISC/CISC

Während bei herkömmlichen Datenverarbeitungssystemen der Befehlssatz 150 bis 600 elementare Befehle umfaßt, liegt die Zahl bei RISC-Systemen zwischen 20 und 40. Diese Entwicklung stützt sich auf Untersuchungen, bei denen man zu dem Ergebnis kam, daß eine Leistungssteigerung gegenüber herkömmlichen Datenverarbeitungssystemen durch eine Beschränkung auf einen elementaren Grundbefehlssatz erreicht werden kann. Außerdem stellte man fest, daß bei umfangreichen Befehlssätzen ohnehin nur etwa 20% der verfügbaren Befehle häufig genutzt werden. Die übrigen Befehle können im Bedarfsfall durch Kombination von Befehlen des Grundvorrats gebildet werden. Dies führt zwar in manchen Fällen zu etwas "längeren" Programmen, die Programmausführungszeit reduziert sich aber dennoch drastisch. Abbildung SYSAR-4 zeigt den Effekt der Leistungssteigerung bei RISC-Architekturen.

Kontrollfragen

1. Was versteht man unter Systemarchitektur?
2. Geben Sie eine Aufzählung der verschiedenen Architekturprinzipien.
3. Erläutern Sie die Struktur eines Von-Neumann-Rechners.
4. Was versteht man unter dem Von-Neumann-Zyklus?
5. Beschreiben Sie den Unterschied zwischen RISC- und CISC-Architektur.

Quellenliteratur

Bauer, F. und Goos, G.: Informatik. 2 Bände, 4. A., Springer Verlag, Berlin et al. 1991

Bode, A. und Händler, W.: Rechnerarchitektur - Grundlagen und Verfahren. Springer Verlag, Berlin et al. 1980

Goldschlager, L. und Lister, A.: Informatik. 3. A., Verlag Hanser, München/Wien 1990

Rechenberg, P.: Was ist Informatik? Verlag Hanser, München/Wien 1991

Vertiefungsliteratur

Chroust, G.: Mikroprogrammierung und Rechnerentwurf. Oldenbourg Verlag, München/Wien 1989

Giloi, W.: Rechnerarchitektur. Springer Verlag, Berlin et al. 1981

Oberschelp, W. und Vossen, G.: Rechneraufbau und Rechnerstrukturen. 2. A., Oldenbourg Verlag, München/Wien 1987

Schwärtzel, H. (Hrsg.): Informatik in der Praxis - Aspekte ihrer industriellen Nutzanwendung. Springer Verlag, Berlin et al. 1986

SYSBE - Systembetrieb und Betriebssysteme

Lernziele

Sie kennen die verschiedenen Betriebsarten eines Computers und können den Systembetrieb systematisieren. Sie können Meßgrößen für die Leistung beschreiben. Sie kennen die Aufgaben eines Betriebssystems. Sie können die Funktionen eines Betriebssystems beschreiben. Sie können einen Überblick über bekannte Betriebssysteme geben.

Definitionen und Abkürzungen

Auftragsverwaltung (job management) = die Überwachung und Zuteilung freier Betriebsmittel auf einzelne Aufträge (Programme oder Programmteile).

Betriebsmittel (production facility) = die Teile eines Computersystems, die in wechselndem Ausmaß von verschiedenen Prozessen belegt werden und deren beschränkte Verfügbarkeit zu Belegungskonflikten führen kann.

MS-DOS = Abkürzung für Microsoft Disk Operating System; ein von Micro Soft für den Betrieb des IBM PCs entwickeltes Betriebssystem.

MVS = Abkürzung für Multiple Virtual Storage Operating System; ein auf IBM Großrechnern verwendetes Betriebssystem.

Prioritätensteuerung (priority processing) = die Bestimmung der Reihenfolge der Aufträge bei der Verarbeitung.

real (real) = aus der Sicht eines bestimmten Objekts physisch vorhanden; im Unterschied dazu: virtuell.

Spool = Abkürzung für Simultaneous Peripheral Operations Online; der logische Bereich eines Speichers, in dem alle Eingaben und Ausgaben zwischengespeichert werden.

Timer (timer) = der Taktgeber (clock), der periodisch elektrische Taktsignale (Quarzimpulsgeber) abgibt und das Startsignal für den Ablauf sämtlicher Operationen ist.

UNIX = ein weitgehend hardwareunabhängiges Betriebssystem für den Einsatz auf PCs, Workstations und Minicomputern.

Unterbrechung (interrupt) = ein durch ein Signal ausgelöster Mechanismus, der ein laufendes Programm unterbricht und ein anderes Programm startet.

virtuell (virtual) = aus der Sicht eines bestimmten Objekts physisch nicht vorhanden; im Unterschied dazu: real.

Wiederanlauf (restart) = das Starten eines Programms nach einer Programmunterbrechung oder nach einem Programmabbruch.

Überblick

Die Leistungsfähigkeit eines Computers hängt nicht nur von technischen Eigenschaften der Hardware und von der Systemarchitektur ab, sondern in einem hohen Ausmaß auch von der verfügbaren Systemsoftware. Systemsoftware ist eine zusammenfassende Bezeichnung für das Betriebssystem sowie alle anwendungsneutralen Dienstprogramme, die zur Abwicklung häufig vorkommender Aufgaben eingesetzt werden. Dazu zählen u.a. Editoren, Übersetzer, Netzsoftware und Datenbanksysteme (vgl. Lerneinheit GSOFT).

Die möglichen Betriebsarten eines Computers werden primär durch das Betriebssystem bestimmt. Eine Systematik der Betriebsarten kann nach mehreren Gesichtspunkten gebildet werden. Bezüglich der Art der Abarbeitung der Aufträge wird zwischen **Einprogrammbetrieb** und **Mehrprogrammbetrieb** unterschieden. Beeinflußt der Benutzer den Steuerungszusammenhang zwischen einem Datenendgerät und dem Computer, so unterscheidet man zwischen **Stapelbetrieb** und **Dialogbetrieb.** Spezielle Formen des Dialogbetriebs sind der **Teilhaberbetrieb**, der **Teilnehmerbetrieb** und der **Time-Sharing-Betrieb.** Nach dem Steuerungszusammenhang zwischen der Ein-/Ausgabe und dem Datenverarbeitungsprozeß wird zwischen **online** und **offline** bzw. zwischen **Stapelbetrieb** und **Echtzeitbetrieb** unterschieden. Weiter kann zwischen **Einzelplatzsystem** und **Mehrplatzsystem** sowie nach organisatorischen Gesichtspunkten zwischen **offenem Betrieb** und **geschlossenem Betrieb** differenziert werden.

Manche der aufgezählten Betriebsarten schließen sich gegenseitig aus, andere ergänzen sich. Sie werden nachfolgend näher erläutert. Der Betrieb eines Computersystems wird, entsprechend den Benutzeranforderungen ("Zielen"), unterschiedlich gestaltet. Die Realisierung einer bestimmten Betriebsart setzt voraus, daß das benutzte Betriebssystem diese Betriebsart unterstützt. Die Betriebssysteme sind zwar je nach Art des Computers (vgl. Lerneinheit GHARD) und Hardware-Herstellers unterschiedlich, dennoch läßt sich eine gleichartige Grundstruktur erkennen. Ihre Darstellung erfolgt im Anschluß an die Erläuterung der Betriebsarten.

Betriebsarten von Computersystemen

Bezüglich der Art der Abarbeitung der Aufträge wird zwischen Einprogrammbetrieb und Mehrprogrammbetrieb unterschieden.

Beim **Einprogrammbetrieb** (engl.: single tasking mode) wird ein Auftrag vollständig abgearbeitet, bevor mit dem nachfolgenden Auftrag begonnen wird. Da

von der Zentraleinheit jeweils nur ein Programm ausgeführt wird, werden ihm alle Betriebsmittel des Datenverarbeitungssystems zugeteilt. Dies führt zu einer schlechten Ausnutzung der Betriebsmittel, da beispielsweise vom Programm zum Zeitpunkt der Eingabe nur der Eingabeprozessor benötigt wird, alle anderen Prozessoren und Funktionseinheiten - obwohl sie zum gegebenen Zeitpunkt vom Programm nicht in Anspruch genommen werden, aber blockiert sind. Der Einprogrammbetrieb ist häufig bei dezentralen Arbeitsplatzsystemen (z.B. bei PCs) anzutreffen. Neuere Entwicklungen unterstützen aber auch bei PCs bereits den Mehrprogrammbetrieb.

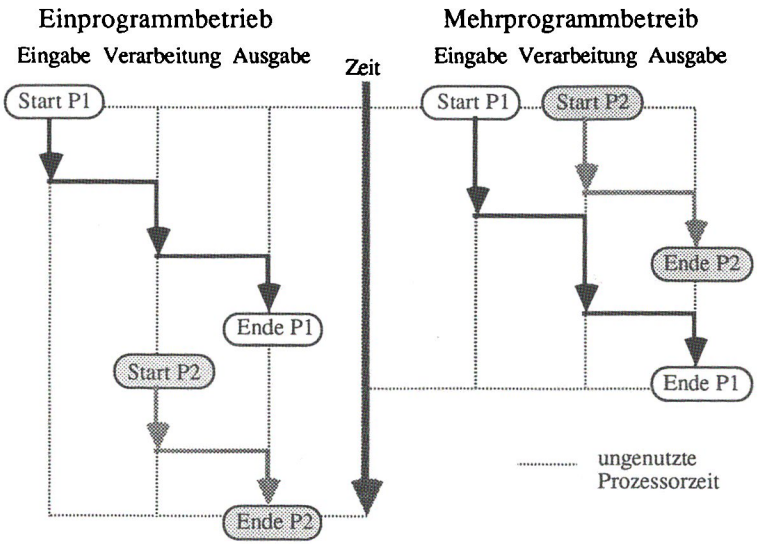


Abb. SYSBE-1: Einprogramm- und Mehrprogrammbetrieb

Der **Mehrprogrammbetrieb** (engl.: multiprogramming oder multitasking mode) ist dadurch gekennzeichnet, daß mehrere Programme gleichzeitig ausgeführt werden können. Dadurch werden die Betriebsmittel des Datenverarbeitungssystems besser als beim Einprogrammbetrieb genutzt, da die von einem Programm nicht benötigten Betriebsmittel für andere Programme freigegeben werden. Da ablaufende Programme unter Umständen langfristig Betriebsmittel belegen, auf die andere Programme während dieser Zeit nicht zugreifen können, benötigt der Mehrprogrammbetrieb eine **Prioritätensteuerung**. Die Prioritätensteuerung regelt die Zuteilung der Betriebsmittel an die Programme. Häufig angewandte Prinzipien, nach denen diese Prioritätensteuerung erfolgt, sind:

- Die Priorität wird extern vom Benutzer festgelegt.
- Die Priorität wird aufgrund der bereits belegten Betriebsmittel nach einem Zuteilungsschlüssel errechnet.
- Programme mit kurzer Laufzeit haben eine höhere Priorität als Programme mit langer Laufzeit.