

Heidelberger Taschenbücher

Sammlung Informatik

Giloi

Rechnerarchitektur



Springer-Verlag

Berlin Heidelberg New York

W. K. Giloi

Rechnerarchitektur

Mit 133 Abbildungen

Springer-Verlag
Berlin · Heidelberg · New York 1981

Wolfgang K. Giloi

Professor of Computer Science, University of Minnesota
und

Professor für Technische Informatik,
Technische Universität Berlin

CIP-Kurztitelaufnahme der Deutschen Bibliothek.

Giloi, Wolfgang:

Rechnerarchitektur/W. K. Giloi. —

Berlin, Heidelberg, New York: Springer. 1981. —

(Heidelberger Taschenbücher; Bd. 208: Sammlung Informatik)

ISBN 3-540-10352-X Springer-Verlag Berlin Heidelberg New York

ISBN 0-387-10352-X Springer-Verlag New York Heidelberg Berlin

Das Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, der Entnahme von Abbildungen, der Funksendung, der Wiedergabe auf photomechanischem oder ähnlichem Wege und der Speicherung in Datenverarbeitungsanlagen bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten.

Bei Vervielfältigungen für gewerbliche Zwecke ist gemäß § 54 UrhG eine Vergütung an den Verlag zu zahlen, deren Höhe mit dem Verlag zu vereinbaren ist.

© Springer-Verlag Berlin, Heidelberg 1981

Printed in GDR.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, daß solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Bindearbeiten: Konrad Triltsch, Würzburg
2145/3020—543210

Heidelberger Taschenbücher Band 208

Sammlung Informatik

Herausgegeben von F. L. Bauer, G. Goos und M. Paul



*Dieses Buch
ist dem großen Rechnerpionier
Konrad Zuse gewidmet*

Vorwort

The types of architectures are established not by architects but by society, according to the needs of the different institutions. Society sets the goals and assigns to the architect the job of finding the means of achieving them.

Encyclopaedia Britannica, Macropaedia 1, 1089

In der bisherigen Entwicklung elektronischer Rechenanlagen konnte etwa alle 5—6 Jahre eine Leistungssteigerung um eine Größenordnung festgestellt werden. Dieser Fortschritt war bisher jedoch fast ausschließlich ein Fortschritt der Hardware-Technologie. Die Entwicklung der Halbleiter-Technologie in Richtung auf hohe Integrationsdichte, hohe Arbeitsgeschwindigkeit und geringe Kosten verlief in den letzten Jahren besonders stürmisch. An die Stelle der Hardware als dominierendem Kostenfaktor einer Rechenanlage ist die Software getreten.

Die geringen Kosten hochintegrierter, standardisierter Bausteine erlauben heute auch bei kleinen bis mittleren Rechnern einen Hardware-Aufwand, den man vor wenigen Jahren nur bei einem Großrechner als angemessen betrachtet hätte. Selbst Mikroprozessoren haben inzwischen eine Wortlänge von 16 oder 32 bit und können mit einem Arbeitsspeicher von mehreren Megabyte ausgestattet werden; die Größe des Arbeitsspeichers ist damit kein Kriterium zur Unterscheidung von „kleinen“ und „großen“ Rechnern. Der große, kostengünstige Speicher ist aber nur ein erster Schritt zur Nutzung der durch die Großintegration eröffneten Möglichkeiten, der als nächster Schritt eine weitere Verbesserung des Preis/Leistungs-Verhältnisses folgen muß. Da die Technologie großintegrierter Rechnerkomponenten verhältnismäßig „langsam“ ist, kann eine Leistungssteigerung nicht mehr, wie in der Vergangenheit, durch einfache Erhöhung der Arbeitsgeschwindigkeit von Rechenwerk und Speicher erzielt werden, sondern nur durch entsprechende Vervielfachung dieser zwar billigen, aber auch verhältnismäßig langsamen Hardware-Betriebsmittel eines Rechners. Damit kommt man erstmalig auch der Verwirklichung von Zielvorstellungen nahe, die man seit etwa zwei Jahrzehnten bereits als äußerst wünschenswert erkannt hat, deren Realisierung bisher aber aus technischen oder wirtschaftlichen Gründen nicht oder nur in unbefriedigendem Maße möglich war: der modularen Erweiterbarkeit und der Ausfalltoleranz einer Anlage.

Diese Entwicklung macht eine Abkehr von der jahrzehntelang dominierenden Architektur der von Neumann-Maschine und eine Suche nach neuen, angemesseneren Architekturformen unumgänglich. Aber auch um die neueren Erkenntnisse der Softwaretechnik hinsichtlich Modularität,

Zuverlässigkeit und Wartbarkeit großer Softwaresysteme, hinsichtlich Datenintegrität, Anpassungsfähigkeit der Betriebssysteme, Realisierung der Konzepte abstrakter Datentypen und Datenmodelle, Realisierung transaktionsorientierter Sprachen und anderem mehr zu nutzen und eine solche Software besser hardwaremäßig zu unterstützen (Schließung der „semantischen Lücke“ zwischen Software und Hardware), sind neue Wege in der Rechnerarchitektur zu beschreiten.

Dieses Buch handelt von der *Rechnerarchitektur*. Betrachtungen über *Rechnerorganisation* — das ist der Aufbau und die Arbeitsweise der in der Architektur verwandten „Hardware-Betriebsmittel“ — werden in dem Maße angestellt, wie diese organisatorischen Maßnahmen bestimmte Prinzipien der Rechnerarchitektur unterstützen (z. B. Speichersegmentierung, Fließbandverarbeitung, u. ä.) und zum Verständnis gewisser Eigenheiten der Strukturen von Rechnerarchitekturen und deren Leistungsfähigkeit betrachtet werden müssen. Dadurch werden gewisse Exkursionen von der Architekturebene in die Ebene der Organisation der Komponenten einer Architektur unvermeidbar sein. Als ergänzende Schrift, die vorwiegend die Organisationsebene behandelt (mit gewissen Exkursionen in die Architekturebene) sei das vorzügliche Buch von Jessen [JES 75] empfohlen.

Aber auch dort, wo von uns organisatorische Fragen behandelt werden, wird der Leser keine Diagramme von Schaltnetzen und Schaltwerken vorfinden. Für den Rechnerarchitekten umfaßt der Begriff der Hardware komplexe Komponenten mit einem bestimmten funktionellen Verhalten, die er als Betriebsmittel beim architektonischen Entwurf eines Rechners einplanen kann, und die in steigendem Maße als fertige, großintegrierte Bausteine erhältlich sein werden. Schaltungen gehören damit in das Arbeitsgebiet des Entwurfsers solcher Bausteine.

In der Vergangenheit wurde die Suche nach innovativen Rechnerarchitekturen zu sehr oft mit reinen Strukturüberlegungen gleichgesetzt, das heißt mit der Suche nach Verbindungsstrukturen von Prozessoren, Speichern und anderen Betriebsmitteln, die besondere Vorteile erhoffen ließen. Erst in den letzten Jahren hat man erkannt, daß solche reinen Strukturüberlegungen nicht weit führen; daß es vielmehr darauf ankommt, sich bereits vor Eintritt in Strukturüberlegungen Klarheit über das verschafft zu haben, was wir unter dem Begriff „Operationsprinzip“ subsummieren, nämlich die Informationsstrukturen und Kontrollstrukturen der gesuchten Rechnerarchitektur.

Wir sehen damit den Rechnerentwurf der Zukunft vorwiegend als einen *top-down*-Entwurfsprozeß: Ausgehend von den Zielen, die mit einer Rechnerarchitektur erreicht werden sollen, legt man zunächst ein Operationsprinzip fest, welches am besten die gestellten Ziele zu verwirklichen verspricht, und das gewählte Operationsprinzip impliziert dann in gewissem Grade die Systemstruktur. In jedem dieser Entwurfsschritte wird aber ein gewisser Freiraum für Entwurfsentscheidungen bestehen bleiben.

Zu der am Anfang stehenden Festlegung der Entwurfsziele (*requirement engineering*) gehören auch bereits detaillierte Überlegungen darüber, wie und mit welchen Sprachen das System später programmiert werden soll (gerade gegen dieses Postulat ist in der Vergangenheit bei den Bemühungen um innovative Rechnerarchitekturen oft — und meist mit sehr nachteiligen Folgen — verstoßen worden). Dabei werden innovative Operationsprinzipien zwangsläufig zu neuartigen Sprachkonstruktionen führen, wie dies bei den Feldrechnern oder den Datenfluß-Architekturen geschehen ist und für die verteilten Polyprozessor-Systeme zu erwarten steht. So wie in der Vergangenheit die von Neumann-Architektur die Entwicklung der „von Neumann-Sprachen“ nach sich zog, so wird vielfach auch weiterhin die Sprachentwicklung der Architekturentwicklung nachfolgen. Innovative Architekturentwürfe, die in Zukunft in stärkerem Maße neben anderen auch das Entwurfsziel der Verringerung der semantischen Lücke zwischen Hardware und Anwendungssoftware verfolgen werden, werden aber auch mehr als bisher eine umgekehrte Entwicklung zeitigen (die bisher nur bei den „Keller-Maschinen“ zu verzeichnen war), nämlich, daß die Architektur der Sprache folgt. Dies soll erklären, warum die Betrachtung von Datentypen und Datenstrukturen bis hin zu den Strukturen höherer Programmiersprachen einen so breiten Raum in einem Buch über Rechnerarchitektur einnimmt, wie dies bei dem vorliegenden Text der Fall ist.

Wie bei jedem neuen Gebiet, so ist auch die bisherige Entwicklung in der Rechnerarchitektur zunächst durch das Vorherrschen von *ad hoc*-Lösungen gekennzeichnet. Systematisierungen folgen in der Regel dem Prozeß der Entwicklung von *ad hoc*-Lösungen erst dann, wenn ein gewisser Erfahrungsschatz erarbeitet und ein gewisser Reifegrad erreicht ist. Der vorliegende Text ist der Versuch einer solchen Systematisierung.

Der Text ist entstanden als das Ergebnis mehrjähriger Vorlesungs- und Forschungstätigkeit des Autors an der University of Minnesota und der Technischen Universität Berlin. Er hat zunächst den Charakter eines Lehrbuchs; darüber hinaus ist er aber auch als allgemeine Monographie angelegt. Insbesondere glauben wir, daß dieses Buch auch dem in der Praxis stehenden Ingenieur, Informatiker oder Mathematiker, der mit der Entwicklung, der Planung oder dem Einsatz von Rechenanlagen befaßt ist, helfen kann, die Leistungsfähigkeit und Grenzen von innovativen Rechner-systemen beurteilen und Entwicklungstendenzen erkennen zu lernen.

Der Verfasser ist Professor Dr. G. Goos und Dr. G. Hommel für wertvolle Kritik und Hinweise zum Dank verpflichtet. Seinen Mitarbeitern, P. Behr, Dr. H. K. Berg und R. Güth verdankt er anregende Diskussionen, und sein besonderer Dank für viele ausführliche Gespräche über zukünftige Rechnerstrukturen richtet sich an C. A. (Sandy) Wilson.

W. Giloi, H. Liebig

Logischer Entwurf digitaler Systeme

Hochschultext

2., überarbeitete Auflage. 1980. 183 Abbildungen.

XII, 328 Seiten

DM 48,-

ISBN 3-540-10091-1

Inhaltsübersicht: Aussagenkalkül und Boolesche Algebra. – Boolesche Algorithmen. – Schaltnetze. – Schaltketten. – Asynchron-Schaltwerke. – Synchron-Schaltwerke und Mikroprozessoren. – Literaturverzeichnis. – Sachverzeichnis.

Aus den Besprechungen: „Das ausgezeichnete Buch von W. Giloi und H. Liebig wird voraussichtlich längere Zeit als Standardwerk für die Informatikausbildung auf dem Gebiet logischer Entwurf digitaler Systeme zu empfehlen sein. Es ist beachtenswert mit welcher Konsequenz die Kapitel einheitlich und folgerichtig aufgebaut sind. An der Darstellung ist zu erkennen, daß das Buch von erfahrenen Hochschullehrern geschrieben wurde. Das Bemühen um Anschaulichkeit und Verständlichkeit sowie die zahlreichen Beispiele, die der Praxis entnommen sind, unterstreichen das.“

Zeitschrift für angewandte Mathematik und Mechanik

H. Liebig

Logischer Entwurf digitaler Systeme

Beispiele und Übungen

Hochschultext

1975. 92 Abbildungen. VIII, 175 Seiten

DM 28,-

ISBN 3-540-06912-7



Springer-Verlag
Berlin
Heidelberg
New York

Das Buch umfaßt 72 Übungsaufgaben und vollständig durchgerechnete Lösungen mit dem Charakter von Entwurfsbeispielen für die systematische Entwicklung von Schaltnetzen und Schaltwerken. Die in dem zugehörigen Hauptwerk "Logischer Entwurf digitaler Systeme" von GILIO/LIEBIG behandelten mathematischen Methoden werden voll angewendet und zum Teil ergänzt. Der aktuellen technischen Praxis wird durch die Auswahl der Aufgaben und der ergänzenden Themen Rechnung getragen.

Studenten wie Ingenieuren wird die gegenüberstellende Darstellung von Formeln und Schaltbildern willkommen sein. Das Buch ist für beide Gruppen gleichermaßen geeignet, für Studenten zum Üben und Entwerfen und für Ingenieure zum Nachschlagen und Entwickeln.

H. Liebig

Rechner- organisation

Hardware und Software digitaler Rechner

Hochschultext

Unter Mitarbeit von T. Flik, K. Horn
1976. 102 Abbildungen. X, 282 Seiten
DM 58,-
ISBN 3-540-07596-8

Inhaltsübersicht: Klassische Rechnerorganisation. – Rechnerhardware. – Rechnersoftware. – Assemblerprogrammierung und Prozessororganisation. – Speicherorganisation. – Ein- und Ausgabeorganisation.

Aus den Besprechungen:

Nach einer Einführung in die klassische Rechnerorganisation, die Hard- und Software von Rechenanlagen werden die Organisationen der Hauptelemente digitaler Rechner – Prozessor, Speicher und Ein-/Ausgabegeräte in verschiedenen Variationen – besprochen... Die prinzipiell möglichen Realisierungen der Einzelorganisationen werden sehr ausführlich behandelt und verglichen, eine große Anzahl detaillierter Beispiele erleichtert das Verständnis...

Gerade durch die große Anzahl praktischer Beispiele und deren Ausführung in einer durch Hardwareentscheidungen ausgezeichnet motivierten Assemblersprache eignet sich das Buch vorzüglich als Nachschlagewerk für Praktiker, aber auch als Einführung in die Probleme der Rechnerorganisation im Rahmen von Hoch- und Fachhochschulen.“



Springer-Verlag
Berlin
Heidelberg
New York

VDI-Zeitschrift

Inhaltsverzeichnis

1 Einleitung	1
1.1 Was ist Rechnerarchitektur?	1
1.2 Zum Stand der Technik bei den Hardware-Komponenten.	3
1.3 Motivation für innovative Rechnerarchitekturen.	10
1.4 Das hierarchische Schichtenmodell eines Rechnersystems	11
1.5 Die Konstituenten einer Rechnerarchitektur	20
1.5.1 Definitionen	20
1.5.2 Abstrakte Datentypen	21
1.6 Taxonomie von Rechnerarchitekturen	24
1.6.1 Allgemeine Bemerkungen	24
1.6.2 Operationsprinzipien	27
1.6.3 Strukturen von Rechnerarchitekturen	28
1.6.4 Die OS-Matrix.	31
2 Die klassische von Neumann-Maschine	33
2.1 Die Struktur der klassischen von Neumann-Maschine	33
2.2 Das Operationsprinzip der von Neumann-Maschine	36
2.3 Berechnung und Ablaufkontrolle durch die von Neumann-Maschine	38
2.4 Programmstrukturen.	46
2.5 Das Petrinetz-Modell für konkurrente Prozesse	54
2.6 Grundlagen der Speicherorganisation im von Neumann-Rechner.	56
2.6.1 Der Working Set eines von Neumann-Programms	57
2.6.2 Schutzmechanismen	59
3 Strukturen von Einprozessor-Systemen	63
3.1 Die zentrale Recheneinheit (CPU)	63
3.1.1 Die minimale von Neumann-Maschine	63
3.1.2 Mehrregister-Maschinen.	65
3.1.3 Anwendung des Pipeline-Prinzips im Prozessor.	67
3.2 Speicherorganisation und Speicherverwaltung	71

3.2.1	Seitenadressierung	71
3.2.2	Abbildung eines kleineren logischen Adreßraums auf einen größeren physikalischen Adreßraum	72
3.2.3	Abbildung eines größeren logischen Adreßraums auf einen kleineren physikalischen Adreßraum	75
3.2.4	Cache-Speicher	78
3.2.5	Speicher-Segmentierung	84
3.3	Capability-Adressierung	86
3.3.1	Capability-Adressierung und Speichersegmentierung	88
3.3.2	Referenz über Capability-Register	89
3.3.3	Einführung eindeutiger Objekt-Identifikatoren	90
3.4	Bus-Organisation	92
3.5	Ein/Ausgabe-Organisation	95
3.6	Mikroprogrammierung und „vertikale Verlagerung“	96
4	Konzepte der Parallelarbeit	101
4.1	Der Parallelismus und seine Nutzung	101
4.2	Die Ebenen der Parallelarbeit	105
4.3	Erkennung von implizitem Parallelismus	108
4.3.1	Automatische Erkennung von Parallelismus auf der Operations-Ebene	109
4.3.2	Automatische Erkennung von Parallelismus auf der Anweisungs-Ebene	109
4.4	Programmierung von Parallelarbeit	113
4.4.1	Anweisungs-Ebene	114
4.4.2	Prozeß-Ebene: Explizite Synchronisation	116
4.4.3	Prozeß-Ebene: Implizite Synchronisation	118
4.4.4	Der <i>Tasking</i> -Mechanismus von Ada (GREEN).	120
4.4.5	Die Multiprogramming-Ebene	124
4.5	Datenstrukturen und Parallelarbeit	125
4.6	Arrays oder Pipelines von Bearbeitungselementen und ihr Wirkungsgrad.	128
4.7	Vergleich zwischen Pipeline und Array von Bearbeitungselementen	134
4.8	Verbindung der PEs zu einem Prozessor-Array	137
4.9	Wirkungsgrad von Multiprozessor-Systemen	141
5	Innovative Rechnerstrukturen mit von Neumann-Operationsprinzip	148
5.1	Zellulare Systeme: Die Holland-Maschine	148
5.1.1	Vorbemerkungen über zellulare Systeme.	148
5.1.2	Die Holland-Maschine	148
5.1.3	Comfords Variante der Holland-Maschine	156
5.1.4	Kritik der Holland-Maschine	157

5.2 Arrays von Bearbeitungselementen; ILLIAC IV	159
5.2.1 Grundsätzliches	159
5.2.2 Das Grundkonzept der ILLIAC IV als Beispiel eines Prozessor-Arrays	160
5.2.3 Die Organisation des ILLIAC IV	162
5.2.4 Programmierung der ILLIAC IV	164
5.2.5 Kritik der ILLIAC IV	169
5.3 System mit einer Vielzahl von Datenprozessoren: CDC 6600	171
6 Feldverarbeitungs-orientierte Rechnerarchitekturen	178
6.1 Vektor-Maschinen	178
6.1.1 Leistungsfähigkeit der Pipeline-Verarbeitung	178
6.1.2 Operationsprinzip der Vektor-Maschinen	182
6.1.3 Struktur von STAR 100 und CYBER 200	185
6.1.4 Struktur der CRAY-1	192
6.2 Verfeinerung des Konzepts der Bearbeitung geordneter Datenmengen; STARLET	194
6.2.1 Strukturbildungsfunktionen und Speicherzugriffsfunktion	194
6.2.2 Die Informationsdarstellung in der STARLET-Maschine	197
6.2.3 Der Struktur-Datentyp VECTOR der STARLET-Maschine	204
6.3 Assoziative Rechner	206
6.3.1 Assoziativspeicher	206
6.3.2 STARAN, ein assoziativer Feldrechner	208
7 Selbst-beschreibende Information und Datentypen-Architekturen	212
7.1 Vorteile einer Typenkenennung	212
7.1.1 Typenabhängige Befehlsausführung	213
7.1.2 Erhöhung der Fehlersicherheit	213
7.1.3 Aufbau von Struktur-Datentypen	215
7.1.4 Unterstützung der Mechanismen höherer Programmiersprachen	216
7.1.5 Unterstützung des Betriebssystems	217
7.2 Hierarchische Typenkenennung	220
7.3 Anwendung des DRAMA-Prinzips: Datentypen-Architekturen	224
7.3.1 Allgemeines über Datentypen-Architekturen	224
7.3.2 Hierarchischer Aufbau von Datentypen aus einem Grundbaustein	228
7.4 Tagged Architectures und DRAMA-Maschinen	233
8 Spracharchitekturen	239
8.1 Prinzipien der Schließung der „semantischen Lücke“ zwischen HLL und ML	239

8.2 Keller-Maschinen	244
8.2.1 Der Maschinen-Datentyp KELLER	244
8.2.2 Anwendung des Kellers zur Berechnung von Ausdrücken und Anweisungen	246
8.2.3 Keller für den Unterprogramm-Aufruf	249
8.2.4 Keller zur Darstellung der Blockstruktur einer HLL	250
8.2.5 Beispiel einer Kellermaschine: P-Maschine.	253
8.3 Unmittelbare Ausführung des HLL-Programms: Die SYMBOL-Maschine	259
8.3.1 Allgemeine Kennzeichnung des SYMBOL-Systems	259
8.3.2 Sprache, Datentypen und Informationseinheiten der SYMBOL-Maschine	262
8.3.3 Speicherverwaltung im SYMBOL-Rechner.	263
8.4 Direkt ausführende Architektur mit Darstellungstranformation: Burroughs B6700	265
8.4.1 Allgemeine Kennzeichnung der B6700.	265
8.4.2 Informationseinheiten und Keller-Organisation der B6700	267
8.4.3 Programmausführung in der B6700	270
8.5 Sprachorientierte Rechnerarchitekturen: APL-Maschinen	271
8.5.1 Eigenschaften von APL	271
8.5.2 Direkte Ausführung von APL	272
8.5.3 Beispiel einer APL-Maschine	274
8.6 DRAMA-Maschinen als sprachunterstützende Architektur	278
9 Datenfluß-Multiprozessorarchitekturen	279
9.1 Einleitung	279
9.2 Der „Prozessor-Baum“ für die Berechnung arithmetischer Ausdrücke	282
9.3 Reduktionsmaschinen (applicative machines)	286
9.4 Datenfluß-Architektur mit fester Zuordnung der Operationseinheiten	288
9.5 Elementares Datenfluß-Schema und elementarer Datenfluß-Prozessor	292
9.6 Verallgemeinertes Datenfluß-Schema und die Datenfluß-Basismaschine	297
9.7 Das LAU-System	300
9.8 Zusammenfassende Betrachtungen über Datenfluß-Architekturen	306
10 Allzweck-Multiprozessorarchitekturen	308
10.1 Multiprozessor-Systeme mit zentralisierter Kontrolle	308
10.1.1 Definitionen	308
10.1.2 Asymmetrische Multiprozessor-Systeme	313

10.1.3	Symmetrische Multiprozessor-Systeme	314
10.1.4	Zentrale Betriebssystem-Aufgaben in Multiprozessor-Systemen	315
10.2	Strukturen von Multiprozessor-Systemen mit zentralisierter Kontrolle.	320
10.2.1	Strukturen von Verbindungseinrichtungen	320
10.2.2	Realisierungen von Verbindungseinrichtungen.	325
10.3	Beispiele für Multiprozessor-Systeme.	335
10.3.1	C.mmp — ein symmetrisches Multiprozessor-System.	335
10.3.2	Cm* — ein symmetrisches Multiprozessor-System	337
10.3.3	Ein Multiprozessor-System mit Hardware-Überwacher	343
10.4	Verteilte Systeme	347
10.4.1	Die drei Klassen verteilter Systeme	347
10.4.2	Verteilte Polyprozessor-Systeme: Allgemeine Betrachtungen	352
10.4.3	Übertragungs- und Zugriffsprotokolle	353
10.4.4	Inter-Prozeß-Kommunikations-Protokolle	358
Literatur		367
Sachverzeichnis		377

1 Einleitung

1.1 Was ist Rechnerarchitektur?

Rechnerarchitektur ist eine Übersetzung des englischen Begriffs *computer architecture*. *Architecture* bedeutet im Englischen *Baukunst* oder *Baustil*; aber auch das einzelne Produkt der Baukunst wird als *architecture* bezeichnet. Die letztgenannte Bedeutung hat das deutsche Wort Architektur nicht. Wenn wir von Rechnerarchitektur sprechen, meinen wir jedoch den vollen Bedeutungsinhalt von *computer architecture*.

Rechnerarchitektur wie Architektur im allgemeinen dient einem Zweck. Wir zitieren hier die Encyclopaedia Britannica („Art of Architecture“, Macropaedia 1,1089): “The types of architecture are established not by architects but by society, according to the needs of the different institutions. Society sets the goals and assigns to the architect the job of finding the means of achieving them.”

Entsprechend ihrem Zweck unterscheidet man in der Baukunst Typen von Architekturen wie die Architektur des Wohnungsbaus, des Fabrikbaus, des Sakralbaus, des Verwaltungsbaus, etc. Spezifischer noch kann man von Sportstätten-Architektur, Flughafen-Architektur, Krankenhaus-Architektur, usw. reden.

Die Grundaufgabe einer Rechenanlage ist die Sammlung, Speicherung, Verarbeitung und Darstellung von Information. Diese Grundaufgabe kann auf den verschiedensten Anwendungsgebieten zu erfüllen sein. Eine Rechenanlage, die auf jedem beliebigen Gebiet einsetzbar sein soll, soll Universalrechner genannt werden. Daneben gibt es Anlagen, die für eine spezielle Aufgabe konzipiert wurden und die deshalb *Spezialrechner* heißen. Dementsprechend ist zwischen Universalrechner-Architekturen und Spezialrechner-Architekturen zu unterscheiden.

Der Computer-Architekt hat seine Aufgabe aber noch nicht damit gelöst, daß er ein architektonisches Konzept vorlegt, welches den vorgegebenen Zweck erfüllt. Es wird vielmehr von ihm erwartet, daß dieses Konzept gewisse Forderungen möglichst optimal erfüllt. Solche Forderungen können unter verschiedenen Aspekten gestellt werden, nämlich

- dem Leistungs-Aspekt,
- dem Ausfalltoleranz-Aspekt,
- dem Erweiterbarkeits-Aspekt,
- dem pragmatischen Aspekt.

Leistungs-Aspekt. Bei manchen Spezialrechnern für Echtzeit-Aufgaben kann mitunter die Forderung nach maximaler absoluter Leistung in Vordergrund stehen, hinter den Kostenerwägungen zurücktreten müssen. Im allgemeinen wird man aber nicht die absolute Leistungs- sondern die Kosteneffektivität, d. h. die Leistung pro Kosteneinheit, zu maximieren trachten (wie immer diese meßbar sein mag). Es ist dabei zu beachten, daß zwei verschiedene Systeme nur dann in ihrer Leistung vergleichbar sind, wenn auch ihre Anwendungsbreite etwa gleich ist. Bei Allzweckrechnern kann es daher nicht darum gehen, die Leistung für eine bestimmte Aufgabe zu maximieren, sondern die mittlere Leistung, betrachtet über alle Anwendungsgebiete.

Ausfalltoleranz-Aspekt. Das Optimierungskriterium ist hier die Gewährleistung einer gewissen Mindest-Verfügbarkeit des Systems. Ein System heißt dann ausfalltolerant, wenn bei Ausfällen einzelner Komponenten des Systems noch ein betriebsfähiger Kern übrigbleibt, so daß eine bestimmte Mindestleistung zu allen Zeiten zur Verfügung steht.

Erweiterbarkeits-Aspekt. Bei diesem Aspekt handelt es sich darum, ein Rechnersystem so zu entwerfen, daß seine Erweiterbarkeit durch verschiedene Ausbaustufen ermöglicht wird.

Pragmatik-Aspekt. Unter diesem Stichwort fassen wir Architekturen zusammen, bei denen eine höhere Programmiersprache unmittelbar durch die Hardware des System interpretiert wird. Das bedeutet, daß die höhere Programmiersprache bereits die Maschinensprache des Systems ist, und daß zwischen der Darstellung eines Programms in der höheren Programmiersprache und der das Programm ausführenden Hardware keine weitere Zwischendarstellung existiert. Das Ziel ist hier, daß der Anwendungsprogrammierer, der Systemprogrammierer und möglichst auch der Wartungstechniker alle dieselbe Sprache sprechen [CHU 75]. Da dies die Beziehung zwischen einem Rechnersystem und den Personen, die mit seiner Programmierung und Wartung betraut sind, betrifft, sprechen wir von dem Pragmatik-Aspekt (wobei wir uns den Begriff *Pragmatik* aus der Semiotik ausleihen). Zum Pragmatik-Aspekt gehört auch die Forderung nach einer möglichst guten *Benutzbarkeit* und *Wartbarkeit* der Anlage.

Es muß betont werden, daß die genannten Aspekte sich keineswegs gegenseitig ausschließen. Gewisse architektonische Konzepte mögen dazu dienen, mehr als einem der genannten Aspekte Genüge zu tun. So mag man bestimmte Betriebsmittel zu einer Rechnerstruktur verbinden, die zunächst ausfalltolerant ist, daneben aber auch eine möglichst hohe Rechenleistung erbringt. Es muß ferner kaum besonders betont werden, daß der Leistungs-Aspekt und der Ausfalltoleranz-Aspekt mit Abstand die wichtigsten Gesichtspunkte beim Entwurf von Rechnerarchitekturen sind. Es ist für die Zukunft zu erwarten, daß als weiterer wichtiger Aspekt innovativer Rechnerarchitekturen der Aspekt der Systemsoftware-Vereinfachung ins Spiel kommt.

Dem Gebäude-Architekten stehen bestimmte Materialien wie Holz, Stein, Beton, Stahl, Glas zur Verfügung. Aus diesen Materialien bildet er zu-