2 Grundlagen der Datenverarbeitung

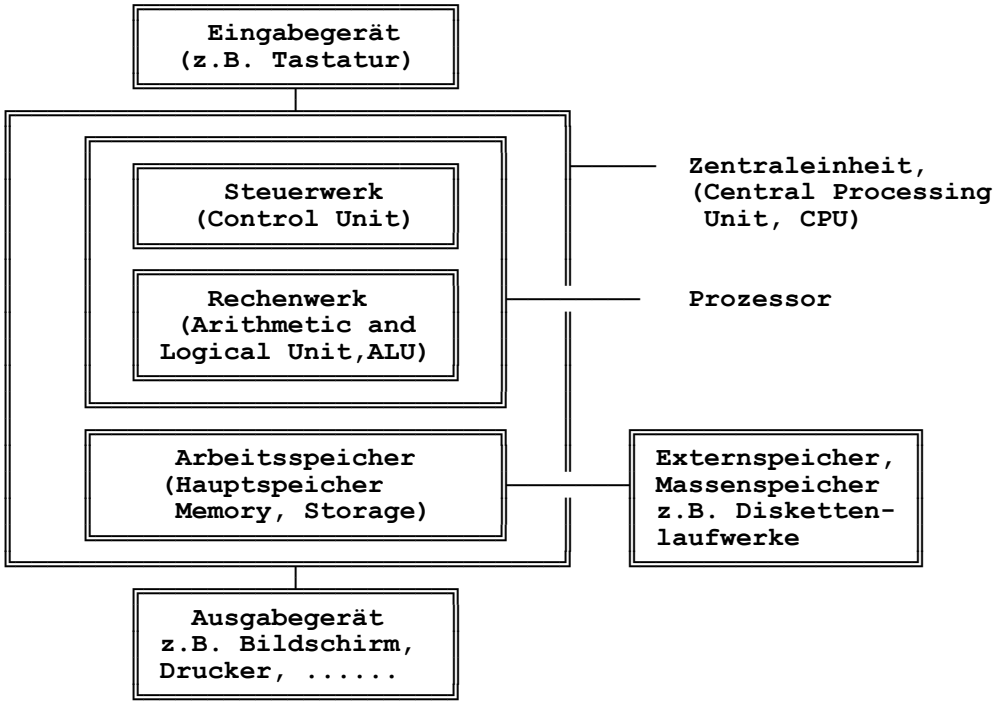
Hardware, Software, Betriebssystem, Übersetzungsprogramme, Programmiersprachen, Zahlensysteme, Zeichencodierung, Mikroprozessoren

Gliederung

2.1 Die Hardware

Die Hardware umfaßt die Geräte und die Datenträger.

Die folgende Skizze zeigt schematisch den Aufbau einer herkömmlichen Datenverarbeitungsanlage:



2.1.1 Die Geräte

a) **Zentraleinheit** (CPU = Central Processing Unit) mit den Komponenten:

a1) **Steuerwerk** (Control Unit)

a2) **Rechenwerk** (ALU = Arithmetic and Logical Unit)

Steuerwerk und Rechenwerk bilden den sog. Prozessor. Bei Mikrocomputern ist der Prozessor ein Mikroprozessor, für MS-DOS/Windows-PCs z.B. Intel 8088, 8086, 80286, 80386, 80486 (alle überholt), aktuell Pentium, Pentium Pro und Pentium MMX oder bei Apple Macintosh Motorola 68000, 68010, 68020, 68030 (alle überholt), aktuell MC 601, 602, 603 und 604.

a3) Arbeitsspeicher (Hauptspeicher, Internspeicher, Memory, Storage)

Für den Arbeitspeicher werden heute fast ausschließlich Halbleiter-Bauelemente eingesetzt. (Magnet-) Kernspeicher sind praktisch nicht mehr anzutreffen.

Nach der Zugriffsmöglichkeit wird beim Arbeitsspeicher zwischen den Grund-Speichertypen RAM und ROM unterschieden:

- **RAM:** **R**andom **A**ccess **M**emory. Schreib/Lese-Speicher.
  Technologisch ist bei Halbleiter-RAMs zwischen dynamischen RAMs (**DRAM**s, Standard) und statischen RAMs zu unterscheiden. DRAMs lassen eine höhere Integrationsdichte zu, sind wesentlich billiger, aber nicht so schnell wie sehr teuren statischen RAMs. Die Weiterentwicklung der DRAM führte zu **EDO RAM.** Extended Data Output RAM. Die ausgelesenen Daten stehen länger am Ausgang an, so daß überlappend und somit etwas schneller gelesen werden kann. Zugriffszeiten bei DRAM und Varianten: 70 oder 60 ns. Nach der Anordnung der RAMs auf kleinen Platinen unterscheidet man zwischen **SIMM** (Single Inline Memory Modul, einseitige Bestückung, früher 30, jetzt 72 Pins) und **DIMM** (Dual Inline Memory Modules, beidseitige Bestückung, 168 Pins).
- **ROM**: **R**ead **O**nly **M**emory. Nur-Lese-Speicher.

Varianten sind:

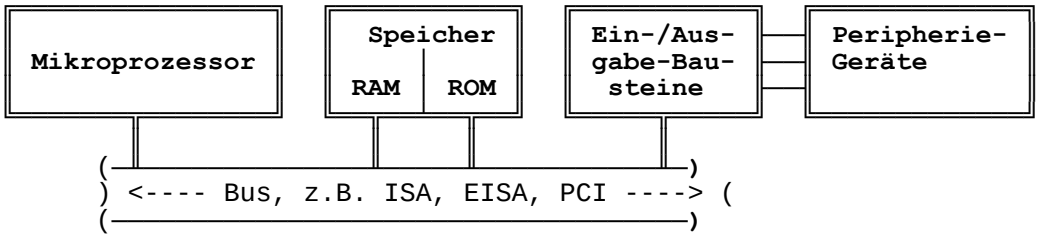
- **PROM**: **P**rogrammable **R**ead **O**nly **M**emory. Programmierbarer Nur-Lese-Speicher.
- **EPROM**: **E**rasable **P**rogrammable **R**ead **O**nly **M**emory. Mit Ultraviolett-Licht oder elektrisch (**EEPROM** = **E**lectrically **E**rasable **PROM**) löschbarer und neu programmierbarer Nur-Lese-Speicher.

b) Eingabeeinheit, z.B. Tastatur

c) Ausgabeeinheit, z.B. Bildschirm, Drucker

d) Externspeicher, z.B. Kassettenlaufwerke, Diskettenlaufwerke, Magnetbandlaufwerke, Magnetplattenlaufwerke. Externspeicher können aber auch als Ein- und Ausgabeeinheiten betrachtet werden.

Der Aufbau eines Mikrocomputers kann schematisch wie folgt dargestellt werden:



2.1.2 Die Datenträger

Beispiele: Kassetten, Disketten, Magnetbänder, Magnetplatten, optische Speicherplatten usw.

Bei Mikrocomputern werden Diskettenlaufwerke und nicht–wechselbare Magnetplatten (Festplatten, Harddisks, früher Winchester gennant) eingesetzt und im zunehmenden Maße optische Speicherplatten. Wechselbare Festplatten (Syquest, 44 MByte bis 270 MByte), Jazz (1GByte) sind seltener anzutreffen. Zur Datensicherung werden oft DAT-Streamer (DAT, Digital Audio Tape) mit Speicherkapazitäten bis 16 GByte (in komprimiertert Form) eingesetzt.

Diskettenformate:

- 5,25–Zoll, DS DD (double sided, double density), 360 KByte. Überholt.
- 5,25–Zoll, DS HD (double sided, high density), 1,2 MByte (AT–Disketten). Überholt.
- 3,5–Zoll, DS DD (double sided, double density), 720 KByte. Überholt.
- 3,5–Zoll, DS, HD (double sided, high density), 1,44 MByte
- Seit 1995: ZIP-Laufwerke und Datenträger mit 100 MByte
- Seit 1997: LS-120-Laufwerke und Datenträger mit 120 MByte

Zu den optischen Speichermedien:

- **CD-ROM**. Compact Disk ROM. Wird vom Hersteller beschrieben und kann nur gelesen werden. Vervielfältigung durch Pressen (sehr preiswert, weniger als 2 DM). Standard-Speicherkapazität: 650 MByte. Für Nachschlagewerke und Zubringer-Medium für große Programmpakete. Entwicklung der Geschwindigkeiten: Standard, Double-Speed, Quadro-Speed, 8-, 10-, 12- und 16-fach-Speed (1997) mit (theoretischen) Datenübertragungsrater von 1800 KByte/s. Zugriffszeiten z.T. schon unter 100 ms. Laufwerkskosten ca. 200 DM. Bislang arbeiten CD-ROM-Laufwerke mit konstantem Datenstrom (CLV, Constant Linear Velocity). Dabei muß der Spindelmotor innen beschleunigen und außen verzögern. In Zukunft aber mit konstanter Drehzahl (CAV, Constant Angular Velocity). Dabei ergeben sich außen höhere Übertragungsrateen (gut für Prospektangaben) als innen. Dafür entfällt das Beschleunigen und Verzögern des Spindelmotors. Die Daten aller CD-ROM-Varianten sind in spiraliger Form von innen nach außen aufgezeichnet.

- **WORM**. Write Once Read Mostly. Kann vom Anwender einmal beschrieben (gebrannt) und beliebig oft gelesen werden. Wird oft als Archivierungsmedium genutzt. Im Jahre 2000 kosten einfache CD-ROM-Brenner (Toaster) nur noch ca. 400 DM und Rohlinge ca. 2 DM.

- **MOD**. Magneto Optical Disk. Beliebig oft beschreib- und lesbar. Speicherkapazitäten von 120 MByte, 600 MByte (beidseitig) und höher.

- Ab 1997: **DVD** CD-ROM. DVD = Digital Video Disk (auch: Digital Versatile Disk). Informationsaufzeichnung ein- oder beidseitig (manuelles Wenden) in einer Ebene oder in zwei Ebenen. Folgende Formate:
  DVD5 einseitig, eine Ebene, 4.7 GByte
  DVD9 einseitig, zwei Ebenen, 8.5 GByte
  DVD10 zweiseitig, eine Ebene, 9.4 GByte. Manuelles Wenden
  DVD18 zweiseitig, zwei Ebenen, 17 GByte. Manuelles Wenden
  DVD-R zweiseitig, eine Ebene, Write Once, 3.8 GByte/Seite
  DVD-RAM zweiseitig, eine Ebene, wiederbeschreibbar, 2.6 GByte/Seite

- **Photo-CD**. Von speziell ausgestatteten Photogeschäften (Finishern) werden Kleinbild- und Mittelformatphotos (verschiedene Photo-CD-Klassen) auf die Photo-CD gescannt. Sehr preiswerter Scan (ca. 2 DM pro Bild). Ursprünglich für den Home-Einsatz (Betrachten von Urlaubsbildern auf dem Fernseher) gedacht. Einsatz aber auch in der Druckindustrie. Die Photo-CD kann in mehreren Sitzungen beschrieben werden. Die Daten können aber nur gelesen und nicht überschrieben werden.

Die Laufwerke der optischen Speicher sind im Vergleich mit Festplattenlaufwerken für magnetische Datenträger zur Zeit noch relativ langsam, was die Zugriffszeit (300 ms, 100 ms, ... 40 ms) und die Übertragungsgeschwindigkeit betrifft. Die Zugriffszeiten bei modernen Festplattenlaufwerken liegen z.T. unter 10 ms.

2.1.3 Zu Bits und Bytes

Die Größe einer Datenverarbeitungsanlage wird sehr oft an der Speicherkapazität des Arbeitsspeichers gemessen. Um ein Zeichen, z.B. einen Buchstaben abzuspeichern, benötigt man 8 bit (Bit = **bi**nary digi**t** = Binärziffer). Das Bit ist die kleinste Informationseinheit (Bit gesetzt: Strom fließt, Stelle magnetisiert, Loch gestanzt, Bit nicht gesetzt: Strom fließt nicht, Stelle nicht magnetisiert, Loch nicht gestanzt usw.). Der Arbeitsspeicher ist in Speicherzellen aufgeteilt, die je nach Rechnersystem 8 bit, 16 bit, usw. umfassen. Eine Gruppe von 8 bit wird vereinbarungsgemäß mit Byte bezeichnet. Es gilt also:

```
1 Byte = 8 bit
```

Mit 8 bit kann man $2^8 = 256$ verschiedene Bit–Muster darstellen. Benutzt man einen Zeichensatz mit 256 verschiedenen Zeichen, dann benötigt man zum Speichern eines Zeichens genau 1 Byte.

Zur Orientierung: Eine Schreibmaschinenseite mit z.B. 50 Zeilen mit je 40 Zeichen enthält insgesamt 2000 Zeichen. Um diese Seite zu speichern, benötigt man deshalb 2000 Byte. Es gilt:

1 KByte = 1 024 Byte	1024 = 2^{10}
1 MByte = 1 024 KByte = 1 048 576 Byte	1048576 = 2^{20}
1 GByte = 1 024 MByte = 1 073 741 824 Byte	1073741824 = 2^{30}
1 TByte = 1 024 GByte = 1 TeraByte = = 1 099 511 627 776 Byte	= 2^{40}

"K" steht für Kilo, "M" steht für Mega, "G" für Giga und "T" für Tera. Im EDV–Jargon werden oft KB und MB oder auch nur K oder M als Kürzel für KByte bzw. MByte usw. benutzt.

Mikrocomputer der ersten Generation mit 8–Bit–Prozessoren (Commodore CBM u.ä.) besaßen einen Arbeitsspeicher mit einer Kapazität von max. 64 KByte = 64*1024 Byte = 65536 Byte. Damit könnte man rund 32 der oben deklarierten Schreibmaschinenseiten abspeichern. Zu bedenken ist aber, daß im Arbeitsspeicher auch das Programm abgespeichert sein mußte und daß Teile des Arbeitsspeichers als ROM–Speicher dem Benutzer nicht für das Abspeichern eigener Daten zur Verfügung standen. Man war happy, mit einer in Basic geschriebenen Textverarbeitung "Dokumente" mit drei Seiten Schreibmaschinenschrift am Stück bearbeiten zu können.

2.2 Die Software

Zur Software gehören sowohl Programme (Anweisungen zum Verarbeiten von Daten) als auch die Daten selbst.

Die Programme werden unterteilt in:

a) **Anwenderprogramme**, die entweder vom Anwender selbst erstellt oder von einem Software–Hersteller bezogen wurden. Beispiele für Anwenderprogramme sind Programme für Textverarbeitung, Kalkulation, Finanz–Buchhaltung usw.

b) **Systemprogramme**, die ihrerseits unterteilt werden in:

b1) **Steuerprogramme**. Diese steuern den Ablauf eines Programmes und das Zusammenwirken der Zentraleinheit mit der Peripherie.

b2) **Dienstprogramme** (Hilfsprogramme, Utilitites). Dienstprogramme sind für den benutzerfreundlichen Betrieb einer Datenverarbeitungsanlage notwendig. Beispiele für Dienstprogramme sind sog. Editoren zum bequemen Erstellen eines Programmes in einer Programmiersprache, Sortierprogramme, Dateiverwaltungsprogramme usw.

b3) **Übersetzungsprogramme** zum Übersetzen eines in einer Programmiersprache geschriebenen Programmes in einen Code, der vom Computer verarbeitet werden kann. Übersetzungsprogramme sind Assembler, Compiler und Interpreter.

Steuerprogramme und Dienstprogramme bilden das sog. **Betriebssystem** (OS = Operating System).

Moderne Computer besitzen Plattenlaufwerke (Disks) wie Magnetplattenlaufwerke oder Diskettenlaufwerke. Als Sammelbezeichnung für darauf abgestimmte Betriebssysteme wird häufig "DOS" als Abkürzung für Disk Operating System benutzt.

2.3 Betriebssysteme für Mikrocomputer

Als Betriebssystem bezeichnet man die elementare Software, die zum Betrieb einer Datenverarbeitungsanlage unbedingt notwendig ist.

Zu den Aufgaben des Betriebssystems zählen u.a.:

- Hardware des Rechners nach dem Einschalten testen und in einen bestimmten Anfangszustand setzen
- Verwalten des Arbeitsspeichers und des Externspeichers
- Datenübertragung zwischen Speicher, Tastatur, Bildschirm und anderen Hardware–Komponenten steuern
- Dem Anwender Befehle für die Kommunikation mit dem Computer zur Verfügung stellen

Anmerkung: Software, die in ROMs, PROMs oder EPROMs gespeichert ist, bezeichnet man auch mit "Firmware".

Für Mikrocomputer stehen u.a. folgende rechner–unabhängige Betriebssyteme zur Verfügung:

- **MS-DOS** (**M**icro**s**oft **D**isk **O**perating **S**ystem), Microsoft, USA. MS-DOS ist Quasi-Standard ("Industriestandard") für 16-, 16/32- und 32/32-Bit-Mikrocomputer (PC, Personal Computer, persönlicher Computer) mit den Intel-Mikroprozessoren 8088, 8086 (adressieren max. 2^{20} = 1MByte), 80286 (adressieren max. 2^{24} = 16 MByte, 80386, 80486 und Pentium (adressieren max. 2^{32} = 4 GByte). Mit MS-DOS können aber in der originalen Form nur 640 KByte verwaltet werden. Das IBM-Betriebssystem PC-DOS ist praktisch mit MS-DOS identisch. MS-DOS gibt es in folgenden Versionen: 1.27 (veraltet), 2.11, 3.1, 3.2, 3.3, 4.0, 5.0, 6.0, 6.1, 6.2 (alle veraltet). 1997 aktuell ist die in Windows 95 enthaltene Version 7.0.

- **DR–DOS** von Digital–Research, USA, später von Novell übernommen. Seit ca. 1994 nicht mehr auf dem Markt.

- **OS/2** von IBM. Ursprünglich zusammen mit Microsoft entwickeltes Betriebssystem mit grafischer Benutzeroberfläche. Später Trennung, Microsoft geht mit Windows eigene Wege. OS/2 gilt als "stabiler" als Windows, zumindest bis zu der Version Windows 95, konnte sich aber wegen mangelnder Unterstützung durch Software-Entwickler gegen Windows nicht durchsetzen. Windows-Programme laufen unter OS/2, allerdings nur in Emulation.

- **Windows**, Microsoft USA. Windows ist bis einschließlich Version 3.3 kein eigenständiges Betriebssystem, sondern eine MS–DOS–Betriebssystemerweiterung mit einer komfortablen grafischen Benutzeroberfläche. Windows ist auf Intel-Prozessoren oder dazu kompatiblen Prozessoren (Cyrix, AMD = Advanced Micro Devices) abgestimmt und somit wie DOS hardwareabhängig. Erst ab Version Windows 95 ist Windows ein eigenständiges Betriebssystem, das aber ein rudimentäres DOS enthält, damit DOS-Programme weiter lauffähig bleiben.

  Die aktuelle Version Windows 95 setzt voraus: Prozessoren ab Intel 80386 (gibt praktisch keinen Sinn, besser Pentium und höhere) und mind. 16 MByte Arbeitsspeicher.

- **Windows NT** (NT = New Technology), Microsoft, ab 1994. Eigenständiges Betriebssystem im Stil von Windows (aber stabiler), das aber nicht mehr an Intel–Prozessoren alleine gebunden ist. 1997 wird Windows NT noch vorrangig als Netzwerks-Betriebssystem (Konkurrenz zu Novell) eingesetzt; der Einsatz in Workstations (Arbeitsstationen) ist aber stark zunehmend. Für 1998/1999 wird ein Verschmelzen von Windows und Windows NT erwartet. Windows-Anwendungen laufen auch unter Windows NT.

- **UNIX,** Bell Laboratories, USA. Betriebssystem für Mehrprogramm-Betrieb (multi-programming, multi tasking) und Mehrbenutzer-Betrieb (multi user) für 16-Bit- und 32-Bit-Mikrocomputer, aber auch für Mini und Groß-Computer. Unixist nicht an einen bestimmten Prozessortyp gebunden. Unix ist zum größten Teil in der Programmiersprache C (siehe später) geschrieben und somit leicht auf unterschiedliche Computer übertragbar.Abwandlungen (Derivate) von Unix sind u.a. Xenix (Microsoft), Sinix (Siemens), AU/X (Apple), AIX (IBM), Ultrix (DEC), **Linux** (Public Domain). Die Unix-Derivate sind nicht alle kompatibel. Die Bedienung von Unix über Kommandozeilen-Befehle ist schwieriger als bei DOS, vor allem wegen

der viel größeren Anzahl von Befehlsoptionen. Komfortabel wird Unix bei Verwendung der grafischen Benutzeroberfläche **X-Window** (nicht X-Windows).

Neben diesen Betriebssystemen gibt es auch noch rechnerabhängige Betriebssysteme für Mikrocomputer, z.B. für den Apple Macintosh. Übertragbarkeit von Programmen setzt u.a. gleiches Betriebssystem voraus.

2.4 Übersetzungsprogramme
Assembler, Compiler und Interpreter

Man unterscheidet zwischen den Übersetzungsprogrammen für

- **maschinenorientierte Programmiersprachen** (prozessor–orientierte Programmiersprachen)

  und

- **problemorientierte Programmiersprachen** ("Höhere" Programmiersprachen, prozessorunabhängig).

- Bei einer maschinenorientierten Programmiersprache wird der **Quell–Code** mit dem Hilfsprogramm **Editor** in **mnemotechnischer Schreibweise** geschrieben und dann mit dem Übersetzungsprogramm **Assembler** in den Maschinencode übersetzt. Die Übersetzung erfolgt *1:1*, d.h. aus *jeder* Anweisung im mnemotechnischen Code resultiert genau *eine* Anweisung im Maschinencode, Makro–Befehle ausgenommen. Oft bezeichnet man auch das mnemotechnische Codieren selbst mit Assembler–Programmierung. Assembler–Programme sind schwieriger zu erstellen als Programme in einer höheren Programmiersprache. Der Vorteil liegt vor allem in der kürzeren Programm–Laufzeit, wenn beim Programmieren die Möglichkeiten des Prozessors optimal genutzt werden und im geringeren Speicherbedarf des übersetzten Programmes. Ein wesentlicher Nachteil besteht darin, daß die Programme nicht auf Computer mit anderen Prozessortypen übertragen werden können, was mit höheren Programmiersprachen im Prinzip möglich ist, wenn auch manchmal wegen mangelhafter Standardisierung etwas schwierig.

- Bei einer höheren Programmiersprache steht nicht der Prozessor, sondern das Problem im Vordergrund. Dementsprechend sind die Befehle mit umgangssprachlichen (meist englischen) Bezeichnungen in Klarsprache festgelegt. Die Übersetzung erfolgt *1:n*, d.h. aus *einer* Anweisung im Quellcode (Source Code) entstehen *n*–Anweisungen im Maschinencode (Object Code).

Bei den Übersetzungsprogrammen für höhere Programmiersprachen ist zu unterscheiden zwischen Compilern und Interpretern.

- **Der Compiler** übersetzt den Quellcode komplett in den Maschinencode. Der Quellcode wird mit dem Hilfsprogramm Editor erstellt. Beim Übersetzen werden Syntaxfehler, d.h. Verstöße gegen die festgelegte grammatikalische Schreibweise der Befehle und teilweise auch Semantikfehler, das sind logische Fehler, festgestellt und ausgegeben. Nach der Fehlerbehebung im Quellcode muß das ganze Programm neu compiliert werden; die Fehlerbehebung kann bei Compilern und Editoren mit geringem Komfort sehr langwierig werden. Mit dem Binder–Programm (engl. Linker) können mehrere compilierte Programmteile zu einem ausführbarem Programm verbunden werden. Änderungen, Korrekturen und Erweiterungen des Programms sind praktisch nur möglich, wenn der Quellcode zur Verfügung steht. Der Vorteil des compilierten Programmes liegt vor allem in der kürzeren Programmlaufzeit, verglichen mit einem Interpreter–Programm.

- **Beim Interpreter** wird bei *jedem* Programmlauf der Quellcode zeilenweise übersetzt *und* sofort ausgeführt (interpretiert). Die Übersetzung, der Maschinencode, wird nach der Ausführung der Zeile "vergessen". Das trifft auch für Programmzeilen zu, die in einer Schleife mehrmals durchlaufen werden. Somit ergeben sich beim Interpreter längere Programmlaufzeiten als beim Compiler. Der Vorteil des Interpreters liegt vor allem in der bequemeren Programmerstellung und Fehlerbehebung. Für das Erstellen des Programms stehen in einer Interpreter–Sprache Edit–Funktionen zur Verfügung, wenn auch meist einfacher Art.

Durch die Wahl der Programmiersprache ist man in der Regel bereits auf Compiler oder Interpreter festgelegt.

Neben dem weitverbreiteten (Standard–) BASIC ist unter den gängigen Programmiersprachen fast nur noch APL als Interpretersprache für allgemeine Anwendungen nennenswert. Fast alle anderen höheren Sprachen sind Compilersprachen. Dazu zählen auch die neueren BASIC–Dialekte Power–Basic und QuickBasic. Erwähnenswert ist aber die Seitenbeschreibungssprache **PostScript** (Firma Adobe, USA), die interpretativ arbeitet. Das Kap. 04 in Datenverarbeitung II bringt einen Einblick in PostScript. Die von Sun 1995 vorgestellte Sprache Java zur Entwicklung von Internet-Anwendungen (Applets) arbeitet ebenfalls interpretativ.

2.5 Überblick über höhere Programmiersprachen

Weltweit existieren über 500 Programmiersprachen. Die folgende Aufzählung kann nur eine kleine Auswahl zeigen:

- **FORTRAN** (**For**mular **Tran**slator). Entstand 1950. Zur Zeit noch die bedeutendste Sprache für technisch/wissenschaftliche Anwendungen. Auf Mikrocomputern aber nur wenig verbreitet. Frühzeitige Standardisierung (FORTRAN IV und FORTRAN 77). Großes Programmangebot.

- **COBOL** (**Co**mmon **B**usiness **O**riented **L**anguage). Enstand 1959. Mitinitiator: US–Verteidigungsministerium. Zur Zeit noch die bedeutendste Sprache für kommerzielle Anwendungen. Auf PCs wegen des großen Sprachumfangs kaum oder nur in Subsets vertreten.

- **ALGOL** (Algorithmic Language). Entstand 1960 an der damaligen Technischen Hochschule München. Für technisch/wissenschaftliche Anwendungen. Lange Zeit Ausbildungssprache an Hochschulen. Heute keine Bedeutung mehr. ALGOL gilt aber als "Mutter" der modernen Sprachen Pascal, Ada und Modula 2.

- **PL/1** (Programming Language 1). 1963 von IBM als Universalsprache für technisch/wissenschaftliche *und* kommerzielle Aufgaben entwickelt. Konnte aber FORTRAN und COBOL nicht verdrängen. Wegen des großen Sprachumfanges auf PCs praktisch nicht vertreten.

- **BASIC** (**B**eginners **A**ll–Purpose **S**ymbolic **I**nstruction **C**ode). 1962 als einfach erlernbare Sprache in Dartmouth, USA, entwickelt. Gewisse Verwandtschaft mit FORTRAN. BASIC dürfte wegen der Mikrocomputer derzeit die weitest verbreitete Programmiersprache der Welt sein. BASIC eignet sich mit Einschränkungen sowohl für technisch/wissenschaftliche als auch für kommerzielle Anwendungen. Die erst spät einsetzenden Standardisierungsbemühungen konnten den Wildwuchs an BASIC–Dialekten nicht mehr eindämmen. Eine Übertragbarkeit der BASIC–Programme ist bei unterschiedlichen Dialekten nur mit großen Einschränkungen gegeben. Die mangelhaften Strukturierungsmöglichkeiten sind ein weiterer Vorwurf gegen BASIC. Die ungezügelte Verwendung des Sprungbefehls GOTO kann zu nicht mehr durchschaubaren Programmen führen. Längst veraltete BASIC-Interpreter: MSBASIC und GWBASIC. Moderne Varianten (die häufig Compiler sind) wie QuickBasic, Qbasic, Power-Basic besitzen Strukturierungsmöglichkeiten und sonstige Merkmale (keine Zeilennummern, Blockstruktur möglich, Prozeduren und Funktionen mit lokalen Variablen, Rekursionen usw.), die denen von Pascal nicht viel nachstehen und in Einzelfällen sogar vorteilhafter sind.

  Durch die Windows-Varianten **Visual Basic** (VB) und **Visual Basic for Application** (VBA, Makrosprache für Windows-Anwendungen), beide von Microsoft, hat Basic ab etwa 1995 eine bemerkenswerte Renaissance erlebt. Mit VB bzw. VBA können wesentlich einfacher Windows-Anwendungen entwickelt werden als z.B. mit C++. Microsoft wird alle Windows-Anwendungen mit der Makrosprache VBA ausstatten. 1997 sind es Excel 7.0 und WinWord 8.0. Siehe auch Delphi.

- **Pascal**. Von Prof. Wirth (ETH Zürich, früher TH München) 1972 vorgestellte Sprache. Obwohl ursprünglich nur als Ausbildungssprache konzipiert, hat sich Pascal relativ schnell durchgesetzt, überwiegend im Ausbildungsbereich und im

technisch/wissenschaftlichen Bereich. Pascal gilt als die Mustersprache für strukturiertes Programmieren. Neben dem standardisierten (und recht spröden) Pascal entstehen mehrere Dialekte. Einige Beispiele: Pascal/Z, OMSI–Pascal, UCSD–Pascal, Quick–Pascal (Microsoft) und Turbo–Pascal (Borland). Turbo-Pascal dominiert eindeutig.

Für die Entwicklung von Windows-Anwendungen wurde von Borland um 1993 "**Pascal for Windows**" auf den Markt gebracht, das sich aber wegen der Komplexität nicht durchsetzen konnte. 1995 bringt Borland "**Delphi**", das ähnlich wie Visual Basic eine relativ komfortable Entwicklung von Pascal-Programmen für Windows ermöglicht. Delphi ist im Gegensatz zu Visual Basic ein echter Compiler. Dennoch bleibt abzuwarten, ob sich Delphi gegenüber der starken Verbreitung von Visual Basic durchsetzen kann. Delphi ist kein Ersatz für Pascal, sondern setzt fast den gesamten Sprachumfang von Turbo-Pascal voraus.

- **Ada**. Nach Auguste Ada Byron (erste Programmiererin der Welt!) benannte Universalsprache. 1980 vorgestellt. Wie COBOL vom US–Verteidigungsministerium initiiert. Von Anfang an standardisiert. Sehr umfangreiche Sprache, mit vielen Elementen von Pascal. Auf Mikrocomputern nur vereinzelt Subsets.

- **Modula 2**. Von Prof. Wirth Anfang der 80er Jahre vorgestellte Nachfolger–Sprache zu Pascal. Soll wegen der Kompaktheit, im Gegensatz zu Ada, auch als Universalsprache für kleinere Computer geeignet sein. Hat sich nicht durchgesetzt.

- **C**. Von den Bell–Laboratorien in Verbindung mit dem Betriebssystem UNIX entwickelt. C ist eine sehr leistungsfähige (und anspruchsvolle) Mischung aus einer maschinenorientierten und einer problemorientierten Programmiersprache und dennoch weitgehenst prozessorunabhängig. Bei der Systemprogrammierung (Entwicklung von Betriebssystemen, Compilern u.ä.) wird die Sprache C zunehmend als Ersatz für Assembler verwendet. Compiler stehen für praktisch alle Rechnertypen zur Verfügung, vom PC bis zum Großrechner.

- Kleine Aufzählung weiterer Programmiersprachen:

APL	Interpretersprache für universelle Anwendung. Besonders für die Behandlung von Feldern (Matrizen) geeignet. Kurzschreibweise (Symbole) der Befehle.
PILOT	Einfache Schulsprache, einfacher als BASIC
LISP	Für "künstliche Intelligenz" (artificial intelligence)
PROLOG	Für "künstliche Intelligenz"
COMAL	BASIC mit Pascal–Eigenschaften
RPG	Programm–Generator für kommerzielle Aufgaben. Relativ starre Programmstruktur
FORTH	Für Prozeßsteuerung
EXAPT	Für Werkzeugmaschinensteuerung
PCL	Page Control Language. Nichtprogrammierbare Seitenbeschreibungs sprache (statische ...) von Hewlett & Packard (HP). Druckersprache

Esc/P	Druckersprache (statisch) von Epson
Prescribe	dito von Kyocera
HP/GL	Graphic Language. Nichtgrammierbare Plottersprache von HP
PostScript	Programmierbare Seitenbeschreibungssprache (dynamische ...) von Adobe
Java	Interpreter für die Entwicklung von kleinen Internet-Anwendungen (Applets). 1996 von Sun vorgestellt. Microsoft entwickelt daraufhin eigene Produkte, die aber nicht ganz kopmatibel sind.
JavaScript	Interpreter für Quelltext in HTML-Seiten. Vom Browserhersteller Netscape entwickelt.

2.6 Zahlensysteme

2.6.1 Das duale Zahlensystem

Das duale Zahlensystem hat bekanntlich die Basis 2. Binär heißt, daß eine Zahl mit zwei Symbolen dargestellt wird. Über die Wertigkeit, die sich aus der Basis ergibt, ist damit streng genommen noch nichts gesagt. Da die feinen Unterschiede für die Datenverarbeitungspraxis belanglos sind, wird im folgenden nur vom Binärsystem gesprochen und die Basis 2 unterstellt. Für die Darstellung des Binärsystems werden üblicherweise die Symbole 0 (für Bit nicht gesetzt) und 1 (für Bit gesetzt) benutzt, obwohl die einschlägige Norm für diese Zwecke die Symbole 0 und L empfiehlt.

Für die weitere Darstellung wird ein aus 8 bit bestehendes Bit–Muster benutzt: Die Zählung der Bits beginnt rechts mit dem niederwertigsten Bit (least significant bit, LSB), es hat die Nummer 0. Das höchstwertige Bit (most significant bit, MSB) hat dann die Nummer 7. Nach DIN sollte die Zählung von 1 bis 8 laufen.

Die allgemein übliche Zählung von 0 bis 7 (und nicht von 1 bis 8) hat den Vorteil, daß sich die dezimale Bit–Wertigkeit des Bits mit:

$$\text{Wertigkeit} = 2^{\text{BitNr}}$$

einfach darstellen läßt, wie folgendes Beispiel zeigt:

```
<————————————————————————————————
  7    6    5    4    3    2    1    0    Bit–Nr
 2^7  2^6  2^5  2^4  2^3  2^2  2^1  2^0    Bit–Wertigkeit
128   64   32   16   8    4    2    1    Bit–Wertigkeit
```

Durch Multiplikation der einzelnen Bit–Wertigkeiten mit 1 (wenn Bit gesetzt) bzw. 0 (wenn Bit nicht gesetzt) und anschließender Summation erhält man das Dezimal–Äquivalent des gesamten Bit–Musters.

Beispiel:

```
<———————————————————————————————
 7    6    5    4    3    2    1    0   Bit–Nr
 2⁷   2⁶   2⁵   2⁴   2³   2²   2¹   2⁰  Bit–Wertigkeit
128   64   32   16   8    4    2    1   Bit–Wertigkeit
———————————————————————————————————————
 0    1    1    0    1    0    1    1   Bit–Muster (Beispiel)
———————————————————————————————————————
 0  + 64 + 32 + 0 +  8  + 0  + 2  + 1   = 107
                                        (Dezimal-äquivalent des
                                         des Bit–Musters = 107)
```

Mit 8 bit kann man $2^8 = 256$ verschiedene Bit–Muster darstellen, mit 16 bit erhält man $2^{16} = 65536$ verschiedene Bit–Muster, mit 32 bit entsprechend $2^{32} = 4\,294\,967\,296$ verschiedene Bit–Muster usw.

Auszug aus der 8-Bit-Muster-Tabelle:

Mit Dez ist das Dezimal–Äquivalent des Bit–Musters nach obigem Beispiel gemeint, mit Hex der entsprechende Wert in hexadezimaler Darstellung. Zur besseren Lesbarkeit sind die Bit–Muster in zwei Vierer–Gruppen unterteilt. Die Hex–Zahlen werden im nächsten Unterpunkt erklärt.

Dez	Bit–Muster	Hex
000	0000_0000	00
001	0000_0001	01
002	0000_0010	02
003	0000_0011	03
004	0000_0100	04
005	0000_0101	05
006	0000_0110	06
007	0000_0111	07
008	0000_1000	08
009	0000_1001	09
010	0000_1010	0A
011	0000_1011	0B
012	0000_1100	0C
013	0000_1101	0D
014	0000_1110	0E
015	0000_1111	0F
016	0001_0000	10
017	0001_0001	11
018	0001_0010	12

Dez	Bit–Muster	Hex
015	0000_1111	0F
016	0001_0000	10
017	0001_0001	11
...		
031	0001_1111	1F
032	0010_0000	20
033	0010_0001	21
...		
063	0011_1111	3F
064	0100_0000	40
065	0100_0001	41
...		
127	0111_1111	7F
128	1000_0000	80
129	1000_0001	81
...		
253	1111_1101	FD
254	1111_1110	FE
255	1111_1111	FF

Tabelle der Zweier-Potenzen von 0 bis 32:

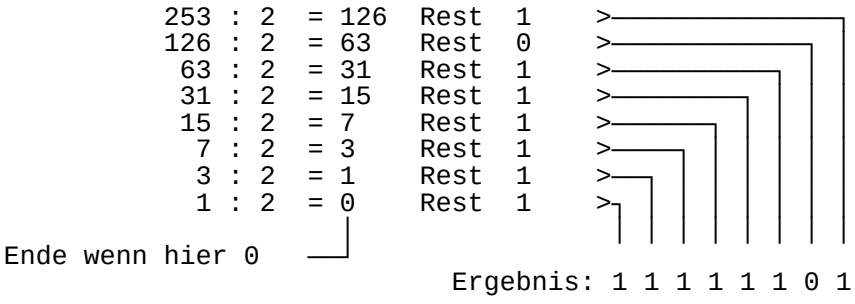
n	2^n Dez	2^n Hex
0	1	0000
1	2	0002
2	4	0004
3	8	0008
4	16	0010
5	32	0020
6	64	0040
7	128	0080
8	256	0100
9	512	0200
10	1 024	0400
11	2 048	0800
12	4 096	1000
13	8 192	2000
14	16 384	4000
15	32 768	8000
16	65 536	01 0000

n	2^n Dez	2^n Hex
16	65 536	0001 0000
17	131 072	0002 0000
18	262 144	0004 0000
19	524 288	0008 0000
20	1 048 576	0010 0000
21	2 097 152	0020 0000
22	4 194 304	0040 0000
23	8 388 608	0080 0000
24	16 777 216	0100 0000
25	33 554 432	0200 0000
26	67 108 864	0400 0000
27	134 217 728	0800 0000
28	268 435 456	1000 0000
29	536 870 912	2000 0000
30	1 073 741 824	4000 0000
31	2 147 483 648	8000 0000
32	4 294 967 296	01 0000 0000

Umwandlung dezimal/binär:

Für die manuelle Umrechnung einer Dezimalzahl in eine Binärzahl kann man die Verfahren benutzen, die mit folgenden Beispielen gezeigt werden:

1. Beispiel: Dezimalzahl 253



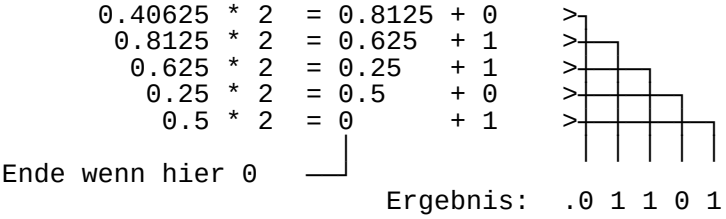
Probe:

Bit-**Muster** **1111 1101**

Entspricht: $1*2^7 + 1*2^6 + 1*2^5 + 1*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0$

128 + 64 + 32 + 16 + 8 + 4 + 0 + 1 = 253

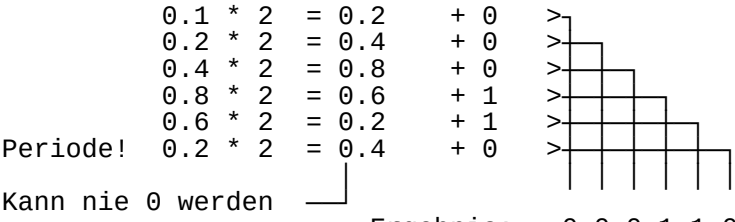
2. Beispiel: Dezimalzahl 0.40625



Probe:

Bit–**Muster** .01101
Entspricht: $0*2^{-1} + 1*2^{-2} + 1*2^{-3} + 0*2^{-4} + 1*2^{-5} =$
$0*0.5 + 1*0.25 + 1*0.125 + 0*0.0625 + 1*0.03125 = 0.40625$

3. Beispiel: Dezimalzahl: 0.1



Probe:

Nach 5 Stellen abgebrochenes Bit–**Muster**: .00011
entspricht: $0*2^{-1} + 0*2^{-2} + 0*2^{-3} + 1*2^{-4} + 1*2^{-5} =$
$0*0.5 + 0*0.25 + 0*0.125 + 1*0.0625 + 1*0.03125 = 0.09375$

Da in der Datenverarbeitung Kommazahlen (genauer Realzahlen) unabhängig von der Größe nur mit einer bestimmten Bytezahl gespeichert werden können, ist bei diesen Daten immer mit einem Fehler zu rechnen. Dieser Fehler kann sich erheblich fortsetzen, so daß das Assoziativ-Gesetz in der numerischen Datenverarbeitung im allgemeinen nicht gültig ist, wie folgendes Beispiel (Ausgabe eines Pascal-Programms) zeigt, bei dem S1 gleich S2 und (S1 - S2) Null sein müßte.

Zum Assoziativ-Gesetz in der numerischen Datenverarbeitung:

Das Assoziativ-Gesetz gilt nicht in der numerischen Datenverarbeitung

S1 = (A - B) + C	S2 = A + (C - B)	S1 - S2
1.11111100000016E+0000	1.11111100000016E+0000	0.00000000000000E+0000
1.11111100000016E+0000	1.11111100000016E+0000	0.00000000000000E+0000
1.11111100000016E+0000	1.11111100000016E+0000	0.00000000000000E+0000
1.11111100000016E+0000	1.11111100000562E+0000	-5.45696821063757E-0012
1.11111100000016E+0000	1.11111100018024E+0000	-1.80079950951040E-0010
1.11111100000016E+0000	1.11111068725586E+0000	-5.45696821063757E-0007
1.11111100000016E+0000	1.11108398437500E+0000	3.12744305119850E-0005
1.11111100000016E+0000	1.10937500000000E+0000	1.73600000016449E-0003

Fehler dieser Art können noch viel größer werden, siehe das spätere Praktikumsbeispiel „Hilbert-Matrix“, mit dem mühelos 1 > 4711 oder ähnlicher Unsinn „bewiesen“ werden kann.

2.6.2 Das hexadezimale Zahlensystem

Die Bezeichnung "hexadezimal" ist eigentlich falsch. Richtiger müßte die Bezeichnung "sedezimal" lauten, da dieses Zahlensystem die Basis 16 hat. Das hexadezimale Zahlensystem (im Jargon das Hex–System) wird in der Datenverarbeitung als Kurzschreibweise für das Binärsystem benutzt. Binäre Zahlendarstellungen sind sehr lang, wie das letzte Beispiel gezeigt hat. Sie sind deshalb für Ein– und Ausgaben nicht sehr geeignet. Andereits ist die Umwandlung dezimal<—>binär zeitaufwendig. Als Lösung bietet sich das Hex–System an, da wegen der Basis 16 eine Gruppe von 4 bit (Tetrade, Nibble) durch 1 Hex–Ziffer dargestellt werden kann. Ein Byte besteht bekanntlich aus 8 bit und kann somit durch 2 Hex–Ziffern dargestellt werden.

Ein Zahlensystem muß soviele verschiedene Zeichen haben, wie es als Basis besitzt. Das Hex–System braucht 16 verschiedene Zeichen, das Binärsystem nur 2.

Vereinbarungsgemäß werden im Hex–System die Ziffernzeichen 0 bis 9 und die Buchstabenzeichen A, B, C, D, E und F nach folgendem Schema benutzt:

dezimal	0	1	2	3	4	5	6	7	9	9	10	11	12	13	14	15
hexadezimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Eine Hex–Zahl kann somit auch nur aus Ziffernzeichen bestehen. Um visuelle Verwechslungen mit Dezimalzahlen in jedem Falle auszuschließen, ist eine zusätzliche Kennung notwendig. In den meisten Basic–Dialekten ist dafür das Symbol **&H** vorgesehen, das der Hex–Zahl voranzustellen ist. In anderen Sprachen oft auch nur voran– oder nachgestelltes Symbol **H**. Groß–/Kleinschreibung der Hex–Zeichen und Symbole beliebig.

In Turbo–Pascal wird das vorangestellte Dollarzeichen **$** als Symbol für Hex–Zahlen benutzt. Üblicherweise werden Hex–Zahlen geradstellig dargestellt, also Stellenzahl 2, 4, 6 usw. Gegebenenfalls wird eine führende Null vorangestellt, um eine gerade Stellenzahl zu erhalten.

Umwandlung dezimal/hexadezimal:

Für die manuelle Umwandlung einer Dezimalzahl in eine Hex–Zahl kann man das von der Umwandlung in Binärzahlen bekannte Verfahren benutzen. Lediglich der Divisor muß auf 16 geändert werden.

Beispiel: Dezimalzahl 2612

```
      2612 : 16 = 163  Rest  4 ——_ 4   >——————
       163 : 16 =  10  Rest  3 ——_ 3   >———————
        10 : 16 =   0  Rest 10 ——_ A   >———————
                                  _
Ende, wenn hier 0 ——┘
                              Ergebnis:    A 3 4
```

Das Ergebnis mit führender Null (gerade Stellenzahl) und Turbo–Pascal–Hex–Symbol: $0A34

Beim manuellen Umwandeln einer Hex–Zahl in eine Dezimalzahl ist die genannte Zeichenzuordnung und die Basis 16 zu beachten.

Beispiel: Die Hex–Zahl $0A34 ist in eine Dezimalzahl umzuwandeln

```
16^3  16^2  16^1 16^0   Wertigkeit
4096 256  16  1    Wertigkeit
————————————————————
0      A   3  4    die Hex–Ziffen
0     10   3  4    die dezimale Darstellung der Hex–Ziffer
————————————————————
0*4096 + 10*256 + 3*16 + 4*1 = 2612
```

2.6.3 Das oktale Zahlensystem

Das bei Mikrocomputern nur noch selten eingesetzte oktale Zahlensystem benutzt die Basis 8. Es ist ebenso wie das hexadezimale Zahlensystem eine Kurzschrift des Binärsystems. Beim Oktal–System werden drei Binärziffern zu einer Oktalziffer zusammengefaßt. Es werden dafür die Zeichen '0'..'7' benutzt. Als Zusatzkennung werden von manchen Programmen die Zeichen 'O' (leicht zu verwechseln mit dem Ziffernzeichen '0') oder 'Q' und benutzt, die je nach Programm vorangestellt oder nachgestellt werden. Turbo–Pascal unterstützt das oktale Zahlensystem nicht.

Hinweis: Das oktale Zahlensystem wird z.B. in **PostScript** zur Eingabe von Zeichen gebraucht, die über die Tastatur nicht erreicht werden.

2.6.4 Gegenüberstellung der Zahlensysteme im Dezimalbereich bis 255

Bei der binären Darstellung wird in der Tabelle das Unterstreichungszeichen nur zur optischen Trennung des sonst etwas unübersichlichen 8-Bit-Musters benutzt.

dez	hex	okt	binär	dez	hex	okt	binär	dez	hex	okt	binär	dez	hex	okt	binär
000	h00	o000	0000_0000	064	h40	o100	0100_0000	128	h80	o200	1000_0000	192	hC0	o300	1100_0000
001	h01	o001	0000_0001	065	h41	o101	0100_0001	129	h81	o201	1000_0001	193	hC1	o301	1100_0001
002	h02	o002	0000_0010	066	h42	o102	0100_0010	130	h82	o202	1000_0010	194	hC2	o302	1100_0010
003	h03	o003	0000_0011	067	h43	o103	0100_0011	131	h83	o203	1000_0011	195	hC3	o303	1100_0011
004	h04	o004	0000_0100	068	h44	o104	0100_0100	132	h84	o204	1000_0100	196	hC4	o304	1100_0100
005	h05	o005	0000_0101	069	h45	o105	0100_0101	133	h85	o205	1000_0101	197	hC5	o305	1100_0101
006	h06	o006	0000_0110	070	h46	o106	0100_0110	134	h86	o206	1000_0110	198	hC6	o306	1100_0110
007	h07	o007	0000_0111	071	h47	o107	0100_0111	135	h87	o207	1000_0111	199	hC7	o307	1100_0111
008	h08	o010	0000_1000	072	h48	o110	0100_1000	136	h88	o210	1000_1000	200	hC8	o310	1100_1000
009	h09	o011	0000_1001	073	h49	o111	0100_1001	137	h89	o211	1000_1001	201	hC9	o311	1100_1001
010	h0A	o012	0000_1010	074	h4A	o112	0100_1010	138	h8A	o212	1000_1010	202	hCA	o312	1100_1010
011	h0B	o013	0000_1011	075	h4B	o113	0100_1011	139	h8B	o213	1000_1011	203	hCB	o313	1100_1011
012	h0C	o014	0000_1100	076	h4C	o114	0100_1100	140	h8C	o214	1000_1100	204	hCC	o314	1100_1100
013	h0D	o015	0000_1101	077	h4D	o115	0100_1101	141	h8D	o215	1000_1101	205	hCD	o315	1100_1101
014	h0E	o016	0000_1110	078	h4E	o116	0100_1110	142	h8E	o216	1000_1110	206	hCE	o316	1100_1110
015	h0F	o017	0000_1111	079	h4F	o117	0100_1111	143	h8F	o217	1000_1111	207	hCF	o317	1100_1111
016	h10	o020	0001_0000	080	h50	o120	0101_0000	144	h90	o220	1001_0000	208	hD0	o320	1101_0000
017	h11	o021	0001_0001	081	h51	o121	0101_0001	145	h91	o221	1001_0001	209	hD1	o321	1101_0001
018	h12	o022	0001_0010	082	h52	o122	0101_0010	146	h92	o222	1001_0010	210	hD2	o322	1101_0010
019	h13	o023	0001_0011	083	h53	o123	0101_0011	147	h93	o223	1001_0011	211	hD3	o323	1101_0011
020	h14	o024	0001_0100	084	h54	o124	0101_0100	148	h94	o224	1001_0100	212	hD4	o324	1101_0100
021	h15	o025	0001_0101	085	h55	o125	0101_0101	149	h95	o225	1001_0101	213	hD5	o325	1101_0101
022	h16	o026	0001_0110	086	h56	o126	0101_0110	150	h96	o226	1001_0110	214	hD6	o326	1101_0110
023	h17	o027	0001_0111	087	h57	o127	0101_0111	151	h97	o227	1001_0111	215	hD7	o327	1101_0111
024	h18	o030	0001_1000	088	h58	o130	0101_1000	152	h98	o230	1001_1000	216	hD8	o330	1101_1000
025	h19	o031	0001_1001	089	h59	o131	0101_1001	153	h99	o231	1001_1001	217	hD9	o331	1101_1001
026	h1A	o032	0001_1010	090	h5A	o132	0101_1010	154	h9A	o232	1001_1010	218	hDA	o332	1101_1010
027	h1B	o033	0001_1011	091	h5B	o133	0101_1011	155	h9B	o233	1001_1011	219	hDB	o333	1101_1011
028	h1C	o034	0001_1100	092	h5C	o134	0101_1100	156	h9C	o234	1001_1100	220	hDC	o334	1101_1100
029	h1D	o035	0001_1101	093	h5D	o135	0101_1101	157	h9D	o235	1001_1101	221	hDD	o335	1101_1101
030	h1E	o036	0001_1110	094	h5E	o136	0101_1110	158	h9E	o236	1001_1110	222	hDE	o336	1101_1110
031	h1F	o037	0001_1111	095	h5F	o137	0101_1111	159	h9F	o237	1001_1111	223	hDF	o337	1101_1111
032	h20	o040	0010_0000	096	h60	o140	0110_0000	160	hA0	o240	1010_0000	224	hE0	o340	1110_0000
033	h21	o041	0010_0001	097	h61	o141	0110_0001	161	hA1	o241	1010_0001	225	hE1	o341	1110_0001
034	h22	o042	0010_0010	098	h62	o142	0110_0010	162	hA2	o242	1010_0010	226	hE2	o342	1110_0010
035	h23	o043	0010_0011	099	h63	o143	0110_0011	163	hA3	o243	1010_0011	227	hE3	o343	1110_0011
036	h24	o044	0010_0100	100	h64	o144	0110_0100	164	hA4	o244	1010_0100	228	hE4	o344	1110_0100
037	h25	o045	0010_0101	101	h65	o145	0110_0101	165	hA5	o245	1010_0101	229	hE5	o345	1110_0101
038	h26	o046	0010_0110	102	h66	o146	0110_0110	166	hA6	o246	1010_0110	230	hE6	o346	1110_0110
039	h27	o047	0010_0111	103	h67	o147	0110_0111	167	hA7	o247	1010_0111	231	hE7	o347	1110_0111
040	h28	o050	0010_1000	104	h68	o150	0110_1000	168	hA8	o250	1010_1000	232	hE8	o350	1110_1000
041	h29	o051	0010_1001	105	h69	o151	0110_1001	169	hA9	o251	1010_1001	233	hE9	o351	1110_1001
042	h2A	o052	0010_1010	106	h6A	o152	0110_1010	170	hAA	o252	1010_1010	234	hEA	o352	1110_1010
043	h2B	o053	0010_1011	107	h6B	o153	0110_1011	171	hAB	o253	1010_1011	235	hEB	o353	1110_1011
044	h2C	o054	0010_1100	108	h6C	o154	0110_1100	172	hAC	o254	1010_1100	236	hEC	o354	1110_1100
045	h2D	o055	0010_1101	109	h6D	o155	0110_1101	173	hAD	o255	1010_1101	237	hED	o355	1110_1101
046	h2E	o056	0010_1110	110	h6E	o156	0110_1110	174	hAE	o256	1010_1110	238	hEE	o356	1110_1110
047	h2F	o057	0010_1111	111	h6F	o157	0110_1111	175	hAF	o257	1010_1111	239	hEF	o357	1110_1111
048	h30	o060	0011_0000	112	h70	o160	0111_0000	176	hB0	o260	1011_0000	240	hF0	o360	1111_0000
049	h31	o061	0011_0001	113	h71	o161	0111_0001	177	hB1	o261	1011_0001	241	hF1	o361	1111_0001
050	h32	o062	0011_0010	114	h72	o162	0111_0010	178	hB2	o262	1011_0010	242	hF2	o362	1111_0010
051	h33	o063	0011_0011	115	h73	o163	0111_0011	179	hB3	o263	1011_0011	243	hF3	o363	1111_0011
052	h34	o064	0011_0100	116	h74	o164	0111_0100	180	hB4	o264	1011_0100	244	hF4	o364	1111_0100
053	h35	o065	0011_0101	117	h75	o165	0111_0101	181	hB5	o265	1011_0101	245	hF5	o365	1111_0101
054	h36	o066	0011_0110	118	h76	o166	0111_0110	182	hB6	o266	1011_0110	246	hF6	o366	1111_0110
055	h37	o067	0011_0111	119	h77	o167	0111_0111	183	hB7	o267	1011_0111	247	hF7	o367	1111_0111
056	h38	o070	0011_1000	120	h78	o170	0111_1000	184	hB8	o270	1011_1000	248	hF8	o370	1111_1000
057	h39	o071	0011_1001	121	h79	o171	0111_1001	185	hB9	o271	1011_1001	249	hF9	o371	1111_1001
058	h3A	o072	0011_1010	122	h7A	o172	0111_1010	186	hBA	o272	1011_1010	250	hFA	o372	1111_1010
059	h3B	o073	0011_1011	123	h7B	o173	0111_1011	187	hBB	o273	1011_1011	251	hFB	o373	1111_1011
060	h3C	o074	0011_1100	124	h7C	o174	0111_1100	188	hBC	o274	1011_1100	252	hFC	o374	1111_1100
061	h3D	o075	0011_1101	125	h7D	o175	0111_1101	189	hBD	o275	1011_1101	253	hFD	o375	1111_1101
062	h3E	o076	0011_1110	126	h7E	o176	0111_1110	190	hBE	o276	1011_1110	254	hFE	o376	1111_1110
063	h3F	o077	0011_1111	127	h7F	o177	0111_1111	191	hBF	o277	1011_1111	255	hFF	o377	1111_1111

2.7 Das Rechnen im Dualsystem. Zum Selbststudium!

Alle arithmetischen Operationen werden letztlich auf Additionen zurückgeführt, auch wenn ein eigener Befehl für die Subtraktion vorhanden ist, oder, wie bei neueren Mikroprozessoren, eigene Befehle für (ganzzahlige) Multiplikation oder Division vorliegen. Mathematische Funktionen wie z.B. Logarithmus oder Sinus werden durch Approximationen dargestellt, die nur die Grundrechnungsarten beinhalten.

Auch der Vergleich von 2 Zeichenketten (engl. string), wie z.B. 'Meier' und 'Huber' ist letztlich eine Addition der Bit–Muster des Zeichen 1 mit dem 2–Komplement des Zeichens 2. Ist das Ergebnis 0, dann sind beide Zeichen gleich. Der Vergleich ist für alle Zeichen der beiden Zeichenketten durchzuführen. Nur wenn alle "Vergleiche" das Ergebnis 0 haben, sind beide Zeichenketten gleich.

2.7.1 Addition

Arithmetische Regeln:

```
  0  +  0  =    0
  0  +  1  =    1
  1  +  0  =    1
  1  +  1  =  1 0
                └── Übertrag auf nächsthöhere Bit–Wertigkeit
```

1. Beispiel: 23 + 18= 41

```
23 dual: 10111     (1*16 + 0*8 + 1*4 + 1*2 + 1*1 = 23)
18 dual: 10010     (1*16 + 0*8 + 0*4 + 1*2 + 0*1 = 18)

       5  4  3  2  1  0     Bit–Nr
      32 16  8  4  2  1     Bit–Wertigkeit dezimal
      ─────────────────
          1  0  1  1  1     23 dual: 10111
      +   1  0  0  1  0     18 dual: 10010
      ─────────────────
       1     1  1           Übertrag
      ─────────────────
       1  0  1  0  0  1     Summe dual: 101001
      ==================
```

Probe: 1*32 + 0*16 + 1*8 + 0*4 + 0*2 * 1*1 = 41

2. Beispiel: 31 + 29 = 60

```
31 dual: 11111     (1*16 + 1*8 + 1*4 + 1*2 + 1*1 = 31)
29 dual: 11101     (1*16 + 1*8 + 1*4 + 0*2 + 1*1 = 29)

       5  4  3  2  1  0     Bit–Nr
      32 16  8  4  2  1     Bit–Wertigkeit dezimal
     ─────────────────────
          1  1  1  1  1     23 dual: 11111
     +    1  1  1  0  1     18 dual: 11101
     ─────────────────────
      1  1  1  1  1         Übertrag
     ─────────────────────
      1  1  1  1  0  0      Summe dual: 111100
     =====================

Probe: 1*32 + 1*16 + 1*8 + 1*4 + 0*2 * 0*1  =  60
```

2.7.2 Multiplikation

Arithmetische Regeln:

```
  0  *  0  =  0
  0  *  1  =  0
  1  *  0  =  0
  1  *  1  =  1
```

1. Beispiel: 15 * 14 = 210

```
15 dual: 1111     (1*8 + 1*4 + 1*2 + 1*1 = 15)
14 dual: 1110     (1*8 + 1*4 + 1*2 + 0*1 = 14)

        1  1  1  1    *    1  1  1  0
     ──────────────────────────────────
        1  1  1  1  ──────┘  |  |  |
           1  1  1  1  ─────┘  |  |
              1  1  1  1  ─────┘  |
                 0  0  0  0  ─────┘
     ──────────────────────────────────
        1  1  1                      2. Übertrag
     ──────────────────────────────────
     1  1  1  1  1                   1. Übertrag
     ──────────────────────────────────
     1  1  0  1  0  0  1  0          Ergebnis
     ================================

Probe: 1*128 + 1*64 + 0*32 + 1*16 + 0*8 + 0*4 + 1*2 + 0*1  =  210
```

2. Beispiel: 15 * 7 = 105

```
15 dual: 1111     (1*8 + 1*4 + 1*2 + 1*1 = 15)
 7 dual:  111     (      1*4 + 1*2 + 1*1 =  7)

        1  1  1  1    *    1  1  1
      ──────────────────────────────
        1  1  1  1  ──────┘  |  |
           1  1  1  1  ─────┘  |
              1  1  1  1  ─────┘
      ──────────────────────────────
        1  1  1                    2. Übertrag
      ──────────────────────────────
      1  1  1  1  1                1. Übertrag
      ──────────────────────────────
      1  1  0  1  0  0  1          Ergebnis
      ============================
```

Probe: `1*64 + 1*32 + 0*16 + 1*8 + 0*4 + 0*2 + 1*1 = 105`

Multiplikationen mit 2, 4, 8, 16 usw. werden rationeller durch Linksverschieben des Bit–Musters um 1 bit, 2 bit, 3 bit, 4 bit usw. ausgeführt. Rechts wird jeweils eine Null angehängt.

Beispiel: 7 * 4 = 28

```
        1 1 1  dual 7
    1 1 1 0 0  Mulitplikation mit 4: um 2 bit nach links
               verschieben
```

Probe: `1*16 + 1*8 + 1*4 + 0*2 + 0*1 = 28`

Die Kombination von Bit–Verschiebungen und Additionen ermöglicht rationelle Multiplikationen für beliebige Zahlen.

2.7.3 Subtraktion

Die Subtraktion wird üblicherweise durch Addition des sog. 2–Komplements der zu subtrahierenden Zahl (Subtrahend) zum Minuenden durchgeführt. Das 2–Komplement ist die Invertierung (Vertauschen von 0 und 1) des Subtrahenden), wobei nach der Invertierung noch eine 1 addiert werden muß. Vor dem Invertieren muß der (duale) Subtrahend mit führenden Nullen auf die gleiche Dualstellenzahl des Minuenden aufgefüllt werden. Wenn bei Addition des 2–Komplements ein End–Übertrag entsteht, dann wird dieser einfach ignoriert.

1. Beispiel: 9 – 3 = 6 (x – y = z)

```
Minuend    9 dual:  1001   (1*8 + 0*4 + 0*2 + 1*1  =  9)
Subtrahend 3 dual:    11   (           1*2 + 1*1  =  3)


1. Schritt: y auf gleiche Stellenzahl wie x auffüllen:  0011
2. Schritt: invertieren (0 und 1 vertauschen):          1100
3. Schritt: 1 addieren                                     1
                                                        ————
                                  2–Komplement von 3: 1101

4. Schritt: Zahl x:                 1001
            + 2–Komplement von 3:   1101
                                    ————
  (eventuellen) End–    ——————————  1  1  Übertrag
  Übertrag ignorieren               ————
                                    0110 Ergebnis
                                    ====
```

Probe: `0*8 + 1*4 + 1*2 + 0*1  =  6`

2. Beispiel: 243 – 27 = 216 (x – y = z)

```
243 dual: 1111 0011 (1*128+1*64+1*32+1*16+0*8+0*4+1*2+1*1  =  243)
 27 dual:    1 1011 (               1*16+1*8+0*4+1*2+1*1  =   27)

1. Schritt: y auf gleiche Stellenzahl wie x auffüllen:  0001 1011

2. Schritt: invertieren (0 und 1 vertauschen):          1110 0100
3. Schritt: 1 addieren                                         1
                                                        —————————
                                  2–Komplement von y: 1110 0101

4. Schritt: Zahl x:                 1111 0011
            + 2–Komplement von y:   1110 0101
                                    —————————
         (eventuellen) End– ———————  111   111  Übertrag
         Übertrag ignorieren        —————————
                                    1101 1000 Ergebnis
                                    =========
```

Probe: `1*128 + 1*64 + 0*32 + 1*16 + 1*8 + 0*4 + 0*2 + 0*1  =  216`

2.7.4 Division

Die Division (z = x/y) wird auf eine fortlaufende Subtraktion des Divisors y vom Dividenden x zurückgeführt.

Für die Subtraktion wird zweckmäßigerweise wieder das 2–Komplement des Divisors benutzt. Das Ergebnis einer 2–Komplement–Addition sei mit *Zwischenergebnis* bezeichnet.

Die Division ist dann beendet,

- wenn das Zwischenergebnis Null wird. In diesem Fall ist die Division ganzzahlig aufgegangen; es gibt keinen Divisonsrest. Das Ergebnis der Division ist die Summe aller End–Überträge.

oder

- wenn bei der Addition des 2–Komplements *kein* End–Übertrag mehr auftritt. Das Ergebnis der Division ist wie vorher die Summe aller End–Überträge. Das Ergebnis der letzten 2–Komplement–Addition, bei der es noch zu einem End–Übertrag kam, ist der Divisionsrest.

Beispiel: 9 / 3 = 3

```
9 dual:              1001
2–Komplement von 3: 1101  (siehe 1. Beispiel Subtraktion)

        1001 Dividend 9
      + 1101 2–Komplement des Divisors 3
       ─────
      1   1  Übertrag  (hier mit 1. End–Übertrag, links)
       ─────
        0110 Zwischenergebnis
       ─────
      + 1101 2–Komplement des Divisors
       ─────
      1 1    Übertrag  (hier mit 2. End–Übertrag, links)
       ─────
        0011 Zwischenergebnis
       ─────
      + 1101 2–Komplement des Divisors
       ─────
      1 111   Übertrag  (hier mit 3. End–Übertrag, links)
       ─────
        0000 Zwischenergebnis Null. Ende, kein Divisionsrest
       ─────

      Summation der End–Überträge:

        1   1. End–Übertrag
        1   2. End–Übertrag
        1   3. End–Übertrag
      ───
       1  Übertrag
      ───
       11 Ergebnis der Division 1*2 + 1*1 = 3,  9/3 = 3
      ===
```

Ähnlich wie bei der Multiplikation werden Divisionen durch 2, 4, 8, 16 usw. rationeller durch Bit–Verschiebungen um 1 bit, 2 bit, 3 bit, 4 bit usw. durchgeführt, allerdings nach rechts.

Beispiel: 16/4 = 4

16: 10000
um 2 bit nach rechts: 100 (1*4 + 2*2 + 1*0 = 4)

2.8 Die Zeichencodierung

Die Zeichencodierung wird ausführlicher im Kap. 13 behandelt. Hier wir lediglich der bei PCs unter DOS verwendete **Ascii-Zeichensatz** (Zeichen bis #127) und die **IBM-Erweiterung** (IBM-Zeichensatz, Codepage 437, Zeichen #128 bis #255) vorgestellt. Im Bereich zwischen 0 und 31 liegen Steuerzeichen. Details dazu in Kap. 13. Da der Zeichensatz 256 Zeichen umfaßt, genügt zur Speicherung eines Zeichens ein Byte (1 Byte = 8 bit, $2^8 = 256$).

Dez	Hex		Dez	Hex		Dez	Hex		Dez	Hex		Dez	Hex		Dez	Hex		Dez	Hex	
032	20		064	40	@	096	60	`	128	80	Ç	160	A0	á	192	C0	└	224	E0	α
033	21	!	065	41	A	097	61	a	129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
034	22	"	066	42	B	098	62	b	130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
035	23	#	067	43	C	099	63	c	131	83	â	163	A3	ú	195	C3	├	227	E3	π
036	24	$	068	44	D	100	64	d	132	84	ä	164	A4	ñ	196	C4	-	228	E4	Σ
037	25	%	069	45	E	101	65	e	133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
038	26	&	070	46	F	102	66	f	134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
039	27	'	071	47	G	103	67	g	135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
040	28	(	072	48	H	104	68	h	136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
041	29	)	073	49	I	105	69	i	137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Ø
042	2A	*	074	4A	J	106	6A	j	138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
043	2B	+	075	4B	K	107	6B	k	139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
044	2C	,	076	4C	L	108	6C	l	140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
045	2D	-	077	4D	M	109	6D	m	141	8D	ì	173	AD	¡	205	CD	═	237	ED	Ø
046	2E	.	078	4E	N	110	6E	n	142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
047	2F	/	079	4F	O	111	6F	o	143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
048	30	0	080	50	P	112	70	p	144	90	É	176	B0	░	208	CF	╨	240	F0	≡
049	31	1	081	51	Q	113	71	q	145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
050	32	2	082	52	R	114	72	r	146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
051	33	3	083	53	S	115	73	s	147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
052	34	4	084	54	T	116	74	t	148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
053	35	5	085	55	U	117	75	u	149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
054	36	6	086	56	V	118	76	v	150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
055	37	7	087	57	W	119	77	w	151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
056	38	8	088	58	X	120	78	x	152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
057	39	9	089	59	Y	121	79	y	153	99	Ö	185	B9	╣	217	D9	┘	249	F9	•
058	3A	:	090	5A	Z	122	7A	z	154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
059	3B	;	091	5B	[	123	7B	{	155	9B	¢	187	BB	╗	219	DB	▉	251	FB	√
060	3C	<	092	5C	\	124	7C	|	156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
061	3D	=	093	5D	]	125	7D	}	157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
062	3E	>	094	5E	^	126	7E	~	158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
063	3F	?	095	5F	_	127	7F		159	9F	f	191	BF	┐	223	DF	▀	255	FF	á

Neben der Codepage 437, die vorrangig in den USA und im englischsprachigen Kanada verwendet wird, ist vor allem in Europa die Codepage 850 häufig im Gebrauch. Die

Unterschiede liegen in etwas mehr als 40 Zeichen im Codebreich >= 128. Die speziellen deutschen Zeichen (Umlaute, Scharf-S) befinden sich aber an den gleichen Positionen.

Windows und Apple **Macintosh** verwenden nicht den Ascii-Zeichensatz mit der IBM-Erweiterung, sondern den **Ansi-Zeichensatz**, der im Codebereich von #0 bis #127 identisch ist mit dem Ascii-Zeichensatz, darüber hinaus aber völlig verschieden ist von der IBM-Erweiterung. Das jedem DR-Studierenden ausgehändigte gelbe Arbeitsblatt enthält den Ascii-Zeichensatz, die IBM-Erweiterung für Codepage 437 und den Ansii-Zeichensatz.

Der **Unicode** verwendet für das Speichern eines Zeichens 2 Byte. Damit sind 2^{16} = 65536 verschiedene Zeichen codierbar. Die ersten 128 Zeichen sind identisch mit dem Ascii-Zeichensatz.

2.9 Mikroprozessoren

In diesem Kapitel werden die wichtigsten technischen Merkmale der Mikroprozessoren vorgestellt, die bei Personal Computern Verwendung finden.

Unter dieser Einschränkung sind praktisch nur zwei Hersteller zu nennen:

- **Intel**, USA
- **Motorola**, USA

Nachbauten (Clones) der Intel-Prozessoren werden von anderen Halbleiterherstellern auf den Markt gebracht. Im Jahre 2000 ist nur noch erwähnenswert:

- **AMD**, USA (Advanced Micro Devices)

In den folgenden Ausführungen sind die Original–Mikroprozessoren der beiden eingangs aufgeführten Hersteller angesprochen.

Neben dem Befehlsumfang der Prozessoren ist die Registerbreite und die Taktfrequenz für die Leistung maßgebend.

Register sind schnelle Speicherzellen auf dem Chip des Mikroprozessors für besondere Aufgaben. Die Register sind entweder 8 bit (frühere Home Computer), 16 bit oder 32 bit breit. Dementsprechend spricht man von 8–Bit–, 16–Bit– und 32–Bit–Mikroprozessoren.

32–Bit–Prozessoren sind 1997 Standard bei Personal Computern, bei Intel die Pentium-Prozessoren.

Die Geschwindigkeit eines Mikroprozessors ist unter sonst gleichen Bedingungen proportional der Taktfrequenz. Die Spannweite reicht von 4,77 MHz (erster IBM–PC

1981), über 8, 10, 12, 16, 20, 33, 60, 90, 120, 133, 166 MHz bis zur derzeitigen Spitze von 200 MHz (Pentuim Pro) bzw. 250 MHz (PowerPC).

Mikroprozessoren können standardmäßig numerische Berechnungen nur mit Ganzzahlen (Integer) durchführen. Gleitkommaberechnungen müssen software-mäßig realisiert werden, was sehr zeitintensiv ist. Mit mathematischen **Koprozessoren** (Arithmetik-Prozessoren, numeric processor) können derartige Berechnungen hardware-mäßig durchgeführt werden. Bezogen auf Einzel-Operationen können durchaus Leistungssteigerungen um den Faktor 100 auftreten. Arithmetik-Prozessoren sind relativ teuer; ihr Einsatz lohnt sich aber bei Aufgaben wie: CAD (Computer Aided Design), Bildtransformationen, Tabellenkalkulation, FEM (Finite Elemente Methode) u.ä. Bei Textverarbeitung bringt ein Arithmetik-Prozessor verständlicherweise nichts. Vor dem Kauf ist zu prüfen, ob das in Frage stehende Programm den Arithmetik-Prozessor benutzt. Ab Turbo-Pascal 5.0 kann der Arithmetik-Prozessor mit dem Compiler-Befehl {$N+} (Vorgriff) benutzt werden. Ab dem Typ 80486 DX und somit auch bei den Pentiums ist bei Intel-Prozessoren der Arithmetik-Prozessor integriet, ebenso wie beim PowerPC der neueren Apple Macintosh.

2.9.1 Die Intel-Mikroprozessoren der Baureihe iAPx86 und Pentium

Die Baureihe besteht aus den Grundtypen 8088, 8086, 80286, 80386, 80486 und 80586. Letztere wurde aus rechtlichen Gründen in "p5" und später in "Pentium" umbenannt. Die Intel–Arithmetik–Prozessoren haben (hatten) die Bezeichnungen 8087, 80287 und 80387.

Der Pentium wurde auf der CeBIT '93 erstmals in Deutschland vorgestellt. Die ersten Versionen wurden mit 60 MHz getaktet. Ebenfalls auf CeBIT '93 wurden von Digital Equipment (DEC) der Mikroprozessor Alpha AXP vorgestellt (64 Bit, 200 MHz).

- Der **8086** (1978, veraltet) ist der "Stammvater" der Intel–Baureihe iAPx86. Er ist ein "echter" 16–Bit–Prozessor, d.h. er besitzt nicht nur 16–Bit–breite Register, sondern auch einen 16–bit breiten Daten–Bus (Bus: Sammelleitung). Somit kann ein 16-Bit-Wort in einem Zyklus vom Arbeitsspeicher in ein Register geladen werden oder umgekehrt. Der 8086 besitzt 20 Adreßleitungen und kann damit 2^{20} = 1 048 576 Byte = 1024 KByte = 1 MByte adressieren. Durch die nur 16-Bit breiten Register war aber ein Kunstgriff vonnöten, um die 20-Bit-Speicheradressen "bedienen" zu können. Siehe spätere Ausführungen zu "Segmentierter Adressierung". Durch die Vorgaben des Betriebssystems MS-DOS stehen für den Anwender nur die ersten 640 KByte zur Verfügung, wobei aber noch ein erheblicher Teil für die Teil des Betriebssystems abgehen, die in den RAM geladen werden müssen. Der Chip des 8086 enthielt über 27 000 Transistorfunktionen. **Der 8086 dient nur noch als Referenz**.

- Der **8088** (veraltet) ist ein Zwitter. Er besitzt die gleichen 16–bit–Register wie der 8086, arbeitet intern also mit 16 bit, besitzt aber nur einen 8–bit–DatenBus. Beim Laden oder Abspeichern eines 16–bit–Wortes benötigt der 8088 somit zwei Speicherzugriffe. Unter sonst gleichen Bedingungen ist der 8088 deshalb etwas langsamer als der 8086. Der erste IBM–PC und auch die vielen Klones waren aus Kostengründen mit einem 8088 ausgestattet und nicht mit dem 8086.

- Der **80286** (veraltet) ist ebenfalls noch ein 16–Bit–Prozessor, d.h. er besitzt Register und Datenbus mit 16 bit.. Der Adressbus ist aber 24 bit breit. Somit ist ein Speicher von maximal 2^{24} = 16 777 216 Byte = 16 MByte adressierbar. Dieser Speicherbereich kann nur durch Modifikation des Betriebssystems genutzt werden. Aus diesen Zeiten stammen noch die Begriffe **Real-Mode** und **Protected-Mode**. Nur im Protected-Mode konnte der zusätzliche Speicherbereich genutzt werden, wenn auch nur in max. 64 KByte großen Blöcken. Auf eine weitere Behandlung wird in der vorliegenden Version des Skriptums verzichtet. Es leben Pentium & Co.!

- Der **80386** (1986, veraltet) und der **80486** (1990, veraltet) sind 32-Bit-Prozessoren. Sie besitzen 32-Bit-Register (bis auf die Segmentregister), 32-Bit-Datenbus und 32 Adreßleitungen. Damit ist ein Speicherbereich von max. 2^{32} = 4 294 967 296 = 4 GByte adressierbar. Bei DOS-Anwendungen ist man aber aus Kompatibiltätsgründen auf den Real-Mode oder Protected-Mode beschränkt. Erst mit Windows wird die Beschränkung aufgehoben. Der 80486 stand in den Versionen 80486 SX (ohne integrierten Koprozessor) und 80486 DX (erster Intel-Prozessor mit integriertem Koprozessor) zur Verfügung. Der Chip des Prozessor 80386 enthielt über 275 000 Transistorfunktionen, der 80486 DX über 1,25 Mio.

- Der **Pentium** (1997 bei MS-DOS- und Windows-Rechnern Standard) ist ein vollständiger 32-Bit-Prozessor, besitzt aber zum Teil 64-Bit-Strukturen. Der Koprozessor ist integriert und auch ein Cache-Speicher auf dem Chip selbst. Cache-Speicher besitzen sehr schnelle Speicherbausteine (statische RAMs) und dienen zum Zwischenspeichern von unmittelbar benötigten Befehlen und Daten. Die Taktfrequenzen reichen bis 200 MHz. Weiterentwicklungen sind der **Pentium Pro** und der **Pentium MMX** (Multimedia Extensions, mit 57 neuen Befehlen). Der Chip des Pentium Pro enthält über 5,27 Mio Transistorfunktionen (Zur Erinnerung: Beim 8086 waren es 27 000 und beim 80486 "nur" 1,25 Mio). Auch der Pentium kann für DOS-Anwendungen im Real-Mode oder Protected-Mode betrieben werden. Sinnvolle Anwendungen setzen aber Windows voraus.

Ein Relikt aus der Intel-Steinzeit: Die segmentierte Adressierung

Ein Merkmal aller Intel–Prozessoren bis einschließlich Typ 80286 ist die ausschließlich **segmentierte Adressierung**. Eine physikalische Speicheradresse (oft auch physische Adresse genannt) setzt sich zusammen aus der Segment–Adresse und der Offset-Adresse. Segmente können bis zu 64 KByte groß sein und müssen bei einem Vielfachen (n = 0, 1, 2, ..) von 16 beginnen. Der Offset ist die Relativ–Adresse zum Segment-

beginn. In der üblichen Hex–Schreibweise der Speicheradresse ist die letzte Ziffer des Segments durch das Vielfache von 16 immer eine (hexadezimale) 0. Man schreibt sie vereinbarungsgemäß nicht an. Ansonsten wird die Speicheradresse im typischen Format ***Segment:Offset*** angegeben, wie folgendes Beispiel zeigt:

```
2D0A:F705
```

Segment: `$2D0A`
Offset: `$F705`

Bei der Bildung der physikalischen Adresse schiebt der Prozessor die (binäre) Segmentadresse um 4 bit nach links, was einer Multiplikation mit 16 entspricht und addiert dann die Offset–Adresse.

Die Rechnung in hex:

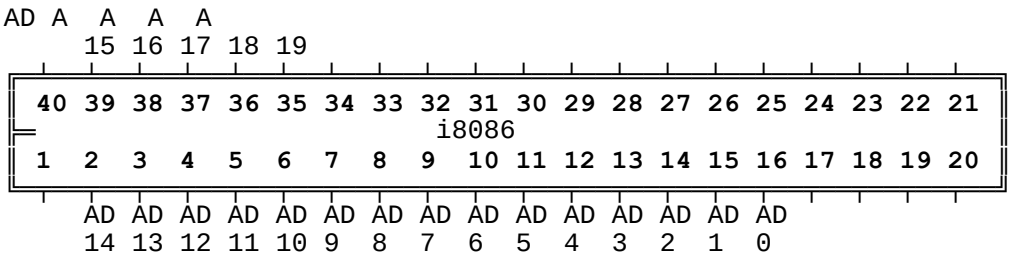
```
Segment * 16:     2D0A0        '0 anhängen, Multiplikation mit 16
+ Offset:          F705
                  _____
physik. Adresse = 3C7A5

dezimal:  3*65536 + 12*4096 + 7*256 + 10*16 + 5 = 247á717
```

Für die Verwaltung der Segment–Adressen verfügen die Intel-Prozessoren über spezielle Register (*CS* für Code–Segment, *DS* für Daten–Segment, *ES* für Extra–Segment und *SS* für Stack–Segment). Obwohl die Register nur 16 bit breit sind, kann der Prozessor durch den Trick der Bit–Verschiebung um 4 bit und der anschließenden Offset-Addition eine 20–Bit–Adresse entsprechend den 20 Adreß-Leitungen berechnen und somit maximal 2^{20} = 1 048 576 Byte = 1 MByte adressieren. Der 80286 besitzt aber 24 Adreß–Leitungen, die im Protected–Mode auch alle genutzt werden. Durch ziemlich trickreiche Maßnahmen (Umdefinition bzw. Aufteilung der Segmentregister in drei Felder, darunter einen Zeiger auf eine Tabelle im Arbeitsspeicher) gelingt es schließlich, die 24 Adreß-Leitungen und den damit möglichen Speicherbereich des 80286 von bis zu 16 MByte zu nutzen.

Das Gegenteil der *segmentierten Adressierung* ist die **lineare Adressierung**, die bei Mikroprozessoren anderer Hersteller üblich und optional auch ab dem Intel 80386 möglich ist. Wie die Bezeichnung vermuten läßt, werden bei der linearen Adressierung die Speicherzellen linear von 0 bis zum Höchstwert durchgezählt.

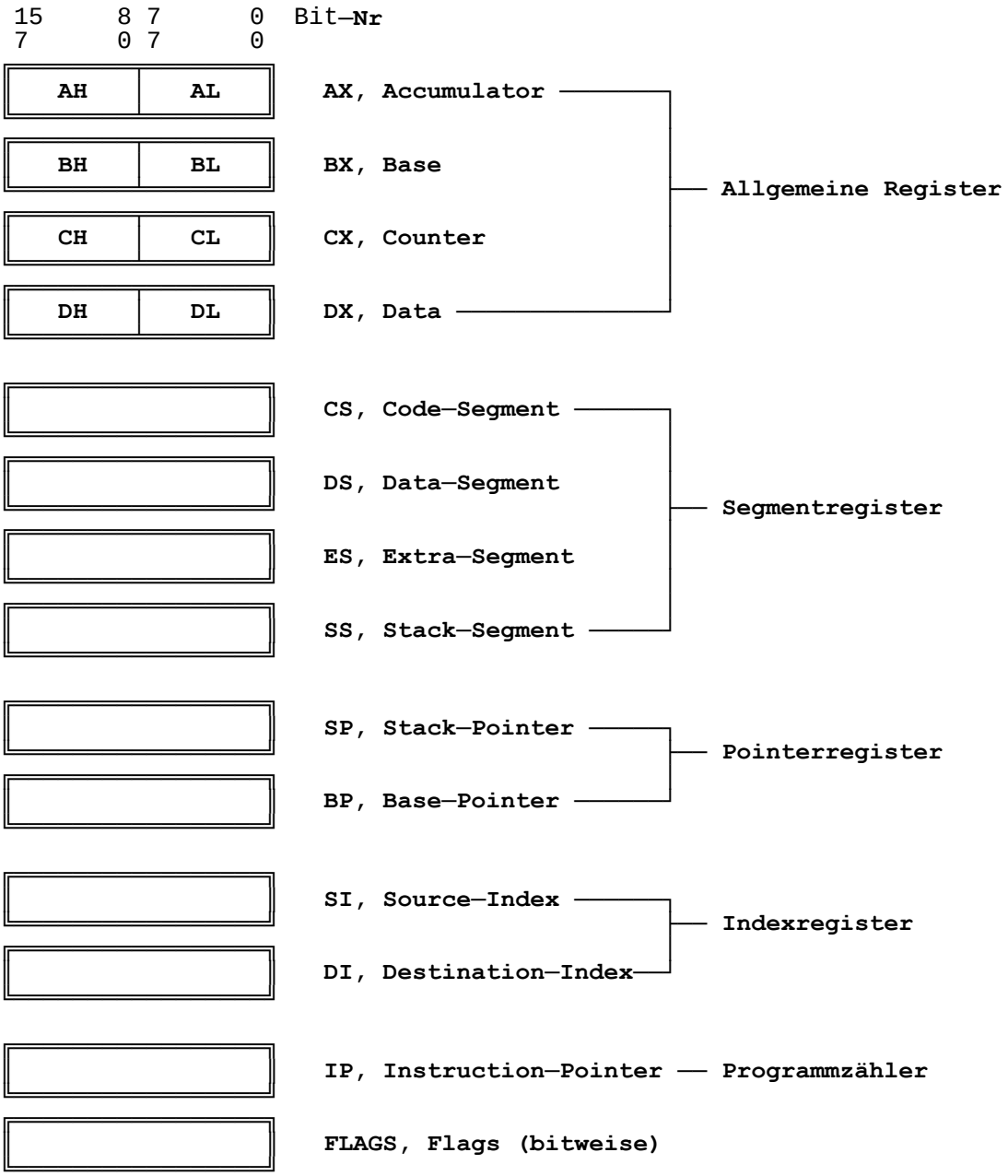
2.9.2 Immer noch die Referenz für DOS-Anwendungen: Der Intel 8086



Der 8086 besitzt 40 Anschlüsse (Pins). In der Zeichnung sind lediglich die 20 Adreßleitungen (A0 bis A19) und die 16 Datenleitungen (D0 bis D15) eingetragen. Man sieht, daß bei diesem Prozessor 16 Leitungen gemeinsam für Daten und Adressen benutzt werden. Durch Mulitplexen wird zwischen Datenleitung und Adreßleitung hin– und hergeschaltet. Der 80286 und der 80386 besitzen dagegen getrennte Leitungen für Adressen und Daten.

Der zum 8086 passende Koprozessor von Intel hat die Bezeichnung i8087.

Die Register des Intel 8086:



a) Die allgemeinen Register AX, BX, CX und DX

Die Register AX, BX, CX und DX sind allgemeine 16-Bit-Register. Sie können aber auch byteweise angesprochen werden. Beispiel: AL und AH. AL steht für Low–Byte

Register A, und AH für High–Byte Register A. Alle vier allgemeinen Register können als Akkumulatoren dienen.

b) Die Segmentregister

Die Segmentregister dienen zum Speichern der Segmentadressen

- Register CS: Segmentadresse des (Programm–) Codes
- Register DS: Segmentadresse der Daten
- Register ES: Extra–Segment. Für bestimmte String–Operationen
- Register SS: Segmentadresse des Stack–Pointers

c) Die Pointerregister

- Register SP: StackÄPointer. Offset bezüglich des StackÄSegmentregisters SS
- Register BP: BaseÄPointer. Offset bezüglich des Registers SS

d) Die Indexregister

- Register SI: Source–Index. Für bestimmte String–Operationen, z.B. Verschiebung von 64–KByte–Blöcken beliebigen Inhalts im gesamten Speicherbereich.
- Register DI: Destination–Index. Siehe Register SI.

e) Der Programmzähler IP

Offset bezüglich des Code–Segmentregisters CS. Immer wenn ein Befehlsbyte gelesen wir, wird IP um 1 erhöht, ausgenommen bei Sprungbefehlen, bei denen die Zieladresse geladen wird. In IP steht somit immer die (Offset–)Adresse des nächsten Befehls.

f) Das Flag–Register (Statusregister)

Beim Flag–Register hat jedes einzelne Bit eine bestimmte Bedeutung. Das Bit Nr. 6 (ZF, zero flag) zeigt z.B. an, wenn das Ergebnis einer arithmetischen oder logischen Operation Null ist. In diesem Fall wird ZF auf 1 gesetzt, im anderem Falle auf 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit–Nr
.	.	.	.	OF	DF	IF	TF	SF	ZF	.	AF	.	PF	.	CF	

Die Bits 1, 3, 5, 12, 13, 14 und 15 des Flag–Registers werden nicht benutzt.

Nr	Name	Zweck	DEBUG–Anzeige gesetzt	DEBUG–Anzeige gelöscht
0	CF	Carry Flag, Übertrag	CY	NC
2	PF	Parity Flag, Parität	PE	PO
4	AF	Auxiliary Carry Flag, Hilfsübertrag	AC	NA
6	ZF	Zero Flag, Null	ZR	NZ
7	SF	Sign Sign, Vorzeichen	NG	PL
8	TF	Trap Flag, Einzelschritt		
9	IF	Interrupt Flag, Unterbrechung	EI	DI
10	DF	Direction Flag, Richtung	DN	UP
11	OF	Overflow Flag, Überlauf	OV	NV

Das **Carry Flag CF** wird gesetzt, wenn bei einer Operation ein Übertrag über das höchstwertige Bit hinaus erfolgt (z.B. bei einer Addition) oder wenn bei einer Subtraktion "geborgt" werden muß. Anderenfalls wird CF gelöscht.

Das **Parity Flag PF** wird gesetzt, wenn die Anzahl der gesetzten Bits im Low–Byte des Ergebnisses geradzahlig ist; anderenfalls wird PF gelöscht.

Das **Auxiliary Carry Flag AF** wird bei BCD–Arithmetik verwendet, bei der jede Dezimalziffer durch ein Halbbyte dargestellt wird. AF wird gesetzt, wenn bei einer Operation ein Übertrag über Bit 3 hinaus erfolgt oder entsprechend "geborgt" werden muß; anderenfalls wird AF gelöscht.

Das **Zero Flag ZF** wird gesetzt, wenn das Ergebnis einer Operation Null ist; anderenfalls wird ZF Bit gelöscht.

Das **Sign Flag SF** wird gesetzt, wenn das höchste Bit des Ergebnisses einer Operation gesetzt ist; anderenfalls wird SF gelöscht. SF ist somit eine Kopie des Vorzeichens des Ergebnisses.

Das **Trap Flag TF** wird in Verbindung mit dem MS–DOS–Programm DEBUG oder ähnlichen Programmen zum Test benutzt. Ist TF gesetzt, dann wird nach jedem Programmschritt ein (Software–) Interrupt ausgelöst.

Das **Interrupt Flag IF** gibt an, ob Hardware–Interrupts zugelassen sind oder nicht. Wird IF gesetzt, dann werden Interrupt–Anforderungen von externen Geräten erkannt und bearbeitet.

Das **Direction Flag DF** wird bei der Verarbeitung von Zeichenketten (strings) verwendet. Wird DF gesetzt, dann werden Strings mit absteigender Adresse verarbeitet, sonst mit aufsteigender Adresse.

Das **Overflow Flag OF** wird gesetzt, wenn nach einer Operation ein Übertrag in des Vorzeichen–Bit hinein oder ein "Borgen" aus diesem heraus erfolgt; anderenfalls wird OF gelöscht. Bei vorzeichenloser Arithmetik kann OF ignoriert werden.

2.9.3 Wie geht es weiter?

Im Jahr 2000 werden die Spitzenprozessoren von Intel und AMD die "Taktfrequenz-Schallmauer" von 1 GHz durchbrechen.

2.9.4 CISC und RISC

Die bislang besprochenen Prozessoren gehören der Kategorie **CISC** an. Die Abkürzung steht für: Complex Instruction Set Computer. Diese Prozessoren besitzen viele und komplexe Befehle. Untersuchungen haben aber gezeigt, daß viele Befehle selten oder fast gar nicht benutzt werden. Intel erwähnt aber, daß der Pentium eine Mischung aus CISC und RISC sei.

Prozessoren der **RISC**-Kategorie (Reduced Instruction Set Computer) besitzen dagegen weniger, dafür aber schneller ausführbare Befehle. Die seltener vorkommenden komplexeren Operationen werden durch mehrere einfache Befehle nachgestellt. Insgesamt soll die RISC-Technik eine wesentliche Reduzierung der Ausführungszeiten bringen.

Typische RISC-Prozessoren sind:

- **PowerPC** (**P**erformance **O**ptimization **w**ith **E**nhanced **P**rocessor **C**hip), entwickelt von Motorola, IBM und Apple. Wird von Motorola gebaut und u.a. beim Apple Macintosh eingesetzt.
- **AlphaPC**, von Digital Equipment (DEC). Einsatz in Workstations
- **R4000**, **R8000**, **R10000** von MIPS (zu Silicon Graphics gehörig). Einsatz in Workstations, z.B. Indigo und Indy.
- **SPARC**, **UltraSPARC** (**S**caleable **P**rocessor **Arc**hitecture), entwickelt von Sun. Bau durch verschiedene Halbleiterhersteller, u.a. Texas Instruments. Einsatz in Workstations, z.B. in Sun Workstations.