# Mikrorechner-technik

Die Intel Mikroprozessor-Familie
80286 - 80386 - 80486

von
Professor Dr. Heinrich Weber
Fachhochschule für Technik Esslingen

R. Oldenbourg Verlag München Wien 1994

**Prof. Dr. rer. nat. Heinrich Weber** gehört dem Fachbereich Technische Informatik der Fachhochschule für Technik Esslingen an. Er arbeitete nach dem Studium der Elektrotechnik in Saarbrücken (Dipl.-Ing.) und der Informatik in Karlsruhe (Dipl. Inform.) von 1985 bis 1991 am Forschungszentrum Informatik (FZI) und der Universität in Karlsruhe.

**Die Deutsche Bibliothek – CIP-Einheitsaufnahme**

**Weber, Heinrich:**
Mikrorechnertechnik : die Intel-Mikroprozessor-Familie 80286 – 80386 – 80486 / von Heinrich Weber. – München ; Wien : Oldenbourg, 1994
ISBN 3-486-22628-2

© 1994 R. Oldenbourg Verlag GmbH, München

Das Werk einschließlich aller Abbildungen ist urheberrechtlich geschützt. Jede Verwertung außerhalb der Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Bearbeitung in elektronischen Systemen.

Gesamtherstellung: R. Oldenbourg Graphische Betriebe GmbH, München

ISBN 3-486-22628-2

# Inhaltsverzeichnis


**1. Grundlagen** 9
1.1. Von Neumann-Architektur 9
1.2. Die Realisierung von Rechenwerken 16
1.3. Logik außerhalb der CPU 18
1.4. Mikrorechnerfamilien 21

**2. Hardware des Mikroprozessors 80286** 25
2.1. Einleitung 25
2.2. Charakteristische Kennzeichen des 80286 27
2.3. Komponenten der CPU 28
2.3.1. Befehls-Pipelining 30
2.3.2. Die Register des 80286 31
2.4. Der Stapelspeicher 36
2.5. Anschluß des Arbeitsspeichers 38
2.5.1. Der Bussteuerbaustein 82288 44
2.6. Memory Mapped- und Isolated I/O 47
2.7. Erzeugung von Takt- und Reset-Signalen 49
2.8. Interrupts 51
2.8.1. Interrupts an INTR 53
2.8.2. Interrupts an NMI 56
2.8.3. Software-Interrupts und Exceptions 57
2.9. Direct Memory Access 60
2.10. Anschluß des Koprozessors 62

**3. Befehlssatz und Programmierung im Real Mode** 67
3.1. Assembler-Notation 70
3.1.1. Allgemeine Assembler-Notation 73
3.1.2. Einfache Assemblerprogramme unter MS-DOS 74
3.2. Befehlsformate 77
3.2.1. Adressierungsarten 77
3.2.2. Aufbau von Maschineninstruktionen 80
3.2.3. Die Zustandsregister 85
3.2.4. Befehlsausführungszeiten 85
3.3. Assemblerbefehle für den Real Mode 86
3.3.1. Transportbefehle 86
3.3.2. Stack-Operationen 88
3.3.3. Ein-/Ausgabeoperationen 90
3.3.4. Arithmetische Operationen 91
3.3.5. Logische Operationen 96
3.3.6. Shift- und Rotate-Befehle 98
3.3.7. Programmverzweigungsbefehle 102
3.3.8. Flagregister 108
3.3.9. Typkonvertierung 109
3.3.10. BCD-Arithmetik 109
3.3.11. String-Befehle und REP-Zusätze 113
3.3.12. Prozessor-Steuerungsbefehle 116
3.4. Assembler-Steueranweisungen 118
3.4.1. Reservierung von Speicherplatz 118

3.4.2. Bedingte Übersetzung 121
3.4.3. Operatoren und Ausdrücke 122
3.4.4. Makros 124
3.4.5. Include 127
3.4.6. Segment 127
3.4.7. Assume 128
3.4.8. Getrennte Übersetzung 129

**4. Der PC als Software-Entwicklungssystem 133**
4.1. Der MS-DOS Standardmonitor und Debugger DEBUG 133
4.1.1. Zusammenfassung der DEBUG-Befehle 137
4.1.2. Programmerstellung per DOS-DEBUG 138
4.2. Der Einsatz von Assembler und Linker 139

**5. Peripheriebausteine 147**
5.1. Der Anschluß von Ein-/Ausgabebausteinen 147
5.2. Paralleler Schnittstellenbaustein 8255 150
5.2.1. Betriebsarten des 8255 152
5.2.2. Betriebsart 0 (Einfache Ein- /Ausgabe) 153
5.2.3. Betriebsart 1 (getastete Ein-/Ausgabe) 154
5.2.4. Betriebsart 2 (getastete bidirektionale Ein-/Ausgabe) 159
5.3. Serielle Ein-/Ausgabe über V.24 Schnittstelle 159
5.3.1. Serieller Schnittstellenbaustein 8251A 162
5.4. Interrupt-Steuerbausteine 171
5.4.1. Der Interrupt-Controllers 8259A 171
5.4.2. Programmierung des Interrupt-Controllers 8259A 172

**6. Der Mikroprozessor 80486 181**
6.1. Übersicht und Anschlußbelegung 181
6.1.1. Allgemeine Daten 181
6.1.2. Anschlußbelegung 183
6.2. Die Registerstruktur des 80486/80386 187
6.2.1. Allgemeine Register 187
6.2.2. Segmentregister 189
6.2.3. Steuerregister 189
6.2.4. Debug-Register 191
6.2.5. Testregister 193
6.2.6. Systemadreßregister 194
6.3. Befehle und Befehlsformate 195
6.4. Betriebs- und Adressierungsarten 198
6.4.1. Betriebsarten 198
6.4.2. Adressierungsarten 199
6.5. Der Virtual Protected Mode 199
6.5.1. Segmente und Pages 199
6.5.2. Segmente 200
6.5.3. Der Protection Mechanismus 212
6.5.4. Paging 220
6.6. Der Pufferspeicher 227
6.6.1. Allgemeines 227
6.6.2. Der Cache-Speicher des 80486 228
6.6.3. Pufferspeicher-Strategie 229
6.6.4. Anschlüsse des Pufferspeichers 230

**7. Anhang** **233**

**8. Literaturverzeichnis** **237**

**9. Index** **239**

# 1. Grundlagen

## 1.1. Von Neumann-Architektur

Trotz der heute vorhandenen großen Zahl von unterschiedlichen Rechnermodellen, angefangen vom Taschenrechner über den Personalcomputer bis hin zum Großrechner, arbeitet die weit überwiegende Zahl nach dem gleichen Prinzip und verfügt über die gleiche Architektur. Diese ist nach ihrem Erfinder Johann von Neumann[1] (Neumann-Architektur) benannt.

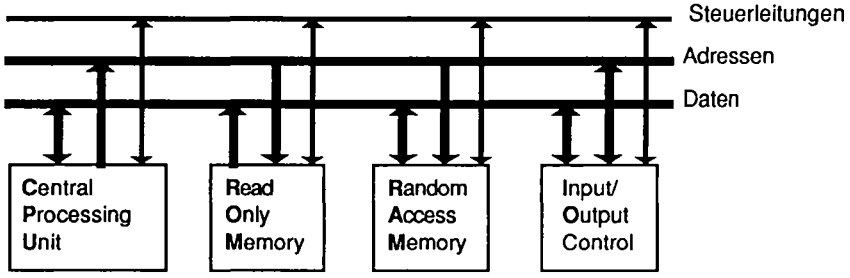



Abb. 1.1-1: Von Neumann-Rechner

Die grundlegenden Bestandteile dieser Rechnerarchitektur sind der Prozessor, verschiedene Arten von Speicherelementen sowie Elemente zur Ein- und Ausgabe von Daten an die Umgebung des Rechners. Die Elemente kommunizieren hierbei über vielpolige Leitungen (sog. Busse) miteinander. Das zentrale Element in dieser Anordnung ist der Prozessor (**CPU**, für central processing unit). In modernen Mikrorechnersystemen sind heute oftmals mehrere Prozessoren, teilweise mit ganz speziellen Aufgaben, im System vorhanden. Typische Beispiele hierfür sind die FPU (Floating Point Unit), der GDP (Graphic Display Processor) oder der DMA-Controller (Direct Memory Access).

Ein klassischer Prozessor besteht aus einem

- **Rechenwerk** (**ALU**, für Arithmetic Logic Unit), einem
- Satz von **Registern** (**RU**, für Register Unit) und einem
- **Steuerwerk** (**CU**, für Control Unit).

Das Rechenwerk übernimmt die Durchführung von arithmetischen (+,-, *, /), logischen (AND, OR, XOR, ...) Verknüpfungen sowie unterschiedlichen Shift-Operationen. Diese Operationen werden mit Wörtern der Länge n, wobei n die Wortbreite des Prozessors bezeichnet, durchgeführt (z.B. 8-Bit-Prozessor).

---

[1] Johann von Neumann: * 1903 in Budapest, † 1957 in Washington. Der erste von Neumann-Rechner "EDEVAC" wurde 1949 von M.V. Wilkes in England (erst 1952 in den USA) fertiggestellt.

Innerhalb des Registerblocks befinden sich Register, welche Operanden aufnehmen können (häufig mehrere, mindestens jedoch ein Akkumulator), und Register, welche Adressen aufnehmen können (Datenzeiger, oftmals auch Datenzähler genannt). Letztere sind in der Regel breiter und erlauben somit die Adressierung eines größeren Speicherbereiches.

Das Steuerwerk (auch Leitwerk genannt) ist ein synchrones Schaltwerk (taktgesteuert!), welches die Abarbeitung eines Befehls steuert. Die Generierung der einzelnen Steuersignale über ein oder mehrere Taktphasen hinweg wird in der Regel durch ein Schaltwerk oder per Mikroprogrammierung vorgenommen. Sie dienen zum Öffnen und Schließen von CPU-internen Datenpfaden sowie zur Ausgabe von Steuersignalen an die Logik außerhalb des Prozessors.

Während die oben aufgeführten Einheiten (evtl. in unterschiedlicher Form) mindestens vorhanden sein müssen, haben moderne Mikroprozessoren häufig noch weitere Einheiten, welche z.B. die Verarbeitungsgeschwindigkeit erhöhen (prefetch queue, Cache Speicher, paging unit...) oder Möglichkeiten zur ereignisgesteuerten Unterbrechung des laufenden Programms realisieren. Letzteres ist insbesondere für die Prozeßdatenverarbeitung von großer Bedeutung.

Eine CPU hat damit z.B. den nachfolgenden prinzipiellen Aufbau:

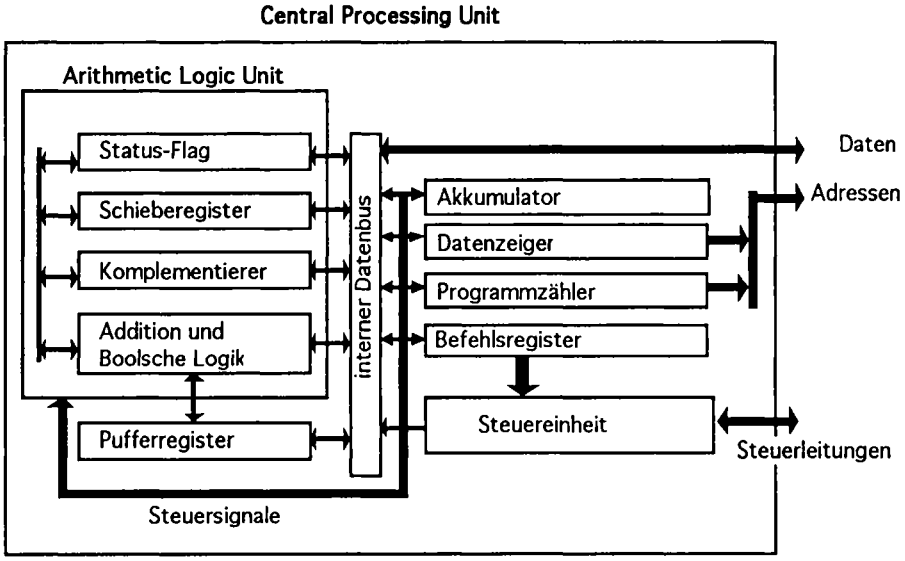



Abb. 1.1-2: Interne Struktur eines Prozessors

Sie kommuniziert über Adreß-, Daten- und Steuerleitungen mit ihrer Umgebung.

Als Bauelement der Elektronik ergibt sich physikalisch für eine CPU z.B. die folgende Struktur:

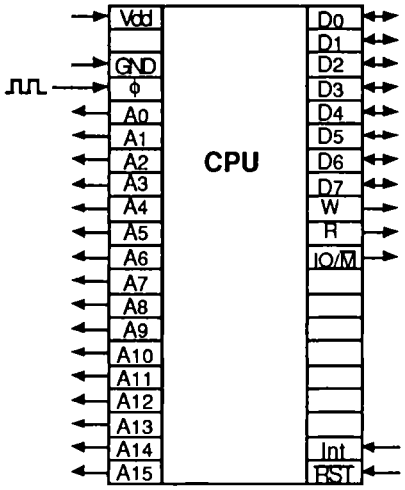



Abb. 1.1-3: Realisierung als Standardbauelement

Hierbei richtet sich die Anzahl der Anschlüsse nach der Komplexität der CPU und reicht von 40 bis zu 168 (z.B. beim 80486). Ein solcher Rechner bzw. die in ihm enthaltene CPU arbeitet dabei grundsätzlich nach folgendem Prinzip:

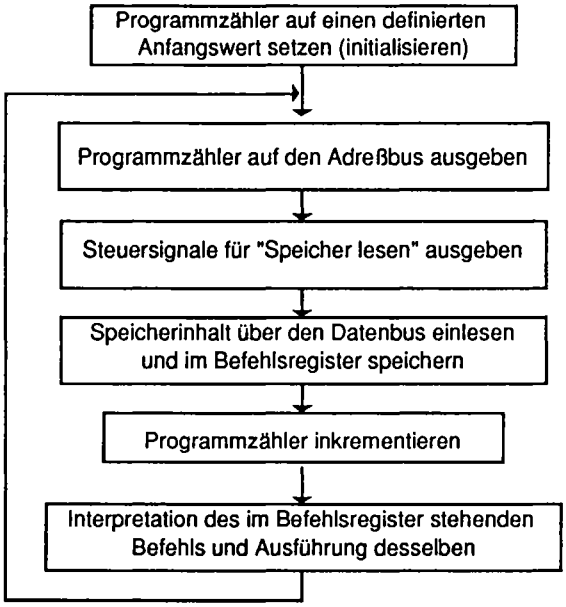



Abb. 1.1-4: Arbeitsweise des von Neumann-Rechners

Um die Arbeitsweise zu demonstrieren, betrachte man nachfolgendes Programmstück sowie seine Abarbeitung durch den Prozessor[2]:

Beispiel:

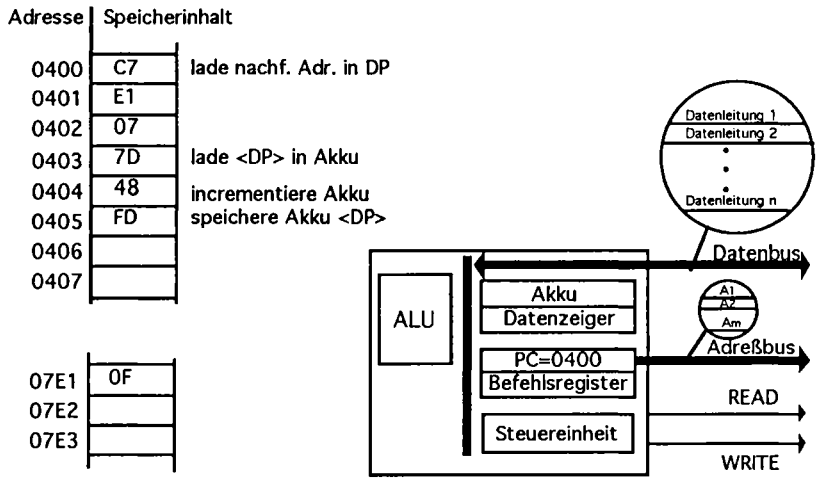



Abb. 1.1-5: Befehlsabarbeitung eines Rechners

Für das folgende Beispiel wird vorausgesetzt, daß der Programmzähler (Program Counter, PC oder auch Instruction Pointer, IP) aktuell den Wert 0400H beinhaltet, d.h. von dieser Adresse im Speicher den nächsten auszuführenden Befehl einliest.

**Op-Code=C7: Lade die nachfolgende Adresse in den Datenzeiger**

1 Programmzähler auf Adreßbus legen
2 READ-Steuersignal auf "1" legen
→ jetzt wird der Speicher veranlaßt, sein Datum (=C7) auf den Datenbus auszugeben
3 Datum in das Befehlsregister (command register, CR) eintragen, READ auf "0" legen
und PC inkrementieren
4 Schaltwerk bzw. Mikroprogramm für Befehl C7 anstoßen
5 PC auf Adreßbus legen
6 READ→1, Datum lesen (=E1) und in DP-low speichern, READ→0, PC inkr.
7 READ→1, Datum lesen (=07) und in DP-high speichern, READ→0, PC inkr.

---

2 Im folgenden werden in Beispielen Zahlen für Adressen und Befehlscodes verwendet. Da für die Darstellung von Zahlen in der Regel Vielfache von 8 Bit verwendet werden, eignet sich zur Notation besonders das Hexadezimalformat ( 2 Hexadezimalziffern für 8 Bit ).

**Op-Code=[7D]: Lade den durch DP adressierten Speicherinhalt in den Akkumulator**

1 PC auf Adreßbus legen
2 READ-Steuersignal auf "1" legen
→ jetzt wird der Speicher veranlaßt, sein Datum auf den Datenbus auszugeben
3 Datum in das Befehlsregister eintragen, READ auf "0" legen und PC inkr.
4 Schaltwerk bzw. Mikroprogramm für Befehl 7D anstoßen
5 DP auf Adreßbus legen
2 READ-Steuersignal auf "1" legen
- jetzt wird der Speicher veranlaßt sein Datum auf den Datenbus auszugeben
3 Datum in den Akku eintragen, READ auf "0" legen und PC inkr.

**Op-Code=[48]: Inkrementiere den Inhalt des Akkumulators**

1 PC auf Adreßbus legen
2 READ-Steuersignal auf "1" legen
→ jetzt wird der Speicher veranlaßt, sein Datum auf den Datenbus auszugeben
3 Datum in das Befehlsregister eintragen, READ auf "0" legen und PC inkr.
4 Schaltwerk bzw. Mikroprogramm für Befehl 48 anstoßen
5 Akkuinhalt in ALU bringen, dort inkr. und zurück in Akku schreiben

**Op-Code= [FD]: Speichere den Akkumulator in die durch DP adressierte Speicherzelle**

1 PC auf Adreßbus legen
2 READ-Steuersignal auf "1" legen
→ jetzt wird der Speicher veranlaßt, sein Datum auf den Datenbus auszugeben
3 Datum in das Befehlsregister eintragen, READ auf "0" legen und PC inkr.
4 Schaltwerk bzw. Mikroprogramm für Befehl FD anstoßen
5 DP auf Adreßbus legen
2 WRITE-Steuersignal auf "1" legen
→ jetzt wird der Speicher veranlaßt das Datum vom Datenbus zu übernehmen
3 Datum in den Akku eintragen, READ auf "0" legen und PC inkr.

Betrachtet man den internen Ablauf innerhalb des Prozessors, so ergibt sich zwischen den verschiedenen Registern und dem internen Datenbus eine ähnliche Art der Programmabarbeitung, wie außerhalb zwischen CPU und externen Speichern oder der Ein-/ Ausgabe.

Die Bearbeitung eines einzelnen Befehls übernimmt innerhalb der CPU das Steuerwerk (CU für Control Unit). Nach dem Einlesen eines Befehlskodes erfolgt dessen Interpretation und Ausführung. Während die Befehlsholphase (Instruction-Fetch) bei allen Prozessorbefehlen gleich ist, lösen verschiedene Befehle während ihrer Ausführung auch unterschiedliche Reaktionen innerhalb und außerhalb der CPU aus. Dabei liefert ein komplexes Schaltwerk oder ein fest eingetragenes Bitmuster (Mikroprogramm) eine Menge von Signalverläufen, welche interne Datenpfade schalten und Register zur Übernahme von Daten veranlassen (triggern).

Zur Durchführung von internen Aktionen wird also an jedes der Datentore bzw. an jedes vorhandene Triggersignal eines internen Registers eine exakt definierte und zeitlich aufeinander abgestimmte Folge von binären Signalen ausgegeben. Die Folge ergibt sich z.B. durch die aufsteigende Adressierung des Mikroprogrammspeichers ab einer, dem Befehl zugeordneten Anfangsadresse, welche von der Steuereinheit abhängig vom eingelesenen und dekodierten Maschinenbefehl eingestellt wird.

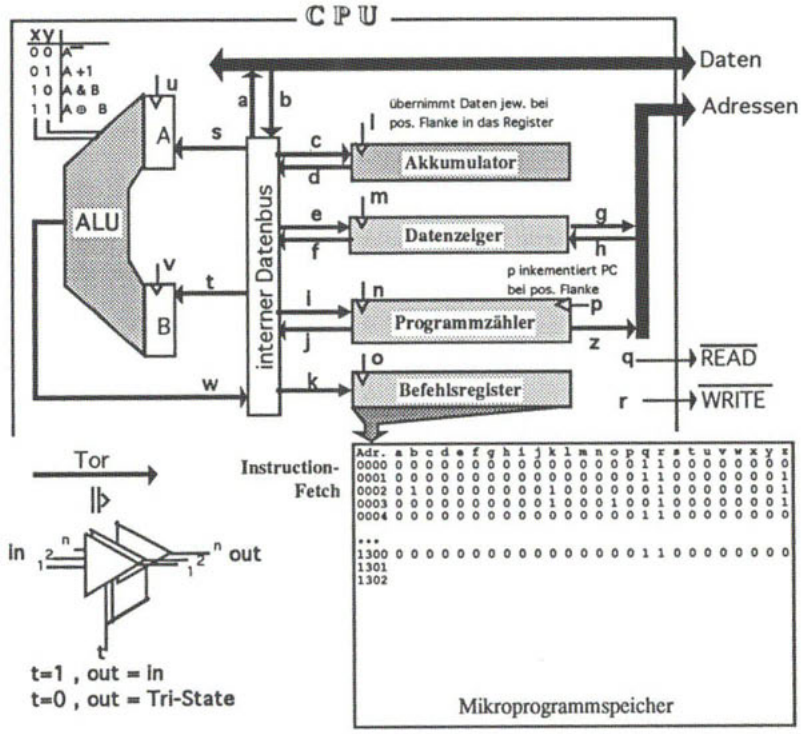



Abb. 1.1-6: Beispiel für die Abläufe innerhalb der CPU

Bei der überwiegenden Zahl der Prozessoren (insbesondere bei einer Realisierung des Steuerwerks als Schaltwerk) liegen diese internen Abläufe unveränderlich fest. Im Fall der Realisierung mit Hilfe eines Mikroprogramms besteht potentiell auch die Möglichkeit, dieses Mikroprogramm teilweise statt

in einem ROM, in einem RAM zu halten und es mit Hilfe von Prozessorbefehlen zu verändern. Prozessoren, welche über diese Möglichkeit verfügen, nennt man **mikroprogrammierbar**. Sie erlauben somit die Definition von speziellen (eigenen) Prozessorbefehlen (Makrobefehlen).

Um unterschiedliche Rechnerkomponenten wie z.B. CPU, Speicher oder I/O-Bausteine an einem gemeinsamen Bus betreiben zu können, müssen die Ausgangstreiber einen zusätzlichen Zustand besitzen, in welchem sie weder "0" noch "1" Signal führen, sondern eine Abtrennung vom Bus bewirken. Nehmen die Ausgänge diesen dritten Zustand ein, so sind sie gegenüber dem gemeinsamen Bus hochohmig geschaltet (**Tri-State-Ausgänge**).

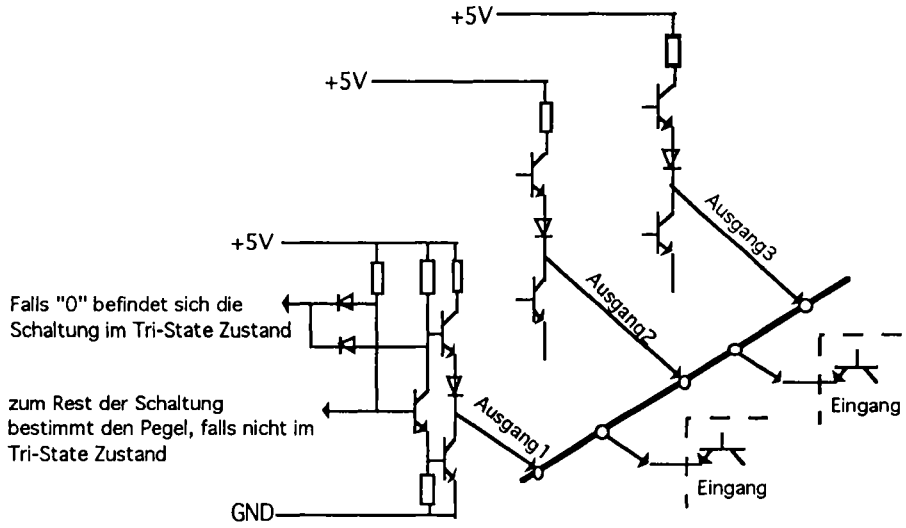



Abb. 1.1-7: Schaltung mit Tri-State-Ausgang

Ein anderer Ausgang kann damit den Pegel auf der gemeinsam genutzten Datenleitung bestimmen. Will man einen "Kurzschluß" vermeiden, so darf sich von allen Ausgängen, welche an einer gemeinsam benutzten Datenleitung angeschlossen sind, nur höchstens einer im aktiven Zustand (d.h. nicht im Tri-State-Zustand) befinden.

## 1.2. Die Realisierung von Rechenwerken

Für zweistellige Verknüpfungen beinhaltet ein typischer Rechenzyklus folgende Operationen:

1. Selektion von zwei Quellregistern aus einem Registerblock
2. Verknüpfung der Registerinhalte in einer ALU
3. Speicherung des Ergebnisses in ein Zielregister in einem Registerblock

Für die Realisierung eines solchen Rechenwerks können Drei-, Zwei- und Ein-Bus Rechenwerke mit ein- bis dreischrittigem Rechenzyklus unterschieden werden. Die logisch einfachste Realisierung eines Rechenwerks besteht in einem Drei-Bus Rechenwerk mit einschrittigem Rechenzyklus:

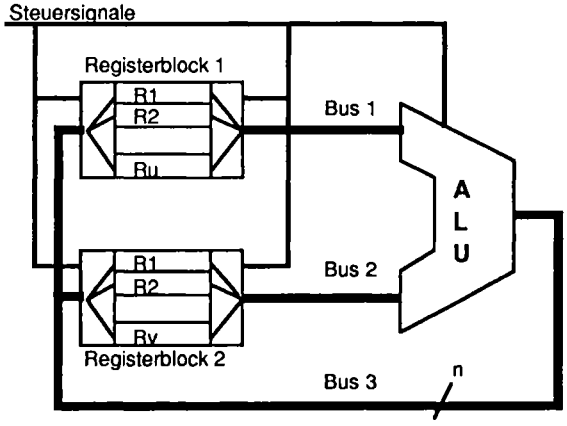



Abb. 1.2-1: Drei-Bus Rechenwerk mit zwei Registerblöcken

Hierbei verfügt jeder Registerblock über einen separaten Eingangs- und Ausgangsmultiplexer sowie Busanschluß. Es können jeweils nur ein Register aus verschiedenen Registerblöcken in der ALU verknüpft werden. Die Steuerleitungen definieren die Operandenregister (Ausgangsmultiplexer), die gewünschte ALU-Operation sowie das Zielregister (Eingangsmultiplexer).

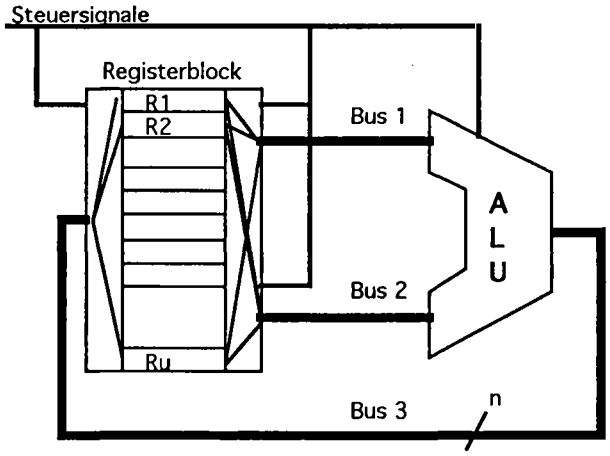



Abb. 1.2-2: Drei-Bus Rechenwerk mit einem Registerblock

Zur Steigerung der Flexibilität können die beiden Registerblöcke auch zusammengefaßt werden. Es sind damit beliebige Registerpaare miteinander verknüpfbar.

Zur Reduktion der notwendigen Ausgangsmultiplexer und Busleitungen auf dem Chip werden Zwei-Bus Rechenwerke eingesetzt. Sie benötigen dann jedoch zwei Schritte, um eine Operation in der ALU durchführen zu können.

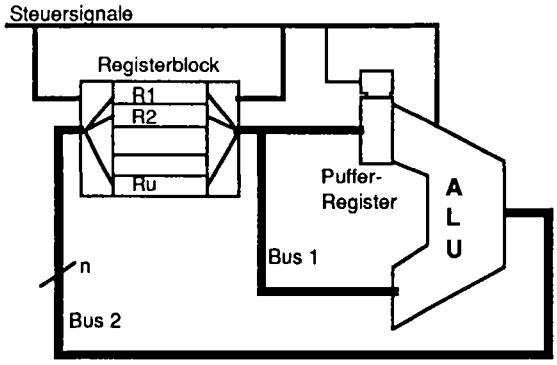



Abb. 1.2-3: Zwei-Bus Rechenwerk mit zweischrittigem Rechenzyklus

Zur weiteren Vereinfachung der Verdrahtung beim Chip-Layout werden auch Ein-Bus Rechenwerke eingesetzt. Hierbei benötigt man zur Durchführung einer ALU-Operation jedoch drei Schritte, wobei sich zuerst die beiden Operanden und dann das Ergebnis auf dem gemeinsamen Bus befinden. Die Operanden werden beide in Puffern zwischengespeichert.

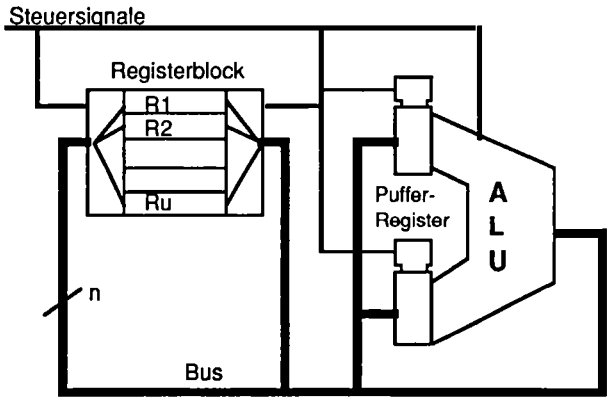



Abb. 1.2-4: Ein-Bus Rechenwerk mit einem Registerblock

## 1.3. Logik außerhalb der CPU

Das Timing der Signale außerhalb der CPU hat in der Regel den folgenden Aufbau:
Jeder Vorgang beginnt z.B. mit der Anstiegsflanke des Taktsignals $\phi$. Zu diesem Zeitpunkt gibt der Prozessor eine Adresse auf den Adreßbus und setzt das Signal zum Lesen eines Datums. Der Speicherbaustein und die evtl. vorhandene Dekodierlogik dekodiert die angelegte Adresse und für den Fall, daß sie angesprochen wurde, legt sie das entsprechende Datum auf den Datenbus. Während das Lesesignal bereits wieder mit der fallenden Flanke von $\phi$ zurückgenommen wird, bleibt das Datum des Speicherbausteins noch bis zum Ansteigen des Taktsignals stabil. Innerhalb dieser Zeit übernimmt der Prozessor das Datum in das intern angesprochene Register (z.B. Befehlsregister oder Akku).

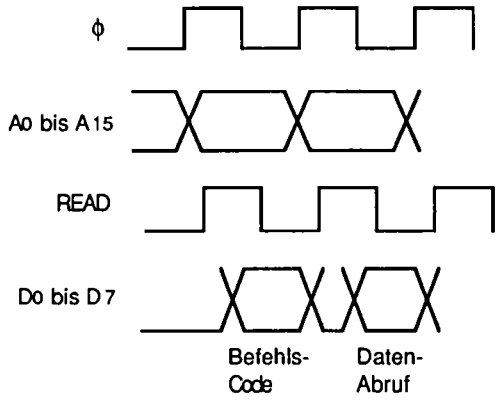



Abb. 1.3-1: Timing außerhalb der CPU

Soll ein Datum geschrieben werden, so wird statt des READ-Signals ein WRITE-Signal ausgegeben. Die Dekodierlogik, welche sich teils innerhalb, teils außerhalb der Speicherbausteine befindet, sorgt dafür, daß (bei vollständiger Dekodierung) genau eine Adresse, bzw. ein Register des Speichers oder eines Ein-/Ausgabebausteins angesprochen wird.

Zum Anschluß eines einfachen Registers an einen Mikrorechner genügt also die nachfolgende Schaltung:

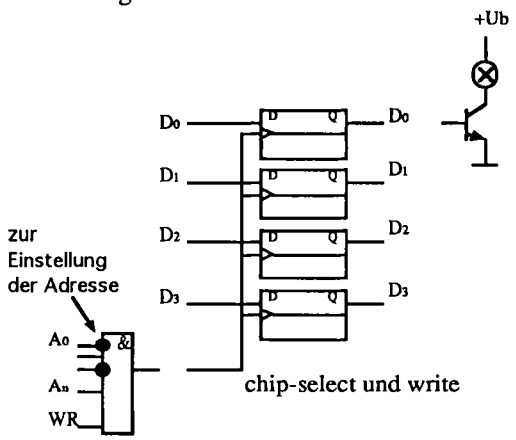



Abb. 1.3-2: Anschluß eines einfachen Registers (Latch)

Mit ihrer Hilfe läßt sich auf einfache Weise ein Ausgabebaustein realisieren. Werden Adreßleitungen und die WRITE-Leitung in geeigneter Weise beschaltet, so übernimmt das Register die momentan auf dem Datenbus befindlichen Daten (hier nur Bit 0 bis 3).

Eine Erweiterung der Schaltung ermöglicht das Auslesen von Registerinhalten. In der u.a. Schaltung können vier Bit adressiert und ein- bzw.

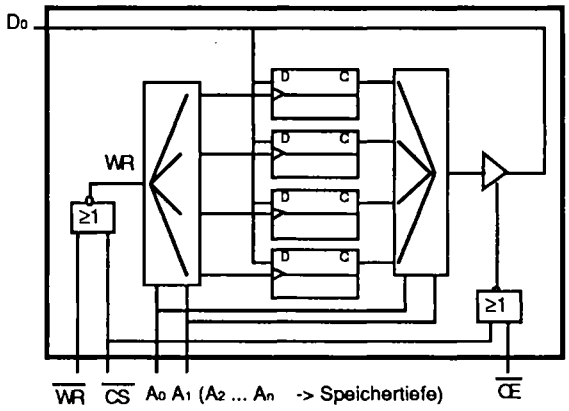
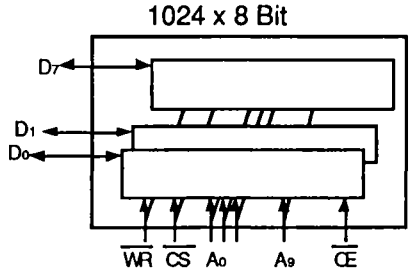



Abb. 1.3-3: Speichermodell

ausgelesen werden. Während der Baustein mit dem Anschluß CS (für Chip-Select) aktiviert wird und mit Hilfe der beiden Adreßleitungen eine der vier Adressen selektiert werden kann, erfolgt mit den Anschlüssen WR (WRITE) und OE (Output-Enable) das Einschreiben bzw. Auslesen von Speicherinhalten.

Werden Multiplexer mit einer größeren Anzahl von Kanälen und einer damit erhöhten Anzahl von Auswahlleitungen ($A_1$ bis $A_m$) verwendet, so erhöht sich der adressierbare Speicherbereich (Speichertiefe). Wird die gesamte Anordnung n-mal dupliziert und bis auf die Anschlüsse $D_i$ (i= 0 bis n-1) alle weiteren miteinander entsprechend verbunden, so erhöht sich die Wortbreite (Speicherbreite).




Die verwendeten Halbleiterspeicher, welche an einen Mikroprozessor angeschlossen werden können, unterscheiden sich grob nach

- Typ (RAM, ROM, EPROM, BIPORT-RAM, ...)
- Organisation (**n** K x **m** Bit, z.B. 32K x 8 Bit)
- Größe (Anzahl Kilobyte **KB**, Megabyte **MB**, Gigabyte **GB**)
  - statische Parameter
    - Spannung, Strom
    - fan-in, fan-out
    - Leistungsaufnahme
  - dynamische Parameter
    - Zugriffszeit
    - Zykluszeit > Zugriffszeit
    - Setup- und Hold-Zeiten

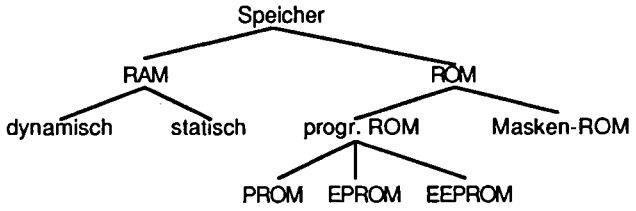



Abb. 1.3-4: Speichertypen

Hierbei stehen:

**RAM** für Random Access Memory, d.h. Schreib-/Lesespeicher mit wahlfreiem Zugriff,
**ROM** für Read Only Memory, d.h. nur Lesespeicher, nicht wie bei RAM,
**PROM** für Programmable ROM, d.h. programmierbarer ROM,
**EPROM** für Electrically PROM, d.h. elektrisch programmierbarer ROM und
**EEPROM** für Erasable EPROM, d.h. löschbarer EPROM.

Während PROM entweder im Verlauf des Herstellungsprozesses oder nach der Herstellung vom Anwender nur einmalig programmiert werden können, bieten (E)EPROM-Bausteine den Vorteil, daß sie elektrisch oder per intensiver Bestrahlung mit UV-Licht mehrfach gelöscht und neu programmiert werden können. Da letztere teurer sind, entscheidet der Anwendungsfall und die Anzahl benötigter Bausteine die Wahl (z.B. Entwicklung oder Konsumgüter).

## 1.4. Mikrorechnerfamilien

Eine Mikrorechnerfamilie besteht aus einem Satz von Bauelementen mit aufeinander abgestimmten Eigenschaften (auch die TTL-Bausteine 74xx bilden z.B. eine Familie).

Voraussetzung für die Verträglichkeit von verschiedenen Bauelementen innerhalb einer Schaltung sind dabei

- definierte Signalpegel,
- abgestimmte Lastfaktoren,
- definierte Steuersignale und Logik,
- abgestimmtes Zeitverhalten (Timing) und in der Regel auch
- gleiche Versorgungsspannung.

Beispiele für die bekanntesten Mikrorechnerfamilien sind:

| 8 Bit Familien | | 16 und 32 Bit Familien | |
|---|---|---|---|
| • 8080(Z80) | Fa. Intel (Fa. Zilog) | • 680x0 | Fa. Motorola |
| • 6800 | Fa. Motorola | • 80x86 | Fa. Intel |
| • 6502 | Fa. Motorola | • Z8000 | Fa. Zilog |
| | | • TMS 9900 | Fa. Texas Instruments |
| | | • 32000 | Fa. National |

Zu den Komponenten einer Rechnerfamilie zählen neben der CPU auch Speicher- und Ein-/ Ausgabebausteine, Adreßdekoder, Interrupt- und DMA-Controller, usw..

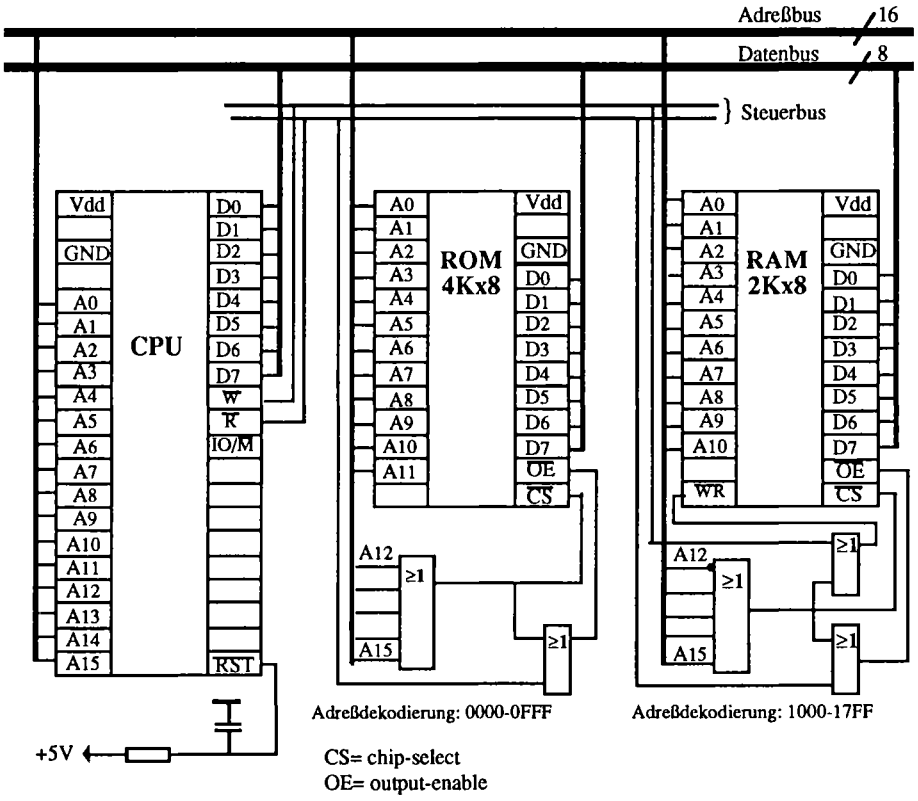



Abb. 1.4-1: Verbindung von Komponenten einer Rechnerfamilie zu einem Modul

In der Praxis gibt es für die verschiedenen Funktionen und Anwendungen oftmals bereits angepaßte, vorgefertigte Module:

- **CPU-Modul**
  - CPU, Koprozessor, DMA, ... ,ROM mit Monitorprogramm, minimaler RAM
  - evtl. eine ser. Schnittstelle zum Anschluß eines Terminals

- **Speichermodul**
  - Hauptspeicher des Systems (1-32 MB)

- **Hard-Disk (HD-) bzw. Floppy-Disk (FD-) Controller**
  - zum Anschluß von Hintergrundspeicher, z.B. Hard- oder Floppy-Disk

- **Ein-/Ausgabemodul**
  - V.24 oder Centronix Schnittstelle, ISDN Anschluß (B+B+D Kanal), ...
  - Feldbus Schnittstelle, Ethernet Anschluß, ...

- **Module zur Ankopplung von Prozeß Ein-/Ausgabe**
  - Digital Ein-/Ausgabe
  - Analog Ein-/Ausgabe
  - Zähler und Zeitgeber

Die Verbindung dieser Module erfolgt über einen Systembus, welcher Adreß-, Daten- und Steuerbus umfaßt und oftmals zahlreiche weitere Leitungen enthält.

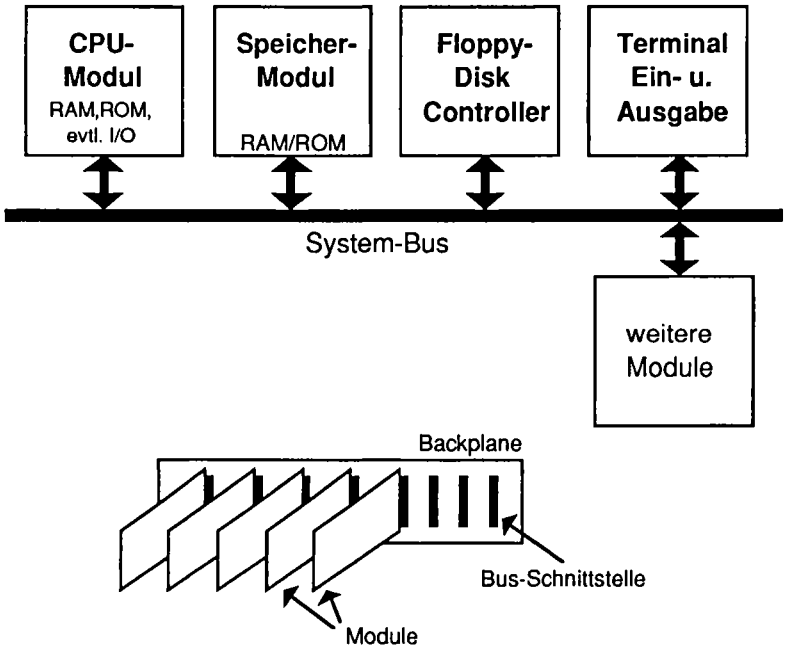



Abb. 1.4-2: Modularer Aufbau eines Rechnersystems

Die Vorteile eines modularen Aufbaus:

- Überschaubarkeit und getrennte Testbarkeit der einzelnen Module
- System ist durch den Anwender frei konfigurierbar
- reparaturfreundlicher Aufbau → erhöhte Verfügbarkeit
- Standardisierung der Busschnittstelle möglich, d.h. mehrere Hersteller können kompatible Module anbieten

Nachteile:

- hoher mechanischer Aufwand (und damit auch Preis)
- auf jedem Modul werden Bausteine zum Ansteuern des Busses benötigt (sog. Anschaltlogik)

- mehr Bausteine heißt höherer Leistungsbedarf, d.h. teureres Netzteil und höhere Betriebskosten
- der Bus bedingt zusätzliche Signalverzögerungszeiten
  d.h. stärkere Begrenzung der Signalanstiegs- und Abfallzeiten

Heute verbreitete Bussysteme sind z.B.:

PC-Slot-Bus (ISA), EISA, Local-Bus, Micro-Channel, Turbo-Channel, VME-Bus, Future-Bus++, Q-Bus, Multibus-II, Nu-Bus, S-Bus, ...

# 2. Hardware des Mikroprozessors 80286

## 2.1. Einleitung

Die Anfänge der Mikroprozessorfamilie der Fa. Intel gehen vor das Jahr 1970 zurück. Nach einem ersten 4-Bit- und 8-Bit-Prozessor, welche einen großen Aufwand an externer Beschaltung erforderten, mehrere zeitlich verschobene Taktsignale benötigten und an mehrere verschiedene Spannungspegel angeschlossen werden mußten, entstand 1974 der Prozessor 8080. Er hatte eine Wortbreite von 8 Bit und konnte 64 KByte Speicher adressieren (16-Bit-Adreßbus).

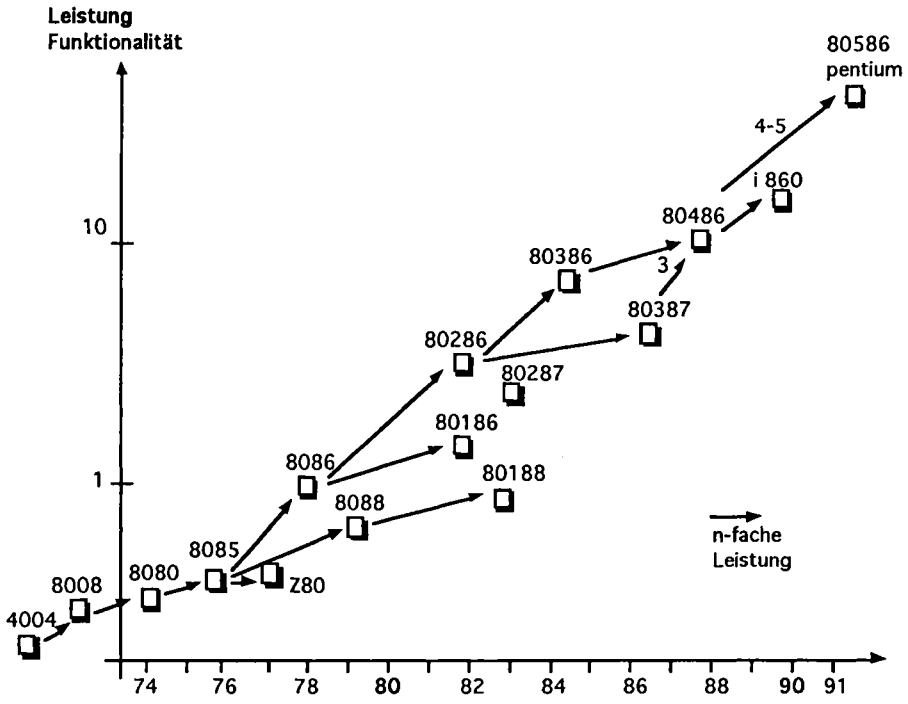



Abb. 2.1-1: Die Intel Mikroprozessorfamilie

Mit der Entwicklung des 16-Bit-Mikroprozessors 8086 vollzog die Fa. Intel 1978 den Übergang zu der Familie der 16-Bit-Mikroprozessoren. Eine Variante dieses Prozessors, der Baustein 8088, welcher intern dem 8086 zwar gleich war (insbesondere befehlskompatibel), jedoch extern mit einem im Zeitmultiplex betriebenen 8 Bit breiten Datenbus auskam, wurde zur Grundlage für die ersten Personalcomputer (PC), welche im industriellen