

Materialsammlung Grundlagen der  
Rechnerarchitektur und Informatik —  
Neuere Entwicklungen  
in der Informationstechnologie

Prof. Dr. Hans-Jürgen Buhl

2005

Fachbereich Mathematik und Naturwissenschaften(C)  
Mathematik und Informatik  
Bergische Universität Wuppertal

Interner Bericht der Integrierten Arbeitsgruppe  
*Mathematische Probleme aus dem Ingenieurbereich*  
IAGMPI – 9505

April 2005  
7. Auflage, 2005

*Praktische Informatik 01*



# Inhaltsverzeichnis

<b>1</b>	<b>Rechnerarchitekturen</b>	<b>1</b>
1.1	Der von-Neumann-Rechner	15
1.2	Modularer Aufbau von Betriebssystemen	43
1.3	CISC/RISC	47
1.3.1	CISC	47
1.3.2	„Überalterung“ von CISC	50
1.3.3	RISC (= reduced instruction set computer)-Designprinzip	53
1.3.4	Historischer CISC-RISC CPU-Vergleich (1992f.)	57
1.3.5	RISC-Marktkonsolidierung	63
1.4	HighEnd-Server und Großrechner	71
1.5	Der Cache zur Datentransferbeschleunigung	72
1.6	Multiuser-Betriebssysteme	73
1.7	Multitasking und Timesharing	73
1.8	Multiuser-Multitasking-Betriebssysteme	74
1.9	Multithreading	75
1.10	Hyperthreading	76
1.11	Dual-core CPUs	80
1.12	Bussysteme — Chipsätze und das Motherboard	83
1.13	Der Weg fort von der von-Neumann-Architektur	87
1.13.1	SPARC	87
1.13.2	Der Intel Pentium	109
1.14	IA64 - Die EPIC-Architektur	119
1.14.1	Itanium	120
1.14.2	Itanium2	123
1.15	AMDs 64-Bit-Architektur	125
1.15.1	Athlon32	125
1.15.2	Athlon64 und Opteron	126
1.16	PentiumM	127
1.17	Pentium Doppelkern-Prozessoren	128
1.18	AMD-Doppelkernprozessoren	130

# Abbildungsverzeichnis

1.1	Computersystem	5
1.2	Der ZUSE Z3 von Konrad Zuse	6
1.3	Universalrechner im Aufbau	17
1.4	„circuit switched“ Datentransfer	19
1.5	Datentypen	24
1.6	Verteilung UNICODE	31
1.7	Bidirectional Ordering	32
1.8	General Scripts	33
1.9	Die Bytes im Speicher ...	41
1.10	... und ihre Reihenfolge im Computerwort	41
1.11	CPU	42
1.12	Windows 95/98	43
1.13	Windows 2000	44
1.14	Solaris/Linux	45
1.15	Solaris Kernel Overview	46
1.16	Microcode und Betriebssystem	46
1.17	CISC-CPU-Aufbau	47
1.18	Microprogrammierte CISC-CPU	48
1.19	Nanoprogrammierte CISC-CPU	49
1.20	Der x86 MOV-Befehl	51
1.21	Der x86 MUL-Befehl	52
1.22	Registerstack am Beispiel SPARC	56
1.23	Der Cache als Daten-Vorratsbehälter	72
1.24	Multitask-Stati	73
1.25	Hyperthreading	77
1.26	Hyperthreading (Fortsetzung)	78
1.27	Dual-core CPU	80
1.28	Verschiedene Geschwindigkeiten im Computer	82
1.29	PC Frontsidebus	83
1.30	Alternative zum FSB: Der Opteron Hypertransport-Link	84
1.31	Hypertransport-Link vs. FSB	84

1.32	Hypertransport-Link (Forts.)	85
1.33	Graphikkarte am PCIe	86
1.34	UltraSPARC-II	102
1.35	4-way UltraSPARC-III	103
1.36	UltraSPARC-III Functional Units	104
1.37	UltraSPARC-IV	105
1.38	CMT mit 4 Threads	106
1.39	CMT des Niagara (?)	107
1.40	CMT: viele mehrfachgethreadete Cores auf einem Chip	108
1.41	Pentium II	110
1.42	PentiumII Execution Units	111
1.43	PentiumII-Design 1	112
1.44	PentiumII-Design 2	113
1.45	P6 Architektur	114
1.46	Dynamic Execution Microarchitecture	115
1.47	x86-Entwicklung	116
1.48	PentiumIII und 4	116
1.49	NetBurst Architektur	117
1.50	Datendurchsatz beim Pentium4	118
1.51	VLIW	119
1.52	IA64	120
1.53	Register des Itanium	121
1.54	Roadmap IA64	122
1.55	Itanium2	123
1.56	Athlon 32 Bit mit FSB	125
1.57	Opteron	126
1.58	PentiumM	127
1.59	Dual Core Pentium D	128
1.60	Pentium D Smithfield	129
1.61	AMD Dualcore CPU	130

# Tabellenverzeichnis

1.1	Zeittafel zur Entwicklung der Computertechnik . . . . .	4
1.2	Merkmale der 1. bis 3. Computergeneration . . . . .	15
1.3	ASCII-Code . . . . .	26
1.4	ISO-Austauschtabelle . . . . .	26
1.5	PC-8 Zeichensatz . . . . .	27
1.6	Zeichensatz für Windows 3.x . . . . .	28
1.7	ISO-8859 Latin 1(ECMA-94 Latin 1) Zeichensatz . . . . .	29
1.8	nationale ISO8859-Varianten . . . . .	29
1.9	UNICODE Version 1.0, Character Blocks 0000-00FF . . . . .	33
1.10	Weitere Zeichenbereiche . . . . .	34
1.11	UNICODE to Adobe Standard Mappings . . . . .	37
1.12	The UNICODE to SGML (ISO DIS 6862.2) Mappings . . . . .	38
1.13	UNICODE to Macintosh Mappings . . . . .	39
1.14	Analyse typischer Computeranwendungen auf ihre Instruktionstypen hin	50
1.15	Von CISC zu RISC . . . . .	53
1.16	Unterscheidungsmerkmale CISC/RISC . . . . .	54
1.17	RISC-CPU's im Vergleich zu Intel 80x86-CPU's . . . . .	57
1.18	CISC PC-CPU's: i80x86 . . . . .	57
1.19	64 Bit RISC-CPU's (1995...) . . . . .	58
1.20	RISC PC-CPU's: PowerPC (Apple und IBM) . . . . .	58
1.21	PC-Bussysteme und Durchsatzraten . . . . .	59
1.22	CISC PC-CPU's: i80x86 (Forts.) . . . . .	59
1.23	Anzahl der Register-Fenster und Integer-Zyklen pro Anweisung	97
1.24	„floating point“-Zyklen pro Anweisung . . . . .	98

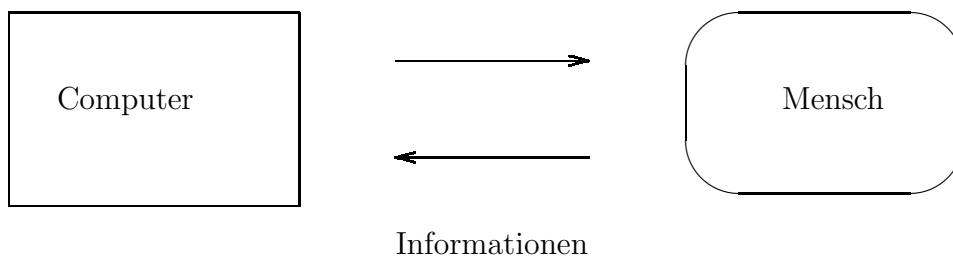
## Literatur

- Internet (siehe Links in dieser Materialsammlung).
- Duden Informatik, Ausgabe für Studium und Praxis, BI, Mannheim.
- Werner u.a.: Taschenbuch der Informatik, Leipzig.
- Stahlknecht/Hasenkamp: Einführung in die Wirtschaftsinformatik, Springer.
- [A. S. Tanenbaum: Structured Computer Organization](#), Prentice-Hall.
- [W. Stallings: Computer Organization and Architecture](#), Prentice-Hall.

# Kapitel 1

## Rechnerarchitekturen

### Vom (mechanischen) Spezialgerät zum Universalrechner



Im Gegensatz

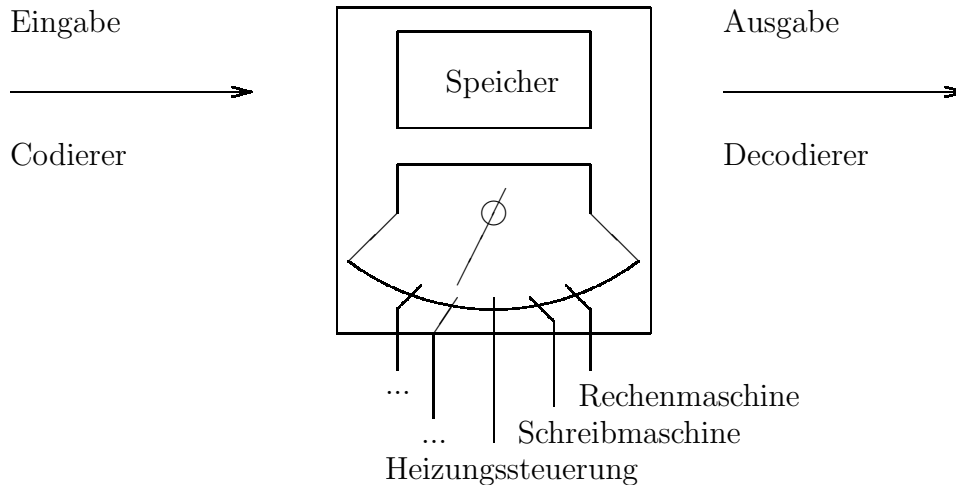
- zur mechanischen Kurbel-Rechenmaschine,
- zur Schreibmaschine,
- zum Heizungsthermostat,
- zur Programmautomatik einer Waschmaschine

ist ein *Computer* ein universell einsetzbares Gerät zur Verarbeitung von Daten (Rechnen, Datenverarbeitung (DVA), Bereitstellung von Informationen). Diese Universalität wird dadurch erreicht, daß

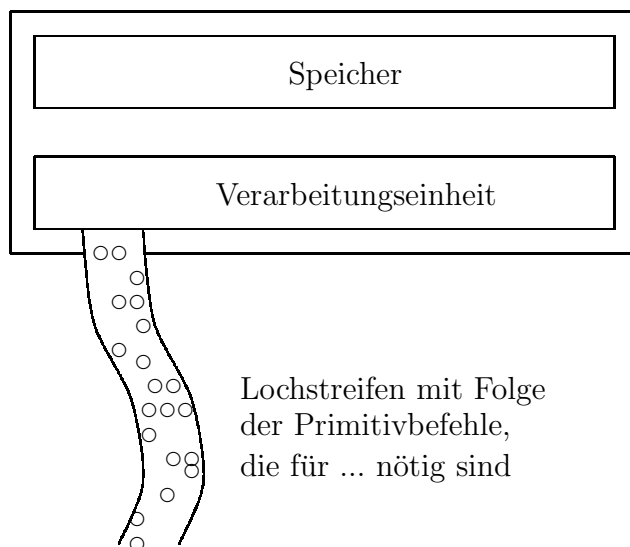
- die Informationen einheitlich als binäre Daten (0/1-Informationen, Spannung da/nicht da) codiert werden (Ein-/Ausgabe),
- nur kurzzeitig verfügbare Daten in ihrer binären Codierung zwischengespeichert werden (Speicher)



und eine *universelle* Binärdatenverarbeitungseinheit durch *Umschaltung* für fast alle denk- und lösbaren Informationsverarbeitungsaufgaben eingesetzt werden kann. Natürlich wird in der folgenden Abbildung keine volle Uni-

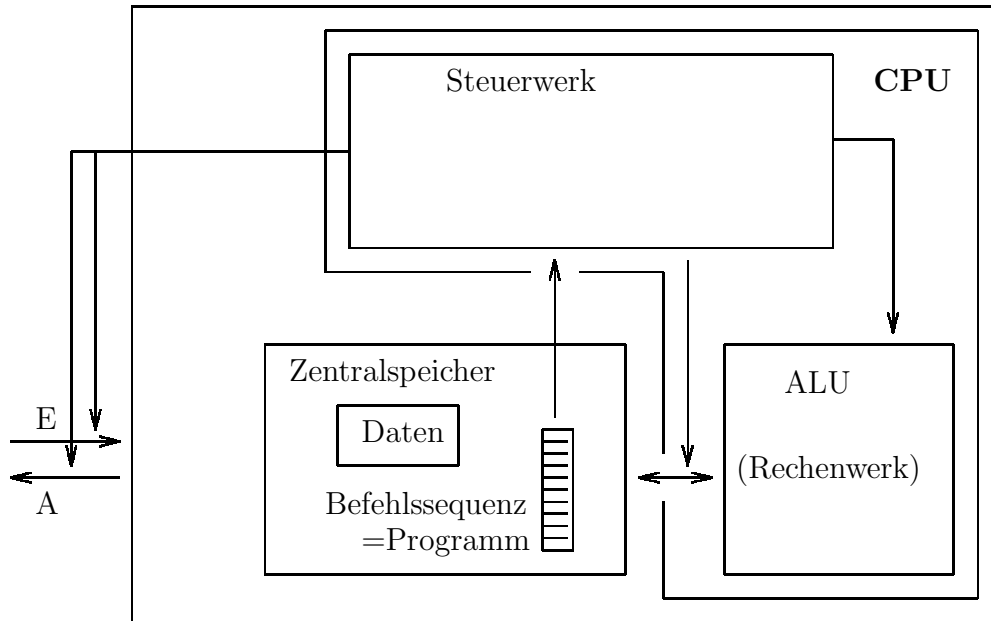


versalität erreicht; lediglich eine durch Schalterstellung definierbare *1 aus N* Spezialauswahl ist realisiert. Sieht man jedoch eine große Anzahl von die Daten nur gering ändernden *Spezialmaschinen*, sogenannte *Primitivbefehle* oder *Maschinenbefehle* vor, und baut aus *sequentieller* Aneinanderreihung solcher Befehle den gewünschten Effekt nach und nach auf, so ist die erstrebte Universalität erreicht. Erkennt man nun schließlich, daß die benötigten



*Sequenzen von Primitivbefehlen* auch nur eine spezielle Art von Information sind, die deshalb — wie die Ein- und Ausgabedaten — ebenfalls codiert

im Speicher abgelegt werden können, so haben wir den prinzipiellen Aufbau eines Computers entwickelt:



Um beliebige Programme in den *Computer* hineinzubekommen und dann ablaufen lassen zu können, ist ein Verwaltungsprogramm, das *Betriebssystem* nötig, das den Computer erst benutzbar macht. Dies ist entweder in nicht-flüchtigem Speicher (ROM = *read only memory*, Inhalt bleibt auch in ausgeschaltetem Zustand erhalten) fest eingebaut oder wird von einer Festplatte, einem Magnetband, ... mit Hilfe des im ROM fest eingebauten *Urladers* (siehe auch im *FOLDOC*) oder *Monitorprogramms* geladen.

**Aufgabe 1.1** Diskutieren Sie die Vor- und Nachteile eines Betriebssystems im ROM und eines von Platte geladenen Betriebssystems.

**Aufgabe 1.2** Welche zusätzlichen Funktionalitäten zu denjenigen eines *Urladers* stellt ein *Monitorprogramm* bereit?

Tabelle 1.1: Zeittafel zur Entwicklung der Computertechnik

1833	CHARLES BABBAGE (1792-1871), Professor an der Universität Cambridge (Großbritannien), entwirft und baut einen programmgesteuerten mechanischen Rechenautomaten, die <i>Analytical engine</i> . Sie enthält ein 4-Spezies-Rechenwerk, 1000 Zahlenspeicher, Lochkartensteuerung und Ergebnisdrukwerk (nicht vollendet).
1941	Vorführung des ersten arbeitsfähigen programmgesteuerten Rechenautomaten ZUSE Z3 in Relaisstechnik durch KONRAD ZUSE (geb. 1910)
1944	Inbetriebnahme des programmgesteuerten elektromechanischen Rechenautomaten Mark 1 von HOWARD H. AIKEN (1900-1973).
1944/46	Formulierung der Prinzipien des <i>von-Neumann-Computers</i> (JOHN VON NEUMANN (1903-1975). Realisiert erstmals mit der EDVAC (1952/53)
1946	Inbetriebnahme des ENIAC, des ersten Computers mit Elektronenröhren durch JOHN P. ECKERT (geb. 1919) und JOHN W. MAUCHLEY (1907-1980). Beginn der Epoche der elektronischen Computer.
1951	Beginn der Serienproduktion elektronischer Computer mit der Anlage UNIVAC I der Firma Remington Rand. Beginn der 2. Computergeneration.
1955	Erster Computer mit Transistoren: TRADIC (Bell. Labor.)
um 1965	Computer-Familie IBM/360. Beginn der 3. Computergeneration. Der Begriff <i>Rechnerarchitektur</i> wird erstmals verwendet.
um 1965	Minirechner PDP-8 (Digital Equipment Corp.). Kleinere Rechner entstehen neben den Mainframes.
1969	Pilotprojekt Weitverkehrsrechnernetz ARPANET (USA) in Betrieb genommen.
1971	4-Bit-Mikroprozessoren i4004 der Firma INTEL Corp. Beginn der Mikroprozessorgeneration. Rascher Übergang zu 8-Bit-Mikroprozessoren.
1975-1980	Personalcomputer auf Mikroprozessorbasis und die Software dafür werden zu Massenartikeln. Beginn der 4. Computergeneration.
um 1978	Erste Installationen lokaler Rechnernetze.
1978-1980	16-Bit-Mikroprozessoren kommen auf den Markt.
1979	Standardvorschlag der ISO » <i>Reference model of open system interconnections</i> « für Rechnernetze.
1981-1985	Personalcomputer mit 16-Bit-Mikroprozessoren werden marktbestimmend; insbesondere PC XT und AT von IBM und Kompatible dazu sowie das Betriebssystem MS-DOS der Firma Microsoft.
ab 1988	Personalcomputer mit 32-Bit-Mikroprozessoren kommen auf den Markt und lösen im Verlauf einiger Jahre die 16-Bit-Systeme ab.
ab 1996	Workstations mit 64-Bit-Mikroprozessoren

Quelle: D.Werner (Hrsg.): Taschenbuch der Informatik; Fachbuchverlag Leipzig, 1995

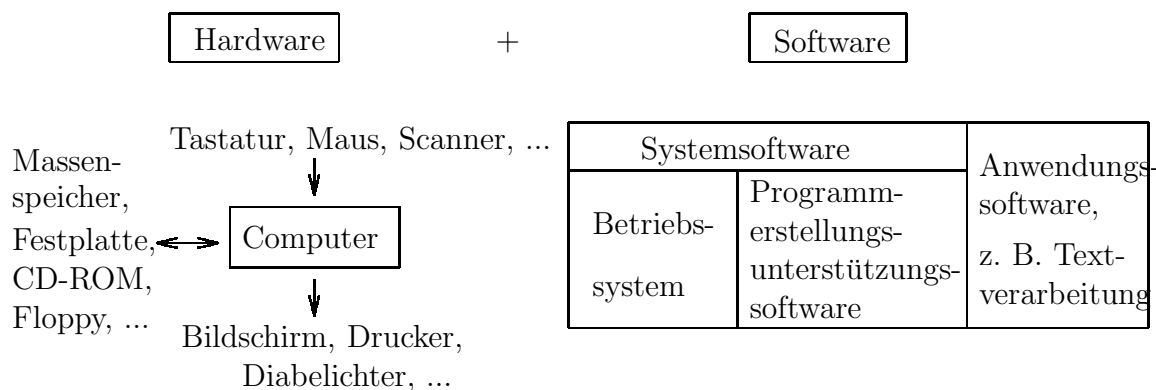


Abbildung 1.1: Computersystem

## Der Windows 2000 Bootprozeß

<http://windows.about.com/od/tipsarchive/1/aa000716a.htm>

## Dual Booting

`boot.ini`: <http://support.microsoft.com/?kbid=99743>

Linux von `boot.ini` aus starten:

[http://www.it-academy.cc/content/article\\_browse.php?ID=1402](http://www.it-academy.cc/content/article_browse.php?ID=1402)

bzw.

[http://support.novell.com/cgi-bin/search/searchtid.cgi?/en/2002/10/fhassel\\_grub\\_nt.html](http://support.novell.com/cgi-bin/search/searchtid.cgi?/en/2002/10/fhassel_grub_nt.html)

## Das BIOS

Das BIOS (= Basic Input Output System) ist der in Intel-basierten Computersystemen im ROM vorhandene Betriebssystemteil, der eng mit dem Motherboard des Rechners verbunden ist:

- BIOS: <http://www.nickles.de/biosguide/html/1.php3>
- Motherboards/Mainboards: <http://www.computerhope.com/network/mboard.htm>
- BIOS-Versionen: etwa in <http://www.heise.de/newsticker/meldung/58654> ...

## Die Zukunft der x86-Firmware: Nachfolger des BIOS

- Übersicht: <http://www.heise.de/newsticker/meldung/47858&words=Efi%20EFI>
- EFI: <http://www.intel.com/technology/efi/efi.htm>

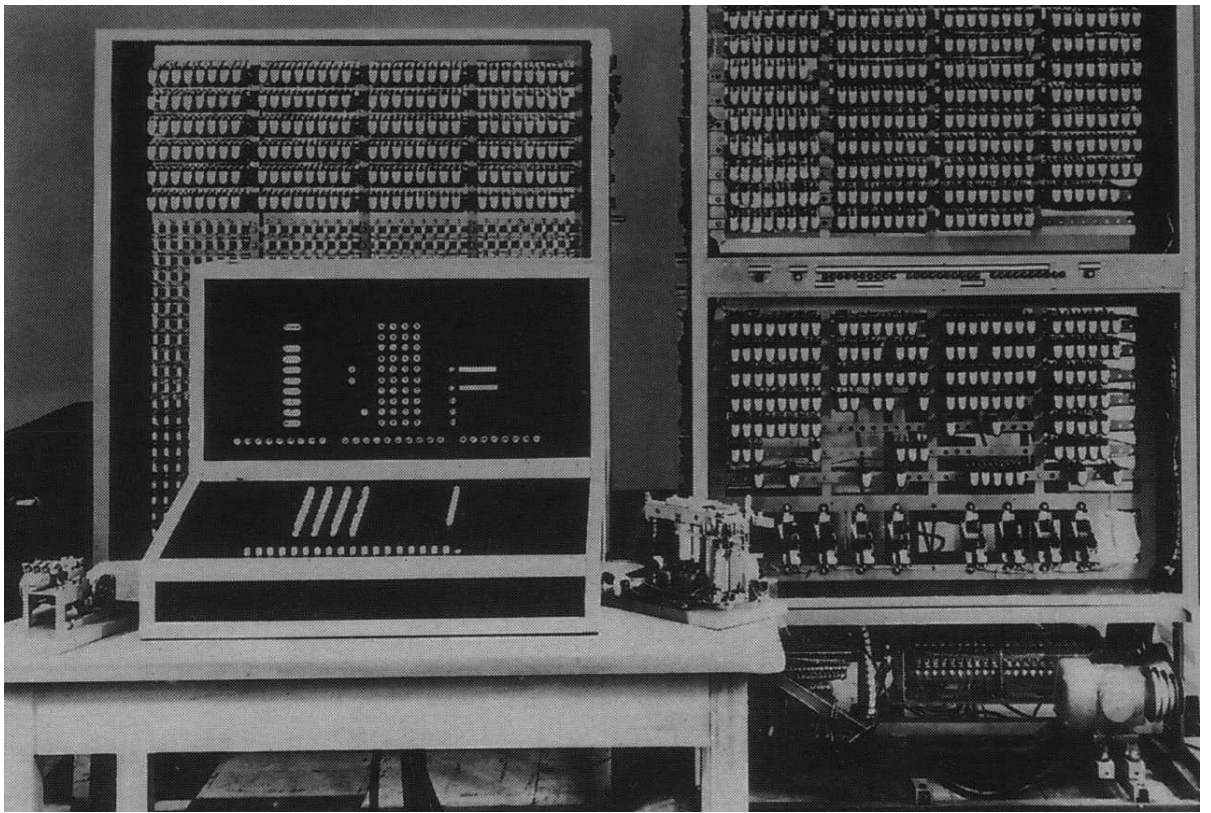


Abbildung 1.2: Der ZUSE Z3 von Konrad Zuse

- CSS: <http://deviceforge.com/news/NS6888038345.html>
- OpenBIOS/LinuxBIOS: <http://www.linuxbios.org/papers/index.html>

## Open Firmware für nicht-x86-Computer

- OpenBootPROM: [http://en.wikipedia.org/wiki/Open\\_Firmware](http://en.wikipedia.org/wiki/Open_Firmware)
- OpenBoot 3.x: <http://docs.sun.com/app/docs/doc/805-4436?q=openboot>

## Server-Controll-Prozessoren für die Server-Kontrolle

- Remote System Control: <http://docs.sun.com/app/docs/doc/805-7998-10?q=RSC>
- Lights Out Management: <http://docs-pdf.sun.com/806-2038/806-2038.pdf>

**Aufgabe 1.3** *Welche Hauptvorteile wird EFI gegenüber dem BIOS mit sich bringen?*

**Aufgabe 1.4** *Welche Dienste können mittels des RSC-Betriebssystems eines Server-RSC-Prozessors in Anspruch genommen werden?*

## Das BIOS-Setup

Das BIOS-Setup erlaubt das Einstellen der Systemkonfiguration:

- Anzeige der vorhandenen Hardware
- Setzen von Datum und Uhrzeit der Hardware-Uhr
- (früher:) Einstellung der Festplattengeometrie (Anzahl der Zylinder, der Spuren, der Sektoren)
- Auswahl der Bootreihenfolge (Platte/n, Diskette, CDROM, USB-Speicherkarte, NIC (Network), ...)
- Konfiguration des Bootvorganges: voller/minimaler POST, Boot-CPU-Geschwindigkeit, ...
- Konfiguration der Schnittstellen (parallele/serielle/Infrarot-/...)
- Konfiguration von Tastatur (Typ, USB-Emulation, ...) und Maus (Typ, bei Laptops: Touchpad parallel, ...)
- Konfiguration der Videokarten (intern/extern/...)
- Auswahl der Prozessorgeschwindigkeit, enable/disable SpeedStep-Technology
- Anzeige des Akkustatus
- Konfiguration des Stromsparmmodus/Powermanagements (für Laptops): getrennt für Netzbetrieb/Akkubetrieb: Abschaltzeiten für Videosignal, Festplatte, ...; Start des Suspend-/Hibernate-Modus; ...
- An/Aus von internem/externem Modem, Netzwerk, ...
- Wakeup on Modem/LAN, Autoboot zu Uhrzeit, ...
- Konfiguration von Bluetooth, wireless LAN, ...
- Konfiguration des Anschlusses an eine Docking-Station
- Vereinbarung von Systempasswörtern: Master, Admin, Festplatten, ...
- ...

## Festplatten-Partitionierung

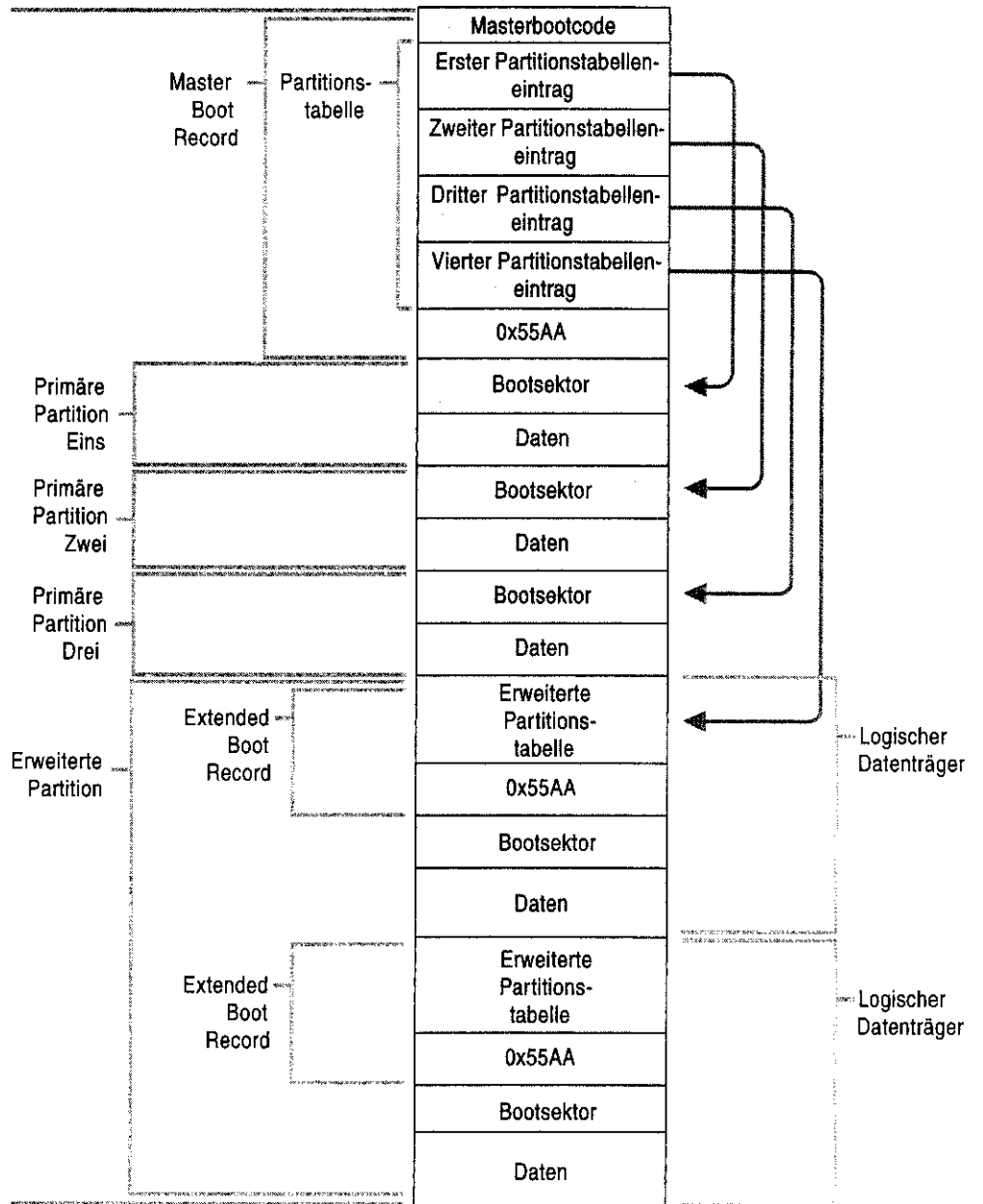
Für einige Zwecke sollte man eine Festplatte in mehrere „logische“ Festplatten unterteilen; sie sieht dann dem Betriebssystem gegenüber wie mehrere („physikalisch“ vorhandene) Festplatten aus:

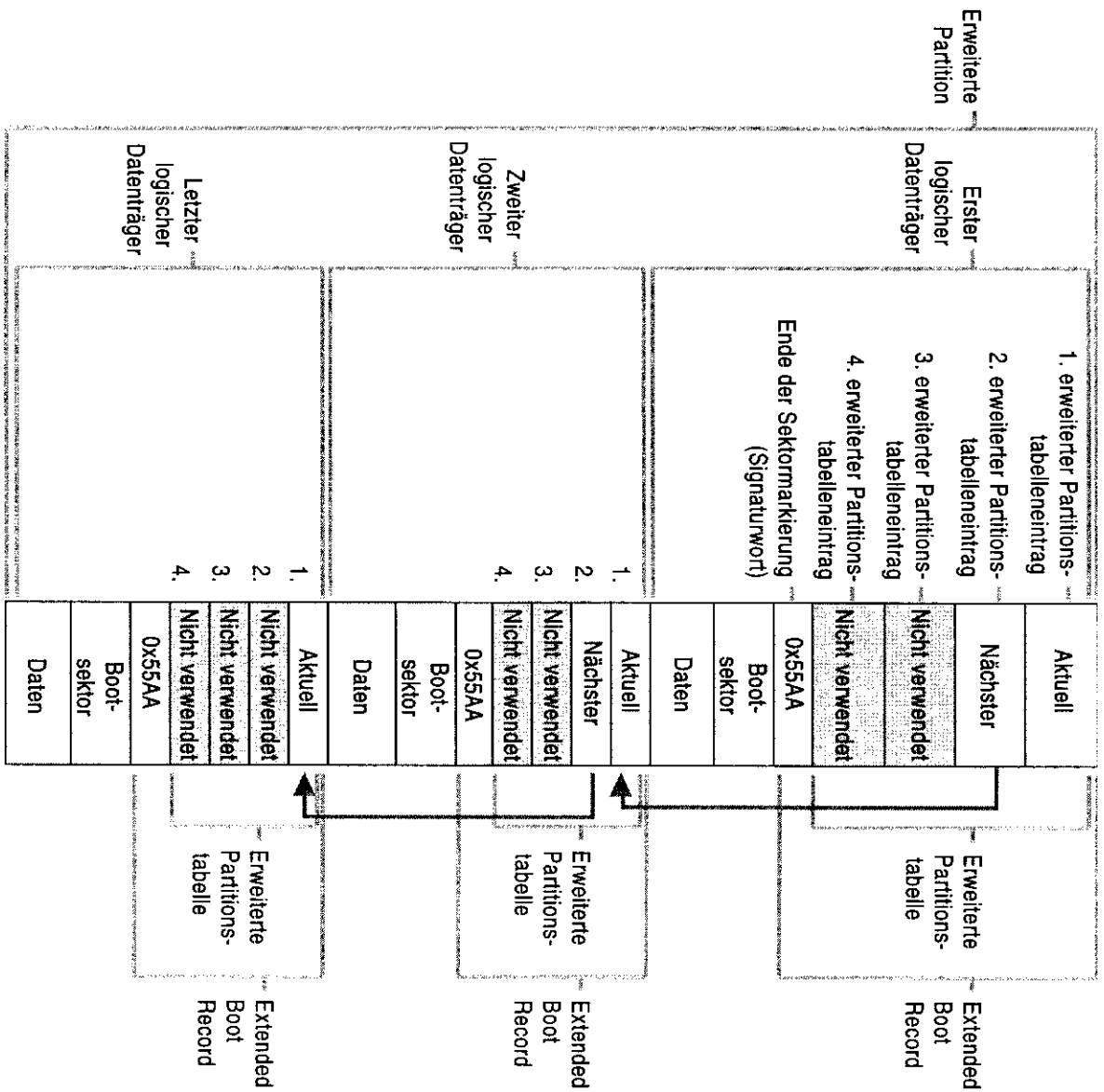
- „Vorspiegelung“ mehrerer kleinerer Festplatten, wenn die Gesamtkapazität sonst für das Betriebssystem/Dateisystem unbrauchbar groß ist (FAT16-Dateisysteme konnten zum Beispiel nur maximal 2 GB groß sein).
- Gleichzeitige Installierung mehrerer Betriebssysteme auf einer Festplatte.
- Logische Trennung der Bereiche einer Platte, die zum Beispiel das Betriebssystem (inklusive Swapping-Bereich), die Anwenderprogramme und die Datenbereiche der einzelnen Benutzer (bei Terminalservern) enthalten.
- Logische Trennung von Bereichen für das Betriebssystem, für die Loggingdatenbereiche des Betriebssystems, Temporärbereiche (wie zum Beispiel 800 MB für das Image einer CD bzw. 5 GB für ein DVD-Image), ...
- Anlegen einer gemeinsamen Daten-Partition (zum Beispiel FAT) zum Datenaustausch zwischen verschiedenen installierten Betriebssystemen.
- Anlegen einer dediziert nur zum Swapping benutzten Partition.
- Einrichten einer temporären Scratch-Partition.
- ...

Man nennt solche „logischen“ Festplatten *Partitionen*. Eine Festplatte wird für die genannten Zwecke *partitioniert*.



Partitionen:



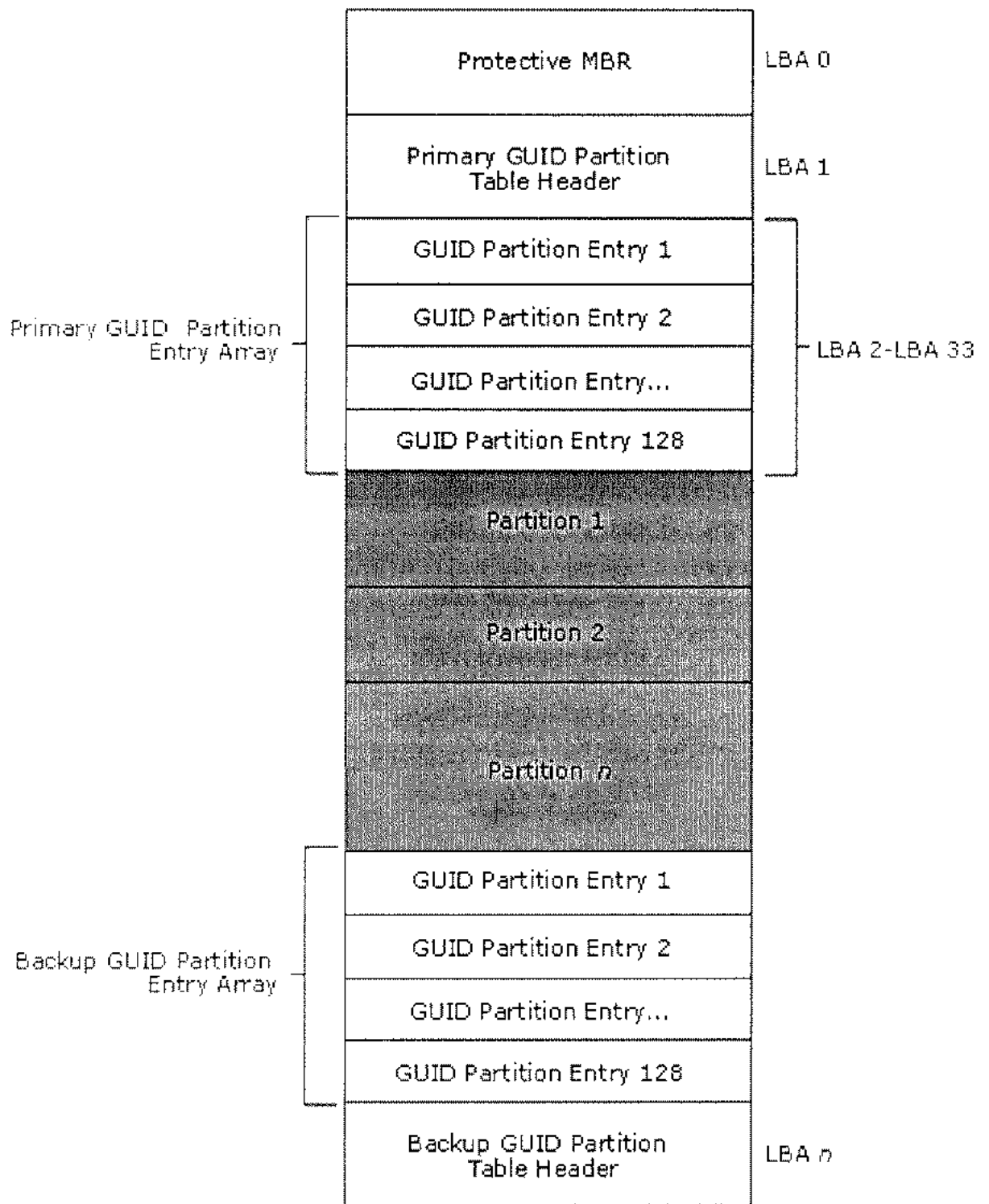


**Aufgabe 1.5** Informationen zur Verwaltung von Partitionen auf IDE-Platten finden Sie unter:

<http://www.microsoft.com/technet/prodtechnol/windowsserver2003/library/TechRef/bdeda920-1f08-4683-9ffb-7b4b50df0b5a.mspx?pf=true>

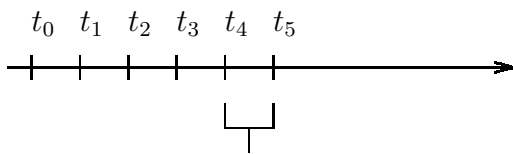
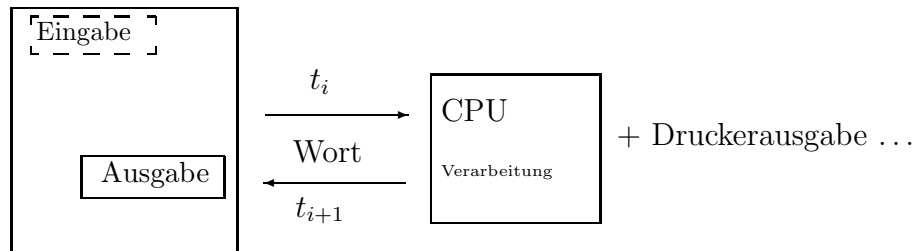
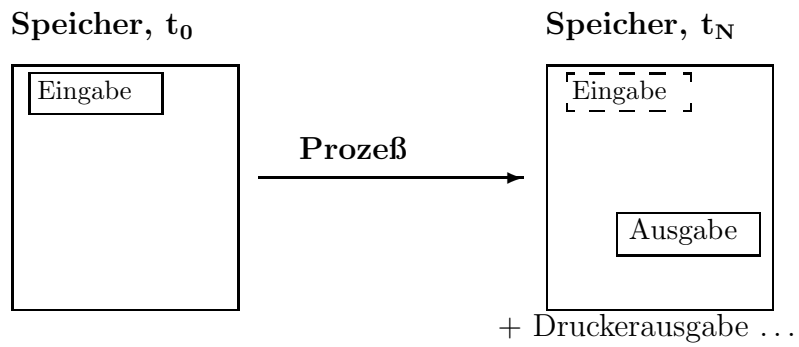
Welche Partitiones-Typen (FAT16, ...) sind dort beschrieben?

Partition Structures on a GPT Disk



```
boot.ini

[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(5)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(5)\WINDOWS="Microsoft Windows XP Professional"
/fastdetect
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Microsoft Windows 2000 Professional"
/fastdetect
```



Machinenzykluszahl heute:  
typisch

$$\frac{1}{100 \text{ MHz}} = \frac{1}{100 \cdot 10^6 \text{ s}^{-1}} = 10 \cdot 10^{-9} \text{ s} = 10 \text{ ns}$$

## 1.1 Der von-Neumann-Rechner

### Die Geschichte der Computerentwicklung

Tabelle 1.2: Merkmale der 1. bis 3. Computergeneration

	<b>1. Generation</b> ab 1951	<b>2. Generation</b> ab 1960	<b>3. Generation</b> ab 1965
Basiselement	Elektronenröhre	Flächentransistor	integrierter Schaltkreis
Arbeitsspeicher	Magnettrommel-, Umlaufspeicher	Ferritkernspeicher	Mikroferritkernspeicher
Externe Speicher	keine	Magnetbandspeicher, -trommelspeicher	Magnetband-, Wechselplattenspeicher
Ein- und Ausgabegeräte	Lochkarten-, Lochstreifengeräte, elektrische Schreibmaschine	wie 1. Generation sowie Walzendrucker	wie 2. Generation sowie Bildschirmgeräte, Datenfernübertragungsgeräte
Programmierung	Maschinencode, Assemblersprache	Assemblersprache, erste problemorientierte Sprachen, einfache Betriebssysteme	Assemblersprache, universelle Sprachen, komplexe Betriebssysteme
Einsatz als	wissenschaftlich-technische Rechner	wissenschaftlich-technische, kommerzielle und Prozeßrechner	universelle und Prozeßrechner, Einbaurechner

Quelle: D.Werner (Hrsg.): Taschenbuch der Informatik; Fachbuchverlag Leipzig, 1995

#### **Merkmale der 4. Generation (heute):**

- VLSI-Schaltkreise
- in wenigen Jahren sich verdoppelnde Rechengeschwindigkeiten, Speichergrößen, Übertragungsgeschwindigkeiten (→ in 4 Jahren ist ein Computersystem „total“ veraltet)
- Software-Entwicklungskosten höher als Hardwarekosten; wegen der schnellen Hardware-Innovationszyklen sind meist „nur“ noch portable Software-Produkte schnell genug verfügbar (UNIX, T<sub>E</sub>X, ...)
- GUI's = *graphical user interfaces*
- Computer werden immer mehr zu IT-Geräten (Datennetzwerke)
- Laserdrucker, fotorealistischer Druck, Scanner mit 24Bit Farbtiefe

#### **5. Generation:** Für nach 2000 angestrebte Merkmale:

- natürlichsprachiger Umgang mit Computern
  - Handschrift
  - Sprache
- Multimedia, Telekonferenzen, Unterhaltung, Teleshopping, T-online, WWW, Heimarbeitsplätze, Fernkurse
- Methoden der Künstlichen Intelligenz (KI)
  - Zeichen-, Bild-, Sprach-, Schrift-, Bewegungserkennung
  - simultane automatische Sprachübersetzung
  - Wissensverarbeitung (automatische Krankheitsdiagnose, ...)
  - Nutzeridentifizierung durch Iris-Mustererkennung ...

# Blockschaltbild eines Computers

Ein von-Neumann-Rechner ist durch folgende Merkmale gekennzeichnet:

## Prinzipien:

1. **Computerbestandteile:** CPU (= Rechenwerk und Steuerwerk), Speicher, Ein- und Ausgabewerk, Bussystem.

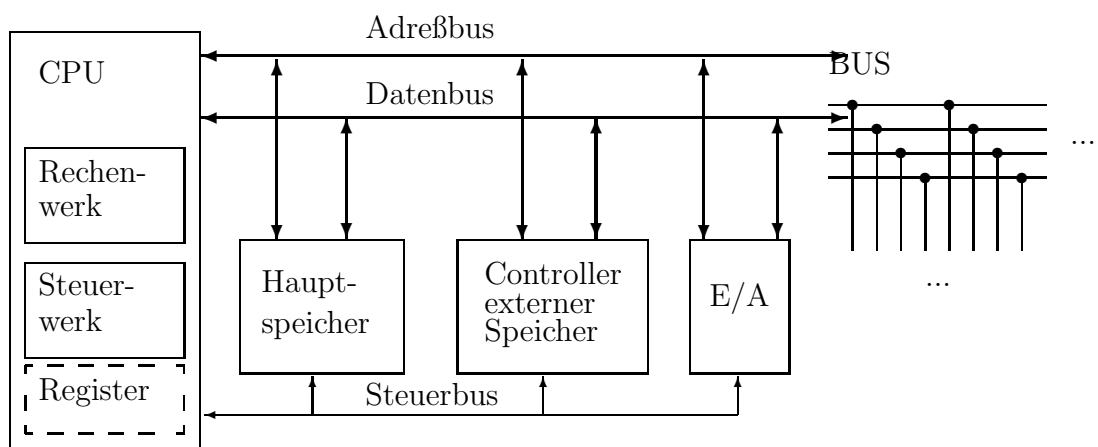


Abbildung 1.3: Universalrechner im Aufbau

2. **Universalität:** Spezialisierung nur durch von außen eingegebenes Programm.
3. **Programme als Daten:** Eingabedaten, Zwischen- und Endergebnisse sowie Programme werden im selben Speicher abgelegt.
4. **Linearer Speicher:** Der Speicher ist in gleichgroße Zellen, die fortlaufend nummeriert sind, eingeteilt.
5. **Sequentielle Ausführung:** Aufeinanderfolgende Befehle eines Programms werden in aufeinanderfolgenden Speicherzellen abgelegt. Beim Programmablauf wird im allgemeinen der Befehlszähler fortlaufend inkrementiert.
6. **Sprünge:** Sprungbefehle ermöglichen ein Durchbrechen der linearen Ausführungsreihenfolge.



7. **Zusätzliche Primitiva:** Datentransferbefehle, arithmetische Operationen, logische und Vergleichsoperationen, (heute auch: Graphik- und Multimediabefehle); unmittelbare, direkte, indizierte, relative, ... Adressierung.
8. **Binäre Codierung:** Daten (Befehle, Adressen, Operanden) werden binär codiert. Geeignete Schaltwerke (Decodierer) sorgen für die richtige Entschlüsselung.

**Aufgabe 1.6** *Informieren Sie sich mit Hilfe von*

*[http://tech-www.informatik.uni-hamburg.de/applets/baukasten/DA/VNR\\_Simulation\\_1.html](http://tech-www.informatik.uni-hamburg.de/applets/baukasten/DA/VNR_Simulation_1.html) über die internen Vorgänge beim Programmablauf in einem von-Neumann-Rechner.*

### **Adressierungsmodi**

Die x86-Adressierungsmodi sind verschiedene Arten, wie Operanden von Maschinensprachebefehlen spezifiziert werden können (vgl.

[http://www.electronics.dit.ie/staff/tscarff/8086\\_address\\_modes/8086\\_address\\_modes.htm](http://www.electronics.dit.ie/staff/tscarff/8086_address_modes/8086_address_modes.htm)).

**Aufgabe 1.7** *Skizzieren Sie jede der dargestellten Adressierungsarten in einer geeigneten Graphik (beteiligte Register und Speicherstellen, alte und neue Inhalte, welche Daten - Zahlenwerte - werden jeweils auf den Adress- bzw. Datenbus gelegt).*

### **Bussteuerung**

Beim geschilderten Computeraufbau stellt sich die Frage, wann welche Systemeinheit welche Bus-Leitungen benutzen darf. Dies wird beim Schreiben von 16 Bit von einer CPU in den Speicher durch den Bus-Controller (CPU oder DMA-Chip) mittels der *Handshake-Steuerbus-Leitungen* gemäß Abbildung 1.4 geregelt.

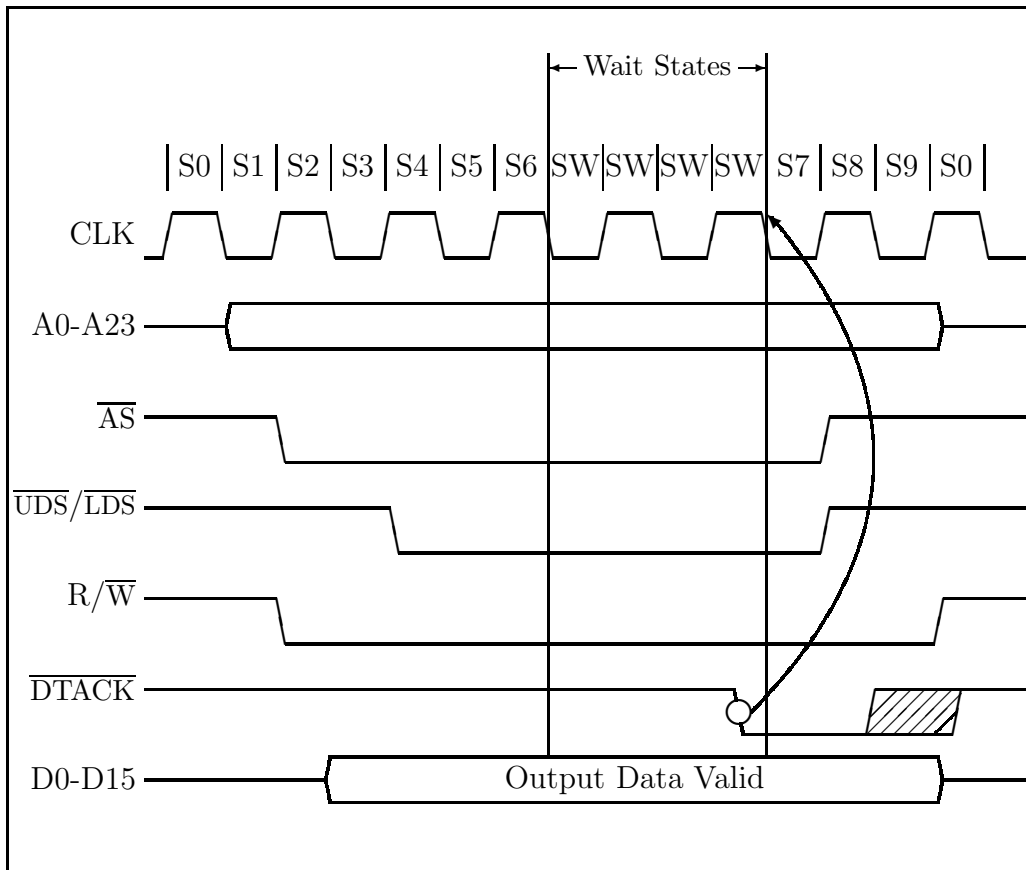


Abbildung 1.4: „circuit switched“ Datentransfer

**Aufgabe 1.8** Beschreibe in eigenen Worten den stattfindenden **Handshake** der Datenübertragung.

Bemerkungen:

- Signalnamen:

CLK	=	clock
A0-A23	=	address bus
$\overline{AS}$	=	address strobe
R/ $\overline{W}$	=	read/write
$\overline{UDS}, \overline{LDS}$	=	upper/lower data strobe
D0-D15	=	data bus
$\overline{DTACK}$	=	data transfer acknowledge

- strobe = Gültigkeitssignal für z.B. die Daten auf dem Adreßbus

Problem:

- Bei der geschilderten, *circuit switched* genannten Bus-Steuerung tritt folgendes Problem auf: Der Bus blockiert, bis die anstehende Operation ausgeführt ist. *Wait states*, etwa zur Benutzung langsamen Speichers erlaubt, verlängern noch die Bus-Blockadezeit.

Moderne Alternative:

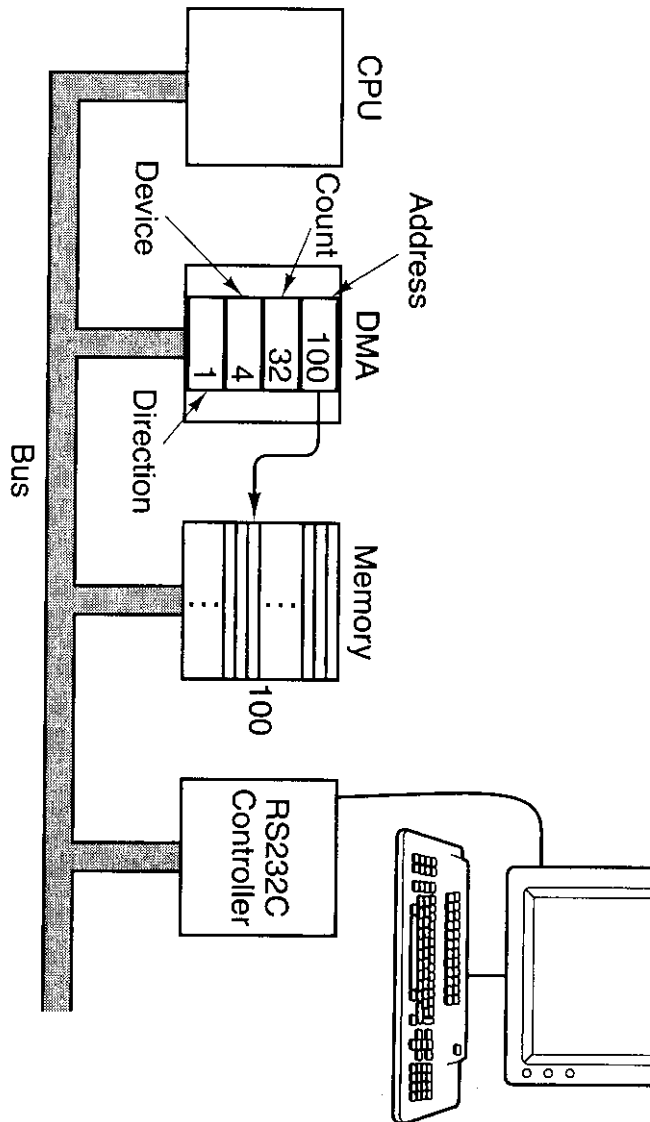
- *Packet switched*: Jeder Datentransfer wird in kleine Pakete zerlegt, die jeweils die Ziel-Adresse und einen Anteil der zu übertragenden Daten enthalten. Diese werden nacheinander nichtblockierend auf den (seriellen) Bus geschickt. Dadurch kann eine weitere CPU (ein weiterer Thread) schon wieder Daten transferieren, obwohl die erste CPU ihren Transfer noch nicht abgeschlossen hat.

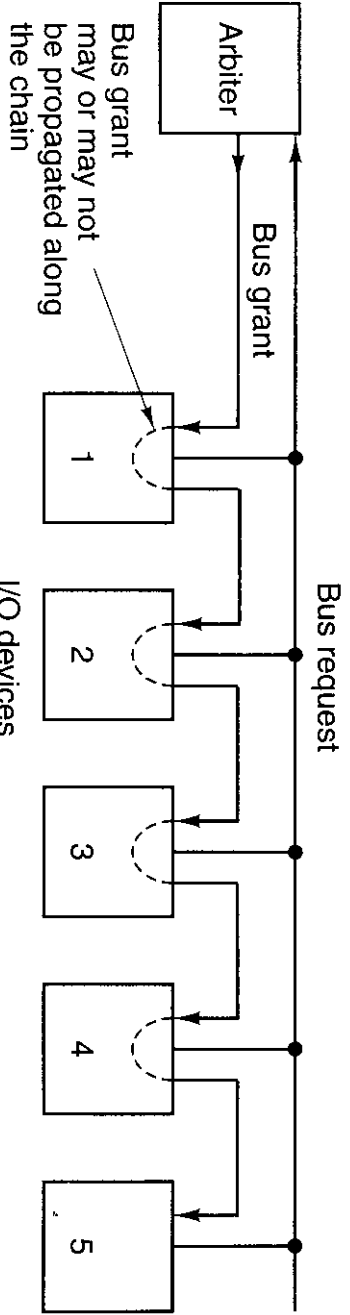
Siehe <http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?query=packet+switched>  
 Ein serieller Bus (für Daten und Adressen) ohne zusätzliche Steuereitungen verursacht das neue Problem: Wie verhindert man, dass mehrere Bus-Geräte gleichzeitig Pakete auf den Bus senden? Eine Lösung:

CSMA/CD = carrier sense multiple access / collision detect

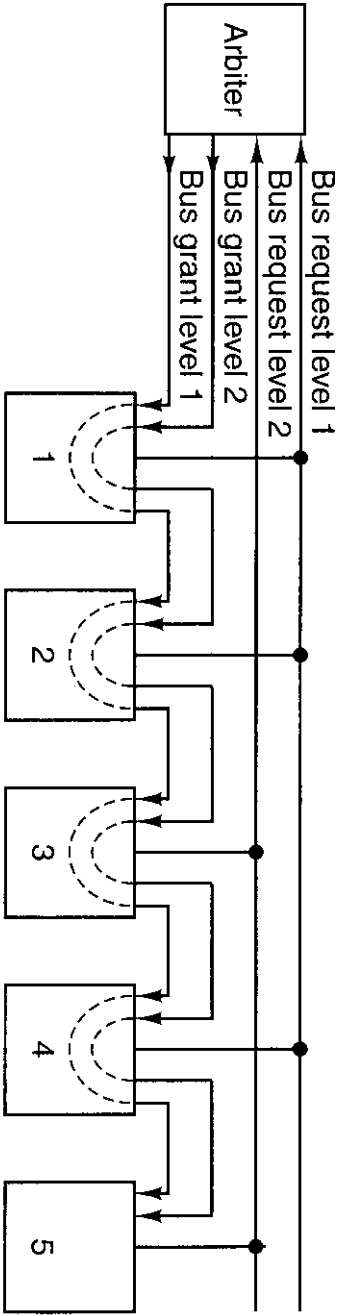
Siehe <http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?query=csma&action=Search>

Datentransfer ohne CPU-Beteiligung





(a)



Mehrere DMA-Controller

## Mehrere Bussysteme

- dual ported memory:  
<http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?query=dual+ported&action=Search>  
[http://en.wikipedia.org/wiki/Dual-ported\\_RAM](http://en.wikipedia.org/wiki/Dual-ported_RAM)
- crossbar switch: <http://en.wikipedia.org/wiki/Crossbar>

## Binäre Codierung

Bemerkung: Speicherinhalte werden je nach „Datentyp“ unterschiedlich interpretiert ([http://en.wikipedia.org/wiki/Computer\\_numbering\\_formats](http://en.wikipedia.org/wiki/Computer_numbering_formats)).

Numerische Datentypen:

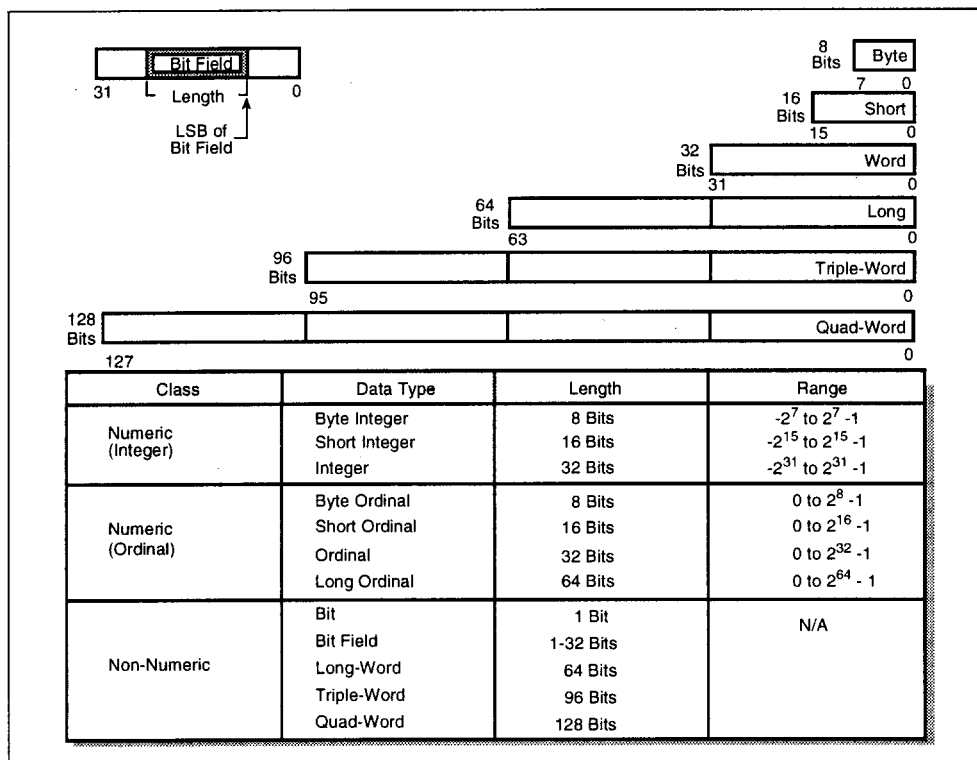


Abbildung 1.5: Datentypen

Siehe:

- Integer: [http://en.wikipedia.org/wiki/Integer\\_%28computer\\_science%29](http://en.wikipedia.org/wiki/Integer_%28computer_science%29)
  - negatives Vorzeichen: <http://en.wikipedia.org/wiki/Sign-magnitude>
- Festkommazahlen: [http://en.wikipedia.org/wiki/Fixed-point\\_arithmetic](http://en.wikipedia.org/wiki/Fixed-point_arithmetic)
- Gleitkommazahlen: [http://en.wikipedia.org/wiki/Floating\\_point](http://en.wikipedia.org/wiki/Floating_point)

- Numerische Mathematik: [http://en.wikipedia.org/wiki/Numerical\\_analysis](http://en.wikipedia.org/wiki/Numerical_analysis)
- Rundungsfehler usw.: [http://en.wikipedia.org/wiki/Floating\\_point#Problems\\_with\\_floating-point](http://en.wikipedia.org/wiki/Floating_point#Problems_with_floating-point)



Nichtnumerische Datentypen:

Der ASCII-Code (*American Standard Code for Information Interchange*) ist auch heute noch Grundlage vieler Zeichencodes.

00	NUL	01	SOH	02	STX	03	ETX	04	EOT	05	ENQ	06	ACK	07	BEL
08	BS	09	HT	0A	NL	0B	VT	0C	NP	0D	CR	0E	SO	0F	SI
10	DLE	11	DC1	12	DC2	13	DC3	14	DC4	15	NAK	16	SYN	17	ETB
18	CAN	19	EM	1A	SUB	1B	ESC	1C	FS	1D	GS	1E	RS	1F	US
20	SP	21	!	22	"	23	#	24	\$	25	%	26	&	27	'
28	(	29	)	2A	*	2B	+	2C	,	2D	-	2E	.	2F	/
30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3A	:	3B	;	3C	<	3D	=	3E	>	3F	?
40	@	41	A	42	B	43	C	44	D	45	E	46	F	47	G
48	H	49	I	4A	J	4B	K	4C	L	4D	M	4E	N	4F	O
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W
58	X	59	Y	5A	Z	5B	[	5C	\	5D	]	5E	^	5F	-
60	`	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6A	j	6B	k	6C	l	6D	m	6E	n	6F	o
70	p	71	q	72	r	73	s	74	t	75	u	76	v	77	w
78	x	79	y	7A	z	7B	{	7C		7D	}	7E	~	7F	DEL

Tabelle 1.3: ASCII-Code

Da sieben Bit, also 128 Zeichen nicht für landesspezifische Sonderzeichen ausreichen, entstanden die landesspezifischen Varianten durch Zeichenersetzung (vgl. Tabelle 1.4).

ISO Nr.	Zeichensatz	Dezimalform												
		ID	35	36	64	91	92	93	94	96	123	124	125	126
6	ANSI ASCII	0U	#	\$	@	[	\	]	^	`	{		}	~
11	Schweden: Namen	0S	#		É	Ä	Ö	Å	Ü	`	ä	ö	å	ü
10	Schweden	3S	#		@	Ä	Ö	Å	^	`	ä	ö	å	
17	Spanien	2S	£	\$	§	ı	Ñ	ı	^	`	°	ñ	ç	~
69	Frankreich	1F	£	\$	à	°	ç	§	^	μ	é	ù	è	
21	Deutschland	1G	#	\$	§	Ä	Ö	Ü	^	`	ä	ö	ü	ß
4	Großbritannien	1E	£	\$	@	[	\	]	^	`	{		}	
16	Portugal	4S	#	\$	§	Ã	Ç	Õ	^	`	ã	ç	õ	°
60	Norwegen 1	0D	#	\$	@	Æ	Ø	Å	^	`	æ	ø	å	
61	Norwegen 2	1D	§	\$	@	Æ	Ø	Å	^	`	æ	ø	å	
2	IRV		#		@	[	\	]	^	`	{		}	
15	Italien	0I	£	\$	§	°	ç	é	^	`	à	ò	è	ì

Tabelle 1.4: ISO-Austauschtabelle

Alternativ wurden acht Bit (256 Zeichen) für landesspezifische Sonderzeichen, mathematische Symbole, graphische Symbole zum Tabellendruck bzw. für Sonderzwecke (Spiele, ...) besetzt, etwa im Industriestandard PC-8 Zeichensatz (Tabelle 1.5)<sup>1</sup>.

0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241
2	↑↓	"	2	B	R	b	r	é	Æ	ó		⊥		Γ	≥
3	!!	#	3	C	S	c	s	â	ô	ú		⊢		π	≤
4	¶	\$	4	D	T	d	t	ä	ö	ñ	⊣			∑	
5	♣	§	%	5	E	U	e	u	à	ò	N			σ	
6	♠	&	6	F	V	f	v	â	û	ª				μ	÷
7		'	7	G	W	g	w	ç	ù	º				τ	≈
8	↑	(	8	H	X	h	x	ê	ÿ	¿				Φ	°
9	○	↓	)	9	I	Y	i	y	ë	Ö				Θ	·
10	→	*	:	J	Z	j	z	è	Û	¬				Ω	·
11	←	+	;	K	[	k	{	ï	ç					δ	√
12		,	<	L	\	l		î	£					∞	<sup>n</sup>
13	↔	-	=	M	]	m	}	ì		í				φ	<sup>2</sup>
14		.	>	N	^	n	~	Ä	Pt	<<				€	
15		/	?	O	_	o		Å	f	>>				∩	
	31	47	63	79	94	111	127	143	159	175	191	207	223	239	255

Tabelle 1.5: PC-8 Zeichensatz

<sup>1</sup>In dieser – wie auch in einigen der folgenden Tabellen – sind leere Codestellen entweder unbesetzt oder wegen Problemen beim Satz der entsprechenden Zeichen in diesem Skript freigelassen worden.

In Windows 3.x wurden jedoch andere Codierungen genutzt (Tabelle 1.6). Der inzwischen verabschiedete Standard der *International Standardization*

NUL			0	@	P	`	p				°	À		à	
0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
		!	1	A	Q	a	q		‘	ı	±	Á	Ñ	á	ñ
1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241
		"	2	B	R	b	r		’	ç	²	Â	Ë	â	ë
2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242
		#	3	C	S	c	s		£	³	Ã	Ö	ã	ö	
3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243
		\$	4	D	T	d	t			´	Ä	Û	ä	ü	
4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244
		%	5	E	U	e	u			µ	Å	Ö	å	ö	
5	21	37	53	69	85	101	117	133	149	165	181	197	213	229	245
		&	6	F	V	f	v			¶	Æ	Ö	æ	ö	
6	22	38	54	70	86	102	118	134	150	166	182	198	214	230	246
BEL		’	7	G	W	g	w		§	·	Ç	×	ç	÷	
7	23	39	55	71	87	103	119	135	151	167	183	199	215	231	247
BS		(	8	H	X	h	x		¨		È	Ø	è	ø	
8	24	40	56	72	88	104	120	136	152	168	184	200	216	232	248
HT		)	9	I	Y	i	y		©	¹	É	Û	é	ù	
9	25	41	57	73	89	105	121	137	153	169	185	201	217	233	249
LF		*	:	J	Z	j	z		ª	º	Ê	Ú	ê	ú	
10	26	42	58	74	90	106	122	138	154	170	186	202	218	234	250
VT	ESC	+	;	K	[	k	{		«	»	Ë	Û	ë	û	
11	27	43	59	75	91	107	123	139	155	171	187	203	219	235	251
FF		,	<	L	\	l			¬	¼	Ï	Û	ì	ü	
12	28	44	60	76	92	108	124	140	156	172	188	204	220	236	252
CR		-	=	M	]	m	}		-	½	Í	Ý	í	ý	
13	29	45	61	77	93	109	125	141	157	173	189	205	221	237	253
SO		.	>	N	^	n	~			¾	Î		î		
14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254
SI		/	?	O	-	o			-	¿	Ï	ß	ï	ÿ	
15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255

Tabelle 1.6: Zeichensatz für Windows 3.x

*Organisation* (ISO), der ISO-8859 Latin 1 (ECMA-94 Latin 1) Zeichensatz (Tabelle 1.7) setzte sich insbesondere bei Workstations und neueren Hard- und Softwareprodukten durch. Neben der Latin-1 Version existieren auch noch einige andere nationale Sonderformen des ISO 8859 Codes (vgl. Tabelle 1.8).

Das Euro-Zeichen und **ISO Latin 9**:

[http://en.wikipedia.org/wiki/ISO\\_8859-15](http://en.wikipedia.org/wiki/ISO_8859-15)

0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241
2	18	34	50	66	81	98	114	130	146	162	178	194	210	226	242
3	19	35	51	67	82	99	115	131	147	163	179	195	211	227	243
4	20	36	52	68	83	100	116	132	148	164	180	196	212	228	244
5	21	37	53	69	84	101	117	133	149	165	181	197	213	229	245
6	22	38	54	70	85	102	118	134	150	166	182	198	214	230	246
7	23	39	55	71	86	103	119	135	151	167	183	199	215	231	247
8	24	40	56	72	87	104	120	136	152	168	184	200	216	232	248
9	25	41	57	73	88	105	121	137	153	169	185	201	217	233	249
10	26	42	58	74	89	106	122	138	154	170	186	202	218	234	250
11	27	43	59	75	90	107	123	139	155	171	187	203	219	235	251
12	28	44	60	76	91	108	124	140	156	172	188	204	220	236	252
13	29	45	61	77	92	109	125	141	157	173	189	205	221	237	253
14	30	46	62	78	93	110	126	142	158	174	190	206	222	238	254
15	31	47	63	79	94	111	127	143	159	175	191	207	223	239	255

Tabelle 1.7: ISO-8859 Latin 1 (ECMA-94 Latin 1) Zeichensatz

Code Set	Name	Coverage	Approved
8859-1	Latin-1	western Europe	15 February 1987
8859-2	Latin-2	eastern Europe	15 February 1987
8859-3	Latin-3	Maltese, Catalan, Galician, Esperanto	15 April 1988
8859-4	Latin-4	Baltic and Nordic region	15 April 1988
8859-5	Cyrillic	Slavic countries	1 December 1988
8859-6	Arabic	Arab countries	15 August 1987
8859-7	Greek	Greece	15 November 1987
8859-8	Hebrew	Israel	1 June 1988
8859-9	Latin-5	8859-1 minus Iceland plus Turkey	15 May 1989

Tabelle 1.8: nationale ISO8859-Varianten

Die aktuellen ISO-8859-Varianten:

[http://en.wikipedia.org/wiki/ISO\\_8859#The\\_Parts\\_of\\_ISO\\_8859](http://en.wikipedia.org/wiki/ISO_8859#The_Parts_of_ISO_8859)

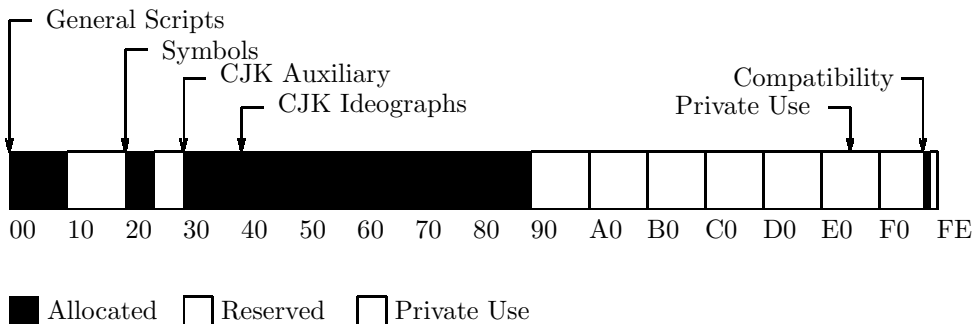
Das Varianten-Durcheinander:

<http://www.unicodecharacter.com/charsets/iso8859.html>

Eine weltweite Anwendbarkeit eines Zeichencodes kann erreicht werden, falls sechzehn Bit für die Codierung zur Verfügung stehen: *UNICODE* (<http://www.unicode.org/charts/>), den neuere Programmiersprachen wie etwa *JAVA* schon benutzen und der zur Zeit Standard auch in den GUIs der Betriebssysteme wird.

Siehe: <http://en.wikipedia.org/wiki/Unicode>

### Statistik UNICODE Version 1.0



CJK = Chinese, Japanese, Korean

Abbildung 1.6: Verteilung UNICODE

Die folgende Tabelle zeigt die Anteile des gesamten im UNICODE zur Verfügung stehenden Platzes, die verschiedenen Schrifttypen in der Version 1.0 bereits zugeteilt wurden:

	<i>Allocated</i>	<i>Unassigned</i>	<i>% Assigend</i>	
General	2336	5856	29%	
Symbols	1290	2806	31%	
CJK symbols	763	261	75%	
Hangul	2350	450	84%	
Han Compatibility	268	4	99%	(Volume 2)
Ideographic & other	20733	22275	48%	(Volume 2)
User Space	5632	N/A	N/A	
Compatibility Zone	362	133	73%	
Special	1	13		
FEFF	1	0		
FFFE, FFFF	N/A	2		
<i>Totals</i>	28706 (assigned)			
	+ 5632 (private use)			
	= 34338 (allocated)			
	52%			

## Die Zukunft des UNICODES

In Zukunft wird der UNICODE Standard um weniger verbreitete und veraltete Schrifttypen erweitert. Schrifttypen dieser Art werden jedoch nicht in ihrer ursprünglichen Form eingebunden, da sich ihr Nutzen schwer einschätzen läßt. So wird bei vielen dieser Schriften eine ausführliche Diskussion nötig sein, bis ein zufriedenstellendes Codierungsschema vorliegt. Die fünf Schriftarten *Ethiopian*, *Burmese*, *Khmer*, *Sinhala* und *Mongolian* werden zum Standard UNICODE hinzugefügt, sobald zuverlässige Informationen über sie vorliegen. Weitere Schriftarten, die für eine mögliche Aufnahme vorgesehen sind, sind

- *Inuktitut/Cree Syllabary*: Das Kommunikationsministerium von Kanada untersucht Standardisierungen von verschiedenen Dialektarten, die von Cree und/oder Inuktitut gesprochen werden und sucht Codierungsschemen.
- *Egyptian Hieroglyphics*: Ein einheitliches Codierungsschema existiert und wird vorangetrieben.
- *Korean Hangul Syllables*: Eventuell werden noch weitere Korean Hangul Dialekte hinzugefügt.

Der Unterschied zwischen der logischen Anordnung von Zeichen und der Anordnung auf dem Bildschirm zeigt die Abbildung 1.7

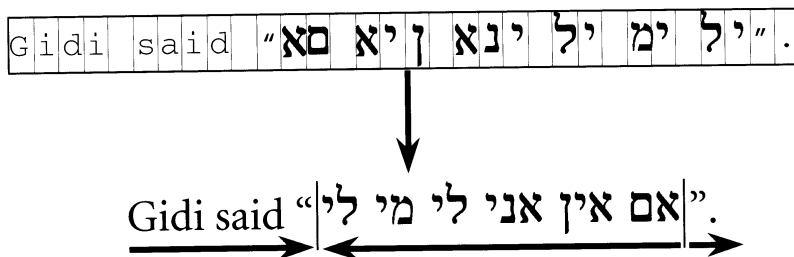


Abbildung 1.7: Bidirectional Ordering

## General Scripts

Im *General Scripts*-Bereich des UNICODEs sind alle lateinischen und nicht-ideographischen Schriftzeichen codiert:

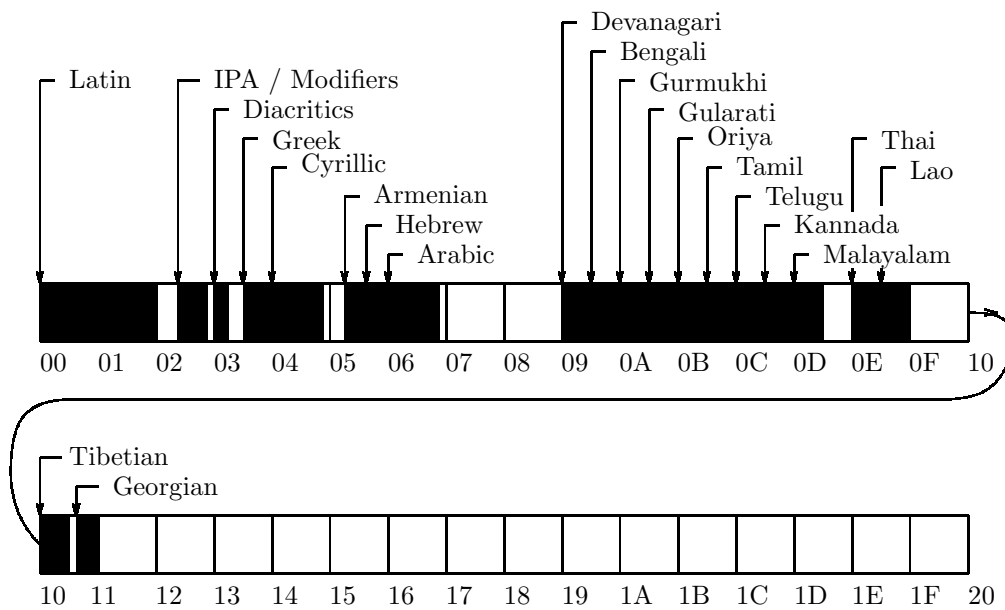


Abbildung 1.8: General Scripts

Control		ASCII						Control		Latin 1					
000	001	002	003	004	005	006	007	008	009	00A	00B	00C	00D	00E	00F
NUL	DLE		0	@	P	'	p	CTRL	CTRL		°	À		à	
SOH	DC1	!	1	A	Q	a	q	CTRL	CTRL	ı	±	Á	Ñ	á	ñ
STX	DC2	"	2	B	R	b	r	CTRL	CTRL		²	Â	Ò	â	ò
ETX	DC3	#	3	C	S	c	s	CTRL	CTRL	£	³	Ã	Ó	ã	ó
EOT	DC4	\$	4	D	T	d	t	CTRL	CTRL		,	Ä	Ô	ä	ô
ENQ	NAK	%	5	E	U	e	u	CTRL	CTRL		µ	Å	Õ	å	õ
ACK	SYN	&	6	F	V	f	v	CTRL	CTRL		¶	Æ	Ö	æ	ö
BEL	ETB	'	7	G	W	g	w	CTRL	CTRL	§	·	Ç	×	ç	÷
BS	CAN	(	8	H	X	h	x	CTRL	CTRL	¨	¸	È	Ø	è	ø
HT	EM	)	9	I	Y	i	y	CTRL	CTRL	©	¹	É	Ù	é	ù
LF	SUB	*	:	J	Z	j	z	CTRL	CTRL	ª	º	Ê	Ú	ê	ú
VT	ESC	+	;	K	[	k	{	CTRL	CTRL	«	»	Ë	Û	ë	û
FF	FS	,	<	L	\	l		CTRL	CTRL	¬	¼	Ì	Ü	ì	ü
CR	GS	-	=	M	]	m	}	CTRL	CTRL	-	½	Í	Ý	í	ý
SO	RS	.	>	N	^	n	~	CTRL	CTRL	-	¾	Î		î	
SI	US	/	?	O	_	o	DEL	CTRL	CTRL	-	¿	Ï	ß	ï	ÿ

Tabelle 1.9: UNICODE Version 1.0, Character Blocks 0000-00FF

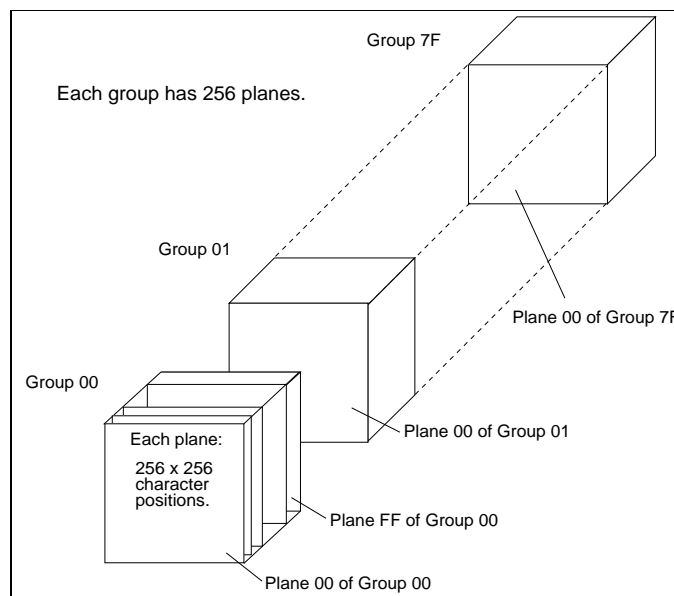


2000..206F	Zeichen für die Zeichensetzung: ,,:;“„ ...
2070..209F	Subscripts und Superscripts: <sup>2,3,4</sup> , ...
20A0..20CF	Währungssymbole: £,\$, ...
20D0..20FF	diakretische Zeichen: ←, →, ...
2100..214F	buchstabenähnliche Zeichen: ℱ, °F, ...
2150..218F	Zahlen: $\frac{1}{3}$ , I, VII, ...
2190..21FF	Pfeile: ↑, ↗, ...
2200..22FF	mathematische Sonderzeichen: ∀, ∃, ∈, ...
2300..23FF	verschiedene technische Sonderzeichen: #, ...
2400..243F	Symbole für Control-Zeichen: NUL, ESC, ...
2440..245F	OCR-Zeichen
2460..24FF	eingerahmte alphanumerische Zeichen: ©, ...
2500..257F	Formular- und Diagrammzeichen: †, ‡,   , ...
2580..259F	Blockgraphik-Zeichen
25A0..25FF	graphische Symbole
2600..26FF	verschiedene Dingbats
2700..27BF	Zapf-Dingbats
...	
3000..303F	CJK-Symbole
3040..309F	Hiragana
30A0..30FF	Katakana
...	

Tabelle 1.10: Weitere Zeichenbereiche

Der UNICODE stellt weitgehende Kompatibilität zu bestehenden Codes durch (verschobenes) Einfügen oder Bereitstellen von Code-Umwandlungstabellen her: 0000..007F entspricht ASCII. Für andere Codes werden UNICODE-Übersetzungstabellen bereitgestellt, z.B. für UNICODE zu SGML (Tabelle 1.12), UNICODE zu Postscript (Tabelle 1.11) oder UNICODE zu MacIntosh (Tabelle 1.13). Analoge Tabellen gibt es zur Übersetzung von UNICODE zu Microsoft Windows, zu PC Code Page Mappings (Latin, Greek,...), zu EBCDIC Code Page Mappings und weiteren.

Die Notwendigkeit, Control-Codes anderer Codierungen auch verfügbar zu haben, und die Weigerung von Japan und Korea, die vereinheitlichte CJK-Codierung (die mit 19000 statt insgesamt über 31000 Codepositionen ausgekommen wäre) zu akzeptieren, führt zum 32Bit Zeichencode *ISO 10646*, der in seinen ersten 65536 Codes die Zeichen des Unicodes benutzt:



Näheres zum ISO- bzw. Unicode lese man bei

<http://wwwinfo.cern.ch/asdoc/WWW/publications/ictp99/ictp99N2705.html>

und

<http://www.unicode.org/>

## Unicode 4.0.0 kommt mit 1226 neuen Zeichen

/28.04.2003 13:56 /

Das **Unicode Konsortium**[1] hat die Major-Version **Unicode 4.0.0**[2] vorgestellt. In der neuen Version wurden 1226 neue Zeichen integriert. Wesentliche Bestandteile von Unicode 4.0 sind neue Symbole für den mathematisch und technischen Bereich, viele individuelle Zeichen wurden zu den Schriftzeichen für Sprachen wie Indisch, Khmer oder Arabisch hinzugefügt. Außerdem wurde das neue Release um die Zeichensätze von Minderheitensprachen wie Limbu, Tai Le oder Osmyana aufgestockt. Auch die sonst oft vernachlässigten historischen Schriften bekamen eine Erweiterung. Damit reagierte das Consortium auf **Probleme bei vorherigen Unicode-Versionen**[3]. Diese waren nicht zur Formulierung von XML-Tags für einige historische Zeichensätze wie Alt-Mongolisch, Burmesisch oder einige kanadische Indianschriften fähig.

95

Unicode ist ein System, in dem die Zeichen oder Elemente aller bekannten Schriftkulturen und Zeichensysteme computertauglich festgehalten werden. Durch dieses System wird es möglich, einem Computer "weltweit" zu sagen, welches Zeichen man dargestellt bekommen will. Voraussetzung ist natürlich, dass der Computer beziehungsweise das ausgeführte Programm das Unicode-System unterstützt. Der Unicode-Standard wird von Industriegrößen wie Apple, Hewlett-Packard, IBM, Microsoft, Oracle, SAP oder Sun eingesetzt. Genauere Informationen über die neue Version finden sich in der **Dokumentation zu Unicode 4.0.0**[4] und in der **Unicode Character Database**[5]. (see[6]/c't)

### URL dieses Artikels:

<http://www.heise.de/newsticker/data/see-28.04.03-002/>

### Links in diesem Artikel:

- [1] <http://www.unicode.org/>
- [2] <http://www.unicode.org/versions/Unicode4.0.0/>
- [3] <http://www.heise.de/newsticker/data/hps-18.10.02-000/>
- [4] <http://www.unicode.org/versions/Unicode4.0.0/>
- [5] <http://www.unicode.org/ucd/>
- [6] <mailto:see@ct.heise.de>

UNIC	ISO Latin1 StdEnc	ZapfDB Symbol	Adobe glyph name	Unicode character name
0020	20	20	space	SPACE
0021	21	21	exclam	ECLAMATION MARK
0022	22	22	quotedbl	QUOTATION MARK
0023	23	23	numbersign	NUMBERSIGN
0024	24	24	dollar	DOLLAR SIGN
0025	25	25	percent	PERCENT SIGN
0026	26	26	ampersand	AMPERSAND
0027	A9		quotesingle	APOSTROPHE-QUOTE
0028	28	28	parenleft	OPENING PARENTHESIS
0029	29	29	parenright	CLOSING PARENTHESIS
002A	2A	2A	asterisk	ASTERISK
002B	2B	2B	plus	PLUS SIGN
002C	2C	2C	comma	COMMA
002D	2D	AD	hyphen	HYPHEN-MINUS
002D		2D	minus	HYPHEN-MINUS
002E	2E	2E	period	PERIOD
002F	2F	2F	slash	SLASH
0030	30	30	zero	DIGIT ZERO
0031	31	31	one	DIGIT ONE
0032	32	32	two	DIGIT TWO
0033	33	33	three	DIGIT THREE
0034	34	34	four	DIGIT FOUR
0035	35	35	five	DIGIT FIVE
0036	36	36	six	DIGIT SIX
0037	37	37	seven	DIGIT SEVEN
0038	38	38	eight	DIGIT EIGHT
0039	39	39	nine	DIGIT NINE
003A	3A	3A	colon	COLON
003B	3B	3B	semicolon	SEMIKOLON

Tabelle 1.11: UNICODE to Adobe Standard Mappings

UNIC	6862.2	SGML	Unicode character name
0021		excl	EXCLAMATION MARK
0023		num	NUMBER SIGN
0024		dollar	DOLLAR SIGN
0025		percent	PERCENT SIGN
0026		amp	AMPERSAND
0027		quot	APOSTROPHE-QUOTE
0028		lpar	OPENING PARENTHESIS
0029		rpar	CLOSING PARENTHESIS
002A		ast	ASTERISK
002B	05.00	plus	PLUS SIGN
002C		comma	COMMA
002D		hyphen	HYPHEN-MINUS
002E		period	PERIOD
002F		sol	SLASH
003A		colon	COLON
003B		semi	SEMICOLON
003C		lt	LESS-THAN SIGN
003D		equals	EQUALS SIGN
003E		gt	GREATER-THAN SIGN
003F		quest	QUESTION MARK
0040		commat	COMMERCIAL AT
005B		lsqb	OPENING SQUARE BRACKET
005C		bsol	BACKSLASH
005D		rsqb	CLOSING SQUARE BRACKET
005E		circ	SPACING CIRCUMFLEX
005F		lowbar	SPACING UNDERSCORE
0060		grave	SPACING GRAVE
007B		lcub	OPENING CURLY BRACKET
007C		verbar	VERTICAL BAR
007D		rcub	CLOSING CURLY BAR
007E		tilde	TILDE
00A0		nbsp	NON-BREAKING SPACE
00A1		ixcl	INVERTED EXCLAMATION MARK
00A2		cent	CENT SIGN
00A3		pound	POUND SIGN

Tabelle 1.12: The UNICODE to SGML (ISO DIS 6862.2) Mappings

UNIC	ROM	SYM	GRK	GK2	HEB	ARB	NAME
0020	20	20	20	20	20/A0	21/A1	SPACE
0021	21	21	21	21	21/A1	21/A1	ECLAMATION MARK
0022	22		22	22	22/A2	22/A2	QUOTATION MARK
0023	23	23	23	23	23/A3	23/A3	NUMBERSIGN
0024	24		24	24	24/A4	24/A4	DOLLAR SIGN
0025	25	25	25	25	25/A5	25	PERCENT SIGN
0026	26	26	26	26	26	26/A6	AMPERSAND
0027	27		27	27	27/A7	27/A7	APOSTROPHE-QUOTE
0028	28	28	28	28	28/A8	28/A8	OPENING PARENTHESIS
0029	29	29	29	29	29/A9	29/A9	CLOSING PARENTHESIS
002A	2A		2A	2A	2A/AA	2A/AA	ASTERISK
002B	2B	2B	2B	2B	2B/AB	2B/AB	PLUS SIGN
002C	2C	2C	2C	2C	2C/AC	2C	COMMA
002D	2D		2D	2D	2D/AD	2D/AD	HYPHEN-MINUS
002E	2E	2E	2E	2E	2E/AE	2E/AE	PERIOD
002F	2F	2F	2F	2F	2F/AF	2F/AF	SLASH
0030	30	30	30	30	30/B0	30	DIGIT ZERO
0031	31	31	31	31	31/B1	31	DIGIT ONE
0032	32	32	32	32	32/B2	32	DIGIT TWO
0033	33	33	33	33	33/B3	33	DIGIT THREE
0034	34	34	34	34	34/B4	34	DIGIT FOUR
0035	35	35	35	35	35/B5	35	DIGIT FIVE
0036	36	36	36	36	36/B6	36	DIGIT SIX
0037	37	37	37	37	37/B7	37	DIGIT SEVEN
0038	38	38	38	38	38/B8	38	DIGIT EIGHT
0039	39	39	39	39	39/B9	39	DIGIT NINE
003A	3A	3A	3A	3A	3A/BA	3A/BA	COLON
003B	3B	3B	3B	3B	3B/BB	3B	SEMICOLON
003C	3C	3C	3C	3C	3C/BC	3C/BC	LESS-THAN SIGN
003D	3D	3D	3D	3D	3D/BD	3D/BD	EQUALS-SIGN
003E	3E	3E	3E	3E	3E/BE	3E/BE	GREATER-THAN SIGN
003F	3F	3F	3F	3F	3F/BF	3F	QUESTION MARK
0040	40		40	40	40	40	COMMERCIAL AT
0041	41		41	41	41	41	LATIN CAPITAL LETTER A
0042	42		42	42	42	42	LATIN CAPITAL LETTER B
0043	43		43	43	43	43	LATIN CAPITAL LETTER C
0044	44		44	44	44	44	LATIN CAPITAL LETTER D
0045	45		45	45	45	45	LATIN CAPITAL LETTER E
0046	46		46	46	46	46	LATIN CAPITAL LETTER F
0047	47		47	47	47	47	LATIN CAPITAL LETTER G
	:						

Tabelle 1.13: UNICODE to Macintosh Mappings

Inzwischen benötigt man für den Unicode (bzw. ISO-10646-Code) 17 (= 1+16) 16Bit-Blöcke, die sogenannten Planes mit Zeichencodes von 000000H...10FFFFH, von denen in Unicode Version 4 96382 Positionen benutzt werden:

[http://de.wikipedia.org/wiki/Liste\\_der\\_Unicode-Bl%C3%B6cke](http://de.wikipedia.org/wiki/Liste_der_Unicode-Bl%C3%B6cke)

Durch eine geschickte Einbettung verschiedener Codebereiche (insbesondere des ASCII-Codes) benötigt ein Unicodezeichen in der sogenannten **UTF-8-Codierung** (<http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?query=UTF-8&action=Search>) zwischen ein und vier Byte; reine ASCII-Texte besitzen je Zeichen genau ein Byte Länge:

<http://de.wikipedia.org/wiki/UTF-8>

Hinweise zur Umcodierung von Textdateien verschiedenster Codes ineinander finden Sie unter:

<http://www.linuxdocs.org/HOWTOs/German-HOWTO-7.html#ss7.2>

Neben der lästigen Codeumwandlungen von Textdateien führt der Übergang von immer mehr Benutzerumgebungen auf UTF-8 zu folgenden wichtigen Konfigurationsaufgaben für Anbieter von Web-Inhalten:

- Webseiten müssen bei direkter Benutzung von mehr als reinen ASCII-Zeichen im HTML-Kopf den benutzten **charset** spezifizieren, etwa durch eine Zeile der Art

```
<meta http-equiv="Content-Type" content="Text/html"; charset="iso-8859-1">
```

Siehe dazu <http://www.cs.tut.fi/~jkorpela/html/chars.html>

sowie <http://www.w3.org/TR/REC-html40/charset.html>.

- Emails müssen mit richtig spezifiziertem **charset** versandt werden:

<http://www.uni-koblenz.de/~pidde/lupe/mail.html>.

Übersicht zu den MIME-Charsets: <http://www.iana.org/assignments/character-sets>.

- URLs sollten zum Schutz vor Adress-Fälschungen verschiedene Zeichenbereiche nur eingeschränkt mischen dürfen

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/57201&words=IETF>

und Web-Browser sollten verschieden codierte aber vom Druckbild gleiche Zeichen unterschiedlich darstellen:

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/56792&words=Umlaut>.

Siehe auch <http://de.wikipedia.org/wiki/IDNA>

und <http://www.faqs.org/rfcs/rfc3492.html>

sowie <http://dcomnet.de/cgi-bin/punycod/punycod.cgi>.

Bemerkung: Byteadressierte Computersysteme

**Memory Contents for Little and Big Endian Example**

ADDRESS	DATA
1000H	12H
1001H	34H
1002H	56H
1003H	78H
1004H	9AH
1005H	BCH
1006H	DEH
1007H	FOH

Abbildung 1.9: Die Bytes im Speicher ...

Access	Example	Register Contents (Little Endian)	Register Contents (Big Endian)
Byte at 1000H	ldob 0x1000, r3	12H	12H
Short at 1002H	ldos 0x1002, r3	7856H	5678H
Word at 1000H	ld 0x1000, r3	78563412H	12345678H
Long-Word at 1000H	ldl 0x1000, r4	78563412H (r4) F0DEBC9AH (r5)	12345678H (r4) 9ABCDEF0H (r5)

Abbildung 1.10: ... und ihre Reihenfolge im Computerwort

**Aufgabe 1.9** Lesen Sie <http://en.wikipedia.org/wiki/UTF-16> und erläutern Sie in eigenen Worten die Probleme der Byteanordnung (hier) am Beispiel der Datenübertragung von UTF-16-Zeichenketten.



## Im Inneren der von-Neumann-CPU

In einer CPU wurden zunächst

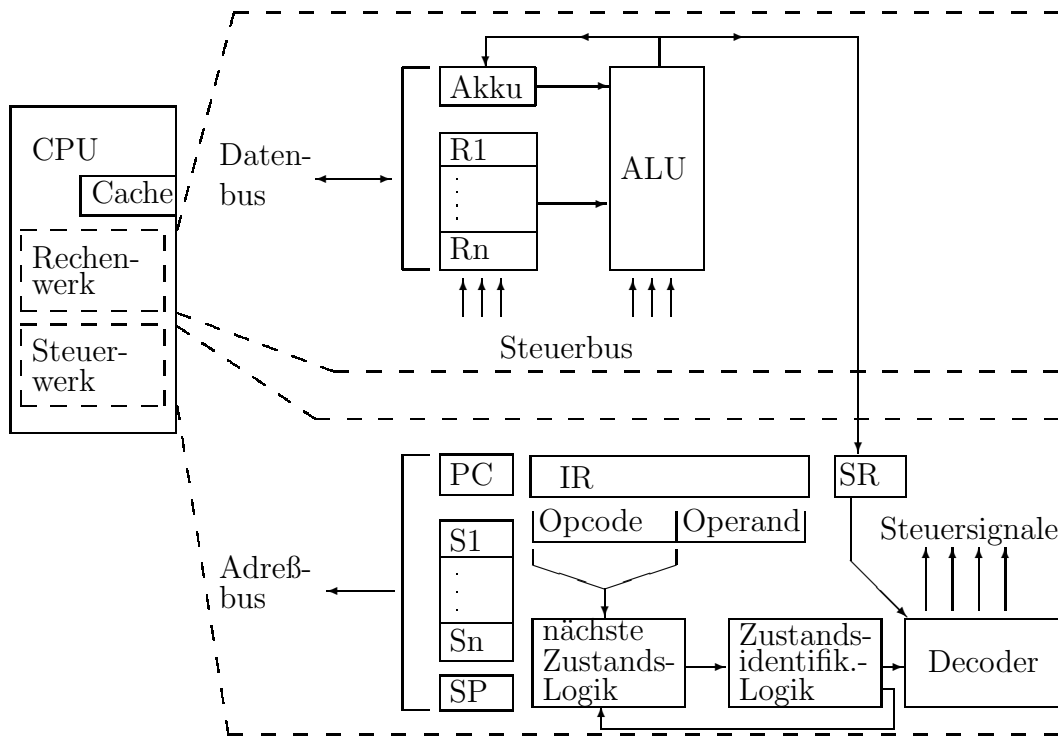


Abbildung 1.11: CPU

die einzelnen Aktionen hardwaremäßig (1 aus N Auswahl der gewünschten Aktion der ALU über CPU-interne Steuerleitungen) realisiert. Die Umsetzung gewisser Bits der Befehlswortes (im IR) in die Aktivierung der richtigen ALU-Steuerleitung geschieht durch Hardware (Halbleiter-Gatter-Schaltungen).

Die Maschinenbefehle einer typischen Intel-CPU sind unter

[http://www.emu8086.com/assembly\\_language\\_tutorial\\_assembler\\_reference/8086\\_instruction\\_set.html](http://www.emu8086.com/assembly_language_tutorial_assembler_reference/8086_instruction_set.html) inspezierbar.

## 1.2 Modularer Aufbau von Betriebssystemen

Betriebssysteme haben die folgenden Aufgaben:

- Auftragsverwaltung (Jobs)
- Betriebsmittelverwaltung:
  - Hauptspeicherverwaltung
  - Dateiverwaltung
  - E/A-Steuerung / Gerätetreiber
  - Ablaufsteuerung / Prozess-/Taskverwaltung
  - Zugriffskontrolle
  - Ausnahmebehandlung
  - ...
- Dienste / Services

Sie setzen sich dabei modular aus Subsystemen zusammen:

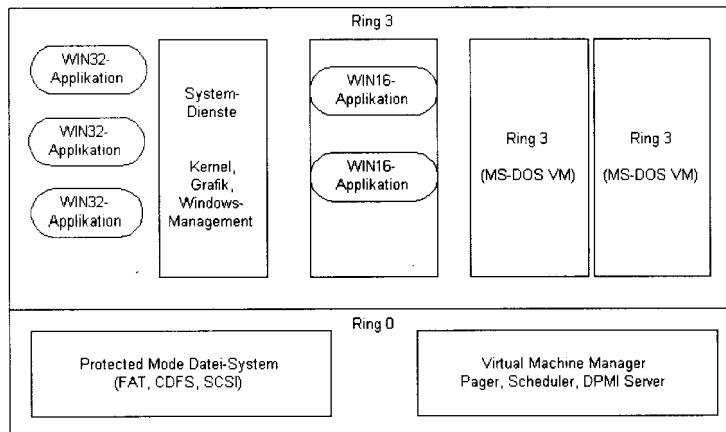


Abbildung 1.12: Windows 95/98

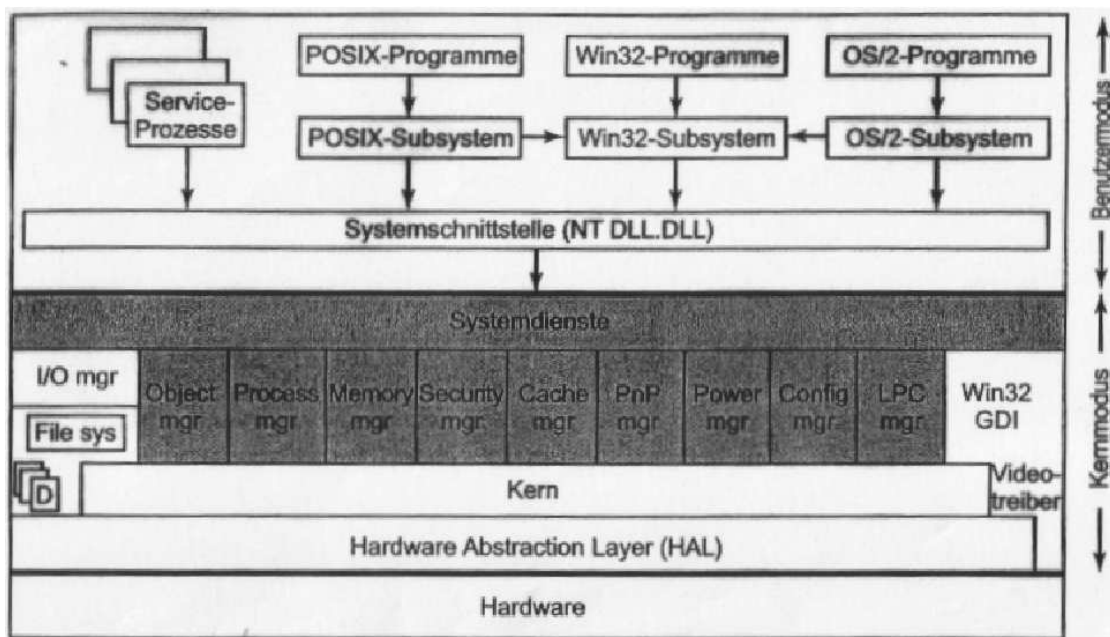


Abbildung 1.13: Windows 2000

Bemerkungen:

[http://files.hanser.de/hanser/docs/20040401\\_244515439-7843.3-446-21497-6.pdf](http://files.hanser.de/hanser/docs/20040401_244515439-7843.3-446-21497-6.pdf)

<http://lexikon.freenet.de/Betriebssystem>

<http://msdn.microsoft.com/msdnmag/issues/01/12/XPKernel/default.aspx>

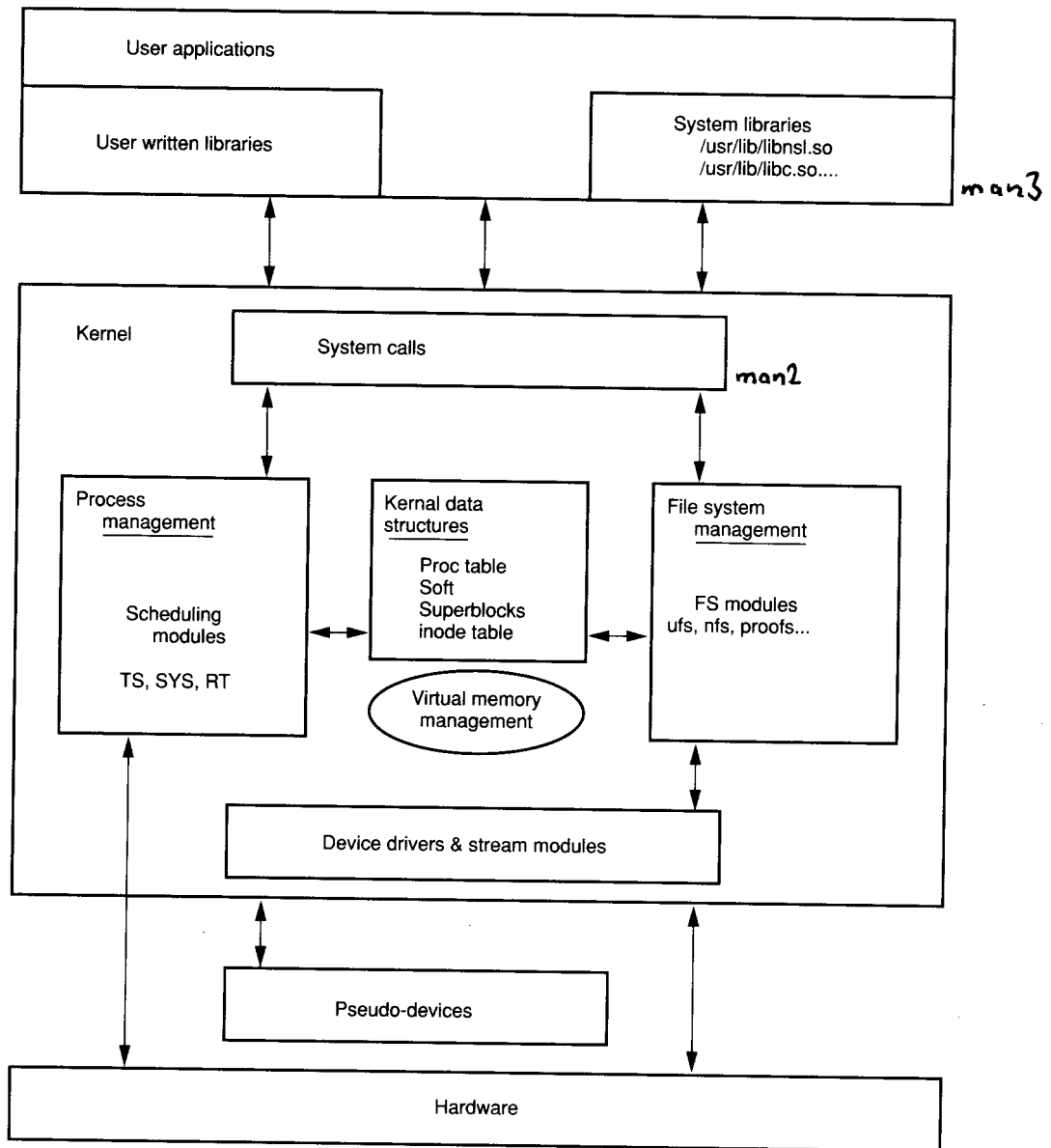


Abbildung 1.14: Solaris/Linux

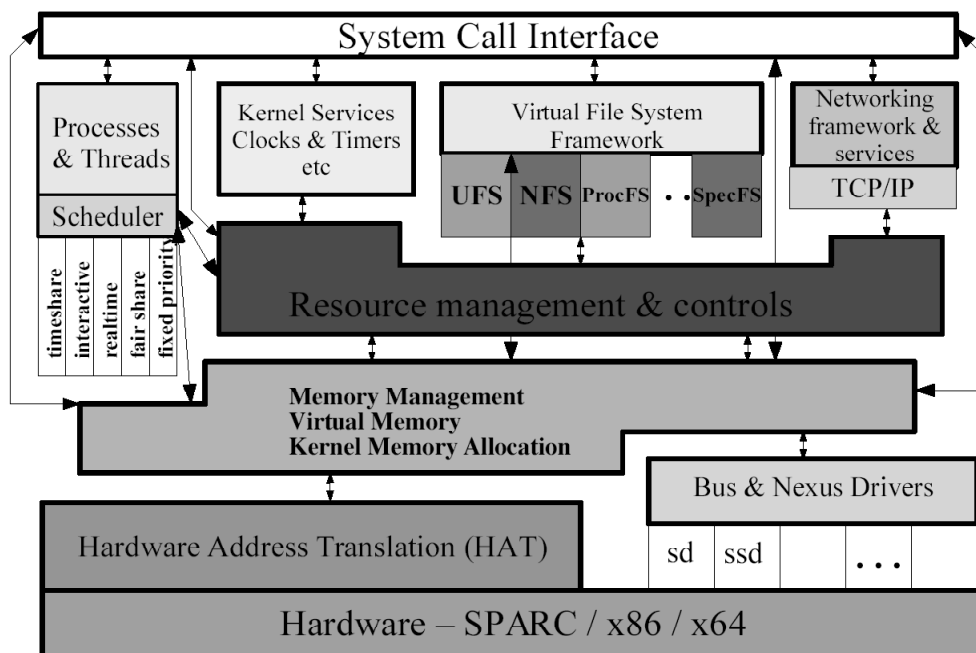


Abbildung 1.15: Solaris Kernel Overview

Bemerkungen:

[http://plg.uwaterloo.ca/~itbowman/CS746G/a1/#Toc\\_2.3](http://plg.uwaterloo.ca/~itbowman/CS746G/a1/#Toc_2.3)

<http://www.linuxhq.com/guides/TLK/tlk.html>

In der folgenden Skizze

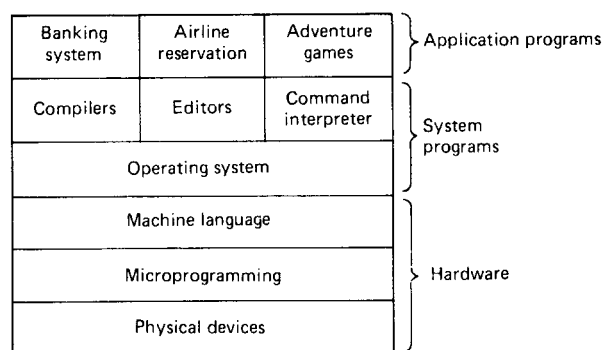


Abbildung 1.16: Microcode und Betriebssystem

wird auf Microprogramming (Microcode) hingewiesen. Was ist darunter zu verstehen?

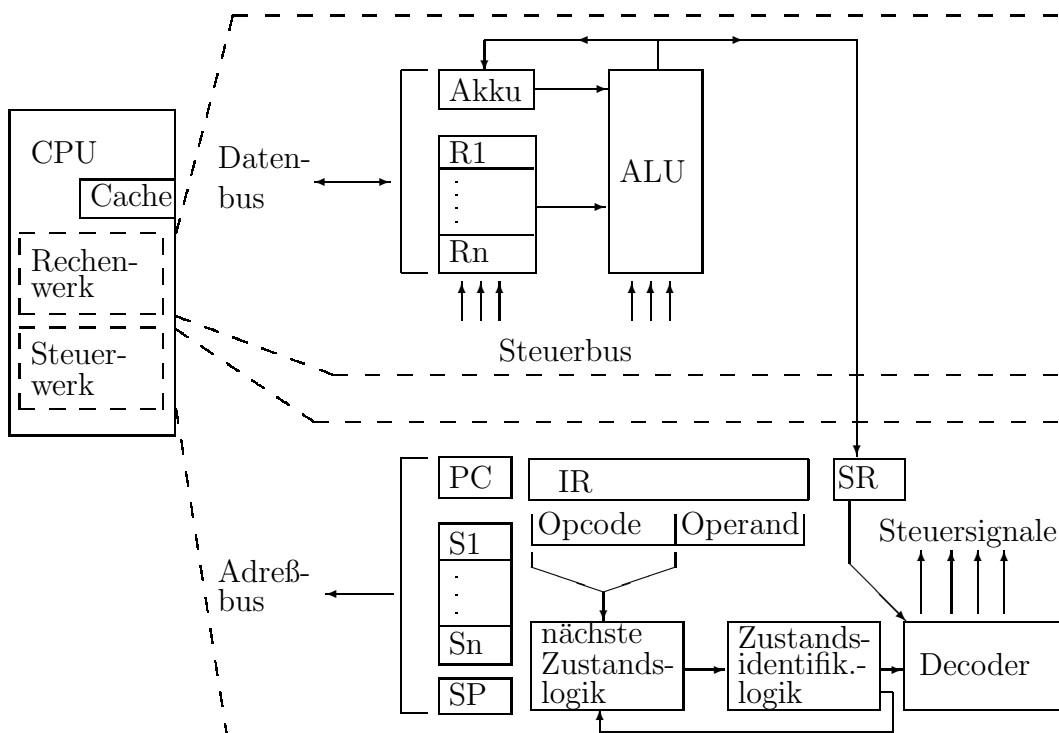
## 1.3 CISC/RISC

### 1.3.1 CISC

Traditionelle CISC's (= complex instruction set computers) sind:

- IBM360
- DEC VAX
- Intel 80386 (80x86)
- Motorola 68030 (680x0)
- ...

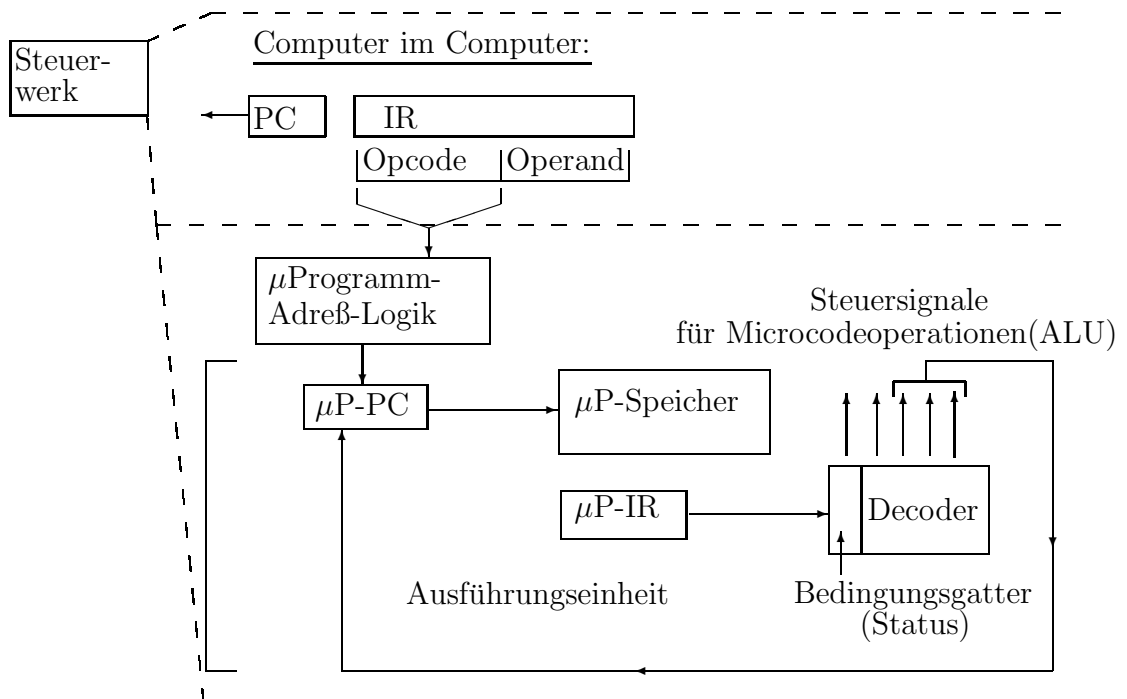
Abbildung 1.17: CISC-CPU-Aufbau



In Zeiten, als der Zeitraum zur Übertragung eines Datenwortes vom Speicher in ein Register oder umgekehrt Größenordnungen länger dauerte als ein

einfacher logischer oder arithmetischer Primitiv-Befehl, mußte zunächst die Programmabarbeitung durch *wait-states* künstlich verlangsamt werden. Man versuchte dann aus Gründen der Ökonomie, eine CPU immer komplexere Primitiv-Befehle (deren Ausführungszeit lediglich anfangs immer noch höchstens eine „Datentransferzeit Register in Speicher“ lang war) ausführen zu lassen. In der damaligen Zeit waren dafür jedoch nicht genügend Schalteroperationen auf einem Chip realisierbar, weshalb man das Rechenwerk der CPU wiederum als Computer aufbaute:

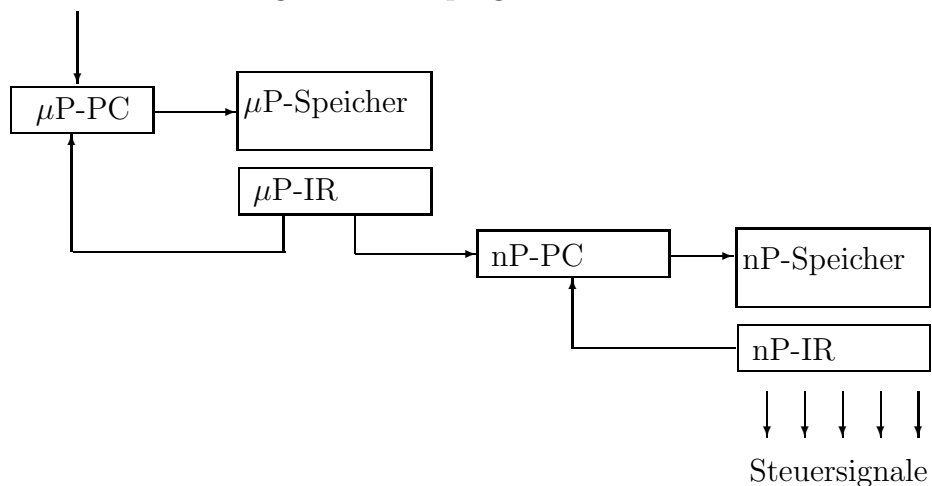
Abbildung 1.18: Microprogrammierte CISC-CPU



Software („SW“) reduziert den Schaltungsaufwand (= Anzahl der Gatter, Anzahl der Leitungen,...) und „interpretiert“ den Maschinencode der CPU (Computer im Computer).

Eventuell werden sogar „noch“ Nanoprogramme benutzt (68000er,...):

Abbildung 1.19: Nanoprogrammierte CISC-CPU



### Das CISC-Designprinzip:

Reduziere die „semantic gap“ zwischen Maschinensprache und Hochsprache durch

- viele komplexe Maschinenbefehle ( $\gtrsim 200$ ), etwa „case“, „while“, ...
- viele Adressierungsmodi, etwa für „Felder“, „Verbunde“, ...
- Unterprogramm-Management im Maschinencode

unter besonderer Beachtung der mindestens um den Faktor 10 langsameren Transferbefehle zum/vom Speicher.

Vergleiche <http://en.wikipedia.org/wiki/CISC>.



### 1.3.2 „Überalterung“ von CISC

Ungefähr 1970 wird

- die Komplexität des Microprogramms immer schlechter zu managen
- der Speicherzugriff schneller
- lohnt sich ein Redesign aufgrund von Messungen „typischer“ Anwendungen:

Tabelle 1.14: Analyse typischer Computeranwendungen auf ihre Instruktionstypen hin

%-Verteilung	SAL Operating System Tanenbaum (1978)	XPL Systempgm. Wortman (1972)	FORTRAN num. Pgm. Knuth (1971)	C Systempgm. Patterson (1982)	Pascal Systempgm. Patterson (1982)
:=	47 <sup>1)</sup>	55	51	38	45
if	17	17	10	43	29
call	25	17	5	12	15
loop	6	5	9	3	5
goto	0	1	9	3	0
sonstiges	5	5	16	1	6

Dabei sind in <sup>1)</sup> 80% der Wertzuweisungen von dem einfachen Typ „Variable := Wert“.

**Konsequenz:** Der Overhead durch das komplexe Microprogramm und die vielen Adressierungsmodi wird nur sehr selten auch ausgenutzt.

## **MOV (80286/80386)**

### **Move**

**Instruction:** Move

**Typical clocks:** (80286) 2-19, (80386) 2-22

**Description:** Copies the source to the destination

**Operation:** There are several MOV instructions, all of which have the same function. They all copy the source operand's contents into the destination operand without destroying the source.

**Syntax:** MOV *destination,source*

**Flags affected:** None

**Flags undefined:** None

**Protected mode exceptions:** A general protection exception, stack fault exception, or descriptor not present exception may be generated if a segment register is being loaded. A general protection exception also may be generated if the destination is in a nonwritable segment. If the CS, DS, or ES segment contains an illegal memory operand effective address, a general protection exception is generated. If the SS segment contains an illegal address, a stack fault exception is generated.

**Real address mode exceptions:** When a word operand is at offset 0FFFFH, INT 13 is generated.

**80386 note:** Word size is 32 bits. Memory addresses are formed using 32-bit effective addresses or 32-bit displacement.

Abbildung 1.20: Der x86 MOV-Befehl

## **MUL (80286/80386)**

**Multiply, unsigned**

**Instruction:** MUL

**Typical clocks:** (80286) 13-21, (80386) 9-41

**Description:** Performs unsigned multiply ( $AX = AL \times$  specified byte)  
Performs unsigned multiply ( $DX:AX = AX \times$  specified word)

**Operation:** A byte operand of the MUL instruction causes the byte to be multiplied by the AL register and the result left in the AX register. CF and OF are reset to 0 if AH is 0; otherwise, they are set to 1.

A word operand of the MUL instruction causes the word to be multiplied by the AX register and the result left in DX:AX. DX contains the high-order 16 bits of the result. CF and OF are reset to 0 if DX is 0; otherwise, they are set to 1. See the 80386 note.

**Syntax:** MUL *source*

**Flags affected:** OF, CF

**Flags undefined:** SF, ZF, AF, PF

**Protected mode exceptions:** If the CS, DS, or ES segment contains an illegal memory operand effective address, a general protection exception is generated. If the SS segment contains an illegal address, a stack fault exception is generated.

**Real address mode exceptions:** When a word operand is at offset 0FFFFH, INT 13 is generated.

Abbildung 1.21: Der x86 MUL-Befehl

### 1.3.3 RISC (= reduced instruction set computer)-Designprinzip

- Analysiere Anwendungen, um die Schlüsseloperationen zu finden.
- Reduziere unter allen Umständen die „Datenpfadzykluszeit“ (Register → ALU → Register) für diese Operationen, also: *keine* Microcode Interpretation (Maschinencode = Microcode)
- Jedes „neue“ Feature ist nur zuzulassen, wenn es häufig benutzt wird und die Mehrzahl der existierenden nicht verlangsamt.

(Wiederhole diesen Design-Zyklus für die anderen CPU-Ressourcen: Cache, Speichermanagement, Gleitkomma-Koprozessoren, ...)

Tabelle 1.15: Von CISC zu RISC

	CISC			RISC		
	IBM 370/168	VAX 11/780	Xerox Dorado	IBM 801	Berkeley RISC I	Stanford MIPS
Jahr	1973	1978	1978	1980	1981	1983
# Operationen	208	303	270	120	39	55
Microcode-Größe [byte]	54K	61K	17K	—	—	—
Befehlslänge [byte]	2–6	2–57	1–3	4	4	4
Ausführungsmodell	reg-reg reg-mem mem-mem	reg-reg reg-mem mem-mem	stack	reg-reg	reg-reg	reg-reg

Siehe auch: [http://en.wikipedia.org/wiki/Reduced\\_Instruction\\_Set\\_Computer](http://en.wikipedia.org/wiki/Reduced_Instruction_Set_Computer)

CISC- und RISC-Architekturen lassen sich durch einige charakteristische Merkmale voneinander unterscheiden:

Tabelle 1.16: Unterscheidungsmerkmale CISC/RISC

Bei RISC üblich	Bei CISC üblich
1. einfache Befehle, 1 Befehl/Zyklus	komplexe Befehle, 1 Befehl/viele Zyklen
2. LOAD/STORE-Architektur (nur LOAD/STORE greift auf Speicher zu)	mem-reg, mem-mem Befehle
3. pipelined	kaum pipelined
4. Hardware, kein Microcode	Microcode
5. feste Befehlslänge	variable Befehlslänge (1...17 byte beim 80386)
6. Wenige Befehle, wenige Adressierungsmodi	viele Befehle und Adressierungsmodi
7. Komplexität im Compiler	Komplexität im Microprogramm
8. Registerstack	feste Registermenge

*Bemerkungen:*

zu 1. Deswegen evtl. sogar keine Multiplikation/Division, die dann in Runtime-Bibliotheken angeboten werden muß.

zu 2. Vorhandene RISC-Varianten: (load|store) (signed|unsigned) (byte|halfword|word)

zu 3. – delayed load/store: Codereorganisation bzw. NOP's

– Pipeline:

Zyklus	1	2	3	4	5	6	7 ...
instr1	fetch	decode	exec	write			
instr2		fetch	decode	exec	write		
instr3			fetch	decode	exec	write	
⋮							

zu 4. in RISC viel Chip-Platz frei: mehr Register realisierbar

zu 5. Die variable Länge benötigte Microcode (oder aufwendige Decoder).

zu 6. – Befehlsdecoder-Länge wächst exponentiell mit der Anzahl der Befehle.

– Geschwindigkeit/Komplexität: wenige Adressierungsmodi

zu 7. In RISC-Systemen sind gute Compiler unumgänglich.

zu 8. Registerstack:

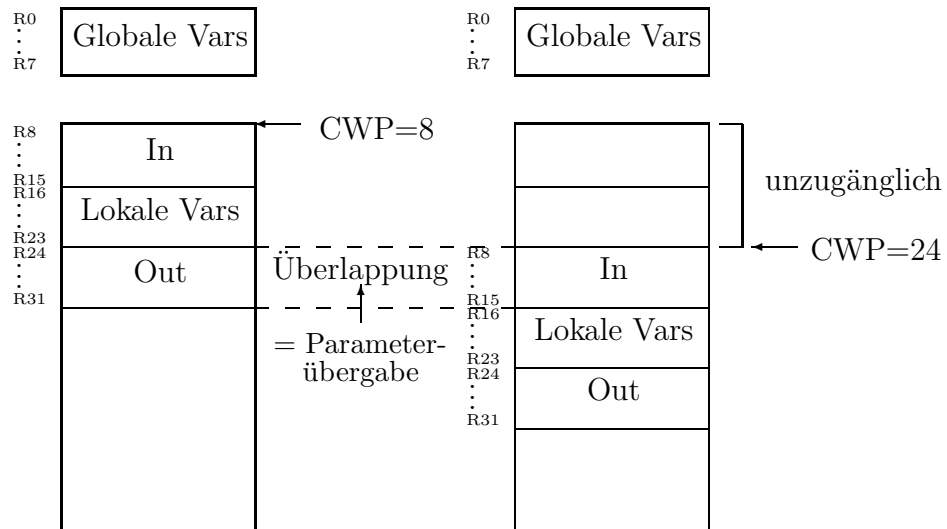
– ~ 500 Register, von denen 32 gleichzeitig verfügbar sind

– überlappende Register-Fenster (s.u.)

– Register-Mehrfachbenutzung („Lebenszeit“ von Variablen)

Der [Registerstack](http://en.wikipedia.org/wiki/Register_window) ([http://en.wikipedia.org/wiki/Register\\_window](http://en.wikipedia.org/wiki/Register_window)) ist auch bei CISC möglich, wenn genügend Chip-Platz verfügbar ist.

Abbildung 1.22: Registerstack am Beispiel SPARC



### 1.3.4 Historischer CISC-RISC CPU-Vergleich (1992f.)

Tabelle 1.17: RISC-CPU's im Vergleich zu Intel 80x86-CPU's

	MIPS 4400	SPARC SS20	DEC- Alpha	Intel Pentium P5	HP PA7100	IBM Power 6264	MIPS R4000SC	SPARC SS10	Intel 486DX2
Taktrate (MHz)	200	75	nom.200	66	100	62,5	100	40	66
# Pipelinestufen		4	7		5	6	8	4	5
max # Instr./Zyklus		3	2		2	4	1	3	1
Cache Instr.	16 KB	20 KB	8 KB	16 KB	—	32 KB	8 KB	20 KB	8 KB
Cache Data	16 KB	16 KB	8 KB	gemeinsam	—	—	8 KB	16 KB	gemeins.
max Durchsatz (MB/s)			3200		k.A.	1000	1200	960	1067
sek. Cache Instr.	1 MB	1 MB	32 MB	256 KB	1 MB	—	4 MB	2 MB	512 KB
sek. Cache Data	insges.	insges.	insges.	gemeinsam	2 MB	64 KB	insges.	gemeins.	gemeins.
max Durchsatz (MB/s)			1067	k.A.	1600	500	800	320	133
Bus-Taktrate (MHz)			67		67	62,5	50	40	33
Bus-Durchsatz (MB/s)	1200		1067		267	1000	400	320	133
SPECint92	140	125,8	69,6	64,5	80,0	59,2	61,7	53,2	32,2
SPECfp92	131	121,2	182,1	56,9	150,6	124,8	63,4	63,4	16,0
SPECint95		3,11			3,13			1,0	
SPECfp95		3,10			4,0			1,0	

Tabelle 1.18: CISC PC-CPU's: i80x86

	i486DX2	i486DX4	Pentium	Pentium	Pentium	Pentium MMX	PentiumPro
Taktrate (MHz)	66	100	100	133	200	200	200
Cache Instr.	8 KB	16 KB	8 KB	8 KB	8 KB	8 KB	8 KB
Cache Data	gemeins.	gemeins.	8 KB	8 KB	8 KB	8 KB	8 KB
sek. Cache	256 KB	256 KB	512 KB	512 KB	512 KB	512 KB	512 KB
SPECint92	32,2	51,4	100,0				
SPECfp92	16,0	26,6	80,6				
SPECint95			3,05	4,1	5,5	6,41	8,58
SPECfp95			2,07	2,5	3,92	4,66	6,48



## 64-Bit Workstation-CPUs:

Tabelle 1.19: 64 Bit RISC-CPUs (1995...)

	UltraSPARC II	Ultra-SPARC I	MIPS R10000	DEC-alpha 21164	HP PA8200	IBM PowerPC 604e
Taktrate (MHz)	336	200	250	533	240	332
Cache Instr.	16 KB	16 KB	32 KB	8 KB	2 MB	32 KB
Cache Data	16 KB	16 KB	32 KB	8 KB	2 MB	32 KB
sek. Cache	4 MB	1 MB	4 MB	96 KB + 4 MB	—	256 KB
SPECint95	14,9	7,81	14,7	16,9	17,4	14,4
SPECfp95	37,6	10,4	62,5	56,7	28,5	12,6

Tabelle 1.20: RISC PC-CPUs: PowerPC (Apple und IBM)

	MPC620	PowerPC „G3“	MPC604e	MPC604	MPC603e	MPC603	MPC601
Taktrate (MHz)	~133	250	332	166	200 ... 240	66/80	50/60/66/80
Cache Instr.	32 KB	32 KB	32 KB		16 KB	8 KB	32 KB
Cache Data	32 KB	32 KB	32 KB		16 KB	8 KB	gemeins.
sek. Cache	1 ... 128 MB	256 ... 1024 KB	256 KB		1 MB	1 MB	1 MB
SPECint92	~225					~60/75	~62 ... 85
SPECfp92	~300					~70/85	~72 ... 105
SPECint95	~5,6	~10	14,4	5,2	7,1		
SPECfp95	~5,6		12,6	4,3	4,2		

Der MPC620 ist ein 64 Bit-Prozessor, der MPC604e hat zwar einen 64 Bit-Datenbus, aber nur einen 32 Bit-Adreßbus.

## Zum Vergleich:

Tabelle 1.21: PC-Bussysteme und Durchsatzraten

	PCI 32	PCI 64	PCI 2.2	PCI-X	PCI-X 2.0 2.0	PCIe x1	PCIe x16
Datenbus- breite (bit)	32	64	64	64	64	seriell	seriell
Taktrate (MHz)	33	33	66	133	266		
Bus-Durchsatz (Mbit/s (peak))	133	266	533	1066	2133	2,5 Gbit/s	40 Gbit/s
Bus-Durchsatz (Mbit/s)	117	234				250 MByte/s je Richtung	4000 MByte/s je R.

	ISA	MCA 1.0	EISA	EISA EMB-133	VL-Bus	VL-Bus 64-bit	PCI 32	PCI 64
Datenbus- breite (bit)	8	32	32	64	32	64	32	64
Taktrate (MHz)	8	4	8	8	33	50	33	33
Bus-Durchsatz (Mbit/s)	8	16	33	125	75	160	117	234

Tabelle 1.22: CISC PC-CPUs: i80x86 (Forts.)

	Pentium II	„Deschutes“
	MMX, AGP, SDRAM	MMX, AGP
Taktrate (MHz)	333	300 ...
Cache Instr.	16 KB	
Cache Data	16 KB	
sek. Cache	512 KB	
SPECint95	12,8	~12
SPECfp95	9,25	~10

MMX („matrix manipulation extensions“) stellt eine Erweiterung des Befehlssatzes für Bild/Audio/Video zu Lasten der FloatingPoint-Befehle dar. Der PentiumPro (P6) ist intern ein RISC-Prozessor mit eingebautem „CISC-RISC translator“ (Kompatibilität!). Für den Pentium II wird ein schnelleres Motherboard mit SDRAM („synchr. dyn. RAM“) benötigt; die Graphikkarte wird aus Geschwindigkeitsgründen nicht mehr über den PCI-Bus sondern über den dedizierten AGP („accelerated graphics ports“) angeschlossen.

Aktuelle Daten entnehme man dem WWW:

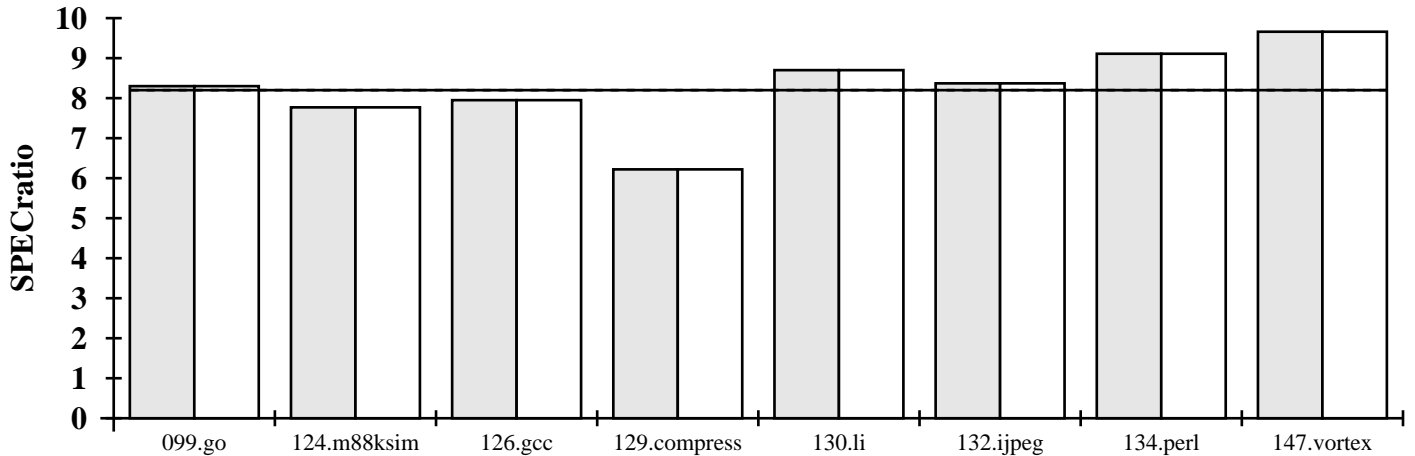
<http://www.specbench.org>, <http://www.ibm.com>,  
<http://www.intel.com>, ...

# SPEC CINT95 Results

©Copyright 1995, Standard Performance Evaluation Corporation

<b>Dell Computer Corporation</b> <b>Dell Dimension XPS Pro200n</b>	<b>SPECint95 = 8.20</b> <b>SPECint_base95 = 8.20</b>
---	---

SPEC license # 14	Tested By: Intel	Test Date: May-96	Hardware Avail: Jun-96	Software Avail: Mar-96
-------------------	------------------	-------------------	------------------------	------------------------



Hardware/Software Configuration for: Dell Dimension XPS Pro200n	Benchmark # and Name	Reference Time	Base Run Time	Base SPEC Ratio	Run Time	SPEC Ratio	
<b>Hardware</b> Model Name: XPS CPU: 200MHz Pentium Pro processor FPU: Integrated Number of CPU(s): 1 Primary Cache: 8KBI+8KBD Secondary Cache: 256KB(I+D) Other Cache: None Memory: 64MB(EDO) Disk Subsystem: 1MB IDE Quantum Fireball Other Hardware: Integrated EIDE disk controller  <b>Software</b> Operating System: UnixWare 2.0, UnixWare SDK Compiler: Intel C Reference Compiler V2.3 File System: UFS System State: Single User	099.go	4600	554	8.30	554	8.30	
	124.m88ksim	1900	244	7.77	244	7.77	
	126.gcc	1700	214	7.95	214	7.95	
	129.compress	1800	289	6.22	289	6.22	
	130.li	1900	218	8.70	218	8.70	
	132.jpeg	2400	287	8.37	287	8.37	
	134.perl	1900	209	9.11	209	9.11	
	147.vortex	2700	280	9.66	280	9.66	
	SPECint_base95 (G. Mean)				8.20		
				SPECint95 (G. Mean)		8.20	

### Notes/Tuning Information

Base and peak flags are the same and use Feedback Directed Optimization  
 Pass1: -tp p6 -ipo -prof\_gen -ircdb\_dir /proj/tmp/IRCDB  
 Pass2: -tp p6 -ipo -prof\_use -ircdb\_dir /proj/tmp/IRCDB  
 -ircdb\_dir is a location flag and not an optimization flag  
 Portability: 124: -DSYSV -DLEHOST 130, 134, 147: -lm 132: -DSYSV 126: -lm -lc -L/usr/ucblib -lucb -lmalloc

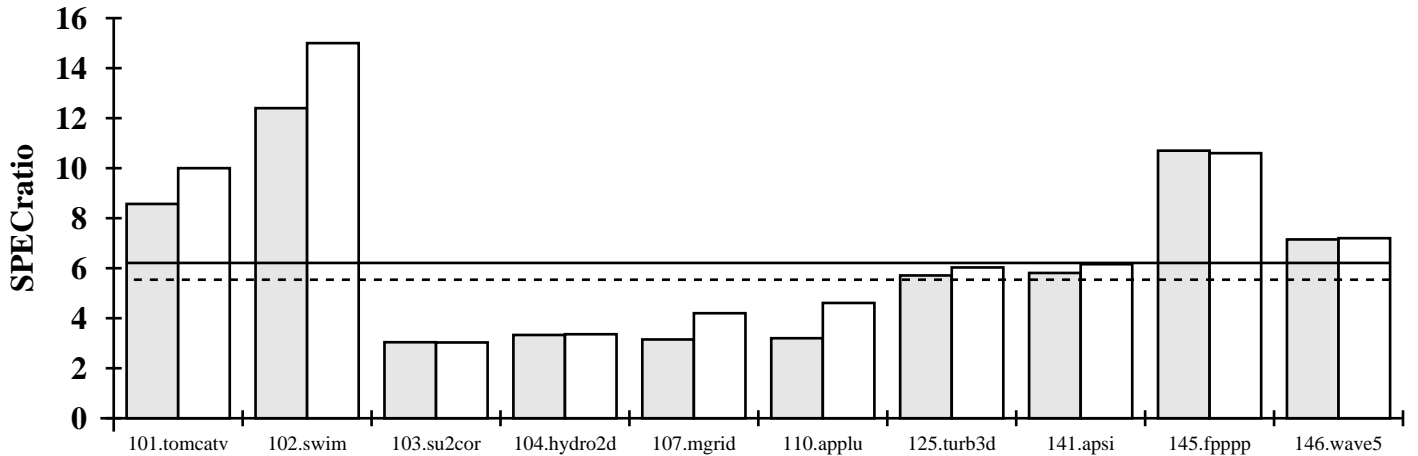
# SPEC CFP95 Results

©Copyright 1995, Standard Performance Evaluation Corporation

**Dell Computer Corporation**  
**Dell Dimension XPS Pro200n**

SPECfp95 = 6.21  
 SPECfp\_base95 = 5.54

SPEC license # 14 | Tested By: Intel | Test Date: May-96 | Hardware Avail: Jun-96 | Software Avail: Mar-96



Hardware/Software Configuration for: Dell Dimension XPS Pro200n		Benchmark # and Name	Reference Time	Base Run Time	Base SPEC Ratio	Run Time	SPEC Ratio
<b>Hardware</b>		101.tomcatv	3700	432	8.57	369	10.0
Model Name:	XPS	102.swim	8600	695	12.4	572	15.0
CPU:	200MHz Pentium Pro processor	103.su2cor	1400	461	3.04	462	3.03
FPU:	Integrated	104.hydro2d	2400	720	3.33	715	3.36
Number of CPU(s):	1	107.mgrid	2500	793	3.15	595	4.20
Primary Cache:	8KBI+8KBD	110.applu	2200	687	3.20	478	4.61
Secondary Cache:	256KB(I+D)	125.turb3d	4100	718	5.71	679	6.03
Other Cache:	None	141.apsi	2100	361	5.81	341	6.16
Memory:	64MB(EDO)	145.fpppp	9600	901	10.7	902	10.6
Disk Subsystem:	1MB IDE Quantum Fireball	146.wave5	3000	420	7.15	417	7.20
Other Hardware:	Integrated EIDE disk controller	SPECfp_base95 (G. Mean)		5.54			
<b>Software</b>				SPECfp95 (G. Mean)		6.21	
Operating System:	UnixWare 2.0, UnixWare SDK						
Compiler:	Intel C/FORTRAN Reference Compiler V2.3						
File System:	UFS						
System State:	Single User						

## Notes/Tuning Information

```

Flags: Base: -tp p6 -ipo -pc 64 -mem
Peak:
101: -tp p6 -ipo -mem -pad -distype -fcon -pc 64
102: -tp p6 -ipo -pad -fcon -pc 64
103: Pass1: -tp p6 -prof_gen -ircdb_dir /proj/tmp/IRCDB
Pass2: -tp p6 -ip -mem -mill -pad -fcon -pc 64 -prof_use -ircdb_dir /proj/tmp/IRCDB
104: Pass1: -tp p6 -prof_gen -ircdb_dir /proj/tmp/IRCDB
Pass2: -tp p6 -ip -mem -mill -pc 64 -prof_use -ircdb_dir /proj/tmp/IRCDB
107: -tp p6 -pad -fcon -pc 64
110: -tp p6 -ipo -mem -mP2OPT_opt_bblock_stat_limit=10000 -pc 64
125: -tp p6 -ip -mem -mill -pad -fcon -pc 64
141: Pass1: -tp p6 -w -prof_gen -ircdb_dir /proj/tmp/IRCDB
Pass2: -tp p6 -pc 64 -prof_use -ircdb_dir /proj/tmp/IRCDB
145: -tp p6 -ipo -pc 64
146: Pass1: -tp p6 -prof_gen -ircdb_dir /proj/tmp/IRCDB
Pass2: -tp p6 -ipo -mem -pad -distype -pc 64 -prof_use -ircdb_dir /proj/tmp/IRCDB
    
```

62

For More  
Information  
Contact:

SPEC  
10754 Ambassador Drive, Suite 201  
Manassas, VA 22110

(703) 331-0180  
info@specbench.org  
http://www.specbench.org

--

Prepared By: --

### 1.3.5 RISC-Marktkonsolidierung

Das Sparc-Lager (unter anderem):

- Sun Microsystems
- Amdahl
- Cray
- Fujitsu
- ICL
- Solbourne
- Toshiba
- Xerox
- Matsushita
- Tadpole Technology
- Goldstar
- Hyundai
- Tattung
- ...

Trotz der Ankündigung

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/46436>

wird die UltraSPARC-Linie mit den Rock-Chips weiterentwickelt, die als Neuerung CMT

([http://iforce.sun.com/protected/solaris10/adoptionkit/general/features/cmt\\_support.html](http://iforce.sun.com/protected/solaris10/adoptionkit/general/features/cmt_support.html))

bringen sollen.

Weitere Informationen zu SPARC: <http://en.wikipedia.org/wiki/SPARC>

## Das HP-PRO-Lager: HP PA (Power Architecture)

- HP
- Hitachi
- Mitsubishi
- Hughes
- Oki
- Yokogawa
- Prime
- Convex
- Sequoia

Abgekündigt:

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/20671>

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/59956>

Weitere Informationen: <http://en.wikipedia.org/wiki/PA-RISC>

## **Das Motorola-Lager (88-Open unter anderem): 88000**

- Motorola
- Data General
- Omron
- Encore Computer
- Dolphin Server Technology
- Sanyo/Icon International Inc.
- Cetic (Thomson CSF)
- Harris Computer Systems Division
- McDonnell Douglas Information Systems
- Micro Focus
- Philips Information Systems
- SAS Institute
- Tadpole Technology
- Unify
- Wordperfect
- Oracle
- Unix International

... leider in der Marktreife zu spät, so dass sowohl NeXT als auch Apple nicht mehr länger auf diese CPU nicht warten wollten und sie nie zum Massengeschäft wurde (Apple benutzte darauf hin den IBM PowerPC).

Weitere Informationen: <http://en.wikipedia.org/wiki/88000>



## Intel: (i860)

Der Intel **i860** war eine gemischte 32/64-Bit CPU der RISC-Architektur, deren FU-Pipelines durch Programmcode erreichbar waren. Die theoretischen CPU-Geschwindigkeiten konnten jedoch in Realität leider nicht erreicht werden, weshalb er lediglich als Graphik-Koprozessor erfolgreich wurde.

Weitere Informationen:

[http://de.wikipedia.org/wiki/Intel\\_i860](http://de.wikipedia.org/wiki/Intel_i860)

<http://encyclopedia.thefreedictionary.com/Intel%20i860>

Neuere Pentium-CPUs sind im Mikrocode RISC-basiert aufgebaut, im Maschinencode jedoch aus Abwärtskompatibilitätsgründen leider nicht. Wenn man jedoch nicht alle „alten“ Maschinenbefehle nutzt, kann eine heutige x86-CPU ähnlich schnell wie eine RISC-CPU arbeiten.

## **Das Digital-Lager: Alpha**

- DEC
- Kubota
- Olivetti
- Raytheon
- Encore Computer
- Advanced Computer Research International
- Carrera Computers
- Cray Research

Abgekündigt:

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/25265>

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/50170>

### Das Mips-Lager (unter anderem):

- Silicon Graphics (Mips)
- DEC
- Pyramid
- SNI
- Siemens
- Olivetti
- Tandem
- NEC
- Sony
- AT&T
- Sumitomo
- Control Data
- Microsoft
- LSI Logic
- Integrated Device Technology
- Toshiba
- Acer
- Carrera Computers

### Abgekündigt:

- <http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/2208>
- <http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/10835>
- <http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/59006>
- [http://en.wikipedia.org/wiki/MIPS\\_architecture#Applications](http://en.wikipedia.org/wiki/MIPS_architecture#Applications)

### Das Power-PC-Lager:

- IBM
- Apple
- ASI
- Bull HN Information Systems Inc.
- Canon
- Ford (für den Einsatz in Kraftfahrzeugen)
- Harris Corp.
- Motorola
- Peacock
- Tadpole Technology
- Taiwan New PC Consortium (24 Hersteller)
- Thomson CSF
- Vobis

### Aktuelle Versionen und schleichender Abschied vom PC-Markt(?):

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/57582>

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/59056>

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/60335>

### Zukunft des PowerPC(?):

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/60402>

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/60522>

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/60858>

<http://en.wikipedia.org/wiki/PowerPC>

**Das Power-Open-Lager:** (siehe auch Power-PC)

- IBM
- Apple
- Bull HN Information Systems Inc.
- Harris Corp.
- Motorola
- Tadpole Technology
- Thomson CSF

Nähere Informationen:

[http://en.wikipedia.org/wiki/IBM\\_POWER](http://en.wikipedia.org/wiki/IBM_POWER)

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/4708>

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/55504>

## 1.4 HighEnd-Server und Großrechner

HighEnd-Server und Großrechner stellen eine Menge von Eigenschaften bereit, die für ausfallsichere Systeme (etwa: EDV einer Telefongesellschaft, ...) nötig sind und zu Offline-Zeiten von unter 3 Sekunden pro Jahr führen:

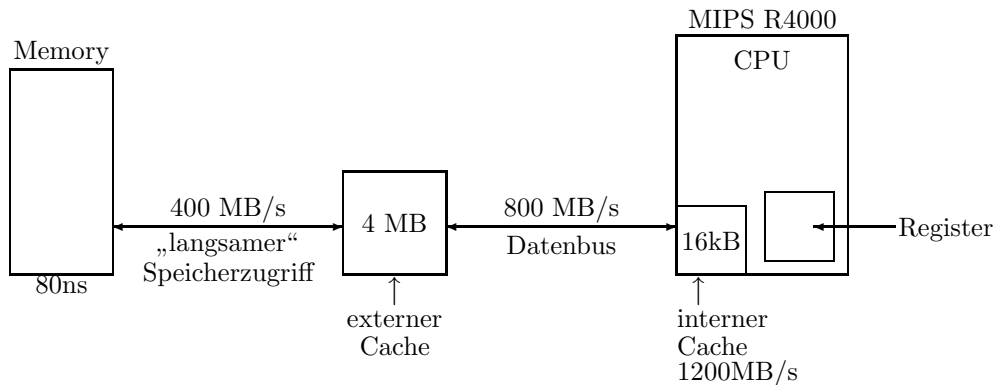
- redundante, im Betrieb komponentenweise ersetzbare Stromversorgung an unabhängigen Stromzulieferungsleitungen und -phasen, **USVs** zur Überbrückung von Stromschwankungen, Dieselaggregate mit Stromgeneratoren oder Ersatzstromversorgungen
- mehrere offline schaltbare CPU-Boards (mit jeweils mehreren Prozessoren)
- redundante „**hot swappable**“ Plattenarrays (Spiegelung oder Paritäts-Redundanz, **RAID**)
- viele **I/O-Channels** (E/A-Prozessoren)
- viele **LPARs**
- „**Capacity on Demand**“

Vertiefung:

<http://www.serverworldmagazine.com/compaqent/2000/06/primer.shtml>  
<http://www-1.ibm.com/servers/eserver/zseries/>

## 1.5 Der Cache zur Datentransferbeschleunigung

Abbildung 1.23: Der Cache als Daten-Vorratsbehälter



**Bemerkungen:** Beachte die verschiedenen Durchsatzraten in (umgekehrter) Proportionalität zur jeweiligen Cache-/Speichergröße!

Hoher Durchsatz wird erreicht bei:

- sequentieller Ausführung von Befehlen,
- Zugriff auf ganze Vektoren (Daten),
- Zugriff auf 64/128 Bit-Gleitkommazahlen bei 32 Bit Datenbusbreite,
- durchschnittlich geringem Bedarf an Daten (aus dem Speicher) pro Befehl

Geringer Durchsatz wird erzielt bei:

- wilden Sprüngen, ...
- Benutzung von verketteten Listen mit „großen“ Knoten (CAD-Programme), ...

**Bemerkung:** Es wurden SparcStation10's ohne SuperCache ausgeliefert, um bei speziellen Anwenderprogrammen das „verlangsamende“ Cache-Füllen zu umgehen.

Vergleiche: [http://en.wikipedia.org/wiki/CPU\\_cache](http://en.wikipedia.org/wiki/CPU_cache) und <http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?cache>

## 1.6 Multiuser-Betriebssysteme

**Multiuser-Betriebssysteme** ermöglichen das (sichere) Arbeiten mehrerer Benutzer auf einem Computer, indem Sie das Konzept des Benutzers (Users) von Ressourcen einführt und diesem die Zugriffskontroll-Steuerung über seine Ressourcen ermöglicht. Daneben wird mit Hilfe eines hierarchisch aufgebauten Ordner/Unterordner/Datei-Namenssystems die Möglichkeit der übersichtlichen Gestaltung der Dateien mehrerer Nutzer erleichtert.

## 1.7 Multitasking und Timesharing

**Cache misses** führen dann zu keinen unnötigen wait-Zyklen, wenn in einem Multitaskingsystem andere Aufträge (Prozesse) auf die CPU-Benutzung warten: Statt „idle“ zu warten braucht die CPU den auf Memory-Daten wartenden Prozess lediglich in die Warteschlange der „blocked“ Prozesse aufzunehmen und kann sich sofort einem der in der „Ready“-Warteschlange direkt produktiv fortsetzbaren Prozess zuwenden:

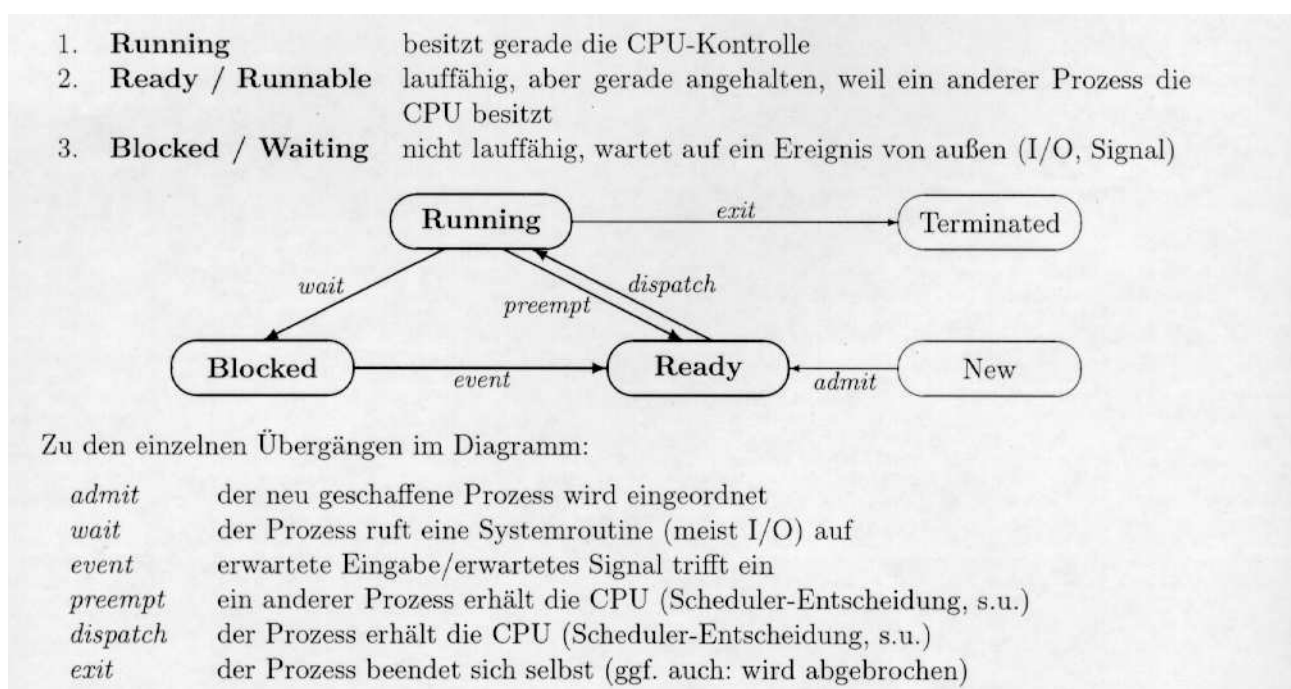


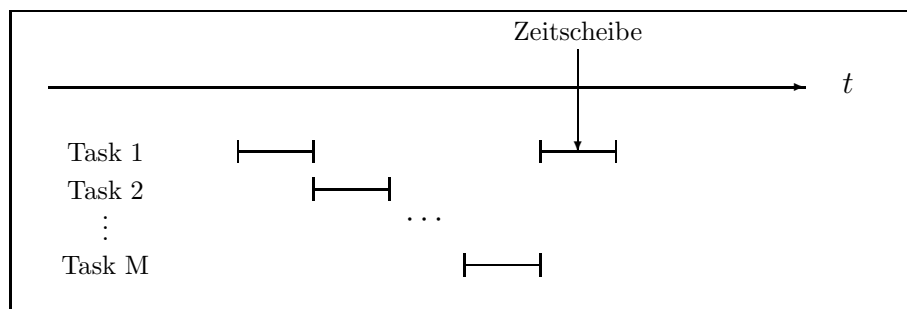
Abbildung 1.24: Multitask-Stati



Um dem interaktiv arbeitenden Computernutzer akzeptabel schnelle Reaktionen auf seine Tastaturbedienungen anbieten zu können, muß zusätzlich eine schnelle Umschaltung der CPU von Benutzerprozess zu Benutzerprozess realisiert werden:

### Prozesse (Tasks)

- Programmeinheiten, die unterbrech- und wiederfortsetzbar sind
- Multitasking:  
1 Rechner erscheint M Benutzern als M jeweils dedizierte einzelne Rechner. Ein Rechner kann (scheinbar) zugleich mehrere Prozesse bearbeiten.



Preemptiv (= „wahlweise vergebbar“) heißt ein Multitasking-Betriebssystem dann, wenn der Betriebssystemkern allein für die Vergabe der Zeitscheiben verantwortlich ist, also nicht auf die kooperative Mitarbeit der Tasks angewiesen ist. Die Tasks werden von „außen“ (spätestens nach Ablauf einer Zeitscheibe) unterbrochen und später wieder fortgesetzt.

## 1.8 Multiuser-Multitasking-Betriebssysteme

... verbinden die Konzepte der Multiuser-Systeme mit denen der „schnellen“ Interaktionsmöglichkeiten einzelner User mittels Timesharing.

Beispiele:

*Windows 3.x*: Multitasking, nicht preemptiv.

*OS/2, Windows 95, Windows NT, UNIX*: preemptives Multitasking.

Bemerkungen zu (Multiuser-) Multitasking-Systemen:

- Benutzer/Benutzer-Gruppen mit Account-Name (**User**), **UserID** und **GroupID** als Eigentümer von Ressourcen
- spezielle User für gewisse Funktionalitäten: **root/Administrator**, ..., **nobody** (als Nutzer mit geringsten Rechten)
- **Passwörter/PINs** für die **Authentifizierung**
- **Passwort-Alternativen**, **Magnetkarten**, **Smartcards**, **RFID-Tags**, ...
- **Autorisierung**
- **Access-Control-Lists**
- **rollenbasierte Zugriffskontrolle**: postmaster, webmaster, uucp, ftpadm, abuse, chief, manager, ...
- „discretionary“ vs. „mandatory“ access control, siehe auch [http://en.wikipedia.org/wiki/Security\\_focused\\_operating\\_system](http://en.wikipedia.org/wiki/Security_focused_operating_system)
- hierarchisch aufgebaute **Ordner-/Datei**struktur zur Übersichtlichkeit

Neben **Multitasking**-Systemen, die den Nutzern per **Timesharing** viele (virtuelle) Computer zur Verfügung stellen, stehen heute auch immer mehr die Möglichkeiten im Vordergrund, einzelnen Applikationen die Fähigkeit zu geben, mehrere Prozessoren für ihre Dienstleistungen „parallel“ zu benutzen, das sogenannte *Multithreading*.

## 1.9 Multithreading

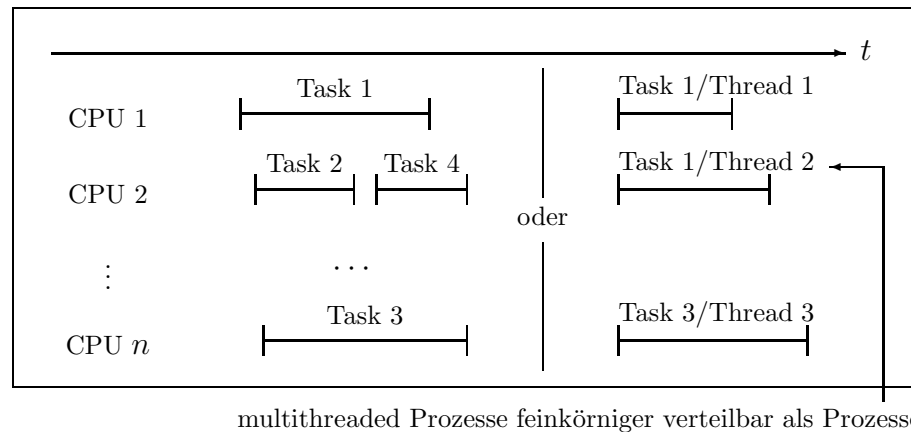
Die einzelnen laufenden Prozesse in Multitasking-Systemen können nur begrenzt an gemeinsamen Aufgaben zusammenarbeiten, da sie lediglich mittels **IPC** beziehungsweise **Netzwerkkommunikation** miteinander kommunizieren können, was doch das Hauptanliegen des Multitasking, einzelne Prozesse andere Prozesse so wenig wie möglich stören zu lassen.

Prozesse, die miteinander IPC betreiben wollen, müssen sich i.a. kostspielig klonen (d.h. Kopien aller wichtigen Systemressourcen erhalten, so dass nachher jeder Prozess lediglich seine eigene Kopie der Ressourcen ändern kann). Dieses kostspielige Klonen wird beim Multithreading vermieden: die einzelnen **Threads** (= unabhängig voneinander auch gleichzeitig an einer gemeinsamen Aufgabe arbeiten könnende Algorithmenteile) behalten den gemeinsamen Speicher, die gemeinsame Datei-Tabelle, ...

Erkauft wird dieses kostengünstige miteinander an einer gemeinsamen Aufgabe arbeiten Können durch einen zusätzlichen Aufwand: Die Applikationen müssen multithreading-fähig programmiert werden und dabei insbesondere darauf achten, dass die einzelnen Threads richtig synchronisiert werden, also z.B. nicht gleichzeitig dieselbe Ressource zu ändern versuchen (**Resourcen-Locking**), dass nicht zwei Threads jeweils auf die Vollendung des anderen warten (**deadlock**), ...

Dann bieten multithreading-fähige Applikationen aber die beiden folgenden Vorteile:

- Kürzere Turn-Around-Zeiten, da mehrere Prozessoren an der Aufgabe gleichzeitig arbeiten können.



- Selbst beim Vorhandensein lediglich eines Prozessors kann beim nötigen Warten eines Threads (der z.B. auf das Einlesen eines Plattensektors wartet) ein anderer Thread der Applikation (z.B. das GUI) weiter laufen.

## 1.10 Hyperthreading

Um die in neueren Pentium-CPU's mehrfach vorhandenen Funktionseinheiten durchschnittlich besser auszulasten, simulieren HT-CPU's (hyper-threading) dem Betriebssystem gegenüber die Existenz von 2 (virtuellen) Prozessoren. Ist dann in einem Prozess-Thread wegen Cache-Miss, Pipeline-Stalling, ... eine Funktionseinheit zeitweise arbeitslos, kann sie für einen anderen Prozess-Thread genutzt werden. Ziel des HT ist also primär eine Verbesserung der Nutzung aller Ressourcen der CPU, nicht unbedingt eine schnellere Ausführung von einzelnen Applikationen.

Durchschnittliche Performance-Steigerung (Turn-Around-Zeiten von multi-threading programmierten Applikationen) ist 20..30 Prozent: In diesem günstigen Falle sind also nicht 2 virtuelle CPUs, sonder etwa 1,3 virtuelle CPUs verfügbar. Bei nicht multi-threading programmierten Applikationen wirkt sich die Verfügbarkeit einer HT-CPU nicht Turn-Around-Zeiten vermindern aus, lediglich der gesamte Prozess-Durchsatz der CPU wird um 20..30 Prozent vergrößert.

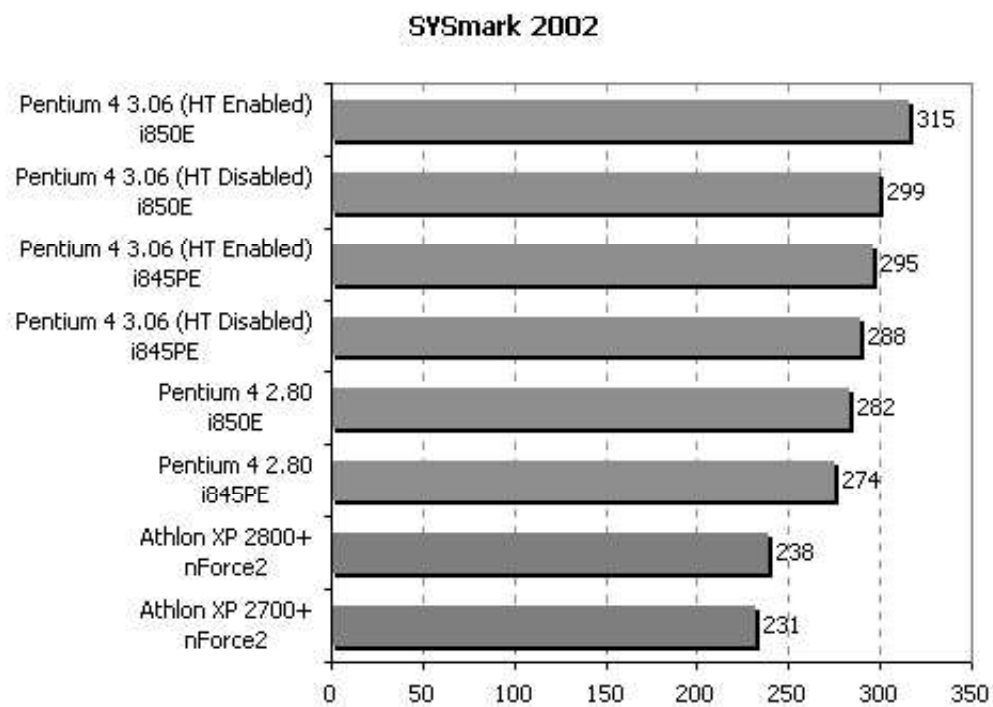


Abbildung 1.25: Hyperthreading

Vergleiche <http://en.wikipedia.org/wiki/Hyperthreading> und [http://www.xbitlabs.com/articles/cpu/display/pentium4-3066\\_2.html](http://www.xbitlabs.com/articles/cpu/display/pentium4-3066_2.html).

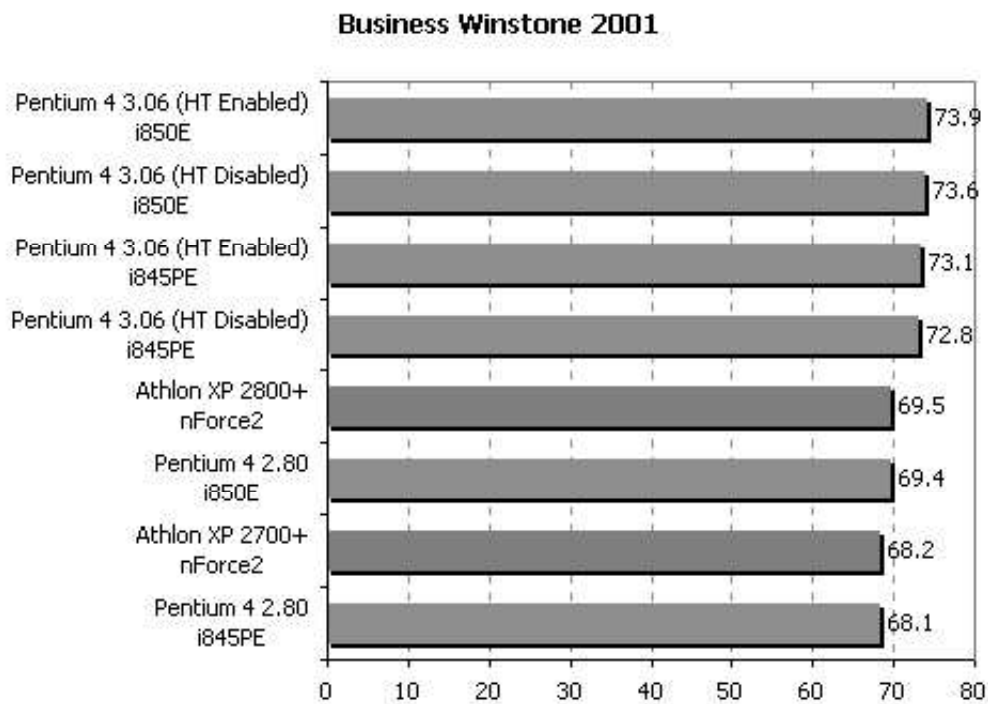


Abbildung 1.26: Hyperthreading (Fortsetzung)

Meßergebnisse:

-----  
 2 HT 1,8 GHz Xeon-Prozessoren  
 -----

The number of threads is 1  
 Creating thread number 0, tid=1026

real 0m1.179s  
 user 0m1.170s  
 sys 0m0.010s  
 -----

The number of threads is 2  
 Creating thread number 0, tid=1026  
 Creating thread number 1, tid=2051

real 0m0.868s  
 user 0m1.720s  
 sys 0m0.010s  
 -----

```
The number of threads is 3
Creating thread number 0, tid=1026
Creating thread number 1, tid=2051
Creating thread number 2, tid=3076
```

```
real    0m0.585s
user    0m1.520s
sys     0m0.000s
```

---

```
The number of threads is 4
Creating thread number 0, tid=1026
Creating thread number 1, tid=2051
Creating thread number 2, tid=3076
Creating thread number 3, tid=4101
```

```
real    0m0.438s
user    0m1.730s
sys     0m0.000s
```

---

...

---

```
The number of threads is 8
Creating thread number 0, tid=1026
Creating thread number 1, tid=2051
Creating thread number 2, tid=3076
Creating thread number 3, tid=4101
Creating thread number 4, tid=5126
Creating thread number 5, tid=6151
Creating thread number 6, tid=7176
Creating thread number 7, tid=8201
```

```
real    0m0.487s
user    0m1.670s
sys     0m0.000s
```

---

## 1.11 Dual-core CPUs

Dual-core CPUs bringen auf einem Chip zwei vollständige übliche CPUs bis hin zum L1-Cache unter.

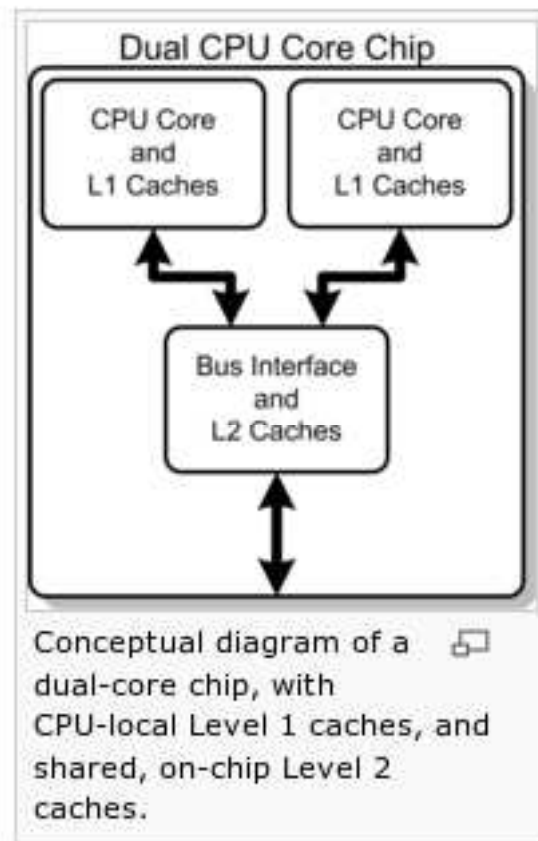


Abbildung 1.27: Dual-core CPU

Vergleiche: <http://en.wikipedia.org/wiki/Dual-core>

Wenn man auf einem Chip zwei halb so schnell getaktete Cores als auf einem Referenzchip unterbringt, so erreicht man die gleiche Anzahl von Operationen pro Sekunde bei lediglich einem viertel so viel Stromverbrauch (und auch Kühlungsbedarf):

[http://www.hpcaconf.org/hpca11/slides/hpca\\_cmt\\_slides.pdf](http://www.hpcaconf.org/hpca11/slides/hpca_cmt_slides.pdf)

[http://enterprise.amd.com/downloadables/Server\\_Blade.ppt](http://enterprise.amd.com/downloadables/Server_Blade.ppt)

Die CPU-Rechenleistung per Watt wird immer wichtiger:

<http://www.xbitlabs.com/news/cpu/display/20050512111032.html>

[http://www.computerbase.de/artikel/hardware/prozessoren/2005/test\\_intels\\_pentium\\_4\\_600-serie/29/](http://www.computerbase.de/artikel/hardware/prozessoren/2005/test_intels_pentium_4_600-serie/29/)

<http://www.cotsjournalonline.com/home/printthis.php?id=100237>

<http://www.heise.de/ct/aktuell/meldung/60324>

<http://www.heise.de/newsticker/meldung/61503>



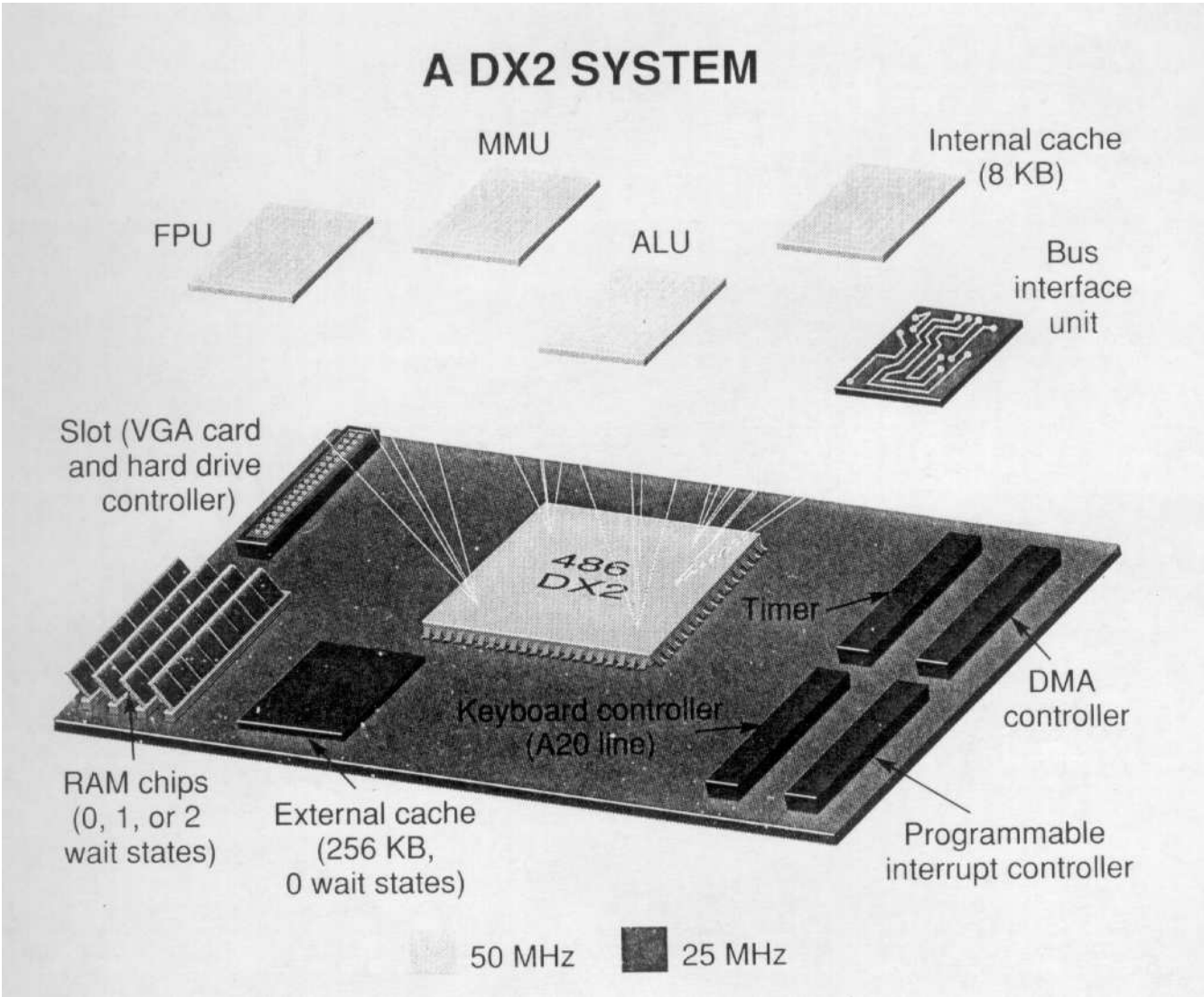


Abbildung 1.28: Verschiedene Geschwindigkeiten im Computer

## 1.12 Bussysteme — Chipsätze und das Motherboard

Die wichtigsten die Bus- und Peripherieinterfaces bedienenden Hardware-schaltungen befinden sich bei Motherboards für Intel-kompatible Rechner im sogenannten Chipsatz:

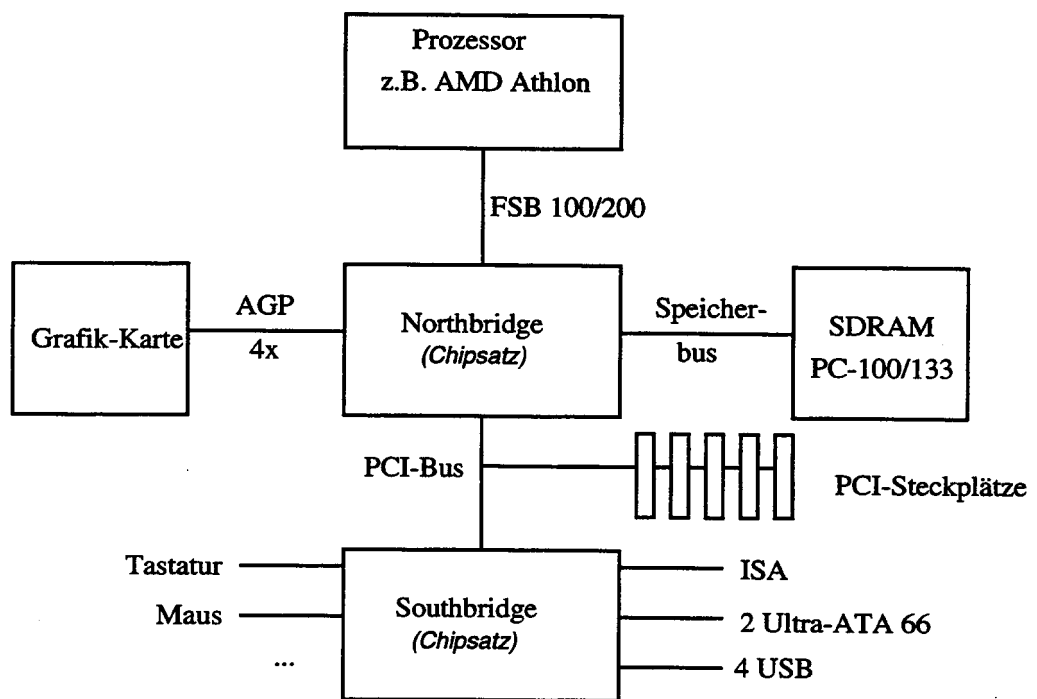


Abbildung 1.29: PC Frontsidebus

Zu genaueren aktuellen Informationen vergleiche im Web:

- [Hinweise zur Bus-Architektur von Intel-PCs](#)
- [Der „Front side bus“](#)
- [Der HyperTransport-Bus](#)
- [PCI und PCI express](#)
- [ATA \(IDE\) und sATA](#)

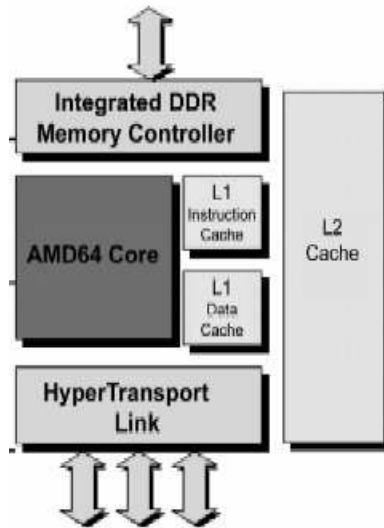


Abbildung 1.30: Alternative zum FSB: Der Opteron Hypertransport-Link

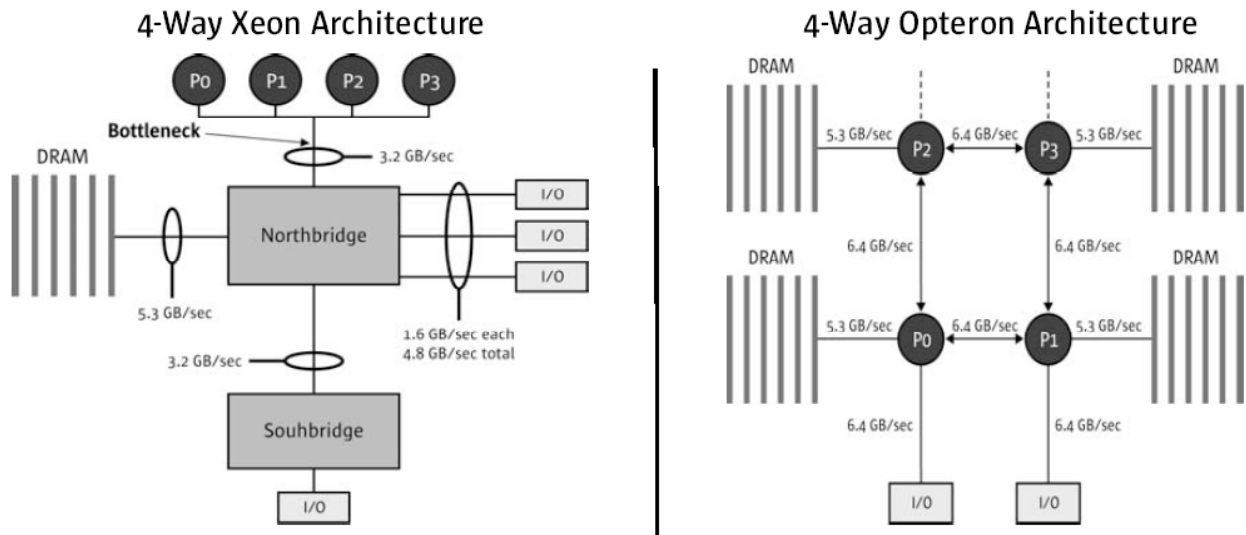


Abbildung 1.31: Hypertransport-Link vs. FSB

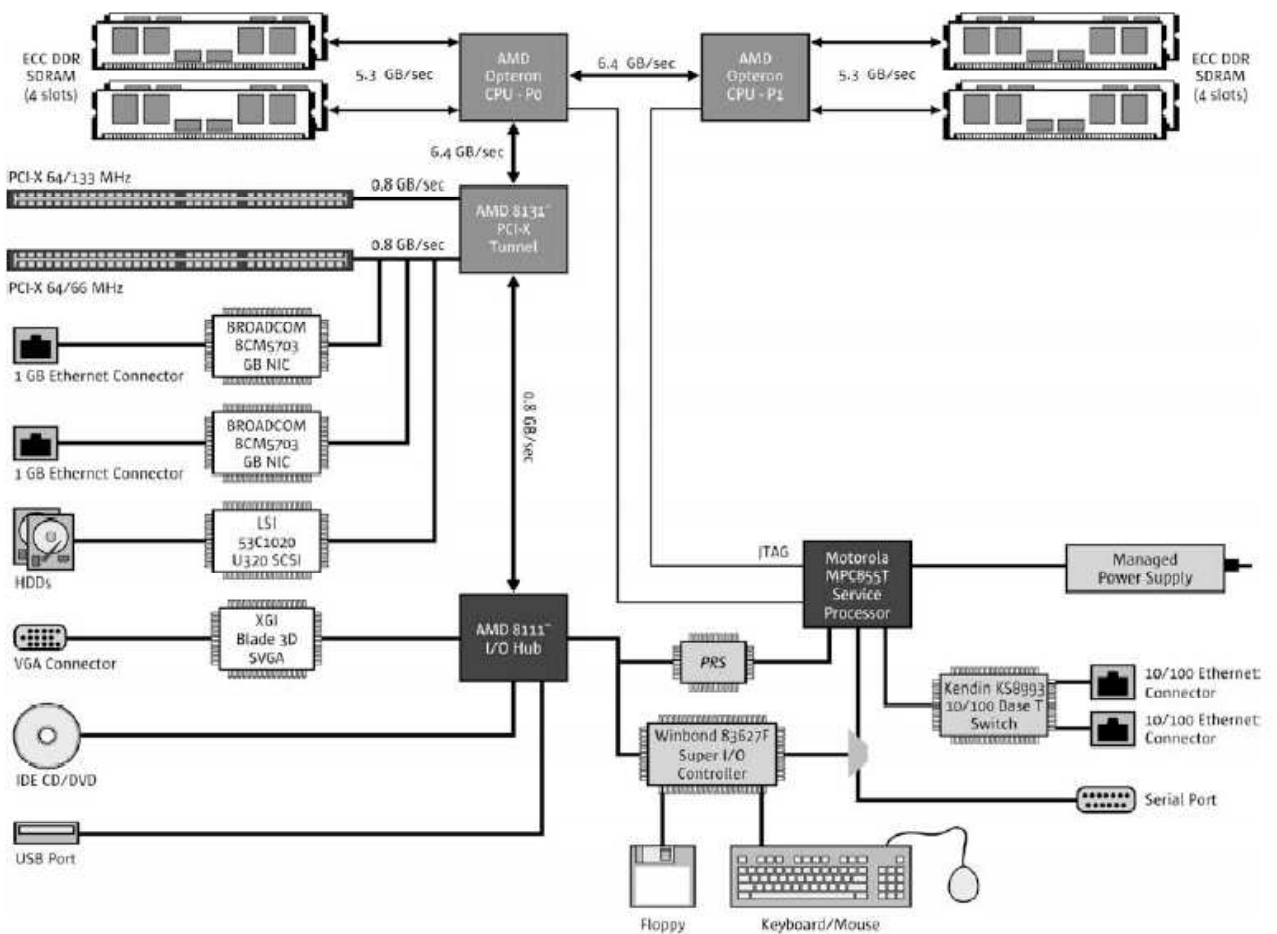


Abbildung 1.32: Hypertransport-Link (Forts.)

# Intel 955X and 945G/P Express Chipset

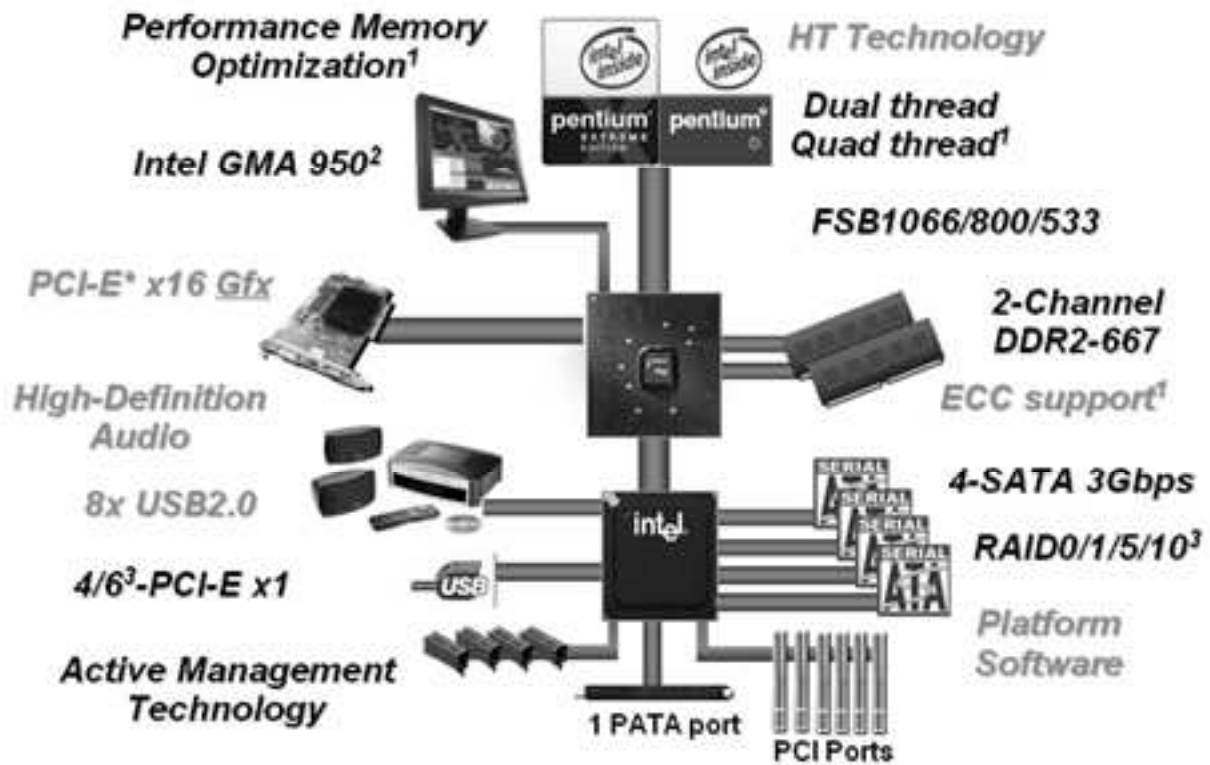


Abbildung 1.33: Graphikkarte am PCIe

Peripherie-Transfargeschwindigkeiten im Vergleich:

<http://www.macinfo.de/hardware/transferspeed.html>

Graphikkarten-Bus:

AGP, AGP-8X

PCIe-Graphikkartenbus

Graphikkarten

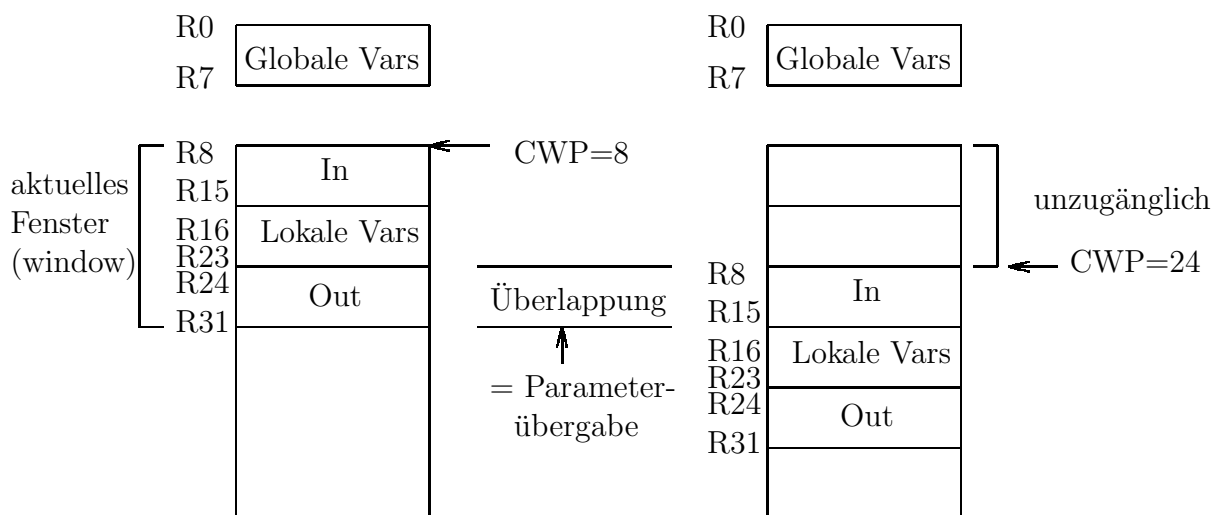
## 1.13 Der Weg fort von der von-Neumann-Architektur

### 1.13.1 SPARC

#### SUPER-SPARC

Die Register des SPARC V8-Prozessors (SUPER-SPARC) sind durch verschiedene Merkmale gekennzeichnet:

##### 1. Registerstack



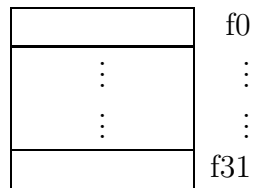
CWP steht für *current window pointer*. Die SPARCclassic besitzt z. B. acht solcher Fensterbereiche. Auf die Register greift man mit *logischen* Namen zu:

```
%g0..%g7  global vars
%i0..%i7  in
%l0..%l7  local vars
%o0..%o7  out
```

mit  $\%g0 \equiv 0$  (fest),  $\%o6 = \%sp$  (stack pointer),  $\%o7$  *return adress* für Unterprogramm-Rücksprung.

##### 2. Y für Multiplikation und Division.

3. Gleitkommaregister:



Je 32Bit-Gleitkommaregister für 16 *double precision* oder 8 *quad precision* Gleitkommazahlen.

4. State-Register sind unter anderem PSR *processor state register*, FSR *floating point state register*, PC *program counter*, nPC *next program counter*, WIM *window invalid maske*, TBR *trap base register*, IU-deferred trap queue, FP-deferred trap queue, ...

## Befehle und Ausführungszeiten bei SPARC(RISC)-ALU's

LDSB	load signed byte sign extended
LDSH	load signed halfword sign extended
LDUB	load unsigned byte zero filled
LDUH	load unsigned halfword zero filled
LD	load word
LDD	load doubleword
LDF	load floating-point
LDDF	load double floating-point
LDFSR	load floating-point state register
STB	store byte
⋮	
LDSTUB	atomic load/store unsigned byte
SWAP	swap register with memory
SETHI	set 22 bit of register
NOP	no operation
AND	
ANDcc	... and modify icc (= integer condition codes) of PSR
OR	
ORN	A or not B
XOR	
XNOR	exclusive NOR = XOR not

SLL	shift left logical
SRL	shift right logical (fill zero)
SRA	shift right arithmetic (fill most significant bit)
ADD	A+B
ADDcc	A+B and modify icc
ADDX	add with carry = A + B + c
:	
SUB	subtract
:	
MULSss	multiply step (zusammen mit Register Y für 64Bit-Ergebnis)
UMUL	unsigned integer multiply
SMUL	signed integer multiply
UDIV	unsigned integer division
SDIV	signed integer division
SAVE	register stack
RESTORE	register stack
Bicc	branch on integer condition codes, z.B.
	BA        branch always
	BN        branch never
	BNE       branch not equal
FBfcc	branch on floating point condition codes
CALL	call and link
JMPL	jump and link
RETT	return from Trap
Ticc	Trap on icc, z.B. TA, TN, ...
:	(read/write state ... registers)
UNIMP	unimplemented
FLUSH	flush instruction memory
FPop	(rounding direction, wahlweise <i>to nearest, towards 0, towards <math>-\infty</math>, towards <math>+\infty</math></i> )
	FiTOs     integer to single
	FiTOd     integer to double
	FiTOq     integer to quad
	FsTOi
	FdTOi



FqTOi  
 FsTOd  
 FsToq  
 ⋮  
 FMOV<sub>s</sub>  
 FNEG<sub>s</sub>  
 FABS<sub>s</sub>  
 FSQRT<sub>s</sub>  
 FADD<sub>s</sub>  
 FADD<sub>d</sub>  
 FADD<sub>q</sub>  
 FSUB<sub>s</sub>  
 FMUL<sub>s</sub>  
 FsMUL<sub>d</sub> entspricht  $s \cdot s \rightarrow d$   
 FdMUL<sub>q</sub> entspricht  $d \cdot d \rightarrow q$   
 FDIV<sub>s</sub>  
 FCMP<sub>s</sub>  
 FCMPE<sub>s</sub> compare and exception it unordered  
 ⋮

In späteren (Ultra-)SPARC-Versionen:

Branch with prediction: BPr, z.B.: BNZ, bt (bt=branch taken)

Prefetch Data: Load to cache (ohne Erfolgsgarantie)

Compare and swap (atomic): zur Prozeßsynchronisation

VIS (Visual instruction set): spezielle „Vektorbefehle“ zur Grafikverarbeitung

## Ein Beispiel-C-Programm ...

```
#include      <stdio.h>
#include      <math.h>

double f(double x);

int main(){

    int i;

    printf(" x      |   f(x)\n");
    printf("-----\n\n");

    for (i = 0; i <=20; i++)
        printf("%f   %f \n", i/10.0, f(i/10.0));
}

double f(double x){

    double arg;

    arg= 2 * M_PI * x + 5.0;
    return sin(arg);

}
```

... und was daraus in SPARC-Maschinenbefehlen wird (cc -S):

```
    .section      ".text",#alloc,#execinstr
    .align      8
    .skip      16

    ! block 0

    .global main
    .type      main,#function
main:
    save      %sp,-120,%sp
```

```

        ! block 1
.L147:

! File sinus-tabelle.c:
!   1 #include      <stdio.h>
!   2 #include      <math.h>
!   3
!   4 double f(double x);
!   5
!   6 int main(){
!   7
!   8     int i;
!   9
!  10     printf(" x          |   f(x)\n");

        sethi    %hi(.L148),%l0
        or      %l0,%lo(.L148),%l0
        call    printf
        mov     %l0,%o0

!  11     printf(" ----- \n\n");

        call    printf
        add     %l0,20,%o0

!  13     for (i = 0; i <=20; i++)

        mov     %g0,%i5

! block 2
.L_y0:
        sethi    %hi(.L_cseg0),%l2
        or      %l2,%lo(.L_cseg0),%l2
        sethi    %hi(.L154),%l3
        or      %l3,%lo(.L154),%l3

.L153:
.L150:

!  14     printf("%f   %f \n", i/10.0, f(i/10.0));

        st      %i5,[%fp-12]

```

```

    ld      [%fp-12],%f4
    fitod  %f4,%f6
    ldd    [%12+0],%f4
    fdivd  %f6,%f4,%f30
    std    %f30,[%fp-24]
    st     %i5,[%fp-12]
    ld     [%fp-12],%f4
    fitod  %f4,%f6
    ldd    [%12+0],%f4
    fdivd  %f6,%f4,%f4
    std    %f4,[%sp+88]
    call   f
    ldd    [%sp+88],%o0
    mov    %i3,%o0
    ldd    [%fp-24],%f30
    std    %f30,[%sp+88]
    ld     [%sp+88],%o1
    ld     [%sp+92],%o2
    std    %f0,[%sp+88]
    ld     [%sp+88],%o3
    call   printf
    ld     [%sp+92],%o4
    add    %i5,1,%i5
    cmp    %i5,20
    ble    .L150
    nop

! block 3
.L155:
.L152:
.L146:
    jmp    %i7+8
    restore
    .size  main,(.-main)
    .align 8
    .align 8
    .skip 16

! block 0

.global f

```

```

        .type    f,#function
f:
        save    %sp,-120,%sp
        st      %i1,[%fp+72]
        st      %i0,[%fp+68]

        ! block 1
.L158:
        ld      [%fp+68],%f4
        ld      [%fp+72],%f5
        std     %f4,[%fp-16]

! File sinus-tabelle.c:
!   15   }
!   16
!   17   double f(double x){
!   18
!   19       double arg;
!   20
!   21       arg = 2 * M_PI * x + 5.0;

        sethi   %hi(.L_cseg1),%l0
        ldd     [%l0+%lo(.L_cseg1)],%f6
        sethi   %hi(.L_cseg2),%l0
        ldd     [%l0+%lo(.L_cseg2)],%f4
        fmuld   %f6,%f4,%f6
        ldd     [%fp-16],%f4
        fmuld   %f6,%f4,%f6
        sethi   %hi(.L_cseg3),%l0
        ldd     [%l0+%lo(.L_cseg3)],%f4
        faddd   %f6,%f4,%f4
        std     %f4,[%fp-24]

!   22       return sin(arg);

        std     %f4,[%sp+88]
        call    sin
        ldd     [%sp+88],%o0
        fmovs   %f1,%f5
        fmovs   %f0,%f4
        std     %f4,[%fp-8]

```

```

        fmovs    %f5, %f1
        fmovs    %f4,%f0
        jmp     %i7+8
        restore

        ! block 2
.L157:
        fmovs    %f5, %f1
        fmovs    %f4,%f0
        jmp     %i7+8
        restore
        .size    f,(.-f)
        .align   8

        .section          ".data1",#alloc,#write
        .align   4
.L148:
        .ascii   " x          |   f(x)\n\000"
        .skip    1
        .type    .L148,#object
        .size    .L148,20
        .align   4
.L149:
        .ascii   " ----- \n\n\000"
        .skip    1
        .type    .L149,#object
        .size    .L149,24
        .align   4
.L154:
        .ascii   "%f    %f \n\000"
        .type    .L154,#object
        .size    .L154,10

        .section          ".rodata",#alloc
        .align   8
.L_cseg0:
        .word    0x40240000,0x0
        .type    .L_cseg0,#object
        .size    .L_cseg0,8
        .align   8
.L_cseg1:

```

```

        .word    0x40000000,0x0
        .type    .L_cseg1,#object
        .size    .L_cseg1,8
        .align   8
.L_cseg2:
        .word    0x400921fb,0x54442d18
        .type    .L_cseg2,#object
        .size    .L_cseg2,8
        .align   8
.L_cseg3:
        .word    0x40140000,0x0
        .type    .L_cseg3,#object
        .size    .L_cseg3,8

        .section        ".bss",#alloc,#write
Bbss.bss:
        .skip    0
        .type    Bbss.bss,#object
        .size    Bbss.bss,0

        .section        ".data",#alloc,#write
Ddata.data:
        .skip    0
        .type    Ddata.data,#object
        .size    Ddata.data,0

        .section        ".rodata",#alloc
Drodata.rodata:
        .skip    0
        .type    Drodata.rodata,#object
        .size    Drodata.rodata,0

        .file    "sinus-tabelle.c"
        .ident   "@(#)stdio.h      1.15      99/10/02 SMI"
        .ident   "@(#)feature_tests.h  1.18      99/07/26 SMI"
        .ident   "@(#)isa_defs.h  1.20      99/05/04 SMI"
        .ident   "@(#)va_list.h   1.13      01/02/08 SMI"
        .ident   "@(#)stdio_tag.h   1.3       98/04/20 SMI"
        .ident   "@(#)stdio_impl.h  1.13      01/11/16 SMI"
        .ident   "@(#)math.h       2.11      00/09/07 SMI"
        .ident   "@(#)math_iso.h  1.2       00/09/07 SMI"

```

```

.ident "@(#)floatingpoint.h 2.5 99/06/22 SMI"
.ident "@(#)ieee.h 2.8 99/10/29"
.ident "acomp: Sun C 5.6 2004/07/15"

.global __fsr_init_value

__fsr_init_value = 0x0
.xstabs ".stab.index", "V=10.0;DBG_GEN=4.14.28;cd;backend;Xs;R="
.xstabs ".stab.index", "/tmp; /opt/SunStudio9/SUNWspro/prod/bin
.xstabs ".stab.index", "main", 42,0,0,0

```

Anweisung	Fujitsu/LSI #1,#2	Cypress/ Fujitsu #3	MicroSPARC (&B5000)	Fujitsu SPARClite	Super SPARC
# of register windows	7	8	7	8	8
ld (32 bit integer)	2	2	1	1	1
ldd (64 bit integer)	3	3	2	2	1
ld (32 bit float)	2	2	1	1	1
ldd (64 bit double)	3	3	2(1)	2	1
st (32 bit integer)	3	3	2	1	1
std (64 bit integer)	4	4	3	2	1
st (32 bit float)	3	3	2(1)	1	1
std (64 bit double)	4	4	3(2)	2	1
taken branches	1	1	1	1	1
untaken branches	2	1	1	1	1
jmp and rett	2	2	2	2	1
integer multiply	N/A	N/A	19	?	4
integer divide	N/A	N/A	39	?	18
issue FP operation	2	1	1	N/A	1

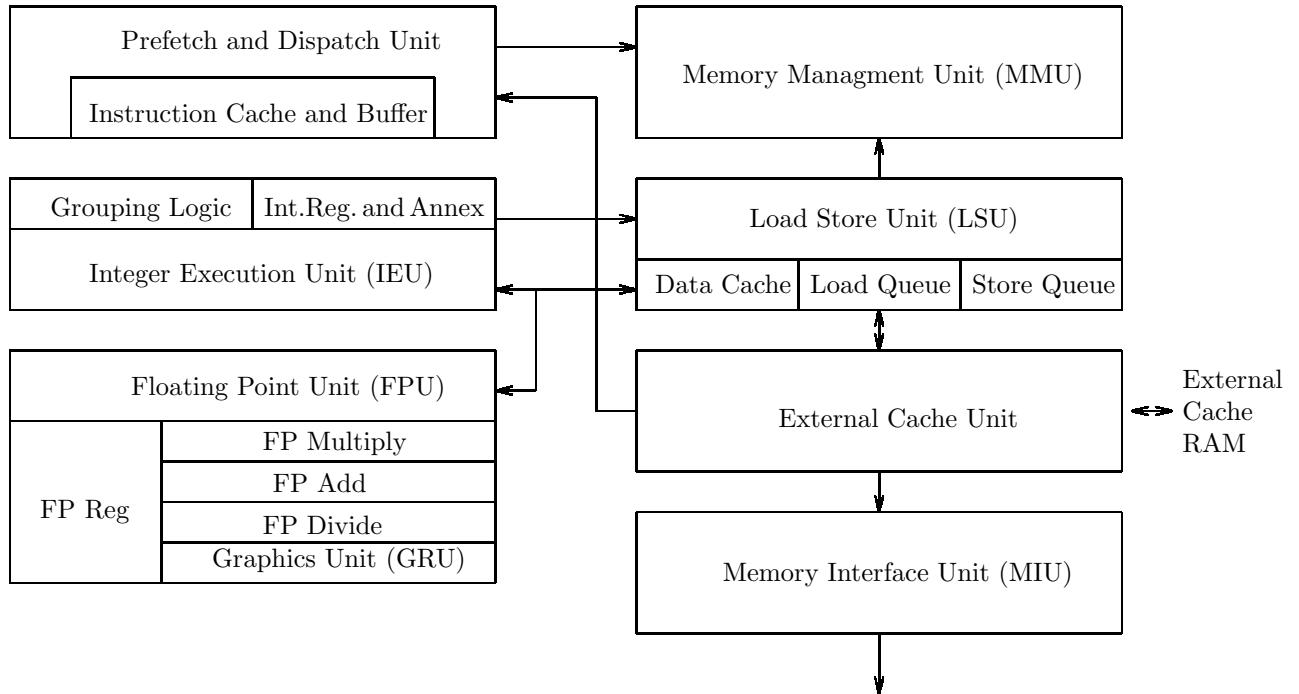
Tabelle 1.23: Anzahl der Register-Fenster und Integer-Zyklen pro Anweisung



Anweisung	FPC &	Weitek	Texas	BIT	MicroSPARC			Super
	TI 8847	3170 & 3171	602	B5000	min	typ	max	SPARC
fitos	8	10	4	2	5	6	13	1
fitod	8	5	4	2	4	6	13	1
fstoir, fstoi	8	5	4	2	6	6	13	1
fdtoir, fdtoi	8	5	4	2	7	7	14	1
fstod	8	5	4	2	2	2	14	1
fdtos	8	5	4	2	3	3	16	1
fmovs	8	3	4	2	2	2	2	1
fnegs	8	3	4	2	2	2	2	1
fabss	8	3	4	2	2	2	2	1
fsqrts	15	60	22	24	6	37	51	6
fsqrtd	22	118	32	45	6	65	80	10
fadds, fsubs	8	5	4	2	4	4	17	1
faddd, fsubd	8	5	4	2	4	4	17	1
fmuls	8	5	4	3	5	5	25	1
fmuld	9	8	6	4	7	9	32	1
fdvis	13	38	16	14	6	20	38	4
fdivd	18	66	26	24	6	35	56	7
fcmps, fcmpes	8	3	4	2	4	4	15	1
fcmpd, fcmped	8	3	4	2	4	4	15	1

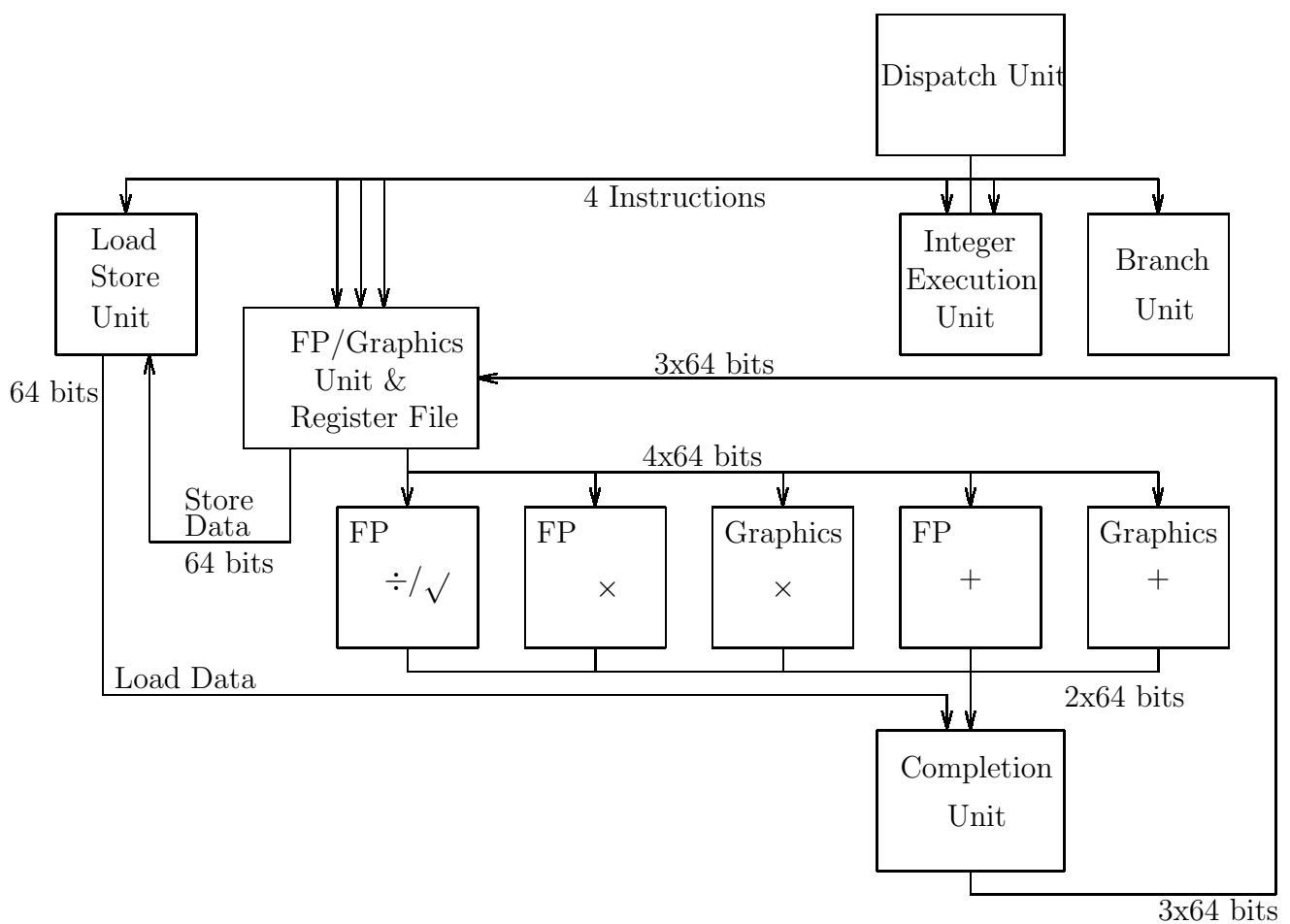
Tabelle 1.24: „floating point“-Zyklen pro Anweisung

## UltraSPARCI



- 64Bit-Architektur (SPARC V9)
- CPU-Befehle für(VIS): 2D-, 3D-Graphik, Real-Time Video
- IU-Pipeline (9 Phasen): fetch-decode-group-execute-cache access-load miss-integer pipe wait- trap resolution-writeback-
- 2 ALU's in IEU (= interger execute unit)
- Branch prediction
- Branch following
- FP-pipeline
- getrennte FP-execution units

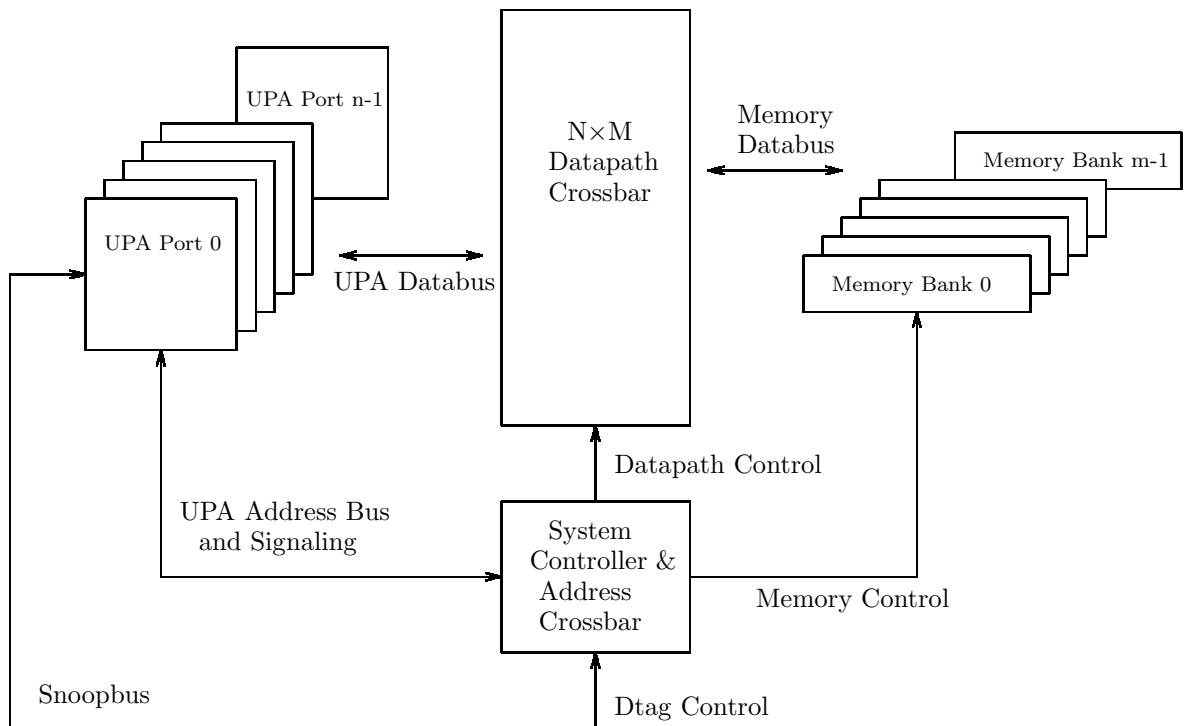
Operation	Throughput (Cycles)	Latency (Cycles)
Add (Single Precision)	1	3
Add (Double Precision)	1	3
Multiply (Single Precision)	1	3
Multiply (Double Precision)	1	3
Divide (Single Precision)	12	12
Divide (Double Precision)	22	22
Square Root (Single Precision)	12	12
Square Root (Double Precision)	22	22



**Superskalare** RISC-Rechner brauchen pro Maschinenbefehl höchstens 1 Takt Ausführungszeit.

Die *Ultra Port Architecture* arbeitet statt mit einem Bus-System mit *packet switched* und *crossbar*-Datentransfer:

### Ultra Port Architecture



## UltraSPARC-II

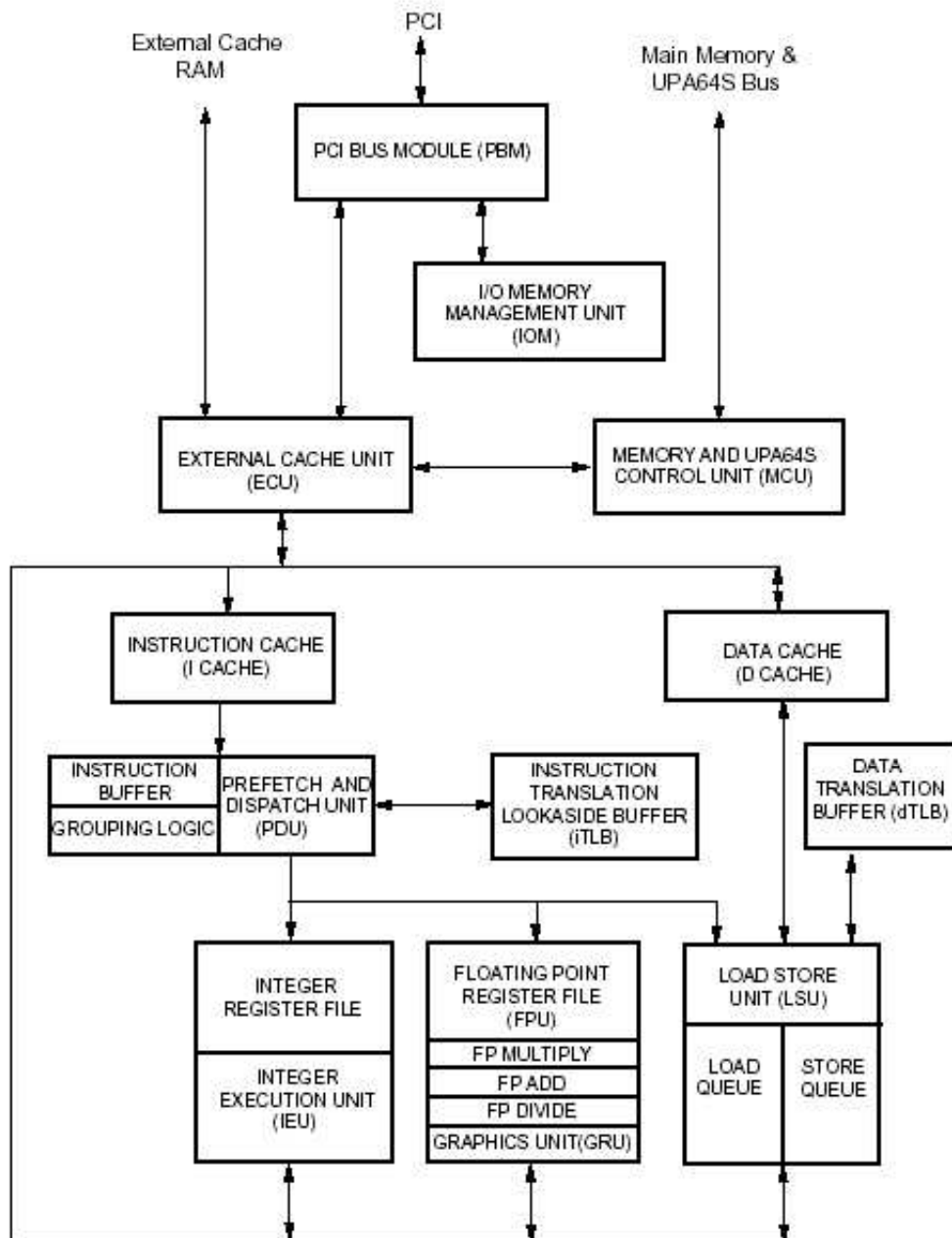


Abbildung 1.34: UltraSPARC-II

- Bis zu 1.92 Gbyte/s Speichertransferzeit
- 4 Instruktionen pro Zyklus (Pipeline der Länge 9)
- 2 ALUs in IEU, 3 FPUs, 1 GRU mit 2 Pipelines
- Real-time MPEG2 Decode
- Prefetch

Siehe auch bei: <http://www.sun.com/processors>

### UltraSPARC-III

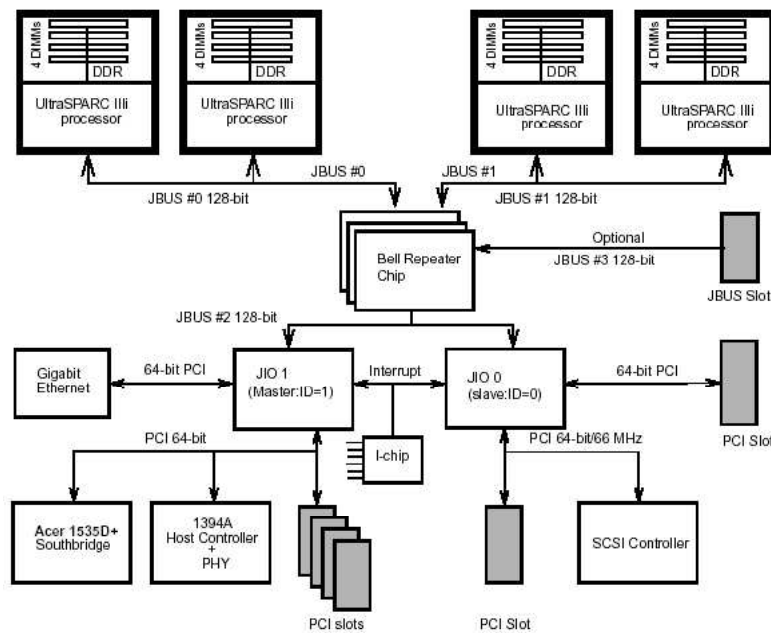


Abbildung 1.35: 4-way UltraSPARC-III

- Bis zu 3.6 GB/s Speichertransferzeit
- IEU mit 2 ALUs, 1 Branch-Unit, 1 Load-/Store-Unit, 2 FPUs, 1 GRU
- 14-stufige CPU-Pipeline
- Cu-Technologie
- speculative execution
- out of order execution
- SIAM-Befehl für Intervall-Arithmetik (set interval arithmetic mode)
- VIS-Extensions (video encoding ... / Multimedia)

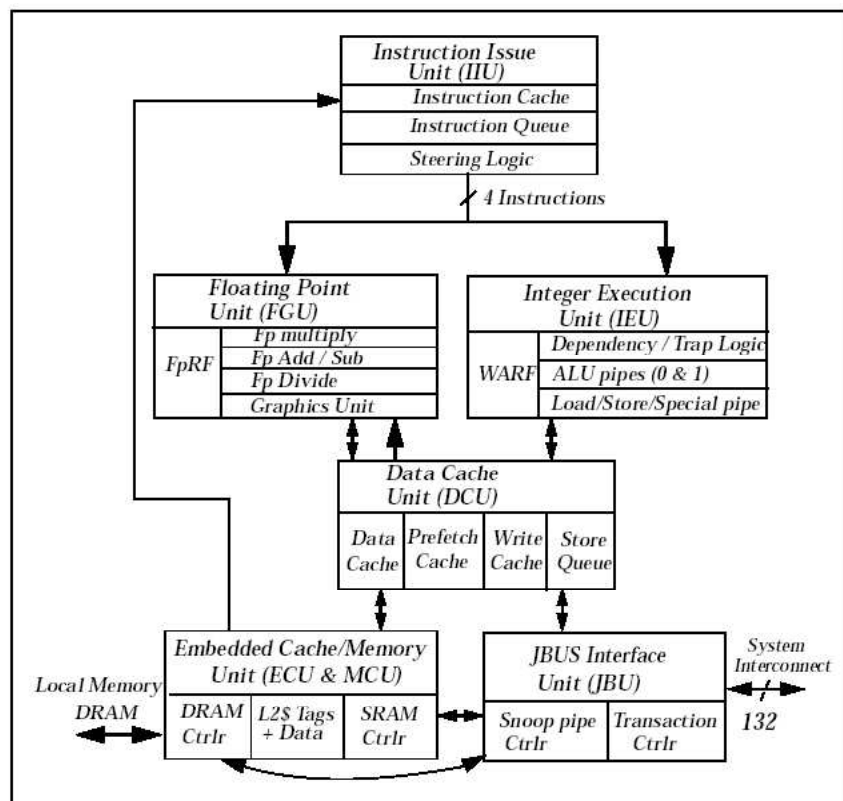


Abbildung 1.36: UltraSPARC-III Functional Units

Siehe auch bei: <http://www.sun.com/sparc/UltraSPARC-III/>

## UltraSPARC-IV

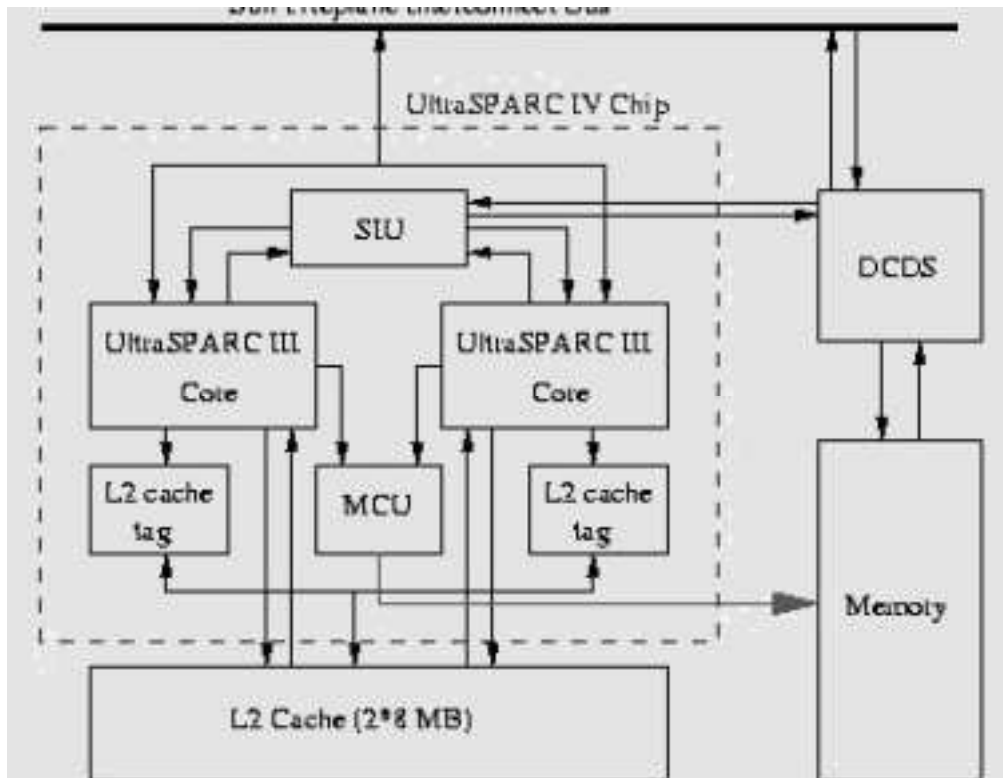


Abbildung 1.37: UltraSPARC-IV

- CMT mit zwei UltraSPARCIII-Pipelines
- speculative execution
- out of order execution
- register mapping
- verbesserte FPU/Cache (fast FFT)
- verbesserte FPU-Arithmetik (IEEE 754)

Siehe bei: <http://www.sun.com/sparc/UltraSPARC-IV/> sowie <http://www.arcade-eu.info/overview/2005/sparcIV.html>



## UltraSPARC-V

Sollte zwei Threads (CMT) sowie VIS Vs. 3 (horizontale Vektorbefehle = Verknüpfung der Teilworte eines Registers; Codierung, ...) bieten.

... ist **abgekündigt** zugunsten von:

## Niagara

.. mit 8 vereinfachten UltraII-Cores zu je 4 Threads (15-fache Geschwindigkeit des UltraSPARC Ii)

## Rock

... erreicht 30-fache Geschwindigkeit der UltraSPARC Ii.

# Chip Multithreading (CMT)

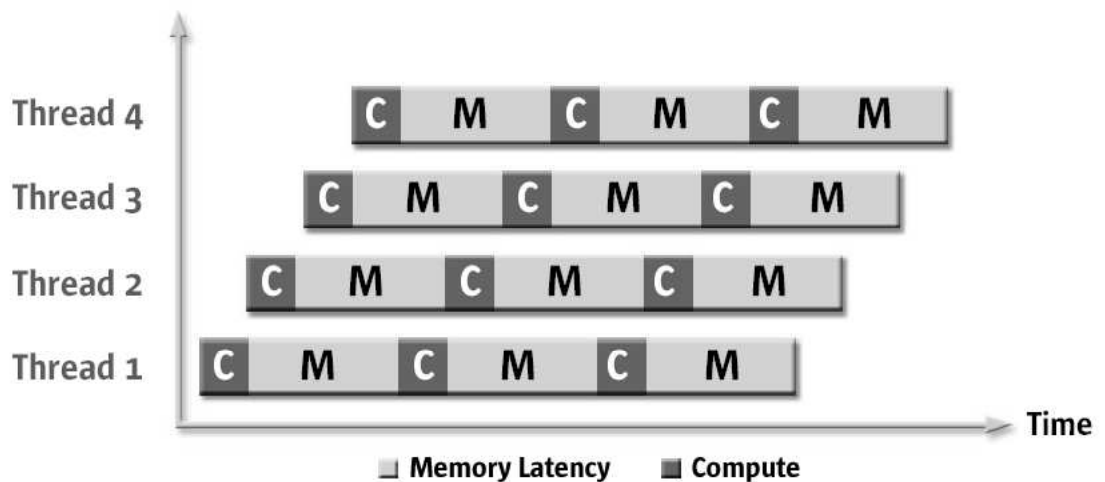


Abbildung 1.38: CMT mit 4 Threads

# CMT – Multiple Multithreaded Cores

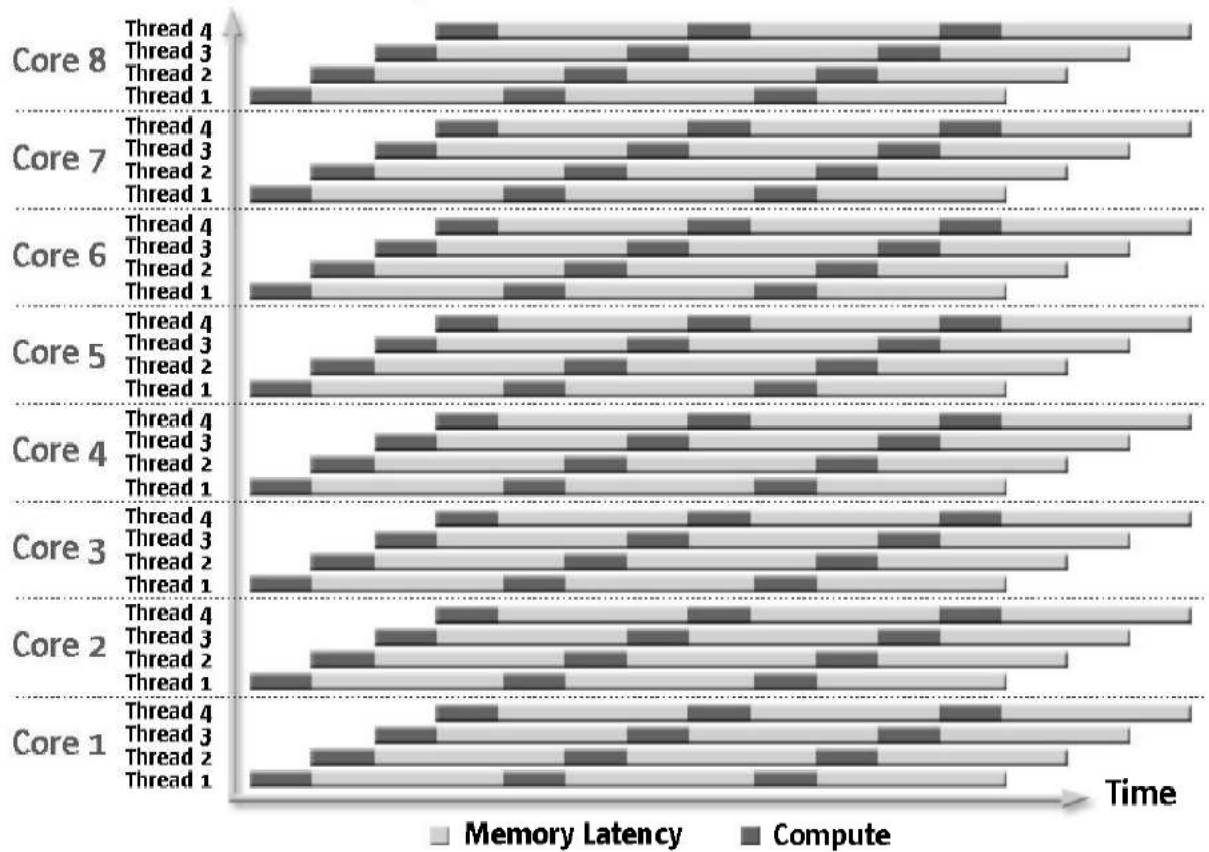


Abbildung 1.39: CMT des Niagara (?)

Sun Roadmap:

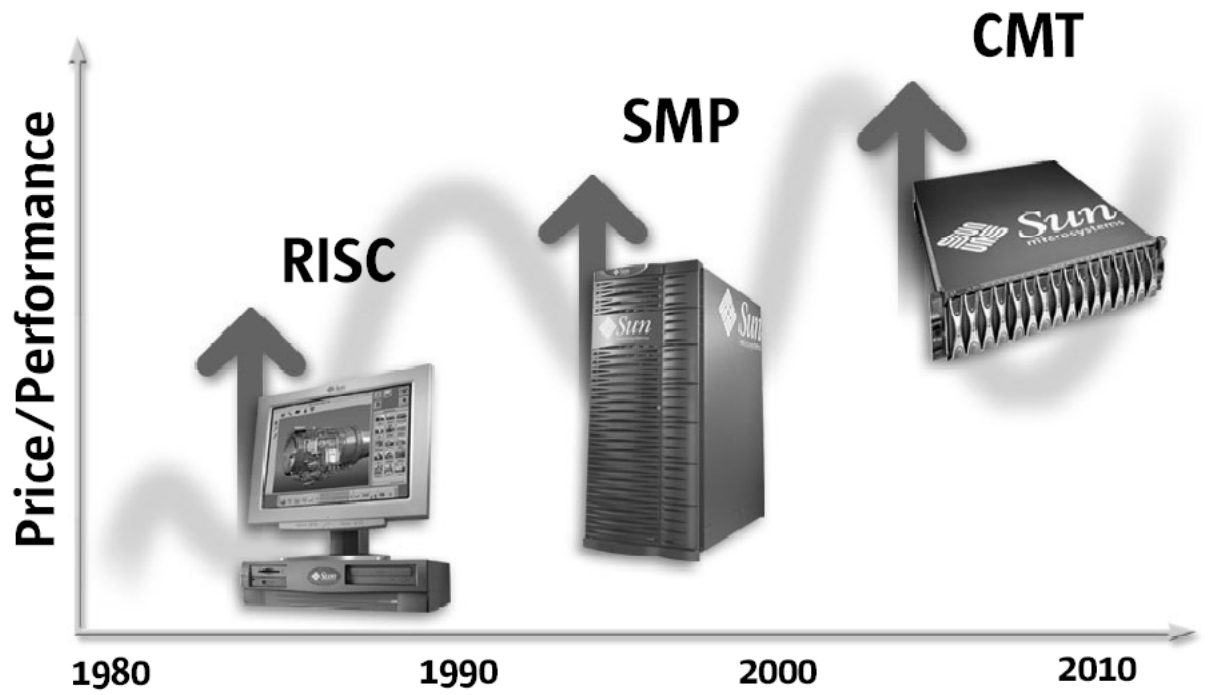


Abbildung 1.40: CMT: viele mehrfachgethreadete Cores auf einem Chip

Weiterführende Literatur:

[http://www.hpcaconf.org/hpca11/slides/hpca\\_cmt\\_slides.pdf](http://www.hpcaconf.org/hpca11/slides/hpca_cmt_slides.pdf)

[http://www.princeton.edu/~jdonald/research/hyperthreading/spracklen\\_opportunities.pdf](http://www.princeton.edu/~jdonald/research/hyperthreading/spracklen_opportunities.pdf)

## 1.13.2 Der Intel **Pentium**

### MMX- und weitere Graphikerweiterungen:

- **MMX** = Matrix-Manipulation-Extensions für 2D-Graphik, Audio, Video (1997: P55C/PentiumII)
- **SSE/STREAMING** 64Bit SIMD-Extensions, auch für 3D-Graphik (1999: PentiumIII)
- **SSE2**: Real-time Video, 128Bit integer-SIMD-Befehle (2000: Pentium4)
- **SSE3**: Wandlung FP zu Integer ohne Rundung (truncate), Arithmetik mit komplexen Zahlen, horizontale Vektorbefehle, 16Byte Ld auf unaligned Adressen, (nicht bei AMD:) Hyperthread-Synchronisations-Befehle: monitor, mwait (2004: Prescott-Pentium4)

### RAM-Architektur

#### Entwicklung der RAM-Bussysteme:

- FP
- EDO
- SDRAM
- DDR-SDRAM
- RDRAM
- Chipsätze für RDRAM (i850), SDRAM (i845) oder SDRAM/DDR-SDRAM (M1671)
- **DDR2**
- **QDR**
- **XDR**
- **XDR2**

## Pentium II



Abbildung 1.41: Pentium II

- MMX = Matrix-Manipulation-Extensions für 2D-Graphik, Audio, Video
- AGP = accelerated Graphics Port
- SDRAM = synchronous dynamic random access memory
- Dynamic Execution Core: multiple branch prediction, data flow analysis, speculative execution
- Dual Independent Bus Architecture (DIB)
- 100 MHz Frontsidebus

Port	Execution Units	Latency/Throughput
0	Integer ALU Unit LEA instructions SHL instructions Integer Multiplication instruction Floating-Point Unit FADD instruction FMUL FDIV Unit MMX ALU Unit MMX Multiplier Unit	Latency 1, Throughput 1/cycle Latency 1, Throughput 1/cycle Latency 1, Throughput 1/cycle Latency 4, Throughput 1/cycle <sup>2</sup> Latency 3, Throughput 1/cycle Latency 5, Throughput 1/2 cycle <sup>1,2</sup> Latency: single precision 17 cycles, double precision 36 cycles, extended precision 56 cycles, Throughput non-pipelined Latency 1, Throughput 1/cycle Latency 3, Throughput 1/cycle
1	Integer ALU Unit MMX ALU Unit MMX Shift Unit	Latency 1, Throughput 1/cycle Latency 1, Throughput 1/cycle Latency 1, Throughput 1/cycle
2	Load Unit	Latency 3 on a cache hit, Throughput 1/cycle <sup>3</sup>
3	Store Address Unit	Latency 3 (not applicable), Throughput 1/cycle <sup>3</sup>
4	Store Data Unit	Latency 1 (not applicable), Throughput 1/cycle

**NOTES:**

See notes following Table 2-1.

Abbildung 1.42: PentiumII Execution Units

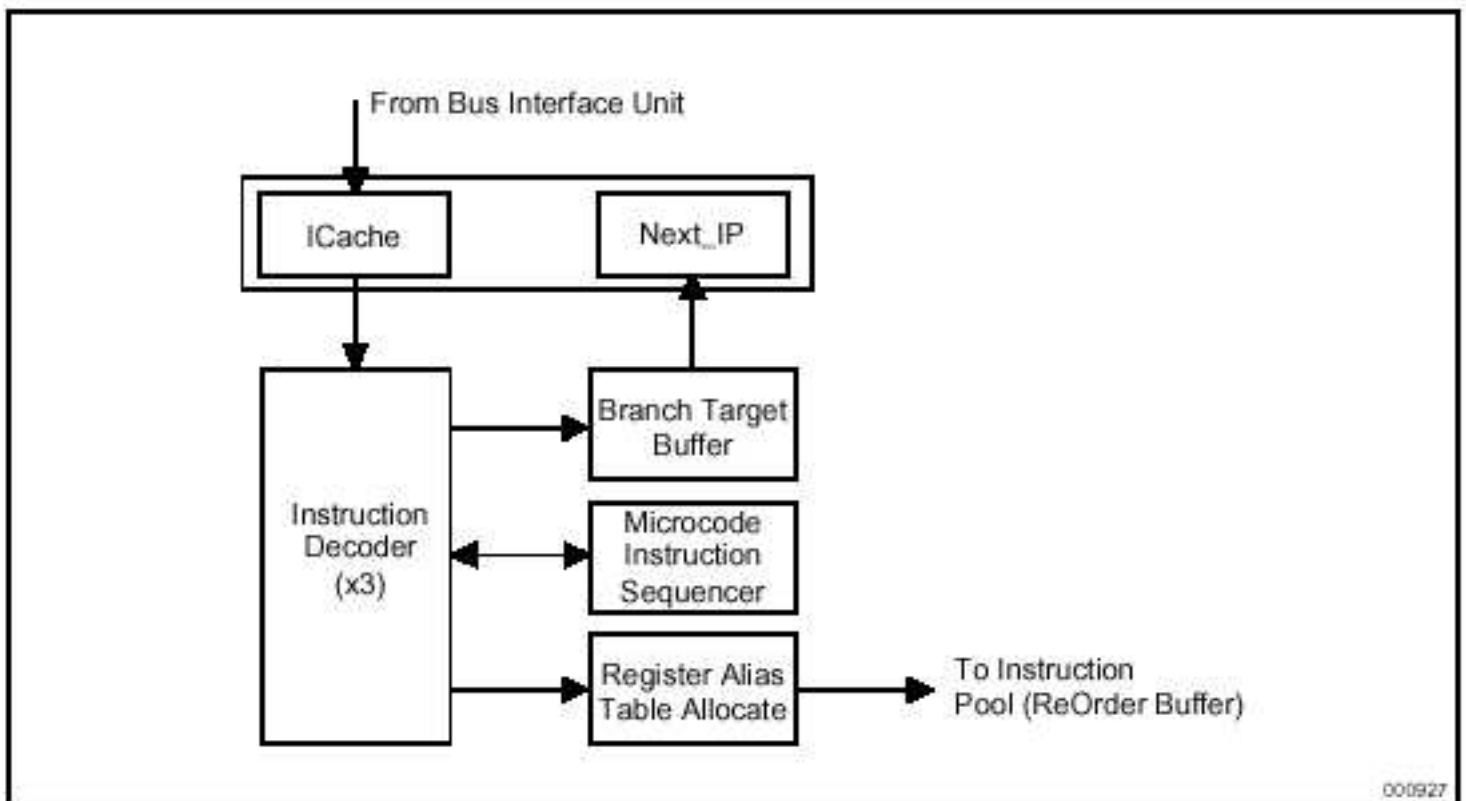


Abbildung 1.43: PentiumII-Design 1

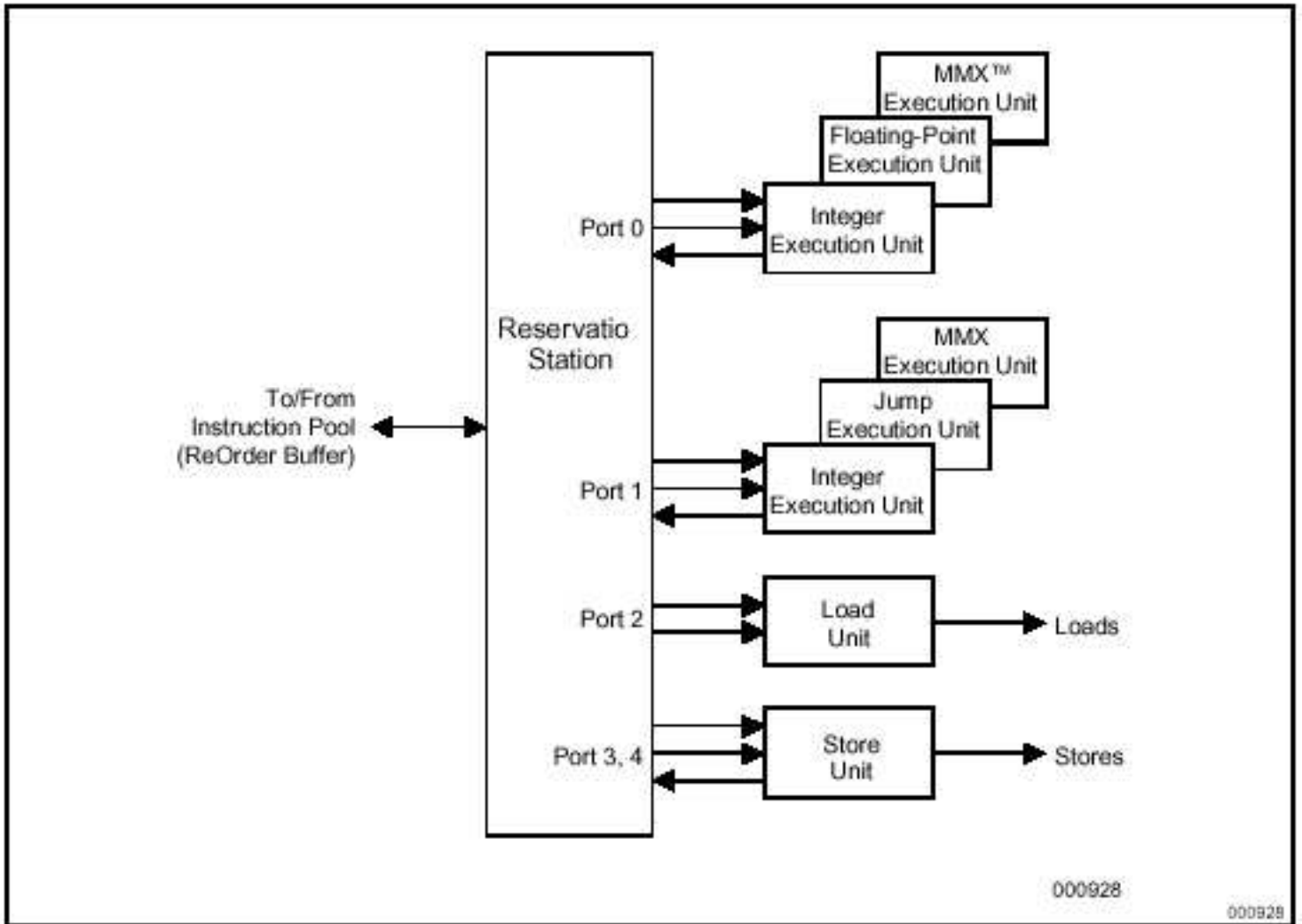


Abbildung 1.44: PentiumII-Design 2



## Pentium III

- SSE = STREAMING 64Bit SIMD-Extensions, auch für 3D-Graphik
- Dynamic Execution Microarchitecture
- Dual Independent Bus
- Intel Processor Serial Number
- 133 MHz Systembus

Siehe auch bei: <http://www.intel.com/pentiumiii/>

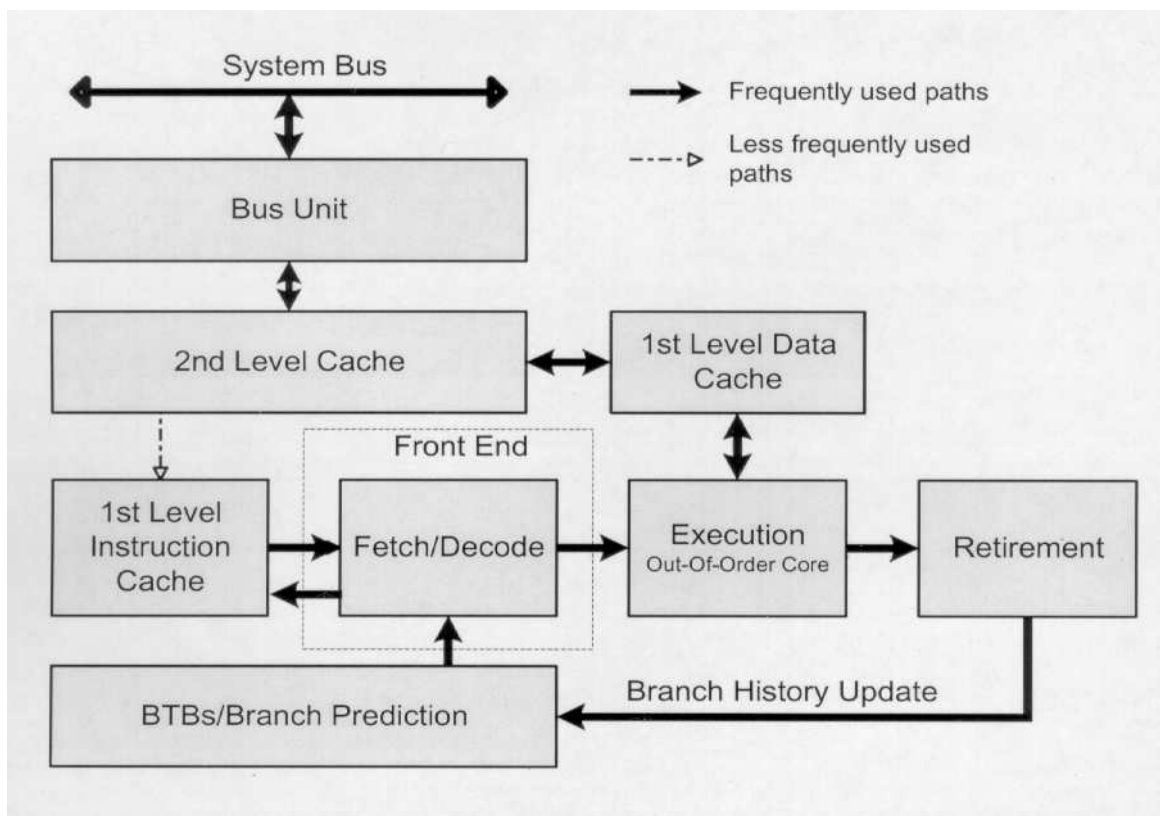


Abbildung 1.45: P6 Architektur

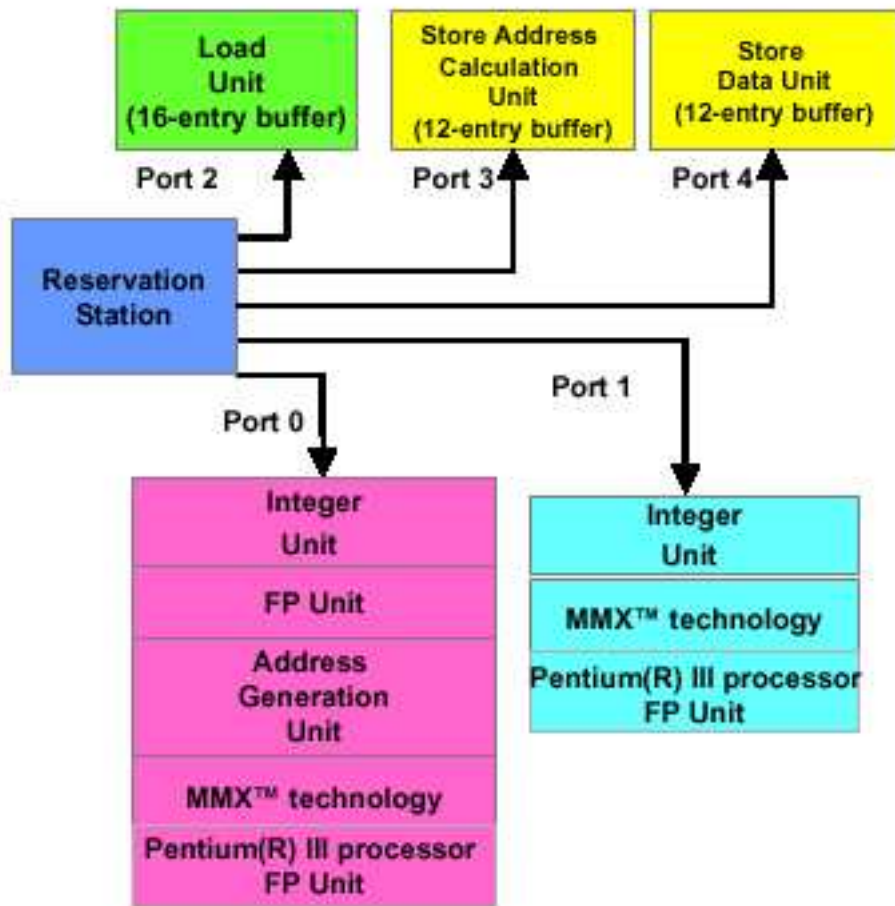


Abbildung 1.46: Dynamic Execution Microarchitecture

Intel Processor	Date Introduced	Max. Clock Frequency at Introduction	Transistors per Die	Register Sizes <sup>1</sup>	Ext. Data Bus Size <sup>2</sup>	Max. Extern. Addr. Space	Caches
8086	1978	8 MHz	29 K	16 GP	16	1 MB	None
Intel 286	1982	12.5 MHz	134 K	16 GP	16	16 MB	Note 3
Intel386 DX Processor	1985	20 MHz	275 K	32 GP	32	4 GB	Note 3
Intel486 DX Processor	1989	25 MHz	1.2 M	32 GP 80 FPU	32	4 GB	L1: 8KB
Pentium Processor	1993	60 MHz	3.1 M	32 GP 80 FPU	64	4 GB	L1:16KB
Pentium Pro Processor	1995	200 MHz	5.5 M	32 GP 80 FPU	64	64 GB	L1: 16KB L2: 256KB or 512KB
Pentium II Processor	1997	266 MHz	7 M	32 GP 80 FPU 64 MMX	64	64 GB	L1: 32KB L2: 256KB or 512KB
Pentium III Processor	1999	500 MHz	8.2 M	32 GP 80 FPU 64 MMX 128 XMM	64	64 GB	L1: 32KB L2: 512KB

Abbildung 1.47: x86-Entwicklung

Intel Processor	Date Introduced	Micro-architecture	Clock Frequency at Introduction	Transistors per Die	Register Sizes <sup>1</sup>	System Bus Bandwidth	Max. Extern. Addr. Space	On-die Caches <sup>2</sup>
Pentium III processor <sup>3</sup>	1999	P6	700 MHz	28 M	GP: 32 FPU: 80 MMX: 64 XMM: 128	Up to 1.06 GB/s	64 GB	32KB L1; 256KB L2
Pentium 4 processor	2000	Intel NetBurst micro-architecture	1.50 GHz	42 M	GP: 32 FPU: 80 MMX: 64 XMM: 128	3.2 GB/s	64 GB	12K $\mu$ op Execution Trace Cache; 8KB L1; 256KB L2

Abbildung 1.48: PentiumIII und 4

## Pentium4

- SSE2: Real-time Video, 128Bit integer-SIMD-Befehle
- NetBurst Microarchitecture: Hyper Pipelined Technology (verdoppelte Pipeline-Länge), Rapid Execution Unit (2 ALUs sind doppelt so schnell wie die restliche CPU getackted, so dass z.B. Integer-Addition in  $\frac{1}{2}$  Taktperiode ausgeführt werden)
- Advanced Dynamic Execution: very deep, out-of-order speculative execution engine, enhanced branch prediction
- Level-1 Cache speichert decodierte Microbefehle
- 400 MHz System Bus

Siehe auch bei: <http://www.intel.com/pentium4/>

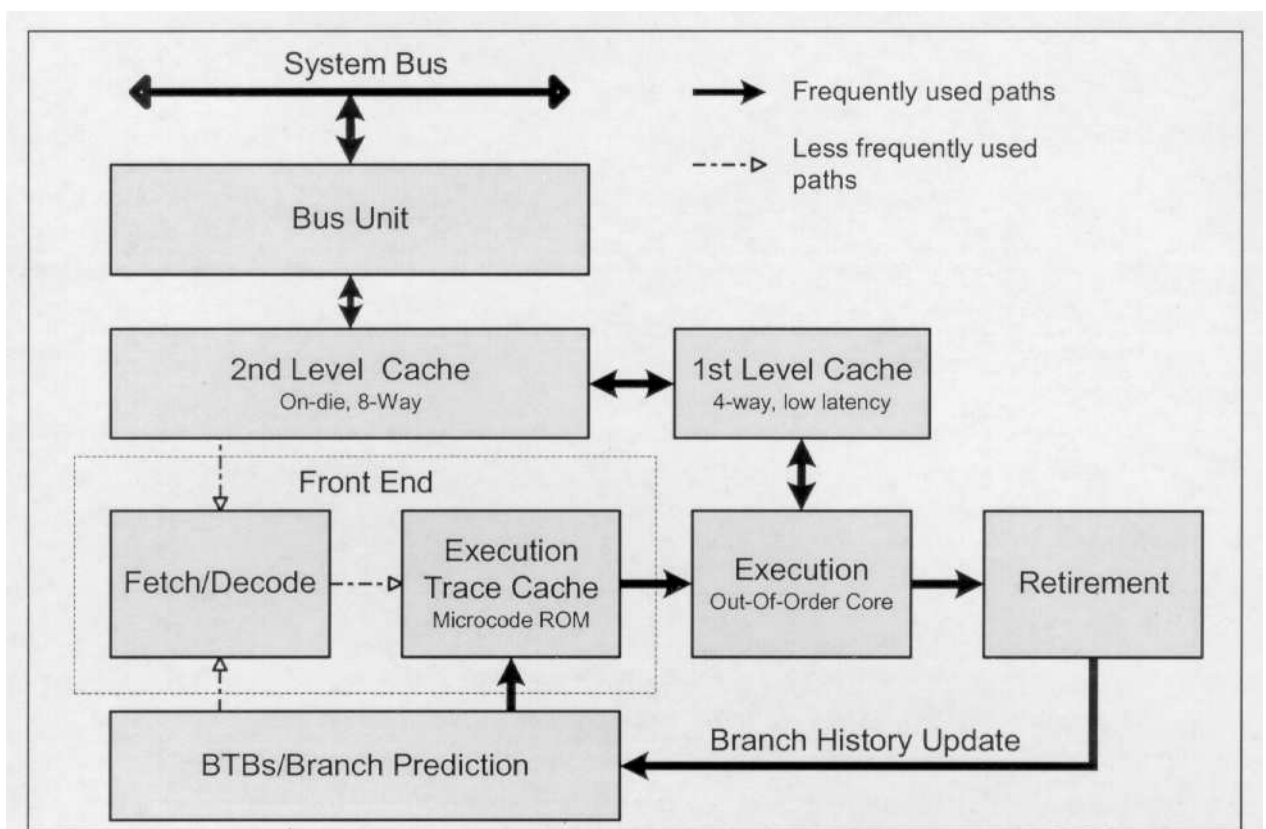


Abbildung 1.49: NetBurst Architektur

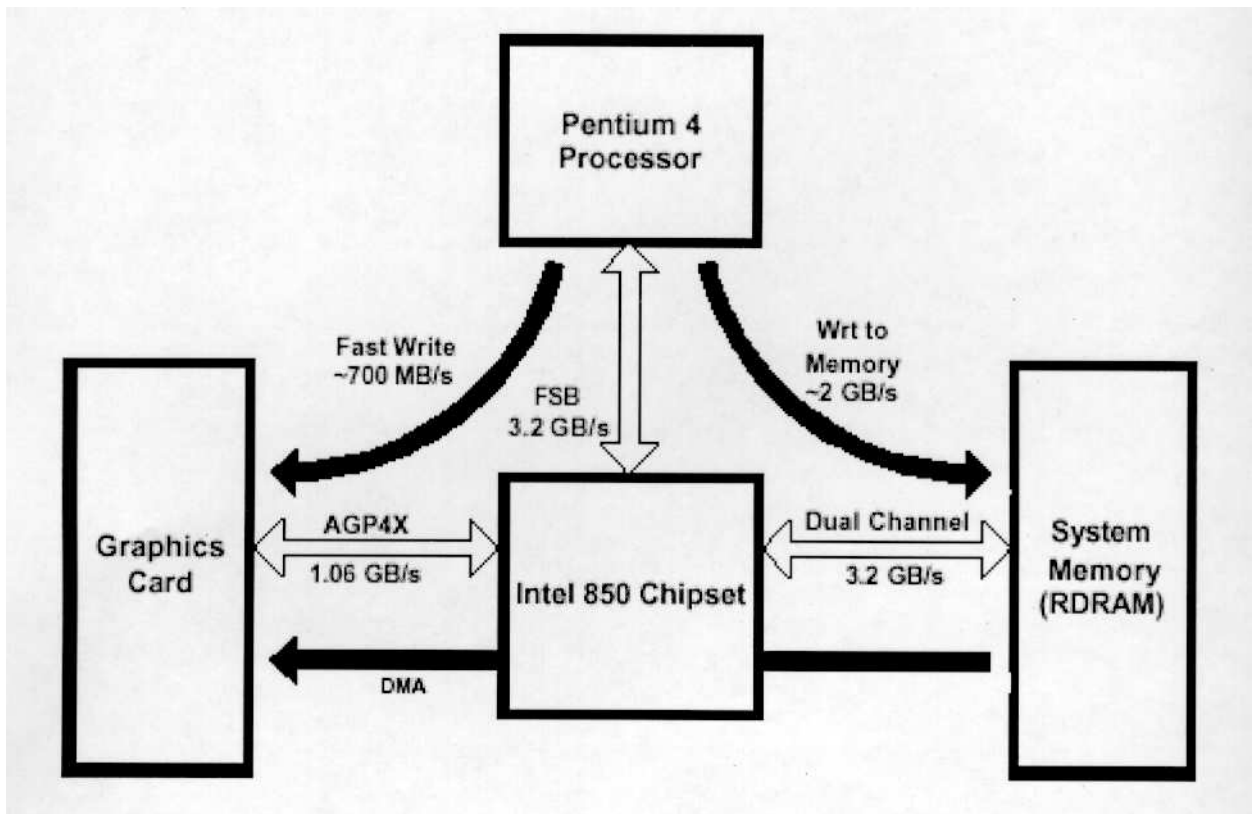


Abbildung 1.50: Datendurchsatz beim Pentium4

Die Zukunft des Pentium4:

<http://www.heise.de/newsticker/meldung/47181>

## 1.14 IA64 - Die EPIC-Architektur

# Explicit Parallel Instruction Computing

- Vermeide “dynamic instruction” scheduling
- Der Compiler bestimmt zur Übersetzungszeit, was parallel ausgeführt werden kann und soll (muß gut optimieren können)
- Breitere Busse zur Durchsatzserhöhung: dadurch insbesondere (V)LIW = very long instruction words

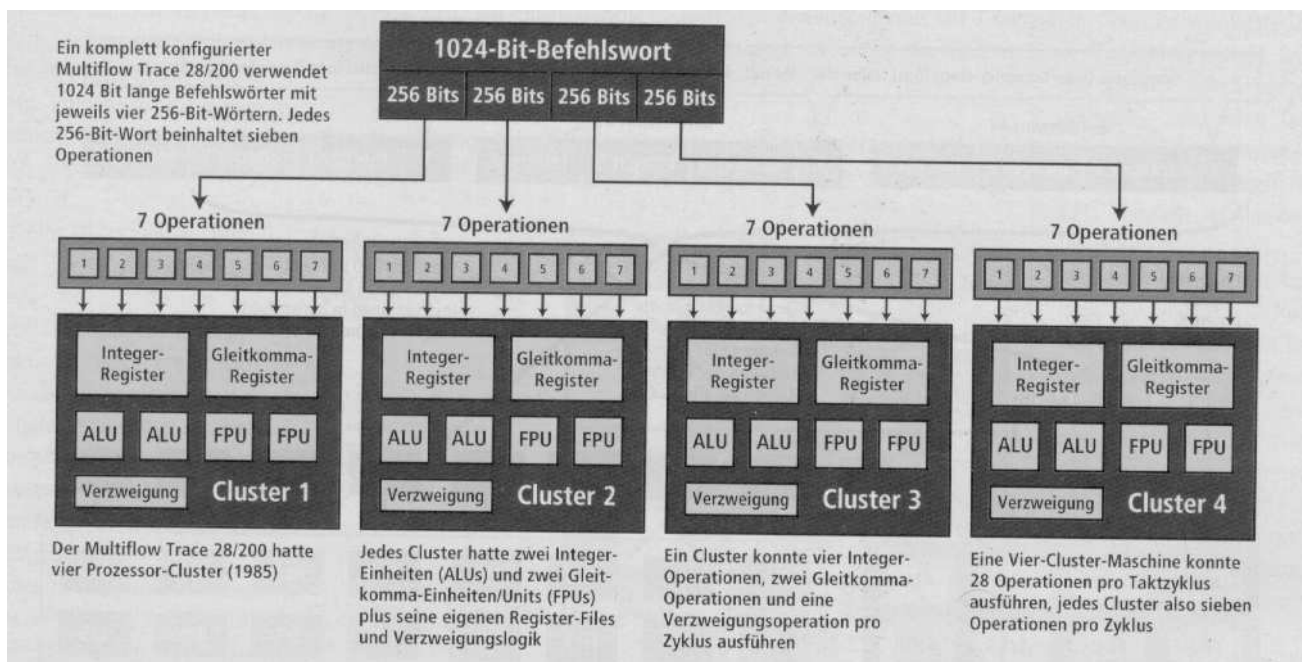


Abbildung 1.51: VLIW

- Branch Predication zur besseren Nutzung der Pipelines
- IUs/FPUs mit vielen SIMD-Befehlen
- Prefetching

### 1.14.1 Itanium

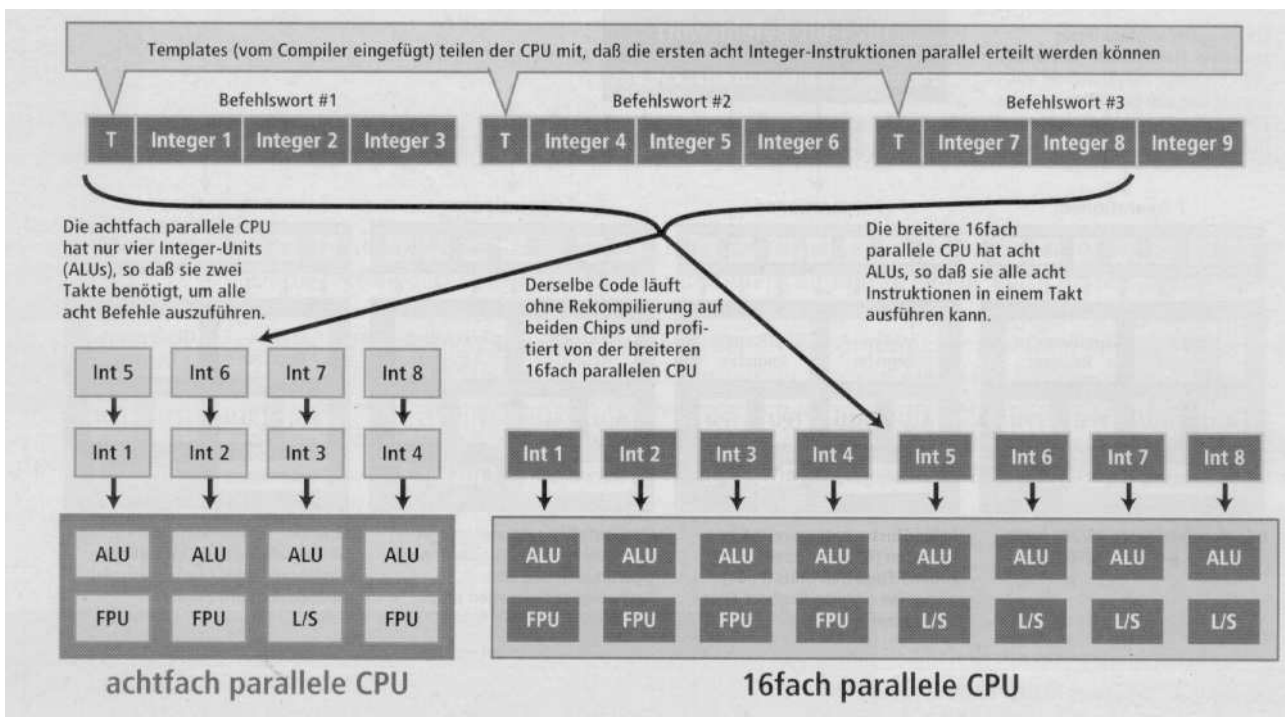


Abbildung 1.52: IA64

- MMX, SIMD, Multimedia-Befehle
- Wide parallel hardware
- 15 execution units: 4 INT, 4 FP, 4 Multimedia, 3 Branch
- L1, L2 und L3-Cache
- 128 FP-Register, rotating
- Registerstack
- Predication und Speculation



- 64Bit-Prozessor (IA64-Architektur)
- 2.1 GB/s Systembus
- 4 GB/s Memorybus
- Binäre IA32-Kompatibilität

Figure 3-1. Application Register Model

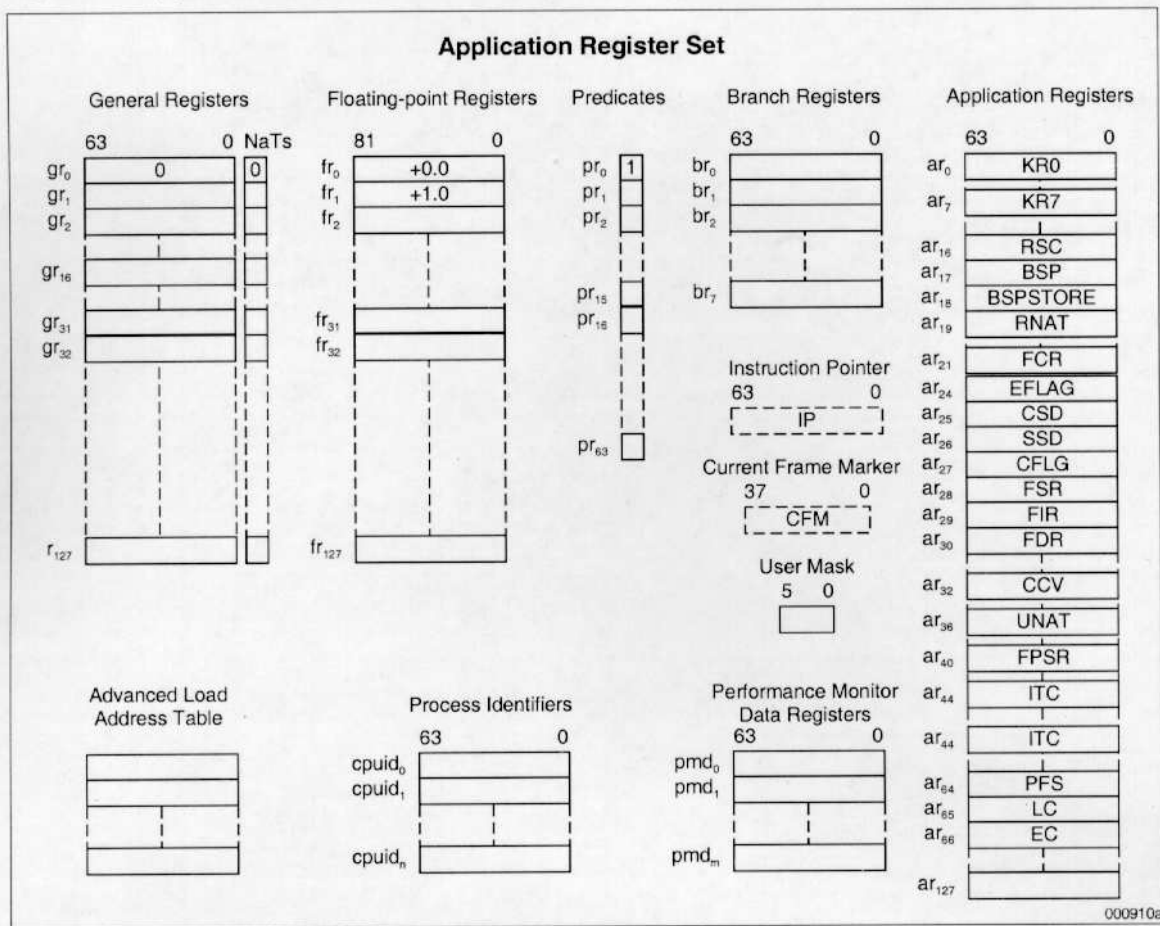
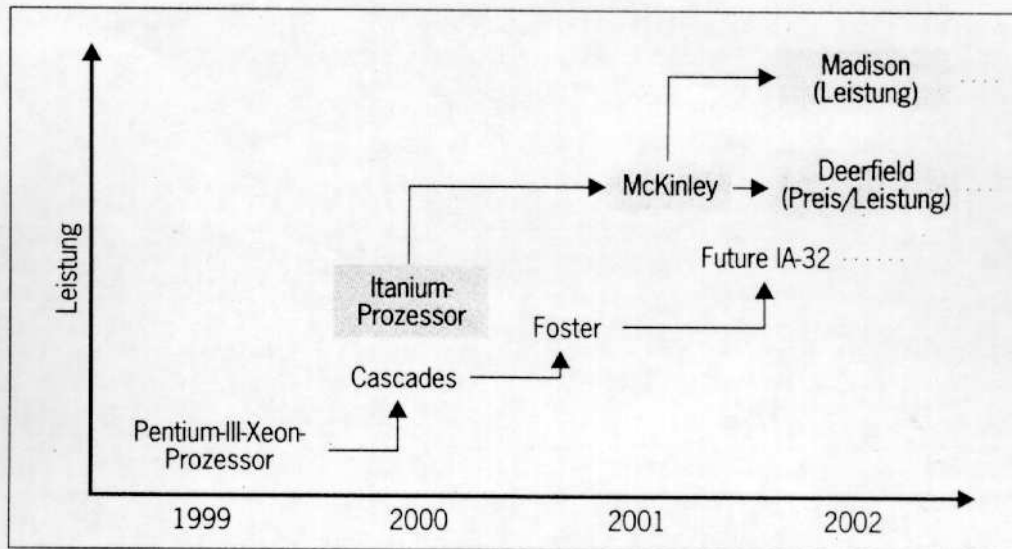


Abbildung 1.53: Register des Itanium



## Roadmap für die IA-64-Architektur



Quelle: Intel

CW 26/01-ww

**Bestandteile des Alpha-Prozessors sollen in Intels Zukunftschip Madison einfließen.**

Abbildung 1.54: Roadmap IA64

## 1.14.2 Itanium2

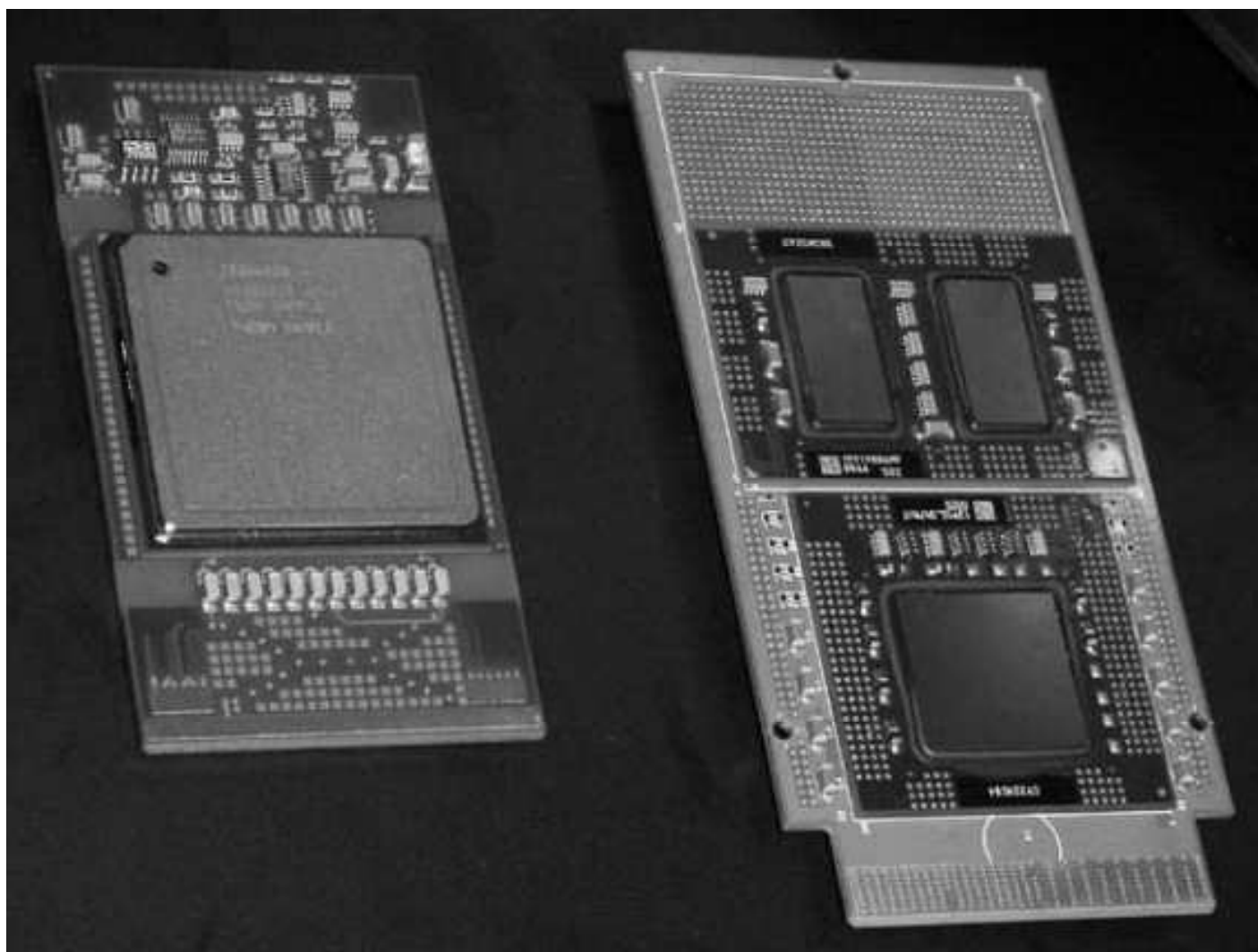


Abbildung 1.55: Itanium2

Weitere Informationen:

- <http://balusc.xs4all.nl/ned/har-cpu-int-ita.php>
- <http://www.heise.de/newsticker/meldung/44719>
- <http://h71028.www7.hp.com/ERC/downloads/5982-9406EN.pdf>
- <http://www.heise.de/newsticker/meldung/60976>
- [https://h30046.www3.hp.com/campaigns/2005/events/integrity\\_servers/images/LE%20Rdmp%2011-04\\_NonNDA.pdf](https://h30046.www3.hp.com/campaigns/2005/events/integrity_servers/images/LE%20Rdmp%2011-04_NonNDA.pdf)

## 1.15 AMDs 64-Bit-Architektur

[http://www.amd.com/us-en/Processors/ProductInformation/0,,30\\_118\\_8796\\_8804,00.html](http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_8796_8804,00.html)

### 1.15.1 Athlon32

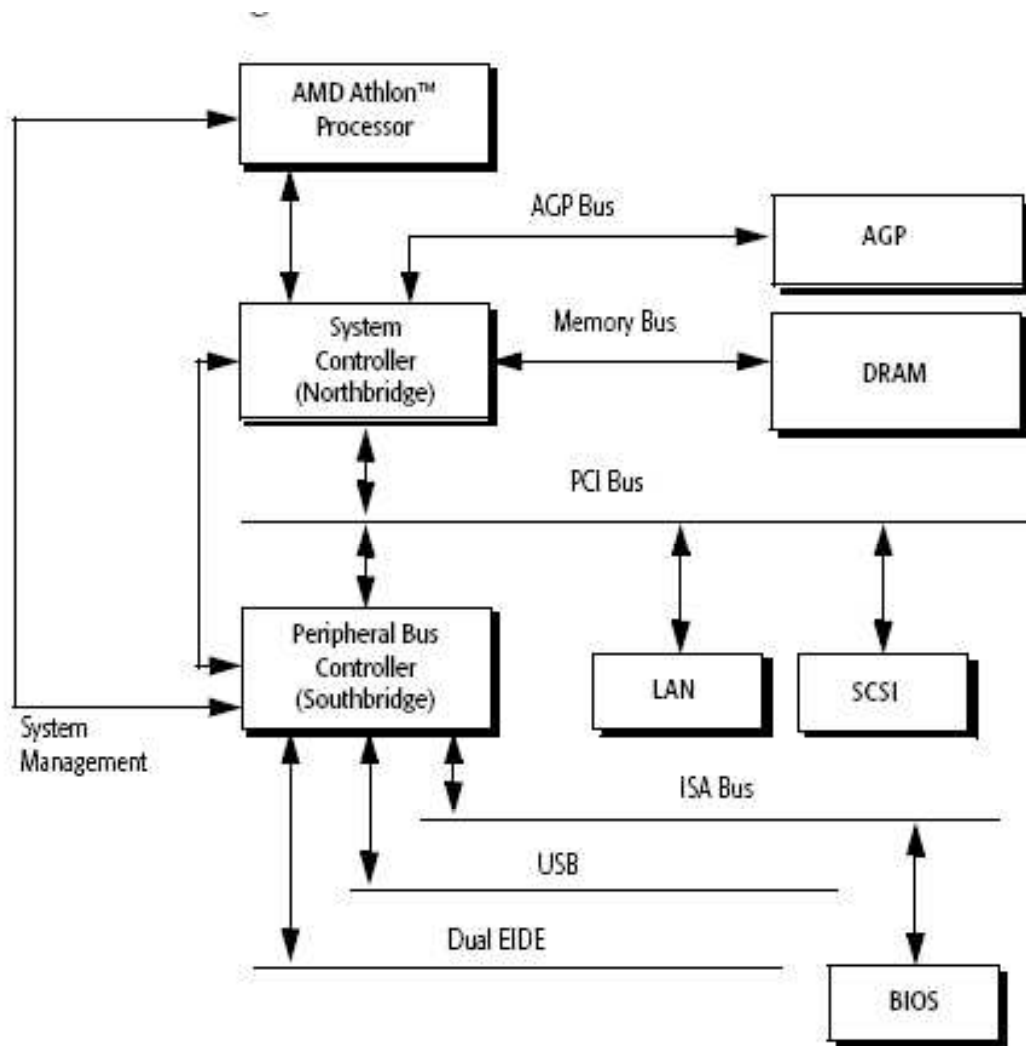


Abbildung 1.56: Athlon 32 Bit mit FSB

Vergleiche: [http://en.wikipedia.org/wiki/List\\_of\\_AMD\\_Athlon\\_microprocessors](http://en.wikipedia.org/wiki/List_of_AMD_Athlon_microprocessors)

### 1.15.2 Athlon64 und Opteron

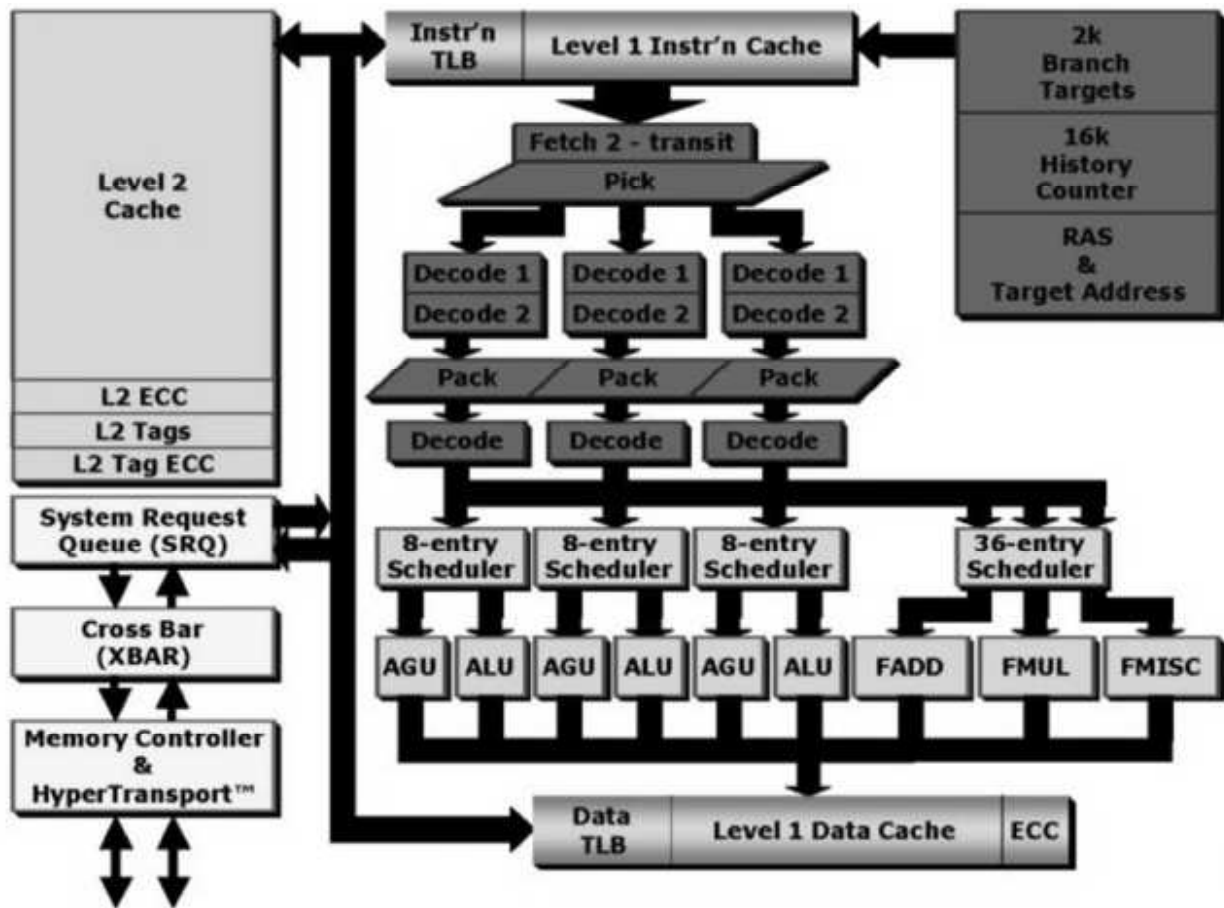


Abbildung 1.57: Opteron

Vergleiche: [http://www.integritylinux.com/pdf/OPTERON\\_AMD64\\_overview.pdf](http://www.integritylinux.com/pdf/OPTERON_AMD64_overview.pdf)

64Bit-Verarbeitung auch im Pentium:  
<http://www.heise.de/newsticker/meldung/44719>

## 1.16 PentiumM

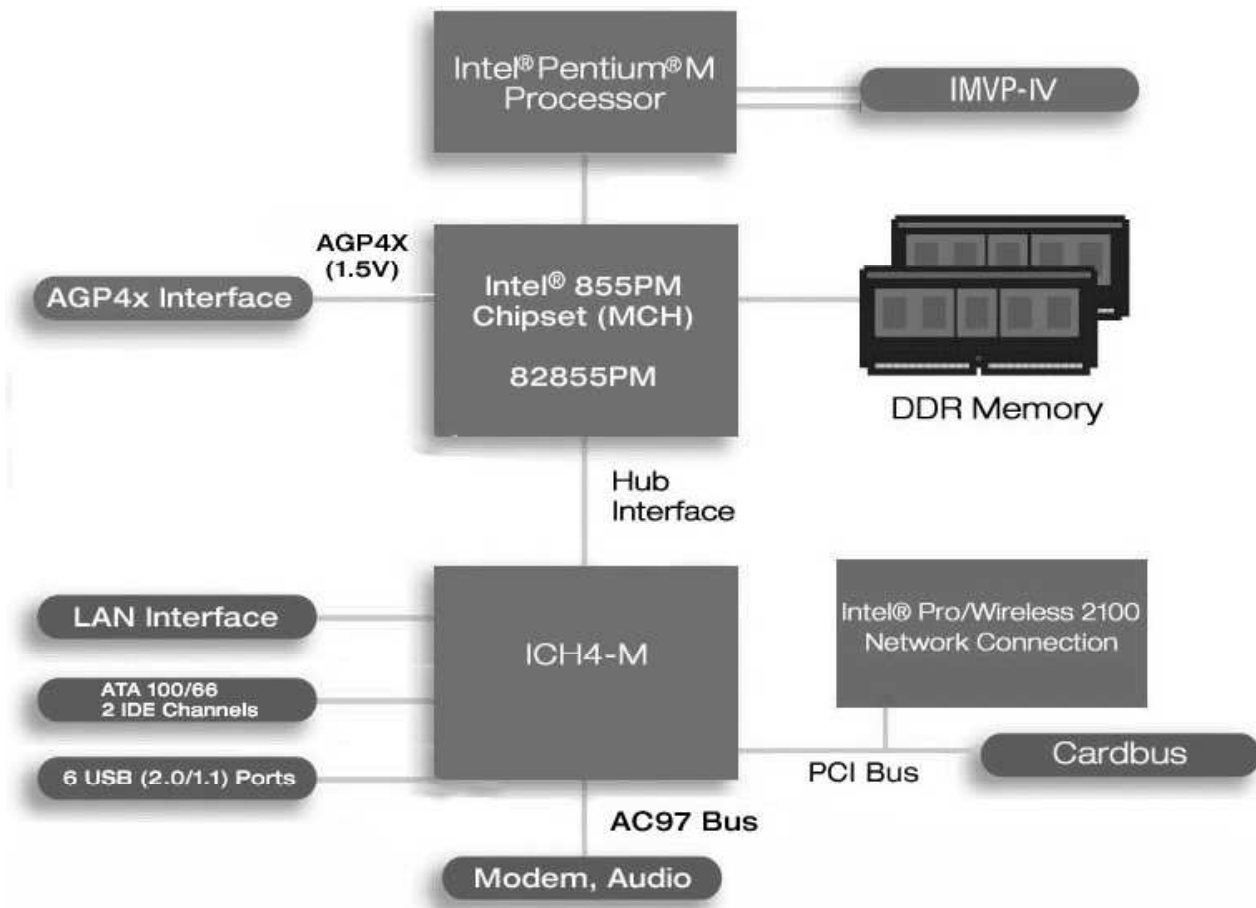


Abbildung 1.58: PentiumM

<http://www.intel.com/products/notebook/processors/pentiumm/>

Als (energiegünstigerer) Ausgangspunkt zur Fortentwicklung eines zukünftigen Pentium-Rechners müssen in die PentiumM-Architektur die vier neuen schon in den Pentium4 integrierten Technologien

- Hyper-Threading
- EM64T
- NX sowie Lagrande-Technologie(TCPA, Trusted Computing, DRM)
- Vanderpool-Technologie

eingebaut werden.

Zur Lagrange-Technologie vergleiche:

<http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html>

<http://moon.hipjoint.de/tcpa-palladium-faq-de.html>

## 1.17 Pentium Doppelkern-Prozessoren

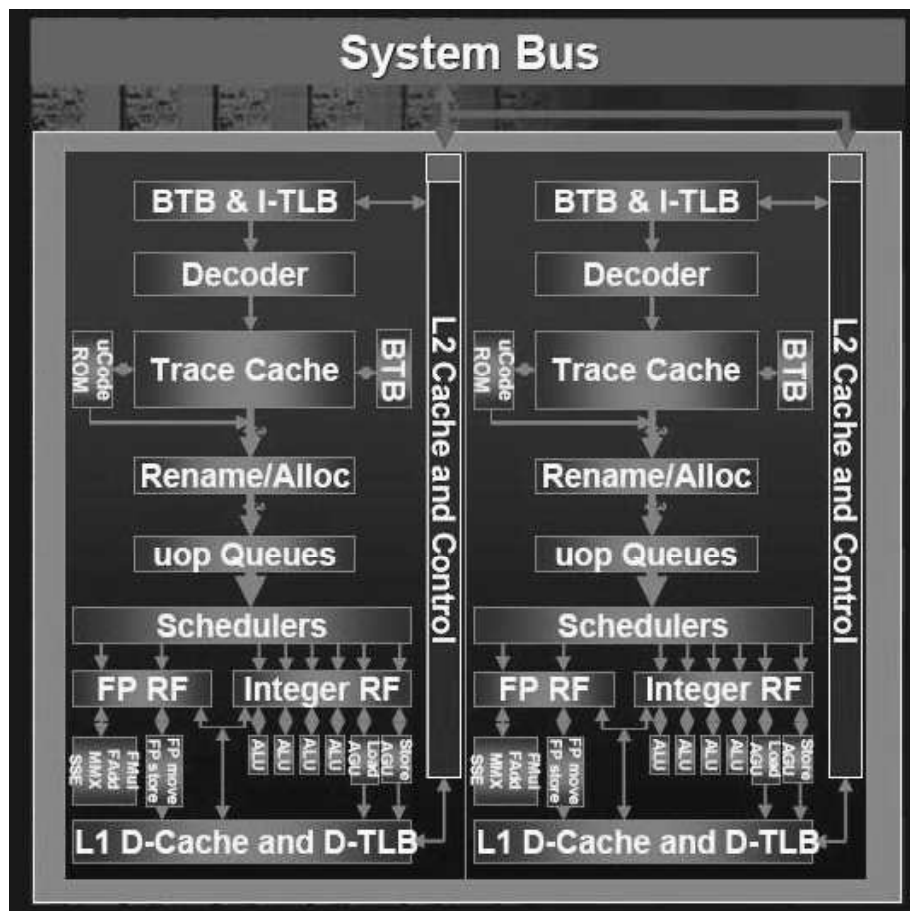
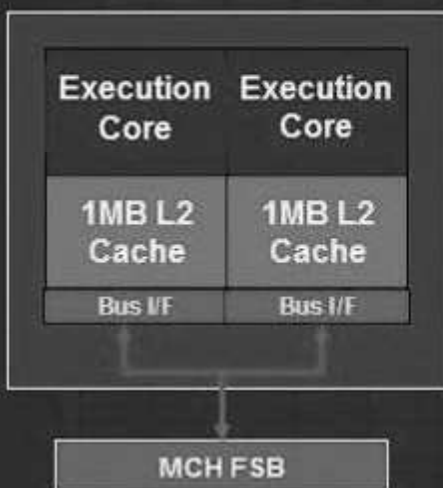


Abbildung 1.59: Dual Core Pentium D

Processor Number <sup>1</sup>	Speed	Cache Size	Intel® EM64T	Execute Disable Bit <sup>2</sup>	Hyper-Threading Technology <sup>1</sup>	Enhanced Intel® Speedstep Technology	Front Side Bus
840	3.20 GHz	2x1MB	Yes	Yes	No	Yes	800
830	3 GHz	2x1MB	Yes	Yes	No	Yes	800
820	2.80 GHz	2x1MB	Yes	Yes	No	No	800

## Intel® Pentium® D Processor Overview (Smithfield)



**Implementation: One piece of silicon; two execution cores**

Summary	
Micro-architecture	NetBurst
L2 Cache	2MB total (1MB per core)
FSB	800 MHz
Intel® EM64T	Yes
Execute Disable Bit	Yes
Socket	LGA 775
Process technology	90nm
Transistors	230M
Die Size	206 mm <sup>2</sup>
Chipset	Intel 945 Express Chipset Family
Availability Target	2Q'05

Abbildung 1.60: Pentium D Smithfield

Weitere Informationen:

[http://www.intel.com/personal/desktopcomputer/dual\\_core/index.htm](http://www.intel.com/personal/desktopcomputer/dual_core/index.htm)

[http://www.intel.com/products/processor/pentium\\_D/](http://www.intel.com/products/processor/pentium_D/)

[http://www.computerbase.de/artikel/hardware/prozessoren/2005/test\\_intels\\_pentium\\_4\\_600-serie](http://www.computerbase.de/artikel/hardware/prozessoren/2005/test_intels_pentium_4_600-serie)



## 1.18 AMD-Doppelkernprozessoren

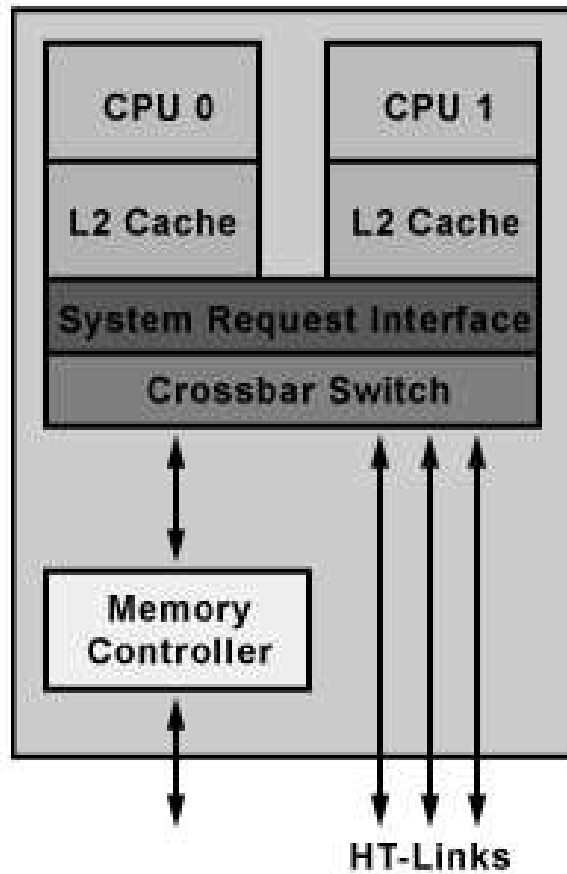


Abbildung 1.61: AMD Dualcore CPU

<http://www.elektronik-kompodium.de/sites/com/1005091.htm>

[http://www.tomshardware.com/cpu/20050509/cual\\_core\\_athlon-11.html](http://www.tomshardware.com/cpu/20050509/cual_core_athlon-11.html)

[http://en.wikipedia.org/wiki/Athlon\\_64\\_X2](http://en.wikipedia.org/wiki/Athlon_64_X2)

<http://en.wikipedia.org/wiki/Multicore>

<http://en.wikipedia.org/wiki/Dual-core>

# Index

- 680x0, 47
- 80x86, 47
- 88000, 65
- Aiken, Howard H., 4
- ASCII, 26
- Babbage, Charles, 4
- Betriebssystem, 3
- Cache, 72
- chip, 48
- CISC, 47, 49, 54
- Computer, 1
- Datendurchsatz, 72
- Datenpfad-Zykluszeit, 53
- DEC-alpha, 57, 67
- Deschutes, 59
- Durchsatz, 72
- Eckert, John P., 4
- EISA, 59
- General Scripts, 32
- Handshake-Steuerbus-Leitungen,  
18
- IBM360, 47
- ISA, 59
- Maschinenbefehle, 2
- Mauchley, John W., 4
- MCA, 59
- Microprogramm, 48
- MIPS, 57, 68
- Monitorprogramm, 3
- Multitasking, 74
- Nanoprogramm, 49
- P2SC, 58
- PCI, 59
- Pentium, 57
- Pentium II, 59
- Pentium MMX, 57
- PentiumPro, 57, 61, 62
- Pipeline, 55
- Power, 70
- Power Architecture, 57, 64
- PowerPC, 58, 69
- preemptiv, 74
- Primitivbefehle, 2
- Rechnerarchitektur, 4
- Registerstack, 56
- RISC, 53, 54
- RISC-Designprinzip, 53
- ROM, 3
- semantic gap, 49
- sequentieller Aneinanderreihung, 2
- SPARC, 57, 63
- SPECfp95, 62
- SPECint95, 61
- Spezialmaschinen, 2
- tasks, 74
- Ultra Part Architecture, 101

Ultra-SPARC, 58  
UNICODE, 31  
Urlader, 3  
  
VAX, 47  
Vektoren, 72  
Vesa Local Bus, 59  
von Neumann, John, 4  
von-Neumann-Computer, 4, 17  
  
wait-states, 48  
  
Zuse, Konrad, 4