

Grundlagen der Rechnerarchitektur —
Neuere Entwicklungen
in der Informationstechnologie

Prof. Dr. Hans-Jürgen Buhl

2001

Fachbereich Mathematik (7)
Institut für Angewandte Informatik
Bergische Universität – Gesamthochschule Wuppertal

Interner Bericht der Integrierten Arbeitsgruppe
Mathematische Probleme aus dem Ingenieurbereich

IAGMPI – 9305

Oktober 1993

5. Auflage, 2001

Praktische Informatik 01

Inhaltsverzeichnis

| | |
|---|------------|
| 1 Rechnerarchitekturen | 1 |
| 1.1 Der von-Neumann-Rechner | 9 |
| 1.2 Modularer Aufbau von Betriebssystemen | 29 |
| 1.3 CISC/RISC | 33 |
| 1.3.1 CISC | 33 |
| 1.3.2 „Überalterung“ von CISC | 36 |
| 1.3.3 RISC (= reduced instruction set computer)-Designprinzip | 39 |
| 1.3.4 Historischer CISC-RISC CPU-Vergleich | 43 |
| 1.4 Der Cache zur Datentransferbeschleunigung | 55 |
| 1.4.1 Multitasking | 56 |
| 1.5 Bussysteme — Chipsätze und das Motherboard | 59 |
| 1.6 Der Weg fort von der von-Neumann-Architektur | 60 |
| 1.6.1 SPARC | 60 |
| 1.6.2 Der Intel Pentium | 74 |
| 1.7 IA64 - Die EPIC-Architektur | 84 |
| 1.7.1 Itanium | 85 |
| 1.8 Zukünftige CPU-Entwicklungen | 88 |
| 2 High Performance Computer | 89 |
| 2.1 Vektorrechner | 89 |
| 2.2 SIMD- und MIMD-Parallelrechner | 95 |
| 2.3 Mehrprozessorsysteme | 100 |
| 2.3.1 Multithreadsysteme | 102 |
| 2.4 Multiprozessor-Server | 103 |
| 3 Rechnerarchitekturen – Fortsetzung | 104 |
| 3.1 Peripheriebus: USB und Firewire | 104 |
| 3.1.1 USB = Universal Serial Bus | 104 |
| 3.1.2 Firewire, IEEE 1394 | 105 |
| 3.1.3 USB2 | 105 |
| 3.1.4 Firewire, IEEE 1394b | 105 |

| | |
|-------------------------------|-----|
| 3.2 Neuronale Netze | 106 |
|-------------------------------|-----|

Abbildungsverzeichnis

| | | |
|------|--|----|
| 1.1 | Computersystem | 5 |
| 1.2 | Der ZUSE Z3 von Konrad Zuse | 6 |
| 1.3 | Universalrechner im Aufbau | 11 |
| 1.4 | „circuit switched“ Datentransfer | 12 |
| 1.5 | Datentypen | 14 |
| 1.6 | Verteilung UNICODE | 19 |
| 1.7 | Bidirectional Ordering | 20 |
| 1.8 | General Scripts | 21 |
| 1.9 | Die Bytes im Speicher ... | 27 |
| 1.10 | ... und ihre Reihenfolge im Computerwort | 27 |
| 1.11 | CPU | 28 |
| 1.12 | Windows 95/98 | 29 |
| 1.13 | Windows NT | 30 |
| 1.14 | Solaris/Linux | 31 |
| 1.15 | Microcode und Betriebssystem | 32 |
| 1.16 | CISC-CPU-Aufbau | 33 |
| 1.17 | Microprogrammierte CISC-CPU | 34 |
| 1.18 | Nanoprogrammierte CISC-CPU | 35 |
| 1.19 | Der x86 MOV-Befehl | 37 |
| 1.20 | Der x86 MUL-Befehl | 38 |
| 1.21 | Registerstack am Beispiel SPARC | 42 |
| 1.22 | Der Cache als Daten-Vorratsbehälter | 55 |
| 1.23 | Multitask-Stati | 57 |
| 1.24 | Cache auf dem Motherboard | 58 |
| 1.25 | PC Frontsidebus | 59 |
| 1.26 | PentiumII Execution Units | 76 |
| 1.27 | PentiumII-Design 1 | 77 |
| 1.28 | PentiumII-Design 2 | 78 |
| 1.29 | P6 Architektur | 79 |
| 1.30 | Dynamic Execution Microarchitecture | 80 |
| 1.31 | x86-Entwicklung | 81 |

| | | |
|------|--|-----|
| 1.32 | PentiumIII und 4 | 81 |
| 1.33 | NetBurst Architektur | 82 |
| 1.34 | Datendurchsatz beim Pentium4 | 83 |
| 1.35 | VLIW | 84 |
| 1.36 | IA64 | 85 |
| 1.37 | Register des Itanium | 86 |
| 1.38 | Roadmap IA64 | 87 |
| 2.1 | Verarbeitungseinheit: Vektor von Worten | 89 |
| 2.2 | Vektorregister und Vektoralu | 90 |
| 2.3 | Rechnerstruktur eines Cray-1-Rechners | 92 |
| 2.4 | Die Register und Funktionseinheiten einer Cray-1 | 93 |
| 2.5 | CRAY-2 (Juni 1985) | 95 |
| 2.6 | NEC-SX2 SIMD, mehrere Pipelines | 97 |
| 2.7 | Systemarchitektur und Datendurchsatz | 99 |
| 2.8 | „shared memory“ und „message passing“ | 101 |

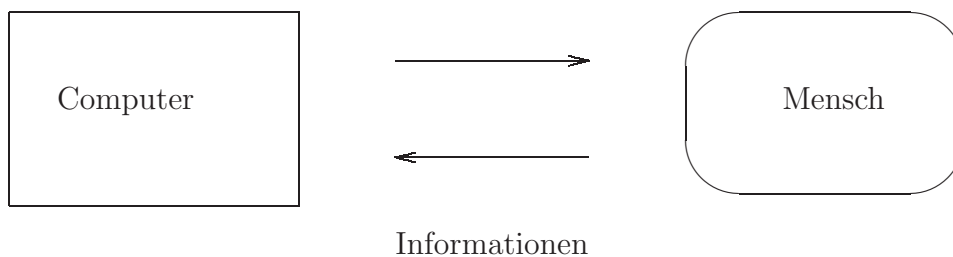
Tabellenverzeichnis

| | | |
|------|--|-----|
| 1.1 | Zeittafel zur Entwicklung der Computertechnik | 4 |
| 1.2 | Merkmale der 1. bis 3. Computergeneration | 9 |
| 1.3 | ASCII-Code | 15 |
| 1.4 | ISO-Austauschtabelle | 15 |
| 1.5 | PC-8 Zeichensatz | 16 |
| 1.6 | Zeichensatz für Windows 3.x | 17 |
| 1.7 | ISO-8859 Latin 1(ECMA-94 Latin 1) Zeichensatz | 18 |
| 1.8 | nationale ISO8859-Varianten | 18 |
| 1.9 | UNICODE Version 1.0, Character Blocks 0000-00FF | 21 |
| 1.10 | Weitere Zeichenbereiche | 22 |
| 1.11 | UNICODE to Adobe Standard Mappings | 24 |
| 1.12 | The UNICODE to SGML (ISO DIS 6862.2) Mappings | 25 |
| 1.13 | UNICODE to Macintosh Mappings | 26 |
| 1.14 | Analyse typischer Computeranwendungen auf ihre Instruktionstypen hin | 36 |
| 1.15 | Von CISC zu RISC | 39 |
| 1.16 | Unterscheidungsmerkmale CISC/RISC | 40 |
| 1.17 | RISC-CPU's im Vergleich zu Intel 80x86-CPU's | 43 |
| 1.18 | 64 Bit RISC-CPU's | 44 |
| 1.19 | PC-Bussysteme und Durchsatzraten | 44 |
| 1.20 | RISC PC-CPU's: PowerPC (Apple und IBM) | 44 |
| 1.21 | CISC PC-CPU's: i80x86 | 45 |
| 1.22 | CISC PC-CPU's: i80x86 (Forts.) | 46 |
| 1.23 | Marktallianzen im RISC-Bereich | 49 |
| 1.24 | Anzahl der Register-Fenster und Integer-Zyklen pro Anweisung | 68 |
| 1.25 | „floating point“-Zyklen pro Anweisung | 69 |
| 2.1 | Klassifikation von Mehrprozessorsystemen | 100 |
| 2.2 | Speicher bei Mehrprozessorsystemen | 101 |

Kapitel 1

Rechnerarchitekturen

Vom (mechanischen) Spezialgerät zum Universalrechner



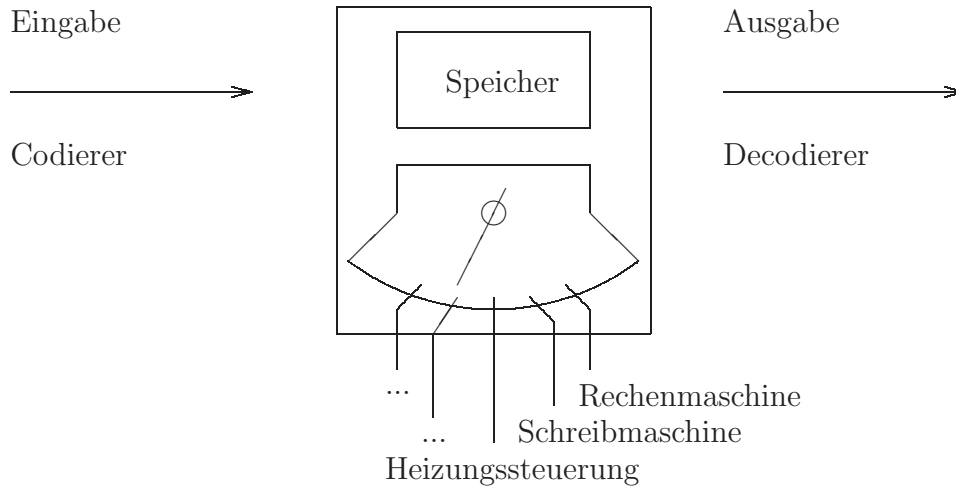
Im Gegensatz

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

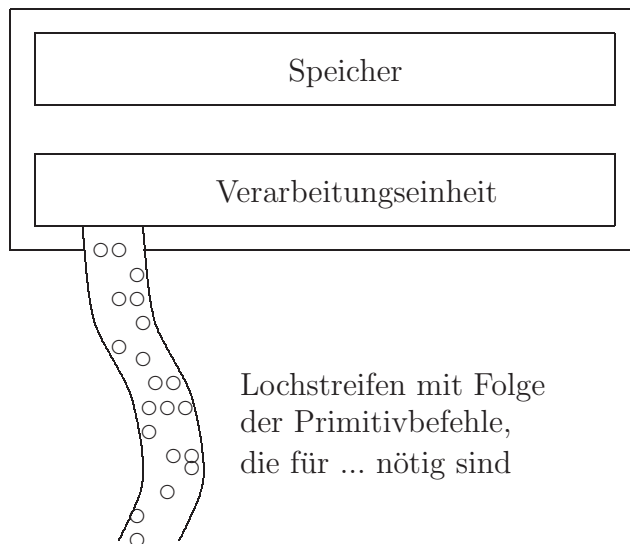
ist ein *Computer* ein universell einsetzbares Gerät zur Verarbeitung von Daten (Rechnen, Datenverarbeitung (DVA), Bereitstellung von Informationen). Diese Universalität wird dadurch erreicht, daß

- 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-

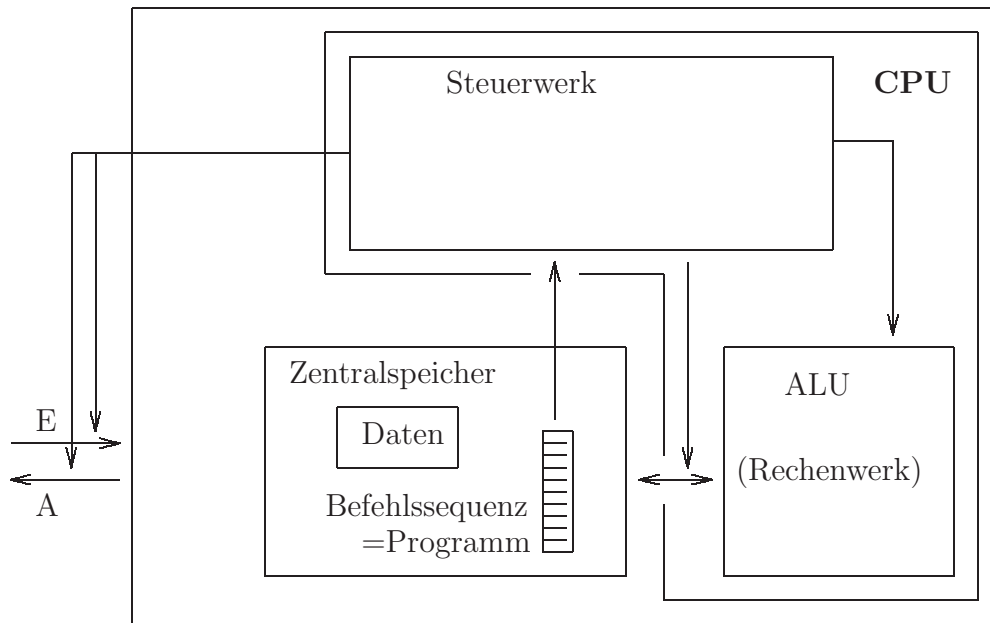


versalität erreicht; lediglich eine durch Schalterstellung definierbare *1 aus N* Spezialauswahl ist realisiert. Sieht man jedoch eine große Anzahl von die Daten nur gering ändernden *Spezialmaschinen*, sogenannte *Primitivbefehle* oder *Maschinenbefehle* vor, und baut aus *sequentieller* Aneinanderreihung 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* oder *Monitorprogramms* geladen.

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

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 Mikroprozessorrära. 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

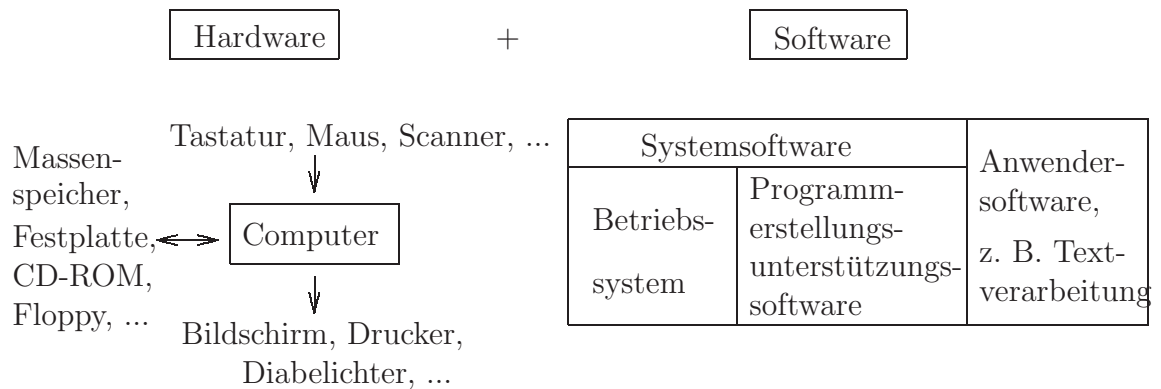


Abbildung 1.1: Computersystem

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.qvctc.commnet.edu/classes/csc277/bios.html>
- Der PC-Bootvorgang: <http://www.qvctc.commnet.edu/classes/csc277/boot.html>
- Motherboards und BIOS: <http://www.abios.com/>
- BIOS-Versionen: etwa in [http://www.heise.de/newsticker/data/jow-19.05.01-000/ ...](http://www.heise.de/newsticker/data/jow-19.05.01-000/)

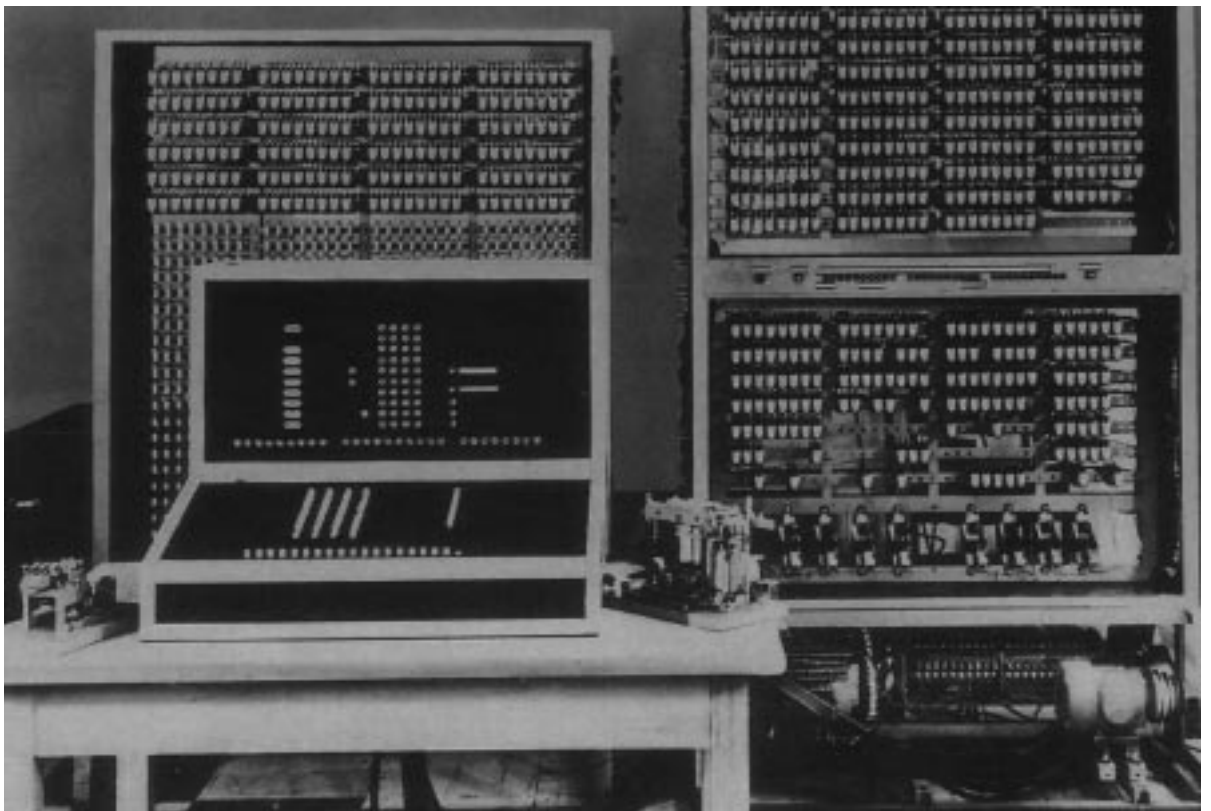
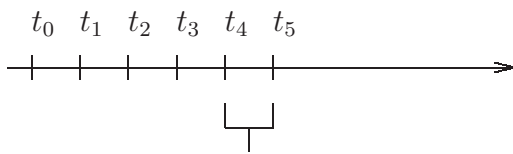
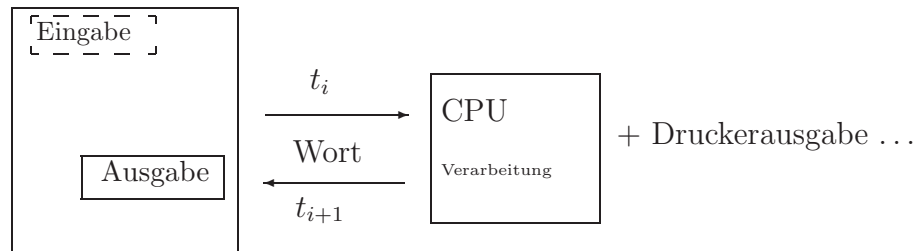
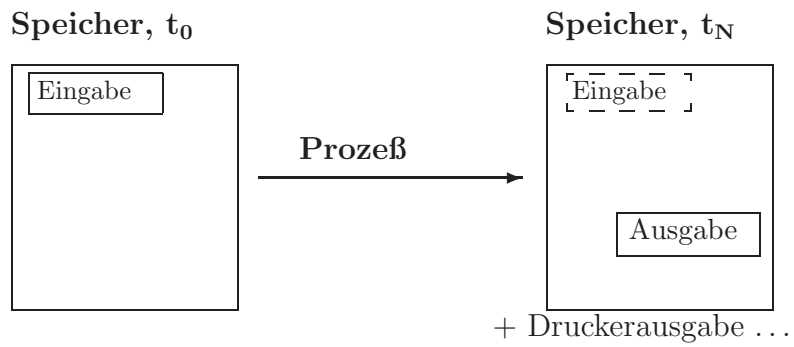


Abbildung 1.2: Der ZUSE Z3 von Konrad Zuse

Das BIOS-Setup

Das BIOS-Setup erlaubt das Einstellen der Systemkonfiguration:

- Setzen von Datum und Uhrzeit
- (früher:) Einstellung der Festplattengeometrie
- Auswahl der Bootdevices (Platte, Diskette, CDROM, ...)
- Auswahl der Reihenfolge der Bootdevices
- Konfiguration der Schnittstellen (parallele/serielle/...)
- Auswahl der Prozessorgeschwindigkeit
- Konfiguration des Stromsparmmodus (für Notebooks)
- Konfiguration des Powermanagements
- Vereinbarung von Systempasswörtern
- ...



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

| | 1. Generation ab 1951 | 2. Generation ab 1960 | 3. Generation ab 1965 |
|------------------------|--|---|---|
| Basiselement | Elektronenröhre | Flächentransistor | integrierter Schaltkreis |
| Arbeitsspeicher | Magnettrommel-, Umlaufspeicher | Ferritkernspeicher | Mikro ferritkernspeicher |
| 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_EX, ...)
- 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.

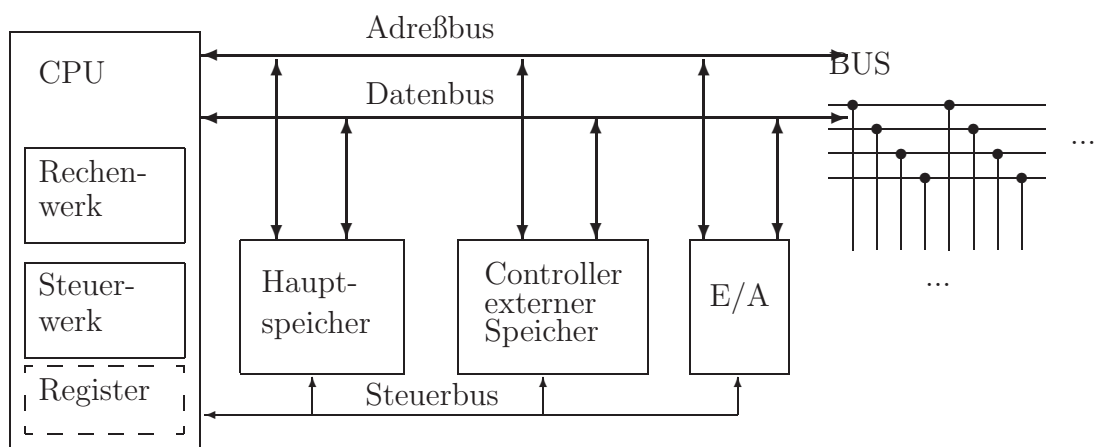


Abbildung 1.3: Universalrechner im Aufbau

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.

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

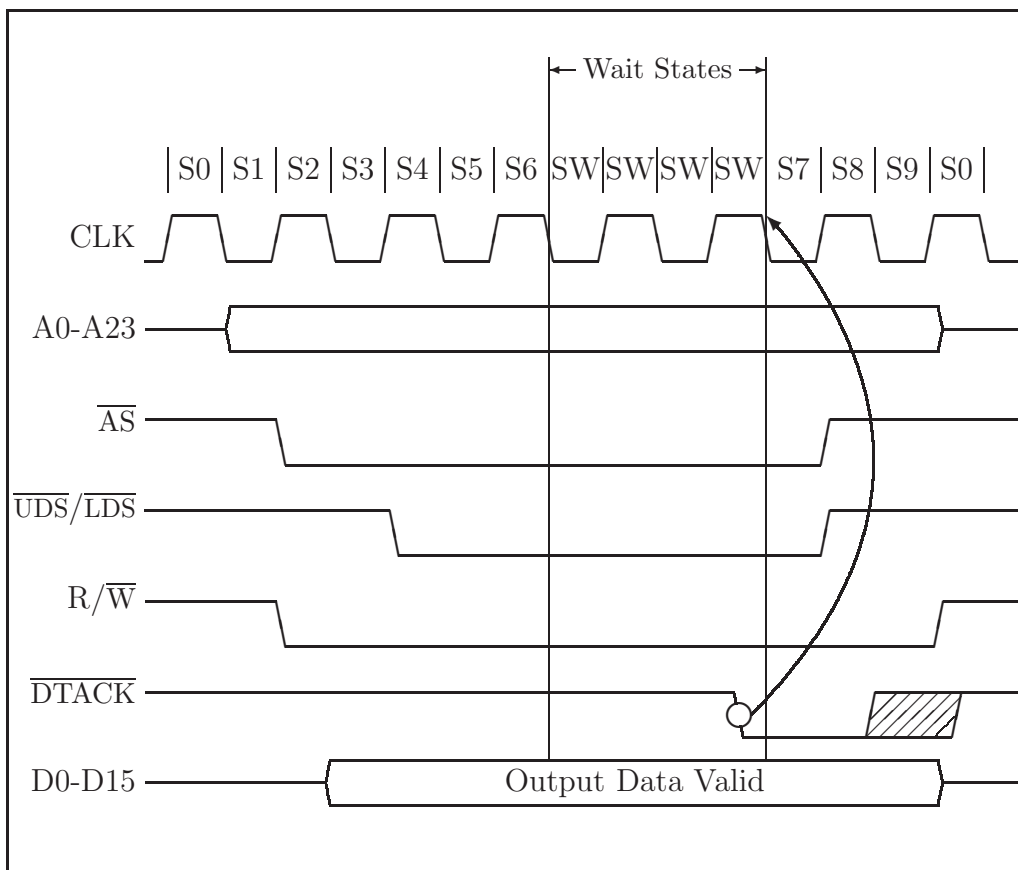


Abbildung 1.4: „circuit switched“ Datentransfer

Aufgabe 1.2 Diskutiere: write Daten von Datenbus D0-D15 in die Speicherzelle mit der auf A0-A23 anliegenden Adresse.

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:

- *circuit switched*: Der Bus blockiert, bis die anstehende Operation ausgeführt ist: *wait states*

Alternative:

- *packet switched*: Jeder Datentransfer wird in kleine Pakete zerlegt. Diese werden nacheinander nichtblockierend auf den Bus geschickt. Dadurch kann eine weitere CPU (ein weiterer Thread) schon wieder Daten transferieren, obwohl die erste CPU ihren Transfer noch nicht abgeschlossen hat.

Bemerkung: Speicherinhalte werden je nach „Datentyp“ unterschiedlich interpretiert

Numerische Datentypen:

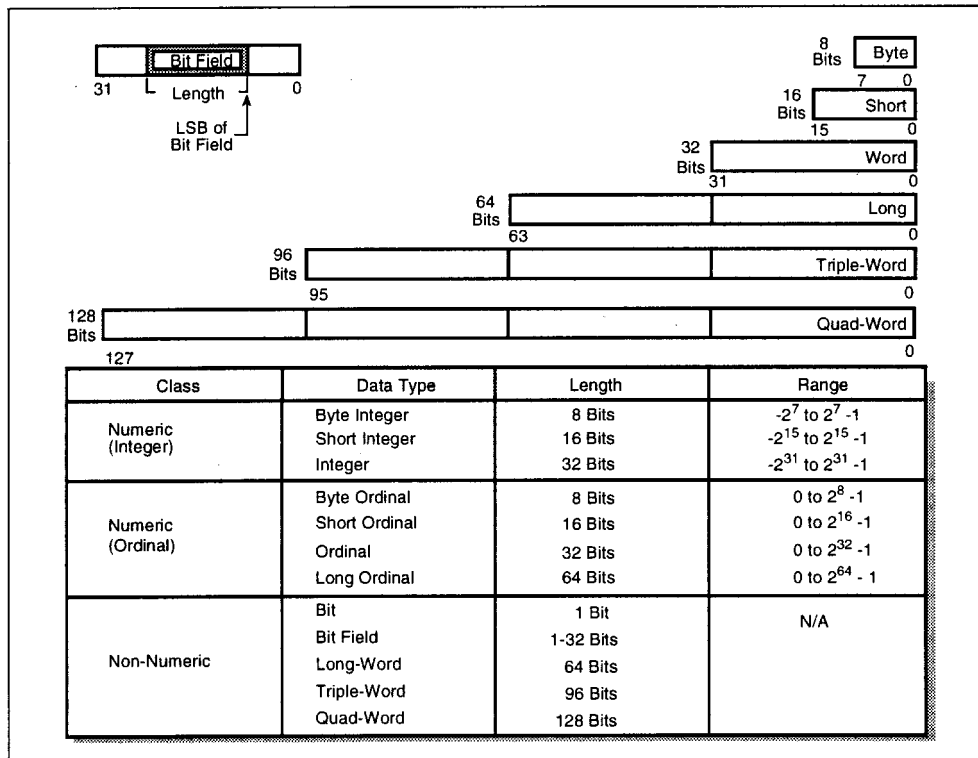


Abbildung 1.5: Datentypen

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)¹.

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|-----|-----|-----|-----|--------------|-----|-----|-----|-----|--------------|
| 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 | ⌋ | ⌌ | ⌍ | Σ | |
| 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 | £ | | | | | ∞ | ⁿ |
| 13 | ↔ | - | = | M |] | m | } | ì | | í | | | | φ | ² |
| 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

¹In 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 | Ò | â | ò |
| 2 | 18 | 34 | 50 | 66 | 81 | 98 | 114 | 130 | 146 | 162 | 178 | 194 | 210 | 226 | 242 |
| | | # | 3 | C | S | c | s | | £ | ³ | Ã | Ó | ã | ó | |
| 3 | 19 | 35 | 51 | 67 | 82 | 99 | 115 | 131 | 147 | 163 | 179 | 195 | 211 | 227 | 243 |
| | | \$ | 4 | D | T | d | t | | | ´ | Ä | Ö | ä | ö | |
| 4 | 20 | 36 | 52 | 68 | 83 | 100 | 116 | 132 | 148 | 164 | 180 | 196 | 212 | 228 | 244 |
| | | % | 5 | E | U | e | u | | | µ | Å | Ö | å | õ | |
| 5 | 21 | 37 | 53 | 69 | 84 | 101 | 117 | 133 | 149 | 165 | 181 | 197 | 213 | 229 | 245 |
| | | & | 6 | F | V | f | v | | | ¶ | Æ | Ö | æ | ö | |
| 6 | 22 | 38 | 54 | 70 | 85 | 102 | 118 | 134 | 150 | 166 | 182 | 198 | 214 | 230 | 246 |
| BEL | | / | 7 | G | W | g | w | | § | · | Ç | × | ç | ÷ | |
| 7 | 23 | 39 | 55 | 71 | 86 | 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 | 87 | 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 | 88 | 105 | 121 | 137 | 153 | 169 | 185 | 201 | 217 | 233 | 249 |
| LF | | * | : | J | Z | j | z | | ª | º | Ê | Û | ê | ú | |
| 10 | 26 | 42 | 58 | 74 | 89 | 106 | 122 | 138 | 154 | 170 | 186 | 202 | 218 | 234 | 250 |
| VT | ESC | + | ; | K | [| k | { | | « | » | Ë | Ü | ë | û | |
| 11 | 27 | 43 | 59 | 75 | 90 | 107 | 123 | 139 | 155 | 171 | 187 | 203 | 219 | 235 | 251 |
| FF | | , | < | L | \ | l | | | ¬ | ¼ | Ï | Û | ì | ü | |
| 12 | 28 | 44 | 60 | 76 | 91 | 108 | 124 | 140 | 156 | 172 | 188 | 204 | 220 | 236 | 252 |
| CR | | - | = | M |] | m | } | | - | ½ | Í | Ý | í | ý | |
| 13 | 29 | 45 | 61 | 77 | 92 | 109 | 125 | 141 | 157 | 173 | 189 | 205 | 221 | 237 | 253 |
| SO | | . | > | N | ^ | n | ~ | | | ¾ | Î | | î | | |
| 14 | 30 | 46 | 62 | 78 | 93 | 110 | 126 | 142 | 158 | 174 | 190 | 206 | 222 | 238 | 254 |
| SI | | / | ? | O | - | o | | | - | ¿ | Ï | ß | ï | ÿ | |
| 15 | 31 | 47 | 63 | 79 | 94 | 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).

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 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 |

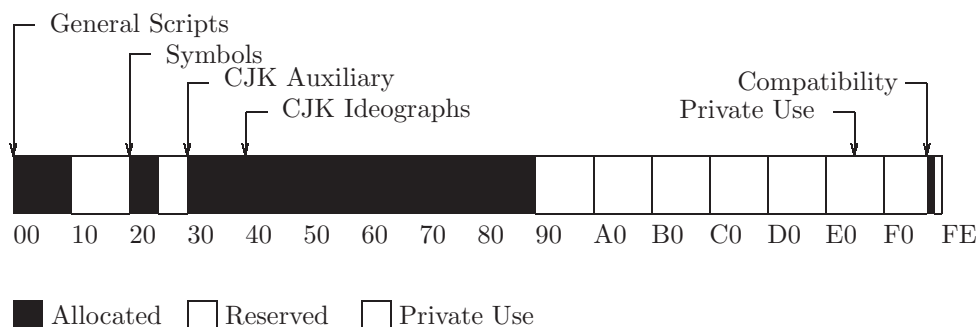
Tabelle 1.7: ISO-8859 Latin 1 (ECMA-94 Latin 1) Zeichensatz

| 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

Eine weltweite Anwendbarkeit eines Zeichencodes kann erreicht werden, falls 16 Bit für die Codierung zur Verfügung stehen: *UNICODE*, der in neuen Programmiersprachen wie etwa *JAVA* schon benutzt wird.

Statistik der 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% | | | |

Mit noch über 30000 unbenutzten Character Positionen besitzt der UNICODE auch für die Zukunft noch genug Raum für weitere Expansionen.

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

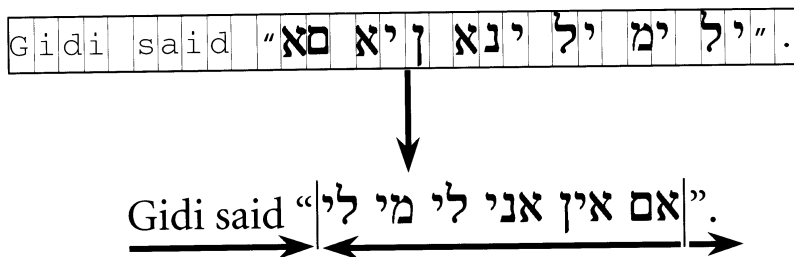


Abbildung 1.7: Bidirectional Ordering

General Scripts

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

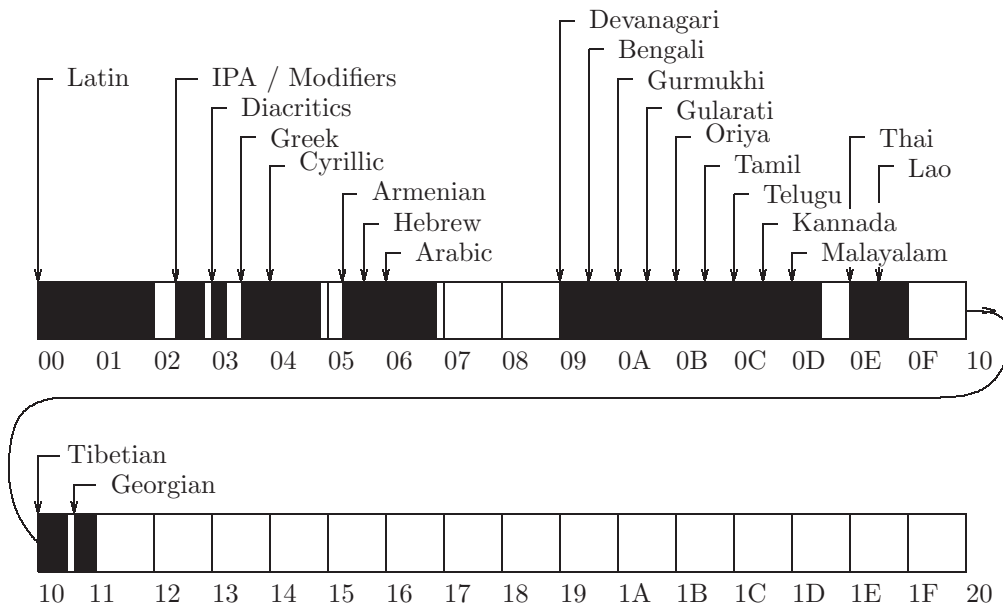


Abbildung 1.8: General Scripts

| 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 | © | ˙ | 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 |

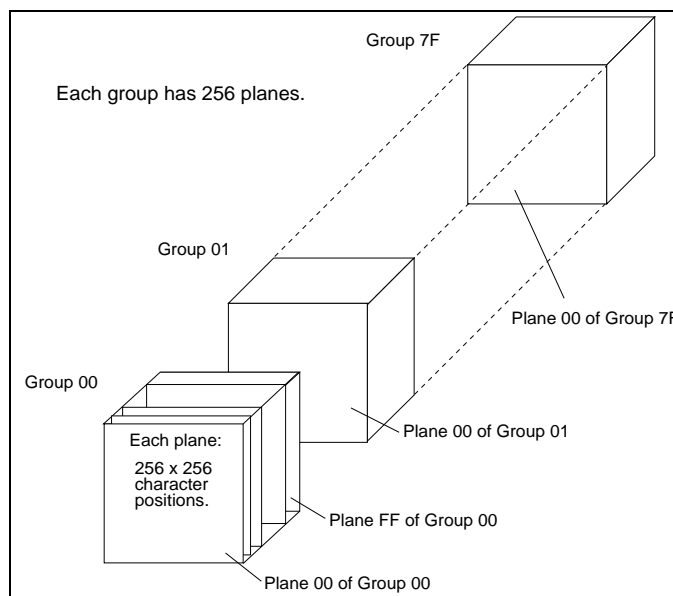
Tabelle 1.9: UNICODE Version 1.0, Character Blocks 0000-00FF

| | |
|------------|---|
| 2000..206F | Zeichen für die Zeichensetzung: ,,:;“„, ... |
| 2070..209F | Subscripts und Superscripts: ^{2,3,4} , ... |
| 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://dns.hti.umich.edu/htistaff/pubs/1997/janete.01/>,
<http://www.indigo.ie/egt/standards/iso10646/bmp-today-table.html>,
<http://wwwinfo.cern.ch/asdoc/WWW/publications/ictp99/ictp99N2705.html>

und

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

nach. Die nun zur Verfügung stehenden 4294967296 Codes dürften voraussichtlich für eine Codierung auch der ausgefallensten (ausgestorbenen) Schriften ausreichen.

| 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

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

Im Inneren der CPU

In einer CPU wurden zunächst

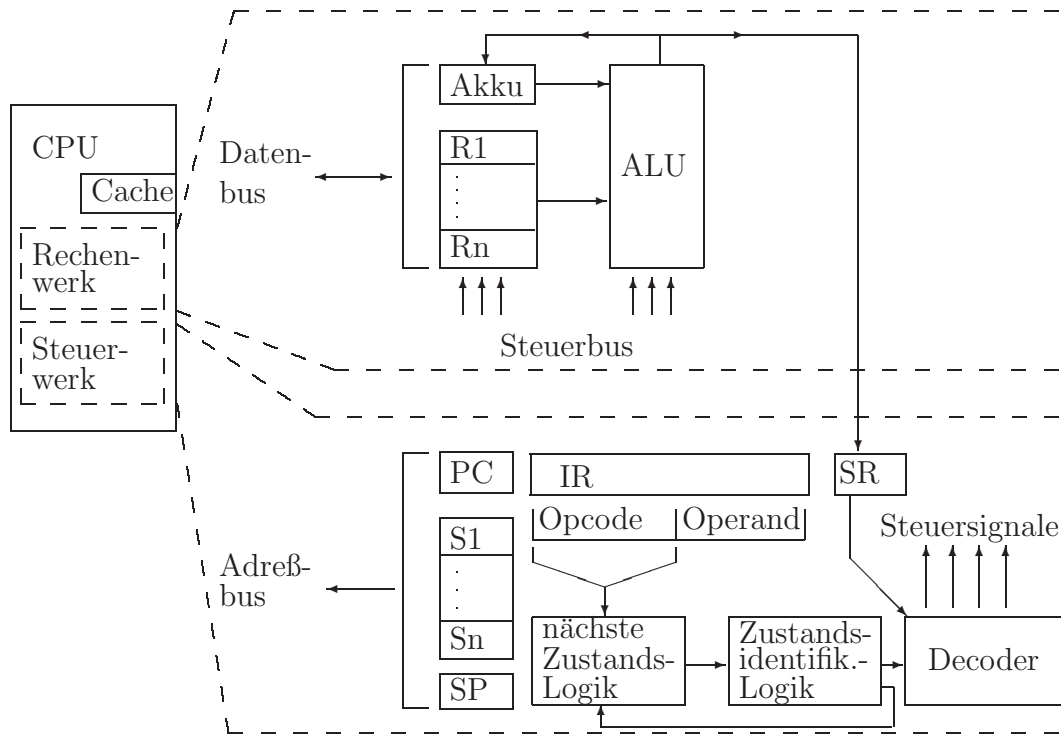


Abbildung 1.11: CPU

die einzelnen Aktionen hardwaremäßig (1 aus N Auswahl der gewünschten Aktion der ALU über CPU-interne Steuerleitungen) realisiert.

1.2 Modularer Aufbau von Betriebssystemen

Betriebssysteme haben die folgenden Aufgaben:

- Auftragsverwaltung (Jobs)
- Betriebsmittelverwaltung:
 - Hauptspeicherverwaltung
 - Datenverwaltung
 - E/A-Steuerung
 - Ablaufsteuerung
 - Zugriffskontrolle
 - Ausnahmebehandlung
 - ...

Sie setzen sich dabei modular aus Subsystemen zusammen:

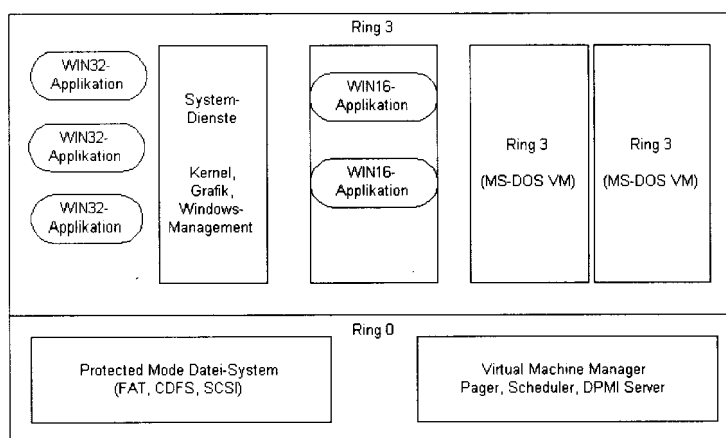


Abbildung 1.12: Windows 95/98

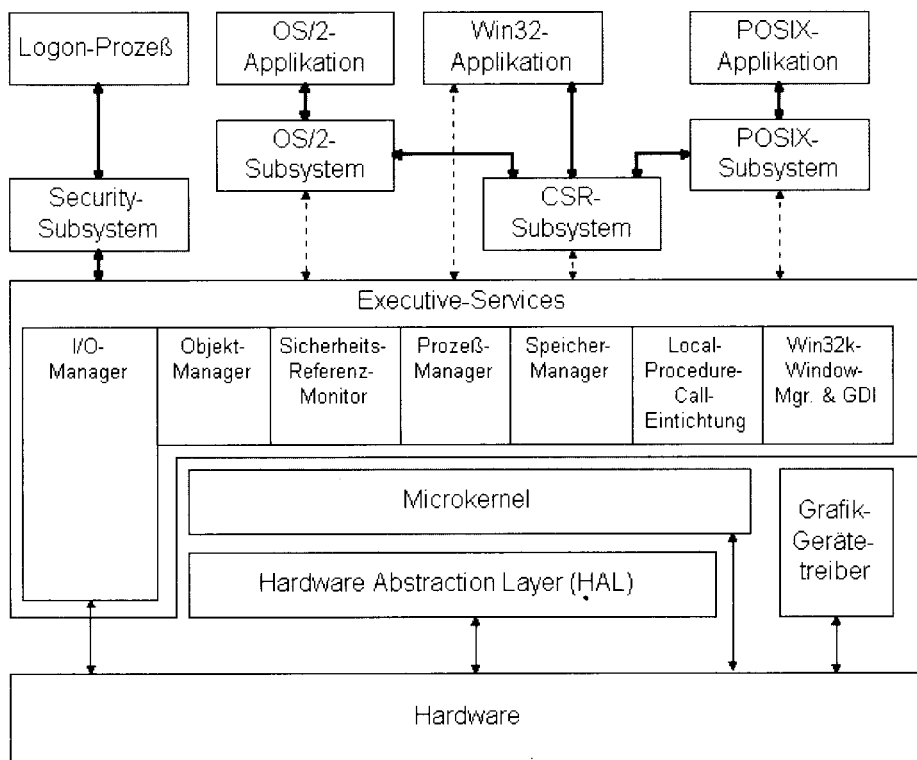


Abbildung 1.13: Windows NT

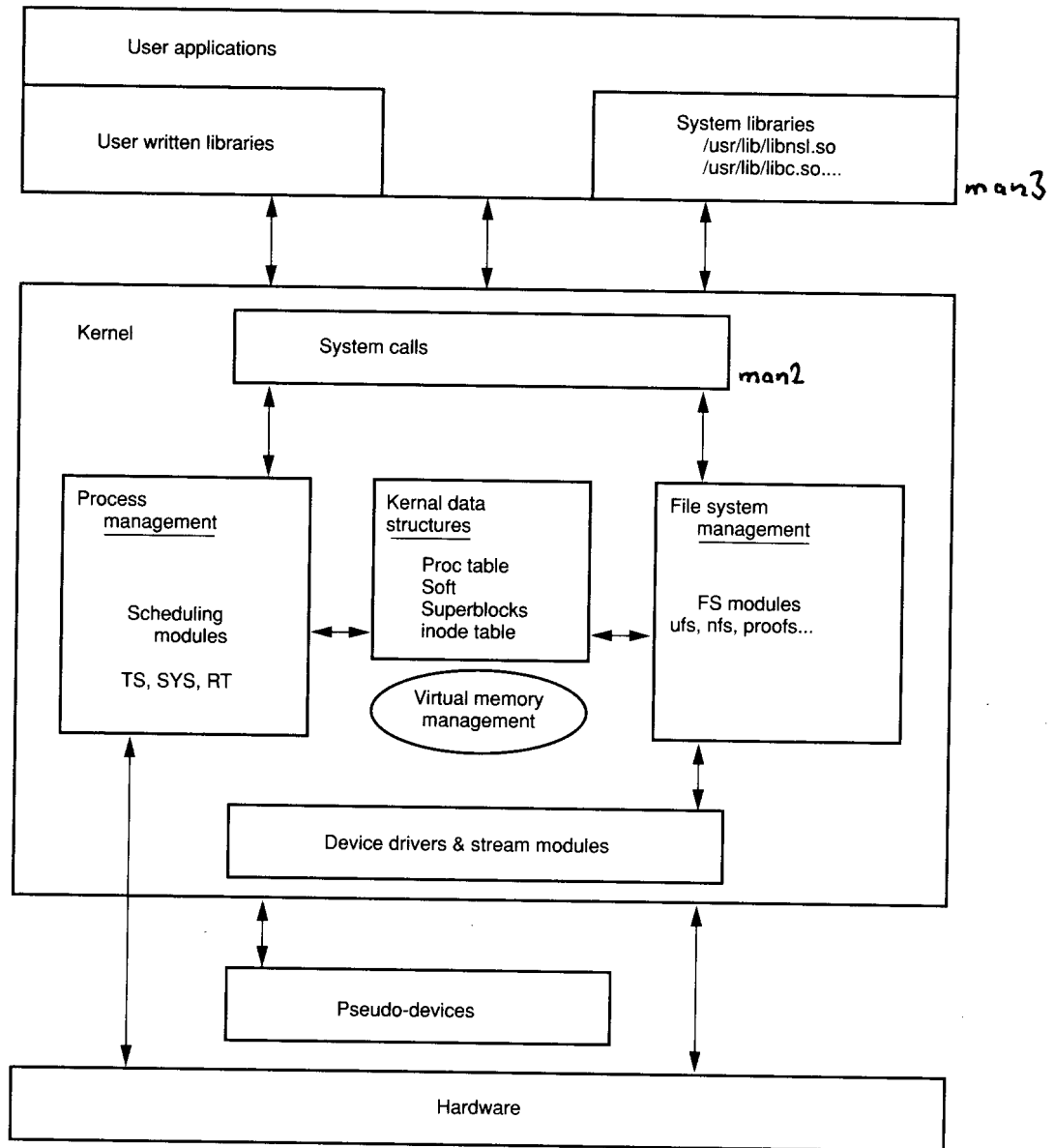


Abbildung 1.14: Solaris/Linux

In der folgenden Skizze

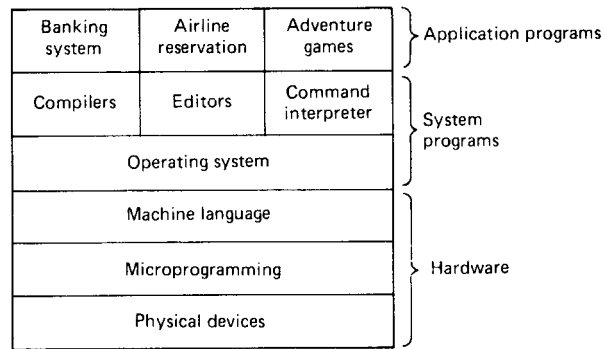


Abbildung 1.15: Microcode und Betriebssystem

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

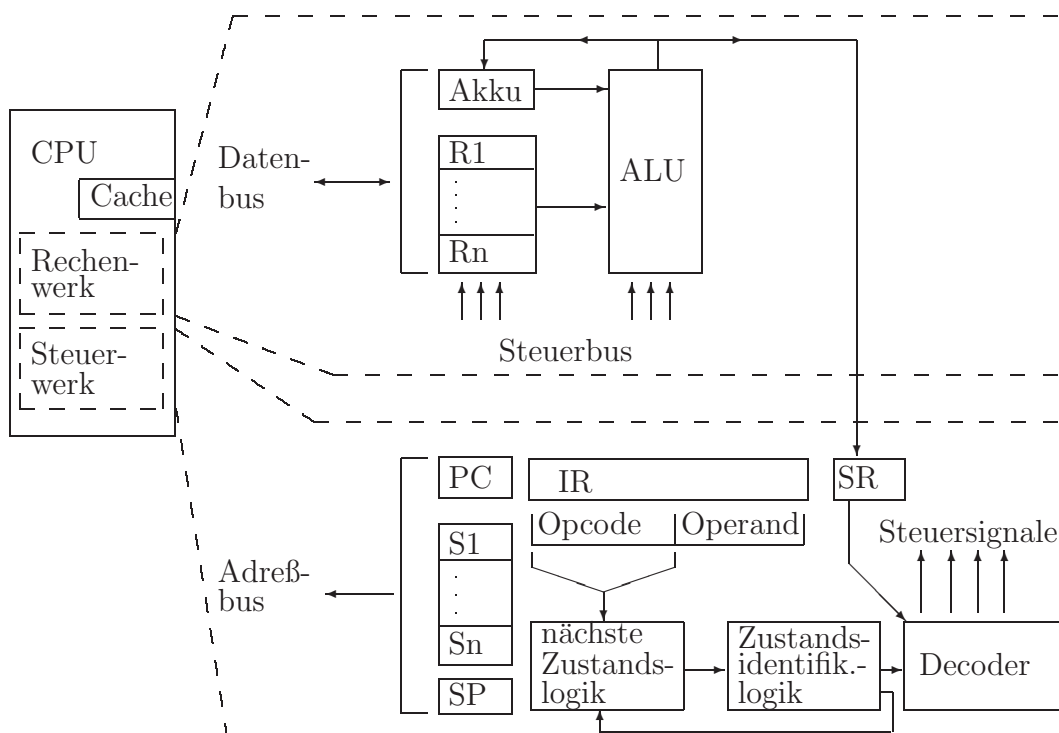
1.3 CISC/RISC

1.3.1 CISC

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

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

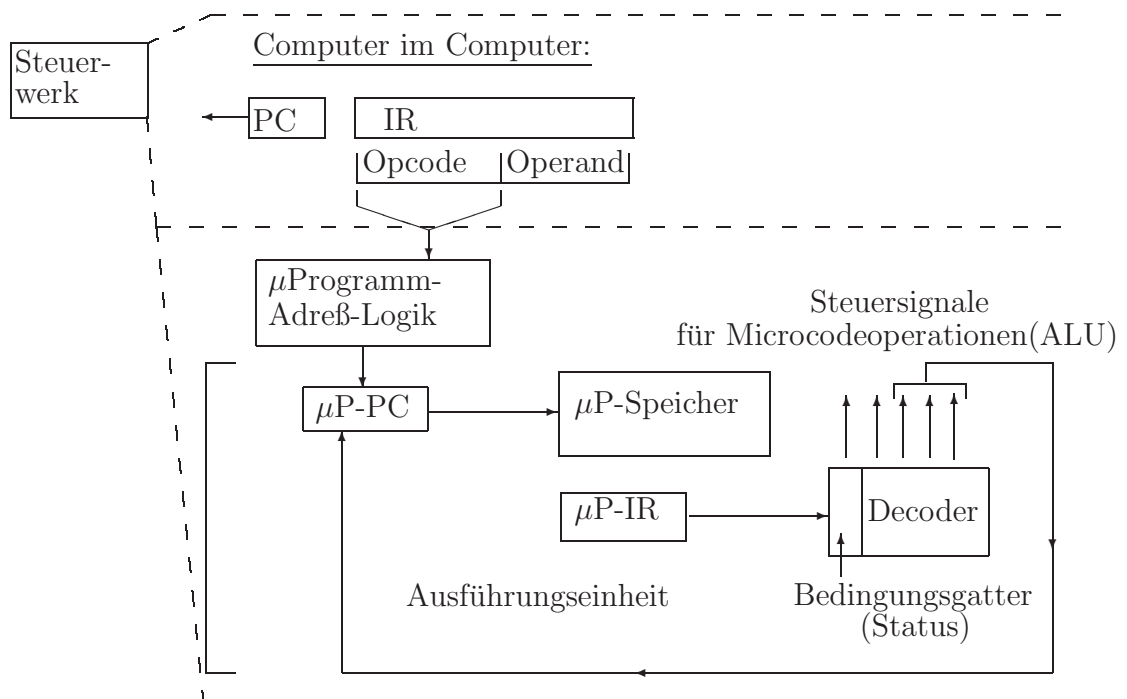
Abbildung 1.16: CISC-CPU-Aufbau



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 Schaltungen auf einem Chip realisierbar, weshalb man das Rechenwerk der CPU wiederum als Computer aufbaute:

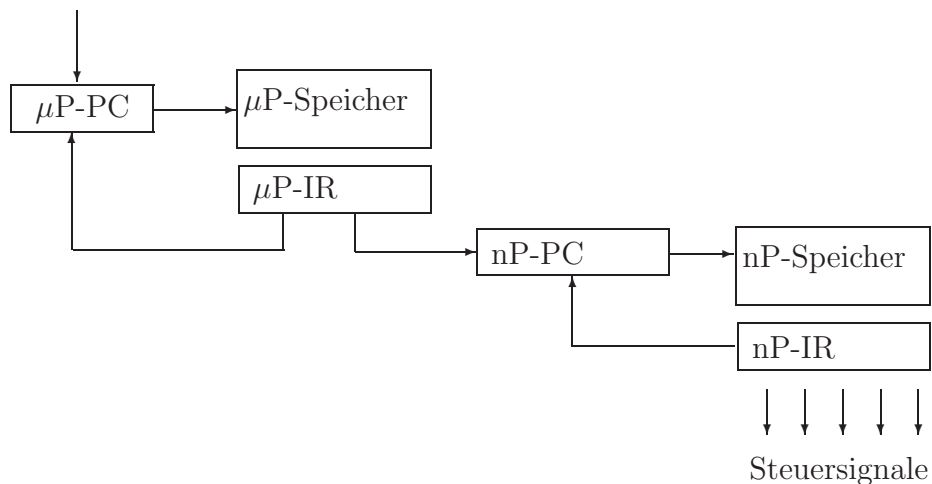
Abbildung 1.17: Microprogrammierte CISC-CPU



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 1.18: Nanoprogrammierte CISC-CPU



Das CISC-Designprinzip:

Reduziere die „semantic gap“ zwischen Maschinsprache 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.

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 ¹⁾ | 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 ¹⁾ 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 1.19: 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 1.20: 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-men men-men | reg- reg reg-men men-men | stack | reg-reg | reg-reg | reg-reg |

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:

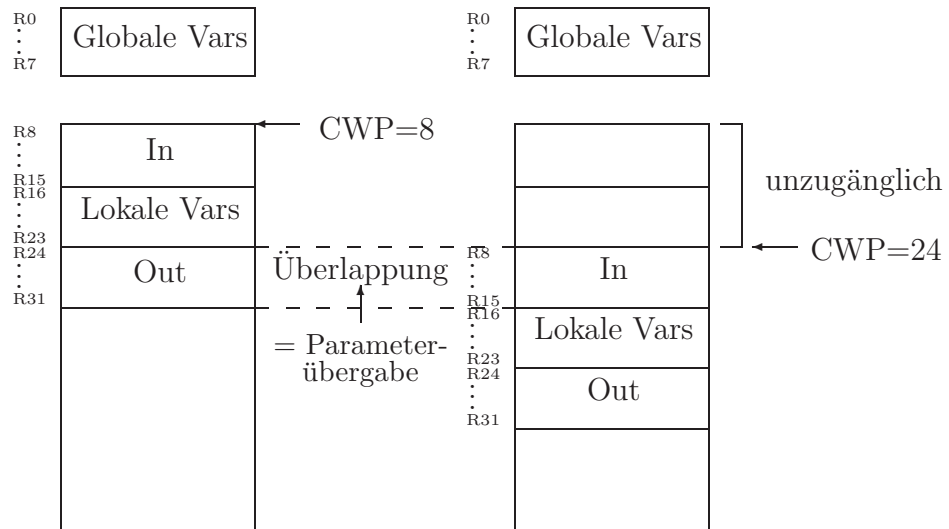
– ~ 500 Register, von denen 32 gleichzeitig verfügbar sind

– überlappende Register-Fenster (s.u.)

– Register-Mehrfachbenutzung („Lebenszeit“ von Variablen)

Der Registerstack ist auch bei CISC möglich, wenn genügend Chip-Platz verfügbar ist.

Abbildung 1.21: Registerstack am Beispiel SPARC



1.3.4 Historischer CISC-RISC CPU-Vergleich

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 | |

Zum Vergleich:

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

Tabelle 1.18: 64 Bit RISC-CPU's

| | 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 1.19: PC-Bussysteme und Durchsatzraten

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

Tabelle 1.20: RISC PC-CPU's: 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 | | |

Tabelle 1.21: 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 |

Tabelle 1.22: CISC PC-CPU's: 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>, . . .

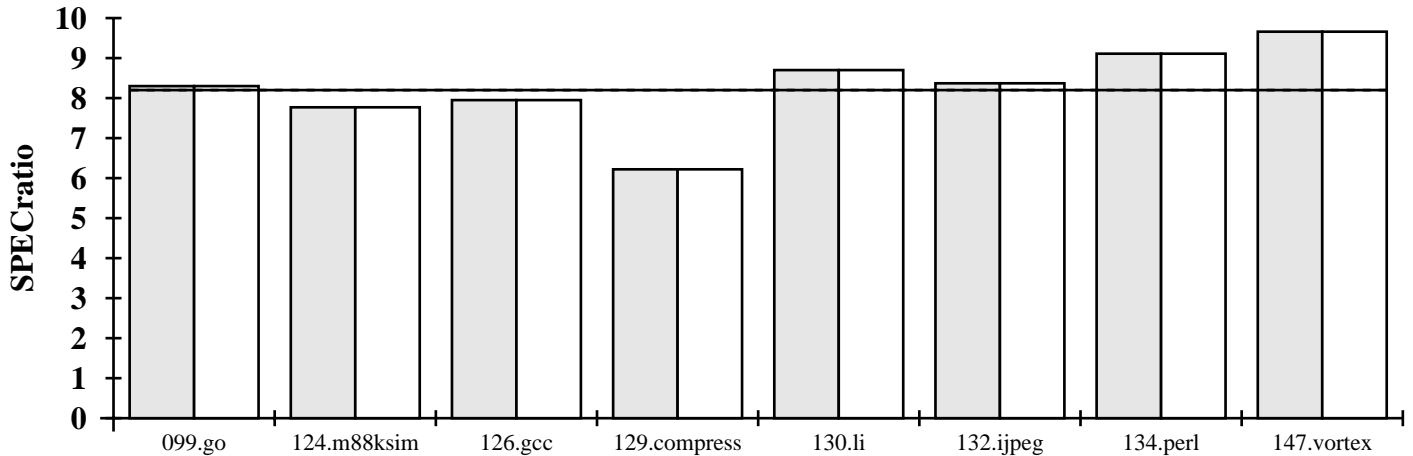
SPEC CINT95 Results

©Copyright 1995, Standard Performance Evaluation Corporation

Dell Computer Corporation
Dell Dimension XPS Pro200n

SPECint95 = 8.20
SPECint_base95 = 8.20

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

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

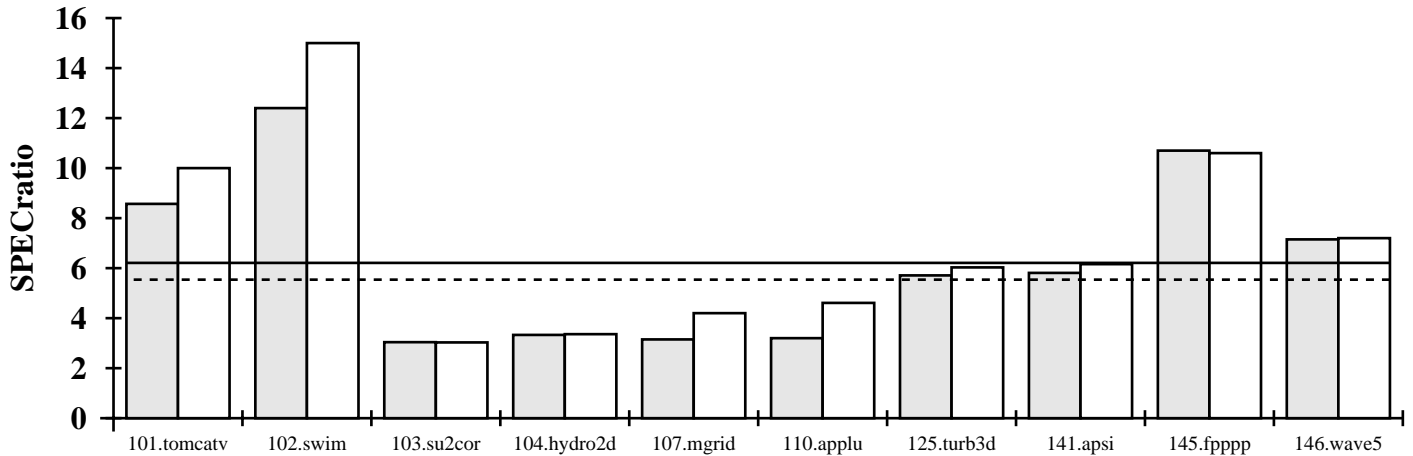
SPEC CFP95 Results

©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 |
|--|---|-------------------------|----------------|--------------------|-----------------|----------|------------|
| Hardware | | 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 | | | |
| Software | | | | 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
    
```

48

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: --

Tabelle 1.23: Marktallianzen im RISC-Bereich

Das Sparc-Lager (unter anderem): SPARC

- Sun Microsystems
- Amdahl
- Cray
- Fujitsu
- ICL
- Solbourne
- Toshiba
- Xerox
- Matsushita
- Tadpole Technology
- Goldstar
- Hyundai
- Tatung
- Force Computers
- LSI Logic
- Tulip Computers
- Meiko

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

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

Das Motorola-Lager (88-Open unter anderem): 88000 (siehe auch Power-PC)

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

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

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

Das Digital-Lager: Alpha

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

Das Mips-Lager (unter anderem): MIPS

- 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

Intel: (i860)

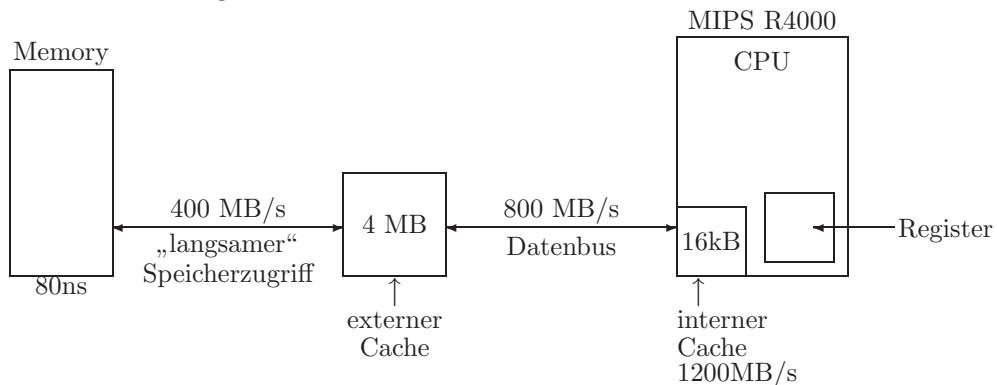
- PentiumPro (nur im Kern ?!)

Das Power-PC-Lager: POWER-PC

- 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

1.4 Der Cache zur Datentransferbeschleunigung

Abbildung 1.22: 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), ...

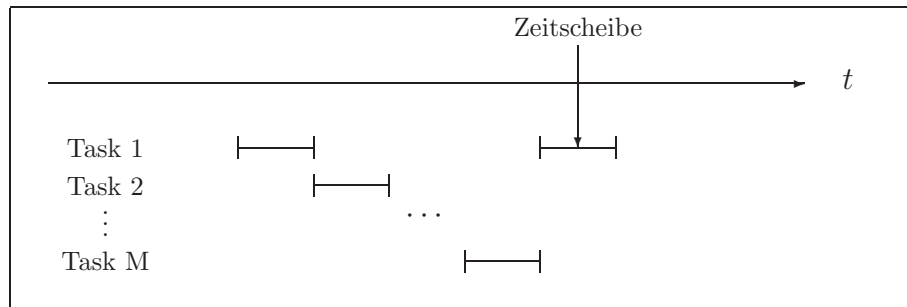
Bemerkung: Es wurden SparcStation10's ohne SuperCache ausgeliefert, um bei speziellen Anwenderprogrammen das „verlangsamende“ Cache-Füllen zu umgehen.

1.4.1 Multitasking

Pagefaults führen dann zu keinen unnötigen wait-Zyklen, wenn in einem Multitaskingsystem andere Aufträge (Prozesse) auf die CPU-Benutzung warten.

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.



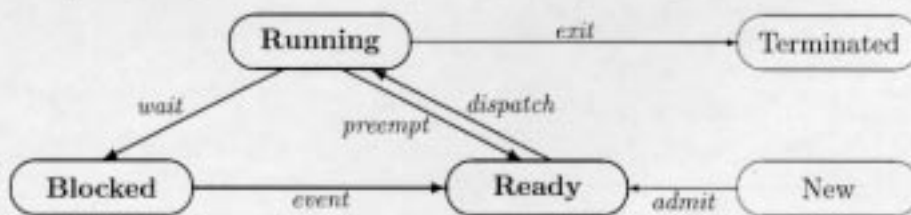
Beispiele:

Windows 3.x: Multitasking, nicht preemptiv.

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

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“ unterbrochen und später wieder fortgesetzt.

1. **Running** besitzt gerade die CPU-Kontrolle
2. **Ready / Runnable** lauffähig, aber gerade angehalten, weil ein anderer Prozess die CPU besitzt
3. **Blocked / Waiting** nicht lauffähig, wartet auf ein Ereignis von außen (I/O, Signal)



Zu den einzelnen Übergängen im Diagramm:

| | |
|-----------------|---|
| <i>admit</i> | der neu geschaffene Prozess wird eingeordnet |
| <i>wait</i> | der Prozess ruft eine Systemroutine (meist I/O) auf |
| <i>event</i> | erwartete Eingabe/erwartetes Signal trifft ein |
| <i>preempt</i> | ein anderer Prozess erhält die CPU (Scheduler-Entscheidung, s.u.) |
| <i>dispatch</i> | der Prozess erhält die CPU (Scheduler-Entscheidung, s.u.) |
| <i>exit</i> | der Prozess beendet sich selbst (ggf. auch: wird abgebrochen) |

Abbildung 1.23: Multitask-Stati

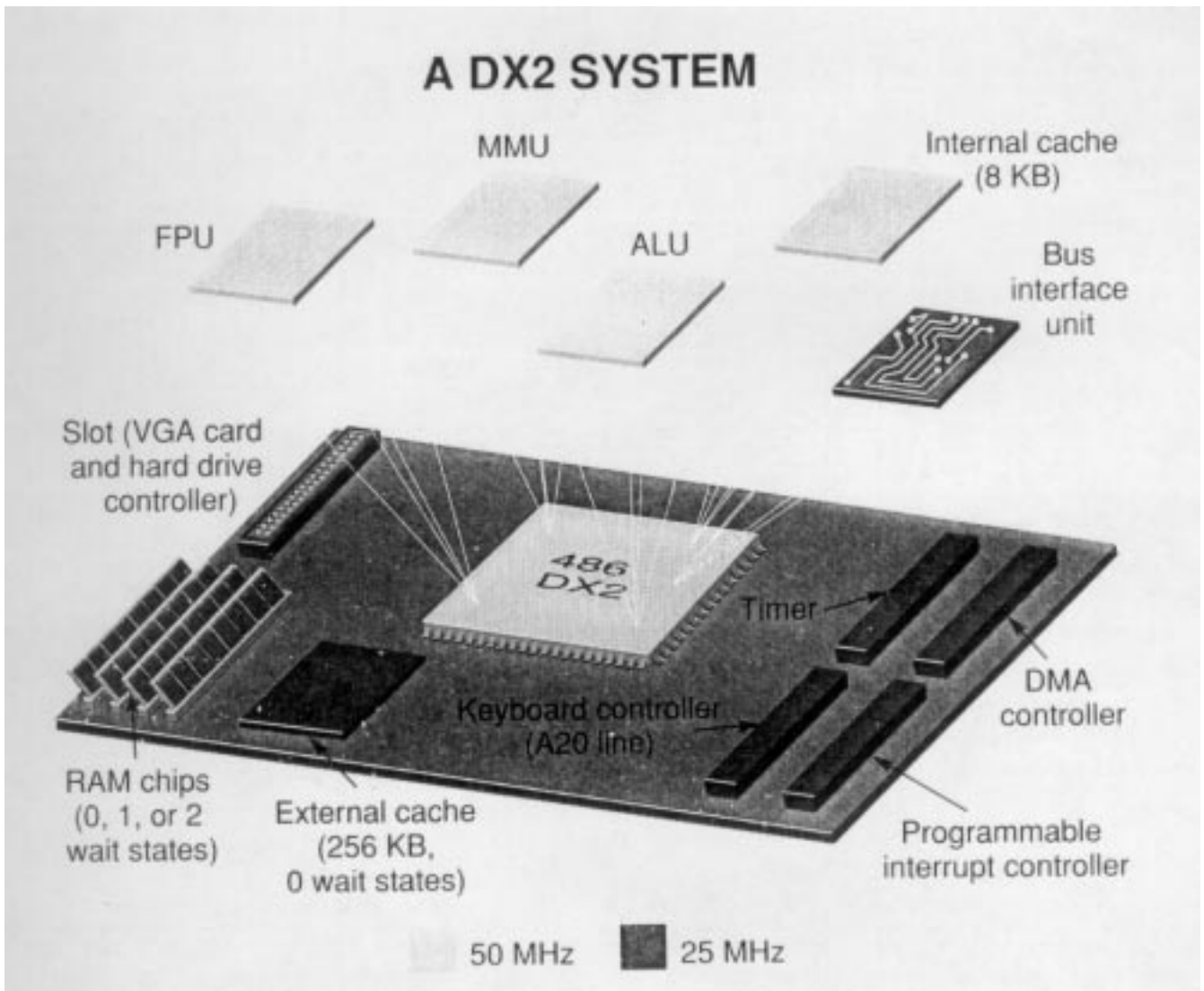


Abbildung 1.24: Cache auf dem Motherboard

1.5 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:

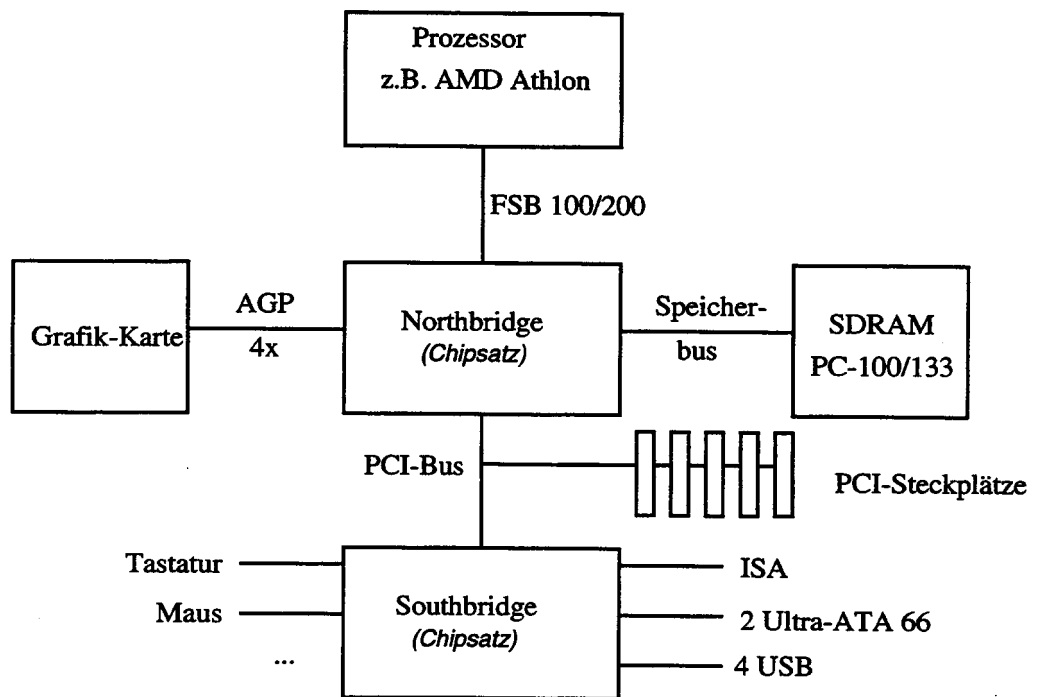


Abbildung 1.25: PC Frontsidebus

Zu genaueren aktuellen Informationen vergleiche im Web:

[Hinweise zur Architektur von Standard-PCs](#)

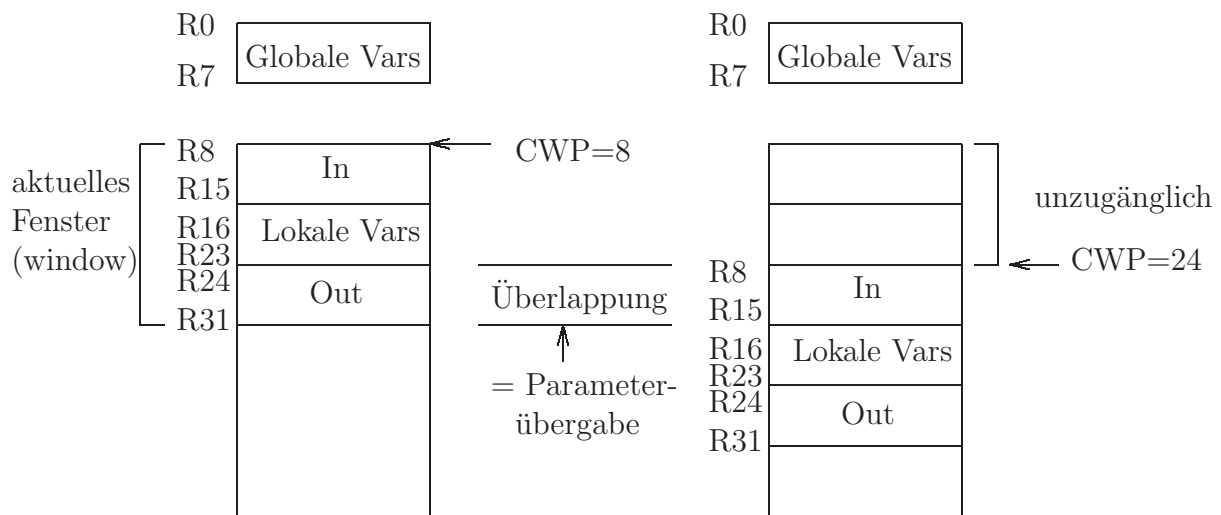
1.6 Der Weg fort von der von-Neumann-Architektur

1.6.1 SPARC

SUPER-SPARC

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

1. Registerstack



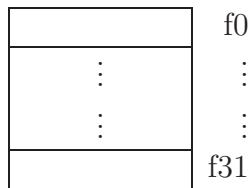
CWP steht für *current window pointer*. Die SPARCclassic besitzt z. B. acht solcher Fensterbereiche. Auf die Register greift man mit *logischen* Namen zu:

```
%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 |
| ADD _{cc} | A+B and modify icc |
| ADDX | add with carry = A + B + c |
| ⋮ | |
| SUB | subtract |
| ⋮ | |
| MUL _{Sss} | 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 $-\infty$, towards $+\infty$</i>) FiTOs integer to single FiTOd integer to double FiTOq integer to quad FsTOi FdTOi FqTOi FsTOd |

```

FsToq
:
FMOVs
FNEGs
FABSs
FSQRTs
FADDs
FADDd
FADDq
FSUBs
FMULs
FsMULd entspricht  $s \cdot s \rightarrow d$ 
FdMULq entspricht  $d \cdot d \rightarrow q$ 
FDIVs
FCMPs
FCMPEs compare and exception it unordered
:

```

Ein Beispiel-PASCAL Programm ...

```

program Tabelle(output);
  var
    i : integer ;
  function f (x : real) : real;
  const
    pi = 3.141592653589793 ;
  var
    arg : real;
  begin
    arg := 2.0 * pi * x + 5.0;
    f   := sin(arg);
  end;
begin
  writeln("f(arg)");
  writeln("-----");
  for i := 0 to 20 do
    writeln(i/10.0,f(i/10.0));
  end;
end;

```

end.

... und was daraus in SPARC-Maschinenbefehlen wird (pc -s):

```
.section ".data "  
.align 4  
.common i,4,4  
.common PROGFRAME,4,4  
.align 4  
.L15:  
.ascii "%21.14e "  
.byte 0  
.section ".text "  
.proc 010  
.global f  
  
.align 4  
f:  
!#PROLOGUE# 0  
sethi %hi(.LF2),%g1  
add %g1,%lo(.LF2),%g1  
save %sp,%g1,%sp  
!#PROLOGUE# 1  
st %i0,[%fp+0x44]  
st %i1,[%fp+0x48]  
  
.L3:  
.L4:  
.stabs "SC3.0.1 07 Nov 1994 Pascal 3.0.3 patch 101912-01 NN ",0x30,0,0xd,0x0  
.ident "SC3.0.1 07 Nov 1994 Pascal 3.0.3 patch 101912-01 "  
.stabs "Tabelle.p ",0x30,0,0x1,0x1  
.file "Tabelle.p "  
.xstabs ".stab.index ", "V=2.0 ",0x3c,0,0,0x33b3734e  
!  
arg := 2.0 * pi * x + 5.0;  
sethi %hi(.L2000000),%o0  
ldd [%o0+%lo(.L2000000)],%f0  
ld2 [%fp+0x44],%f2  
fmuld %f0,%f2,%f0  
sethi %hi(.L2000001),%o1  
ldd [%o1+%lo(.L2000001)],%f4  
faddd %f0,%f4,%f6  
std %f6,[%fp+-0x20]
```

```

!      f := sin(arg);
      ldd    [%fp+-0x20],%o0
      call   __sin,2
      nop
      std    %f0,[%fp+-0x18]
.L6:
.L7:
      ldd    [%fp+-0x18],%f0
      b      .LE2
      nop
.LE2:
      ret
      restore
      .optim "-O Q R S "
      .LF2 = -128
      .LP2 = 96
      .LST2 = 96
      .LT2 = 96
      .type f,#function
      .size f,-f
      .section ".rodata "
      .align 8
.L2000000: .word 0x401921fb,0x54442d18
      .align 8
.L2000001: .word 0x40140000,0x0
      .section ".text "
      .proc 022
      .global program

      .align 4
program:
      !#PROLOGUE# 0
      sethi  %hi(.LF1),%g1
      add    %g1,%lo(.LF1),%g1
      save   %sp,%g1,%sp
      !#PROLOGUE# 1
.L8:
.L9:
      .xstabs ".stab.index ", "Tabelle ",0x2a,0,0,0
      call   __PC0__tb_init,0
      nop

```

```

sethi    %hi(PROGFRAME),%o2
st       %fp,[%o2+%lo(PROGFRAME)]
!        writeln( "f(arg) ");
set      output,%o0
call     __PC0__UNIT,1
nop
st       %o0,[%fp+-0x8]
ld       [%fp+-0x8],%o3
ld       [%o3+0xc],%o1
mov      0xa,%o0
call     fputc,2
nop
!        writeln( "—— ");
set      output,%o0
call     __PC0__UNIT,1
nop
st       %o0,[%fp+-0x8]
ld       [%fp+-0x8],%o4
ld       [%o4+0xc],%o1
mov      0xa,%o0
call     fputc,2
nop
!        for i := 0 to 20 do
.L12:    st       %g0,[%fp+-0x10]
ld       [%fp+-0x10],%o5
sethi    %hi(i),%o7
st       %o5,[%o7+%lo(i)]
!        writeln(i/10.0, f(i/10.0));
set      output,%o0
call     __PC0__UNIT,1
nop
st       %o0,[%fp+-0x8]
ld       [%fp+-0x8],%l0
ld       [%l0+0xc],%o0
sethi    %hi(.L2000002),%l1
ldd      [%l1+%lo(.L2000002)],%f8
ld       [%fp+-0x10],%f10
fitod    %f10,%f12
fdivd    %f12,%f8,%f12
st       %f12,[%sp+.LP1]

```



```

    ld    [%sp+.LP1],%o2
    st    %f13,[%sp+.LP1]
    ld    [%sp+.LP1],%o3
    set   .L15,%o1
    call  fprintf,4
    nop
    sethi %hi(.L2000002),%l2
    ldd   [%l2+%lo(.L2000002)],%f14
    ld    [%fp+-0x10],%f16
    fitod %f16,%f18
    fdivd %f18,%f14,%f18
    st    %f18,[%sp+.LP1]
    ld    [%sp+.LP1],%o0
    st    %f19,[%sp+.LP1]
    ld    [%sp+.LP1],%o1
    call  f,2
    nop
    std   %f0,[%sp+.LP1+0x8]
    ld2   [%sp+.LP1+0x8],%o2
    ld    [%fp+-0x8],%l3
    ld    [%l3+0xc],%o0
    set   .L15,%o1
    call  fprintf,4
    nop
    ld    [%fp+-0x8],%l4
    ld    [%l4+0xc],%o1
    mov   0xa,%o0
    call  fputc,2
    nop
.L13:
    ld    [%fp+-0x10],%l5
    add   %l5,0x1,%l5
    st    %l5,[%fp+-0x10]
    ld    [%fp+-0x10],%l6
    cmp   %l6,0x14
    bg    .L16
    nop
    b     .L12
    nop
.L16:
.L14:

```

```

.L11:
.L17:
        b        .LE1
        nop
.LE1:
        ret
        restore
        .optim "'-O Q R S "
        .LF1 = -128
        .LP1 = 96
        .LST1 = 112
        .LT1 = 112
        .type    program,#function
        .size    program,-program
        .section ".rodata "
        .align 8
.L2000002: .word 0x40240000,0x0
        .section ".data "
        .align 4

```

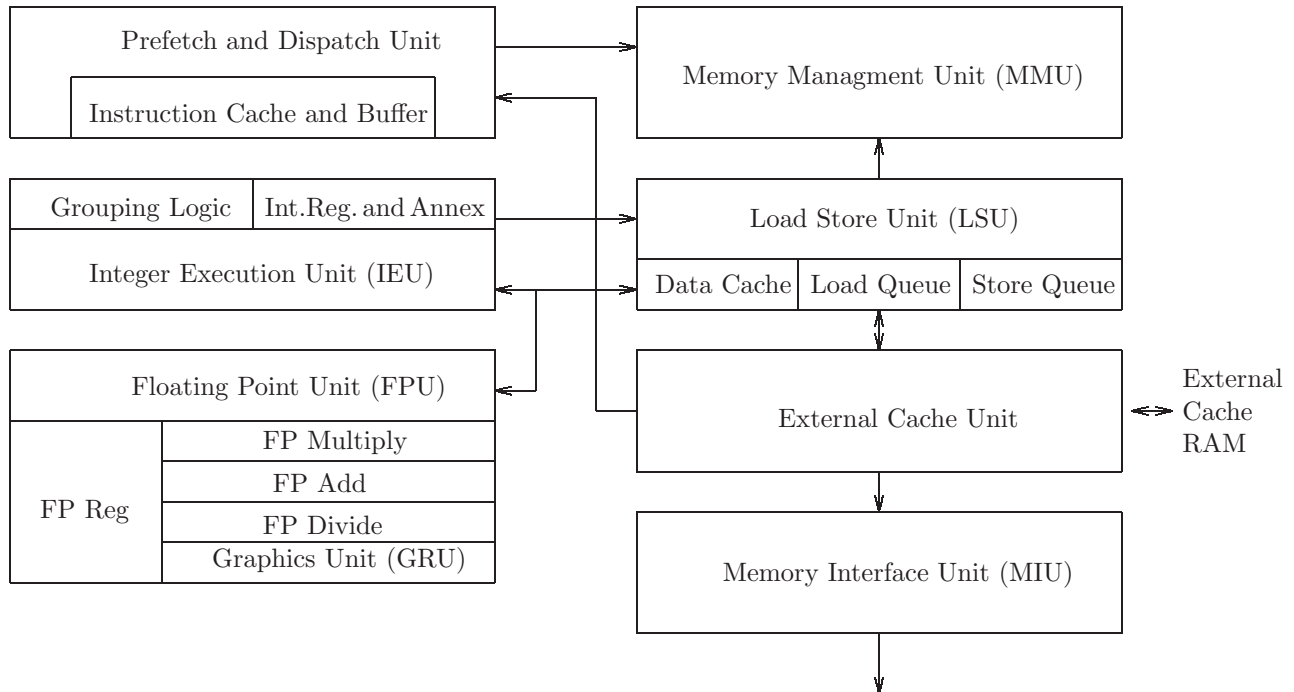
| Anweisung | Fujitsu/LSI #1,#2 | Cypress/ Fujitsu #3 | MicroSPARC (&B5000) | Fujitsu SPARClite | Super SPARC |
|-----------------------|----------------------|------------------------|------------------------|----------------------|----------------|
| # of register windows | 7 | 8 | 7 | 8 | 8 |
| ld (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 1.24: 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 |

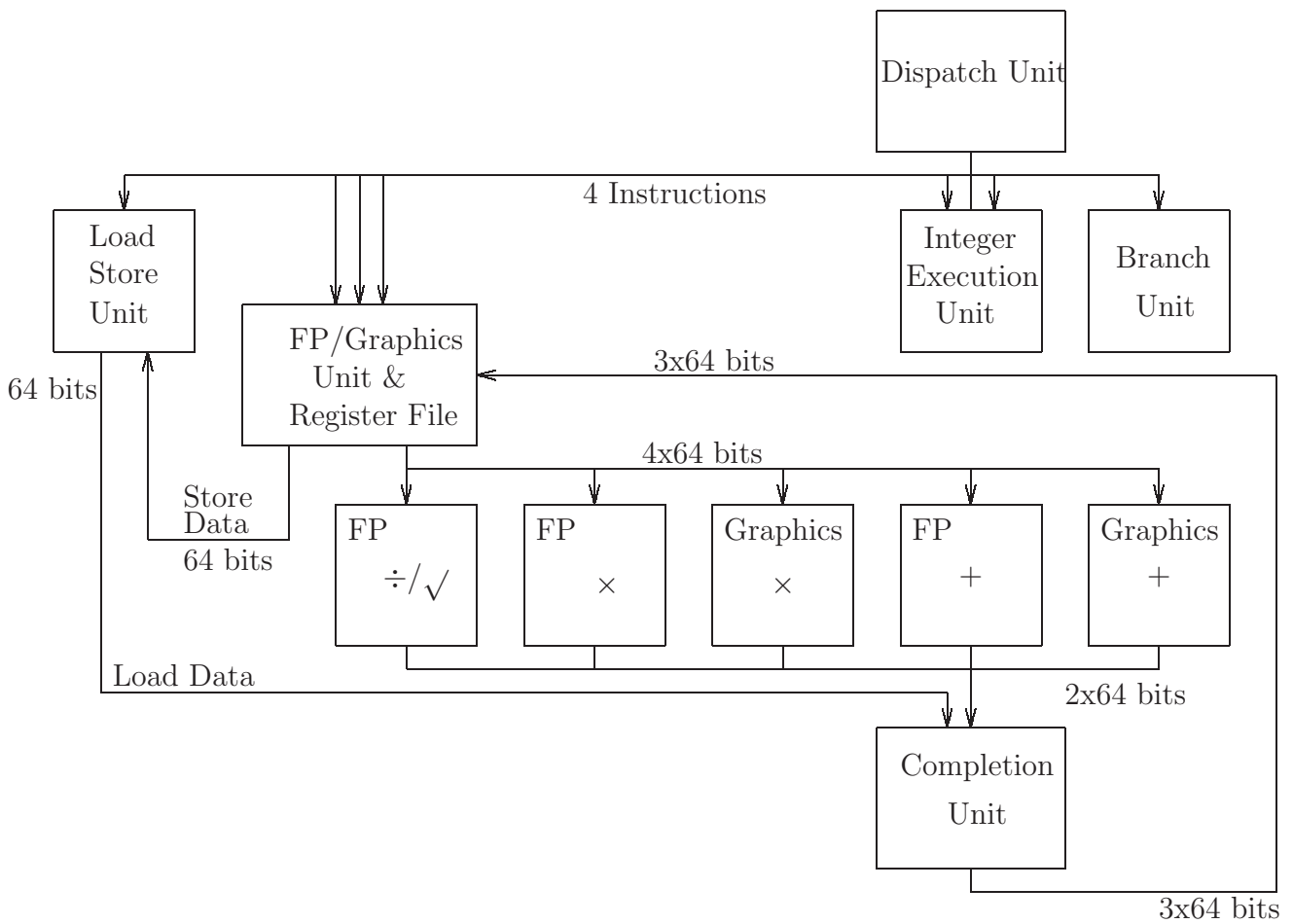
Tabelle 1.25: „floating point“-Zyklen pro Anweisung

UltraSPARCI



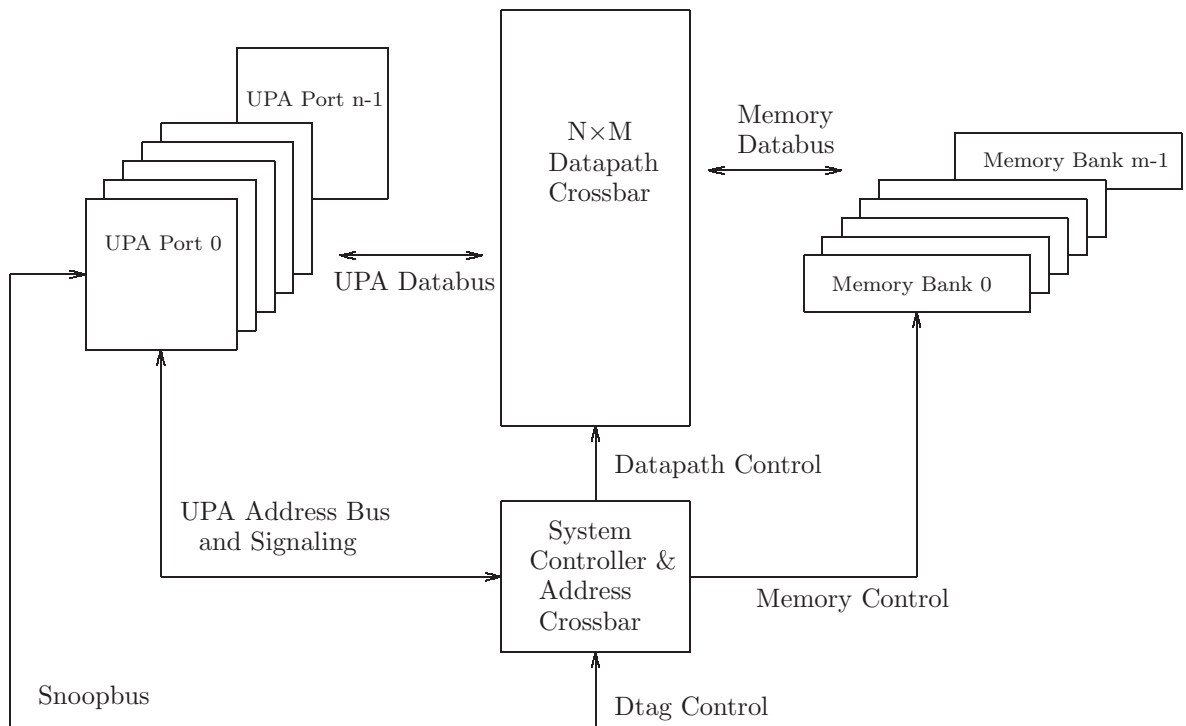
- 64Bit-Architektur (SPARC V9)
- CPU-Befehle für(VIS): 2D-, 3D-Graphik, Real-Time Video
- IU-Pipeline (9 Phasen): fetch-decode-group-execute-cache access-load miss-integer pipe wait- trap resolution-writeback-
- 2 ALU's in IEU (= interger execute unit)
- Branch prediction
- Branch following
- FP-pipeline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



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

Ultra Port Architecture



UltraSPARCI

- Bis zu 1.92 Gbyte/s Speichertransferzeit
- 4 Instruktionen pro Zyklus (Pipeline der Länge 9)
- 4 IUs, 2 FPU, 2GUs
- Real-time MPEG2 Decode
- Prefetch

Siehe auch bei: <http://www.sun.com/microelectronics/UltraSPARC-II/>

UltraSPARCI

- 9.6 GB/s Adressbus
- Bis zu 3.6 GB/s Speichertransferzeit
- 14-stufige CPU-Pipeline
- VIS-Extensions

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

1.6.2 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)

RAM-Architektur

- FP
- EDO
- SDRAM
- DDR-SDRAM
- RDRAM
- Chipsätze für RDRAM (i850), SDRAM (i845) oder SDRAM/DDR-SDRAM (M1671)

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

Siehe auch bei: <http://www.intel.com/PentiumII/home.htm> und bei <http://www.intel.com/design/PentiumII/>

| 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 ² Latency 3, Throughput 1/cycle Latency 5, Throughput 1/2 cycle ^{1,2} 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 ³ |
| 3 | Store Address Unit | Latency 3 (not applicable), Throughput 1/cycle ³ |
| 4 | Store Data Unit | Latency 1 (not applicable), Throughput 1/cycle |

NOTES:

See notes following Table 2-1.

Abbildung 1.26: PentiumII Execution Units

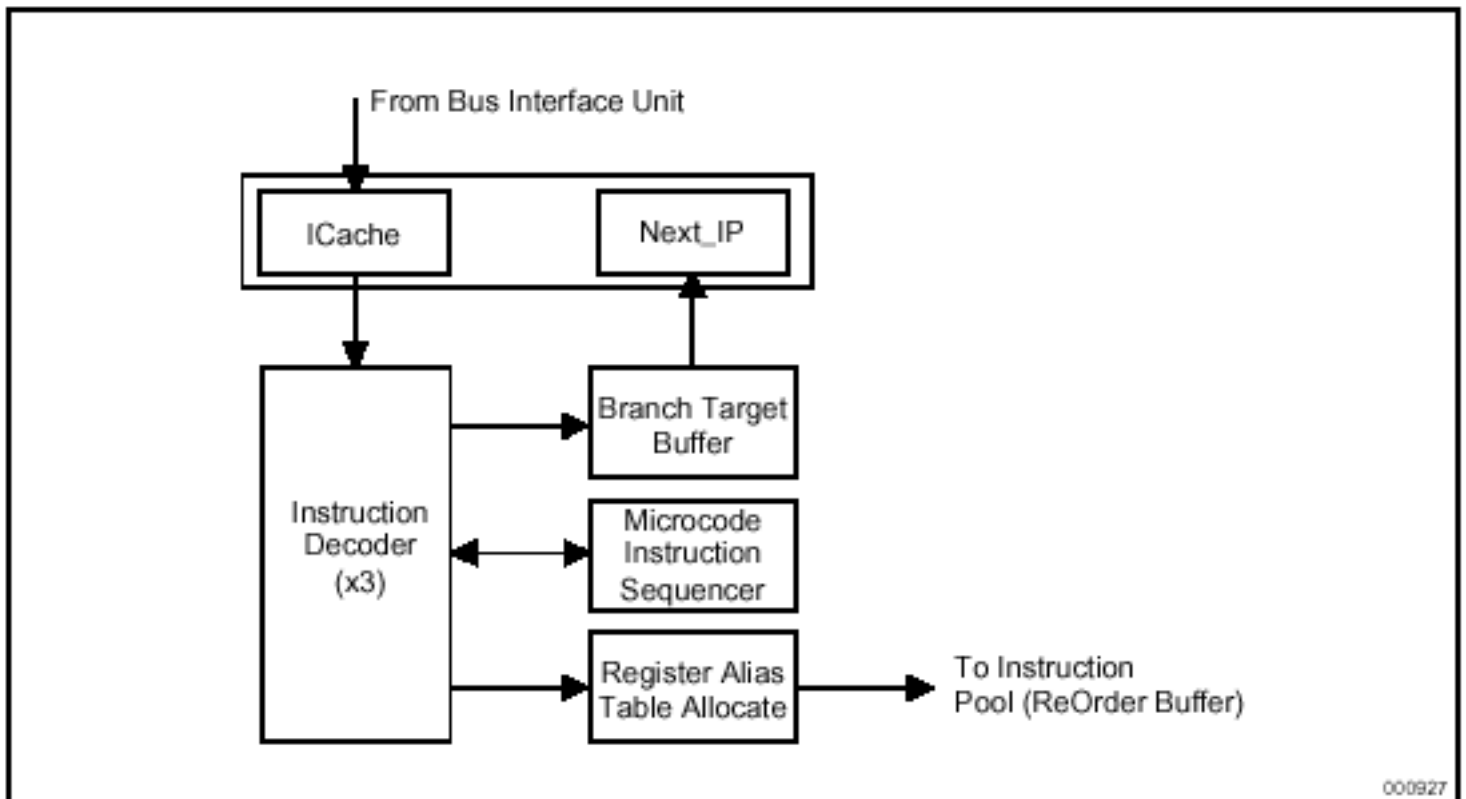


Abbildung 1.27: PentiumII-Design 1

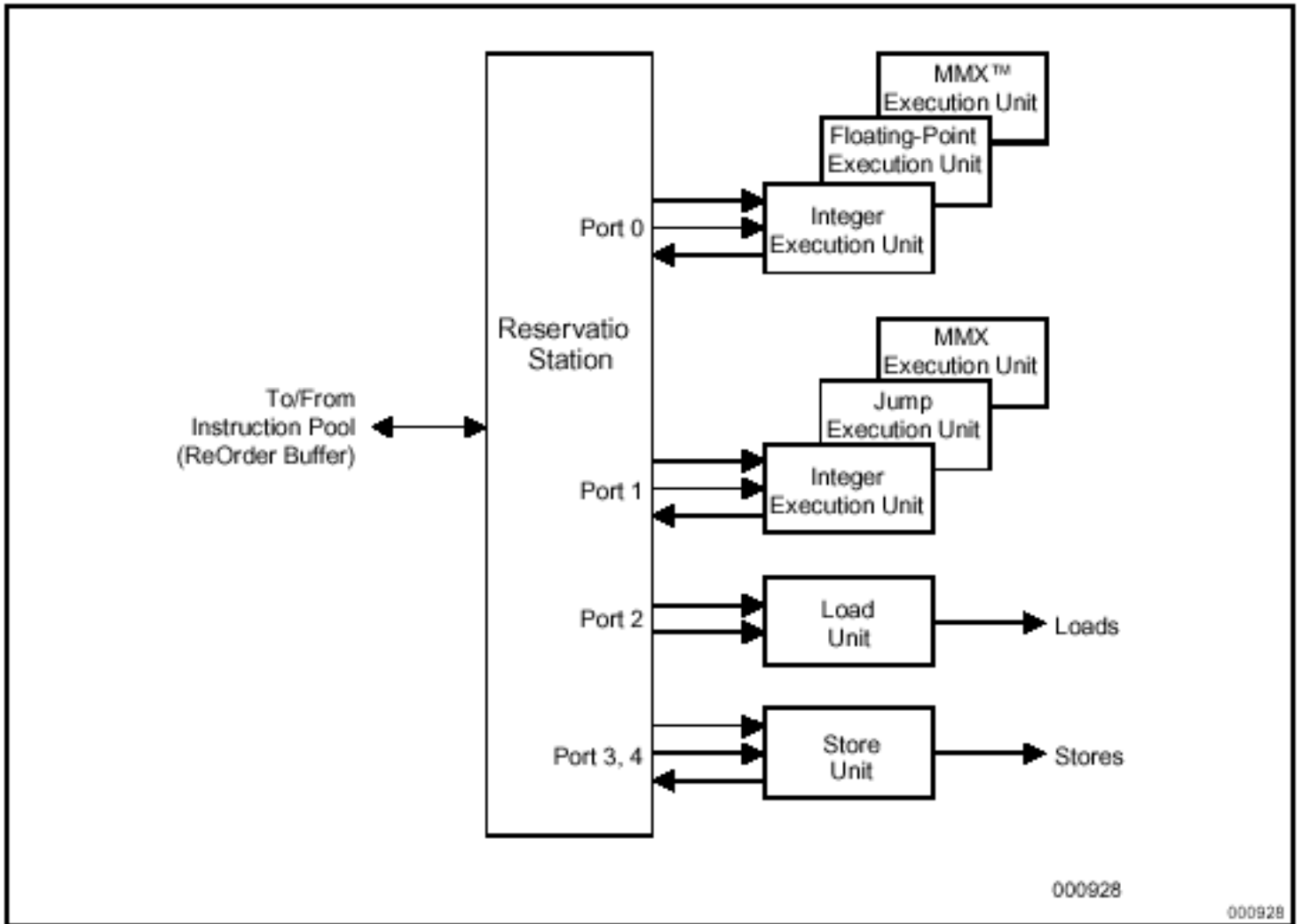


Abbildung 1.28: PentiumII-Design 2

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/design/PentiumIII/> und bei <http://www.intel.com/pentiumiii/>

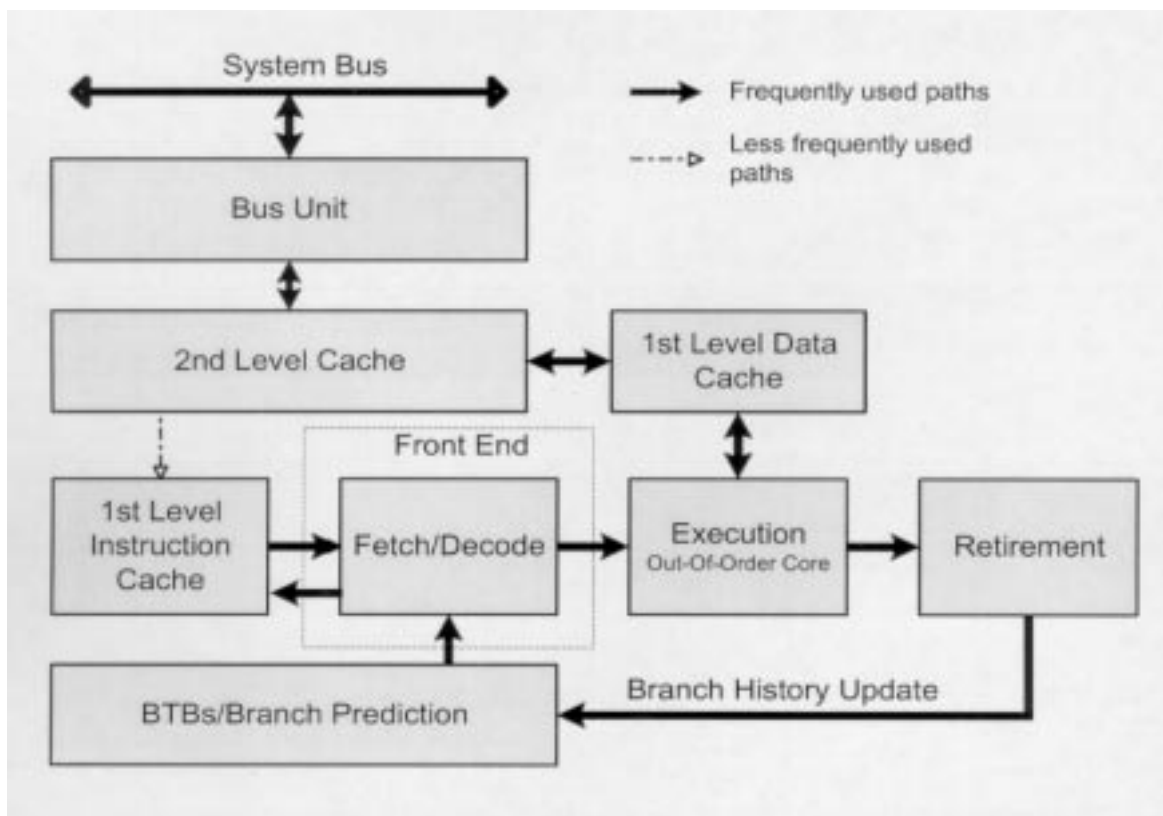


Abbildung 1.29: P6 Architektur

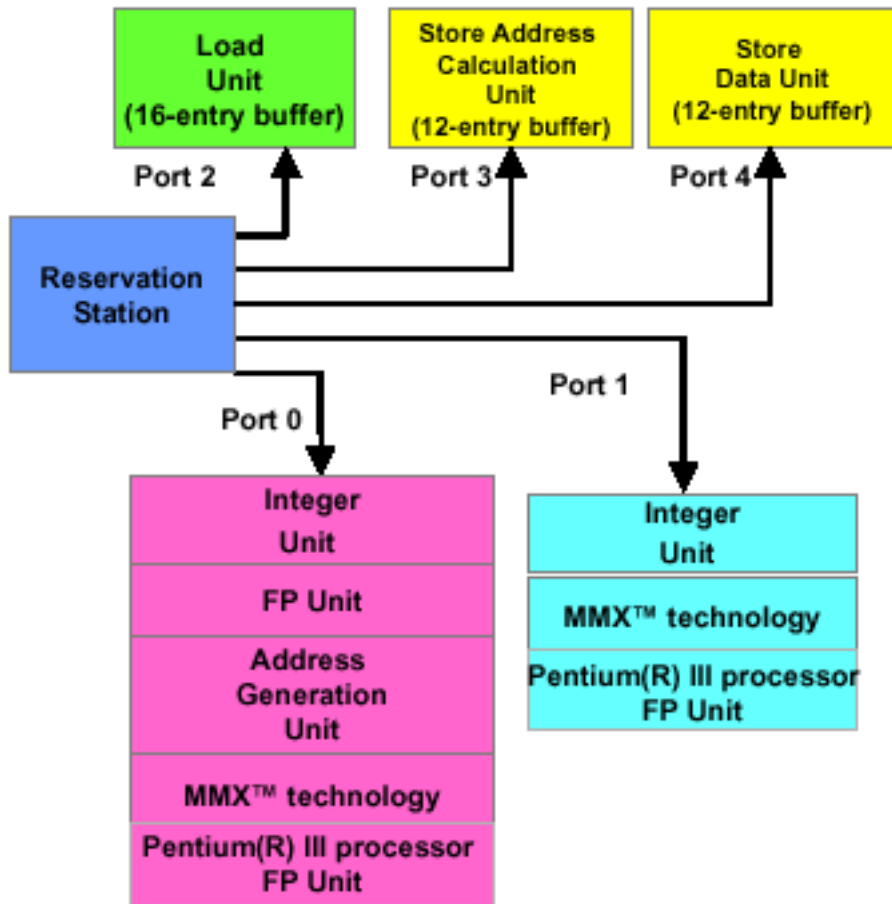


Abbildung 1.30: Dynamic Execution Microarchitecture

| Intel Processor | Date Introduced | Max. Clock Frequency at Introduction | Transistors per Die | Register Sizes ¹ | Ext. Data Bus Size ² | 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 1.31: x86-Entwicklung

| Intel Processor | Date Introduced | Micro-architecture | Clock Frequency at Introduction | Transistors per Die | Register Sizes ¹ | System Bus Bandwidth | Max. Extern. Addr. Space | On-die Caches ² |
|------------------------------------|-----------------|-----------------------------------|---------------------------------|---------------------|--|----------------------|--------------------------|--|
| Pentium III processor ³ | 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 μ op Execution Trace Cache; 8KB L1; 256KB L2 |

Abbildung 1.32: 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/design/Pentium4/> und bei <http://www.intel.com/pentium4/>

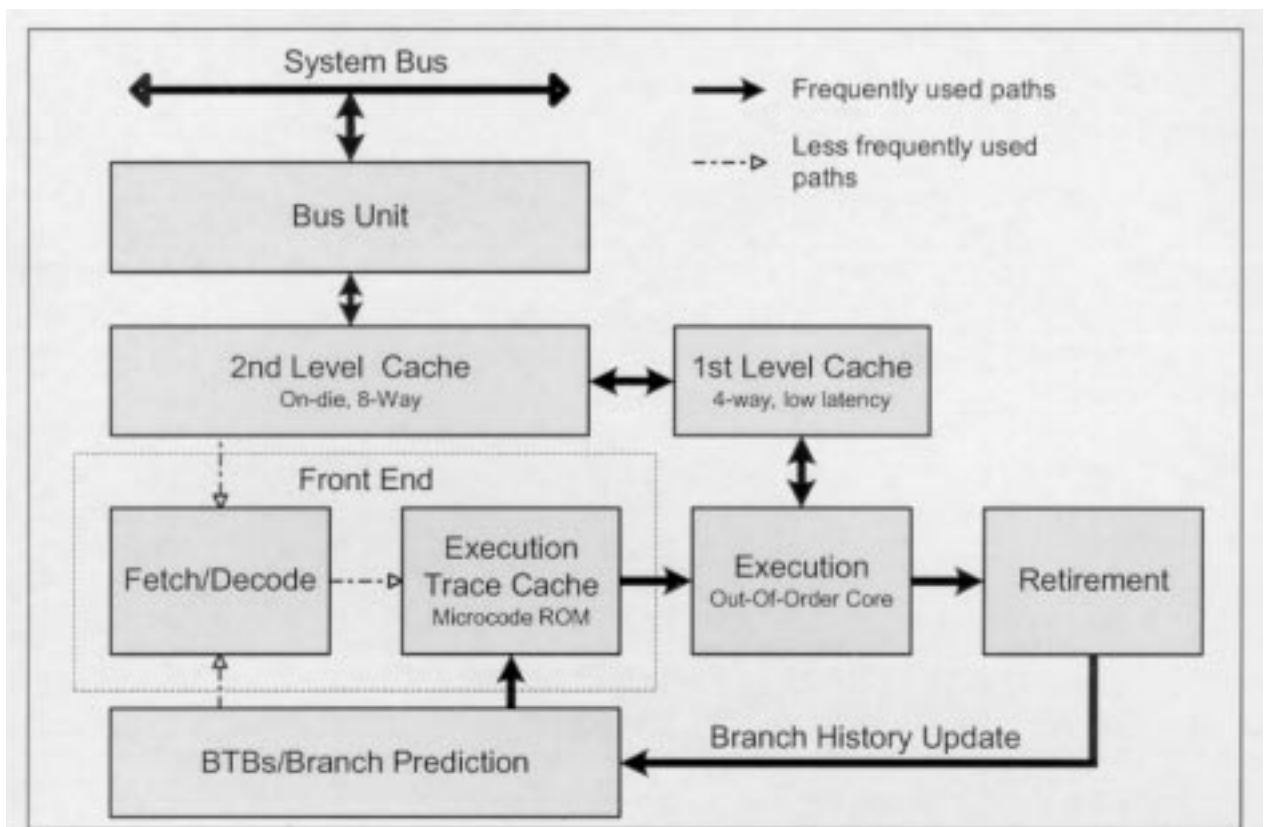


Abbildung 1.33: NetBurst Architektur

Für CPU-Leistungsangaben siehe auch bei:

http://www.intel.com/procs/perf/pentiumiii_820/productivity/specint2000.htm

http://www.intel.com/procs/perf/pentiumiii_820/3d/specfp2000.htm

<http://developer.intel.com/procs/perf/pentium4/productivity/specint2000.htm>

<http://developer.intel.com/procs/perf/pentium4/3d/specfp2000.htm>

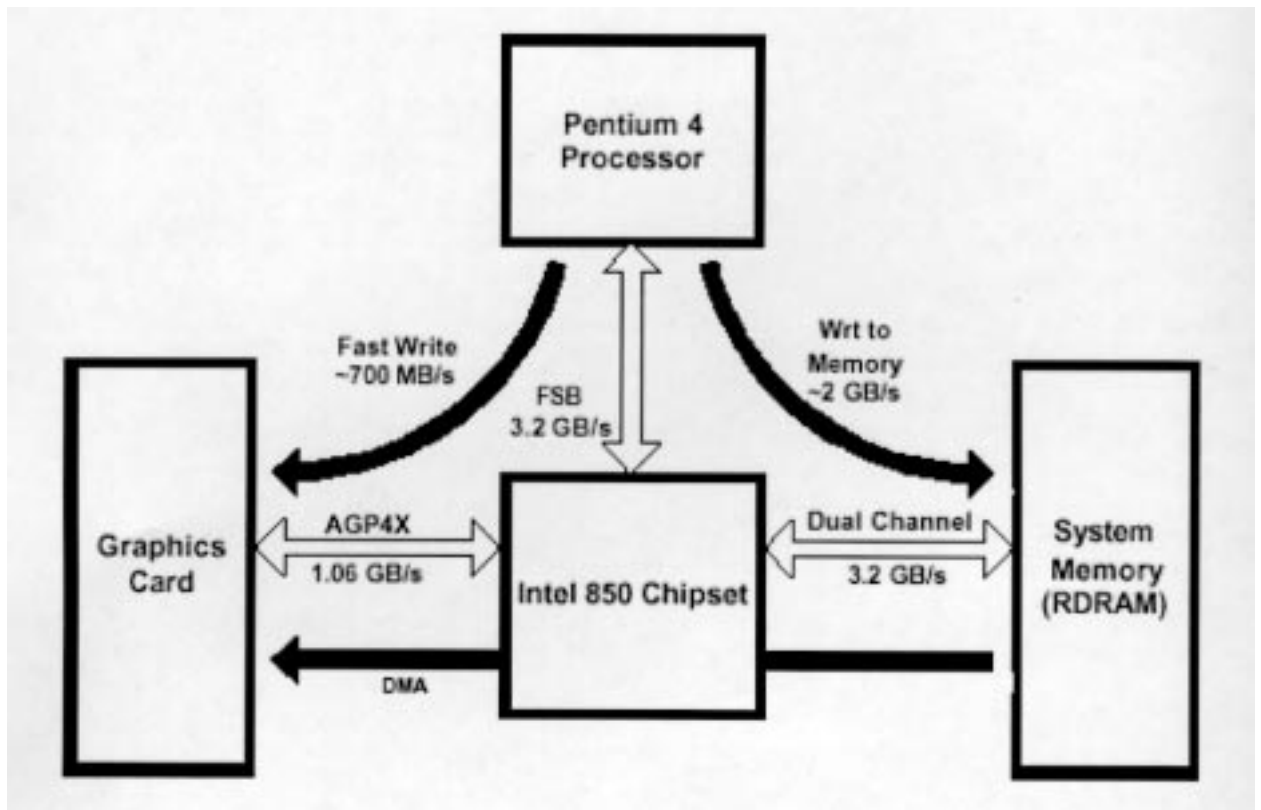


Abbildung 1.34: Datendurchsatz beim Pentium4

1.7 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

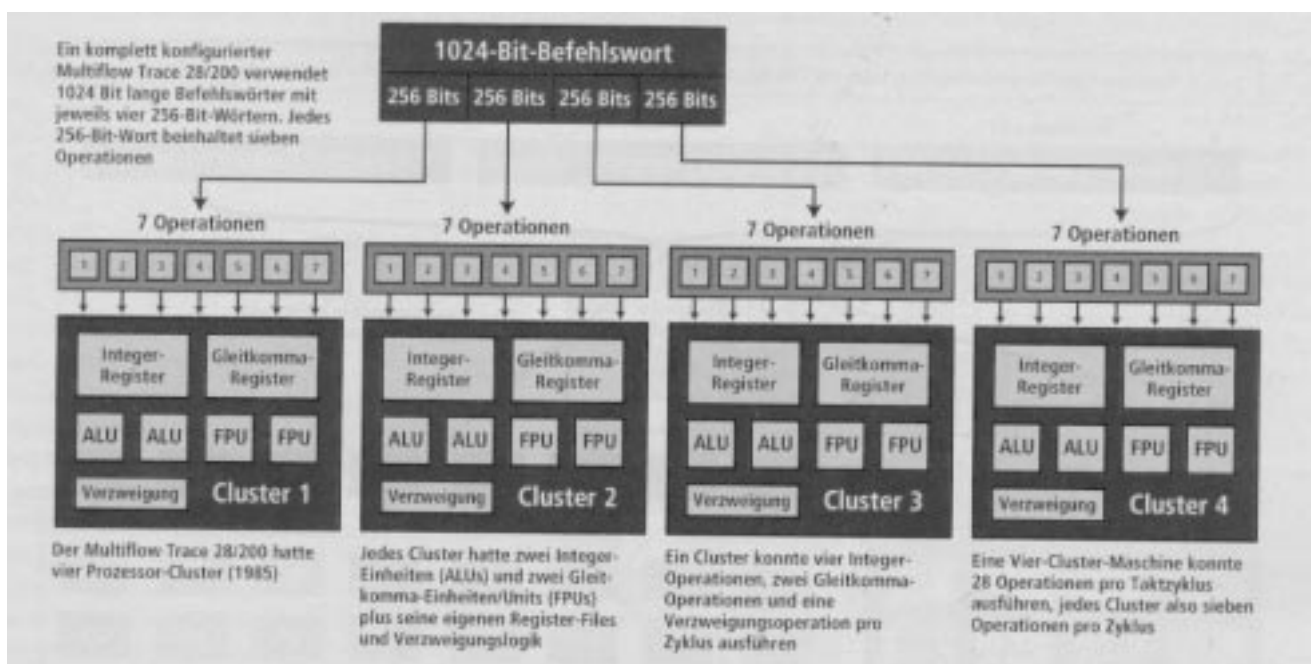


Abbildung 1.35: VLIW

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

1.7.1 Itanium

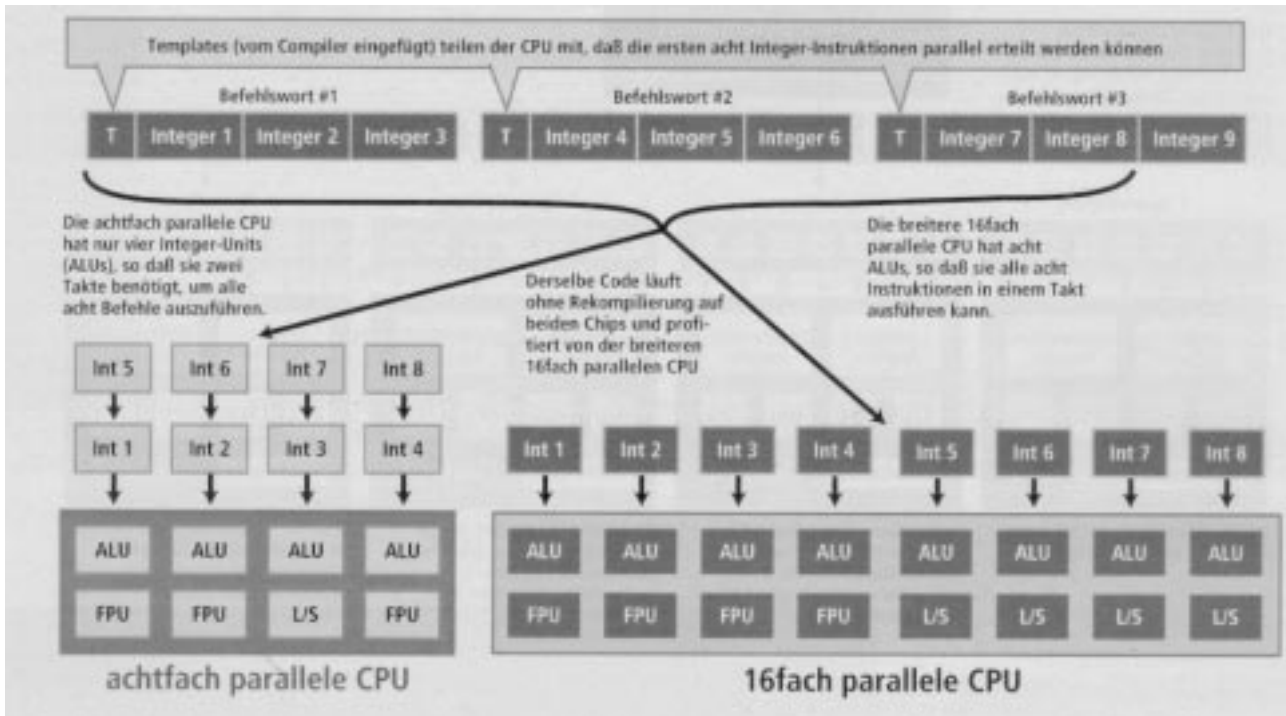


Abbildung 1.36: IA64

- 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

- <http://developer.intel.com/eBusiness/products/ia64/overview/bm012101.htm#5>
- <http://developer.intel.com/eBusiness/products/ia64/overview/bm012101.htm#6>

Figure 3-1. Application Register Model

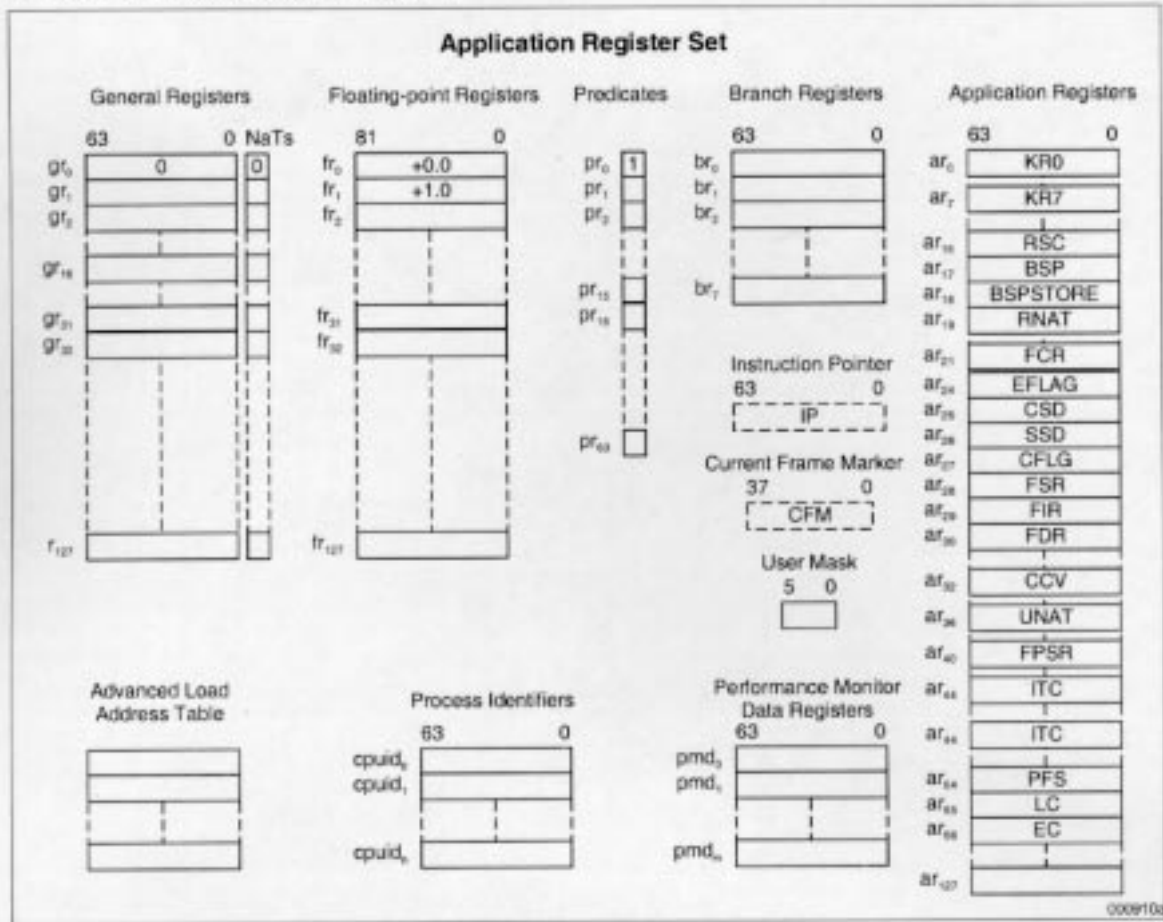
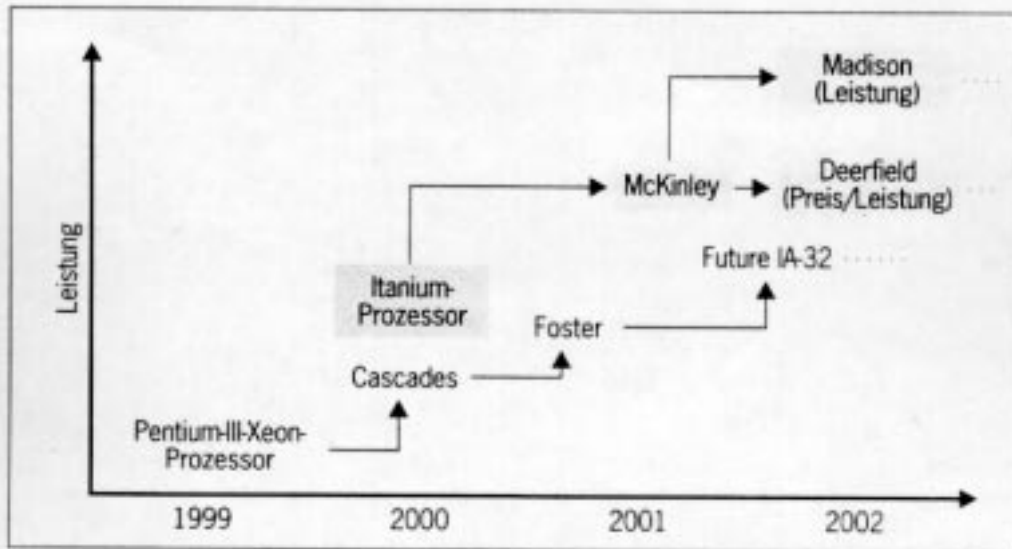


Abbildung 1.37: Register des Itanium

Roadmap für die IA-64-Architektur



Quelle: Intel

CW 26/01-ww

Bestandteile des Alpha-Prozessors sollen in Intels Zukunftschiff Madison einfließen.

Abbildung 1.38: Roadmap IA64

1.8 Zukünftige CPU-Entwicklungen

- Kupfertechnologie (statt Aluminium) für Leiterbahnen
- Dünnschicht-Technologie (mehrere Lagen Schaltkreise übereinander)
- FRAM = Ferroelectric Random Access Memory
- Kristall als Speicher
- Tesafilm, der Superspeicher?
- Wärmeproblem: Wasserkühlung?
- 20 nm-Transistoren — 2007: 20 GHz?
- Silizium-Germanium-Chips: 210 GHz
- Einelektron-Transistoren (Kohlenstoff-Nanoröhrchen)
- Supraleitende Transistoren
- Photonik
- Quantencomputer

Kapitel 2

High Performance Computer

2.1 Vektorrechner

Statt der Verarbeitung einzelner Worte pro Taktperiode versucht man ganze Vektoren/Takt zu verarbeiten (Vervielfältigung der ALU):

Abbildung 2.1: Verarbeitungseinheit: Vektor von Worten

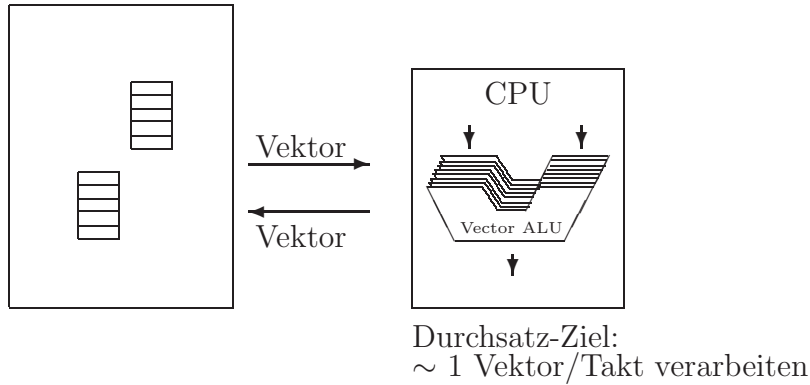
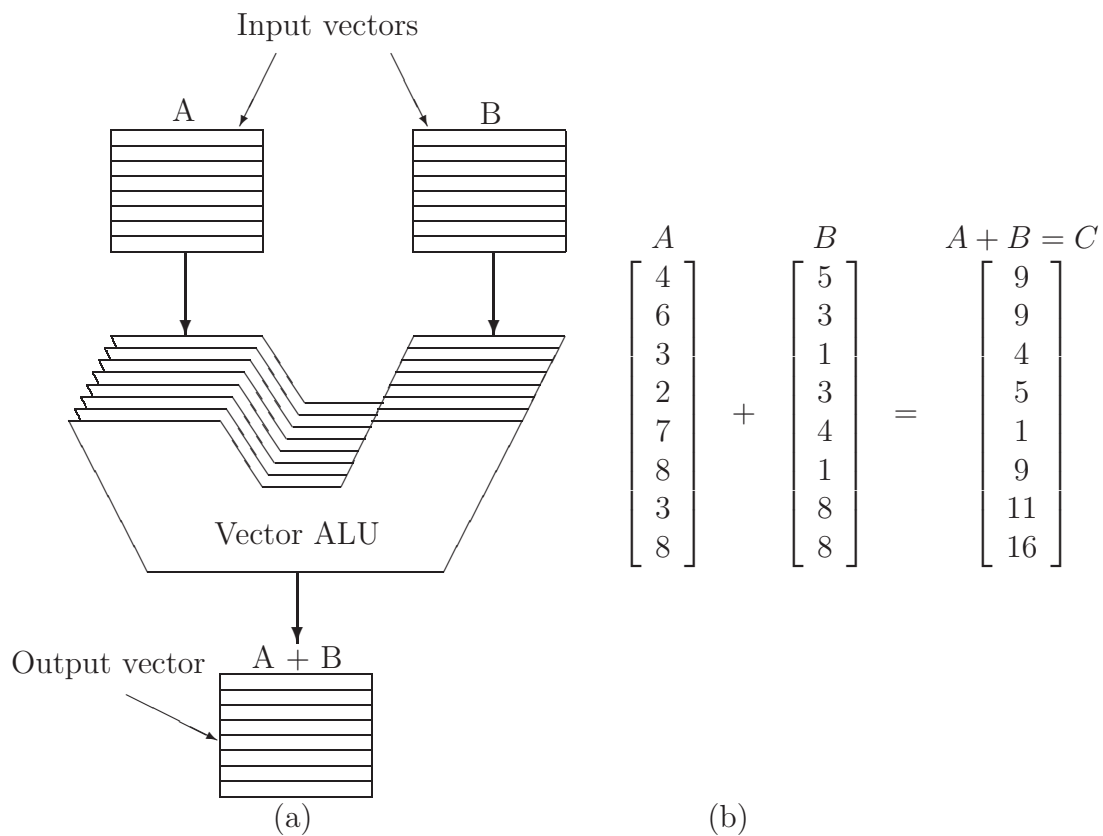


Abbildung 2.2: Vektorregister und Vektoralu



(a) Vektor-ALU.

(b) Beispiel für die Vektor-Addition.

Andere Beispiele für Vektoroperationen:

- $C_i = f_1(A_i)$ mit $f_1 \in \{\cos, \sqrt{\quad}, \dots\}$ komponentenweise ausgewertet
- $s = \sum_{i=1}^N A_i$ Summe, Minimum, Produkt
- Minimum
- Produkt
- Maximum
- ⋮

N ALU's sind i.a.:

- zu teuer, da jede einzelne ALU sehr schnell sein soll
- unflexibel in der Vektor-Länge

Kompromiß: „eine ALU im Pipelinebetrieb statt eines Vektors von ALU's“

FPU-Pipeline (baue ALU aus „unabhängigen“ Einheiten auf):

Beispiel: Subtraktion

| | |
|---------------------|---|
| 1) fetch operands | $1.082 \cdot 10^{12}, 9.212 \cdot 10^{11}$ |
| 2) adjust exponent | $1.082 \cdot 10^{12}, 0.9212 \cdot 10^{12}$ |
| 3) execute „-“ | $0.1608 \cdot 10^{12}$ |
| 4) normalize result | $1.608 \cdot 10^{11}$ |

$$C_i = A_i - B_i \quad \forall i = 1 \dots N$$

| Zyklus | 1 | 2 | 3 | 4 | 5 | ... |
|-----------|------------|------------|-------------|-------------|-------------|-----|
| fetch | B_1, C_1 | B_2, C_2 | B_3, C_3 | B_4, C_4 | B_5, C_5 | |
| adjust | — | B_1, C_1 | B_2, C_2 | B_3, C_3 | B_4, C_4 | |
| execute | — | — | $B_1 - C_1$ | $B_2 - C_2$ | $B_3 - C_3$ | |
| normalize | — | — | — | $B_1 - C_1$ | $B_2 - C_2$ | |

$\swarrow \quad \swarrow$
 $B_1 - C_1 \quad B_2 - C_2$

Resümee:

- Ein Einzelbefehl (z.B. Subtraktion) braucht 4 Zyklen Zeit.
- Nach einer Startup-Zeit von 4 Zyklen (Füllen der Pipe) wird dann jedoch pro Zyklus ein Endergebnis geliefert. (Das ist allerdings nur eine Komponente des Ergebnis-Vektors.)
- Alte Vektorrechner benötigen eine Vektorlänge ≥ 100 , damit sie schneller als ein Skalarrechner arbeiten; neuere Vektorrechner kommen jedoch schon mit einer Vektorlänge von $\geq 2 \dots 4$ aus.
- **Problem:** „Datenabhängigkeit“

Ein Wert ist noch in der Pipe, obwohl er schon als neues Argument benötigt wird.

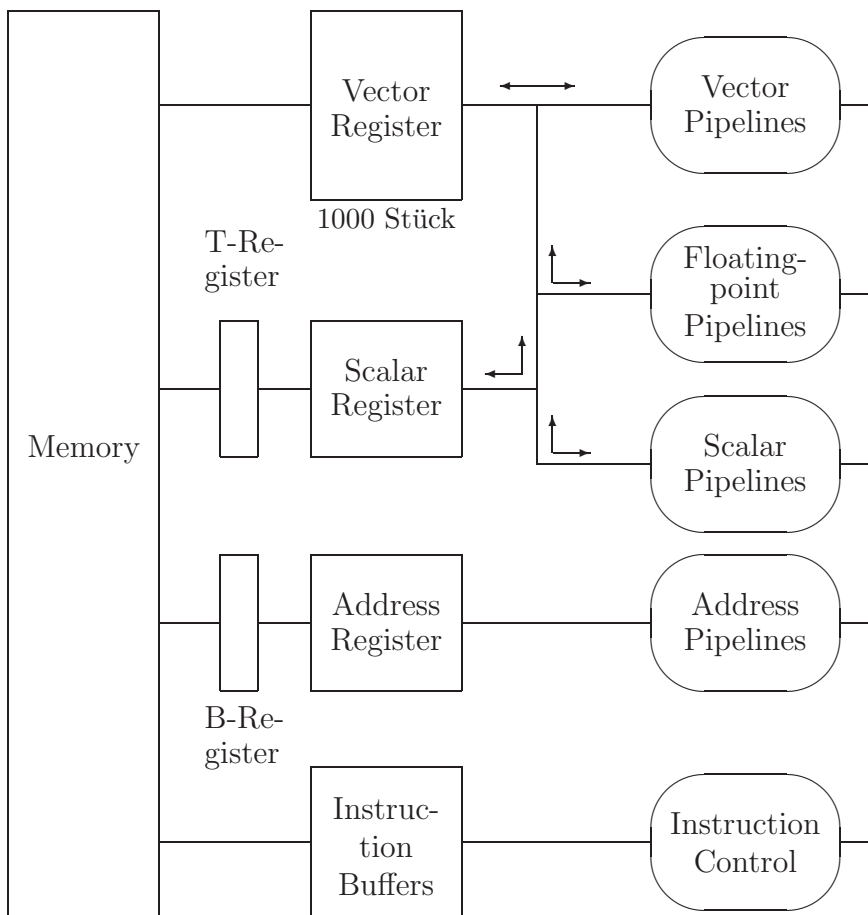
- Jede ALU arbeitet denselben Befehl ab (SIMD = „single instruction multiple data“)
- COMPILER vektorisieren z.T. automatisch.

Beispiel: Cray-1 (1976)

mehrere parallel arbeitende Pipelines,
viele Register,
Taktzeit ~ 12.5 ns

} peak_Leistung: 250 MFlops

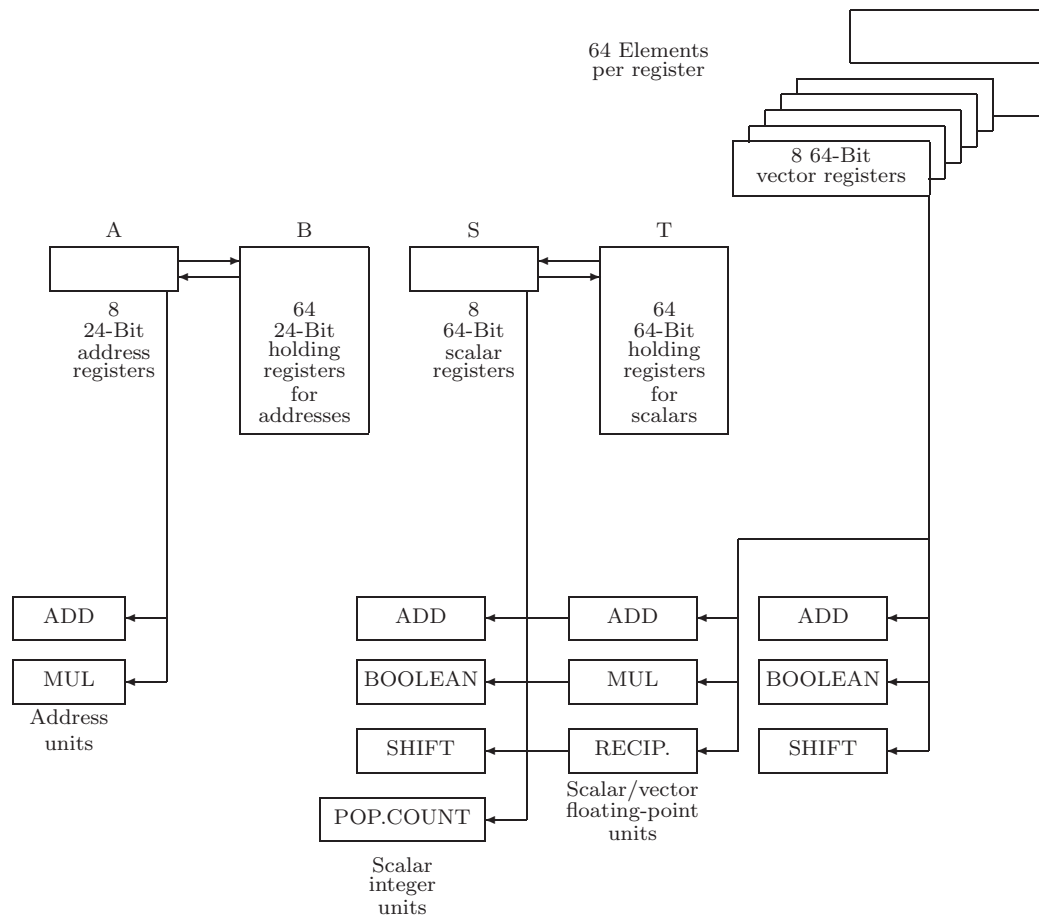
Abbildung 2.3: Rechnerstruktur eines Cray-1-Rechners



Chaining von Pipelines

- Vector Pipelines – Additionen
 - Verschieben
 - logische Operationen
 - Floatingpoint Pipelines – Additionen
 - Multiplikationen
 - Divisionen
 - Address Pipelines – Adreß-Multiplikationen
 - Adreß-Additionen
- } von Ganzzahlvektoren
- } von Gleitkommazahlvektoren

Abbildung 2.4: Die Register und Funktionseinheiten einer Cray-1



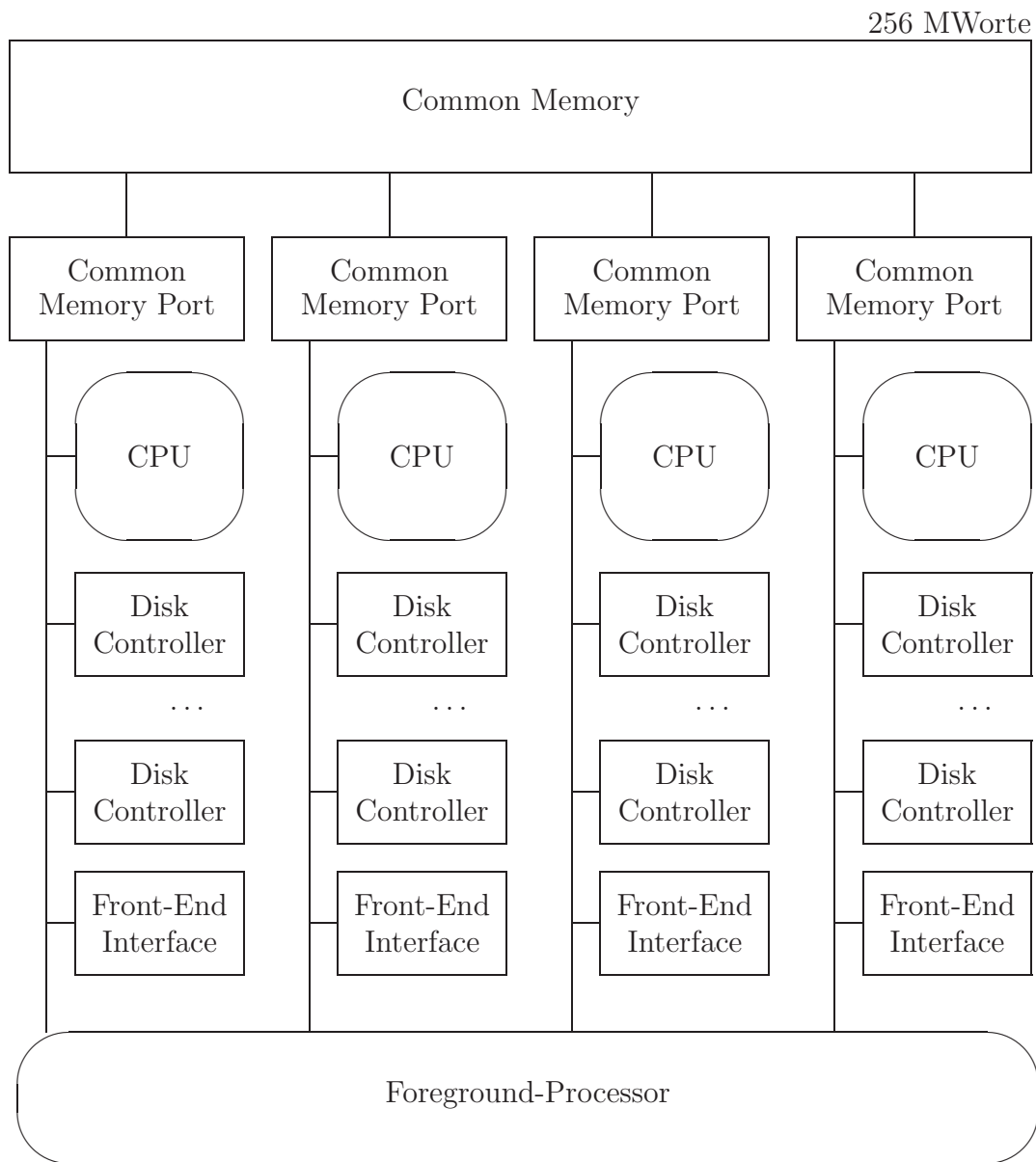
Weitere Geschwindigkeitssteigerungen¹ sind erreichbar durch

ein speichergekoppeltes **Multiprozessorsystem**, das mehrere Zentraleinheiten (MIMD = „multiple instruction multiple data“) oder nur mehrere ALU's (SIMD = „single instruction multiple data“) enthalten kann.

¹ Selbst heutige PC-CPU's enthalten parallel benutzbare Funktionseinheiten (ein oder mehrere Integer-Units, ein oder mehrere FP-Units, . . .), die häufig im Sinne einer Pipeline arbeiten (z.B. beim Pentium: zwei 32 Bit-Integer-Pipelines und eine FP-Pipeline). Naturgemäß sind Programme aus wenigen einfachen Befehlen viel besser parallel zu „dispatchen“ (zur parallelen Abarbeitung vorzubereiten und „in Marsch zu setzen“), weshalb das CISC-RISC-Translatorprinzip des PentiumPro zwar genaugenommen einer Microcode-CPU entspricht, aber das Design des „Microcodes“ als RISC-Code bessere Parallelisierbarkeit und deshalb schnellere CPU's ermöglicht (PentiumPro: dreiweg Superskalarausführung, „speculative execution“, „branch prediction“, 14-stufige Superpipeline).

2.2 SIMD- und MIMD-Parallelrechner

Abbildung 2.5: CRAY-2 (Juni 1985)



Rechnerstruktur eines Cray-2-Rechners

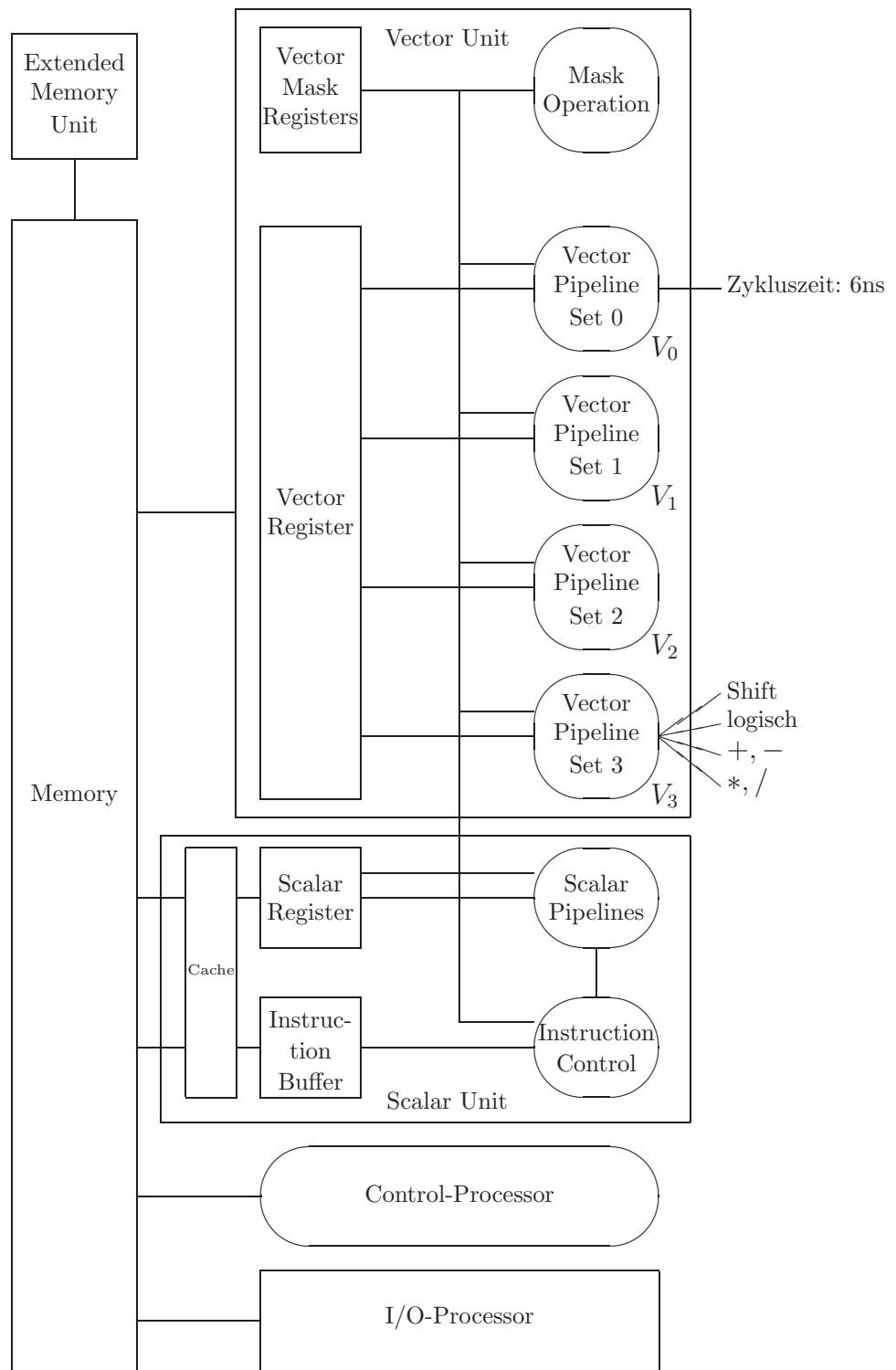
6 ... 12× schneller als Cray-1

| | | |
|----------------------------|-----------------------------------|----------------------------------|
| Cray-3 | 16 CPU's mit 2ns Taktzeit | |
| Cray-4 | 64 CPU's mit 1ns Taktzeit | ~ 128 GFLOPS (~ 1000× Cray-1) |
| Cray T90 ² | 32 CPU's mit je 2 Vektorpipelines | ~60 GFLOPS |
| Cray Origin 2000 | 2048 CPU's | ~2.5 TFLOPS |
| Cray T3E-1200 ³ | 2048 CPU's | ~2.5 TFLOPS |

²Vgl. auch <http://www.cray.com/hpc/>

³Vgl. auch <http://www.cray.com/products/systems/crayt3e/1200/>

Abbildung 2.6: NEC-SX2 SIMD, mehrere Pipelines



Überlappende Abarbeitung der Vector-Pipeline-Sets:

V_0 1,5,9, ...
 V_1 2,6,10, ...
 V_2 3,7,11, ...
 V_3 4,8,12, ...

nichtsichtbar, automatisch
Start-Up-Zeit: 60ns

NEC SX-3/11 mit:

- 256MB Hauptspeicher
- 1GB Erweiterungsspeicher (max. 2GB)
- 20GB Platte
- IBM 3084Q als „front end“ Rechner
- 2.9 ns Zykluszeit
- skalar: 170 MFlops peak, vektoriell: $4 \times 345 \text{MFlops} = 1.386 \text{GFlops}$ pro Vektorpipeline

SX-3/44 mit:

4 Pipelines pro CPU \rightarrow 5.5 GFlops/CPU
4 CPU's in Modell 44 \rightarrow 22 GFlops

NEC SX-4 mit: ⁴

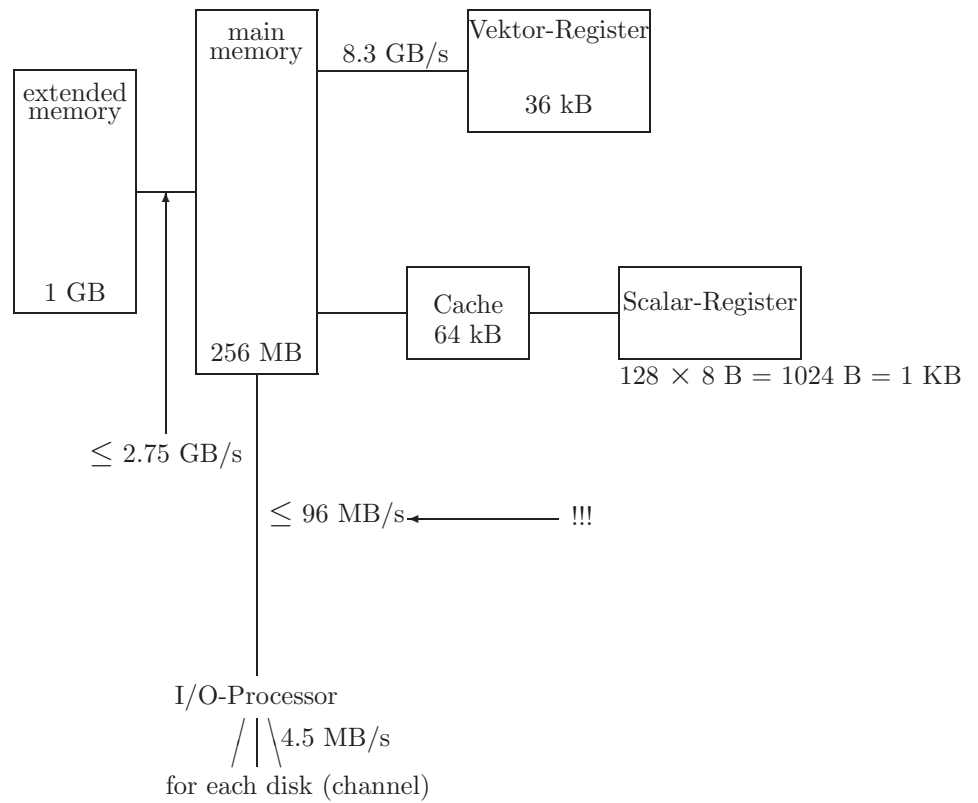
- 512 CPUs mit je 8 Pipelines \rightarrow 1 TFlops⁵

⁴Vgl. <http://www.hpc.comp.nec.co.jp/sx-e/sx-world/no18/report.html>

⁵Vgl. <http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?prefix>

Kritisch bei High-Performance Rechnern ist auch heute noch, das I/O-Subsystem mit einer geeignet hohen Durchsatzrate an die CPUs zu koppeln:

Abbildung 2.7: Systemarchitektur und Datendurchsatz



Einen Überblick zu den jeweils aktuell leistungsstärksten Supercomputern bietet:

<http://www.top500.org/>

2.3 Mehrprozessorsysteme

Welche der Systemeinheiten wird vervielfältigt?

- ALU
- ALU und Steuerwerk
- ALU, Steuerwerk und Speicher

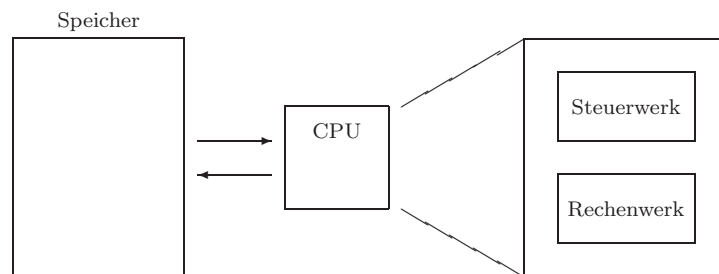


Tabelle 2.1: Klassifikation von Mehrprozessorsystemen

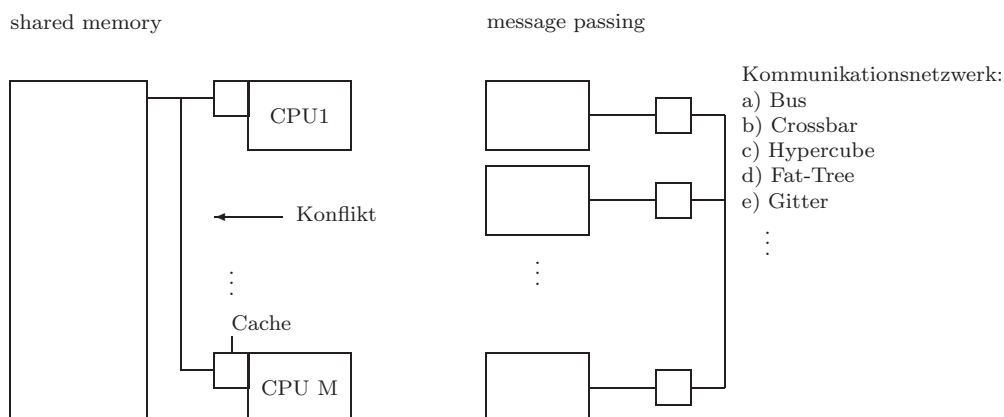
| | CPU's | | Memory | Beispiel |
|------------------------------|------------------------------|-------------|--------|------------|
| | Steuerwerke | Rechenwerke | | |
| von Neumann | 1 | | 1 | „alte“ PCs |
| Vektorrechner | 1 | N | 1 | Cray-1 |
| Multiprozessor-Vektorrechner | M (mit je N Rechenwerken) | | 1 | Cray-2 |

Bemerkung: Rechenwerke oder Pipelines

Tabelle 2.2: Speicher bei Mehrprozessorsystemen

| Multi- prozessor mit | CPU's | | Memory | Beispiel |
|-------------------------|-------------|-------------|--------|-------------------|
| | Steuerwerke | Rechenwerke | | |
| shared memory | M | | 1 | AlphaServer |
| message passing | M | | M | Transputersysteme |
| „Mischung“ | M | | M+1 | ... |

Abbildung 2.8: „shared memory“ und „message passing“



Unter M+1 soll verstanden werden: „Ein gemeinsamer und je ein lokaler Speicher.“

Zuweilen klassifiziert man Multiprozessorsysteme durch folgendes zweidimensionale Schema:

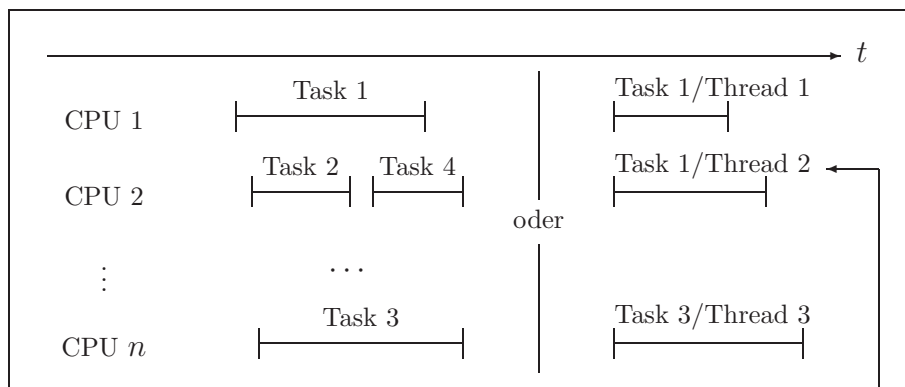
| SW \ HW | homogen | inhomogen |
|--------------|---------|-----------|
| symmetrisch | | |
| asymmetrisch | | |

Häufig: 1 Steuerprozessor und N Arbeitsprozessoren

Aktuell wird daran geforscht, zwischen „shared memory“ und „distributed memory“ (=message passing systems) im sogenannten „virtuellen distributed memory“ den Effizienzvorteil des „distributed memory“ mit der einfacheren Bedienbarkeit des „shared memory“ zu kombinieren.

2.3.1 Multithreadsysteme

- parallele, unabhängig durchführbare Teile eines Prozesses, die deshalb auf Multiprozessorsystemen (\neq Multiprozesssystemen) durch verschiedene CPU's durchgeführt werden können: An der Aufgabe eines Prozesses arbeiten mehrere Prozessoren zugleich.



Prozesse feinkörniger verteilbar

2.4 Multiprozessor-Server

Die Programmierung von Anwendungen, die mehrere Prozessoren im MIMD-Modus nutzen sollen, ist i.a. sehr arbeitsaufwendig. Deshalb gibt es auch nicht viele solche Programme, die auf jedem Knoten wirklich unterschiedliche Algorithmen ablaufen lassen.

Einfacher ist es zum Beispiel, ein Betriebssystem so zu konzipieren, daß bei einem Mehrbenutzerbetrieb die einzelnen Prozesse und Threads jeweils einzelnen Prozessoren zur Abarbeitung übergeben werden: Multiprozessor-Server mit hohem Prozeß-/Thread-Durchsatz.

Für solche DV-Anlagen ist dann jedoch eine hohe Verfügbarkeit unbedingte Voraussetzung: unterbrechungsfreie Stromversorgung, Hotswap, redundante Komponenten, ...

Der hohe Preis von solchen Servern ist jedoch nicht immer akzeptabel, so daß zur Zeit an *verteilten Betriebssystemen* gearbeitet wird, die mehrere billigere PCs oder Workstations zu einem virtuellen Mehrprozessorserver vereinigt erscheinen lassen: Man meldet sich dann an einem „virtuellen“ Rechner an und arbeitet auf einem gerade freien Rechner des ganzen Clusters. Beim nächsten Anmelden arbeitet man eventuell auf einem anderen Rechner, was man jedoch überhaupt nicht merkt, da das Betriebssystem die ganze Arbeitsumgebung unabhängig von dem gerade wirklich benutzten Rechner einheitlich erscheinen läßt.

Wirkliche Multiprozessor-Server sind dann nur noch in den Fällen wirtschaftlich, wenn in Abteilungsservern spezielle Dienste (Datenbanken, WWW-Intranets, Fileexport (NFS) , ...) bereitgestellt werden müssen.

Aktuelle Daten zu Multiprozessor-Servern findet man unter:

<http://www.sun.de/Produkte/Hardware/index.html>
<http://www.sgi.de/produkte/index.html>
<http://www.compaq.com/products/servers/index.html>
<http://www.hewlett-packard.de/enterprise/>
<http://www.ibm.com/products/>
...

Kapitel 3

Rechnerarchitekturen – Fortsetzung

3.1 Peripheriebus: USB und Firewire

3.1.1 USB = Universal Serial Bus

- Packet switched Peripherie-Bus
- LowSpeed: 1.5MBit/s, FullSpeed: 12 MBit/s
- Hot-swapping
- Eingebaute Stromversorgung für Endgeräte mit geringem Stromverbrauch
- Audioübertragung mit fest ausgehandelter Bandbreite
- kaskadierende Hub's (maximal 7 Ebenen inkl. root-Hub), bis zu 127 Geräte
- Kabellänge maximal 5 m
- PC-kontrolliertes Master-Slave Protokoll
- seit 1996 (weit verbreitet erst seit standardmäßig in Chipsets vorhanden)
- einheitlicher "Ersatz" für die serielle und die parallele Schnittstelle für Tastatur, Maus, Monitor, Drucker, lowspeed Scanner, digitaler Fotoapparat, ISDN-Adapter, Ethernetkarten (10 MBit/s), ...

3.1.2 Firewire, IEEE 1394

- Packet switched Hochgeschwindigkeitsperipherie-Bus
- 100, 200, 400 MBit/s
- Hot-swapping
- Daisy-Chaining bis zu 16 Kabeln, maximal 63 Geräte
- Kabellänge maximal 4.5 m
- Peer-to-Peer

3.1.3 USB2

- abwärtskompatibel zu USB 1.1
- HighSpeed: 480 MBit/s
- auch geeignet für: Festplatten, DVD-Laufwerke, CD-Brenner mit mehr als 4x Geschwindigkeit, Fast Ethernet (100 MBit/s), Camcorder (≥ 28 MBit/s), ...
- noch nicht verfügbar/erprobt

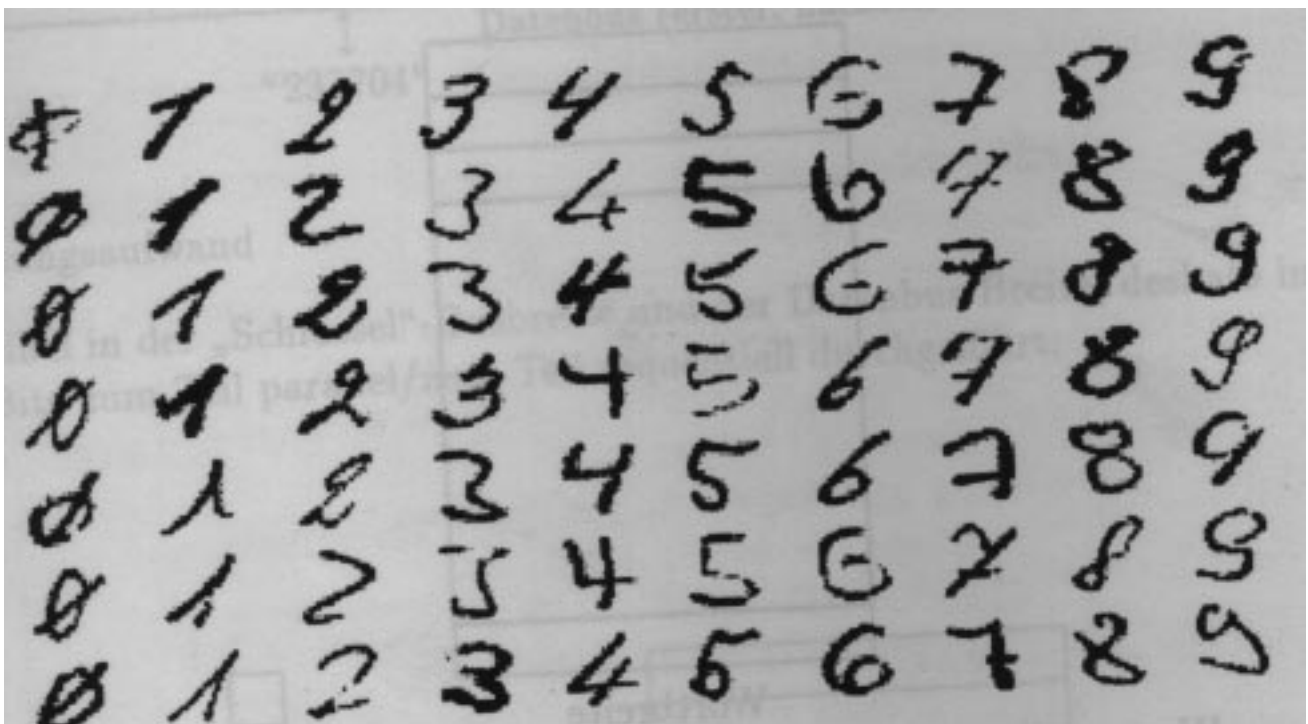
3.1.4 Firewire, IEEE 1394b

- geplante Erweiterung des Firewire
- 800, 1600 MBit/s
- bis zu 100 m Übertragungstrecke möglich

3.2 Neuronale Netze

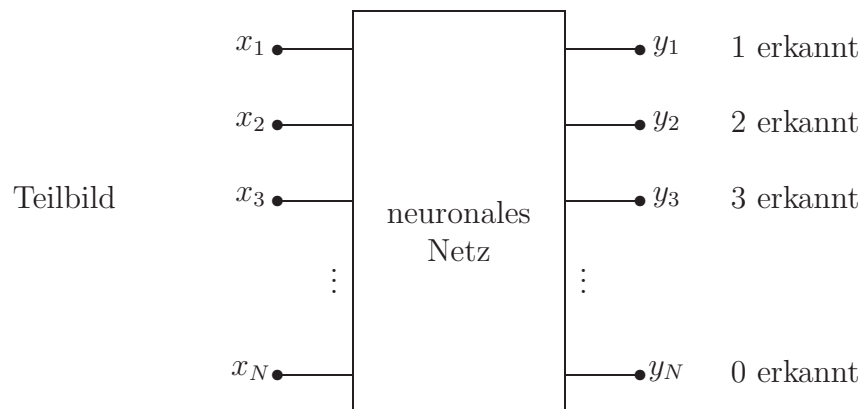
Komplexe Aufgaben sind zum Teil nicht mehr analytisch (d.h. durch Algorithmen) lösbar bzw. müssen für jede Eingabedatenkonfiguration aufwendig wieder neu gelöst werden.

Beispiel:



Handgeschriebene Ziffern
(größennormierte Darstellung mit einer 16×16 Grauwertmatrix)

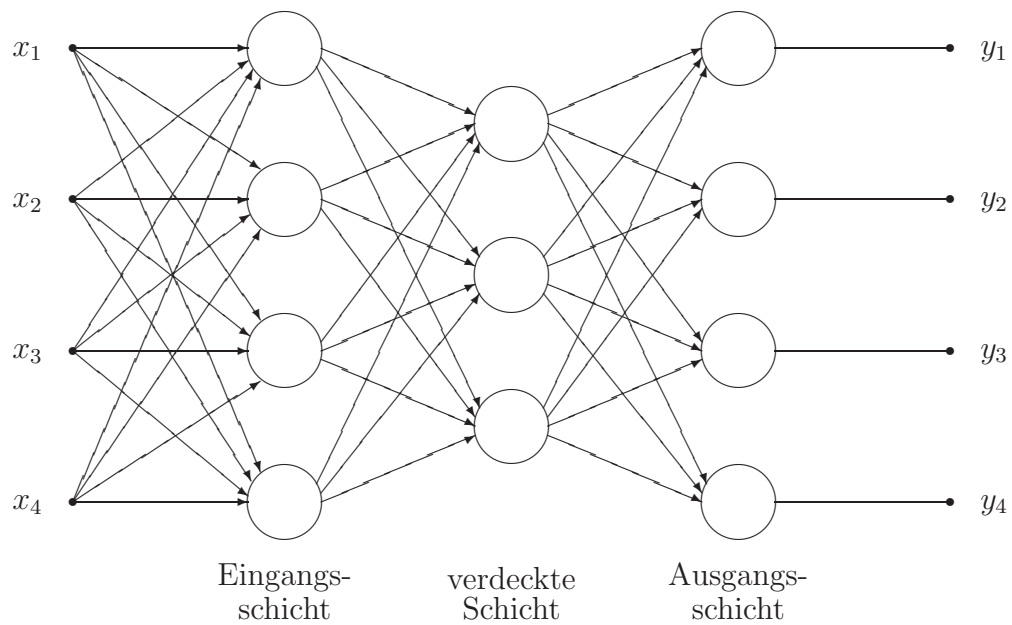
Die Erkennung handgeschriebener Ziffern (etwa für die Postleitzahlen-Erkennung) ist analytisch nicht sehr gut beschreibbar. Deshalb wird versucht, ein dem menschlichen Gehirn ähnlich komplexes, „lernendes“ Schaltwerk — ein neuronales Netz — zur Problemlösung einzusetzen.



Neuronales Netz mit Ein- und Ausgangsvektoren

Merkmale („künstlicher“) neuronaler Netze:

- *viele* Neuronen
- feste problemspezifische Verschachtelung der Neuronen
- Verbindungen gewichtet (hemmend, fördernd)
- jedes Neuron agiert durch
 - ständige gewichtete Aufsummierung aller Eingänge und „Schwellwert“-Ausgabe auf andere Neuronen
- häufig modularer Aufbau: Schichten



Bei einigermaßen komplexen Netzen sind die benötigten Gewichte für die gewünschte Aufgabe nicht mehr berechenbar; deshalb:

Training / Lernen der Gewichte

durch „Rückkopplung“ oder „Backpropagation“.

Vorteile beim Einsatz von neuronalen Netzen:

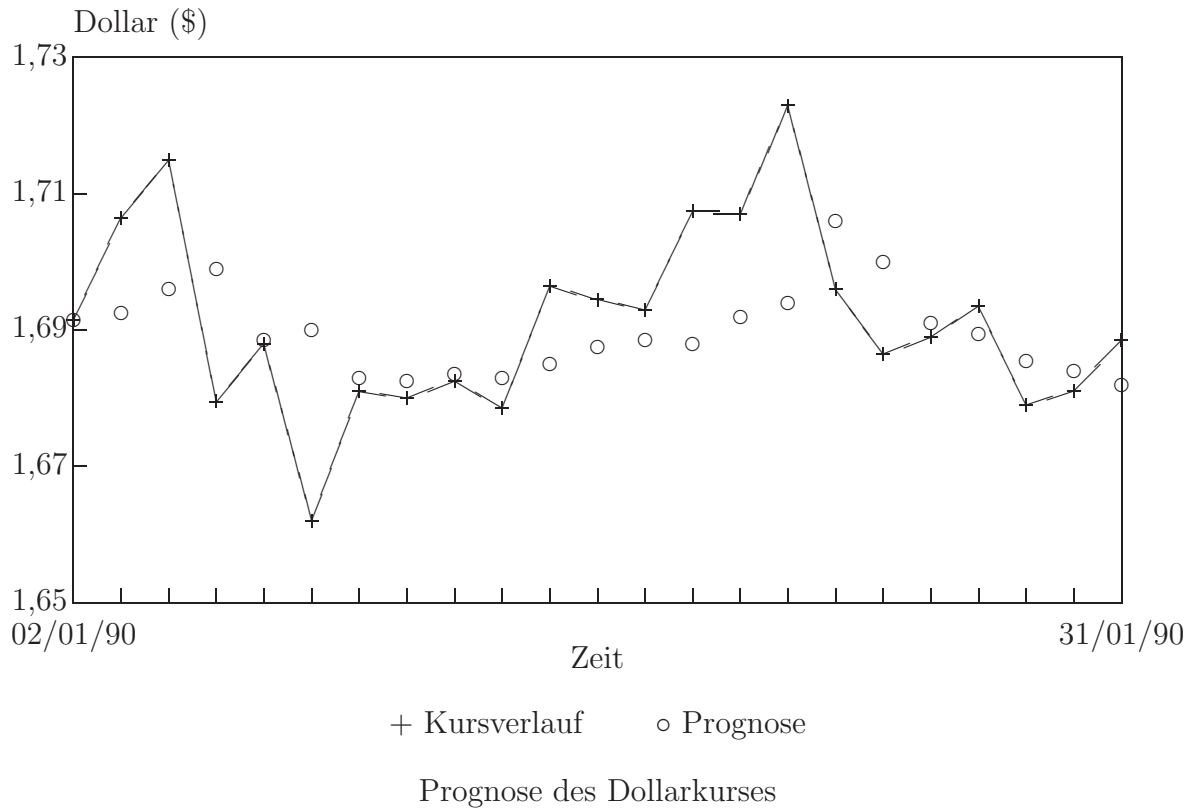
- Lernfähigkeit (jedoch lange Dauer)
- Generalisierung (erlernt auch das Verhalten für „ähnliche“ Situationen, d.h. auch für nicht gelernte Eingabesituationen werden „sinnvolle“ Ausgaben erzeugt)
- Fehlertoleranz (gegenüber dem Ausfall einzelner Neuronen oder Eingangsdaten)
- Unempfindlichkeit gegen Rauschen

Nachteile:

- Analyse der Klassifikationskriterien unmöglich
- Lernen (= Programmieren) nur durch Training möglich
- das Trainingsprogramm muß auf die zu lernende Aufgabe abgestimmt sein; das Erreichen dieser Abstimmung ist häufig nur unklar zu erkennen
- Lernen sehr zeitaufwendig (dadurch aber keine Analyse der Probleme nötig)
- sehr eingeschränkt einsetzbar, meist Klassifikation, Mustererkennung, Optimierung, Kontrollaufgaben (Robotersteuerung)
- kein universelles neuronales Netz, d.h für jede Aufgabe
 - spezielles Netz
 - spezielle Verbindungsstruktur
 - spezielle Trainingsdaten

In gewissem Rahmen verlangt der Einsatz von Expertensystemen ebenfalls keine analytische Durchdringung des zu lösenden Problems (sondern „nur“ eine Sammlung von „Wissen“ in Form von Fakten und Regeln). Im Gegensatz zu neuronalen Netzen liefert aber der Einsatz von Expertensystemen eine nachvollziehbare Begründung der Problemlösung.

Anwendungen:



- Bei Kursprognosen wurde eine Trefferquote von $\sim 56\%$ erreicht.
- Mustererkennung beim Lesen von Postleitzahlen.
- Klassifizierung von Gefahrensituationen
- ...

Vergleich: Elektronik – Biologie, künstliche — natürliche Neuronen

| | Elektronik | Biologie |
|--------------------|--|--|
| Clock-Frequenz | $10^8 s^{-1}$ | $10^3 s^{-1}$ Frequenz der Neuronenimpulse |
| Speicherdichte | 10^7 Bit/cm ⁻³ | 10^{12} Synapsen/cm ⁻³ |
| Verbindungen | 10 fan-in, fan-out von digitalen Gattern | 10^4 Synapsen pro Neuronen |
| Anzahl Elemente | 10^9 Transistoren | 10^{11} Neuronen, 10^{15} Synapsen |
| Leistungsverbrauch | 1 Kilowatt | 50 Watt |

Eine Auswahl von elektronischen neuronalen Netzen:

| | | Verbindungen pro Sekunde (x 10 ⁹) | Verbindungen Auflösung | Anzahl Neuronen | Anzahl Verbin- dungen | Anzahl Gewichte | Technologie | Lernen |
|---|--|---|-----------------------------------|--------------------|-----------------------------|--------------------|--|---|
| 1 | M.Griffin et al. Adaptive Solutions | 1,6 | 9 Bit x 16 Bit fixed point | 64 | 64 | 16.384 | 0,8 μm CMOS 26,2 mm x 27,5 mm | auf einem Chip verschiedene Algorithmen |
| 2 | M.Asai et al. Hitachi | 1,5 | 8 Bit x 8 Bit product | 576 | 576 | 36.864 | 0,8 μm CMOS Gate Array, Wafer 5 Inch | nicht auf dem Wafer |
| 3 | M.A.Sivilotti et al. Neural Semiconductor | 2 | 4 Bit x 4 Bit product | 32 | 1.024 | 1.024 | ? | nicht auf dem Chip |
| 4 | H.Graf, D.Henderson AT&T Bell Labs. | 80 | 1 Bit x 4 Bit product flexible | 256 | 8.192 | 8.192 | 0,9 μm CMOS 4,5mm x 7 mm | nicht auf dem Chip |
| 5 | Y.Arima et al. Mitsubishi Electric | 28 | 1 Bit x 6 Bit product | 336 | 28.000 | 28.000 | 1 μm CMOS 14,5 mm x 14,5 mm | auf dem Chip Boltzmann |
| 6 | R.Allen, J.Alspector and A.Jayakumar, Bellcore | 0,1 | 5 Bit x 5 Bit product | 32 | 992 | 992 | 1,25 μm CMOS | auf dem Chip Boltzmann |
| 7 | R.Benson et al. Intel Corp. | 2 | 6 Bit x 6 Bit product | 64 | 10.240 | 10.240 | 1 μm CMOS EEProm | nicht auf dem Chip |

Quelle:Weber

Die ersten drei Schaltungen sind digitale Prozessoren, während die folgenden vier zumindest teilweise analoge Schaltungen sind.

Index

- 680x0, [33](#)
- 80x86, [33](#)
- 88000, [51](#)

- Aiken, Howard H., [4](#)
- ASCII, [15](#)

- Babbage, Charles, [4](#)
- Betriebssystem, [3](#)

- Cache, [55](#)
- chip, [34](#)
- CISC, [33](#), [35](#), [40](#)
- Computer, [1](#)
- Cray Origin 2000, [96](#)
- Cray T3E-1200, [96](#)
- Cray T90, [96](#)
- Cray-1, [92](#)
- Cray-2, [95](#)
- Cray-3, [96](#)
- Cray-4, [96](#)

- Datendurchsatz, [55](#), [99](#)
- Datenpfad-Zykluszeit, [39](#)
- DEC-alpha, [43](#), [52](#)
- Deschutes, [46](#)
- Durchsatz, [55](#)

- Eckert, John P., [4](#)
- EISA, [44](#)

- General Scripts, [20](#)

- Handshake-Steuerbus-Leitungen,
[12](#)

- Hotswap, [103](#)

- IBM360, [33](#)
- ISA, [44](#)

- Maschinenbefehle, [2](#)
- Mauchley, John W., [4](#)
- MCA, [44](#)
- Mehrprozessorsysteme, [100](#)
- message passing, [101](#)
- Microprogramm, [34](#)
- MIMD, [94](#)
- MIPS, [43](#), [53](#)
- Monitorprogramm, [3](#)
- Multitasking, [56](#)
- Multithreadsysteme, [102](#)

- Nanoprogramm, [35](#)
- NEC-SX2, [97](#)
- NEC-SX3/11, [98](#)
- NEC-SX3/44, [98](#)
- NEC-SX4, [98](#)

- P2SC, [44](#)
- PCI, [44](#)
- Pentium, [43](#), [45](#)
- Pentium II, [46](#)
- Pentium MMX, [45](#)
- PentiumPro, [45](#), [47](#), [48](#)
- Pipeline, [41](#), [91](#)
- Power, [51](#), [52](#)
- Power Architecture, [43](#), [50](#)
- PowerPC, [44](#), [54](#)
- preemptiv, [56](#)
- Primitivbefehle, [2](#)

Rechnerarchitektur, 4
redundant, 103
Registerstack, 42
RISC, 39, 40
RISC-Designprinzip, 39
ROM, 3

semantic gap, 35
sequentieller Aneinanderreihung, 2
Server, 103
shared memory, 101
SIMD, 91, 94
SPARC, 43, 49
SPECfp95, 48
SPECint95, 47
Spezialmaschinen, 2

tasks, 56
Transputer, 101

Ultra Part Architecture, 72
Ultra-SPARC, 44
UNICODE, 19
Urlader, 3
USV, 103

VAX, 33
Vektoren, 55
Vektorrechner, 89
Vektorregister, 90
Vesa Local Bus, 44
virtuell, 101, 103
von Neumann, John, 4
von-Neumann-Computer, 4, 11

wait-states, 34

Zuse, Konrad, 4