1.3 Field replaceable units (FRU's)

Am Beispiel des kleinsten RISC-Rechners von Sun Microsystems, der ELC:

- Systemplatine, darauf:
  - NVRAM, d.h. non volatile (= nicht flüchtiger) Speicher
  - Speichermodule SIMM's (= single inline memory module)
- Bildschirm/Monitor

Und in größeren Rechnern:

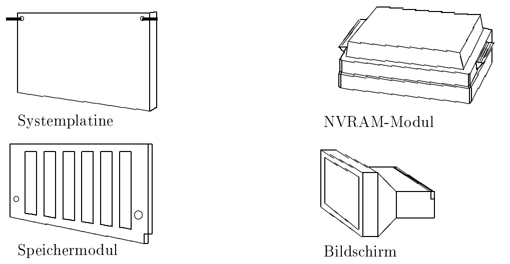
- **SBUS**-Karten: z.B. SCSI-Interface, ...

$\xrightarrow{\text{für I/O}}$ „offener" Chip-Bus, direkt mit internem CPU-Bus verbunden, keine Treiberlogik auf Karte notwendig, postkartengroß, 96 poliger Stecker, maximal 10 Watt, 32 Bit Daten, $\leq$ 100 MB/s (aktuell 20...25 MB/s), Hardware-Adresse entspricht Steckplatz, Karte identifiziert sich bei Systemstart selbständig, kein Jumper nötig.

- **MBUS**-Karten: Prozessor (IU + FPU + Cache Controller + Cache Memory), pro Karte 1 oder 2 Prozessoren

$\xrightarrow{\text{für Memory}}$ „offener" Memory-Bus, 40 MHz Takt, 64 Bit Daten, $\leq$ 320 MB/s für Memory oder CPU's geeignet.

Abbildung 1.8: FRU's der SPARC-ELC

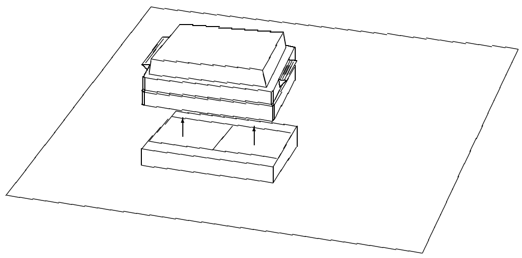


Auszüge aus Reparaturanleitungen für Feld-Techniker:

- **Entfernen des NVRAM-Moduls:**

  Fassen Sie das Gehäuse an den Griffen und ziehen Sie das NVRAM kräftig nach oben ab (siehe Abbildung):

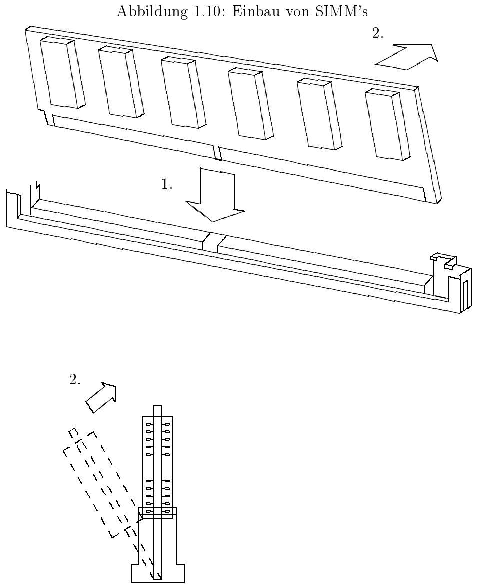
Abbildung 1.9: Entfernen des NVRAM-Moduls



Vorsicht: Die Systemplatine und das NVRAM-Modul bestehen aus hochempfindlichen elektronischen Bauteilen, die gegen statische Aufladung empfindlich sind. Vergewissern Sie sich, daß Sie beim Austausch der Komponenten alle Sicherheitsbestimmungen einhalten.

- **Einbau von Speichermodulen:**

Abbildung 1.10: Einbau von SIMM's

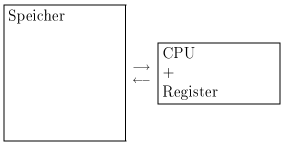


Hinweis: Es spielt keine Rolle, welches Modul Sie in welchen Steckplatz installieren. Sie können jedoch nicht ein Modul zwischen zwei anderen bereits installierten Modulen montieren, da die Sockel dafür zu eng nebeneinander liegen.

Vorsicht: Vergewissern Sie sich, daß Sie den Steckplatz MEM1 nicht frei lassen, wenn Sie auf Ihrer Systemplatine weniger als vier Module installieren. Das System arbeitet nicht richtig, wenn kein MEM1-Modul installiert ist.

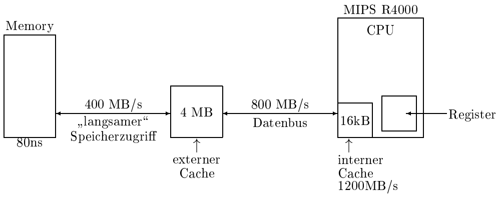
1.4 SIMD/MIMD und „data parallel programming“ (SPMD)

Abbildung 1.11: Der Flaschenhals zwischen CPU und Memory



1.4.1 Cache-Speicher

Abbildung 1.12: Der Cache als Daten-Vorratsbehälter



Bemerkungen:

- 80ns $\longleftrightarrow \frac{1}{80 \cdot 10^{-9}s} = 12.5 \cdot 10^6$ Zugriffe/s
- Bus: 50 MHz und 8 Byte Busbreite führt zu 400 MB/s
- im SIMM müssen also 32 Bit-Zugriffe „parallel“ erfolgen
- externer und interner Cache bilden einen Puffer, der mit “look ahead“ bei einer Trefferquote von $\sim 90 - 95\%$ betrieben wird.

- Es handelt sich um eine „schnelle CPU“ mit 100 MHz; also ergeben sich bei einem Durchsatz von 1200 MB/s durchschnittlich $\leq$ 12B pro Taktperiode $\leq$ 3 Worte/Eintakt-Befehl (32 Bit Befehl und 64 Bit Daten).

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:

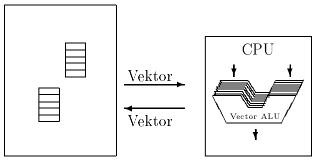
- wilden Sprüngen, ...
- Benutzung von verketteten Listen mit „großen“ Knoten (CAD-Programme), ...

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

1.4.2 Vektorrechner

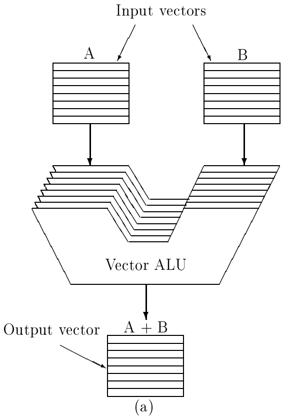
Statt der Verarbeitung einzelner Worte pro Taktperiode versucht man ganze Vektoren/Takt zu verarbeiten (Vervielfältigung der ALU):

Abbildung 1.13: Verarbeitungseinheit: Vektor von Worten



Durchsatz-Ziel:
$\sim$ 1 Vektor/Takt verarbeiten

Abbildung 1.14: Vektorregeister und Vektoralu



(a)

$$
\begin{array}{ccccc}
A & & B & & A+B=C \\
\begin{bmatrix} 4\\6\\3\\2\\7\\8\\3\\8 \end{bmatrix} & + & \begin{bmatrix} 5\\3\\1\\3\\4\\1\\8\\8 \end{bmatrix} & = & \begin{bmatrix} 9\\9\\4\\5\\1\\9\\11\\16 \end{bmatrix}
\end{array}
$$

(b)

(a) Vektor-ALU.

(b) Beispiel für die Vektor-Addition.

Andere Beispiele für Vektoroperationen:

$C_i = f_1(A_i)$ mit $f_1 \in \{\cos, \sqrt{\ }, \ldots\}$ komponentenweise ausgewertet

$s = \sum_{i=1}^{N} A_i$ Summe, Minimum, Produkt
Minimum
Produkt
Maximum
$\vdots$

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	$1.608 \cdot 10^{11}$

$C_i = A_i - B_i \; \forall \; i = 1 \ldots N$

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	—	—	—	$B_1 - C_1$	$B_2 - C_2$	

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

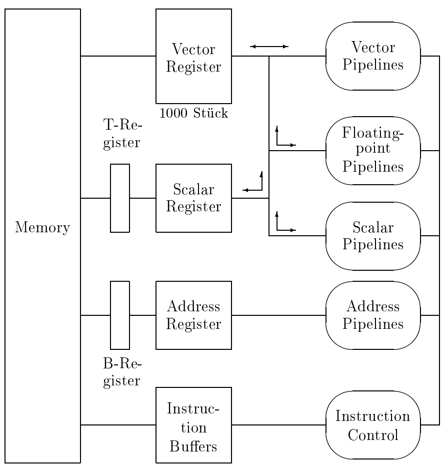
$\boxed{\textbf{Resümee:}}$

- 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 $\geq 2 \ldots 4$ aus.
- **Problem:** „Datenabhängigkeit“
  $\boxed{\text{Ein Wert ist noch in der Pipe, obwohl er schon als neues Argument benötigt wird.}}$
- Jede ALU arbeitet denselben Befehl ab (SIMD = „single instruction multiple data“)
- COMPILER vektorisieren z.T. automatisch.

Beispiel: Cray-1 (1976)

mehrere parallel arbeitende Piplines,
viele Register,
Taktzeit ~ 12.5 ns
} peak_Leistung: 250 MFlops

Abbildung 1.15: Rechnerstruktur eines Cray-1-Rechners



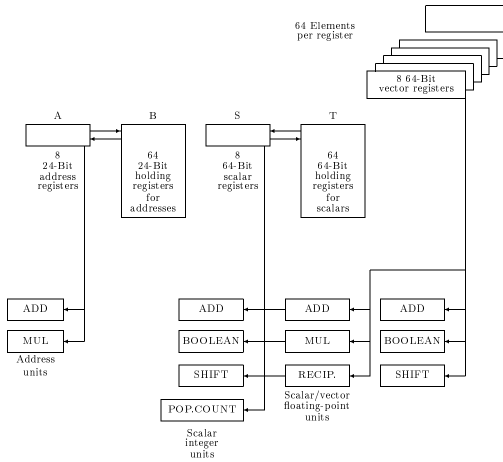
Chaining von Pipelines

Vector Pipelines – Additionen
– Verschieben
– logische Operationen
} von Ganzzahlvektoren

Floatingpoint Pipelines – Additionen
– Multiplikationen
– Divisionen
} von Gleitkommazahlvektoren

Address Pipelines – Adreß-Multiplikationen
– Adreß-Additionen

Abbildung 1.16: Die Register und Funktionseinheiten einer Cray-1



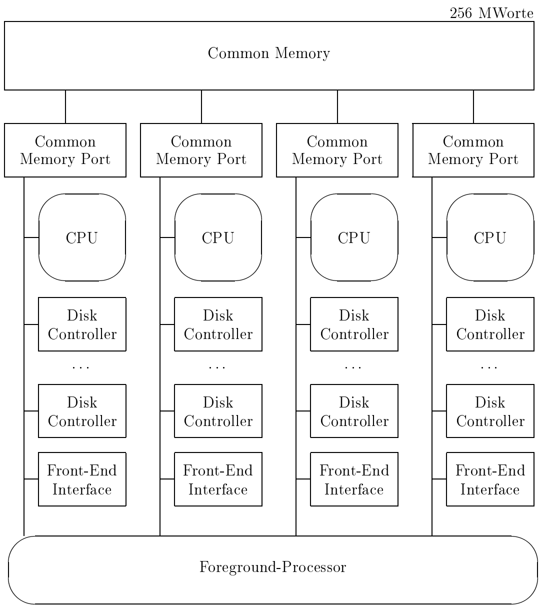
Weitere Geschwindigkeitssteigerungen[1] sind erreichbar durch

> 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-CPUs 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 CPUs ermöglicht (PentiumPro: dreiweg Superskalarausführung, „speculative execution“, „branch prediction“, 14-stufige Superpipeline).

1.4.3 SIMD- und MIMD-Parallelrechner

Abbildung 1.17: CRAY-2 (Juni 1985)

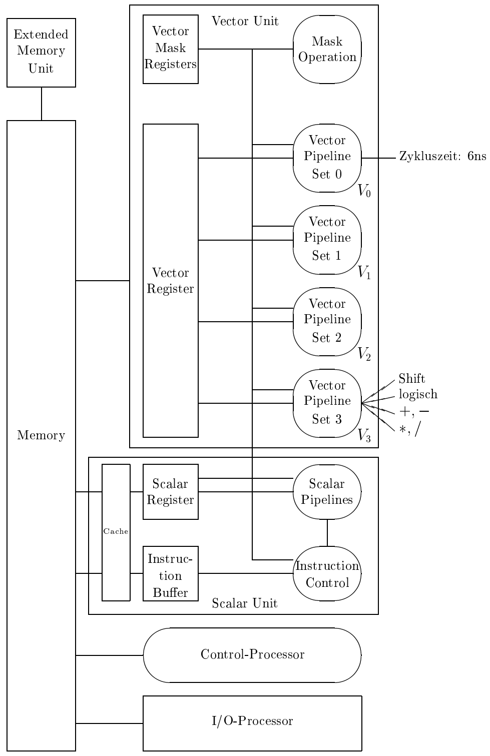


Rechnerstruktur eines Cray-2-Rechners 6...12× schneller als Cray-1

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

[2]Vgl. auch `http://www.cray.com/hpc/`
[3]Vgl. auch `http://www.cray.com/products/systems/crayt3e/1200/`

Abbildung 1.18: 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× 345MFlops = 1.386GFlops pro Vektorpipeline

SX-3/44 mit:

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

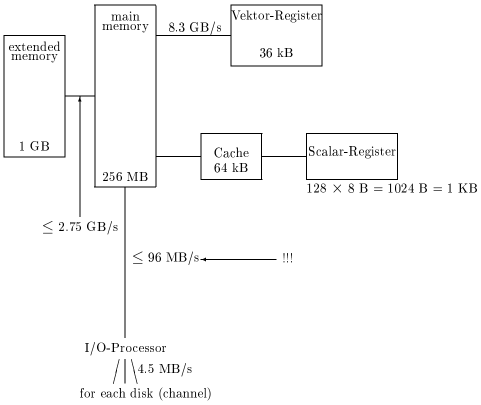
NEC SX-4 mit: [4]

- 512 CPUs mit je 8 Pipelines → 1 TFlops[5]

[4] Vgl. `http://www.nec.com`
[5] Vgl. `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:

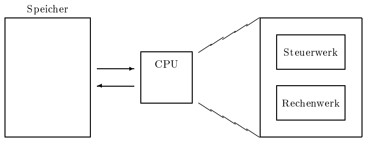
Abbildung 1.19: Systemarchitektur und Datendurchsatz



Einen Überblick zu den jeweils aktuell leistungsstärksten Supercomputern bietet:

`http://top500.rz.uni-mannheim.de/`

1.4.4 Mehrprozessorsysteme



Welche der Systemeinheiten wird vervielfältigt?

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

Tabelle 1.11: Klassifikation von Mehrprozessorsystemen

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

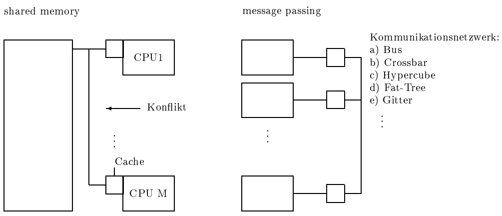
Bemerkung: Rechenwerke oder Pipelines

Tabelle 1.12: Speicher bei Mehrprozessorsystemen

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

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

Abbildung 1.20: „shared memory“ und „message passing“



Zuweilen klassifiziert man Multiprozessorsysteme durch folgendes zweidimensionale Schema:

SW \ HW	homogen	inhomogen
symmetrisch		
unsymmetrisch		

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.

Kopplung der Komponenten: schnelle Busse

Zum Beispiel bei Sun-Workstations:

- **SBus:** für I/O, zum Beispiel: SCSI
  - offener Chip-Bus, direkt mit internem CPU-Bus verbunden
  - keine Treiberlogik auf Karte nötig (je ≤ 10 Watt)
  - keine Jumper nötig (Selbstidentifikation)
  - 96-poliger Stecker, 32 Bit Daten
  - ≤ 100 MB/s (aktuell 20...25 MB/s)

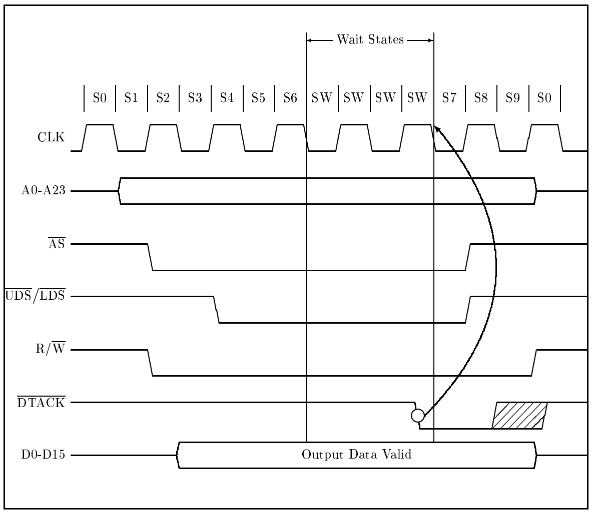
  } circuit switched (Wort-Transfer ist blockierend)
- **MBus:** (Design für ≤ 4 Prozessoren – SPARCStation 10)
  - offener, schneller Memory Bus
  - 40 MHz Takt
  - 64 Bit Daten
  - ≤ 320 MB/s (effektiv 105 MB/s)

  } circuit switched (Wort-Transfer ist blockierend)
- **XDBus:** (für SPARCServer 2000/1000, Design für ≤ 8 Prozessoren)
  - offener, schneller Memory/CPU-Bus
  - 40 MHz
  - 64 Bit Daten
  - 320 MB/s (effektiv 250 MB/s)

  } packet switched, deshalb: überlappende Arbeitsweise möglich

„circuit switched“ Datentransfer: Wait States

Abbildung 1.21: Zeitdiagramm „circuit switched“ Datentransfer



Wait States während einer Schreib-Operation

Bemerkungen:

- Signalnamen:

CLK	=	clock
A0–A24	=	address bus
AS	=	address strobe
$R/\overline{W}$	=	read/write
UDS,LDS	=	upper/lower data strobe
D0–D15	=	data bus
DTACK	=	data transfer acknowledge

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

Zusammenfassend:

- circuit switched: Der Bus blockiert, bis die anstehende Operation ausgeführt ist: „wait states“
- packet switched[6]: 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.

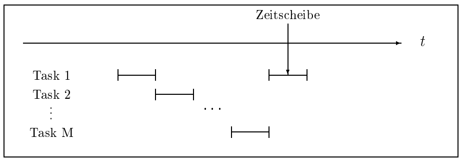
  Problem: Steuerung der CPU nichtsequentiell (mehrere Prozesse lösen sich beim Warten ab, Threads, ...)

[6] Der USB-Anschluß (*Universal Serial Bus*) an neueren PCs bietet eine paketorientierte Datenübertragung mit bis zu 12 Mbps zur Kommunikation von CPU und Peripherie. Insbesondere ermöglicht er die Übertragung von Audiodaten mit zuvor fest ausgehandelter Bandbreite.

Multiprozess- und Multiprozessorsysteme

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:

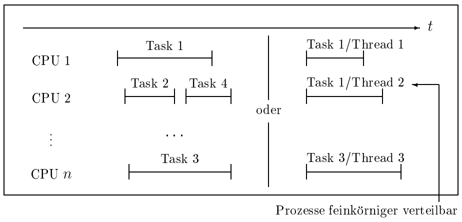
Windows 3.x: Multitasking, nicht preemptiv.

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

Preemtiv (= „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.

Threads

- 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.



1.4.5 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 unterschieldliche 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) , ...) bereitgestelt werden müssen.

Aktuelle Daten zu Multiprozessor-Servern findet man unter:

```
http://www.sun.de/Produkte/Server/index.html
http://www.sgi.de/produkte/index.html
http://www.dec.de/products/Alphaserver/Alphaserver.html
http://www.hewlett-packard.de/enterprise/
http://www.de.ibm.com/go/general/produkte/produkte_hp.html
...
```