TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik | Institut Systemarchitektur | Professur Rechnernetze

WS 2012
LV Informatik-I für Verkehrsingenieure

2. Rechnerarchitektur
2.2 fortgeschrittene Computer

Dr. rer.nat. D. Gütter

Mail: Dietbert.Guetter@tu-dresden.de
WWW: wwwpub.zih.tu-dresden.de/~guetter/

Computer ab ca. 1964

Großrechner

Massendatenverarbeitung und wissenschaftliche Berechnungen
Stapelverarbeitung
max. Durchsatz und minimale Kosten

Arbeitsstationen

Softwareentwicklung und wissenschaftliche Berechnungen
Dialogverarbeitung
kurze Bearbeitungsdauer, dialogorientierte Bedienerschnittstelle

Prozeßrechner

Steuerung von technologischen Prozessen
Echtzeitverarbeitung
garantierte Reaktionszeiten auf äußere Ereignisse

Privilegien

- **privilegierter Status** für die Betriebssystemroutinen

  → voller Zugriff auf den gesamten Hauptspeicher (u.a. Ressourcen)

- … evtl. mehrere Privilegierungsstufen

- **unprivilegierter Status** für die Anwendungsprogramme

  → Einschränkung des Hauptspeicherzugriffes auf "eigenen" Bereich (hardwaremäßig überwacht!)

Vorteil

Beim Lauf eines fehlerhaften Programmes ist das Überschreiben der Speicherbereiche anderer Prozesse unmöglich.

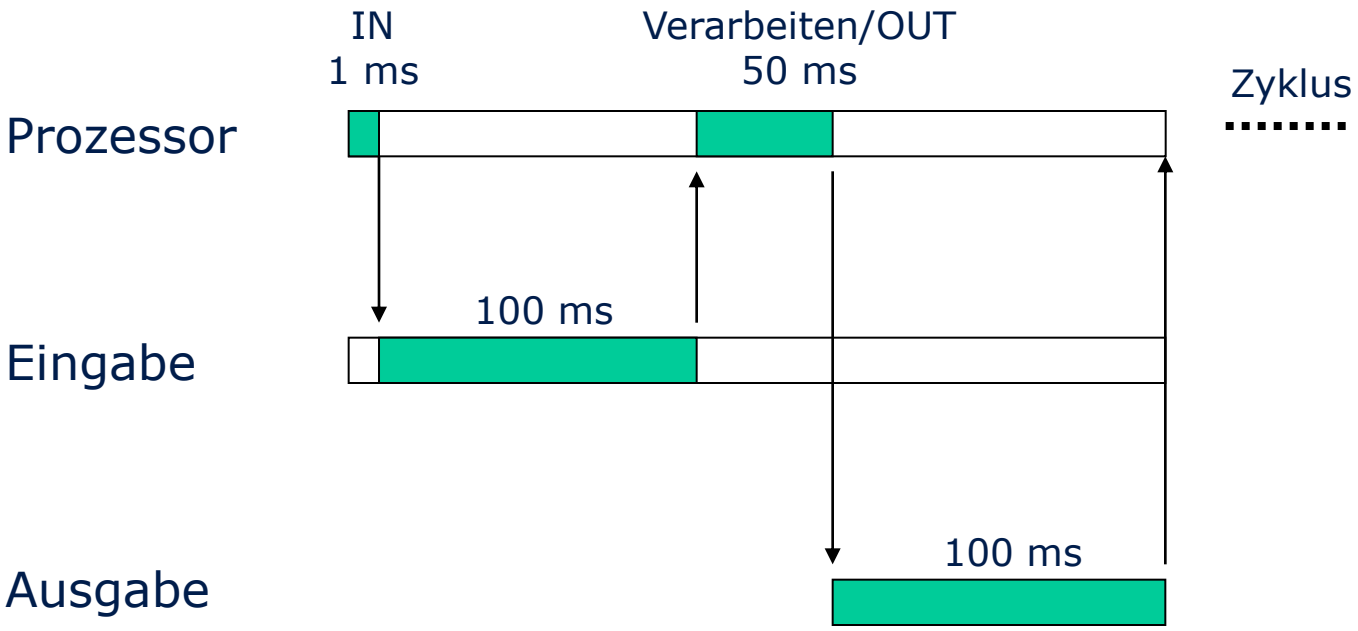
→ verringerte Absturzgefahr, erleichterte Fehleranalyse

Höhere Systemeffizienz

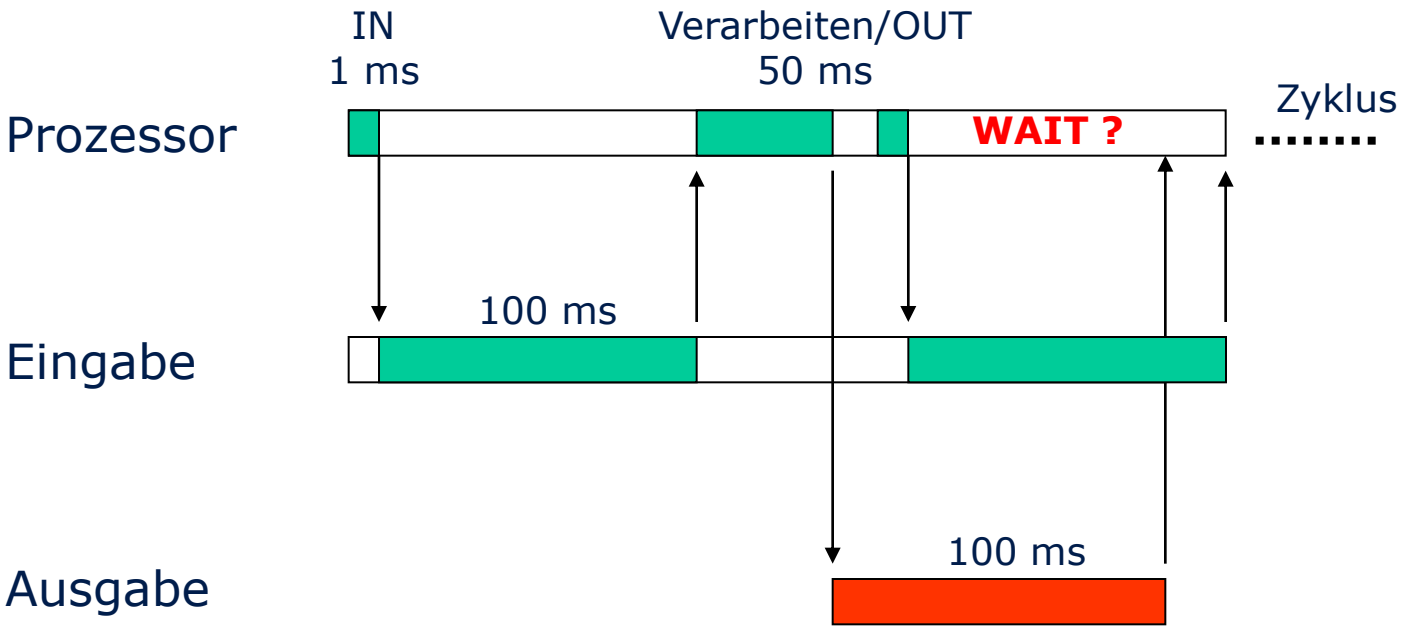
Optimierung für Multitasking, Echtzeitverarbeitung, …

- **komfortablerer Gerätezugriff**
  durch eine leistungsfähige Datenverwaltung
- **Verringerung von Wartezeiten**
  durch asynchrone Arbeit von ZVE und Peripherie
- **Reaktion auf äußere Ereignisse**
  durch ein Unterbrechungsbehandlungssystem
- **bessere Ressourcenauslastung**
  durch parallele Abarbeitung mehrerer Programme

Synchrone Arbeit



Asynchrone Arbeit



Cache

soll Datenzugriffszeit auf (relativ) langsame Geräte verkürzen

- Nutzung eines schnellen Pufferspeichers
- Verwaltung der Nachrichten im Pufferspeicher
- asynchrone Arbeit von Cachenutzer und Cacheverwalter

- **Vermeidung von Zugriffen** durch Arbeit mit gepufferten Datenkopien

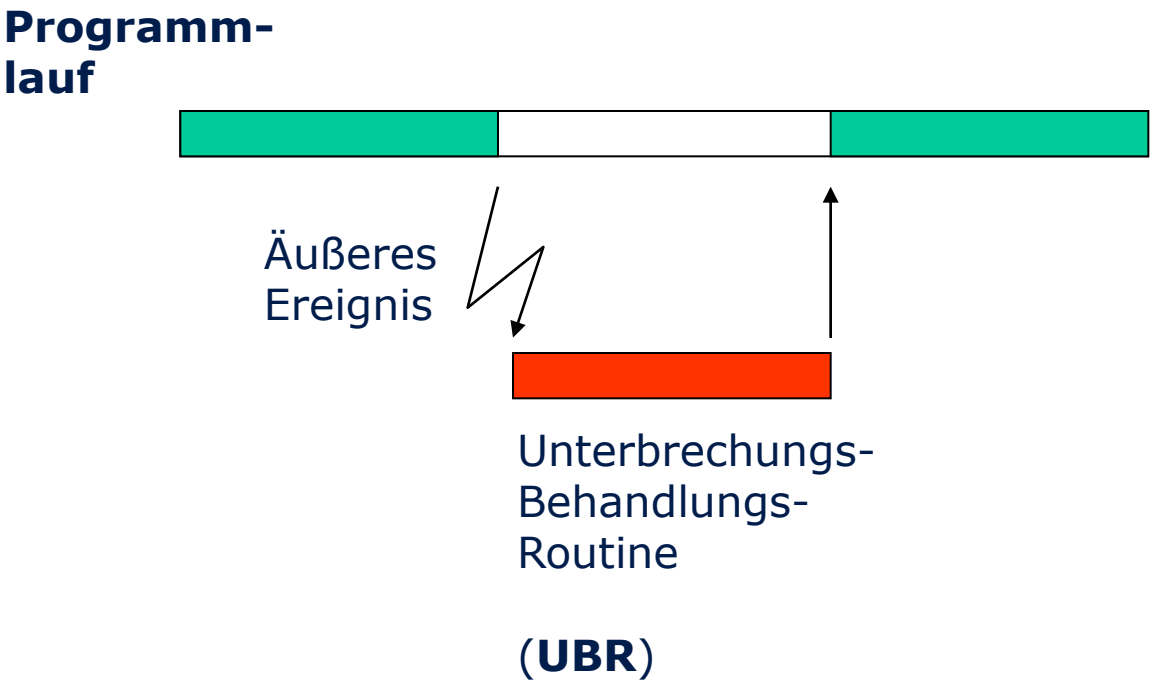
- Problem Konsistenz bei Schreib-Cache !
- Nutzung eines Cache's ist transparent

Anwendungsbeispiele

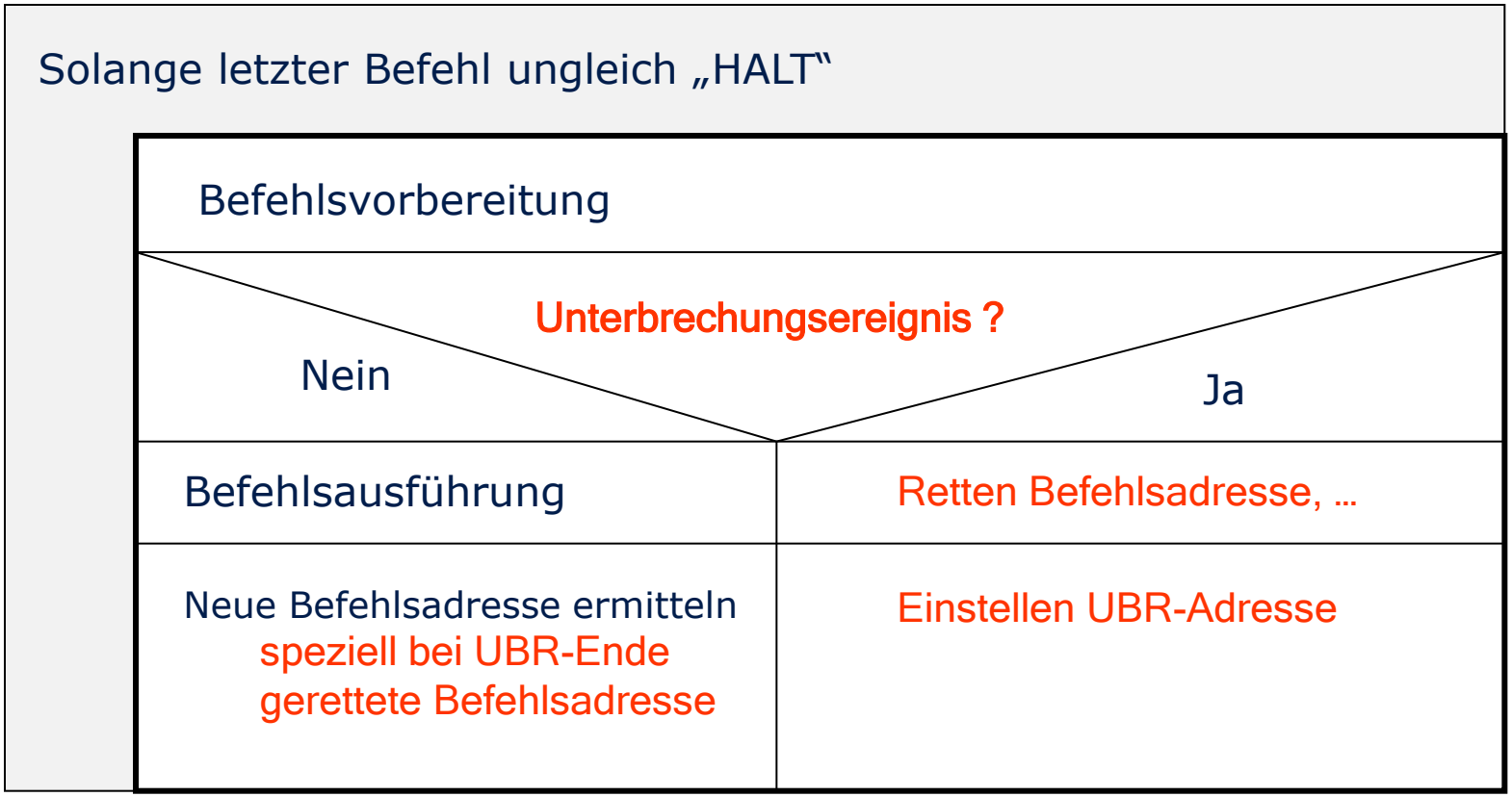
- Prozessorcache
- Festplattencache
- WWW-Seitencache

Unterbrechungsbehandlungssystem

Hardware erlaubt Prozessor Reaktion auf äußere Ereignisse (Unterbrechungen)



UBR-Behandlung durch Steuerwerk



Flaschenhals des v.Neumann-Automaten

Prozessoren verarbeiten nur einen Befehl in jedem Zeitpunkt
Erhöhung der Prozessorleistung nur möglich durch

- (leistungsfähigere Befehle)
  - Wortbreitenerhöhung
  - Mehroperandenverarbeitung
  - …
- höheren Systemtakt
  - physikalische Grenzen in Sicht
    z.Zt. Takt < 1 Nanosekunde

→ **andere Computerarchitekturen**
mit zeitparallelerer Bearbeitung mehrerer Befehle

Flynn - Einteilung Computer in vier Klassen

SISD (Single Instruction Single Data)

Ein Prozessor greift auf einen Speicher zu, evtl. prozessorinterne Pipeline

SIMD (Single Instruction Multiple Data)

Vektorrechner mit Array-Prozessoren
gleichzeitige Ausführung der Berechnungen mit verschiedenen Daten

MISD (Multiple Instruction Single Data)

mehrere Prozessoren bearbeiten gleichzeitig dieselben Daten,
praktisch ohne Bedeutung

MIMD (Multiple Instruction Multiple Data)

mehrere Prozessoren greifen auf unabhängige Speicher zu
zeitparallele Bearbeitung mehrerer Programme möglich

Prozessor - Pipeline

Die Abarbeitung eines Maschinenbefehls erfordert mehrere Takte,
→ Zeitverlust
z.B. während CPU-interner Arbeit ist Bus untätig

Pipeline: mehrere Befehle gleichzeitig im Prozessor
in verschiedenen Bearbeitungsstufen

z.B. Intel – Pentium 5 Befehle in einer Pipeline möglich

- Befehlsholstufe Holen Befehlen aus Prozessorcache
- Dekodierstufe 1 Auswerten Befehlskode;
- Dekodierstufe 2 Bestimmung der Operandenadressen
- Ausführungsstufe Bearbeitung in Arithmetisch-Logischer Einheit (ALU)
- Schreibstufe Operationsergebnisse abspeichern

Stau-Probleme: ungleiche Taktzahl bei Befehlen
abhängige Befehlsfolgen

Prozessoren mit mehreren Kernen

Prozessor kann aus mehreren Subprozessoren (Kernen) bestehen
→ gleichzeitige Abarbeitung mehrerer Programme
gleichzeitige Abarbeitung vieler Befehle eines Programmes

Prozessor

- Aufteilung der Programmbefehlsfolgen auf mehrere Kerne
- Parallele Arbeit in den Pipelines der Kerne

Probleme

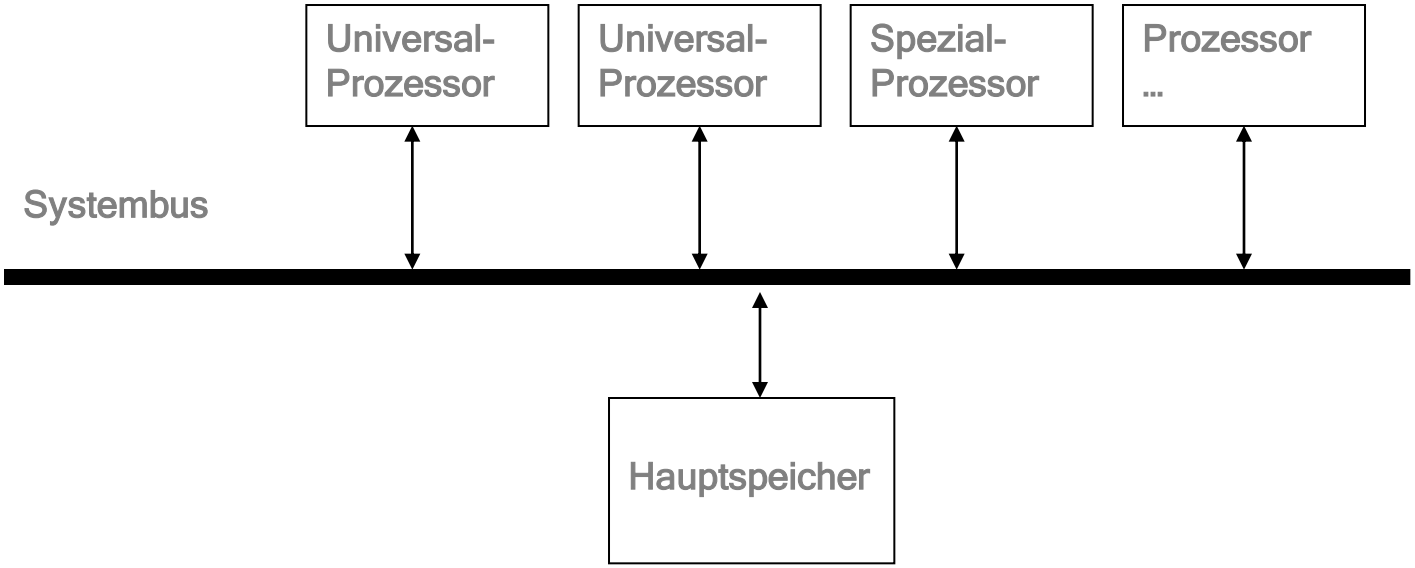
- Stau / Konsistenz / Synchronisation
- Energieaufnahme, Wärme, …

Normaler Programmkode ungeeignet
→ Compiler müssen optimierten Kode erzeugen

Mehrprozessorsysteme

Systeme können mehrere Prozessoren besitzen
→ gleichzeitige Abarbeitung mehrerer Programme
gleichzeitige Abarbeitung vieler Befehle eines Programmes

Problem: Systembus wird evtl. zum Engpaß



Prozessor - Cache

Schneller interner Speicher im Prozessor
zur Aufzeichnung der letzten Hauptspeicherzugriffe

Adresse	Inhalt
…	…

Lese-Cache — HS-Inhalte möglichst vom schnellen Cache lesen
nicht vom (langsameren) Hauptspeicher
Zeitersparnis, außerdem Busentlastung(!)

Schreib-Cache — Prozessor schreibt Daten in Cache, nicht in HS
danach asynchroner Transport in HS
Zeiteinsparung durch Parallelabeit

Problem — **Datenkonsistenz** in Mehrprozessorsystemen
Cache-Inhalt u.U. nicht aktuell,
wenn andere Prozessoren den HS-Inhalt ändern

Prozessor Cache - Nutzungsbeispiele

- **Stack im Cache**

  Stack-Daten werden eigentlich nur im Prozessor benötigt

  → Busentlastung
  wesentliche Erhöhung der Abarbeitungsgeschwindigkeit

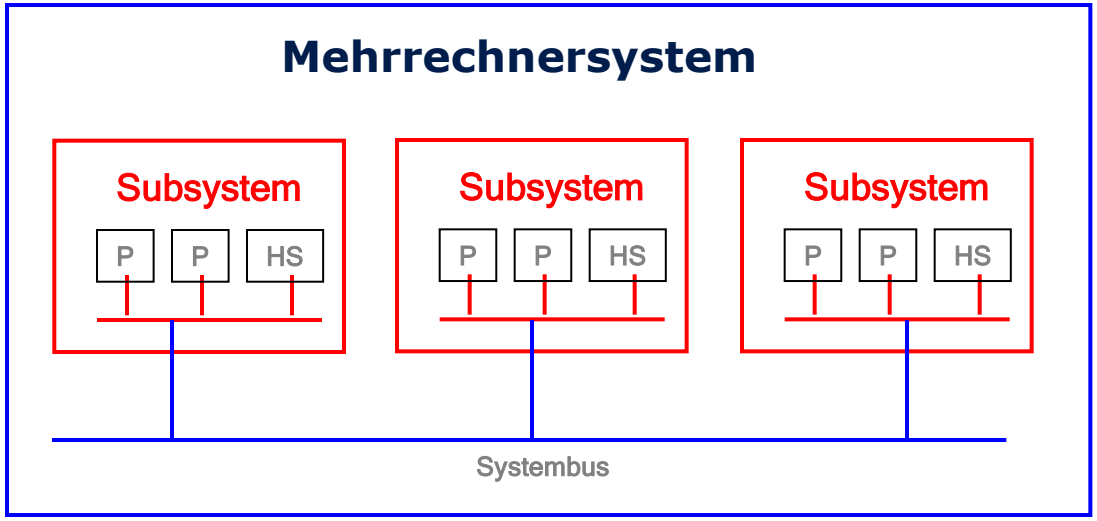
- **Zyklus** mit 100 Durchläufe mit je 100 Befehlen

  → Lesen 100 Befehle aus HS beim ersten Durchlauf
  (danach Kopien aller Befehle im Cache)

  99 weitere Zyklen kein HS-Zugriff mehr erforderlich
  zum Lesen der Befehle

Mehrrechnersysteme

bestehen aus mehreren autonom lauffähigen Recheneinheiten



kein Busengpaß → massiv-parallele Systeme möglich (Superrechner), mit evtl. tausenden Subsystemen

Schwierig: Aufteilung der Lösung eines Problems auf viele parallele Systeme

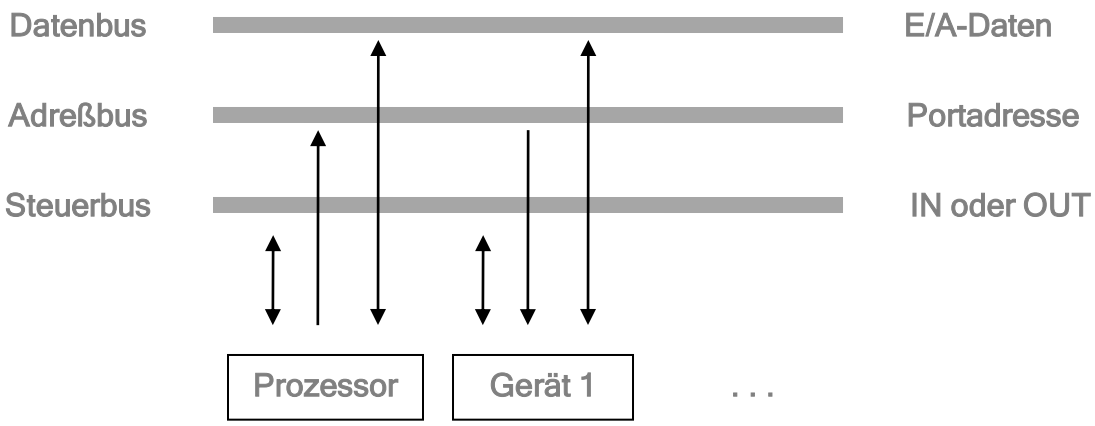
EA-Hardware bei PC-Architektur (vereinfacht)

Zugriff auf E/A-Geräte ähnlich wie Zugriff auf Hauptspeicher

- Initiative geht immer vom Prozessor aus,
- Geräte beobachten Bus und reagieren, wenn sie angesprochen werden.

Prozessor gibt Signale auf **Steuerbus**,

- Entscheidung Hauptspeicherzugriff oder E/A-Operation
- Lesen oderSchreiben
- am Adreßbus liegt Hauptspeicher- bzw. Geräteadresse (Portadresse) an.
- über Datenbus werden die Daten vom/zum Ziel übertragen.



Gerätezugriff

jedes E/A-Gerät muß eine eindeutige Portadresse besitzen
einige Geräte sind sind Bestandteil der PC-Architektur,
deren Portadressen sind durch Industriestandards festgelegt

Beispiel:

Zugriff auf CMOS-RAM der batteriegepufferten Echtzeituhr
(gehört zur PC-Ausstattung, belegt u.a. Ports 70H und 71H)

E/A-Befehle für Lesen der aktuellen Stunde des laufenden Tages

```
MOV  AL,04H   ;Position des Parameters im CMOS-RAM
OUT  70H,AL   ;Ausgabe der Position an Echtzeituhr
IN     AL,71H   ;Eingabe der aktuellen Stunde (BCD-Format) von Echtzeituhr
```

Entsprechend könnten auch Minute und Sekunde gelesen werden.

Treiber - Aufgaben

- **Schnittstellen**bereitstellung für Betriebssystem (s. Kap.3)
  zur Nutzung aller Funktionen realer E/A-Geräte

- **möglichst einheitliche Form** des Aufrufes von Treiber-Routinen
  Funktionen und Parameter gerätespezifisch

- **Synchronisation**
  der Zusammenarbeit zwischen Computer und Geräten

- Arbeit im **privilegiert**en Status

- Dienstleistungsniveau der Treiber
  oft über dem Aufgabenspektrum realer Geräte

  → Zugriff auf **virtuelle Geräte** mit verbesserten Eigenschaften

Beispiele für gerätespezifische Treiberfunktionen

Grafikkarte

- Lesen/Schreiben ASCII-Zeichen an Zeilen-/Spaltenposition
- Lesen/Schreiben Farbpunkt an Pixelposition
- Schreiben vorgegebener Grafikelemente, z.B. eines Kreises
- *Rollen, mehrere virtuelle Fenster*

Drucker

- Ausgabe Druckzeichen; Ausgabe Pixel
- Test Ausgabestatus, ggf. Unterbrechungssignal bei Ausgabe-Ende
- *Kantenglättung, Emulation Postscriptdruck*

Magnetplattengerät

- Steuerung des Gerätes, z.B. Positionieren Magnetköpfe
- Lesen/Schreiben/Löschen von Datenblöcken
- E/A-Statusabfrage, ggf. Unterbrechungssignal bei E/A-Ende
- *Lese-/Schreib-Cache*

Unterbrechungssystem Intel-Architektur

Polling zeitzyklisches Abfragen eines EA-Gerätes, z.B. Feststellen Alarmzeit
nicht sehr zweckmäßig, weil hohe Systembelastung
→ wünschenswerte asynchrone Arbeit von Prozessor/Gerät

Unterbrechungssystem
hat prinzipiell 256 verschiedene Unterbrechungen,
die durch eine 8-bit-Interruptnummer bezeichnet werden.
Adressen der UB-Behandlungsroutinen am Anfang des Hauptspeichers (jeweils 4 Byte)

Die ersten fünf Interrupts sind hardwaremäßig festgelegt (interne UB)

INT 0	bei Division durch Null
INT 1	nach jedem Befehl bei Einzelschrittbetrieb (für Testzwecke)
INT 2	bei NMI-Signal (Nichtmaskierbarer Interrupt) am Prozessor für externe Unterbrechungen höchster Priorität, z.B. bei Spannungsausfall
INT 3	bei Erreichen eines Haltepunktes im Testbetrieb
INT 4	bei Datenüberlauf nach Rechenoperationen

Unterbrechungsbehandlung

interne Unterbrechungen bei Ausnahmeereignissen

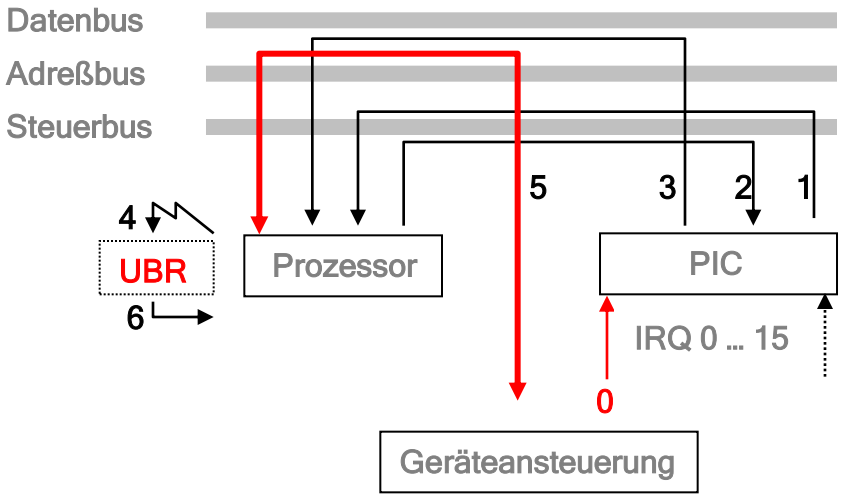
- Division durch Null
- Spannungsausfall

externe Unterbrechungen bei asynchroner Arbeit mit Geräten

- Initialisierung einer Unterbrechungsbehandlungsroutine
- Starten EA-Operation
- Warten auf Unterbrechung (E/A-Ende-Meldung vom Gerät)
- Reaktion auf Unterbrechung mit Interruptbehandlung
- Ende EA anzeigen

Externe Unterbrechungen in Intel-Architektur

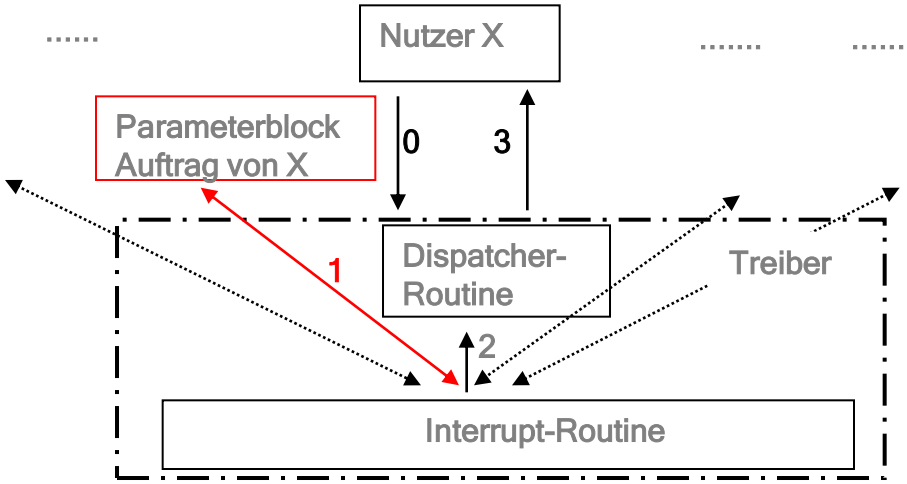
0. Gerät meldet UB (IRQ 0 bis IRQ 15) an beim Interrupt-Controller PIC
1. PIC gibt Signal über Steuerbus zum Prozessor.
   (Sammel-)Unterbrechung der Befehlsabarbeitung im Prozessor
2. Prozessor quittiert
3. liest anschließend die zum Ereignis gehörende UB-Nummer vom PIC.
4. Aufruf Unterbrechungsbehandlungroutine UBR
5. Innerhalb von UBR Datenaustausch mit Gerät
6. Nach UBR-Ende Fortsetzung des unterbrochenen Programms



Innerhalb UBR
Retten/Wiederherstellen aller Parameter des unterbrochenen Programmes

Treiber – Nutzung für E/A-Operationen

0. Auftragsübergabe durch Nutzer an Dispatcherroutine
   Anlegen Parameterblock
   Nutzer geht in Wartezustand (Abfrageschleife E/A-Ende)
1. Interruptroutine liest die Auftragsparameter aus Parameterblock,
   tauscht Nachrichten mit Geräten aus und
   trägt Zwischenergebnisse bis Bearbeitungsende in Parameterblock ein
2. Rückmeldung an Dispatcherroutine
3. Fertigmeldung an Nutzer, Wartezustand beendet



Moderne Betriebssysteme
nutzen die Wartezeit für andere parallele Aufgaben (mehrere Nutzer).

Treiber Tastatur

Tastatur

- enthält eigenen spezialisierten Prozessor, Hauptspeicher, …
  bei Tastendruck wird gespeichert:
  Zeile/Spalte im Tastaturfeld,
  ggf. für mehrere gleichzeitig gedrückte Tasten

- Übergabe der Eingaben an Hauptprozessor/Treiber, (Interrupt, …)
  Puffern der Eingaben bis zur Abnahme

Treiber

- Zuordnung Tasten zu Zeichen des nationalen Alphabets
  Berücksichtigung von Umschalttasten, z.B. Groß-/Kleinbuchstabe, …
  und Mehrtastenbetätigung, z.B. „AltGr"+"Q" wird „@"

- Eliminieren „geprellter" Tasteneingaben (Zeitüberwachung, …)
  Problem: Tasten federn meist nach, dann
  scheinbar mehrere Zeicheneingaben schnell hintereinander

Treiber Grafikkarte

Bildwiederholspeicher BWSP (meist intern in Grafikkarte)

- 2-dimensionale Einteilung in Zeilen und Spalten
  für Zeichen (Textmodus) bzw. Pixel (Grafikmodus)

- Bildschirmdarstellung durch Hardware der Grafikkarte
  häufig RGB-Darstellung (24 bit, je 8 für Rot/Grün/Blau)
  BWSP-Größe bestimmt erzielbare Auflösung
  z.B. 1024x768 Pixel a´ 24 bit → 2 359 296 Byte

- Betriebssystem schreibt Darstellungsinformationen in BWSP
  kartenspezifische Treiber erforderlich, aber
  einheitliche Treiber-Schnittstelle entsprechend Betriebssystem

Zusatzfunktionen der Treiber (Unterstützung durch Grafikkartenhardware)

- Rollen von Zeilen im Textmodus
- Unterstützung der Arbeit mit Bildschirmfenstern
- Grafikfunktionen, wie
  Zeichnen geometrischer Figuren, Drehbewegungen usw.

Treiber Drucker

Arbeitsweise

- logisch ähnlich Grafikkarte, ebenfalls Text- und Grafikmodus
  zusätzlich Synchronisation, da Drucker vergleichsweise langsam

- Arbeitsteilung zwischen Treiber und Drucker
  interne Arbeit der Treiber sehr modellspezifisch

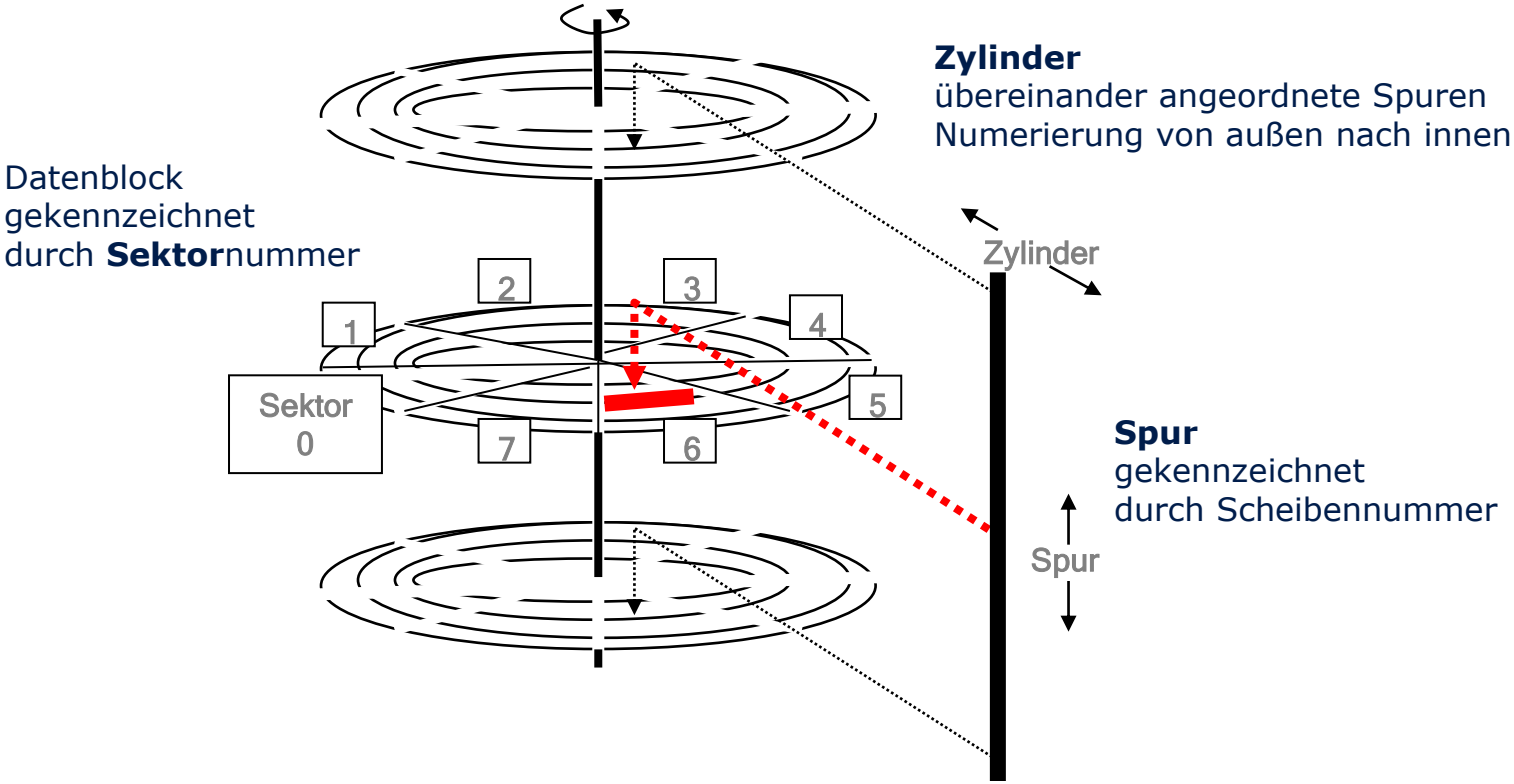
- Nutzerfunktionen

  - zur Zeichen-/Pixelausgabe (interne Zeichensätze/Schriften)
  - zur Ausgabe geometrischer Figuren
  - zum Abfragen des Druckerstatus (abnahmebereit ja/nein)
  - evtl. Unterstützung von Seitenbeschreibungssprachen
    Postscript, pdf, …

- Interne Funktionen
  z.B. Kantenglättung durch Interpolation
  Vermeidung unschöner Treppenstufen

Magnetplattengeräte

bestehen aus übereinander angeordneten rotierenden Magnetscheiben,
auf denen sich Datenblöcke in konzentrischen Kreisen (Spuren)
mit einer festen Anzahl von Sektoren befinden.



Treiber Magnetplatte

Aufgaben

- Direktzugriff (Lesen/Schreiben) organisieren
  adressierte Datenblöcke (Zylinder-, Spur- und Sektornummer)
- Zugriffssteuerung durch Zusammenarbeit Treiber/Hardware
  Zugriffskamm an richtige Zylinderposition,
  auswählen Magnetkopf entsprechend Spurnummer,
  warten bis sich durch Rotation der angegebene Sektor nähert,
  Lesen/Überschreiben Datenblockinhalt
- Cache für Festplattenzugriff organisieren
- Zugriffszeit nicht exakt vorhersehbar
  Bewegung des Zugriffskammes sehr zeitaufwendig,
  warten auf Sektor dauert durchschnittlich eine halbe Umlaufdauer.
- Initialisierung bzw. Formatierung erforderlich
  Magnetplatte beschreiben mit adressierbaren, aber „leeren" Blöcken

RAID (Redundant Arrays of Inexpensive Disks)

Zusammenfassung mehrerer Platten zu Arrays (RAID)
Zugriff wie auf eine Platte, organisiert durch Treiber und Hardware
6 RAID Niveaustufen

RAID-0

- Unterteilung der Datenträger in gleichgroße Stücke (chunks)
- stückchenweise Abspeicherung auf verschiedenen Platten (stripes)
- Erhöhung der Lese-/Schreibgeschwindigkeit
- keine Redundanz

	Platte 1	**Platte 2**	**Platte 3**
Streifen 1	Daten A1	Daten A2	Daten A3
Streifen 2	Daten A4	Daten A5	Daten A6
...	...	...	...

RAID (Redundant Arrays of Inexpensive Disks)

RAID-1

- Daten werden gespiegelt (Schreiben auf zwei Platten)
- Schreiben wird dadurch evtl. etwas langsamer
- Lesen kann schneller erfolgen als bei einer Einzelplatte, da auf beiden Platten gleichzeitig gelesen werden kann
- Plattenausfall → kein Datenverlust (Nutzung der zweiten Platte)
- hohe Fehlertoleranz erkauft durch doppelten Speicheraufwand

RAID-2 / RAID-3 mit redundanter Checksequenz zur Fehlererkennung
(zusätzliche Platte bei RAID-3), Fehlerkorrektur oft möglich

RAID-4 / Raid-5 / RAID-6
zusätzliche Platten, Checksequenz in Streifen,
ab RAID-5 Checksequenz verteilt auf alle Platten

1 Plattenausfall kann kompensiert werden,
bei RAID-6 sogar 2 Ausfälle

RAID (Redundant Arrays of Inexpensive Disks)

RAID-4

- wie RAID-0; eine zusätzliche *redundante* Platte
- redundante Platte enthält pro Streifen eine Kontrollsequenz
- Bildung der Kontrollsequenz durch bitweises Exclusiv-Oder
- bei Ausfall *einer* beliebigen Platte kein Datenverlust

	Platte 1	**Platte 2**	**Platte 3**	**Parität**
Streifen 1	1010...	0101...	0011...	1100...

Beispiel: Ausfall Platte 2

	1010	Chunk Platte 1
exo	0011	Chunk Platte 3
exo	1100	Chunk Paritätsplatte
→	0101	Chunk Platte2 (rekonstruiert)

Speicherschutz (für Hauptspeicher)

Speicherschutz erforderlich,
wegen evtl. Speicherbereichsüberschreitungen bei Programmlauf

- Schreiben - Ursache von Störungen/Abstürzen usw.
- Lesen - Datenschutzprobleme usw.

Zugriffsschutz erfordert Hardwareunterstützung

z.B.

- Zuordnung von Speicherschutzschlüsseln für Programme (zur Laufzeit)

- bei HS-Zugriff Kontrolle des Speicherschutzschlüssels ggf. Fehleranzeige „illegaler Speicherzugriff“

Virtueller Speicher

Hauptspeichernutzung vereinfacht sich,
wenn die Computerhardware **gestreute Adressierung** unterstützt.

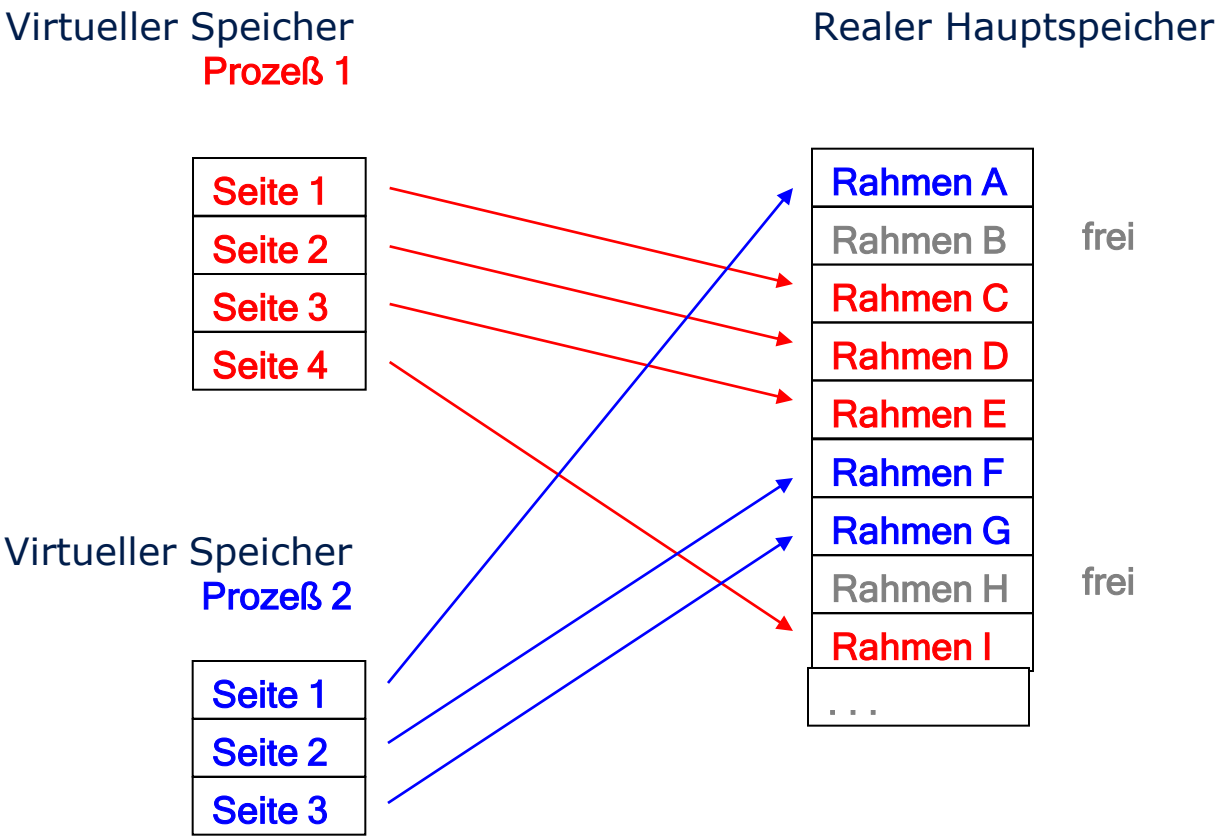
- Maschinenbefehle
  Zugriff nicht auf realen Hauptspeicher
  sondern auf virtuellen Speicher

- Virtueller Adreßraum
  Einteilung in **Seiten** (meist) fester Größe

- Realer Speicher
  Einteilung in **Rahmen** (Größe gleich Seitengröße)

- Laden
  Seiten werden in Rahmen „eingespannt".
  Kein zusammenhängender Speicherbereich erforderlich, da
  Befehlsabarbeitung in virtueller(!) Adreßreihenfolge erfolgt.

Gestreute Adressierung



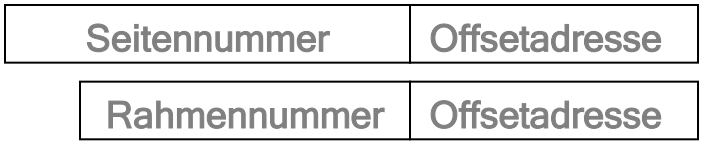
Virtuelle Adressen

virtuelle Adressen werden innerhalb der Seite unterteilt in

- Seitennummer
- Relativadresse (Offsetadresse)
  Seitengröße 2^m Byte → Offsetadressen jeweils m Bit

realen Adressen werden entsprechend unterteilt in

- Rahmennummer
- Offsetadresse



Speicherzugriffe zweistufig (transparent zur Laufzeit)

1. Logischer Zugriff auf virtuelle Adresse
   Abbildung Seitennummer auf Rahmennummer
   (wesentlicher Zeitmehraufwand nur bei Seitenwechsel !)
2. Zugriff auf Realspeicher

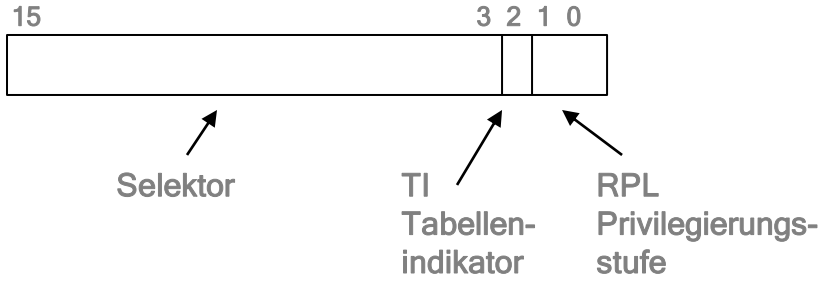
Intel Pentium - Hauptspeicherorganisation

Pentium: Weiterentwicklung I8086 ff. zur 32-Bit-Architektur

z.B. AX (16-bit-Rechenregister) → EAX (32-bit-Register)
CS (16-bit-Segmentregister) → CS (16-bit-Register)
…
4 Privilegierungsstufen

- Real Mode (s. Kap.2.1; 16-bit-Adressierungstechnik I8086)
- **Protected Mode** mit Segment-Swapping oder Paging
  je nach Einstellung im Prozessor

Segmentregister beinhalten keine Segmentadressen, sondern **Selektoren** (Zeiger auf Segment-Deskriptoren)



Deskriptortabellen

TI zeigt alternativ auf

GDT (globale Deskriptortabelle)	für alle Task
LDT (lokale Deskriptortabelle)	für aktive Task

Selektor zeigt auf Segment-Deskriptoreintrag (8 byte) in GDT/LDT

Segment-Deskriptor enthält u.a

- Segment im Hauptspeicher ja/nein
- Basisadresse Segment (32 bit)
- Größe Segment (20 bit)
  1 MByte bzw. 4 GByte je nach Granularität (Byte bzw. 4KByte)

2^{13} Selektoren in GDT und in LDT, d.h. max. 16384 Segmente
→ Taskgröße bis 64 Tbyte

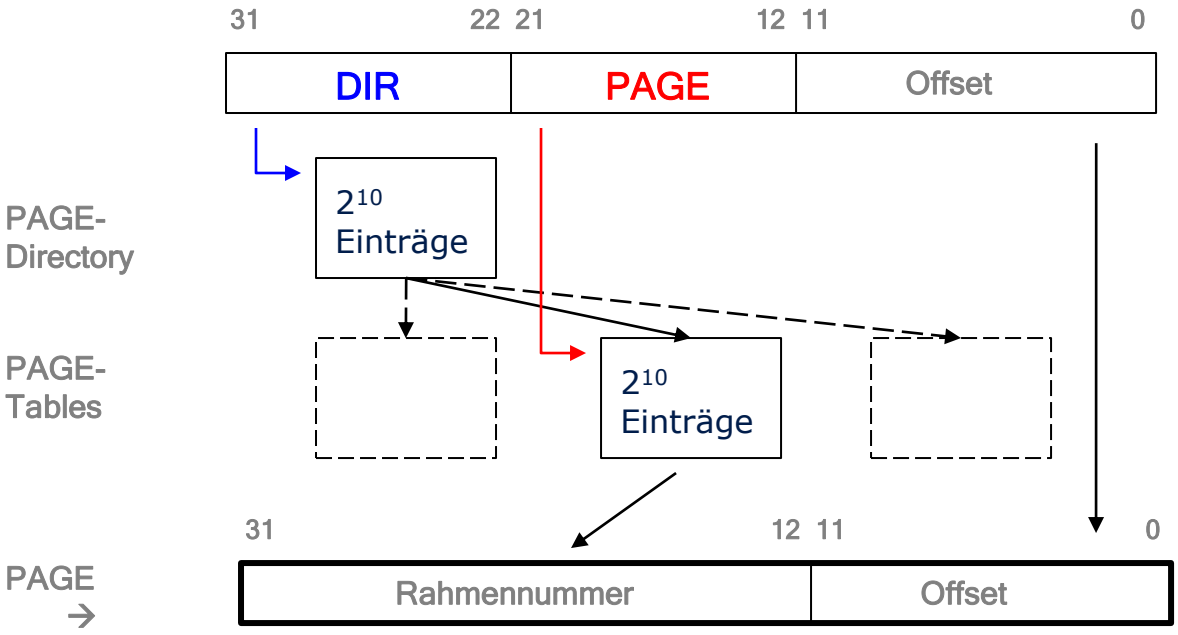
Swapping Ein-/Auslagern kompletter Segmente je nach Bedarf
praktische Größenbeschränkung durch Realspeicher

Paging

andere Interpretation des Segmentdeskriptors, falls Paging eingestellt
→ **lineare 32-bit-Adresse** (anstelle Segmentanfangsadresse)

2-stufige Seitentabellen (Hauptspeichereinsparung)

DIR	Selektor für Page-Directory	→ Adresse Page-Table (20 bit + 12 x "0")
PAGE	Selektor für Page-Table	→ Rahmennummer (20 bit)

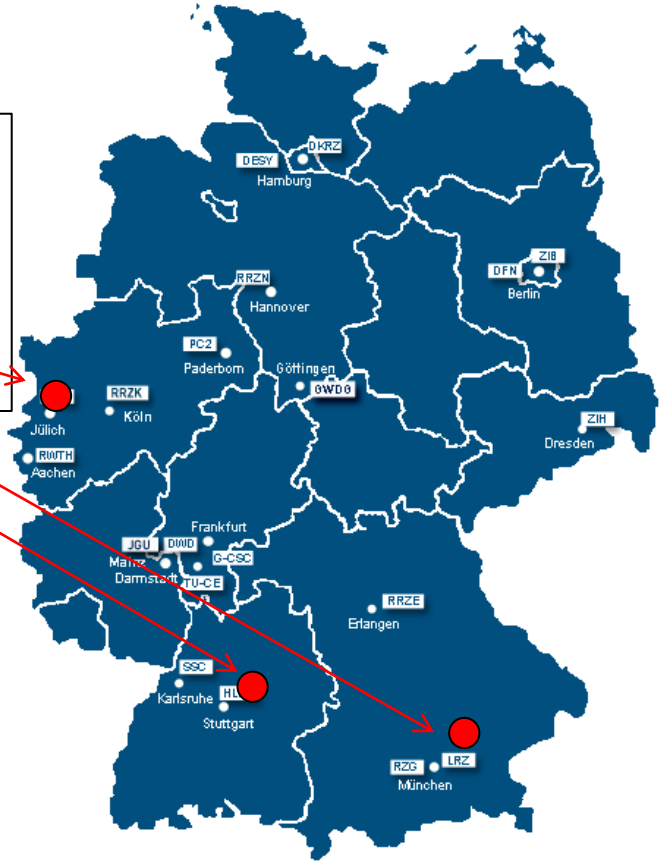


Hochleistungsrechnen als Schlüsseltechnologie

Gauß-Allianz Deutschland
(http://www.gauss-allianz.de/de/ueber-die-gauss-allianz/mitglieder)

GCS
(Gauss Centre for Supercomputing)

3 nationale Zentren
- Jülich
- Garching
- Stuttgart

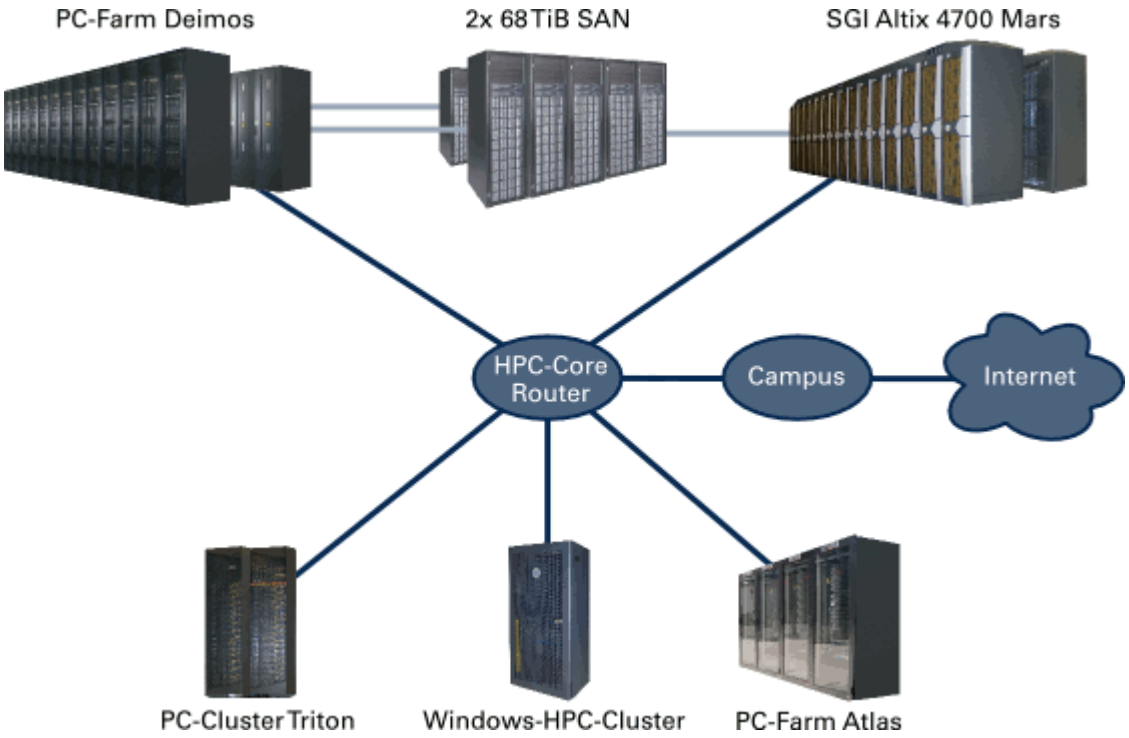


Weitere Mitglieder

- RWTH Aachen
- …
- TU Dresden
- …

Hochleistungsrechnen an der TU Dresden

Zentrum für Informationsdienste und Hochleistungsrechnen (**ZIH**)



(http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/hpc)

Hochleistungsrechner

SGI Altix 4700 TU Dresden, ZIH

- 2007 installiert (2013 Nachfolger geplant)
- 2.048 Intel Itanium (Dual Core) → 11.9 TFlops/s
- 6,5 TByte Hauptspeicher
- 50 000 Prozessorstunden täglich
- Betriebssystem „Novell SUSE Linux Enterprise Server 10"
- hoher Energieverbrauch

zusätzlich mehrstufige Massenspeicherung (SAN)

- mit SGI InfiniteStorage Data Migration Facility (DMF) Anbindung an SGI Altix mit 8 Gbyte/s
- 136 TByte Plattenspeicher
- Archivierung durch Bandsystem (1 PByte)

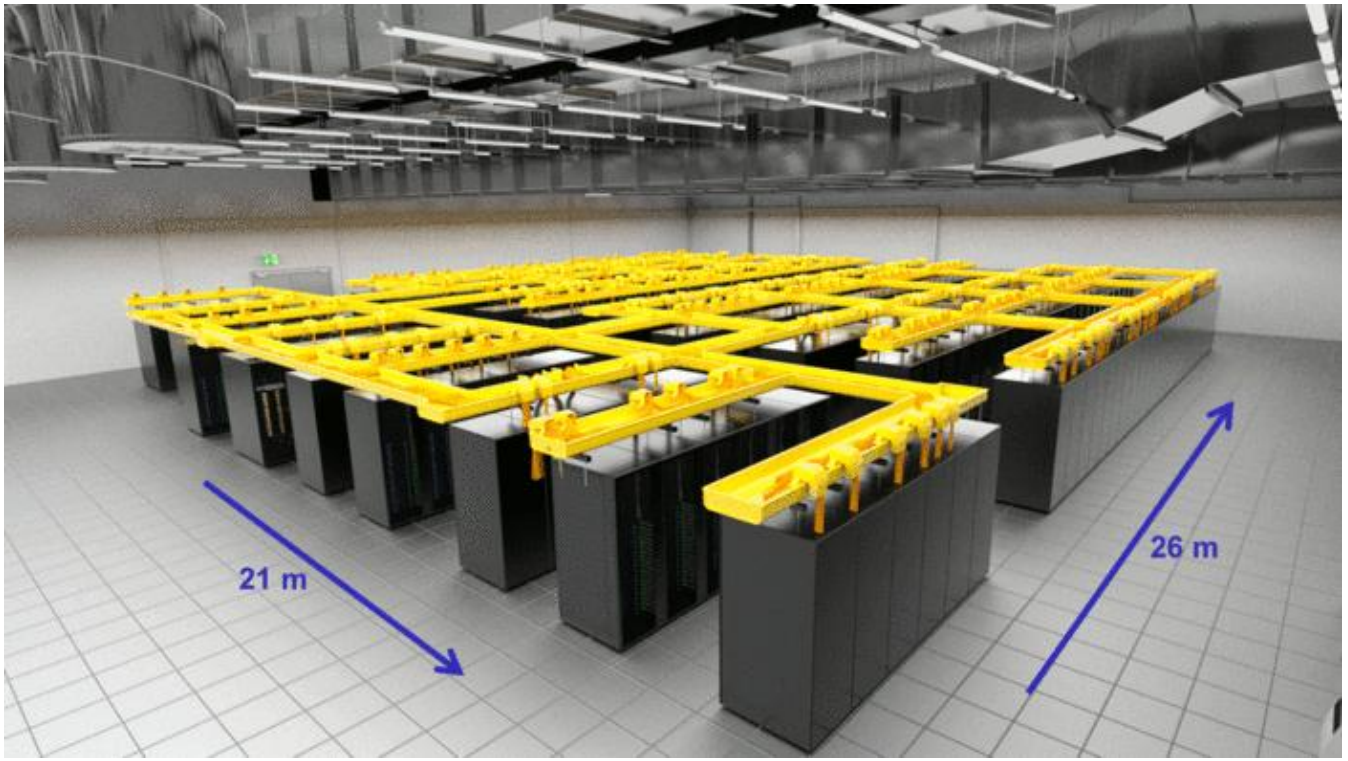
LRZ (Leibniz-Rechenzentrum) Garching/München

31.07.2006 Inbetriebnahme SGI Altix 4700

(Video: http://www.youtube.com/watch?v=MI38ZJhASJw)

LRZ (2)

20.7.2012 SuperMUC
3 PFlops/s (1. Platz in Europa, 4. Platz in der Welt)



(http://www.lrz.de/services/compute/supermuc/systemdescription/)