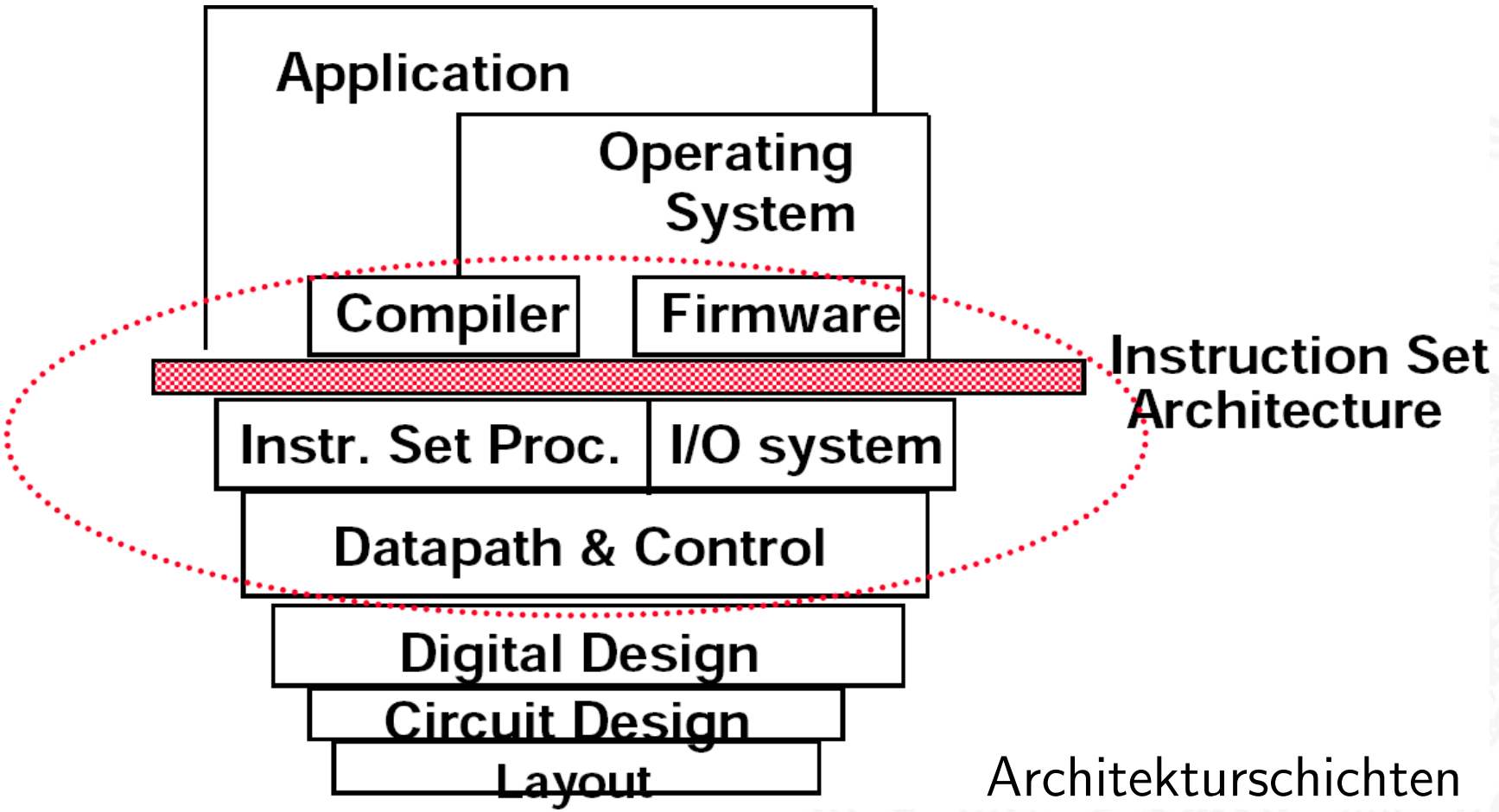
Gliederung

Schnittstellen zu anderen Inhalten des Studiums



Architekturschichten

Schnittstellen zu anderen Inhalten des Studiums (cont.)

Sehr starke Wechselwirkungen der Rechnerarchitektur mit anderen Bereichen der Informatik

- Rechnerorganisation
- Betriebssysteme
- Compilerbau
- Warteschlangentheorie

Der Bereich der Rechnerarchitektur liefert genug Stoff für mehrere eigene Vorlesungen, deshalb kann hier nur einführende Übersicht folgen

Schnittstellen zu anderen Inhalten des Studiums (cont.)

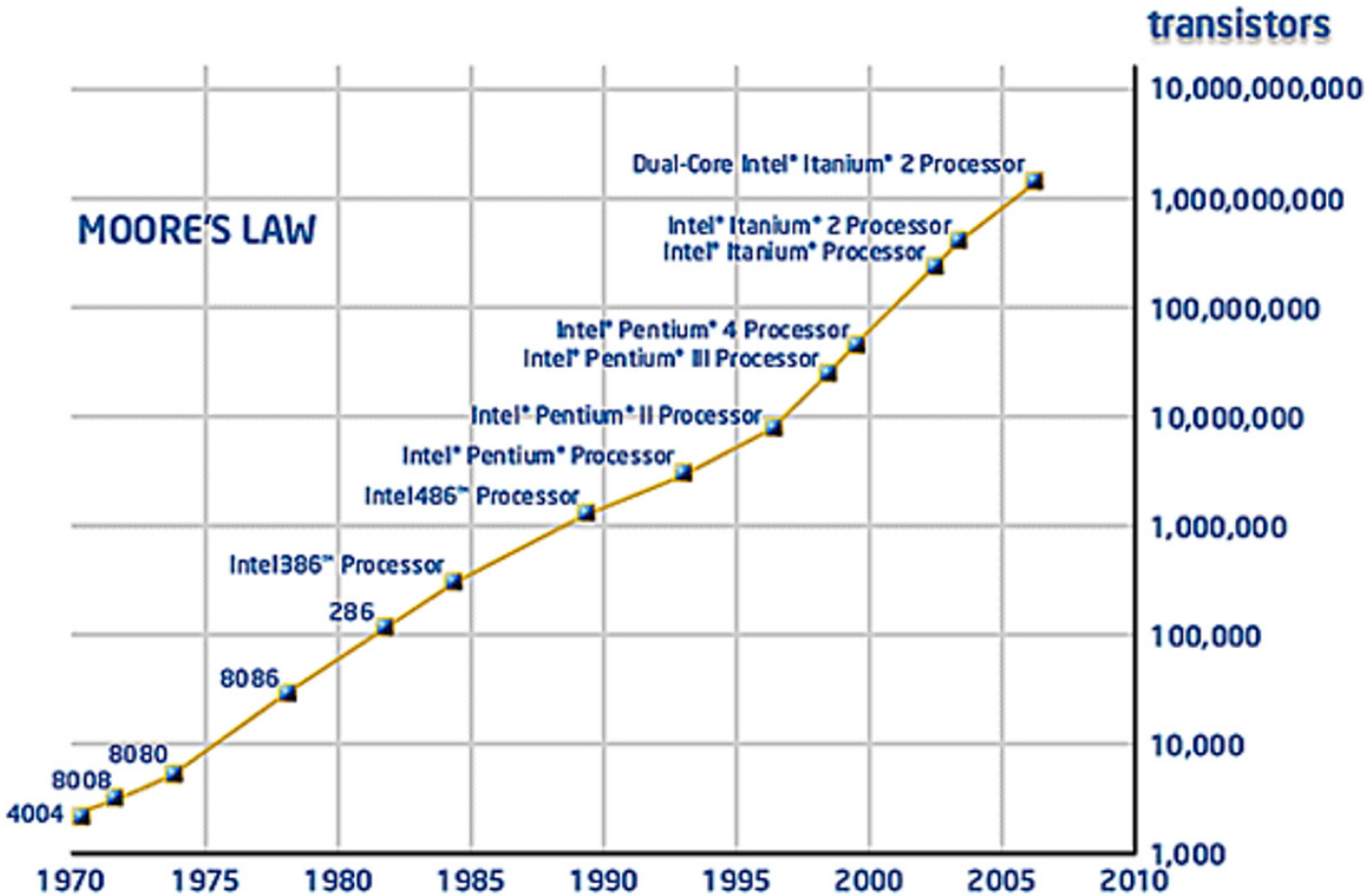
Literatur – besonders diese

- D. Patterson, J. Hennessy: *Rechnerorganisation und -entwurf; Die Hardware/Software-Schnittstelle* [PH05]
- D. Patterson, J. Hennessy: *Computer Organization and Design; The Hardware/Software Interface* [PH04]
- J. Hennessy, D. Patterson: *Rechnerarchitektur; Analyse, Entwurf, Implementierung, Bewertung* [HP94]
- J. Hennessy, D. Patterson: *Computer architecture; A quantitative approach* [HP06]
- A. Tanenbaum: *Computerarchitektur; Strukturen, Konzepte, Grundlagen* [Tan06]

Schnittstellen zu anderen Inhalten des Studiums (cont.)

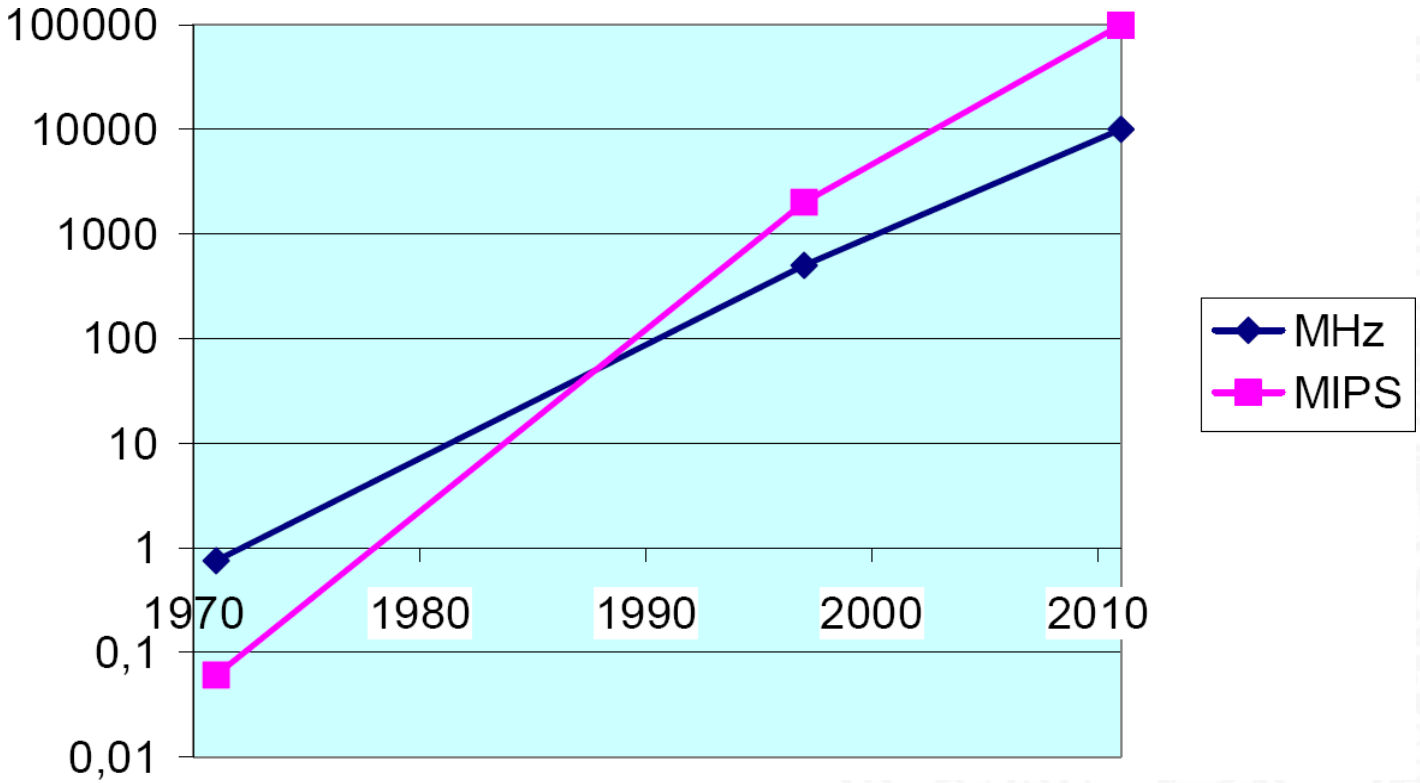
- `http://de.wikipedia.org` und `http://en.wikipedia.org`
- C. Märtin: *Einführung in die Rechnerarchitektur: Prozessoren und Systeme* [Mär03]
- D. Comer: *Essentials of Computer Architecture* [Com05]
- W. Oberschelp, G. Vossen: *Rechneraufbau und Rechnerstrukturen* [OV06]
- M. Dal Cin: *Rechnerarchitektur* [DC96]

Was tun mit Moore's Law?



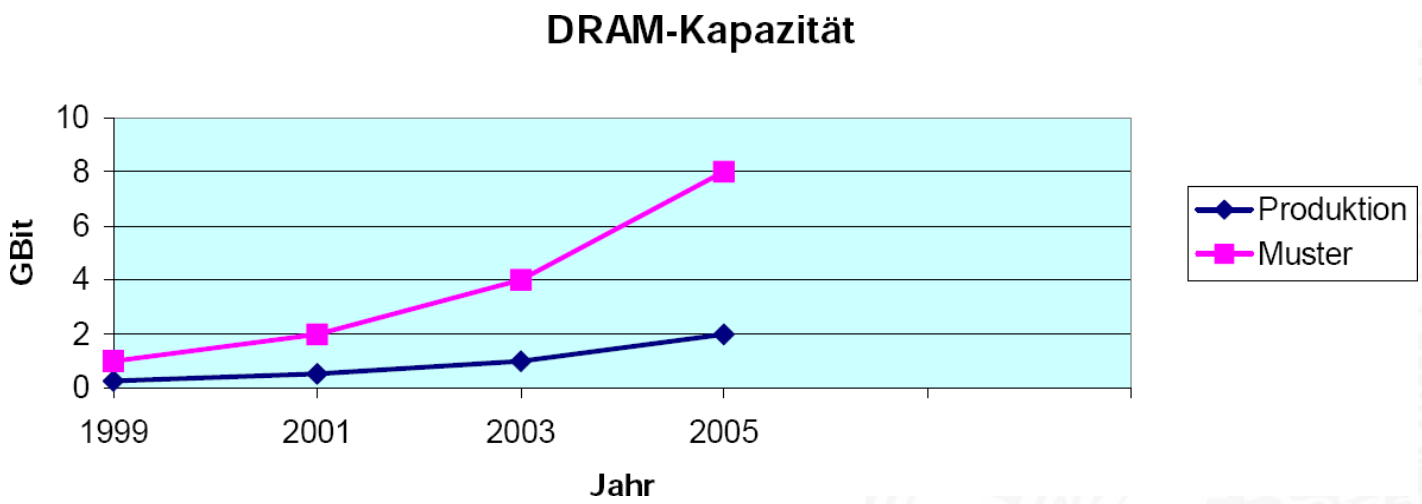
Was tun mit Moore's Law? (cont.)

- Universalrechner: CPUs, Mikrocontroller...



Was tun mit Moore's Law? (cont.)

- reguläre Strukturen: alle Arten von RAM



- . . . im Folgenden geht es um die Frage, nach welchen Prinzipien Prozessoren aufgebaut sind und damit: wie man sie entwirft

Was ist Rechnerarchitektur?

Definitionen

1. *The term architecture is used here to describe the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behaviour, as distinct from the organization and data flow and control, the logical and the physical implementation.* [Amdahl, Blaauw, Brooks]

2. *The study of computer architecture is the study of the organization and interconnection of components of computer systems. Computer architects construct computers from basic building blocks such as memories, arithmetic units and buses.*

Was ist Rechnerarchitektur? (cont.)

From these building blocks the computer architect can construct anyone of a number of different types of computers, ranging from the smallest hand-held pocket-calculator to the largest ultra-fast super computer. The functional behaviour of the components of one computer are similar to that of any other computer, whether it be ultra-small or ultra-fast.

By this we mean that a memory performs the storage function, an adder does addition, and an input/output interface passes data from a processor to the outside world, regardless of the nature of the computer in which they are embedded. The major differences between computers lie in the way of the modules are connected together, and the way the computer system is controlled by the programs. In short, computer architecture is the discipline devoted to the design of highly specific and individual computers from a collection of common building blocks. [Stone]

Was ist Rechnerarchitektur? (cont.)

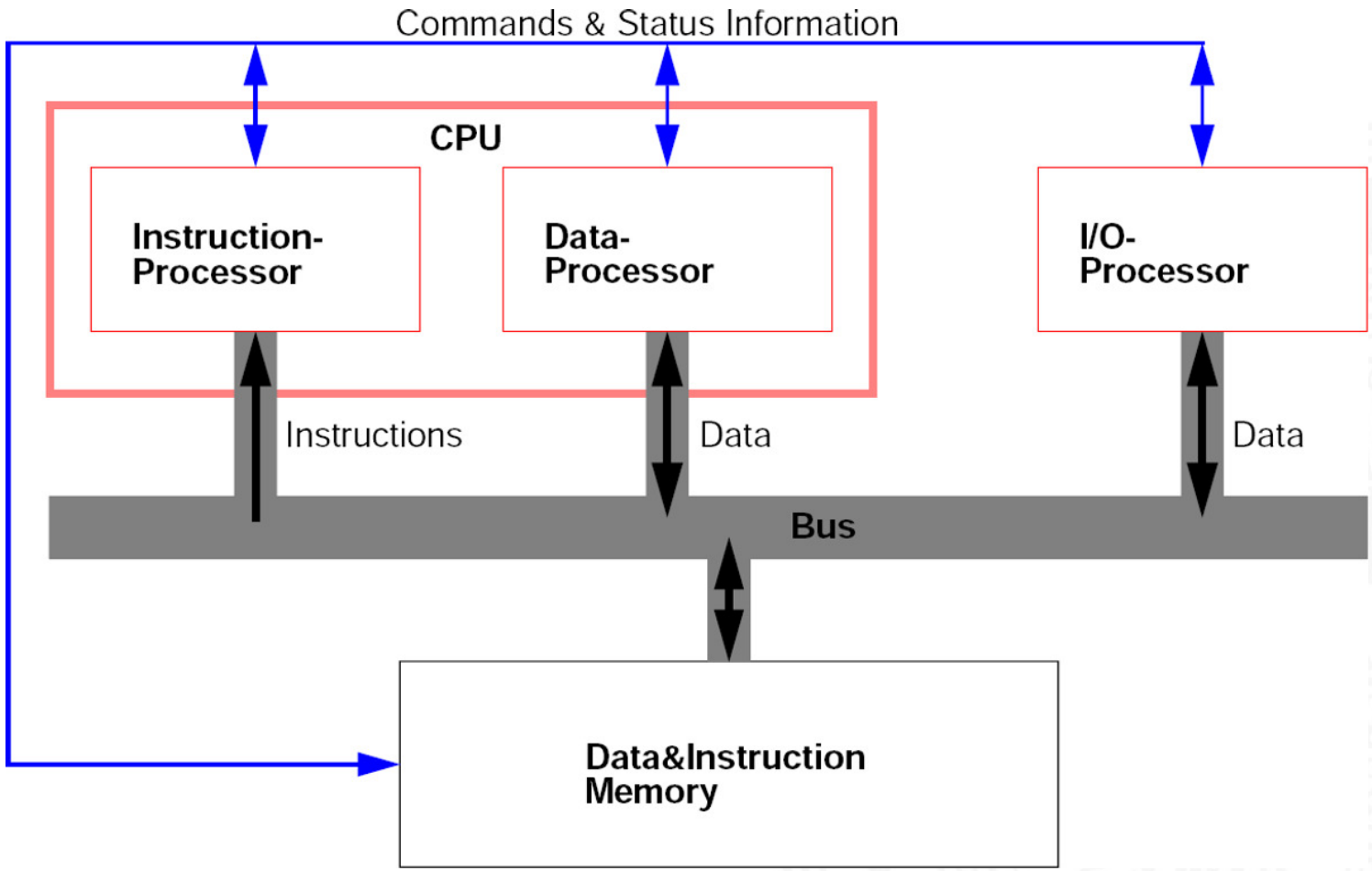
Zwei Aspekte der Rechnerarchitektur

1. Operationsprinzip: das funktionelle Verhalten der Architektur
   = Programmierschnittstelle
   = ISA – **I**nstruction **S**et **A**rchitecture
2. Hardwarestruktur: beschrieben durch Art und Anzahl der Hardware-Betriebsmittel sowie die sie verbindenden Kommunikationseinrichtungen
   = Mikroarchitektur, beispielsweise „von-Neumann“ Architektur

von-Neumann Architektur

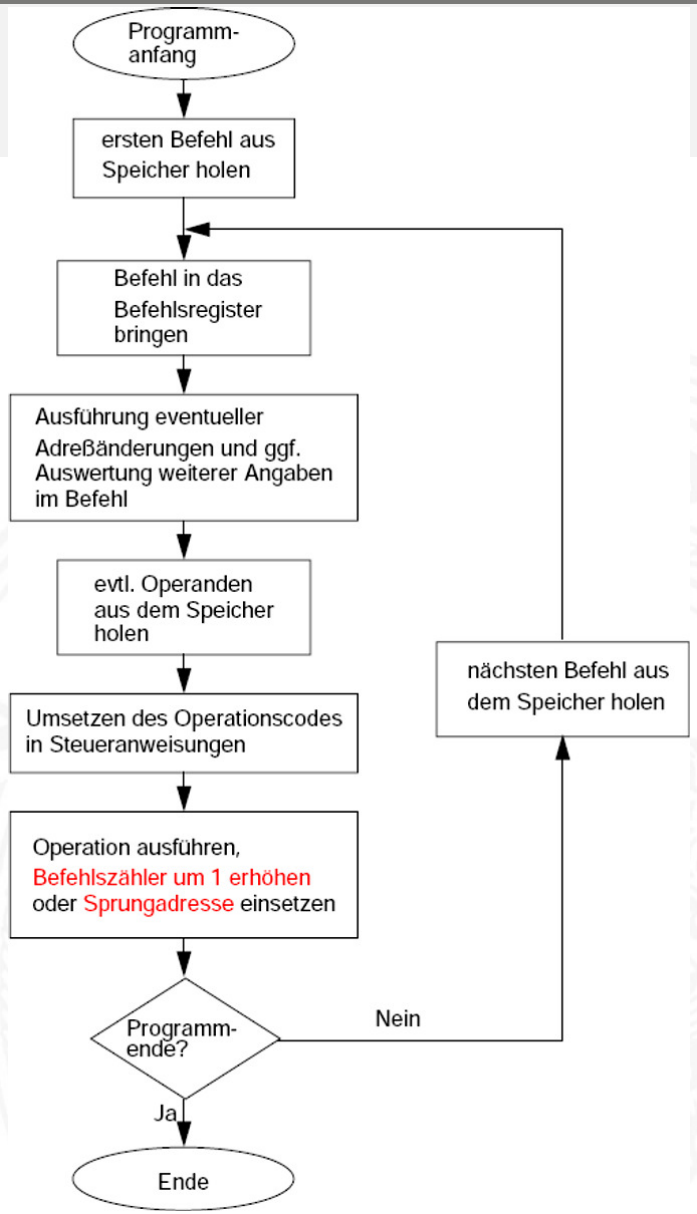
- Historie: 1945 entwickelt (John von Neumann)
- Abstrakte Maschine mit minimalem Hardwareaufwand
- Komponenten
  - zentralen Recheneinheit: CPU
    - logisch unterteilt in
    1. Datenprozessor / Rechenwerk / Operationswerk
    2. Befehlsprozessor / Leitwerk / Steuerwerk
  - Speicher für Daten und Befehle, fortlaufend adressiert
  - Ein/Ausgabe-Einheit zur Anbindung peripherer Geräte
  - Bussystem(e) als verbinden diese Komponenten
  - die Struktur ist unabhängig von dem Problem, das Problem wird durch austaschbaren Speicherinhalt (Programm) beschrieben

von-Neumann Architektur (cont.)



von-Neumann Architektur (cont.)

- Paradigma der Programmverarbeitung
  - Programm als Sequenz elementarer Anweisungen (Befehle)
  - als Bitvektoren im Speicher codiert
  - Interpretation (Operanden, Befehle und Adressen) ergibt sich aus dem Kontext (der Adresse)
  - zeitsequenzielle Ausführung der Instruktionen



von-Neumann Architektur (cont.)

– „von-Neumann Flaschenhals“: Zugriff auf Speicher

... vieles davon gilt Heute noch, außer

- *parallele*, statt sequentieller Befehlsabarbeeitung
- *Speicherhierarchie*, z.T. getrennte Daten und Instruktionsspeicher

Kriterien beim Entwurf

- Architekt sucht beste Lösung im Suchraum möglicher Entwürfe
- Kriterien „guter“ Architekturen:
  - hohe Rechenleistung
  - zuverlässig, robust
  - modular, skalierbar
  - einfach handhabbar, programmierbar
  - orthogonal
  - ausgewogen
  - wirtschaftlich, adäquat
  - ...

Kriterien beim Entwurf (cont.)

Skalierbarkeit Hinzufügen weiterer Module (ohne zusätzliche Änderungen) verbessert das System
⇒ Erweiterbarkeit, Wirtschaftlichkeit

Orthogonalität Jedes Modul hat eine definierte Funktionalität; keine zwei Module bieten die gleiche Funktionalität
⇒ Wartbarkeit, Kosten, Handhabbarkeit

Adäquatheit Die Kosten eines Moduls sind adäquat zur Funktion
⇒ Performance, Kosten

Virtualität Elimination physikalischer Grenzen: virtueller Prozessor, virtueller Speicher, virtuelle Kanäle...
⇒ skalierbar, ausbaubar, einfache Programmierung

Kriterien beim Entwurf (cont.)

Transparenz unwichtige Details werden verborgen
⇒ einfache Programmierung

Performance-Transparenz Änderung der System-Konfiguration ohne die Funktionalität zu beeinflussen
⇒ Skalierbarkeit, Wartbarkeit, Zuverlässigkeit

Größentransparenz Erweiterung des System, so dass sich die Performance verbessert
⇒ Skalierbarkeit, Kosten

Fehlertransparenz System verbirgt, maskiert oder toleriert Fehler
⇒ Zuverlässigkeit

- Die Punkte sind hier für die *Mikroarchitektur* formuliert, gelten aber gleichermaßen für die *ISA*

Kriterien zur Architekturbewertung

Kenngrößen zur Bewertung

- Taktfrequenz
- Werte die sich aus Eigenschaften der Architektur ergeben
- Ausführungszeiten von Programmen
- Durchsätze
- statistische Größen
- ...

Die Wahl der Kenngrößen hängt entscheidend von der jeweiligen Zielsetzung ab

Kriterien zur Architekturbewertung (cont.)

Verfahren zur Bestimmung der Kenngrößen

- *Benchmarking*: Laufzeitmessung bestehender Programme
  - Standard Benchmarks
    SPEC Standard Performance Evaluation Corporation
    `http://www.spec.org`
    TPC Transaction Processing Performance Council
    `http://www.tpc.org`
  - profilspezifische Benchmarks: SysMark, PCmark, Winbench etc.
  - benutzereigene Anwendungsszenarien
- *Monitoring*: Messungen während des Betriebs
- Modelltheoretische Verfahren: Analytische Modelle, Simulation...

Kenngrößen

Taktfrequenz

- In den letzten Jahren erfolgreich beworben!

⇒ *für die Leistungsbewertung aber völlig ungeeignet*

theoretische Werte

- MIPS – **M**illion **I**nstructions **P**er **S**econd
- MFLOPS – **M**illion **Fl**oating Point **O**perations **P**er **S**econd
  - keine Angabe über die Art der Instruktionen und deren Ausführungszeit
  - nicht direkt vergleichbar
- innerhalb einer Prozessorfamilie sinvoll

Kenngrößen (cont.)

Ausführungszeit

- Benutzer: *Wie lange braucht mein Programm?*
- Gesamtzeit: Rechenzeit +
  Ein-/Ausgabe, Platten- und Speicherzugriffe...
- CPU-Zeit: Unterteilung in System- und Benutzer-Zeit

Unix `time`-Befehl: `597.07u 0.15s 9:57.61 99.9%`

`597.07`	user CPU time [sec.]
`0.15`	system CPU time
`9:57.61`	elapsed time
`99.9`	CPU/elapsed [%]

Kenngrößen (cont.)

Theoretische Berechnung der CPU-Zeit (user CPU time)

- CPU-Zeit $= IC \cdot CPI \cdot T$

IC	Anzahl auszuführender Instruktionen	**I**nstruction **C**ount
CPI	mittlere Anzahl Takte pro Instruktionen **C**ycles **p**er **I**nstruction	
T	Taktperiode	

- IC kleiner: weniger Instruktionen
  - bessere Algorithmen
  - bessere Compiler
  - mächtigerer Befehlssatz

Kenngrößen (cont.)

- *CPI* kleiner: weniger Takte pro Instruktion
  - parallel Befehle ausführen: VLIW...
  - parallel Teile der Befehle bearbeiten: Pipelining, Super-Skalar...
- *T* kleiner: höhere Taktfrequenz
  - Technologie
- genauere Untersuchung wenn CPI über die Haufigkeiten und Zyklenanzahl einzelner Befehle berechnet wird
- so lassen sich beispielsweise alternative Befehlssätze miteinander vergleichen

Kenngrößen (cont.)

CPU-Durchsatz

- RZ-Betreiber
  - *Wie viele Aufträge kann die Maschine gleichzeitig verarbeiten?*
  - *Wie lange braucht ein Job im Mittel?*
  - *Wie viel Arbeit kann so pro Tag erledigt werden?*

$\Rightarrow$ Latenzzeit: *Wie lange dauert es, bis mein Job bearbeitet wird?*

$\Rightarrow$ Antwortzeit: *Wie lange rechnet mein Job?*

- Modellierung durch Warteschlangentheorie: Markov-Ketten, stochastische Petri-Netze...

Kenngrößen (cont.)

statistische Werte zur Zuverlässigkeit

- Betriebssicherheit des Systems: „Quality of Service“
- Fehlerrate: Fehlerursachen pro Zeiteinheit
  Ausfallrate: Ausfälle pro Zeiteinheit
  - *Fault:* Fehlerursache
  - *Error:* fehlerhafter Zustand
  - *Failure:* ein Ausfall ist aufgetreten
- MTTF Mean Time To Failure
  MTBF Mean Time Between Failures
  MTTR Mean Time To Repair
  MTBR Mean Time Between Repairs

Kenngrößen (cont.)

- Überlebenswahrscheinlichkeit, *Reliability*
  - Maß für kontinuierlichen korrekten Service
  - Wahrscheinlichkeit, dass System nach Zeit t noch funktioniert, wenn es zum Zeitpunkt t_0 funktioniert hat:

    $R(t) = e^{-\lambda t}$ zeitunabhängige Ausfallrate: λ
- Verfügbarkeit, *Availability*
  - Maß für korrekten Service, u.U. mit Unterbrechungen
  - Wahrscheinlichkeit, dass System zum Zeitpunkt t funktioniert:

    $A(t) =$ konstant
- Sicherheit, *Safety*
  - Maß für Service ohne katastrophale Fehler
  - Wahrscheinlichkeit, dass System trotz internem Fehler keinen (katastrophalen) Ausfall hat

Kenngrößen (cont.)

- Robustheit, *Robustness*
  - Maß für das korrekte Systemverhalten trotz fehlerhafter Eingaben
  - Wahrscheinlichkeit, dass System trotz internem Fehler nicht ausfällt und keine falschen Ausgaben produziert
- Sicherheit, *Security*
  - Maß für den Aufwand um unerlaubten Zugriff auf Informationen zu verhindern
- Wartbarkeit, *Maintainability*
  - Maß für den Aufwand das System funktionsfähig zu halten

Bewertung von Maßnahmen

Wie wirken sich Verbesserungen der Rechnerarchitektur aus?

- Speed-Up: Verhältnis von Ausführungsszeiten *vor* und *nach* der Verbesserung

  Speed-Up $= T_{\mathbf{v}orVerbesserung} / T_{\mathbf{n}achVerbesserung}$
- werden nur Teile der Berechnung beschleunigt, Faktor F:

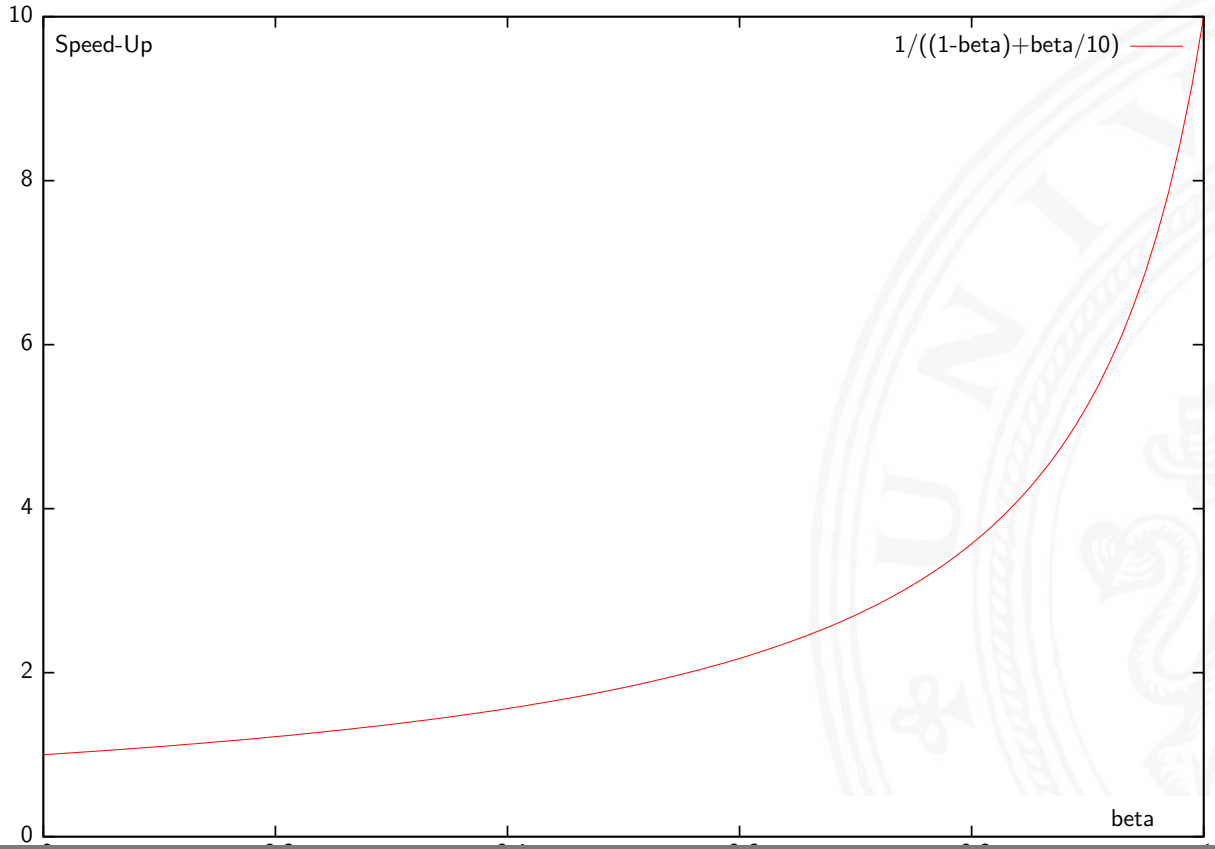
$$\begin{aligned} T &= T_{\mathbf{o}hneEffekt} + T_{\mathbf{m}itEffekt} \\ T_n &= T_{v,o} + T_{v,m}/F_{Verbesserung} \end{aligned}$$

$\Rightarrow$ den „Normalfall“, den häufigsten Fall beschleunigen, um den größten Speed-Up zu erreichen

Bewertung von Maßnahmen (cont.)

- Amdahlsches Gesetz (s.u.)

$$\text{Speed-Up} = \frac{1}{(1-\beta)+\beta/F_{Verbesserung}} \text{ mit } \beta = T_{v,m}/T_v$$



$F_{Verbesserung} = 10$

Bewertung von Maßnahmen (cont.)

Amdahlsches Gesetz

- Beschleunigung der Bearbeitung durch Parallelausführung mit N Prozessoren (Gene Amdahl '67)
- Speed-Up $= \frac{1}{(1-\beta)+f_k(N)+\beta/N} \leq \frac{1}{(1-\beta)}$

N	# Prozessoren als Verbesserungsfaktor
β	Anteil parallelisierbarer Berechnung
$1-\beta$	Anteil nicht parallelisierbarer Berechnung
$f_k()$	Kommunikationsoverhead zwischen den Prozessoren

- Aufgaben verteilen
- Arbeit koordinieren
- Ergebnisse zusammensammeln

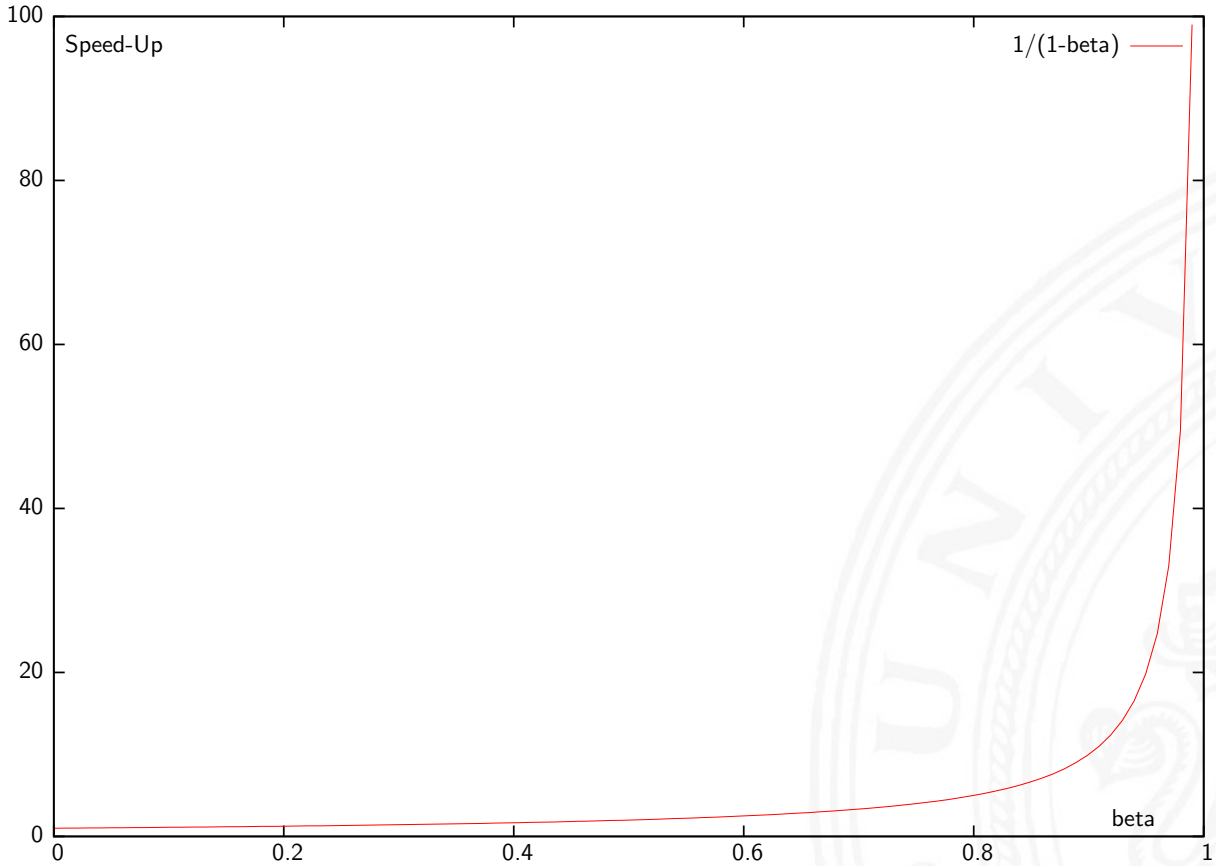
Bewertung von Maßnahmen (cont.)

- *Welche Auswirkungen hat das in der Praxis?*

N	β	Speed-up
2	0,40	1,25
4	0,40	1,43
4536	0,80	5,00
9072	0,99	98,92

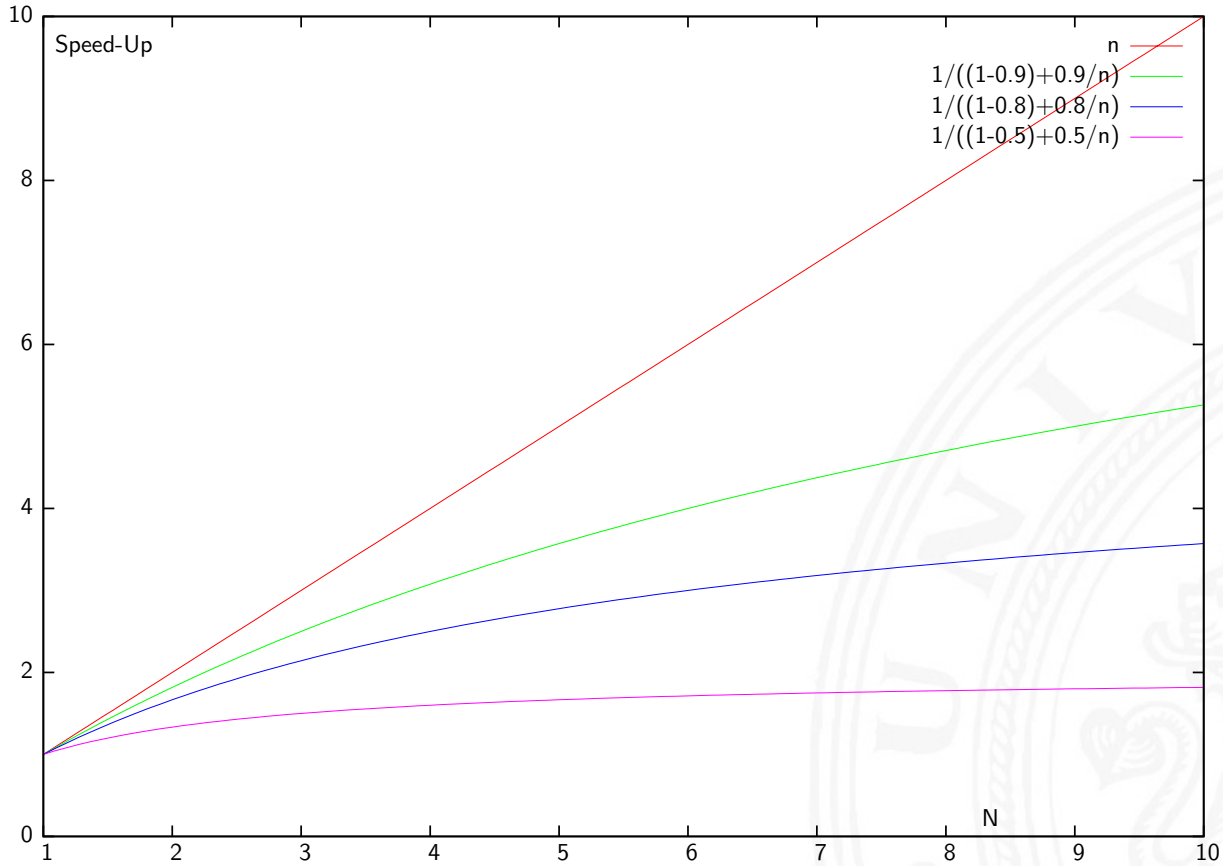
  - Enttäuschend: zwei Prozessoren sind *nicht* doppelt so schnell
  - . . . immerhin bei Multitasking und mehreren Prozessen kommt man auf große β

Bewertung von Maßnahmen (cont.)



$F_{Verbesserung} = \infty$

Bewertung von Maßnahmen (cont.)



Bewertung von Maßnahmen (cont.)

Gustafsonsches Gesetz

- Umkehrung von Amdahl's Gesetz
- Speed-Up eines Parallelrechners gegenüber einem ein-Prozessor Rechner

$$
\begin{aligned}
T_n &= T_{v,o} + N \cdot T'_{v,m} \\
\text{Speed-Up} &= \frac{T_{v,o} + N \cdot T'_{v,m}}{T_{v,o} + T'_{v,m}} \\
&= 1 + (N-1)\beta' \quad \text{mit } \beta' = \frac{T'_{v,m}}{T_{v,o} + T'_{v,m}}
\end{aligned}
$$