

Wolfgang E. Nagel
Center for Information Services and High Performance Computing (ZIH)

Rechnerarchitektur II

Definitionen und Prinzipien



Gliederung

- Begriffe und Definitionen
- Einflusskomplexe
- Entwurf eines Rechnersystems
 - Zielsetzungen
 - Gestaltungsgrundsätze
 - Randbedingungen
- Bemerkungen zum klassischen Digitalrechner
- Aufgaben und Ziele der Rechnerarchitektur

Begriffe und Definitionen

Begriffe und Definitionen

- *Rechnerarchitektur* wurde von englischsprachigen Begriff *computer architecture* abgeleitet
- *Computer architecture* ist Teildisziplin des *computer engineering*:
 - Überwiegend ingenieurmäßige Herangehensweise bei Entwurf und Optimierung von Rechnersystemen
- Zwei Deutungen des englischen Begriffs „*architecture*“:
 - Baukunst, Baustil
 - Einzelprodukte der Baukunst
- Gilt nicht für den deutschen Begriff Architektur
- Rechnerarchitektur
 - Relativ junge Fachdisziplin
 - etwa 1943 entstand der erste digitale Ziffernrechenautomat
 - erst um 1962 wurde die Rechnerarchitektur kreiert

Zur Definition der Rechnerarchitektur

- Von Anfang der 60er Jahre bis Ende der 70er Jahre versuchte man, die interne Struktur und Organisation eines Rechners vor dem Nutzer (Programmierer) zu verbergen.
- Architektur: Ausdruck insbesondere der Möglichkeiten der Programmierschnittstelle
 - Maschinenbefehlssatz
 - Registerstruktur
 - Adressierungsmodi
 - Unterbrechungsbehandlung
 - Ein- und Ausgabe-Funktionalität

Beispiel: Intel Xeon Skylake-SP

Instruktionsausführung

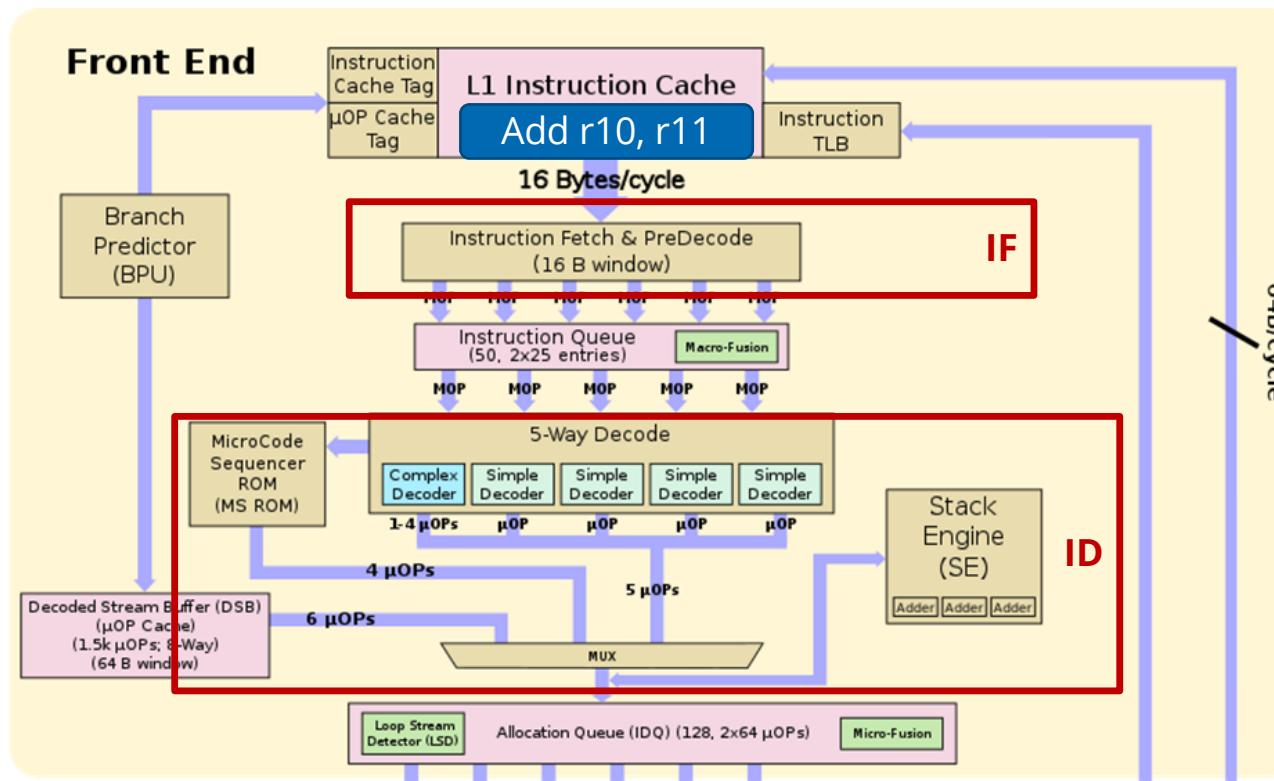
Mehrere Zyklen pro Instruktion

- Holen der Instruktion (Instruction fetch cycle / **IF**)
- Dekodieren der Instruktion (Instruction decode/register fetch cycle / **ID**)
- Ausführen/Adressberechnung (Execution/effective address cycle / **EX**)
- Speicherzugriff (Memory access / **MEM**)
- Schreiben des Ergebnisses (Write-back cycle / **WB**)

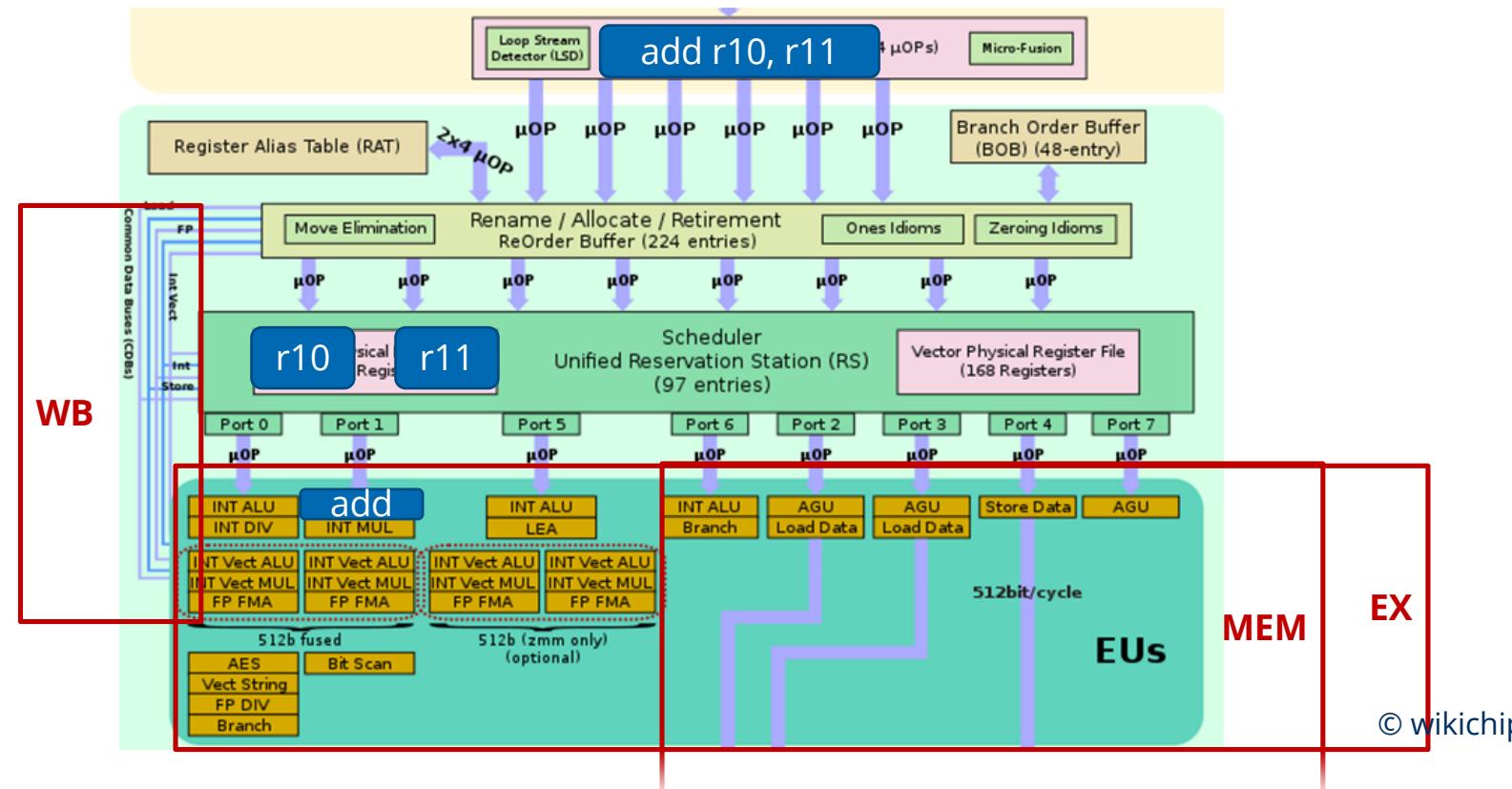
(Pipeline lt. Hennessy&Patterson (2007). Computer Architecture - A Quantitative Approach.)

- Hier am Beispiel des aktuellen Intel Server Prozessors (Skylake-SP)
- Welche Bestandteile brauchen wir dafür?

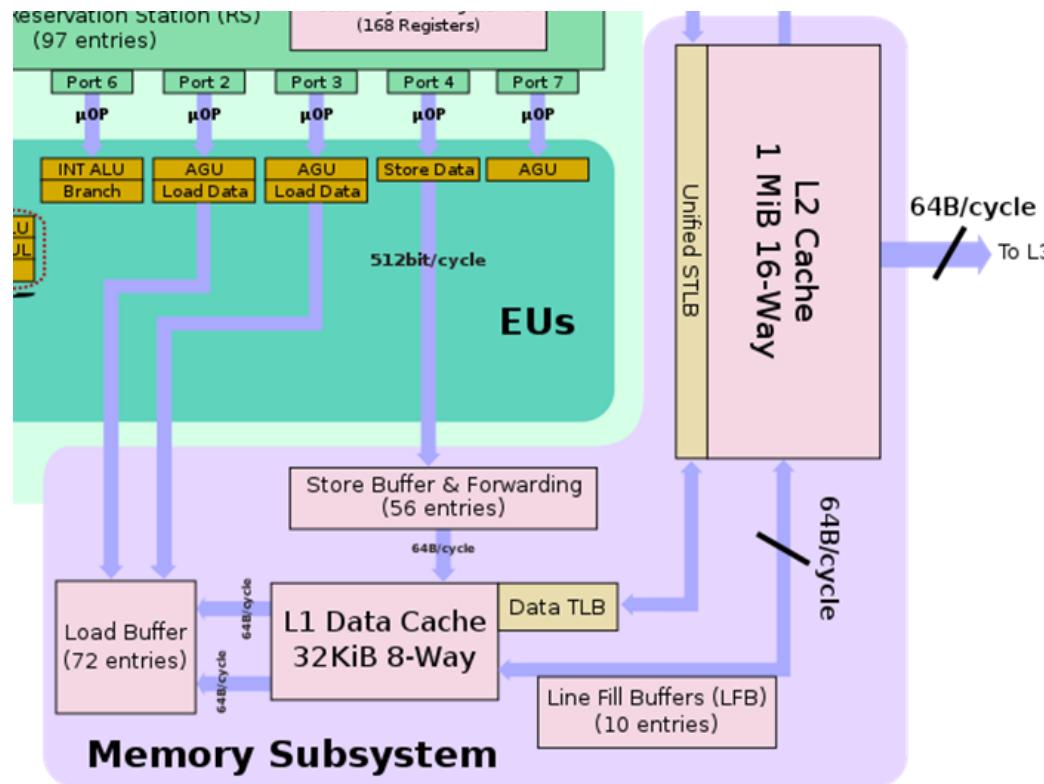
Instruction Fetch IF, Instruction Decode ID



Execute EX, Write-back WB



Memory access MEM



© wikichip

Welche Bestandteile?

- Ausführungseinheiten
- Puffer/Speicher
- Verbindungsnetzwerke/Busse
- Diese Einteilung sehen wir gleich wieder

Klassifikation nach Giloi

Gilioi: Rechnerarchitektur/Hardwarestruktur/...

— Hardwarestruktur

Art und Anzahl der Hardware-Betriebsmittel und ihrer Verbindungseinrichtungen

— Operationsprinzip

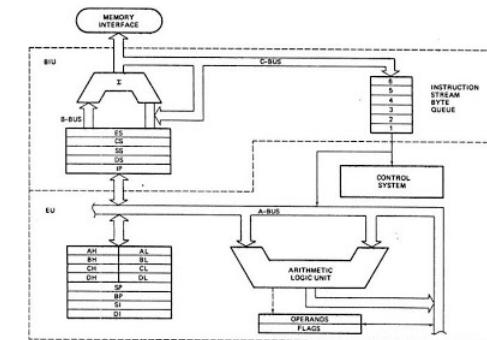
– Informationsstruktur

Interne Struktur von Code und Daten

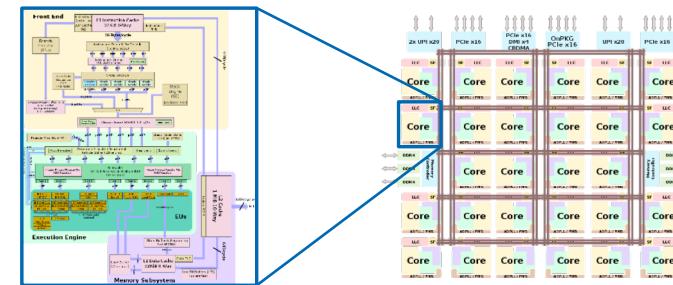
– Steuerungs-(Kontroll-)Struktur:

Spezifikation der Algorithmen für die Interpretation und Transformation der Maschinendaten – zeitliche Wirkungsweise

8086:

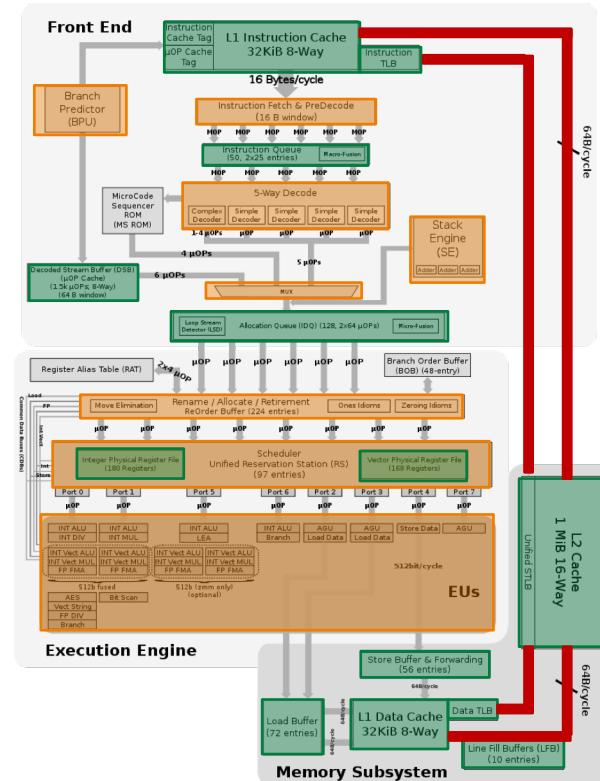
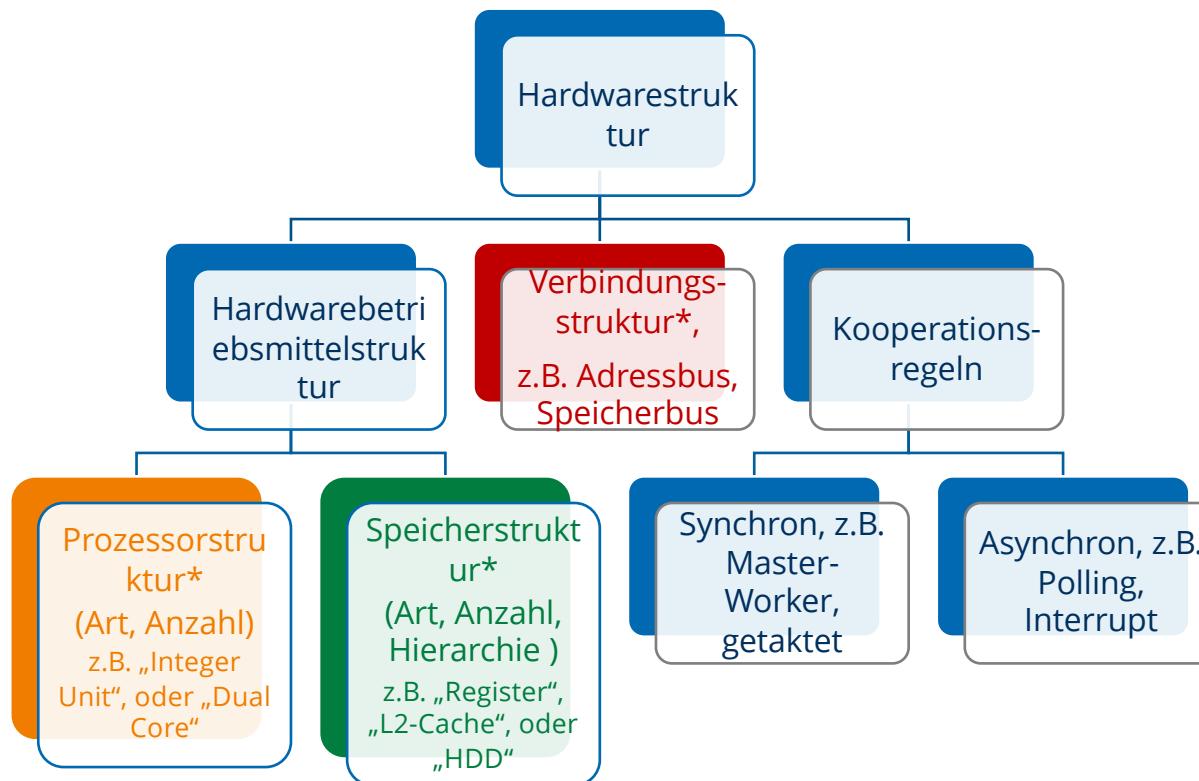


Intel Xeon Skylake-SP:



© wikichip

Komponenten der Hardwarestruktur



Rechnerarchitektur ist mehr als nur die Hardware!

Gilio: Rechnerarchitektur/Operationsprinzip/Informationsstruktur/...

— Hardwarestruktur

Art und Anzahl der Hardware-Betriebsmittel und ihrer Verbindungseinrichtungen

— Operationsprinzip

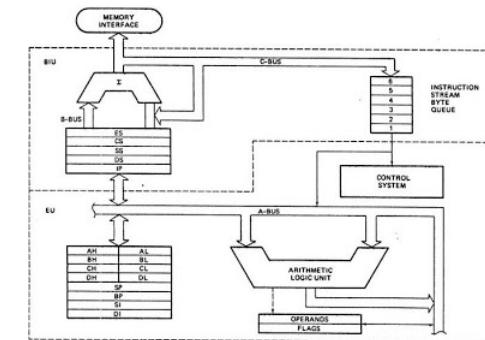
– Informationsstruktur

Interne Struktur von Code und Daten

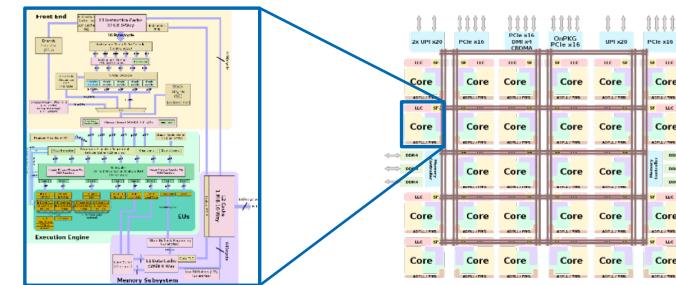
– Steuerungs-(Kontroll-)Struktur:

Spezifikation der Algorithmen für die Interpretation und Transformation der Maschinendaten – zeitliche Wirkungsweise

8086:



Intel Xeon Skylake-SP:



© wikichip

Informationsstruktur sichtbar in Befehlssatz und Registern

- Klassen von Datenobjekten
 - niedere Elementardatentypen (entsprechen Festkomma-DT), z.B. Byte/Halfword/Word/... (un)signed
 - höhere Elementardatentypen (entsprechen Gleitkomma-DT), z.B. floating point, single/double precision
 - Gruppendatentypen (gehören zu den Strukturdatentypen), z.B. Vektoren ,Felder, sequentielle Listen, Stack, Adressstack
 - Strukturdatentypen, z.B. verkettete Listen

X86

- Addition gibt es für:
 - Byte/Halfword/Word/Double Word (un)signed (Add, 64 bit general purpose Register)
 - Float, Double (z.B. FADD, 80 bit Fließkommaregister)
 - Vektoren (z.B. ADDPS, 128 – 512 bit Vektorregister)

Informationsstruktur beinhaltet auch die Darstellung

- Menge der Maschinendarstellungen der Datenobjekttypen
(nur beispielhaft, nicht vollständig)
 - 1-er Komplement für Integer
 - 2-er Komplement für Integer
 - Vorzeichen-Betrags-Notation für Integer
 - ungepackte BCD-Darstellung
 - gepackte BCD-Darstellung
 - Floating-Point-Darstellungen seit Anfang der 90er Jahre durchgesetzt:
 - IEEE 754-1985 mit single precision (32 bit), double precision (64 bit)
 - Big-/little-Endian

Informationsstruktur beinhaltet auch den Befehlssatz

- Menge von *Funktionen*, die auf die Datenobjekte anwendbar sind, d.h. die unterschiedlichen Befehle des Maschinenbefehlssatzes
- Transferfunktionen (Reg.-Sp.; Reg.-Reg.; [Sp.-Sp.]; Reg.-I/O)
- Datenmanipulationen (arith., Vergleich, log., Schiebe., Rotation)
- Steuerflussoperationen (Sprung [unbedingt], Verzweigung [bedingt], Unterprogrammaufruf [unbedingt/bedingt], Return Sub)
- Maschinensteuerung (Traps [synchr.], Interr. [asyn.], Sp.-verw.)

X86

- Addition
- Multiplikation
- Load
- Store
- ...

Datentypen +
Maschinendarstellung + Befehle
=
Informationsstruktur

Gilioi: Rechnerarchitektur/Operationsprinzip/Steuerungsstruktur/...

— Hardwarestruktur

Art und Anzahl der Hardware-Betriebsmittel und ihrer Verbindungseinrichtungen

— Operationsprinzip

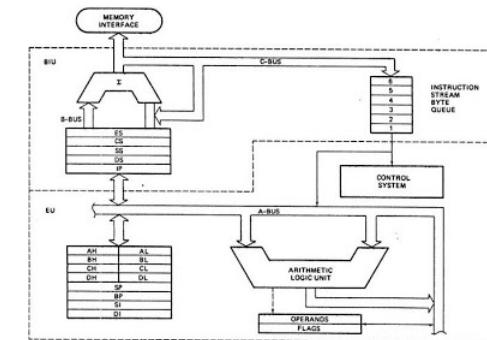
– Informationsstruktur

Interne Struktur von Code und Daten

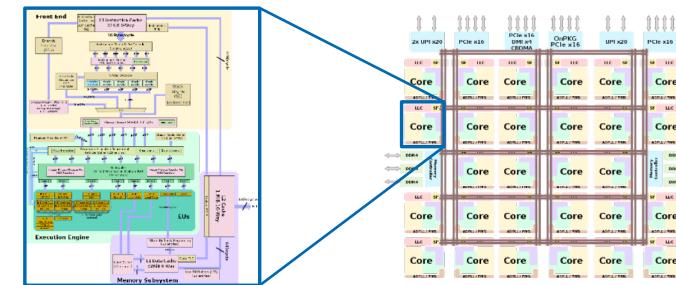
– Steuerungs-(Kontroll-)Struktur:

Spezifikation der Algorithmen für die Interpretation und Transformation der Maschinendaten – zeitliche Wirkungsweise

8086:



Intel Xeon Skylake-SP:



© wikichip

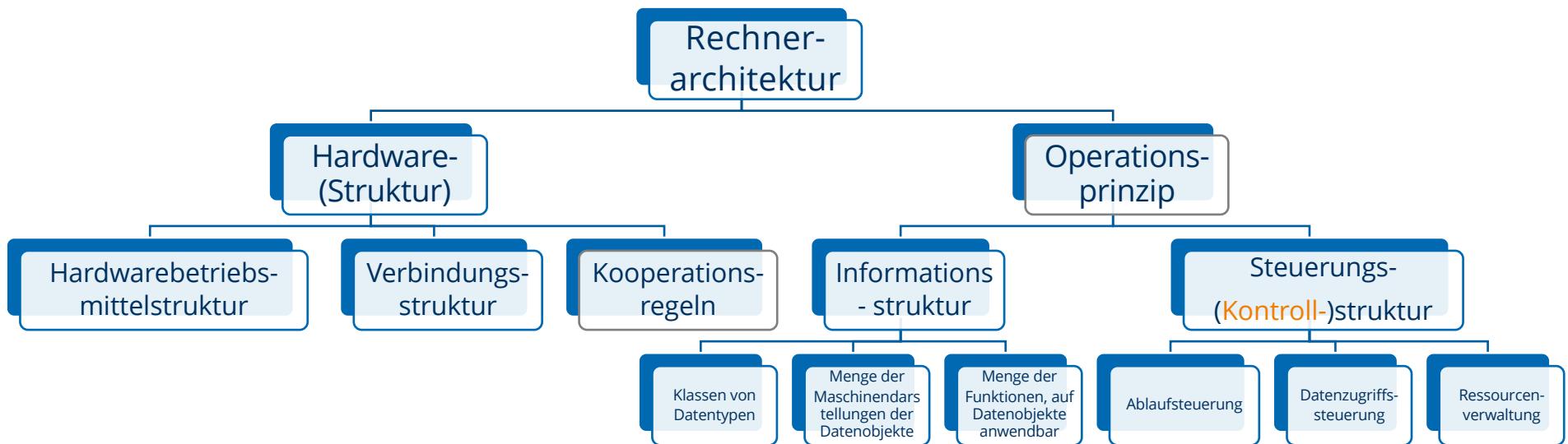
Steuerungs-(Kontroll-)Struktur sagt wie es auf HW umgesetzt wird

- Ablaufsteuerung
- **PC-getrieben: typische von-Neumann-Ablaufsteuerung; prozedural**
- Datengetrieben: eingesetzt in Datenflussarchitekturen
- Anforderungsgetrieben: eingesetzt in Reduktionsarchitekturen
- Datenzugriffssteuerung
- **Zugriff über Adresslogik [eventuell Multiport mit unterschiedlichen Zugriffsstrategien (Prioritäten; fair; ...)]**
- einfache Wertzuweisung (single assignment)
[jede Variable darf nur 1x geschrieben werden und danach 1x gelesen werden; danach ist kein weiterer Zugriff möglich; Vervielfachungen sind durch Broadcast möglich]
- **assoziativer Zugriff**
[Adresse und Inhalt werden gemeinsam abgespeichert, sog. inhaltsadressierbarer Speicher (content addressable memory (CAM))]

Steuerungs-(Kontroll-)Struktur sagt wie es auf HW umgesetzt wird

- Ressourcenverwaltung
- Ressourcenzuweisung:
 - Prozessorzuweisung
(Zuordnung des auszuführenden Codes zu unterschiedlichen Units [z.B. bei mehreren Integer- bzw. Floating-point-Units] oder vollständigen CPUs; eventuell von Daten zu Rechenwerken Feldrechners)
 - Speicherverwaltung (siehe RA I)
- Kommunikation zwischen den Ressourcen
 - ... über entsprechende Kommunikationsprotokolle
(ISO/OSI-Schichtenmodell, Cache-Kohärenz-Protokolle für unterschiedliche Verbindungsnetze)
- Steuerungsstruktur legt fest, wie Informationsstrukturen auf Hardwarestrukturen gemappt werden

Kompletter Baum – Beschreibung nach Giloi (Siehe Übung 01)



Taxonomie

Gilioi hat auf der Basis dieser Definition der Rechnerarchitektur ...
...eine Taxonomie der Rechnerarchitektur entwickelt.

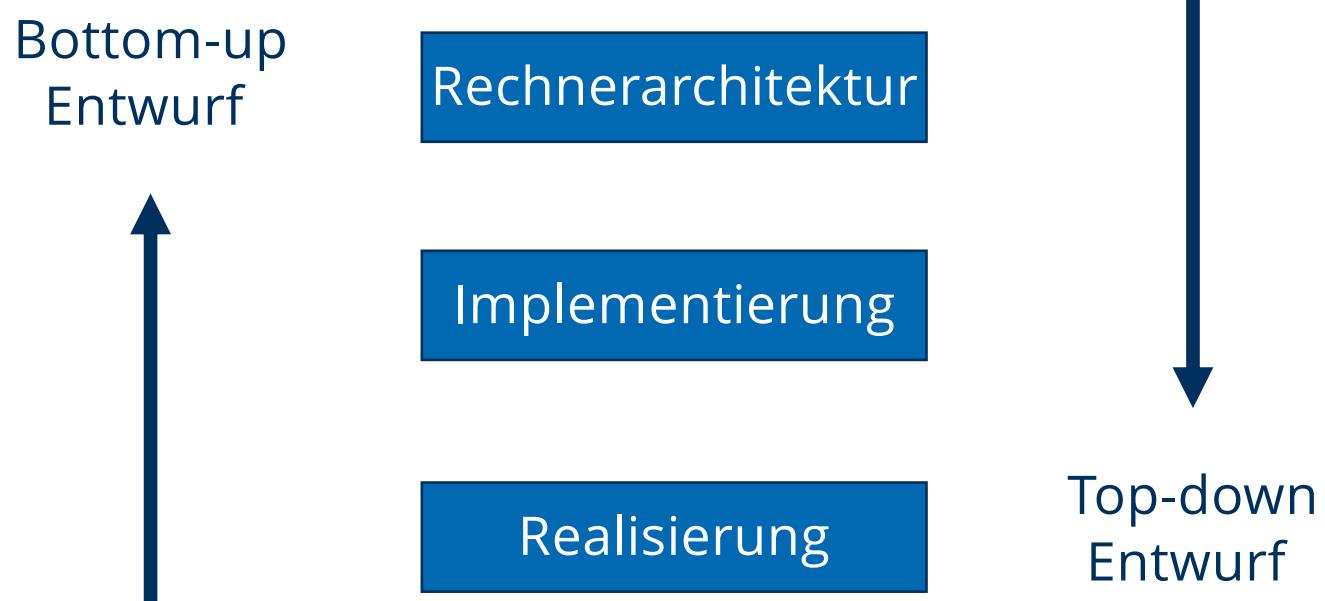
Begriff der Taxonomie:

- Taxonomie ist nicht irgendeine Klassifizierung, sondern ein Ordnungsprinzip, das genügend Spielraum zur genauen Einordnung bereits vorliegender Ergebnisse der Rechnerarchitekturforschung besitzt sowie einer systematischen Suche nach neuen Architekturformen gerecht wird.

Details: Später im entsprechenden Abschnitt!

Rechnerentwurf

Dreiphasenmodell zum Entwurf eines Rechnersystems

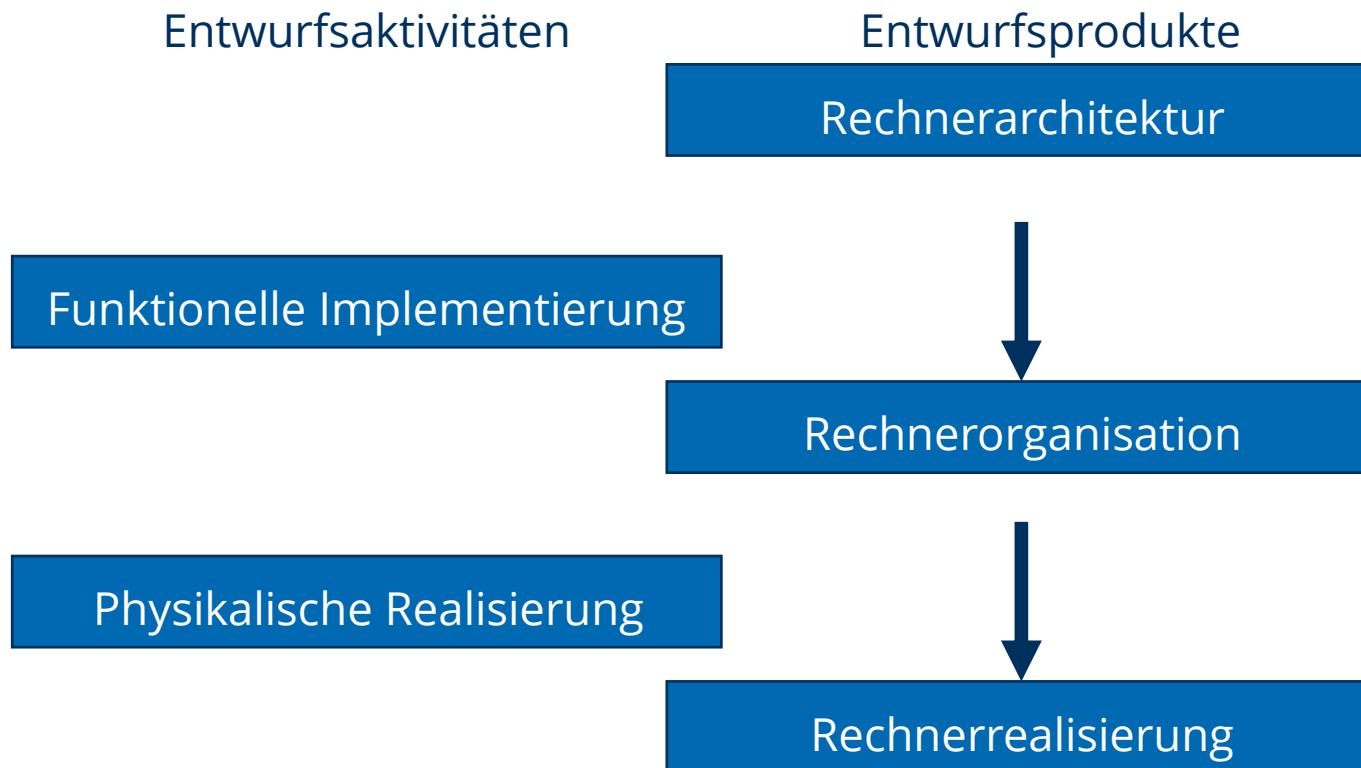


Rückwirkungen durch den technologischen Stand



- Bottom-up typischerweise nur solange, wie kein Variantenreichtum
- Heute: breites Spektrum z.B. von unterschiedlich leistungsfähigen und kostenaufwendigen Halbleitertechnologien
- Bottom-up-Entwurf wurde vor der Entstehung des Familienkonzeptes praktiziert

Dreiphasenmodell nach Entwurfsaktivitäten und Entwurfsprodukten



Begriffe

Rechnerarchitektur:

- Operationsprinzip und Struktur des gesamten Rechnersystems

Rechnerorganisation:

- Funktionelle Arbeitsweise der einzelnen Hardware-Betriebsmittel einer Rechenanlage einschließlich:
 - Verbindungseinrichtungen
 - zeitabhängige Wechselwirkung zwischen den einzelnen Hardware-Betriebsmitteln
 - Details von Daten- und Steuerfluss

Rechnerrealisierung:

- Logischer Entwurf und physikalische Realisierung der einzelnen Hardware- Betriebsmittel
 - kombinatorische Schaltungen
 - analoge und digitale Schaltungstechnik
 - E-Technik
 - Halbleitertechnologie

Modifiziertes Dreiphasenmodell zum Entwurf eines RS

Rechnerarchitektur nach [Hen07]

Realisierung

Rechnerarchitektur nach [Tan06]

Befehlssatzarchitektur

Implementierung

[Tan06] A. S. Tanenbaum. Computerarchitektur - Strukturen - Konzepte - Grundlagen. Prentice Hall, 2006.

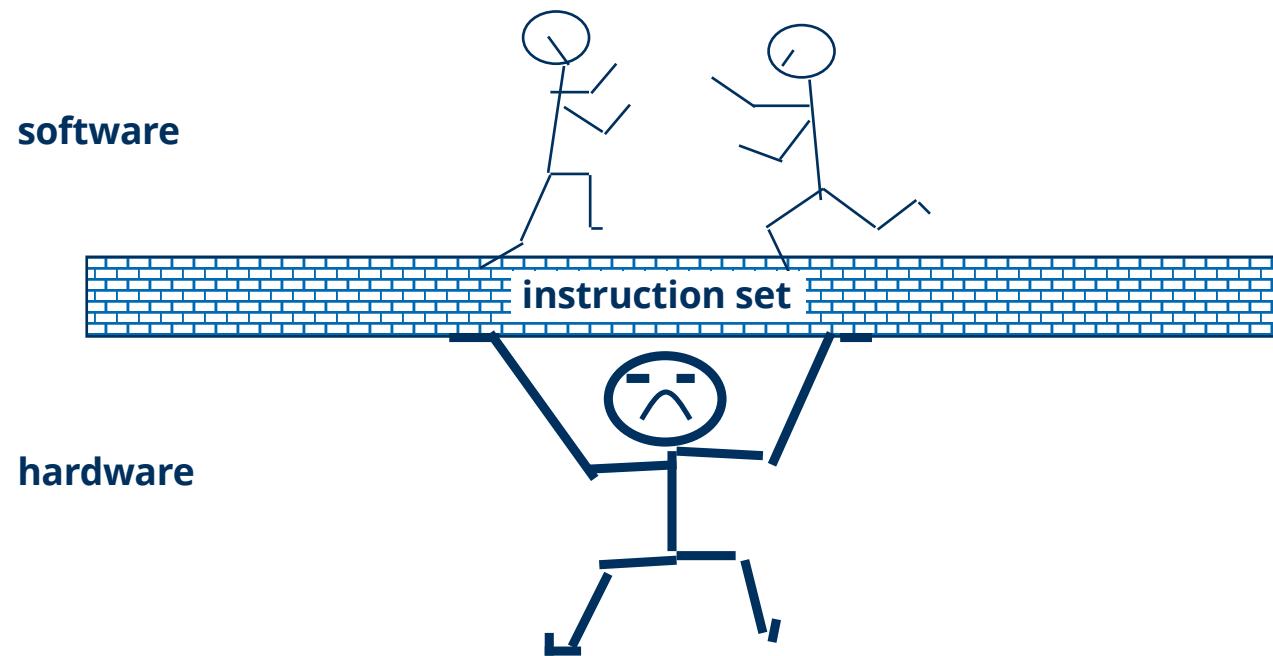
[Hen07] J. L. Hennessy and D. A. Patterson. Computer Architecture - A Quantitative Approach. Morgan Kaufmann, 2007.

Architektur-Definition (Tanenbaum)

Allgemeine Architektur-Definition nach Tanenbaum ([Tan06], S. 24)

- “Den Satz von Datentypen, Operationen und Merkmalen jeder Ebene bezeichnet man als ihre **Architektur** (Architecture). Die Architektur betrifft die Aspekte, die für den Benutzer der jeweiligen Ebene sichtbar sind.”
- “Dagegen haben **Implementierungsdetails**, wie z.B. die für die Speicherchips verwendete Technologie, **nichts** mit der Architektur zu tun.”
- “Computerarchitektur und Computerorganisation bedeuten in der Praxis das Gleiche.”

The Instruction Set: a Critical Interface



Architekturbegriff (Hennessy/Patterson, [Hen07], S. 8 ff.)

"We use the term **instruction set architecture** (ISA) to refer to the actual programmer-visible instruction set in this book. The ISA serves as the boundary between the software and hardware."

"The implementation of a computer has two components: **organization** and **hardware**."

"The term **organization** includes the high-level aspects of a computer's design, such as the memory system, the memory interconnect, and the design of the internal processor or CPU (central processing unit – where arithmetic, logic, branching, and data transfer are implemented). For example, two processors with the same instruction set architectures but very different organizations are the AMD Opteron 64 and the Intel Pentium 4."

Architekturbegriff (Hennessy/Patterson, [Hen07], S. 8 ff.)

“**Hardware** refers to the specifics of a computer, including the detailed logic design and the packaging technology of the computer. Often a line of computers contains computers with identical instruction set architectures and nearly identical organizations, but they differ in the detailed hardware implementation. For example, the Pentium 4 and the Mobile Pentium 4 are nearly identical, but offer different clock rates and different memory systems, making the Mobile Pentium 4 more effective for low-end computers.”

“In this book the word **architecture** covers all three aspects of computer design – instruction set architecture, organization and hardware.”

Zusammenfassung Klassifikationen und Definitionen

Rechnerarchitektur =

- a) Hardwarestruktur* + Operationsprinzip* (Gilo)

* Wird weiter unterteilt

- b) Befehlssatzarchitektur + Implementierung (Tanenbaum)

- c) Befehlssatzarchitektur + Implementierung + Realisierung (Hennessy/Patterson)

Mit welchen Disziplinen interagiert die RA?

Wechselwirkung mit anderen Disziplinen

- Betriebssysteme
 - Topologie
 - Hardware-Entwurf
 - Compilerbau
 - Softwaretechnik
 - Software-Entwurf
 - Software-Ergonomie
 - Algorithmenentwurf
-
- Anforderungen Betriebssysteme an RA:
 - Virtueller Speicher [Tan06], S. 459 ff.
 - Memory Management Unit
 - Virtuelle I/O-Instruktionen: abstraktere I/O-Funktionalität gegenüber der ISA-Ebene [Tan06], S. 482 ff.
 - Anforderungen RA an Betriebssysteme:
 - Registersicherung bei Prozesswechsel
 - Rchte Nutzung von Schlafzuständen/Frequenzen
 - Weiterreichung von Hardwareschnittstellen (z.B. Performance Counter)

Wechselwirkung mit anderen Disziplinen

- Betriebssysteme
- Topologie
- Hardware-Entwurf
- Compilerbau
- Softwaretechnik
- Software-Entwurf
- Software-Ergonomie
- Algorithmenentwurf
- Ursprünglich Teilgebiet der Mathematik zur Untersuchung der Struktur von Punktmengen und Räumen einschließlich ihrer Klassifizierung
- Daraus folgte: Gestaltung von Verbindungseinrichtungen in Multiprozessorsystemen (Ring, Mesh, Baum)
- Für die Darstellung von Verbindungseinrichtungen ist auch die Graphentheorie bedeutsam

Wechselwirkung mit anderen Disziplinen

- Betriebssysteme
- Topologie
- Hardware-Entwurf
- Compilerbau
- Softwaretechnik
- Software-Entwurf
- Software-Ergonomie
- Algorithmenentwurf
- Die Sammlung von Anforderungen bildet die strukturelle und organisatorische Entwurfsspezifikation der Teilkomponenten eines Rechners.
- Zur Darstellung dienen Hardware-Beschreibungssprachen (HDL).

Wechselwirkung mit anderen Disziplinen

- Betriebssysteme
- Topologie
- Hardware-Entwurf
- Compilerbau
- Softwaretechnik
- Software-Entwurf
- Software-Ergonomie
- Algorithmenentwurf
- Codegenerator eines Compilers hängt von Architektur und Befehlssatz des Zielprozessors ab.
- Hier spielt auch die Mikroarchitektur eine große Rolle: z.B.
 - Welche Funktionseinheiten gibt es?
 - Wie lange dauern einzelne Befehle?
- Compilieren für den Befehlssatz oder für die Architektur?

Wechselwirkung mit anderen Disziplinen

- Betriebssysteme
- Topologie
- Hardware-Entwurf
- Compilerbau
- Softwaretechnik
- Software-Entwurf
- Software-Ergonomie
- Algorithmenentwurf
- Nutzung unterschiedlicher Programmierparadigmen und -modelle zur bestmöglichen Nutzung architektonischer Möglichkeiten von Prozessoren und Rechnersystemen
- Anregungen:
 - Objektorientierung und gekapselte Speichermodule (distributed local memory architectures)
 - Datenparallele Programmiersprachen und datenparallele Verarbeitung
- Entwicklungsaufwand für parallele Software!

Wechselwirkung mit anderen Disziplinen

- Betriebssysteme
- Topologie
- Hardware-Entwurf
- Compilerbau
- Softwaretechnik
- Software-Entwurf
- Software-Ergonomie
- Algorithmenentwurf
- Optimale Programme erfordern geeignete Lösungsalgorithmen
- Besonders drastische Bedingungen bestehen bei Parallelverarbeitung durch erforderliche Parallelalgorithmen und Parallelisierung sequenzieller Algorithmen
- Optimale Parallelalgorithmen können zum Einsatz sogenannter **Systolischer Arrays** führen
- Verweis auf später: Architekturkonzepte der Parallelverarbeitung

Einflusskomplexe der Rechnerarchitektur

Anwendungsanforderungen

Kompatibilitäts- und
Portabilitätsforderungen

Rechnerarchitektur

Hardwaretechnologien

Software-Technologien

Einflusskomplexe der Rechnerarchitektur

Anwendungsanforderungen

- Natürliche Kommunikation
 - Sprache, Schriftzeichen, Bilder, Graphiken
- Wissensverarbeitung
 - Lernen, Schlussfolgern

Kompatibilitäts- und Portabilitätsforderungen

Rechnerarchitektur

Hardwaretechnologien

Software-Technologien

Einflusskomplexe der Rechnerarchitektur

Anwendungsanforderungen

- Natürliche Kommunikation
 - Sprache, Schriftzeichen, Bilder, Graphiken
- Wissensverarbeitung
 - Lernen, Schlussfolgern

Kompatibilitäts- und Portabilitätsforderungen

- Software weiter nutzen → ausgereifte Entwicklungen, aber kann entwicklungshemmend werden
- Portabilitätsforderungen
- Standardisierte Schnittstellen

Rechnerarchitektur

Hardwaretechnologien

Software-Technologien

Einflusskomplexe der Rechnerarchitektur

Anwendungsanforderungen

- Natürliche Kommunikation
 - Sprache, Schriftzeichen, Bilder, Graphiken
- Wissensverarbeitung
 - Lernen, Schlussfolgern

Kompatibilitäts- und Portabilitätsforderungen

- Software weiter nutzen → ausgereifte Entwicklungen, aber kann entwicklungshemmend werden
- Portabilitätsforderungen
- Standardisierte Schnittstellen

Rechnerarchitektur

Hardwaretechnologien

Software-Technologien

- Rechnergestützter Software-Entwurf
- Daten- und Wissensbasis-Veraltungssysteme
- Automatischer Software-Entwurf

Einflusskomplexe der Rechnerarchitektur

Anwendungsanforderungen

- Natürliche Kommunikation
 - Sprache, Schriftzeichen, Bilder, Graphiken
- Wissensverarbeitung
 - Lernen, Schlussfolgern

Kompatibilitäts- und Portabilitätsforderungen

- Software weiter nutzen → ausgereifte Entwicklungen, aber kann entwicklungshemmend werden
- Portabilitätsforderungen
- Standardisierte Schnittstellen

Rechnerarchitektur

Hardwaretechnologien

- Halbleiterschaltkreise
- Massenspeicher
- Bus-Systeme

Software-Technologien

- Rechnergestützter Software-Entwurf
- Daten- und Wissensbasis-Veraltungssysteme
- Automatischer Software-Entwurf

Was muss Rechnerarchitektur umsetzen? Einflusskomplexe

Entwurf eines Rechnersystems

Was sollte eine Rechnerarchitektur alles umsetzen?

We are stuck with technology when what we really want is just stuff that works.

The Salmon of Doubt by Douglas Adams

Was wäre **Ihnen** wichtig?

Entwurf eines Rechnersystems

Kompromissfindung zwischen

- Zielsetzungen
 - Anwendungsbereiche, Funktionalität, Verfügbarkeit,...
- Gestaltungsgrundsätzen
 - Modularität, Sparsamkeit, Fehlertoleranz,...
- Randbedingungen
 - Technologie, Finanzen, Umwelt,...

Entwurf eines Rechnersystems - Zielsetzungen

- Anwendungsbereich
- Verarbeitungsgeschwindigkeit
- (Binärcode-)Kompatibilität
- Stromverbrauch
- Benutzerfreundlichkeit
- Verlässlichkeit/Robustheit
- Erweiterbarkeit/Skalierbarkeit
- Flexibilität (Universalrechner statt Spezialrechner)

Entwurf eines Rechnersystems – Zielsetzungen Anwendungsbereiche

- Technisch-wissenschaftlicher Bereich
 - Strömungsmechanik
 - Materialforschung
 - ...
- Kommerzieller Bereich
 - Datenbankanwendungen
 - Internet, Suchmaschinen
 - Optimierung von Geschäftsprozessen
 - ...
- Eingebettete Systeme
 - Verarbeitung digitaler Medien
 - Automatisierungstechnik
 - Telekommunikation

Entwurf eines Rechnersystems – Zielsetzungen

Benutzerfreundlichkeit

- Beziehung zwischen einem Rechnersystem und Nutzer:
 - Anwendungsprogrammierer
 - Problemorientierte Erstellung von Anwendungen
 - Administration
 - Verwaltung, Wartung verteilter Systemressourcen
 - Bedienung durch nicht-technisches Personal, Anwender
- Gestaltung der Schnittstelle zwischen dem Rechnersystem und seinem Benutzer
 - Ziel der Softwareentwicklung
 - Techniken der benutzerfreundlichen Oberfläche erst durch spezielle Hardware-Techniken möglich

Entwurf eines Rechnersystems – Zielsetzungen

Verlässlichkeit/Robustheit

- Gewährleistung einer minimalen Verfügbarkeit des Systems
 - Bei Ausfällen von Komponenten muss ein betriebsfähiger Kern bereit sein
 - Vielfach Verwendung redundanter Komponenten
 - Wichtig für sicherheitskritische Anwendungen

Entwurf eines Rechnersystems – Zielsetzungen

Erweiterbarkeit/Skalierbarkeit

- Eine Rechnerfamilie ist in Ausbaustufen skalierbar für verschiedene Anwendungen
 - Module sind durch andere mit verbesserter Technologie ersetzbar
 - Architekturentwurf ist langlebiger als eine konkrete Maschine.
- Skalierbarkeit ist ein wesentliches Erfordernis aller Rechnersysteme
→ Chancen auf dem Markt
- z.B. SGI-Slogan:
Paying by growing von Einstiegs-Servern bis zu HPC-Maschinen
- Motivation: Architektur kennen und schätzen lernen → Ideen für neue Projekte

- Uns beschäftigt: Welche architektonischen Voraussetzungen sind für die Erweiterbarkeit erforderlich?

Entwurf eines Rechnersystems – Gestaltungsgrundsätze Konsistenz

- Eigenschaft eines Systems mit folgerichtigem, schlüssigem Aufbau
- Vorausschauender Entwurf einer Rechner- bzw. Prozessorarchitektur, der zu erwartenden Architekturerweiterungen schon Rechnung trägt
- Beispiel: MIPS-Prozessor-Familie R2000/3000/4000/6000/.../12000
 - Reserven im Codierungsrahmen
 - Reserven in Flag-Registern für weitere Steuer- und Statusinformationen
 - Reserven für weitere Exception-Codes
 - Reserven für weitere Co-Prozessoren

Entwurf eines Rechnersystems – Gestaltungsgrundsätze Konsistenz

Codierungsrahmen
für main-OPC (MIPS-
Prozessoren R2000)

28..26 IR-Bits 31..29	0	1	2	3	4	5	6	7
0	SPECIAL	BCOND	J	JAL	BEQ	BNE	BLEZ	BGTZ
1	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
2	COP0	COP1	COP2	COP3	+	+	+	+
3	+	+	+	+	+	+	+	+
4	LB	LH	LWL	LW	LBU	LHU	LWR	+
5	SB	SH	SWL	SW	+	+	SWR	+
6	LWC0	LWC1	LWC2	LWC3	+	+	+	+
7	SWC0	SWC1	SWC2	SWC3	+	+	+	+

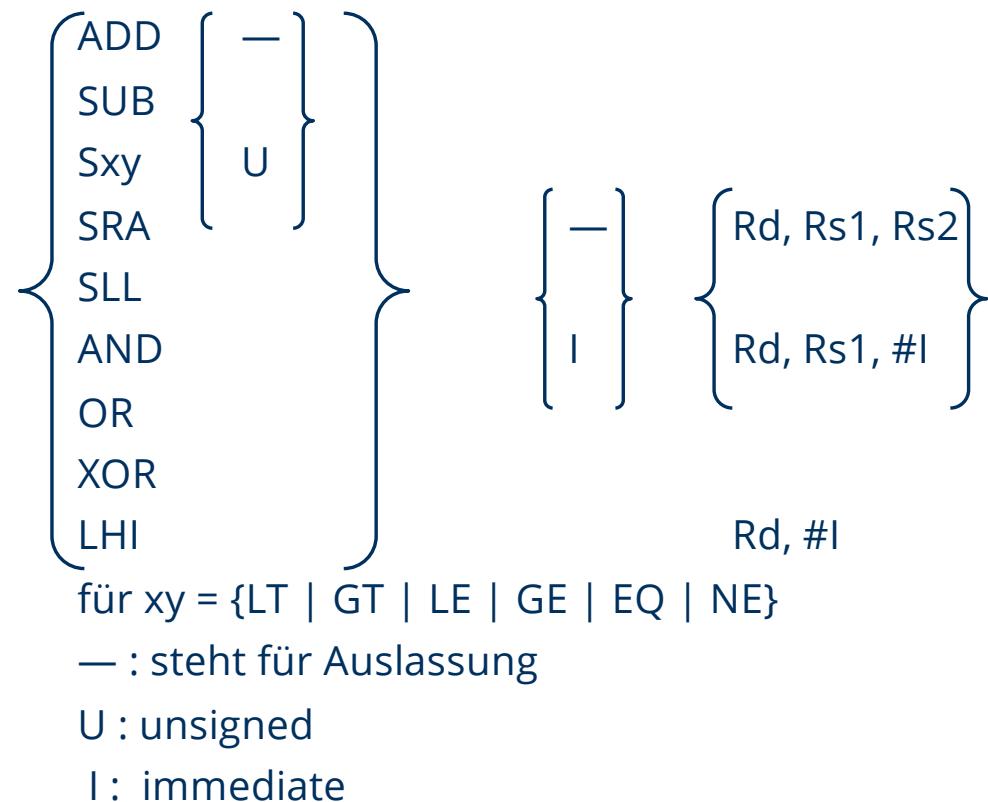
Entwurf eines Rechnersystems – Gestaltungsgrundsätze

Orthogonalität/Modularität

- Funktional unabhängige Teilelemente sind unabhängig voneinander spezifiziert und realisiert
- Standardisierung hat immer größere Bedeutung
- Hauseigene Lösungen ohne Zweitanbieter relativ chancenlos
- Wichtiger Gestaltungsgrundsatz für Befehlssätze
 - vereinfachte Code-Erzeugung in Compilern:
 - spezieller Befehl wird konstruiert durch freie Kombination der Befehlselemente:
 - Operation
 - Datentyp
 - Adressierungsart
 - Registernummern

DLX ALU-Operationen

Zusammengefasste
formale Darstellung
der ALU-
Operationen der
DLX-Architektur



Beispiele für ALU-Operationen der DLX-Architektur

- ADD Rd, Rs1, Rs2
- ADDI Rd, Rs1, #I
- ADDU Rd, Rs1, Rs2
- ADDUI Rd, Rs1, #I

Entwurf eines Rechnersystems - Gestaltungsgrundsätze

Symmetrie

- Gleichartige Dinge werden im System auch gleichartig verarbeitet

Angemessenheit

- Die Elemente eines Systems sind angemessen, wenn ihre Funktionen bei der Lösung der vorgesehenen Problemstellung ausgeschöpft werden

Sparsamkeit

- Kosten des Systems möglichst gering halten. Sie ist in Abhängigkeit von der Technologie zu sehen.
- Die Dimensionierung einer Systemeinheit muss der vorgesehenen Beanspruchung entsprechen.

Entwurf eines Rechnersystems - Gestaltungsgrundsätze

Wiederverwendbarkeit

- Einsatz von Komponenten, die von vorne herein auf allgemeine Verwendbarkeit hin entwickelt wurden

Transparenz/Abstraktion

- Ein System ist transparent, wenn verschiedene Funktionen des Gesamtsystems unsichtbar bleiben
- Verbergen von Systemeigenschaften vor dem Anwender (nicht im Sinne von Durchsichtigkeit)

Virtualität

- Eigenschaft eines Systems, Funktionen anzubieten, die real gar nicht vorhanden sind
- Verbergen der Begrenzung einer Systemimplementierung vor dem Nutzer

Entwurf eines Rechnersystems - Randbedingungen

Technologische

- Erreichbare Taktfrequenz
- Strukturbreite auf Chips
- Anzahl der Transistoren pro Chip
- maximale Verlustleistung (Wärmeabfuhr)

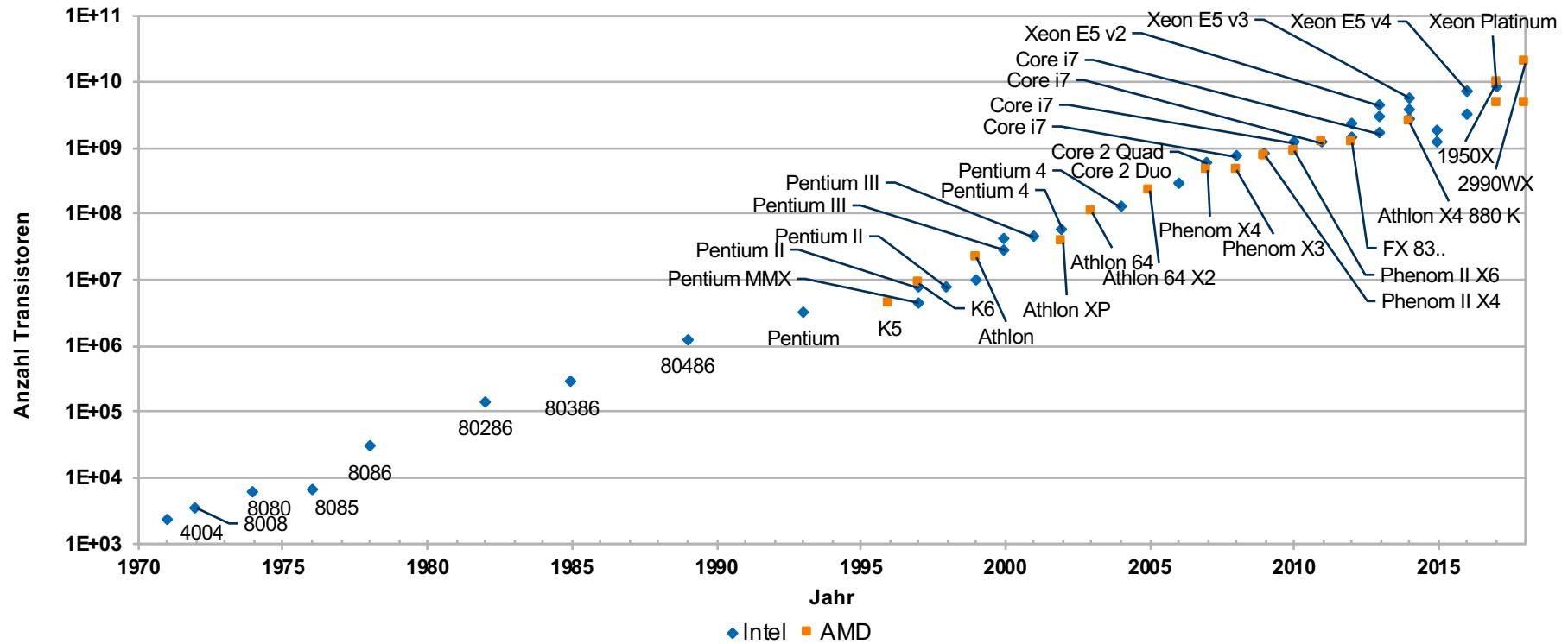
Finanzielle

- Entwicklungskosten für eine neue x86-Prozessorgeneration ca. 400 Millionen Dollar
- Meist Verwendung von Standardprozessorkernen

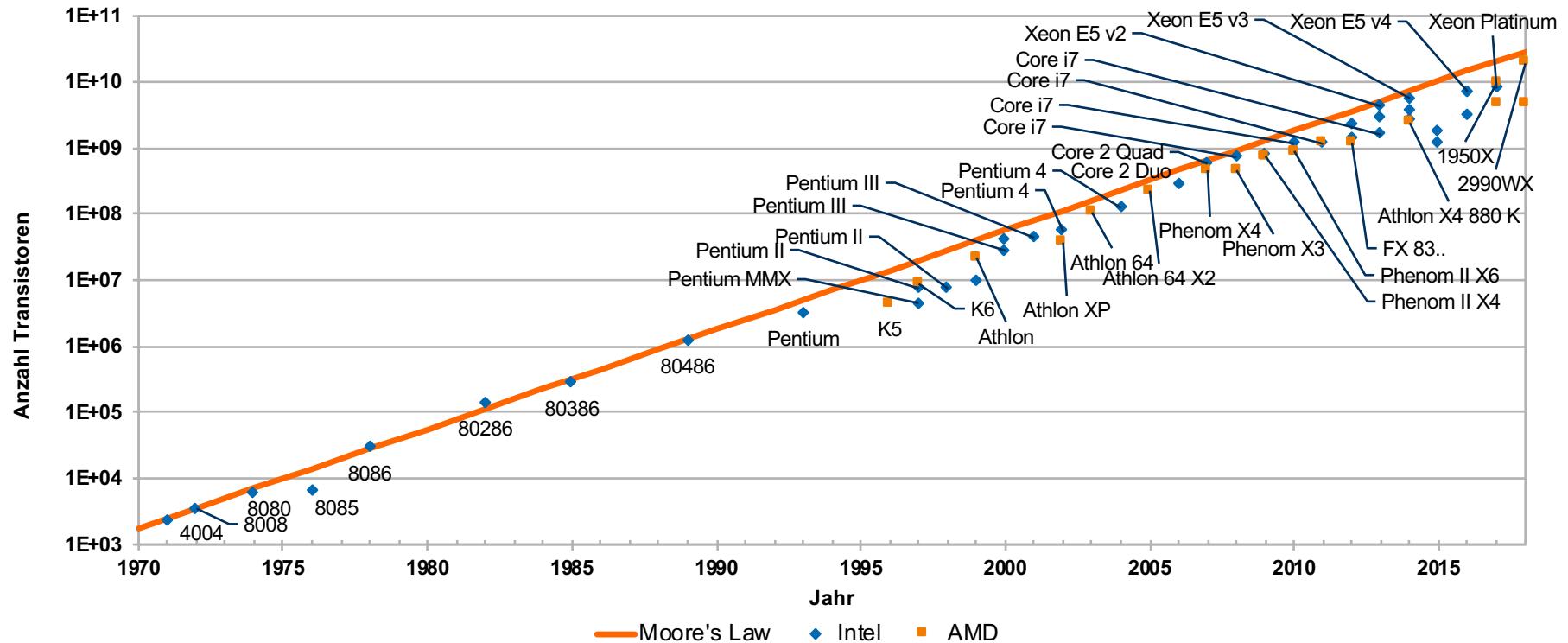
Käuferakzeptanz

- Gestaltungsalternativen bzgl. des Preis-/Leistungsverhältnisses auswählen

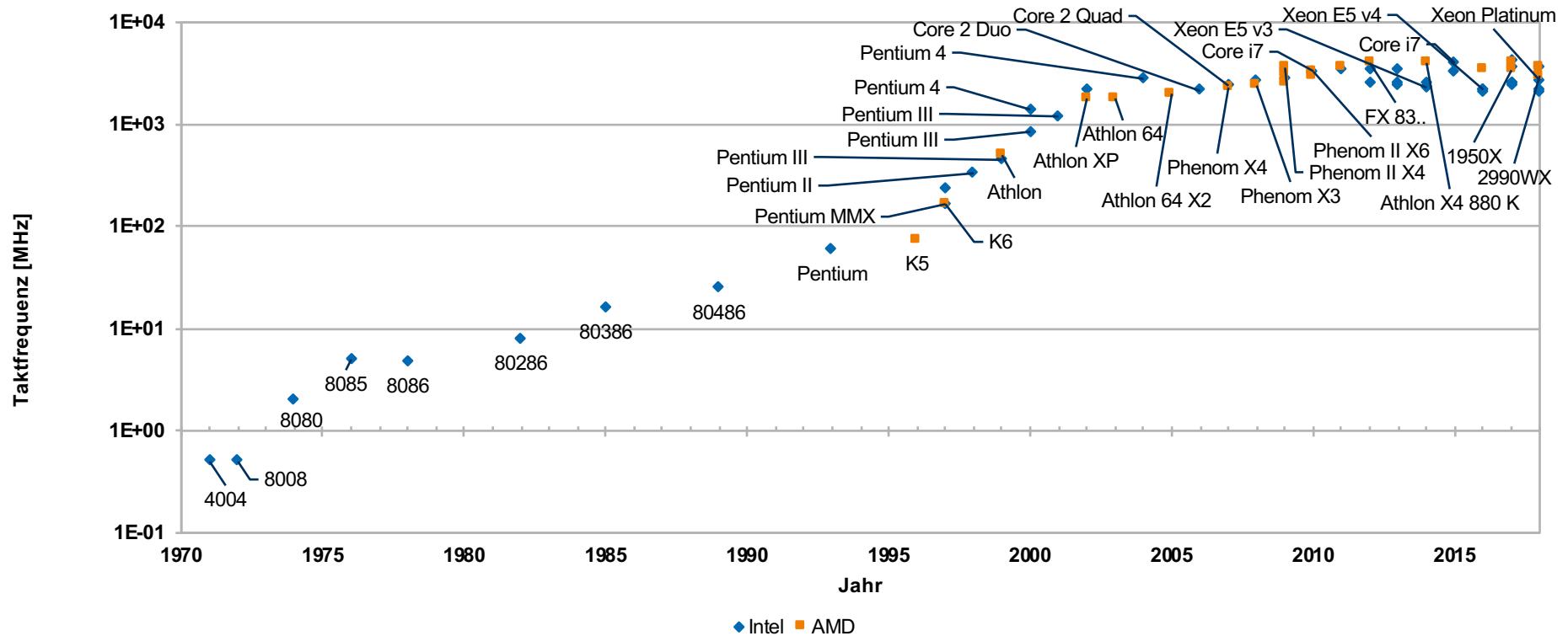
Entwurf eines Rechnersystems – Randbedingungen: Moore's Law



Entwurf eines Rechnersystems – Randbedingungen: Moore's Law

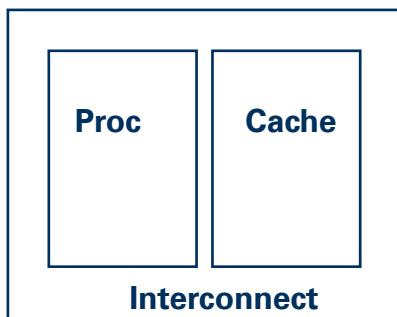


Entwurf eines Rechnersystems – Randbedingungen: Taktfrequenz



Technology: A Closer Look

- Basic advance is decreasing feature size (λ)
 - Circuits become either faster or lower in power
- Die size is growing too
 - Clock rate improves roughly proportional to improvement in λ
 - Number of transistors improves like λ^2 (or faster)
- Performance > 100x per decade
 - clock rate < 10x, rest is transistor count
- How to use more transistors?
 - Parallelism in processing
 - multiple operations per cycle reduces CPI
 - Locality in data access
 - avoids latency and reduces CPI
 - also improves processor utilization
 - Both need resources, so tradeoff
- Fundamental issue is resource distribution, as in uniprocessors

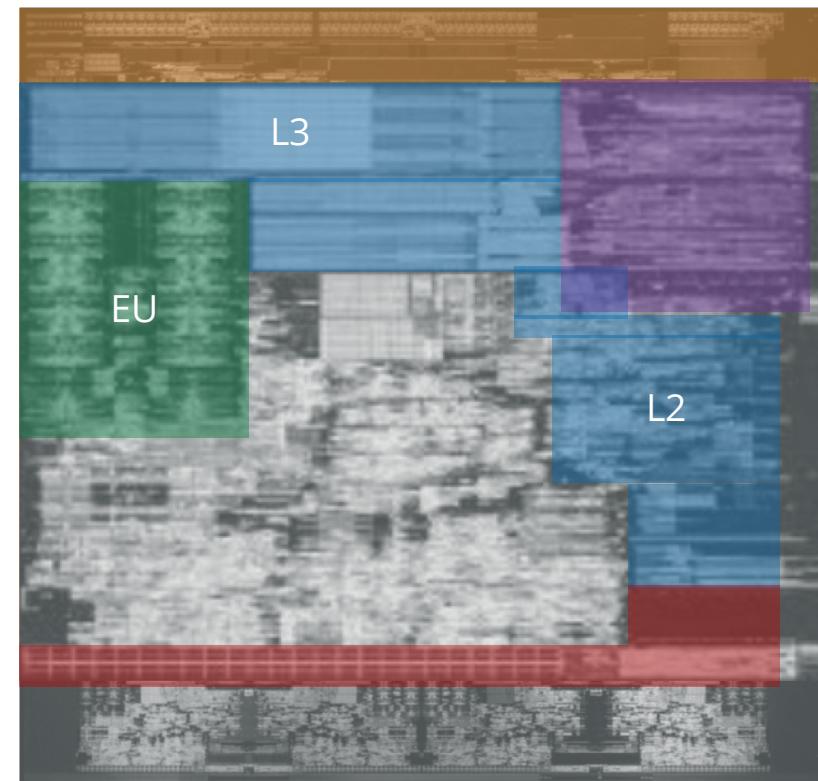


Architectural Trends

- Architecture translates technology's gifts into performance and capability
- Resolves the tradeoff between parallelism and locality
 - Current microprocessor: 1/3 compute, 1/3 cache, 1/3 off-chip connect
 - Tradeoffs may change with scale and technology advances
- Understanding microprocessor architectural trends
 - Helps build intuition about design issues or parallel machine
 - Shows fundamental role of parallelism even in "sequential" computers

Architectural Trends – Transistor Usage

- Ca 1/3 off-chip Kommunikation (PCIe, DRAM, UPI)
- Ca ¼ Cache
- Ca 8% Execution Units
- Remaining?
 - Voltage Control / Power Delivery
 - On-Chip-Network
 - Out-Of-Order Engine
 - Speculative Execution
 - ...



Architectural Trends

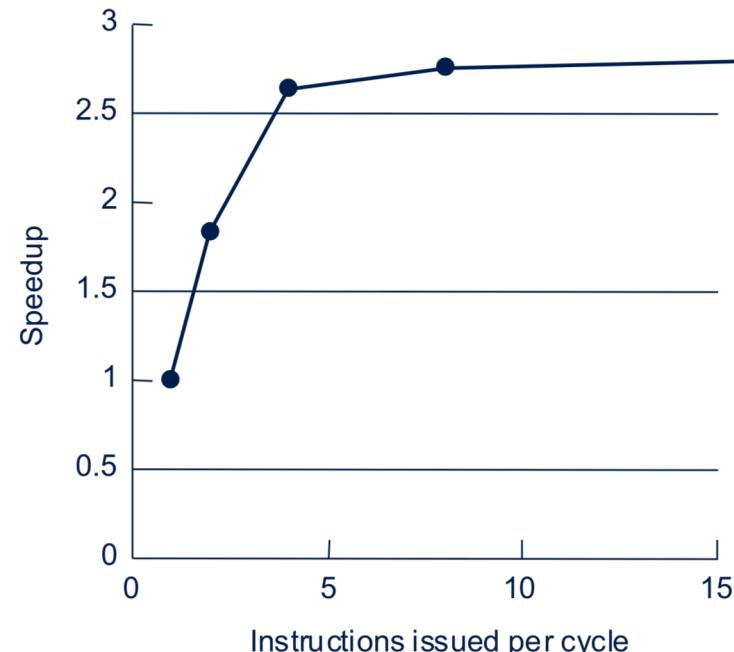
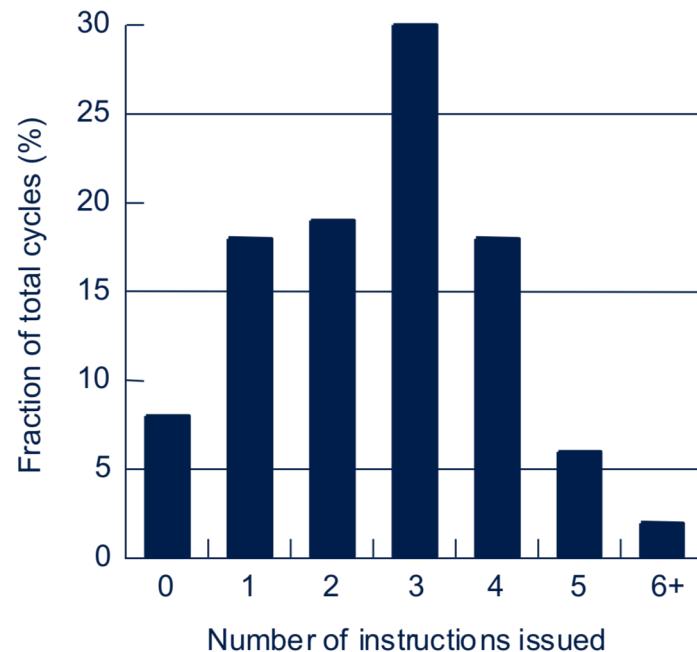
Greatest trend in VLSI generation is increase in parallelism

- Up to 1985: bit level parallelism: 4-bit → 8 bit → 16-bit
 - slows after 32 bit
 - adoption of 64-bit now under way, 128-bit far (not performance issue)
 - great inflection point when 32-bit micro and cache fit on a chip
- Mid 80s to mid 90s: instruction level parallelism
 - pipelining and simple instruction sets, + compiler advances (RISC)
 - on-chip caches and functional units → superscalar execution
 - greater sophistication: out of order execution, speculation, prediction
 - to deal with control transfer and latency problems

Architectural Trends

- End 90s: thread level parallelism and/or chip multiprocessors
 - simultaneous multithreading (virtual processors)
 - multi-core chips (starting with dual-cores)
- Next step: reconfigurable computing

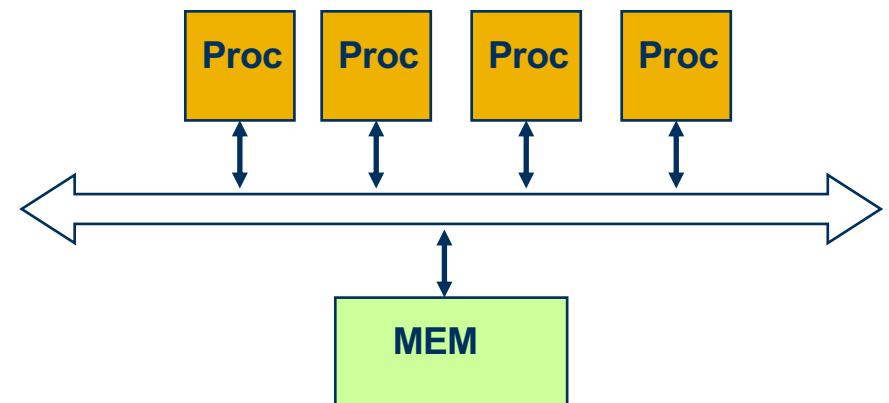
How far will Instruction Level Parallelism go?



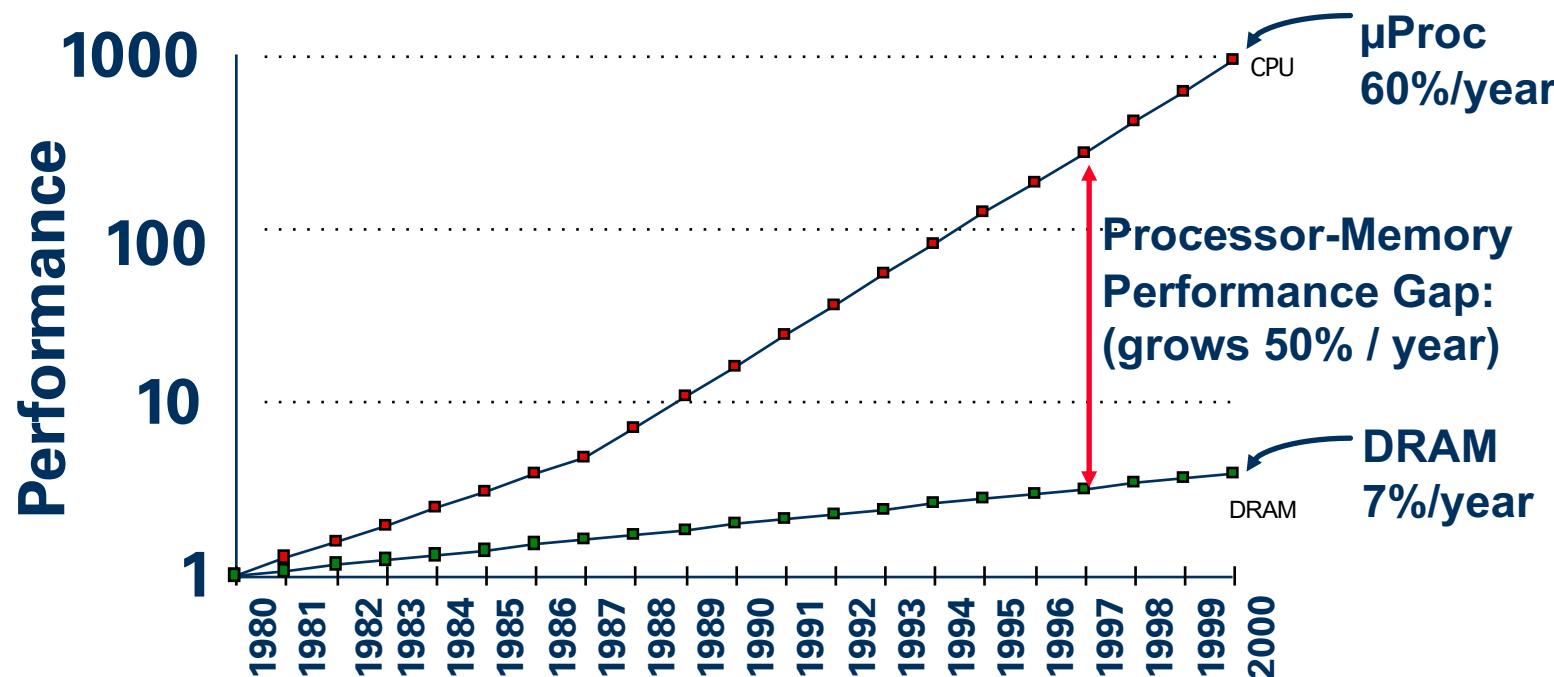
Infinite resources and fetch bandwidth, perfect branch prediction and renaming
real caches and non-zero miss latencies

Thread Level Parallelism “on board”

- Micro on a chip makes it natural to connect many to shared memory
- dominates server and enterprise market, moving down to desktop
- Faster processors began to saturate bus, then bus technology advanced
- today, range of sizes for bus-based systems, desktop to large servers

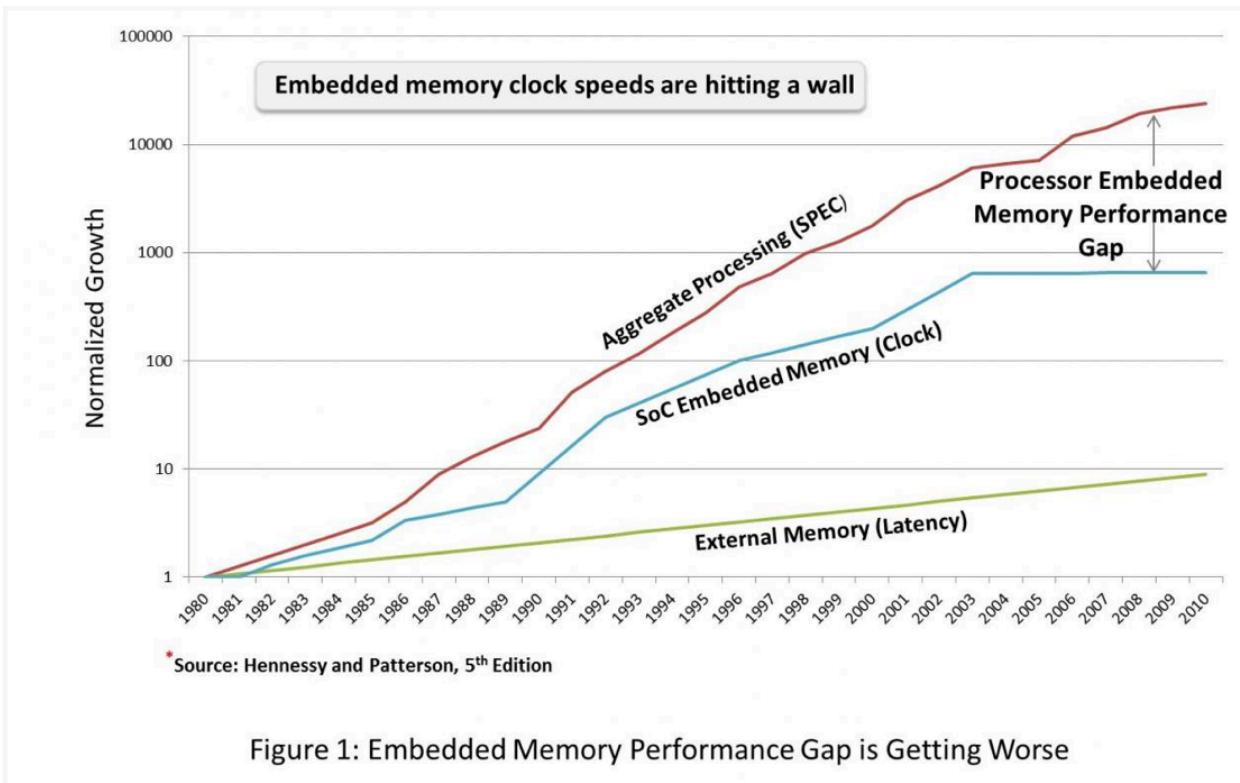


Processor Limit: DRAM Gap (from Gordon Bell, SC'99)



- Alpha 21264 processor full cache miss / instructions executed: $180 \text{ ns}/1.7 \text{ ns} = 108 \text{ clks} \times 4 \text{ or } 432 \text{ instructions}$
- Caches in Pentium Pro: 64% area, 88% transistors (Taken from Patterson-Keeton Talk to SigMod)

Processor Limit: DRAM Gap



Was wäre, wenn das Gehalt ...?

- Parameter
 - 16 Euro ist Basis
 - 54% Wachstum pro Jahr
 - 40 Jahre
- Zunächst 16 Euro → kaufe ein Buch
- Im dritten Jahr 58 Euro → kaufe ein Computer-Spiel
- Im 16. Jahr 16.000 Euro → kaufe ein Auto
- Im 22. Jahr 213.500 Euro → kaufe eine Haus
- Im 40. Jahr ~ $\frac{1}{2}$ Milliarde Euro → kaufe eine Menge

**Man muss fundamental neue Wege finden,
wie das Geld ausgegeben werden soll!!**

Bemerkungen zum klassischen Digitalrechner

Bemerkungen zum klassischen Digitalrechner

- Das Operationsprinzip des klassischen Digitalrechners ist maßgeblich durch die rein sequentielle Programmabarbeitung durch einen einzigen Prozessor bestimmt
- Dieses Prinzip beruht auf:
 - der zentralen Steuerung durch ein Steuerwerk (Leitwerk, Control Processor, Instruction Processor)
 - der zentralen Verarbeitung durch ein Rechenwerk (Register-ALU, Data Processor)

Bemerkungen zum klassischen Digitalrechner Harvard-Architektur

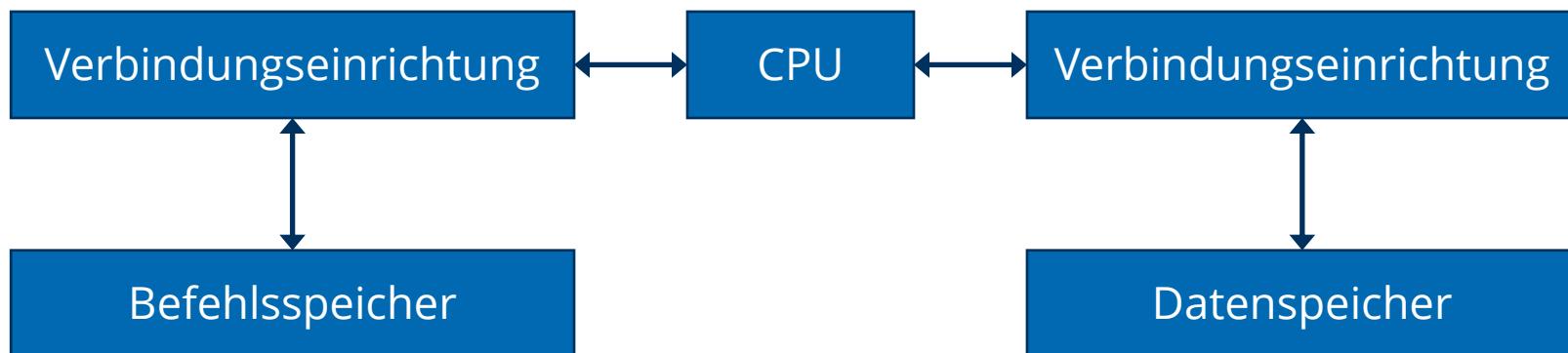
Obwohl sich bei dem klassischen Digitalrechner die v. Neumann-Rechnerarchitektur (1945/46) durchgesetzt hat, gibt es den Harvard-Rechnerarchitektur-Ansatz, der bereits ab 1940 von Howard Aiken in Harvard in Rechnern angewendet wurde [Hen07]:

- Mark-I: elektromechanischer Rechner
- Mark-II: Relaismaschine
- Mark-III: Elektronenröhrenrechner
- Mark-IV: Elektronenröhrenrechner

Bemerkungen zum klassischen Digitalrechner

Harvard-Architektur

(vereinfachte Darstellung)



Die Harvard-Rechnerarchitektur, als Vorläufer des von-Neumann-Rechners, ist gekennzeichnet durch physikalisch getrennte Speicher für Daten und Befehle, d. h. getrennte Buszuordnung

Bemerkungen zum klassischen Digitalrechner Harvard-Architektur - Vor- und Nachteile

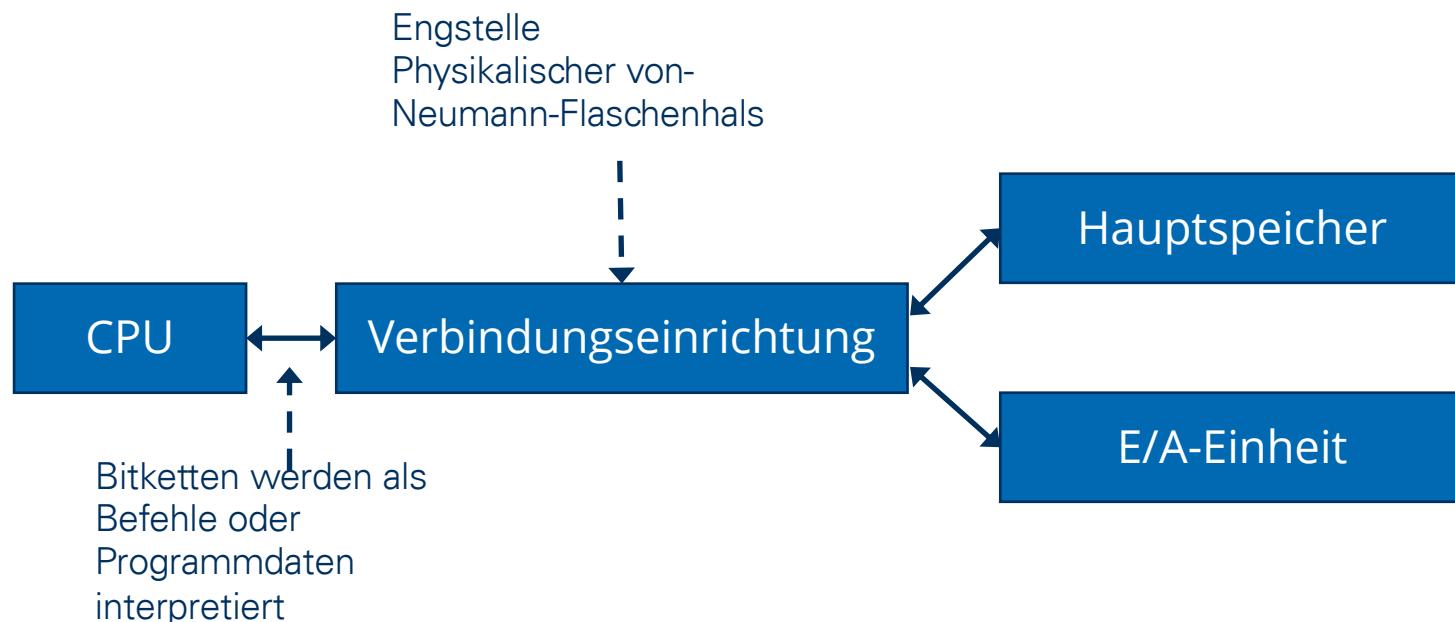
- Vorteile:
 - geringere Wartezeiten
 - einfache Busverwaltung
- Nachteile:
 - höherer Verdrahtungsaufwand (hohe Kabelkosten)
 - verminderte Flexibilität bei der Ausnutzung der Speicher (keine Austauschbarkeit)
- Die Grundidee der Harvard-Architektur wurde bei der Optimierung der Speicherhierarchie wieder aufgegriffen
- First Level Cache wird zumeist in die physikalischen Bestandteile unterteilt:
 - Instruction Cache
 - Data Cache
- Second Level Caches verwenden i.d.R. gemeinsamen Speicher für Befehle und Programmdaten

Bemerkungen zum klassischen Digitalrechner Von-Neumann-Rechnerarchitektur

- Architektur des minimalen Hardware-Aufwands
- Einzigartige Verbindung von
 - Einfachheit (\rightarrow niedrige Kosten)
 - Flexibilität (\rightarrow universell anwendbar)
- Grundelemente sind:
 - Leitwerk und Rechenwerk (CPU)
 - Hauptspeicher
 - Eingabeeinheit und Ausgabeeinheit
 - Verbindungseinrichtung

Bemerkungen zum klassischen Digitalrechner Von-Neumann-Rechnerarchitektur

(vereinfachte Darstellung)



Bemerkungen zum klassischen Digitalrechner Von-Neumann-Rechnerarchitektur

Eigenschaften:

- Durch die Abarbeitung des Programms werden
 - Speicherinhalte geändert
 - Ein- und Ausgaben realisiert
- Die gerade als Vorteile genannten
 - Einfachheit und
 - maximale Flexibilität
- bedingen folgende Nachteile:
 - Befehle und Programmdaten müssen über einen Kanal zwischen Speicher und Prozessor transportiert werden (sog. physikalischer v.-Neumann-Flaschenhals)
 - streng sequentielle Abarbeitung (sog. intellektueller v.-Neumann-Flaschenhals)

Bemerkungen zum klassischen Digitalrechner Von-Neumann-Rechnerarchitektur

Weiterer Nachteil:

- Große semantische Lücke zwischen den für die Benutzungsschnittstelle typischen höheren Programmiersprachen und den im von-Neumann-Speicher enthaltenen von-Neumann-Variablen

Unzureichende Verarbeitungsleistung

Problem 1:

- Die von-Neumann-Variable: das ist der einzelne Speicherplatz, der „Behälter“, aus dem die CPU einzelne Wörter entnehmen bzw. in den sie solche Wörter ablegen kann.

Lösungen 1a zur Verbesserung der unzureichenden Verarbeitungsleistung:

- Substitution skalarer Speicherzellen-Inhalte durch komplexere Informationseinheiten (z.B. Vektoren/Felder)
- Zugriff zu den einzelnen Datenelementen in diesen Strukturdatentypen nicht durch Adressierung seitens der CPU, sondern durch spezielle Hardware-Einrichtungen außerhalb der CPU (z.B. Adressgeneratoren)

Ergebnis:

- Minderung des physikalischen von-Neumann-Flaschenhalses
- Verringerung der semantischen Lücke

Unzureichende Verarbeitungsleistung

Problem 1:

- Die von-Neumann-Variable: das ist der einzelne Speicherplatz, der „Behälter“, aus dem die CPU einzelne Wörter entnehmen bzw. in den sie solche Wörter ablegen kann.

Lösungen 1b zur Verbesserung der unzureichenden Verarbeitungsleistung:

- Organisationsformen, in denen die Daten selbst-identifizierend sind, so dass der Zugriff auf sie nicht durch Adressen , sondern assoziativ, erfolgen kann

Ergebnis:

- Minderung des physikalischen von-Neumann-Flaschenhalses

Ungenügende Abarbeitungssicherheit

Die ungenügende Abarbeitungssicherheit hängt andererseits eng mit den Vorteilen der von-Neumann-Variable zusammen:

- einmaliges Schreiben sichert unbegrenztes Lesen einer von-Neumann-Variable
- theoretisch unbegrenzter Vorrat an Variablen, abhängig von der verfügbaren Adressierungsbreite

Ungenügende Abarbeitungssicherheit

Problem 2a:

- Lesen einer Variable der noch kein gültiger Wert zugewiesen wurde.

Lösungen 2a zur Verbesserung der ungenügenden Abarbeitungssicherheit:

- Einführung einer logischen Variable mit einem Bit „Eventual Value Variable“, die beim Laden des Programms auf „Leer“ gesetzt wird.
- nach dem ersten Schreiben des Wertes wird die logische Variable auf den Zustand „Besetzt“ verändert.
- Damit ist READ AFTER WRITE gesichert!

Ungenügende Abarbeitungssicherheit

Problem 2b:

- Fehlerhafte Adressierung führt zu Fehlinterpretation (z.B. Befehle als Daten und umgekehrt).

Lösungen 2b zur Verbesserung der ungenügenden Abarbeitungssicherheit:

- Einführung einer logischen Variable „Tagged Variable“ im Compiler oder der Rechnerarchitektur, d.h. die von-Neumann-Variable wird um ein „Tag“ (Etikett) zur Kennzeichnung des Datentyps erweitert.
- Rechnerarchitekturen auf dieser Basis werden als „Tagged Architectures“ bezeichnet.
- Dieses Prinzip kann von 1 bit-Tags (Unterscheidung: Instruction/Data) auf Tags mit mehreren Bits erweitert werden, so dass beispielsweise zur Laufzeit Datentyp-Prüfungen möglich sind (in dem Sinne, ob die Datentypen der Quelloperanden zum zugehörigen Befehl passen).

Ungenügende Abarbeitungssicherheit

Problem 2b:

- Fehlerhafte Adressierung führt zu Fehlinterpretation (z.B. Befehle als Daten und umgekehrt).

Lösungen 2b zur Verbesserung der ungenügenden Abarbeitungssicherheit:

- Einführung einer logischen Variable „Tagged Variable“ im Compiler oder der Rechnerarchitektur, d.h. die von-Neumann-Variable wird um ein „Tag“ (Etikett) zur Kennzeichnung des Datentyps erweitert.
- Rechnerarchitekturen auf dieser Basis werden als „Tagged Architectures“ bezeichnet.
- Dieses Prinzip kann von 1 bit-Tags (Unterscheidung: Instruction/Data) auf Tags mit mehreren Bits erweitert werden, so dass beispielsweise zur Laufzeit Datentyp-Prüfungen möglich sind (in dem Sinne, ob die Datentypen der Quelloperanden zum zugehörigen Befehl passen).

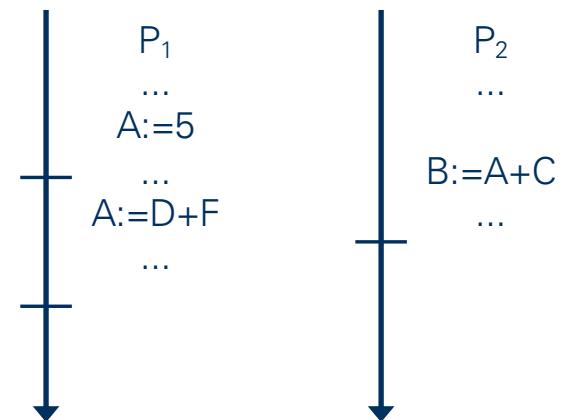
Ungenügende Abarbeitungssicherheit

Problem 2c:

- Beim Einsatz von von-Neumann-Variablen in einer parallelen Verarbeitung kann es zum Zugriff auf nicht aktuelle Werte kommen.

Lösungen 2c zur Verbesserung der ungenügenden Abarbeitungssicherheit:

- Substitution der von-Neumann-Variable durch die „Single Assignment Variable“ der Datenflussarchitektur, d.h. in einer dem Wert zugeordneten logischen Variablen wird gekennzeichnet, ob ein einmal lesbarer Wert geschrieben wurde.



Aufgaben und Ziele der Rechnerarchitektur

Aufgaben der Rechnerarchitektur

Rechnerarchitektur ist eine praxisnahe Forschungsdisziplin, in der schwerpunktmäßig folgende Aufgaben bestehen:

- Architekturanalyse bestehender Rechnersysteme und ihrer Komponenten, wie Prozessoren, Speicher, Verbindungseinrichtungen u.a.
- Beobachtung der Evolution von Rechnerfamilien und Architekturklassen sowie Ableitung neuer Architekturrichtungen
- Entwurf und Synthese neuer leistungsfähiger Rechensysteme mit bewährten Entwurfsmethoden und automatisierten Werkzeugen
- Umsetzung von Leistungsanforderungen, die von Anwendungsbereichen vorgegeben werden, in Struktur und Organisationsformen für Rechner und deren Komponenten

Ziele der Rechnerarchitektur

- Leistungssteigerung durch Architekturverbesserungen
- Steigerung der Nutzerakzeptanz durch benutzergerechte System- und Anwendersoftware
- Entwurf ausbaufähiger Rechnerarchitekturen, die konkurrenzfähig bleiben und Weiterentwicklungen mit reduzierten Kosten gestatten