Angewandte Computer- und Biowissenschaften

HOCHSCHULE MITTWEIDA
University of Applied Sciences

Fraunhofer SIT

Bundeskriminalamt

Betriebssysteme

Betriebssystemarchitektur

Autor: Ronny Bodach, Felix Fischer

Stand: 27.04.2022

Agenda

1. **Grundlegender Aufbau von Rechnern**
2. **Von Neumann Rechnerarchitektur**
3. **Harvard-Architektur**
4. **Computerarchitektur nach Tanenbaum**
    1. **Transistorebene**
    2. **Logische Ebene**
    3. **Microarchitektur**
    4. **Instruction Set Architecture**
    5. **Betriebssystem**
5. Betriebssystem-klassifikation
6. Interrupts
7. Prozess, Task und Thread
8. Scheduling
9. Parallelität und Nebenläufigkeit
10. Speicherverwaltung

Grundlegender Aufbau von Rechnern

Klassifikationsprinzipien

Nach ihrem Rechenprinzip

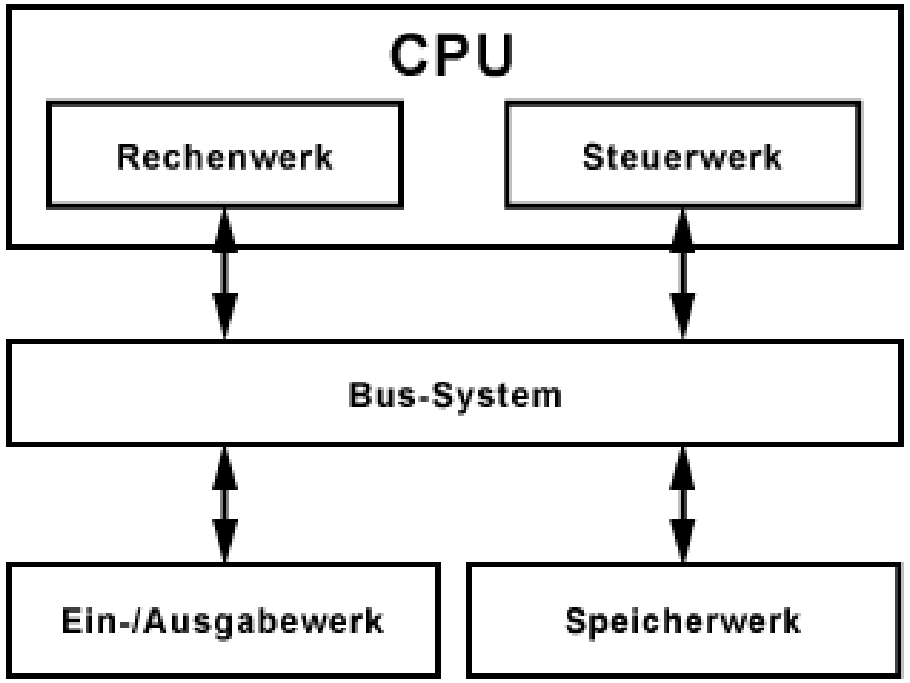
- Von Neumann (Steuerfluss)
- Harvard-Architektur
- Datenfluss (Zündregel, Petrinetze)
- Reduktion (Funktionsaufruf)
- Objektorientiert (Methodenaufruf)

Nach dem Architektur-Grundkonzept

- Vektorrechner (Pipeline)
- Array-Computer (Data-Array)
- Assoziativ-Rechner (Assoziativ-Speicher)

Von Neumann Rechnerarchitektur

- Rechner besteht aus 5 Einheiten:
  - Steuerwerk
  - Rechenwerk
  - Speicher
  - Ein- und Ausgabewerk
- universell (Struktur des Rechners ist unabhängig vom zu lösenden Problem)
- Programm und Daten werden im gleichen Speicher abgelegt
- Speicher sind in gleichgroße Zellen unterteilt, die fortlaufend nummeriert sind



Von Neumann Rechnerarchitektur

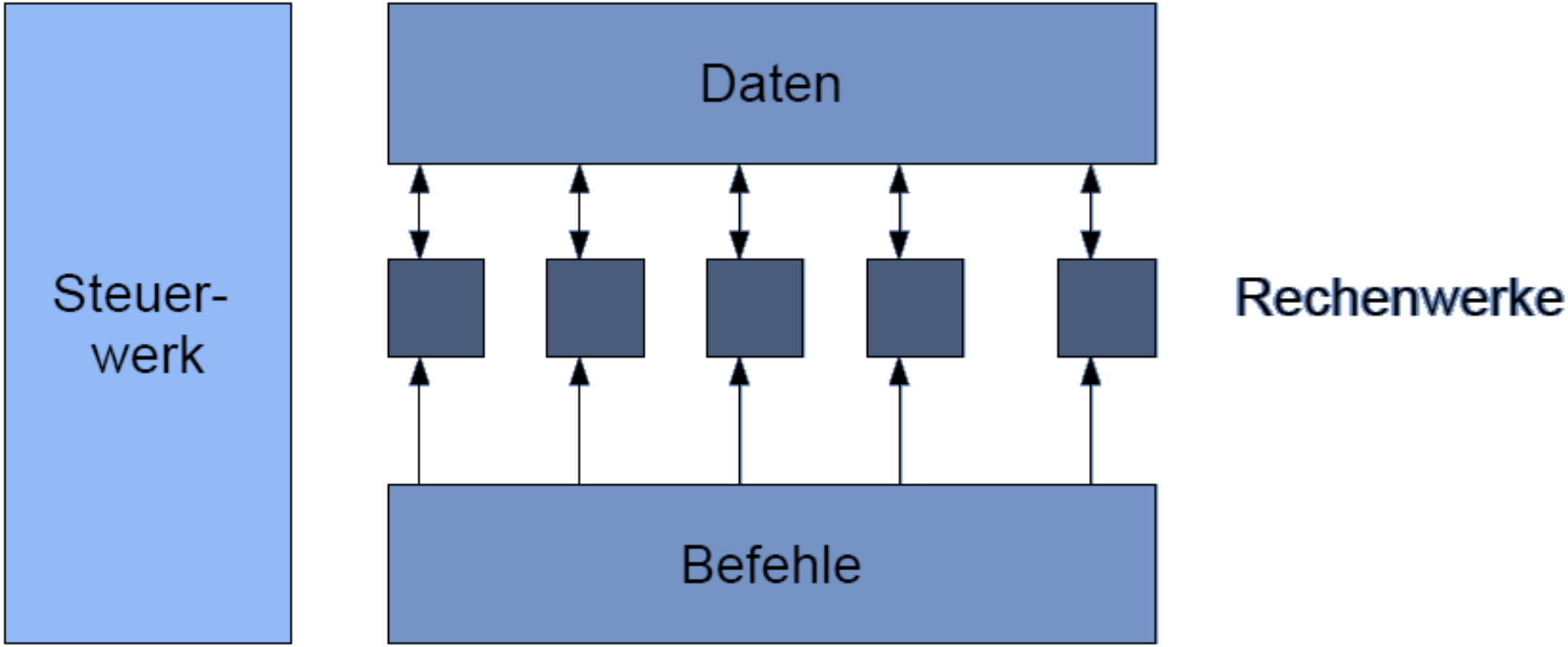
- aufeinander folgende Befehle eines Programms werden in aufeinander folgenden Speicherzellen abgelegt
- durch Sprungbefehle kann von der sequentiellen Bearbeitung abgewichen werden
- es existiert ein Befehlssatz für:
  - arithmetische Befehle
  - logische Befehle
  - Transportbefehle
  - Verzweigungsbefehle
  - sonstige Befehle
- alle Daten werden binär kodiert

Harvard Rechnerarchitektur

Befehlsspeicher ist physisch vom Datenspeicher getrennt

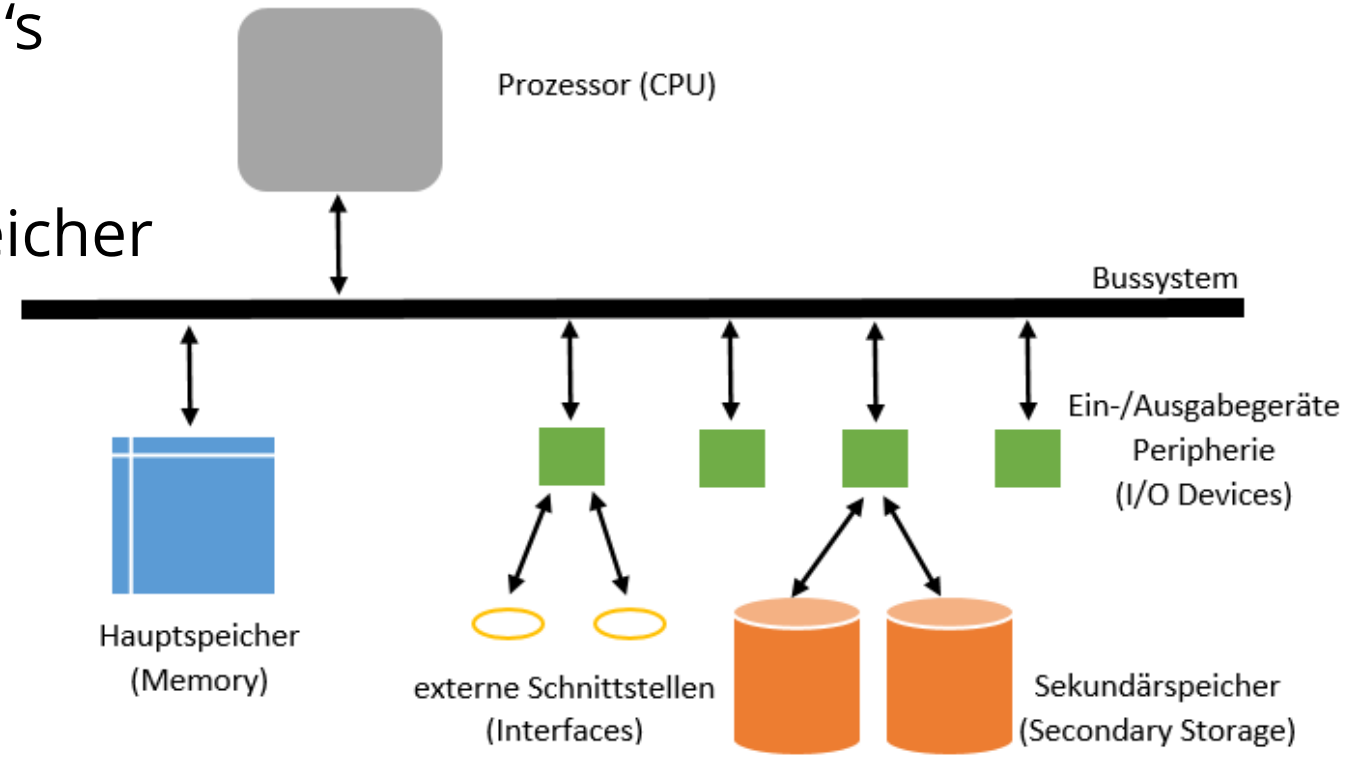
Vorteile	Nachteile
gleichzeitiges Laden und Speichern von Befehlen und Daten	komplexer als Von.-Neumann-Architektur
Programmcode ist schreibgeschützt	Datenspeicher kann nicht als Programmspeicher verwendet werden und umgekehrt
Datenwortbreite kann sich von Befehlswortbreite unterscheiden	

Harvard Rechnerarchitektur



Moderne Rechnerarchitektur

- Mischform aus Von-Neumann- und Harvard-Architektur
- getrennte Daten- und Befehlsbereiche
- getrennte Caches und MMU‘s
- getrennte Busse
- externen gemeinsamen Speicher



Leistungserhöhung

Gebiet	Maßnahme
Architektur	Pipelines, Superskalarität, Spekulative Ausführung, Caches, Busbreite
Optimierung von Software	Compileroptimierung, Parallelisierung, effizientere Algorithmen
Siliziumbasis	Transistordichte und Taktraten

Computerarchitektur nach Tanenbaum

Computerarchitektur nach Tanenbaum

Aufteilung in Ebenen:

- Anwendungsebene (Anwendungssoftware)
- Assemblerebene (Beschreibung von Algorithmen, Link und Bind)
- Betriebssystem (Speichermanagement, Prozesskommunikation)
- Instruction Set Architecture (6502, 8080, z80, x86, x64, Arch64)
- Microarchitektur (RISC, CISC)
- Logische Ebene (Arithmetik, Register, Schieberegister)
- Transistorebene (Transistoren, MOS, Flip Flop)

Computer-architektur nach Tanenbaum

- Anwendungsebene
- Assemblerebene
- Betriebssystem
- Instruction Set Architecture
- Microarchitektur
- Logische Ebene
- **Transistorebene**

Transistorebene

Computer kennen nur zwei Zustände mit Informationsgehalt:

- an, aus
- Strom fließt, fließt nicht
- wahr (true), falsch (false)
- Magnetfeldänderung, keine Magnetfeldänderung
- Höhenänderung, keine Höhenänderung
- 1, 0

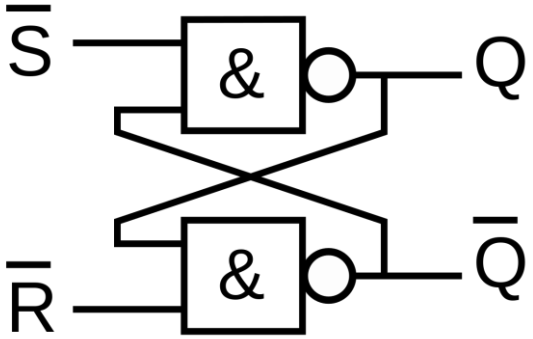
Transistorebene

Computer-Hardware arbeitet (fast) ausschließlich auf dieser digitalen binären Grundlage:

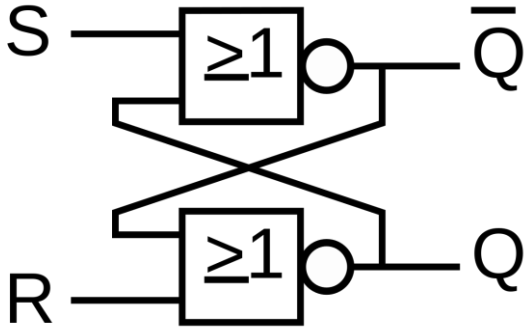
- Rechenoperationen der Informationsverarbeitung:
  - UND
  - ODER
  - NOT
- Informationsspeicherung erfolgt Bit-weise, d.h. in Form einzelner Nullen und Einsen

Speicherung von Bits

- Speicherung durch elektrischer Ladung
- z.B. durch Kondensatoren oder Flip-Flops.
- Einzelne Flip-Flops (Transistor/Bausteine) speichern eine Ziffer (1 Bit):
  - elektrische Ladung vorhanden → 1
  - elektrische Ladung nicht vorhanden → 0



Flip-Flop mit NAND-Gattern



Flip-Flop mit NOR-Gattern

Register

Mehrere Flip-Flops (Register) speichern eine komplette Zahl in Registerbreite:

- 8 Bit = 1 Byte
- Wort = 16 Bit / 32 Bit (Word)
- Doppelwort = 32 Bit / 64 Bit (Double Word, DWord)
- Vierfachwort = 64 Bit / 128 Bit (Quad Word, QWord)

Die Wortbreite hängt vom Prozessor und seiner Architektur ab ebenso das Byte-Ordering.

Endian

- Little-Endian
  - niedrigste Wertigkeit zuerst
  - verwendet von x64
  - Hundertzwanzig = 021

- Big-Endian
  - größte Wertigkeit zuerst
  - wie altägliches Dezimalsystem
  - Hundertzwanzig = 120

Computer-architektur nach Tanenbaum

- Anwendungsebene
- Assemblerebene
- Betriebssystem
- Instruction Set Architecture
- Microarchitektur
- **Logische Ebene**
- Transistorebene

Informationsverarbeitung

- Rechnerarithmetik
  - Addition, Subtraktion, Multiplikation, Division
  - Festkomma- und Gleitkommaarithmetik

- Natürliche Zahlen
  - in positionaler Darstellung: Ziffern, Position der Ziffern als Gewichtung
  - Zahlen sind darstellbar in jeder beliebigen Basis b

Zahlenbasis

Basis	Zeichen	Beispiel 140
2 (binär)	0, 1	10001100
8 (oktal)	0, 1, 2, 3, 4, 5, 6, 7	214
10 (dezimal)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	140
16 (hexadezimal)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	8C

Boolsche Operationen

- Konjunktion AND
- Disjunktion OR
- Negation NOT
- Kontravalenz XOR

A	B	A and B	A or B	not A	A xor B
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

Beispiel Binäradition:

```
      10011
    + 1001
-----------
       11   Übertrag (Carry)
===========
Ergebnis: 11100
```

Overflow und Underflow

	1	0	1	1	0	1	0	1	Register A
+	1	0	0	1	1	0	0	0	Register B
1	0	1	0	0	1	1	0	1	Ergebnis im Register

Wird nicht im Register gespeichert.
Setzt jedoch Carry-Flag.

	0	0	1	1	0	1	0	1	Register A
-	1	0	0	1	1	0	0	0	Register B
1	1	0	0	1	1	1	0	1	Ergebnis im Register

Lösungen Over- und Underflow

- Lösung 1: Halbaddierer mit Carrybit
- Lösung 2: Volladdierer mit Carrybit in neuen Eingang
- Lösung 3: Paralleles Addierwerk aus Volladdierern

Darstellung negativer Zahlen

- durch Vorzeichen und Betrag
- ein Bit repräsentiert das Vorzeichen
- die anderen Bits repräsentieren den Betrag der Zahl
- Beispiel:

  01001 = **+**9

  11001 = **-**9

- Nachteile:
  - explizite Auswertung des Vorzeichens
  - zwei Nullen

Einerkomplement

Eine negative Zahl -x wird binär durch das bitweise Komplement der entsprechenden positiven Zahl x dargestellt.

- Komplementierung (6) = 0110 wird zu (−6) = 1001.
- Beispiel für n=4:

Dezimal	0	1	2	3	4	5	6	7
Binär	0000	0001	0010	0011	0100	0101	0110	0111

Dezimal	-0	-1	-2	-3	-4	-5	-6	-7
Binär	1111	1110	1101	1100	1011	1010	1001	1000

zwei Nullen

Einerkomplement

Die Addition/Subtraktion muss in mehreren Schritten erfolgen:

1. Normale Binäraddition
2. Zusätzliche Addition eines eventuellen Übertrags aus der ersten Binäraddition
3. die bei der 2. Addition entstehenden Überträge werden gestrichen

$$\begin{array}{r} 5 \\ +-6 \\ \hline =-1 \end{array} \qquad \begin{array}{rr} 0101 & (5) \\ +1001 & (-6) \\ \hline =1110 & (-1) \end{array}$$

$$\begin{array}{r} 5 \\ +-3 \\ \hline =\ \ 2 \end{array} \qquad \begin{array}{rr} 0101 & (5) \\ +1100 & (-3) \\ \hline =1|0001 & (-14) \\ +1 & (1) \\ \hline =\not{1}|0010 & (2) \end{array}$$

Zweierkomplement

Beim Zweierkomplement werden die negativen Zahlen durch Bestimmung des Einerkomplements und einer zusätzlichen Addition von 1 gebildet.

- 2er Komplementierung (6)=0110 wird zu 1001+0001=(-6)= 1010.
- Beispiel für n=4:

Dezimal	0	1	2	3	4	5	6	7
Binär	0000	0001	0010	0011	0100	0101	0110	0111

Dezimal	-1	-2	-3	-4	-5	-6	-7	-8
Binär	1111	1110	1101	1100	1011	1010	1001	1000

Zweierkomplement

- Addition
  - normale vorzeichenlose Binäraddition
- Subtraktion
  1. Negation der zu subtrahierenden Zahl
  2. Addition der negierten Zahl

Beispiel: (5 − 6) = (5 + (−6))

$$\begin{array}{r} 5 \\ +-6 \\ \hline =-1 \end{array} \qquad \begin{array}{rr} 0101 & (5) \\ +1010 & (-6) \\ \hline =1111 & (-1) \end{array}$$

$$\begin{array}{r} 5 \\ +-3 \\ \hline =\ 2 \end{array} \qquad \begin{array}{rr} 0101 & (5) \\ +1101 & (-3) \\ \hline =\not{1}|0010 & (2) \end{array}$$

Festkomma-Darstellung von Zahlen

- feste Kommaposition (Beispiel: Eurobetrag in Centschreibweise)
- nur bei spezieller Hardware zum Einsatz, wie z.B. digitalen Signalprozessoren (DSPs)

- Beispiel: n = 4, Komma an Position k = 2
- Registerinhalt 0110 bedeutet:
  - 01,10 binär
  - 1,5 dezimal ($0 * 2^1 + 1 * 2^0 + 1 * 2^{-1} + 0 * 2^{-2} = 2^0 + 2^{-1} = 1{,}5$)
- negative Zahlen analog (Vorzeichen erstes Bit)

Gleitkommadarstellung

- Ziel: Darstellung ***großer und kleiner Zahlen*** mit gleichem Verfahren
- Gleitkomma in engl. floating point
- Datentyp real, float oder double aus gängigen Programmiersprachen
- Annäherung an reellen Zahlen
- Umsetzungsidee:
  - Nutzung einer wissenschaftlichen Potenzschreibweise
  - Darstellung einer Anzahl von Ziffern *(Mantisse)* plus
  - Darstellung der Position des Kommas (Gleitkomma) durch Exponenten

Gleitkommadarstellung

Bei der Potenzschreibweise im wissenschaftlichen Bereich wird das Komma durch den Exponenten in der Stelle verschoben:

- Verschiebung nach rechts für kleine Zahlen:

$$\mathbf{0,00021 = 0,21 * 10^{-3}}$$

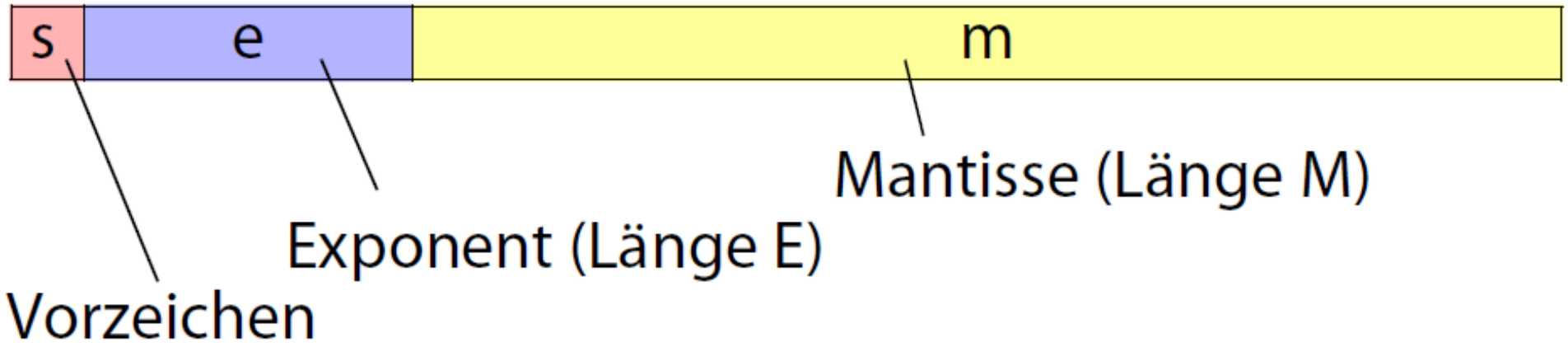
- Verschiebung nach links für große Zahlen:

$$\mathbf{1600000,00021 = 0,160000000021 * 10^{7}}$$

Die binäre Gleitkommadarstellung erweitert die Festkommadarstellung um einen binär-kodierten Exponenten.

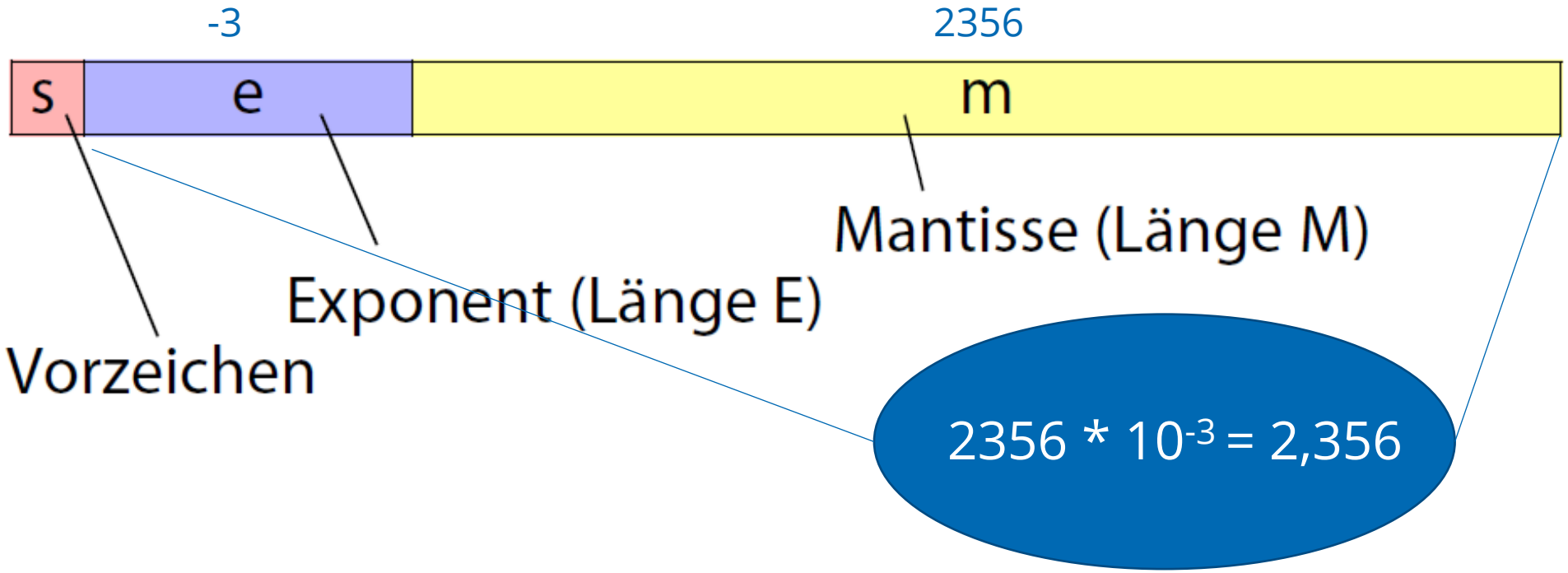
Gleitkommadarstellung

- IEEE 754 Standard definiert Gleitkommadarstellung
- Aufbau einer IEEE 754 Gleitkommazahl:



Gleitkommadarstellung

- IEEE 754 Standard definiert Gleitkommadarstellung
- Aufbau einer IEEE 754 Gleitkommazahl:



IEEE 754 Formatdefinitionen

	Single Precision	*Double Precision*	*Quad Precision*
Gesamtlänge *(N)*	32 Bit	64 Bit	128 Bit
Vorzeichen	1 Bit	1 Bit	1 Bit
Mantisse *(M)*	23 Bit	52 Bit	112 Bit
Exponent *(E)*	8 Bit	11 Bit	15 Bit
Bias *(B)*	127	1023	16383
$\|x_{min}\|$ (norm.)	$2^{-126} \approx 10^{-38}$	$2^{-1022} \approx 10^{-308}$	$2^{-16382} \approx 10^{-4932}$
$\|x_{min}\|$ (denorm.)	$2^{-149} \approx 10^{-45}$	$2^{-1074} \approx 10^{-324}$	$2^{-16492} \approx 10^{-4965}$
$\|x_{max}\|$	$(2 - 2^{-23}) * 2^{127} \approx 10^{38}$	$(2 - 2^{-52}) * 2^{1023} \approx 10^{308}$	$(2 - 2^{-112}) * 2^{16383} \approx 10^{4932}$

Zusammenfassung Binäre Arithmetik

- Boolesche Algebra:
  - Grundlage für die Computer-Hardware
  - Werte 0 und 1 zum Rechnen
  - Operationen der Konjunktion, Disjunktion und Negation
- Addition:
  - Halbaddierer addiert zwei Bits
  - produziert Summe und Carry
  - Volladdierer addiert zwei Bits und Carry
  - üblicherweise Addierwerk aus Volladdierern
- Subtraktion:
  - 2er-Komplement

Repräsentation von Texten

- Einfacher Ansatz:
  - Codierung A=0, B=1, C=10 … und dann fortgesetzte Binärcodierung
- Probleme:
  - Welche Sonderzeichen existieren im Zeichensatz?
  - Welche Zeichen sollen wie codiert werden?
  - ➢ Austausch von Texten zwischen Computern

Lösung: Standardisierte Zeichensätze

ASCII - Codierung

- American Standard Code for Information Interchange – ASCII
- verabschiedet 1963 von der American Standards Organization
- für die USA entwickelt
- 7-Bit Code
- codiert Zeichen
- Steuercodes zur Kontrolle von Geräten
  z.B. 0x0D CR (Carriage Return) für Wagenrücklauf bei Druckern

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	*NUL*	*SOH*	*STX*	*ETX*	*EOT*	*ENQ*	*ACK*	*BEL*	*BS*	*HT*	*LF*	*VT*	*FF*	*CR*	*SO*	*SI*
1...	*DLE*	*DC1*	*DC2*	*DC3*	*DC4*	*NAK*	*SYN*	*ETB*	*CAN*	*EM*	*SUB*	*ESC*	*FS*	*GS*	*RS*	*US*
2...	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

ISO-8859-1 / ISO Latin 1

- Motivation / Problem:
  - viele wichtige Zeichen fehlen bei ASCII 😟
- Ansatz:
  - „Längere Codierung“
  - Erweiterung des Zeichensatzes durch ISO-8859-1 / ISO Latin 1
  - ISO = International Organization for Standardization
  - 8-Bit Code mit vielen Sonderzeichen für westeuropäische Sprachen
  - z.B. ß, ä, â, à, á, î, ì, í, ö, ê, è, ë, é, æ, £, …

ISO-8859-1 / ISO Latin 1

Problem:

Einige wichtige Zeichen fehlen

- französische Sonderzeichen
- €-Symbol
- 😟 Emojis

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	*nicht belegt*															
1...																
2...	*SP*	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8...	*nicht belegt*															
9...																
A...	*NBSP*	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	*SHY*	®	¯
B...	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C...	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D...	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E...	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F...	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Unicode 

- verwaltet vom Unicode-Konsortium (http://www.unicode.org)
- verschiedene Codierungsformate (UTF - Unicode Transformation Format):
  - UTF-8 (8 Bits)
  - UTF-16 (16 Bits)
  - UTF-32 (32 Bits)
- längere Unicode-Formate ergänzen kürzere Formate
- Zusatzinformationen wie z.B. Schreibrichtung
- kontinuierlich um neue Zeichen ergänzt
- Ziel:
  - Codierung aller in Gebrauch befindlicher Schriftsysteme und Zeichen

Unicode

- derzeitiger defacto Standard
- 17 Planes zu je 65.536 Zeichen
- 6 Planes derzeit genutzt
- restliche Planes für spätere Nutzung

- Basic Multilingual Plane (BMP, 0):
  - grundlegender mehrsprachiger Codebereich
  - enthält aktuell gebräuchliche Schriftsysteme
  - Satzzeichen und Symbole

Unicode

- Supplementary Multilingual Plane (SMP, 1):
  - historische Schriftsysteme
  - weniger gebräuchliche Zeichen
  - z.B. Domino- und Mahjonggsteine

- Supplementary Ideographic Plane (SIP, 2):
  - ergänzender ideographischer Bereich
  - selten benutzte fernöstliche CJK-Schriftzeichen
  - CJK = China, Japan, Korea

Unicode Transformation Format (UTF) Forensische Bedeutung

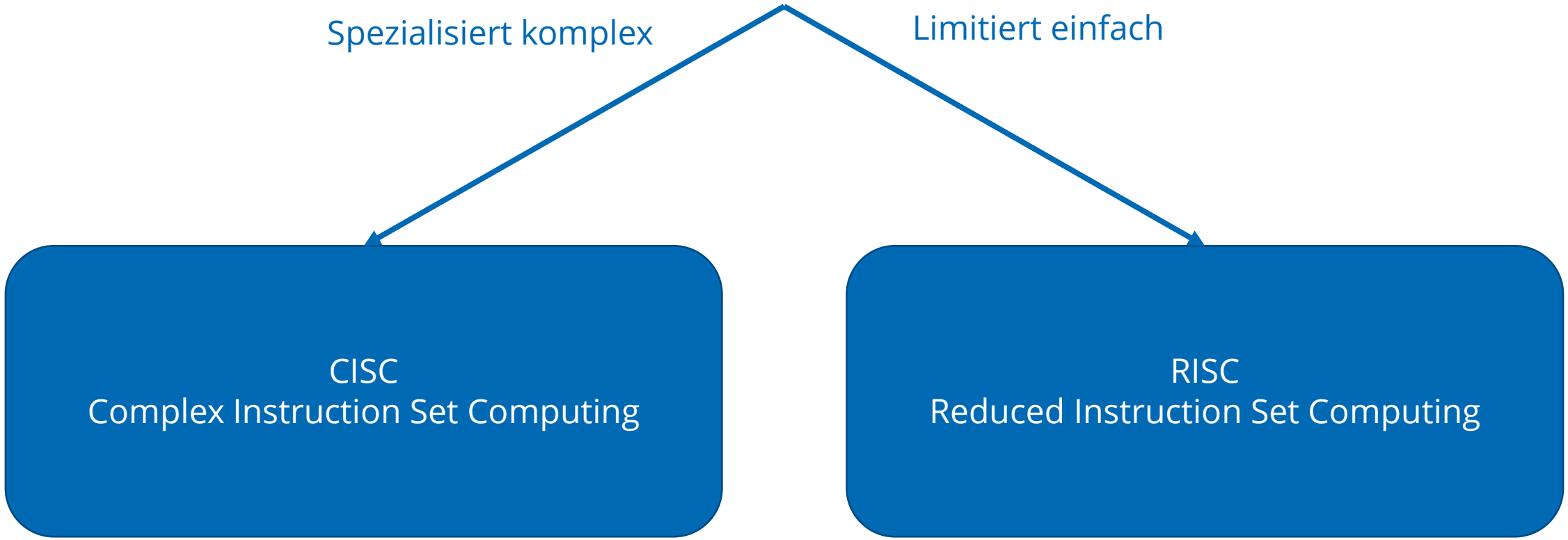
Veränderung (de)

```
56 00 00 00|65 00 00 00|72 00 00 00|E4 00 00 00|6E 00 00 00|64 00 00 00  | UTF-32LE
00 00 00 56|00 00 00 65|00 00 00 72|00 00 00 E4|00 00 00 6E|00 00 00 64  | UTF-32BE
V          |e          |r          |ä          |n          |d             | Veränd
65 00 00 00|72 00 00 00|75 00 00 00|6E 00 00 00|67 00 00 00               | UTF-32LE
00 00 00 65|00 00 00 72|00 00 00 75|00 00 00 6E|00 00 00 67               | UTF-32BE
e          |r          |u          |n          |g                         | erung
56 00|65 00|72 00|E4 00|6E 00|64 00|65 00|72 00|75 00|6E 00|67 00        | UTF-16LE
00 56|00 65|00 72|00 E4|00 6E|00 64|00 65|00 72|00 75|00 6E|00 67        | UTF-16BE
V    |e    |r    |ä    |n    |d    |e    |r    |u    |n    |g             | Veränderung
56|65|72|C3 A4|6E|64|65|72|75|6E|67                                      | UTF-8
V |e |r |ä    |n |d |e |r |u |n |g                                       | Veränderung
```

Computer-architektur nach Tanenbaum

- Anwendungsebene
- Assemblerebene
- Betriebssystem
- Instruction Set Architecture
- **Microarchitektur**
- Logische Ebene
- Transistorebene

Grundlegender Aufbau der Microarchitektur



CISC (Complex Instruction Set Computing)

- großer Befehlsumfang
- Anweisungen mit komplexen Aufgaben
- Komplexe Adressierungsmöglichkeiten
- optimiert auf Spezialaufgaben
- AMD x64 / Intel ia64 (intern jedoch auch RISC)

Funktionsprinzip CISC

- Befehl durch Mikrocode im ROM-Speicher des CISC-Prozessors definiert
- Decoder-Einheit teilt Befehl in:
  - Maschinenbefehl
  - Adressierungsart
  - Adressen
  - Register
- Befehlsausführung:
  - Mikrocode (atomare Anweisungen) an Nanoprozessor geschickt
  - Nanoprozessor schaltet komplexe Schaltkreise

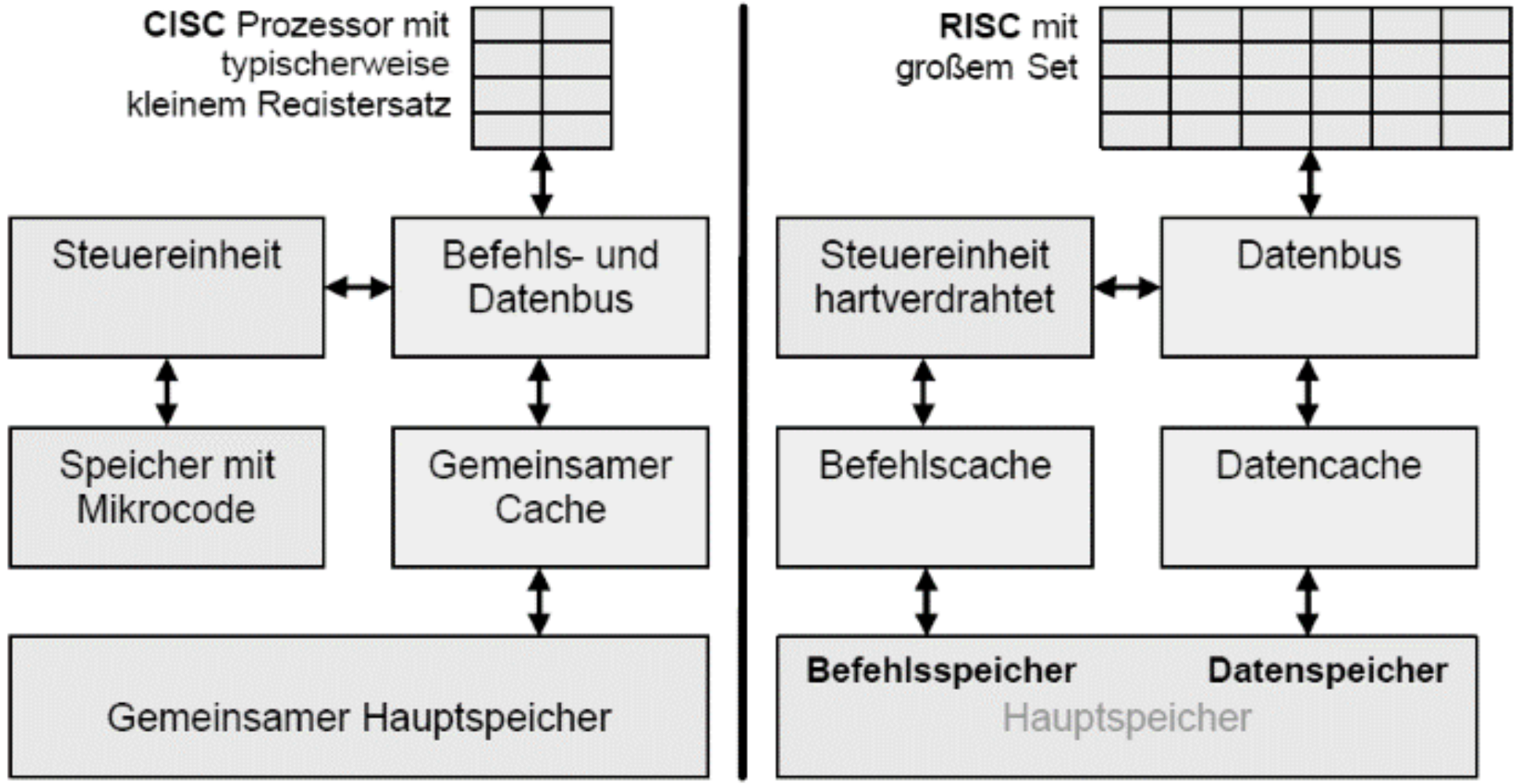
RISC (Reduced Instuction Set Computer)

- sehr wenige Anweisungen
- einfache, elementare Instruktionen
- optimiert auf Pipelining
- Ziel ein Takt pro Befehl (superskalar)
- RISC-Befehle viel schneller als CISC-Befehle geladen
- einfacheren Prozessoraufbau
- einfache, digitale, fest verdrahtete Schaltungen
- ARM aarch64, PowerPC, OpenRISC

Funktionsweise RISC

- unabhängige Verarbeitungseinheiten
- mehrere getrennte, interne Bussysteme
- Parallelverarbeitung der Befehle
- alle RISC-Befehle haben das gleiche Format
- nur eine Möglichkeit sie zu laden oder zu speichern

Gegenüberstellung RISC / CISC



Computer-architektur nach Tanenbaum

- Anwendungsebene
- Assemblerebene
- Betriebssystem
- **Instruction Set Architecture**
- Microarchitektur
- Logische Ebene
- Transistorebene

Formale Spezifikation

- Beschreibung des Befehlssatzes
- binärer Kodierung des Befehlssatzes
- Beschreibung der CPU-Verhaltensweise während Betriebszustände und Ereignisse
  - Verhalten der CPU bei Unterbrechungsanforderung
  - Startadresse der Befehlsabarbeitung
  - Initialisierung der Register nach Reset
  - Aufbau wichtiger Datenstrukturen

Mikroprozessor

Man spricht davon, dass ein Mikroprozessor eine Befehlssatzarchitektur implementiert bzw. unterstützt, wenn er alle im Sinne der Regeln dieser Befehlssatzarchitektur gültigen Programme in der vorgesehenen Art und Weise ausführen kann.

- viele Befehlssatzarchitekturen historisch gewachsen
- Abwärtskompatibilität nicht immer gegeben
- in Praxis:
  - undokumentierte Eigenschaften
  - vermeintlich unbedeutende Details
  - ➢ Bestandteil einer Befehlssatzarchitektur

Virtuelle Maschine

- Befehlssatzarchitektur = formale Definition
- Implementierung kann virtual geschehen
- Prozessor-Emulierung auf abweichender Architektur
- Hypervisor übernimmt Virtualisierung
- ermöglicht Softwarekompatibilität
- Beispiele:
  - Konsolenemulation (PS1, GB, GBA)
  - Linuxkernel für Intel Itanium-CPU

Computer-architektur nach Tanenbaum

- Anwendungsebene
- Assemblerebene
- **Betriebssystem**
- Instruction Set Architecture
- Microarchitektur
- Logische Ebene
- Transistorebene

Definitionen Betriebssystem

Nach DIN 44300:

„... die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften der Rechenanlage die Basis der möglichen Betriebsarten des digitalen Rechensystems bilden und die insbesondere die Abwicklung von Programmen steuern und überwachen.“

Nach Tanenbaum:

„... eine Software-Schicht ..., die alle Teile des Systems verwaltet und dem Benutzer eine Schnittstelle ... anbietet, die einfacher zu verstehen und zu programmieren ist [als die direkte Programmierung der Hardware].“

Definitionen Betriebssystem

Silberschatz/Galvin

„… ein Programm, das als Vermittler zwischen Rechnernutzer und Rechner-Hardware fungiert. Der Sinn des Betriebssystems ist eine Umgebung bereitzustellen, in der Benutzer bequem und effizient Programme ausführen können.“

Brinch Hansen

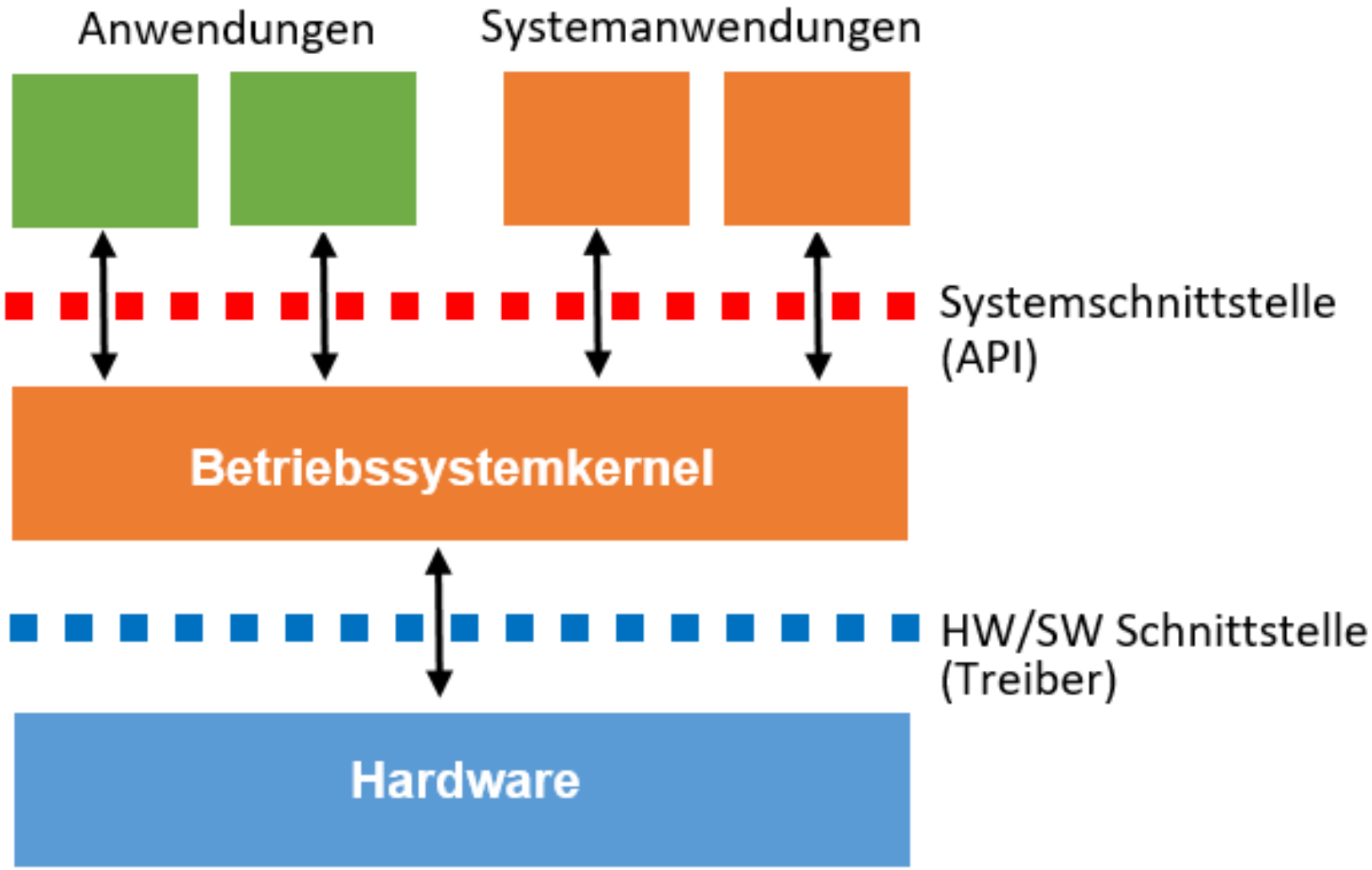
„… der Zweck eines Betriebssystems [liegt] in der Verteilung von Betriebsmitteln auf sich bewerbende Benutzer.“

Zusammenfassung der Definitionen

- Software zur Steuerung und Überwachung der Betriebsarten
- Software zur Betriebsmittelverwaltung
- Vermittler zwischen Hardwareebene und Benutzerebene
- Abstraktionsebene für bequeme und effiziente Programmnutzung
- Benutzerschnittstelle für einfache Programmierung

- kurz: Fundament für Anwenderprogramme

Schematischer Aufbau Betriebssystem



Zentrale Aufgaben eines Betriebssystems

- **Betriebsmittelverwaltung**
  (Betriebsmittel = **Hardware-** oder **Software-Ressource)**
- Verwaltung mehrerer Prozesse
- Aus- und Einlagern von Prozessen
- Verwaltung des Hauptspeichers
- Versorgung aller Prozesse mit benötigten Teilen des Hauptspeichers
- CPU-Scheduling (Auswahl des jeweils nächsten Prozesses für die CPU)
- Datei- und Verzeichnisverwaltung auf Datenträgern
- Verwaltung der Ein- und Ausgabegeräte

Zusammenfassung

Zusammenfassung

Ziel dieser Veranstaltung war es sich neben dem klassischen Aufbau eines Rechners nach der von Neumann Architektur mit den wichtigsten Ebenen zu beschäftigen, die für die Funktion eines Betriebssystems notwendig sind.

Neben der Architektur haben Sie die Mikroarchitektur sowie die Grundlagen der Arithmetik und der Arbeitsweise der logischen Ebene bis hin zur Transistorebene kennengelernt.

Für die Forensik bedeutsame Rechenarithmetiken und Zeichen Codierungen waren ebenso Bestandteil, wie der grundlegende Hintergrund für was ein Betriebssystem zuständig ist.

Vielen Dank

Prof. Ronny Bodach

Hochschule Mittweida | University of Applied Sciences
Technikumplatz 17 | 09648 Mittweida
Fakultät Angewandte Computer- und Biowissenschaften

T +49 (0) 3727 58-1011
F +49 (0) 3727 58-21011
bodach@hs-mittweida.de
www.cb.hs-mittweida.de

Haus 8 | Richard-Stücklen Bau| Raum 8-205
Am Schwanenteich 6b| 09648 Mittweida

Felix Fischer

Hochschule Mittweida | University of Applied Sciences
Technikumplatz 17 | 09648 Mittweida
Fakultät Angewandte Computer- und Biowissenschaften

fische11@hs-mittweida.de
www.cb.hs-mittweida.de

HOCHSCHULE MITTWEIDA
University of Applied Sciences

hs-mittweida.de