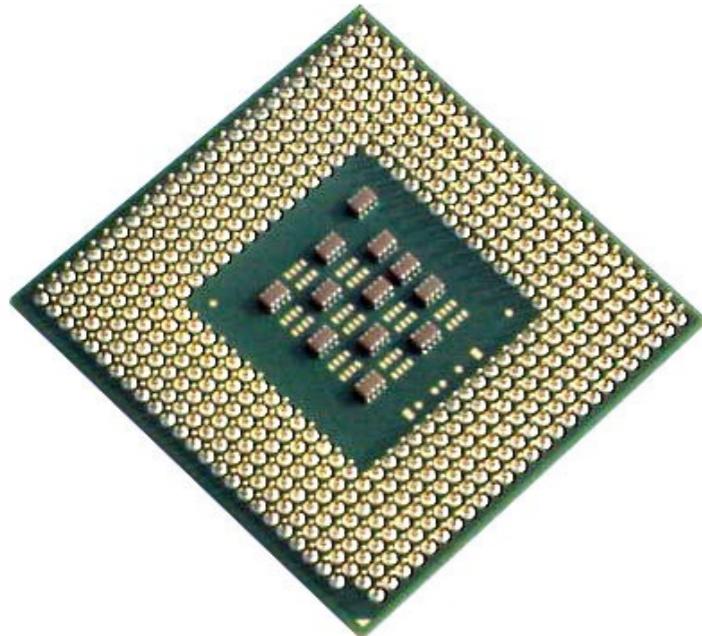
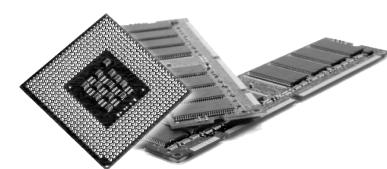


# Datenverarbeitungs- Systeme

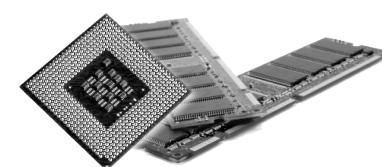
SS 2018/19





## Vorlesung

- Mittwoch: 10:00 - 13:15 Uhr (D003)
- Sprechstunde: Donnerstag 10.30 - 11.30 oder nach Vereinbarung
- Kontakt: [rudolf.hackenberg@oth-regensburg.de](mailto:rudolf.hackenberg@oth-regensburg.de)



# Übung



## Übungen

- Gruppe 1
- Gruppe 2
- Gruppe 3
- Gruppe 4
- Abgabe der Übungen ist Pflicht
- Abgabeliste unterschreiben lassen!

Fakultät Informatik und Mathematik  
Vorlesung Hardwaregrundlagen  
Prof. Dr. Rudolf Hackenberg  
Wintersemester 2013/2014

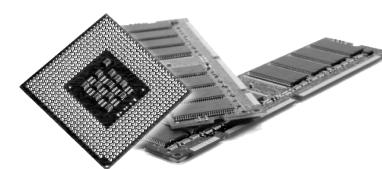
### Abgabeliste

Name:  
Matrikelnummer:

Diese Liste ist zur Prüfung mitzubringen. Die Erfüllung des Übungspensums ist Voraussetzung für die Prüfungszulassung.

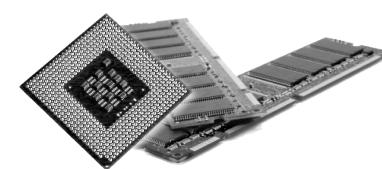
	Unterschrift
Übungsblatt 1	
Übungsblatt 2	
Übungsblatt 3	
Übungsblatt 4	
Übungsblatt 5	
Übungsblatt 6	
Übungsblatt 7	
Übungsblatt 8	

Abgabeliste für Hardwaregrundlagen



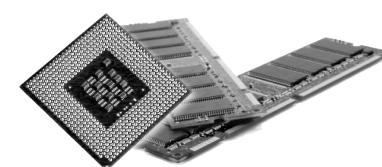
## E-Learning

- <https://elearning.uni-regensburg.de/course/view.php?id=19740>
  - Skript
  - Übungsblätter
  - Forum / Nachrichten
  - Ankündigungen



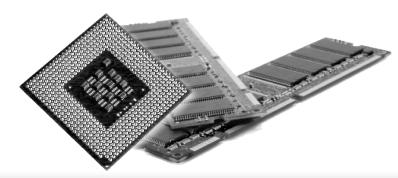
## Prüfung

- Art: schriftlich
- Dauer: 90 Minuten
- keine Unterlagen zugelassen
- Inhalte:
  - Vorlesung
  - Übungen

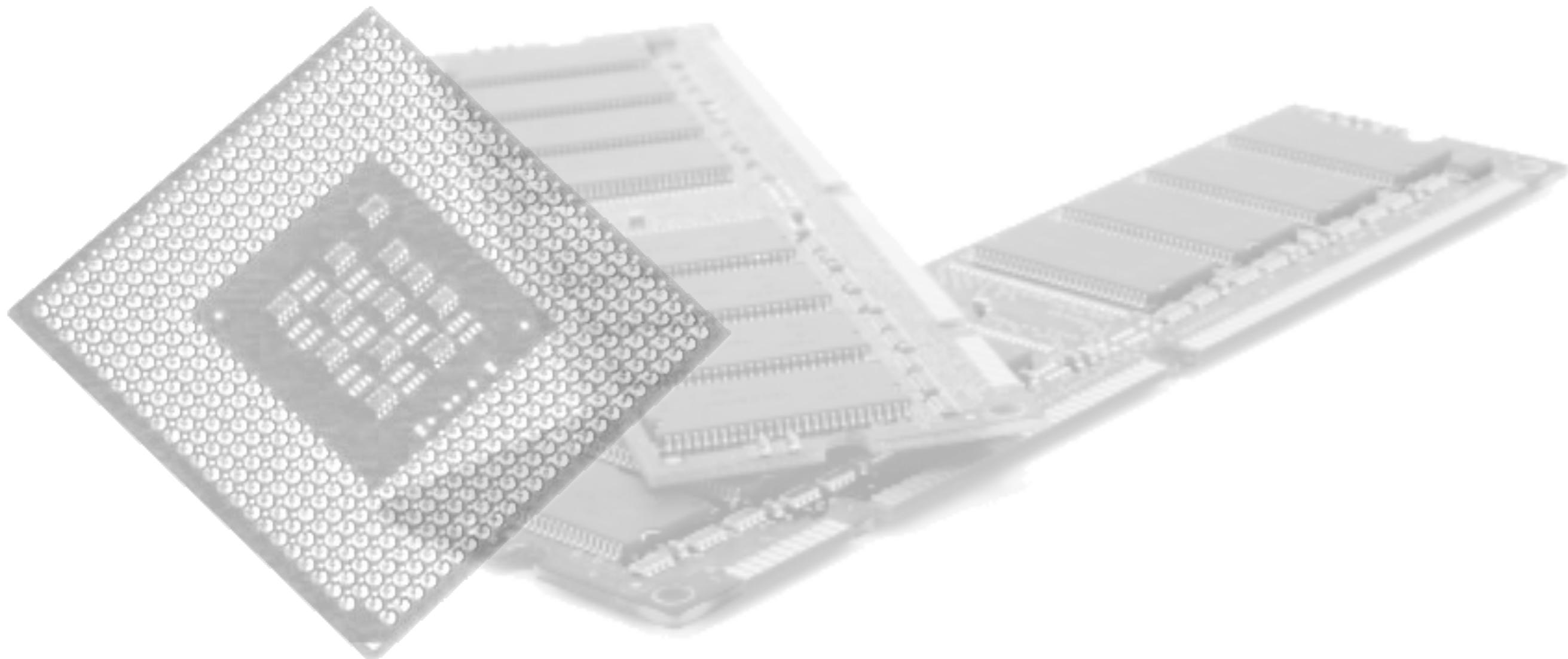


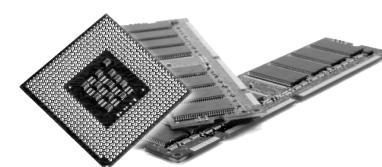
## Inhalte der Vorlesung

- Grundlagen
- Zahlendarstellung / Arithmetik / Konvertierung
- Datenverarbeitungssysteme
- Hard- und Software / von-Neumann-Architektur
- Mikroprozessoren
- Befehlszyklus / interne Organisation 8086
- OPCODE / Interrupts
- Assembler
  - Grundlagen
  - Adressierung / arithmetische Befehle / Schleifen
  - Ein- und Ausgabe
  - ...



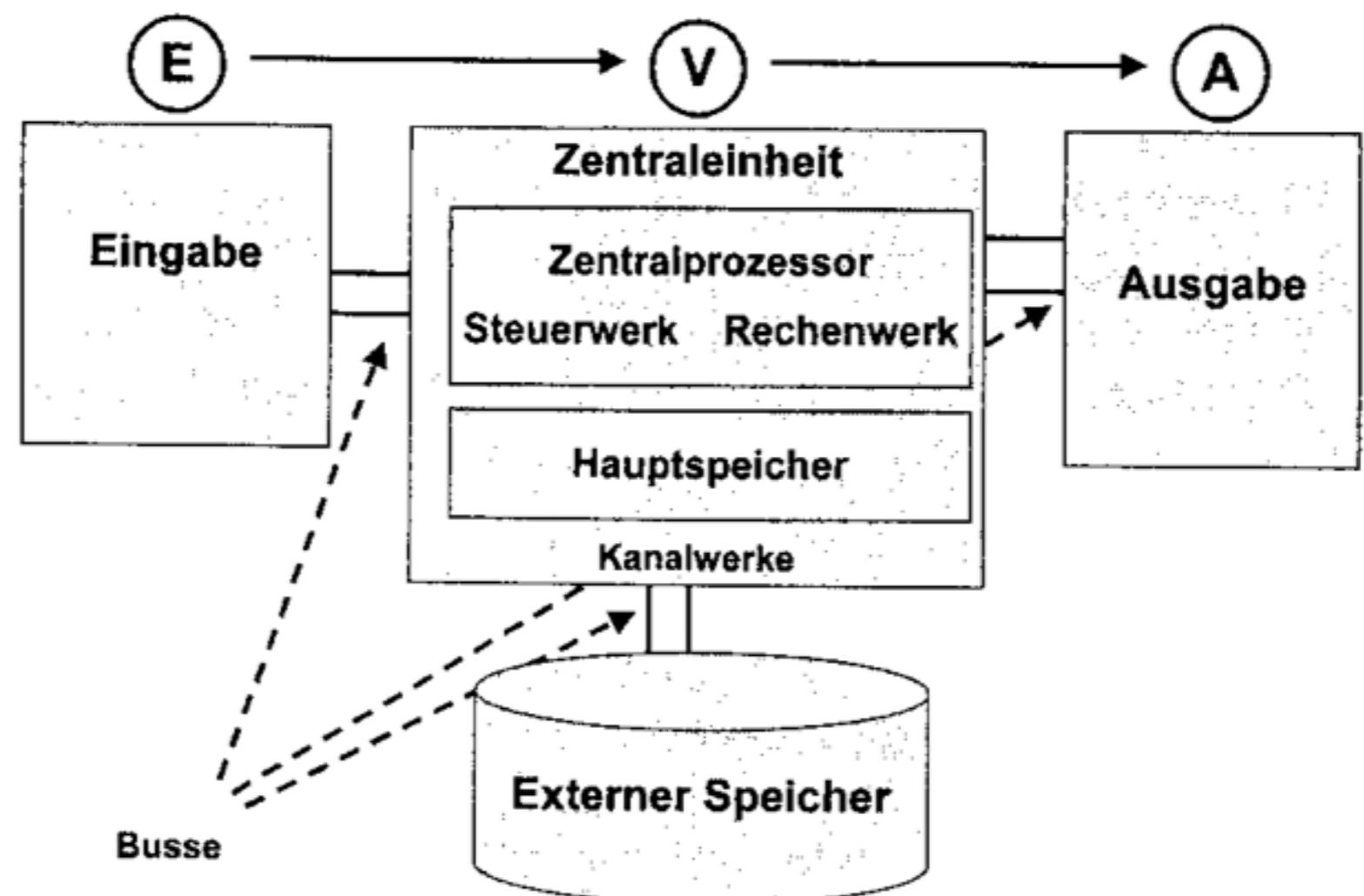
## Grundlagen

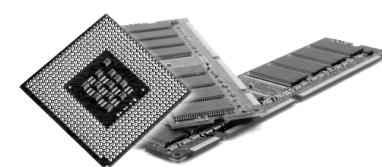




## EVA - Prinzip

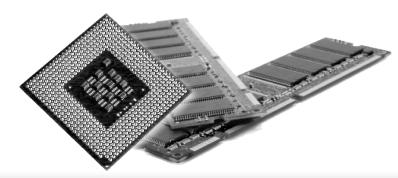
- Eingabe
- Verarbeitung
- Ausgabe



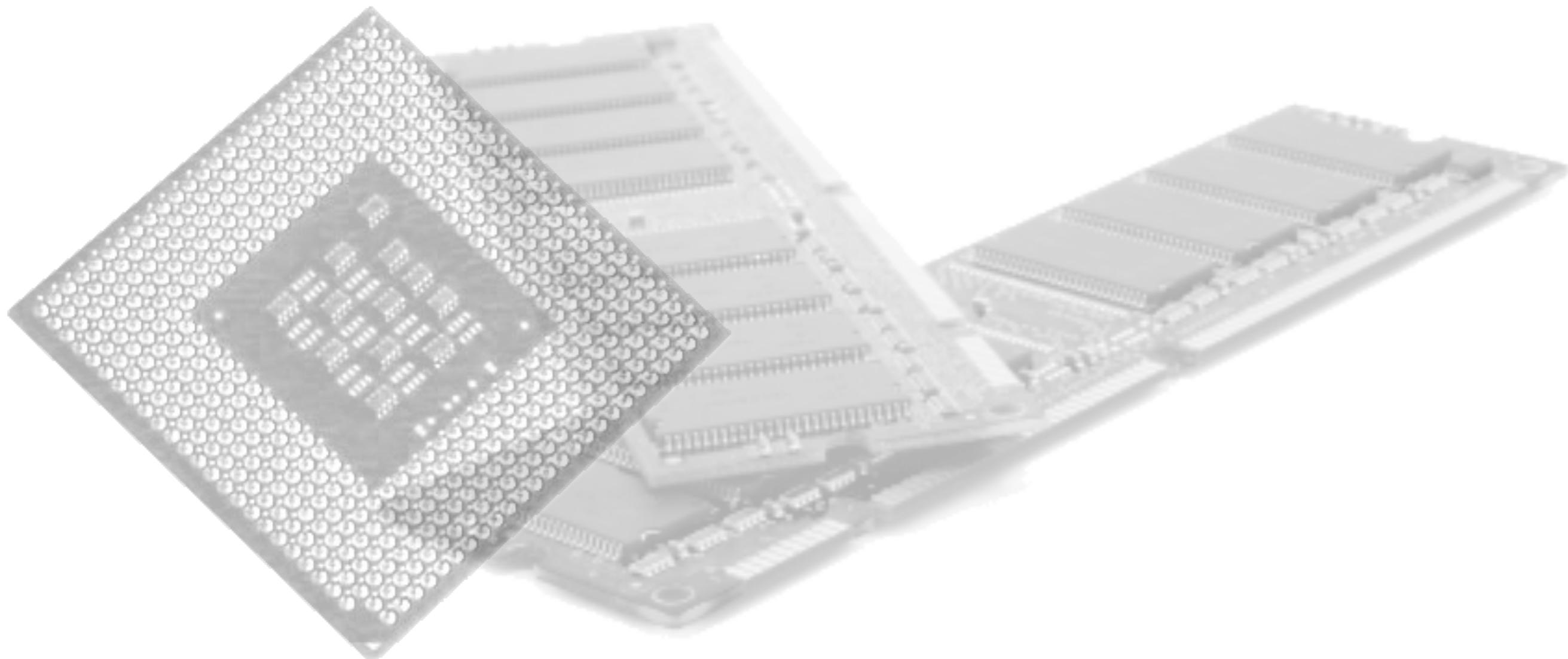


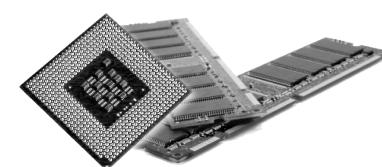
## Begriffe (Definition)

- Ein **Datum** (Plural: **Daten**) ist eine Folge von digitalen Zeichen
- Eine **Information** ist ein Datum mit inhaltlicher Bedeutung
- Eine **Nachricht** ist eine Zeichenfolge, die eine Information vermittelt
- Ein **Signal** ist die physikalische Darstellung von Nachrichten oder Daten.



## Zahlendarstellung

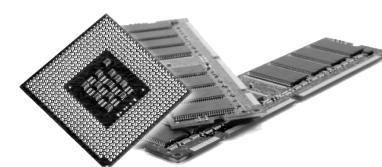




# Stellenwertsystem

- Dezimal: 0,1,2,3,4,5,6,7,8,9

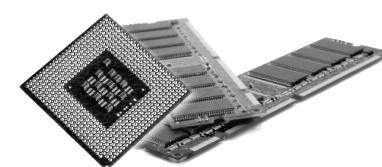
Dezimal		Dezimal
0		
1		
...		
9		
<b>1 0</b>	$= 1 \times 10^1 + 0 \times 10^0$	$= 10 + 0 = 10$
<b>1 1</b>	$= 1 \times 10^1 + 1 \times 10^0$	$= 10 + 1 = 11$
<b>1 2</b>	$= 1 \times 10^1 + 2 \times 10^0$	$= 10 + 2 = 12$
...		
<b>2 8</b>	$= 2 \times 10^1 + 8 \times 10^0$	$= 20 + 8 = 28$
...		
<b>1 3 4</b>	$= 1 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$	$= 100 + 30 + 4 = 134$
...		



# Stellenwertsystem

- Oktal: 0,1,2,3,4,5,6,7

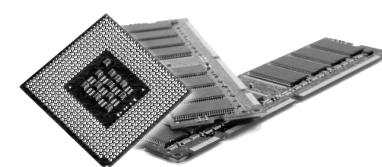
Oktal		Dezimal
0		
1		
...		
7		
<b>1 0</b>	$= 1 \times 8^1 + 0 \times 8^0$	$= 8 + 0 = 8$
<b>1 1</b>	$= 1 \times 8^1 + 1 \times 8^0$	$= 8 + 1 = 9$
<b>1 2</b>	$= 1 \times 8^1 + 2 \times 8^0$	$= 8 + 2 = 10$
...		
<b>2 0</b>	$= 2 \times 8^1 + 0 \times 8^0$	$= 16 + 0 = 16$
...		
<b>1 3 4</b>	$= 1 \times 8^2 + 3 \times 8^1 + 4 \times 8^0$	$= 64 + 24 + 4 = 92$
...		



# Stellenwertsystem

- Hexadezimal: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

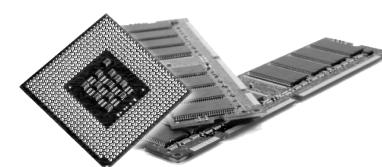
Hexadezimal	Dezimal
0	
...	
9	
<b>A</b>	= 10
<b>B</b>	= 11
...	
<b>F</b>	= 15
<b>1 0</b>	$= 1 \times 16^1 + 0 \times 16^0 = 16 + 0 = 16$
<b>1 1</b>	$= 1 \times 16^1 + 1 \times 16^0 = 16 + 1 = 17$
...	
<b>1 C 4</b>	$= 1 \times 16^2 + 12 \times 16^1 + 4 \times 16^0 = 256 + 192 + 4 = 452$
...	



# Stellenwertsystem

- Binär: 0,1

Binär		Dezimal
0		
1		
1 0	$= 1 \times 2^1 + 0 \times 2^0$	$= 2 + 0 = 2$
1 1	$= 1 \times 2^1 + 1 \times 2^0$	$= 2 + 1 = 3$
1 0 0	$= 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	$= 4 + 0 + 0 = 4$
1 0 1	$= 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	$= 4 + 0 + 1 = 5$
1 1 0	$= 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	$= 4 + 2 + 0 = 6$
1 1 1	$= 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	$= 4 + 2 + 1 = 7$
1 0 0 0	$= 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	$= 8 + 0 + 0 + 0 = 8$
1 0 0 1	$= 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$	$= 8 + 0 + 0 + 1 = 9$
...		
1 1 0 1 1 0	$= 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$	$= 32 + 16 + 0 + 4 + 2 + 0 = 54$
...		



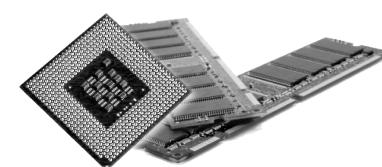
# Stellenwertsystem

## Definition

Ein Stellenwertsystem zur Basis  $b$  ist wie folgt definiert:

$$(a_n \dots a_3 a_2 a_1 a_0, a_{-1} a_{-2} a_{-3}) = \sum_{i=-\infty}^n a_i b^i$$

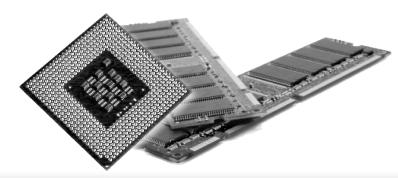
$$b \in N, b > 1, a_i \in Z, a_i \in [0, b-1]$$



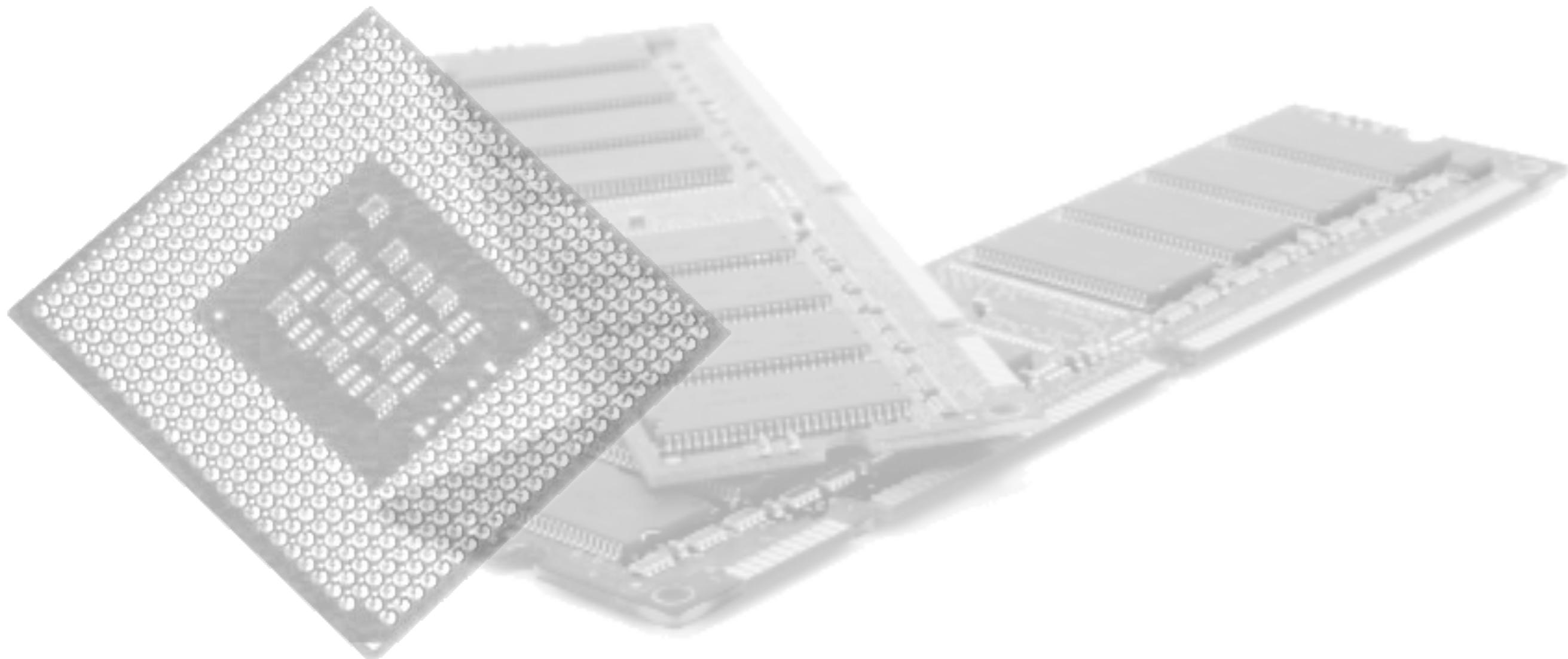
# Stellenwertsysteme

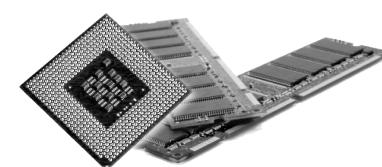
Basis b	Bezeichnung	Ziffernbereich
2	Binär, Dual	0,1
8	Oktal	0,1,...,7
10	Dezimal	0,1,...,9
16	Hexadezimal	0,1,...,9,A,B,C,D,E,F

- Der Wert einer Zahl (=Ziffernfolge) ist abhängig von der Basis. Ohne Angabe der Basis ist der Wert nicht eindeutig definiert.



## Arithmetik





# Addition und Subtraktion

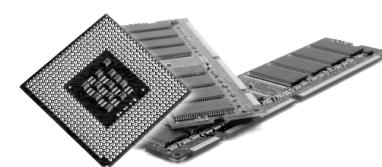
Für die beiden  $b$ -adischen Zahlen  $x$  und  $y$  gemäß

$$x = (x_N \dots x_2 x_1 x_0, x_{-1} x_{-2} \dots x_{-M}) = \sum_{i=-M}^N x_i b^i$$

$$y = (y_N \dots y_2 y_1 y_0, y_{-1} y_{-2} \dots y_{-M}) = \sum_{i=-M}^N y_i b^i$$

wird die Summe  $x + y$  bzw. die Differenz  $x - y$  nach folgender Regel gebildet:

$$x \pm y = \sum_{i=-M}^N x_i b^i \pm \sum_{i=-M}^N y_i b^i = \sum_{i=-M}^N (x_i \pm y_i) b^i$$



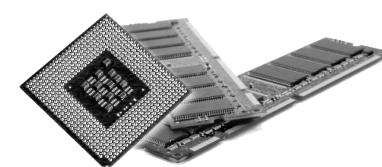
# Addition und Subtraktion

## Hinweis:

Für den eventuell eintretenden Ziffern- Überlauf ( $x_i + y_i > b - 1$ ) wird ein **Übertrag** eingeführt, der auf die nächst links stehende Ziffer addiert wird.

$$\begin{array}{rcl} x & = & 1260315,2 \\ y & = & 1271423,3 \\ \hline x + y & = & 2531738,5 \end{array}$$

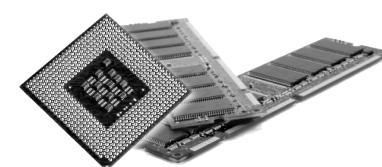
The diagram shows a vertical addition of two floating-point numbers. The numbers are aligned by their decimal points. The digit '6' in the first number and the digit '7' in the second number are highlighted with a yellow box, indicating they are being added together. A small orange '1' is positioned above the yellow box, indicating a carry-over (Übertrag) from this column to the next.



# Addition und Subtraktion

- Binär:
  - $11,625_{10} + 13,25_{10}$

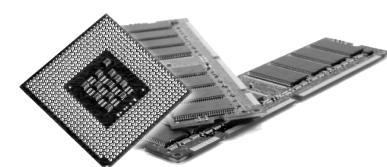
$$\begin{array}{rcl} x & = & 1011,101 \\ y & = & 1101,010 \\ \hline x+y & = & 11000,111 \end{array}$$



# Multiplikation

Das Produkt der beiden  $b$ -adischen Zahlen  $x$  und  $y$  berechnet sich wie folgt:

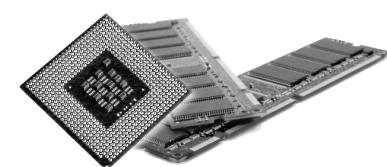
$$\begin{aligned}x * y &= \left( \sum_{i=-M}^N x_i b^i \right) * \left( \sum_{j=-M}^N y_j b^j \right) \\&= \sum_{i,j=-M}^N (x_i b^i * y_j b^j) = \sum_{i,j=-M}^N (x_i * y_j b^{i+j})\end{aligned}$$



# Multiplication

$$\begin{array}{r} 1 \ 4 \ , \ 2 \ * \ 3 \ 7 \ , \ 4 \\ \hline 4 \ 2 \ 6 \ 0 \ 0 \\ 9 \ 9 \ 4 \ 0 \\ 5 \ 6 \ 8 \\ \hline 1 \ 2 \ 1 \\ \hline 5 \ 3 \ 1 \ 0 \ 8 \end{array}$$

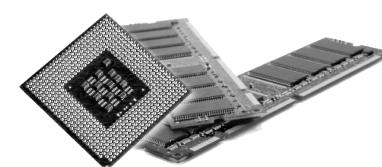
$$14,2 * 37,4 = 531,08$$



# Multiplikation

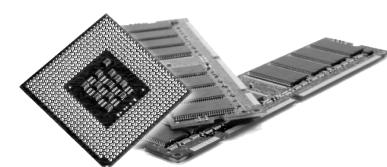
- Gleiches Prinzip für Binärzahlen
- Beispiel:

$$\begin{array}{r} 1 \quad 0 \quad 1 \quad 1 \quad * \quad 1 \quad 0 \quad 1 \\ \times \quad 1 \quad 0 \quad 1 \quad 1 \\ \hline & & & & 0 \\ & & & 1 \quad 0 \quad 1 \quad 1 \\ & & & \hline & 1 & \\ \hline 1 \quad 1 \quad 0 \quad 1 \quad 1 \end{array}$$



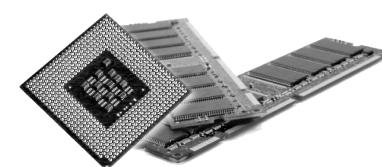
## Division

- Dezimalsystem
  - Betrachtung erste Ziffer des Dividenden
    - wenn Divisor n-mal Bestandteil ( $n = \{0, \dots, 9\}$ )
      - n als nächste Ziffer anschreiben
      - n-faches des Divisors von Dividenden subtrahieren
      - für weitere Dividenden-Stelle bis 0 wiederholen



## Division

$$\begin{array}{r} 9 & 2 & 8 & 8 & : & 3 & 6 & = & 2 & 5 & 8 \\ \hline - & 7 & 2 & & & & & & & & \\ \hline & 2 & 0 & 8 & & & & & & & \\ - & 1 & 8 & 0 & & & & & & & \\ \hline & 2 & 8 & 8 & & & & & & & \\ - & 2 & 8 & 8 & & & & & & & \\ \hline & & & & 0 & & & & & & \end{array}$$

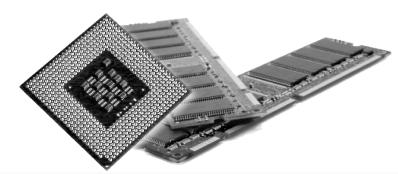


# Division

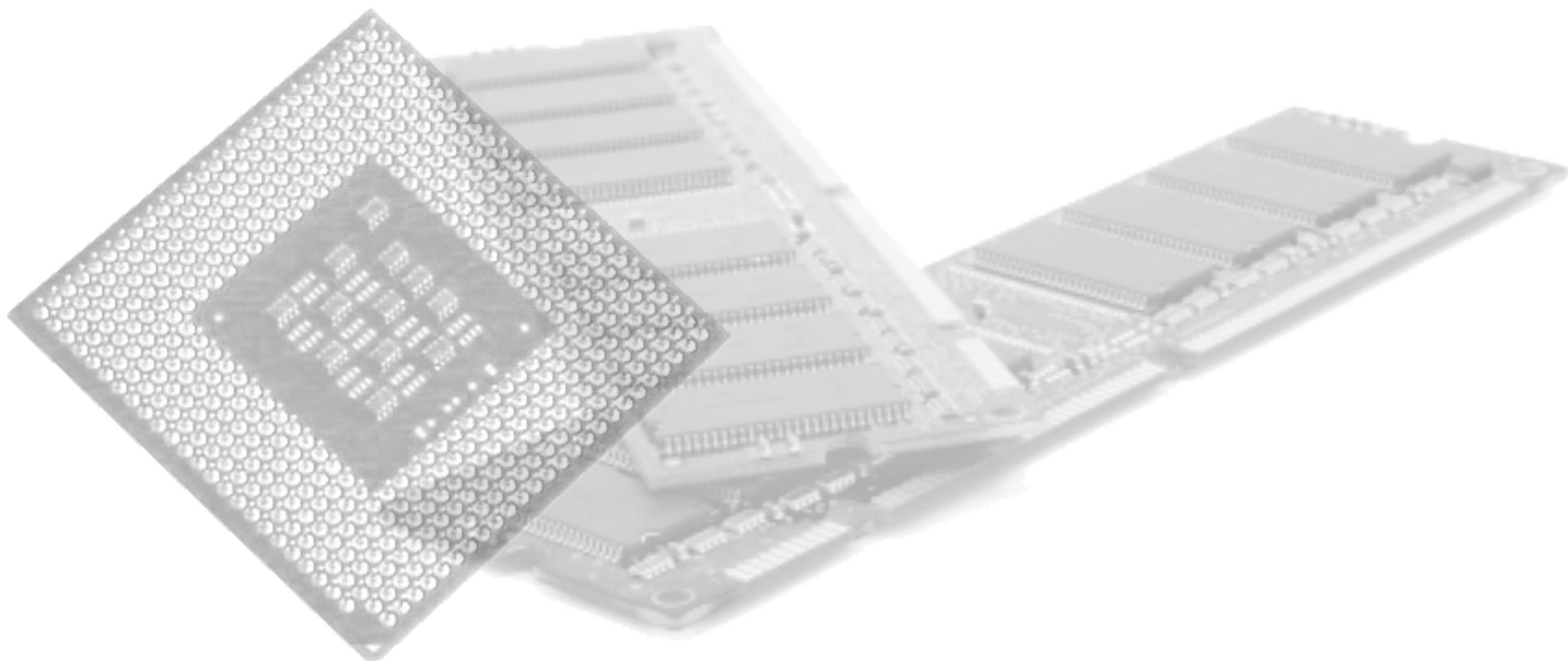
- Gleiches Prinzip für Binärzahlen
- Divisor immer 0 oder 1 mal Bestandteil des Dividenden

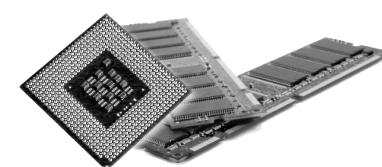
- Beispiel:

$$\begin{array}{r} 1 \quad 0 \quad 1 \quad 0 \quad 1 \\ \hline - & 1 \quad 0 \quad 1 \\ & \hline 1 \quad 0 \quad 0 \\ - & 1 \quad 1 \\ & \hline 1 \quad 1 \\ - & 1 \quad 1 \\ & \hline 0 \end{array} = \boxed{1 \quad 1 \quad 1}$$



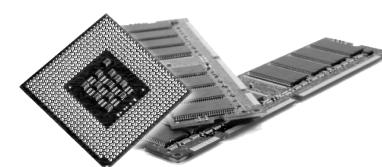
## Konvertierung





## Konvertierung

- Umformung von einem Stellenwertsystem in ein anderes
- z.B.  $1011_{(2)} = 11_{(10)}$  (binär nach dezimal)
- Horner-Schema
- Verfahren der fortgesetzten Division mit Rest
- Spezialfälle bei der Konvertierung
- Festkommazahlen
- Gleitkommazahlen

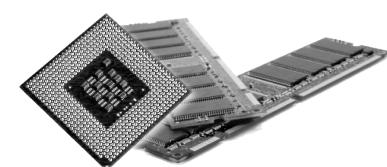


# Horner-Schema

## Algorithmus

Eine  $c$ -adische Zahl  $u$  wird in eine  $b$ -adische Zahl  $v$  konvertiert, indem die folgende Formel angewandt wird:

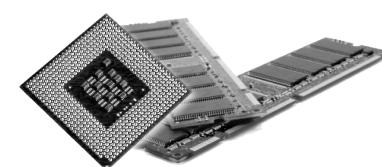
$$(u)_c = \sum_{i=0}^n u_i b^i = ((\dots(u_n b + u_{n-1})b + u_1)b + u_0) = (v)_b$$



# Horner-Schema

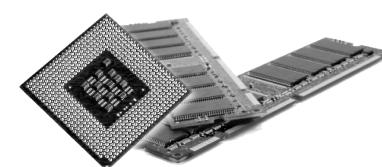
- $2173_{(8)}$  in Dezimal:

$$\begin{array}{ccccccc} 2 & 1 & 7 & 3 & (8) & = \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\ (( ( 2 * 8 + 1 ) * 8 ) + 7 ) * 8 + 3 & = \\ 1 & 1 & 4 & 7 & (10) \end{array}$$



## Horner-Schema

- Ausnutzung des (dualen) Stellenwertsystems wiederholte Verschiebung von Ziffern (div/mod)
- Umwandlung dual → dezimal:
  - Dualziffern von links nach rechts durchlaufen
  - Ziffer (1 oder 0) addieren, Resultat mit 2 multiplizieren (außer nach letzter Stelle)
- Umwandlung dezimal → dual:
  - Dezimalzahl wiederholt durch 2 teilen (ganzzahlig)
  - jeweils Rest (0 oder 1) notieren
  - notierte Ziffern ergeben Dualzahl (umgekehrt)



# Horner-Schema

Kovertierung von binär nach dezimal:

$$(11001)_2 =$$

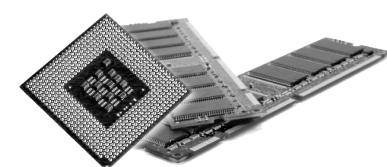
$$1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0 =$$

$$\left( \left( (1 * 2 + 1) * 2 + 0 \right) * 2 + 0 \right) * 2 + 1 =$$

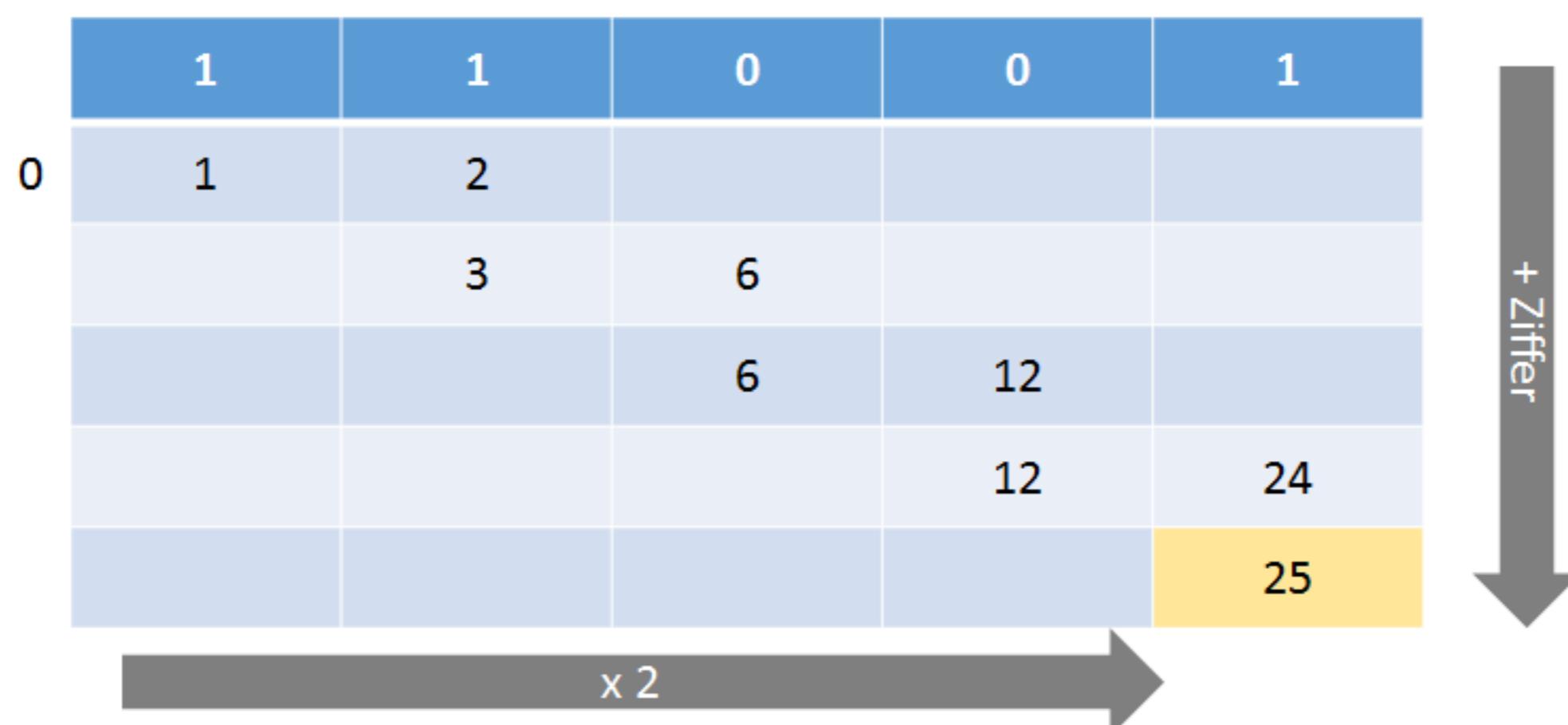
$$\left( \left( (2 + 1) * 2 + 0 \right) * 2 + 0 \right) * 2 + 1 =$$

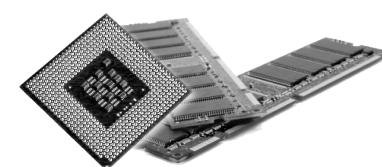
$$(3 * 2 * 1 + 0) * 2 + 1 =$$

$$12 * 2 + 1 = 25 = (25)_{10}$$



## Horner-Schema





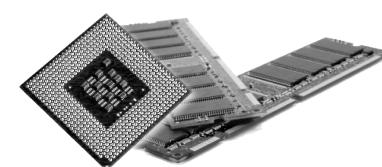
# Verfahren der fortgesetzten Division mit Rest

- Das Verfahren der fortgesetzten Division mit Rest stellt eine Umkehrung des Horner-Schemas dar.

$$z = q * d + r$$

$$z = (z \text{ div } d) * d + (z \text{ mod } d)$$

$$\begin{aligned} z &= (a_n a_{n-1} \dots a_1 a_0)_2 = a_n * 2^n + a_{n-1} * 2^{n-1} + \dots + a_1 * 2^1 + a_0 * 2^0 \\ &= (a_n * 2^{n-1} + a_{n-1} * 2^{n-2} + \dots + a_1) * 2^1 + a_0 * 2^0 \\ &= (a_n a_{n-1} \dots a_1)_2 * 2^1 + a_0 \end{aligned}$$



# Beispiel

**$z = 29$**

**$z \text{ div } 2$**

**$z \text{ mod } 2$**

29

14

1

14

7

0

7

3

1

3

1

1

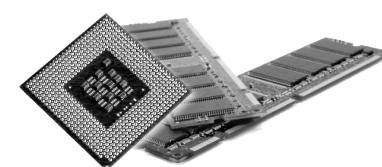
1

0

1

Rest von unten nach oben gelesen  
ergibt das Ergebnis

$\Rightarrow (1\ 1\ 1\ 0\ 1)_2$



## Beispiel 0,...

 $z = 29$  $z \text{ div } 2$  $z \text{ mod } 2$ 

29

14

7

3

1

14

0

to do  
Umrechnen von 0,25 (z.B.)

1

0

1

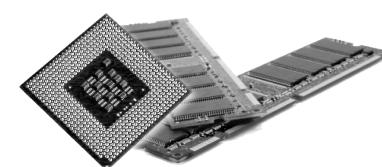
1

1



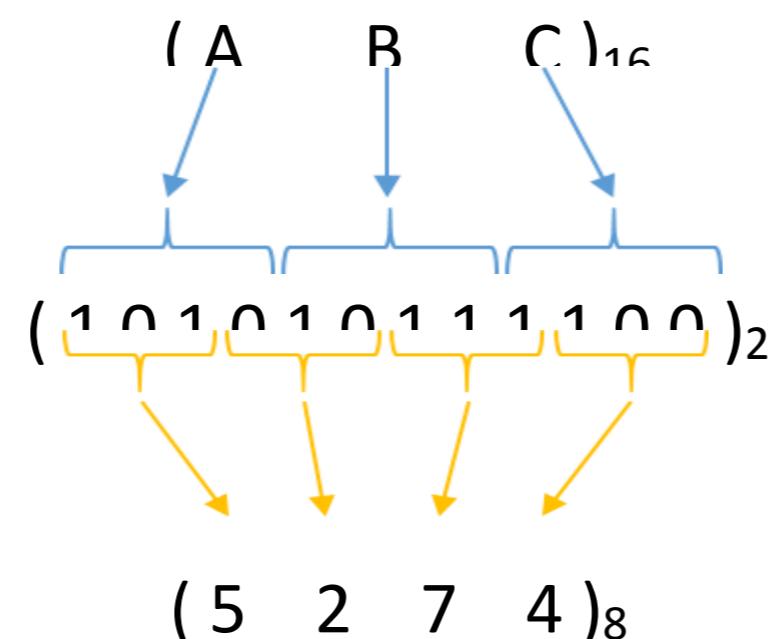
Rest von unten nach oben gelesen  
ergibt das Ergebnis

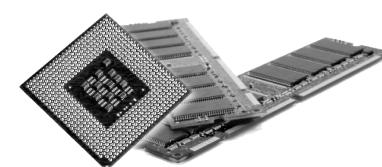
 $\Rightarrow (1\ 1\ 1\ 0\ 1)_2$



# Spezialfälle

- Konvertierungen zwischen **hexadezimal – binär** und **oktal** sind sehr einfach durchzuführen:

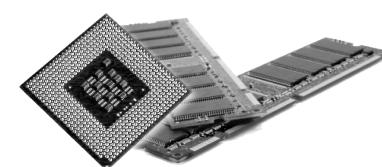




# Ganze Zahlen

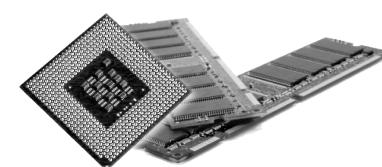
- Die Nutzung des 2er-Komplement ermöglicht die Darstellung von negativen und positiven ganzen Zahlen
- Vorzeichenbit repräsentiert Zahlenbereich
- Subtraktion durch Umwandlung in Addition mit negativen Zahlen

$$z = -a_n * 2^n + a_{n-1} * 2^{n-1} + \dots + a_1 * 2^1 + a_0 * 2^0$$



## 2er Komplement

- Von rechts bis zur ersten 1 abschreiben (incl. erste 1) und dann die restlichen Stellen umdrehen (aus 1 wird 0 und aus 0 wird 1):
- **100000011 10** <= von rechts beginnen
- Umdrehen, abschreiben
- **011111100 10**

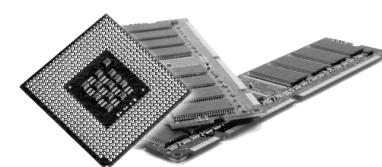


# 2er Komplement

- Interpretation:
  - das höchstwertige Bit hat eine negative Wertigkeit

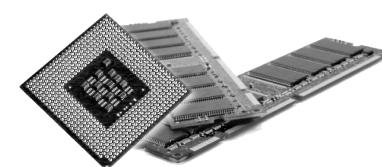
Wertigkeit	-128	64	32	16	8	4	2	1	Dezimal
Bitfolge	0	0	0	1	1	0	1	0	= 26
Bitfolge	1	1	1	0	0	1	1	0	= -26

- $00011010_{(2)} = 16 + 8 + 2 = 26$
- $11100110_{(2)} = -128 + 64 + 32 + 4 + 2 = -26$



# 2er Komplement - Subtraktion

- $7_{10} - 9_{10}$ :  $0111_2 - 1001_2$ 
  - 2er Komplement von  $1001$ : **0111**
  - Addition:  $0111 + 0111 = 1110$
  - negatives Ergebnis: 2er Komplement von  $1110$ : **0010** ( $-2_{10}$ )
- $8_{10} - 5_{10}$ :  $1000_2 - 0101_2$ 
  - 2er Komplement von  $0101$ : **1011**
  - Addition:  $1000 + 1011 = 10011$
  - positives Ergebnis: 1. Stelle weglassen: **0011** ( $3_{10}$ )



# Festkommazahlen

- Darstellung durch Kommmazahlen mit fester Anzahl n Stellen vor dem Komma und m Stellen nach dem Komma (Festkommadarstellung, fixed point)

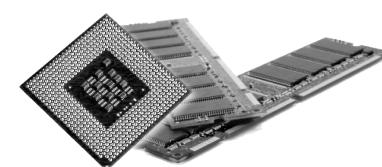
$$z = (a_n a_{n-1} \dots a_1 a_0 , b_1 b_2 \dots b_{m-1} b_m)_2$$

$$= (a_n * 2^n + a_{n-1} * 2^{n-1} + \dots + a_0 , b_1 * 2^{-1} + \dots + b_m * 2^{-m})_2$$

- Behandelte Rechenverfahren können direkt übernommen werden, evtl. Anpassungen
- Stellenkorrektur

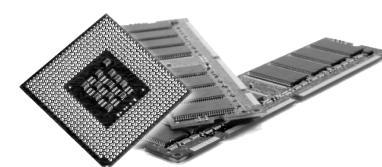
$$0000\ 1010 * 0000\ 1100 = 0111\ 1000$$

$$0000\ 1,010 * 0000\ 1,100 = 0000\ 1,111$$



# Festkommazahlen - Hinweise

- Genauigkeitsverlust bei kleinen Beträgen
  - $00123,456 : 100 = 00001,234$ 
    - d.h. zwei signifikante Ziffern gehen verloren
- Überlauf bei hohen Beträgen
  - $00123,456 \cdot 1000 = (1)23456,000$ 
    - zu viele Stellen für vorgesehene Darstellung
- Daher **Gleitkommadarstellung (floating point)**
  - Idee: signifikanten Ziffern und ihrer Position getrennt dargestellt (Exponentialschreibweise)

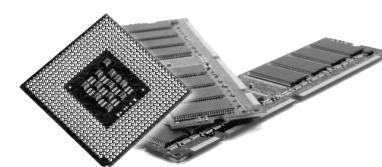


# Gleitkommazahlen

- Nutzung von Exponent und Mantisse
  - Exponent zeigt Nachkommastelle bzw. Position der Mantisse (in Bezug zu einer Basis)
  - Mantisse zeigt darstellbare Stellen

$$123,456 = \underset{\text{Mantisse (m)}}{1,23456} \times \underset{\text{Basis (b)}}{10^2}^{\underset{\text{Exponent (e)}}{2}}$$

- Darstellung nicht eindeutig
- Deshalb „Normalisierung“ erforderlich

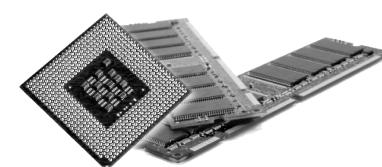


# Gleitkommazahlen

- Hauptstandard: IEEE-Format als Normalisierung
- Single/Double Precision

Größe	32 Bit	64 Bit
Vorzeichen	1	1
Exponent	8	11
Mantisse	23	52

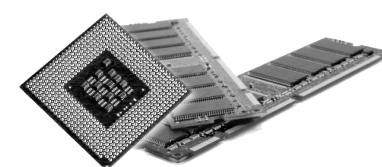
- Vorzeichen: 0 positiv, 1 negativ
- Exponent: Wertebereich -126 bis +127
- Mantisse: normalisiert auf 1,...
  - 0 nicht normalisierbar -> minimale Mantisse/Exponent
  - Sonderfälle: NaN und „Unendlich“



# Gleitkommazahlen - Arithmetik

- Addition/Subtraktion
  - Normalisierung beider Zahlen: gleicher Exponent (höhere)
  - Mantissenwerte addieren / subtrahieren
  - Ggf. Normalisierung der Ergebnisse

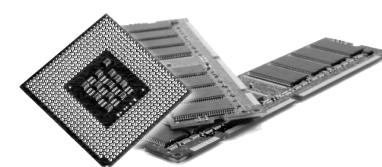
$$\begin{aligned} & 1,234 \quad \times 10^4 \quad - \quad 9,876 \quad \times 10^3 \\ = & 1,234 \quad \times 10^4 \quad - \quad 0,9876 \quad \times 10^4 \\ = & 0,2464 \quad \times 10^4 \\ = & 2,464 \quad \times 10^3 \end{aligned}$$



# Gleitkommazahlen - Arithmetik

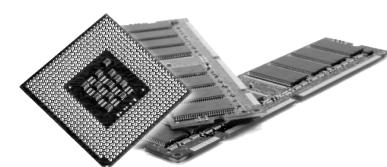
- Multiplikation/Division
  - Mantissenwerte multiplizieren / dividieren
  - Exponentenwerte addieren / subtrahieren
  - Ggf. Normalisierung

$$\begin{aligned} & 1,234 \quad \times 10^4 \quad \times 9,876 \quad \times 10^3 \\ = & 1,234 \quad \times 9,876 \quad \times 10^4 \quad \times 10^3 \\ = & 12,187 \quad \times 10^7 \\ = & 1,2187 \quad \times 10^8 \end{aligned}$$



## Gleitkommazahlen - Konvertierung

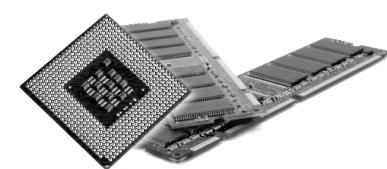
- Umrechnung zwischen dezimal – binär durch Horner-Schema (erweiterte Form)
  - Vorkommazahl umrechnen
  - Nachkommazahl umrechnen
  - Normalisierung
  - Exponent berechnen
  - Vorzeichen bestimmen
  - Ergebnis darstellen



# Konvertierungen

- Hexadezimaltabelle / Binärtabelle

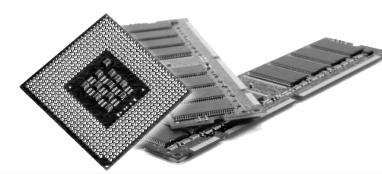
Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bin	00	01	10	11	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111



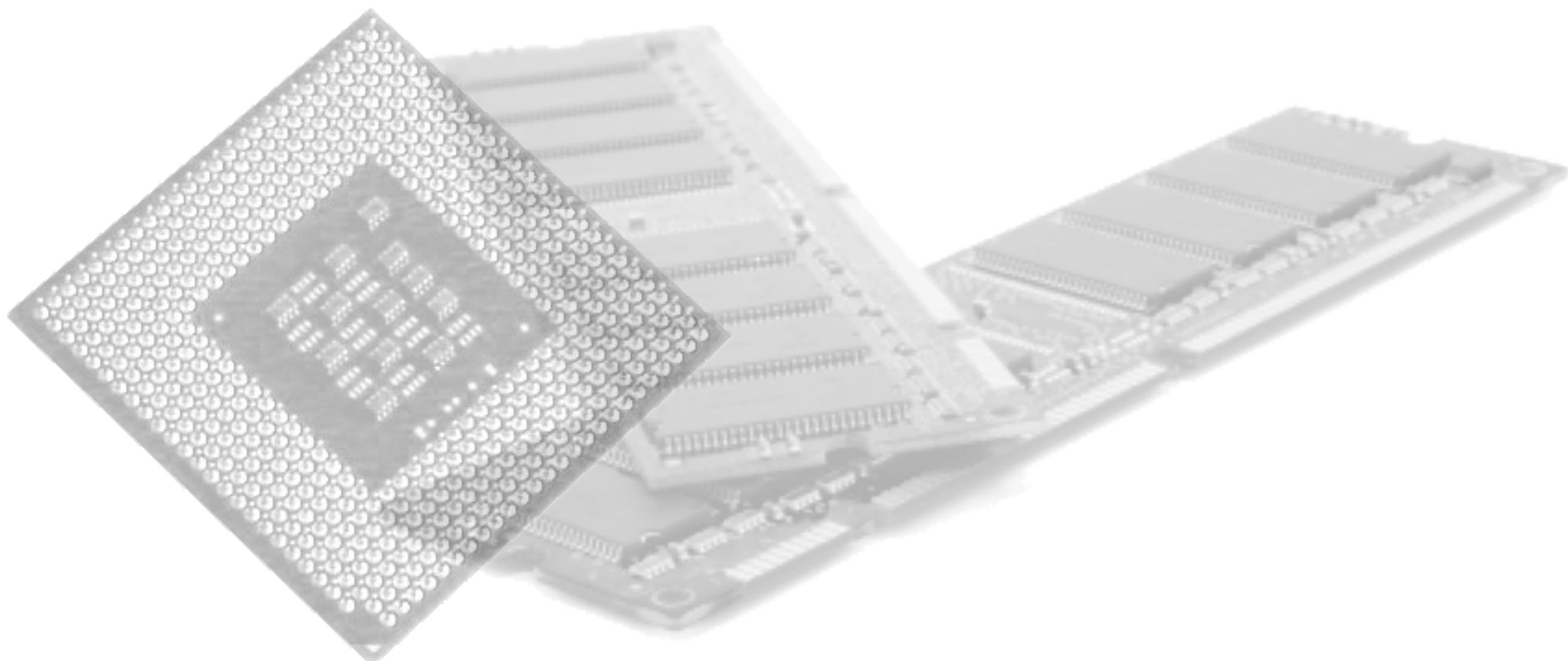
# Zusammenfassung

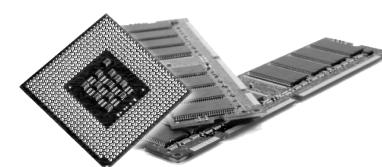
- Horner-Schema
    - $(( ( a_3 * \text{Basis} + a_2 ) * \text{Basis} ) + a_1 ) * \text{Basis} + a_0$
  - Spezialfälle
    - Hex  $\leftrightarrow$  Bin  $\leftrightarrow$  Oct
  - Gleitkommazahlen

$$123,456 = \underset{\text{Mantisse (m)}}{1,23456} \times \underset{\text{Basis (b)}}{10^2} \underset{\text{Exponent (e)}}{2}$$



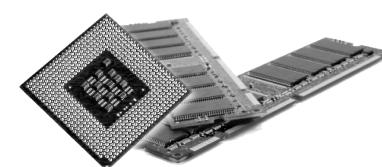
# Datenverarbeitungssysteme





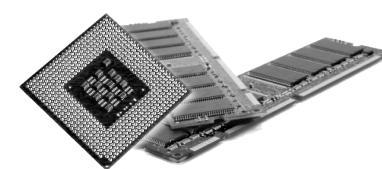
## Datenverarbeitungssystem (DVS)

- Programmgesteuertes digitale Systeme zur numerischen und nichtnumerischen Datenverarbeitung
- System:
  - Hardware
  - Software



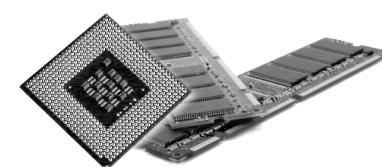
## Datenverarbeitungssystem (DVS)

- Computer (früher)
  - **Groß-/Super-Computer** - DV-Anlagen
  - **Mini-Computer** – Labor- bzw. Prozessrechner für wissenschaftliche und technische Anwender (Mehrbenutzersysteme)
  - **Micro-Computer** - sehr kompakte Systeme auf der Basis integrierter Steuer- und Verarbeitungsbausteine (Homecomputer), u.a. Fertigung
  - **Microcontroller** – spezialisierte Systeme z.B. Peripherie, KFZ, Meß-/ Regel-/ Steuer- Systeme, Entertainment



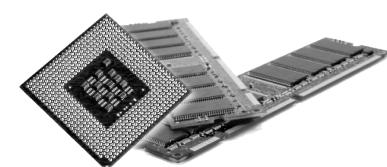
## Grundfunktionen

- **RISC** – Reduced Instruction Set Computer
  - Kleiner Befehlssatz, Fest gedrahtet, Pipelining, großes Registerset
- **CISC** - Complex Instruction Set Computers
  - Mikroprogramme
  - Erleichterung für Programmierer
- hybride **RISC / CISC** - heute



## RISC

- kleiner Befehlssatz
  - fast alle Befehle lassen sich innerhalb eines Taktcs ausführen
- einfacher Chip
- Aufwand bei der Software (längerer Quellcode)
- weniger Stromverbrauch
- ARM-Architektur

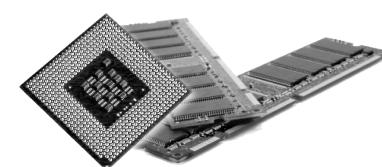


## CISC

- komplizierteres Chipdesign
- Aufwand bei der Hardware
- Weniger Code
- x86-Architektur (Intel)

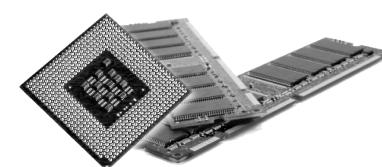


Intel Pentium 4



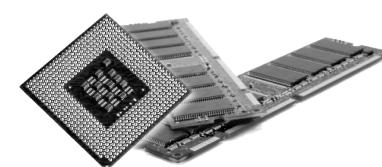
# Vergleich RISC - CISC





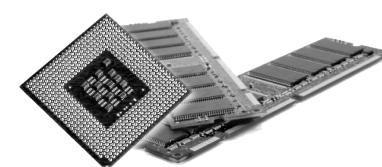
## RISC / CISC

- Übersetzen den x86-Befehlssatz zunächst in RISC-Mikro-Instruktionen konstanter Länge
  - mikro-architektonische Optimierungen
- erste hybride x86-Prozessor: Intel Pentium Pro.



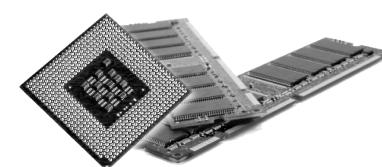
## Informationseinheiten

- **BIT** (binary digit) - Kurzform für Binärziffer
- **BYTE** – 8 zusammen betrachtete Binärziffern
- **WORD** - Eine Folge von Zeichen, die in einem bestimmten Zusammenhang als eine Einheit betrachtet wird.
- Wenn ein WORD einen Befehl enthält, so spricht man von einem Speicherwort.



## ASCII

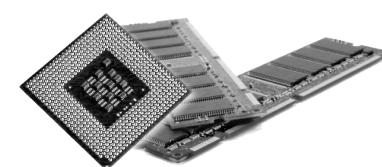
- American Standard Code for Information Interchange
- „Internationaler“ Standard zur Interpretation von Zeichen (der Tastatur)
- Nutzung von 7-Bit Speicher für Dezimalzahl
  - Enthält Steuerzeichen (NULL) am Anfang
  - Klein- und Großbuchstaben in alphabetischer Reihenfolge
  - Ziffern in natürlicher Reihenfolge



## ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0 000	NUL	(null)	32	20 040	&#32;	Space		64	40 100	&#64;	Ø	ø	96	60 140	&#96;	‘	‘
1	1 001	SOH	(start of heading)	33	21 041	&#33;	!	!	65	41 101	&#65;	A	a	97	61 141	&#97;	‘	‘
2	2 002	STX	(start of text)	34	22 042	&#34;	”	”	66	42 102	&#66;	B	b	98	62 142	&#98;	‘	‘
3	3 003	ETX	(end of text)	35	23 043	&#35;	#	#	67	43 103	&#67;	C	c	99	63 143	&#99;	‘	‘
4	4 004	EOT	(end of transmission)	36	24 044	&#36;	\$	\$	68	44 104	&#68;	D	d	100	64 144	&#100;	‘	‘
5	5 005	ENQ	(enquiry)	37	25 045	&#37;	%	%	69	45 105	&#69;	E	e	101	65 145	&#101;	‘	‘
6	6 006	ACK	(acknowledge)	38	26 046	&#38;	&	&	70	46 106	&#70;	F	f	102	66 146	&#102;	‘	‘
7	7 007	BEL	(bell)	39	27 047	&#39;	‘	‘	71	47 107	&#71;	G	g	103	67 147	&#103;	‘	‘
8	8 010	BS	(backspace)	40	28 050	&#40;	(	(	72	48 110	&#72;	H	h	104	68 150	&#104;	‘	‘
9	9 011	TAB	(horizontal tab)	41	29 051	&#41;	)	)	73	49 111	&#73;	I	i	105	69 151	&#105;	‘	‘
10	A 012	LF	(NL line feed, new line)	42	2A 052	&#42;	*	*	74	4A 112	&#74;	J	j	106	6A 152	&#106;	‘	‘
11	B 013	VT	(vertical tab)	43	2B 053	&#43;	+	+	75	4B 113	&#75;	K	k	107	6B 153	&#107;	‘	‘
12	C 014	FF	(NP form feed, new page)	44	2C 054	&#44;	,	,	76	4C 114	&#76;	L	l	108	6C 154	&#108;	‘	‘
13	D 015	CR	(carriage return)	45	2D 055	&#45;	-	-	77	4D 115	&#77;	M	m	109	6D 155	&#109;	‘	‘
14	E 016	SO	(shift out)	46	2E 056	&#46;	.	.	78	4E 116	&#78;	N	n	110	6E 156	&#110;	‘	‘
15	F 017	SI	(shift in)	47	2F 057	&#47;	/	/	79	4F 117	&#79;	O	o	111	6F 157	&#111;	‘	‘
16	10 020	DLE	(data link escape)	48	30 060	&#48;	0	0	80	50 120	&#80;	P	p	112	70 160	&#112;	‘	‘
17	11 021	DC1	(device control 1)	49	31 061	&#49;	1	1	81	51 121	&#81;	Q	q	113	71 161	&#113;	‘	‘
18	12 022	DC2	(device control 2)	50	32 062	&#50;	2	2	82	52 122	&#82;	R	r	114	72 162	&#114;	‘	‘
19	13 023	DC3	(device control 3)	51	33 063	&#51;	3	3	83	53 123	&#83;	S	s	115	73 163	&#115;	‘	‘
20	14 024	DC4	(device control 4)	52	34 064	&#52;	4	4	84	54 124	&#84;	T	t	116	74 164	&#116;	‘	‘
21	15 025	NAK	(negative acknowledge)	53	35 065	&#53;	5	5	85	55 125	&#85;	U	u	117	75 165	&#117;	‘	‘
22	16 026	SYN	(synchronous idle)	54	36 066	&#54;	6	6	86	56 126	&#86;	V	v	118	76 166	&#118;	‘	‘
23	17 027	ETB	(end of trans. block)	55	37 067	&#55;	7	7	87	57 127	&#87;	W	w	119	77 167	&#119;	‘	‘
24	18 030	CAN	(cancel)	56	38 070	&#56;	8	8	88	58 130	&#88;	X	x	120	78 170	&#120;	‘	‘
25	19 031	EM	(end of medium)	57	39 071	&#57;	9	9	89	59 131	&#89;	Y	y	121	79 171	&#121;	‘	‘
26	1A 032	SUB	(substitute)	58	3A 072	&#58;	:	:	90	5A 132	&#90;	Z	z	122	7A 172	&#122;	‘	‘
27	1B 033	ESC	(escape)	59	3B 073	&#59;	;	;	91	5B 133	&#91;	[	{	123	7B 173	&#123;	‘	‘
28	1C 034	FS	(file separator)	60	3C 074	&#60;	<	<	92	5C 134	&#92;	\		124	7C 174	&#124;	‘	‘
29	1D 035	GS	(group separator)	61	3D 075	&#61;	=	=	93	5D 135	&#93;	]	}	125	7D 175	&#125;	‘	‘
30	1E 036	RS	(record separator)	62	3E 076	&#62;	>	>	94	5E 136	&#94;	^	~	126	7E 176	&#126;	‘	‘
31	1F 037	US	(unit separator)	63	3F 077	&#63;	?	?	95	5F 137	&#95;	_	DEL	127	7F 177	&#127;	‘	‘

Source: [www.LookupTables.com](http://www.LookupTables.com)



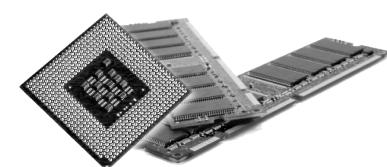
## UTF-8

- Universal Character Set Transformation Format
- 8 Bit
- mit ASCII kompatibel



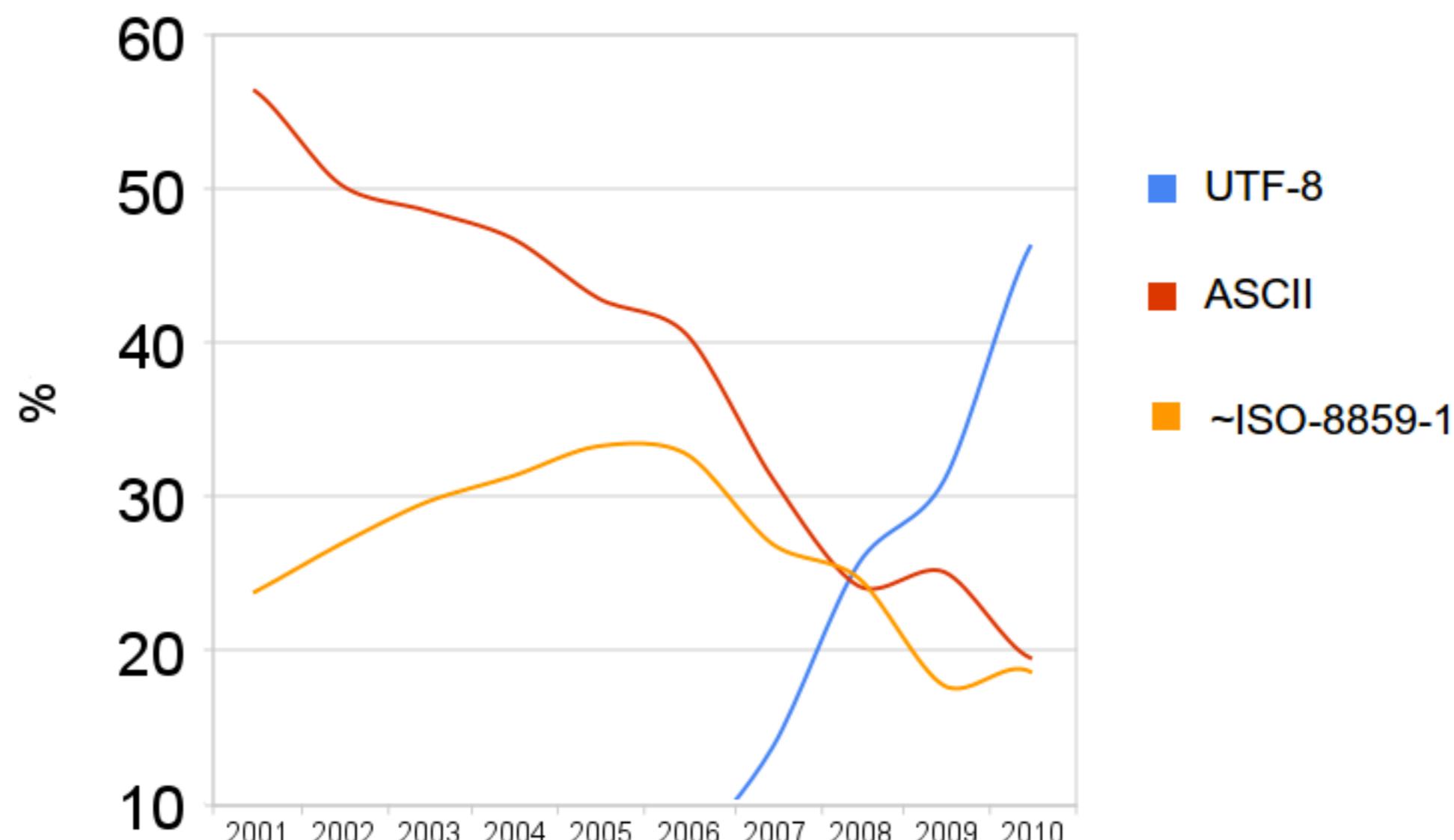
Beispiele für UTF-8 Kodierungen

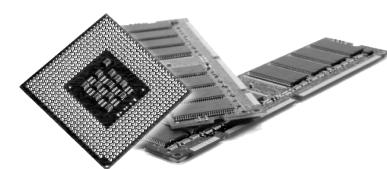
Zeichen	Unicode	Unicode binär	UTF-8 binär	UTF-8 hexadezimal
Buchstabe y	U+0079	00000000 01111001	01111001	79
Buchstabe ä	U+00E4	00000000 11100100	11000011 10100100	C3 A4
Zeichen für eingetragene Marke ®	U+00AE	00000000 10101110	11000010 10101110	C2 AE
Eurozeichen €	U+20AC	00100000 10101100	11100010 10000010 10101100	E2 82 AC
Violinschlüssel ♫	U+1D11E	00000001 11010001 00011110	11110000 10011101 10000100 10011110	F0 9D 84 9E



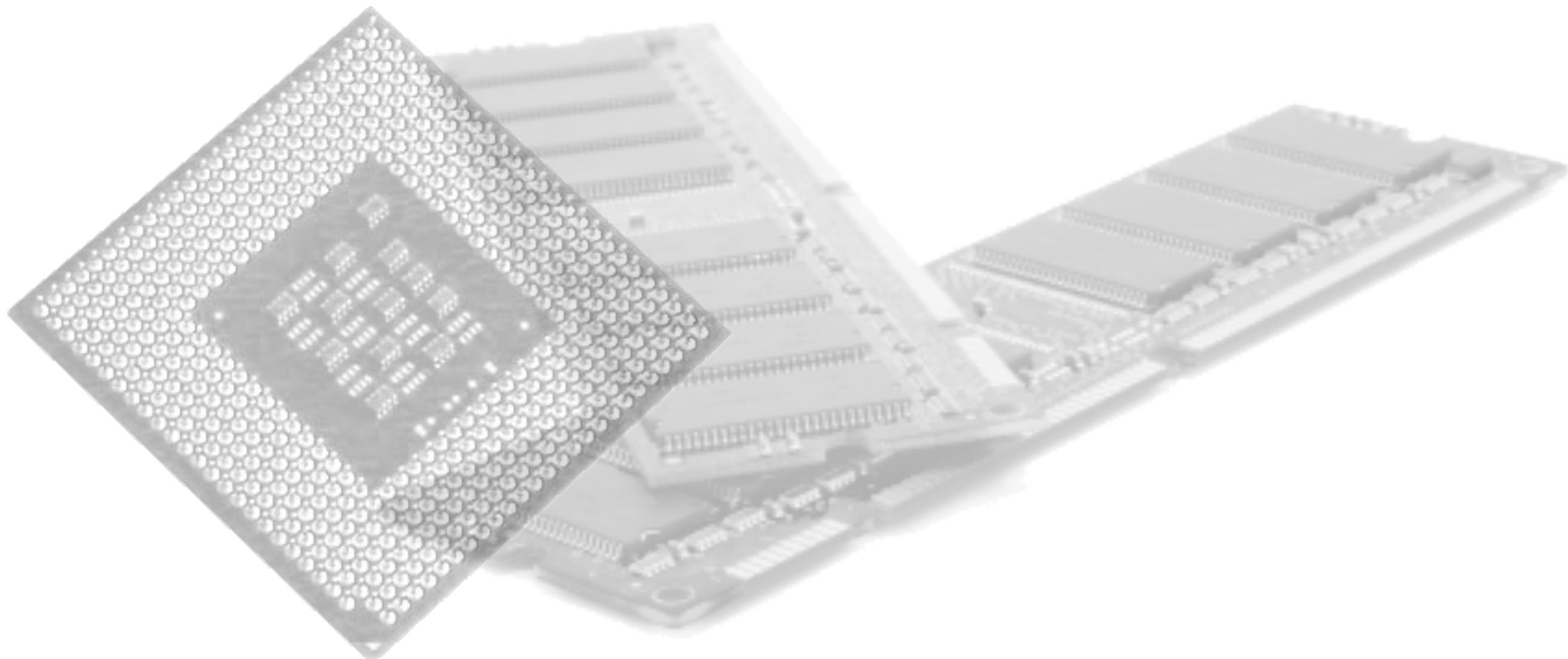
## UTF-8

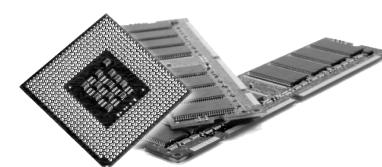
Growth of UTF-8 on the Web





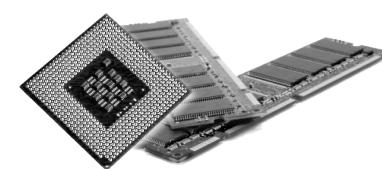
## Hardware und Software





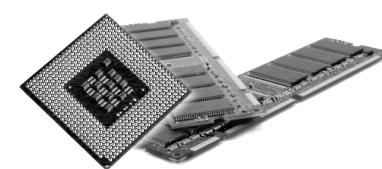
## Software

- Sammelbegriff für Programme und die zugehörigen Daten
- Die Software bestimmt das Handeln des softwaregesteuerten Gerätes
- Die Hardware führt die Software aus
- ermöglicht eine individuelle Arbeitsweise auf starrer Hardware



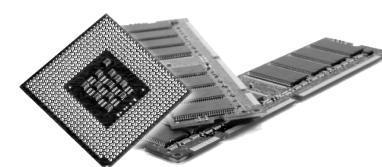
## Software

- Systemsoftware
  - entlastet den Benutzer von Verwaltungsaufgaben
  - ermöglicht eine komfortable Bedienung der DV
- Anwendungssoftware
  - Die Gesamtheit der Anwendungsprogramme (z.B: Textverarbeitung, Datenbanken, Buchhaltung, Kalender)

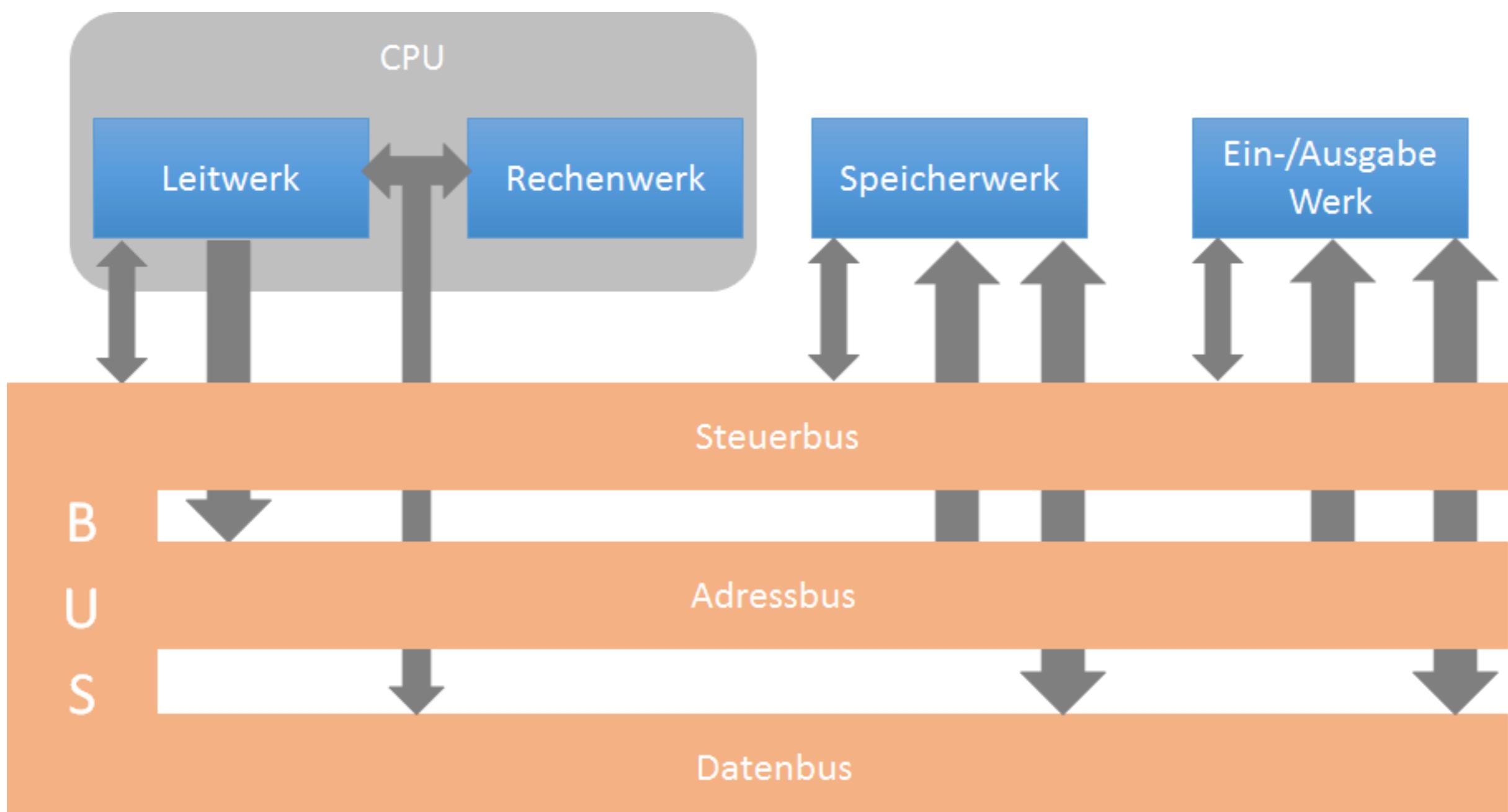


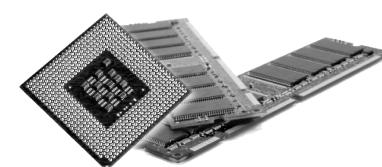
## Klassische Rechnerarchitekturen

- Die meisten heutzutage verwendeten Computer sind nach dem Konzept der Von-Neumann-Architektur aufgebaut:
  - Problemunabhängige Struktur
  - gemeinsamer Speicher für Computerprogrammbefehle und Daten
  - SISD- Architektur (Single Instruction, Single Data)



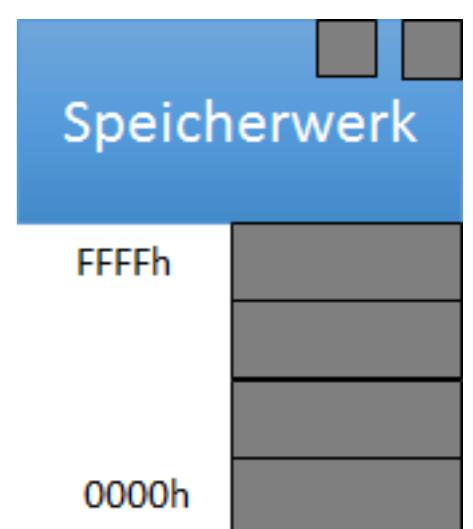
## Der von-Neumann Rechner

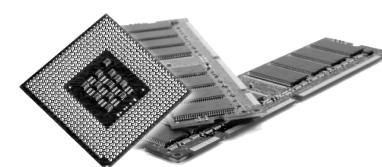




## Der von-Neumann Rechner

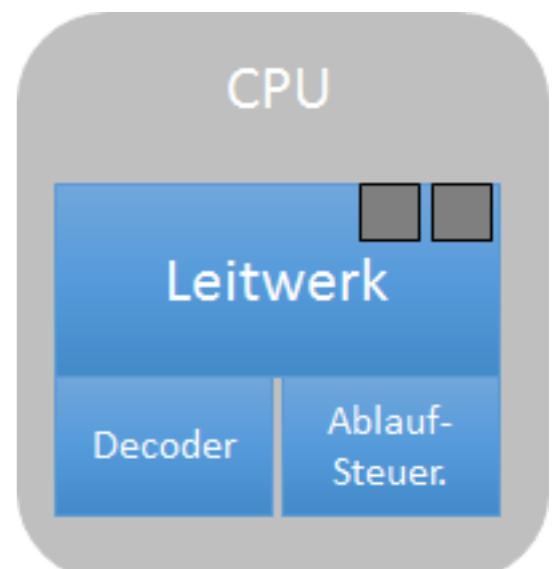
- Nimmt Programmbefehle und Daten auf
- Binär verschlüsselte Daten
- Prinzipiell kein Unterschied zwischen Daten und Befehlen
- Adress- und Datenregister
- Unterteilung in Speicherplätze (Speicherzellen, Speicherworte)
- Angesprochen über Adressen

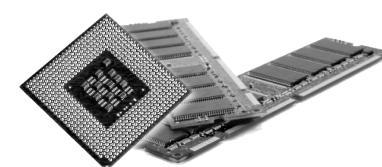




## Der von-Neumann Rechner

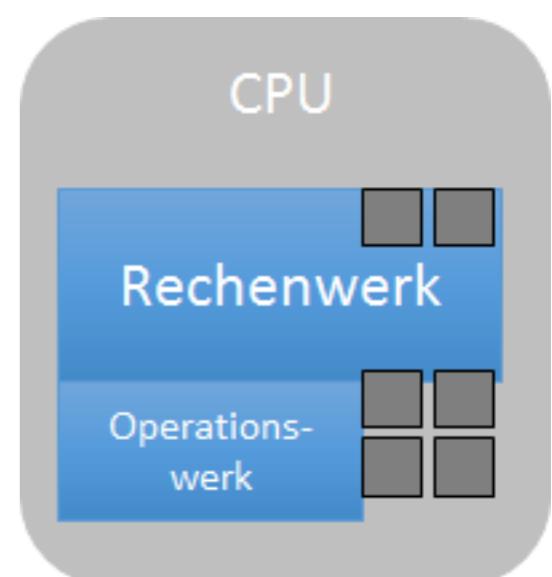
- Steuerwerk oder Control Unit (CU)
- Dekodiert Befehle und steuert den Programmablauf (Ablaufsteuerung)
- Befehlszähler und -Register
- Befehle werden interpretiert und deren Ausführung veranlasst, gesteuert und überwacht.

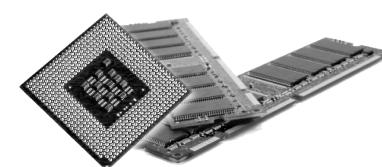




## Der von-Neumann Rechner

- Execution Unit (EX)
- Verknüpft (arithmetische und logische Befehle) und verändert die zu bearbeitenden Daten
- Besitzt Registerblock, Operationswerk (ALU – arithmetic logic unit) und Statusregister

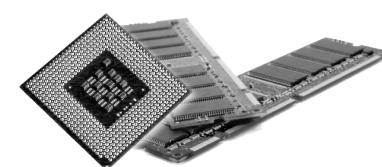




## Der von-Neumann Rechner

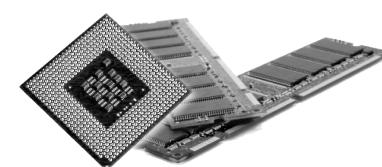
- Übernahme bzw. Übergabe von Programmen und Daten
- Ziel/Quelle können externe Einheiten oder auch Speicher sein

E/A - Werk



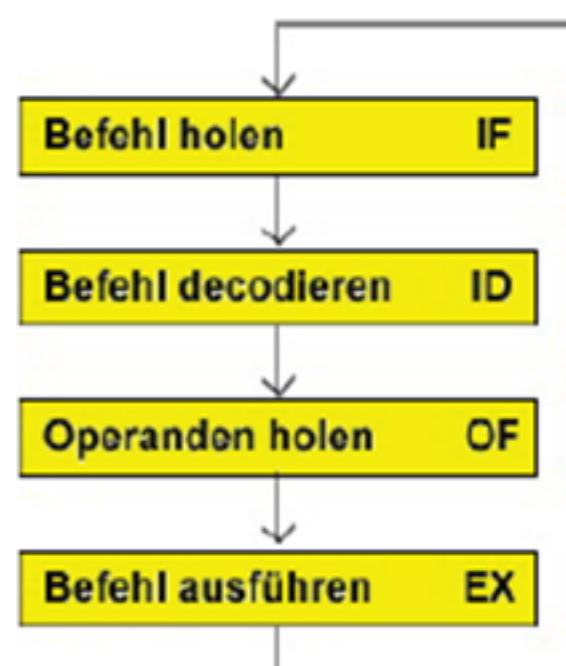
## Arbeitsweise

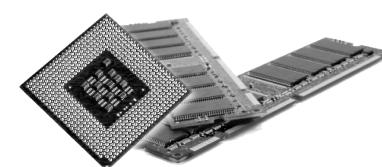
- 4-Phasen für Befehlsverarbeitung:
  - Befehlsholphase (**I**nstruction **F**etch)
  - Befehlsentschlüsselungsphase (**I**nstruction **D**ecode)
  - Befehlsoperandenholphase (**O**perand **F**etch)
  - Befehlausführungsphase (**I**nstruction **E**Xecute)
- Häufig wird Phase 3 (OF) als Bestandteil der Phase 4 (EX) betrachtet



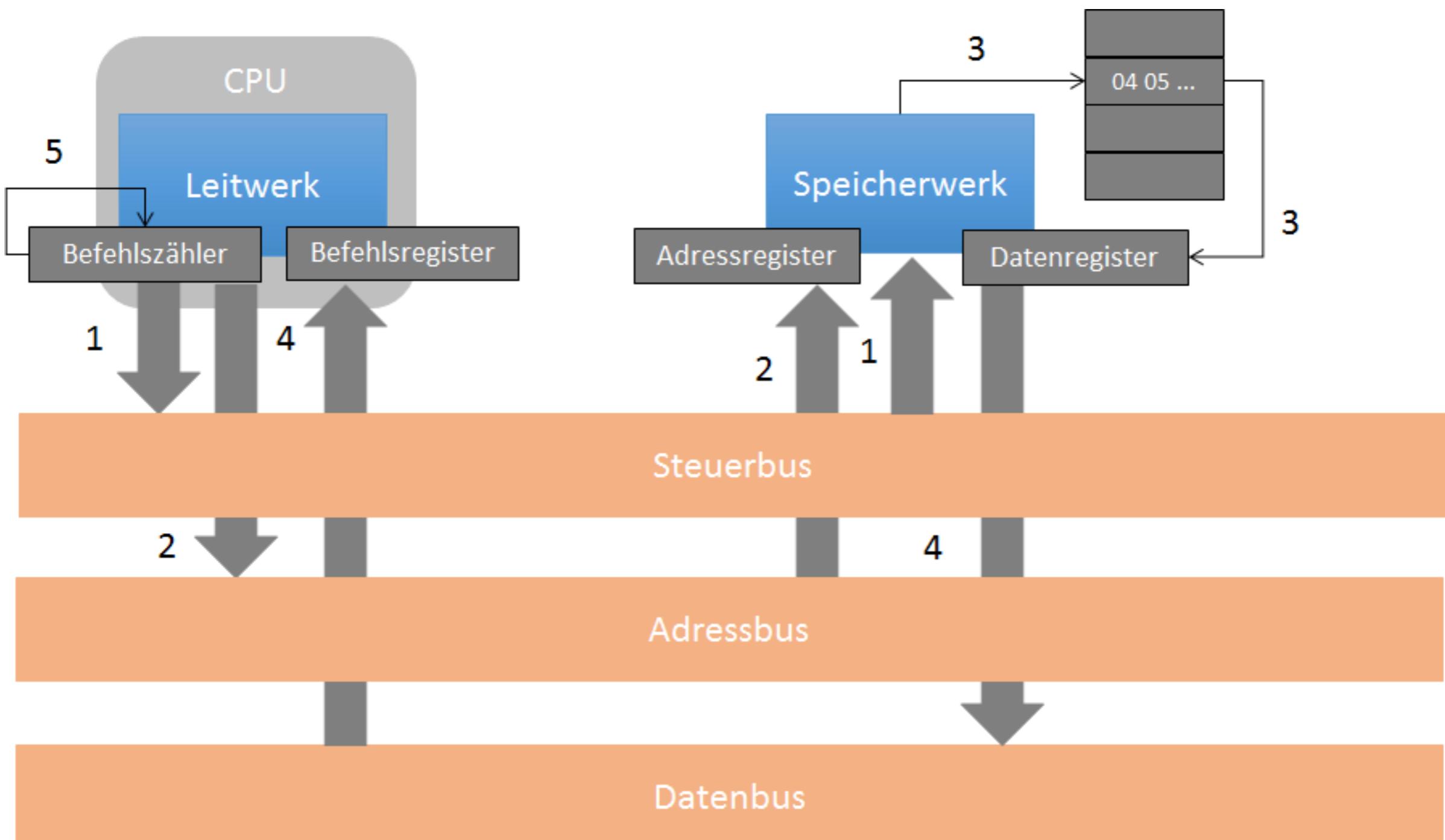
## Befehlszyklus

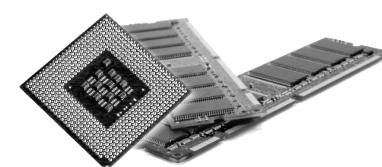
- Der Befehlszyklus wird von der CPU ständig durchlaufen:
  - die Befehle stehen im Speicher
  - das Leitwerk "weiß" jederzeit, welcher Befehl als nächster auszuführen ist
  - die Adresse des nächsten auszuführenden Befehls steht in einem speziellen Register des Leitwerks, dem Befehlszähler (Program Counter, PC, BZ)
  - üblicherweise stehen aufeinanderfolgende Befehle in aufeinander folgenden Speicherzellen, der zuerst auszuführende Befehl hat die niedrigste Adresse
  - zu Beginn des Programms wird der BZ mit dessen Startadresse geladen





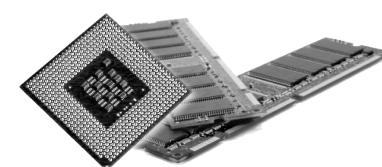
# Arbeitsweise - fetch





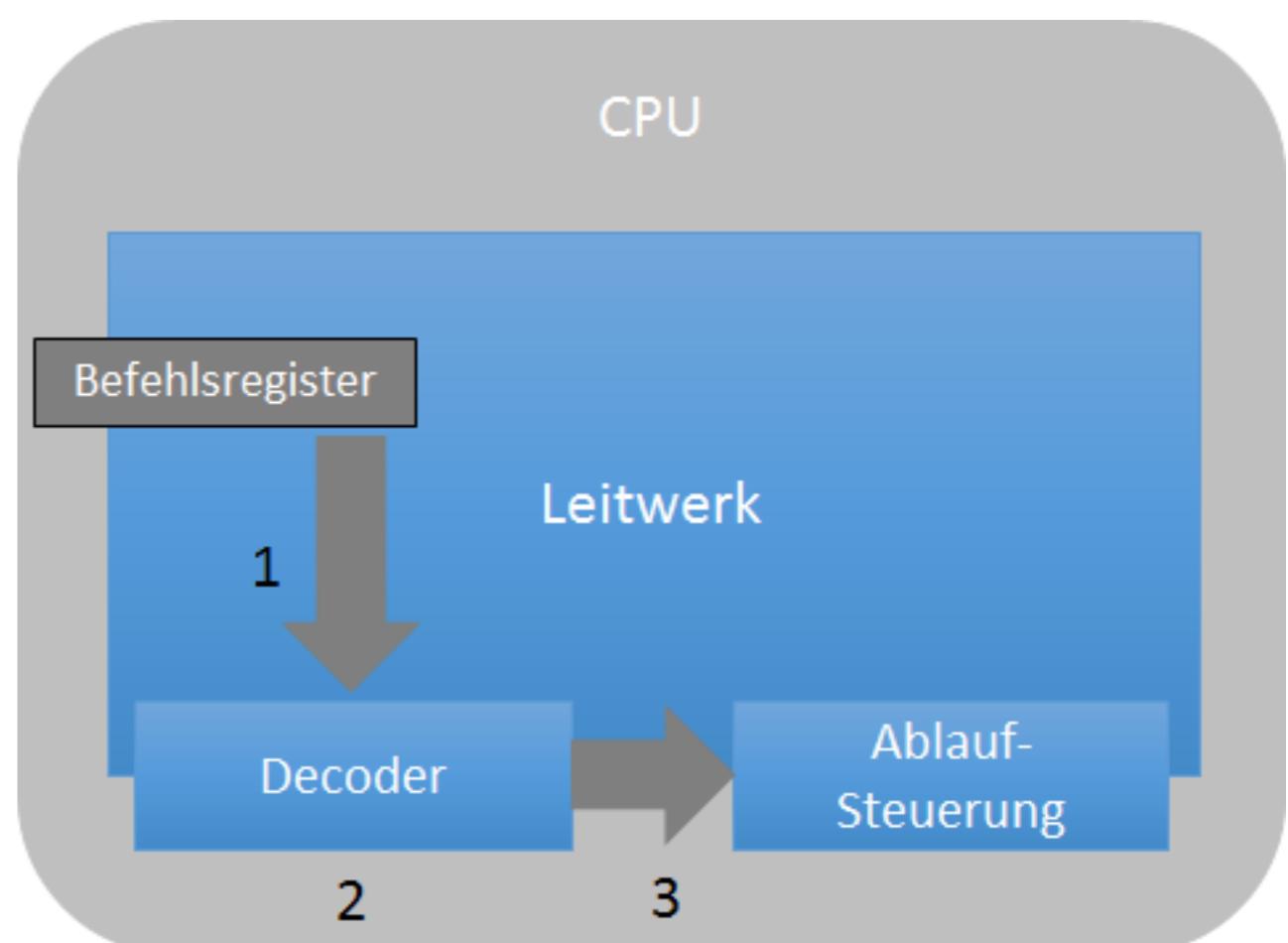
## Arbeitsweise - fetch

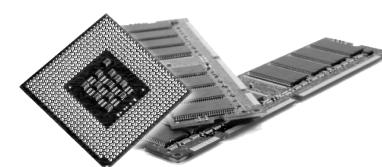
1. Das Leitwerk aktiviert über den Steuerbus das Adresswerk und den -Bus
2. Das Leitwerk sendet die Adresse des Befehlszählers über den Adressbus an das Speicherwerk zur Speicherung im Adressregister
3. Das über den im Adressregister gespeicherten Wert adressierte Speicherwort wird ausgelesen und im Datenregister gespeichert
4. Das Befehlswort wird über den Datenbus im Befehlsregister gespeichert
5. Befehlszähler wird erhöht oder gesetzt (Sprungbefehle)



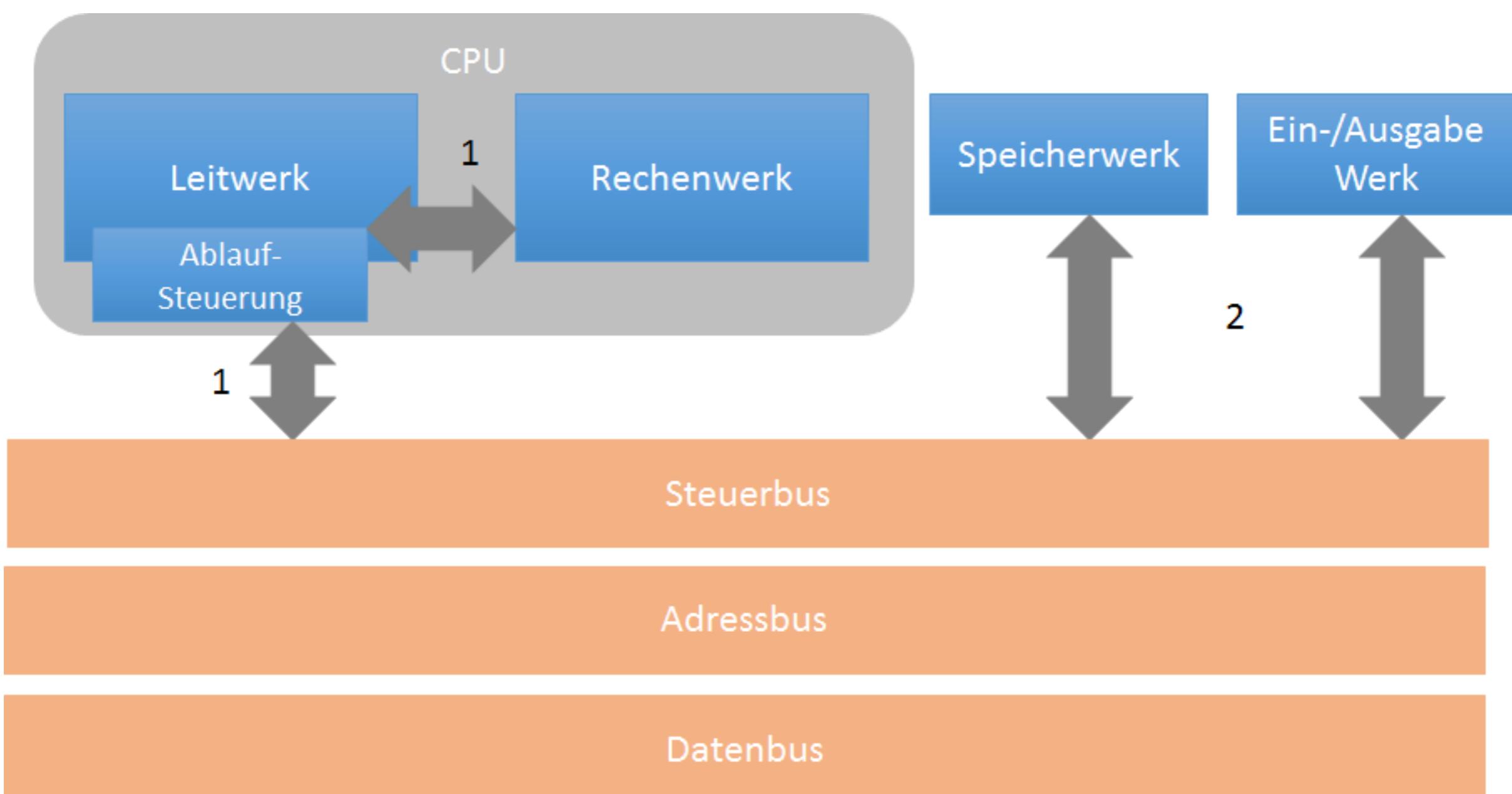
## Arbeitsweise - decode

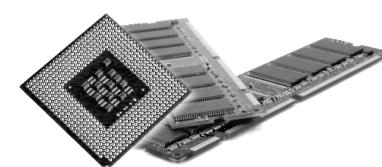
1. Das Befehlswort wird aus dem Befehlsregister an den Decoder gesendet
2. Der Decoder entschlüsselt das Befehlswort
3. Der Befehl wird zur weiteren Abarbeitung an die Ablaufsteuerung gesendet





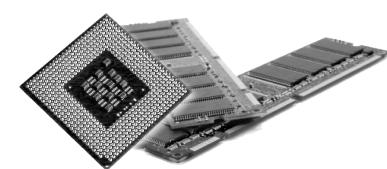
## Arbeitsweise - execute





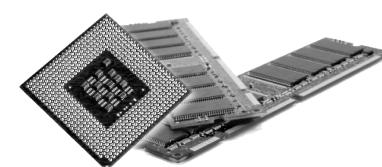
## Arbeitsweise - execute

1. Die Ablaufsteuerung aktiviert je nach Befehl das Rechenwerk oder das Adress- oder E/A-Werk über den Steuerbus
2. Je nach Ergebnis oder Folgeschritt werden Daten über die entsprechenden Busse übertragen (z.B. Operanden)



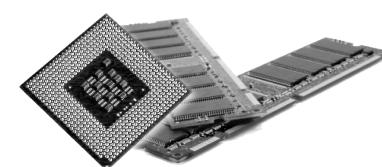
## Vorteile

- Problemunabhängig / Universell
- Geringer Hardwareaufwand
- Minimaler Speicheraufwand
- Deterministischer Programmablauf wegen sequenzieller Arbeitsweise (**SISD** = single instruction, single data)



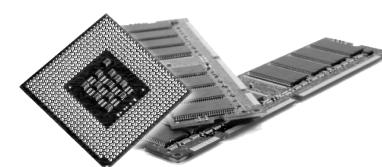
## Nachteile

- Von-Neumann-Flaschenhals: Transport von Daten und Befehlen über den Bus
- Keine Parallelität der Arbeitsschritte
- Aufgabenspezifische Register (1x) erlauben keine Multiuser/-threading-Unterstützung
- Memory-Wall: Verarbeitungsgeschwindigkeit der CPU höher als Zugriff auf Speicher



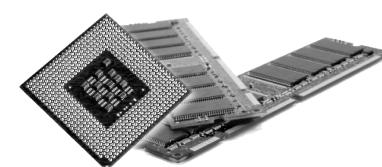
## Innovative Rechnerarchitekturen

- Erhöhung der Leistungsfähigkeit des v. Neumann-Rechners
  - **Verlagerung der Prozessorfunktionen bei der Ein-/Ausgabe auf das E/A-Werk** ("intelligente" Schnittstellen, E/A-Prozessoren, Vor-Rechner, Front-End-Rechner)
  - Weiterentwicklung des Architekturprinzips durch Steuerung der einzelnen Funktionseinheiten über eigene Prozessoren
  - Bearbeiten einzelner, spezieller Befehle durch Spezialprozessoren, die abwechselnd mit der "normalen" CPU arbeiten (z. B. Arithmetik-Koprozessor).



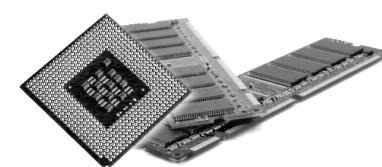
## Innovative Rechnerarchitekturen

- Vermeiden oder Mildern des v. Neumann-Flaschenhalses
  - Parallelisierung verschiedener Abläufe im Computer
  - mehreren parallelen Rechenwerke, mit gemeinsamem Steuerwerk -> **SIMD** (single instruction, multiple data)



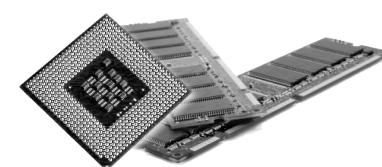
## Innovative Rechnerarchitekturen

- Asynchrone Trennung von Befehlshol und -ausführung; Pipelining
  - mehrere hintereinander geschaltete Rechenwerke
  - Es wird ein Befehl nach dem anderen abgearbeitet
  - Während des 2. Taktes des 1. Befehls wird der zweite Befehl geholt usw.



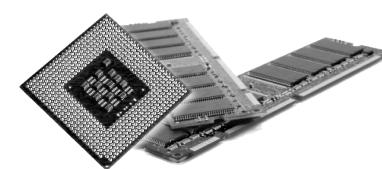
## Innovative Rechnerarchitekturen

- Verlagerung der Datenhaltung; Caching
  - kB-große Speicherzellen nahe an der CPU
  - Cache wird durch Pre-fetching gefüllt und von Controller verwaltet
  - Nachteile bei Cache-Misses im Fall von Sprungbefehlen

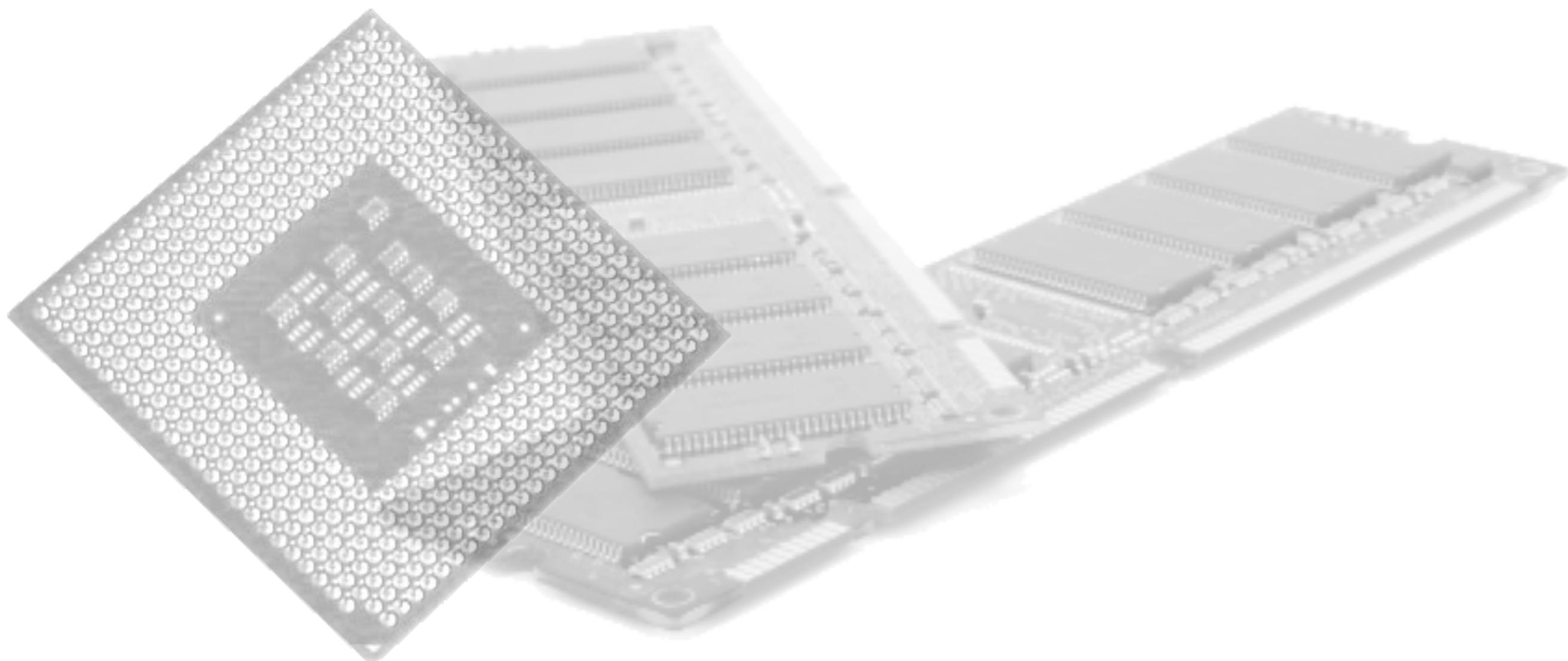


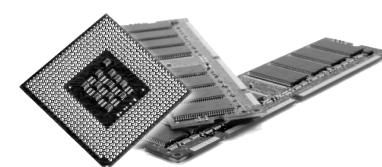
## Weiterführende Rechenwerke

- Teilaufgaben werden auf Grund des Pipelinings überlappend ausgeführt (**MIMD**: multiple instruction, multiple data):
  - Vektorrechner - Kombination mehrerer Pipelines
  - Multiprozessorsysteme - Gleicher Speicher, unabhängige Prozessoren
  - Polyprozessorsysteme - Multiprozessorsysteme mit verteilter Kontrolle. CPU's arbeiten autonom, kommunizieren aber miteinander.

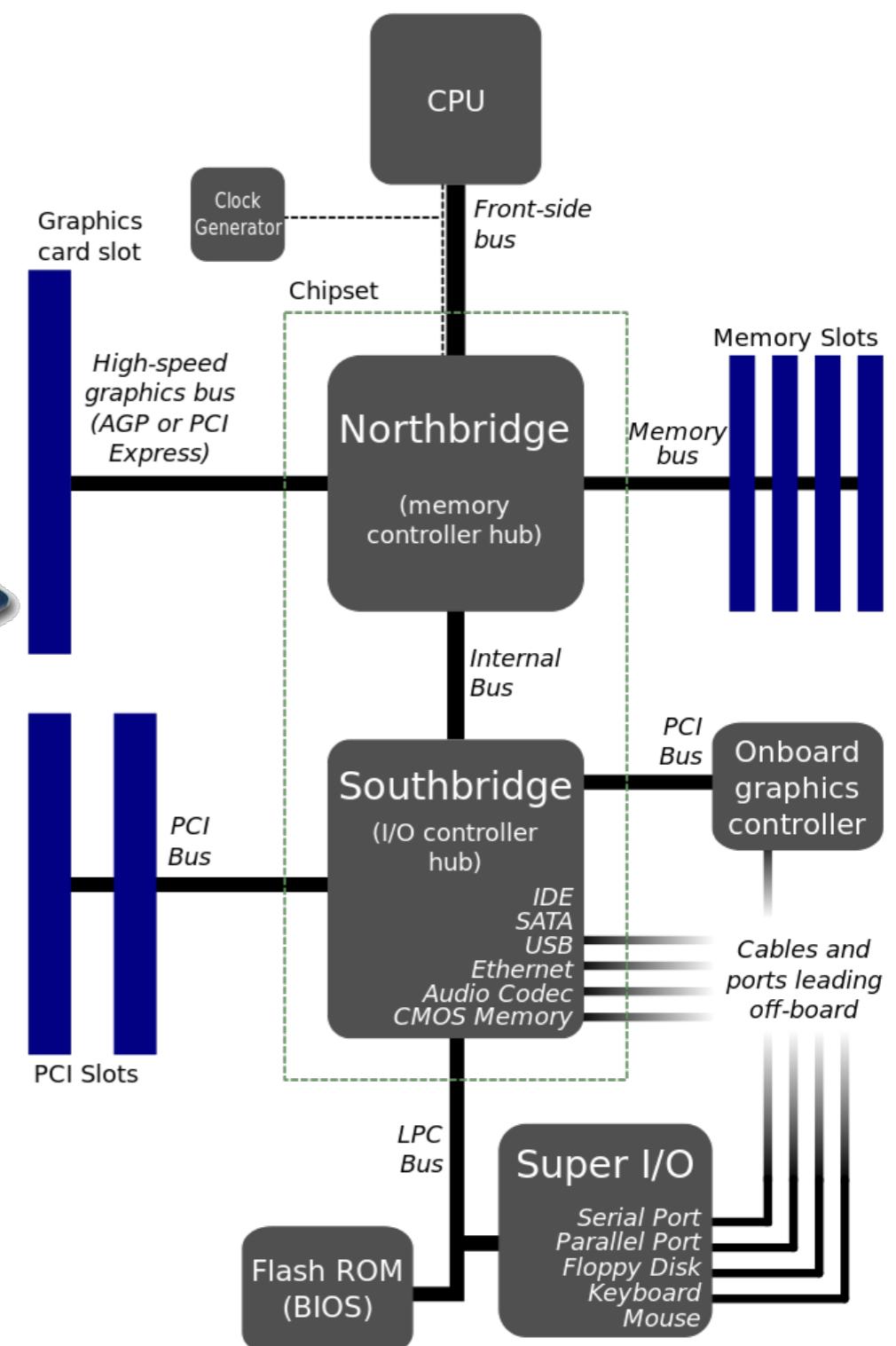
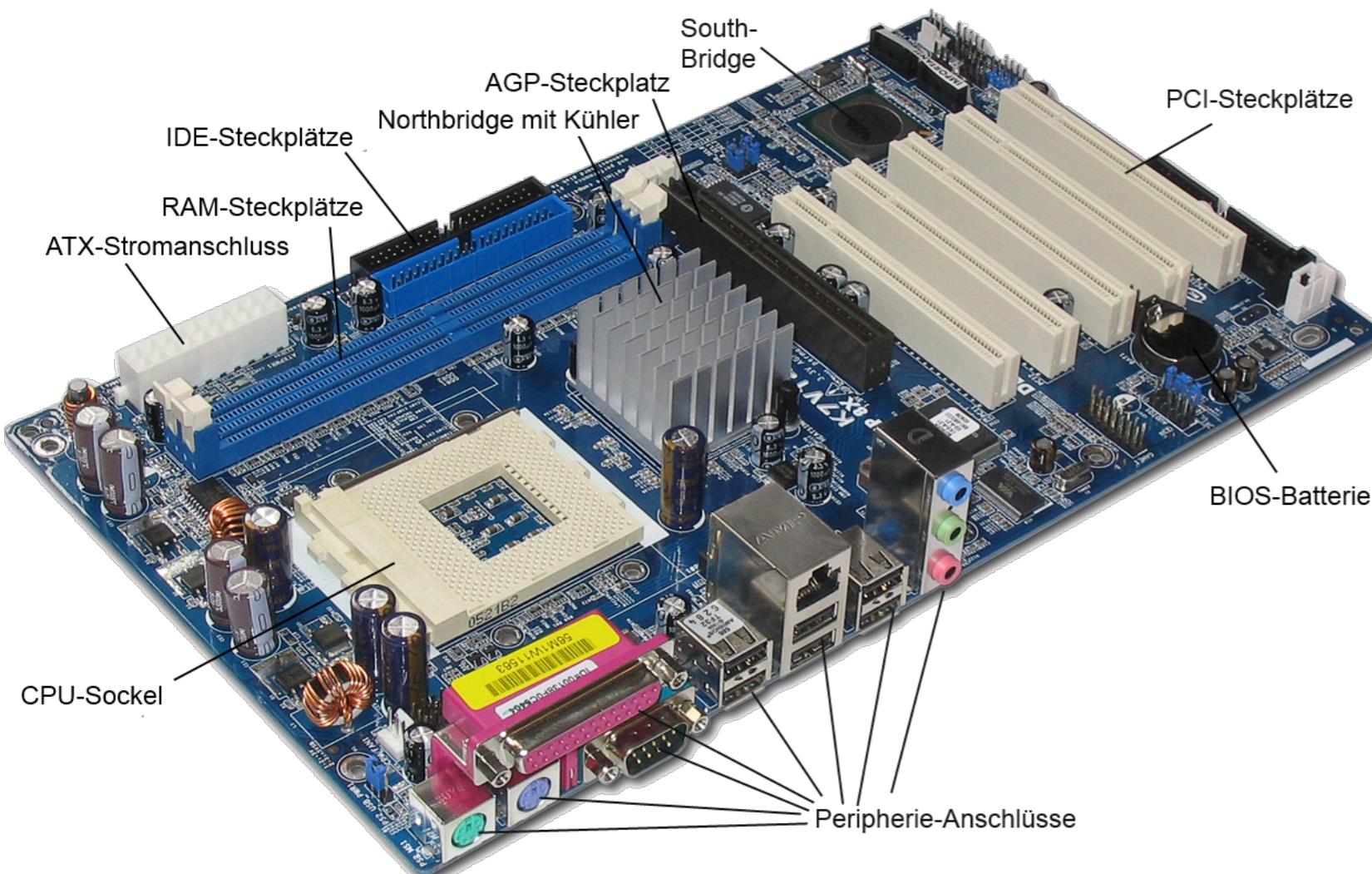


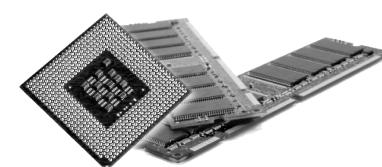
## Moderne Architekturen





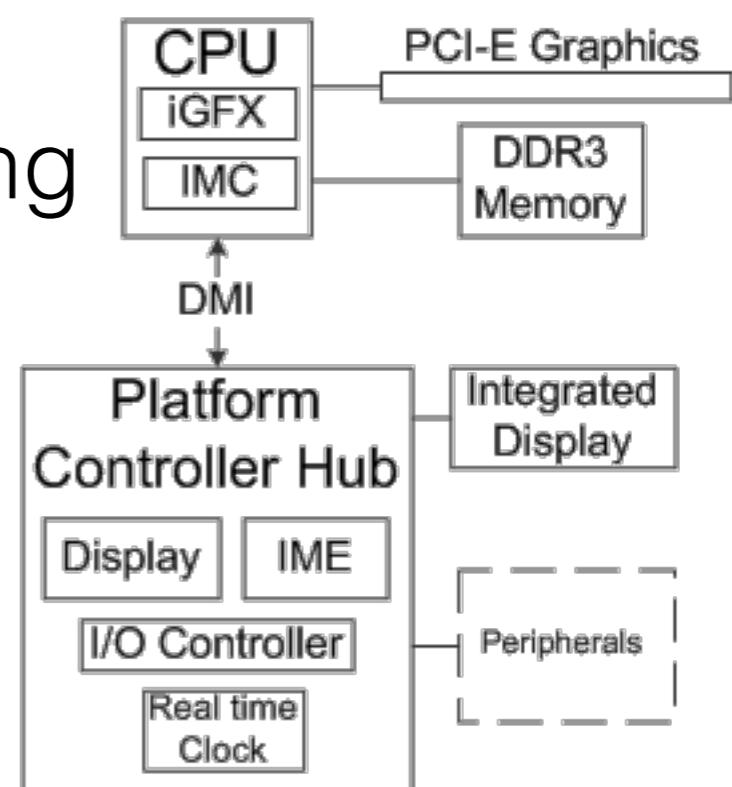
## Moderne Architekturen

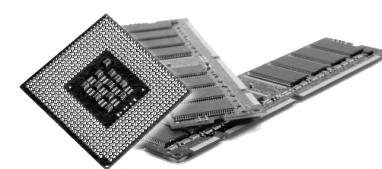




# Platform Controller Hub

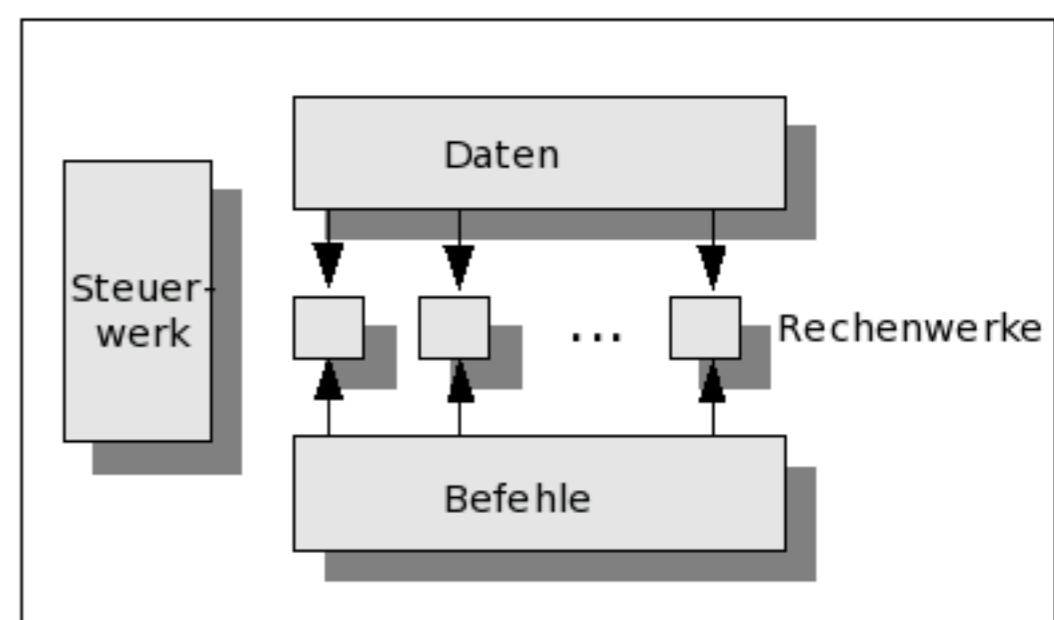
- 2008 vorgestellt von Intel
- Speichercontroller und PCI-Anbindung in die CPU integriert
- Direct Media Interface (DMI)
- PCH übernimmt die verbleibenden Funktionen der Northbridge und die Funktionen der Southbridge

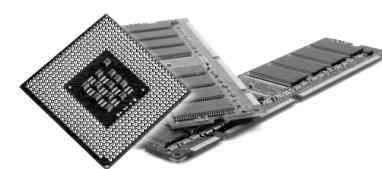




## Harvard-Architektur

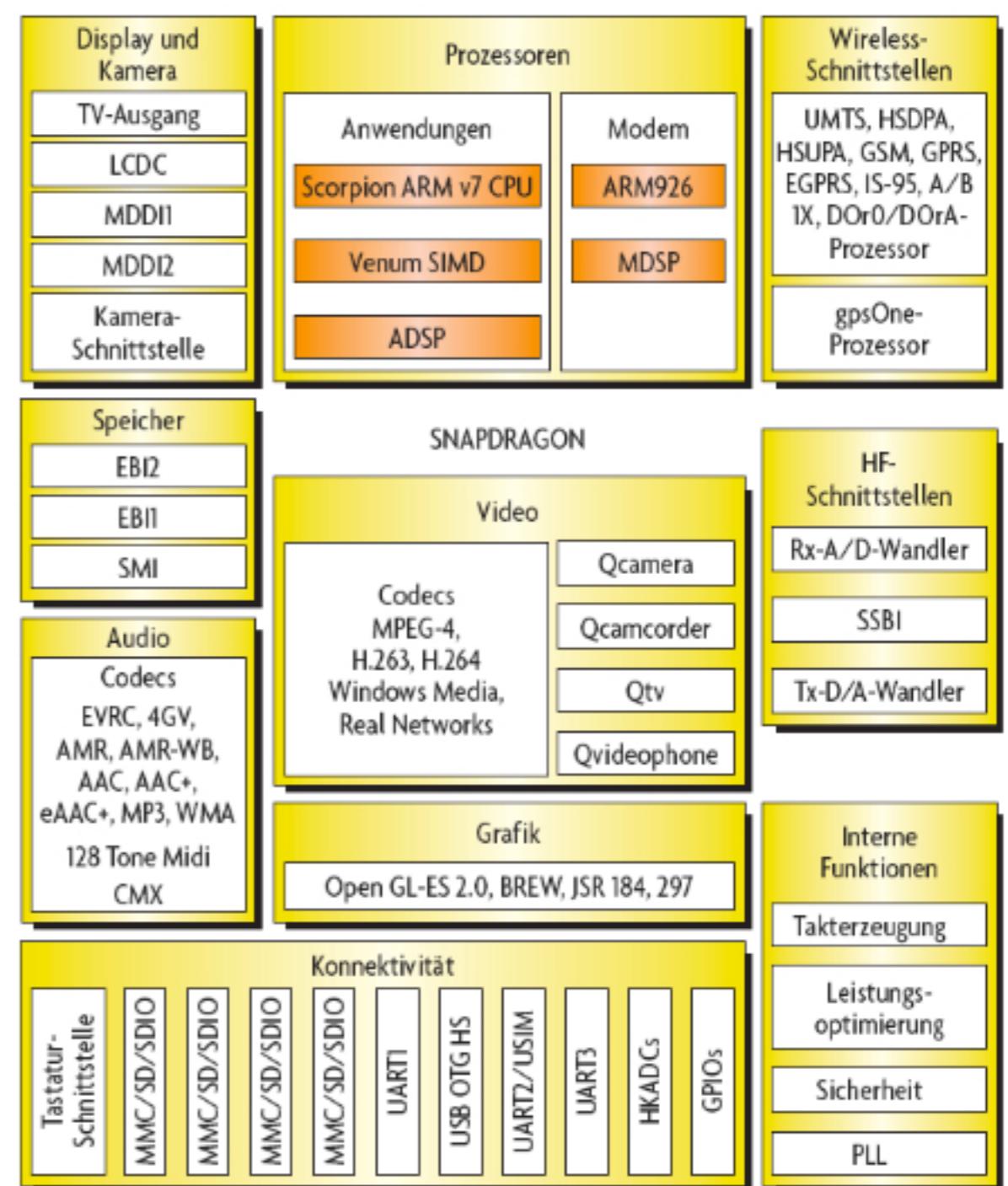
- Daten und Befehle werden getrennt gespeichert
- 2 getrennte Bussysteme
- gleichzeitiges Laden von Daten und Befehlen möglich
- z.B. ARM-Architektur

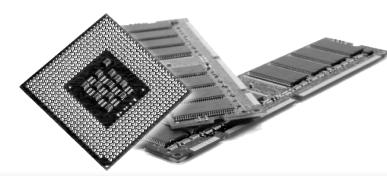




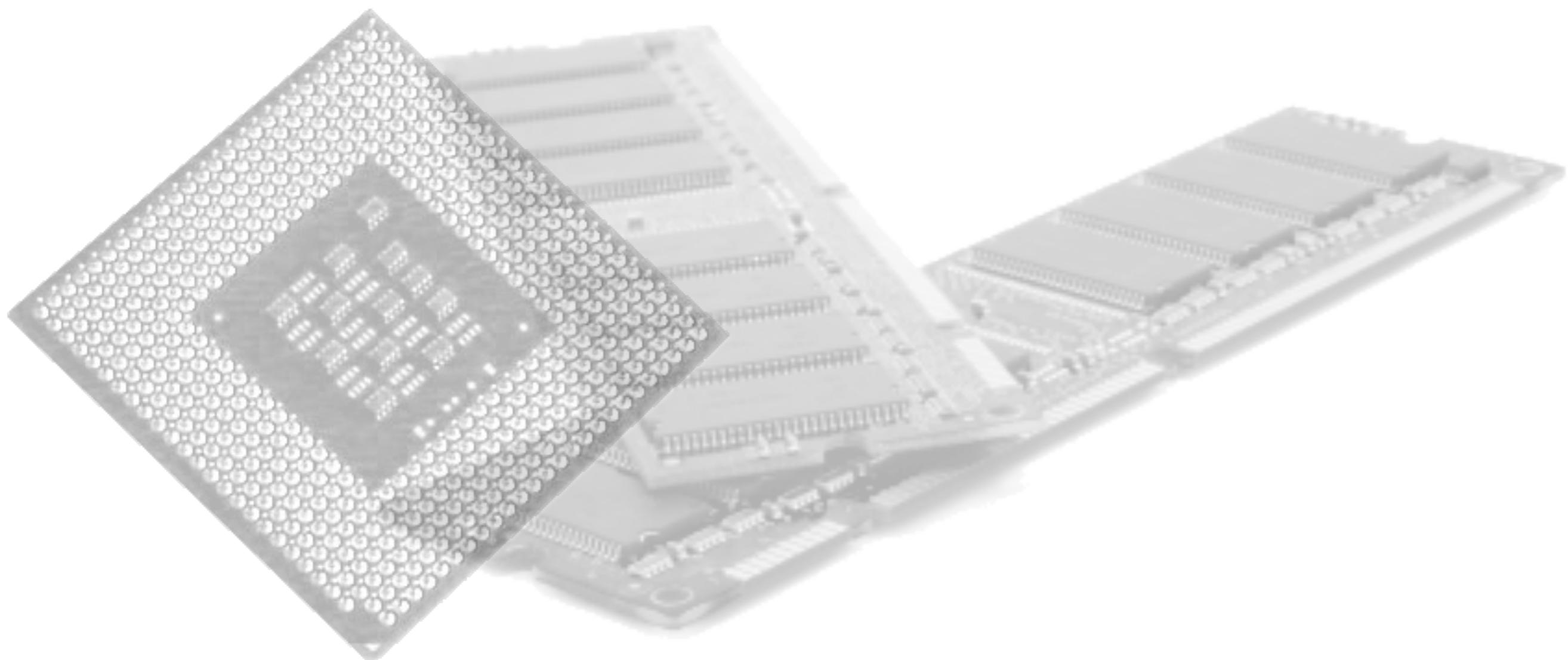
# System-on-a-Chip

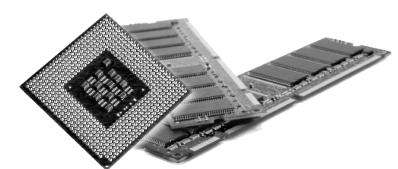
- Integration (fast) aller Funktionen auf einen Chip
- embedded Systems



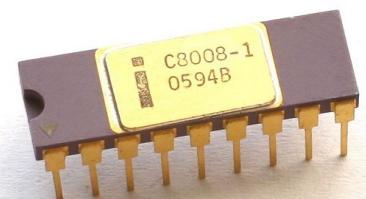


## Mikroprozessoren

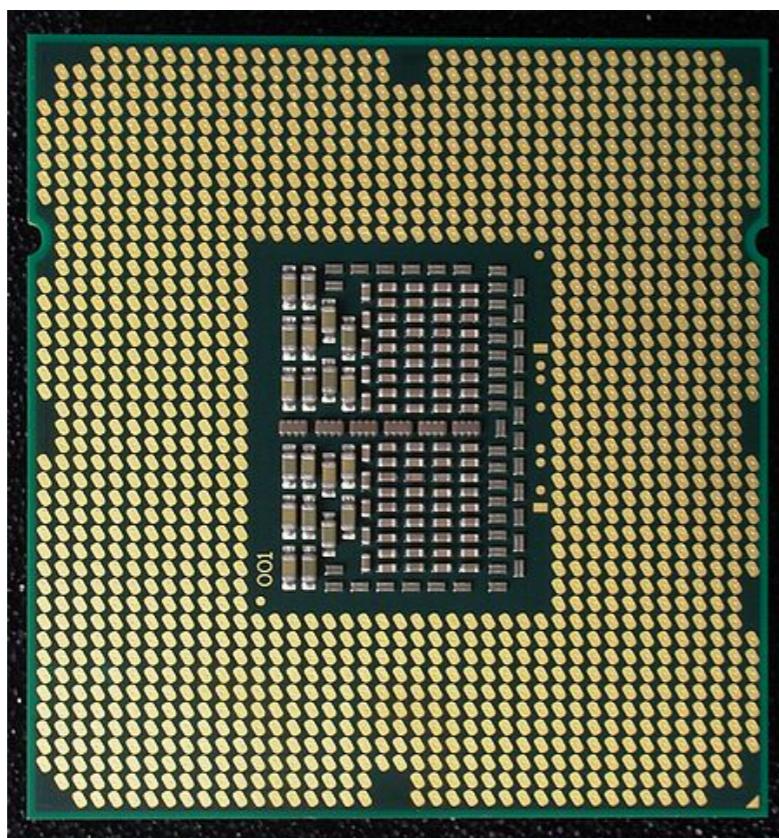


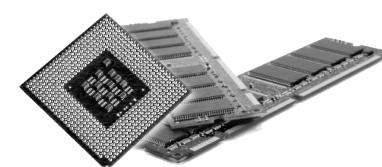


## Mikroprozessor



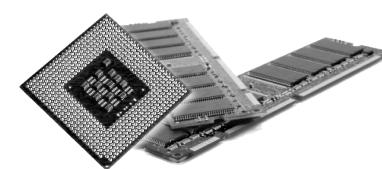
- Prozessor in sehr kleinem Maßstab
- alle Bauteile des Prozessors auf einem Mikrochip



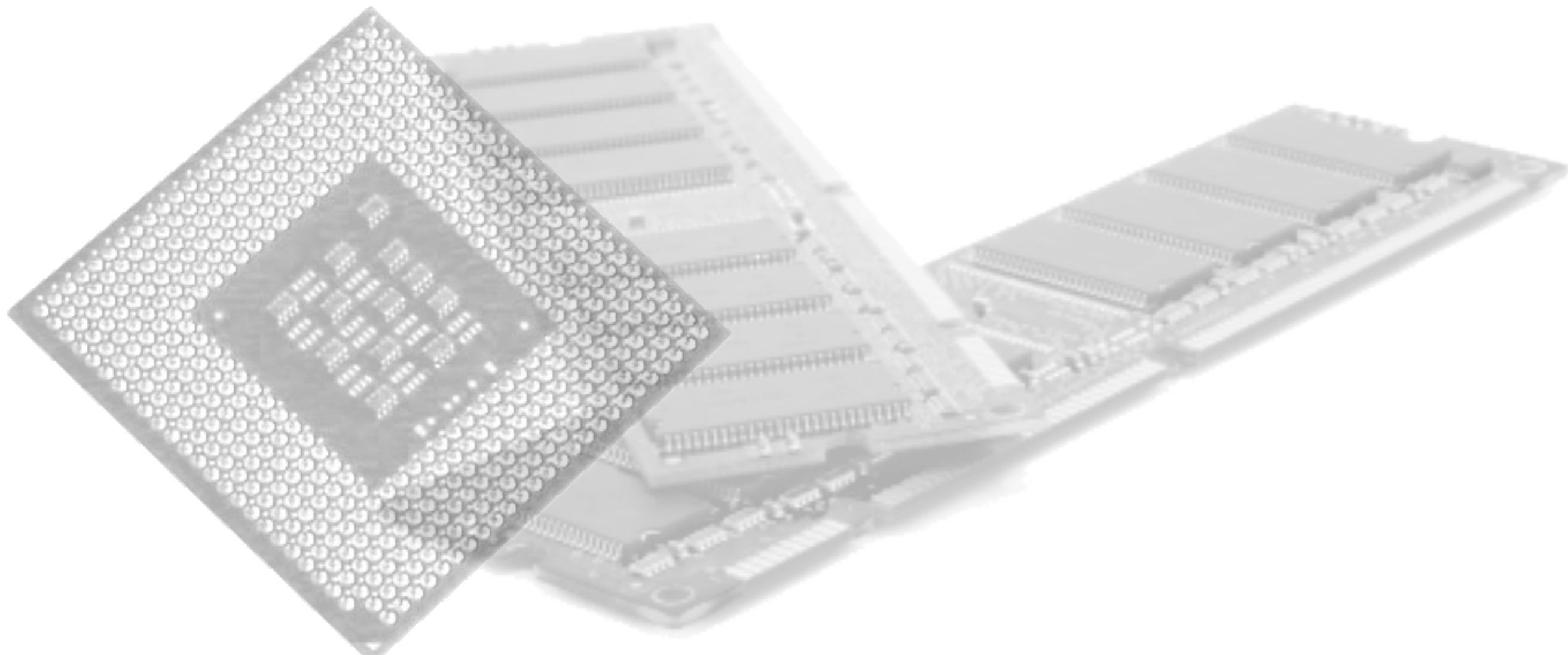


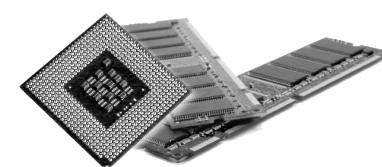
## Mikroprozessor

- Intel 8086 (1978)
  - 16 Bit
  - Urvater der x86 Familie
- x86
  - für 64 Bit: x86-64 (oder x64)
  - 8086 erste Generation
  - Intel Core i-Serie aktuelle, 9. Generation



## Ergänzungen Zahlensysteme + Arithmetik



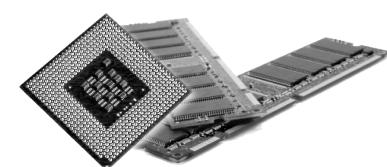


# Umrechnung nach Dezimal

- $101,11_{(2)} = 1 * \mathbf{2^2} + 0 * 2^1 + 1 * \mathbf{2^0} + 1 * \mathbf{2^{-1}} + 1 * \mathbf{2^{-2}}$   
 $= \mathbf{4} + 0 + \mathbf{1} + \mathbf{0,5} + \mathbf{0,25} = 5,75$

$$2^{-2} = \frac{1}{2^2}$$

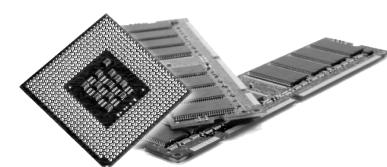
- $5,5_{(16)} = 5 * 16^0 + 5 * \mathbf{16^{-1}} =$   
 $5 + 5 * \mathbf{0,0625} = 5,3125$



# von Dezimal zu Binär

- $1,8125_{(10)}$  zu Binär:

$$\begin{array}{rcl} \mathbf{0,8125} & *2 & = \mathbf{1,625} \\ \mathbf{0,625} & *2 & = \mathbf{1,25} \\ \mathbf{0,25} & *2 & = \mathbf{0,5} \\ \mathbf{0,5} & *2 & = \mathbf{1,0} \\ & & \downarrow \\ & & = 1, \mathbf{1101}_{(2)} \end{array}$$

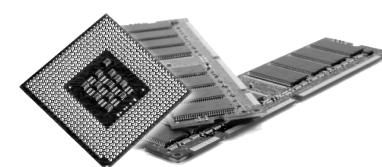


# Addition im 5-er System

- $423,2_{(5)} + 14,4_{(5)} =$

$$\begin{array}{r} 4 \quad 2 \quad 3, \quad 2 \\ + \quad \quad 1 \quad 4, \quad 4 \\ \hline 4 \quad 4 \quad 3, \quad 1 \end{array}$$

Übertrag bei 5

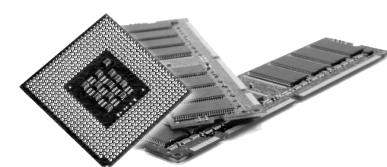


# Subtraktion in Binär

- $110_{(2)} - 11_{(2)} =$

2 von der nächsten Stelle "holen"

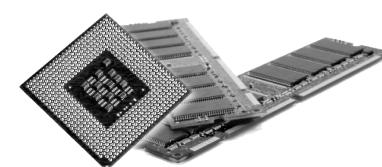
$$\begin{array}{r} & 1 & 1 & 0 \\ - & & 1 & 1 \\ \hline & 1 & 1 & \end{array}$$



# Multiplizieren in Hex

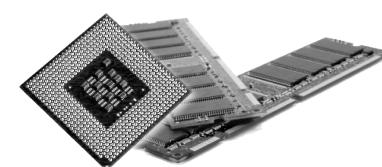
- $19_{(16)} * A1_{(16)} =$

$$\begin{array}{r} 1 \quad 9 \\ * \quad \quad \quad A \quad 1 \\ F \quad A \\ \hline 1 \quad 9 \\ \hline F \quad B \quad 9 \end{array}$$



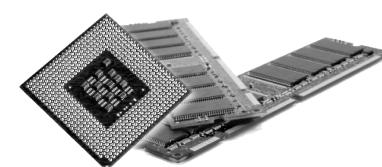
## Gleitkommazahlen

- 12,25(10) in Binäre Gleitkommazahl umwandeln:
  - Vorzeichen: 1 Bit (0: positiv, 1: negativ)
  - Länge des Exponenten: 5 Bit
  - Länge der Mantisse: 6 Bit
  - Normalisierung auf 1,...



## Gleitkommazahlen

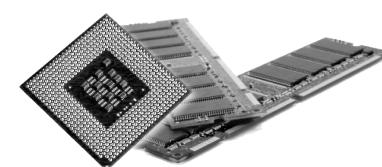
- 12,25<sub>(10)</sub> in Binäre Gleitkommazahl umwandeln:
  - Vorzeichen:
    - positiv -> 0
  - Umwandeln:
    - 12<sub>(10)</sub> = 1100<sub>(2)</sub>
    - 0,25<sub>(10)</sub> = 0,01<sub>(2)</sub>
  - Normalisieren:
    - 1100,01 = 1,10001 \* 2<sup>3</sup>



# Gleitkommazahlen

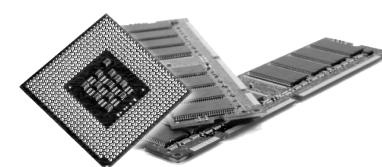
- 12,25<sub>(10)</sub> in Binäre Gleitkommazahl umwandeln:
  - Normalisieren:
    - $1100,01 = 1,10001 \cdot 2^3$
  - Mantisse:
    - 10001
  - Exponent:
    - $3_{(10)} = 11_{(2)}$

	VZ	Exponent					Mantisse					
12,25	0	0	0	0	1	1	1	0	0	0	1	0



## Gleitkommazahlen IEEE 754

- $x = s * m * b^e$
- Exponent:
  - fester Biaswert B wird addiert:  $E = e + B$
  - $B = 2^{r-1} - 1$
  - r = Anzahl der Stellen des Exponenten



## Schreibweise Zahlensysteme

- **5<sub>(16)</sub>**
- 5<sub>16</sub>
- [5]<sub>16</sub> Bsp.: Hexadezimal
- 5<sub>Hex</sub>
- 5<sub>Hexadezimal</sub>
- **5h**
- 0x5