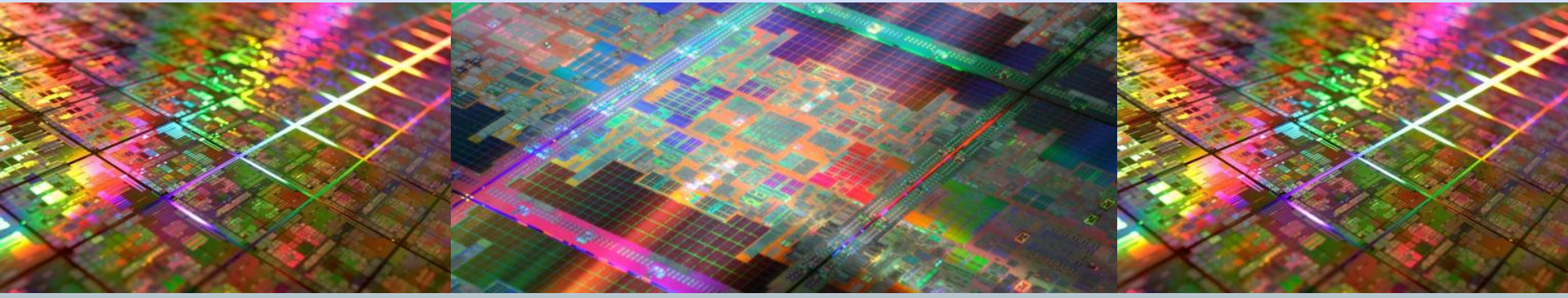
Grundlagen

Technische Grundlagen der Informatik für Wirtschaftsinformatik

Stefan Podlipnig

TU Wien



Lernziele

- Kennenlernen grundlegender Prinzipien der Rechnerorganisation
- Kennenlernen der Vorteile der binären Darstellung
- Verstehen der Darstellung von Zahlen in Rechnern und Durchführen von einfachen Berechnungen

ALLGEMEINER ÜBERBLICK

Was wird in dieser LV besprochen?

- Wie werden Zahlen in einem Rechner dargestellt?
- Wie werden Programme, die in einer höheren Programmiersprache geschrieben sind, in die Sprache der Hardware übersetzt?
- Was ist die Schnittstelle zwischen der Software und der Hardware?
- Was bestimmt die Leistung (*performance*) eines Programms?
- Welche Techniken können von den Rechnerarchitekten eingesetzt werden um die Leistung zu verbessern?
- Wie kommunizieren Teile eines Rechners miteinander?
- Wie kommunizieren Rechner untereinander?

Acht zentrale Ideen in der Rechnerarchitektur

- Design basierend auf dem Mooreschen Gesetz
- Abstraktion um Entwürfe zu vereinfachen
- Den häufig vorkommenden Fall optimieren
- Leistung durch parallele Verarbeitung
- Leistung durch Pipelining
- Leistung durch Vorhersage
- Speicherhierarchien
- Zuverlässigkeit durch Redundanz

KOMPONENTEN EINES RECHNERS

Komponenten eines Rechners

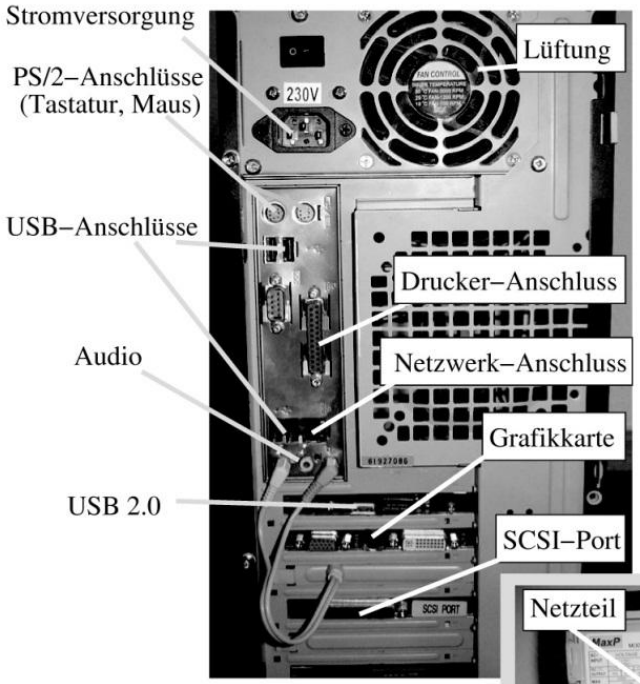
- Fünf klassische Komponenten
  - Rechenwerk (*Arithmetic Logic Unit*, *ALU*)
  - Steuer- oder Leitwerk (*Control Unit*, *CU*)

  } Prozessor (*processor*)

  - Hauptspeicher (*Memory*)
  - Eingabe (*Input*)
  - Ausgabe (*Output*)

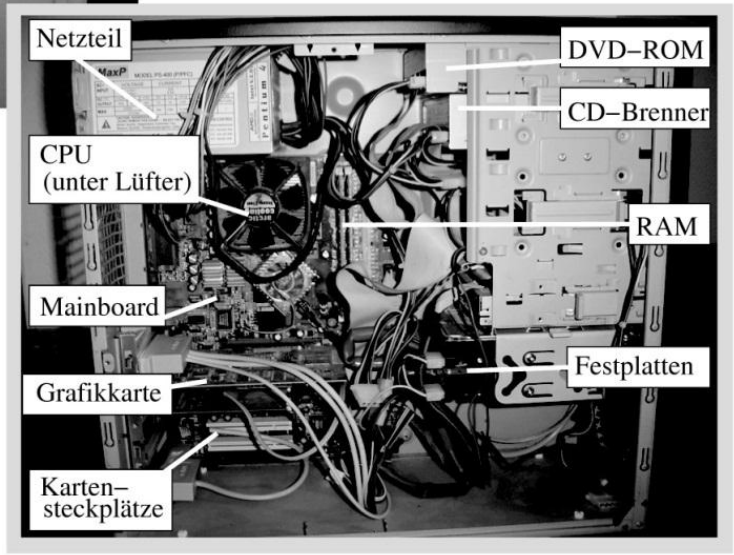
  } Eingabegeräte: Tastatur, Maus, Scanner, Mikrophon usw.
  Ausgabegeräte: Bildschirm, Beamer, Drucker, Plotter, Lautsprecher, usw.
  Ein- und Ausgabe gleichzeitig: Festplatten, CD/DVD, SSD, Netzwerkkarte, Touchscreen usw.
- Historischer Ursprung in den Arbeiten von John von Neumann (Mitte der 1940er)
  - Rechner besteht aus: Rechenwerk, Steuerwerk, Speicher, Ein- und Ausgabeeinheiten, Verbindungssystem (Bussystem)
  - Programme und Daten werden gemeinsam im Speicher abgelegt

Anatomie eines Rechners (historisch)



Rückseite eines PCs

Inneres eines PCs



Anatomie eines Rechners (modern)

- Prozessor (CPU, *Central Processing Unit*)
  - Rechenwerk
  - Leitwerk
  - Cache-Speicher
    - Kleiner schneller Speicher, der als Puffer für den Hauptspeicher dient

- Hauptspeicher (auch Arbeitsspeicher)
  - Beinhaltet Programme und Daten

- Motherboard (Mainboard)
  - Die wichtigsten Komponenten (Prozessor, Speicher, Schnittstellen) des Rechners sind darauf miteinander verschaltet

Abstraktionsschichten

Programmierung

Rechner speichert Daten und führt Befehle aus

Architektur

Struktur eines Rechners (Register, Rechenwerk, Steuerwerk, interne Bussysteme, Ein-/Ausgabeeinheit)

Schnittstelle zwischen Hard- und Software (Instruktionen, Registersatz und Adressierungsarten)

Digitale Logik

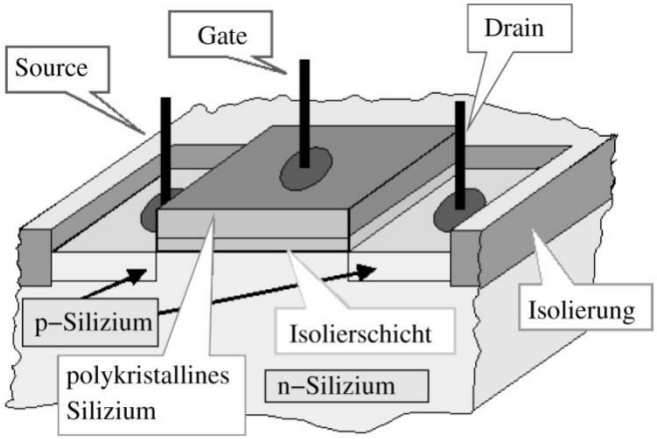
Schaltnetze und Schaltwerke zur Verknüpfung und Speicherung der Daten

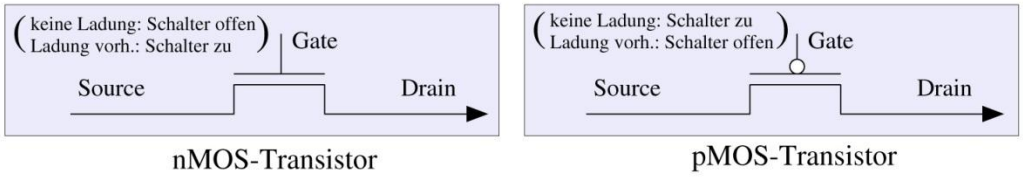
Elektronik

Elektronische Schaltkreise, die binäre Signale erzeugen, verknüpfen und speichern

Prozessoren und Transistoren

- Prozessor besteht aus Millionen bzw. Milliarden von Transistoren
- Transistor = Ein-/Ausschalter, der elektrisch gesteuert wird





Trends - Strukturgröße

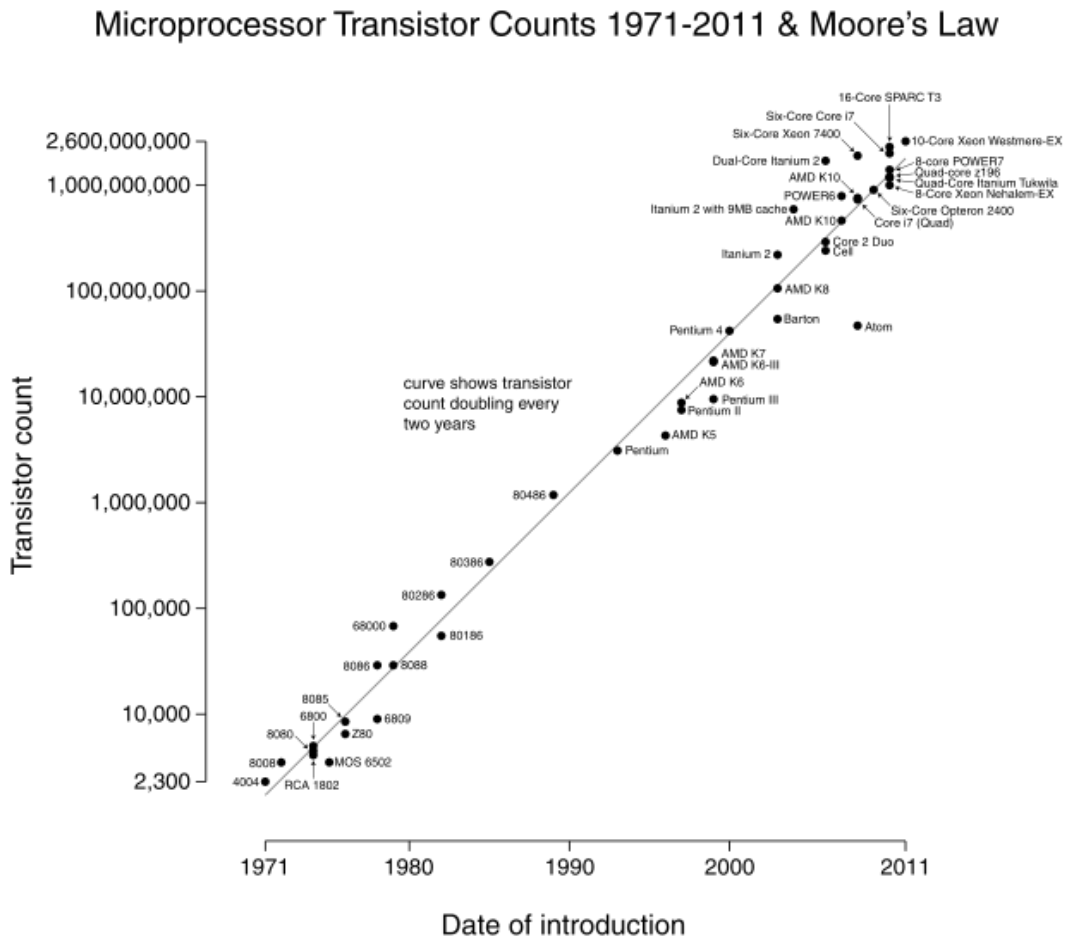
- Strukturgröße (*feature size*)
  - Kantenlänge eines Transistors (vereinfacht)
- Beispiele (Intel-Prozessoren)

Jahr	Prozessor	Transistoren	Strukturgröße	Fläche
1971	4004	2 300	10 µm	12 mm^2
1978	8086	29 000	3 µm	33 mm^2
1985	80386	275 000	1.5 µm	104 mm^2
1993	Pentium (P5)	3,1 Millionen	800 nm	294 mm^2
2000	Pentium 4	42 Millionen	180 nm	217 mm^2
2007	Core2 Duo	411 Millionen	45 nm	107 mm^2
2015	Core i7 (Skylake)	1,175 Milliarden	14 nm	122 mm^2

- Hinweis: 1 µm = 10^{-6} m, 1nm = 10^{-9} m

Trends – Mooresches Gesetz

- Das „Gesetz“ von Moore (1965) besagt, dass sich die Anzahl der Transistoren je Chip alle 2 Jahre (alternativ 18 Monate) verdoppelt



[http://commons.wikimedia.org/wiki/File:Transistor_Count_and_Moore%27s_Law_-_2011.svg , 20.02.2016]

Binäre Signale

- Transistoren sind Ein-/Ausschalter
  - Werden elektrisch gesteuert
  - Erzeugen binäre Signale
- Alphabet der Binärziffern = {0, 1}
  - **Binärzeichen** oder **Bits** (Symbole des binären Alphabets)
  - Bit von Binary digit = Binärziffer
- Gruppen von Binärziffern
  - Mit *n* Binärziffern können 2^n Kombinationen gebildet werden
    - Beispiel für n = 2: *00*, *01*, *10*, *11*
  - Für *x* Kombinationen benötig man zumindest $\lceil log_2\ x \rceil$ Bits

Formale Modellierung

Code

- Zuordnungsvorschrift zwischen zwei Alphabeten
  - Alphabete können (müssen aber nicht) verschieden sein
- Beispiel (Bitmuster - Kleinbuchstaben)
  - 00000 = a
  - 00001 = b
  - 00010 = c
  - …
- Binärcodierung
  - Die Codierung irgendeines Alphabets durch Folgen von Binärzeichen
- **Universeller Einsatz des binären Alphabets**
  - **Alle denkbaren endlichen Alphabete** lassen sich durch Folgen von Binärzeichen ausdrücken

Vorteile der binären Darstellung

- Geringe Störempfindlichkeit
  - Durch zweiwertige physikalische Größen, z. B.
    - logische 0: 0 - 2 Volt
    - logische 1: 3 - 5 Volt
  - Zwei Werte müssen eindeutig unterscheidbar sein
- Verlustlose Speicherung
  - Speicherung stetig veränderlicher Werte führt zu Genauigkeitsverlust (z.B. Audiokassette)
  - Binäre Werte lassen sich einfach und sicher speichern

Byte

- Byte
  - Entspricht 8 Bits (meistens)
- Beispiel für 2 Bytes
  - 10001010 10111010
- Maßeinheit für z.B.
  - Hauptspeichergrößen
  - Festplattengrößen
- Kürzere Schreibweise
  - Präfixe für eine große Anzahl von Bytes (Kilo, Mega usw.)
- **Achtung**
  - Unterschiedliche Verwendung der Präfixe in der Literatur und Praxis

Maßeinheiten – Dezimalpräfixe

- SI-Präfixe
  - Dezimalpräfixe für die Verwendung im internationalen Einheitensystem

Exp.	Langform	Präfix	Exp.	Langform	Präfix
10^{-3}	0,001	Milli	10^{3}	1.000	Kilo
10^{-6}	0,000001	Mikro	10^{6}	1.000.000	Mega
10^{-9}	0,000000001	Nano	10^{9}	1.000.000.000	Giga
10^{-12}	0,000000000001	Pico	10^{12}	1.000.000.000.000	Tera
10^{-15}	0,000000000000001	Femto	10^{15}	1.000.000.000.000.000	Peta
10^{-18}	0,000000000000000001	Atto	10^{18}	1.000.000.000.000.000.000	Exa
10^{-21}	0,000000000000000000001	Zepto	10^{21}	1.000.000.000.000.000.000.000	Zetta
10^{-24}	0,000000000000000000000001	Yokto	10^{24}	1.000.000.000.000.000.000.000.000	Yotta

Maßeinheiten – Binärpräfixe

- Besondere, an die SI-Präfixe angelehnte, explizite Binärpräfixe
  - Beispiele
    - Kibibyte = 2^{10} Bytes
    - Mebibyte = 2^{20} Bytes
    - Gibibyte = 2^{30} Bytes
- In dieser LV gelten daher folgende Notationen

Notation	Name	Potenz	Wert
kB	Kilobyte	10^3	1 000 Bytes
MB	Megabyte	10^6	1 000 000 Bytes
GB	Gigabyte	10^9	1 000 000 000 Bytes
TB	Terabyte	10^{12}	1 000 000 000 000 Bytes

Notation	Name	Potenz	Wert
KiB	Kibibyte	2^{10}	1 024 Bytes
MiB	Mebibyte	2^{20}	1 048 576 Bytes
GiB	Gibibyte	2^{30}	1 073 741 824 Bytes
TiB	Tebibyte	2^{40}	1 099 511 627 776 Bytes

Zweierpotenzen

- Grundlegende Zweierpotenzen

Zweierpotenz	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9
Wert der Potenz	1	2	4	8	16	32	64	128	256	512

- Weitere

Zweierpotenz	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}
Wert der Potenz	1024	2048	4096	8192	16384	32768	65536	131072	262144	524288

- Beispiele für Verwendung und Berechnung
  - Ein Speicher hat 2^{16} Bytes
    - 2^{16} Bytes = $2^6 \times 2^{10}$ Bytes = 64 Kibibytes
  - Eine SSD speichert 2^{34} Bytes
    - 2^{34} Bytes = $2^4 \times 2^{30}$ Bytes = 16 Gibibytes
  - Eine Festplatte speichert 4 Tebibytes
    - 4 Tebibytes = $2^2 \times$ Tebibytes = 2^{42} Bytes
    - oder 2^{32} Kibibytes oder 2^{22} Mebibytes oder 2^{12} Gibibytes

ZAHLENDARSTELLUNG – GRUNDLAGEN

Zahlensysteme - Dezimalsystem

- Dezimales Zahlensystem (Zehnersystem)
  - Stellenwertsystem
- Darstellung mit
  - Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - An bestimmten Stellen (Einer, Zehner, Hunderter, Tausender, ….)
  - Basis 10

Ziffer	**1**	**9**	**2**	**3**
Stelle	Tausender	Hunderter	Zehner	Einer
Wert der Stelle	10^3	10^2	10^1	10^0

  - Wert der Zahl: $1x10^3 + 9x10^2 + 2x10^1 + 3x10^0 = 1923$

Zahlensysteme – allgemeines Stellenwertsystem

- Basis b
  - Zahl = $(\ldots a_3 a_2 a_1 a_0)_b$
  - Wert der Zahl = $\ldots a_3 b^3 + a_2 b^2 + a_1 b^1 + a_0 b^0$
- Beispiel
  - Zahl = $(1923)_{10}$
  - Wert = $1x10^3 + 9x10^2 + 2x10^1 + 3x10^0 = 1923$
- Basis
  - Anzahl der verwendeten Ziffern, $b \geq 2$
  - Ziffern: Zeichen mit dezimalem Wert 0, 1, … ,b – 1
- Nachkommastellen
  - $= (\ldots a_3 a_2 a_1 a_0 . a_{-1} a_{-2} \ldots)_b = \ldots a_3 b^3 + a_2 b^2 + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2}$

Beispiele

- Beispiel: $(123.12)_{10}$ ist ident mit der gewohnten Notation 123.12 für das geläufige Zehnersystem

$$
\begin{aligned}
(a_2 a_1 a_0 \,.\, a_{-1} a_{-2})_b &= a_2 \cdot b^2 + a_1 \cdot b^1 + a_0 \cdot b^0 + a_{-1} \cdot b^{-1} + a_{-2} \cdot b^{-2} \\
(123.12)_{10} &= 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 1 \cdot 10^{-1} + 2 \cdot 10^{-2} \\
&= 100 + 20 + 3 + 0.1 + 0.02 \\
&= (123.12)_{10}
\end{aligned}
$$

- Weiteres Beispiel (Basis = 6)

$$
\begin{aligned}
(520.3)_6 &= 5 \cdot 6^2 + 2 \cdot 6^1 + 0 \cdot 6^0 + 3 \cdot 6^{-1} \\
&= 180 + 12 + 0 + 0.5 \\
&= (192.5)_{10}
\end{aligned}
$$

Zahlensysteme – Binärsystem

- Basis 2
- Verwendete Ziffern: 0 und 1
- Beispiel (Tabelle rechts)
  - 4 Bits zur Verfügung
  - 16 (2^4) Zahlen darstellbar
- Beispiele für Umrechnung
  - $(0011)_2 = 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 2 + 1 = (3)_{10}$
  - $(1010)_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 8 + 2 = (10)_{10}$

binär	dezimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Zahlensysteme – Oktalsystem, Hexadezimalsystem

- Oktalsystem
  - Basis 8
  - Verwendete Ziffern: 0, 1, 2, 3, 4, 5, 6, 7
  - Beispiele für Anwendungen
    - Dateizugriffsrechte unter Unix
    - Transpondercode bei Flugzeugen
- Hexadezimalsystem
  - Basis 16
  - Verwendete Ziffern: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - Einfachere Notation von Binärzahlen
    - 1 Byte wird mit 2 Hex-Ziffern dargestellt
    - Wird bei Hex-Editoren verwendet

Überblick und Beispiele

binär	oktal	hexadezimal	dezimal
0000	00	0	0
0001	01	1	1
0010	02	2	2
0011	03	3	3
0100	04	4	4
0101	05	5	5
0110	06	6	6
0111	07	7	7
1000	10	8	8
1001	11	9	9
1010	12	A	10
1011	13	B	11
1100	14	C	12
1101	15	D	13
1110	16	E	14
1111	17	F	15

binär	oktal	hexadezimal	dezimal
10000	20	10	16
10001	21	11	17
10010	22	12	18
10011	23	13	19
10100	24	14	20
11110	36	1E	30
101000	50	28	40
110010	62	32	50
1100100	144	64	100

binär	oktal	hexadezimal	dezimal
11111	37	1F	31
100000	40	20	32
11111111	377	FF	255
100000000	400	100	256
111111111111	7777	FFF	4095
1000000000000	10000	1000	4096

Addition (Beispiel Binärsystem)

- Wie im Dezimalsystem
  - Übertrag muss aber korrekt bestimmt werden
- Regeln
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 10
    - Übertrag (wie z.B. bei 5 + 7 im Dezimalsystem)
    - Die Eins muss eine Stelle weiter links berücksichtigt werden
      - Dabei gilt dann auch: 1 + 1 + 1 = 3 = 11 im Binärsystem (1 anschreiben, 1 Übertrag)
  - Beispiele

	0	1	1	1
+	0	0	1	1
	1	1	1	
=	1	0	1	0

	0	1	0	1	0
+	0	1	1	0	0
	1				
=	1	0	1	1	0

Addition (weitere Beispiele)

- Oktalsystem
  - $(3)_8 + (4)_8 = (7)_8$
  - $(3)_8 + (5)_8 = (10)_8$
  - Größeres Beispiel rechts

	0	4	7	7
+	0	3	7	6
	1	1	1	
=	1	0	7	5

- Hexadezimalsystem
  - $(3)_{16} + (9)_{16} = (C)_{16}$
  - $(8)_{16} + (9)_{16} = (11)_{16}$
  - Größere Beispiele

	5	6	7	0
+	5	6	7	0
=	A	C	E	0

	0	8	9	A
+	0	7	C	F
	1	1	1	
=	1	0	6	9

Subtraktion im Binärsystem

- Regeln
  - 0 – 0 = 0
  - 0 – 1 = –1 (erfordert Sonderbehandlung)
  - 1 – 0 = 1
  - 1 – 1 = 0
- 0 – 1 als Spezialfall
  - Aus 0 – 1 wird 10 – 1
  - Ergebnis ist 1
  - Die vor die Null gedachte Eins muss als Übertrag an die nächste Stelle geschrieben und von dieser zusätzlich abgezogen werden
- Beispiele

	0	1	1	1
–	0	0	1	1
=	0	1	0	0

	1	1	0	1	0
–	0	1	1	0	0
	1	1			
=	0	1	1	1	0

Schnelle Konvertierungen (1)

- Oktalsystem
  - Konvertierung in (und vom) Binärsystem recht einfach
    - Eine Oktalziffer entspricht drei Bits (Basis $8 = 2^3$)
  - Beispiele
    - $(110111000010)_2 = 110\ 111\ 000\ 010 = (6702)_8$
    - $(527)_8 = (101\ 010\ 111)_2$
- Hexadezimalsystem
  - Konvertierung in (und vom) Binärsystem recht einfach
    - Eine Hexadezimalziffer entspricht vier Bits (Basis $16 = 2^4$)
  - Beispiele
    - $(110111000010)_2 = 1101\ 1100\ 0010 = (DC2)_{16}$
    - $(BAD1)_{16} = 1011\ 1010\ 1101\ 0001 = (1011101011010001)_2$

Schnelle Konvertierungen (2)

- Hinweise
  - Beim Konvertieren ganzer Zahlen immer rechts beginnen
  - Führende Nullen beachten bzw. weglassen
- Beispiele
  - $(10101)_2$ = 010 101 =$(25)_8$
  - $(15)_{16}$ = 0001 0101 =$(10101)_2$

Konvertierung zwischen beliebigen Zahlensystemen

- Man unterscheidet zwischen
  - Konvertierung von ganzen Zahlen und
  - Konvertierung von Zahlen mit Nachkommastellen
- Man kann jede reelle Zahl als Summe einer ganzen Zahl und einer Zahl, deren Vorkommaanteil gleich 0 ist, darstellen
  - Getrennte Umrechnung möglich
  - Nicht benötigter Anteil wird als 0 angenommen

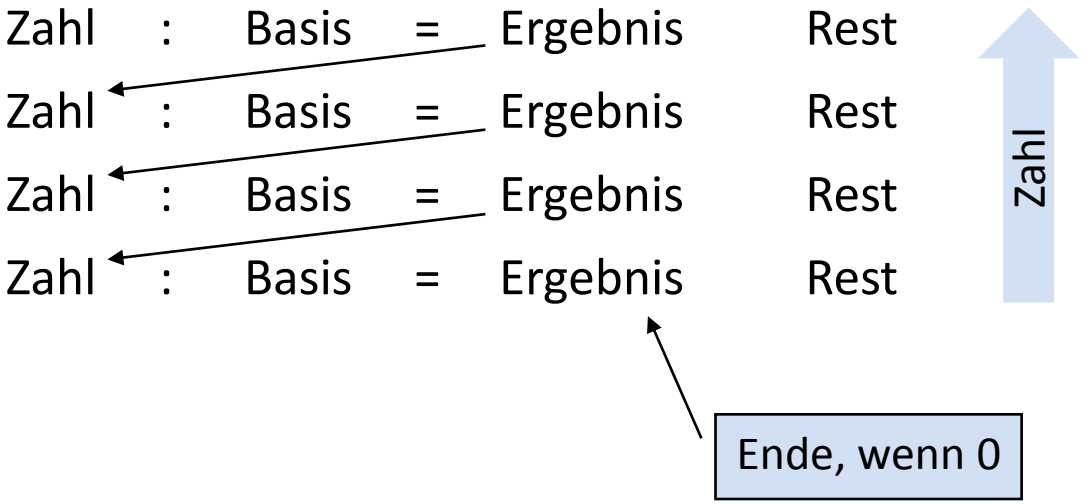
Konvertierung (dezimal zu binär)

- Beispiel: $(6)_{10}$ in das binäre Zahlensystem umrechnen
- Das Stellenwertsystem der binären Zahlendarstellung lautet

$$z = (\ldots a_3 a_2 a_1 a_0)_2$$
$$= \ldots + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0$$
$$= (((\ldots + a_3) \cdot 2 + a_2) \cdot 2 + a_1) \cdot 2 + a_0$$

- $z - a_0$ ist durch 2 teilbar, d.h. dass a_0 als Rest bei der Division von z durch 2 auftritt
  - Für $(6)_{10}$: $a_0 = 0$
- Sei $z_1 = (z - a_0)/2$, a_1 ist der Rest bei der Division von z_1 durch 2
  - Für $(6)_{10}$: $z_1 = (6 - 0)/2 = 3$, $a_1 = 1$ (Rest von 3:2)
- Auf ähnliche Weise: $a_2 = 1$
- $(6)_{10} = (110)_2$
- Diese Überlegungen kann man verallgemeinern

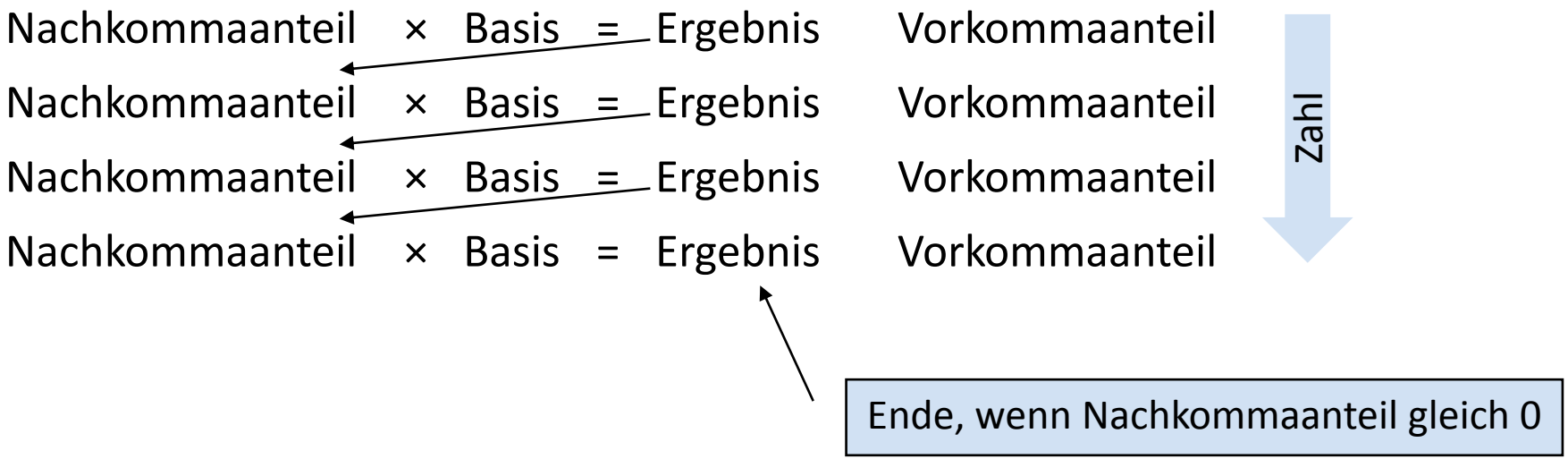
Schema für die Konvertierung ganzer Zahlen



- Beispiel: $(105)_{10} = (1101001)_2$

105	:	2	=	52	1
52	:	2	=	26	0
26	:	2	=	13	0
13	:	2	=	6	1
6	:	2	=	3	0
3	:	2	=	1	1
1	:	2	=	0	1

Schema für die Konvertierung des Nachkommaanteils



- Beispiel: $(0.8125)_{10} = (0.1101)_2$

0.8125	×	2	=	1.625	1
0.625	×	2	=	1.25	1
0.25	×	2	=	0.5	0
0.5	×	2	=	1.0	1

- Hinweis: Sehr oft wird nur eine bestimmte Anzahl von Nachkommastellen berechnet (Ungenauigkeit!)

Konvertierung in weitere Zahlensysteme (Beispiele)

- Berechnung: $(895)_{10} = (37F)_{16}$

895	:	16	=	55	F
55	:	16	=	3	7
3	:	16	=	0	3

- Berechnung: $(0.6875)_{10} = (0.54)_{8}$

0.6875	×	8	=	5.5	5
0.5	×	8	=	4.0	4

Direkte Konvertierung (Nach dem Komma mit den Blöcken links beginnen):
$(0.54)_8 = 0.101\ 100 = (0.1011)_2$

- Berechnung: $(36.6875)_{10} = (100100.1011)_2$

36	:	2	=	18	0
18	:	2	=	9	0
9	:	2	=	4	1
4	:	2	=	2	0
2	:	2	=	1	0
1	:	2	=	0	1

0.6875	×	2	=	1.375	1
0.375	×	2	=	0.75	0
0.75	×	2	=	1.5	1
0.5	×	2	=	1.0	1

Konvertierung – Spezialfälle

- Endliche Zifferndarstellung geht **für manche Zahlen** bei der Konvertierung in ein anderes Zahlenformat verloren
- Beispiel: $(0.1)_{10} = (0.00011001100\ldots)_2$

0.1	×	2	=	0.2	0
0.2	×	2	=	0.4	0
0.4	×	2	=	0.8	0
0.8	×	2	=	1.6	1
0.6	×	2	=	1.2	1
0.2	×	2	=	0.4	0
0.4	×	2	=	…….	

Konvertierung – Verallgemeinerung

- Ursprüngliche Zahl mit Basis b unterschiedlich von 10
  - Rechengang im Zahlensystem mit der Basis b
- Beispiel: $(34)_5 = (10011)_2$

34	:	2	=	14	1
14	:	2	=	4	1
4	:	2	=	2	0
2	:	2	=	1	0
1	:	2	=	0	1

- Beispiel: $(135)_7 = (1023)_4$

135	:	4	=	24	3
24	:	4	=	4	2
4	:	4	=	1	0
1	:	4	=	0	1

Zahlennotation in dieser Vorlesung

- Verschiedene Notationen in dieser Vorlesung
- Basis wird nur angegeben, wenn diese aus dem Kontext nicht klar hervorgeht
- Beispiele

Notation	Zahlensystem	Dezimal	Hinweis
$(42)_{16}$	Hexadezimal	66	
42	Dezimal	42	
1001001_2	Binär	73	
0x42	Hexadezimal	66	Wird in dieser Vorlesung bei Speicheradressen und MIPS-Datenwörtern verwendet
42h	Hexadezimal	66	
56A	„Intuitiv forderndes“ Hexadezimal	1386	

ZAHLENDARSTELLUNG - GANZE ZAHLEN

Darstellung ganzer Zahlen im Rechner

- Interne Repräsentation als Folge von Bytes
- Anzahl der Bytes definiert den darstellbaren Zahlenbereich
  - Entspricht in der Regel einer Zweierpotenz 2^n
- Beispiele
  - 1 Byte = $[0 - 2^8 - 1] = [0 - 255]$
  - 2 Byte = $[0 - 2^{16} - 1] = [0 - 65535]$
  - 4 Byte = $[0 - 2^{32} - 1] = [0 - 4294967295]$

Bitwertigkeit

- Bitwertigkeit
  - Stellenwert eines einzelnen Bits, den es durch seine Position innerhalb einer Binärzahl hat
- MSB (*most significant bit*)
  - Das höchstwertige Bit
  - Steht an der Stelle mit dem höchsten Stellenwert
- LSB (*least significant bit*)
  - Das niedrigstwertige Bit
  - Besitzt den niedrigsten Stellenwert 1
- Beispiel
  - **1**0011000
  - MSB = **1**, LSB = **0**

Bytereihenfolge

- Reihenfolge der einzelnen Bytes im Speicher
  - **Big-Endian**-Architekturen
    - Das höchstwertige Byte wird zuerst im Speicher abgelegt
    - MIPS ist ein Beispiel dafür (siehe Foliensatz über Befehlssatz)
  - **Little-Endian**-Architekturen
    - Das Byte mit der niedrigsten Wertigkeit (Bit 0 bis Bit 7) wird zuerst im Speicher abgelegt
- Beispiel (schematisch)
  - 439041101 = **00011010 00101011 00111100 01001101**
  - Big-Endian

Adresse	0	1	2	3	4	5
Inhalt	00011010	00101011	00111100	01001101	.....	.....

  - Little-Endian

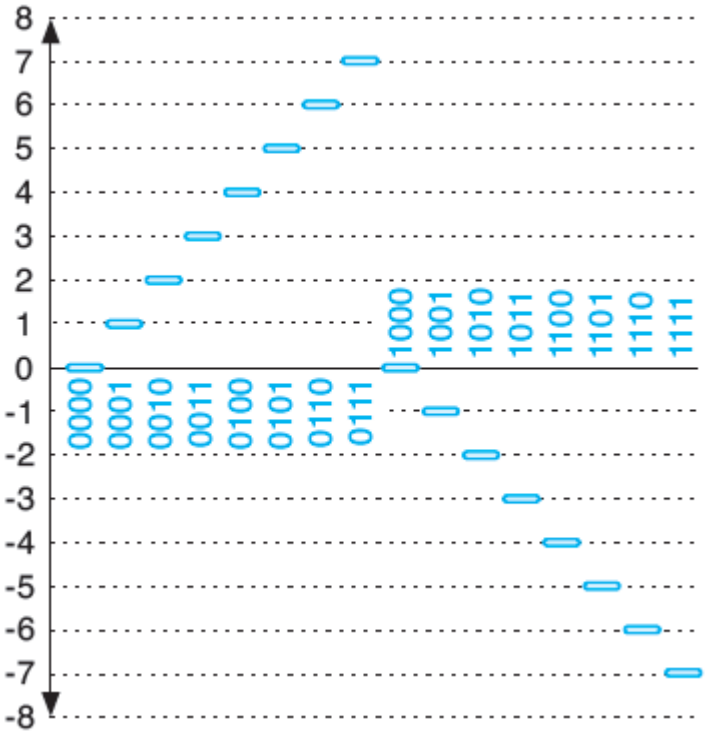
Adresse	0	1	2	3	4	5
Inhalt	01001101	00111100	00101011	00011010	.....	.....

Repräsentation von negativen Zahlen

- Bisher nur positive Zahlen betrachtet
  - Reicht bei bestimmten Programmen aus
  - Programmiersprachen wie C bieten Datentypen an, die nur positive Bereiche abdecken (unsigned)
- Darstellungsmöglichkeiten für negative Zahlen
  - Vorzeichenbitdarstellung
  - Einerkomplement
  - Zweierkomplement

Vorzeichenbitdarstellung

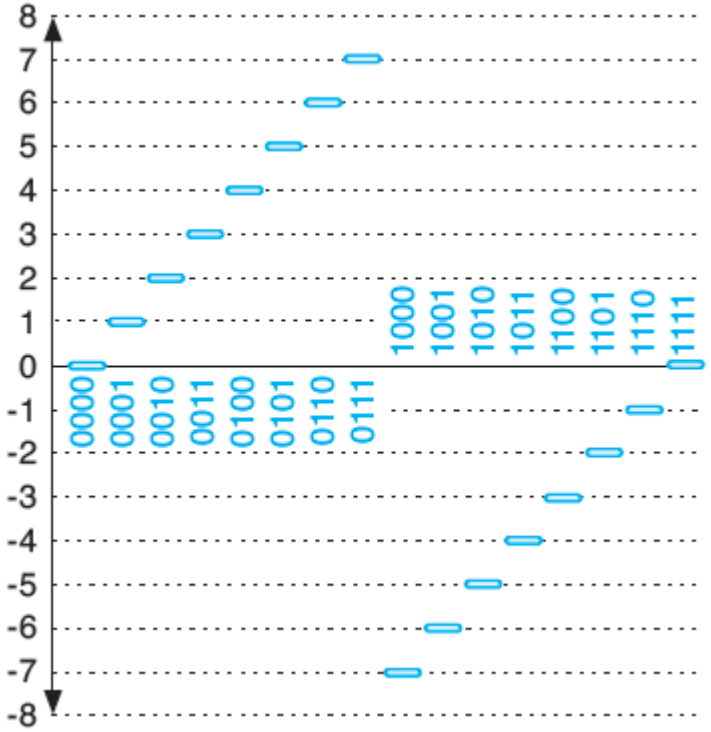
- Explizite Darstellung des Vorzeichens durch ein festgelegtes Bit
  - Z.B. 0 für positive und 1 für negative Zahlen
- Vierstellige Binärzahlen (Beispiel rechts)
- Probleme
  - Zwei Darstellungen für 0
  - Binärarithmetik funktioniert nicht mehr
  - Beispiel



	0	1	0	1	(5)
+	1	1	1	0	(-6)
= 1	0	0	1	1	(-3)

Einerkomplement (1)

- Negative Zahlen
  - Durch Invertieren aller Bits
- Beispiel
  - 7 = 0111 daher ist –7 = 1000
- Vierstellige Binärzahlen (Beispiel rechts)
- Noch immer Problem bei 0
- Addition (Schritte)
  - Ausführen der gewöhnlichen Binäraddition
  - Aufaddieren des Übertrags
  - Streichen verbleibender Überträge
  - Beispiele

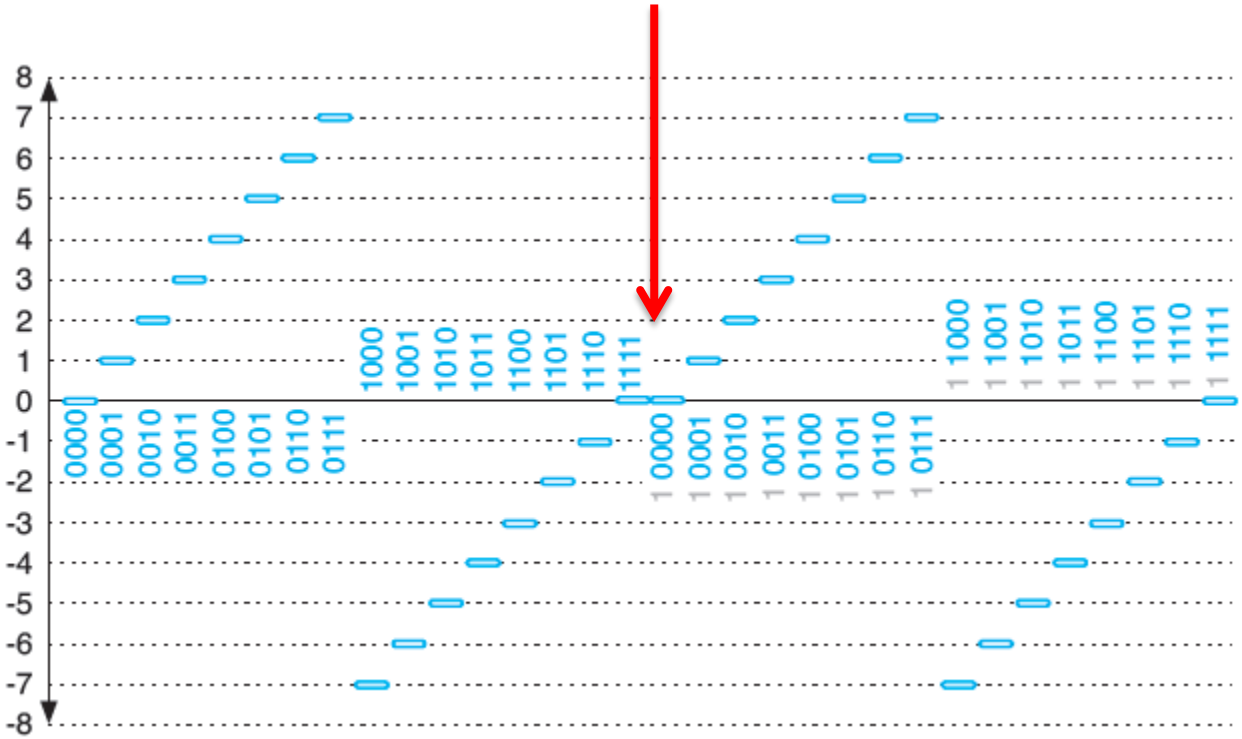


	0	1	0	1	(5)
+	1	0	0	1	(-6)
=	1	1	1	0	(-1)

	0	1	0	1	(5)
+	1	1	0	0	(-3)
=	1	0	0	0	1 (-14)
+	0	0	0	1	(1)
=	~~1~~	0	0	1	0 (2)

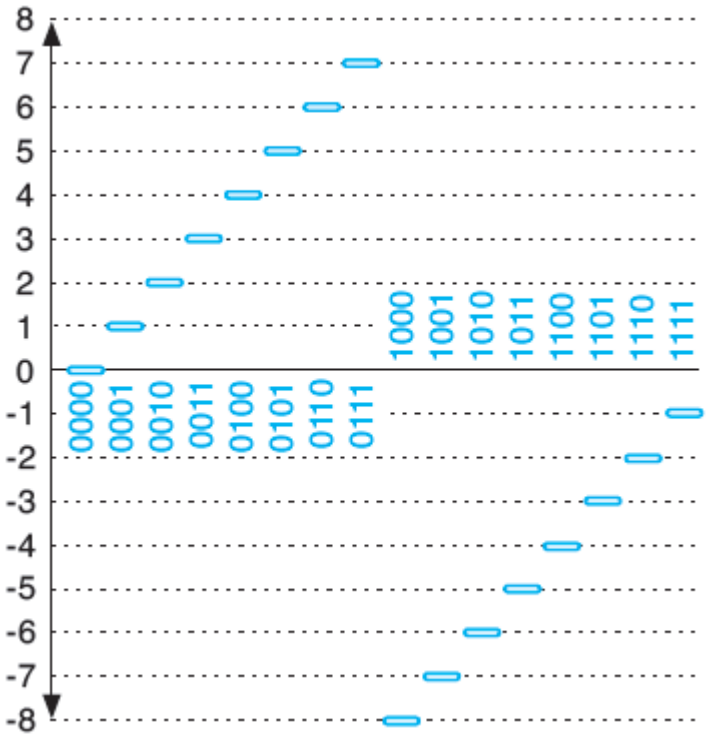
Einerkomplement (2)

- Übertragsregel
  - Doppeldarstellung der Null
  - Ergebnis ist um 1 zu klein



Zweierkomplement (1)

- Negative Zahl
  - Bilden des Einerkomplements
  - Addition von 1
- Beispiel
  - 4 = 0100 daher ist –4 = 1100 (1011 + 0001)
- Vierstellige Binärzahlen (Beispiel rechts)
- Addition
  - Wie bei positiven Zahlen
  - Streichen verbleibender Überträge



	0	1	0	1	(5)
+	1	0	1	0	(-6)
=	1	1	1	1	(-1)

	0	1	0	1	(5)
+	1	1	0	1	(-3)
=	~~1~~	0	0	1	0 (2)

Zweierkomplement (2)

- In der Praxis sehr oft eingesetzt
- Eigenschaften
  - Allgemein geht der Bereich bei n Bits von -2^{n-1} bis $+2^{n-1} - 1$ (Verlust der Symmetrieeigenschaft)
  - Nicht negative Zahlen haben die gleiche vorzeichenlose und vorzeichenbehaftete Darstellung
    - Gilt auch für Einerkomplement und Vorzeichenbitdarstellung
  - Einige spezielle Zahlen
    - 0000 0000 … 0000 = 0
    - 1111 1111 … 1111 = –1
  - Kleinste darstellbare Zahl: 1000 0000 … 0000
  - Größte darstellbare Zahl: 0111 1111 … 1111

ZAHLENDARSTELLUNG – REELLE ZAHLEN

Darstellung

- Rationale oder reelle Zahlen
  - Wie stellt man diese Zahlen mit einer fixen Anzahl an Bits dar?
- Festkommazahlen
  - Komma an fixer Position
  - Jede Kommazahl z wird durch Skalierung auf eine ganze Zahl z' abgebildet
  - Rechner arbeitet nur auf ganzer Zahl
- Gleitkommazahlen
  - Darstellung der Kommazahl durch Mantisse und Exponent

Festkommazahl

- Zahl zur Basis b mit einer festen Zahl von k Nachkommastellen

$$z = (z_{n-k-1}\, z_{n-k-2} \ldots z_1\, z_0 \,.\, z_{-1}\, z_{-2} \ldots z_{-k+1}\, z_{-k})_b$$

$$= z_{n-k-1} \cdot b^{n-k-1} + z_{n-k-2} \cdot b^{n-k-2} + \ldots + z_1 \cdot b^1 + z_0 \cdot b^0 + z_{-1} \cdot b^{-1} + z_{-2} \cdot b^{-2} + \ldots + z_{-k+1} \cdot b^{-k+1} + z_{-k} \cdot b^{-k} = \sum_{i=-k}^{n-k-1} z_i \cdot b^i$$

  - Die Ziffern $z_{n-k-1}\, z_{n-k-2} \ldots z_1\, z_0$ entsprechen dem **ganzzahligen Teil**
  - Die Ziffern $z_{-1}\, z_{-2} \ldots z_{-k}$ entsprechen dem **gebrochenen Teil**
  - Die feste Kommaposition k kennt nur der Anwender (Programm)
  - Der Rechner arbeitet mit skalierten ganzen Binärzahlen $z' = z \cdot 2^k$
- Beispiel: 8-Bit Binärzahl z´ = 01101110
  - Für k = 3 gilt: $z = 01101.110_2 = 2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-2} = 13{,}75_{10}$
- Problem
  - Fixiertes k

Gleitkommazahl (1)

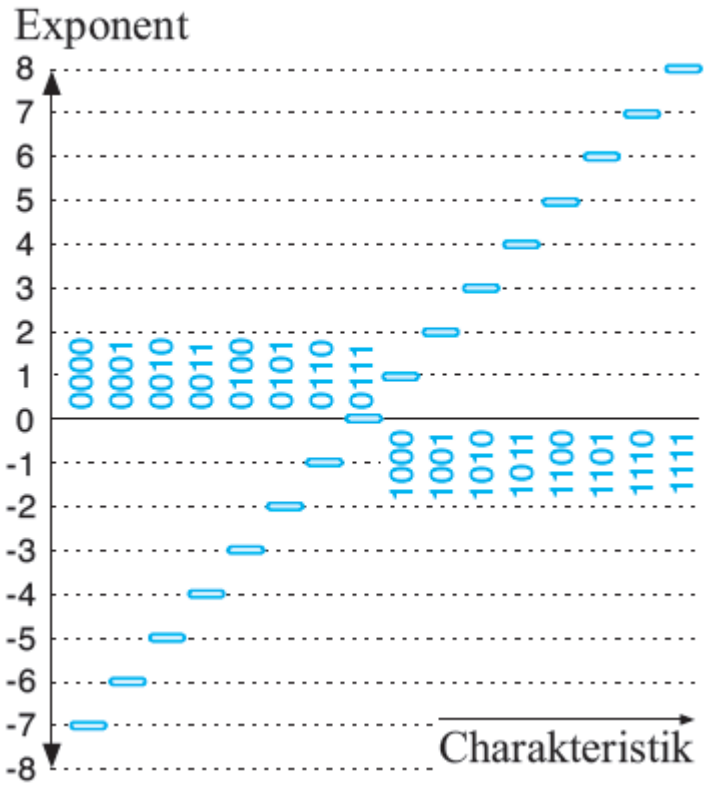
- In vielen Anwendungen wird eine große Zahlendynamik benötigt
  - Sehr kleine und sehr große Zahlen sollen einheitlich dargestellt werden
- Allgemeine Gleitkommazahl zur Basis r („radix“)
  - $\mathbf{x = a \times r^e}$
  - Argument oder Mantisse (*fraction*) a
  - Exponent (*exponent*) oder Charakteristik e
- Normalisierte Gleitkommazahl zur Basis r
  - Für a gilt: $1 \leq a < r$
- Beispiele (für r = 10)
  - 0,0000002345 entspricht $2{,}345 \times 10^{-7}$
  - 1024500000,0 entspricht $1{,}0245 \times 10^{9}$

Gleitkommazahl (2)

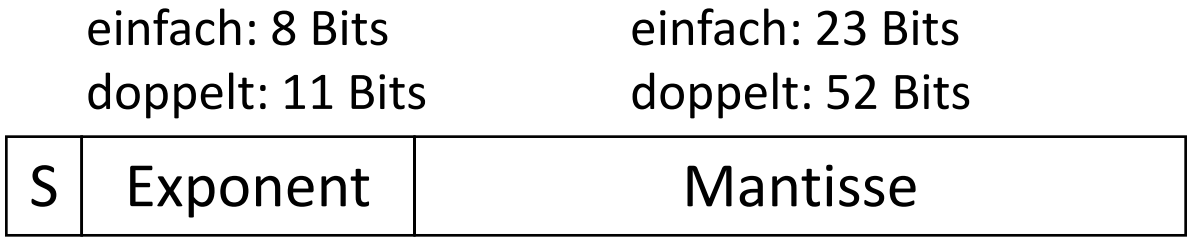
- Eine binäre Gleitkommazahl x ist definiert durch $x = a \times 2^e$
  - Mit m-stelliger Mantisse a und p-stelligem Exponenten e
- Normalisierte binäre Gleitkommazahl $x \neq 0$
  - Für a gilt: $1 \leq a < 2$
- Häufig Darstellung des Exponenten mit Bias b
  - $x = a \times 2^{e-b}$
  - Wahl von $b = 2^{p-1} - 1$ bewirkt Transformation des Bereiches für den Exponenten e von $0 \ldots 2^p - 1$ in $-(2^{p-1} - 1) \ldots 2^{p-1}$
    $\Rightarrow$ einfache Kodierung positiver und negativer Exponenten
- Früher unterschiedliches Gleitkommaformat in jedem Prozessor
  - Heute überwiegend Verwendung des IEEE 754 Standards
- Zwei Formate bei IEEE 754
  - Einfache Genauigkeit (*single precision*)
  - Doppelte Genauigkeit (*double precision*)

Darstellung des Exponenten (Charakteristik)

- Darstellung mit Bias (oder Exzess-Darstellung)
  - Verschieben um einen konstanten Wert
- 4-Bit Exponenten (Beispiel rechts)
  - Bias = 7
- Eigenschaften
  - Bitmuster kann als vorzeichenlose Binärzahl behandelt werden
  - Lexikalischer Größenvergleich möglich
  - Exponent kann durch Subtraktion der Konstante zurückerhalten werden



IEEE 754



$$x = (-1)^S \times (1 + \text{Mantisse}) \times 2^{(\text{Exponent} - \text{Bias})}$$

- S: Vorzeichen(0 $\Rightarrow$ nicht negativ, 1 $\Rightarrow$ negativ)
- Normalisiert: $1.0 \leq |\text{Signifikand}| < 2.0$
  - Signifikand = 1 + Mantisse
  - Das Bit der führenden 1 ist bereits inbegriffen
  - 1 vor dem Komma wird nicht kodiert $\Rightarrow$ erhöhte Präzision
- Exponent
  - Darstellung mit Bias (Exzesscodierung)
  - Repräsentation: Exponent + Bias
    - Einfach: Bias = 127
    - Doppelt: Bias = 1023

Einfache Genauigkeit

- Exponenten 00000000 und 11111111 sind reserviert
- Kleinster Betrag (normalisiert)
  - Exponent: 00000001
  - Eigentlicher Exponent = 1 – 127 = –126
  - Mantisse: 000…00 → Signifikand 1.0
  - $1.0 \times 2^{-126} \approx 1.2 \times 10^{-38}$
- Größter Betrag (normalisiert)
  - Exponent: 11111110
  - Eigentlicher Exponent = 254 – 127 = +127
  - Mantisse: 111…11 ≈ Signifikand 2.0
  - $2.0 \times 2^{+127} \approx 3.4 \times 10^{+38}$

Doppelte Genauigkeit

- Exponenten 0000…00 und 1111…11 sind reserviert
- Kleinster Betrag (normalisiert)
  - Exponent: 00000000001
  - Eigentlicher Exponent = 1 – 1023 = –1022
  - Mantisse: 000…00 → Signifikand 1.0
  - $1.0 \times 2^{-1022} \approx 2.2 \times 10^{-308}$
- Größter Betrag (normalisiert)
  - Exponent: 11111111110
  - Eigentlicher Exponent = 2046 – 1023 = +1023
  - Mantisse: 111…11 ≈ Signifikand 2.0
  - $2.0 \times 2^{+1023} \approx 1.8 \times 10^{+308}$

Beispiel

- Darstellung von –0.75
  - $-0.75 = (-1)^1 \times 1.1_2 \times 2^{-1}$
  - S = 1
  - Mantisse = $1000\ldots00_2$ ($0.11 \times 2^0 = 1.1 \times 2^{-1}$)
  - Exponent = –1 + Bias
    - Einfach: –1 + 127 = 126 = 01111110_2
    - Doppelt: –1 + 1023 = 1022 = 01111111110_2
- Einfach: 1 01111110 1000…00
- Doppelt: 1 01111111110 1000…00

Beispiel

- Welche Zahl wird durch folgende einfach genaue Zahl dargestellt?
- 1 10000001 01000…00
  - S = 1
  - Mantisse = $01000\ldots00_2$
  - Exponent = $10000001_2 = 129$
  - $x = (-1)^1 \times (1 + 0.25) \times 2^{(129 - 127)}$
    $= (-1) \times 1.25 \times 2^2$
    $= -5.0$
- Ergebnis: –5.0

Spezialfälle

- $e = e_{min} = (00..00)_2 = 0$ und $e = e_{max} = (11..11)_2$ werden zur Kodierung besonderer Zahlen verwendet

  x = +0 („**positive Zero**“): e = 0, m = 0, s = 0

  x = –0 („**negative Zero**“): e = 0, m = 0, s = 1

  $x = +\infty$ („**positive Infinity**“): $e = e_{max}$, m = 0, s = 0

  $x = -\infty$ („**negative Infinity**“): $e = e_{max}$, m = 0, s = 1

  x = NaN („**Not a Number**“): $e = e_{max}$, $m \neq 0$, s beliebig

  $x = (-1)^s \times 0.f \times 2^{1-b}$ („**Denormalized Number**“): e = 0, $m \neq 0$

- Denormalisierte Gleitkommazahlen
  - Ermöglichen die Darstellung sehr kleiner Werte im Bereich 2^{1-b-m} ... 2^{1-b}

Ausnahmesituation

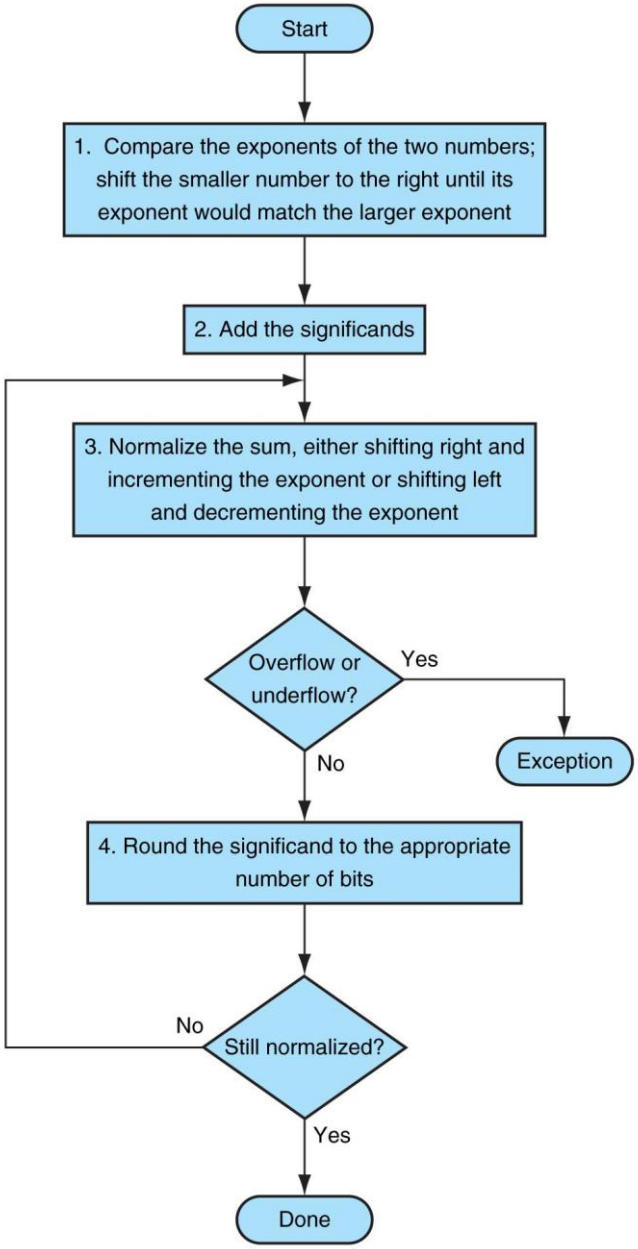
- Wenn für x gilt: $e \geq e_{max}$
  - Generierung von $+\infty$, falls $x > 0$
  - Generierung von $-\infty$, falls $x < 0$
- Einige Rechenregeln für ∞:
  - $\infty + x = \infty$ (falls $x \neq -\infty$), $\infty - x = \infty$ (falls $x \neq \infty$),
  - $\pm x / 0 = \pm\infty$ (falls $x \neq 0$), $\infty \cdot x = \pm\infty$ (falls $x \neq 0$)
- Unbestimmtes Ergebnis (Beispiele)
  - $\infty \cdot 0 = NaN$, $0 / 0 = NaN$, $\infty - \infty = NaN$,
  - Ferner gilt für alle Operationen: $f(x, NaN) = NaN$
- Wenn für x gilt: e = 0
  - Generierung von x = 0 („flushing to zero“)
  - Generierung einer denormalisierten Darstellung von x

IEEE 754 – Weitere Informationen

- Ist nicht nur eine Definition des Formats
- Weitere Festlegung
  - Algorithmen für das Runden
  - Arithmetische Operationen
  - Umwandlung zwischen Formaten
  - Ausnahmebehandlung
  - …

[http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=30711, 23.02.2016]

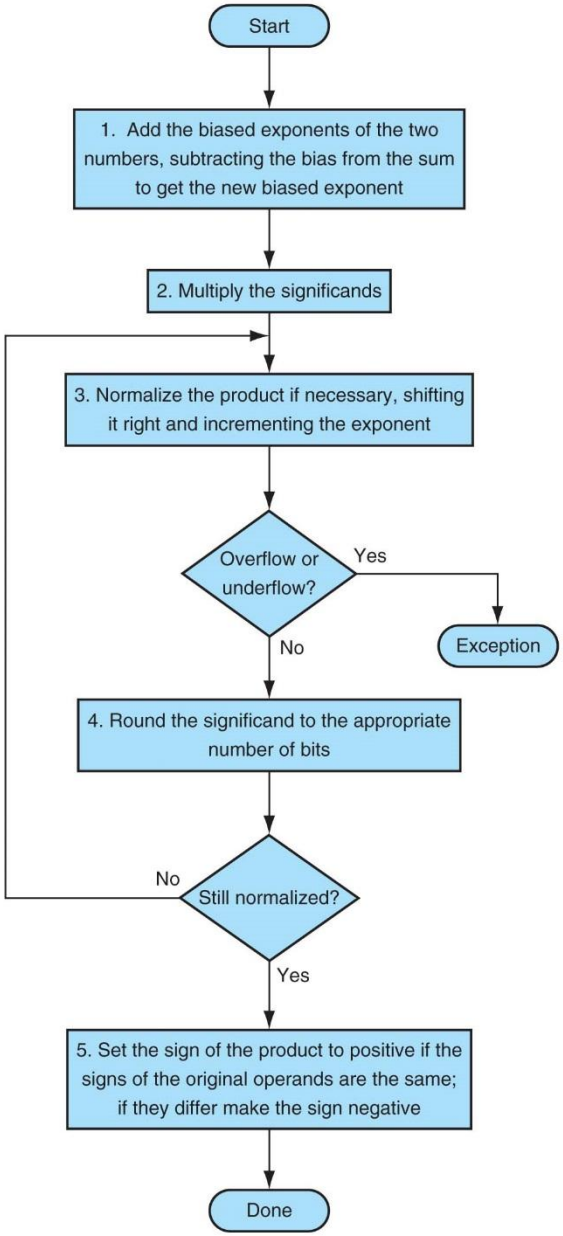
Gleitkomma-Addition



Gleitkomma-Addition (Beispiel)

- Binärzahlen mit 4 Bits
  - $1.000_2 \times 2^{-1} + -1.110_2 \times 2^{-2}$ (0.5 + –0.4375)
- Anpassen der Exponenten
  - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1}$
- Addieren der Signifikanden
  - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1} = 0.001_2 \times 2^{-1}$
- Normalisieren und Überprüfen
  - $1.000_2 \times 2^{-4}$, kein Überlauf
- Runden (nicht notwendig)
  - $1.000_2 \times 2^{-4} = 0.0625$

Gleitkomma-Multiplikation

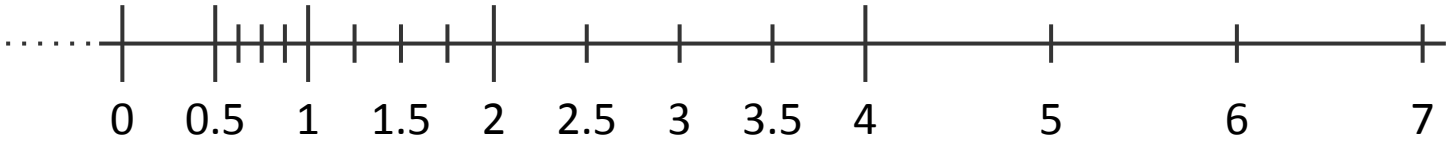


Gleitkomma-Multiplikation (Beispiel)

- Binärzahlen mit 4 Bits
  - $1.000_2 \times 2^{-1} \times -1.110_2 \times 2^{-2}$ (0.5×-0.4375)
- Exponenten addieren
  - Unbiased: $-1 + -2 = -3$
  - Biased: $(-1 + 127) + (-2 + 127) = -3 + 254 - 127 = -3 + 127$
- Signifikanden multiplizieren
  - $1.000_2 \times 1.110_2 = 1.110_2 \Rightarrow 1.110_2 \times 2^{-3}$
- Normalisieren (nicht notwendig)
  - $1.110_2 \times 2^{-3}$ kein Überlauf
- Runden (nicht notwendig)
  - $1.110_2 \times 2^{-3}$
- Vorzeichen bestimmen: +ve × –ve $\Rightarrow$ –ve
  - $-1.110_2 \times 2^{-3} = -0.21875$

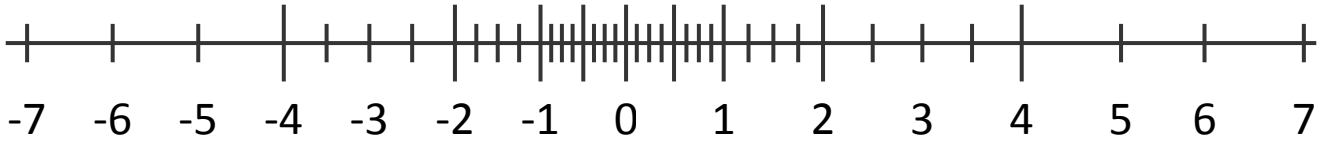
Genaue Arithmetik

- Gleitkommazahlen können ein Näherungswert für eine Zahl sein, die nicht dargestellt werden kann
- Beispiel
  - Vereinfachtes System: 2 Bit für Mantisse (4 Zahlen), $e_{min} = -1$, $e_{max} = 2$
  - Zahlenstrahl (nur normalisierte Gleitkommazahlen)



  - Zahlenstrahl (mit denormalisierten Gleitkommazahlen)

4.5? Im Bereich, aber nicht darstellbar!



- IEEE 754
  - 2^{53} Zahlen mit doppelter Genauigkeit, die skaliert werden können

Runden

- Zahlen, die nicht exakt dargestellt werden können (z.B. 4.5 auf der vorherigen Folie), müssen gerundet werden
- IEEE 754
  - 3 zusätzliche Bits für präzises Runden (Guard-, Round- und Sticky-Bit)
    - Werden nur während der Berechnung verwendet
  - Auswahl des Rundungsmodus
    - Programmierer kann diesen bei der Programmierung verwenden
- Nicht alle Gleitkommaeinheiten implementieren alle Optionen
- Kompromiss zwischen Komplexität der Hardware und praktischen Anforderungen

Assoziativität

- Achtung: Gleitkommaarithmetik ist nicht assoziativ
- Beispiel

		(x+y)+z	x+(y+z)
x	-1,50E+38		-1,50E+38
y	1,50E+38	0,00E+00	
z	1,0	1,0	1,50E+38
		1,00E+00	0,00E+00

LEISTUNG

Leistung

- Mit der Anzahl der Transistoren erhöht sich die Leistung
  - Was bedeutet aber Leistung wirklich?
- Beispiele
  - Durchschnittliche Ausführungszeit eines Programms über einen längeren Zeitraum (*sustained performance*)
  - Maximale Leistung (*peak performance*) eines Prozessors/Systems
    - Z.B. Integer-Operationen pro Sekunde
  - Durchschnittliche Leistung (*average performance*) eines Prozessors/Systems für eine bestimmte Arbeitslast
    - Durchschnittliche Anzahl an Integer-Operationen pro Sekunde

Faktoren

- Mehrere Faktoren beeinflussen die Leistung eines Programms

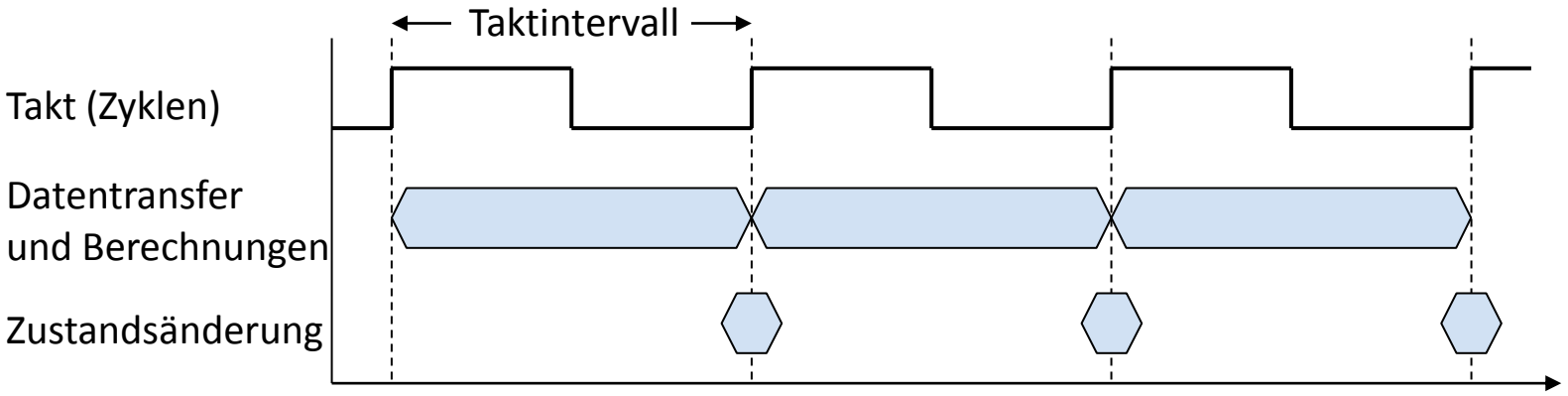
Hardware- und Softwarekomponente	Wie beeinflusst diese Komponente die Leistung?	Wo wird dieses Thema behandelt?
Algorithmus	Bestimmt sowohl die Anzahl der Anweisungen im Quellprogramm als auch die Anzahl der ausgeführten Ein- /Ausgabeoperationen	z.B. LV Algorithmen und Datenstrukturen
Programmiersprache, Compiler und Architektur	Bestimmt die Anzahl der Maschinenbefehle für jede Anweisung des Quellprogramms	Foliensatz 3
Prozessor und Speichersystem	Bestimmt, wie schnell Befehle ausgeführt werden können	Foliensatz 5, 6
Ein-/Ausgabesystem	Bestimmt, wie schnell Ein-/Ausgabeoperationen ausgeführt werden können	Foliensatz 7

Antwortzeit und Durchsatz

- Antwortzeit (*response time*) oder Ausführungszeit
  - Die Zeit, die der Rechner insgesamt benötigt, um eine Aufgabe zu erledigen
  - Gesamte Ausführungszeit (*wall-clock time*, *real time*)
    - Gesamtzeit für eine Aufgabe (inklusive Ein-/Ausgabe, Overhead vom Betriebssystem)
  - CPU-Ausführungszeit (CPU-Zeit, *cpu time*)
    - Tatsächliche Zeit für die Bearbeitung einer Aufgabe (ohne Overhead)
- Durchsatz (*throughput*)
  - Die Anzahl der Aufgaben, die pro Zeiteinheit ausgeführt werden

Taktzyklus

- Fast alle Rechner verwenden einen Takt
  - Binäres Rechtecksignal
    - Wechselt in periodischen Abständen zwischen 0 und 1
  - Legt fest, wann Ereignisse innerhalb der Hardware stattfinden
- Taktzyklus oder auch Tick, Takt, Zyklus (*clock cycle*)



- Taktzykluszeit oder Taktintervall (*clock period*)
  - Die Länge eines Taktzyklus, z.B. 250ps = 0.25ns = 250×10^{-12}s

Taktrate

- Taktgeschwindigkeit oder Taktrate (*clock frequency* (*rate*))
  - Das Inverse des Taktintervalls (Taktrate = 1/Taktzykluszeit)
  - Zyklen pro Sekunde
  - Die Einheit ist Hertz (Hz)
- Beispiele

Taktrate	Taktzykluszeit
1 Hz	1 s
1 KHz (10^3 Hz)	1 ms (10^{-3} s)
100 KHz (10^5 Hz)	10 µs (10^{-5} s)
500 KHz (5×10^5 Hz)	2 µs
1 MHz (1×10^6 Hz)	1 µs = 1000 ns
400 MHz (4×10^8 Hz)	2.5 ns
1 GHz (1×10^9 Hz)	1 ns = 1000 ps
2 GHz (2×10^9 Hz)	0.5 ns = 500 ps
4 GHz (4×10^9 Hz)	250 ps

CPU-Zeit für ein Programm

$$\text{CPU-Zeit} = \text{CPU-Taktzyklen} \times \text{Taktzykluszeit}$$
$$= \frac{\text{CPU-Taktzyklen}}{\text{Taktrate}}$$

- Leistung kann verbessert werden durch
  - Reduktion der Anzahl der Taktzyklen (für ein Programm)
  - Reduktion der Taktzykluszeit (Erhöhung der Taktrate)
- Aber
  - Viele Techniken, die die Anzahl der Taktzyklen senken, erhöhen möglicherweise auch die Taktzykluszeit!

Taktzyklen pro Befehl

- Befehlszähler (*instruction count*)
  - Anzahl der durch das Programm ausgeführten Befehle
- Taktzyklen pro Befehl (*Clock Cycles per Instruction*, *CPI*)
  - Durchschnittliche Anzahl der Taktzyklen pro Befehl für ein Programm
  - Durch die CPU-Hardware bestimmt
    - Prozessoren mit den gleichen Befehlen können unterschiedliche CPI-Werte haben (unterschiedlicher Hardwareaufbau)

$$\text{CPU-Taktzyklen} = \text{Befehlszähler} \times \text{CPI}$$

$$\text{CPU-Zeit} = \text{Befehlszähler} \times \text{CPI} \times \text{Taktzyklus zeit}$$

$$= \frac{\text{Befehlszähler} \times \text{CPI}}{\text{Taktrate}}$$

- Wie misst man diese Werte?

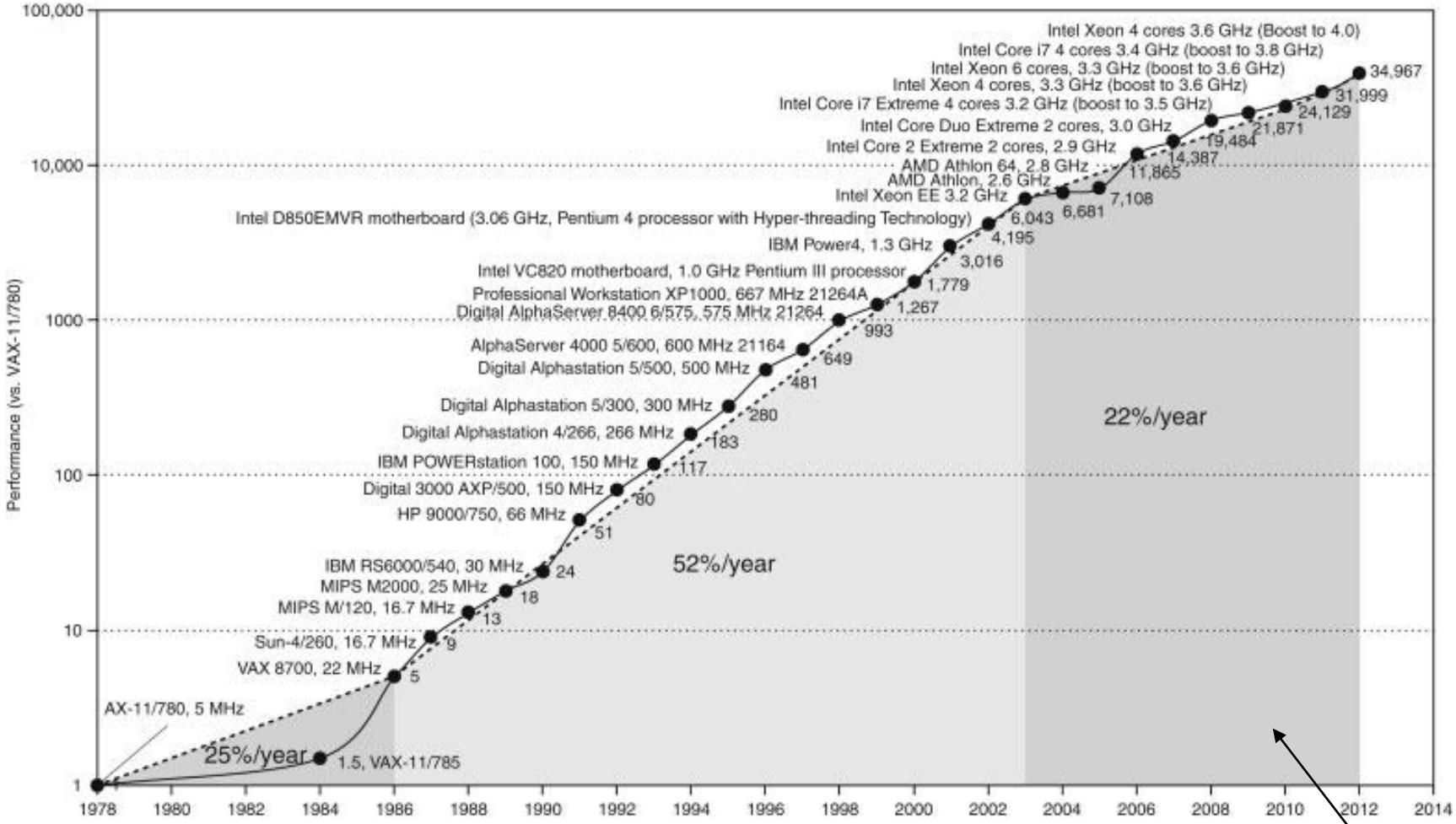
Benchmark

- Aussagekraft von Maßzahlen ist begrenzt
- Ausführungsgeschwindigkeit anhand „realer“ Programme messen
- Benchmark
  - Fest definiertes Referenzprogramm
  - Ausführung unter reproduzierbaren Bedingungen
    - Messung sollte bei Wiederholung (fast) gleiches Ergebnis liefern
- Benchmark-Kollektion
  - Mehrere Programme mit unterschiedlichen Eigenschaften

SPEC

- Standard Performance Evaluation Corporation (SPEC)
- Benchmark-Kollektionen
  - Leistung unter „realen“ Bedingungen bewerten
  - http://www.spec.org/benchmarks.html
- Beurteilung von Prozessorleistung (SPEC CPU2006)
  - SPECint2006 (ganzzahlige Operationen)
    - https://www.spec.org/cpu2006/CINT2006/
  - SPECfp2006 (Gleitkommaoperationen)
    - https://www.spec.org/cpu2006/CFP2006/
  - Endergebnis
    - Zwei Zahlen, die die Leistung bezogen auf eine Referenzmaschine angeben
  - Programme und Referenzmaschine ändern sich über die Jahre

Zunahme Prozessorrechenleistung (SPECint)



Einschränkungen: Verzögerung bei Speicherzugriffen etc.

Multicore-Prozessoren

- Durch Einschränkungen nur mehr 22% pro Jahr
  - Wandel in Hinblick auf das Design der Mikroprozessoren - Multicore-Prozessoren
- Multicore-Prozessoren
  - Mehr als ein Prozessor (*core*) pro Chip
  - Steigert den Durchsatz aber nicht unbedingt die Antwortzeit
  - Plan für die nahe Zukunft
    - Verdopplung der Cores pro Chip alle 2 Jahre
- Konsequenzen
  - Man kann sich nicht mehr darauf verlassen, dass die Software ohne Neuprogrammierung durch reine Hardware-Innovationen beschleunigt wird
  - Man muss für eine bessere Antwortzeit die Programme an die Multicore-Prozessoren anpassen

ZUSAMMENFASSUNG

Zusammenfassung

- Abstraktionsschichten
- Komponenten eines Rechners
  - Rechenwerk, Steuerwerk, Hauptspeicher, Eingabe, Ausgabe
- Zahlensysteme
  - Binär-, Oktal- und Hexadezimalsystem
  - Darstellung negativer Zahlen
  - Gleitkommazahlen, IEEE 754
- Leistung
  - Takt, Benchmark

Literatur

- Grundliteratur
  - D. A. Patterson, J. L. Hennessy: **Computer Organization and Design**, 5. Auflage, Morgan Kaufmann, 2013 – Kapitel 1 und 3
  - H. Herold, B. Lurz, J. Wohlrab, **Grundlagen der Informatik**, 2. Auflage, Pearson Studium, 2012 – Kapitel 5 und 13
  - D. W. Hoffmann: **Grundlagen der Technischen Informatik**, 4. Auflage, Hanser, 2014 – Kapitel 2 und 3