Universität Bielefeld
Technische Fakultät

R|V|S
Rechnernetze und
Verteilte Systeme

Technische Informatik I

Vorlesung 2: Zahldarstellung

Joachim Schmidt
jschmidt@techfak.uni-bielefeld.de

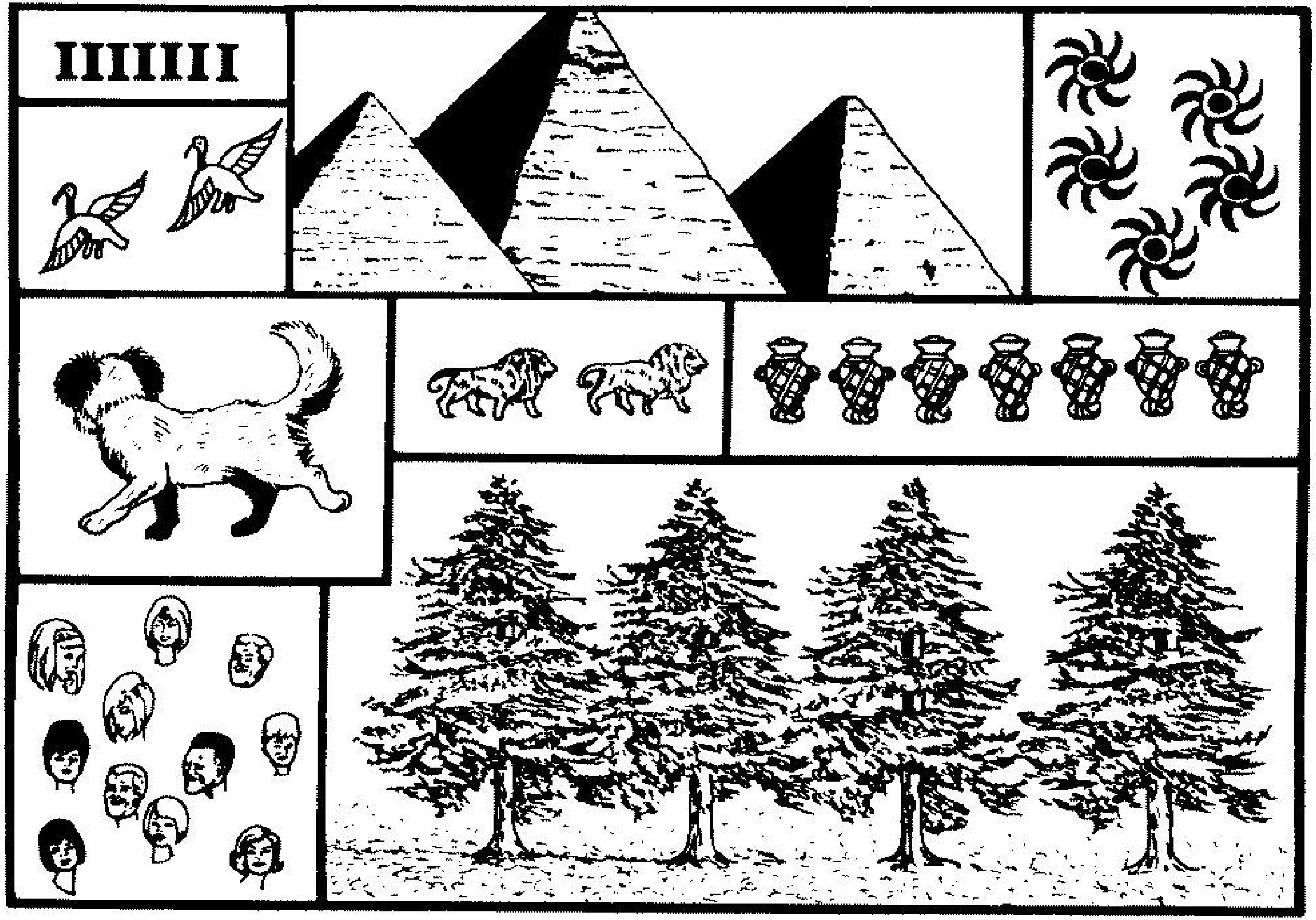
Übersicht

- Geschichte der Zahlen
- Zahlensysteme
  - Basis / Basis-Umwandlung
- Zahlsysteme im Computer
  - Binärsystem, Hexadezimalsystem, BCD
  - Negative Zahlen
  - 1er / 2er Komplement
  - Floating Point - wissenschaftliche Zahlen
- ASCII / Unicode-Kodierung
- Schlußfolgerung

Zahlwahrnehmung

- Wieso hat der Mensch den Zahlbegriff entwickelt?
  - Beobachtung der Natur
  - Wunsch, sich darüber zu verständigen
- Voraussetzung: Alle Menschen haben die gleiche Zahlwahrnehmung
- Ist das wirklich so?

Zahlwahrnehmung



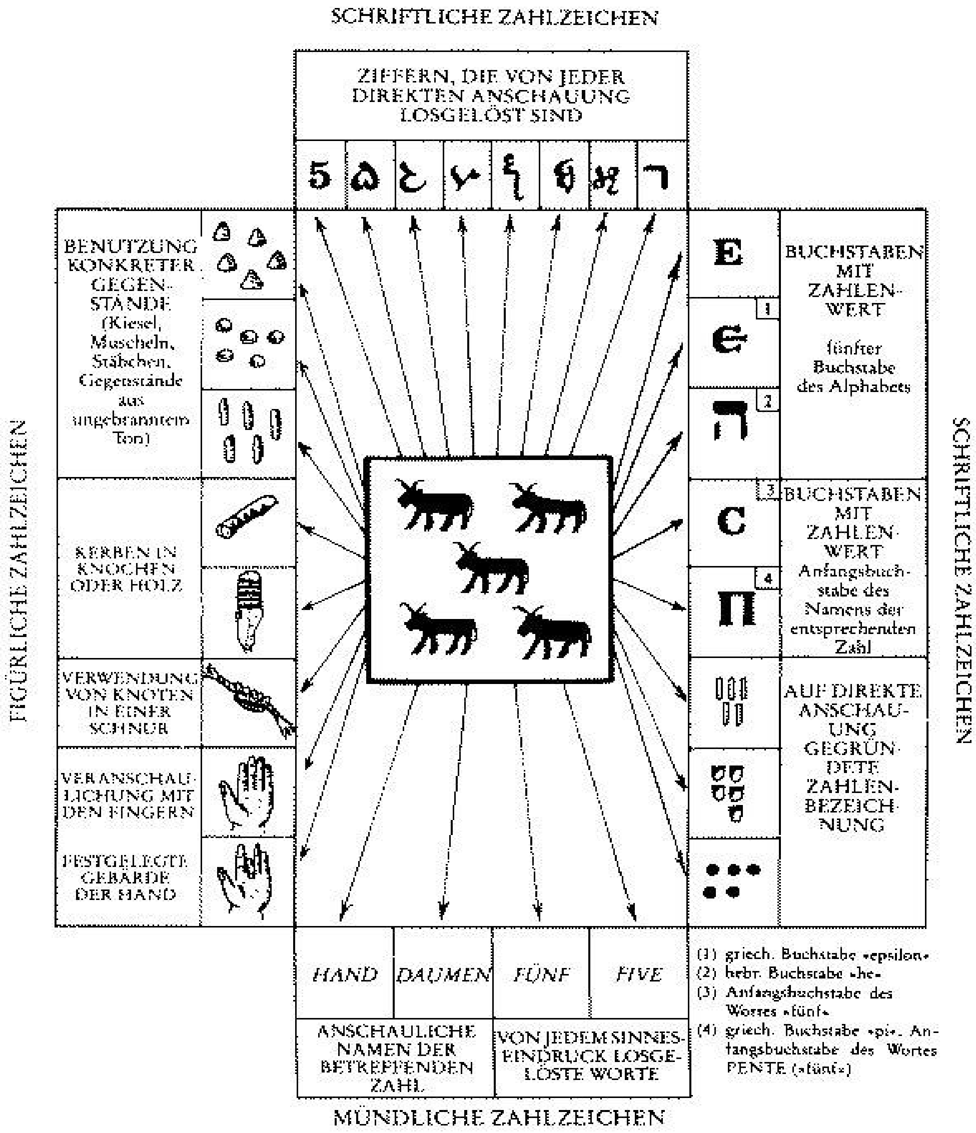
Zählen

- Zählen $\neq$ Zahlgefühl
- Natürliche Grenze liegt bei 3-5
- Für alles weitere muss man zählen
- Man benötigt ein Zahlensysstem

Zahlensysteme

- Was macht ein Zahlensystem aus?
  - Künstlich geschaffenes System
  - Bijektion Objekt/Symbol $\leftrightarrow$ Element einer Menge
  - Symbole in Beziehung setzten $\Rightarrow$ Rechnen
  - Sollte möglichst allgemein verständlich sein

Zahlzeichen



Zahlzeichen

- Der Mensch hat viele verschiedene Möglichkeiten entwickelt, Zahlen symbolisch darzustellen
- Konkrete Zahlzeichen
  - Gegenstände aller Art
  - Kerben in Knochen oder Holz
  - Geknotete Schnüre
  - Gesten mit Fingern, Zehen und anderen Körperteilen

Zahlzeichen

- Mündliche Zahlzeichen
  - Die Einheit: “Sonne“, “Mond”
  - Das Paar: “Augen“, “Flügel eines Vogels“
  - Die Drei: “Blätter des Klees”
  - Die Vier: “Pfoten eines Tieres”
- Vom Sinneseindruck losgelöste Zahlwörter
  - “eins”, “zwei”, “drei”
- Schriftliche Zahlzeichen
  - Graphische Zeichen aller Art

Zahlensysteme mit verschiedenen Basen

Dualsystem	Repräsentation im Rechner	Basis 2
Quinärsystem	Zählen mit einer Hand	Basis 5
Oktalsystem	Menschenlesbare Darstellung von Maschinenzahlen	Basis 8
Dezimalsystem	Zählen mit den Fingern	Basis 10
Doudezimalsystem	„Dutzend“, gut zu rechnen	Basis 12
Hexadezimalsystem	Menschenlesbare Darstellung von Maschinenzahlen	Basis 16
Vigesimalsystem	Zählen mit Fingern und Zehen	Basis 20
Sexagesimalsystem	Astronomie, Mathematik	Basis 60

Eigenschaften der Basis

Man benötigt eine für den Menschen überschaubare Größenordnung als Basis

kleinere Basis	Grössere Basis
längere Darstellung	kürzere Darstellung
einfacheres System	schwierigeres System
wenig Zahlzeichen	viele Zahlzeichen
überschaubares "1x1"	quadratisch wachsendes "1x1"

Zahlendarstellung

- Basis des Zahlensystems: B
- Ziffer: $a_i \in \{0, 1, 2, \ldots, B-1\}$
- Zahl: $\langle a_0, a_1, a_2, \ldots, a_n \rangle$

  geschrieben: $a_n a_{n-1} \ldots a_2 a_1 a_0$
- Wert: $a_0 * B^0 + a_1 * B^1 + \ldots + a_n * B^n$

  $= \sum_{i=1}^{n} a_i * B^i$

Zahlendarstellung

Binary	1	1	1	1	1	0	1	0	0	0	1

$$1 \times 2^{10} + 1 \times 2^{9} + 1 \times 2^{8} + 1 \times 2^{7} + 1 \times 2^{6} + 0 \times 2^{5} + 1 \times 2^{4} + 0 \times 2^{3} + 0 \times 2^{2} + 0 \times 2^{1} + 1 \times 2^{0}$$

$$1024 + 512 + 256 + 128 + 64 + 0 + 16 + 0 + 0 + 0 + 1$$

Octal	3	7	2	1

$$3 \times 8^{3} + 7 \times 8^{2} + 2 \times 8^{1} + 1 \times 8^{0}$$

$$1536 + 448 + 16 + 1$$

Decimal	2	0	0	1

$$2 \times 10^{3} + 0 \times 10^{2} + 0 \times 10^{1} + 1 \times 10^{0}$$

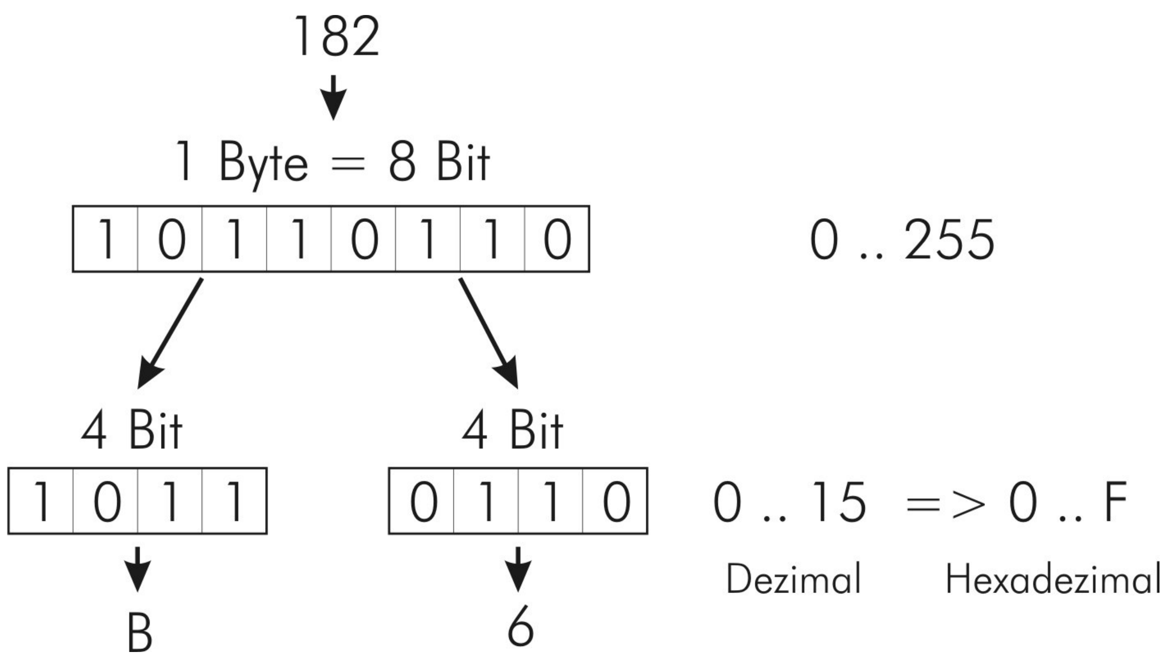
$$2000 + 0 + 0 + 1$$

Hexadecimal	7	D	1

$$7 \times 16^{2} + 13 \times 16^{1} + 1 \times 16^{0}$$

$$1792 + 208 + 1$$

Umwandlung Binär ↔ Hexadezimal

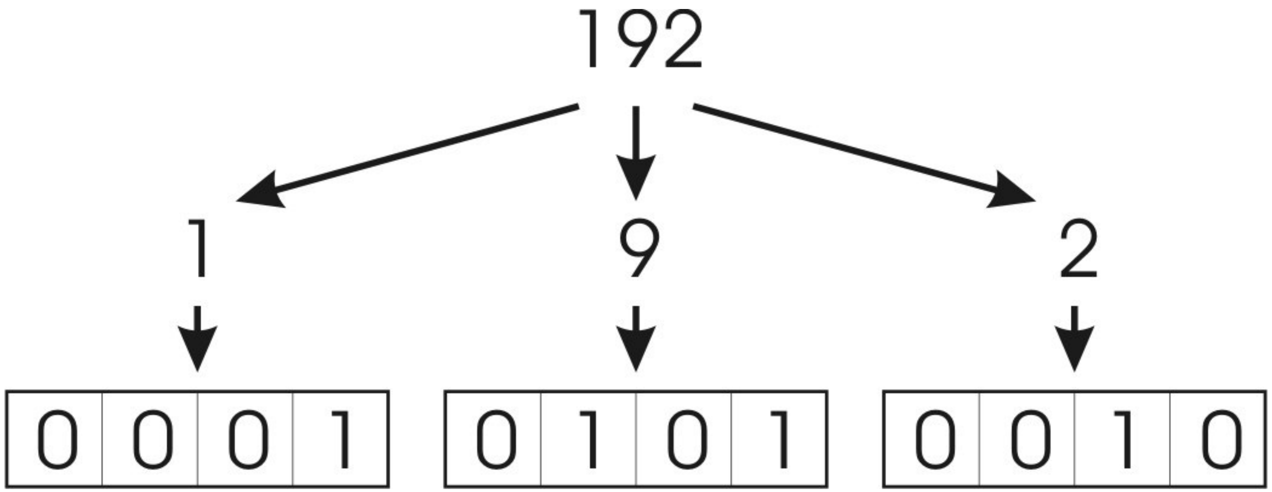


- Einfache Lesbarkeit
- Höhere Informationsdichte

Verschiedene Zahlensysteme

Binär	Oktal	Dezimal	Hexadezimal
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
1001	11	9	9
1010	12	10	A
10000	20	16	10
10100	24	20	14

BCD – Binary Coded Decimal



- Zahlenformat in COBOL
- Darstellung von Brüchen
- Vermeidung von Rundungsfehlern
- Anwendung bei Versicherungen / Banken

Basiskonversion

- Wunsch: Jedes Zahlensystem in jedes andere überführen
- Algorithmus:
  - Umzurechnende Zahl: A, neue Zahl: B=0
  - Alte Basis: B_A, neue Basis: B_B
  - Schrittzähler: n=0
    - Teile (ganzzahlig mit Rest) Zahl A durch neue Basis B_B
    - Speichere Ergebnis wieder in A zur Basis B_A
    - Füge den Rest als Ziffer zur Basis B_B an Stelle n in B ein
    - Nächster Schritt: n += 1
  - Solange, bis Zahl A=0

Basiskonversion

Quotients	Remainders
1492	
746	0
373	0
186	1
93	0
46	1
23	0
11	1
5	1
2	1
1	0
0	1

1 0 1 1 1 0 1 0 1 0 0 $= 1492_{10}$

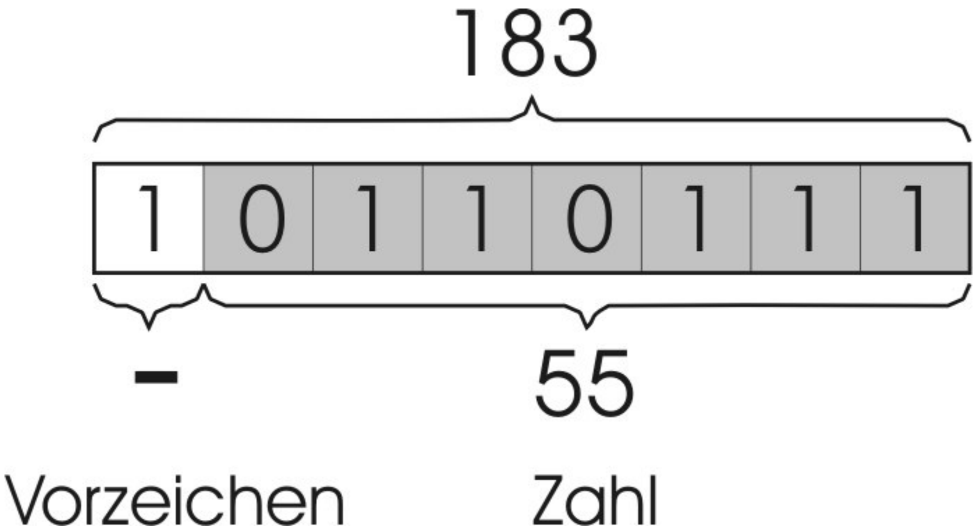
Grundrechenarten Binär

- Angelehnt an schriftliche Addition / Multiplikation
- Wegen Basis 2 sehr einfach
- Multiplikation zurückgeführt auf Addition

$$
\begin{array}{rcccccccc}
 & 1 & 0 & 1 & 0 & * & 1 & 1 & 0 \\
\hline
0*110= & & & & 0 & 0 & 0 & & \\
1*110= & & & 1 & 1 & 0 & & & \\
0*110= & & 0 & 0 & 0 & & & & \\
1*110= & 1 & 1 & 0 & & & & & \\
\hline
\Sigma & 1 & 1 & 1 & 1 & 0 & 0 & & \\
\hline\hline
\end{array}
$$

Negative Zahlen

- Wunsch: Auch negative Zahlen darstellen



- 1. Bit codiert Vorzeichen (Sign Bit)
- $\Rightarrow$ Weniger Bits für eigentliche Zahl übrig
- Vorsicht beim Rechnen!

Einerkomplement

- Andere Darstellung negativer Zahlen
- Ersetzte jede 0 durch eine 1, jede 1 durch 0
- Bitweise Negation
- Enthält ebenfalls Vorzeichenbit

Einerkomplement

- Darstellung ist eindeutig

+20: 00010100

-20: 11101011

- Zwei Darstellungen der Null:

+0: 00000000

-0: 10000000

Zweierkomplement

- Weitere Darstellung negativer Zahlen
- Ersetzte jede 0 durch eine 1, jede 1 durch 0
- Bitweise Negation
- Addiere 1
- Enthält Vorzeichenbit

Zweierkomplement

- Es gibt keine „negative 0“
- Hat andere „Singularität“:

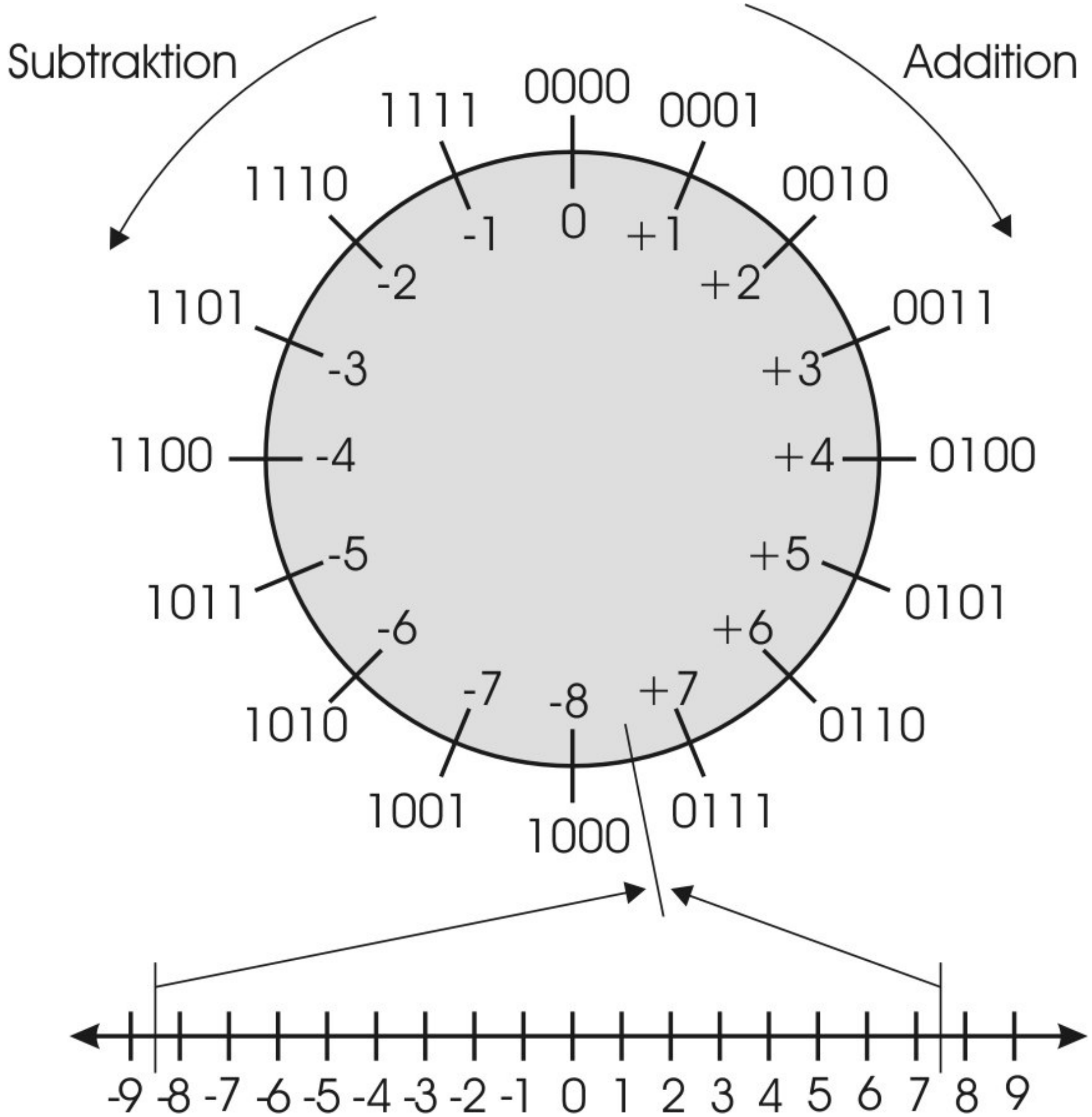
  10000000 ist sein eigenes Komplement:

  01111111 + 00000001 = 10000000
- Unsymmetrisch

  +20: 00010100

  -20: 11101100

Graphische Darstellung: 2er Komplement



Subtraktion in Komplementdarstellung

- Vorteil: Subtraktion = Addition einer negativen Zahl
- Nur ein Algorithmus / Schaltwerk nötig
- Problem: Overflow / Bereichsüberschreitung

Subtraktion in Komplementdarstellung

Decimal	1's complement	2's complement
10	00001010	00001010
+ (–3)	11111100	11111101
+7	1 00000110	1 00000111
	carry 1	discarded
	00000111	

Fließkommazahlen – Warum?

- Auch rationale Zahlen darstellen
- Sehr große und sehr kleine Zahlen darstellen
- Lieber feste Anzahl relevanter Stellen als fester Wertebereich
- Keine feste Position für den Dezimalpunkt
- Floating Point (FP)
- Fließkomma-Maschinenzahlen (FKM)

Fließkommazahlen – Wie?

- Angelehnt an wissenschaftliche Notation:

$$1226700000 = 1.2267 * 10^{10}$$

$$3.1415926 = 3.1415926 * 10^{0}$$

$$= 1.5707963 * 2^{1}$$

- Darstellung ist nicht eindeutig

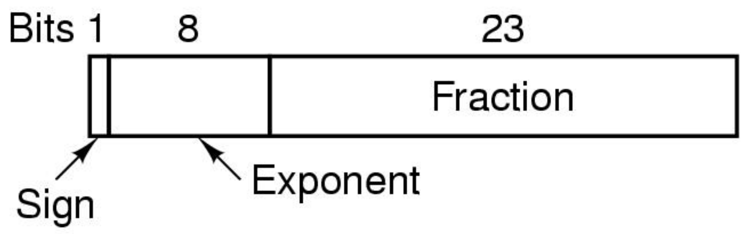
Prinzip Fließkommazahlen

- Jede reelle Zahl läßt sich ausdrücken durch

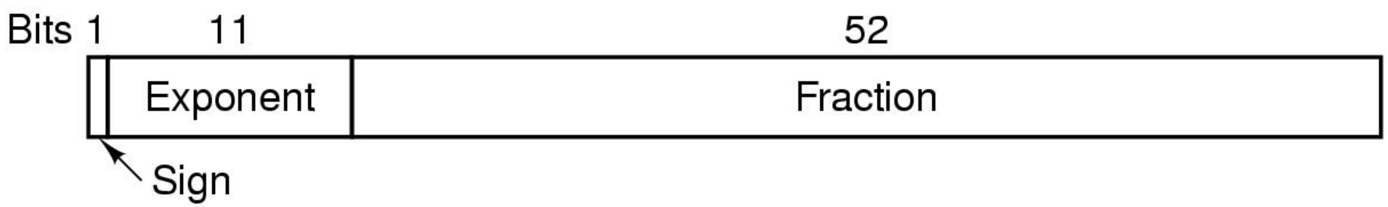
$$\pm \text{Mantisse} * \text{Basis}^{\text{Exponent}}$$

- Basis im wissenschaftlichen System: 10
- Basis im Rechner: 2 (bzw. 2^n)

Darstellung Fließkommazahlen

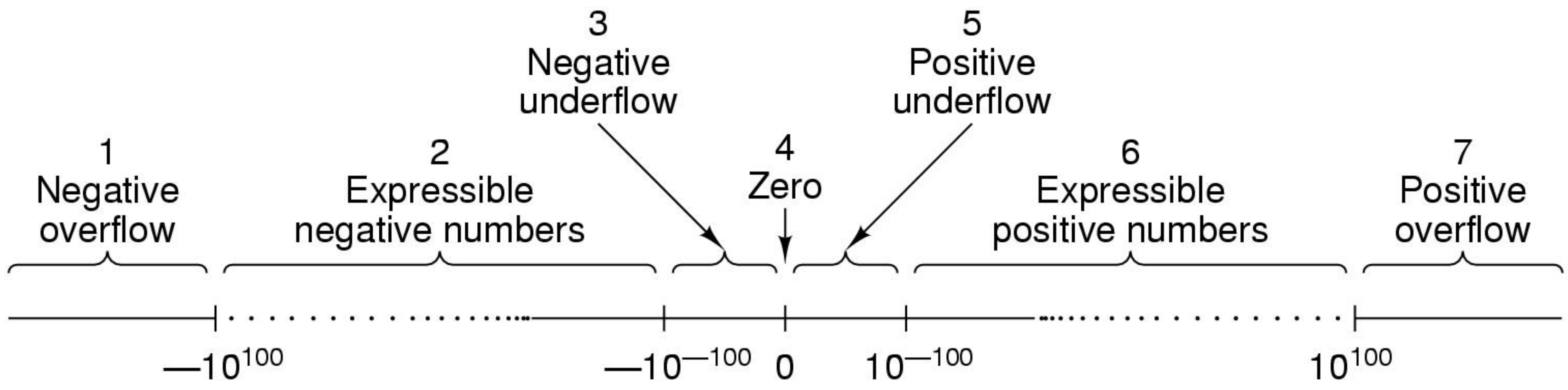


(a)



(b)

Dynamik Fließkommazahlen



Standards Fließkommazahlen

- Standards definiert in IEEE 754, 1985
- Single Precision (32 Bit)
  - Bereich: 10^{-38} .. 10^{38}
- Double Precision (64 Bit)
  - Bereich: 10^{-308} .. 10^{308}
- Extended Precision (80 Bit)
  - Meist nur intern benutzt

Normierung Fließkommazahlen

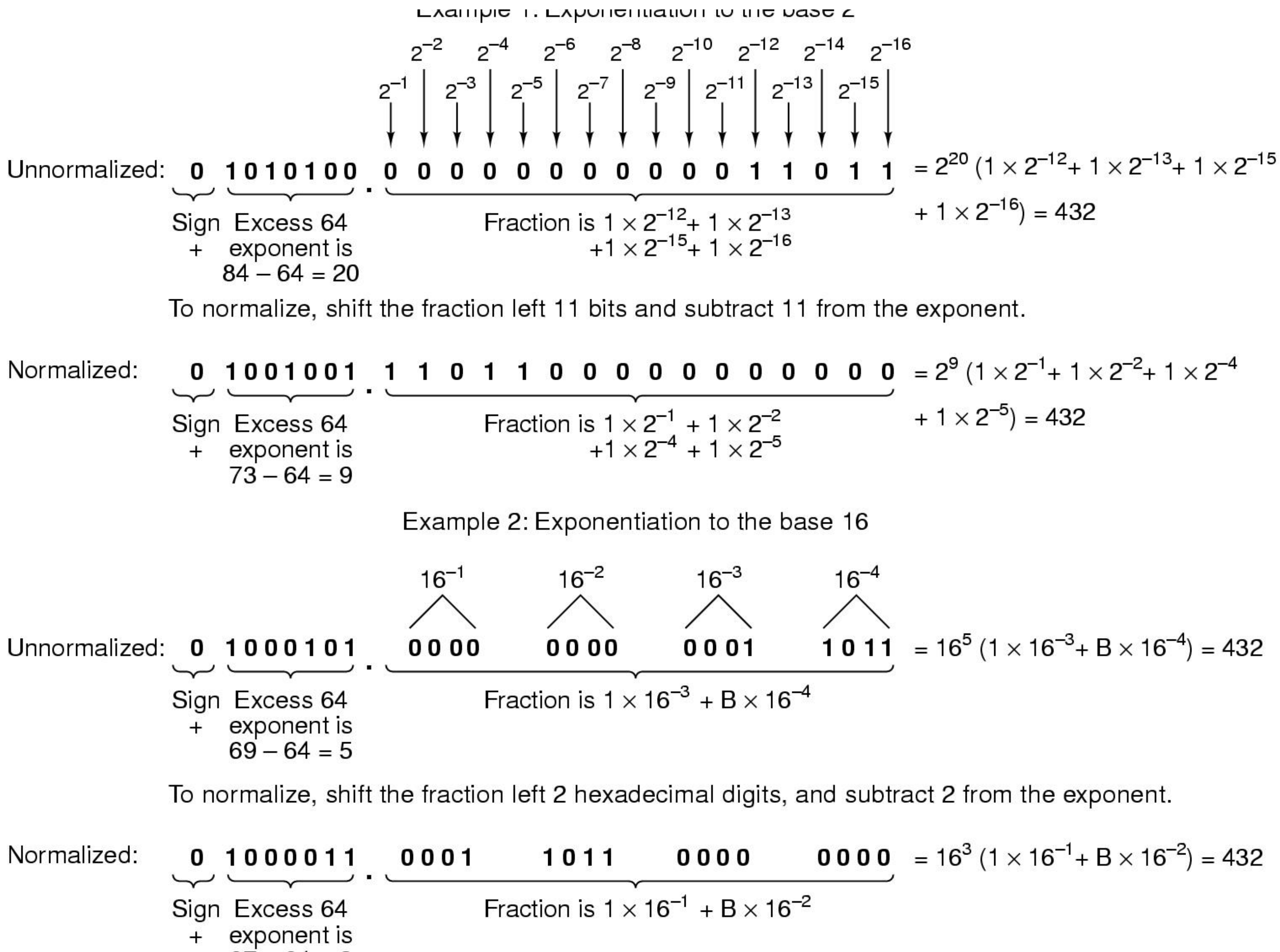
- 1 Vorzeichenbit: 0 = positiv, 1 = negativ
- Mantisse: normalisiert, d.h. führendes Bit = 1
  (kann deshalb weggelassen werden)
- Basis = 2
- Exponent + 127, um das Vorzeichen des Exonenten nicht abspeichern zu müssen

Normierung Fließkommazahlen

	Sign	Exponent	Fraction
Normalized	±	0 < Exp < Max	Any bit pattern
Denormalized	±	0	Any nonzero bit pattern
Zero	±	0	0
Infinity	±	1 1 1…1	0
Not a number	±	1 1 1…1	Any nonzero bit pattern

Sign bit

Fließkommazahlen



Probleme Fließkommazahlen

- Darstellung nicht eindeutig $\quad A == B?$
- Nicht jede reelle Zahl läßt sich exakt darstellen:

$$^{1}/_{3} = 0.3333333333...$$

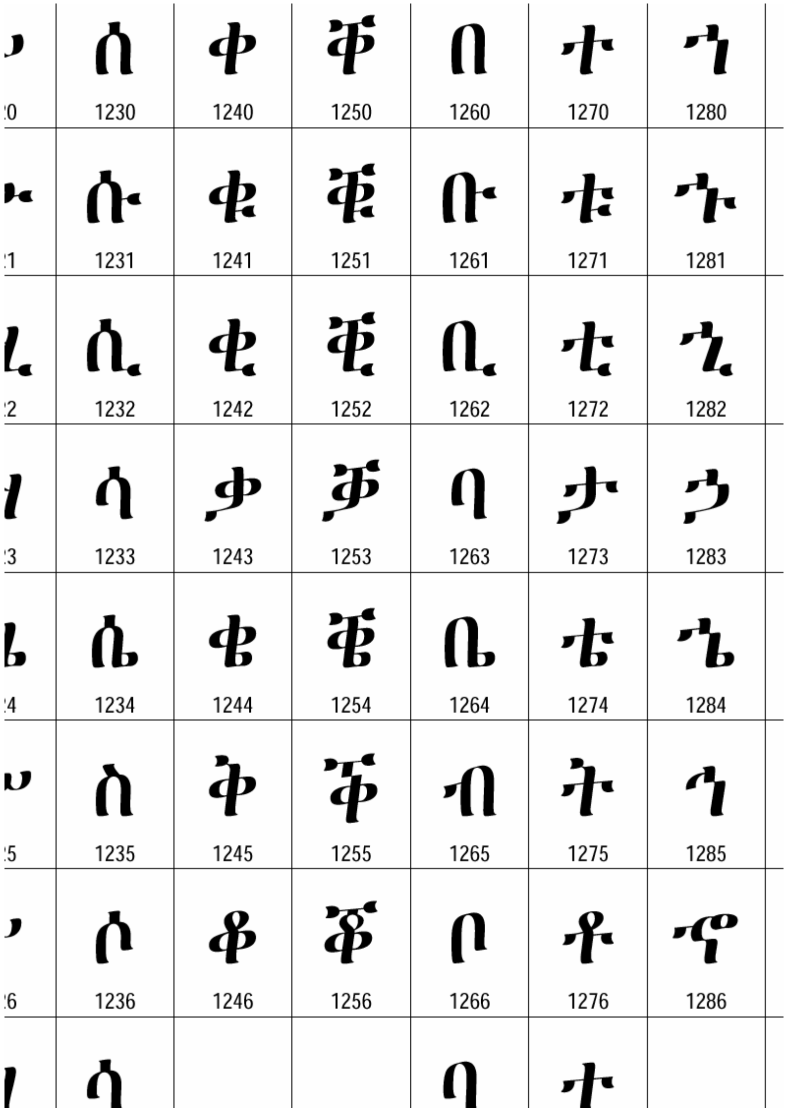
- Rechenoperationen (insbesondere Multiplikation und Division) relativ aufwendig, Berechnung im Floating-Point-Prozessor

  $\rightarrow$ VL: Numerische Methoden der Informatik

ASCII

- American Standard Code for Information Interchange
- 127 Zeichen (7 Bit)
  - Alphabet
  - Ziffern
  - Sonderzeichen
  - Steuerzeichen
- Erweiterung: 256 Zeichen (8 Bit)
- Unicode: 65535 Zeichen (16 Bit)

Unicode


1230 1240 1250 1260 1270 1280
1231 1241 1251 1261 1271 1281
1232 1242 1252 1262 1272 1282
1233 1243 1253 1263 1273 1283
1234 1244 1254 1264 1274 1284
1235 1245 1255 1265 1275 1285
1236 1246 1256 1266 1276 1286


F925 F935 F945 F955 F965 F975 F985
F926 F936 F946 F956 F966 F976 F986
F927 F937 F947 F957 F967 F977 F987
F928 F938 F948 F958 F968 F978 F988
F929 F939 F949 F959 F969 F979 F989
F92A F93A F94A F95A F96A F97A F98A
F92B F93B F94B F95B F96B F97B F98B

Schlußfolgerung I

- Jede Zahl / Information läßt sich im Binärsystem darstellen
- Die meisten Rechenoperationen lassen sich auf elementare Operationen (Addition) zurückführen

Schlußfolgerung II

- Jede Zahl / Information läßt sich beliebig genau im Rechner darstellen
- Jede Rechenoperation läßt sich beliebig genau durchführen
- Für Grundrechenarten gibt es effiziente Algorithmen
- Einschränkungen durch Geschwindigkeit und Speichergröße

Literatur und Links

- Universalgeschichte der Zahlen
  Georges Ifrah, GLB Parkland, 1998
- Structured Computer Organization
  Andrew S. Tanenbaum, Prentice Hall, 1999
- Vorlesung:
  Numerische Methoden der Informatik
  (voraussichtlich WS 02/03)
- elearn.rvs.uni-bielefeld.de

Universität Bielefeld
Technische Fakultät

R|V|S
Rechnernetze und
Verteilte Systeme

Technische Informatik I

Nächste Woche:
Vorlesung 3: Bool’sche Algebra

Mirco Hilbert
mail@mirco-hilbert.de