Einführung in die Informatik

1.1 Geschichte der Digitaltechnik

- DIGITAL: aus Lateinischem (digitus = Finger ☝)
- Historische Meilensteine der Digitaltechnik:
  - Gottfried Wilhelm Leibniz (1679): Entwicklung duales Zahlensystem (und mechanischer Rechenmaschinen)
  - George Boole (1854): Erweiterung um algebraisches System
  - Conrad Zuse u.a. (1936/37): erste elektrische Rechenmaschinen (Relais- und Röhrentechnik)
  - Bardeen, Brattain und Shockley (1947): Germaniumtransistor
  - 1954: Siliziumtransistoren
  - 1971: erster 4-Bit Mikroprozessor (Firma Intel) als integrierte Schaltung (IC)
  - danach: exponentieller Anstieg der Transistoren pro IC
- 1 Bit = {0,1} (2 Werte)

1 2 4 8 16 32 64 128

1.2 Zahlendarstellung

- Dargestellter Wert $W=d_{n-1}\cdot 10^{n-1}+d_{n-2}\cdot 10^{n-2}+\ldots+d_1\cdot 10^1+d_0\cdot 10^0$ Aufgabe: 1204 = $1\cdot 10^3+2\cdot 10^2+0\cdot 10^1+4\cdot 10^0$
- Allgemein: $W=z_{n-1}\cdot B^{n-1}+\ldots+z_1\cdot B^1+z_0\cdot B^0$ (B=Basis, z.B. 2,8,10,16)
- Dualsystem (B=2): 1 Bit = 1 Ziffer = {0,1} (2 Werte) - Aufgabe: 1010 als Potenzsumme&Dezimalzahl darstellen: 1010=… $1\cdot 2^3+1\cdot 2^1$
- Dezimalzahl in Dualzahl konvertieren – Beispiel 13, Aufgabe 222_{10} konvertieren:

  222 − 128 = 94; 94 − 64 = 30; 30 − 16 … 11011110

- 4 Bits = 1 Nibble = {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}, jede Hexadezimalziffer entspricht einer 4-Bit Dualzahl

  z.B.: $13_{(10)}$= $D_{(16)}$=0xD, Aufgabe – als Dual- und Dezimalzahl darstellen: 0xABCD= 1010.1011.1100.1101 | $10\cdot 16^3+11\cdot 16^2+12\cdot\ldots+13$ 43981

1.3 Einheiten von Dualzahlen (IEC)

- 1 Bit = 1 Ziffer = {0,1}
- 4 Bits = 1 Nibble = {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}
- 8 Bits = 2^3 Bits = 2 Nibble = 1 Byte
- IEC-Norm: 1KiBit = 2^{10} Bits = 1024 Bits (1 KiBi-Bit≠1000 Bits!)
- Entspr.: 1 Kibi Byte = 1 KiB = 2^{10} B = 1024 B (= $2^{10} * 2^3$ Bits = 2^{13} Bits)
- Mebi-Byte: 1 MiB = 2^{20} Bytes
- Gibi-Byte: 1 GiB = 2^{30} Bytes
- Tebi-Byte: 1 TiB = 2^{40} Bytes

In Si-Normen Doppeldeutigkeiten von Einheiten: z.B. 1KB = 1000B oder 1KB = 1024B

1.4 Additionen von Dualzahlen

	4	3	2	1	0
7		0	1	1	1
F	$+_1$	1_1	1_1	1_1	1
16	1	0	1	1	0

Aufgabe: $01101101_{(2)}$ + $10100101_{(2)}$

Addition im Zweierkomplement

- Vorzeichenbits aufdoppeln (=kopieren) und wie gewohnt binär addieren
- Wie kann man entscheiden, ob das Additionsergebnis (n+1 Bits) auch als n-Bit Zahl darstellbar ist?

1.5 Zweierkomplement

- n-Bit Dualzahlen von -2^{n-1} bis $2^{n-1}-1$
- Vorgehen Vorzeichenumkehr – pos. <-> neg. Zahl – z.B. +6 → -6:
- Einerkomplement bilden (bitweise invertieren): 0110 → 1001
- +1 (eins dazu addieren): 1001+1=1010 = -6_{10}
- Aufgabe: Zweierkomplement zu -15_{10} bestimmen

	4	3	2	1	0
-5	1	1	0	1	1
4	+ 0	0	1	0	0
-1	1	1	1	1	1

1.6 gebrochene Zahlen / Kommazahlen

- Wert-Darstellung von gebrochenen Zahlen (B=10): $W=d_{n-1}\cdot 10^{n-1}+d_{n-2}\cdot 10^{n-2}+\ldots+d_1\cdot 10^1+d_0\cdot 10^0+d_{-1}\cdot 10^{-1}+d_{-2}\cdot 10^{-2}+d_{-3}\cdot 10^{-3}+\ldots$
- z.B.: $299{,}8_{(10)}=2\cdot 10^2+9\cdot 10^1+9\cdot 10^0+8\cdot 10^{-1}$
- allgemein: $w=z_{n-1}\cdot b^{n-1}+\ldots+z_1\cdot b^1+z_0\cdot b^0+z_{-1}\cdot b^{-1}+z_{-2}\cdot b^{-2}+\ldots$
- $299{,}8_{(10)}=?_{(2)}$

Vorkomma-Teil	Nachkomma-Teil
299 / 2 = 149 Rest 1 (LSB) 149 / 2 = 74 Rest 1 … 1001 => 10010__11=0X12_	0,8 * 2 = 1,6 => 1 + 0,6 (MSB) 0,6 * 2 = 1,2 => 1 + 0,2 => $0{,}8_{(10)}=0{,}\overline{11}_{(2)}=0{,}\_{(16)}$

- Hinweis: 299,8 ist im Dualsystem unendlich lang => Rundungsfehler bei Rückkonvertierung: 0,11__=$2^{-1}+2^{-2}+\ldots$=0,75 (vergleiche Loncapa/EDX-Gleichheitsprüfung von Dezimalzahlen!)
- Fließkommazahlen: wissenschaftliches Format: z.B. $c \approx 2{,}998\cdot 10^8\,\frac{m}{s}$ (2,998 = Mantisse (m), 8 Exponent (e))
- in Digitaltechnik: b=2, üblich $1 \le m < b$ => MSB ist immer 1 außer für die Zahl 0 (Wertebereich von m: [1;2) => man muss MSB nicht abspeichern (hidden bit), Sonderkodierungen für Zahlen $+0, -0, +\infty, -\infty$, NaN (not a number)
- Fließkommazahlen – IEEE 754 Norm: $x = s\cdot m\cdot 2^e = (-1)^S\cdot\left(1+\frac{M}{2^{(n_M)}}\right)\cdot 2^{(E-\mathrm{Bias})}$
  - S=Sign=Vorzeichenbit: 0 wenn x≥0, sonst 1; M: [1;2) – nur Nachkommastellen abspeichern (hidden bit); E: Bias/Offset von $2^{n-1}-1$, um neg. Exponenten zu ermöglichen (z.B. Single: Erhöhung tatsächlicher Exponent um 127) Zusammenhänge: $m = 1+\frac{M}{2^{(n_M)}}$; $M = 2^{(n_m)}\cdot(m-1)$; $e = E - Bias$; $E = e + Bias$; $s = (-1)^S$

Datentyp	Mantisse [Bits](n_M)	E: Exponent [Bits] / Bias
Half	10	5 (e: [-14;15]) / 15
Single	23	8 (e: [-126;127]) / 127
Double	52	11 (e: [-1022;1023]) / 1023

- z.B.: $2{,}998\cdot 10^8 \approx 2^{20}\cdot 285{,}912_{(10)}10^8$ -> Single S:0 E:10011011 M:00011101111010010111100
- Aufgabe: -9,25 (Datentyp Single)
- Aufgabe: Welche *Werte* für Exponenten E fehlen?

Sonderfälle Exponenten (n_E=Anzahl Bits Exponent; n_M=Anzahl Bits Mantisse):

- $E=0$: **kein** hidden Bit, höchstwertigstes Bit von M ist Vorkommabit $\Rightarrow x = (-1)^S\cdot M\cdot 2^{(-n_M-Bias)}$ mit $M = 2^{(n_m)}\cdot m$
- $E = 2^{n_M}-1 = 11\ldots 1$: $x = \begin{cases}(-1)^S\cdot\infty & \text{für } M=0\\ NaN \text{ (not a number)} & \text{für } M\neq 0\end{cases}$

Aufgaben: a. Darstellung Zahl 0, b. kleinste mögliche pos. Zahl, c. warum Vorzeichen bei 0 interessant? d. Beispiele für Rechenoperationen mit Ergebnis NaN

Kleinbuchstaben=Zahlenwerte in Exponentialdarstellung; Großbuchstaben=im Speicher abgelegte Dualzahlen

222 : 8 = 27 R 6
27 : 8 = 3 R3
3 : 8 = 0 R 3

$\underline{326_8} = 222_{10} = \underline{\underline{0xDE_{16}}} = \underline{\underline{3@_{64}}}$

9ABCD

222 : 16 = 13 R14 E
13 : 16 = 0 R13 D

222 : 64 = 3 R30
3 : 64 = 0 R3

30 - 9 = 21

A : 1010
B : 1011
C : 1100
D : 1101
E : 1110
F : 1111