Einführung in die IT

Teil 2:

Zahlensysteme
Binäre Arithmetik
Informationstheorie
Codierungen

Bernd Schöner

DHBW Mannheim

Darstellung von Zahlen

- Für das praktische Rechnen verwendet man dem Problem angepasste Ziffernsysteme. Für Menschen am geläufigsten ist das Dezimalsystem. Für die digitale Datenverarbeitung sind jedoch Ziffernsysteme günstiger, die dem Umstand Rechnung tragen, dass für die Darstellung von Zahlen in digitalen Rechenanlagen nur die beiden Ziffern 0 und 1 verwendet werden.
- Als elementare Maßeinheit für Information dient das Bit. Eine Nachricht (z.B. ein Text od. eine Zahl) hat einen Informationsgehalt von n Bit.
- Der Informationsgehalt einer Nachricht ist eine Größe, die angibt, wie viel Information in dieser Nachricht übertragen wurde.
- Er bezeichnet die minimale Anzahl von Bits, die benötigt werden, um ein Zeichen (also eine Information) darzustellen oder zu übertragen.
- Am häufigsten kommen für diesen Zweck das Binärsystem und das Hexadezimalsystem, bisweilen auch das Oktalsystem zur Anwendung.

Zahlensysteme

- Unter einem Zahlensystem zur Basis b verstehen wir ein Stellenwertsystem, bei dem Werte aus der Addition einzelner Zahlen aus der Menge {0, 1, …, b-1} multipliziert mit Potenzen der Basis b dargestellt werden: Die n-stellige Zahl mit m Nachkommastellen hat den Wert:

$$\sum_{k=-m}^{n-1} d_k b^k = d_{n-1}b^{n-1} + \ldots + d_0 + d_{-1}b^{-1} + \ldots + d_{-m}b^{-m}$$

- Bei Zahlensystemen mit einer Basis b > 10 werden die Buchstaben A-Z für Ziffern mit Wert ab 10verwendet.
- Der Informationsgehalt einer Nachricht ist eine Größe, die angibt, wie viel Information in dieser Nachricht übertragen wurde.
- Er bezeichnet die minimale Anzahl von Bits, die benötigt werden, um ein Zeichen (also eine Information) darzustellen oder zu übertragen.
- Am häufigsten kommen für diesen Zweck das Binärsystem (101_{bin}) und das Hexadezimalsystem ($2A_{hex}$) , bisweilen auch das Oktalsystem (37_{okt}) zur Anwendung.
- Werden Zahlen ohne einen Index zum Zahlensystem angegeben, so ist hier die Dezimaldarstellung gemeint.

Überblick über reelle Zahlen

<table>
<tr><td colspan="6">reelle Zahlen ($\mathbb{R}$)</td></tr>
<tr><td colspan="4">rationale Zahlen ($\mathbb{Q}$)</td><td colspan="2" rowspan="2">irrationale Zahlen</td></tr>
<tr><td colspan="3">ganze Zahlen ($\mathbb{Z}$)</td><td colspan="2">Bruchzahlen</td></tr>
<tr><td>positive ganze Zahlen ($\mathbb{N} = \mathbb{Z}^{+}$)</td><td>Null</td><td>negative ganze Zahlen ($\mathbb{Z}^{-}$)</td><td>positive</td><td>negative</td><td>positive</td><td>negative</td></tr>
</table>

Dezimalsystem/1

- Im Dezimalsystem (Zehnersystem) wird eine natürliche Zahl z als Summe von Potenzen zur Basis B = 10 dargestellt **(*Summenformel)***:

  $z = a_n 10^n + a_{n-1}\ 10^{\ n-1} +. . .+ a_2 10^2 + a_1 10^1 + a_0 10^0$

  $a_0, a_1, ..a_n \in \{0, 1..9\}$

  Basis : 10_{dez}

  $n \in \mathbb{N}$

Beispiel : $2789 = 2 \times 10^3 + 7 \times 10^2 + 8 \times 10^1 + 9 \times 10^0$

Dezimalsystem/2

- Erweitert man dieses Konzept um negative Exponenten, so lassen sich auch Dezimalbrüche, d. h. rationale Zahlen r darstellen:

  $r = a_n 10^n + ... + a_1 10^1 + a_0 10^0 + a_{-1} 10^{-1} + a_{-2} 10^{-2} + ... + a_{-m} 10^{-m}$

  $a_0, a_1, ..a_n \in \{0, 1..9\}$,

  Basis : 10_{dez}

  $r \in \mathbb{Q}$

- Eine rationale Zahl hat in obiger Notation also n+1 Vorkommastellen und m Nachkommastellen.

  Beispiel: Die Zahl 257,34 lautet in der oben definierten Darstellung:

  $257{,}34 = 2 \times 10^2 + 5 \cdot 10^1 + 7 \cdot 10^0 + 3 \cdot 10^{-1} + 4 \cdot 10^{-2}$

Dezimalsystem/3

- Unendliche, d. h. nicht abbrechender Dezimalbruch oder eine reelle Zahl mit unendlich vielen Stellen:

$i = a_n 10^n + ... + a_1 10^1 + a_0 10^0 + a_{-1} 10^{-1} + a_{-2} 10^{-2} + ... + a_{-\infty} 10^{-\infty}$

$a_0, a_1, ..a_n \in \{0, 1..9\}$

Basis : 10_{dez}

$i \in \mathbb{R} \backslash \mathbb{Q}$

Beispiel: $\pi = 3.141.. = 3 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2} + 1 \times 10^{-3} ..$

Bemerkung: $\mathbb{R} \backslash \mathbb{Q}$ steht für die Menge der irrationalen Zahlen

Dualsystem

- Auf Grund der Repräsentation von Daten in DV-Anlagen durch die beiden Zustände „0“ und „1“ bietet sich in diesem Bereich das Dualsystem (auch Zweiersystem oder Binärsystem) an. Es arbeitet mit der Basis *B* = 2 und den beiden Grundziffern *G* = *{0,1}*

$$b = \underbrace{a_n 2^n + a_{n-1}\, 2^{n-1} + \ldots + a_1 2^1 + a_0 2^0}_{r \in \mathbb{Z}}$$

$$\underbrace{\underbrace{a_{-1}2^{-1} + a_{-2}2^{-2} + \ldots + a_{-m}2^{-m}}_{i \in \mathbb{Q}} + \ldots + a_{-\infty}2^{-\infty}}_{i \in \mathbb{R}\setminus\mathbb{Q}}$$

$a_n, \ldots, a_1, a_0, \ldots, a_{-m}, a_{-\infty} \in \{0, 1\}$

Basis : 2_{dez}

Beispiel: $10010 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$ $(= 18_{dez})$

Hexadezimalsystem

- Eine noch kompaktere Zahldarstellung ergibt sich, wenn man jeweils vier Binärziffern zu einer Hexadezimalziffer zusammenzieht. Dieses Ziffernsystem ist u. a. deshalb sehr praktisch, weil die mit einem Byte codierbaren Zahlen gerade mit zweistelligen Hexadezimalzahlen geschrieben werden können.

$$\underbrace{h = a_n 16^n + a_{n-1} 16^{n-1} + \ldots + a_1 16^1 + a_0 16^0}_{h \in \mathbb{Z}}$$

$$\underbrace{\underbrace{+ a_{-1} 16^{-1} + a_{-2} 16^{-2} + .. + a_{-m} 16^{-m}}_{h \in \mathbb{Q}} + \ldots + a_{-\infty} 16^{-\infty}}_{H \in \mathbb{R} \setminus \mathbb{Q}}$$

$a_0, a_1, ..a_n \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

$a_{-m}, a_{-\infty} \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

Basis : 16_{dez}

Beispiel: $0x1C4F = 1 \times 16^3 + C \times 16^2 + 4 \times 16^1 + F \times 16^0 = 7247_{dez}$, bzw. $0001\ 1100\ 0100\ 1111_{bin}$

Hexadezimalsystem

- Die Zahlenwerte von Brüchen ergeben sich durch Multiplizieren der Stellenwerte mit B^{-k}, wobei B die Basis des Ziffernsystems ist und k die Position, also z. B. 1 für die erste und 2 für die zweite Nachkommastelle.
- Im Hexadezimalsystem entspricht der ersten Nachkommastelle also der dezimale Zahlenwert $16^{-1} = 0{,}0625$, der zweiten $16^{-2} = 0{,}00390625$ usw.
  - $1/16 = 0{,}0625$
  - $1/16^{-2} = 1/256 = 0{,}00390625$
- Im Binärsystem hat die erste Nachkommastelle den Wert $2^{-1} = 0{,}5$, die zweite den Wert $2^{-2} = 0{,}25$. Allgemein hat in einem Zahlensystem mit Basis B die k-te Nachkommastelle den dezimalen Wert $B^{-k.}$
  - ½ = 0,5
  - $1/2^{-2} = ¼ = 0{,}25$

Zahlendarstellungen in Dez, Bin, Hex

Dez	Bin	Hex	Dez	Bin	Hex
0	0000	0x0	8	1000	0x8
1	0001	0x1	9	1001	0x9
2	0010	0x2	10	1010	0xA
3	0011	0x3	11	1011	0xB
4	0100	0x4	12	1100	0xC
6	0101	0x5	13	1101	0xD
6	0110	0x6	14	1110	0xE
7	0111	0x7	15	1111	0xF

0 - 15dez in Binär-, Dezimal- & Hexadezimaldarstellung

Bits and Bytes

- Ein Bit (ein „binary digit“) bezeichnet die kleinste Informations- und Zähleinheit in einem Computer in Form einer einstelligen Binärzahl. Es kann die Werte 0_{bin} und 1_{bin} bin annehmen. Eine achtstellige Binärzahl wird als ein Byte bezeichnet.
- Ein Byte lässt sich damit entweder mit einer 8-stelligen Binärzahl, mit 3-stelligen Oktal- oder Dezimalzahlen oder mit 2-stelligen Hexadezimalzahlen darstellen.
- Beispiel 2.4. Die Umrechnung zwischen Binär-, Oktal- und Hexadezimaldarstellung

Bits werden in 8er Blöcken zu Bytes zusammengefasst,

1010 1100	1100 1110	1010 0011	1111 1010	…
1. Byte	2. Byte	3. Byte	4. Byte	

Stellenwertigkeit 2^0 – 2^7 in Bin, Dez, Hex.

	Dez	Bin	Hex
2^0	1	0000 0001	0x01
2^1	2	0000 0010	0x02
2^2	4	0000 0100	0x04
2^3	8	0000 1000	0x08
2^4	16	0001 0000	0x10
2^5	32	0010 0000	0x20
2^6	64	0100 0000	0x40
2^7	128	1000 0000	0x80

Beispiel: Was ist 1111 1111 in Dez und Hex?

Konvertierung Bin, Okt, Hex -> Dez

Die Umrechnung von Binär-, Oktal- oder Hexadezimalzahlen zu Dezimalzahlen erfolgt durch Addition der Produkte der Ziffern mit ihrem Stellenwert.

Die Binärzahl 1011,01 hat die dezimale Darstellung
$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \cdot 1^{-2} = 8 + 2 + 1 + 0{,}25 = 11{,}25$

Die Oktalzahl 64_{okt} hat die dezimale Darstellung
$6 \times 8^1 + 4 \times 8^0 = 48 + 4 = 52$

Die Hexadezimalzahl $x = F8_{hex}$ hat die dezimale Darstellung
$15 \times 16^1 + 8 \times 16^0 = 240 + 8 = 248$

Konvertierung Dez -> Bin durch Divisionsmethode

Man dividiert die umzuwandelnde Dezimalzahl durch die größte Potenz von 2, die kleiner ist als diese Dezimalzahl und notiert als erste (höchstwertige) Binärstelle eine 1. Das Ergebnis wird nun durch die nächst kleinere Potenz von 2 dividiert; das Resultat, also 0 oder 1, gibt die nächste Binärstelle an, usw. bis schließlich nach der Division durch $2^0 = 1$ das Verfahren abbricht Auf analoge Weise kann man eine in irgendeinem Zahlensystem angegebene Zahl in eine beliebige andere Basis umwandeln.

116 : 64 = 1
-64
52 : 32 = 1
-32
20 : 16 = 1
-16
4 : 8 = 0
-0
4 : 4 = 1
-4
0 : 2 = 0
-0
0 : 1 = 0

116 $_{\textbf{dez}}$ ***= 111 0100*** $_{\textbf{bin}}$

Konvertierung Dez -> Bin und Hex durch Restwertmethode

Die fortgesetzte Division einer Dezimalzahl durch Basis *B,* der Divisionsrest liefert die Koeffizienten a_0 bis a_n für die Darstellung dieser Zahl zur Basis *B*.
Beispiel: Es soll die Zahl 10172_{dez} mit der Restwertmethode in duale und hexadezimale Schreibweise umgewandelt werden.

Die Umwandlung in eine Dualzahl folgt aus der fortgesetzten Division durch 2:

10172 : 2 = 5086 Rest 0
5086 : 2 = 2543 Rest 0
2543 : 2 = 1271 Rest 1
1271 : 2 = 635 Rest 1
635 : 2 = 317 Rest 1
317 : 2 = 158 Rest 1
158 : 2 = 79 Rest 0
79 : 2 = 39 Rest 1
39 : 2 = 19 Rest 1
19 : 2 = 9 Rest 1
9 : 2 = 4 Rest 1
4 : 2 = 2 Rest 0
2 : 2 = 1 Rest 0
1 : 2 = 0 Rest 1

Ergebnis: $10172_{dez} = 10011110111100_{bin}$.

Fortgesetzte Division durch 16 liefert für die Umwandlung in eine Hexadezimalzahl:
10172 : 16 = 635 Rest 12 (= C)
635 : 16 = 39 Rest 11 (= B)
39 : 16 = 2 Rest (=7)
2 : 16 = 0 Rest (=2)

Ergebnis: $10172_{dez} = 27BC_{hex}$.

Konvertierung Dezimalbrüche -> Bin und Hex durch Restwertmethode

Es soll die Dezimalzahl 39,6875 als Dualzahl dargestellt werden.

Vorkommastellen, wie gehabt: Ergebnis: $39_{dez} = 100111_{bin}$.

Für die Konversion der Nachkommastellen, erhält man die Koeffizienten (= Ziffern) a_{-1}, a_{-2}, ... bei fortgesetzter Multiplikation mit der Basis als den ganzzahligen Teil (d. h. die Vorkommastellen des Multiplikationsergebnisses) erhält. Man lässt für den folgenden Schritt die Vorkommastellen weg und setzt dieses Verfahren fort, bis die Multiplikation eine ganze Zahl ergibt oder bis im Falle eines nicht abbrechenden Dezimalbruchs die gewünschte Genauigkeit erreicht ist.

0,6875 · 2 = 1,375 1 abspalten
0,3750 · 2 = 0,750 0 abspalten
0,7500 · 2 = 1,500 1 abspalten
0,5000 · 2 = 1,000 1 abspalten (fertig, Ergebnis ganzzahlig)

Ergebnis für die Nachkommastellen: $0{,}6875_{dez} = 0{,}1011_{bin}$.

Insgesamt hat man also: $39{,}6875_{dez} = 100111{,}1011_{bin} = 27{,}B_{hex}$.
Im Hexadezimalsystem ist die Rechnung viel kürzer in nur einem Schritt durchführbar:
0,6875 · 16 = 11,000 11=B abspalten

Umwandlung in Hexadezimalzahlen durch Zusammenfassen von Binärstellen

$$97_{dez} = \underbrace{0110}_{6}\ \underbrace{0001}_{1} = 0x61$$

$$531_{dez} = \underbrace{0010}_{2}\ \underbrace{0001}_{1}\ \underbrace{0011}_{3} = 0x213$$

$$4003_{dez} = \underbrace{1111}_{F}\ \underbrace{1010}_{A}\ \underbrace{0011}_{3} = = 0xFA3$$

- Bei Bedarf werden führende Nullen zugefügt.

Umwandlung von Hexadezimalzahlen in Binärzahlen und Dezimal

$$\mathit{2E4}_{\text{hex}} = \underbrace{\mathit{0010}}_{\mathit{2}}\,\underbrace{\mathit{1110}}_{\mathit{E}}\,\underbrace{\mathit{0100}}_{\mathit{4}}{}_{\text{bin}} = \mathit{1011100100}_{\text{bin}}$$

Führende Nullen werden unterdrückt.

Übungsaufgabe:

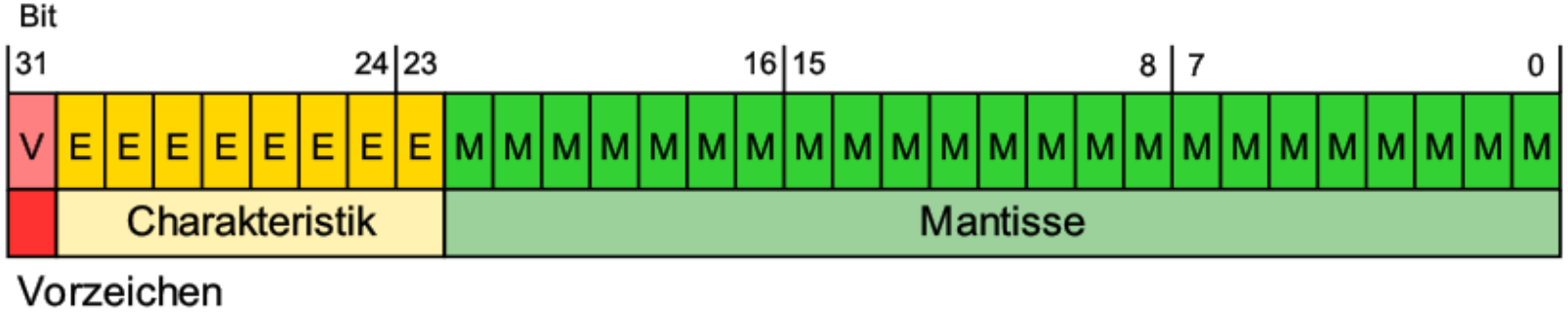
$$\mathit{1011100100}_{\text{bin}} = \mathit{??}_{\text{dez}}$$

Gleitkommazahlen

- Für die Darstellung sehr kleiner oder sehr großer Zahlen sind Festkommazahlen nicht geeignet. Günstiger und in der Computertechnik seit Zuse üblich sind Gleitkommazahlen in halblogarithmischer Darstellung.
- In Gleitkommadarstellung schreibt man Zahlen in der Form $m \cdot be$. Dabei ist die Mantisse m eine Festkommazahl und be ein Faktor mit Basis b und Exponent e.
- **Beispiele:** Dezimale Gleitkommazahlen mit der Basis $b = 10$:
  - $0{,}000112 = 1{,}12 \cdot 10^{-4}$, $7123458 = 7{,}123458 \cdot 10^{6}$, $-24{,}317 = -2{,}4317 \cdot 10^{1}$.
- Darstellung in ***Normalform**:* man schreibt die Mantisse m so, dass genau eine Stelle vor dem Dezimalpunkt eine von Null verschiedene Ziffer ist.
- Die Genauigkeit der Zahldarstellung hängt offenbar von der Stellenzahl der Mantisse ab, der darstellbare Zahlenbereich von der Basis und vor allem vom Exponenten.

Gleitkommazahlen nach IEEE 754 Standard

- Gleitkommazahlen mit Basis $b = 2$.
- Nach dem Standard3 IEEE 754 ist eine binäre Gleitkommazahl z mit Vorzeichenbit s, binärer Mantisse 1, f (in Normalform) und Exponent e wie folgt definiert:



- Exponent wird nicht direkt als e, sondern in Form der sog. Charakteristik $c = e+B$ gespeichert wird. Zum tatsächlichen Exponenten e wird also eine Verschiebung (Bias) B addiert, die so gewählt ist, dass der Nullpunkt für e in die Mitte des zur Verfügung stehenden Wertebereichs $[0,2B+1]$ verschoben wird. Auf diese Weise können Exponenten zwischen $e = -B$ (entsprechend $c = 0$) und $e = B+1$ (entsprechend $c = 2B+1$) dargestellt werden
- Bei einer kurzen Gleitkommazahl werden 32 Bit verwendet, wobei 8 Bit für die Charakteristik $c = e+127$ mit Wertebereich $[0,255]$ und Bias $B = 127$ zur Verfügung stehen.

Beispiel: Umwandlung Dez in Gleitkomma

1. Die Dezimalzahl wird in eine Binärzahl umgewandelt, ggf. mit Nachkommastellen.
2. Das Komma wird so weit verschoben, bis die Normalform erreicht ist. Bei Verschiebung um je eine Stelle nach links oder rechts wird der Exponent *e* der Basis 2 um eins erhöht bzw. um eins erniedrigt.
3. Das Vorzeichen *s* der Zahl (positiv: 0, negativ: 1) wird in das MSB des ersten Byte geschrieben.
4. Zum Exponenten *e* wird 127 addiert, das Ergebnis *c* = *e*+127 wird in binäre Form mit 8 Stellen umgewandelt. Ist der Exponent positiv, so hat das führende Bit von *c* den Wert 1, sonst hat es den Wert 0.
5. In die Bytes 2 (anschließend an das bereits für den Exponenten verwendete MSB), 3 und 4
6. werden schließlich die Nachkommastellen *f*0 *f*1 . . . *f*22 der Mantisse eingefügt.

Beispiel: 148,625 ist in eine binäre Gleitkommazahl umzuwandeln.

1. Schritt: $148{,}625_{dez} = 10010100{,}101_{bin}$
2. Schritt: $10010100{,}101 = 1{,}0010100101 \cdot 2^7$, Normalform ist erreicht, der Exponent ist e = 7
3. Schritt: Das Vorzeichen der Mantisse ist positiv, das MSB lautet also s = 0
4. Schritt: Charakteristik: c = e+127 = $134_{dez} = 10000110_{bin}$
5. Schritt: Ergebnis: $0100\ 0011\ 0001\ 0100\ 1010\ 0000\ 0000\ 0000_{bin} = 4314A000_{hex}$

Byte 1 Byte 2 Byte 3 Byte 4

Binäre Arithmetik

- Die Rechenregeln für Binärzahlen sind ganz analog zu den Rechenregeln für Dezimalzahlen definiert.
- Es genügt, sich auf die binäre Addition, Subtraktion, Multiplikation, Division und die logischen Operationen zu beschränken.

Übersicht der Grundrechenarten und logischen Operationen

Operation	Operator	Log. OP
Addition	+	
Subtraktion	-	
Multiplikation	X	
Division	:	
Kontravalenz	$\veebar$	XOR
Disjunktion	$\vee$	OR
Konjunktion	$\wedge$	AND
Inversion	¬	NOT

Anmerkung: Bei der Verwendung von „$\vee$“ für das logische Oder wird hingegen „$\veebar$“ für das Exklusiv-Oder verwendet.

Binäre Arithmetik, Wahrheitstabellen

- a XOR b = (a ∧¬ b) ∨ (¬a ∧ b)
- NOT: ¬1 = 0 ¬0 = 1

A	B	A ∨ B	A ∧ B	A ⊻ B
1	1	1	1	0
1	0	1	0	1
0	1	1	0	1
0	0	0	0	0

- **Beispiel:** Logische Verknüpfungen werden stellenweise ausgeführt:

$$\begin{array}{r} 10011 \\ \vee\ 10101 \\ \hline = 10111 \end{array} \qquad \begin{array}{r} 10011 \\ \wedge\ 10101 \\ \hline = 10001 \end{array} \qquad \begin{array}{r} \neg\ 10101 \\ \hline = 01010 \end{array}$$

Binäre Arithmetik, Addition

- Die Rechenregeln für die binäre Addition zweier Binärziffern lauten:
  - 0+0 = 0, 0+1 = 1, 1+0 = 1, 1+1 = 0 Übertrag 1 .

- **Beispiel:** Binäre Addition

  Die Aufgabe 11+14 = 25 soll in binärer Arithmetik gelöst werden:

```
   1011
+  1110
------------------
   111      Übertrag
= 11001     Ergebnis
```

  Die Aufgabe 151,875+27,625 = 179,5 soll in binärer Arithmetik gelöst werden:

```
  10010111.111
+     11011.101
------------------
    111111 11     Übertrag
= 10110011.100    Ergebnis
```

Binäre Arithmetik, Subtraktion

- Die Rechenregeln für die binäre Subtraktion zweier Binärziffern lauten:
  - 0−0 = 0, 1−1 = 0, 1−0 = 1, 0−1 = 1 Übertrag −1 .

- **Beispiel:** Direkte Binäre Subtraktion

  Die Aufgabe 13−11 = 2 soll in binärer Arithmetik gelöst werden:

1101	
− 1011	
−1	Übertrag
= 0010	Ergebnis

Einerkomplement/ Stellenkomplement

- Wird gebildet durch Invertierung aller Stellen
- Zur Darstellung eines negativen Werts geht ein Bit verloren, der Zahlenbereich umfasst nicht das MSB:
  - Most Significant Bit (MSB) 10110110
  - MSB repräsentiert das Vorzeichen: MSB 0→+, 1 → -

- **Beispiele:**

  Die Bitfolge 10000001 würde die negative Dezimalzahl -1 darstellen, da MSB = 1

  01010111 87_{dez}
  10101000 -87_{dez} (Einerkomplement)

Zweierkomplement

- Ermöglicht die Rückführung der Subtraktion auf die Addition
- In Computern technisch vergleichsweise einfach umzusetzen
- Wird gebildet durch Addition von 1 zum Einerkomplement

- **Beispiel:**

01010111	87_{dez}
10101000	-87_{dez} (Einerkomplement)
10101001	-87_{dez} (Zweierkomplement)

Wir rechnen: 87-87, mit Hilfe der Zweierkomplementmethode:

```
  01010111
 +10101001
 ----------
  11111110        Übertrag
= 00000000        Ergebnis      Übertrag an vorderster Stelle wird gestrichen
```

Beispiel 1: Subtraktion durch Addition des Zweierkomplements

87 - 46 = ?

46_{dez} : 00101110

Einerkomplement von 46_{dez} : 11010001

Addition von 1 1

Zweierkomplement -46_{dez} :11010010

87_{dez}: 01010111

Addition -46_{dez} :11010010

Übertrag 10101100

Ergebnis 00101001

87 - 46 = 41 = 0010 1001

Anmerkung: Tritt ein Übertrag auf, wird das MSB automatisch 0

Beispiel 2: Subtraktion durch Addition des Zweierkomplements

- negative Ergebnisse

2 - 4 = -2

4_{dez} : 00000100

Einerkomplement von 4_{dez} : 11111011

Addition von 1 1

Zweierkomplement -4_{dez} :11111100

2_{dez}: 00000010

Addition -4_{dez} : 11111100

Ergebnis 11111110

Einerkomplement: 00000001

Zweierkomplement: 00000010

Anmerkung: kein Übertrag bleibt das MSB also 1. Zur Bestimmung des Betrags des Ergebnisses bildet man abermals das Zweierkomplement. Dies liefert dann eine positive Zahl. Das negative Vorzeichen ist ja wegen MSB = 1 bereits bekannt.

Übungen

- Konvertieren Sie nach der Divisionsmethode 285_{dez} in die binäre bzw. hexadezimale Darstellung.
- Addieren Sie die Binärwörter 01010010 + 01100111.
- Subtrahieren Sie die Binärwörter 10101101 – 01101011 (direkte Methode).
- Subtrahieren Sie die Binärwörter 10101101 – 01101011 (Zweierkomplement).
- Wenden Sie die XOR-Operation auf 10110101 $\veebar$ 01101011 an.

Informationstheorie
- eine kurze Einführung

Claude Shannon, Begründer der Informationstheorie, US-amerikanischer Mathematiker und Elektrotechniker

- „A Mathematical Theory of Communication“, 1949
- Codierung von Information
- Übertragung codierter Information
- Math. Definition von Informationsgehalt
- Definition der Entropie von Information

Nachricht und Information

- Eine Nachricht lässt sich als Folge von Zeichen auffassen, die von einem Sender (Quelle) ausgehend, in irgendeiner Form einem Empfänger (Senke) übermittelt wird.
- Relevant vor allem in IT und Nachrichtentechnik
- Die Nachrichtenübermittlung erfolgt dabei im technischen Sinne über einen Kanal.
- Umformung der Originalnachricht in das für die Übertragung erforderliche Medium erforderlich, Codierer.
- Empfängerseite: Decodierer wandelt die empfangenen Signale in eine durch den Empfänger lesbare Form.
- Nachrichten lassen sich optimieren bzw. verkürzen .. (In heutiger Zeit u.U. nicht mehr so relevant, wegen Rechnerleistung, Netzwerkbandbreite, …)

Terminologie

- **Zeichen x:** Kleinste Einheit die nicht weiter unterteilt werden kann, z.B. ein einzelner Buchstabe ’A’, ein Bit ’1’, ein Symbol ♥ etc., oder auch landesspezifische Zeichen wie ä, ö, ç
- **Alphabet A:** Definierte endliche Menge von Zeichen, Zeichenvorrat, z.B. lateinisches, kyrillisches, binäres Alphabet A={A..Z}, A={A..Z,a..z}, A={0,1}
- **Nachricht bzw. Wort N:** Eine aus den Zeichen eines Alphabets gebildete Zeichenfolge.
- **Nachrichtenraum N(A) bzw. A*:** Die Menge aller Nachrichten, die mit den Zeichen eines Alphabets A gebildet werden können.
- **Beispiele: Für Alphabete:**

1. {a, b, c, . . . , z} Die Menge aller Kleinbuchstaben in lexikografischer Ordnung.
2. {0, 1, 2, . . . , 9} Die Menge der ganzen Zahlen 0 bis 9 mit der Ordnungsrelation „<“.
3. {♦,♥,♠,♣} Die Menge der Spielkartensymbole in der Reihenfolge ihres Spielwertes.
4. {2, 4, 6, . . . } Die unendliche Menge der geraden natürlichen Zahlen mit Ordnung „<“.
5. {0, 1} Die Binärziffern 0 und 1 mit 0 < 1.

Interpretation und Information

- Die Extraktion von Information aus einer Nachricht setzt eine Zuordnung (Abbildung) zwischen Nachricht und Information voraus, die Interpretation genannt wird.
- Die Interpretation einer Nachricht ist jedoch nicht unbedingt eindeutig, sondern subjektiv. Das gilt besonders für die Bedeutung, die eine Nachricht tragen kann. Ein und dieselbe Nachricht kann bisweilen auf verschiedene Weisen interpretiert werden.
- Zum Beispiel kann das Wort „Fuchsschwanz“ dienen. Dabei kann man je nach Kontext an den Schwanz eines Fuchses denken kann, oder aber an eine Säge.
- Die Interpretationsvorschrift muss auch nicht unbedingt offensichtlich sein, Verschlüsselung.
- Die Lehre von der Verschlüsselung von Nachrichten oder Kryptologie entwickelte sich als ein Teilgebiet der Informatik mit großer praktischer Bedeutung.
- „Information“ ist also ein sehr vielschichtiger Begriff, der mathematisch nicht einfach und vor allem auch nicht allgemein in all seinen Facetten fassbar ist.
- Daher sind im Sinne der Informatik Informationen, im Gegensatz zu Nachrichten, nicht exakt definierbare abstrakte Objekte.

Codierung von Buchstaben, Ziffern und Sonderzeichen

- Für die Kodierung lateinischer Zeichen wird fast nur noch bei Großrechnern die 8-Bit-Kodierung **EBCDIC** verwendet, die IBM parallel zu ASCII für sein System/360 entwickelte.
- Der **American Standard Code for Information Interchange (ASCII)**, ab 1963, ist eine 7-Bit-Zeichenkodierung. Sie dient als Grundlage für spätere, auf mehr Bits basierende Kodierungen für Zeichensätze.
- Die Zeichenkodierung definiert 128 Zeichen, bestehend aus 33 nicht druckbaren sowie den folgenden 95 druckbaren Zeichen, beginnend mit dem Leerzeichen.
- Druckbare Zeichen: lateinisches Alphabet in Groß- und Kleinschreibung, die zehn arabischen Ziffern sowie einige Interpunktionszeichen, wie Satzzeichen und andere Sonderzeichen.
- Die nicht druckbaren Steuerzeichen enthalten Ausgabezeichen wie Zeilenvorschub oder Tab, Übertragungsende, etc.
- Heute wird fast immer zur Erweiterung von ASCII ein 8-Bit-Code verwendet, 256 Zeichen.
- Auch 8-Bit-Codes bieten zu wenig Platz, um alle Zeichen, die gebraucht werden, gleichzeitig unterzubringen. Es wurden **Zeichensatztabellen** definiert, die den darstellbaren Zeichen und Steuerzeichen Zahlenwerte zuordnen, auch Codetabellen oder Codepages.
- Beispiele für Codepages: ***https://de.wikipedia.org/wiki/Zeichensatztabelle***.

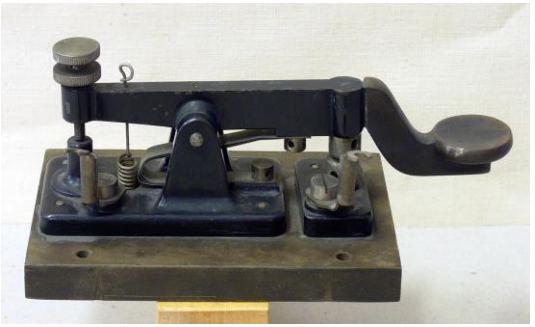
Beispiel erweiterter ASCII-Code für die Codepage 850

000	NUL	033	!	066	B	099	c	132	ä	165	Ñ	198	ã	231	þ
001	Start Of Header	034	"	067	C	100	d	133	à	166	ª	199	Ã	232	Þ
002	Start Of Text	035	#	068	D	101	e	134	å	167	º	200	╚	233	Ú
003	End Of Text	036	$	069	E	102	f	135	ç	168	¿	201	╔	234	Û
004	End Of Transmission	037	%	070	F	103	g	136	ê	169	®	202	╩	235	Ù
005	Enquiry	038	&	071	G	104	h	137	ë	170	¬	203	╦	236	ý
006	Acknowledge	039		072	H	105	i	138	è	171	½	204	╠	237	Ý
007	Bell	040	(	073	I	106	j	139	ï	172	¼	205	═	238	¯
008	Backspace	041	)	074	J	107	k	140	î	173	¡	206	╬	239	´
009	Horizontal Tab	042	*	075	K	108	l	141	ì	174	«	207	¤	240	-
010	Line Feed	043	+	076	L	109	m	142	Ä	175	»	208	ð	241	±
011	Vertical Tab	044	,	077	M	110	n	143	Å	176	░	209	Ð	242	‗
012	Form Feed	045	-	078	N	111	o	144	É	177	▒	210	Ê	243	¾
013	Carriage Return	046	.	079	O	112	p	145	æ	178	▓	211	Ë	244	¶
014	Shift Out	047	/	080	P	113	q	146	Æ	179	│	212	È	245	§
015	Shift In	048	0	081	Q	114	r	147	ô	180	┤	213	ı	246	÷
016	Delete	049	1	082	R	115	s	148	ö	181	Á	214	Í	247	¸
017	-- frei --	050	2	083	S	116	t	149	ò	182	Â	215	Î	248	°
018	-- frei --	051	3	084	T	117	u	150	û	183	À	216	Ï	249	¨
019	-- frei --	052	4	085	U	118	v	151	ù	184	©	217	┘	250	·
020	-- frei --	053	5	086	V	119	w	152	ÿ	185	╣	218	┌	251	¹
021	Negative Acknowledge	054	6	087	W	120	x	153	Ö	186	║	219	█	252	³
022	Synchronous Idle	055	7	088	X	121	y	154	Ü	187	╗	220	▄	253	²
023	End Of Transmission Block	056	8	089	Y	122	z	155	ø	188	╝	221	¦	254	■
024	Cancel	057	9	090	Z	123	{	156	£	189	¢	222	Ì	255	
025	End Of Medium	058	:	091	[	124	|	157	Ø	190	¥	223	▀		
026	Substitude	059	;	092	\	125	}	158	×	191	┐	224	Ó		
027	Escape	060	<	093	]	126	~	159	ƒ	192	└	225	ß		
028	File Seperator	061	=	094	^	127	⌂	160	á	193	┴	226	Ô		
029	Group Seperator	062	>	095	_	128	Ç	161	í	194	┬	227	Ò		
030	Record Seperator	063	?	096	`	129	ü	162	ó	195	├	228	õ		
031	Unit Seperator	064	@	097	a	130	é	163	ú	196	─	229	Õ		
032		065	A	098	b	131	â	164	ñ	197	┼	230	µ		

Der Unicode

- Der Unicode ist eine Erweiterung des ASCII-Codes, mit dem sämtliche Zeichensysteme in den bekannten Schriftkulturen ausgedrückt werden können (Erste Version 1991). https://unicode.org/main.html.
- Unicode als internationalen Standard (ISO 10646) und integriert in Version 12.0 im März 2019 inzwischen 150 Schriftsysteme mit 137292 Zeichen und Emoji.
- ‚Alle' modernen Betriebssysteme, Programmiersprachen und Anwendungen basieren inzwischen auf dem Unicode-System. Dazu gehören zahlreiche Sonderzeichen, mathematische Formelzeichen, fernöstliche Symbole und Angaben über die Schreibrichtung und Stellung der Zeichen. Details findet man auf der Unicode-Webseite 2.
- Die Umsetzung des Unicode-Zeichensatzes in konkrete Codierungen erfolgt mit dem Unicode Transformation Format (UTF). Dieses existiert in verschiedenen Ausprägungen.
- UTF-8 verwendet zur Codierung zwischen 1 und 4 Byte. Vorteil, dass die 1-Byte Codierung mit dem 127-Bit ASCII Code übereinstimmt und ist bei Verwendung von lateinischen Zeichen sehr effizient (im Vergleich zu 2 Byte bei UTF-16).
- UTF-32 verwendet beispielsweise immer genau 4 Byte, viel Speicherplatz ist erforderlich (vgl. 1 Byte bei ASCII)
- UTF-16 benötigt pro Zeichen 2 Byte (für die „gebräuchlichen" Symbole) oder 4 Byte (für die „exotischeren" Symbole).
- flexiblere Varianten UTF-16 (Windows, OS-X) und UTF-8 (WWW, Email, Unix) sind weiter verbreitet
- **Erst durch neuere Speicher- und Rechnertechnologien möglich.**

Weitere bekannte Codierungen, Beispiele



Von Hannes Grobe (talk) - Schulhistorische Sammlung Bremerhaven, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=4626456

- Morse–Codierung (Samuel Morse (1791 - 1872), war ein gebräuchlicher Code zur telegrafischen Übermittlung von Buchstaben, Ziffern und weiterer Zeichen. Wird heute noch in der Luft- und Schifffahrt verwendet, ansonsten kaum noch Bedeutung.

- Als Strichcode, oder Barcode wird eine optoelektronisch lesbare Schrift (Abbildung von Daten in binären Symbolen) bezeichnet, die aus verschieden breiten, parallelen Strichen und Lücken besteht. in den 1970er Jahren begann die Verbreitung in Supermarktketten.

- Der QR-Code (englisch Quick Response) besteht aus einer quadratischen Matrix aus schwarzen und weißen Quadraten, die die kodierten Daten binär darstellen (Firma Denso Wave, 1994). entwickelt wurde. Die Daten im QR-Code sind durch einen fehlerkorrigierenden Code erweitert. Dadurch wird der Verlust von bis zu 30 % des Codes toleriert, d. h., er kann auch dann noch dekodiert werden. Die Verwendung des QR-Codes ist lizenz- und kostenfrei.

Multimediale Daten

- Grafiken, Bilder
  - Rastergrafik, NetPBM-Format (Network Portable BitMap)
    - Grafik wird aus einer Folge einzelner Rasterpunkte dargestellt und für jeden Punkt einzeln der Farbwert und eventuelle Transparenz definiert (abhängig von der Farbtiefe)
  - Ein Format für Vektorgrafik ist das SVG-Format für Scalable Vector Graphics. Hier können mit allgemeinen Pfaden, Formen und Texten samt weiterführenden Transformationen und Effekten als Text oder mit Programmen erfasst werden.
- Audio, wie WAVE, FLAC-Format (Free Lossless Audio Codec) und MP-3
- Videodaten sind typischerweise eine Kombination von bewegtem Bild und Tondaten. Einzelbildspeicherung bei Motion JPEG, Animated Portable Network Graphics oder bei Animated GIF-Format. Videoformate wie MP4 speichern nur wenige Bilder komplett ab, und versuchen durch Speichern von Unterschieden wiederum kombiniert mit einer Reduktion von für Menschen unwichtigen Informationen den Speicherbedarf deutlich zu reduzieren.

Anwendungsbereiche von Codierung bzw. Codes

- Physische Übertragung von Information
  - Leitungscodierung, Kanalcodierung
  - Fehlerkorrigierende Übertragung von Information
  - (Effiziente) Persistierung von Information
  - Dateisysteme, Kompressionsverfahren
  - Verschlüsselung, sym.-/ asymmetrische Verfahren

Codesicherung

- **Problem:** Störungen können dazu führen, dass einzelne Bits nicht richtig übertragen werden (0 statt 1/ 1 statt 0).
- **Lösung:** Durch Hinzufügen zusätzlicher Bits bzw. geschickter Auswahl der Codeworte kann die Störanfälligkeit eines Codes erhöht werden, so dass eine Fehlererkennung bzw. eine Fehlerkorrektur möglich ist.

Grössenordnungen

$1\mathrm{K} = 1024 = 2^{10}$ (K = kilo)

$1\mathrm{M} = 1024 \cdot 1024 = 2^{20}$ (M = mega)

$1\mathrm{G} = 1024 \cdot 1024 \cdot 1024 = 2^{30}$ (G = giga)

$1\mathrm{T} = 1024 \cdot 1024 \cdot 1024 \cdot 1024 = 2^{40}$ (T = tera)

$1\mathrm{P} = 1024 \cdot 1024 \cdot 1024 \cdot 1024 \cdot 1024 = 2^{50}$ (P = Peta)

$1\mathrm{E} = 1024 \cdot 1024 \cdot 1024 \cdot 1024 \cdot 1024 \cdot 1024 = 2^{60}$ (E = exa)

Für Längen, Zeiten:

$1\mathrm{m} = 10^{-3}$ (m = milli)

$1\mu = 10^{-6}$ (μ = mikro)

$1\mathrm{n} = 10^{-9}$ (n = nano)

$1\mathrm{p} = 10^{-12}$ (p = pico)

$1\mathrm{f} = 10^{-15}$ (f = femto)

Datentypen und Wertebereiche von Zahlen

Typ	Länge		Max. Wert
Unsigned short	16 Bit		0 – 65 536
Short	16 Bit	1 Vorzeichenbit	-32 768 - 32 768
Unsigned int	32 Bit		-2,147,483,648 to 2,147,483,647
Int	32 Bit	1 Vorzeichenbit	0 to 4,294,967,295
Unsigned long	64 Bit		-922,3372,0368,5477,5808 to 922,3372,0368,5477,5807
Long	64 Bit	1 Vorzeichenbit	0 to 1844,6744,0737,0955,1615
Unsigned char	1 Byte		0 to 255
Char	1 Byte	1 Vorzeichenbit	-128 to 127
Float	32 Bit	6 Dezimalstellen	1.2E-38 to 3.4E+38
Double	64 Bit	15 Dezimalstellen	2.3E-308 to 1.7E+308

- Short und Int hängen von der Wortgrösse ab, je nach Hardware Architektur können das 8, 16, 32 oder 64 Bit sein.
- Je nach Hardware Architektur und Programmiersprache können kleine Unterschiede existieren.