STUDIUM PLUS
DUALES STUDIUM

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Zahlensysteme und Textcodierung

Inhalt

STUDIUM PLUS DUALES STUDIUM
THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Summenformel Positionssysteme

Ein Positionssystem mit der Basis B ist ein Zahlensystem, in dem eine Zahl x nach Potenzen von B zerlegt wird.

Eine natürliche Zahl *x* wird durch folgende Summe dargestellt:

$$x = \sum_{i=0}^{n-1} b_i * B^i$$

Es gilt:

- B = Basis des Zahlensystems ($B \in \mathbb{N}, B \geq 2$)
- b = Ziffern ($b_i \in \mathbb{N}_0, 0 \leq b_i < B$)
- N = Anzahl der Stellen

Übung

$(11001)_2 = (\quad ? \quad)_{10}$

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

$16 \quad + \quad 8 \qquad\qquad + 1 \qquad = 25_{10}$

$(315)_8 = (\quad ? \quad)_{10}$

$3 * 8^2 + 1 * 8^1 + 5 * 8^0$

$3 * 64 + 1 * 8 + 5 * 1 \qquad = 205_{10}$

$(C9)_{16} = (\quad ? \quad)_{10}$

$12 * 16^1 + 9 * 16^0$

$192 \quad + \quad 9 \qquad = 201_{10}$

Konvertierung

- Konvertieren vom Dezimalsystem in andere Positionssysteme
  - Methode: Division mit Rest

Methode: Division mit Rest

Teilen wir eine natürliche Zahl z (Dividend) durch eine andere natürliche Zahl $d \neq 0$ (Divisor), erhalten wir einen Quotienten q und einen Rest r.

Es gilt dann offensichtlich der Zusammenhang:
`z = q · d + r` mit `0 ≤ r < d`

`z` = Zahl, `q` = Quotient (entspr. Ergebnis der Division),
`d` = Divisor (der „Teiler“), `r` = Rest

Wir unterscheiden

- die Operation des exakten Dividierens wird als `div` bezeichnet
- die Operation, die zwei Zahlen den Divisionsrest zuordnet, wird als `mod` bezeichnet.

Beispielsweise gilt:

- `5000 div 16 = 312` und
- `5000 mod 16 = 8` `denn: 5000 : 16 = 312 Rest 8`

Die obige Gleichung können wir unter Zuhilfenahme der Operatoren `div` und `mod` allgemein schreiben als:

- `z = (z div d) · d + (z mod d)`

Konvertieren vom Dezimalsystem in andere Positionssysteme

Folgender Algorithmus kann zur Umwandlung einer Dezimalzahl `x` in ein Zahlensystem mit der Basis `n` verwendet werden:

1. `x / n = y Rest z`
2. Mache `y` zum neuen `x` und fahre wieder mit Schritt 1 fort, wenn dieses neue `x` ungleich `0` ist, ansonsten fahre mit Schritt 3 fort.
3. Die ermittelten Reste `z` von unten nach oben nebeneinander geschrieben ergeben dann die entsprechende Zahl im Ziel-Positionssystem mit der Basis `n`

Konvertieren vom Dezimalsystem ins Binärsystem

Beispiel: $(76)_{10} \rightarrow (?)_2$

Zur Umrechnung dividiert man die Zahl in Dezimaldarstellung sukzessive durch 2 und merkt sich die Reste bei diesen Divisionen. In umgekehrter Folge eingesammelt ergibt dies die Zahl in Binärdarstellung.

$(\qquad\qquad)_2$

Konvertieren vom Dezimalsystem ins Binärsystem

Beispiel: $(76)_{10} \rightarrow (?)_2$

76 / 2 = 38; Rest 0
38 / 2 = 19; Rest 0
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

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

Konvertieren vom Dezimalsystem ins Hexadezimalsystem

Unser Verfahren zur Umrechnung funktioniert auch für die Basen 8 und 16

Beispiel: $(24572)_{10} \rightarrow (?)_{16}$

24572 / 16 = 1535; Rest 12
1535 / 16 = 95; Rest 15
95 / 16 = 5; Rest 15
5 / 16 = 0; Rest 5

$(5\ F\ F\ C)_{16}$

Konvertierung

- Konvertierung von anderen Systemen in das Dezimalsystem
  - Methode: Horner-Schema

Konvertierung von anderen Systemen in das Dezimalsystem via Horner

- Eine im Positionssystem mit Basis B dargestellte natürliche Zahl n:

$$n = \sum_{i=0}^{N} b_i \cdot B^i$$

- lässt sich mit Hilfe des Hornerschemas wie folgt darstellen:

$$n = (\ldots(((b_N \cdot B + b_{N-1}) \cdot B + b_{N-2}) \cdot B + b_{N-3}) \cdot B + \ldots + b_1) \cdot B + b_0$$

- Das Horner-Schema ist ein Umformungsverfahren für Polynome
  - Polynomdivision, Berechnung von Nullstellen und von Ableitungen
  - Ermöglicht damit auch Umrechnungen von beliebigen Zahlensystemen

- Mit Hilfe dieser Darstellung können Konvertierungen in das Dezimalsystem durchgeführt werden

Horner-Schema: $(11011101)_2 \rightarrow (?)_{10}$

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

$B_n \quad B_{n-1} \quad B_{n-2} \quad B_{n-3} \quad B_{n-4} \quad B_{n-5} \quad B_{n-6} \quad B_{n-7}$

Wir spiegeln die Formel

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

$B_{n-7} \qquad\qquad\qquad\qquad B_n$

Wir verwenden Hilfsvariable t_0 bis t_n

STUDIUM PLUS
DUALES STUDIUM

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

$$p(x) = b_0 + x(b_1 + x(\ldots + x(b_{n-2} + x(b_{n-1} + xb_n)))\ldots)$$

$$t_0 := \boxed{b_n}$$

$$t_1 := b_{n-1} + xb_n = \boxed{b_{n-1} + xt_0}$$

$$t_2 := b_{n-2} + x(b_{n-1} + xb_n) = \boxed{b_{n-2} + xt_1}$$

$$t_3 := b_{n-3} + x(b_{n-2} + x(b_{n-1} + xb_n)) = \boxed{b_{n-3} + xt_2}$$

$$t_n := b_0 + x(b_1 + x(\ldots + x(b_{n-2} + x(b_{n-1} + xb_n)))\ldots) = \boxed{b_0 + xt_{n-1}}$$

Horner-Schema: $(11011101)_2 \rightarrow (221)_{10}$

STUDIUM PLUS DUALES STUDIUM

THM TECHNISCHE HOCHSCHULE MITTELHESSEN

$$(11011101)_2$$

$$p(2) = 1 + 2(0 + 2(1 + 2(1 + 2(1 + 2(0 + 2(1 + 2 \times 1))))))$$

$t_0 := 1$

$t_1 := 1 + 2 \times 1 = 1 + 2 \times t_0$ 3

$t_2 := 0 + 2(1 + 2 \times 1) = 0 + 2 \times t_1$ 6

$t_3 := 1 + 2(0 + 2(1 + 2 \times 1)) = 1 + 2 \times t_2$ 13

$t_4 := 1 + 2(1 + 2(0 + 2(1 + 2 \times 1))) = 1 + 2 \times t_3$ 27

$t_5 := 1 + 2(1 + 2(1 + 2(0 + 2(1 + 2 \times 1)))) = 1 + 2 \times t_4$ 55

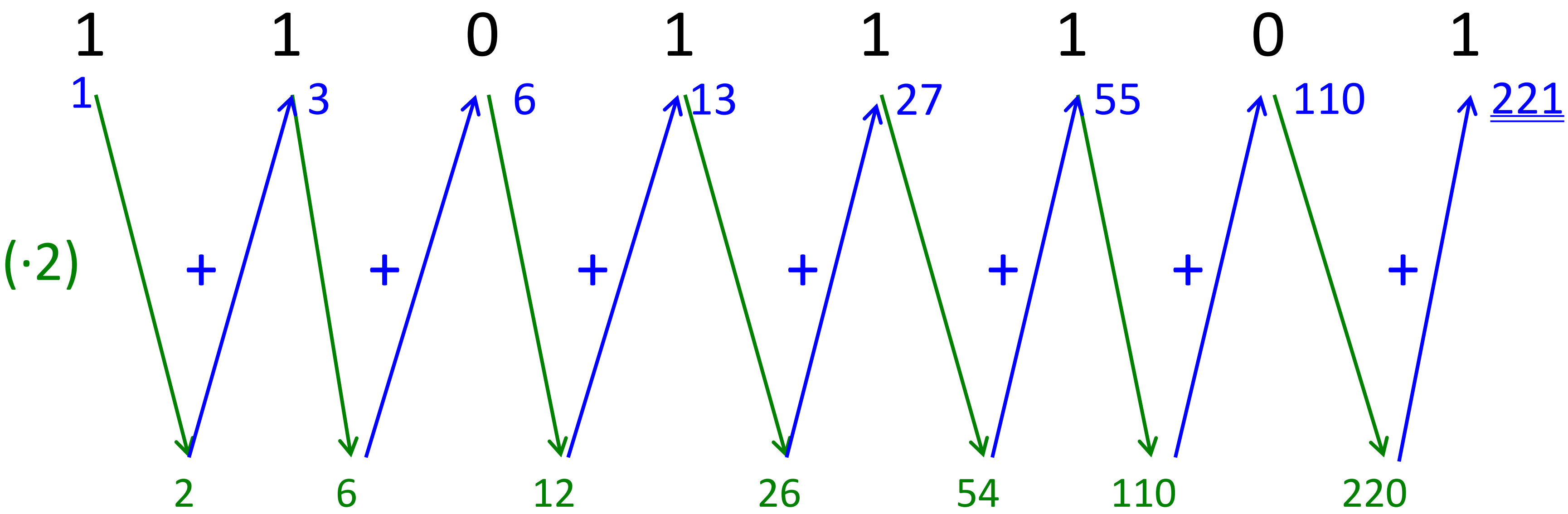
$t_6 := 0 + 2(1 + 2(1 + 2(1 + 2(0 + 2(1 + 2 \times 1))))) = 0 + 2 \times t_5$ 110

$t_7 := 1 + 2(0 + 2(1 + 2(1 + 2(1 + 2(0 + 2(1 + 2 \times 1)))))) = 1 + 2 \times t_6$ 221

Horner-Schema: $(11011101)_2 \rightarrow (221)_{10}$

STUDIUM PLUS
DUALES STUDIUM

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN



Für die blauen Werte gilt. $1=t_0$, $3=t_1$, $6=t_2$,….,$221=t_7$

Konvertierung Zusammenfassung

- Konvertieren vom Dezimalsystem in andere Positionssysteme
  - Methode: Division mit Rest
- Konvertierung von anderen Systemen in das Dezimalsystem
  - Methode: Summenformel
  - Methode: Horner-Schema

Inhalt

Lernziel

- Sie kennen die Möglichkeiten der Vorzeichenbehandlung
- Sie können die Methode 2er-Komplement für negative ganze Zahlen erläutern

Darstellung ganzer Zahlen – Vorzeichendarstellung (Einerkomplement)

- Bisher betrachtet: Natürliche Zahlen (positive, ganze Zahlen)
- Ganze Zahlen schließen negative Zahlen mit ein
- Folglich sind der absolute Zahlenwert und das Vorzeichen von Bedeutung
- Naheliegende Möglichkeit?
  - Das Vorzeichen wird über das führende Bit ausgedrückt.
    0 → positive Zahl; 1 → negative Zahl
  - Man spricht von der **Vorzeichendarstellung**

Quelle: http://www.hki.uni-koeln.de/

- Beispiel: Vorzeichendarstellung in 4 Bit:

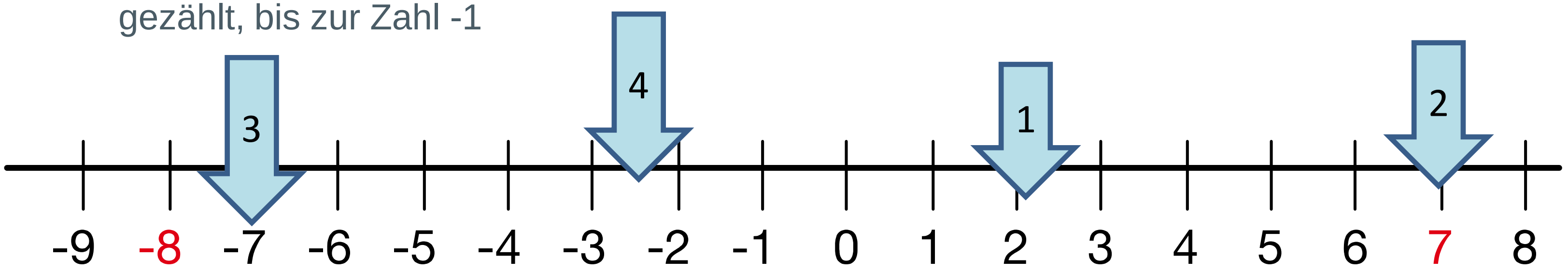
0000 = +0	0100 = +4	1000 = -0	1100 = -4
0001 = +1	0101 = +5	1001 = -1	1101 = -5
0010 = +2	0110 = +6	1010 = -2	1110 = -6
0011 = +3	0111 = +7	1011 = -3	1111 = -7

Nachteile der Vorzeichendarstellung

- Bei näherer Betrachtung hat die Vorzeichendarstellung eine Reihe von Nachteilen:
  - Die Zahl 0 wird durch zwei verschiedene Bitfolgen darstellt (als '+0' und als '-0').
  - Das Rechnen ist komplizierter geworden. Es ist nicht mehr so einfach möglich, Zahlen untereinander zu schreiben und zu addieren.
- Die Zweierkomplementdarstellung ist eine Variante, die diese Probleme vermeidet und deshalb zu einer gebräuchlichen Darstellung geworden ist.

Die Zweierkomplementdarstellung

- Zweierkomplementdarstellung ist die gebräuchliche interne Repräsentation ganzer positiver und negativer Zahlen
- Beispiel (4 Bit): Zahlenbereich von $2^4 = 16$ abdeckbar.
  - Der Bereich ist frei wählbar, also z.B. die 16 Zahlen von -8 bis +7
  - Um Dezimalzahlen abzubilden, wird von 0 beginnend aufwärts gezählt, bis die obere Grenze (+7) erreicht ist. Anschließend wird an der unteren Grenze (-8) fortgefahren und aufwärts gezählt, bis zur Zahl -1



1000 = -8	1100 = -4	0000 = 0	0100 = 4
1001 = -7	1101 = -3	0001 = 1	0101 = 5
1010 = -6	1110 = -2	0010 = 2	0110 = 6
1011 = -5	1111 = -1	0011 = 3	0111 = 7

Quelle: http://www.hki.uni-koeln.de/

Die Zweierkomplementdarstellung

- Nun offenbart sich, wieso für 4 Bit der Bereich von -8 bis +7 gewählt wurde:
  - Bei dem mit 0 beginnenden Hochzählen wird bei der neunten Bitfolge zum ersten Mal das erste Bit zu 1 (Bei den Zweierkomplementzahlen stellt das erste Bit das Vorzeichen dar; d.h. man springt ab der 9. Bitfolge in den negativen Bereich).
  - Weiterer Vorteil: Die „0“ kommt nur einmal vor
  - Darstellbar Wertebereich mit dem Zweierkomplement: Zahlen von -2^{N-1} bis $2^{N-1}-1$
  - Zweierkomplementzahlen werden auch als *signed binary numbers* oder *signed integers* bezeichnet

Vergleich

Einerkomplement

1000 = -0	1100 = -4
1001 = -1	1101 = -5
1010 = -2	1110 = -6
1011 = -3	1111 = -7

Zweierkomplement

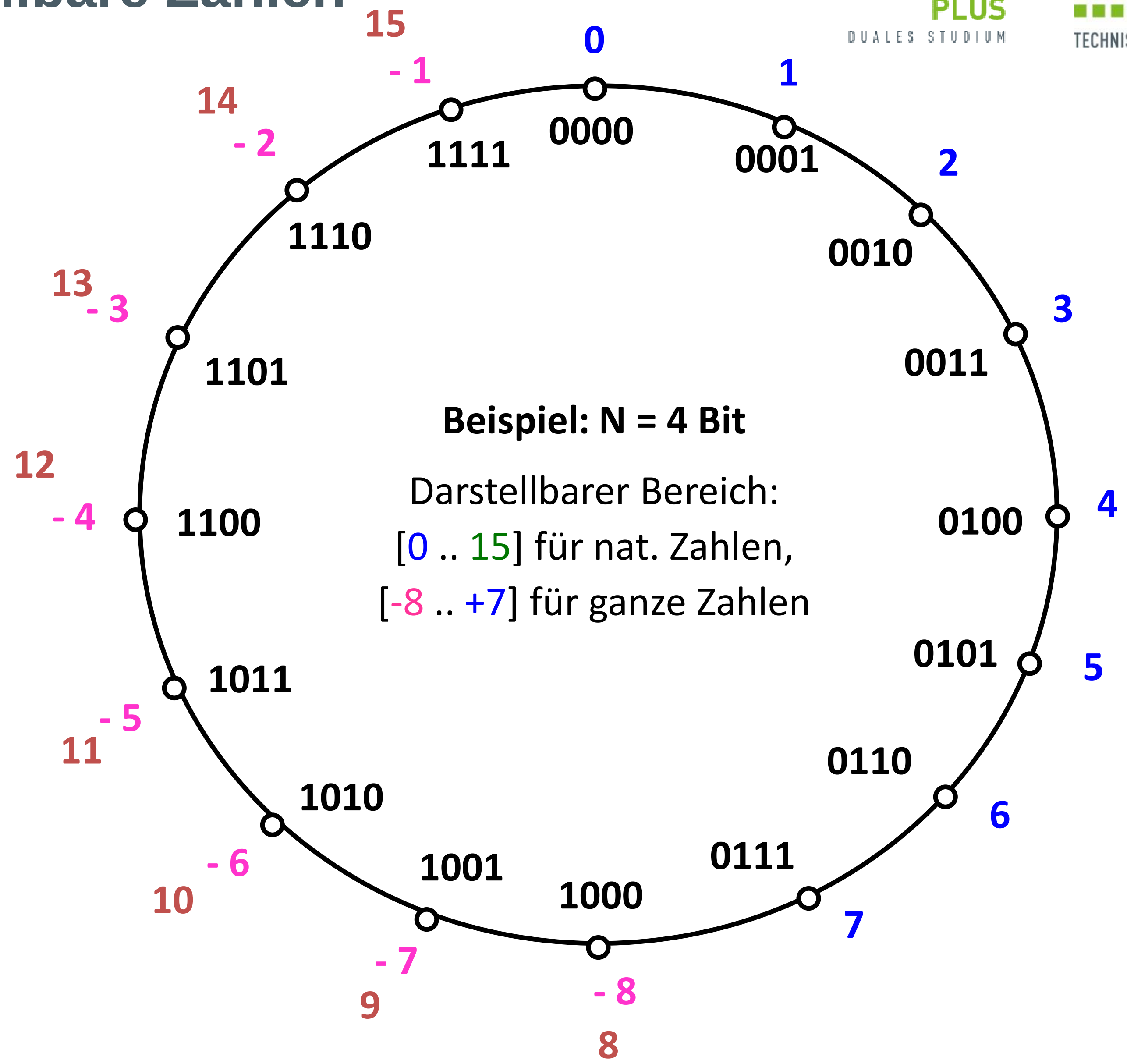
1000 = −8	1100 = −4
1001 = −7	1101 = −3
1010 = −6	1110 = −2
1011 = −5	1111 = −1

Berechnung des Zweierkomplements

- Das Einerkomplement lässt sich durch bitweises Vertauschen der Werte 0 und 1 erhalten (Invertierung)
- Das Zweierkomplement resultiert aus Bildung des Einerkomplements und anschließender Addition von 1
- Algorithmus:
  - Die gewünschte Zahl wird binär codiert
  - Die binäre Darstellung wird invertiert
  - Zur Invertierten Darstellung wird (binär) eins addiert
- Beispiel: Zweierkomplementdarstellung von -5

1. Binärcodierung	$(0101)_2 = (5)_{10}$
2. Invertierung	$(1010)_2$
3. Addition +1	$(1011)_2$

Darstellbare Zahlen



Inhalt

Addition von Binärzahlen

Die Addition von Binärzahlen folgt direkt der bekannten schriftlichen Addition von Dezimalzahlen.

```
      0111 0000          (=112₁₀)
+     0101 0101          (= 85₁₀)
      1100 0101          (=197₁₀)
```

Aber:

```
      1100 0000          (=192₁₀)
+     0101 0101          (= 85₁₀)
    1 0001 0101          (=277₁₀)
```

Überlauf!

Bei dieser Addition wird der darstellbare Bereich verlassen!
Der Überlauf wird ignoriert.
Das kann bei jeder Wortbreite passieren – bei höherer Wortbreite erst bei größeren Zahlen.

Subtraktion von Binärzahlen

Die Subtraktion wird im Binärsystem auf eine Negation (Zweier Komplement) mit anschließender Addition zurückgeführt.

Beispiel: $(112)_{10} - (85)_{10} = (27)_{10}$

```
   01110000   (= 112₁₀)
   01010101   (=  85₁₀)
Zu          (=  -85₁₀) kommen wir

    10101010   =Einerkomplement
    10101011   =Zweierkomplement

   01110000   (= 112₁₀)
  +10101011   (= -85₁₀)
  ---------
  100011011  (= 27₁₀)
```

Die erste Bit-Position (Endübertrag) bleibt unberücksichtigt, da bei der Umwandlung des Subtrahenden nur ein achtstelliger Wert berücksichtigt wurde.

Woher kommt der Name?

Sei B die die Basis des Zahlensystem
Im Falle B= 2

Dann kann man das Einer-Komplement so schreiben: (B-1)-Komplement

Dann kann man das Zweier-Komplement so schreiben: (B)-Komplement

Multiplikation und Division von Binärzahlen

- Multiplikation und Division funktionieren wie bei den Dezimalzahlen
- Multiplikation: Produkte werden untereinander geschrieben und addiert, eventuelle Überträge eine Stelle nach links mitgenommen
- Division:
  - Divisor unter erste Stelle des Dividenden schreiben und subtrahieren
  - Anschließend Ergebnisse von links nach rechts aufschreiben
  - Passt der Divisor mindestens einmal in die ersten Stellen des Dividenden, dann entsteht beim Ergebnis eine 1. Falls nicht, entsteht beim Ergebnis eine 0 und es muss eine weitere Stelle des Dividenden nach unten gezogen werden

Inhalt

STUDIUM PLUS
DUALES STUDIUM
THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Lernziel

- Sie kennen die Möglichkeiten rationale und reelle zu kodieren
- Sie können Rechner-interne Darstellung nach IEEE-754 erläutern

Darstellung reeller Zahlen

- In einem endlichen Intervall liegen nur endlich viele ganze Zahlen, aber unendlich viele rationale bzw. reelle Zahlen.
- Zwei zentrale Fragen:
  - Wie lässt sich das Komma darstellen, wo 0 und 1 aus Kodierungssicht bereits belegt sind?
  - Wie lässt sich der unendliche Zahlenbereich zwischen zwei reellen Zahlen mit endlichen vielen Ziffern (Bits) darstellen?

- Lösung: **Gleitpunktdarstellung** (Kommastelle ist Bestandteil der Zahl)
- Wissenschaftliche Notation gibt die **Kommaposition** über einen **Exponenten** an.
- Beispielsweise kann die Zahl **3,84** auch so geschrieben werden:
  - $\mathbf{384 \cdot 10^{-2}}$
  - $\mathbf{0{,}0384 \cdot 10^{2}}$
  - $\mathbf{38{,}4 \cdot 10^{-1}}$

Gleitpunktdarstellung

- In der Praxis werden fast ausschließlich Gleitpunktzahlen für die Speicherung "reeller Zahlen" verwendet.
  - Beachte: Echte "reelle" Zahlen lassen sich nie genau in einem Computer speichern, da es für sie definitionsgemäß keine endliche Darstellung gibt.
  - Die sog. *"real numbers"* im Computer sind mathematisch gesehen immer rationale Zahlen, d.h. Näherungswerte für reelle Zahlen.

- Mit der Gleitpunktdarstellung möchte man:
  - ein möglichst großes Intervall "reeller" Zahlen umfassen,
  - die Genauigkeit der Darstellung an die Größenordnung der Zahl anpassen: bei kleinen Zahlen sehr hoch, bei großen Zahlen niedrig.

- Daher speichert man neben dem reinen Zahlenwert (Mantisse) auch einen Exponenten (i.d.R. zur Basis 2 oder 10), der die Kommaposition in der Zahl angibt.

Gleitpunktdarstellung

- Die Gleitpunktdarstellung besteht aus folgenden Komponenten:
  - dem Vorzeichenbit**: V** (gibt an, ob die vorliegende Zahl positiv oder negativ ist)
  - der Mantisse**: M** (besteht aus Binärziffern $m_1 \ldots m_n$ und gibt den Wert der vorliegenden Zahl an)
  - dem Exponenten**: E**. (Der Exponent ist eine Binärzahl, zum Beispiel im Bereich -127 bis +127, die angibt, mit welcher Potenz einer **Basiszahl b** die vorliegende Zahl zu multiplizieren ist)

- Das Tripel (V, M, E) wird als $\mathbf{V \cdot M \cdot b^E}$ interpretiert.

Normierte Gleitpunktzahlen

- Eine zur Basis 2 **normierte Gleitpunktzahl** ist eine solche, bei der der Exponent so gewählt wird, dass die Zahl in der Form

  $\pm 1.m_1 m_2 ... m_{n-1} m_n * 2^E$ dargestellt werden kann.

- **Motivation für eine Normierung:** Es soll für jede darstellbare Zahl genau eine Darstellung als Gleitpunktzahl existieren.
- Da durch die Normierung im Binärsystem immer eine 1 vor dem Komma stehen muss, kann man diese auch weglassen.
- In der Mantisse werden dann nur noch die Stellen hinter dem Komma notiert! (Die führende "1," steht also gedacht links vor der Mantisse.).
- Durch Einsparung der führenden 1 können die Mantissenbits optimal ausgenutzt werden, was besonders bei unendlich vielen Nachkommastellen (periodische Zahlen) eine höhere Rechengenauigkeit ermöglicht.

Gleitpunktzahlen nach IEEE 754

- Nach IEEE 754 (Institute of Electrical and Electronics Engineers) normierte Gleitpunktzahlen verwenden b = 2 als Basiszahl.

- Zwei von IEEE verabschiedete Normen werden heute in den meisten Rechnern verwendet:

Typ	Bit	Aufteilung
short real, auch float (einfache Genauigkeit)	32	V = 1 Bit, E = 8 Bit, M = 23 Bit
long real, auch double (doppelte Genauigkeit)	64	V = 1 Bit, E = 11 Bit, M = 52 Bit

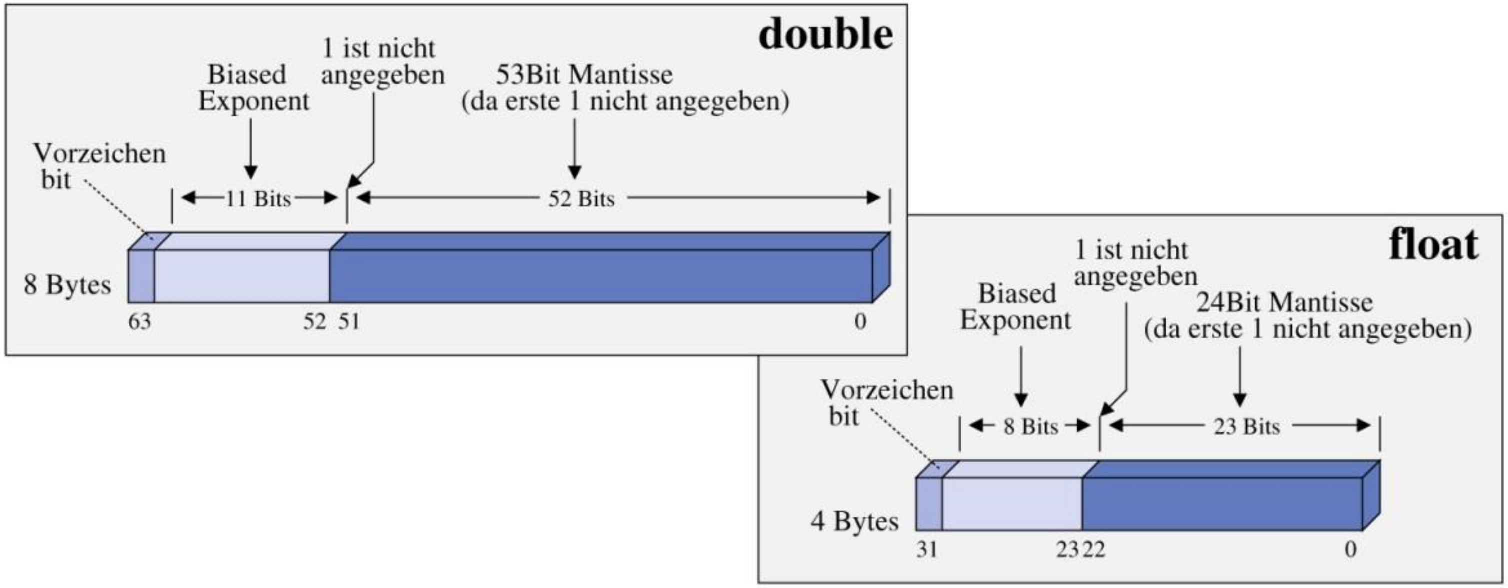


Abbildung 3.4: IEEE-Format für `float` und `double`

Konvertierung reeller Zahlen in 32-Bit IEEE 754 Gleitpunktzahlen / 1

1. Das Vorzeichen wird mit 0 (pos. Zahlen) und 1 (neg. Zahlen) codiert.
2. Konvertierung des ganzzahligen Anteils.
3. Der gebrochene Anteil wird sukzessive mit 2 multipliziert.
   a. Wenn das Ergebnis größer (oder gleich) 1 ist, wird 1 subtrahiert und eine 1 als Ergebnis notiert.
   b. Wenn das Ergebnis kleiner als 1 ist, wird erneut mit 2 multipliziert und eine 0 als Ergebnis notiert.
   c. Diese Berechnung wird solange wiederholt, bis das Ergebnis gleich 0 ist.
   d. (Achtung! Dieser Prozess kann unendlich lange fortschreiten. Die Genauigkeit der Berechnung wird durch die für die Mantisse verfügbare Anzahl der Stellen begrenzt!)
4. Das Ergebnis wird als Festkomma-Zahl notiert.

Konvertierung reeller Zahlen in 32-Bit IEEE 754 Gleitpunktzahlen / 2

STUDIUM PLUS DUALES STUDIUM
THM TECHNISCHE HOCHSCHULE MITTELHESSEN

5. Die Festkomma-Zahl wird normiert, d.h. das Komma wird so weit nach rechts oder links verschoben, bis vor dem Komma eine 1 steht.
6. Die Anzahl der Stellen, um die das Komma verschoben wird, ergibt den Wert des Exponenten.
7. Wird das Komma nach rechts verschoben, ist der Exponent negativ, wird er nach links verschoben, ist der Exponent positiv.
8. Der Exponent e wird als natürliche Zahl e' codiert mit Wert e' = e + 127 (BIAS). e' ist der „biased exponent".
9. Die Nachkommastellen (!) werden genommen und auf die Breite der Mantisse nach rechts mit Nullen aufgefüllt.
10. Die Bits werden in der Reihenfolge Vorzeichen - Exponent - Mantisse angeordnet.

Zusammenfassung IEEE 754 (short real)

STUDIUM PLUS DUALES STUDIUM | THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Für eine IEEE-754 Gleitkommazahl benötigt man:

- Vorzeichen der Mantisse (1 Bit, 0 bei pos. oder 1 bei neg. Zahl)
- Mantisse (Binär, normiert, 1 links vom Komma wird nicht dargestellt)
- Exponent (Binäre Ganzzahl; Vorzeichenloses Binärformat nach Addition eines sogenannten Bias; Wert des Bias abhängig vom Genauigkeitsgrad – short real = 127, long real = 1023)

- Beispiel: $(17{,}625)_{10}$ entspricht der Binärzahl $1{,}0001101 \cdot 2^4$

V	Exponent ------->								Mantisse ------------------------------->														
0	1	0	0	0	0	0	1	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	...	0

$(17{,}625)_{10} = (0100\ 0001\ 1000\ 1101\ 0000\ 0000\ 0000\ 0000)_2$

- Biased Exponent ergibt sich als:

Bias =	0111 1111 = $(127)_{10}$
+ wirkl. Exp. =	0000 0100 = $(4)_{10}$
	1000 0011 = $(131)_{10}$

Beispiel

Wie wird die Dezimalzahl 13, 875 als Gleitkommazahl im Format *real short* dargestellt?

0	Vorüberlegung	V CCCCCCCC MMMMMMMMMMMMMMMMMMMMMMM Vorzeichen 1 · Charakteristik 8 · Mantisse 23

1	Das Vorzeichen wird mit 0 (pos. Zahlen) und 1 (neg. Zahlen) codiert	Vorzeichen von $13{,}875_{10}$ betrachten: Positiv, somit **0**
2	Konvertierung des ganzzahligen Anteils	Vorkommazahl in Binär 13 = **1101**
3	Der gebrochene Anteil wird sukzessive mit 2 multipliziert. a) Wenn das Ergebnis größer (oder gleich) 1 ist, wird 1 subtrahiert und eine 1 als Ergebnis notiert. b) Wenn das Ergebnis kleiner als 1 ist, wird erneut mit 2 multipliziert und eine 0 als Ergebnis notiert. a. Diese Berechnung wird solange wiederholt, bis das Ergebnis gleich 0 ist.	Nachkommazahl in Binär (Leserichtung ↓) 0,875 * 2 = 1,75 **1** 0,75 *2 = 1,5 **1** 0,5 *2 = 1 **1** 0 * 2 =0 0 Hinweis ....$2^2 + 2^1 + 2^0$, $2^{-1} + 2^{-2} + 2^{-3} + \ldots$ ← Leserichtung → :2 *2
4	Das Ergebnis wird als Festkommazahl notiert	Vorkommazahl(Binär) und Nachkommazahl(Binär) zusammenführen: **1101,111**

Beispiel

5	Die Festkomma-Zahl wird normiert, d.h. das Komma wird so weit nach rechts oder links verschoben, bis vor dem Komma eine 1 steht.	**1 1 0 1,** 111 **1 1 0 ,1**111 1 **1 1 , 01**111 2 **1, 101**111 3 Die Mantisse lautet **101**111.
6	Die Anzahl der Stellen, um die das Komma verschoben wird, ergibt den Wert des Exponenten.	Anzahl der verschobenen Stellen: 3 bzw. Multiplikation der Zahl mit 2^3 somit Exponent = 3
7	Wird das Komma nach rechts verschoben, ist der Exponent negativ, wird er nach links verschoben, ist der Exponent positiv.	Der Exponent ist positiv, da das Komma nach links verschoben wurde
8	Der Exponent e wird als natürliche Zahl e' codiert mit Wert e' = e + 127 (BIAS). e' ist der „biased exponent“.	Bias + Exponent = Charakteristik 127 + 3 = 130 = 128 + 2 in Binär: 10000010 Hinweis: Dezimal in Dual, dann +127. Dual in Dezimal, dann -127

9	Die Nachkommastellen (!) werden genommen und auf die Breite der Mantisse nach rechts mit Nullen aufgefüllt.	`1`
		`MMMMMMMMMMMMMMMMMMMMMMM`
		`10111100000000000000000`

10	Reihenfolge Vorzeichen - Exponent - Mantisse angeordnet.	`V CCCCCCCC MMMMMMMMMMMMMMMMMMMMMMM`
		`0 10000010 10111100000000000000000`
		`In Hex`
		`0100 0001 0101 1110 0000 0000 0000 0000`
		`4 1 5 E 0 0 0 0`

Formel für die Darstellung einer Gleitpunktzahl im IEEE-Format

STUDIUM PLUS DUALES STUDIUM | THM TECHNISCHE HOCHSCHULE MITTELHESSEN

$$\underbrace{(-1)^S}_{\text{SIGN}} \times \underbrace{(2^{B-bias})}_{\text{EXPONENT}} \times \underbrace{(1.f_N \ldots\ldots f_0)}_{\text{MANTISSE}}$$

MANTISSE. N=22 (float = 23 Stellen);
N=51 (double = 52 Stellen);

EXPONENT. B = Biased Exponent (zu speichernder Exp.)

SIGN. S = 0 (positiv); S = 1 (negativ)

Beispiel:

V	Exponent ------->								Mantisse ------------------------------>														
0	1	0	0	0	0	0	1	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	...	0

$$\Rightarrow \quad (-1)^{\boxed{0}} \times (2^{\boxed{131}-127}) \times (1.\boxed{0001101})$$

$$\Rightarrow \quad = 1 \cdot 2^4 \cdot 1{,}0001101 \qquad = (17{,}625)_{10}$$

Gegenüberstellung *float* und *double* nach IEEE-754

	einfach (float)	doppelt (double)
Vorzeichen-Bits	1	1
Exponenten-Bits	8	11
Mantissen-Bits	23	52
Bits insgesamt	32	64
BIAS	127	1023
Exponentenbereich	[-126, 127]	[-1022, 1023]

Sonderfälle des IEEE-Formats:

Biased Exponent	Mantisse	Bedeutung
111..111 (=255 bzw. =2047)	≠ 0	not a number
111..111 (=255 bzw. =2047)	000..000 (=0)	±∞
000..000 (=0)	000..000 (=0)	±0

Inhalt

Textdarstellung / Zeichenkodierung

- Um Texte in einem Rechner darzustellen, codiert man Alphabet und Satzzeichen in Bitfolgen.
- Mit einem Alphabet von 26 Kleinbuchstaben, ebenso vielen Großbuchstaben, einigen Satzzeichen wie etwa Punkt, Komma und Semikolon und Spezialzeichen wie '+', '&', '%' hat eine normale Schreibmaschinentastatur eine Auswahl von knapp 100 Zeichen.
- Die Information, wo ein Zeilenumbruch stattfinden oder wo ein Text eingerückt werden soll, codiert man ebenfalls wie ein Zeichen.
- Beispiele:
  - das CR-Zeichen (von englisch **C**arriage **R**eturn = Wagenrücklauf),
  - das LF-Zeichen (von englisch **L**ine **F**eed = Zeilenvorschub),
  - das Tabulatorzeichen Tab.
- Diese Zeichen haben bei der Darstellung von Text eine formatierende Wirkung, werden aber nicht als Zeichen gedruckt. Sie heißen daher auch nicht-druckbare Zeichen.

Zeichenkodierung: ASCII, EBCDIC, ...

- Mit einem Byte (= 8 Bits) kann man $2^8 = 256$ verschiedene Zeichen darstellen.
- Um Texte mit Hilfe von Bitfolgen zu speichern, bedarf es einer Vorschrift, die den Zeichen eines Zeichensatzes eindeutig Byte(-nummern) zuordnet. Eine solche Zuordnungsvorschrift nennt man einen Code. Häufig verwendete und international akzeptierte Codes sind:
  - ASCII: **American Standard Code of Information Interchange**
  - EBCDIC: **Extended Binary Codes Decimal Interchange Code**
- Diese Codes folgen einer gewissen Systematik, so stehen z.B.
  - alle Kleinbuchstaben,
  - alle Großbuchstaben,
  - die Ziffern 0 bis 9

    in ihrer "natürlichen" Reihenfolge.
- Die folgende Tabelle enthält einige Beispiele für die Zuordnung von Bytes zu verschiedenen Zeichen.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Der ASCII-Code

- Zur Realisierung von länderspezifischen Abweichungen steht der Code-Bereich [128 .. 255] für Zeichen wie z.B. "ä" (ASCII 132), "ö" (ASCII 148) "ü" (ASCII 129) zur Verfügung
  Weiterhin werden Zeichen dargestellt, mit denen man einfache graphische Darstellungen wie Rahmen und Schraffuren zusammensetzen kann.

- Diese Zeichen können über die numerische Tastatur eingegeben werden. Dazu muss diese aktiviert sein (dies geschieht durch die Taste "Num"), danach kann bei gedrückter "Alt"-Taste der dreistellige ASCII-Code eingegeben werden.

- Leider ist auch die Auswahl der sprachspezifischen Sonderzeichen eher zufällig und bei weitem nicht ausreichend für die vielfältigen Symbole fremder Schriften. Daher wurden von der „International Standardisation Organisation“ (ISO) verschiedene ASCII-Erweiterungen normiert. In Europa ist dazu die ASCII-Erweiterung Latin-1 nützlich, die durch die Norm ISO 8859-1 beschrieben wird.

Unicode

- Neuer Standard: Unicode – versucht, sämtliche relevanten Zeichen aus den unterschiedlichsten Kulturkreisen in einem universellen Code zu vereinigen
- Version 1: Zunächst eine 16-Bit Codierung, folglich maximal 65.536 darstellbare Zeichen
- Die ersten 128 Zeichen des Unicode-Zeichensatzes sind identisch mit dem ASCII-Code, die folgenden 128 identisch mit ISO-Latin 1
- Unicode ist als UCS (Universal Character Set) bzw. als ISO-10646 vom Unicode-konsortium und der ISO standardisiert
- ISO geht in Definition von UCS noch einen Schritt weiter als das Unicode-Konsortium: Es werden sowohl eine 16-Bit Codierung (UCS-2) als auch eine 32-Bit Codierung (UCS-4) festgelegt
- Speicherung von Texten im Unicode benötigt zwei (UCS-2) bis viermal (UCS-4) soviel Speicherplatz, wie im ASCII-Code

Quelle: http://www.hki.uni-koeln.de/

UTF-8 (UCS Transformation Format 8)

- UTF ist andere Codierung von UCS bzw. Unicode
- UTF-8: Aufbau und Verfahren
  - UTF-8 ist eine Mehrbyte-Codierung (bis zu 4 Byte) bzw. ein Code von variabler Bit-Breite
  - ASCII-Zeichen werden bei UTF-8 mit 1 Byte codiert, in dem das erste Bit immer Null ist: `0xxx xxxx`
  - Jedes Byte, das mit einer `1` beginnt, gehört zu einem aus mehreren Bytes bestehenden UTF-8 Code: `110x xxxx 10xx xxxx` = 2 Byte Code
  - Besteht ein UTF-8 Code aus n ≥ 2 Bytes, beginnt das erste Byte (Startbyte) mit n Einsen, gefolgt von einer Null und jedes n-1 folgende Byte mit der Bitfolge 10 (siehe Beispiel oben)

    Ein 3-Byte Code sieht also folgendermaßen aus: `1110 xxxx 10xx xxxx 10xx xxxx`
  - Mit den 16 „verfügbaren“ Bits können alle 16-Bit UCS-2 Codes dargestellt werden
- UTF-8 codierte Dateien sind voll abwärtskompatibel zu 7-Bit ASCII und vergrößern den Umfang von Dateien aus dem amerikanischen und europäischen Bereich gar nicht oder nur unwesentlich

Quelle: http://www.hki.uni-koeln.de/

Überblick UTF-8 Kodierung

Unicode-Bereich	UTF-8 Kodierung	Bemerkung	Möglichkeiten (theoretisch)
0000 0000 – 0000 007F	0xxx xxxx	Bereich der dem ASCII-Code entspricht	$2^7 = 128$
0000 0080 – 0000 07FF	110xxxxx 10xxxxxx	Erstes Byte beginnt immer mit 110, die folgenden Bytes mit 10. Die xxxxx stehen für die Bits des Unicode-Zeichenwerts. Dabei wird das niederwertigste Bit des Zeichenwerts auf das rechte x im letzten Byte abgebildet, die höherwertigen Bits fortschreitend von rechts nach links. Die Anzahl der Einsen vor der ersten 0 im ersten Byte ist gleich der Gesamtzahl der Bytes für das Zeichen (In Klammern jeweils die theoretisch maximal möglichen).	$2^{11} – 2^7$ (2^{11}) = 1920 (2048)
0000 0800 – 0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx		$2^{16} – 2^{11}$ (2^{16}) = 63.488 (65.536)
0001 0000 – 0010 FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx		2^{20} (2^{21}) = 1.048.576 (2.097.152)

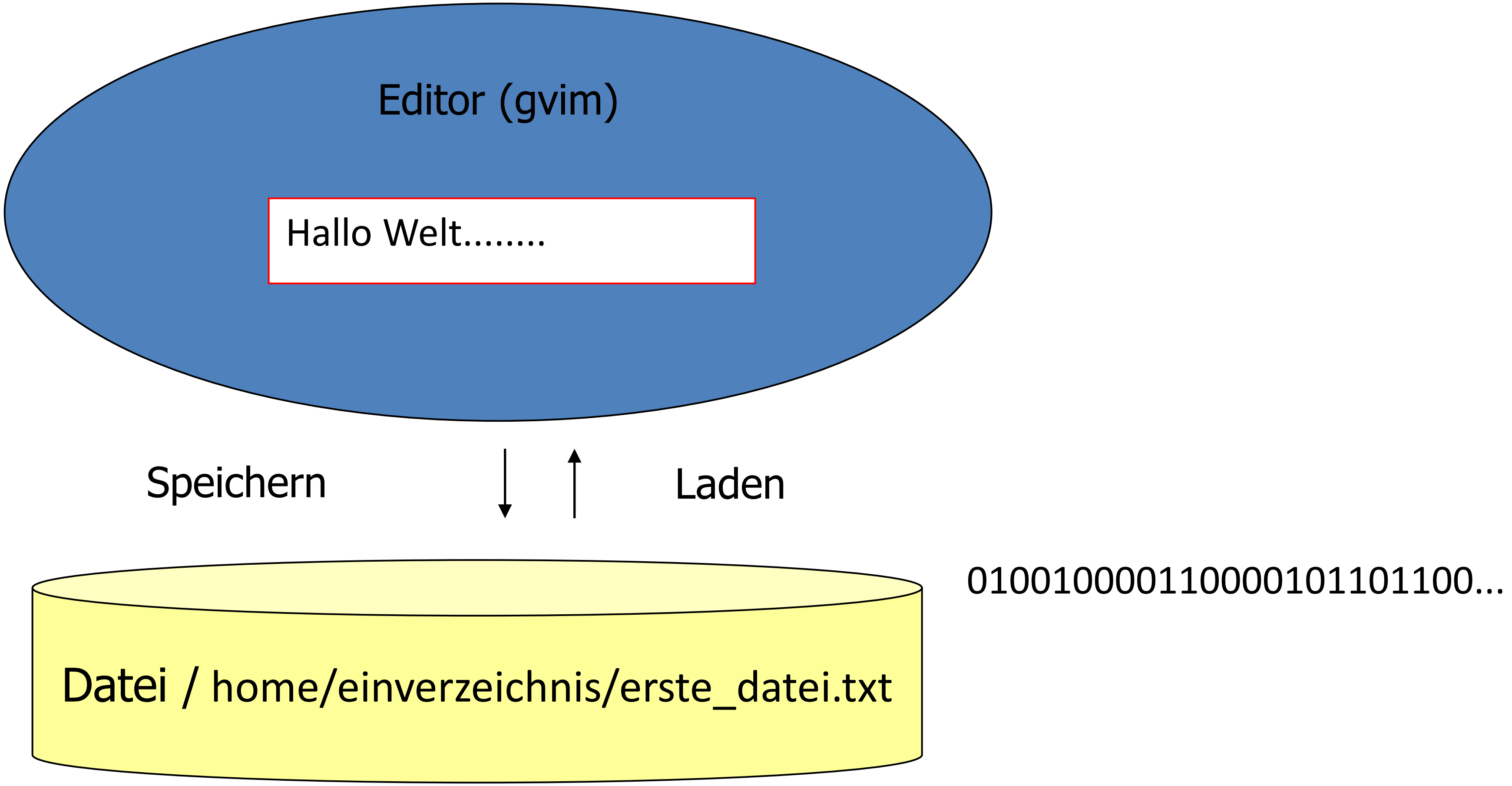
Quelle: http://de.wikipedia.org/wiki/UTF-8

Inhalt

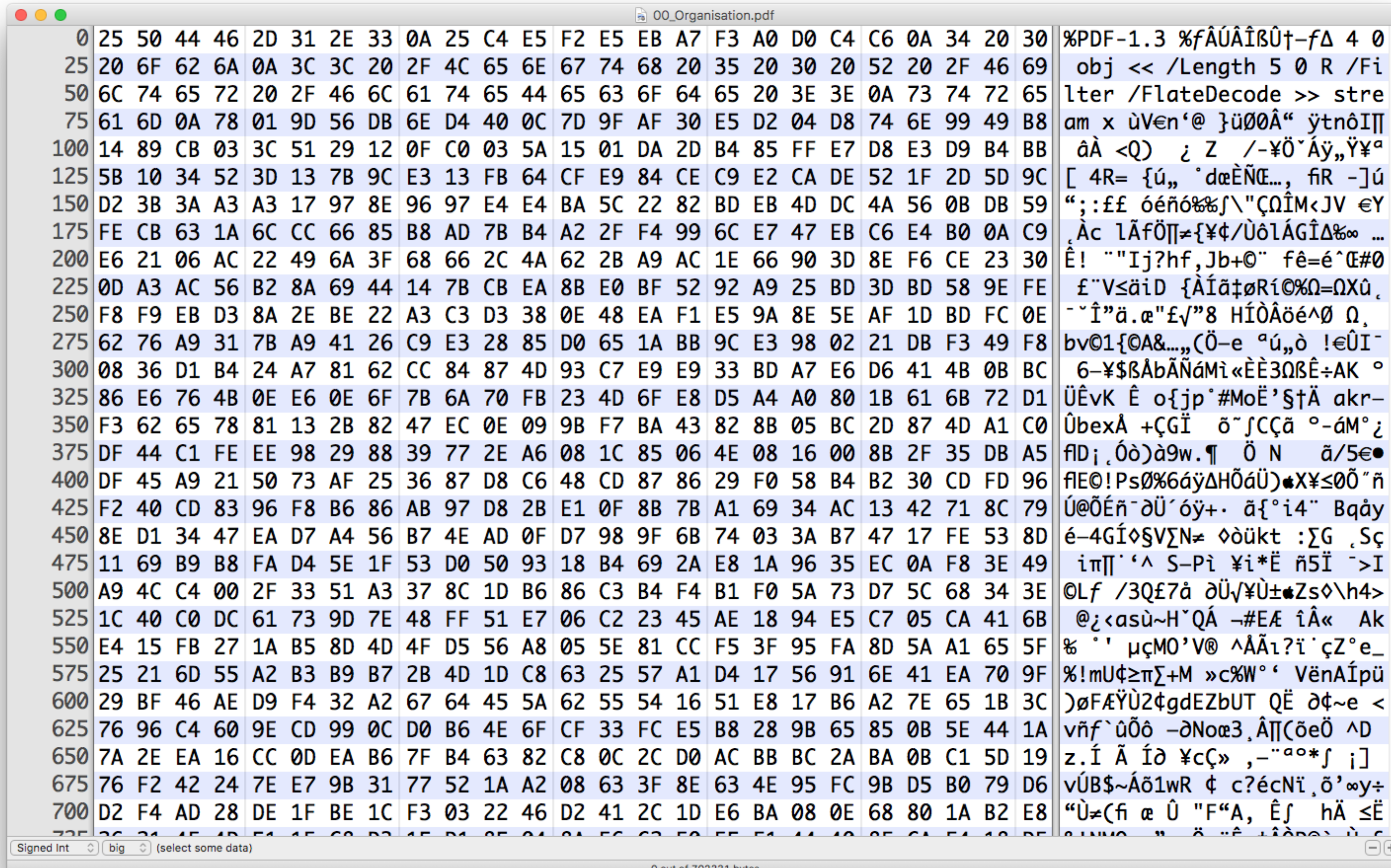
Was ist eine Datei?

- Eine Datei ist eine Folge von Bits bzw. von Bytes. Dateien dienen dazu, Daten für längere Zeit (genauer: über eine Sitzung hinaus) zu speichern.
- So wie ein Buch einen Titel hat, hat eine Datei einen Dateinamen.
- Der Dateiinhalt ist eine Folge von Bits bzw. Bytes, die nach Maßgabe der Datei-Beschreibung (Protokoll) z.B. als (Binär-) Zahlen, Texte, Bilder, Graphiken, ausführbarer Programm-Code etc. interpretiert werden.
- Die Erweiterung des Namens (extension) gibt an, welcher Art die gespeicherten Daten sind (z.B. .doc, .txt, .ppt, .pas, .java, .exe).
- Dateien werden in Verzeichnissen zusammengefasst und können vom Betriebssystem über einen Pfad angesprochen werden, z.B. /home/einverzeichnis/erste_datei.txt.

Interpretation von Daten

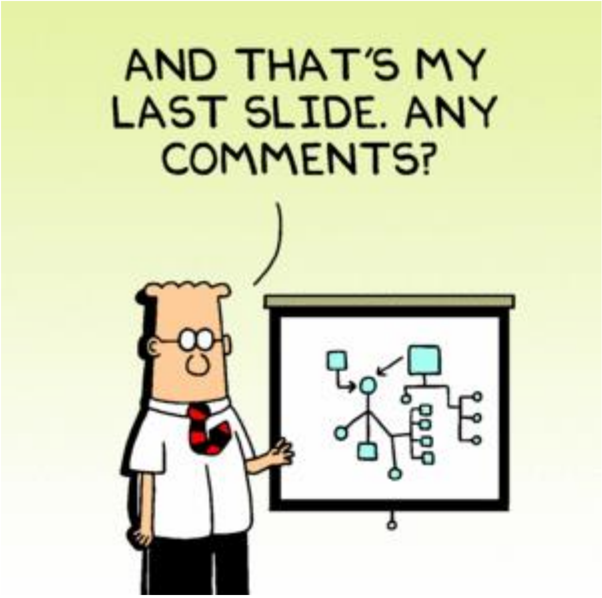


HEX-Editor


00_Organisation.pdf
0 25 50 44 46 2D 31 2E 33 0A 25 C4 E5 F2 E5 EB A7 F3 A0 D0 C4 C6 0A 34 20 30
25 20 6F 62 6A 0A 3C 3C 20 2F 4C 65 6E 67 74 68 20 35 20 30 20 52 20 2F 46 69
50 6C 74 65 72 20 2F 46 6C 61 74 65 44 65 63 6F 64 65 20 3E 3E 0A 73 74 72 65
75 61 6D 0A 78 01 9D 56 DB 6E D4 40 0C 7D 9F AF 30 E5 D2 04 D8 74 6E 99 49 B8
100 14 89 CB 03 3C 51 29 12 0F C0 03 5A 15 01 DA 2D B4 85 FF E7 D8 E3 D9 B4 BB
125 5B 10 34 52 3D 13 7B 9C E3 13 FB 64 CF E9 84 CE C9 E2 CA DE 52 1F 2D 5D 9C
150 D2 3B 3A A3 A3 17 97 8E 96 97 E4 E4 BA 5C 22 82 BD EB 4D DC 4A 56 0B DB 59
175 FE CB 63 1A 6C CC 66 85 B8 AD 7B B4 A2 2F F4 99 6C E7 47 EB C6 E4 B0 0A C9
200 E6 21 06 AC 22 49 6A 3F 68 66 2C 4A 62 2B A9 AC 1E 66 90 3D 8E F6 CE 23 30
225 0D A3 AC 56 B2 8A 69 44 14 7B CB EA 8B E0 BF 52 92 A9 25 BD 3D BD 58 9E FE
250 F8 F9 EB D3 8A 2E BE 22 A3 C3 D3 38 0E 48 EA F1 E5 9A 8E 5E AF 1D BD FC 0E
275 62 76 A9 31 7B A9 41 26 C9 E3 28 85 D0 65 1A BB 9C E3 98 02 21 DB F3 49 F8
300 08 36 D1 B4 24 A7 81 62 CC 84 87 4D 93 C7 E9 E9 33 BD A7 E6 D6 41 4B 0B BC
325 86 E6 76 4B 0E E6 0E 6F 7B 6A 70 FB 23 4D 6F E8 D5 A4 A0 80 1B 61 6B 72 D1
350 F3 62 65 78 81 13 2B 82 47 EC 0E 09 9B F7 BA 43 82 8B 05 BC 2D 87 4D A1 C0
375 DF 44 C1 FE EE 98 29 88 39 77 2E A6 08 1C 85 06 4E 08 16 00 8B 2F 35 DB A5
400 DF 45 A9 21 50 73 AF 25 36 87 D8 C6 48 CD 87 86 29 F0 58 B4 B2 30 CD FD 96
425 F2 40 CD 83 96 F8 B6 86 AB 97 D8 2B E1 0F 8B 7B A1 69 34 AC 13 42 71 8C 79
450 8E D1 34 47 EA D7 A4 56 B7 4E AD 0F D7 98 9F 6B 74 03 3A B7 47 17 FE 53 8D
475 11 69 B9 B8 FA D4 5E 1F 53 D0 50 93 18 B4 69 2A E8 1A 96 35 EC 0A F8 3E 49
500 A9 4C C4 00 2F 33 51 A3 37 8C 1D B6 86 C3 B4 F4 B1 F0 5A 73 D7 5C 68 34 3E
525 1C 40 C0 DC 61 73 9D 7E 48 FF 51 E7 06 C2 23 45 AE 18 94 E5 C7 05 CA 41 6B
550 E4 15 FB 27 1A B5 8D 4D 4F D5 56 A8 05 5E 81 CC F5 3F 95 FA 8D 5A A1 65 5F
575 25 21 6D 55 A2 B3 B9 B7 2B 4D 1D C8 63 25 57 A1 D4 17 56 91 6E 41 EA 70 9F
600 29 BF 46 AE D9 F4 32 A2 67 64 45 5A 62 55 54 16 51 E8 17 B6 A2 7E 65 1B 3C
625 76 96 C4 60 9E CD 99 0C D0 B6 4E 6F CF 33 FC E5 B8 28 9B 65 85 0B 5E 44 1A
650 7A 2E EA 16 CC 0D EA B6 7F B4 63 82 C8 0C 2C D0 AC BB BC 2A BA 0B C1 5D 19
675 76 F2 42 24 7E E7 9B 31 77 52 1A A2 08 63 3F 8E 63 4E 95 FC 9B D5 B0 79 D6
700 D2 F4 AD 28 DE 1F BE 1C F3 03 22 46 D2 41 2C 1D E6 BA 08 0E 68 80 1A B2 E8
%PDF-1.3 %ƒÂÚÂÎßÛ†–ƒΔ 4 0
obj << /Length 5 0 R /Fi
lter /FlateDecode >> stre
Signed Int
big
(select some data)
0 out of 702331 bytes

Selbstkontrolle

- Sie kennen die Begriffe Codierung und Interpretation
- Sie kennen die üblichen Zahlensysteme (Ganze Zahlen, Gebrochene Zahlen)
- Sie können mit dem Dual-, Oktal, Hexadezimalsystem umgehen
- Sie können die Konvertierungsalgorithmen anwenden
- Sie kennen Rechenoperationen im Dualsystem
- Sie kennen übliche Codes zur Darstellung von Zeichen


AND THAT'S MY
LAST SLIDE. ANY
COMMENTS?