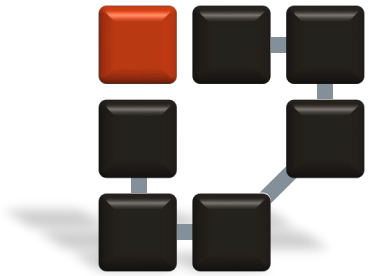
Informatik 1 für Nebenfachstudierende
Grundmodul

Datendarstellung – Teil 2

Kai-Steffen Hielscher
Folienversion: 29. Oktober 2019



Informatik 7
Rechnernetze und
Kommunikationssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Inhaltsübersicht

- Kapitel 1 - Einführung und Übersicht
  - Was ist Informatik?
  - Grundbegriffe
  - **Datendarstellung**
  - Hardware von Computersystemen
  - Grundsoftware üblicher Computersysteme

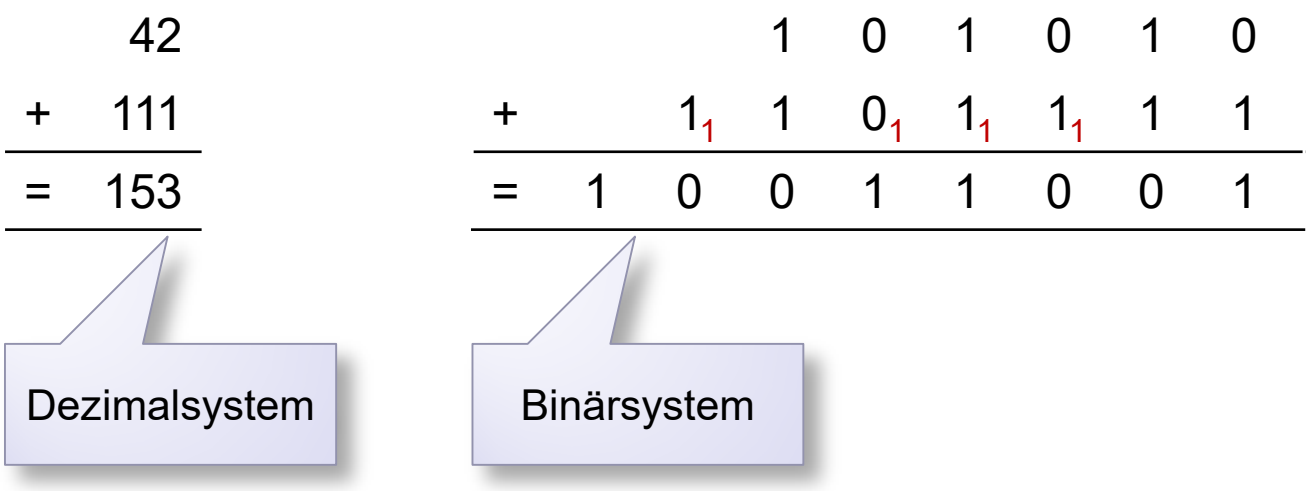
Addition im Binärsystem

- Übertrag bei Addition von Dualzahlen
  - Bei Addition von Dualzahlen entsteht ein **Übertrag**, wenn das Ergebnis > 1 ist.

1. Zahl	2. Zahl	Summe	Übertrag
0	0	0 + 0 = 0	0
0	1	0 + 1 = 1	0
1	0	1 + 0 = 1	0
1	1	1 + 1 = 0	1

Addition im Binärsystem

- Stellenweise Addition mit Übertrag
  - Zwei aus mehreren Ziffern bestehende Dualzahlen werden ganz analog zur Addition von Dezimalzahlen addiert (Addition des Übertrags zur nächsthöherwertigen Position)
  - Beispiel:

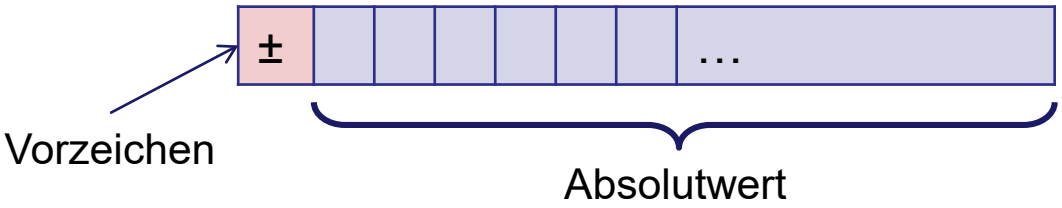


Wiederholung: Zahlendarstellung im Rechner

- Die **Vorzeichendarstellung** kann durch ein Vorzeichenbit gelöst werden, das an den Anfang oder das Ende eines Worts gestellt ist.
  - Eine 0 bedeutet z.B. ein positives, eine 1 ein negatives Vorzeichen.
  - Bei der Tetradendarstellung wird dementsprechend eine Tetrade (= 4 Bit) für das Vorzeichen reserviert.

Wiederholung: Vorzeichendarstellung

- **Darstellung ganzer Zahlen**
  - Ganze Zahlen ::= Natürliche Zahlen + Negative (‘nat.‘) Zahlen
    { … -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, … }
  - Kodierung des absoluten Zahlenwertes und Vorzeichen´+´ bzw. ´-´, d.h. für diese 2 Möglichkeiten genau 1 weiteres Bit Information nötig
  - sogen. **Vorzeichendarstellung**:
    - Erstes Bit für Vorzeichen: **0** für "**+**" und **1** für "**-**"
    - weitere Bits für den Absolutwert
  - Beispiel:
    - Wortlänge sei 4 Bit
    - Ein Bit für Vorzeichen
    - Drei Bit für den Absolutwert
    - Zahlen von -7 bis +7 darstellbar



Z = ± | Absolutwert |

Vorzeichendarstellung

- **Darstellung ganzer Zahlen:**
  Wortlänge sei 4 Bit

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

- Man erkennt folgende Nachteile:
  - Zahl 0 durch zwei verschiedene Bitfolgen dargestellt

Vorzeichendarstellung

- Addition problematisch:

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

-3	1011
+ +5	+ 0101
= +2	= 0000

Fehler!
= +0, nicht +2

Zahlendarstellung im Rechner

- **Frage:**
  Existiert Bit-Darstellung für (negative) ganze Zahlen, so daß Addition zweier ganzer Zahlen immer korrekt ist?

	-3		????	
+	+5	+	0101	welche Bitfolge für "-3"?
=	+2	=	0010	

- **Lösung: Zweierkomplement**

	-3		1101	
+	+5	+	0101	Bitfolge "1101" für "-3".
=	+2	=	0010	

- Bitfolge 1101 (-3) ist Zweierkomplement zu 0011 (+3).

Komplementdarstellung

- Das **Komplement** einer Folge von Nullen und Einsen erhält man, indem man stellenweise negiert, jede 0 durch 1 und jede 1 durch 0 ersetzt.
- Bei der Darstellung ganzer Zahlen zur Basis B unterscheidet man zwischen dem B-Komplement und dem (B-1)-Komplement einer Zahl.
- Das **B-Komplement** der Zahl Z ist definiert als diejenige Zahl Ż, für die die Beziehung

  $$Z + \dot{Z} = B^n$$

  gilt (n ist gleich der Länge der Darstellung von Z).
- Das **(B-1)-Komplement** von Z ist diejenige Zahl Ż für die

  $$Z + \dot{Z} = B^n - 1$$

  ist.
- Das **1er-Komplement** erhält man bei Binärzahlen, indem man Stelle für Stelle negiert, d.h. 1 durch 0 und 0 durch 1 ersetzt.
- Das **2er-Komplement** erhält man, indem man das 1er-Komplement bildet und zum Ergebnis 1 addiert.

Komplementdarstellung

- Beispiele: (B=2, n = 4, Z = 1011)
  - 1er-Komplement von Z: Ż = 0100
  - 2er-Komplement von Z: Ż = 0101
- Die Komplementbildung wird bei der Durchführung arithmetischer Operationen auf Digitalrechnern verwendet.
- Die Subtraktion zweier Zahlen a und b realisiert man durch die Addition von a mit dem 2er-Komplement von b.
- Da eine Rechenanlage eine feste Wortlänge besitzt, können nur endlich viele verschiedene Zahlen, z.B. 0 bis $2^8 - 1 = 255$ dargestellt werden.

Komplementdarstellung

- Um auch negative Zahlen verarbeiten zu können, hat man bei der Komplementdarstellung folgende Konvention vereinbart:
- Sei n die Wortlänge. Von den 2^n verschiedenen Zuständen der Speicherzellen definiert man
  - 0, 1, …, 2^{n-1} -1 als Darstellungen für positive Zahlen und
  - 2^{n-1}, 2^{n-1}+1, ..., 2^n-1 als Darstellungen für die negativen Zahlen -2^{n-1} , -2^{n-1}+1, ..., -1.

Komplementdarstellung

- Die Bezeichnung „Komplementdarstellung“ rührt daher, dass man durch Bildung des 2er-Komplements aus der Zahldarstellung von x die Zahldarstellung von -x erhält.
- Ob eine Zahl positiv oder negativ ist, erkennt man am ersten Bit der Zahldarstellung: Bei positiven Zahlen besitzt dieses Bit den Wert 0, bei negativen den Wert 1.
- Beispiel: (n = 8, darstellbare Zahlen: -128 bis 127)

Größte darstellbare Zahl:	0 1 1 1 1 1 1 1	+ 127
Kleinste darstellbare Zahl:	1 0 0 0 0 0 0 0	- 128
	1 1 1 1 1 1 1 1	- 1
	0 1 1 0 0 0 0 1	+ 97
	1 0 0 1 1 1 1 1	- 97

Darstellung von Gleitkommazahlen

- Bei der Gleitkommadarstellung wird neben der Ziffernfolge zusätzlich die Größenordnung mit Hilfe eines Exponenten formuliert
- **Beispiele:**
  - $(0.00015)_{10} = 0.15 * 10^{-3}$
  - $(325000.0)_{10} = 0.325 * 10^{6}$
- Eine **Gleitkommazahl** Z ist definiert durch $Z = M * B^{E}$ und
  - M mit |M| < 1 heißt **Mantisse**, B ist die **Basis** des Zahlensystems und E ist der **Exponent**.
  - Damit die Gleitkommadarstellung eindeutig ist muss die Mantisse in **Normalform** vorliegen, d.h. die erste Stelle muss von Null verschieden sein.
  - Mantisse und Exponent besitzen jeweils ein Vorzeichen.
  - Um beim Exponenten das Vorzeichen einzusparen, wird oft eine positive **Charakteristik** C verwendet, die durch Addition einer Konstanten K entsteht:

$$C = E + K \Rightarrow Z = M * B^{C-K}$$

Darstellung von Gleitkommazahlen

- **Beispiele:**
  - Es gebe 2 Stellen für die Charakteristik C, d.h. $0 \leq C \leq 99$, so dass

    mit K=50 für E der Darstellungsbereich $-50 \leq E \leq +49$ vorliegt.
  - $Z = +0.478 * 10^{-3}$ ⇨ M = +4780000, E = -03 , C = 47
  - $Z = +0.5137201 * 10^{5}$ ⇨ M = +5137201, E = +05, C = 55
  - $Z = -0.31438 * 10^{2}$ ⇨ M = -3143800, E = +02 , C = 52
  - Darstellung einer Dezimalzahl im **dualen Dezimalcode** (BCD-Code, **B**inary **C**oded **D**ecimals):
  - $Z = +0.654 * 10^{-12}$ ⇨ $Z = (0\ 0110\ 0101\ 0100\ 0011\ 1000)_{BCD}$

    $+ \quad 6 \quad 5 \quad 4 \quad 3 \quad 8$

    mit M = +654 und C = 38.

Gleitkommaarithmetik

- **Addition:**
  - Gegeben: $Z_1 = (M_1,C_1)$ und $Z_2 = (M_2,C_2)$; gesucht: $Z = Z_1 + Z_2 = (M,C)$
  - Gilt $C_1 = C_2 = C$ dann werden die Mantissen M_1 und M_2 einfach addiert.
  - Andernfalls muss die kleinere Mantisse um so viele Stellen nach rechts verschoben werden, wie es der Betrag $|C_1 - C_2|$ definiert.
  - Führt die Addition von M_1 und M_2 zu $|M| \geq 1$, muss eine zusätzliche **Normalisierung** erfolgen: dabei wird M um eine Stelle nach rechts geschoben und die Charakteristik C um 1 erhöht.
- **Beispiel:** $(0.219)_{10} * 10^{-4} + (0.501)_{10} * 10^{-2}$ (Mantissenlänge: 3)
  - Für K=50 gilt: $M_1 = 219$, $C_1 = 46$, $M_2 = 501$ und $C_2 = 48$
  - Die Charakteristik des 1. Summanden ist um $|C_1 - C_2| = 2$ kleiner, seine Mantisse wird daher um 2 Stellen nach rechts geschoben

    ⇨ $M_{1\_neu} = 002$, $C_{1\_neu} = C_2 = 48$

    ⇨ $M = M_{1\_neu} + M_2 = 002 + 501 = 503$, $C = 48$ ⇨ $Z = (0{,}503)_{10} * 10^{-2}$
  - Damit ist bei der Angleichung von C_1 und C_2 wegen der beschränkten Mantissenlänge von 3 Dezimalstellen ein **Stellenverlust** aufgetreten.

Gleitkommaarithmetik

- **Beispiel:** Genauigkeitsverlust wegen Normalisierung

$$
\begin{array}{lll}
 & 0.219 * 10^{-2} & \\
+ & 0.889 * 10^{-2} & \\
 & & \\
 & 219\ 48 & \\
+ & 888\ 48 & \\
\hline
= & 1107\ 48 & = 1.107 * 10^{-2} \\
\approx & 110\ 49 & = 0.110 * 10^{-1}
\end{array}
$$

Wegen der begrenzten Mantissenlänge von 3 wurde die Ziffer 7 abgeschnitten.

⇨ Besser wäre eine arithmetische Rundung : $(111{,}49) \approx 0.111 * 10^{-1}$!

Gleitkommaarithmetik

- **Subtraktion:** Subtraktion ist die Addition zweier Zahlen mit ungleichen Vorzeichen.
- **Beispiel:**

```
          0.219 * 10^-4          Standarddarstellung
-         0.501 * 10^-2

           219 46                Gleitpunktdarstellung (M/C)
-          501 48

           002 48                Anpassung von Mantisse und Charakteristik
-          501 48
=         -499 48    = -0.499 * 10^-2
```

Gleitkommaarithmetik

- **Multiplikation:** Man multipliziert die Mantissen und addiert die Exponenten bzw. die Charakteristiken, d.h.
  - $M = M_1 * M_2$
  - $E = E_1 + E_2$
  - $C = C_1 + C_2 - K$
  - Gegebenenfalls ist anschließend zu normalisieren, d.h. Linksshift der Mantisse um 1 Stelle und Dekrementierung (Wert um 1 verkleinern) der Charakteristik.
- **Division:** Man dividiert die Mantissen und subtrahiert die Exponenten bzw. die Charakteristiken, d.h.
  - $M = M_1 / M_2$
  - $E = E_1 - E_2$
  - $C = C_1 - C_2 + K$
  - Auch bei der Division kann eine Normalisierung notwendig werden, d.h. Rechtsshift der Mantisse um 1 Stelle und Inkrementierung (Wert um 1 vergrößern) der Charakteristik.

Numerische Effekte

- Folgende **Fehlermöglichkeiten** und **Problemfälle** wurden bereits in den Beispielen veranschaulicht:
  - beschränkte Genauigkeit durch endliche Wortlänge
  - Konvertierungsfehler durch die Transformation vom Dezimalsystem ins Dualsystem und zurück
  - Stellenverlust bedingt durch das Angleichen der Charakteristik (durch Verschieben der Mantisse gehen rechtsbündig Stellen verloren)
  - Normalisierung kann zum **Abschneiden** einer Stelle bzw. zur Rundung führen
- Es kann auch passieren, dass Operationen gar nicht ausführbar sind, weil das Ergebnis außerhalb des zulässigen Zahlenbereichs liegen würde.
  - **Additionsüberlauf:** Statt der Charakteristik wird jetzt der Exponent $-49 \leq E \leq +49$ verwendet:

```
          219 +49
+         821 +49
-----------------
=        1040 +49
=         104 +50
```

Numerische Effekte

- Weitere Fehlersituationen sind der
  - Divisionsüberlauf
  - Multiplikationsüberlauf
  - Multiplikationsunterlauf
  - Divisionsunterlauf
- Entsteht zur Laufzeit eines Programms einer der obigen Fehler, führt das zum Programmabbruch, und eine Fehlermeldung wie "exponent overflow" bzw. "exponent underflow" wird ausgegeben.
- Auch das Inkrementieren, also die sukzessive Addition von 1 kann zu einer Fehlersituation führen solange wir mit Gleitpunkzahlen arbeiten.
  - **Beispiel:**

    Ein Rechner habe eine Gleitkommaarithmetik mit 6-stelliger Mantisse, jede Stelle verschlüsselt durch eine Tetrade, und einen Exponentenbereich von $-49 \leq E \leq +49$.

    ⇨ Die größte darstellbare Zahl ist $0.999999 * 10^{49}$.

    Diese Zahl ist durch einfaches Inkrementieren nicht erreichbar:

Numerische Effekte

$$\begin{array}{r} 0.100000 * 10^7 \\ +\ 0.100000 * 10^1 \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{r} 0.100000 * 10^7 \\ +\ 0.000000 * 10^7 \\ \hline = 0.100000 * 10^7 \end{array}$$

Die zu addierende 1 wird durch die Angleichung rechts vergessen, und die Inkrementierung bleibt bei $0.1 * 10^7$ stehen!

Da der Rechner hier keine Fehlermeldung oder Warnung produziert, können besonders unangenehme, nicht unmittelbar erkennbare Fehler auftreten!

⇨ **Fazit:** bei der Interpretation von Ergebnisdaten ist Vorsicht geboten, es sollte immer eine Plausibilitätsbetrachtung durchgeführt werden und bei besonders wichtigen Zwischenergebnissen alternative Berechnungen zur mehrheitlichen Bestätigung der Korrektheit der numerischen Berechnungen erfolgen.

Beispiel Raumfähre: 3-aus-5-Mehrheitsentscheidung!

Standardzahlenformate

- Für praktische Zwecke kann ein Computer immer nur mit einem endlichen Teilbereich der ganzen Zahlen umgehen.
  - Die Größe des benötigten Bereiches richtet sich nach der Anwendung.
  - In den meisten Programmiersprachen gibt es mehrere ganzzahlige Datentypen mit unterschiedlichen Wertebereichen.
  - Je größer der zur Verfügung gestellte Bereich ist, um so mehr Bits werden zur Darstellung eines jeden Wertes aus diesem Bereich benötigt.
  - Dies ist immer ein Vielfaches von 8 (Wortlänge des Computers), da ein Byte die kleinste Gruppe von Bits ist, die ein Rechner verarbeiten kann.
  - Durch die Wahl eines genügend großen Bereiches muss der Programmierer dafür sorgen, dass der Bereich nicht überschritten wird.

Standardzahlenformate

- Die folgende Tabelle zeigt die Bereiche und ihre Namen, wie sie beispielsweise von der Programmiersprache **Java** bereitgestellt werden zusammen mit der Größe die für die Speicherung einer Zahl benötigt wird.

Standardzahlenformate

Ganze Zahlen

Bereich	Größe	Delphi	Java
-128 … 127	8 Bit	`Shortint`	`byte`
-32768 … 32767	16 Bit	`Integer`	`short`
-2^{31} … $2^{31}-1$	32 Bit	`Longint`	`int`
-2^{63} … $2^{63}-1$	64 Bit		`long`
0 … 255	8 Bit	`Byte`	
0 … 65535	16 Bit	`Word`	

Gleitkommazahlen

Bereich	Bytes	Delphi	Java
±2.9 E -39 … 1.7 E 38	6	`Real`	
±1.5 E -45 … 3.4 E 38	4	`Single`	`float`
±5.0 E -324 … 1.7 E 308	8	`Double`	`double`
±3.4 E -4932 … 1.1 E 4932	10	`Extended`	

Daten und Informationen

- Daten sind Bitfolgen. Findet man Daten, lässt sich noch keine Information daraus extrahieren.
- Eine Folge von Bits oder Bytes hat für sich genommen keine Bedeutung. Erst wenn man weiß, wie diese Bytes zu interpretieren sind, erschließt sich deren Bedeutung und damit eine Information.
- Betrachten wir etwa die Bitfolge

  0100 0100 0110 0101 0111 0010 0010 0000 0100 0010 0110 0001 0110 1100 0110 1100 0010 0000 0110 1001 0111 0011 0111 0100 0010 0000 0111 0010 0111 0101 0110 1110 0110 0100 0010 1110
  - Interpretation als Folge von Hexadezimal-Ziffern liefert:

    44 65 72 20 42 61 6C 6C 20 69 73 74 20 72 75 6E 64 2E
  - Interpretation als Folge von 1-Byte-Zahlen, im Bereich -128 ... 127 liefert:

    68 101 114 32 ...
  - Interpretation als Folge von 2-Byte-Zahlen liefert:

    17509 29216 16993
  - Interpretation als ASCII-Zeichenfolge liefert: ???

Daten und Informationen

- Wir halten also fest: **die Bedeutung von Daten erschließt sich erst durch die Kenntnis der benutzten Repräsentation**.
- Die Interpretation von Daten nennt man, wie wir wissen, Abstraktion.
- Mit einem Rechner kann man Daten lesen und verändern, z.B.
  - Lesen von Daten
  - Verknüpfung von Daten anhand arithmetischer oder logischer Operationen
  - Speichern von veränderten Daten auf einem externen Medium.
- Information wird also durch Daten repräsentiert.
- Wenn wir Information verarbeiten wollen, müssen wir die informations-verarbeitenden Operationen durch Operationen auf den entsprechenden Daten nachbilden.
- **Informationsverarbeitung** bedeutet demnach, dass Information zunächst auf Daten abgebildet wird, diese Daten verändert werden und aus den entstandenen Daten die neue Information abstrahiert wird.
- Das folgende Diagramm fasst das Gesagte noch einmal zusammen:

Daten und Informationen