Helmut Herold
Bruno Lurz
Jürgen Wohlrab

Grundlagen der Informatik

2., aktualisierte Auflage

ALWAYS LEARNING
PEARSON

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation
in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet
über <http://dnb.dnb.de> abrufbar.

10 9 8 7 6 5 4 3 2 1

14 13 12

ISBN 978-3-86894-111-1(Print);978-3-86326-526-7(PDF);978-3-86326-044-6(ePUB)

Programmleitung: Birger Peil, bpeil@pearson.de
Development: Alice Kachnij, akachnij@pearson.de
Korrektorat: Margarete Lurz, Herzogenaurach
Einbandgestaltung: Thomas Arlt, tarlt@adesso21.net
Herstellung: Monika Weiher, mweiher@pearson.de
Satz: le-tex publishing services GmbH, Leipzig
Druck und Verarbeitung: Drukarnia Dimograf, Bielsko-Biala

Printed in Poland

Speicherung und Interpretation von Information

3

ÜBERBLICK

3.1 Rätsel: Umfüllprobleme

1. Wie kann man 6 Liter Wasser von einem Fluss abfüllen, wenn zum Messen nur ein 4-Liter-Eimer und ein 9-Liter-Eimer zur Verfügung stehen?
2. Eine Bauersfrau soll aus einem oben offenen Bottich voll Essig genau einen Liter abmessen, hat dazu jedoch nur ein 3-l- und ein 5-l-Gefäß. Wie erreicht sie dies am besten?
3. Eine Kanne mit 8 Liter Fassungsvermögen ist vollgefüllt mit Wein. Wie kann man 4 Liter Wein abfüllen, wenn zwei leere Kannen mit 5 Liter und 3 Liter Fassungsvermögen zur Verfügung stehen?
4. In einem Fass befinden sich 18 Liter Wein. Diese Menge soll mittels eines 2-l-Bechers, eines 5-l-Kruges und eines 8-l-Eimers so verteilt werden, dass sich die Hälfte des Weines in dem Fass, ein Drittel des Weines in dem Eimer und ein Sechstel des Weines in dem Krug befindet. Welche Umfüllungen sind dazu notwendig?

3.2 Unterschiedliche Zahlensysteme

Als Beginn der Datenverarbeitung kann die Erfindung von Zahlensystemen und die Verarbeitung von Zahlen angesehen werden. Durch die Abbildung auf Zahlen können unterscheidbare Objekte, wie die Anzahl der Schafe in einer Herde oder die Anzahl von Getreidesäcken quantifiziert werden. Die Zahlen mussten dann miteinander verglichen, addiert oder subtrahiert, d. h. allgemein verarbeitet werden.

Zahlensysteme wurden in der Vergangenheit sehr unterschiedlich konzipiert. Fast alle Zahlensysteme beruhen auf dem Abzählen mit den Fingern. Es findet sich daher fast überall in mehr oder weniger unterschiedlichen Varianten das *Zehnersystem*.

3.2.1 Das römische Zahlensystem

In den Zahlensystemen der Ägypter und Römer wurde der Wert einer Zahl einfach durch die Form und die Anzahl der Zeichen bestimmt. Die Regeln des römischen Zahlensystems sind im Folgenden aufgeführt.

- Verfügbare Ziffern sind:
  `I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, M = 1000`
- Während die Ziffern `I`, `X`, `C` und `M` beliebig oft nebeneinander stehen können, dürfen die Ziffern `V`, `L` und `D` nicht wiederholt nebeneinander angegeben werden. Stehen gleiche Zeichen nebeneinander, so werden ihre Zahlenwerte addiert, wie z. B.:

II	=	2 = 1 + 1	XXX	=	30 = 10 + 10 + 10
CC	=	200 = 100 + 100	MMM	=	3000 = 1000 + 1000 + 1000

  Jedoch dürfen die Zeichen `I`, `X` und `C` nicht mehr als dreimal nebeneinander angegeben werden. `M` kann beliebig oft nebeneinander angegeben werden.
- Die Zeichen `V`, `L` und `D` dürfen in einer Zahl nur einmal vorkommen.
- Steht das Zeichen für eine kleinere Einheit rechts neben dem Zeichen einer größeren Einheit, dann wird die kleinere Einheit auf die größere addiert, wie z. B.

```
VI      =   6 = 5 + 1
XIII    =  13 = 10 + 1 + 1 + 1
DCCLVI  = 756 = 500 + 100 + 100 + 50 + 5 + 1
```

- Steht das Zeichen für eine kleinere Einheit links neben dem Zeichen einer größeren Einheit, dann wird die kleinere Einheit von der größeren subtrahiert, wie z. B.

```
IV   =  4 = 5 - 1
IX   =  9 = 10 - 1
XXIX = 29 = 10 + 10 + 10 - 1
```

- Es dürfen nicht zwei oder mehrere kleinere Einheiten von der rechts stehenden größeren Einheit abgezogen werden. Von zwei möglichen Schreibweisen wählt man heute meist die kürzere:

```
IL      =                    49 (XLIX = 49)
VD      =                   495 (XDV = 495)
MIM     = (MCMIC = 1999; MCMXCIX = 1999)
MDCCVL  =                (MDCCXLV = 1745)
```

- Tabelle von römischen Zahlen:

```
   I   1  |    X  10  |    C  100
  II   2  |   XX  20  |   CC  200
 III   3  |  XXX  30  |  CCC  300
  IV   4  |   XL  40  |   CD  400
   V   5  |    L  50  |    D  500
  VI   6  |   LX  60  |   DC  600
 VII   7  |  LXX  70  |  DCC  700
VIII   8  | LXXX  80  | DCCC  800
  IX   9  |   XC  90  |   CM  900
          |           |    M 1000
```

Um sich arabische Zahlen in römische bzw. umgekehrt umwandeln zu lassen, werden im Zusatzmaterial entsprechende begleitende Programme vorgestellt.

3.2.2 Positionssysteme

In den Systemen der Babylonier, Chinesen, Mayas und Inder hing der Wert einer Zahl von der Form und der Position der Zeichen ab. Solche Systeme heißen auch *Positions- oder Stellenwertsysteme*. Zur Darstellung benötigen sie ein zusätzliches Zeichen für die Ziffer 0. Der große Vorteil von Positionssystemen besteht darin, dass sie sehr einfache Rechenregeln besitzen. Unser heutiges Zahlensystem stammt aus Indien und gelangte über den nahen Osten zu uns, weshalb man auch heute noch von *arabischen Ziffern* spricht. Es ist ein Positionssystem mit der Basis zehn. Auch die ersten mechanischen Rechenmaschinen verwendeten das Zehnersystem.

Heutige elektronische Rechner verwenden das Dualsystem, ein Positionssystem, das mit zwei Ziffern 0 und 1 auskommt. Solche Dualzahlen besitzen bei gleichem Wert erheblich mehr Stellen, da eine Stelle ja nur zwei Werte repräsentieren kann. Der Grund für die Verwendung des Dualsystems in heutigen Rechnern ist allein der, dass es technisch erheblich einfacher ist viele elektronische Elemente mit nur jeweils

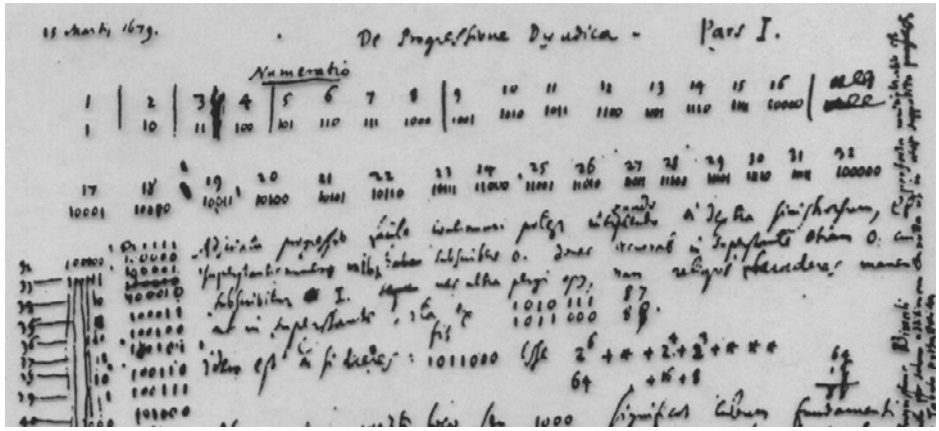

Abbildung 3.1: Leibniz-Traktat bezüglich Dualzahlen von 1679

zwei Zuständen (Strom bzw. kein Strom) zu bauen, als weniger Elemente mit dann jeweils zehn Zuständen für das Zehnersystem.

Bereits Leibniz kannte und beschäftigte sich mit dem Dualsystem, sein Ursprung liegt aber vermutlich schon erheblich früher in China. Unsere heutigen modernen DV-Maschinen können neben Zahlen auch alphanumerische Zeichen und Bilder speichern und verarbeiten. Die dazugehörigen Daten werden dabei in der Maschine ausschließlich binär kodiert (als Binärzahlen bestehend nur aus Nullen und Einsen) gespeichert.

3.2.3 Positionssysteme bei natürlichen Zahlen

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

- Eine natürliche Zahl `n` wird durch folgende Summe dargestellt:

$$n = \sum_{i=0}^{N-1} b_i \cdot B^i \qquad \text{wobei Folgendes gilt:}$$

  - `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

- Namen für einige Zahlensysteme:

  B = 2: *Dualsystem*
  B = 8: *Oktalsystem*
  B = 10: *Dezimalsystem*
  B = 16: *Hexadezimalsystem*
  B = 12: *Zwölfersystem* (in der Informatik nicht gebräuchlich)

▶ Übung

Wie viele Ziffern stehen im Dezimalsystem zur Verfügung? Beachten Sie den Unterschied zwischen *Zahl* und *Ziffer*!

- dezimal:

$$
\begin{aligned}
n = (2017)_{10} &= 2\cdot 10^3 + 0\cdot 10^2 + 1\cdot 10^1 + 7\cdot 10^0\\
\text{in Kurzform} &: 2\cdot 10^3 + \quad + 1\cdot 10^1 + 7\cdot 10^0\\
\text{oder} &: 2000 + \quad + 10 + 7
\end{aligned}
$$

$$
\begin{aligned}
n = (7508)_{10} &= 7\cdot 10^3 + 5\cdot 10^2 + 0\cdot 10^1 + 8\cdot 10^0\\
\text{in Kurzform} &: 7\cdot 10^3 + 5\cdot 10^2 + \quad + 8\cdot 10^0\\
\text{oder} &: 7000 + 500 + \quad + 8
\end{aligned}
$$

- oktal:

$$
\begin{aligned}
n = (315)_8 &= 3\cdot 8^2 + 1\cdot 8^1 + 5\cdot 8^0\\
&= 3\cdot 64 + 1\cdot 8 + 5\cdot 1\\
&= 192 + 8 + 5 = (205)_{10}
\end{aligned}
$$

$$
\begin{aligned}
n = (777)_8 &= 7\cdot 8^2 + 7\cdot 8^1 + 7\cdot 8^0\\
&= 7\cdot 64 + 7\cdot 8 + 7\cdot 1\\
&= 448 + 56 + 7 = (511)_{10}
\end{aligned}
$$

- dual:

$$
\begin{aligned}
n = (11001)_2 &= 1\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0\\
&= 1\cdot 16 + 1\cdot 8 + 0\cdot 4 + 0\cdot 2 + 1\cdot 1\\
&= 16 + 8 + 0 + 0 + 1 = (25)_{10}
\end{aligned}
$$

Da es für die „Ziffern“ `zehn`, `elf`, …, `fünfzehn` im Hexadezimalsystem keine eigene Zifferndarstellung gibt, nimmt man hierfür die Buchstaben `A`, `B`, `C`, `D`, `E`, `F` bzw. `a`, `b`, `c`, `d`, `e`, `f`.

▶ Übung

1. Wie viele Ziffern stehen im Oktalsystem zur Verfügung?
2. Geben Sie alle Ziffern des Oktalsystems an!
3. Wie viele Ziffern stehen im Hexadezimalsystem zur Verfügung?
4. Geben Sie alle Ziffern des Hexadezimalsystems an!

$(C9)_{16} = 12\cdot 16^1 + 9\cdot 16^0 = (201)_{10}$
$(fee)_{16} = 15\cdot 16^2 + 14\cdot 16^1 + 14\cdot 16^0 = (4078)_{10}$

Tabelle 3.1

Tabelle für die Zahlendarstellung in fünf verschiedenen Zahlensystemen

Dual	Oktal	Dezimal	Hexadezimal	Zwölfersystem
0	0	**0**	0	0
1	1	**1**	1	1
10	2	**2**	2	2
11	3	**3**	3	3
100	4	**4**	4	4
101	5	**5**	5	5
110	6	**6**	6	6
111	7	**7**	7	7
1000	10	**8**	8	8
1001	11	**9**	9	9
1010	12	**10**	a	a
1011	13	**11**	b	b
1100	14	**12**	c	10
1101	15	**13**	d	11
1110	16	**14**	e	12
1111	17	**15**	f	13
10000	20	**16**	10	14
10001	21	**17**	11	15

▶ Übung

Stellen Sie die folgenden Zahlen in ihrer Summenschreibweise dar und geben Sie ihre entsprechenden Werte im Dezimalsystem an:

$$(312)_4, (1202)_{16}, (ab1)_{12}, (101011)_2, (705)_8, (ABC)_{16}, (1111)_2, (127)_8$$

▶ Übung

In welchem Zahlensystem stellt folgende Gleichung eine wahre Aussage dar?
$42 + 242 = 16^2$

Das begleitende Programm `konvert.c`, das im Zusatzmaterial vorgestellt wird, liest eine Zahl aus einem beliebigen Zahlensystem ein und konvertiert diese dann in alle Zahlensysteme zwischen 2 und 36.

Chinesische Zahlen – Beispiel zu den Eigenschaften von Positionssystemen

Denken Sie sich eine Zahl zwischen 1 und 26 aus. Dann betrachten Sie nacheinander die folgenden sechs Tabellen:

1	4	7	**2**	5	8	**3**	4	5
10	13	16	11	14	17	12	13	14
19	22	25	20	23	26	21	22	23
6	7	8	**9**	10	11	**18**	19	20
15	16	17	12	13	14	21	22	23
24	25	26	15	16	17	24	25	26

Befindet sich die ausgewählte Zahl in einer der Tabellen, so schreiben Sie die Zahl auf, die sich oben links (fett gedruckt) in dieser Tabelle befindet. Danach addieren Sie die aufgeschriebenen Zahlen. So kommt immer wieder die zu Anfang gewählte Zahl als Ergebnis heraus. Z. B. ist die Zahl 17 im zweiten, im vierten und im fünften Quadrat enthalten. Wenn man die drei ersten Zahlen dieser Quadrate addiert, ergibt sich: 2 + 6 + 9 = 17.

Bei diesen Tabellen handelt es sich um eine geschickte Kodierung für das 3er System. Darin ist 3 die Basis und zur Darstellung einer Zahl stehen die Ziffern 0, 1 und 2 zur Verfügung. Wenn man 3 Stellen zur Verfügung hat, so kann man im 3er System alle Zahlen zwischen 0 und 26 darstellen:

$$
\begin{aligned}
(0)_3 &= (0)_{10} \\
(1)_3 &= (1)_{10} = 1*3^0 \\
(2)_3 &= (2)_{10} = 2*3^0 \\
(10)_3 &= (3)_{10} = 1*3^1 + 0*3^0 \\
(11)_3 &= (4)_{10} = 1*3^1 + 1*3^0 \\
&\quad \ldots\ldots\ldots \\
(121)_3 &= (16)_{10} = 1*3^2 + 2*3^1 + 1*3^0 \\
(122)_3 &= (17)_{10} = 1*3^2 + 2*3^1 + 2*3^0 \\
&\quad \ldots\ldots\ldots \\
(221)_3 &= (25)_{10} = 2*3^2 + 2*3^1 + 1*3^0 \\
(222)_3 &= (26)_{10} = 2*3^2 + 2*3^1 + 2*3^0
\end{aligned}
$$

Allgemein gilt für eine Zahl n:

$$n = (0 \text{ oder } 1 \text{ oder } 2)\cdot 3^2 + (0 \text{ oder } 1 \text{ oder } 2)\cdot 3^1 + (0 \text{ oder } 1 \text{ oder } 2)\cdot 3^0$$

$$n = (0 \text{ oder } \mathbf{9} \text{ oder } \mathbf{18}) + (0 \text{ oder } \mathbf{3} \text{ oder } \mathbf{6}) + (0 \text{ oder } \mathbf{1} \text{ oder } \mathbf{2})$$

Die fett gedruckten Zahlen finden Sie in den linken oberen Ecken der Tabellen wieder. Wird nun eine Zahl entsprechend dem 3er System zerlegt, dann befindet sich diese Zahl, falls die Zerlegung eine 1 enthält, in der ersten Tabelle. Enthält die Zerlegung eine 2, so befindet sich diese Zahl in der zweiten Tabelle usw. Enthält die Zerlegung eine 9, so befindet sich diese Zahl in der sechsten Tabelle. Folglich erhält man wieder die Ausgangszahl n, indem man die einzelnen Zahlen aus den oberen Ecken der betroffenen Tabellen addiert.

▶ Übung: Der Computer errät eine gedachte Ziffernfolge („Superhirn“)

Nehmen wir an, dass Sie ein Programm schreiben sollen, das das Spiel „Moo“ realisiert. Beim Spiel „Moo“ handelt es sich um eine Computerversion zu dem bekannten Spiel „Superhirn“ (auch unter dem Namen „Mastermind“ bekannt). Die Aufgabe Ihres Programms ist es dabei, eine vom Benutzer ausgedachte Zahlenkombination zu erraten. Wie viele Ziffern (z) und Positionen (p) zur Verfügung stehen, muss der Benutzer am Anfang eingeben. Danach soll das Programm dem Benutzer immer Lösungsvorschläge vorgeben. Der Benutzer muss dann eingeben, wie viele Ziffern in diesem Lösungsvorschlag an der richtigen Position sind und wie viele Ziffern zwar richtig sind, aber sich noch an der falschen Position befinden. Mögliche Abläufe der begleitenden Programme `moo.c` und `Moo.java`:

```
Wie viele Ziffern:  10
Wie viele Positionen: 3
0 0 0    ? 0,0
1 1 1    ? 0,0
2 2 2    ? 0,0
3 3 3    ? 1,0
4 4 3    ? 0,1
5 3 5    ? 1,0
6 3 6    ? 1,0
7 3 7    ? 2,0
8 3 7    ? 2,0
 ....  Ok, ich habe die Kombination gefunden: 9 3 7
```

```
Wie viele Ziffern:  5
Wie viele Positionen: 6
0 0 0 0 0 0    ? 1,0
1 1 1 1 1 0    ? 0,1
2 2 2 2 0 2    ? 2,0
3 3 3 3 0 2    ? 2,1
4 4 3 2 0 4    ? 3,3
4 2 4 3 0 4    ? 3,3
 ....  Ok, ich habe die Kombination gefunden: 4 3 2 4 0 4
```

Ihre Vorgehensweise ist dabei folgende:

- Sie speichern sich zunächst alle möglichen Kombinationen.
- Immer wenn der Benutzer den Computervorschlag bewertet, also die richtigen Positionen und Ziffern eingegeben hat, geht das Programm wie folgt vor: Es bewertet alle noch nicht gestrichenen Kombinationen, indem es für jede Kombination annimmt, dass dies eine mögliche richtige Lösung wäre. Handelt es sich bei dieser Kombination um eine potenzielle Lösung, so müsste für diese Kombination die Benutzerbewertung bezüglich des Computervorschlags zutreffen. Trifft dies nicht zu, wird diese Kombination gestrichen. Dieses Verfahren wird für jede noch nicht gestrichene Kombination durchgeführt.
- Anschließend bietet dieses Programm dem Benutzer die erste noch nicht gestrichene Kombination zur erneuten Bewertung an usw.

Beantworten Sie zu dieser Aufgabenstellung nun folgende Fragen:

- Wie viele Kombinationen gibt es bei `z` Ziffern und `p` Positionen?
- Geben Sie für folgende Konstellationen alle möglichen Kombinationen an!

Ziffern	Positionen	Kombinationen
2	3	
4	2	
5	2	

- Können Sie aus dieser Tabelle Rückschlüsse auf Positionssysteme ziehen?
- Beschreiben Sie, wie man abhängig von der Ziffernzahl z und der Positionszahl p die jeweils benötigten Kombinationen erzeugen kann!

3.2.4 Positionssysteme bei gebrochenen Zahlen

Bei gebrochenen Zahlen trennt ein Punkt (Komma im Deutschen) in der Zahl den ganzzahligen Teil der Zahl vom gebrochenen Teil (Nachkommateil). Solche Zahlen lassen sich durch folgende Summenformel beschreiben:

$$n = \sum_{i=-M}^{N-1} b_i \cdot B^i \qquad \text{wobei Folgendes gilt:}$$

- `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 vor dem Punkt (Komma)
- `M` = Anzahl der Stellen nach dem Punkt (Komma)

$$\begin{array}{lcllllll}
(17.05)_{10} & = & 1 \cdot 10^1 & + & 7 \cdot 10^0 & + & 0 \cdot 10^{-1} & + & 5 \cdot 10^{-2} \\
(3758.0)_{10} & = & 3 \cdot 10^3 & + & 7 \cdot 10^2 & + & 5 \cdot 10^1 & + & 8 \cdot 10^0 \\
(9.702)_{10} & = & 9 \cdot 10^0 & + & 7 \cdot 10^{-1} & + & 0 \cdot 10^{-2} & + & 2 \cdot 10^{-3} \\
(0.503)_{10} & = & 0 \cdot 10^0 & + & 5 \cdot 10^{-1} & + & 0 \cdot 10^{-2} & + & 3 \cdot 10^{-3}
\end{array}$$

► Übung: Geben Sie zu folgenden Zahlen die Summenform und die Darstellung im Dezimalsystem an:

$$(1573.4)_8\,, \quad (ABC.CBA)_{16}\,, \quad (1011.1101)_2\,, \quad (0.4)_8$$

▶ Übung: Formel zu π

Durch welche der drei folgenden Summendarstellungen lässt sich $\pi = 3.1415927\ldots$ darstellen? Geben Sie zu den entsprechenden Möglichkeiten die Werte zu `m`, `n`, a_0, a_{-1} und a_{-7} an:

$$1.\sum_{i=0}^{n-1} a_i \cdot 10^i, \quad 2.\sum_{i=-m}^{n-1} a_i \cdot 10^i, \quad 3.\sum_{i=-\infty}^{n-1} a_i \cdot 10^i$$

3.3 Dual-, Oktal- und Hexadezimalsystem

In der Informatik spielen das Dual-, Oktal- und Hexadezimalsystem eine zentrale Rolle.

3.3.1 Das Dualsystem und das Bit im Rechner

Nochmals zur Wiederholung: Das von uns verwendete Zehnersystem ist ein Positionssystem. Dies bedeutet, dass jeder Position in einer Zahl ein bestimmter Wert zugeordnet wird, der eine Potenz von 10 ist.

Da das Zehnersystem, in dem 10 verschiedene Ziffern $0, 1, 2, \ldots, 9$ existieren, technisch schwer zu realisieren ist, benutzt man in Rechnern intern das Dualsystem, bei dem nur zwei Ziffern, 0 und 1, verwendet werden. Die beiden Ziffern des Dualsystems lassen sich technisch relativ leicht nachbilden:

$$0 = \text{kein Strom, keine Spannung}$$
$$1 = \text{Strom, Spannung}$$

Eine einzelne Binärstelle (0 oder 1), die ein Rechner speichert, wird als **Bit** bezeichnet. Das ist die Abkürzung für „*BInary digiT*“, also Binärziffer. Es handelt sich dabei um die kleinste Informationseinheit, die ein Computer verarbeiten kann.

Wie wir zuvor gesehen haben, handelt es sich auch beim Dualsystem um ein Positionssystem, in dem jeder Position in einer Zahl ein bestimmter Wert zugeordnet wird, der jedoch hier nun eine Potenz von 2 ist:

$$10011 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 16 + 0 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1$$
$$= 19 \text{ (im Zehnersystem)}$$

Das Zahlensystem gibt man dabei meist tiefgestellt an, wie z. B.:

$(10011)_2$	= $(19)_{10}$	oder:	$10011_{(2)}$	=	$19_{(10)}$
$(1011)_2$	= $(11)_{10}$	oder:	$1011_{(2)}$	=	$11_{(10)}$
$(11010110)_2$	= $(214)_{10}$	oder:	$11010110_{(2)}$	=	$214_{(10)}$

3.3.2 Konvertieren zwischen Dual- und Oktalsystem

Neben dem Dualsystem ist in der Informatik noch das Oktalsystem wichtig, da es in einer engen Beziehung zum Dualsystem steht. Es gilt nämlich: $2^3 = 8$ (Basis des Oktalsystems).

▶ Übung: Wie viele Dualstellen werden zur Darstellung der Ziffern im Oktalsystem maximal benötigt?

- Um eine im Dualsystem dargestellte Zahl ins Oktalsystem zu konvertieren, bildet man von rechts beginnend so genannte *Dualtriaden* (Dreiergruppen). Nachfolgend wird dies anhand der Dualzahl $110111001110010_{(2)}$ gezeigt, die der Oktalzahl $67162_{(8)}$ entspricht.

```
110  111  001  110  010  | Dualzahl
 6    7    1    6    2   | Oktalzahl
```

- Bei der Umwandlung einer Oktalzahl in ihre Dualdarstellung geht man den umgekehrten Weg. Nachfolgend wird dies anhand der Oktalzahl $3614_{(8)}$ gezeigt, die der Dualzahl $011110001100_{(2)}$ entspricht.

```
 3    6    1    4   | Oktalzahl
011  110  001  100  | Dualzahl
```

Es ist offensichtlich, dass ein Mensch sich die Zahl $3614_{(8)}$ wesentlich leichter merken kann als $011110001100_{(2)}$. Die Konvertierung von dieser leicht merkbaren Oktalzahl in die zugehörige Dualzahl ist dann – wie wir gesehen haben – sehr einfach möglich.

3.3.3 Konvertieren zwischen Dual- und Hexadezimalsystem

Neben dem Dualsystem ist in der Informatik des Weiteren noch das Hexadezimalsystem wichtig, da es in einer engen Beziehung zum Dualsystem steht. Es gilt nämlich: $2^4 = 16$ (Basis des Hexadezimalsystems).

▶ Übung

Geben Sie die Dualdarstellung der Ziffern des Hexadezimalsystems an! Wie viele Dualstellen werden zur Darstellung der Hexdezimalziffern maximal benötigt?

Um eine im Dualsystem dargestellte Zahl ins Hexadezimalsystem zu konvertieren, bildet man von rechts beginnend so genannte *Dualtetraden* (Vierergruppen).

$(\mathbf{ADA})_{16} = (\mathbf{101011011010})_2 = (\mathbf{5332})_8$

Hexadezimal: A D A | Dreiergruppen: 101 011 011 010

Vierergruppen: 1010 1101 1010 | Oktal: 5 3 3 2

$(\mathbf{753})_8 = (\mathbf{111101011})_2 = (\mathbf{1EB})_{16}$

Oktal: 7 5 3 | Vierergruppen: 1 1110 1011

Dreiergruppen: 111 101 011 | Hexadezimal: 1 E B

$(\mathbf{1011101011101})_2 = (\mathbf{175D})_{16} = (\mathbf{13535})_8$

Vierergruppen: 1 0111 0101 1101 | Dreier: 1 011 101 011 101

Hexadezimal: 1 7 5 D | Oktal: 1 3 5 3 5

$(\mathbf{11010111111111010})_2 = (\mathbf{1AFFA})_{16} = (\mathbf{327772})_8$

Vierer: 1 1010 1111 1111 1010 | Dreier: 11 010 111 111 111 010

Hexa: 1 A F F A | Oktal: 3 2 7 7 7 2

Es ist wieder offensichtlich, dass ein Mensch sich die Zahl $1EB_{(16)}$ wesentlich leichter merken kann als $111101011_{(2)}$. Die Konvertierung von dieser leicht merkbaren Hexadezimalzahl in die zugehörige Dualzahl ist dann – wie wir gesehen haben – sehr einfach möglich. Um also eine im Dualsystem dargestellte Zahl im Hexadezimalsystem (Oktalsystem) darzustellen, ist folgendermaßen vorzugehen:

1. Man teile die Ziffernfolge der Dualdarstellung von rechts nach links in Tetraden (Triaden).
2. Man ersetze die Dualtetraden (Dualtriaden) durch die ihnen entsprechenden Ziffern des Hexadezimalsystems (Oktalsystems) und den Basisindex 2 durch 16 (8).

▶ Übung:

Konvertieren Sie möglichst effizient die Zahl $(ABBA)_{16}$ in das Oktalsystem!

Die obigen Regeln gelten auch für gebrochene Zahlen, wenn man Dualtetraden bzw. -triaden vom Punkt (Komma) aus nach links und rechts bildet.

▶ Übung: Duale, oktale und hexadezimale Darstellung von gebrochenen Zahlen

Ergänzen Sie die folgenden Tabellen, so dass die jeweilige gebrochene Zahl in allen drei Darstellungsformen (dual, oktal und hexadezimal) vorliegt!

Dualsystem	Oktalsystem	Hexadezimalsystem
110 1110,0011		
		ABC,DE

3.4 Konvertierungsalgorithmen

3.4.1 Konvertieren von anderen Systemen in das Dezimalsystem

Eine in einem Positionssystem mit der 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:

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

$(1578)_{10} = ((1 \cdot 10 + 5) \cdot 10 + 7) \cdot 10 + 8$
$(754)_8 = (7 \cdot 8 + 5) \cdot 8 + 4 = (492)_{10}$

Mit Hilfe dieser Darstellung können Konvertierungen in das Dezimalsystem einfach durchgeführt werden.

▶ Übung

Konvertieren Sie folgende Zahlen unter Zuhilfenahme des Hornerschemas in das Dezimalsystem: $(375)_8$, $(1210)_8$, $(888)_9$, $(ADA)_{16}$

3.4.2 Konvertieren vom Dezimalsystem in andere Positionssysteme

Für die Umwandlung einer Dezimalzahl `x` in ein Zahlensystem mit der Basis `n` kann folgender Algorithmus 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 Dualzahl.

Nachfolgend zwei Beispiele für die Umwandlung einer Zahl aus dem Dezimal- in das Dualsystem:

$(30)_{10} = ?_2$

x			y		z
30	: 2 =	15	Rest	0	
15	: 2 =	7	Rest	1	
7	: 2 =	3	Rest	1	
3	: 2 =	1	Rest	1	
1	: 2 =	0	Rest	1	

$(43)_{10} = ?_2$

x			y		z
43	: 2 =	21	Rest	1	
21	: 2 =	10	Rest	1	
10	: 2 =	5	Rest	0	
5	: 2 =	2	Rest	1	
2	: 2 =	1	Rest	0	
1	: 2 =	0	Rest	1	

Die Reste z von unten nach oben nebeneinander geschrieben liefern dann die gesuchte Dualzahl: $\mathbf{(30)_{10} = 11110_2}$ $\mathbf{(43)_{10} = (101011)_2}$

▶ Übung: Wandeln Sie die folgenden Dezimalzahlen in das entsprechende Positionssystem um

$(445)_{10}$ = in das Dualsystem
$(7294)_{10}$ = in das Oktalsystem
$(87599)_{10}$ = in das Hexadezimalsystem
$(1234)_{10}$ = in das Siebenersystem
$(77875)_{10}$ = in das Dreiersystem
$(754398)_{10}$ = in das Dualsystem

Kann man diese letzte Zahl eventuell auch effizienter konvertieren?

Zum Konvertieren von Zahlen können Sie zum einen das auf Seite 52 erwähnte Programm `konvert.c` verwenden oder aber auch die beiden begleitenden Programme `dezkonvert.c` und `Dezkonvert.java`, die das Konvertieren von Dezimalzahlen schrittweise anzeigen, wie z. B.:

```
Gib Basis des Zielsystems ein (2<=Basis<=36): 16
Gib die zu wandelnde Zahl aus dem Zehnersystem ein: 45054
45054 : 16 =  2815 Rest 14 (E)
 2815 : 16 =   175 Rest 15 (F)
  175 : 16 =    10 Rest 15 (F)
   10 : 16 =     0 Rest 10 (A)
      ---> 45054(10) = AFFE(16)
```

3.4.3 Konvertieren echt gebrochener Zahlen

Eine echt gebrochene Zahl `n` ($n < 1$):

$$n = \sum_{i=-M}^{-1} b_i \cdot B^i$$

lässt sich auch mit Hilfe des *Hornerschemas* wie folgt darstellen:

$$n = \frac{1}{B} \cdot \left(b_{-1} + \frac{1}{B} \cdot \left(b_{-2} + \frac{1}{B} \cdot \left(b_{-3} + \ldots + \frac{1}{B} \cdot \left(b_{-M+1} + \frac{1}{B} \cdot b_{-M} \right) \ldots \right) \right) \right)$$

wie z. B. die Zahl:

$$0.193_{(10)} = \frac{1}{10} \cdot \left(1 + \frac{1}{10} \cdot \left(9 + \frac{1}{10} \cdot 3 \right) \right)$$

Mit Hilfe dieser Darstellung können wieder Konvertierungen von anderen Systemen in das Dezimalsystem einfach durchgeführt werden.

Algorithmus zur Konvertierung echt gebrochener Dezimalzahlen

Für die Umwandlung des Nachkommateils einer Dezimalzahl in ein anderes Positionssystem existiert folgender Algorithmus, wobei `B` die Basis des Zielsystems ist:

1. $x \cdot B = y$ Überlauf z (`z` = ganzzahliger Anteil)
2. Mache Nachkommateil von y zum neuen x und fahre mit Schritt 1 fort, wenn dieses neue x ungleich 0 ist und noch nicht genügend Nachkommastellen ermittelt sind, ansonsten fahre mit Schritt 3 fort.
3. Schreibe die ermittelten Überläufe von oben nach unten nach `0.` nebeneinander, um die entsprechende Dualzahl zu erhalten.

$(\mathbf{0.34375})_{\mathbf{10}} = (\mathbf{0.01011})_{\mathbf{2}}$

x		y		z
0.34375	· 2 =	0.6875	Ueberl.	0
0.6875	· 2 =	1.375	Ueberl.	1
0.375	· 2 =	0.75	Ueberl.	0
0.75	· 2 =	1.5	Ueberl.	1
0.5	· 2 =	1.0	Ueberl.	1
0	· 2 =	0.0	Ueberl.	0

Die Überläufe z von oben nach unten nach 0. nebeneinander geschrieben liefern dann die gesuchte Zahl.

$(\mathbf{0.408203125})_{\mathbf{10}} = (\mathbf{0.321})_{\mathbf{8}}$

x		y		z
0.408203125	· 8 =	3.265625	Ueberl.	3
0.265625	· 8 =	2.125	Ueberl.	2
0.125	· 8 =	1	Ueberl.	1
0	· 8 =	0	Ueberl.	0

Genauigkeitsverluste bei der Umwandlung gebrochener Dezimalzahlen

Manche gebrochenen Zahlen, die sich ganz genau im Dezimalsystem darstellen lassen, lassen sich leider nicht ganz genau als Dualzahl darstellen. Typische Beispiele dafür sind Zahlen, die sich im Dualsystem nur durch eine periodische Ziffernfolge repräsentieren lassen, wie z. B. $0.1_{(10)} = 0.0001100110011\ldots_{(2)}$:

x	y	z
0.1 ∗ 2 =	0.2	Überlauf 0
0.2 ∗ 2 =	0.4	Überlauf 0
0.4 ∗ 2 =	0.8	Überlauf 0
0.8 ∗ 2 =	1.6	Überlauf 1
0.6 ∗ 2 =	1.2	Überlauf 1
0.2 ∗ 2 =	0.4	Überlauf 0
0.4 ∗ 2 =	0.8	Überlauf 0
0.8 ∗ 2 =	1.6	Überlauf 1
0.6 ∗ 2 =	1.2	Überlauf 1

Das Bitmuster `0011` wiederholt sich hier ständig und es gilt somit:

$$0.1_{(10)} = 0.0\ \ 0011\ \ 0011\ldots_{(2)}$$

Solche Ungenauigkeiten treten dann natürlich auch in den Rechnern auf, die ja mit dem Dualsystem arbeiten. Darauf wird später noch näher eingegangen.

▶ Übung: Konvertieren Sie die folgenden Zahlen!

$(0.375)_{10}$ = im Dualsystem?
$(0.25)_{10}$ = im Fünfersystem?
$(0.19)_{10}$ = im Hexadezimalsystem?

Zum Konvertieren von echt gebrochenen Zahlen können Sie auch die begleitenden Programme `gebrkonv.c` und `Gebrkonv.java` verwenden, die im Zusatzmaterial vorgestellt werden und die das Konvertieren von echt gebrochenen Dezimalzahlen schrittweise anzeigen.

3.4.4 Konvertieren unecht gebrochener Zahlen

Um eine unecht gebrochene Zahl zu konvertieren, muss diese in ihren ganzzahligen Teil und ihren echt gebrochenen Teil aufgeteilt werden, die dann getrennt von einander zu konvertieren sind.

$(12.25)_{10} = (1100.01)_2$

Ganzzahliger Teil: $(12)_{10} = (1100)_2$	**Echt gebrochener Teil:** $(0.25)_{10} = (0.01)_2$
12 : 2 = 6 Rest 0	0.25 * 2 = 0.5 Überlauf 0
6 : 2 = 3 Rest 0	0.5 * 2 = 1 Überlauf 1
3 : 2 = 1 Rest 1	0 * 2 = 0 Überlauf 0
1 : 2 = 0 Rest 1	

3.5 Rechenoperationen im Dualsystem

3.5.1 Addition

Für die **duale Addition** gilt allgemein:

```
0 + 0                       = 0
0 + 1                       = 1
1 + 0                       = 1
1 + 1                       = 0 Übertrag 1
1 + 1 + 1 (vom Übertrag)    = 1 Übertrag 1
```

```
  0 1 0 1 1 0 1 =  45
  0 1 1 0 1 1 0 =  54
-----------------
  1 1 1 1       = Übertrag
-----------------
  1 1 0 0 0 1 1 =  99
```

▶ Übung: Addition im Dualsystem

Lösen Sie die folgenden Aufgaben, indem Sie die Dezimalzahlen zuerst in das Dualsystem umwandeln und dann im Dualsystem die Addition durchführen:
$(123)_{10} + (204)_{10} = ?_2$, $(15)_{10} + (31)_{10} = ?_2$, $(105)_{10} + (21)_{10} = ?_2$

► Übung: Addieren Sie die folgenden Dualzahlen:

```
1)              2)              3)              4)
  0 1 0 1 1 0 1   0 1 1 0 0 1     0 1 1 0 0 1 0   0 1 0 1 1 0 1
+ 0 0 0 1 0 1 1 + 0 0 1 1 0 0   + 0 0 1 1 0 1 0 + 0 0 0 1 1 1 1
+ 0 0 1 0 0 0 1 + 0 0 0 0 1 1   + 0 0 0 1 1 0 0 + 0 0 0 1 0 0 0
+ 0 0 0 1 0 1 0 + 0 0 1 0 0 1   + 0 0 1 0 0 1 1 + 0 0 1 0 1 0 1
                                                + 0 0 0 1 1 0 1
```

Zum Addieren von Dualzahlen können Sie auch die begleitenden Programme `dualadd.c` und `Dualadd.java` verwenden, die im Begleitmaterial vorgestellt werden.

3.5.2 Subtraktion und Darstellung negativer Zahlen

Negative Zahlen werden üblicherweise durch ihren Betrag mit vorangestelltem Minuszeichen dargestellt. Diese Darstellung wäre auch rechnerintern denkbar, hat jedoch den Nachteil, dass man eine gesonderte Vorzeichenrechnung durchführen müsste und man ein Rechenwerk benötigt, das sowohl addieren als auch subtrahieren kann. Um mit einem reinen Addierwerk auszukommen, versucht man, die Subtraktion auf eine Addition zurückzuführen. Dies geschieht durch das Verfahren der *Komplementbildung*. Man unterscheidet zwei Arten der Komplementbildung, wobei `B` für das Zahlensystem steht: *B-Komplement* und *(B-1)-Komplement*

Im Dualsystem könnte man also mit dem *Zweier-Komplement* (B-Komplement) oder mit dem *Einer-Komplement* ((B-1)-Komplement) arbeiten.

Da das B-Komplement technisch leichter realisierbar ist, wird vorwiegend mit dem B-Komplement (Zweier-Komplement) gearbeitet. Der Vollständigkeit halber und zum Vergleich werden hier beide Komplemente vorgestellt.

Negation von Zahlen mit dem B-Komplement (Zweier-Komplement)

Wir nehmen hier einmal an, dass wir vier Bits zur Verfügung haben, wobei das erste Bit das Vorzeichenbit ist. Hierfür wären dann die in Abbildung 3.2 gezeigten Bitkombinationen möglich.

Unter Verwendung eines Zahlenrings wird dann die in Abbildung 3.2 gezeigte Zuordnung von ganzen Zahlen getroffen. In dieser Darstellung wird die Zahl Null $(000\ldots00_{(2)})$ als positive Zahl aufgefasst. Dadurch wird die Darstellung *unsymmetrisch*, denn es gilt bei *s* verfügbaren Stellen Folgendes:

- kleinste darstellbare negative Zahl: $-B^{s-1}$: Im Zweier-Komplement gilt somit für die kleinste darstellbare negative Zahl: -2^{s-1}:

  bei s = 4: $-2^{4-1} = -2^3 = -8$
  bei s = 8: $-2^{8-1} = -2^7 = -128$
  bei s = 16: $-2^{16-1} = -2^{15} = -32768$
  bei s = 32: $-2^{32-1} = -2^{31} = -2147483648$

- größte darstellbare positive Zahl: $B^{s-1} - 1$: Im Zweier-Komplement gilt somit für die größte darstellbare positive Zahl: $2^{s-1} - 1$:

  bei s = 4: $2^{4-1} - 1 = 2^3 - 1 = 7$
  bei s = 8: $2^{8-1} - 1 = 2^7 - 1 = 127$
  bei s = 16: $2^{16-1} - 1 = 2^{15} - 1 = 32767$
  bei s = 32: $2^{32-1} - 1 = 2^{31} - 1 = 2147483647$

Mit unseren vier Bits könnten wir also Zahlen aus dem Wertebereich `-8..7` darstellen. Hier drängt sich jetzt nur noch die Frage auf, nach welchem Prinzip die einzelnen negativen Zahlen den entsprechenden Bitkombinationen zugeordnet werden.

Alle Kombinationen, bei denen das 1. Bit (Vorzeichenbit) gesetzt ist, repräsentieren dabei negative Zahlen:

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

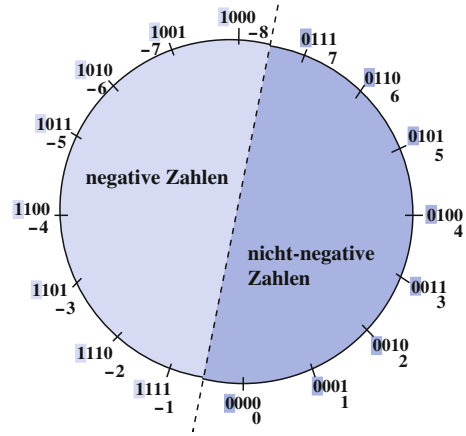

Abbildung 3.2: Zahlenring für vier Bits, wobei das erste Bit das Vorzeichenbit ist

Regeln für die Bildung eines Zweier-Komplements

1. Ist das 1. Bit mit 1 besetzt, so handelt es sich um eine negative Zahl.
2. Der Wert einer negativen Zahl wird dabei im Zweier-Komplement dargestellt. Zweier-Komplement zu einem Wert bedeutet dabei, dass zunächst jedes einzelne Bit invertiert (umgedreht) wird, und dann auf die so entstandene Bitkombination die Zahl 1 aufaddiert wird.

Zweier-Komplement zu 5:
Dualdarstellung von 5: 0101

Komplementieren von 5: 1010
+ 1: 0001
= −5: 1011

Zweier-Komplement zu -5:
Dualdarstellung von −5: 1011

Komplementieren von −5: 0100
+ 1: 0001
= 5: 0101

Der Vorteil einer solchen Komplement-Darstellung ist, dass eine Maschine nicht subtrahieren können muss, sondern jede Subtraktion `a - b` durch eine Addition `a + -b` realisieren kann, wie es in den Beispielen von Abbildung 3.3 gezeigt ist.

In Abbildung 3.3 hat der vorne stattfindende Überlauf des Bits keinen Einfluss auf die Richtigkeit des Ergebnisses. Das gilt allerdings nicht allgemein. Wenn nämlich das Ergebnis nicht im darstellbaren Zahlenbereich liegt, dann erhält man bei einem Überlauf ein falsches Ergebnis, wie es das folgende Beispiel zeigt.

```
2 - 4 = 2 + -4
   0010 =   2
+  1100 =  −4
---------
   1110 =  −2
```

```
6 - 2 = 6 + -2
    0110 =   6
+   1110 =  −2
----------
  1|0100 =   4   Das vorne überlaufende Bit wird weggeworfen
```

Abbildung 3.3: Addition mit und ohne Überlauf

Bei fünf verfügbaren Stellen soll die Subtraktion $(-9)_{10} - (13)_{10}$ im Dualsystem mit Hilfe des B-Komplements durchgeführt werden.

Darstellbarer Zahlenbereich: $-2^4..2^4 - 1 = -16.. + 15$

$$\begin{array}{rl} & -(9)_{10}: \quad (10111)_2 \\ + & (-13)_{10}: \quad (10011)_2 \\ \hline & (+10)_{10}: 1|\,(01010)_2 \end{array}$$

Das vorne überlaufende Bit geht verloren → falsches Ergebnis

▶ Übung

Bilden Sie zu den folgenden Zahlen das entsprechende B-Komplement:
$10101_{(2)}$, $785_{(10)}$, $AFFE_{(16)}$, $453_{(16)}$, $124_{(5)}$

▶ Übung

Subtrahieren Sie die folgenden Zahlen im B-Komplement mit 8 verfügbaren Stellen und `B=2`:

$$(57)_{10} - (122)_{10}$$
$$(43)_{10} - (11)_{10}$$
$$(17)_{10} - (109)_{10}$$

▶ Übung

Subtrahieren Sie die folgenden Zahlen im B-Komplement mit 5 verfügbaren Stellen und `B=10`:

$$(25737)_{10} - (18547)_{10}$$
$$(2737)_{10} - (4578)_{10}$$

Zur B-Komplementbildung und zum Subtrahieren von Zahlen im B-Komplement können Sie auch das begleitende Programm `subtraktion.c` verwenden, das im Begleitmaterial zu diesem Buch vorgestellt wird.

Negation von Zahlen mit dem (B-1)-Komplement (Einer-Komplement)

Wie bereits zuvor erwähnt, lässt sich das B-Komplement technisch leichter realisieren, weshalb auch vorwiegend mit dem B-Komplement (Zweier-Komplement) gearbeitet. Der Vollständigkeit halber und zum Vergleich wird hier das (B-1)-Komplement (Einer-Komplement) vorgestellt. Wir nehmen hier an, dass wir vier Bits zur Verfügung haben, wobei das erste Bit das Vorzeichenbit ist. Hierfür wären dann folgende Bitkombinationen möglich:

```
0000 +0  (positive Null)
0001  1
0010  2
0011  3
0100  4
0101  5
0110  6
0111  7
---------------------------------------
1000 −7
1001 −6
1010 −5
1011 −4  Alle Kombinationen, bei denen das 1. Bit (Vorzeichenbit)
1100 −3  gesetzt ist, repräsentieren dabei negative Zahlen.
1101 −2
1110 −1
1111 −0  (negative Null)
```

Anders als beim B-Komplement ist die Zahlendarstellung hierbei *symmetrisch*.

Regeln für die Bildung eines Einer-Komplements

1. Ist das 1. Bit mit 1 besetzt, so handelt es sich um eine negative Zahl (eventuell die negative Null 111…111).
2. Der Wert einer negativen Zahl wird dabei im Einer-Komplement dargestellt. Einer-Komplement zu einem Wert bedeutet dabei, dass zunächst jedes einzelne Bit invertiert (umgedreht) wird.
3. Führt die Addition des Komplements zu einem Überlauf einer 1, muss zu dem Ergebnis noch diese 1 hinzuaddiert werden („*Einer-Rücklauf*“).

$(14)_{10} - (7)_{10}$ im (B-1)-Komplement
bei 5 verfügbaren Stellen und `B=2`:

Einer-Komplement zu $(7)_{10} = (00111)_2$**:**
Komplementieren von 7 (−7): 11000
Dualdarstellung von 14 : 01110
+ −7 : 11000
−−−−−−−−−−−−−
: 1|00110
Aufaddieren von 1 : 00001
−−−−−−−−−−−−−
= 7 : 00111

$(9)_{10} - (13)_{10}$ im (B-1)-Komplement
bei 5 verfügbaren Stellen und `B=2`:

Einer-Komplement zu $(13)_{10} = (01101)_2$**:**
Komplementieren von 13 (−13): 10010
Dualdarstellung von 9: 01001
+ −13: 10010
−−−−−−−−−−−−−
= −4: 11011

▶ Übung

Subtrahieren Sie folgende Zahlen im (B-1)-Komplement mit 8 verfügbaren Stellen und `B=2`:

$(57)_{10} - (122)_{10}$, $(43)_{10} - (11)_{10}$, $(17)_{10} - (109)_{10}$

Subtrahieren Sie folgende Zahlen im (B-1)-Komplement mit 5 verfügbaren Stellen und `B=10`:

$(25737)_{10} - (18547)_{10}$ $(2737)_{10} - (4578)_{10}$

3.5.3 Multiplikation und Division

Die ganzzahlige Multiplikation bzw. Division wird in einem Rechner zwar allgemein mittels wiederholter Addition durchgeführt, aber in den Sonderfällen des Multiplikators bzw. Divisors von 2, 4, 8, … kann die Multiplikation bzw. Division einfach auch durch eine Verschiebung von entsprechend vielen Bits nach links bzw. rechts erfolgen: Bei 2 (2^1) um 1 Bit, bei 4 (2^2) um 2 Bits, bei 8 um 3 (2^3) Bits usw.

dezimal : $(20)_{10}$ x $(8)_{10}$ = 160_{10}
dual : $(10100)_2$ x $(1000)_2$ = $(10100000)_2$ [10100 | 000]

dezimal : $(20)_{10}$: $(4)_{10}$ = 5_{10}
dual : $(10100)_2$: $(100)_2$ = $(101)_2$ [101 | 00]

Der Vollständigkeit halber wird im Begleitmaterial zu diesem Buch trotzdem die duale Multiplikation und Division entsprechend den Regeln vorgestellt, die wir im Zehnersystem anwenden, wenn wir per Hand multiplizieren.

3.5.4 Konvertieren durch sukzessive Multiplikation und Addition

Für die Konvertierung aus einem beliebigen Positionssystem in ein anderes beliebiges Positionssystem kann auch der folgende Algorithmus verwendet werden, wobei die Berechnung im Zielsystem mit der Basis B des Ausgangssystems durchgeführt wird:

$$
\begin{array}{lcll}
b_n \cdot B & = & a_1 & \\
a_1 + b_{n-1} & = & a_2 & \\
a_2 \cdot B & = & a_3 & \\
a_3 + b_{n-2} & = & a_4 & \\
\ldots & \ldots & \ldots & \\
a_{2n-1} + b_0 & = & \mathrm{x} & \text{im Zielsystem (mit der Basis B)}
\end{array}
$$

Konvertieren der Zahl $(2314)_{10}$ in das Dualsystem
Die Basis $B = (10)_{10}$ hat im Dualsystem die Darstellung $(1010)_2$ und die Dezimalziffern 2, 3, 1, 4 haben im Dualsystem folgende Darstellungen: $(10)_2$, $(11)_2$, $(1)_2$, $(100)_2$.

$10 \cdot 1010 = 10100$
$+ 11 = 10111$
$\cdot 1010 = 11100110$
$+ 1 = 11100111$
$\cdot 1010 = 100100000110$
$+ 100 =$ **100100001010**

► Übung

Konvertieren Sie die Zahl $(11100)_2$ in das Dezimalsystem!
Konvertieren Sie die Zahl $(555)_6$ in das Dezimalsystem!
Konvertieren Sie die Zahl $(0110110)_2$ in das Sechsersystem!

Copyright