HANSER

Leseprobe

Taschenbuch der Informatik

Herausgegeben von Uwe Schneider

ISBN: 978-3-446-42638-2

Weitere Informationen oder Bestellungen unter

http://www.hanser.de/978-3-446-42638-2

sowie im Buchhandel.

2 Informationsdarstellung

Rainer G. Spallek, Thomas B. Preußer

2.1 Information

Information ist der bestimmende Begriff für den Gegenstand der Informatik (→ 1.1), /2.11/, /2.16/, /2.10/.

Information (information): neues, verhaltensbestimmendes Wissen über ein Ereignis, einen Tatbestand oder einen Sachverhalt in der Wirklichkeit. Information ist Beseitigung von Ungewissheit.

❑ *Beispiel*: Information in Zahlen, in einem Text, in einem Bild

Information kann unter drei wesentlichen Blickwinkeln betrachtet werden:
- **Syntax** Strukturierung und formale Darstellung
- **Semantik** Aussageinhalt
- **Pragmatik** Bedeutung und Konsequenz.

Die technische Verarbeitung von Information erfolgt auf der syntaktischen Ebene durch die Verarbeitung der darstellenden Daten (→ 2.2). Mit der Codierung (→ 2.2.3), Bewertung und Messung von Information befassen sich die Informations- und die Codierungstheorie /2.16/, /2.10/. Auch das Wissen über die semantische und insbesondere auch die pragmatische Interpretation von Information durch den Empfänger kann technisch beispielsweise zur Datenkompression (→ 2.6) ausgenutzt werden. Insbesondere sind verlustbehaftete Kompressionen von Audio- und Videodaten weit verbreitet, die für die menschliche Wahrnehmung keine oder nur eine geringe pragmatische Konsequenz in Form von Qualitätseinbußen haben.

2.2 Daten, Zeichen, Maschinenwort

2.2.1 Konzept

Daten (data) sind strukturiert codierte Informationen, die als solche rechentechnisch formal verarbeitet werden können.

► *Anmerkung*: Die Hardware eines Rechners ist auf die Verarbeitung von wenigen genau definierten Datenformaten beschränkt. Die Universalität des Rechners wird durch die Verkettung der auf ihnen definierten Grundoperationen und durch die Strukturierung der Basisdatenformate zu größeren Komplexen wie Felder und Records erreicht.

Bezüglich ihrer Aufgaben werden drei Klassen von Daten unterschieden:
- **Verarbeitungsdaten** stellen die zu verarbeitenden Informationen und Verarbeitungsergebnisse dar (Operanden, z. B. Festkommazahlen → 2.3.4),
- **Programmdaten** bestimmen Art und Reihenfolge der Verarbeitungsschritte (Operationen, z. B. Befehle → 2.5),
- **Steuerdaten** steuern und registrieren Arbeitsmodi des Computers und unterstützen die geordnete Programmabarbeitung sowie die Übergänge zwischen verschiedenen Programmen (z. B. *Programmstatuswort*).

Daten bestehen im allgemeinen Fall aus *Zeichenfolgen*, die aus einem *Zeichenvorrat* nach bestimmten Regeln erzeugt werden. Diesbezüglich unterscheidet man:
- **numerische Daten** (Zahlen), die aus numerischen *Zeichen* (Ziffern) und eventuell einem Vorzeichen gebildet werden,
- **alphabetische Daten** (z. B. Personennamen), die nur aus alphabetischen *Zeichen* (Buchstaben) gebildet werden,
- **alphanumerische Daten**, die aus beliebigen Zeichen (Ziffern, Buchstaben und Sonderzeichen wie +, ∗, /, [, §, . . .) gebildet werden.

2.2.2 Darstellung und Quantisierung

Daten werden im Computer mit physikalischen Größen (Signalen) dargestellt, bei denen typischerweise genau zwei Zustände unterschieden werden:
- hohe/niedrige Spannung auf einer Leitung (high/low),
- elektrische Ladung/keine elektrische Ladung in einer Speicherzelle.

> Abstrahierend von den jeweiligen physikalischen Signalen, werden den beiden unterschiedenen Zuständen die **Binärzeichen** 0 und 1 zugeordnet. Mit diesen erfolgt eine logische Informationscodierung unabhängig von der momentan genutzten physikalischen Repräsentation.

Bei der externen Übertragung und Speicherung von Daten erfolgt häufig eine feinere Quantisierung der physikalischen Größe. Die Zahl der unterschiedenen Zustände entspricht oft einer Zweierpotenz, um eine einfache Umcodierung in die rechnerinterne binäre Darstellung zu ermöglichen.

Ein **Maschinenwort** (auch: **Binärwort**, word) ist eine geordnete Folge fester Länge von Binärzeichen. Es wird im Computer als Einheit geführt und verarbeitet. Die Anzahl der in einem Maschinenwort zusammengefassten Binärzeichen ist die **Wortlänge**. Ihre Zähleinheit ist das **Bit** (Akronym aus *Binary Digit*). Bei der Verwendung von Bit als Maßeinheit wird die Kleinschreibung bevorzugt (→ Bild 2.1).

Die Wortlänge ist eine charakteristische Eigenschaft einer Rechnerarchitektur (→ 3) und beschreibt ihren Parallelitätsgrad auf niedrigster Ebene, der **Bitebenenparallelität** (→ 3.5.1).

Für einige kurze Längen von Bitfolgen sind die in Tabelle 2.1 angegebenen Bezeichnungen üblich. Besonders verbreitet ist die Zähleinheit **Byte** (→ Bild 2.2):

$$\boxed{1\ \text{Byte} = 8\ \text{bit}} \tag{2.1}$$

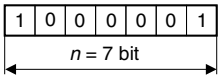

Bild 2.1 Codierung des Buchstabens A in einem Maschinenwort der Länge 7 bit (Code ASCII → Tabelle am Buchende, links)

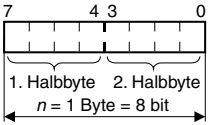

Bild 2.2 Struktur eines Bytes

Tabelle 2.1 Benannte Längen kurzer Bitfolgen

Länge in Bit	Bezeichnung	Anzahl unterscheidbarer Wörter (→ 2.2.3)	Anwendung zur Codierung von …
3	Triade	$2^3 = 8$	Oktalziffern
4	Tetrade, Halbbyte, Nibble	$2^4 = 16$	(Dezimal-,) Hexadezimalziffern
8	Byte, Oktett	$2^8 = 256$	Alphanumerischen Zeichen

Im Kontext konkreter Rechnerarchitekturen wird häufig die Breite des Maschinenwortes als weitere Zähleinheit verwendet, von der noch markante Vielfache abgeleitet werden. Wie in Tabelle 2.2 gezeigt, ergeben sich je nach Basiswortlänge verschiedene konkrete Bitbreiten für die gleichen Bezeichnungen. Die zugrunde gelegte Basiswortlänge kann hierbei auch aus der historischen Entwicklung der Architektur herrühren. So wird bei der x86-Architekturfamilie von Intel ein Wort immer noch mit einer Länge von 16 Bit angenommen, obwohl die Entwicklung schon längst bei 32-Bit- und 64-Bit-Architekturen angelangt ist.

Zur Bemessung großer Datenmengen und insbesondere zur Angabe von Speicherkapazitäten kommen die Binärpräfixe aus Tabelle 2.3 zur Anwendung. Diese sind von den Dezimalpräfixen des SI-Einheitensystems abgeleitet, wachsen jedoch jeweils um den Faktor $2^{10} = 1024$ und nicht wie im Dezimalsystem mit $10^3 = 1000$. Die prozentualen Unterschiede zwischen den korres-

Tabelle 2.2 Verbreitete benannte Vielfache eines Maschinenworts

Bezeichnung		Bitbreite	
		16-Bit- bzw.	32-Bit-Maschinenwort
Halbwort	Halfword	(1 Byte)	16
Wort	Word	16	32
Doppelwort	Doubleword	32	64
Vierfachwort	Quadword	64	128
Achtfachwort	Octaword	128	256

pondierenden Binär- und Dezimalpräfixen wachsen mit der Größenordnung der Präfixe. Zur Unterscheidung dient das „i“ im Vorsatz, das in den Namen als *bi* für *binär* integriert wird.

► *Beachte*: Diese Unterscheidung wurde erst 1998 in der Norm IEC 60 027-2 standardisiert. Sie fehlt deshalb in älterer Literatur und wird auch heute in eindeutigen Kontexten oft noch weggelassen.

Tabelle 2.3 Binärpräfixe

Vorsatz	Name	Quantität
Ki	Kibi	$2^{10} =$ 1 024
Mi	Mebi	$2^{20} = 1024\,\text{Ki} =$ 1 048 576
Gi	Gibi	$2^{30} = 1024\,\text{Mi} =$ 1 073 741 824
Ti	Tebi	$2^{40} = 1024\,\text{Gi} =$ 1 099 511 627 776
Pi	Pebi	$2^{50} = 1024\,\text{Ti} =$ 1 125 899 906 824 624

❑ *Beispiel*: 2 KiByte = 2048 Byte = 16 384 bit

2.2.3 Codierung

Begriffe

Ein **Code** bestimmt eine pragmatische Darstellungsvorschrift, d. h. die **Codierung**, von Informationseinheiten mithilfe der Zeichen eines **Alphabets**. In Verallgemeinerung des Begriffes wird die Menge der verwendeten **Codewörter** zu einem Alphabet höherer Ordnung.

Zur Spezifikation endlicher Codes kommt häufig eine Codetafel zur Anwendung:

Eine **Codetafel** ist die tabellarische Zuordnung der Codierungen von semantisch äquivalenten Informationseinheiten unter Verwendung verschiedener Codes. Eine solche Zuordnung erfolgt meist eineindeutig.

► *Beachte*: Eine Codetafel ist nur dann von Nutzen, wenn die Interpretation eines der dargestellten Codes (intuitiv) bekannt ist.

❑ *Beispiel*: Die Codetafel zum Morsecode bildet die bekannten Buchstaben- und Ziffernsymbole auf Folgen aus den Morsezeichen Strich und Punkt ab.

Im technischen Umfeld sind Codes mit fester Codewortlänge weit verbreitet. Über einem Alphabet Σ mit $|\Sigma|$ Symbolen können bei einer Wortlänge von n Zeichen

$$k = |\Sigma|^n \tag{2.2}$$

Codewörter unterschieden werden.

Ein Code heißt genau dann **dicht**, wenn dessen feste Wortlänge die minimale ist, mit der die zu unterscheidenden Informationseinheiten noch codiert werden können.

Konkret bedeutet das, dass die feste Wortlänge n eines Codes über dem Alphabet Σ bei q zu unterscheidenden Informationseinheiten die folgende Bedingung erfüllt:

$$|\Sigma|^{n-1} < q \leqq |\Sigma|^n \tag{2.3}$$

Die Zahl der Stellen, an denen sich zwei Codewörter eines Codes mit fester Wortlänge unterscheiden, ist deren **HAMMING-Distanz**. Die minimale HAMMING-Distanz zweier Codewörter eines Codes ist die HAMMING-Distanz des Codes.

Ein Code, der nicht auf alle möglichen Codierungen zurückgreift und eine HAMMING-Distanz größer als 1 aufweist, ermöglicht es, Verfälschungen von Codewörtern (z. B. bei der Datenübertragung) zu erkennen oder gar zu korrigieren. Codes mit diesen Eigenschaften heißen **fehlererkennend** (Error Detecting Code – EDC) bzw. **fehlerkorrigierend** (Error Correcting Code – ECC) /2.16/, /2.21/.

Ein Code erfüllt die **Präfixeigenschaft**, wenn kein Codewort Präfix eines anderen Codewortes ist. Ein solcher Code heißt **Präfixcode**.

In einem Präfixcode sind Nachrichten aus mehreren Codewörtern immer eindeutig in die Einzelwörter zerlegbar. Relevant ist diese Eigenschaft insbesondere für Codes mit variabler Codewortlänge, da dort die Wortgrenzen nicht schon durch die feste Wortlänge gegeben sind. Ein Code mit fester Wortlänge erfüllt die Präfixeigenschaft trivialerweise, da ein Codewort nur ein Präfix von sich selbst sein kann.

Binäre Blockcodes

Codes über dem Alphabet $\Sigma = \{0, 1\}$ mit fester Codewortlänge heißen **binäre Blockcodes**.

Aufgrund der internen Datendarstellung im Rechner genießen binäre Blockcodes eine hohe technische Bedeutung. Ein solcher Code ist *dicht*, falls:

$$2^{n-1} < q \leqq 2^n \tag{2.4}$$

Eine Variante des dichten Binärcodes ist der **GRAY-Code**, dessen besondere Eigenschaften sich erst aus der Ordnung der Codewörter ergeben. Wie in Tabelle 2.4 gezeigt, unterscheiden sich beim GRAY-Code benachbarte Codewörter lediglich an einer Bitposition. Diese Eigenschaft wird beispielsweise zur fehlerfreien Übernahme von Zählerständen über Taktbereichsgrenzen in FIFO-Puffern genutzt.

*Tabelle 2.4 Beispiel für einen 3-Bit-*GRAY*- und einen 1-aus-n-Code*

Dezimal	Binär	GRAY	1-aus-*n*
0	000	000	0000 0001
1	001	001	0000 0010
2	010	011	0000 0100
3	011	010	0000 1000
4	100	110	0001 0000
5	101	111	0010 0000
6	110	101	0100 0000
7	111	100	1000 0000

Einen praktisch bedeutsamen nicht dichten Binärcode stellt der **1-aus-*n*-Code** (**1-Hot-Code**) dar, dessen Codewörter genau eine 1 enthalten (→ Tabelle 2.4). Dieser findet Anwendung in der Arbitrierung von Zugriffen auf nur exklusiv nutzbare Ressourcen (z. B. Busse), in der Adressdecodierung zur Aktivierung der einen ausgewählten Speicherzeile oder -spalte und auch in der Codierung von Zuständen von Steuerautomaten. Für letztere ist die besonders einfache Erkennung eines speziellen Automatenzustandes durch die Auswertung eines einzelnen Bits von Bedeutung. Ferner sind in einem 1-aus-*n*-Code alle Einzelbitfehler und viele andere durch einfache Zählung der Einsen leicht erkennbar.

Eine einfache Möglichkeit, binäre Codes gegen unerkannte Einfachfehler abzusichern, ist das Hinzufügen eines **Paritätsbits**. Dieses ergänzt die Anzahl der Einsen im Codewort, wie für den Code festgelegt, auf eine gerade oder ungerade Anzahl. Alle Verfälschungen eines einzelnen Bits führen damit zu Worten mit falscher Parität, die keine gültigen Codewörter darstellen. Es

können so sogar alle ungeradzahligen Bitfehler erkannt werden, geradzahlige bleiben dagegen unerkannt, da sie zu einem anderen, aber gültigen Codewort führen.

2.3 Zahlendarstellung

2.3.1 Stellenwertsysteme

Aufgrund der internen binären Codierung von Daten im Rechner bieten sich zur Darstellung numerischer Daten neben dem Dezimalsystem noch andere Systeme an. Besondere Bedeutung genießen hierbei die Stellenwertsysteme (positional number systems) mit den Basen 2, 8 und 16. Allen Stellenwertsystemen ist gemein, dass sie ganze Zahlen (ohne Nachkommastellen) und reelle Zahlen (mit Nachkommastellen) durch Wörter über einer endlichen Ziffernmenge repräsentieren. Der Beitrag einer Ziffer zum Zahlenwert wird dabei maßgeblich durch ihre Position innerhalb der Ziffernfolge relativ zum Komma bestimmt.

Ein Stellenwertsystem ist charakterisiert durch:
- seine Basis B (auch Radix genannt), und
- seine Ziffernmenge D.

Der Wert einer Zahl $Z = z_{n-1} \ldots z_1 z_0, z_{-1} \ldots z_{-m}$ mit $z_i \in D$ berechnet sich zu:

$$\begin{aligned} Z &= B^{n-1} z_{n-1} + \ldots + B^1 z_1 + B^0 z_0 + B^{-1} z_{-1} + \ldots + B^{-m} z_{-m} \\ &= \sum_{i=-m}^{n-1} B^i \cdot z_i \qquad (2.5) \\ &= (\ldots (z_{n-1} B + z_{n-2}) B + \ldots + z_1) B + z_0 \\ &\quad + \frac{1}{B} \left(z_{-1} + \frac{1}{B} \left(z_{-2} + \ldots + \frac{1}{B} \left(z_{-m+1} \frac{1}{B} z_{-m} \right) \ldots \right) \right) \qquad (2.6) \end{aligned}$$

❑ *Beispiel*: Im Dezimalsystem mit $B = 10$ und $D = \{0, 1, 2, \ldots, 9\}$ gilt:

$$\begin{aligned} 436{,}95 &= 4 \cdot 10^2 + 3 \cdot 10^1 + 6 \cdot 10^0 + 9 \cdot 10^{-1} + 5 \cdot 10^{-2} \\ &= (4 \cdot 10 + 3) \cdot 10 + 6 + \frac{1}{10} \cdot \left(9 + \frac{1}{10} \cdot 5 \right) \end{aligned}$$

In konventionellen Stellenwertsystemen gilt $D = [0, B) = \{0, 1, \ldots, B-1\}$. In diesen ist die Darstellung eines codierten Zahlenwertes eindeutig. Es ist ferner jede ganze Zahl darstellbar und jede reelle Zahl durch das Hinzufügen weiterer Nachkommastellen beliebig genau approximierbar.

Insbesondere für die interne Zahlendarstellung in arithmetischen Schaltungen kommen auch andere unkonventionelle Stellenwertsysteme zum Einsatz (→ Tabelle 2.5). Diese verwenden häufig auch negative Ziffern, die ihr Vorzeichen – zur Unterscheidung von der Subtraktion – über der Betragsziffer tragen.

Tabelle 2.5 Verwendung unkonventioneller Stellenwertsysteme

Basis	Ziffern	Verwendung
2	0, 1, 2	Carry-Save: Mehroperandenadditon, Multiplikation
2	$\overline{1}$, 1	Nicht-restaurierende Division
2	$\overline{1}$, 0, 1	SRT-Division (SWEENEY, ROBERTSON, TOCHER)
4	$\overline{2}$, $\overline{1}$, 0, 1, 2	BOOTH-Multiplikation, ROBERTSON-Division

► *Hinweis*: Im Weiteren werden hier nur konventionelle Stellenwertsysteme betrachtet. Für andere und deren Anwendungen sei auf die weiterführende Literatur verwiesen: /2.17/, /2.7/.

Ist die Basis einer Zahlendarstellung nicht aus dem Kontext erkennbar, wird sie durch Zusätze markiert. Universell anwendbar ist das Anfügen der tief gestellten Basis im Dezimalsystem. Für die Basen 2, 8 und 16 kommen oft auch die Präfixe 0b, 0 bzw. 0x zur Anwendung. Ziffernwerte jenseits der 9 werden fortlaufend durch die Buchstaben des lateinischen Alphabets dargestellt, wobei A den Wert 10 annimmt.

❑ *Beispiel*: $0x7A = 7A_{16} = 122_{10} = 172_8 = 111\,1010_2$

In Stellenwertsystemen kann prinzipiell mit den aus dem Dezimalsystem bekannten Verfahren zur Addition, Subtraktion, Multiplikation und Division gerechnet werden. In jedem Stellenwertsystem entspricht die Multiplikation mit einer Basispotenz B^k einer Rechtsverschiebung des Kommas um k Stellen, eine Division durch B^k einer Linksverschiebung um k Stellen.

Besonders kurz lassen sich die Rechenregeln für die vier Grundrechenarten im **Dualsystem** mit $B = 2$ und $D = \{0, 1\}$ beschreiben (→ Tabelle 2.6).

Tabelle 2.6 Rechenregeln ($\ddot{u}_{\mu+1}$ Übertrag in die nächsthöhere Stelle)

Addition	Subtraktion	Multiplikation	Division
$0+0=0$ $0+1=1$ $1+0=1$ $1+1=0 \rightarrow \ddot{u}_{\mu+1}=1$	$0-0=0$ $0-1=1 \rightarrow \ddot{u}_{\mu+1}=-1$ $1-0=1$ $1-1=0$	$0 \cdot 0 = 0$ $0 \cdot 1 = 0$ $1 \cdot 0 = 0$ $1 \cdot 1 = 1$	$0 \div 0$ verboten $0 \div 1 = 0$ $1 \div 0$ verboten $1 \div 1 = 1$
$\begin{array}{rr} & 101 \\ & +\ 011 \\ \ddot{u} \rightarrow & 111 \\ \hline & 1000 \end{array}$	$\begin{array}{rr} & 101 \\ & -\ 011 \\ \ddot{u} \rightarrow & -1 \\ \hline & 010 \end{array}$	$\begin{array}{r} 101 \cdot 011 \\ 101 \\ 101 \\ 000 \\ \hline 1111 \end{array}$	$10110 \div 11 = 111$ $\begin{array}{r} -11 \\ \hline 0101 \\ -\ 11 \\ \hline 0100 \\ -\ 11 \\ \hline 001 \end{array} \rightarrow$ Rest

Von praktischer Bedeutung sind ferner das **Oktalsystem** mit $B = 8$ und $D = \{0, 1, 2, 3, 4, 5, 6, 7\}$ und insbesondere das **Hexadezimalsystem** mit $B = 16$ und $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}\}$. Durch höhere Zweierpotenzen als Basis erlauben sie Binärdaten einfach und kompakter darzustellen. Tabelle 2.7 gibt die Operationstafeln für die Addition und die Multiplikation im Hexadezimalsystem an.

Tabelle 2.7 Addition und Multiplikation im Hexadezimalsystem

+ ·	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
	1														
2	2	4	5	6	7	8	9	A	B	C	D	E	F	10	11
		4													
3	3	6	6	7	8	9	A	B	C	D	E	F	10	11	12
			9												
4	4	8	C	8	9	A	B	C	D	E	F	10	11	12	13
				10											
5	5	A	F	14	A	B	C	D	E	F	10	11	12	13	14
					19										
6	6	C	12	18	1E	C	D	E	F	10	11	12	13	14	15
						24									
7	7	E	15	1C	23	2A	E	F	10	11	12	13	14	15	16
							31								
8	8	10	18	20	28	30	38	10	11	12	13	14	15	16	17
								40							
9	9	12	1B	24	2D	36	3F	48	12	13	14	15	16	17	18
									51						
A	A	14	1E	28	32	3C	46	50	5A	14	15	16	17	18	19
										64					
B	B	16	21	2C	37	42	4D	58	63	6E	16	17	18	19	1A
											79				
C	C	18	24	30	3C	48	54	60	6C	78	84	18	19	1A	1B
												90			
D	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	1A	1B	1C
													A9		
E	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	1C	1D
														C4	
F	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	1E
															E1

2.3.2 Konvertierungsverfahren

Die Konvertierung zwischen Stellenwertsystemen unterschiedlicher Basis erfolgt auf der Grundlage der Gleichungen (2.5) oder (2.6). Diese eröffnen zwei Herangehensweisen:

- die Berechnung des Zahlenwertes im Zielsystem oder
- das fortlaufende Abtrennen der ganzen Basisvielfachen im Herkunftssystem nach dem HORNER-Schema gemäß Gleichung (2.6).

Typischerweise wird das Verfahren so gewählt, dass im gewohnten Dezimalsystem gerechnet werden kann. Beide Verfahren sind in Bild 2.3 an Beispielen illustriert.

$X_2 = 1011{,}101_2$

$X_{10} = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$

a) $X_{10} = 11{,}625_{10}$

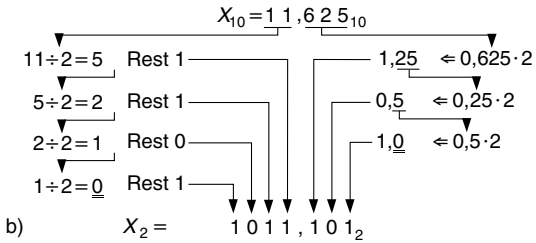

Bild 2.3 a) Schema für das Konvertieren von Dual- in Dezimalzahlen. Oktal- bzw. Hexadezimalzahlen werden nach dem gleichen Schema in Dezimalzahlen konvertiert, jedoch ist die Basis 2 durch 8 bzw. 16 zu ersetzen.
b) Schema für das Konvertieren von Dezimal- in Dualzahlen. Dezimalzahlen werden nach dem gleichen Schema in Oktal- bzw. Hexadezimalzahlen konvertiert, jedoch ist an Stelle von 2 mit 8 bzw. 16 zu multiplizieren oder zu dividieren.

Besonders einfach ist die Konvertierung zwischen Stellenwertsystemen mit kohärenten Basen mit $B_1 = B_2^k$. Bei diesen gibt es eine eineindeutige Abbildung zwischen einer Ziffer des Systems zur Basis B_1 und einem Block von k aufeinander folgenden Ziffern im System zur Basis B_2. So bildet eine Binärtriade auf genau eine Oktalziffer und eine Binärtetrade auf genau eine Hexadezimalziffer ab. Die Umwandlung erfolgt jeweils durch die am Komma ausgerichtete Bildung von Ziffernblöcken.

❑ *Beispiel*: Oktalsystem $X_8 = 1\ 7\ \ 5\ \ 2\ ,\ 3\ \ 4\ \ 7$
Dualsystem $X_2 = 111\,110\,101\,0,\,011\,100\,111$
Hexadezimalsystem $X_{16} = 3\ \ E\ \ A\ ,\ 7\ \ 3\ \ 8$