

Prof. Dr.-Ing. Dr. h. c. J. Becker

becker@kit.edu

Karlsruher Institut für Technologie (KIT)

Institut für Technik der Informationsverarbeitung (ITIV)

Digitaltechnik

Zahlensysteme

- **Zahlensysteme** dienen zur transparenten **Darstellung** von Zahlen durch **geeignete Ziffern** und deren **systematischen Anordnung**
-> elementar für die **Mechanisierung/Automatisierung** in der **Digital-** sowie **Rechentechnik**
- **Polyadische Zahlensysteme** geben den **Ziffern** ihren **Wert** in **Abhängigkeit** von ihrer **Stelle** innerhalb der systematischen stellenorientierten **Anordnung**
- Das **Dezimalsystem** ist der uns **geläufigste Vertreter** der polyadischen Zahlensysteme
- Die **Stellenwerte** entsprechen den **Potenzen** der **Basis** des jeweiligen **polyadischen Zahlensystems**

Beispiele:

$$1689_{10} = 1 * 10^3 + 6 * 10^2 + 8 * 10^1 + 9 * 10^0 = 1689$$

$$1022_3 = 1 * 3^3 + 0 * 3^2 + 2 * 3^1 + 2 * 3^0 = 35$$

$$1011_2 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 = 11$$

$$41_{12} = 4 * 12^1 + 1 * 12^0 = 49$$

Allgemein: Aufbau einer Zahl **N**

$$N = d_n * R^n + \dots + d_1 * R^1 + d_0 * R^0$$

N: Zahl im Zahlensystem

R: Basis, (Grundzahl, Radix) $R \geq 2$

Rⁱ: Wertigkeit der i-ten Stelle

d_i: Ziffer der Stelle i

Z: Menge der Ziffern: $d_i \in Z = \{0, 1, 2, \dots, R-1\}$

Dualsystem (Binärsystem):

$$R_B = 2 \quad (\text{Basis})$$

$$Z_B = \{0,1\}$$

Beispiel:

$$1101_2 = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 13$$

■ Lässt sich in **Hardware effizient darstellen** und **verarbeiten**

■ bei relativ **kleinen Zahlen** im Dualsystem

→ sehr **viele Ziffern** zur **Darstellung** notwendig

→ man fasst oftmals **3** oder **4 Ziffern zusammen**

→ resultiert in **Oktal-** oder **Hexadezimalsystem**

Zahlendarstellung im Dualsystem:

■ Dualzahlen:

- Repräsentation einer Zahl mit **Binärstellen** in **polyadischer Darstellung** und Interpretation
- wenn **Ziffernvorrat** in **einer Stelle erschöpft**, erfolgt ein **Übertrag** in nächsthöhere Stelle

■ Dualsystem:

- grundlegende Zahlensystem für die numerische Verarbeitung in der Digitaltechnik

Dezimalzahlen				Dualzahlen				
10^3	10^2	10^1	10^0	2^4	2^3	2^2	2^1	2^0
			0					0
			1					1
			2				1	0
			3				1	1
			4			1	0	0
			5			1	0	1
			6			1	1	0
			7			1	1	1
			8		1	0	0	0
			9		1	0	0	1
		1	0		1	0	1	0
		:	:					:
		:	:					:
	1	9		1	0	0	1	1
	2	0		1	0	1	0	0

Polyadische Zahlensysteme: Beispiele (2)

Oktalsystem:

$$R_O = 8$$

$$Z_O = \{0, 1, 2, 3, 4, 5, 6, 7\}$$

- 1 Ziffer im Oktalsystem entspricht genau 3 Ziffern im Dualsystem

Beispiele: Oktalzahlen

Dezimal:	0	1	2	3	4	5	6	7	8	9
Oktal:	0 _O	1 _O	2 _O	3 _O	4 _O	5 _O	6 _O	7 _O	10 _O	11 _O

Dezimal:	10	11	12	13	14	15	16	17	18	19
Oktal:	12 _O	13 _O	14 _O	15 _O	16 _O	17 _O	20 _O	21 _O	22 _O	23 _O

Hexadezimalsystem:

$$R_H = 16$$

$$Z_H = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$$

- 1 Ziffer im Hexadezimalsystem entspricht genau 4 Ziffern im Dualsystem
- 2 Ziffern im Hexadezimalsystem zusammengefasst -> 1 Byte (8 Bit)

■ Beispiele: Hexadezimalzahlen

Dezimal:	0	1	...	9	10	11	12	13	14	15	16
Hexadezimal:	0 _H	1 _H	...	9 _H	A _H	B _H	C _H	D _H	E _H	F _H	10 _H
Dezimal:	17	18	...	158	159	160	161	162	163	164	...
Hexadezimal:	11 _H	12 _H	...	9E _H	9F _H	A0 _H	A1 _H	A2 _H	A3 _H	A4 _H	...

Umrechnung: Dualsystem -> Oktal- bzw. Hexadezimalsystem

$$110_B = 6_O$$

$$011_B = 3_O$$

$$110\ 011_B = 63_O$$

$$1011\ 0010_B = B2_H$$

$$255_D = FF_H$$

■ **3 Ziffern** im **Dualsystem** werden zu genau **1 Ziffer** im **Oktalsystem** zusammengefasst

■ **4 Ziffern** im **Dualsystem** werden zu genau **1 Ziffer** im **Hexadezimalsystem**

zusammengefasst

-> sehr gut für **byte-orientierte**

Darstellungen

-> 2 Hexadezimalzahlen = 1 byte

Also:

→ kompaktere Darstellung grosser Dualzahlen im Oktal- bzw. Hexadezimalsystem

→ effizienter Darstellung, Codewandlung sowie Verarbeitung

Wandlung von Zahlensystemen:

- Wenn die **Basis** eines **Zahlensystems** die **Potenz der Basis** eines **anderen Zahlensystems** ist, so sind **Zahlen** zwischen diesen Systemen **sehr leicht umzuwandeln**
- **Zusammenfassung** von **mehreren Stellen** zu einer **einzelnen Ziffer**

Beispiel 1:

$$\begin{array}{ccc} (1011 & 0101 & 0001)_B \\ \Downarrow & \Downarrow & \Downarrow \\ (B & 5 & 1)_H \end{array}$$

■ Wandlung:

-> Dual- in Hexadezimalsystem

Beispiel 2:

$$\begin{array}{ccc} (5 & 2 & 1)_O \\ \Downarrow & \Downarrow & \Downarrow \\ (101 & 010 & 001)_B \end{array}$$

■ Wandlung:

-> Oktal- in Dualsystem

Wandlung von Zahlensystemen:

- Falls **keine** besondere **Beziehung zwischen** den **Basen** besteht, kann man die **Umrechnung über** das **Dezimalsystem** durchführen

Wandlung zunächst ins Dezimalsystem:

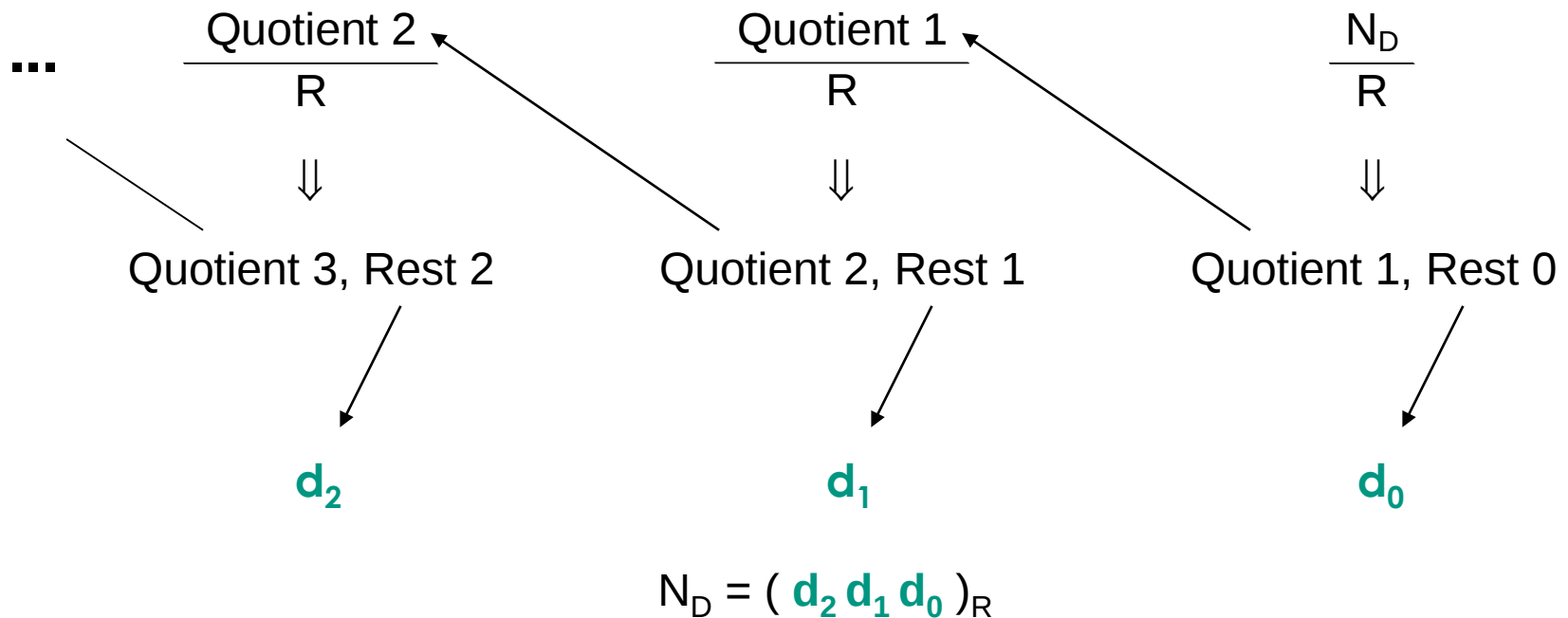
- Bekannte Formel:** $N_D = d_n * R_D^n + \dots + d_1 * R_D^1 + d_0 * R_D^0$

Beispiel: $(FA7E)_H = ?_D$

$$\begin{array}{ccccccc} (& \mathbf{F} & & \mathbf{A} & & \mathbf{7} & & \mathbf{E} &)_H \\ & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & \\ (& \mathbf{F} * 10^3 & + & \mathbf{A} * 10^2 & + & \mathbf{7} * 10^1 & + & \mathbf{E} * 10^0 &)_H \\ & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & \\ (& \mathbf{15} * 16^3 & + & \mathbf{10} * 16^2 & + & \mathbf{7} * 16^1 & + & \mathbf{14} * 16^0 &)_D \\ & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & \\ (& 61440 & + & 2560 & + & 112 & + & 14 &)_D \\ & & & & & & & & \\ & = & \mathbf{64126}_D & & & & & & \end{array}$$

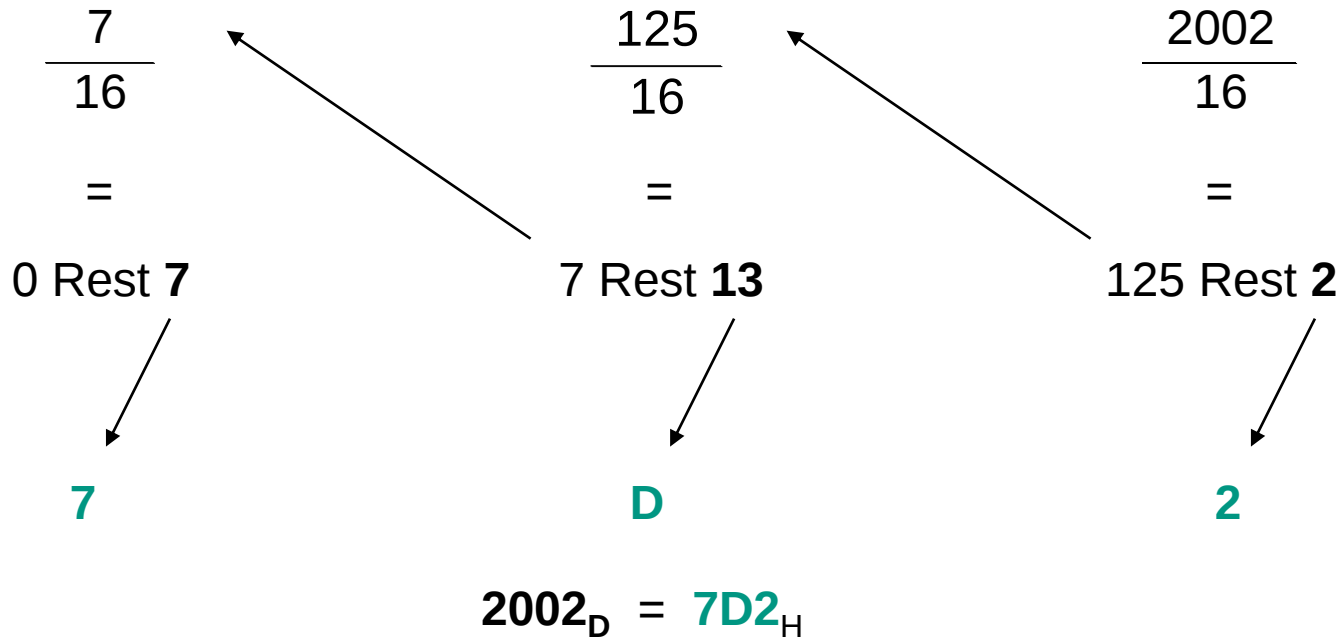
Wandlung von Zahlensystemen:

- Mit Hilfe einer **Ganzzahldivision mit Rest** kann **Umwandlung vom Dezimalsystem in System** mit der **Basis R** wie folgt durchgeführt werden:
 - > N_D wird durch R dividiert (Ergebnis Stelle d_0),
 - > entstehende Quotienten rekursiv nach gleichem Schema verarbeiten



Beispiel: Wandlung der Dezimalzahl 2002 in Hexadezimalzahl

- Umrechnung von 2002_D in das Hexadezimalsystem ($R=16$):



- **Wichtig:** es ist wesentlich **einfacher** Zahlensysteme umzuwandeln, wenn man **Ziffern jeweils separat** und **einzel**n behandeln kann
- Bei der oft gebrauchten **Umwandlung** vom **Dezimalsystem** in das **Dualsystem** ist dies **nicht möglich**
- Abhilfe schafft der **BCD-Code** (**B**inary **C**oded **D**ecimal)
- Dieser stellt **jede Dezimalziffer** durch eine **4-Bit Dualzahl** dar

$$49_D = (0100\ 1001)_{BCD}$$

- Die **10 Codewörter**, die den **Ziffern** zugeordnet sind, werden **Tetrad**e genannt.
Die **unbenutzten Codewörter** werden **Pseudotetrad**e genannt

Pseudotetrad: 1010, 1011, 1100, 1101, 1110, 1111

- **Automatisierte Rechenoperationen** sind von **zentraler Bedeutung**

→ **Basisoperationen** sind: **+, -, *, /**

- **Technische Realisierung:** **Kosten** (Platzbedarf) und **Performanz** wichtig!

→ **Überlegung:** welche Operationen direkt als Schaltung realisieren ?

- **Rechnen** kann man grundsätzlich in **allen Zahlensystemen**

- Im **Dualsystem** benötigt man besonders **wenig Regeln**:

- **Regel 1:** $0 + 0 = 0$

- **Regel 2:** $0 + 1 = 1 + 0 = 1$

- **Regel 3:** $1 + 1 = 10$

- **Regel 4:** $1 + 1 + 1 = 11$

Addition:

- **Dualzahlen** können mit den **vorangegangenen Regeln** so **addiert** werden, wie man das schriftliche Addieren schon in der Grundschule lernt

$$\begin{array}{rcccccc} & 1 & 0 & 1 & 1 & 0 & 0 \\ + & & 1 & 1 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 & & & \\ \swarrow & \swarrow & \swarrow & \swarrow & & & \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{array} \quad \text{Übertrag}$$

benutzte elementare Regel: **3 3 4 3 2 1**

Subtraktion:

- Beim **Subtrahieren** von **Dualzahlen** bedient man sich einem Trick: der **Komplementbildung**
- Statt beispielsweise zwei 8-Bit Dualzahlen direkt zu subtrahieren **erweitert** man mit einer **9-stelligen Dualzahl** wie folgt:

$$a - b = (a + \underbrace{(1\ 0000\ 0000_B - b)}_c) - 1\ 0000\ 0000_B$$

$\underbrace{\hspace{10em}}_d$

- Das **Zweierkomplement** **c** lässt sich einfach wie folgt berechnen:

$$c = 1\ 0000\ 0000_B - b = (1111\ 1111_B - b) + 1_B$$

Subtraktion:

- Dadurch hat man **allgemeine Subtraktion** durch **folgendes ersetzt**:

$$a - b = (a + (1111\ 1111_B - b + 1)) - 1\ 0000\ 0000_B$$

- **Addition** und **Inkrementierung (+1)** und **zwei spezielle Subtraktionen**

- Einfache Subtraktionen (Sonderfälle):

- $(1111\ 1111_B - b)$ berechnet man durch Invertieren aller 8 Bits von b

- $(d - 1\ 0000\ 0000_B)$ berechnet man wie folgt:

- Wenn Bit 9 von d gesetzt ist:

- Ergebnis positiv -> Bit 9 von d abschneiden, fertig

- Wenn Bit 9 von d gleich null ist:

- Ergebnis negativ -> Betrag erhält man durch
Zweierkomplement: Betrag = $(1111\ 1111_B - d) + 1$

Beispiel: Subtraktion: $21 - 3 = ?$

■ $a = 21_D = 0001\ 0101_B$

■ $b = 3_D = 0000\ 0011_B$

■ $c = (1\ 0000\ 0000_B - b) = 1111\ 1111_B - b + 1 = 1111\ 1100_B + 1 = 1111\ 1101_B$

a:		0	0	0	1	0	1	0	1
+ c:	+	1	1	1	1	1	1	0	1
		1	1	1	1	1	1		1
		↘	↘	↘	↘	↘	↘		↘
d:	1	0	0	0	1	0	0	1	0

■ $a - b = (d - 1\ 0000\ 0000_B) = 0001\ 0010_B = 18_D$

Beispiel: Subtraktion: **3 - 21 = ?**

■ **a** = $3_D = 0000\ 0011_B$

■ **b** = $21_D = 0001\ 0101_B$

■ **c** = $(1\ 0000\ 0000_B - \mathbf{b}) = 1111\ 1111_B - \mathbf{b} + 1 = 1110\ 1010_B + 1 = 1110\ 1011_B$

a:		0	0	0	0	0	0	1	1
+ c:	+	1	1	1	0	1	0	1	1
d:		0	1	1	1	0	1	1	0

1 1

↙
↙

■ **a - b** = $(\mathbf{d} - 1\ 0000\ 0000_B) < 0$

■ $|\mathbf{a} - \mathbf{b}| = (\text{Zweierkomplement von } \mathbf{d}) = 1111\ 1111_B - \mathbf{d} + 1$
 $= 0001\ 0010_B = 18_D$

BCD-Code (Binary Coded Decimal)

Rechnen im BCD-Code

- **Zentrale Stellung** des **Dezimalsystems**:
 - **Konversion** stärker unter **Betonung** der **dezimalen Aspekte**
 - **dezimalstellenweise Wandlung** in **Dualform** (Binärform)
- Wie in den **unterschiedlichen Zahlensystemen** können auch **BCD-kodierte Zahlen verrechnet** werden
- Bei der **BCD-Addition**: **Pseudotetraden** können entstehen
- **Zur Korrektur**: zu **Pseudotetraden** und **bei jedem Übertrag** noch **$6_D = 0110_B$** addiert werden
- Bei der **Addition** von **BCD-kodierten Zahlen** kann der **Bereich** der **Pseudotetraden** nur **einmal „überschritten“** werden
 - es genügt in jedem Fall eine Korrektur

■ Addition von 839_D und 862_D

Endergebnis: 1 0111 0000 0001 = 1701_D

Erweiterungen BCD-Code (1): STIBITZ-Code

- es hat nicht an Versuchen gefehlt die **negativen Eigenschaften** des **BCD-Codes** zu **vermeiden**
- Für die **Subtraktion** mit Hilfe der **Komplementbildung** ist es nützlich, dass die **Codewörter** mit **Ziffern symmetrisch angeordnet** sind
- Der **STIBITZ-** oder **Exzess-3-Code** erreicht dies, indem **alle Tetraden** im **Vergleich** zum **BCD-Code um $3_D = 0011_B$ größer** sind:

Dezimal	0	1	2	3	4
STIBITZ	0011	0100	0101	0110	0111
STIBITZ	1000	1001	1010	1011	1100
Dezimal	5	6	7	8	9

Erweiterungen BCD-Code (1): Addition im STIBITZ-Code

- **Codewörter** für Ziffern 0, ..., 4 sind **spiegelbildlich komplementär** zu Ziffern 5, ..., 9
- **Bei Übertrag:** 0011_B muss **addiert** werden
- **Ohne Übertrag:** es muss 0011_B **abgezogen** werden, was durch **Addition** von 1101_B **ohne Übertrag** erreicht werden kann

$$\begin{array}{r}
 \begin{array}{ccc}
 0100 & 0110 & 1100 \\
 + 0101 & 0111 & 0101 \\
 \hline
 1 & 111\textcolor{red}{1} & 1
 \end{array}
 \end{array}
 \begin{array}{r}
 139_D \\
 + 242_D \\
 \hline
 \end{array}$$

$$\begin{array}{ccc}
 1001 & 1110 & 0001
 \end{array}$$

Übertrag

Korrektur:

$$\begin{array}{r}
 \begin{array}{ccc}
 \textcolor{red}{X} +1101 & \textcolor{red}{X} +1101 & \textcolor{red}{+0011} \\
 \textcolor{red}{X} 1 & \textcolor{red}{X} 1 & 11 \\
 \hline
 0110 & 1011 & 0100
 \end{array}
 \end{array}
 = 381_D$$

Erweiterungen BCD-Code (1): AIKEN-Code

- Auch beim **Aiken-Code** sind die Codes der **Ziffern symmetrisch** angeordnet, um eine **einfache Komplementbildung** zu ermöglichen

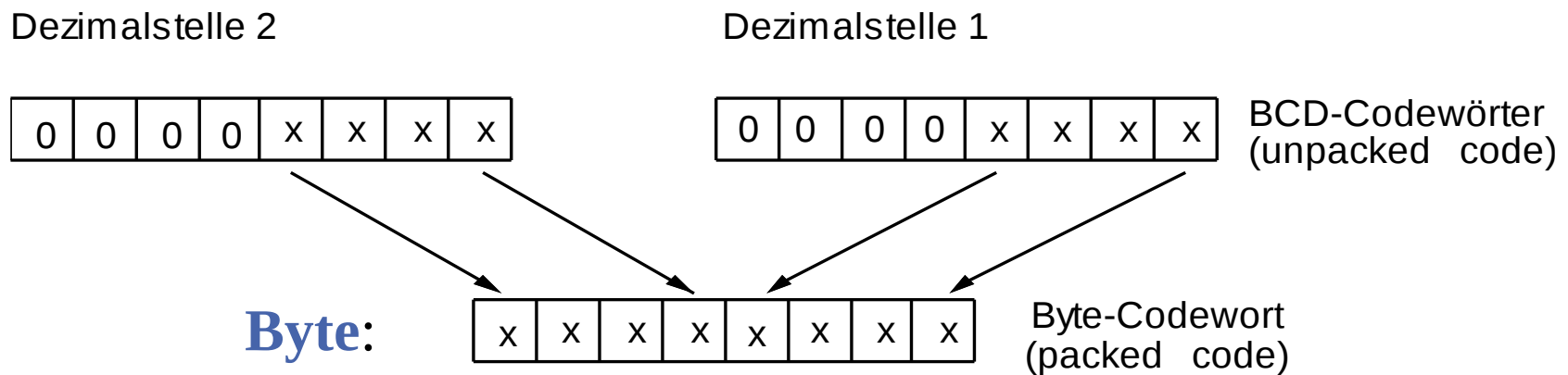
Dezimal	0	1	2	3	4
AIKEN	0000	0001	0010	0011	0100
AIKEN	1111	1110	1101	1100	1011
Dezimal	9	8	7	6	5

- Der **Aiken-Code** wird auch **2-4-2-1-Code** genannt, da dies den **Stellenwerten** der einzelnen **Binärstellen** entspricht

- $1101_{\text{AIKEN}} = 1 * 2 + 1 * 4 + 0 * 2 + 1 * 1 = 7_D$

Kompakte Zahlendarstellung:

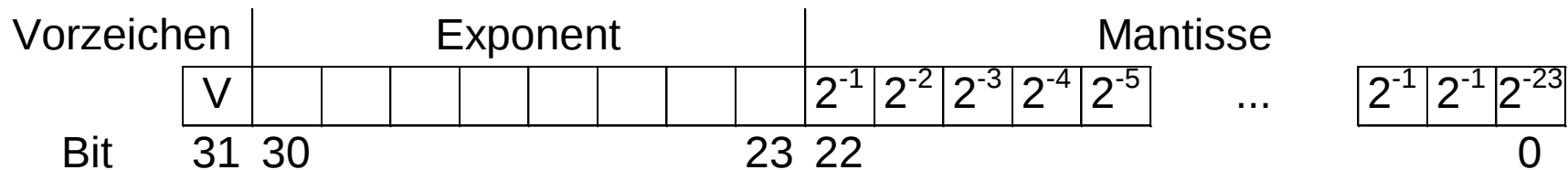
- **vorgestellte Codes** für Dezimalziffern verwenden 4 Bit pro Ziffer
→ **Zusammenfassung** von **2 binär** dargestellten **Dezimalstellen** zu einem **Byte**
- **Kompaktere Darstellung** von BCD-Zahlen und Codes



- Die bisher vorgestellten Codes haben das **Problem**, dass durch die Anzahl der Codewörter auch der **Zahlenraum eingeschränkt** ist
- **Problemursache:**
den **Ziffern** sind **feste Stellenwerte** zugeordnet, womit allenfalls sogenannte **Festkommazahlen** realisierbar sind
- Die Lösung ist, wie bei wissenschaftlichen Zahlennotationen üblich, die Angabe eines **zusätzlichen Exponenten**
- Die Darstellung von Zahlen durch **Mantisse** und **Exponent** wird ***Gleitkommazahl*** genannt
- **Beispiel:**
 $1,2345 * 10^{67}$

- Da **Gleitkommazahlen** auf sehr unterschiedliche Weise kodiert werden können, hat der **IEEE Richtlinien** festgelegt, wie Gleitkommazahlen dargestellt werden sollen

Gleitkomma-Zahlendarstellung gemäß IEEE-Standard



Exponent E	Mantisse M	Wert
255	$\neq 0$	ungültig (NaN)
255	0	$- 1^V \cdot \infty (\pm \text{unendlich})$
$0 < E < 255$	M	$- 1^V \cdot 2^{E-127} \cdot (1, M)$
0	$\neq 0$	$- 1^V \cdot 2^{-126} \cdot (0, M)$
0	0	$- 1^V \cdot 0$

Gleitkomma-Zahlendarstellung gemäß IEEE-Standard:

Die Zahl π mit 7 Dezimalstellen hinter dem Komma im **IEEE-Format** dargestellt:

$$\begin{aligned}\pi &\approx 3,1415926_D \approx 11,0010010000111110110\ 10_B \\ &= (-1^0 \times 2^{128-127})_D \times 1,1001001000011111011\ 010_B \\ &\quad \quad \quad / \quad \quad | \quad \quad \quad / \end{aligned}$$

Ergebnis: $\pi \approx 0\ 10000000\ 1001001000011111011010$