Kap. 4: Repräsentierung von Information in Rechensystemen

Einführung und Überblick (2)

Def

- ***Daten*** **sind Informationen, die nach eindeutigen Regeln in Rechensystemen gespeichert und verarbeitet werden ("Datenverarbeitung").**

  Auf der untersten Abstraktionsebene sind diese durch die Hardware eines Rechensystems bestimmt.

  Auf höheren Abstraktionsebenen sind Daten

  - **nach algorithmischen Gesichtspunkten strukturierte Informationen (z.B. Personaldaten).**
  - **i.d.R. von einem Programmierer entworfen (zusammengesetzte Datentypen)**
  - **durch Compiler auf die elementaren Informationsdarstellungen der Hardware abgebildet.**
- ***Nachrichten*** **sind Daten, die zur Kommunikation (Übertragung) von Information verwendet werden.**

Einführung und Überblick (3)

- In diesem Kapitel: Besprechung der in heutigen Rechnern üblichen Repräsentierungen für die elementaren Datentypen zusammen mit ihren jeweiligen Grundlagen.
- Als elementar werden in heutigen Rechnern i.d.R. angesehen:
  - Bitfolgen
  - Zahlen in verschiedenen Darstellungen
  - Zeichenketten
- In heutigen Rechnern wird jegliche Information in binärer Form codiert dargestellt und verarbeitet

  $\Rightarrow$ die elementaren Datentypen werden abgebildet auf Binärwörter (vgl. Kap. 2.4).
- Im folgenden Kapitel 4:
  Besprechung der allgemeinen Grundlagen der Codierung (Vorgehensweise: vom Konkreten zum Abstrakten).

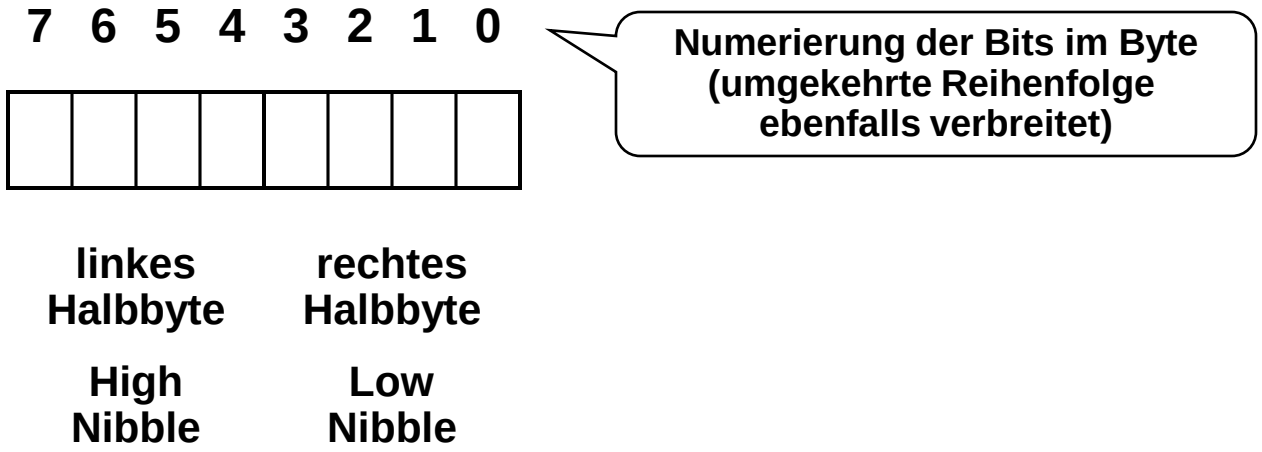
4.2 Bitfolgen

- **Wiederholung (vgl. Kap. 2.4): Für $\sum = \{0,1\}$ heißen die Elemente der Menge $\sum^n$ n-Bit-Wörter oder Binärwörter der Länge n.**
- **Informationen werden i.d.R. nicht auf der Bit-Ebene betrachtet, sondern zu größeren Einheiten zusammengefaßt, die bei variabler Länge als *Bitfolgen*, bei fester Länge auch als *Bitvektoren* bezeichnet werden.**
- **Bezeichnungen für Bitfolgen bestimmter Länge:**

Länge n in Bit	*Bezeichnung*	*Anzahl versch. Wörter*	*Anwendung*
1	Bit (b)	2	
3	Triade	$2^3=8$	Oktalziffern (vgl. 3.3)
4	Tetrade, Halb-Byte, Nibble	$2^4=16$	Hexadezimal- und Dezimalziffern (vgl. 3.3)
8	Byte	$2^8=256$	Zeichen (vgl. 3.4)

Byte

- **Die kleinste adressierbare Einheit im Arbeitsspeicher heutiger Rechner ist ein Byte:**



- **Die Bytes im Speicher sind fortlaufend numeriert.**

Maschinenwörter

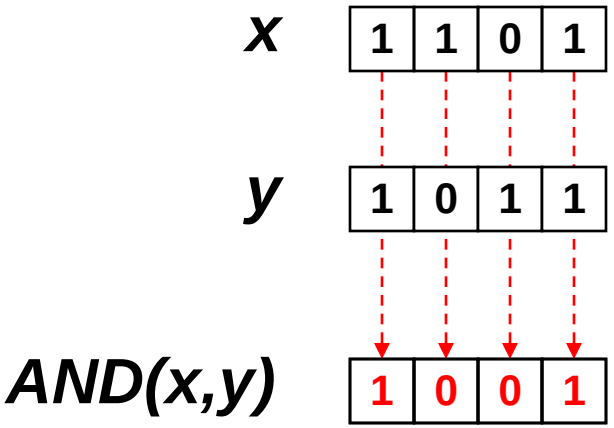
- **Die Hardware eines Rechensystems verwaltet i.d.R. nur Binärwörter von wenigen festen Längen. Solche Binärworter heißen *Maschinenwörter*.**
- **Günstige Längen von Maschinenwörtern sind durch verschiedene Faktoren bestimmt wie (vgl. Kap. 6):**
  - **Länge der kleinsten im Speicher adressierbaren Einheit**
  - **Länge der vom/zum Speicher transferierten Einheiten**
  - **Längen der Verarbeitungseinheiten (elementaren Datentypen) des Prozessors**
  - **Breite von Datenpfaden und Ein-/Ausgabe-Schnittstellen**
- **Übliche Maschinenwortlängen sind Vielfache von Bytes:**
  - **typisch: 32 Bit (4 Bytes) ("32-Bit-Prozessor")**
  - **aber auch z.B. 8 Bit und 16 Bit bei einfachen Mikroprozessoren / Mikrocontrollern**
  - **64 Bit bzw. 128 bit bei derzeitigen Hochleistungsprozessoren bzw. Graphik-Prozessoren**
  - **entsprechend Halbwort, Doppelwort, Vierfachwort.**

Bezeichnungen für Maßeinheiten

Abk.	*Bezeichnung*	=	*Vergleich*
B	Byte		Zeichen
KB	Kilobyte	2^{10} B = 1024 B ≈ 10^3 B	Seite ≈ 4.000 Zeichen
MB	Megabyte	1024 KB = 2^{20} B = 1.048.576 B ≈ 10^6 B	Arbeitsspeicher eines PCs heute ca. 128-256 MB
GB	Gigabyte	1024 MB = 2^{30} B = 1.073.741.824 B ≈ 10^9 B	Festplatte heute ca. 8-64 GB
TB	Terabyte	1024 GB = 2^{40} B ≈ 10^{12} B	große Datenbestände in Rechenzentren
	Pentabyte, ...	...	

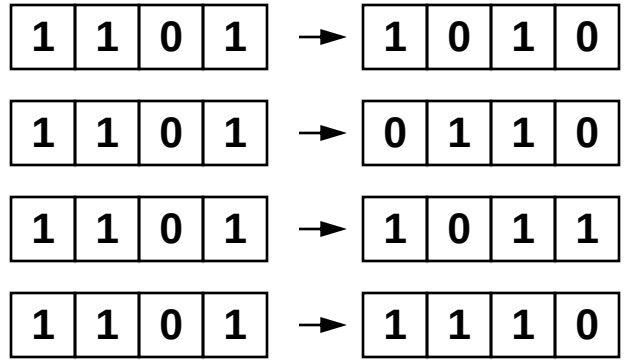
Durchführung der logischen Operationen

- Gleichzeitiges (paralleles) Anwenden der geforderten Boole'schen Funktion auf die korrespondierenden Bitstellen
- Beispiel:



Weitere Operationen auf Bitvektoren

- **Unterstützung in vielen Prozessoren für**
  - **Setzen und Löschen einzelner Bits**
  - **Test eines Bits, ob gesetzt**
  - **Verschieben nach links (shift left)**
  - **Verschieben nach rechts (shift right)**
  - **Rotieren nach links (rotate left)**
  - **Rotieren nach rechts (rotate right)**



4.3 Zahlensysteme, Zahlendarstellungen, Arithmetik

- **Betrachtung von zweckmäßigen Darstellungen für Zahlenmengen und zugehörigen arithmetischen Operationen**
- **Problem: Die aus der Mathematik üblichen Zahlenbereiche (natürliche Zahlen $\mathbb{N}$, ganze Zahlen $\mathbb{Z}$, rationale Zahlen $\mathbb{Q}$, reelle Zahlen $\mathbb{R}$) sind unendlich, die Anzahl der möglichen Codewörter in Maschinenwörtern fester Länge aber endlich.**

$\Rightarrow$ **exaktes Rechnen nur mit beschränkten Zahlenmengen oder approximative (angenäherte) Zahlendarstellungen und Operationen**

Überblick

1. Grundlagen Zahlensysteme
2. Konvertierung von Zahlenwerten
4. Darstellung ganzer Zahlen
4. Darstellung von Festkommazahlen
5. Darstellung von Gleitkommazahlen

4.4.1 Zahlensysteme

- **Grundlage:**
  Stellenwertsysteme (*polyadische* Systeme, *B-adische* Systeme)
- **Eine natürliche Zahl $n \in \mathbb{N}$ kann dargestellt werden durch**

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

Dabei ist

  - **B die Basis des Zahlensystems $B \in \mathbb{N}$, $B \geq 2$,**
  - **b_i sind Zahlenkoeffizienten $b_i \in \{0, ..., B-1\}$ (Ziffern),**
  - **nur endlich viele b_i sind $\neq 0$; sei N der größte Index.**

- **In der Kurzform werden nur die signifikanten Koeffizienten notiert:**

$$n = (b_N b_{N-1} b_{N-2} \ldots b_1 b_0)_B$$

Angabe der Basis B

  d.h. führende Nullen werden unterdrückt.

- **Jede natürliche Zahl läßt sich zu jeder Basis B darstellen.**
- **Die Koeffizientenfolge (b_i) ist bei gegebener Basis B eindeutig bestimmt.**

Wichtige B-adische Zahlensysteme

- **Die für die Informatik wichtigsten Basen**

B=10	Dezimalsystem	Ziffern 0, ..., 9
B=2	Dual-/Binärsystem	Ziffern 0, 1
B=8	Oktalsystem	Ziffern 0, ..., 7
B=16	Hexadezimalsystem Sedezimalsystem	Ziffern 0, ..., 9, A, B, C, D, E, F

A 10_{10}
B 11_{10}
C 12_{10}
D 13_{10}
E 14_{10}
F 15_{10}

- **Beispiel:**
  unterschiedliche Darstellungen der gleichen Zahl 28_{10}

$28_{10} = 2*10^1 + 8*10^0$

$103_5 = 1*5^2 + 0*5^1 + 3*5^0$

$11100_2 = 1*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 0*2^0$

$34_8 = 3*8^1 + 4*8^0$

$1C_{16} = 1*16^1 + C*16^0$

$= 12_{10}$

Anmerkungen

- **Je größer die Basis ...**
  - **um so weniger Ziffern benötigt man zur Darstellung einer Zahl**
  - **um so schwieriger ist das "kleine Einmaleins" (1...B mal 1...B)**
- **Bedeutung des Hexadezimalsystems**
  - **besser lesbar als Dualdarstellung derselben Zahl**
  - **unmittelbare Umwandlung zum/vom Dualsystem (vgl. auch 4.4.2)**
  - **breite Anwendung in der Datenverarbeitung**
    - **Angabe von Arbeitsspeicheradressen**
    - **Inhalte von Maschinenwörtern**
    - **Operanden für Bitfeld-Operationen**
    - **C-Notation: Beispiel 0xFFFF**
    - **...**

Darstellung von Brüchen

- Ein Bruch kann durch negative Exponenten dargestellt werden:

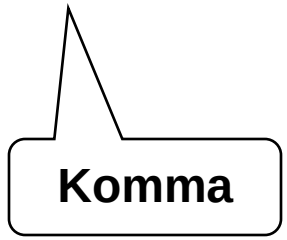
$$z = \sum_{i=-M}^{-1} b_i B^i$$

- Zusammengefaßt ergibt sich damit zur Darstellung rationaler Zahlen (und zur Approximation reeller Zahlen):

$$x = \sum_{i=-M}^{N} b_i B^i = \sum_{i=-M}^{-1} b_i B^i + \sum_{i=0}^{N} b_i B^i$$

- bzw. in der Kurzform der signifikanten Koeffizienten (Ziffern):

$$x = (b_N b_{N-1} b_{N-2} \ldots b_1 b_0 , b_{-1} b_{-2} \ldots b_{-M})_B$$



Horner-Schema

- andere Schreibweise, bedeutend wegen vereinfachter Berechnungsweise (vgl. 4.4.2)
- für natürliche Zahlen:

$$n = \sum_{i=0}^{N} b_i B^i = ((\ldots (b_N * B + b_{N-1}) * B + \ldots + b_2) * B + b_1) * B + b_0$$

- für den gebrochenen Anteil:

$$z = \sum_{i=-M}^{-1} b_i B^i = ((\ldots (b_{-M} / B + b_{-M+1}) / B + \ldots + b_{-2}) / B + b_{-1}) / B$$

4.4.2 Konvertierung von Zahlenwerten

- **Aufgabe**
  - **(a) Gegeben sei eine natürliche Zahl n $\in \mathbb{N}$ zur Ausgangsbasis B' (hier häufig 10 oder 2). Bestimme die Zahlendarstellung von n zur Zielbasis B (hier häufig 2 oder 10).**
  - **(b) analog für Brüche**
- **Methoden basieren auf:**
  - **Potenzreihendarstellung**
  - **Horner-Schema**
- **Unterscheidung**
  - **Rechnen im Ausgangssystem**
  - **Rechnen im Zielsystem**

Division durch fallende Potenzen der Zielbasis

- **Rechnen im Ausgangssystem**
- **Grundlage:**

$$n = \sum_{i=0}^{N} b_i B^i = b_N*B^N + b_{N-1}*B^{N-1} + ... + b_2*B^2 + b_1*B + b_0$$

- **Verfahren**
  - **(1) Bestimme höchste Potenz N mit $B^N \leq n$**
  - **(2) Bestimme Koeff. $b_N = \lfloor n / B^N \rfloor$ ($\lfloor x \rfloor$ größte ganze Zahl kleiner gleich x)**
  - **(3) Bilde $R_{N-1} := n - b_N*B^N$**
  - **(4) Betrachte analog die nächstkleinere Potenz i (N-1) und bestimme den Koeff. $b_i = \lfloor R_i / B^i \rfloor$**
  - **(5) Setze $R_{i-1} := R_i - b_i*B^i$**
  - **(6) Wiederhole (4) und (5) bis i=0**
  - **(7) $(b_N\, b_{N-1} ...\, b_2\, b_1\, b_0)_B$ ist die Darstellung von n zur Basis B.**

Beispiel 1

- Konvertiere 122_{10} zur Basis B=2
- N = 6

R_i	$/ B^i$	b_i
n=122	2^6=64	b_6= 1
122-64=58	2^5=32	b_5= 1
58-32=26	2^4=16	b_4= 1
26-16=10	2^3=8	b_3= 1
10-8=2	2^2=4	b_2= 0
2-0=2	2^1=2	b_1= 1
2-2=0	2^0=1	b_0= 0

$2^8 = 256$
$2^7 = 128$
$2^6 = 64$
$2^5 = 32$
$2^4 = 16$
$2^3 = 8$
$2^4 = 4$
$2^1 = 2$
$2^0 = 1$

$122_{10} = 1111010_2$

Beispiel 2

- Konvertiere $n=122_{10}$ zur Basis B=5
- N = 2

R_i	$/ B^i$	b_i
n=122	$5^2=25$	$b_2= 4$
122-4*25=22	$5^1=5$	$b_1= 4$
22-4*5=2	$5^0=1$	$b_0= 2$

$\Rightarrow$ $122_{10} = 442_5$

$5^4 = 625$
$5^3 = 125$
$5^2 = 25$
$5^1 = 5$
$5^0 = 1$

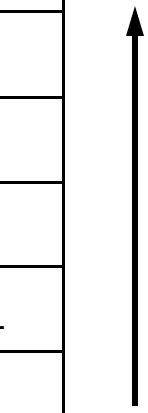
Horner-Schema für natürliche Zahlen (a)

- **Grundlage:**

$$n = \sum_{i=0}^{N} b_i B^i = ((\ldots(b_N * B + b_{N-1}) * B + \ldots + b_2) * B + b_1) * B + b_0$$

- **(a) Rechnen im Ausgangssystem**
- **Verfahren: schrittweise Division von $n \in \mathbb{N}$ durch die Zielbasis B**

Schritt	Division	Quotient	Rest
1	n / B	$(\ldots(b_N*B + b_{N-1})*B + \ldots + b_2)*B + b_1$	b_0
2	(n/B)/B	$\ldots(b_N*B + b_{N-1})*B + \ldots + b_2$	b_1
...	...	...	...
N	n / B^N	b_N	b_{N-1}
N+1	n / B^{N+1}	0	b_N

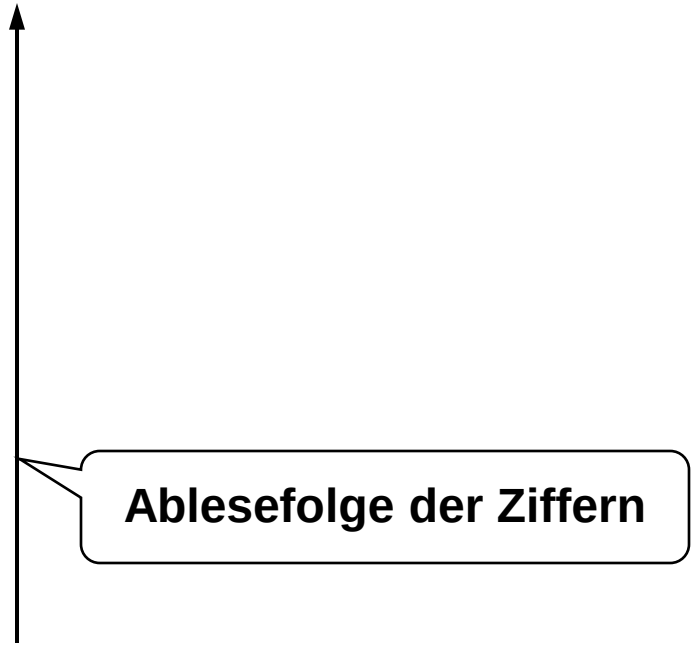


⇒ $n = (b_N \; b_{N-1} \ldots b_2 \; b_1 \; b_0)_B$

Beispiel 1

- **Konvertiere 122_{10} zur Basis B=2**
- **Rechnen im Quellsystem (hier Dezimalsystem)**

Schritt	/ B	Quotient	Rest
122	/2	61	0
61	/2	30	1
30	/2	15	0
15	/2	7	1
7	/2	3	1
3	/2	1	1
1	/2	0	1



$$122_{10} = 1111010_2$$

Beispiel 2

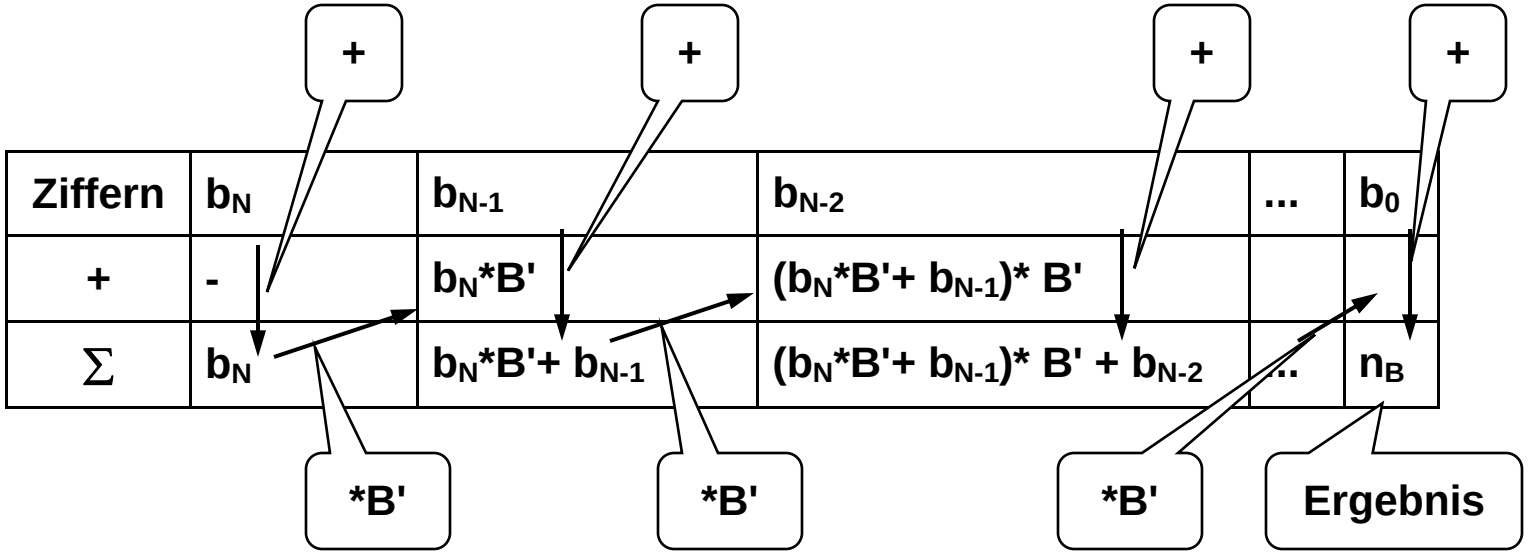
- **Konvertiere 122_{10} zur Basis B=5**
- **Rechnen im Quellsystem (hier Dezimalsystem)**

Schritt	/ B	Quotient	Rest
122	/5	24	2
24	/5	4	4
4	/5	0	4

$\Rightarrow 122_{10} = 442_5$

Horner-Schema für natürliche Zahlen (b)

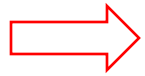
- **(b) Rechnen im Zielsystem**
- **Verfahren:**
  - **Darstellen aller Ziffern und der Ausgangsbasis B' im Zielsystem**
  - **Auswerten des Horner-Schemas von "innen" nach "außen"**



Beispiel 1

- Konvertiere 1111010_2 zur Basis B=10
- Rechnen im Zielsystem (hier Dezimalsystem)
- $B'=2_{10}$

Ziffern	1	1	1	1	0	1	0
+	–	1*2	3*2	7*2	15*2	30*2	61*2
$\sum$	1	3	7	15	30	61	122



$1111010_2 = 122_{10}$

Beispiel 2

- Konvertiere $2AC_{16}$ zur Basis B=10
- Rechnen im Zielsystem (hier Dezimalsystem)
- $B'=16_{10}$

		A_{16}	C_{16}
Ziffern	2	10	12
+	-	2*16	42*16
$\sum$	2	42	684

$$2AC_{16} = 684_{10}$$

Konvertierung gebrochener Zahlen

- Getrennte Behandlung von Vorkommateil und Nachkommateil
- Vorkommateil ist natürliche Zahl, Umwandlung wie beschrieben
- Umwandlung des Nachkommateils entsprechend dem Horner-Schema für Brüche (nächste Folie)
- Weitere Probleme
  - Anzahl der notwendigen Schritte nicht von vornherein bekannt
  - ein gegebener endlicher Bruch muss zur Zielbasis nicht endlich sein

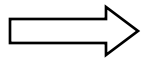
Horner-Schema für Brüche (a)

- **Grundlage:**

$$z = \sum_{i=-M}^{-1} b_i B^i = ((\ldots(b_{-M} / B + b_{-M+1}) / B + \ldots + b_{-2}) / B + b_{-1}) / B$$

- **(a) Rechnen im Ausgangssystem**
- **Verfahren: schrittweise Multiplikation von $n \in \mathbb{N}$ mit der Zielbasis B**

Schritt	Mult.	Produkt (Bruchteil)	Ganzteil b_i
1	$z * B$	$(\ldots(b_{-M}/B + b_{-M+1})/B + \ldots + b_{-3})/B + b_{-2})/B$	b_{-1}
2	$(z*B)*B$	$(\ldots(b_{-M}/B + b_{-M+1})/B + \ldots + b_{-3})/B$	b_{-2}
...	...	...	...
M-1	$z * B^{M-1}$	b_{-M}/B	b_{-M+1}
M	$z * B^{M}$	0	b_{-M}



$z = (0 , b_{-1}\, b_{-2} \ldots b_{-M})_B$

Beispiel

- Konvertiere $0{,}21_{10}$ zur Basis B=2
- Rechnen im Quellsystem (hier Dezimalsystem)

Wert	* B	Produkt (Zwischenerg.)	Bruchteil	Ganzteil
0,21	*2	0,42	0,42	0
0,42	*2	0,84	0,84	0
0,84	*2	1,68	0,68	1
0,68	*2	1,36	0,36	1
0,36	*2	0,72	0,72	0
0,72	*2	1,44	0,44	1
0,44	*2	0,88	...	...

$\Rightarrow 0{,}21_{10} = 0{,}001101\ldots_2$

Horner-Schema für Brüche (b)

- **(b) Rechnen im Zielsystem**
- **entspricht dem Vorgehen bei natürlichen Zahlen mit Division statt Multiplikation**
- **Verfahren:**
  - **Darstellen aller Ziffern und der Ausgangsbasis B' im Zielsystem**
  - **Auswerten des Horner-Schemas von "innen" nach "außen":**
    - **(1) kleinstwertige Ziffer durch die Ausgangsbasis in deren Zieldarstellung dividieren**
    - **(2) Ergebnis zur nächsthöherwertigen Stelle addieren**
    - **(3) Ergebnis aus (2) wieder durch Ausgangsbasis in deren Zieldarstellung dividieren**
    - **(4) wiederholen (2) und (3), bis alle Stellen verarbeitet sind**

Beispiel

- Konvertiere $0{,}01011_2$ zur Basis B=10
- Rechnen im Zielsystem (hier Dezimalsystem)
- $B'=2_{10}$

niederwertigste Ziffer b_{-M}

höchstwertigste Ziffer b_{-1}

Ziffern	1	1	0	1	0	,
+	-	1/2= 0,5	1,5/2= 0,75	0,75/2= 0,375	1,375/2= 0,6875	0,6875/2= 0,34375
$\sum$	1	1,5	0,75	1,375	0,6875	0,34375

$\Rightarrow 0{,}01011_2 = 0{,}34375_{10}$

Schnelle Konvertierung bei "verwandten" Basen

- **Für die Praxis von Bedeutung ist die schnelle Konvertierung zwischen Dual-, Oktal-, und Hexadezimalsystem**
- **Gilt allgemeiner für Basen B' und B mit $B = B'^{k}$, k ganzzahlig.**
- **Dann lassen sich Gruppen von k Ziffern der Darstellung zur Basis B' zusammenfassen zu einer Ziffer der Basis B, beginnend am Komma nach links und rechts.**
- **Beispiele:**

$1111010_2 = 1|111|010_2 = 172_8$

$1111010_2 = 111|1010_2 = 7A_{16}$

$2BC_{16} = 10|1011|1100_2 = 1|010|111|100_2 = 1274_8$

$1101001,11011_2 = 1|101|001,110|11_2 = 151,66_8$

$1101001,11011_2 = 110|1001,1101|1_2 = 69,D8_{16}$

Arithmetik in Stellenwertsystemen

- In Stellenwertsystemen beliebiger Basis B lassen sich prinzipiell die Verfahren des schriftlichen Rechnens anwenden, wie man sie vom Dezimalsystem her kennt
  - Addition mit Übertrag
  - Subtraktion mit Borgen
  - Multiplikation mit stellenrichtiger Addition
  - Division mit Rest
- Beispiel:

```
  24,13₅       24,13₅      24*13₅       10223₅:13₅ = 321₅
+ 12,34₅     - 12,34₅    -------         44
-------      -------        24          ---
  42,02₅       11,24₅       132          32
                          -------         31
                             422₅         ---
                                          13
                                          13
```

- Die in Rechensystemen angewendete Arithmetik wird im folgenden Abschnitt behandelt.

4.4.3 Darstellung ganzer Zahlen

- **Rechensysteme verwenden das Dual-/Binärsystem zur Speicherung und Verarbeitung aller Zahlendarstellungen.**
- **Maschinenwörter werden zur Aufnahme von Zahlendarstellungen verwendet.**
- **Die Wortlänge (Wortbreite) legt die Anzahl der darstellbaren Zahlenwerte fest:**

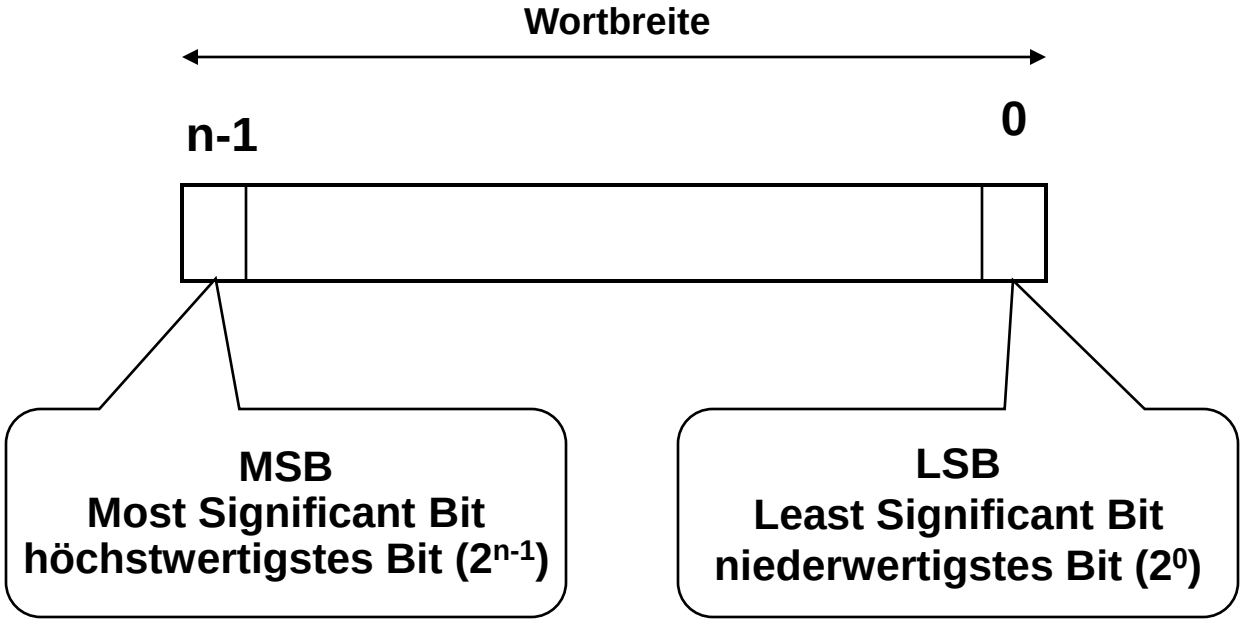
Wortlänge [bit]	Anzahl Darstellungen
8	$2^8 = 256$
16	$2^{16} = 65536$
32	$2^{32} \approx 4.3*10^9$
64	$2^{64} \approx 1.8*10^{19}$

- **Vorrangiges Ziel: exaktes Rechnen mit ganzen Zahlen**

$\Rightarrow$ **Umgang nur mit beschränkten Zahlenmengen möglich**

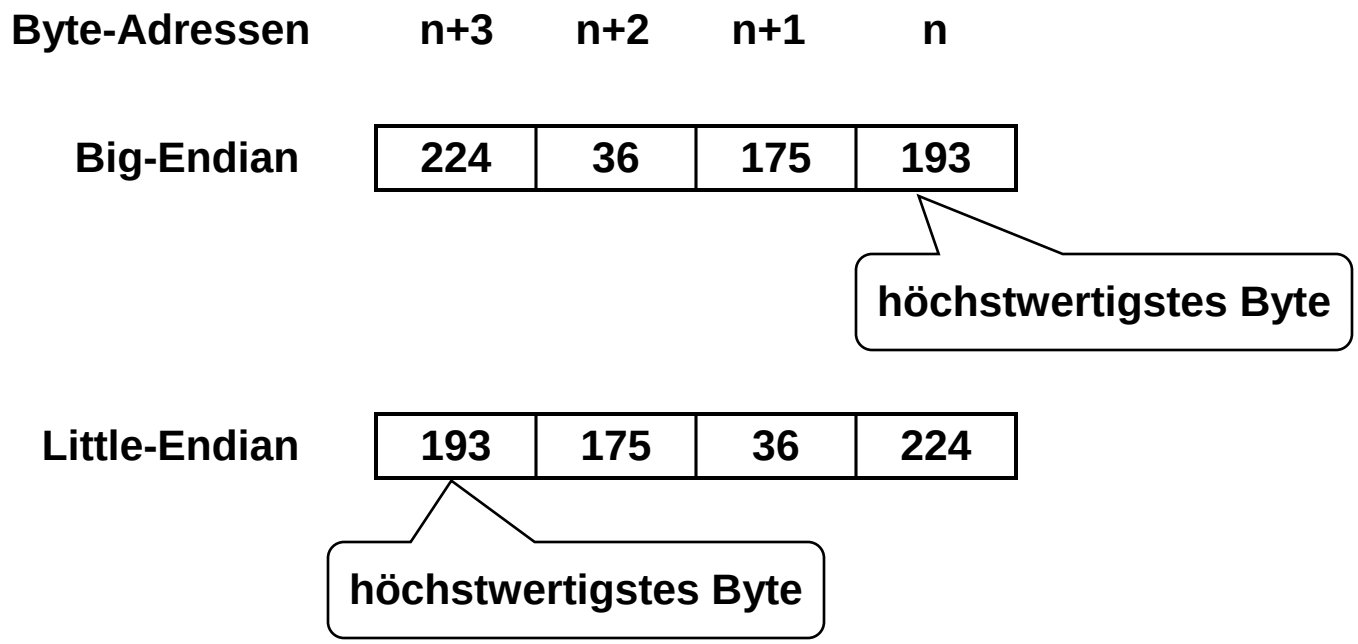
Bit-Positionen in Maschinenwörtern

- **Festlegung ausgezeichneter Bit-Positionen in n-Bit-Maschinenwörtern (typisch):**



Byte Ordering

- **Technisch sind zwei unterschiedliche Adressierungsweisen der Bytes in einem Maschinenwort möglich:**
  - ***Big-Endian*: höherwertige Stelle in Byte mit niederer Adresse. Beispiele: Sun SPARC, Motorola, IBM Mainframe**
  - ***Little-Endian*: niederwertige Stelle in Byte mit niederer Adresse. Beispiele: Intel x86, DEC VAX**



Vorzeichenlose ganze Zahlen

- Verwendung für einen Bereich natürlicher Zahlen $\mathbb{N}$ einschl. 0 sowie für Adressen von Speicherwörtern
- Programmiersprachenebene: *unsigned integer*
- Nutzung des gesamten Maschinenworts zur Darstellung der Zahl in Dualdarstellung

Wortlänge [bit]	Wertebereich	C (typisch)
n	$0 \ldots 2^n - 1$	
8	0 ... 255	(`unsigned char`)
16	0 ... 65535	`unsigned short int`
32	0 ... 4.294.967.295	`unsigned int`
64	$0 \ldots 2^{64} - 1$	`unsigned long int`

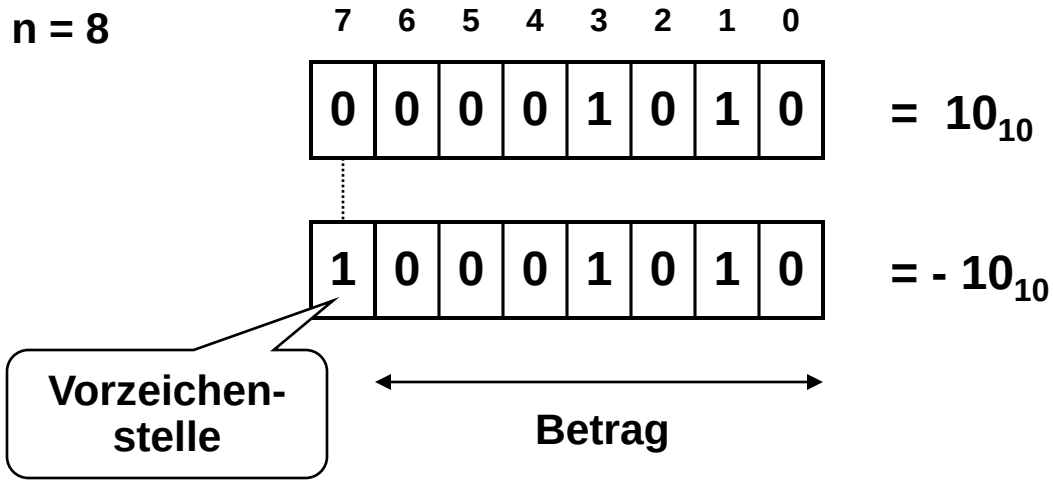
Vorzeichenbehaftete ganze Zahlen

Überblick:

- Verwendung für einen Bereich ganzer Zahlen $\mathbb{Z}$
- Programmiersprachenebene: *signed integer*
- Alternativen zur Darstellung
  - Vorzeichen/Betrags-Darstellung
  - Excess-Darstellung
  - Komplementdarstellung
    - B-1 - Komplement
    - B - Komplement
- Die verschiedenen Alternativen besitzen jeweils Vor- und Nachteile.
- In Hinblick auf die technische Realisierung der Arithmetik in einem Prozessor (Arithmetisch-Logische Einheit, vgl. Kap. 6) besitzt die Komplementdarstellung die meisten Vorteile.

Vorzeichen/Betrags-Darstellung

- **Zahlendarstellung mit Vorzeichen-Bit (engl. *sign bit*) und Betrag in einem n-Bit-Maschinenwort**
  - **Nutzung des höchstwertigen Bits (MSB) zur Aufnahme des Vorzeichens von z**
    - **MSB = 0: $z \geq 0$**
    - **MSB = 1: $z < 0$**
  - **Nutzung der restlichen Bits des Wortes zur Dualdarstellung des Betrags von z.**
- **Beispiel**
  - **n = 8**



Vorzeichen/Betrags-Darstellung (2)

- **Konsequenzen**
  - **Zahlenbereich $-(2^{n-1}-1) \ldots (2^{n-1}-1)$**
  - **Zwei Darstellungen der "Null" (00...0 und 10...0, genannt 0 und -0)**
  - **Unterschiedliche Behandlung von Subtraktion und Addition, Fallunterscheidungen**
  - **relativ hoher Hardware-Aufwand wäre notwendig**

$\Rightarrow$ **in realen Prozessoren nicht verwendet**

Excess-Darstellung

Def

- Sei z eine ganze Zahl. Dann heißt z' = z + k die *Excess-k-Darstellung* von x (Hinzuaddieren eines festen Betrags (Excess)).
- Anwendung:
  - Maschinenwortlänge n
  - $-2^{n-1} \leq z \leq 2^{n-1}-1$, $k = 2^{n-1}$, $\Rightarrow$ $0 \leq z' \leq 2^n-1$
- Beispiel
  - $n = 8$, $k = 2^{n-1} = 128$
  - $-128 \leq z \leq 127$
  - $z' = z+128$ ist die Excess-128-Darstellung von z, $\Rightarrow$ $0 \leq z' \leq 255$
- Vorteile/Nachteile
  - Inkrementieren/Dekrementieren wie bei vorzeichenlosen ganzen Zahlen
  - Ordnungsbeziehung bleibt erhalten: $y' \leq z' \Rightarrow y \leq z$
  - Korrektur bei Addition/Subtraktion: $y'+z' = y+k + z+k = (y+z)' +k \neq (y+k)'$
- Anwendung
  - Exponentendarstellung von Gleitpunktzahlen (vgl. 4.4.5)
  - Analog/Digital- und Digital/Analog-Wandler (vgl. 4.5)

Komplement-Bildung

Def

- Sei B Basis eines Stellenwertsystems, n die betrachtete Wortlänge, z eine ganze Zahl zur Basis B.
  Das *B-Komplement* $^{(B)}z$ von z wird definiert durch $z + {}^{(B)}z = B^n$.
- Es gilt:

$$ {}^{(B)}z = B^n - z = 1 + \sum_{i=0}^{n-1} (B-1)\, B^i - \sum_{i=0}^{n-1} z_i\, B^i = 1 + \underbrace{\sum_{i=0}^{n-1} (B-1-z_i)\, B^i}_{{}^{(B-1)}z} $$

$^{(B-1)}z$ heißt *(B-1)-Komplement* oder *Stellenkomplement*

- Es gilt damit: $z + {}^{(B-1)}z = B^n - 1$ sowie ${}^{(B)}z = {}^{(B-1)}z + 1$
- Das (B-1)-Komplement ergibt sich also durch stellenweise Komplement-Bildung zur größten Ziffer (B-1) der betrachteten Basis.
- Das B-Komplement ergibt sich aus dem (B-1)-Komplement durch Addition von 1.

Beispiel 1

- Betrachtung der Basis B=10 (Dezimalsystem)
  - (B-1)-Komplement: Neuner-Komplement (Ergänzung zu 9)
  - B-Komplement: Zehner-Komplement
- Beispiel

  n=6 Stellen

  $z = 003910_{10}$

  $^{(9)}z = 996089_{10}$ — stellenweise Ergänzung zu B-1=9

  $+1$

  $^{(10)}z = 996090_{10}$ — $^{(10)}z = {}^{(9)}z + 1$

  $z + {}^{(10)}z = 10^6$

Beispiel 2

- **Betrachtung der Basis B=2 (Dualsystem)**
  - **(B-1)-Komplement: Einer-Komplement (Ergänzung zu 1)**
  - **B-Komplement: Zweier-Komplement**
- **Einer-Komplement entspricht stellenweiser Invertierung (Hardware-mäßig einfach zu implementieren!)**
- **Beispiel**

n=8 Stellen

$z = 00111010_2$

$^{(1)}z = 11000101_2$ — **stellenweise Invertierung**

$+1_2$

$^{(2)}z = 11000110_2$ — $^{(2)}z = {}^{(1)}z + 1$

Einer-Komplement-Darstellung ganzer Zahlen

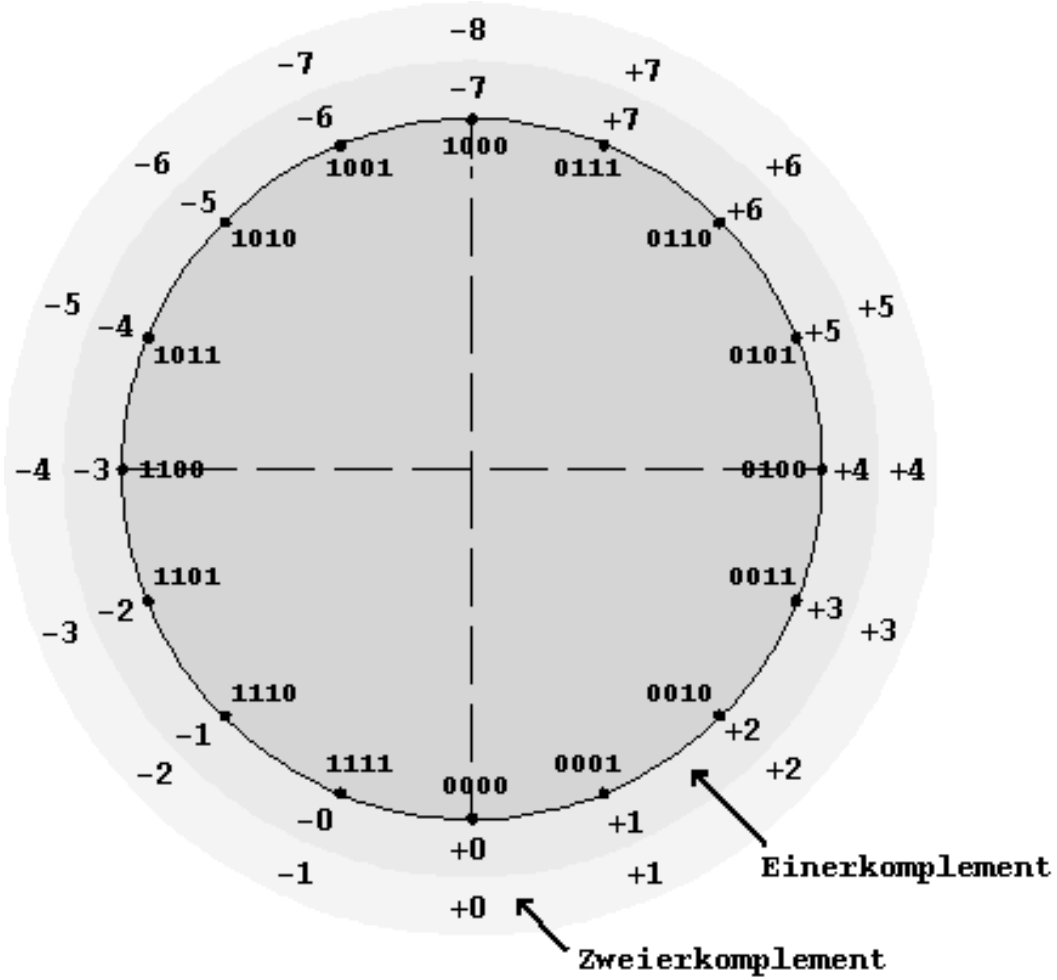
- Zahlendarstellung in einem n-Bit-Maschinenwort
- Aufteilung des Darstellungsbereichs in 2 Hälften:
  - die nicht-negativen ganzen Zahlen 0, ..., 2^{n-1}-1 werden in Dualdarstellung wie in der Vorzeichen/Betragsweise dargestellt.
  - die negativen ganze Zahlen -(2^{n-1}-1), ..., -0 werden durch das Einer-Komplement der betragsgleichen positiven Zahl dargestellt.
- Vorteil:
  - symmetrischer Bereich dargestellter Zahlen
- Nachteile:
  - doppelte Darstellung der Null (00...0 und 11...1)
  - Korrektur bei Addition/Subtraktion, z.B.
    $^{(1)}y+^{(1)}z = B^n\text{-}y\text{-}1 + B^n\text{-}z\text{-}1 = B^n + B^n\text{-}(y+z)\text{-}1\ \text{-}1 = {}^{(1)}(y+z)\ \text{-}1 \neq {}^{(1)}(y+z)$

Zweier-Komplement-Darstellung ganzer Zahlen

- Zahlendarstellung in einem n-Bit-Maschinenwort
- Aufteilung des Darstellungsbereichs in 2 Hälften:
  - die nicht-negativen ganzen Zahlen 0, ..., 2^{n-1}-1 werden in Dualdarstellung wie bisher dargestellt.
  - die negativen ganze Zahlen -2^{n-1}, ..., -1 werden durch das Zweier-Komplement der betragsgleichen positiven Zahl dargestellt und belegen dadurch den Bereich 2^{n-1}-1, ..., 2^n-1.
- Nachteil:
  - asymmetrischer Bereich dargestellter Zahlen (Absolutbetrag von -2^{n-1} nicht darstellbar)
- Vorteile:
  - Beseitigung der Nachteile der Einer-Komplement-Darstellung
  - Vorzeichen einer Zahl ist weiter am MSB ablesbar: Eine Zahl ist negativ $\Leftrightarrow$ MSB=1
  - Einfache Arithmetik (s.u.)

Verdeutlichung

- **Zahlenkreis (n=4)**



Wertebereiche für ganze Zahlen

- **Betrachtung**
  - **n-Bit Maschinenwörter**
  - **zweier-Komplement-Darstellung ganzer Zahlen**
- **Programmiersprachenebene: *signed integer***

Wortlänge [bit]	Wertebereich	C (typisch)
n	$-2^{n-1} \ldots 2^{n-1}-1$	
8	- 128 ... 0 ... 127	(`signed char`)
16	- 32.768 ... 0 ... 32767	`signed short int`
32	- 2.147.483.648 ... 2.147.483.647	`signed int`
64	$-2^{63} \ldots 2^{63}-1$	`signed long int`

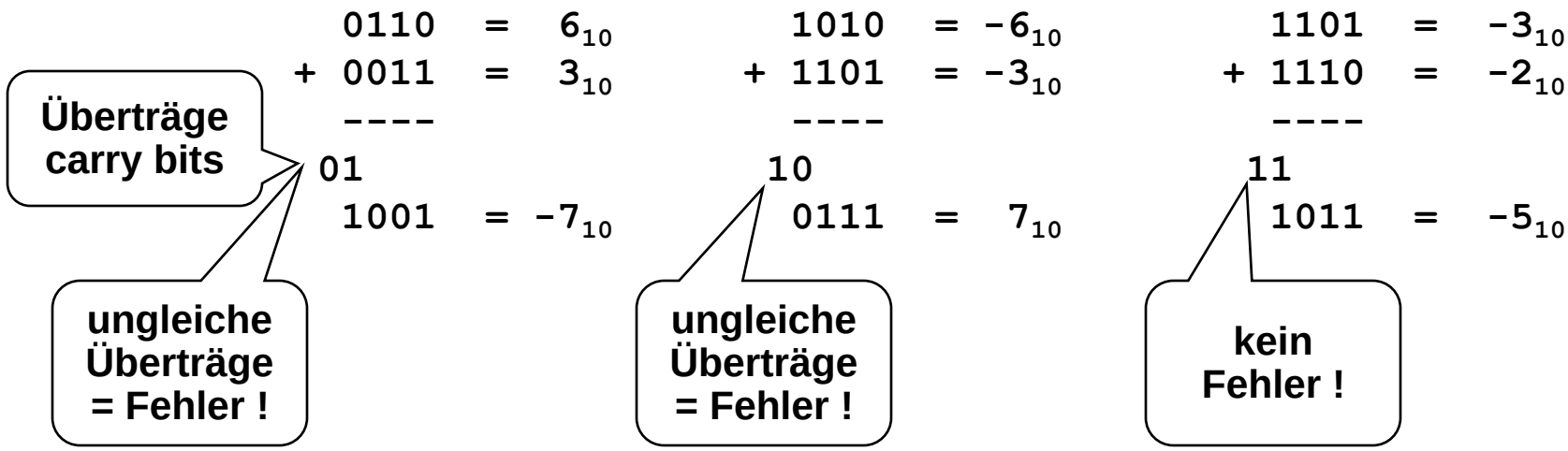
Arithmetik ganzer Zahlen

- **Beschränkung auf Zweier-Komplement-Darstellung**
- **Für Addition/Subtraktion wird nur ein Addierwerk benötigt:**
  - **Summe y+z zweier ganzer Zahlen kann durch gewöhnliche Addition im Dualsystem gebildet werden.**
  - **Subtraktion y-z wird durch Addition des Zweier-Komplements von z erbracht.**
- **Das Vorzeichen wird wie eine normale Stelle behandelt!**
  Ein evtl. Übertrag in die n+1. Stelle wird nicht weiter beachtet.
  (außer zur Fehlererkennung, siehe nächste Folie)
- **Beispiele (n=4):**

0011	=	3_{10}	0111	=	7_{10}	1101	=	-3_{10}
+ 0010	=	2_{10}	+ 1010	=	-6_{10}	+ 1110	=	-2_{10}
----			----			----		
0101	=	5_{10}	(1)0001	=	1_{10}	(1)1011	=	-5_{10}

Erkennung von Überlauf

- ***Überlauf*** **(*Overflow*): das Ergebnis einer Operation ist außerhalb des darstellbaren Zahlenbereichs und damit ungültig.**
- **Erkennungsregel:**
  Bei der Addition zweier Zahlen in Zweier-Komplement-Darstellung findet genau dann ein Überlauf statt, wenn die Überträge (*Carry*) in die n. Stelle und in die gedachte (n+1). Stelle verschieden sind. (Realisierung durch einfache Hardware-Schaltung möglich)
- **Beispiel:**



Arithmetik ganzer Zahlen (2)

- **Multiplikation und Division könnten prinzipiell durch fortgesetzte Addition/Subtraktion und Verschieben basierend auf den Betragswerten und evtl. Komplementierung erfolgen.**
- **Tatächlich existieren in den meisten heutigen Prozessoren spezielle zusätzliche Multiplizierwerke, die eine schnelle Multiplikation erlauben.**
- **Rechenbeispiel:**

```
10110*1001          11011100:10100 = 1011
----------          10100
  10110             -----
     10110            11110
----------            10100
  11000110            -----
                       10100
                       10100
                       -----
                           0
```

4.4.4 Darstellung von Festkommazahlen

- **Prinzip: In der Zahlendarstellung wird an einer beliebigen aber festen Stelle ein Komma angenommen.**
- **Zahlendarstellung und Arithmetik könnten weiter binär sein (vgl. 4.4.2).**
- **Beispiel: $5{,}25_{10} = 2^2 + 2^0 + 2^{-2} = 101{,}01_2$**
- **Da das Komma nur gedacht ist, bliebe die Durchführung der Operationen bis auf die extern notwendige Verwaltung des Kommas unverändert.**

Probleme

- **Ungenauigkeit bei der Konvertierung**
  - **Dezimalzahlen mit endlich vielen Nachkommastellen haben häufig keine endliche Dualdarstellung (vgl. 4.4.2)**
- **Rundungsfehler**
  - **Sorgfältige Wahl der Ausführungsreihenfolge und Genauigkeit von Zwischenergebnissen ist notwendig**
  - **Beispiel:**
    - **2 Vorkommastellen, 1 Nachkommastelle**
    - **(00,2 * 00,3) * 20,0 = 00,0 * 20,0 = 00,0**
    - **00,2 * (00,3 * 20,0) = 00,2 * 06,0 = 01,2 !!!**
- **Ungenauigkeiten bei kaufmännischen Anwendungen werden oft als nicht akzeptabel angesehen (Buchführung "auf den Pfennig genau").**

⇒ **Einführung einer Dezimalarithmetik im Dualsystem durch sogenannte BCD-Zahlen (*Binary Coded Decimal*).**

Festkomma-Prinzip durch gedachte Anzahl von Nachkommastellen

BCD-Darstellung

- **Darstellung der Dezimalziffern in 4-stelligen Dualzahlen**

Dezimal-ziffer	BCD-Darstellung
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
	1010
	1011
	1100
	1101
	1110
	1111

ungültig:
Pseudotetraden

- **Dezimalzahlen werden ziffernweise in BCD-Darstellung überführt**

BCD-Arithmetik

- **BCD-Zahlen werden wie gewöhnliche Dualzahlen addiert unter Beachtung von zwei Korrekturen:**
  - **(a) Bei Übertrag in die nächste Tetrade muß Korrekturwert 6 (0110_2) zur ausgehenden Tetrade addiert werden (Differenz der Basen 16 und 10).**
  - **(b) Tritt Pseudotetrade als Ergebnis auf, so muß Korrekturwert 6 (0110_2) addiert werden.**

    $\Rightarrow$ **Korrekte BCD-Ziffer wird erzeugt**

    Übertrag in nächsthöhere Stelle wird generiert (Korrektur nach (a) entfällt)
- **Beispiele:**

zu (a)

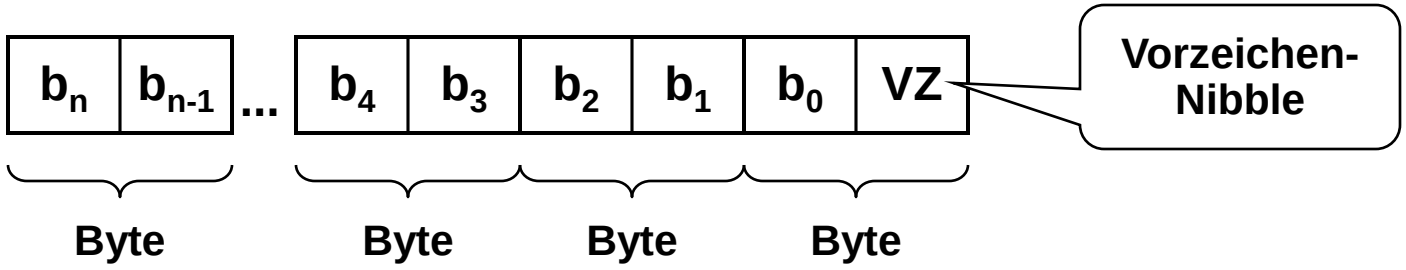
```
    1000  =  8₁₀
+   1001  =  9₁₀
    ----
0001 0001 = 11₁₀
+   0110  =  6₁₀
-----------
0001 0111 = 17₁₀
```

zu (b)

```
    1000  =  8₁₀
+   0100  =  4₁₀
    ----
    1100  = 12₁₀  Pseudotetrade
+   0110  =  6₁₀  Korrekturwert
-----------
0001 0010 = 12₁₀
```

Beispiel

- **Speicherung und Verarbeitung von BCD-Zahlen ist unterschiedlich in verschiedenen Prozessor-Architekturen**
- **Beispiel (IBM /360, Mainframes):**
  - **Format: packed decimal**
  - **je zwei BCD-Dezimalziffern b_i werden in einem Byte (High und Low Nibble) gespeichert.**
  - **variabel lange Darstellung von 1...31 Dezimalstellen in 1-16 Bytes**
  - **gedachtes, extern verwaltetes Komma**
  - **Vorzeichen wird in separatem Nibble dargestellt, gültige Vorzeichen sind alle Pseudotetraden + : CAFE, - : BD**
  - **separates BCD-Rechenwerk**



4.4 Zeichenketten

- **Ziel: Repräsentierung von Wörtern über einem ungeordneten Zeichenvorrat oder einem Alphabet von Schriftzeichen.**
- **Ein Schriftzeichen-Alphabet umfaßt i.d.R.**
  - **Buchstaben (Groß- und Klein-Buchstaben)**
  - **Ziffern**
  - **druckbare Sonderzeichen**
  - **evtl. sogenannte Steuerzeichen, die durch ein verarbeitendes Gerät interpretiert werden.**

  Man spricht daher auch von alphanumerischen Codes.
- **Die wichtigsten alphanumerischen Codes, die im weiteren vorgestellt werden, sind:**
  - **CCITT-Code No. 2 (historisch)**
  - **ASCII**
  - **erweiterter ASCII Code (PC8)**
  - **EBCDIC**
  - **UNICODE**

Operationen auf Zeichenketten

- **Zeichenketten werden in Rechensystemen in Bytefolgen gespeichert.**
- **Maschinenoperationen sind häufig auf Zeichenketten einer bestimmten maximalen Länge beschränkt (z.B. 255 Zeichen).**
- **Neben Operationen zum Kopieren von Zeichenketten (allgemeiner von Bytefolgen) ist das Vergleichen von Zeichenketten in Hinblick auf die lexikographische Ordnung (vgl. Kap. 2.4) besonders wichtig.**

4.4.1 CCITT-Code No. 2

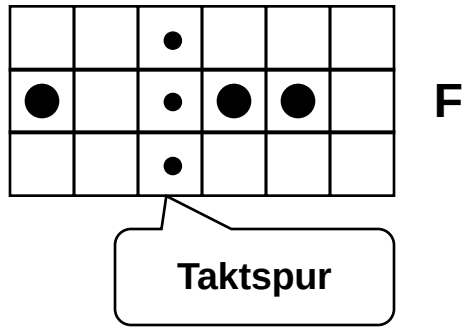
- **CCITT:**
  Comité Consultatif International Telegraphique et Telephonique, jetzt International Telecommunications Union (ITU) genannt
- **von D. Murray entwickelter Fernschreibcode**
- **5-Bit Code (2^5=32 Codewörter)**
  (entsprechend 5 Spuren eines Lochstreifens mit Taktspur)
- **Steuerzeichen zur Umschaltung zwischen Buchstabenmodus und Ziffern/Sonderzeichen-Modus verdoppelt den Vorrat an Codewörtern**

Code-Tabelle CCITT-2

Code-Nr.	Dual-code	Buch-stabe	Ziffer
1	11000	A	-
2	10011	B	?
3	01110	C	:
4	10010	D	WAY
5	10000	E	3
6	10110	F	(NA)
7	01011	G	(NA)
8	00101	H	(NA)
9	01100	I	8
10	11010	J	*bell*
11	11110	K	(
12	01001	L	)
13	00111	M	.
14	00110	N	,
15	00011	O	9
16	01101	P	0

Code-Nr.	Dual-code	Buch-stabe	Ziffer
17	11101	Q	1
18	01010	R	4
19	10100	S	'
20	00001	T	5
21	11100	U	7
22	01111	V	=
23	11001	W	2
24	10111	X	/
25	10101	Y	6
26	10001	Z	+
27	00010	CR	
28	01000	LF	
29	11111	LS	
30	11011	FS	
31	00100	*space*	
32	00000	*(unused)*	

WAY	Who Are You? (Wer da?)
bell	Klingel
CR	Carriage Return (Wagenrücklauf)
LF	Line Feed (Zeilenvorschub)
LS	Letter Shift (Buchstabenumsch.)
FS	Figure Shift (Ziffernumschaltung)
space	Zwischenraum
(NA)	Not Assigned (nicht definiert, reserviert)
(unused)	nicht benutzt



4.4.2 ASCII-Code

- ASCII: American Standard Code for Information Interchange (1963)
- nationale amerikanische Version, auch als US-ASCII bezeichnet
- entspricht CCITT-Code No. 5
- keine Umlaute, diese in oft nicht standardisierten Varianten statt einiger Sonderzeichen
- sehr verbreitet zur Codierung von Zeichenketten in Rechensystemen
- 7-Bit Code (2^7=128 Codewörter)
- Speicherung eines Zeichens in einem Byte
- Steuerzeichen (Control Codes) in den Positionen 0-31 und 127

Code-Tabelle US-ASCII

HEX	ASCII	HEX	ASCII	HEX	ASCII	HEX	ASCII
00	NUL	10	DLE	20	SP	30	0
01	SOH	11	DC1	21	!	31	1
02	STX	12	DC2	22	"	32	2
03	ETX	13	DC3	23	#	33	3
04	EOT	14	DC4	24	$	34	4
05	ENQ	15	NAK	25	%	35	5
06	ACK	16	SYN	26	&	36	6
07	BEL	17	ETB	27	'	37	7
08	BS	18	CAN	28	(	38	8
09	HT	19	EM	29	)	39	9
0A	LF	1A	SUB	2A	*	3A	:
0B	VT	1B	ESC	2B	+	3B	;
0C	FF	1C	FS	2C	,	3C	<
0D	CR	1D	GS	2D	-	3D	=
0E	SO	1E	RS	2E	.	3E	>
0F	SI	1F	US	2F	/	3F	?

HEX	ASCII	HEX	ASCII	HEX	ASCII	HEX	ASCII
40	@	50	P	60	`	70	p
41	A	51	Q	61	a	71	q
42	B	52	R	62	b	72	r
43	C	53	S	63	c	73	s
44	D	54	T	64	d	74	t
45	E	55	U	65	e	75	u
46	F	56	V	66	f	76	v
47	G	57	W	67	g	77	w
48	H	58	X	68	h	78	x
49	I	59	Y	69	i	79	y
4A	J	5A	Z	6A	j	7A	z
4B	K	5B	[	6B	k	7B	{
4C	L	5C	\	6C	l	7C	|
4D	M	5D	]	6D	m	7D	}
4E	N	5E	^	6E	n	7E	~
4F	O	5F	_	6F	o	7F	DEL

HEX	ASCII	Bedeutung
02	STX	Textanfang *(start of text)*
03	ETX	Textende *(end of text)*
07	BEL	Klingel *(bell)*
09	HT	Tabulator *(horizontal tabulator)*
0A	LF	Zeilenvorschub *(line feed)*
0C	FF	Seitenvorschub *(form feed)*
0D	CR	Wagenrücklauf *(carriage return)*
1B	ESC	Umschaltung *(Escape)*

Ordnung:

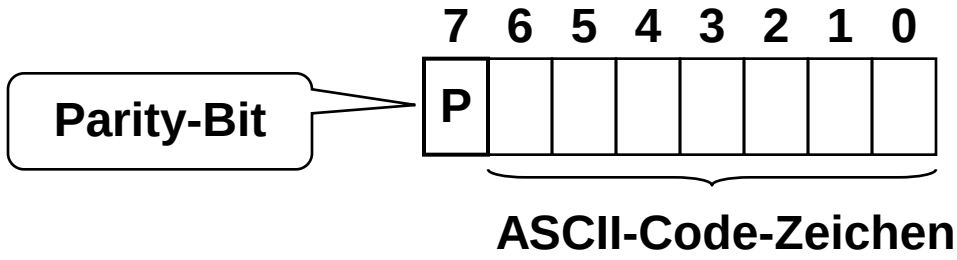
es gilt inbesondere

Leerzeichen<0<1<...<9<A<B<...<Z<a<b<...<z

einige wichtige Steuer-Codes

Paritätsprüfung

- **Das zur Speicherung eines Zeichens in einem Byte nicht benötigte Bit kann zur Fehlerkontrolle (Paritätsprüfung) eingesetzt werden**



- **Unterscheidung**
  - **gerade Parität (even parity):**
    P=0, wenn Anzahl der "1" im Code bereits gerade ist,
    P=1 sonst (damit insgesamt wieder gerade Anzahl)
  - **ungerade Parität (odd parity): umgekehrt**
- **Paritätsprüfung (parity check)**
  - **Erkennung von 1-Bit-Übertragungsfehlern (alle ungeraden Anzahlen)**
  - **Berechnetes Parity-Bit wird mit übertragen**
  - **Empfänger berechnet seinerseits das Parity-Bit und vergleicht es mit dem empfangenen: bei Ungleichheit Fehler erkannt.**

4.4.3 Erweiterter ASCII-Code (PC8)

- auf 8-Bit erweiterter ASCII-Code
- Zeichen 0...127 entsprechen dem 7-Bit ASCII-Code
- das 8. Bit wird in die Codierung einbezogen, um die Anzahl der codierbaren Zeichen auf 2^8=256 zu erhöhen.
- insbesondere genutzt für länderspezifische Sonderzeichen und semigraphische Symbole
- gebräuchlicher Zeichensatz auf PCs

Code-Tabelle PC8

Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01	☺	SOH
^B	2	02	☻	STX
^C	3	03	♥	ETX
^D	4	04	♦	EOT
^E	5	05	♣	ENQ
^F	6	06	♠	ACK
^G	7	07	•	BEL
^H	8	08	◘	BS
^I	9	09	○	HT
^J	10	0A	◙	LF
^K	11	0B	♂	VT
^L	12	0C	♀	FF
^M	13	0D	♪	CR
^N	14	0E	♫	SO
^O	15	0F	☼	SI
^P	16	10	►	DLE
^Q	17	11	◄	DC1
^R	18	12	↕	DC2
^S	19	13	‼	DC3
^T	20	14	¶	DC4
^U	21	15	§	NAK
^V	22	16	▬	SYN
^W	23	17	↨	ETB
^X	24	18	↑	CAN
^Y	25	19	↓	EM
^Z	26	1A	→	SUB
^[	27	1B	←	ESC
^\	28	1C	∟	FS
^]	29	1D	↔	GS
^^	30	1E	▲	RS
^_	31	1F	▼	US

Dec	Hex	Char
32	20	sp
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂†

Dec	Hex	Char
128	80	Ç
129	81	ü
130	82	é
131	83	â
132	84	ä
133	85	à
134	86	å
135	87	ç
136	88	ê
137	89	ë
138	8A	è
139	8B	ï
140	8C	î
141	8D	ì
142	8E	Ä
143	8F	Å
144	90	É
145	91	æ
146	92	Æ
147	93	ô
148	94	ö
149	95	ò
150	96	û
151	97	ù
152	98	ÿ
153	99	Ö
154	9A	Ü
155	9B	¢
156	9C	£
157	9D	¥
158	9E	₧
159	9F	ƒ

Dec	Hex	Char
160	A0	á
161	A1	í
162	A2	ó
163	A3	ú
164	A4	ñ
165	A5	Ñ
166	A6	ª
167	A7	º
168	A8	¿
169	A9	⌐
170	AA	¬
171	AB	½
172	AC	¼
173	AD	¡
174	AE	«
175	AF	»
176	B0	▒
177	B1	▒
178	B2	▒
179	B3	│
180	B4	┤
181	B5	╡
182	B6	╢
183	B7	╖
184	B8	╕
185	B9	╣
186	BA	║
187	BB	╗
188	BC	╝
189	BD	╜
190	BE	╛
191	BF	┐

Dec	Hex	Char
192	C0	└
193	C1	┴
194	C2	┬
195	C3	├
196	C4	─
197	C5	┼
198	C6	╞
199	C7	╟
200	C8	╚
201	C9	╔
202	CA	╩
203	CB	╦
204	CC	╠
205	CD	═
206	CE	╬
207	CF	╧
208	D0	╨
209	D1	╤
210	D2	╥
211	D3	╙
212	D4	╘
213	D5	╒
214	D6	╓
215	D7	╫
216	D8	╪
217	D9	┘
218	DA	┌
219	DB	▒
220	DC	▒
221	DD	▒
222	DE	▒
223	DF	▒

Dec	Hex	Char
224	E0	α
225	E1	ß
226	E2	Γ
227	E3	π
228	E4	Σ
229	E5	σ
230	E6	µ
231	E7	τ
232	E8	Φ
233	E9	Θ
234	EA	Ω
235	EB	δ
236	EC	∞
237	ED	φ
238	EE	ε
239	EF	∩
240	F0	≡
241	F1	±
242	F2	≥
243	F3	≤
244	F4	⌠
245	F5	⌡
246	F6	÷
247	F7	≈
248	F8	°
249	F9	∙
250	FA	·
251	FB	√
252	FC	ⁿ
253	FD	²
254	FE	■
255	FF	

† ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS).
The DEL code can be generated by the CTRL+BKSP key.

4.4.4 EBCDIC

- EBCDIC: Extended Binary Coded Decimal Interchange Code
- 8-Bit Code
- Als Erweiterung des BCD-Codes zur Anwendung in IBM System/360-Rechnern von IBM entwickelt und von anderen Herstellern übernommen
- heute noch in Großrechnern (Mainframes) angewendet

4.4.5 Unicode

- **Der Unicode-Zeichensatz ist ein relativ neues, international standardisiertes Code-System, das alle Schriftzeichen der verbreiteten internationalen Schriften sowie historischer Schriftarten enthält, insbesondere auch Chinesich/Japanisch/Koreanisch (CJK).**
- **Standardisierung durch Unicode-Konsortium (http://www.unicode.org)**
- **Unicode besitzt hohe Bedeutung für die Internationalisierung von Programmen.**
- **Unicode ist konform zur internationalen Norm ISO/IEC 10646 (Universal Character Set, UCS).**
- **Unicode definiert *kein* äußeres Erscheinungsbild (Glyph) für Zeichen, wie dies Fonts tun (z.B. Arial, Helvetica).**

Unicode-Zeichen

- **Jedes Zeichen besitzt eine 16-Bit-Zeichennummer.**
- **$\Rightarrow$ 65536 Zeichen (in Version 2.0 des Unicode-Standards sind 38885 Zeichen dokumentiert)**
- **Erweiterungsmechanismus durch Zeichenpaare für mehr als eine Million Zeichen**
- **Jedes Zeichen wird eindeutig über seine Nummer oder seine (ebenfalls standardisierte) textuelle Beschreibung identifiziert**
- **gebräuchlichste Schreibweise:**
  U+*xxxx*, wobei *xxxx* eine vierstellige hexadezimale Zahl ist.
- **Beispiele:**
  - **U+0041 "LATIN CAPITAL LETTER A"**
  - **U+20AC "EURO SIGN" (Euro-Währungszeichen)**

Zeichenbereiche

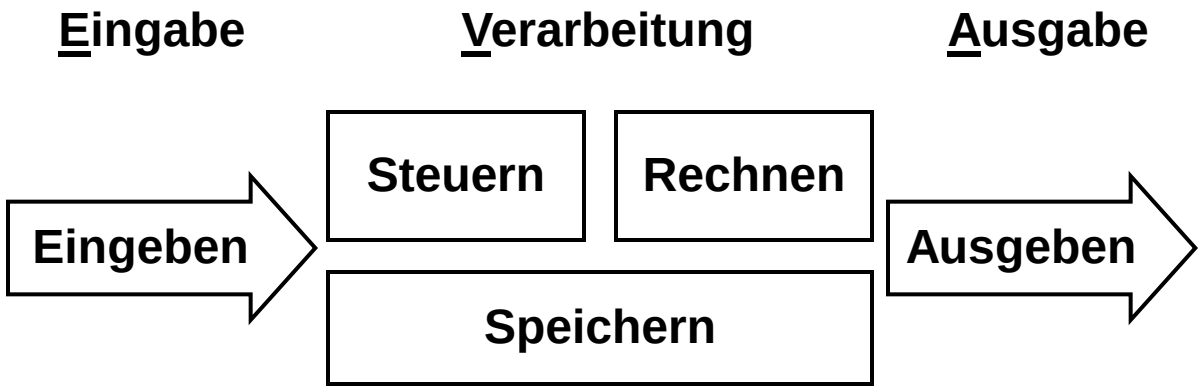
- **Das gesamte Unicode-System ist in Zeichenbereiche (*Scripts*) aufgeteilt. Die Zeichenbereiche spiegeln jeweils eine bestimmte Schriftkultur oder einen Satz von Sonderzeichen wider.**
- **Der Bereich U+0000 bis U+007F "C0 Controls and Basic Latin" entspricht genau dem ASCII-Standard.**
- **Weitere Beispiele:**
  - **U+0080 bis U+00FF enthält "C1 Controls and Latin-1 Supplement"**
  - **U+0370 bis U+03FF Griechisch**
- **Zeichenbereiche sind unterschiedlich groß**
  - **ASCII: 128 Zeichen**
  - **CJK Block enthält Tausende von Zeichen**

Zeichenrepräsentierung

- **Der Unicode Standard sieht verschiedene Codierungen des Zeichensatzes vor, entsprechend den ISO 10646 Transformationsformaten UTF-8 und UTF-16, die ohne Informationsverlust ineinander überführt werden können.**
- **UTF-16:**
  - **Kanonische Speicherung jedes Unicode-Zeichens in 2 Bytes mit einem Inhalt entsprechend der Zeichennummer**
  - **Beispiel: U+0041 $\Rightarrow$ 0x0041**
- **UTF-8:**
  - **Unicode-Zeichen werden verschieden lang in 1, 2 oder 3 Bytes kodiert.**
  - **erlaubt kompaktere Darstellungen (vgl. Kap. 4).**
  - **die Codierung für den Zeichenbereich U+0000 bis U+007F "C0 Controls and Basic Latin" entspricht genau dem ASCII-Code**

4.5 Ein- / Ausgabe

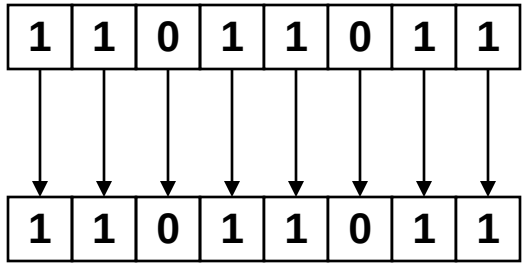
- **Modell eines Rechensystems (vgl. Kap. 1)**



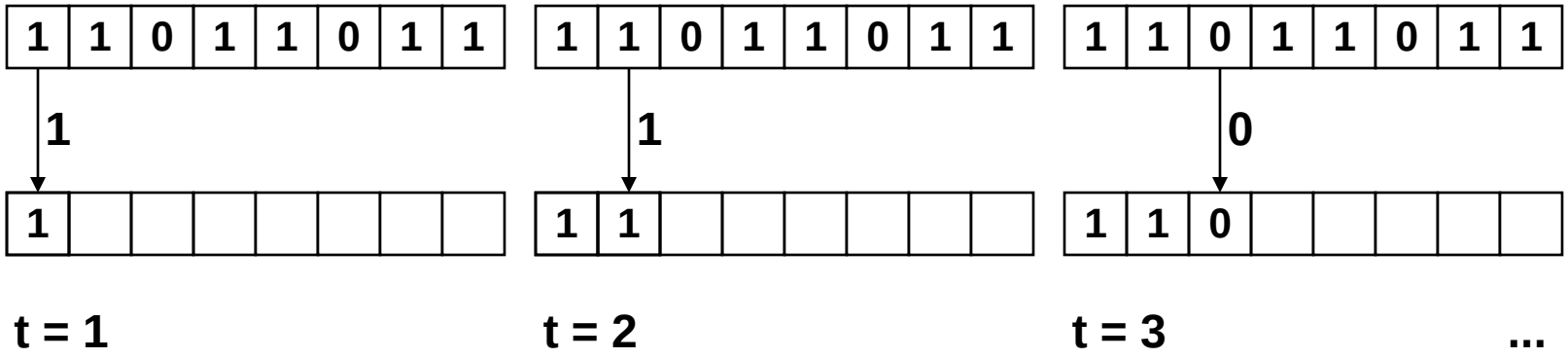
- **Die Verarbeitung geschieht auf Basis digitaler Informationen, Ein- und Ausgabe von/in die Umgebung des Rechensystems kann auf digitalen Signalen oder analogen Signalen basieren.**

Parallele und sequentielle Übertragung

- **Die Übertragung einer Signalmenge zwischen den Komponenten eines Rechensystems bzw. von/zur Umgebung des Rechensystems kann *parallel* im Raum oder *sequentiell* in der Zeit geschehen.**
- **parallele Übertragung:**



- **sequentielle Übertragung:**



Analoge und digitale Signale

Def

- **Der zeitlich veränderliche Verlauf einer physikalischen Größe heißt *Signal.***

  Beispiele: Spannungssignal, akustisches Signal, optisches Signal.

- **Ein Signal heißt *analog* oder *kontinuierlich*, wenn es kontinuierliche Werte entsprechend einem Intervall aus der Menge der reellen Zahlen annehmen kann (unendlich viele Werte).**

  Beispiele: Spannungssignale als Sensorwerte von physikalischen Größen wie Temperatur, Lautstärke, Helligkeit, Winkel, ...

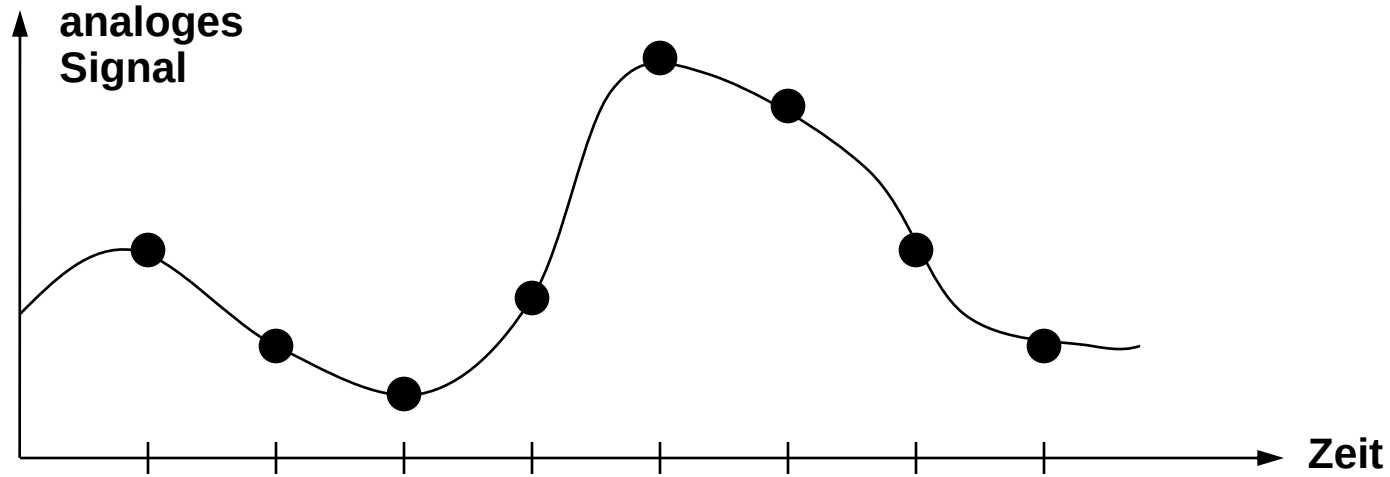
- **Ein Signal heißt *digital*, wenn es nur endlich viele Werte annehmen kann.**

  Beispiel: Schalter (offen oder geschlossen)

Rasterung/Abtastung

Def

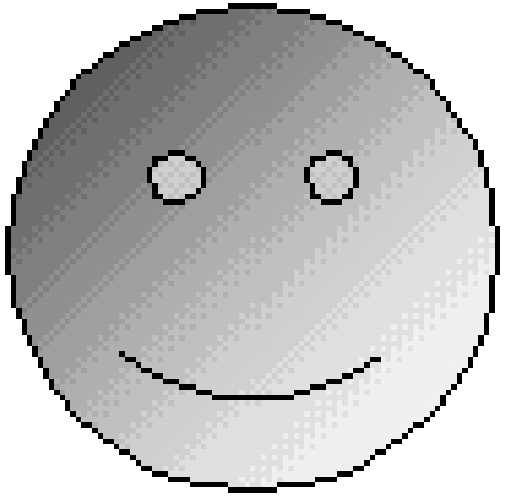
- ***Rasterung*** **oder** ***Abtastung*****: der Wert eines (analogen oder digitalen) Signals wird nur zu einzelnen Zeitpunkten (z.B. mit festem Intervall) oder einzelnen Ortspunkten bestimmt. Die Größe wird dadurch** ***diskretisiert*****. Der ermittelte Wert kann z.B. einem Funktionswert im betrachteten Intervall oder einem Mittelwert entsprechen.**



- **Anmerkung: Die Bedingungen, die erfüllt sein müssen, damit die Abtastung zu keinem Informationsverlust führt, werden durch das sogenannte** ***Abtasttheorem*** **von Shannon definiert (vgl. Vorlesung Informationstheorie).**

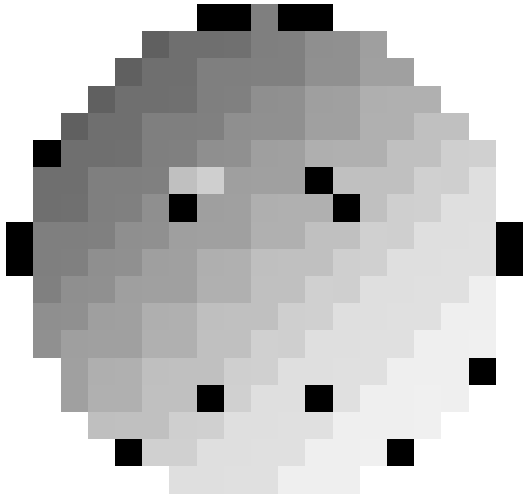
Rasterung/Abtastung (2)

- **Beispiel: 2-dimensionale Rasterung (Ortspunkte)**



Original

in Wirklichkeit auch schon (feiner) gerastert

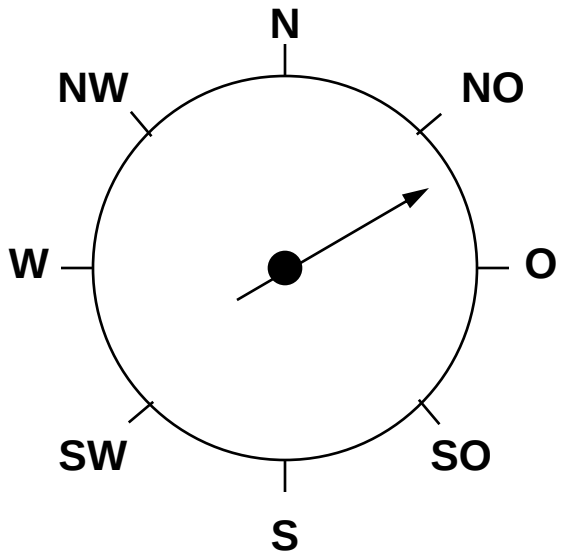


Rasterung

Quantelung

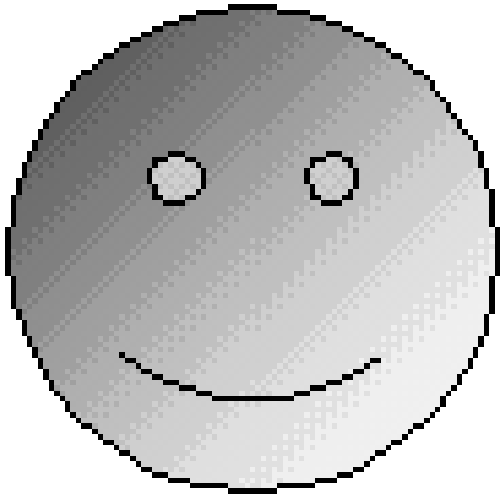
Def

- **Unter *Quantelung* versteht man die Abbildung eines analogen Signals in ein digitales Signal (Diskretisierung des Werts).**
- **Beispiel 1:**
  - **Windrichtung: Winkelbereich zwischen 0 und 360 Grad wird in 8 gleiche Abschnitte unterteilt, die mit {N, NO, O, SO, S, SW, W, NW} bezeichnet werden. Es wird der nächstgelegene Wert zugeordnet.**



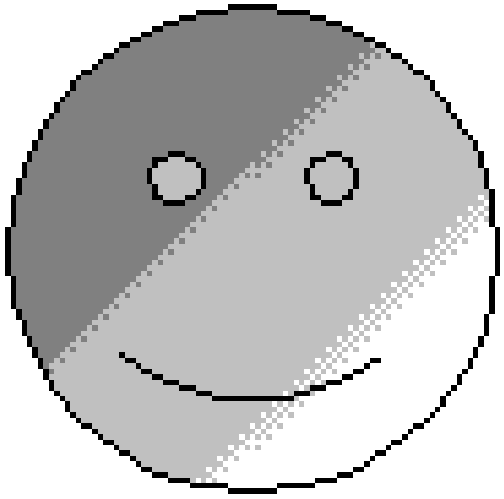
Quantelung (2)

- **Beispiel 2: 2-dimensionale Quantelung (Ortspunkte)**



Original

in Wirklichkeit auch schon gequantelt



Quantelung

auf 4 Grauwerte

Quantelung (3)

- **Die technische Realisierung einer Quantelung geschieht häufig durch sogenannte *Analog/Digital-Wandler*, die einen vorgegebenen Spannungsbereich in n unterschiedliche Werte abbilden (z.B. n=1024).**
- **Die Kombination aus Abtastung und Quantelung wird auch als *Pulse-Code-Modulation (PCM)* bezeichnet.**

  Beispiel:
  Audio-CD: Abtastung 44.100 Mal/sec (44.1 kHz), Wertebereich 2^{16}

4.1 Einführung und Überblick

- **Wiederholung (Kap. 2):**
  Information:
  - **abstrakter Bedeutungsgehalt (Semantik)**
  - **äußere Form von Information: Repräsentation oder Darstellung.**
- **Die Hardware eines Rechensystems ist auf wenige, genau festgelegte Informationsrepräsentierungen zugeschnitten und gestattet deren Manipulation durch Maschinenbefehle.**

Quellen

- U. Rembold, P. Levi: "Einführung in die Informatik für Naturwissenschaftler und Ingenieure", 3. Auflage, Hanser-Verlag, 1999 (Kap. 2.1.)
- D. Werner u.a.: "Taschenbuch der Informatik", Fachbuchverlag Leipzig, 1995 (Kap. 4.1)
- F. Mayer-Lindenberg: "Konstruktion digitaler Systeme", Vieweg-Verlag, 1998 (Kap. 2)
- H. Dispert, H.-G. Heuck: "Einführung in die Technische Informatik und Digitaltechnik", Vorlesungsskript FH Kiel (Kap. 8), http://www.e-technik.fh-kiel.de/universe/digital/dig0_00.htm
- The Unicode Consortium: "The Unicode Standard", Addison-Wesley, 1992
- http://www.unicode.org