GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Informatik II

SS 2005

Kapitel 2: Zahlen und Logik

Teil 1: Zahlen

Dr. Michael Ebner
Dipl.-Inf. René Soltwisch

Lehrstuhl für Telematik
Institut für Informatik

Überblick

- Zahlen
  - **Informationsdarstellung**
  - Zahlensysteme
  - Rechnerarithmetik
- Logik
  - Aussagenlogik und logische Gatter
  - Prädikatenlogik

Nachricht, Daten, Information (1)

- Nachricht
  - „Zeichen oder kontinuierliche Funktion, die zum Zwecke der **Weitergabe** Information aufgrund bekannter oder unterstellter Abmachungen darstellen.“ (DIN 44300)
- Daten
  - „Zeichen oder kontinuierliche Funktion, die zum Zwecke der **Verarbeitung** Information aufgrund bekannter oder unterstellter Abmachungen darstellen.“ (DIN 44300)
- Information
  - „**Inhalt** von Nachrichten oder Daten“

- **Nachricht** — Codierung Information — **Weitergabe**
- **Daten** — Codierung Information — **Verarbeitung**
- **Information** — **Bedeutung/Inhalt**

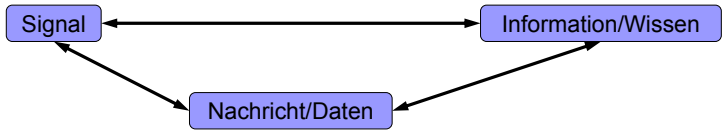
Nachricht, Daten, Information (2)

- Gleiche Informationen sind in verschiedenen Codierungen möglich
- Für verschiedene Empfänger hat die gleiche Information oft unterschiedliche Bedeutung
- Information muss für den Empfänger nicht neu sein, evtl. ist der Informationsgehalt auch leer

	Information	Codierung	Interpretation	Zweck
Nachricht	‘5 Grad’	menschliche Sprache	Duden	Weitergabe der Information in einem Gespräch
Datum	‘5 Grad’	1011’0101 0010’0000 1100’0111 1111’0010 0110’0001 0110’0100	ASCII	Weiterverarbeitung in einem Textverarbeitungsprogramm

Informationsdarstellung

- Zur Übertragung von Nachrichten/Daten müssen diese in Signale umgesetzt werden.
- **Signal**: physikalische Darstellung von Nachrichten/Daten
  - Signaltypen: Radiowellen, Rauchsignale, Stromfluss, Schallwellen, usw.
- **Signalparameter**: Ausprägung eines Signals mit dessen Hilfe Nachrichten/Daten dargestellt werden.
  - Parameterarten**:** Frequenz, Farbe, Form, Spannung, Lautstärke, usw.

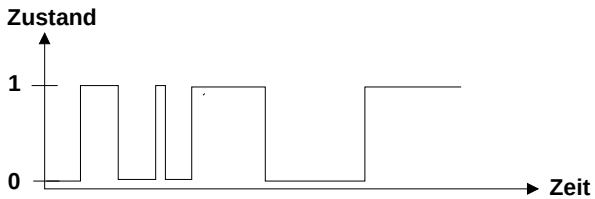


Analoge Daten & Digitale Daten

- **Analoge Daten**
  - Physikalische Prozesse werden im Allgemeinen durch kontinuierliche Funktionen beschrieben. Diese liefern im Prinzip unendlich viele Werte, wobei sich diese aufgrund der Unschärferelation nicht unbedingt unterscheiden lassen.
  - Wo begegnen uns analoge Daten?
    - Quecksilberthermometer, Uhren, Schallplatten, Audio-Kassetten, Waagen
- **Digitale Daten**
  - Quantisierung, d.h. der eventuell unendliche Wertebereich wird auf endlich viele Werte abgebildet
  - Wo begegnen uns digitale Daten?
    - DVD, CD, Computerdisketten, digitale Uhren, digitale Thermometer, MP3

Vorteile von digitalen Daten

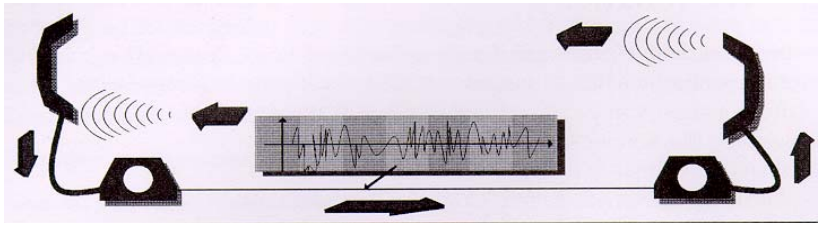
- Auswahl diskreter Werte zur digitalen Interpretation erleichtert die rechnergestützte Verarbeitung und kann auch die Genauigkeit erhöhen
- Daten werden in Rechnern praktisch ausschließlich in digitaler Form dargestellt/verarbeitet. Meist in Form von „binären“ Signalen
- **Binäre Signale:** Signale, die nur zwei Zustände annehmen können.



Wie kommt man in einer ‘analogen‘ Welt zu digitalen Daten?

- Beispiel: Digitalisierung von Sprache
- Analoge Telephonie basiert auf der “Ähnlichkeit” von elektronischen Wellen und Schallwellen

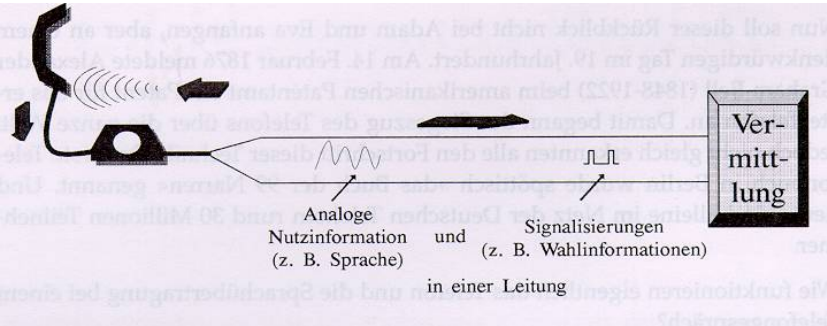
Sprachübertragung in der analogen Telephonie (‘analog‘ ≅ ‘ähnlich‘)



- Anmerkung: Prinzip gilt auch für die Digitalisierung von Musik (z.B. MP3), erfordert aber eigene Annahmen.

Sprach- und Signalisierungsübertragung

- Zusätzlich zur Nutzinformation werden Signalinformationen übertragen.



Digitalisierung: Prinzip (1/3)

- **„digital“ ≅ „ziffernmäßig“**
- Beispiel:
  - Menge von Kartoffeln in einem Sack lässt sich durch zählen oder wiegen beschreiben.
    - 100 kg lässt sich durch die Folge 1-0-0 im Dezimalsystem beschreiben
    - Digitaltechnik: binärer Zeichenvorrat, d.h. 0 und 1

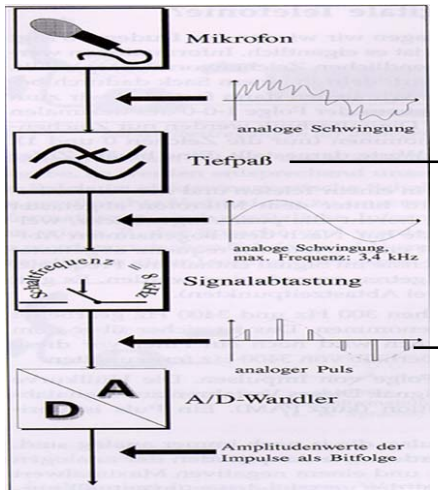
      $100_{10} \cong 1100100_2$

- Ähnlich lassen sich elektrische Wellen in einem Telefon und der Telefonleitung beschreiben.
- Fazit: *Elektrische Welle wird in regelmäßigen Abständen “abgehorcht” und der dann aktuelle Wert wird notiert*

Digitalisierung (2/3)

- **Abtasttheorem**
  - Ein Signal muss mindestens mit einer (Abtast-) Frequenz abgetastet werden, die doppelt so hoch ist wie die höchste im Signal enthaltene Frequenz. Andernfalls kann das Signal nicht originalgetreu reproduziert werden.
  - Abtastfrequenz = 1/(Abstand zwischen zwei Abtastpunkten)
- **Telephonie:**
  - Frequenzen (Töne): 300 Hz bis 3400 Hz
  - Abtastfrequenz: 8000 Hz
  - Tiefpass: Filter für Frequenzen > 3400 Hz
  - Durch Abtastung entsteht eine Folge von Impulsen (Werten). Hüllkurve dieser Abtastwerte ergibt wieder das alte Signal.
  - *Verfahren*: Pulsamplitudenmodulation (PAM)

Digitalisierung (3/3)



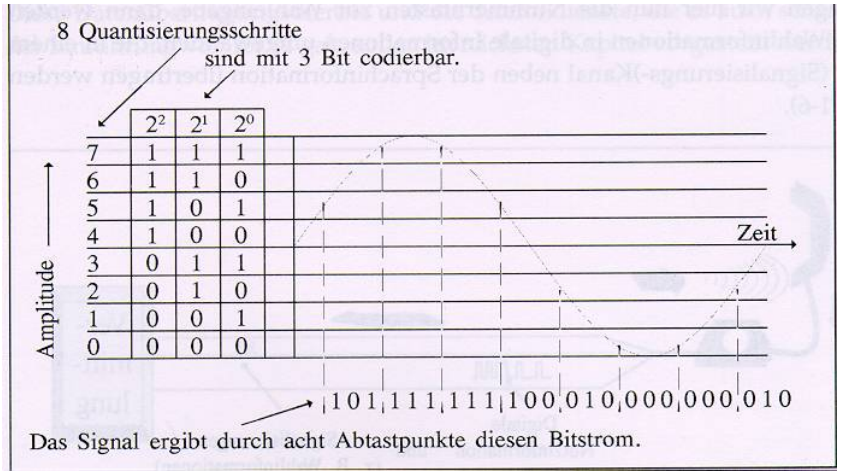
•Filter für Frequenzen >3400Hz

•(---------- Hüllkurve angedeutet)

Quantisierung (1/2)

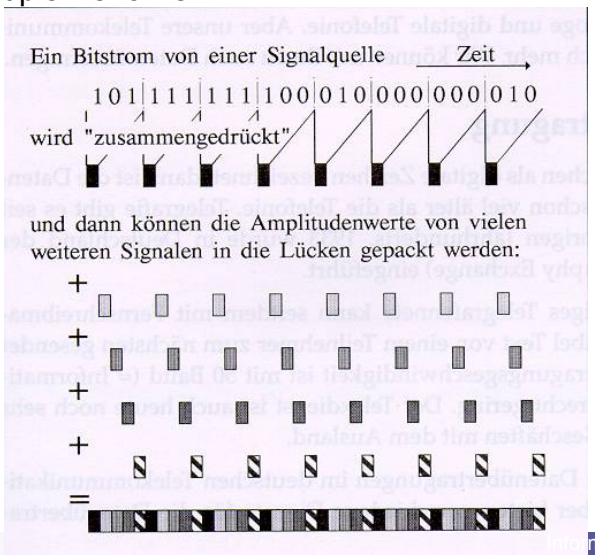
- Impulse sind ja immer noch analog, d.h. die gemessenen Werte sind für die digitale Repräsentation u.U. noch zu genau. Sie müssen zu digitalen Signalen verarbeitet werden. (⇒ Einschränkung auf eine darstellbare endliche Wertemenge)
  - Maximal mögliche Amplitudenwert wird in eine endliche Anzahl von Amplitudenstufen unterteilt (=quantisiert)
  - Je mehr Stufen (Quantisierungsschritte) benutzt werden, desto genauer wird das Originalsignal abgetastet und desto originalgetreuer kann es wieder gewonnen werden.
- **Telekom-Netzbetreiber** verwenden $2^8 = 256$ Quantisierungsschritte (mit 8Bit darstellbar)
  - **Bitrate eines Fernsprechkanals?**
    - Abtastfrequenz: 8000Hz = 8000/s
    - 8Bit * 8000/s = 64000 Bit/s = 64 kBit/s

Quantisierung (2/2)



	2^2	2^1	2^0
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
0	0	0	0

Zeitmultiplexverfahren



Datenaufbewahrung

- Physikalische Prozesse mit deren Hilfe sich Signale und Informationen aufbewahren und wiederherstellen lassen
- Je nach Anwendung kann man zwischen RAM und ROM unterscheiden:

	Speichermedium	Wert 0	Wert 1
ROM	Webstuhl-Brettchen	Bohrung	Keine Bohrung
	Lochstreifen/Lochkarte	Lochung	Keine Lochung
	Kippschalter	Kontakt zu	Kontakt offen
RAM	Magnetschicht	↑H	↓H
	Transistor	0,5 V	4,5 V

Codierung - Grundlagen

- **Alphabet (Zeichenvorrat):**
  - Endliche Menge von Zeichen, die zur Darstellung von Informationen benutzt wird
  - **Beispiele:**
    - Ziffern: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
    - Alphanumerisches Alphabet: {a,...,z, A,...,Z, 0,...,9}
    - Binärzeichen: {0, 1}
- **Wort:**
  - Folge von Zeichen, die in einem bestimmten Kontext als Einheit betrachtet werden
  - **Beispiele:**
    - Zahlwörter: 105, 75, 73, 15, ...
    - Schreibwörter: Kohlkopf, Hunger, Hund
    - Binärwörter: 001, 10010010, 1, 0, ...
- Bemerkungen zum Kontext:
  - in Englisch hat das Wort 'Kohlkopf' keine Bedeutung
  - '010' kann von Rechner mit Wortlänge 8 nicht verarbeitet werden

Codierung - Codes (1/2)

- **Code:**
  - zwei Zeichenvorräte: **Urbildmenge** und **Bildmenge**
  - eindeutige Abbildung von der Urbildmenge auf die Bildmenge
- **Binärcode:**
  - Code, bei dem jedes Zeichen der Bildmenge ein Wort aus Binärzeichen (Binärcode) ist.

Codierung - Codes (2/2)

- **Beispiel:** Codierung der Zahlen 0 ... 9 als Binärwörter
  - **Abkürzungen**
    BCD: **B**inary **C**oded **D**ecimal
    EBCDIC: **E**xtended **B**inary **D**ecimal **I**nterchange **C**ode
    ASCII: **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange

Dezimalzahl	BCD (Dualzahl)	EBCDIC	ASCII (P: Paritätsbit)
0	0000	1111'0000	P011'0000
1	0001	1111'0001	P011'0001
2	0010	1111'0010	P011'0010
3	0011	1111'0011	P011'0011
4	0100	1111'0100	P011'0100
5	0101	1111'0101	P011'0101
6	0110	1111'0110	P011'0110
7	0111	1111'0111	P011'0111
8	1000	1111'1000	P011'1000
9	1001	1111'1001	P011'1001
+	1010		P010'1011
-	1011		P010'1101

Alphanumerische Codes

- Sind zur gemeinsamen Darstellung von Zahlen und Buchstaben in Texten notwendig.
- Alphabet:

26	Großbuchstaben	(A ... Z)
26	Kleinbuchstaben	(a ... z)
10	Zahlen	(0 ... 9)
62	**Zeichen**	

- 62 Zeichen lassen sich in einem Binärwort der Länge 6 codieren (26 = 64).
- Was ist mit nationalen Sonderzeichen und Satzzeichen?
  Zum Beispiel: Ä Ü Ö , ; : - + ( )
  - Man benutzt daher einen **7 Bit Code** mit dem sich 128 Zeichen (27 Zeichen) codieren lassen.

Codierung - ASCII Code (1/2)

- **ASCII: A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
- **8 - Bit Code:**
  - 7 Bit zur Codierung von Zeichen
  - 1 Bit Paritätsbit

Bits 76543210	P000	P001	P010	P011	P100	P101	P110	P111
0000	NULL	DC_0		0	@	P	‘	p
0001	SOM	DC_1	!	1	A	Q	a	q
0010	EOA	DC_2	„	2	B	R	b	r
0011	EOM	DC_3	#	3	C	S	c	s
0100	EOT	DC_4	$	4	D	T	d	t
0101	WRU	ERR	%	5	E	U	e	u
0110	RU	SYNC	&	6	F	V	f	v
0111	BELL	LEM	‘	7	G	W	g	w
1000	FE	S_0	(	8	H	X	h	x
1001	HT/SK	S_1	)	9	I	Y	i	y
1010	LF	S_2	*	:	J	Z	j	z
1011	V/TAB	S_3	+	;	K	□	k	
1100	FF	S_4	,	<	L	\	l	ACK
1101	CR	S_5	-	=	M	□	m	UC
1110	SO	S_6	.	>	N	↑	n	ESC
1111	SI	S_7	/	?	O	←	o	DEL

Codierung - ASCII Code (2/2)

- Die meisten Computer arbeiten jedoch mit einer 8 Bit Codierung ohne Paritätsbit, d.h. anstelle von 2^7 (=128) Zeichen werden 2^8 (= 256) Zeichen kodiert. Man spricht dann von *“Extended Character Sets“*.
- Extended Character Sets werden z.B. benutzt um Linienzeichnungen (z.B. ┐ oder ╲) und andere wichtige Zeichen zu kodieren (z.B. ☺ oder ✂).
- Natürlich auch zur Berücksichtigung von nationalen Varianten (z.B. ß, ü, Ü, ö, Ö, ä, Ä, û, ú, ù, Æ, æ, è, é, Ó).
- Typischerweise reichen 256 Zeichen jedoch nur aus um eine nationale Variante zu unterstützen, wodurch es dann wieder viele verschiedene nationale Extended Character Sets gibt.
- Ein Ausweg aus diesem Chaos soll der **Unicode** sein, einer 16 Bit Kodierung von Zeichen, bei der die ersten 128 Kodierungen der normalen 7 Bit ASCII entsprechen.

Der Unicode-Standard (1/4)

- **Allgemeines**
  - Universeller Standard zur Zeichencodierung
  - Voll Identisch mit ISO/IEC 10646:2003: Information Technology Universal Multiple- Octet Coded Character Set (UCS)
  - Ziele:
    - Codierung multilingualer Texte
    - Einfacher Austausch von Textdateien über nationale Grenzen hinweg
    - Einbeziehung mathematischer und technischer Symbole
  - Unicode ist Standardcodierung bei Java
  - Unterstützung in C++ durch Datentyp *wchar_t und* Klassenbibliotheken wie QT

Der Unicode-Standard (2/4)

- **Zeichenumfang**
  - Jedes Zeichen ist eindeutig einem 16-Bit-Codewert zugewiesen → ≈ 65.500 Zeichen; ausreichend für die Codierung aller Zeichen aller geschriebenen Sprachen
  - Codierung (v.4.0)

Alphabetics, Symbols	11 649
CJK Ideographs	27 786???
Hangul Syllables	11 172
Private Use	6 400
Surrogates	2 048
Controls	65
Not Characters	34
Total assigned 16-bit code values	59 213
Unassigned 16-bit code values	6 3236

  - Unicode entspricht in den unteren 128 Werten dem ASCII-Code

Der Unicode-Standard (3/4)

- **Zeichencodierung**
  - Unterstützte Sprachen: Latein, Griechisch, Chinesisch/Japanisch/Koreanisch, Kyrillisch, Arabisch, Bengalisch, Gujarati,...
  - Zusammengesetzte Zeichen (z. B. Umlaute) können wahlweise dargestellt werden:
  - durch „vorkomponierte“ Zeichen (aus Kompatibilität), z. B. *ä*, oder
  - durch Kombination von Grundzeichen und Diakritika (vorteilhaft bei Sortierung), z. B. a“
  - Änderung der Schreibrichtung kann durch spezielle Zeichen eingeleitet werden, z. B. beim Wechsel zwischen Englisch und Arabisch
  - Unicode legt nicht das Aussehen (glyph) von Zeichen fest; statt dessen Zuweisung eines Namens, z. B. „LATIN CAPITAL LETTER A“

Der Unicode-Standard (4/4)

- **Quellen**
  - Unicode Consortium (http://www.unicode.org)
  - The Unicode Consortium. The Unicode Standard, Version 4.1.0, defined by: *The Unicode Standard, Version 4.0* (Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1), as amended by *Unicode 4.0.1* (http://www.unicode.org/versions/Unicode4.0.1) and by *Unicode 4.1.0* (http://www.unicode.org/versions/Unicode4.1.0).

Codierung - Schichtenmodell

- Schichtenmodell als **Gedankenmodell**

C3
C2
C1
C0

Codierungen sind häufig geschachtelt. D.h. eine höhere Schicht stützt sich auf die nächst tiefere ab. Die höhere Schicht nimmt Zeichen entgegen, ignoriert aber deren tiefere Codierung.

- Beispiel**:**

C++ – Schlüsselwörter
Großbuchstaben ISO
ISO 7-Bit
8-Bit Bytes
Binärstellen, Bits
Schaltzustände
Elektronenebene

Rückblick: Information und Informationsdarstellung

- Signal, Daten, Nachricht, Information
- Analoge Daten & digitale Daten
- Codierungen
  - Zahlencodes
  - alphanumerische Codes
- Codes
  - BCD Code
  - EBCDIC Code
  - ASCII Code
  - Unicode
- Schichtung von Codierungen

Überblick

- Zahlen
  - Informationsdarstellung
  - **Zahlensysteme**
  - Rechnerarithmetik
- Logik
  - Aussagenlogik und logische Gatter
  - Prädikatenlogik

Polyadische Zahlensysteme (1/2)

- Potenzen zu einer Basis B als Stellenwert

$$n = \sum_{i=0}^{N-1} a_i * B^i \qquad (B, a_i \in N_0, B > 1)$$

$$= a_{N-1} * B^{N-1} + a_{N-2} * B^{N-2} + ... + a_1 * B^1 + a_0$$

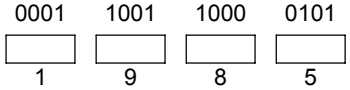
$$= ((...(a_{N-1} * B) + a_{N-2}) * B + ...)*B + a_1) *B + a_0$$

(Horner Schema)

- **Konvention:**
  $<ZAHL>_{<BASIS>}$ besagt das <ZAHL> einen Wert im Zahlensystem mit der Basis <Basis> beschreibt.
  z.B. 123_{10} 123 im Dezimalsystem,
  1110_2 1110 im Dualsystem

Polyadische Zahlensysteme (2/2)

- z.B. Codierung von 1985_{10}
  - BCD (kein polyadisches Zahlensystem):



  - Dezimalsystem (Basis 10):
    $1985_{10} = \mathbf{1} * 10^3 + \mathbf{9} * 10^2 + \mathbf{8} * 10^1 + \mathbf{5} * 10^0$
    $= ((\mathbf{1} * 10) + \mathbf{9}) * 10 + \mathbf{8}) * 10 + \mathbf{5}$
  - Dualsystem (Basis 2):
    11111000001_2
    $= \mathbf{1}*2^{10} + \mathbf{1}*2^9 + \mathbf{1}*2^8 + \mathbf{1}*2^7 + \mathbf{1}*2^6 + \mathbf{0}*2^5 + \mathbf{0}*2^4 + \mathbf{0}*2^3 + \mathbf{0}*2^2 + \mathbf{0}*2^1 + \mathbf{0}*2^0$
    $=(((((((((\mathbf{1}*2 + \mathbf{1})*2 + \mathbf{1})*2 + \mathbf{1})*2 + \mathbf{1})*2 + \mathbf{0})*2 + \mathbf{0})*2 + \mathbf{0})*2 + \mathbf{0})*2 + \mathbf{0})*2 + \mathbf{1}$
    $=1985_{10}$

Zahlensysteme mit Zweierpotenz als Basis

- Binärsystem (Basis 2)
- Vierersystem (Basis 4)
- Oktalsystem (Basis 8)
- Hexadezimalsystem (Basis 16) (Zeichenvorrat: 0...9, A...F)
- **Einfache Umrechnung:** „Umgruppieren der Binärstellen“
- **Beispiel:**
  $1463_{10} = 10110110111_2$

=	01	01	10	11	01	11	
	1	1	2	3	1	3	$= 112313_4$
=	010	110	110	111			
	2	6	6	7			$=2667_8$
=	0101	1011	0111				
	5	B	7				$=5B7_{16}$

Umrechnung zwischen Zahlensystemen

- Im allgemeinen reicht es nicht aus einfach nur die Binärstellen umzugruppieren.
- **Empfehlung für die manuelle Umrechnung:**

I. Darstellung mit Basis B1

(1) ↓ ***** — Multiplikative Umrechnung im Zielsystem (hier Basis 10)

II. Darstellung mit Basis 10

(2) ↓ **/** — Dividierende Umrechnung im Quellsystem (hier Basis 10)

III. Darstellung mit Basis B2

Beispiel für manuelle Umrechnung (1/3)

- **$DB7_{16} = ???_7$**
- **(1) Multiplikative Umrechnung mit Basis 10**
  (Einfaches ‘Ausmultiplizieren’)

$$
\begin{aligned}
DB7_{16} &= D_{16} * 16_{10}^2 + B_{16} * 16_{10}^1 + 7_{16} * 16_{10}^0 \\
&= D_{16} * 256_{10} + B_{16} * 16_{10} + 7_{16} \\
&= 13_{10} * 256_{10} + 11_{10} * 16_{10} + 7_{10} \\
&= 3328_{10} + 176_{10} + 7_{10} \\
&= 3511_{10}
\end{aligned}
$$

oder entsprechend dem Horner Schema

$$
\begin{aligned}
DB7_{16} &= ((D_{16} * 16_{10}) + B_{16}) * 16_{10} + 7_{16} \\
&= ((13_{10} * 16_{10}) + 11_{10}) * 16_{10} + 7_{10} \\
&= (208_{10} + 11_{10}) * 16_{10} + 7_{10} \\
&= 3504_{10} + 7_{10} \\
&= 3511_{10}
\end{aligned}
$$

Beispiel für manuelle Umrechnung (2/3)

- **(2) Dividierende Umrechnung im Quellsystem**
  (Quellsystem: Basis 10, Zielsystem: Basis 7)

3511_{10}	/	7	=	501	Rest	4	=	a_0
501	/	7	=	71	Rest	4	=	a_1
71	/	7	=	10	Rest	1	=	a_2
10	/	7	=	1	Rest	3	=	a_3
1	/	7	=	0	Rest	1	=	a_4
0	/	7	=	0	Rest	0	=	a_5
...	/	7	=	0	Rest	0	=	a_6
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	0	=	a_n

$(a_4a_3a_2a_1a_0)_7 = 13144_7$

Beispiel für manuelle Umrechnung (3/3)

- **Also:**

$DB7_{16}$ = 3511_{10} = 13144_7

↑ Schritt (1) (Multiplikation) ↑ Schritt (2) (Division)

Empfehlung für manuelle Umrechnung zwischen Zahlensystemen (Zusammenfassung)

Beispiel: $DB7_{16} = ???_7$

I. Darstellung mit Basis B1

(1) ↓ * — Multiplikative Umrechnung im Zielsystem (hier Basis 10)

$DB7_{16} = ((D_{16} * 16) + B_{16}) * 16 + 7_{16}$
$= ((13 * 16) + 11)*16 +7$
$= (208 + 11) * 16 + 7$
$= 219 * 16 + 7$
$= 3504 + 7$
$= 3511_{10}$

II. Darstellung mit Basis 10

(2) ↓ / — Dividierende Umrechnung im Quellsystem (hier Basis 10)

3511	: 7 = 501	Rest 4 =	a_0
501	: 7 = 71	Rest 4 =	a_1
71	: 7 = 10	Rest 1 =	a_2
10	: 7 = 1 Rest	3 =	a_3
1	: 7 = 0 Rest	1 =	a_4

III. Darstellung mit Basis B2

$DB7_{16} = 3511_{10} = 13144_7$

Direkte Umrechnung einer Zahlendarstellung mit Basis B1 in eine Zahlendarstellung mit der Basis B2 (1/2)

- **Multiplikative Umrechnung im Zielsystem**

Beispiel: $DB7_{16} = ???_7$

Einfaches 'Ausmultiplizieren' ($16_{10} = 10_{16} = 22_7$, $16_{10}^2 = 256_{10} = 514_7$)

$DB7_{16} = D_{16} * 514_7 + B_{16} * 22_7 + 7_{16}$
$= 16_7 * 514_7 + 14_7 * 22_7 + 10_7$
$= 12463_7 + 341_7 + 10_7$
$= 13144_7$

oder entsprechend dem Horner Schema

$DB7_{16} = ((D_{16} * 22_7) + B_{16}) * 22_7 + 7_{16}$
$= ((16_7 * 22_7) + 14_7) * 22_7 + 10_7$
$= (415_7 + 14_7) * 22_7 + 10_7$
$= 13134_7 + 10_7$
$= 13144_7$

Direkte Umrechnung einer Zahlendarstellung mit Basis B1 in eine Zahlendarstellung mit der Basis B2 (2/2)

- **Dividierende Umrechnung im Quellsystem**

Beispiel: $DB7_{16} = ???_7$

$DB7_{16}$	/	7_{16}	=	$1F5_{16}$	Rest 4 = a_0
$1F5_{16}$	/	7_{16}	=	47_{16}	Rest 4 = a_1
47_{16}	/	7_{16}	=	A_{16}	Rest 1 = a_2
A_{16}	/	7_{16}	=	1_{16}	Rest 3 = a_3
1_{16}	/	7_{16}	=	0_{16}	Rest 1 = a_4

$a_4\ a_3\ a_2\ a_1\ a_0 = 13144_7$

Darstellung von Brüchen

- Brüche werden als negative Potenzen der Basis dargestellt.

$z_B = \sum_{i=-1}^{-N} a_i * B^i$ $\quad (B, a_i \in N_0, B > 1, a_i<B)$

Horner Schema:

$$z = \frac{1}{B}(\mathbf{a_{-1}} + \frac{1}{B}(\mathbf{a_{-2}} + \frac{1}{B}(\mathbf{a_{-3}} + \ldots + \frac{1}{B}(\mathbf{a_{-N+1}} + \frac{1}{B}\mathbf{a_{-N}}) \ldots)$$

Darstellung: $z_B = 0 , a_{-1} a_{-2} a_{-3} a_{-4} \ldots a_{-N}$

Wiederholte Multiplikation mit der Basis B bringt die Ziffern a_i vor das Komma:

$z_{B*B} = a_{-1} + \sum_{i=-2}^{-N} a_i * B^i$ $\quad$ (wobei immer gilt : $\sum_{i=-2}^{-N} a_i * B^i < 1$)

$$z = \mathbf{a_{-1}} + \frac{1}{B}(\mathbf{a_{-2}} + \frac{1}{B}(\mathbf{a_{-3}} + \ldots + \frac{1}{B}(\mathbf{a_{-N+1}} + \frac{1}{B}\mathbf{a_{-N}}) \ldots)$$

z.B.: 0.5310 * 1010 = 5,310

Handhabung von unechten Brüchen

- Vor dem Komma anfallende Ziffern entsprechen dem Rest bei der Division im Quellsystem (vgl. Folie 2.1-36).
- Für Umrechnung Aufspalten in
  - **ganze Zahl** und
  - **echter Bruch**.

Umrechnung im Quellsystem

- Beispiel: **$12{,}02_3 = ??{,}??_2$**

Ganzahliger Anteil: $12_3 = 101_2$ **(Bemerkung: $12_3 = 5_{10}$)**
Echter Bruch: $0{,}02_3 = 0{,}???_2$

- Dividierende Umrechnung im Quellsystem (vgl. Folie 2.1-36) bedeutet **sukzessive Division durch 1/B**

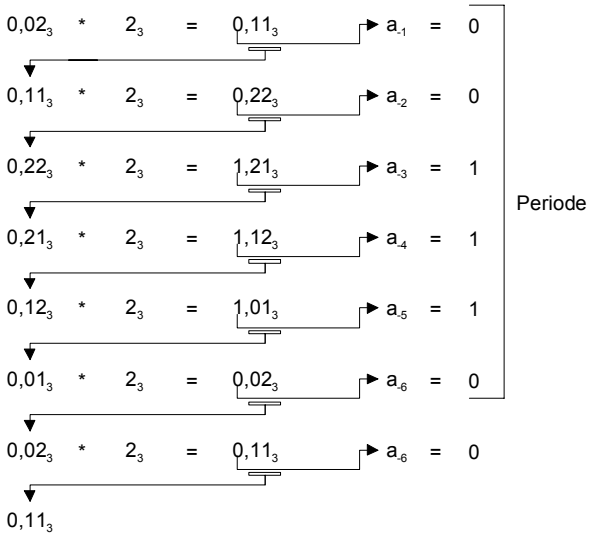
  (Division mit Bruch = Multiplikation mit Kehrwert des Bruches)

- Sukzessive Division durch 1/B entspricht einer **Sukzessiven Multiplikation mit der Basis B**.

Der ganzzahlige Anteil bei den Multiplikationsschritten bildet die a_i im Zielsystem.

Beispiel: Umrechnung im Quellsystem (1/2)

- **$0{,}02_3 = 0{,}???_2$**



d.h. $0{,}02_3 = 0{,}001110_2$

Beispiel: Umrechnung im Quellsystem (2/2)

- **Ergebnis:**

$$12{,}02_3 = 101{,}\overline{001110}_2$$

$$\downarrow \qquad\qquad \downarrow$$

$$5{,}\overline{2}_{10} \qquad \sim 5{,}22217_{10}$$

- Darstellung von Brüchen in unterschiedlichen Zahlensystemen kann zu Ungenauigkeiten führen.

Umrechnen im Zielsystem (1/2)

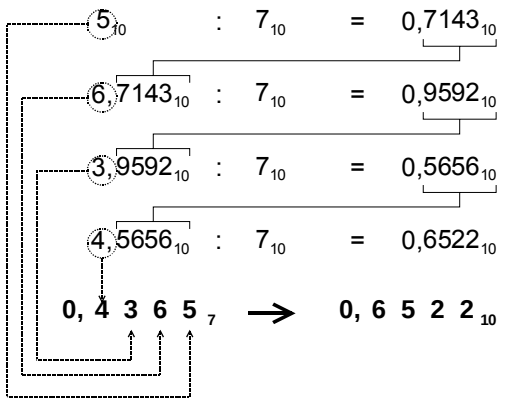
- z.B. $0.4365_7 \Rightarrow ???_{10}$
  (Erinnerung: multiplikative Umrechnung, vgl. Folie 2-34)

  ⇨ wg. Darstellung als negativer Exponent der Basis heißt das **Division**
- Division nach Horner mit Anschreiben der Ziffern in umgekehrter Reihenfolge

- $0{,}4365_7 = \frac{1}{7} * (4 + \frac{1}{7} * (3 + \frac{1}{7} * (6 + \frac{1}{7} \ 5)))$

  $= (((\frac{5}{7} + 6) * \frac{1}{7} + 3) * \frac{1}{7} + 4) * \frac{1}{7}$

Umrechnen im Zielsystem (2/2)



Überblick

- Zahlen
  - Informationsdarstellung
  - Zahlensysteme
  - **Rechnerarithmetik**
- Logik
  - Aussagenlogik und logische Gatter
  - Prädikatenlogik

Zahlendarstellung und Rechnen im Dualsystem (1/3)

- Durch N Bits lassen sich 2^N Zahlenwerte codieren
- Nur positive Zahlen, Wertebereich: 0 ... $(2^N–1)$
- Positive und negative Zahlen:
  - (N-1) - Bit Zahl mit Vorzeichen
  - darstellbarer Wertebereich: $[-2^{N-1} ... 0 ... 2^{N-1}-1]$
- „Most significant“ Bit als Vorzeichencodierung

7	6	5	4	3	2	1	0
↑	„most significant bit“						
0	= positive Zahl						
1	= negative Zahl						

- N = 8 ⇨ Wertebereich: 0 ... 255 oder -128 ... +127
- N= 16 ⇨ Wertebereich: 0 ... 65535 oder -32768 ... +32767

Zahlendarstellung und Rechnen im Dualsystem (2/3)

- N Bits erlauben 2^N Möglichkeiten zur Zahlendarstellung, d.h. nur eine Darstellung mit endlicher Genauigkeit.
  - Bei positiven Zahlen:
    Zahlen n und $n + B^N$ sind nicht unterscheidbar
    (höchstwertige Bit geht verloren)
  - Bei positiven und negativen Zahlen:
    Additionsüberlauf kann zu negativen Ergebnissen führen
  - Beispiel:

```
   0  1 0 0 1 1 1 0          78
+  0  1 0 1 1 0 0 0      +   88
-------------------     -------
   1  0 1 0 0 1 1 0         166     > 128
```

-38

negatives Vorzeichen

Zahlendarstellung und Rechnen im Dualsystem (3/3)

- **Was macht man?**
  - man führt Operationen auf einem Zahlenring durch
  - Darstellung (Codierung) der Zahlen als Binärworte

3 Darstellungsformen für Zahlen

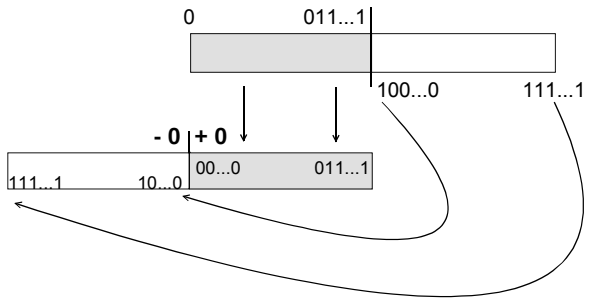
1. Vorzeichen und Betrag (engl. Sign-/Magnitude)
2. Stellenkomplement
3. Basiskomplement

- **Warum 3 Darstellungsformen?**
  - Ziel: möglichst einfache ALU
    (CPU Baustein, der Rechenoperationen durchführt)
  - Wie macht man das?
    - Subtraktion wird auf die Addition zurückgeführt
    - Multiplikation wird auf die Addition zurückgeführt
    - Division wird auf die Multiplikation (d.h. Addition) zurückgeführt

Darstellung mit Vorzeichen und Betrag (1/2)

- Wortlänge N
- N-1 Bits beschreiben den **Betrag** der Zahl
- 1 Bit beschreibt das Vorzeichen der Zahl
- zwei Darstellungen für die NULL
- engl. 'Sign-/Magnitude Representation'
- Die Vorzeichen und Betrag Darstellung entspricht dem was man so kennt.

Darstellung mit Vorzeichen und Betrag (2/2)



- **Beispiel für diese Darstellung:**

Dezimal	Vorzeichen Betrag
+ 92	0000 0101 1100
- 92	1000 0101 1100

Nachteile der Betrag-/Vorzeichen-Darstellung

- Man benötigt Addier- und Subtrahierwerk in der ALU
- Man benötigt spezielle Logik um zu bestimmen, ob addiert oder subtrahiert werden soll
- Beispiel: die Operanden x und y sollen addiert werden

  es sind folgende Fälle zu unterscheiden: $+x \Rightarrow x \geq 0$ $\quad -x \Rightarrow x < 0$

  $+y \Rightarrow y \geq 0$ $\quad -y \Rightarrow y < 0$

Fall	Operanden	Auszuführende Operation
1	+x +y (zwei positive Operanden)	Addition x + y
2	-x-y (zwei negative Operanden)	Addition -(x + y)
3	|positiver Operand| ≥ |negativer Operand| +x, -y, |x| ≥ |y| bzw. +y, -x, |y| ≥ |x|	Subtraktion x – y bzw. y – x
4	|negativer Operand| ≥ |positiver Operand| +x, -y, |x| < |y| bzw. +y, -x, |y| < |x|	Subtraktion –(|y| – |x|) bzw. – (|x| – |y|)

Komplementdarstellungen

1. Stellenkomplement (B-1 Komplement)
2. Basiskomplement

- **Ziel:** Rückführung der Subtraktionen auf die Addition
- **Komplementbildung:** $\underline{b} = C - b$ (für geeignetes C)
- **Subtraktion durch Addition des Komplements:**

$$a + \underline{b} = a - b + C$$
$$\Leftrightarrow \quad a - b = a + \underline{b} - C$$

- d.h. wenn (1.) *das Komplement leicht zu bilden ist*
  und (2.) *die Reduktion mod C einfach ist,*
  dann kann die Subtraktion auf eine Addition des Komplements zurückgeführt werden

Stellenkomplement (1/2)

- C für das Stellenkomplement: $\mathbf{C = B^N - 1}$
- Darstellung einer Zahl a: $n = \sum_{i=0}^{N-1} a_i * B^i$
- Komplementbildung:

$$\underline{a} = C - a = (B^N - 1) - \sum_{i=0}^{N-1} a_i * B^i$$

$$= \sum_{i=0}^{N-1} B^{i+1} - \sum_{i=0}^{N-1} B^i - \sum_{i=0}^{N-1} a_i * B^i$$

$$B^N - 1$$

$$= \sum_{i=0}^{N-1} ((B-1) - a_i) * B^i$$

- **$((B-1) - a_i)$ bedeutet:**
  (B-1)-Komplement kann für jede Stelle (= Stellenkomplement) gebildet werden.

Stellenkomplement für die Basis 2

- Wie funktioniert die (B-1)-Komplementbildung für die Basis 2?
- einsetzen in die bekannten Formeln:

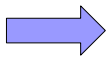
$$a = \sum_{i=0}^{N-1} a_i * 2^i$$

$$\underline{a} = \sum_{i=0}^{N-1} ((2-1) - a_i) * 2^i$$

$$= \sum_{i=0}^{N-1} (1 - a_i) * 2^i \qquad (1 - a_i) \text{ d.h. Invertierung jeder Stelle der Dualzahl}$$

- Beispiel: $x = x_5\, x_4\, x_3\, x_2\, x_1\, x_0 = 0\ 1\ 0\ 0\ 1\ 0$

$x_5 = 0 \rightarrow \underline{x_5} = 1 - x_5 = 1 - 0 = 1$
$x_4 = 1 \rightarrow \underline{x_4} = 1 - x_4 = 1 - 1 = 0$
$x_3 = 0 \rightarrow \underline{x_3} = 1 - x_3 = 1 - 0 = 1$
$x_2 = 0 \rightarrow \underline{x_2} = 1 - x_2 = 1 - 0 = 1$
$x_1 = 1 \rightarrow \underline{x_1} = 1 - x_1 = 1 - 1 = 0$
$x_0 = 0 \rightarrow \underline{x_0} = 1 - x_0 = 1 - 0 = 1$



$\underline{x} = 1\ 0\ 1\ 1\ 0\ 1$

Stellenkomplement und andere Basen

- Formeln: $a = \sum_{i=0}^{N-1} a_i * B^i$; $\quad \underline{a} = \sum_{i=0}^{N-1} ((B-1) - a_i) * B^i$
- im Dualsystem: *Komplementbildung ≙„Bits stellenweise invertieren“*

- Man kann das Stellenkomplement auch auf andere Zahlensysteme (neben dem Dualsystem) anwenden,
  z.B. Dezimalsystem (d.h. B = 10):

  $x = 3\ 5\ 7$

$x_2 = 3 \rightarrow \underline{x_2} = (10-1) - x_2 = 9 - 3 = 6$
$x_1 = 5 \rightarrow \underline{x_1} = (10-1) - x_1 = 9 - 5 = 4$
$x_0 = 7 \rightarrow \underline{x_0} = (10-1) - x_0 = 9 - 7 = 2$

$\underline{x} = 6\ 4\ 2$

Eigenschaften Stellenkomplement (Dualsystem)

- Im Rechner werden die negativen Zahlen als Komplemente der positiven Zahlen dargestellt.
- Das „most significant Bit“ beschreibt das Vorzeichen der repräsentierten Zahl:

  0 ⇒ +

  1 ⇒ –

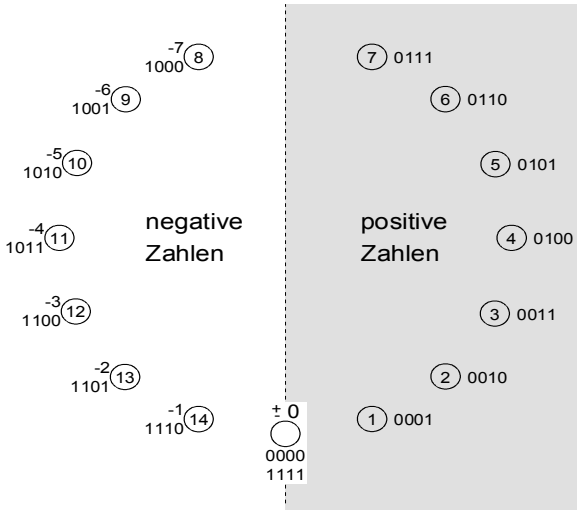
- Zwei Darstellungen der Null

  0 0 0 ... 0 Und 1 1 1 ... 1

- Darstellbarer Zahlenbereich

  $-(½ * B^N - 1)$ ... $+(½ * B^N - 1)$

Zahlenring im Stellenkomplement



Stellenkomplement: Reduktion mod C (1/7)

- **Erinnerung:** Stellenkomplement ist nur sinnvoll, wenn
  - Komplementbildung einfach ist ✓ (Stellenweise invertieren)
  - Reduktion mod C einfach ist
- Reduktion mod C: $d = a - b = a + \underline{b} - C$
- Es sind 3 Fälle zu unterscheiden:

1. Fall (zwei positive Zahlen) $a, b > 0 ;\ a > b \Rightarrow d > 0$
2. Fall (zwei positive Zahlen) $a, b > 0 ;\ a \leq b \Rightarrow d \leq 0$
3. Fall (zwei negative Zahlen) $d = -a - b < 0 ;\ |d| = a + b$

Stellenkomplement: Reduktion mod C (2/7)

- **1. Fall: a, b > 0 ; a > b ⇒ d > 0**
  - Abschätzung: $B^N \leq a + \underline{b} < 2*B^N$
    d.h.: Es tritt ein Überlauf von 1 in die (nicht existierende) (N+1).te Stelle auf.
  - Warum?

$$a + \underline{b} = a + (C - b)$$
$$= a + (B^N - 1 - b)$$
$$= a - b - 1 + B^N$$
$$0 < d < B^N$$

  - Ignorieren des Übertrags entspricht einer Subtraktion von B^N
  - Aufaddieren einer 1 entspricht dann einer Subtraktion von $C = B^N - 1$
  - d.h. man muß den Überlauf addieren um zum richtigen Ergebnis zu gelangen.
  - Stichwort: **“Einserrücklauf“**

Stellenkomplement: Reduktion mod C (3/7)

- 1. Fall: 3 Beispiele

I. Dezimalsystem (N = 3)

a – b	Komplementbildung	a + *b* – 1
65		065
– 43	*43* = 999 – 043 = 956	+ 956
		1 021
		→ 1
22		022

II. Dualsystem (N = 5)

a – b		a + *b* – 1		Dezimal
0 1 1 1 0		0 1 1 1 0		14
– 0 0 1 1 1	⇒	+ 1 1 0 0 0	(Komplement)	– 7
		1 0 0 1 1 0		
		→ 1		
		0 0 1 1 1	⇒	7

Stellenkomplement: Reduktion mod C (4/7)

- 1. Fall: 3 Beispiele

III. Dualsystem (N = 5)

Dezimal	a + *b* – 1
1 4	0 1 1 1 0
– 0	+ 1 1 1 1 1
	1 0 1 1 0 1
	→ 1
1 4	0 1 1 1 0

Bemerkung 1:
00000 = 11111

Bemerkung 2:
„negative Null stört nicht“

Stellenkomplement: Reduktion mod C (5/7)

- **2. Fall: a, b > 0 ; a ≤ b ⇒ d ≤ 0**
  - Differenz d = a – b ist negativ, d.h. d muß selbst in der Komplementdarstellung vorliegen: **d ⇒ C – |d|**

$$d = -|d| = a + \underline{b} - C \;\Leftrightarrow$$
$$C - |d| = a + \underline{b} \;\leq\; C = B^N - 1$$

  - d.h. die Addition des Komplements liefert bereits **d** in der richtigen Darstellung!
  - Es tritt kein Überlauf in die (N+1).te Stelle auf, weil: **a + *b* ≤ B^N – 1**
  - Beispiel:

dezimal	dual		a + *b*
5	– 0 0 1 0 1		+ 0 0 1 0 1
– 7	0 0 1 1 1	⇒	1 1 0 0 0
– 2			1 1 1 0 1

Stellenkomplement: Reduktion mod C (6/7)

- **3. Fall: d = –a – b < 0 ; |d| = a + b**
  - **d** in Komplementdarstellung **d = C - |d|** ist erwünscht

$$\begin{aligned} \underline{a} + \underline{b} &= (C - a) + (C - b) \\ &= (C - (a + b)) + C \\ &= (C - |d|) + C \\ &= |d| + (B^N - 1) \end{aligned}$$

verschwindet durch ignorieren des Überlaufs und Einserrücklauf

Gewünschte Ergebnis

  - Überlauf in die (N+1).te Stelle und Einserrücklauf liefern **|d|** in Komplementdarstellung (d.h. das gewünschte Ergebnis)

Stellenkomplement: Reduktion mod C (7/7)

- **3. Fall: d = –a – b < 0 ; |d| = a + b**
  - Beispiel:

		Dual
–	7	1 1 0 0 0
–	5	+ 1 1 0 1 0
		1 1 0 0 1 0
		⌊⌋
		⌊ ⌋ ⟶ 1
–	**12**	**1 0 0 1 1**

(12 ≅ 0 1 1 0 0)

Basiskomplement (B-Komplement)

- **C für das Basiskomplement:** $C = B^N$
- **Komplementbildung:**

$$ {}^{B}\underline{a} = B^N - \sum_{i=0}^{N-1} a_i * B^i = (B^N - 1) - (\sum_{i=0}^{N-1} a_i * B^i) + 1 $$

$$ = {}^{B-1}\underline{a} + 1 $$

  - d.h. die Komplementbildung ist für das Basiskomplement etwas aufwendiger
    - (B–1)–Komplement bilden
    - Aufaddieren einer 1
  - Beispiel:

	5 ≅	0 0 1 0 1
(B-1)-Komplement:	${}^{B-1}\underline{5}$ ≅	1 1 0 1 0
Aufaddieren von 1:		+ 1
	${}^{B}\underline{5}$ ≅	1 1 0 1 1

Eigenschaften Basiskomplement (Dualsystem)

- Negative Zahlen werden als Komplemente der positiven Zahlen dargestellt.
- Das „most significant Bit“ beschreibt das Vorzeichen der Zahl.

  0 ⇒ +

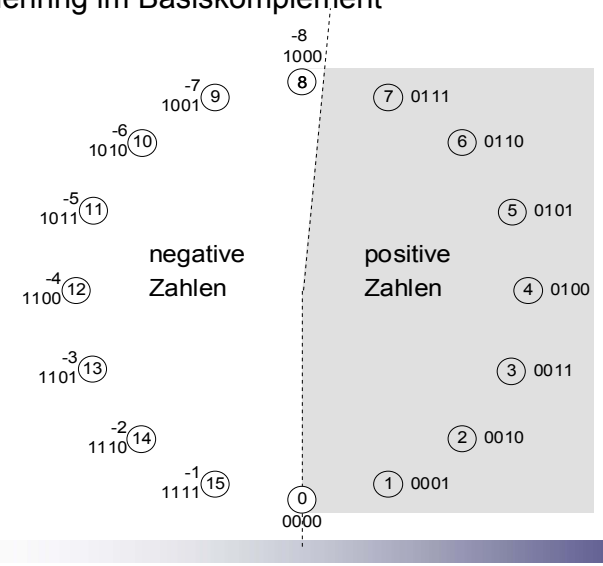
  1 ⇒ –

- Es gibt nur eine Null
  Bemerkung: Komplement der Null liegt beim Basiskomplement, anders als beim Stellenkomplement, nicht mehr im darstellbaren Zahlenbereich (Übertrag in die (n+1).te Stelle).

- Darstellbarer Zahlenbereich

  $-(½ * B^N)$ ... $+(½ * B^N - 1)$

Zahlenring im Basiskomplement



Reduktion mod C im Basiskomplement (1/2)

- einfach, da C = B^N,
  d.h. Überlauf in die (N+1).te Stelle ignorieren entspricht bereits der Subtraktion von B^N ⇒ „**kein Einserrücklauf**“
- Beispiele:
  - Dezimalsystem (N=3)

$$\begin{array}{rcl} 065 & & 065 \\ -\ 043 & {}^{B}043 = 956 + 1 \quad \Rightarrow & +\ 957 \\ \hline & & 1\ 022 \\ & & \Downarrow \\ \mathbf{022} & & \mathbf{022} \end{array}$$

  - Dualsystem (N=5)

dezimal	Dual	
14	0 1 1 1 0	
– 7	+ 1 1 0 0 1	(≅ $^{B}7$)
7	1 0 0 1 1 1	

Reduktion mod C im Basiskomplement (1/2)

- Beispiele:
  - Dualsystem (N=5)

dezimal	Dual	
5	0 0 1 0 1	
– 7	+ 1 1 0 0 1	
– 2	1 1 1 1 0	(≅ $^{B}2$)

Fall d < 0: Ergebnis ist bereits in Komplementdarstellung

Stellen- und Basiskomplement (1/2)

- Überschreiten des zulässigen Zahlenbereichs
  - Bei Operanden mit ungleichem Vorzeichen ist ein Überschreiten des zulässigen Zahlenbereichs nicht möglich:

    $|a - b| \leq |a|$ oder $|a - b| \leq |b|$

  - **Überschreitung möglich, wenn:**

    a, b < 0 oder a, b > 0

    - da das oberste Bit das Vorzeichen repräsentiert, kann bei Überschreitung des Zahlenbereichs das Vorzeichen wechseln.
    - In so einem Fall muß:
      - ein Überlauf angezeigt werden,
      - eine arithmetische Fehlerbedingung aktiviert werden
      - „Overflow error“ angezeigt werden,
      - ....
      - usw.

Stellen- und Basiskomplement (2/2)

- **Fall 1: a, b > 0**
  - Ein Fehler entsteht dann, wenn gilt $a + b \geq B^N\text{-}1$
  - Das vorderste (‚most significant‘) Bit wird gesetzt und als falsches Resultat entsteht eine negative Zahl.
  - Es gibt jedoch keinen Überlauf in die (N+1).te Stelle.
- **Fall 2: a, b < 0**
  - Ein Fehler entsteht dann, wenn gilt

    $|a| + |b| > B^N\text{-}1$ (Basiskomplement)

    $\geq B^N\text{-}1$ (Stellenkomplement)
  - Das Vorzeichenbit wird dann zurückgesetzt, es gibt einen Überlauf in die (N+1).te Stelle und es entsteht ein falsches ‚positives‘ Resultat.
  - Beim Stellenkomplement führt dieses normalerweise zum Einserrücklauf

Gegenüberstellung der 3 Zahlencodierungen

	Rechenwerk	Negation	Einserrücklauf
Vorzeichen-/ Betrag	Add / Sub	Vorzeichen invertieren	Nein
Stellenkomplement	Add	Ziffern komplementieren	Ja
Basiskomplement	Add	Stellenkomplement+ 1	Nein

- **Stellenkomplement:**
  - Verzögerung beim Rechnen wg. Einserrücklauf
  - Zwei Darstellungen der NULL
- **Basiskomplement**
  - Mehraufwand beim Herstellen des Komplements
  - Verzögerung beim Negieren
- **Praxis:** heute fast nur noch Basiskomplement

Multiplikation und Division (1/2)

- zurückführen auf Addition (bzw. Subtraktion)
- **Multiplikation:**
  Wiederholtes Verschieben (Shift-Operationen) und Addieren im Dualsystem

```
0 1 0 1 1 0 1 0 1  *  0 1 0 1 1 0
                      0 0 0 0 0 0 0 0 0
                    0 1 0 1 1 0 1 0 1
                  0 0 0 0 0 0 0 0 0
                +   0 1 0 1 1 0 1 0 1
                      0 1 0 1 1 0 1 0 1
                        0 0 0 0 0 0 0 0 0
                0 0 1 1 1 1 1 0 0 0 1 1 1 0
                Shift-Operationen
```

(Genügend Wortbreite für das Resultat angenommen)

Multiplikation und Division (2/2)

- **Division:**

```
1 1 0 1 0 1 0 0 1  : 1 0 0 1 0   = 1 0 1 1 1
– 1 0 0 1 0
  1 0 0 0 1 0                          Rest: 1 0 1 1
–   1 0 0 1 0
    1 0 0 0 0 0
–   1 0 0 1 0
        1 1 1 0 1
      – 1 0 0 1 0
          1 0 1 1
```

  - Wiederholtes Verschieben und Subtrahieren im Dualsystem
  - Schwierig bei negativem Divisor
    - eine Lösung: rechnen mit absoluten Beträgen und anschließend Vorzeichenrechnung
  - Fehlermeldung für Divisor = 0 („Zerodivide“)

BCD–Darstellung & BCD–Arithmetik (1/4)

- 4.te Alternative zur Darstellung und zum Rechnen mit ganzen Zahlen
- BCD–Darstellung und BCD–Arithmetik wird von verschiedenen Mikroprozessoren direkt unterstützt
- BCD–Codierung:

0_{10}	≙	0000_{BCD}	‘+‘	≙	1010
1_{10}	≙	0001_{BCD}	‘-‘	≙	1011
2_{10}	≙	0010_{BCD}			1100
3_{10}	≙	0011_{BCD}			1101
4_{10}	≙	0100_{BCD}			1110
5_{10}	≙	0101_{BCD}			1111
6_{10}	≙	0110_{BCD}			
7_{10}	≙	0111_{BCD}			
8_{10}	≙	1000_{BCD}			
9_{10}	≙	1001_{BCD}			

ungenutzte Werte

6 Codierungen, die nicht zur Zahlen-darstellung benutzt werden

BCD–Darstellung & BCD–Arithmetik (2/4)

- Beispiel 1:

dezimal	BCD –Darstellung	
23	0010	0011
+14	0001	0100
37	0011	0111

  - normale Dualzahlenaddition liefert hier das korrekte Ergebnis
- Beispiel 2:

dezimal	BCD –Darstellung		
29	0010	1001	
+14	0001	0100	
3?	0011	1101	→ ungültiger BCD Code
(6)	1	0110	→ Korrektur
43	0100	0011	

  - Bei Überträgen und beim Erreichen von ungültigen BCD-Codierungen liefert die Addition von 6_{10} (= 0110_{BCD}) das richtige Ergebnis

BCD–Darstellung & BCD–Arithmetik (3/4)

- Beispiel 3: 4739 + 1287 = 6026

0100	0111	0011	1001	
0001	0010	1000	0111	
		1		
0101	1001	1100	0000	→ Übertrag
			0110	→ + 6
0101	1001	1100	0110	→ ungültiger BCD Code
	1	0110		→ + 6
0101	1010	0010	0110	→ ungültiger BCD Code
1	0110			→ + 6
0110	0000	0010	0110	
⇓	⇓	⇓	⇓	
6	0	2	6	

BCD–Darstellung & BCD–Arithmetik (4/4)

- **Abschließende Bemerkungen:**
  - Der BCD-Code erlaubt auch das Rechnen mit Festzahlen, d.h. Zahlen bei denen die Anzahl der Stellen hinter dem Komma festliegt
  - Wenn der Ablauf „Zahleneingabe, Arithmetik, Zahlenausgabe“ ohne Umwandlung ins Binärsystem erfolgen soll, wird der BCD-Code bevorzugt.
  - Der BCD-Code wird vorwiegend im technischen Bereichen verwendet, z.B. für 7-Segment-Anzeigen zur Darstellung von Zahlen.

Gleitkommazahlen: Motivation

- Häufig berichteter Fehler in GCC (GNU Compiler Collection)
  - Quelle: http://gcc.gnu.org/bugs.html#known (Stand 19. April 2004)
  - Non-bugs: „The following are not actually bugs, but are reported often enough to warrant a mention here.“
  - Inkorrekte Handhabung von Gleitkommazahlen, z.B.:

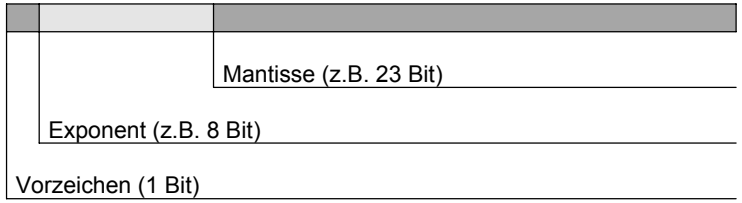
```
#include <iostream>
int main() {
  double a = 0.5;
  double b = 0.01;
  std::cout << (int)(a / b) << std::endl;
  return 0;
}
```

  - In Abhängigkeit von der verwendeten Genauigkeit und Rundung wird als Ergebnis 50 (korrekt) oder 49 (falsch) geliefert.

⇨ Kein Fehler des Compilers, sondern Einschränkung von Gleitkommazahlen

Gleitkommazahlen: „Floating Point Numbers“ (1/3)

- Rechner speichert die Position des Dezimalpunktes
  z.B. $0.473 * 10^{15}$ oder $0.3715 * 10^{-7}$
- Allgemein: $\mathbf{X = m * B^e}$ (m: **Mantisse**, B: **Basis**, e: **Exponent)**
  - Bit-Darstellung:



  - Exponent wird meist zur Basis 2, 10 oder 16 angegeben.
  - Führende Nullen der Mantisse sollten möglichst vermieden werden (‚**Normalisieren**‘)

Gleitkommazahlen: „Floating Point Numbers“ (2/3)

- Darstellung negativer Exponenten:
  - bereits bekannte Möglichkeiten:
    - Vorzeichen-/Betragdarstellung
    - Komplementdarstellung (Stellen- oder Basiskomplement)
  - **aber, die allgemein übliche Form ist die Darstellung mittels einer sog. Charakteristik**

$$ch = e + K$$

z.B. mit $K = B^N/2$ (sog. BIAS) (bei N-stelligen Exponenten)

⇨ d.h. anstelle des Exponenten wird die Charakteristik dargestellt

- Speziell: $\mathbf{X = m * B^{ch}}$ (m: **Mantisse**, B: **Basis**, ch: C**harakteristik)**

Gleitkommazahlen: „Floating Point Numbers“ (3/3)

- **Beispiel: +26.625** soll als 32 Bit Gleitkommazahl dargestellt werden
  - **Format:** 1 Bit Vorzeichen,
    8 Bit Exponent (Charakteristik),
    23 Bit Mantisse

1. **Schritt: Dezimalzahl ⇨ Dualzahl**
   $+(26{,}625)_{10} = +(11010.101)_2$
2. **Schritt: Normalisieren** (Anpassen der Mantisse m, $1/B \le m < 1$)
   $(11010.101)_2 * 2^0 = (0.11010101)_2 * 2^5$
3. **Schritt: Charakteristik berechnen:**
   $N=8 \Rightarrow K = 2^7 = 128$
   $ch = e + K = 5_{10} + 128_{10} = (101)_2 + (10000000)_2 = (10000101)_2$

31	30 23	22 0
0	1 0 0 0 0 1 0 1	1 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Vorz	**ch**	**M**

Gleitkommadarstellung nach IEEE Standard 754 (1/5)

- Normenvorschlag der IEEE Computer Society, der von fast allen Mikrocomputer-Herstellern übernommen worden ist (z.B. Intel ⇨ 80X86, Motorola ⇨ 680X0)
  - **32 Bit Format:**

31	30 23	22 0
S	Charak. **E**	Mantisse **M**

(Basis 2)

  - $\mathbf{X = (-1)^S * 2^{E-127} * (1.M)_2}$ mit $0 < E < 255$, **Bias = 127**

    Mantisse ist normiert: **1.M, d.h.**
    - **führende 1 wird nicht dargestellt**
    - **Erhöhung der Genauigkeit um eine Stelle**

Gleitkommadarstellung nach IEEE Standard 754 (2/5)

- **Beispiel**: x= -1.5

$x = (-1)^S * 2^{E-127} * (1.M)_2$ mit $0 < E < 255$

Bias = 127

31	30 23	22 0
1	0 1 1 1 1 1 1 1	1 0 0 0 0
S	Charakteristik E	Mantisse M

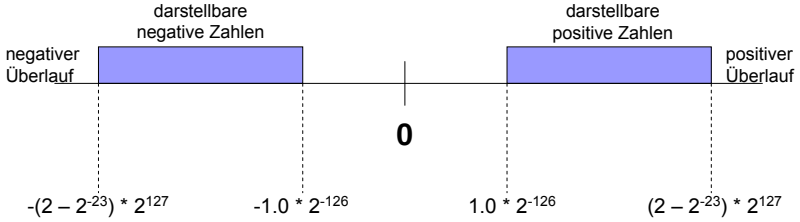
$(-1)^1 * 2^{127-127} * (1.1)_2 =$

$-1 * 2^0 * (1.1)_2 =$

$-(1.1)_2 = -1.5_{10}$

Gleitkommadarstellung nach IEEE Standard 754 (3/5)

- **Zahlenbereiche**
  - $1 * 2^{-126} \le |x| \le (2 - 2^{-23}) * 2^{127}$ oder dezimal umgerechnet
  - $1.18 * 10^{-38} \le |x| \le 3.40 * 10^{38}$



  - Es existiert eine Lücke im Bereich der darstellbaren Zahlen, die insbesondere die 'NULL' enthält.
  - Man benötigt eine eigene Darstellung der 'NULL'

Gleitkommadarstellung nach IEEE Standard 754 (4/5)

- **Besonderheiten des IEEE Formats**
  - **Not-a-Number:** Darstellung „ungültiger“ Zahlen, z.B. Division durch 0, Wurzel aus negativer Zahl.
  - **Überlauf** wird als unendlich dargestellt, d.h. es kann mit unendlich (∞) weitergerechnet werden.
  - **Unterlauf** kann denormalisiert dargestellt werden, d.h. es kann mit geringerer Genauigkeit weitergerechnet werden.
  - **Null:** eigene Darstellung

E	M	x	Erklärung
255	≠ 0	NaN	Not a Number
255	= 0	$(-1)^S * \infty$	±∞
0 < E < 255	beliebig	$(-1)^S * 2^{E-127} * (1.M)_2$	Normalbereich
0	≠ 0	$(-1)^S * 2^{126} * (0.M)_2$	Denormalisiert
0	= 0	$(-1)^S * 0$	0

Gleitkommadarstellung nach IEEE Standard 754 (5/5)

- **Abschließende Bemerkungen:**
  - Neben dem 32-Bit Format existieren noch 64- und 80-Bit Formate. Sie erlauben das Rechnen mit größerer Genauigkeit, aber prinzipiell folgt die Darstellung dem vorgestellten Mechanismus.
  - Die Norm geht davon aus, dass zuerst mit beliebiger Genauigkeit gerechnet wird und danach auf das jeweilige Zielformat gerundet wird. Daher verwenden die meisten Implementierungen eine höhere Genauigkeit für interne Berechnungen.
  - Moderne Gleitkommarecheneinheiten ermöglichen z.B. auch trigonometrische und logarithmische Operationen.
  - Links zu Demonstrationswerkzeugen und weitergehende Informationen über IEEE 754 sind auf der Vorlesungsseite angegeben.

Größerer Fehler bei präziserer Genauigkeit sind möglich

Beispiel: Zahlenbereiche für Typ double in « nearest to even » Modus:

16.0 + X = 16.0

- **Sun Solaris :** 64 bits registers
  X ∈ [-8.8817841970012523e-16 .. 1.7763568394002505e-15]
- **Intel Linux :** 80 bits registers (more accurate handling of expressions)
  X ∈ [-8.8861210056911943e-16 .. 1.7772242011382389e-15]

→ Der Fehler ist **größer** auf der präziseren Architektur (Intel Linux)

Gleitkommaarithmetik: Multiplikation

- **$Z = X * Y = (m[X] * m[Y]) * B^{e[X]+e[Y]}$**

  m[X]: *Mantisse von X*, m[Y]: *Mantisse von Y*
  e[X]: *Exponent von X*, e[Y]: *Exponent von Y*

  - **Multiplikation** der Mantissen
  - **Addition** der Charakteristiken (- Bias)

V_X	CH_X	M_X
⊕	+	*
V_Y	CH_Y	M_Y
	- Bias (127)	
=	=	=
V_Z	CH_Z	M_Z

  - Gegebenenfalls **normalisieren**
  - Division entsprechend
  - Sonderregeln für 0 als Operand

Gleitkommaarithmetik: Addition und Subtraktion

- **$Z = X + Y = (m[X] * B^{e[X]-e[Y]} + m[Y]) * B^{e[Y]}$** für: e[x] ≤ e [y]

  m[X]: *Mantisse von X*, m[Y]: *Mantisse von Y*
  e[X]: *Exponent von X*, e[Y]: *Exponent von Y*

  1. **Berechne e[X] – e[Y]**
  2. **Skalieren:** Angleichen der Mantisse des kleineren Exponenten
  3. **Addition der Mantissen**
  4. gegebenenfalls **Normalisieren**

  - Subtraktion entsprechend

Gleitkommaarithmetik: Beispiel für Addition

- Basis = 10, vierstellige Mantisse
- $Z = X + Y$; $X = 0.5320 * 10^{-2;}$ $Y = 0.4162 * 10^{1}$
- **Berechne e[X] – e[Y]:** -2 – 1 = -3
- **Skalieren von m[X]:** $0.5320 * 10^{-3} = 0.0005_{320} * 10^{0}$
  (Verschieben um 3 Stellen nach Rechts, evtl. mit Genauigkeitsverlust)
- **Addition der Mantissen:** Übernahme des größeren Exponenten

	0.4162	*	101
+	0.0005	*	101
	0.4167	*	101

- **Normalisieren entfällt**

Überblick

- Zahlen
  - Informationsdarstellung
  - Zahlensysteme
  - Rechnerarithmetik
- **Logik**
  - Aussagenlogik und logische Gatter
  - Prädikatenlogik