Kapitel 3: Darstellung von Informationen

- **Zeichencodierungen**
- **Binär-, Hexadezimal-, Zweierkomplementzahlen**
- **Gleitpunktzahlen**
- **Pixel-, Vektorgrafik**
- **Bewegtbilder**

Lernziele

Sie können

- **unterschiedliche Zeichencodierungen mit Vor- und Nachteilen nennen**
- **Verschiedene Zahlenformate erläutern und Einsatzgebiete nennen**
- **Gleitpunktzahlen nennen**
- **Verschiedene Grafikformate mit Ihren Vor- und Nachteilen nennen**

Alphanumerische Zeichendarstellung

- Alphanumerische Daten bestehen aus Ziffern oder Zahlen, Buchstaben oder Sonderzeichen
- Beispiele: DIPL.-INFORM., <Zeichenkette12>, Sehr geehrte ..., Agent 007
- Eindeutige Zuordnung von Zeichen zu Bitkombinationen notwendig => Kodierung
- Kodierung von Buchstaben, Ziffern, Sonderzeichen
- EBCDI-Code und ASCII-Code

Bestandteile einer Codierung

- **Abstrakter Zeichenvorrat: Menge der codierten Zeichen**

  Achtung: es wird nicht die graphische Darstellungsformen (Glyph) codiert

  Beispiel: “Latin Capital Letter A” hat Glyphen: A **A** *A* ***A*** A **A** *A* ***A***

  …
- **Codetabelle: ordnet jedem Zeichen eine Codeposition zu (natürliche Zahl)**
- **Allgemein: Ein Code *f* über den Alphabeten *A* und *B* ist eine (eindeutige) Abbildung (= Codierung) der Form f:A→B. Er ordnet Wörtern aus Symbolen des Alphabets *A* Wörter aus dem Alphabet *B* zu.**
- **Der Code heißt** entzifferbar**, wenn es eine eindeutige Umkehrabbildung f^{-1} gibt, die jedem Nachrichtenwort aus *B* wieder das ursprüngliche Wort aus *A* zuordnet.**

Alphanumerische Zeichendarstellung - EBCDIC

- **EBCDI-Code**
- **„Extended Binary Coded Decimal Information Code“**
- **Kode für binär-verschlüsselte Dezimalzeichen**
- **Verwendung auf (IBM) Großrechnern und deren Nachbauten**
- **Aufteilung in zwei Halbbytes (oder Tetraden)**
- **Das linke Halbbyte oder die 1. Tetrade nennt man Zonenteil.**
- **Das rechte Halbbyte oder die 2. Tetrade nennt man Zifernteil**

Alphanumerische Zahlendarstellung - EBCDIC

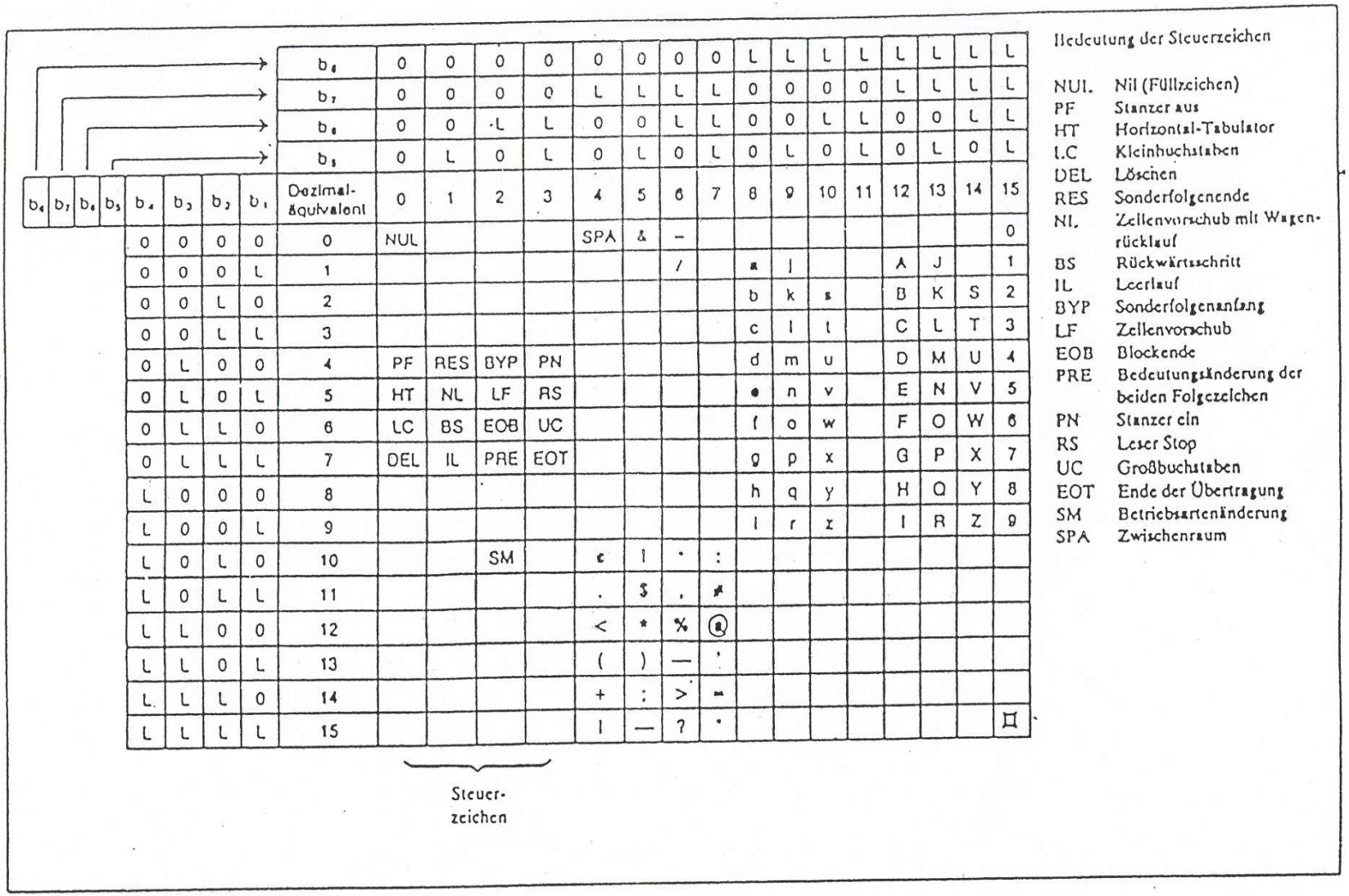
- **EBCDI-Code**
  - Beispiele: Darstellung des Buchstabens A und des Worts Hallo im EBCDI-Code

A <=> 1100 0001 dual oder 12 1 dezimal

1100 ist dabei der Zonenteil (1.Tetrade),
0001 der Zifferntеil (2. Tetrade)

Hallo <=> 12 8 8 1 9 3 9 3 9 6 (dezimal)

Alphanumerische Zahlendarstellung - EBCDIC



b_4 b_7 b_6 b_5	b_4	b_3	b_2	b_1	Dezimal-äquivalent	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	0	0	0	0	NUL				SPA	&	–									0
	0	0	0	L	1							/		a	j			A	J		1
	0	0	L	0	2									b	k	s		B	K	S	2
	0	0	L	L	3									c	l	t		C	L	T	3
	0	L	0	0	4	PF	RES	BYP	PN					d	m	u		D	M	U	4
	0	L	0	L	5	HT	NL	LF	RS					e	n	v		E	N	V	5
	0	L	L	0	6	LC	BS	EOB	UC					f	o	w		F	O	W	6
	0	L	L	L	7	DEL	IL	PRE	EOT					g	p	x		G	P	X	7
	L	0	0	0	8									h	q	y		H	Q	Y	8
	L	0	0	L	9									i	r	z		I	R	Z	9
	L	0	L	0	10			SM		¢	!		:								
	L	0	L	L	11					.	$	,	#								
	L	L	0	0	12					<	*	%	@								
	L	L	0	L	13					(	)	—	'								
	L	L	L	0	14					+	;	>	=								
	L	L	L	L	15					|	—	?	"								□

Steuer-zeichen

Bedeutung der Steuerzeichen

NUL	Nil (Füllzeichen)
PF	Stanzer aus
HT	Horizontal-Tabulator
LC	Kleinbuchstaben
DEL	Löschen
RES	Sonderfolgenende
NL	Zeilenvorschub mit Wagenrücklauf
BS	Rückwärtsschritt
IL	Leerlauf
BYP	Sonderfolgenanfang
LF	Zeilenvorschub
EOB	Blockende
PRE	Bedeutungsänderung der beiden Folgezeichen
PN	Stanzer ein
RS	Leser Stop
UC	Großbuchstaben
EOT	Ende der Übertragung
SM	Betriebsartenänderung
SPA	Zwischenraum

Tabelle EBCDIC

ASCII

- **(American Standard Code for Information Interchange)**
- **7 Bit Zeichenkodierung**
- **1967 standardisiert (zuletzt 1986 aktualisiert)**
- **umfasst**
  - lateinisches Alphabet in Groß und Kleinschreibung
  - zehn arabische Ziffern
  - einige Satz- und Steuerzeichen
- **umfasst** keine **„Sonderzeichen“ wie ÄÖÜäöüßçšñáèû…**

ASCII-Tabelle (7 Bit)

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	*NUL*	*SOH*	*STX*	*ETX*	*EOT*	*ENQ*	*ACK*	*BEL*	*BS*	*HT*	*LF*	*VT*	*FF*	*CR*	*SO*	*SI*
1…	*DLE*	*DC1*	*DC2*	*DC3*	*DC3*	*NAK*	*SYN*	*ETB*	*CAN*	*EM*	*SUB*	*ESC*	*FS*	*GS*	*RS*	*US*
2…	*SP*	!	"	#	$	%	&	´	(	)	*	+	,	-	.	/
3…	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4…	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5…	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6…	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7…	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	*DEL*

siehe auch http://de.wikipedia.org/wiki/ASCII-Tabelle

Beispiel einer Zeichencodierung

- **Text**

  Hallo Welt

- **ASCII-Zeichen**

  72 97 108 108 111 32 87 101 108 116

- **Bitfolgen**

  01001000 01100001 01101100 01101100 01101111
  00100000 01010111 01100101 01101100 01110100

- **hexadezimale Notation**

  48 61 6C 6C 6F 20 57 65 6C 74

Erweiterter ASCII-Code (ISO 8859-X)

- **alle 8 Bit tragen Zeichen-Information**
- **128 weitere Zeichen**
- **sprachspezifische Sonderzeichen wie Ä,Ö, ß, Rahmen, Schraffur, …**
- **es gibt 16 Erweiterungen, die von der ISO genormt sind**
- **Beispiele:**
  - ISO 8859-1, Latin-1, Westeuropäisch
  - ISO 8859-7, Griechisch
- **die ersten 128 Zeichen aller Erweiterungen sind gleich**

Unicode

- **Version 1.0.0 Okt 1991, Industriestandard,**
- **gepflegt vom gemeinnützigem „Unicode Consortium“**
- **Benutzt (in der Grundform) 2 Byte →65.536 Zeichen codierbar**
- **global verwendbar, da Schriftzeichen aller wichtigen Sprachen codierbar**
  - auch kyrillische, japanische, koreanische Zeichen
  - z. B. 20.000 Han-Ideogramme (Japan und China)
  - beinhaltet ISO 8859-X-Zeichen
  - Ūnĭcōdĕ, Юникод, Γιούνικοντ, უნიკოდი, 유니곧, 什麼是, 什么是, ユニコード, يونِكود, יוניקאָד, यूनिकोड,

Unicode aktuell

- seit Version 2.0 maximal 1.114.112 ($2^{16} + 16*2^{16}$) Unicode-Zeichen vorgesehen:
  2^{16} ursprünglichen Unicode-Zeichen und zusätzlich 16 weitere Bereiche (sog. Planes) dieser Größe
- in Unicode 5.0 (Juli 2006) sind nur 99.089 Codes tatsächlich konkreten Zeichen zugeordnet
  → 9% des Coderaumes
- erste 128 Zeichen identisch mit ASCII
- Neuere Sprachen (Java, XML) arbeiten mit Unicode

Unicode / ISO 10646 Zeichenlänge

- **Problem: Jedes Zeichen muss (je nach Version) mit bis zu 4 Byte gespeichert werden**
  → benötigt 4mal so viel Speicher wie ASCII
- **Idee: Statt für jedes Zeichen 4 Byte zu nutzen, werden wichtige Zeichen mit weniger Bytes, unwichtige mit mehr Bytes codiert**
- **Realisierung: z.B. UTF-8**

UTF-8 8-bit Unicode Transformation Format

- verbreitetste Codierung für Unicode-Zeichen
- jedem Unicode-Zeichen wird eine speziell kodierte Bytekette von variabler Länge zugeordnet
- UTF-8 benutzt bis zu 4 Byte, auf die sich alle 1.114.112 Unicode-Zeichen abbilden lassen
- UTF-8 hat zentrale Bedeutung als globale Zeichenkodierung im Internet

Vorteile von UTF-8

- **häufige Zeichen werden kurz, seltene Zeichen länger codiert**
  → dadurch wird weniger Speicherplatz als bei UCS-2 gebraucht
- **Zukunftssicherheit durch Vielzahl der noch nicht belegten Codepositionen**
- **zusammengehörende Bytes werden auch mitten im Text erkannt**

UTF-8 Codierung

Unicode-Bereich	UTF-8-Kodierung	Bemerkungen	Möglichkeiten	
0000 0000– 0000 007F	0xxxxxxx	In diesem Bereich (128 Zeichen) entspricht UTF-8 genau dem ASCII-Code: Das höchste Bit ist *0*, die restliche 7-Bit-Kombination ist das ASCII-Zeichen.	2^7	128
0000 0080– 0000 07FF	110xxxxx 10xxxxxx	Das erste Byte enthält binär 11xxxxxx, die folgenden Bytes 10xxxxxx. Die *x* stehen für die fortlaufende Bitkombination des Unicode-Zeichens. Die Anzahl der Einsen ausgehend von der höchsten 0 im ersten Byte ist die Anzahl der Bytes für das Zeichen. (In Klammern jeweils die theoretisch maximal möglichen.)	$2^{11} - 2^7$ (2^{11})	1.920 (2.048)
0000 0800– 0000 FFFF	1110xxxx 10xxxxxx10 xxxxxx		$2^{16} - 2^{11}$ (2^{16})	63.488 (65.536)
0001 0000– 0010 FFFF [0001 0000– 001F FFFF]	11110xxx 10xxxxxx10 xxxxxx 10xxxxxx		2^{20} (2^{21})	1.048.576 (2.097.152)

UTF-8 Codierung Beispiel 1

Zeichen	Unicode	Unicode binär	UTF-8 binär	UTF-8 hexadezimal
Buchstabe *y*	U+0079	00000000 01111001	01111001	0x79
Buchstabe *ä*	U+00E4	00000000 11100100	11000011 10100100	0xC3 0xA4
Zeichen für eingetragene Marke ®	U+00AE	00000000 10101110	11000010 10101110	0xC2 0xAE
Eurozeichen €	U+20AC	00100000 10101100	11100010 10000010 10101100	0xE2 0x82 0xAC
Violinschlüssel	U+1D11E	00000001 11010001 00011110	11110000 10011101 10000100 10011110	0xF0 0x9D 0x84 0x9E

UTF-8 Codierung Beispiel 2

CP	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	UTF8
0000																	00
0010																	10
0020		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	20
0030	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	30
0040	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	40
0050	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	50
0060	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	60
0070	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		70
0080																	C280
0090																	C290
00A0		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬		®	¯	C2A0
00B0	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	C2B0
00C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	C380
00D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	C390
00E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	C3A0
00F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	C3B0

UTF-8 Codierung Beispiel 3

CP	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	UTF8
2200	∀	∁	∂	∃	∄	∅	∆	∇	∈	∉	∊	∋	∌	∍	∎	∏	E28880
2210	∐	∑	−	∓	∔	∕	∖	∗	∘	∙	√	∛	∜	∝	∞	∟	E28890
2220	∠	∡	∢	∣	∤	∥	∦	∧	∨	∩	∪	∫	∬	∭	∮	∯	E288A0
2230	∰	∱	∲	∳	∴	∵	∶	∷	∸	∹	∺	∻	∼	∽	∾	∿	E288B0
2240	≀	≁	≂	≃	≄	≅	≆	≇	≈	≉	≊	≋	≌	≍	≎	≏	E28980
2250	≐	≑	≒	≓	≔	≕	≖	≗	≘	≙	≚	≛	≜	≝	≞	≟	E28990
2260	≠	≡	≢	≣	≤	≥	≦	≧	≨	≩	≪	≫	≬	≭	≮	≯	E289A0
2270	≰	≱	≲	≳	≴	≵	≶	≷	≸	≹	≺	≻	≼	≽	≾	≿	E289B0
2280	⊀	⊁	⊂	⊃	⊄	⊅	⊆	⊇	⊈	⊉	⊊	⊋	⊌	⊍	⊎	⊏	E28A80
2290	⊐	⊑	⊒	⊓	⊔	⊕	⊖	⊗	⊘	⊙	⊚	⊛	⊜	⊝	⊞	⊟	E28A90
22A0	⊠	⊡	⊢	⊣	⊤	⊥	⊦	⊧	⊨	⊩	⊪	⊫	⊬	⊭	⊮	⊯	E28AA0
22B0	⊰	⊱	⊲	⊳	⊴	⊵	⊶	⊷	⊸	⊹	⊺	⊻	⊼	⊽	⊾	⊿	E28AB0
22C0	⋀	⋁	⋂	⋃	⋄	⋅	⋆	⋇	⋈	⋉	⋊	⋋	⋌	⋍	⋎	⋏	E28B80
22D0	⋐	⋑	⋒	⋓	⋔	⋕	⋖	⋗	⋘	⋙	⋚	⋛	⋜	⋝	⋞	⋟	E28B90
22E0	⋠	⋡	⋢	⋣	⋤	⋥	⋦	⋧	⋨	⋩	⋪	⋫	⋬	⋭	⋮	⋯	E28BA0
22F0	⋰	⋱	⋲	?	?	?	?	?	?	?	?	?	?	?	?	?	E28BB0

Beispiel: Darstellung der Zahl 73

- **Unterscheide: Zahl als mathematisches Objekt und ihre Darstellung durch Zeichenfolgen.**
- **Beispiele für die Darstellung der Zahl 73**
  - Dezimalzahl „73“$_{10}$
  - Hexadezimal „49“$_{16}$ oder „0x49“ in Java
  - Binärzahl „ 1001001“$_{2}$

Stellenwertsystem

- **Interpretation der Zahldarstellungen in der Stellenschreibweise (Stellenwertsysteme)**
- **Beispiele:**
  - $73_{10} = 7*10^1 + 3*10^0 = 70+3$
  - $1001001_2 = 1*2^6 +0*2^5+0*2^4+1*2^3+0*2^2+0*2^1+1*2^0 = 64+8+1$
  - $49_{16} = 4*16^1 + 9*16^0 = 64+9$
- **allgemein:**
  - $(a_n\ a_{n-1}...\ a_1\ a_0)_b = a_n*b^n + a_{n-1}*b^{n-1}...+ a_1*b+ a_0$
  - b : Basis des Zahlensystems (meist 10, 2 oder 16)
  - a_i ist eine Ziffer (d. h. Zahlzeichen) mit $a_i < b$

Stellenwertsystem

- Darstellung einer Zahl durch eine Folge von alphanumerischen Zeichen (Ziffern)
- Jeder Stelle ist ein unterschiedlicher Stellenwert zuordnet
- Der Stellenwert ist immer eine Potenz der Basis des Zahlensystems und hängt von der Stelle ab
- Häufige Schreibweise: ein tief gestellter Index hinter einer Ziffernfolge stellt die Basis des Zahlensystems dar
- Wenn Ziffernfolgen keine Indexzahl haben, dann ist die Ziffernfolge als Dezimalzahl zu interpretieren
- Die dargestellte Zahl ist die Summe der mit den Stellenwerten multiplizierten Zahlenwerten der Ziffern

Binärzahlen → Dezimalzahlen

- **Binärzahlen sind Bitstrings (d. h. Zeichenfolgen aus dem Alphabet {0,1}) , die nach der Stellenwertmethode mit der Basis 2 als natürliche Zahlen interpretiert werden**
- **Beispiele:**
  - $1_2 = 1$
  - $111_2 = 7$
  - $10001_2 = 17$
  - $10000000000_2 = 1024$
- **Grundrechenarten wie im Dezimalsystem**

Dezimalzahlen → Binärzahlen

- **Es gibt mehrere Möglichkeiten der Umrechnung ins Dualsystem.**
- **Eine Möglichkeit: Divisionsmethode (auch Modulo-Methode genannt) am Beispiel 41_{10}**

$$
\begin{array}{rcrcl}
41 & :2 & = & 20 & \text{Rest } 1 \\
20 & :2 & = & 10 & \text{Rest } 0 \\
10 & :2 & = & 5 & \text{Rest } 0 \\
5 & :2 & = & 2 & \text{Rest } 1 \\
2 & :2 & = & 1 & \text{Rest } 0 \\
1 & :2 & = & 0 & \text{Rest } 1
\end{array}
\quad \uparrow
$$

- **Die entsprechende Dualzahl ergibt sich durch Notation der errechneten Reste von unten nach oben: 101001_2**

Eigenschaften

- **Stellenwert-Zahlensystem zur Basis 16**
- **nutzt zur Darstellung die Ziffern 0-9 und die Buchstaben A-F → 16 Zeichen**
- **hexadezimal (von griech. „hexa“ und lat. „decem“)**
- **Wird verwendet zur abkürzenden Notation von Bitmustern bzw. Zahlen**
- **Beispiele:**
  - Farbangaben in HTML (bgcolor= "#C0FF3A ") rot: C0, grün: FF, blau 3A
  - Codepositionen in ASCII- / Unicode-Tabellen

Hex	Dualsystem				Dez
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
A	1	0	1	0	10
B	1	0	1	1	11
C	1	1	0	0	12
D	1	1	0	1	13
E	1	1	1	0	14
F	1	1	1	1	15

Darstellung negativer Zahlen ?

- **Möglichkeit: Vorzeichencodierung in einem Bit**
  - 1 : Minuszeichen
  - 0 : Plus
- **Nachteile:**
  - 0 hat zwei Darstellungen
  - Additionsalgorithmus nicht mehr so einfach
  - Beispiel -3 + 5 = 2

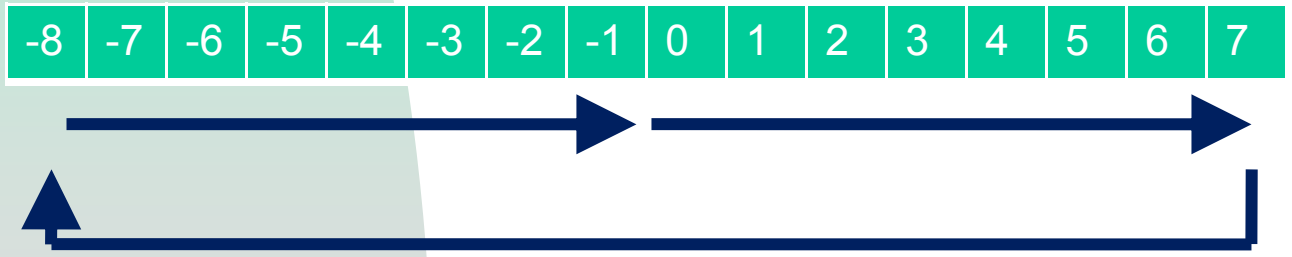
1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	1
?	?	?	?	?	?	?	?

Einführung der 2K-Zahlen an Hand eines Beispiels

- Die 2K-Zahlen haben im Beispiel eine feste Länge von N=4 Bit
- Wie viel Zahlen sind mit 4 Bits darstellbar? (16)
- Es sollen ungefähr gleich viel positive und negative Zahlen dargestellt werden (incl. der Null)
- Welche 16 Zahlen „symmetrisch“ zum Nullpunkt sollten das sein?

Zweierkomplementdarstellung

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



Interpretation von 2K-Zahlen

- **Wie kann man eine negative Zahl erkennen?**
  → eine 1 in der ersten Stelle steht für das negative Vorzeichen
- **Welcher Zahlenraum ist mit n Bits darzustellen?**
  → in 2K-Zahlen mit n Bits sind die Zahlen -2^{n-1} bis $2^{n-1}-1$ darstellbar
- **2K-Zahlen haben eine feste Länge, d.h. führende Nullen können nicht weggelassen werden**
- **Vergleich mit Binärzahl-Interpretation?**
  → Der Unterschied zur Binärzahl-Interpretation liegt bei positiven Zahlen nur in der ersten Stelle (Vorzeichen)

Allgemeine Formel und Addition

- **Wie lautet die allgemeine Formel zur Berechnung des Dezimalwertes?**
  → $(b_n\ b_{n-1}....b_1\ b_0\ )_{2K} = -\ b_n * 2^n + b_{n-1} * 2^{n-1} +....+ b_1 * 2 + b_0$
- **Addition an den Stellen n-1 bis 0 wie bei Binärzahlen**
  - Ausnahme: Overflow-Analyse an der Stelle n-1
  - Zusätzlich: Für die Stelle n ist eine unterschiedliche Overflow-Analyse notwendig (warum?)
- **Vorteil: Prozessor benötigt kein spezielles Addierwerk für 2K-Operation (nur zusätzliche Überprüfung von Sonderfällen)**

Woher kommt der Name Zweierkomplementzahl?

- Das Komplement eines Bitstrings entsteht, in dem alle 1en durch 0en und alle 0en durch 1en ersetzt werden
- Die Negation einer Zahl b wird durch Komplementbildung der 2K-Darstellung von b und anschließender Addition von 1 erzeugt
- Beispiel: Negiere 19
  $-19 = -\ 010011_{2K} = (101100_{2K} + 000001_{2K}) = 101101_{2K} = -19$
- „Beweis“ am Beispiel
  es gilt immer: Die Addition einer 2K-Zahl mit ihrem Komplement ergibt -1.
- $101101_{2K} + 010010_{2K} = 111111_{2K} = -1$

Ganze Zahlen in Java

- **ganze Zahlen werden in Java intern als Zweierkomplementzahl dargestellt**

Datentyp	Länge in Bit	Zahlbereich
byte	**8 Bit**	**-128…127**
short	**16 Bit**	**-32.768…32.767**
int	**32 Bit**	**$-2^{31}…2^{31}-1$**
long	**64 Bit**	**$-2^{63}…2^{63}-1$**

Gleitpunktzahlen (GPZ)

- **andere Bezeichnungen:**
  - Real-Zahlen
  - Floatingpoint-Zahlen
  - Fließkommazahlen
  - Gleitkommazahlen
- **genutzt zur Darstellung der reeller Zahlen**
- **jedoch: nicht alle reellen Zahlen sind im Computer exakt darstellbar**

gebrochene Zahlen

- Zahlen mit Komma- (Punkt)-Werten
- Die gebrochene Dezimalzahl 75,385 wird interpretiert als:
  $7*10^{1} + 5*10^{0} + 3*10^{-1} + 8*10^{-2} + 5*10^{-3}$
- Format und Interpretation gebrochener Binärzahlen:
  $X = X_n X_{n-1} \ldots X_1 X_0 \, . \, Y_1 Y_2 \ldots Y_{m-1} Y_m$
- Interpretation:
- $X = X_n*2^{n} + X_{n-1}*2^{n-1} + \ldots + X_1*2^{1} + X_0 + Y_1*2^{-1} + Y_2*2^{-2} + \ldots + Y_{m-1}*2^{-m+1} + Y_m 2^{-m}$

Beispiel gebrochene Binärzahlen

Gebrochene Binärzahl	Gebrochene Dezimalzahl
0.1	**0,5**
0.01	**0,25**
0.11	**0,75**
1.11	**1,75**
111.111	**7,875**
11011101011.1001110111	**1771,6162109375**
10101010.10011001	**170,59765625**
0.0001100110011001…	**0,1**

Motivation für Gleitpunktzahlen

- **Beispiel**
  - Gleitpunktdarstellung: $-309.471 * 10^{19}$
  - Ohne die Zehnerpotenzmultiplikation würden 22 Stellen benötigt
  - in Java: -309.471e+19
- **Nachteile des Formats mit fester Stellenzahl ohne Exponententeil: Betragsmäßig sehr große oder kleine Zahlen sind evtl. nicht darstellbar, da die Stellenzahl für Ganzzahl- und Bruchanteil jeweils geringer ist als die vorgegebene Gesamtstellenzahl**
- **Gleitpunktzahlen sind gebrochene Zahlen mit Verschiebung im Stellensystem**

Gleitpunktzahlen-Format

- **Gleitpunktzahlen (GP-Zahlen) bestehen prinzipiell aus drei Teilen (Bitstrings):**
  - 1. Vorzeichen V,
  - 2. Exponent E,
  - 3. Mantisse M,
- **Grobformel für die Interpretation: $V^* M * 2^E$**
- **IEEE-754 Norm für Gleitpunktzahlen**
  - short real:
    Vorzeichen: 1 Bit, Exponent: 8 Bit, Mantisse: 23 Bit
    gesamt: 32 Bit
  - long real:
    Vorzeichen: 1 Bit, Exponent: 11 Bit, Mantisse: 52 Bit
    gesamt: 64 Bit

Genauigkeit von Gleitpunktzahlen

- **0,1 als Gleitpunktzahl**

V (1 Bit)	E (8 Bit)	Mantisse (23 Bit)	Zahlenwert
0	**01111011**	**10011001100110011001100**	**0,0999999940…**
0	**01111011**	**10011001100110011001101**	**0,1000000015…**

- **Verschieden lange Gleitpunktzahlen**

Bit	Vor-zeichen	Exponent	Mantisse	Gültige Dezimalstellen	Im Bereich von	bis
32	**1 Bit**	**8 Bit**	**23 Bit**	**~ 7**	**$\pm 1 * 10^{-38}$**	**$\pm 3*10^{38}$**
64	**1 Bit**	**11 Bit**	**52 Bit**	**~ 15**	**$\pm 1*10^{-308}$**	**$\pm 1*10^{308}$**
80	**1 Bit**	**15 Bit**	**64 Bit**	**~ 19**	**$\pm 1*10^{-4932}$**	**$\pm 1*10^{4932}$**

Rundungsfehler

- **prinzipielles Problem: nicht alle reellen Zahlen sind exakt darstellbar**
  →Rundung notwendig
  →Rundungsfehler bei der Darstellung
- **weitere Rundungsfehler beim Rechnen**
- **Welche Operationen sind besonders kritisch für die Rundungsfehler?**
- **Beispiel 0,1**
  - Wie groß ist jeweils der Rundungsfehler?
  - Welche Darstellung ist näher an 0,1?
  - (Oben $6*10^{-9}$ zu niedrig, Unten $1,5*\ 10^{-9}$ zu hoch)

Bildinformation

- Bilder sind optische Darstellungen von Sachverhalten oder Vorgängen
- Vektor- oder Pixelgrafik
- Die rechnergestützte Bildverarbeitung dient zur unmittelbaren oder später unveränderten Wiedergabe und / oder Übermittlung der Bildinformation.
- Weitere Aufgaben sind die Bildgenerierung, Bildanalyse und die Bildaufbereitung
- Festbilder – Bewegte Bilder
- 2- dimensional (Büroinformationssystem)
- 3 – dimensional (CAD – computerunterstütztes Konstruieren)

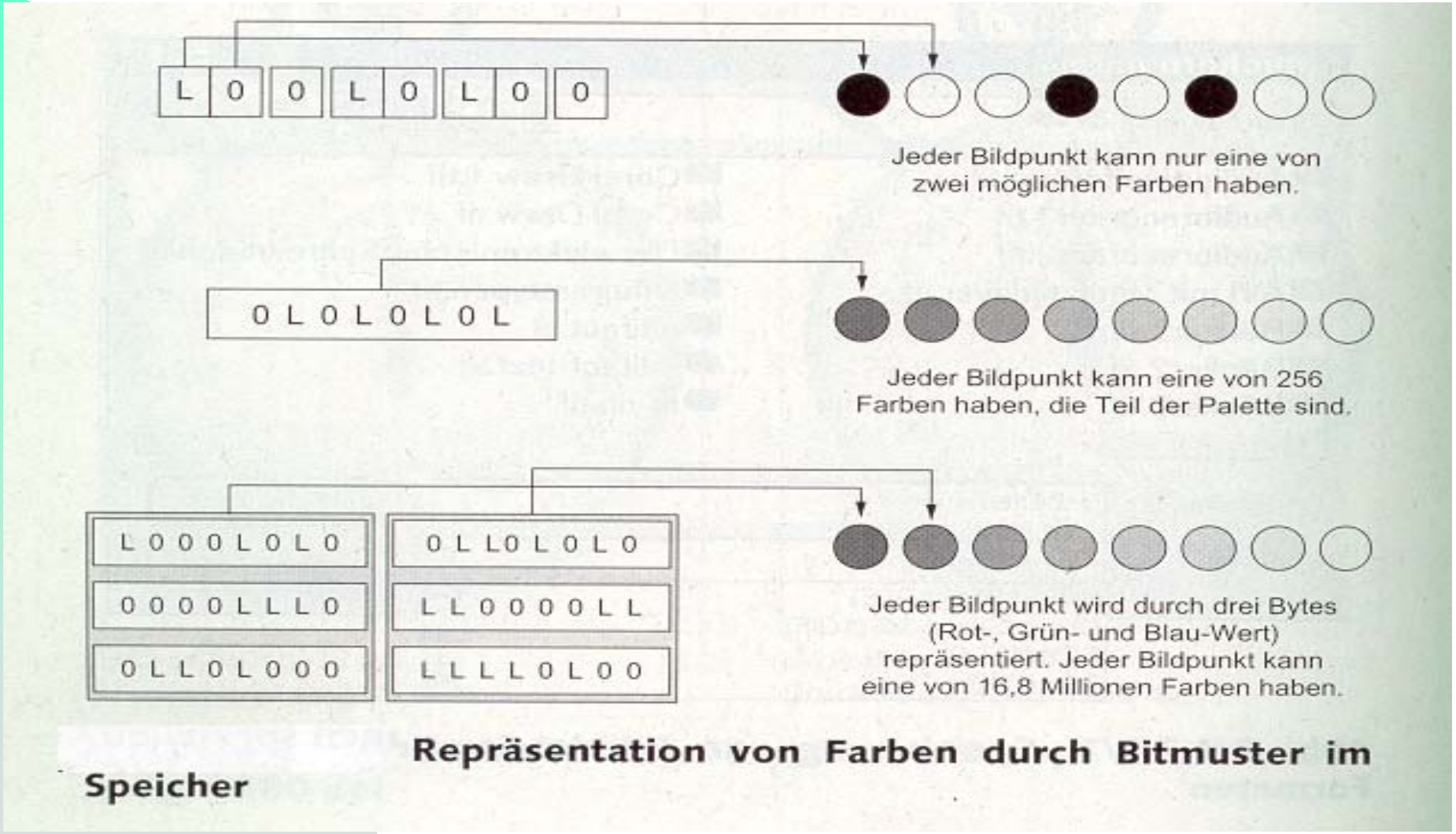
Bildinformation - Festbilder - Pixelgrafik

- Die Pixelgrafik ist eine Matrix von Punkten
- Jeder einzelne Bildpunkt drückt einen Farb- oder Dichtewert aus und kann separat bearbeitet werden.
- Diese Bildart ist gut geeignet, um die kontinuierlichen Graustufen und Farbtöne von Fotos darzustellen.
- Qualität und Datenumfang eines Rasterbildes werden durch die Bildgröße, die Auflösung, die Farbtiefe und den Kompressionsgrad beeinflusst
- Je größer ein Bild und je feiner die Auflösung, desto höher sind der Speicherbedarf und der Zeitbedarf bei der Übertragung

Bildinformation - Festbilder - Pixelgrafik

- Schwarz – Weiß – Bild: Abspeicherung nur 2 Farben d.h. pro Bildpunkt 1 Bit notwendig
- 256 Grautöne bedürfen 8 Bit pro Pixel
- Farbbilder mit RGB-Farbmodell (Rot – Grün – Blau)
- Grundfarben Rot, Grün, und Blau in unterschiedlicher Intensität übereinander projiziert
- Bei hoher Qualität sind pro Farbe 256 Dichtestufen von 0 (keine Farbe) bis 255 (volle Farbe)
- Bei 256 Dichtestufen für die drei Grundfarben sind pro Pixel 24 Bits (3 Bytes) Speicherplatz erforderlich, die $256 \times 256 \times 256 = 2^{24} = 16{,}8$ Millionen Farben (True - Color - Darstellung)

Bildinformation - Festbilder - Pixelgrafik



Repräsentation von Farben durch Bitmuster im Speicher

Bildinformation - Festbilder - Pixelgrafik

- **Annahme: Bild mit 1280x1024 Punkten Auflösung**
  → Speicherbedarf
  1280x1024x3=3.932.160 Bytes = 3,75 MB Speicher
  → Übertragungszeit:
  8 Minuten bei einer 64kBit/s-Verbindung
- **Schwarz-Weiß-Bild benötigt nur 164 KB**
  1280x1024 Bits (1 Bit pro Bildpunkt)
- **Größen beziehen sich auf den Bildspeicher/ Hauptspeicherbedarf**
- **Abspeicherung auf Festplatte mit entsprechender Kompression d.h. nicht alle Bildpunkte werden gespeichert**

Bildinformation - Festbilder - Pixelgrafik

- **Dateiformate von Pixelgrafiken:**
  - BMP (Windows Bitmap)
  - GIF (Graphics Interchange Format von Compuserve
    LZW verlustfrei kompromiert, maximal 256 Farben
  - PNG (Portable Network Graphics)
  - JPEG (Joint Photographic Experts - Group): hohe Kompression bis Faktor 25
  - TIFF (Tagged Image File): verlustfreie Kompression möglich, Einsatz in Designstudios

Bildinformation - Pixelgrafik

- **Speicherbedarf eines Beispielbildes bei verschiedenen Dateiformaten:**
- **Vorlage: 4x2,7 cm, 300 dpi, 480x319px**

Dateiformat	Kompression	Gesamtgröße der Datei in KB
TIFF	**Ohne / LZW**	**476 / 347**
BMP	**Ohne (RLE wenig sinnvoll)**	**449**
PNG	**ZIP/Deflate (24 Bit/8 Bit)**	**296 / 86**
GIF	**LZW, 8 Bit**	**98**
JPG	**Qualität 100, 60, 10**	**154, 41, 12**

Bildinformation - Festbilder - Vektorgrafik

- Vektorgrafik unterschiedet sich grundlegend von Pixelgrafik
- Ein Bild wird nicht als Muster einzelner Punkte, sondern durch mathematisch definierte Objekte aus Linien und/oder Kurven mit Füllungen beschrieben
- Bildobjekte werden mathematisch einzeln definiert und können unabhängig voneinander einzeln bearbeitet (editieren, ändern, verschieben) werden.
- Besonders geeignet für Zeichnungen, Illustrationen und rechnergestützte Konstruktion (CAD) geeignet
- Bekannte Formate: CorelDraw (.cdr), SVG, WMF, EPS

Film, Animation und virtuelle Realität

- **Bewegte Bilder variieren in der Wahrnehmung des Empfängers**
- **Der Eindruck der Bewegung wird durch die rasche Aufeinanderfolge der gezeigten Bilder erreicht**
- **Virtuelle Realität ist ein mittels Echtzeit - Animation nachgebildeter, dreidimensionaler Ausschnitt der realen Welt**
- **Anwendungen der virtuellen Realität sind z.B. Modellsimulationen im Flugzeugbau, Crash-Test im Automobilbau**

Film, Animation und virtuelle Realität

- Je höher die Bildrate, desto besser ist die Qualität von Bewegtbildern
- Die Untergrenze für die Wahrnehmung kontinuierlicher Bewegungen liegt bei etwa 14 Einzelbildern pro Sekunde fps (engl. frames per second)
- Die Fernsehstandards sehen 25 fps (PAL, SECAM in Europa) bzw. 30 fps (NTSC in USA) vor
- Wegen der enormen Datenmenge ist zur Speicherung und Übertragung von Bewegtbildern immer eine Kompression erforderlich
- AVI-Dateiformat bietet üblicherweise eine Auflösung von 160 x 120 Punkten und 15 Bildern pro Sekunde. Für die Wiedergabe ist kein zusätzliches Ausgabegerät erforderlich.
- Quasi - Standard für die Wiedergabe von Filmen ist MPEG

Lernkontrolle

Kennen Sie

- **Vor- und Nachteile unterschiedlicher Zeichencodierungen?**
- **Gründe für die unterschiedliche Speicherung von Zahlenformate?**
- **Realisierungen der Zahlendarstellungen?**
- **Verschiedene Grafikformate mit Ihren Vor- und Nachteilen?**