

### Darstellung von Text

- 7 Bit pro Zeichen genügen ( $2^7 = 128$ )
  - 26 Kleinbuchstaben
  - 26 Großbuchstaben
  - 10 Ziffern
 } **alphanumerische Zeichen**
- Sonderzeichen wie '&', '!', '"'
- nicht druckbare Steuerzeichen, z.B.
  - CR (carriage return = Wagenrücklauf)
  - TAB (Tabulator)
  - BEL (bell = Klingelzeichen)

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 16

### ASCII-Code

- Wie ordnet man den Zeichen je eine Zahl (Code) zu?
- Tabelle mit 128 Zeichen
- Einträge werden durchnummeriert
- jedes Zeichen erhält als Code seine Position in der Tabelle als 7 stellige Binärzahl
 

$b_7b_6b_5b_4b_3b_2b_1b_0$
- das achte Bit eines Bytes wurde ursprünglich als Paritäts-Bit (parity bit) genutzt (s.u.)

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 17

- Tabelle kann im Prinzip beliebig gewählt werden
  - Sender und Empfänger von Informationen müssen aber dieselbe Tabelle verwenden

**American Standard Code for Information Interchange**

- seit 1968 festgelegt
  - 8-Bit-Code, höchstwertiges Bit ist Parity-Bit
- Systematik
  - die 10 Ziffern folgen aufeinander
    - Bits  $b_3b_2b_1b_0$  sind Binärdarstellung des Zahlenwertes
  - Klein- und Großbuchstaben sind jeweils alphabetisch sortiert
- Gibt noch viele andere Tabellen!

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 18

### Parity-Bit

- Parity-Bit  $b_7 = \sum_{i=0}^6 b_i$ 
  - ungerade Parität (*odd parity*) bedeutet:
    - $b_7 = 1 \Leftrightarrow$  Anzahl 1-Bits in  $b_0 \dots b_6$  ist ungerade
  - gerade Parität (*even parity*) bedeutet:
    - $b_7 = 1 \Leftrightarrow$  Anzahl 1-Bits gerade
- 1-Bit-Fehler bei der Datenübertragung können so erkannt werden

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 19

### ASCII-Tabelle

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL				ACK	BEL	BS	HT	LF	VT	FF	CR				
1	DLE						CAN			ESC						
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
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	{	—	}	~	`

- 00 - 1F = Steuerzeichen (*control characters, non-printable characters*)
  - BS = backspace, LF = linefeed, CR = carriage-return, ...

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 20

### Länderspezifische Zeichen

- länderspezifische Zeichen fehlen, z.B.
  - Ä, Ö, Ü, ä, ö, ü, ß, §, Å, æ, ¥
- Lösungsansatz (1986)
  - verzichten auf das Parity-Bit
  - 8. Bit für Verdopplung der Code-Tabelle nutzen
- Vorteil
  - in den meisten europäischen Regionen können alle sprach-spezifischen Zeichen dargestellt werden
- Nachteil
  - verschiedene Regionen interpretieren dieselben Codes unterschiedlich
  - Benutzung verschiedener Code-Tabellen in verschiedenen Regionen
  - Datenaustausch schwierig

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 21

## Unicode

- Standard seit 1993
- 16-Bit Code (65536 Zeichen)
- Unicode definiert eine Code-Tabelle für einen **universellen Zeichensatz**, der alle Schriften der Welt umfassen soll; Unicode deckt bereits die standardisierten Zeichensätze ab
- Unicode 3.0 (seit 2000)
  - 57709 Codes festgelegt (nur noch 7827 frei)
- weitere Informationen, siehe
  - <http://www.unicode.org>

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 22

## Grundbegriffe

- Zeichen (*character*)** ist elementare Texteinheit
  - abhängig von der Sprache
  - abhängig von der Verarbeitungsfunktion: spanisches „ll“ wird für das Sortieren als ein Zeichen betrachtet, aber für Eingabe und Anzeige als zwei Zeichen
  - Abstraktion von graphischen Ausprägungen (Glyphen, *glyphs*); das mit dem ISO-Namen „Latin Capital Letter A“ benannte Zeichen als Abstraktion für die Glyphen

A A A A A

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 23

- Zeichensatz (*character repertoire*)** = Menge, Vorrat von Zeichen; wird normalerweise definiert über die Angabe von Namen für die Zeichen und eine Reihe von typischen Glyphen für die Zeichen
- Schrift, auch Alphabet (*script* oder *alphabet*)** als ein Zeichensatz für eine bestimmte Sprache oder Familie von Sprachen: lateinische Schrift, griechische Schrift, hebräische Schrift, chinesische Schrift

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 24

- Code-Tabelle** für einen Zeichensatz (*code table* oder (*coded*) *character set*) = Tabelle, in der jedes Zeichen eines Zeichensatzes eine Nummer erhält; die Nummern werden in aufsteigender Reihenfolge von 0 an vergeben und können Lücken enthalten; Größe der Code-Tabelle bestimmt die Zahl der Bits, die zur Darstellung der Zeichen-Codes notwendig ist (8 Bit = 256 Werte, 16 Bit = 63.536 Werte)
- Code** oder **Code-Position** eines Zeichens in Bezug auf eine Code-Tabelle (*code set position*, *code point*, *character number*) als die dem Zeichen in der Code-Tabelle zugeordnete Nummer

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 25

- Kodierung** oder **Kodierungsschema (*encoding scheme*, *character encoding*)** als eine Abbildung, die eine Folge von Zeichen in eine Folge von Bytes übersetzt
  - modal vs. nicht-modal (Zeichen-Morphismus)
  - feste vs. variable Code-Länge
- Achtung:** Unicode definiert nur die Code-Tabelle! Unicode selbst definiert noch nicht, wie eine Zeichenkette in eine Byte-Folge abgebildet wird! Das ist Sache des Kodierungsschemas!
  - 16-Bit Code-Tabelle = 65.536 Positionen (mögl. Zeichen)
  - Es gibt Kodierungen UTF-8, UTF-16, UCS2, etc.

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 26

## Standardisierte Code-Tabellen

Sprache	Code-Tabelle	Umfang
Englisch	US-ASCII (ISO 646 IRV:1991)	94
Franz.	ISO 8859-1:1987	191
Chinesisch	GB 2312-80	7.445
Chinesisch	Big 5	13.523
Japanisch	JIS X 0208-1990	6.897
Koreanisch	KS C 5601-1992	8.224
alle	ISO/IEC 10646-1:1993 (Unicode 2.0)	38.885

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 27

### Kodierungen

Kodierung	Code-Tabelle	Länge
Shift-JIS	JIS X 0208-1990	2
EUC-JP	JIS X 0208-1990 and JIS X 0212-1990	2
Big 5	Big 5	2
EUC-KR	KS C 5601-1992	2
UTF-8	Unicode 2.0	≤ 3

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 28

- ### Kodierungsschemata von Unicode
- UCS2:
    - kodiert eine 16-Bit-Binärzahl als zwei Bytes
  - UTF-8:
    - kodiert 16-Bit-Binärzahl mit ein bis drei Bytes
    - kodiert 31-Bit-Binärzahl mit ein bis sechs Bytes
    - transparent für ASCII-Zeichen von 0 bis 127: kodiert in einem Byte mit höchstwertigem Bit 0
    - übrige Binärzahlen kodiert durch Folgen von n Bytes, von denen jedes das höchstwertige Bit auf 1 gesetzt hat; das erste Byte hat n Bits auf 1 und das n+1-te auf 0 gesetzt, die folgenden Bytes beginnen mit 10; die übrigen Positionen dienen der Kodierung der Binärzahl
- G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 29

32-Bit-Binärzahl	Byte-Sequenz UTF-8	Bits
00000000-0000007F	0xxxxxxx	7
00000080-000007FF	110xxxxx 10xxxxxx	11
00000800-0000FFFF	1110xxxx 10xxxxxx 10xxxxxx	16
00080000-001FFFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	21
00200000-03FFFFFF	111110xx 10xxxxxx 10xxxxxx 10xxxxxx	26
04000000-7FFFFFFF	1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx	31

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 30

- Jeder Zeichen-Code muß in dem kürzest-möglichen Block kodiert werden
  - beim Dekodieren muß darauf geachtet werden, daß nur gültige Byte-Sequenzen dekodiert werden  
beispielsweise ist 11000000 10000001 ungültig, da die 32-Bit-Binärzahl 00000001 durch die kürzere Byte-Sequenz 00000001 darstellbar ist
- G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 31

### Darstellung von ganzen Zahlen

- Ein Zahlssystem (*number system*) besteht aus
  - endlich vielen Ziffern (*digits*) und
  - einer Vorschrift,
    - wie Zeichenreihen, die aus diesen Ziffern gebildet wurden, als Zahl-Werte zu interpretieren sind
- Arabische Zahlssysteme zur Basis  $\beta$ 
  - Natürliche Zahl  $z = z_n \dots z_0$  hat den Wert
 
$$z = \sum_{i=0}^n z_i \beta^i$$
 mit  $0 \leq z_i < \beta$

Arithmetica

Pythagoras mit Rechenbrett      Boethius mit arabischen Zahlen

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 32

- Basis  $\beta$  der Darstellung wird **Radix** genannt
  - Namen einiger wichtiger Zahlssysteme
    - Radix 10: Dezimaldarstellung
    - Radix 2: Binärdarstellung
    - Radix 8: Oktalдарstellung
    - Radix 16: Hexadezimaldarstellung
  - Zur Kennzeichnung der Basis wird diese oftmals als Subscript angegeben
    - $7_{10} = 7_8 = 111_2$
    - $9_{10} = 11_8 = 1001_2$
    - $15_{10} = 17_8 = 1111_2$
- G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 33

### ASCII-Zahlen

- Zahlen inmitten von anderem Text werden als ASCII-Codes dargestellt, z.B.
 
$$(42)_{10}: (00110100\ 00110010)_{ASCII}$$
- verschwendet Speicherplatz
  - N Bits für N/8 Ziffern, d.h.  $10^{N/8} = 2^{0.42 \cdot N}$  Zahlen
  - N Bits können aber  $2^N$  verschiedene Zahlen kodieren
- einfache (ziffernweise) Umwandlung von ASCII-Zahlen in Dezimalzahlen und umgekehrt
- arithmetische Operationen sind aufwendig

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 34

### BCD-Zahlen

- BCD-Zahlen (*Binary Coded Decimals*)
  - binär kodierte Dezimalzahlen
  - für eine Ziffer genügen 4 Bit
  - Beispiel
 
$$(42)_{10} = (0100\ 0010)_{BCD}$$
  - immer noch verschwendisch, da mit N Bits nur  $2^{0.83N}$  Zahlen dargestellt werden können
  - einfache (ziffernweise) Umwandlung von Dezimalzahlen in BCD-Zahlen und umgekehrt
  - arithmetische Operationen immer noch aufwendig

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 35

### Binärdarstellungen

- Idee
  - Zahlen mit Stellenwertkodierung darstellen
    - Vorzeichen (falls nicht nur positive Zahlen dargestellt werden sollen)
    - Ziffern
    - Wert der Stelle, an der die Ziffer steht
  - Beispiel Dezimalsystem
 
$$(4711)_{10} = 4 \cdot 10^3 + 7 \cdot 10^2 + 1 \cdot 10^1 + 1 \cdot 10^0$$
  - Analog Binärsystem
 
$$(11010)_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = (26)_{10}$$

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 36

### Vorzeichenlose (positive) Zahlen

$w : \{0, 1\}^n \mapsto \mathbb{N}_0$  natürliche Zahlen mit 0

$$b_{n-1} \dots b_1 b_0 \mapsto w(b_{n-1} \dots b_1 b_0) = \sum_{i=0}^{n-1} b_i 2^i$$

- $2^n$  Zahlen mit n Bits darstellbar:  $0, \dots, 2^n - 1$
- jede Zahl hat eine eindeutige Kodierung in n Bits

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 37

### Umwandlung ins Binärsystem

- Umwandlung binär  $\rightarrow$  dezimal
  - trivial
  - Addition der Zweierpotenzen
- Umwandlung: dezimal  $\rightarrow$  binär
  - Zusammensetzen aus Zweierpotenzen
  - Umwandlung von "links nach rechts" durch Bestimmung der größten Zweierpotenz, die kleiner oder gleich der umzuwandelnden Zahl ist
    - suche aus Tabelle die größte Zweierpotenz, die kleiner oder gleich der Zahl ist
    - subtrahiere sie von der Zahl
    - verfahre genauso mit der Differenz, bis sie 0 ist

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 38

### Divisionsmethode

- Besser:
  - Umwandlung von "rechts nach links" mit ganzzahliger Division durch 2 mit Rest

$$w = \sum_{i=0}^{n-1} b_i 2^i = \sum_{i=1}^{n-1} b_i 2^i + b_0 = 2 \left( \sum_{i=0}^{n-2} b_{i+1} 2^i \right) + b_0$$

- $b_0$  ist der ganzzahlige Rest bei der Division durch 2
- Verfahre ebenso mit dem Ergebnis der Division

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 39

- exakte ganzzahlige Division
  - bei der Division einer ganzen Zahl  $z$  durch einen ganzzahligen Divisor  $d$  entsteht ein ganzzahliger Quotient  $q$  und ein ganzzahliger Rest  $r$  mit
 
$$z = q \cdot d + r \quad \text{mit} \quad 0 \leq r < d$$
  - Operationen, die den Quotienten und den Rest bestimmen
 
$$q = z \text{ div } d$$

$$r = z \text{ mod } d$$

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 40

Beispiel für Divisionsmethode

$z$	$z \text{ div } 2$	$z \text{ mod } 2$
42	21	0
21	10	1
10	5	0
5	2	1
2	1	0
1	0	1
0	0	0
0	0	0

- Resultat:  $(42)_{10} = (101010)_2$

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 41

Oktalsystem, Hexadezimalsystem

- Oktalsystem
  - Darstellung der Zahlen zur Basis 8
  - Ziffern 0, ..., 7
 
$$(d_{n-1}d_{n-2} \dots d_0)_8 = d_{n-1}8^{n-1} + d_{n-2}8^{n-2} + \dots + d_08^0$$
- Hexadezimalsystem
  - Darstellung der Zahlen zur Basis 16
  - Hexziffern: 0, ..., 9, A, ..., F
 
$$(d_{n-1}d_{n-2} \dots d_0)_{16} = d_{n-1}16^{n-1} + d_{n-2}16^{n-2} + \dots + d_016^0$$
- Vorteil der Basen 8 und 16
  - triviale Umrechnungen in und aus dem Binärsystem durch Zusammenfassen der entsprechenden Bits

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 42

Umrechnung: binär ↔ oktal

o.B.d.A. sei  $n$  ein Vielfaches von 3

$$z = (b_{n-1} \dots b_1 b_0) = \sum_{i=0}^{n-1} b_i 2^i$$

$$= \underbrace{(b_{n-1} 2^{n-1} + \dots + b_4 2^4 + b_3 2^3)}_{(b_{n-1} 2^2 + \dots + b_4 2^2 + b_3 2^0) \cdot 8^{n/3-1}} + \underbrace{(b_5 2^5 + b_4 2^4 + b_3 2^3)}_{(b_5 2^2 + b_4 2^2 + b_3 2^0) \cdot 8^1} + \underbrace{(b_2 2^2 + b_1 2^2 + b_0 2^0)}_{(b_2 2^2 + b_1 2^2 + b_0 2^0) \cdot 8^0}$$

$$= \sum_{i=0}^{n/3-1} o_i 8^i = (o_{n/3-1} \dots o_1 o_0)_8$$

binär

011 001 101 100 010

3      1      5      4      2

oktal

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 43

Addition von Binärzahlen

- Zahlen können in jedem Zahlensystem wie gewohnt addiert werden

	Dezimal	Binär	Oktal	Hexadezimal
	2 7 5 2	0 1 0 0 1 0	2 7 5 2	2 7 C A
+	4 2 6 1	1 0 0 1 1 1	4 2 6 1	A F 9 3
=	7 0 1 3	1 1 1 1 0 0 1	7 2 3 3	D 7 5 D

- der Computer verwendet meist feste Wortbreiten
  - ein Übertrag (*carry*) über die höchstwertige Stelle hinaus (*overflow*) wird häufig ignoriert
  - "die Rechnung stimmt nur modulo  $2^n$ "

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 44

Vorzeichenbehaftete ganze Zahlen

- höchstwertiges Bit kodiert Vorzeichen

$w : \{0, 1\}^n \rightarrow \mathbb{Z}$  ganze Zahlen

$$(b_{n-1} b_{n-2} \dots b_0) \mapsto w(b_{n-1} b_{n-2} \dots b_0) = (b_{n-1}, \sum_{i=0}^{n-2} b_i 2^i)$$

Vorzeichenbit

gesonderte Behandlung des Vorzeichens

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 45

### Darstellung als Betrag und Vorzeichen

- Zahlenwert ergibt sich aus
 
$$w(b_{n-1}b_{n-2}\dots b_0) = \begin{cases} + \sum_{i=0}^{n-2} b_i 2^i & , \text{ falls } b_{n-1} = 0 \\ - \sum_{i=0}^{n-2} b_i 2^i & , \text{ falls } b_{n-1} = 1 \end{cases}$$
- Beispiele für n=6
  - $w(011010) = +(1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0) = (+26)_{10}$
  - $w(111010) = -(1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0) = (-26)_{10}$

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 46

- Zahlenbereich ist **symmetrisch**
  - Wertebereich:  $-(2^{n-1}-1), \dots, +2^{n-1}-1$
  - insgesamt  $2^n-1$  Zahlen
- Darstellung der 0 ist **redundant**
  - $w(0000\dots 0) = w(1000\dots 0) = (0)_{10}$
- Vergleichsoperationen schwer zu implementieren

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 47

### Addition bei Betrag und Vorzeichen

- Addition negativer Zahlen
  - komplizierter als normale binäre Addition

$(0011)_2$	$(3)_{10}$
$+(1001)_2$	$+(-1)_{10}$
$(1100)_2 \neq$	$(2)_{10}$

- technisch zwar realisierbar
- die folgende **Zweierkomplementdarstellung** vermeidet aber alle genannten Nachteile

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 48

### Zweierkomplementdarstellung

- Nutze aus, daß  $-z \equiv 2^n - z \pmod{2^n}$
- Wert in Zweierkomplementdarstellung
 
$$w(b_{n-1}b_{n-2}\dots b_0) = -b_{n-1}2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i$$
- für  $b_{n-1}=0$  erhält man die positiven Zahlen wie gehabt
- für  $b_{n-1}=1$  wird  $2^{n-1}$  vom Wert subtrahiert
- Zahlenbereich:  $-2^{n-1}, \dots, 2^{n-1}-1$
- Zahlenbereich **nicht symmetrisch!**
- Darstellung der 0 **nicht redundant**
- bevorzugte Methode zur Darstellung ganzer Zahlen in heutigen Computern

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 49

### Zweierkomplement im Zahlenkreis

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 50

### Zweierkomplement, negative Werte

- Alternative Bestimmung des Wertes einer negativen Zahl
  - Invertiere alle Bits
  - Bestimme den Wert der nun positiven Zahl
  - Addiere eine 1
  - Das Ergebnis ist der Betrag der ursprünglichen negativen Zahl
- Beispiel
  - $(1011001)_2$
  - 1. Invertieren:  $(0100110)_2$
  - 2.  $= 32+4+2 = 38$
  - 3.  $= 38+1 = 39$
  - 4. Ergebnis:  $-39$
- Laut Formel
  - $(1011001)_2$
  - 1.  $= -64+16+8+1$
  - 2.  $= -64+25$
  - 3.  $= -39$
- Bitweises vertauschen allein definiert das **Einerkomplement**

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 51

### Beweis für alternative Berechnung

Invertierung aller Bits entspricht Subtraktion vorzeichenloser Zahlen von  $(11111...1)_2$ , denn z.B.

$$(011001...1)_2 + (100110...0)_2 = (11111...1)_2$$

Also gilt für  $b_{n-1} = 1$

$$\begin{aligned} \sum_{i=0}^{n-1} b_i 2^i + 1 &= (2^n - 1) - \sum_{i=0}^{n-1} b_i 2^i + 1 \\ &= 2^n - b_{n-1} 2^{n-1} - \sum_{i=0}^{n-2} b_i 2^i = 2^{n-1}(2 - 1) - \sum_{i=0}^{n-2} b_i 2^i \\ &= -(-2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i) \end{aligned}$$

Alternativer Bew.:  $x - y \equiv x + (2^n - y) \equiv x + \bar{y} \pmod{2^n}$

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 52

### Addition in Zweierkomplementdarstellung

- Addition funktioniert wie normale Binäraddition vorzeichenloser Zahlen
- Übertrag aus der höchstwertigen Stelle ignorieren
- Beispiele

$(0011)_2$	$(3)_{10}$	$(1100)_2$	$(-4)_{10}$
$+(1001)_2$	$+(-7)_{10}$	$+(1101)_2$	$+(-3)_{10}$
$(1100)_2$	$(-4)_{10}$	$1(1001)_2$	$(-7)_{10}$

- Beweis: selbst durch Fallunterscheidung

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 53

### Einbettung in längere Zahldarstellungen

- Zahlen müssen sich an die Wortlängen des verwendeten Computers orientieren
- Problem: Einbettung einer Zahl in eine Darstellung mit größerer Wortlänge

$$w(b_{n-1}b_{n-2} \dots b_0)$$

$$= \begin{cases} w(b_{n-1}0 \dots 0b_{n-2} \dots b_0) & \text{Betrag + Vorzeichen} \\ w(b_{n-1} \dots b_{n-1}b_{n-2} \dots b_0) & \text{1er- oder 2er-Kompl.} \end{cases}$$

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 54

### Beweis zur Einbettung

- nur für Zweierkomplement
- zu zeigen:

- Fall:  $b_{n-1} = 0$   $w(b_{n-1}b_{n-2} \dots b_0) = w(b_{n-1}b_{n-2} \dots b_0)$   
- positive Zahl, trivial
- Fall:  $b_{n-1} = 1$

$$\begin{aligned} &w(1b_{n-2} \dots b_0) \\ &= -1 \cdot 2^n + 1 \cdot 2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i = 2^{n-1}(-2 + 1) + \sum_{i=0}^{n-2} b_i 2^i \\ &= -1 \cdot 2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i \\ &= w(1b_{n-2} \dots b_0) \end{aligned}$$

G. Zachmann Informatik 1 - WS 05/06 Repräsentation von Daten 55