
— Informatik I —

Modul 2: Rechnerarithmetik



**Universität
Zürich^{UZH}**



Modul 2: Rechnerarithmetik (1)

- ❑ Zahlensysteme
- ❑ Zahlendarstellung
- ❑ Grundrechenarten
- ❑ Zeichendarstellung

Rechnerarithmetik

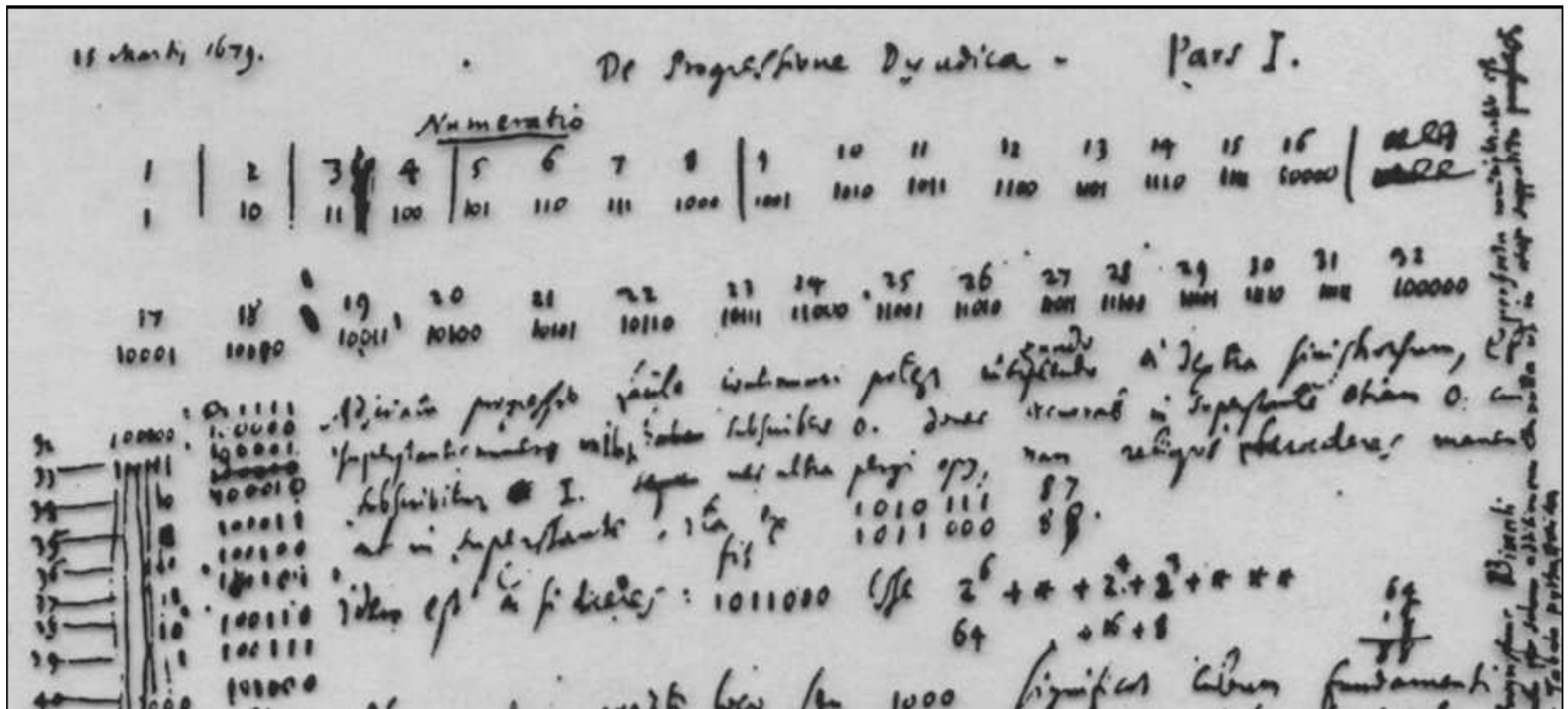
- Die Rechnerarithmetik soll als Beispiel eingeführt werden, wie größere Informationseinheiten elektronisch verarbeitet werden.
- Hierzu werden zunächst die formalen Grundlagen erarbeitet:
 - Zahlensysteme
 - Zahlendarstellungen
 - Grundrechenarten
 - Zeichendarstellungen
 - Boole'sche Algebra (Modul 3)
- Die Rechnerarithmetik bildet gleichzeitig die entscheidende Grundlage, beliebige Informationen und Informationseinheiten zu behandeln, sprich zu berechnen, umzuwandeln, zu speichern oder zu kommunizieren.

Zahlensysteme – Römische Zahlen

I	1		X	10		C	100
II	2		XX	20		CC	200
III	3		XXX	30		CCC	300
IV	4		XL	40		CD	400
V	5		L	50		D	500
VI	6		LX	60		DC	600
VII	7		LXX	70		DCC	700
VIII	8		LXXX	80		DCCC	800
IX	9		XC	90		CM	900
						M	1000

Zahlensysteme – Leibniz'sche Dualzahlen

- Leibniz-Traktat aus dem Jahre 1679
- Vermutlich kommen Ideen zum Dualzahlensystem aus China



Formale Grundlagen

- ❑ Menschen rechnen gewöhnlich im Dezimalzahlensystem.
- ❑ Rechner rechnen gewöhnlich im Dualzahlensystem.

→ Eine Konvertierung ist erforderlich

- ❑ Daneben werden weitere Zahlensysteme wie Oktalzahlensystem oder Hexadezimalzahlensystem (eigentlich: Sedezimal) zur kompakteren Darstellung der sehr langen Dualzahlen verwendet.

→ Es ist notwendig, die Zusammenhänge und mathematischen Grundlagen dieser Zahlensysteme zu verstehen.

Zahlensysteme (1)

- Gängigste Form: Stellenwertsysteme
- Zahlendarstellung in Form einer Reihe von Ziffern z_i , wobei das Dezimalkomma (-punkt) rechts von z_0 plaziert sei:

$z_n z_{n-1} \dots z_1 z_0 , z_{-1} z_{-2} \dots z_{-m}$ z.B. 1234,567

- Jeder Position i der Ziffernreihe ist ein Stellenwert zugeordnet, der eine Potenz b^i der Basis b des Zahlensystems ist.

- Der Wert X_b der Zahl ergibt sich dann als Summe der Werte aller Einzelstellen $z_i b^i$:

$$X_b = z_n b^n + z_{n-1} b^{n-1} + \dots + z_1 b + z_0 + z_{-1} b^{-1} + \dots + z_{-m} b^{-m} = \sum_{i=-m}^n z_i b^i$$

Zahlensysteme (2)

□ Interessante Zahlensysteme in der Informatik

b	Zahlensystem	Zahlenbezeichnung
2	Dualsystem	Dualzahl
8	Oktalsystem	Oktalzahl
10	Dezimalsystem	Dezimalzahl
16	Hexadezimalsystem (Sedezimalsystem)	Hexadezimalzahl (Sedezimalzahl)

- Hexadezimalsystem: Die „Ziffern“ 10 bis 15 werden mit den Buchstaben A bis F dargestellt.
- Dualsystem: Wichtigstes Zahlensystem im Rechner
- Oktal- und Hexadezimalsystem: Leicht ins Dualsystem umwandelbar, besser zu verstehen als lange 0-1-Kolonnen.

Zahlensysteme (3)

- Eine einzelne Binärstelle (0 oder 1), die ein Rechner speichert, wird als **Bit** bezeichnet. Das ist die Abkürzung für „*Binary digit*“, also Binärziffer. Es handelt sich um die kleinste Informationseinheit, die ein Computer verarbeiten kann.
- Auch beim Dualsystem handelt es sich um ein Positionssystem. Der Wert einer Position ist hier jedoch eine Potenz von 2:

$$10011 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 16 + 0 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 \\ = 19 \text{ (im Zehnersystem)}$$

- Beispiele:

$(10011)_2$	$=$	$(19)_{10}$	oder :	$10011_{(2)}$	$=$	$19_{(10)}$
$(1011)_2$	$=$	$(11)_{10}$	oder :	1011_2	$=$	$11_{(10)}$
$(11010110)_2$	$=$	$(214)_{10}$	oder :	$11010110_{(2)}$	$=$	$214_{(10)}$

Zahlensysteme (4)

Dual	Oktal	Dezimal	Hexadezimal	Zwölfersystem
0	0	0	0	0
1	1	1	1	1
10	2	2	2	2
11	3	3	3	3
100	4	4	4	4
101	5	5	5	5
110	6	6	6	6
111	7	7	7	7
1000	10	8	8	8
1001	11	9	9	9
1010	12	10	a	a
1011	13	11	b	b
1100	14	12	c	10
1101	15	13	d	11
1110	16	14	e	12
1111	17	15	f	13
10000	20	16	10	14
10001	21	17	11	15

Zahlensysteme (5)

- Bei gebrochenen Zahlen trennt ein Punkt (Komma im Deutschen) in der Zahl den ganzzahligen Teil der Zahl vom gebrochenen Teil (Nachkommateil). Solche Zahlen lassen sich durch folgende Summenformel beschreiben:

$$n = \sum_{i=-M}^{N-1} b_i \cdot B^i \quad \text{wobei Folgendes gilt:}$$

- $B =$ Basis des Zahlensystems ($B \in \mathbb{N}, B \geq 2$)
- $b =$ Ziffern ($b_i \in \mathbb{N}_0, 0 \leq b_i < B$)
- $N =$ Anzahl der Stellen vor dem Punkt (Komma)
- $M =$ Anzahl der Stellen nach dem Punkt (Komma)

Konvertierung: Euklidischer Algorithmus

- Umwandlung vom Dezimalsystem in ein System zur Basis b

- 1. Methode: Euklidischer Algorithmus:

$$\begin{aligned} Z &= z_n 10^n + z_{n-1} 10^{n-1} + \dots + z_1 10 + z_0 + z_{-1} 10^{-1} + \dots + z_{-m} 10^{-m} \\ &= y_p b^p + y_{p-1} b^{p-1} + \dots + y_1 b + y_0 + y_{-1} b^{-1} + \dots + y_{-q} b^{-q} \end{aligned}$$

Die Ziffern werden sukzessive, beginnend mit der höchstwertigen Ziffer, berechnet.

- 1. Schritt: Berechne p gemäß der Ungleichung $b^p \leq Z < b^{p+1}$ (setze $i = p$)
- 2. Schritt: Ermittle y_i und den Rest R_i durch Division von Z_i durch b^i : $y_i = Z_i \text{ div } b^i$; $R_i = Z_i \text{ mod } b^i$;
- 3. Schritt: Wiederhole 2. Schritt für $i = p-1, \dots$ und ersetze dabei nach jedem Schritt Z durch R_i , bis $R_i = 0$ oder bis b^i (und damit der Umrechnungsfehler) gering genug ist.

Euklidischer Algorithmus: Beispiel

Umwandlung von $15741,233_{10}$ ins Hexadezimalsystem:

1. Schritt: $16^3 \leq 15741,233 < 16^4 \rightarrow$ höchste Potenz 16^3
2. Schritt: $15741,233 : 16^3 = 3$ Rest 3453,233
3. Schritt: $3453,233 : 16^2 = D$ Rest 125,233
4. Schritt: $125,233 : 16 = 7$ Rest 13,233
5. Schritt: $13,233 : 1 = D$ Rest 0,233
6. Schritt: $0,233 : 16^{-1} = 3$ Rest 0,0455
7. Schritt: $0,0455 : 16^{-2} = B$ Rest 0,00253
8. Schritt: $0,00253 : 16^{-3} = A$ Rest 0,000088593
9. Schritt: $0,000088593 : 16^{-4} = 5$ Rest 0,000012299
(\rightarrow Fehler)

$\rightarrow 15741,233_{10} \approx 3D7D,3BA5_{16}$

Konvertierung: Horner Schema

- Umwandlung vom Dezimalsystem in ein Zahlensystem zur Basis b
- 2. Methode: Abwandlung des Horner Schemas
- Hierbei müssen der ganzzahlige und der gebrochene Anteil getrennt betrachtet werden.
- Umwandlung des ganzzahligen Anteils:
- Eine ganze Zahl $X_b = \sum_{i=0}^n z_i b^i$ kann durch fortgesetztes Ausklammern auch in folgender Form geschrieben werden:

$$X_b = ((\dots(((y_n b + y_{n-1}) b + y_{n-2}) b + y_{n-3}) b \dots) b + y_1) b + y_0$$

Horner Schema: Beispiel

- Die gegebene Dezimalzahl wird sukzessive durch die Basis **b** dividiert.
- Die jeweiligen ganzzahligen Reste ergeben die Ziffern der Zahl X_b in der Reihenfolge von der niedrigstwertigen zur höchstwertigen Stelle.

Wandle 15741_{10} ins Hexadezimalsystem um:

$$15741_{10} : 16 = 983 \quad \text{Rest } 13 \quad (D_{16})$$

$$983_{10} : 16 = 61 \quad \text{Rest } 7 \quad (7_{16})$$

$$61_{10} : 16 = 3 \quad \text{Rest } 13 \quad (D_{16})$$

$$3_{10} : 16 = 0 \quad \text{Rest } 3 \quad (3_{16})$$

$$\rightarrow 15741_{10} = 3D7D_{16}$$

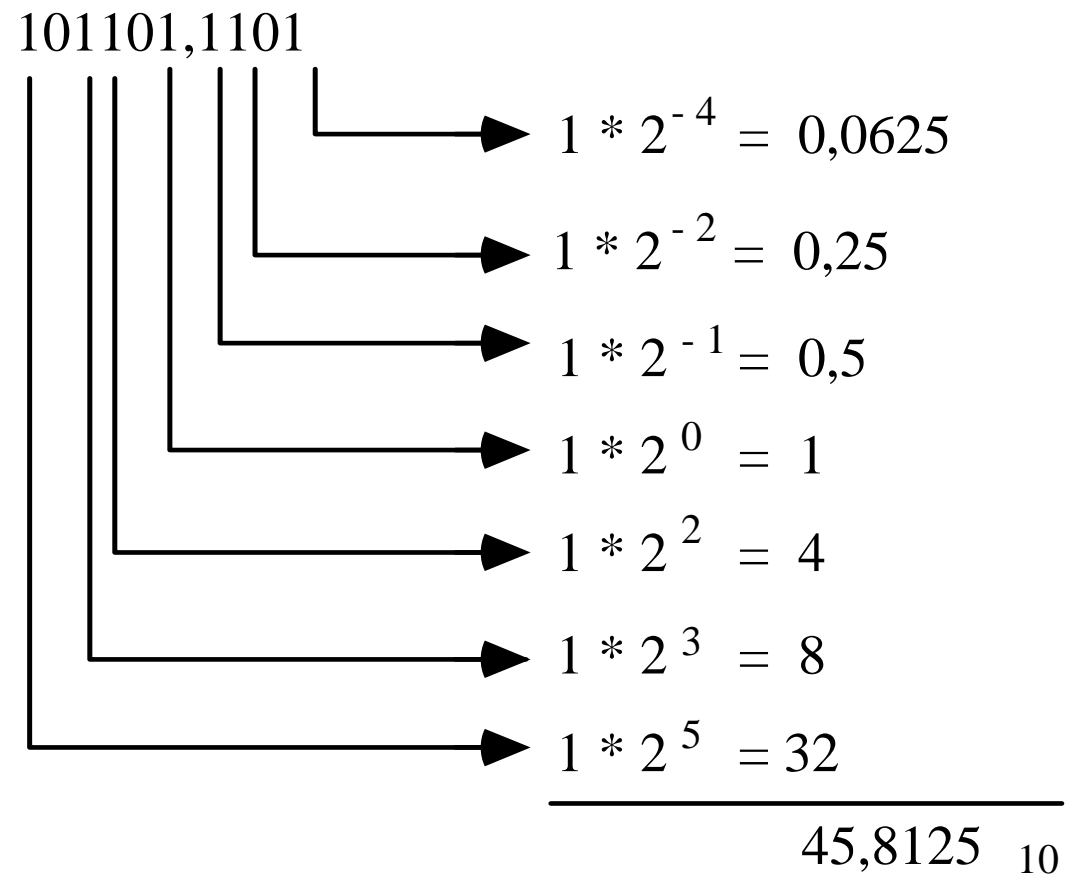
Umwandlung: Basis $b \rightarrow$ Dezimalsystem

- Die Werte der einzelnen Stellen der umzuwandelnden Zahl werden in dem Zahlensystem, in das umgewandelt werden soll, dargestellt und nach der Stellenwertgleichung aufsummiert.
- Der Wert X_b der Zahl ergibt sich dann als Summe der Werte aller Einzelstellen $z_i b^i$:

$$X_b = z_n b^n + z_{n-1} b^{n-1} + \dots + z_1 b + z_0 + z_{-1} b^{-1} + \dots + z_{-m} b^{-m} = \sum_{i=-m}^n z_i b^i$$

Beispiel

Konvertiere $101101,1101_2$ ins Dezimalsystem



Umwandlung beliebiger Stellenwertsysteme

- Man wandelt die Zahl ins Dezimalsystem um und führt danach mit Methode 1 oder 2 die Wandlung ins Zielsystem durch.
- Spezialfall:
 - Ist eine Basis eine Potenz der anderen Basis, können einfach mehrere Stellen zu einer Ziffer zusammengefasst werden oder eine Stelle kann durch eine Folge von Ziffern ersetzt werden.

□ Wandlung von $0110100,110101_2$ ins Hexadezimalsystem

□ $2^4 = 16 \Rightarrow 4$ Dualstellen $\rightarrow 1$ Hexadezimalstelle

dual

0110100,110101



00110100,11010100



hexadezimal

3 4 , D 4

Ergänzen von Nullen zur
Auffüllung auf Vierergruppen

Spezialfall: Umwandlungen Dual-/Oktalsystem

□ Konvertieren zwischen Dual- und Oktalsystem

- Neben dem Dualsystem ist in der Informatik noch das Oktalsystem wichtig, da es in einer engen Beziehung zum Dualsystem steht. Es gilt nämlich: $2^3 = 8$ (Basis des Oktalsystems).
- Um eine im Dualsystem dargestellte Zahl ins Oktalsystem zu konvertieren, bildet man von rechts beginnend so genannte *Dualtriaden* (Dreiergruppen).

110	111	001	110	010		Dualzahl
6	7	1	6	2		Oktalzahl

□ Konvertieren zwischen Dual- und Oktalsystem

- Bei der Umwandlung einer Oktalzahl in ihre Dualdarstellung geht man den umgekehrten Weg.

3	6	1	4		Oktalzahl
011	110	001	100		Dualzahl

- Es ist offensichtlich, dass ein Mensch sich die Zahl $3614_{(8)}$ wesentlich leichter merken und damit umgehen kann, als $011110001100_{(2)}$.

Modul 2: Rechnerarithmetik (1)

- ❑ Zahlensysteme
- ❑ Zahlendarstellung
- ❑ Grundrechenarten
- ❑ Zeichendarstellung

Darstellung negativer Zahlen

- ❑ Für die Darstellung negativer Zahlen in Rechnern werden vier verschiedene Formate benutzt :
- ❑ Darstellung mit Betrag und Vorzeichen
- ❑ Stellenkomplement-Darstellung (Einerkomplement-Darstellung)
- ❑ Zweierkomplement-Darstellung
- ❑ Offset-Dual-Darstellung / Exzeß-Darstellung

Darstellung mit Betrag und Vorzeichen

- Eine Stelle wird als Vorzeichenbit benutzt.
- Ist das am weitesten links stehende Bit (MSB, most significant bit):
 - MSB = 0 → positive Zahl
 - MSB = 1 → negative Zahl

Beispiel:

$$\begin{array}{rcl} 0001\ 0010 & = & +18 \\ 1001\ 0010 & = & -18 \end{array}$$

- **Nachteile:**
 - Bei Addition und Subtraktion müssen die Vorzeichen der Operanden gesondert betrachtet werden.
 - Es gibt zwei Repräsentationen der Zahl 0 (mit positivem und mit negativem Vorzeichen)

Stellenkomplement / Einerkomplement (1)

- Stellenkomplement der entsprechenden positiven Zahl.
- Um eine Zahl zu negieren, wird jedes Bit der Zahl komplementiert.
- Dies entspricht dem Einerkomplement:

Komplementbildung

$$z_{ek} = (2^n - 1) - z$$

Bsp: $4 = 0100_2 \quad \rightarrow \quad -4 = 1011_{ek}$

$$-4 = 2^4 - 1 - 4 = 11_{10} = 1011_2$$

- Negative Zahlen sind wiederum durch ein gesetztes Bit in der ersten Stelle charakterisiert.
- Vorteil gegenüber der Darstellung mit Vorzeichenbit:
 - Erste Stelle bei Addition und Subtraktion muß nicht gesondert betrachtet werden.
 - Aber: **Es gibt weiterhin zwei Darstellungen der Null**

Stellenkomplement / Einerkomplement (2)

- Regeln für die Bildung eines Einerkomplements
 - Ist das 1. Bit mit 1 besetzt, so handelt es sich um eine negative Zahl (eventuell die „negative“ Null 111...111).
 - Der Wert einer negativen Zahl wird dann im Einer-Komplement dargestellt. Einer-Komplement zu einem Wert bedeutet, dass zunächst jedes einzelne Bit invertiert (umgedreht) wird.
 - Führt die Addition des Komplements zum Überlauf einer 1, muß zum Ergebnis diese 1 hinzuaddiert werden („*Einer-Rücklauf*“).

Einer-Komplement zu $(13)_{10} = (01101)_2$:
Invertieren von 13 (-13): 10010
Dualdarstellung von 9: 01001
+ -13 : 10010

= -4 : 11011

Zweierkomplement-Darstellung (1)

□ Man addiert nach der Stellenkomplementierung noch eine 1

□ Man erhält so das Zweierkomplement: $z_{zk} = 2^n - z$

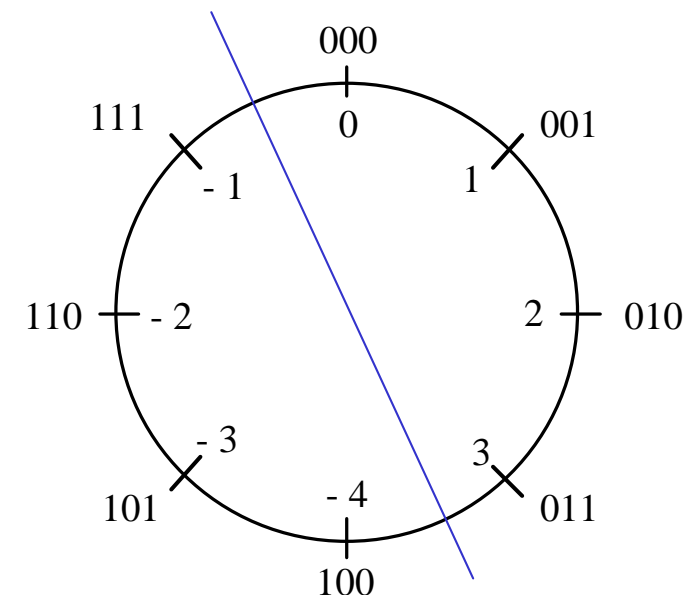
0 . . . 0 → Einerkomplement 1 . . . 1

→ Zweierkomplement 0 . . . 0

□ **Nachteil:**

- Unsymmetrischer Zahlenbereich. Die kleinste negative Zahl ist betragsmäßig um 1 größer als die größte positive Zahl

□ 3-Bit-Zweierkomplementzahlen (Beispiel):



Zweierkomplement-Darstellung (2)

- Alle anderen negativen Zahlen werden um 1 verschoben, das MSB bleibt aber gleich 1.
- Aus der ersten Stelle kann das Vorzeichen der Zahl abgelesen werden
- Aus dieser Konstruktion ergibt sich der Stellenwert des MSB einer Zweierkomplementzahl mit $n+1$ Bit zu -2^n :

$z_n z_{n-1} \dots z_0$ hat den Wert:

$$Z = -z_n \cdot 2^n + z_{n-1} \cdot 2^{n-1} + \dots + z_0$$

Beispiel

Die Zahl -77_{10} soll mit 8 Bit dargestellt werden

$$77_{10} = 0100\ 1101_2$$

Bitweise komplementieren

Mit Vorzeichenbit : $-77 = 1100\ 1101_2$

Einerkomplement : $-77 = 1011\ 0010_2$

Zweierkomplement : $-77 = 1011\ 0011_2$

Addition von 1

Offset-Dual- (Exzeß-)Darstellung

- ❑ Wird hauptsächlich bei der Exponenten-Darstellung von Gleitkommazahlen benutzt.
- ❑ Die Darstellung einer Zahl erfolgt in Form ihrer **Charakteristik (bias)**.
- ❑ Der gesamte Zahlenbereich wird durch Addition einer Konstanten (Exzeß, Offset) so nach oben verschoben, daß die kleinste (negative) Zahl die Darstellung **0...0** erhält.
- ❑ Bei **n** Stellen ist der Offset daher **2^{n-1}**
- ❑ Der Zahlenbereich ist hier auch asymmetrisch.

Zusammenfassung der Möglichkeiten

Darstellung mit				
Dezimalzahl	Betrag + Vorzeichen	Einerkomplement	Zweierkomplement	Charakteristik
-4	---	---	1 0 0	0 0 0
-3	1 1 1	1 0 0	1 0 1	0 0 1
-2	1 1 0	1 0 1	1 1 0	0 1 0
-1	1 0 1	1 1 0	1 1 1	0 1 1
0	1 0 0, 0 0 0	1 1 1, 0 0 0	0 0 0	1 0 0
1	0 0 1	0 0 1	0 0 1	1 0 1
2	0 1 0	0 1 0	0 1 0	1 1 0
3	0 1 1	0 1 1	0 1 1	1 1 1

Fest- und Gleitkommazahlen

- Zahlendarstellung auf dem Papier:

Ziffern 0 .. 9

Vorzeichen + -

Komma (Punkt) , .

- Zahlendarstellung im Rechner:

Binärziffern 0, 1

➔ spezielle Vereinbarungen für die Darstellung von Vorzeichen und Komma/Punkt im Rechner sind erforderlich

- Darstellung des **Vorzeichens**:

– Wurde im vorigen Abschnitt behandelt

- Darstellung des **Kommas** mit zwei Möglichkeiten:

– Festkommadarstellung

– Gleitkommadarstellung

Festkomma-Zahlen (1)

- Vereinbarung:
 - Das Komma sitzt innerhalb des Maschinenwortes, das eine Dualzahl enthalten soll, an einer festen Stelle.

- Meist setzt man das Komma hinter die letzte Stelle.

- Andere Zahlen können durch entsprechende Maßstabsfaktoren in die gewählte Darstellungsform überführt werden.

- Negative Zahlen:
 - Meist Zweierkomplement-Darstellung.

- Festkomma-Darstellungen werden heute hardwareseitig nicht mehr verwendet, jedoch bei der Ein- oder Ausgabe!

Festkomma-Zahlen (2)

- Datentyp "integer" (Ganzzahlen) ist ein spezielles Festkommaformat.
- Manche Programmiersprachen erlauben die Definition von Ganzzahlen unterschiedlicher Länge.
- Beispiel "C": "short int", "int", "long int", "unsigned"

Datentyp	DEC-VAX (einer der Urahnen)		IBM-PC, Apple Macintosh	
	Anzahl der Bits	Zahlenbereich	Anzahl der Bits	Zahlenbereich
short int	16	$-2^{15} \dots 2^{15}-1$	16	$-2^{15} \dots 2^{15}-1$
int	32	$-2^{31} \dots 2^{31}-1$	16	$-2^{15} \dots 2^{15}-1$
long int	32	$-2^{31} \dots 2^{31}-1$	32	$-2^{31} \dots 2^{31}-1$

Gleitkomma-Darstellung (1)

- Zur Darstellung von Zahlen, die betragsmäßig sehr groß oder sehr klein sind, verwendet man die **Gleitkommadarstellung**.
- Sie entspricht einer halblogarithmischen Form

$$X = \pm \text{Mantisse} \cdot b^{\text{Exponent}}$$

- Die Basis b ist für eine bestimmte Gleitkomma-Darstellung fest (meist 2 oder 16) und braucht damit nicht mehr explizit repräsentiert zu werden.
- Gleitkommazahlen werden meist *nicht* im Zweierkomplement, sondern mit **Betrag** und **Vorzeichen** dargestellt.

Gleitkomma-Darstellung (2)

- Bei der **Mantisse** ist die Lage des Kommas wieder durch Vereinbarung festgelegt (meist links vom MSB).
- Der **Exponent** ist eine ganze Zahl, die in Form ihrer Charakteristik dargestellt wird.
- Für die Charakteristik und die Mantisse wird im Rechner eine **feste Anzahl von Speicherstellen** festgelegt.
- Die Länge der Charakteristik $y-x$ bestimmt die Größe des Zahlenbereichs.
- Die Länge der Mantisse x legt die Genauigkeit der Darstellung fest.



$$\text{Dezimalzahl} = (-1)^{Vz} * (0, \text{Mantisse}) * b^{\text{Exponent}}$$

$$\text{Exponent} = \text{Charakteristik} - b^{(y-1) - x}$$

Normalisierung

- ❑ Legt man für die Zahl 0 ein spezielles Bitmuster fest, ist die erste Stelle der Mantisse in normalisierter Form immer gleich 1.
- ❑ Die erste Stelle der Mantisse braucht im Maschinenformat gar nicht erst dargestellt zu werden, d.h. (0,1)
- ❑ Man spart ein Bit bei der Speicherung oder gewinnt bei gleichem Speicherbedarf ein Bit an Genauigkeit.
- ❑ Bei arithmetischen Operationen und bei der Konversion in andere Darstellungen darf diese Stelle natürlich nicht vergessen werden.

Beispiel (1)

3 verschiedene Maschinenformate mit je 32 Bit und $b = 2$.

Die Zahl 7135_{10} wird in jedem dieser Formate dargestellt.

a) Festkommadarstellung mit Zweierkomplement

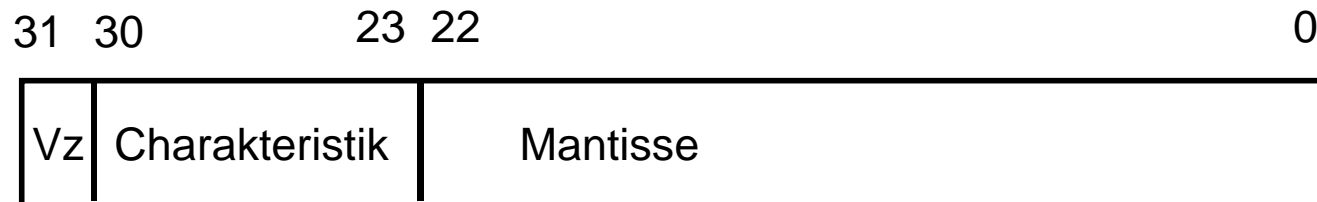
31 30 0

Vz																														
----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

$0\ 000\ 0000\ 0000\ 0000\ 0001\ 1011\ 1101\ 1111_2 = 0000\ 1BDF_{16}$

Beispiel (2)

b) Gleitkommadarstellung, normalisiert:



$$0 \ 100 \ 0101 \ 1 \ 110 \ 1111 \ 0111 \ 1100 \ 0000 \ 0000_2 = 46EF \ 7C00_{16}$$

c) Gleitkommadarstellung, normalisiert, erste "1" implizit:



$$0 \ 100 \ 0101 \ 1 \ 101 \ 1110 \ 1111 \ 1000 \ 0000 \ 0000_2 = 46DE \ F800_{16}$$

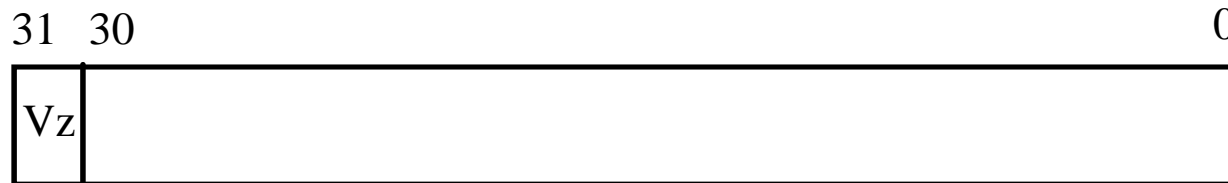
<http://www.h-schmidt.net/FloatConverter/IEEE754.html>

Darstellbarer Zahlenbereich (1)

- Die Anzahl darstellbarer Zahlen (Bitkombinationen) ist zwar in allen drei Fällen gleich (2^{32})
- Der Bereich und damit die Dichte darstellbarer Zahlen auf dem Zahlenstrahl ist aber sehr unterschiedlich.

Darstellbarer Zahlenbereich (2)

- Format a: Zahlen zwischen -2^{31} und $2^{31}-1$



- Format b:



negative Zahlen: $-(1-2^{-23}) \cdot 2^{127}$... $-0,5 \cdot 2^{-128}$,

positive Zahlen $0,5 \cdot 2^{-128}$... $(1-2^{-23}) \cdot 2^{127}$,

und *Null*

Darstellbarer Zahlenbereich (3)

- Format c: normalisierte Gleitkommadarstellung

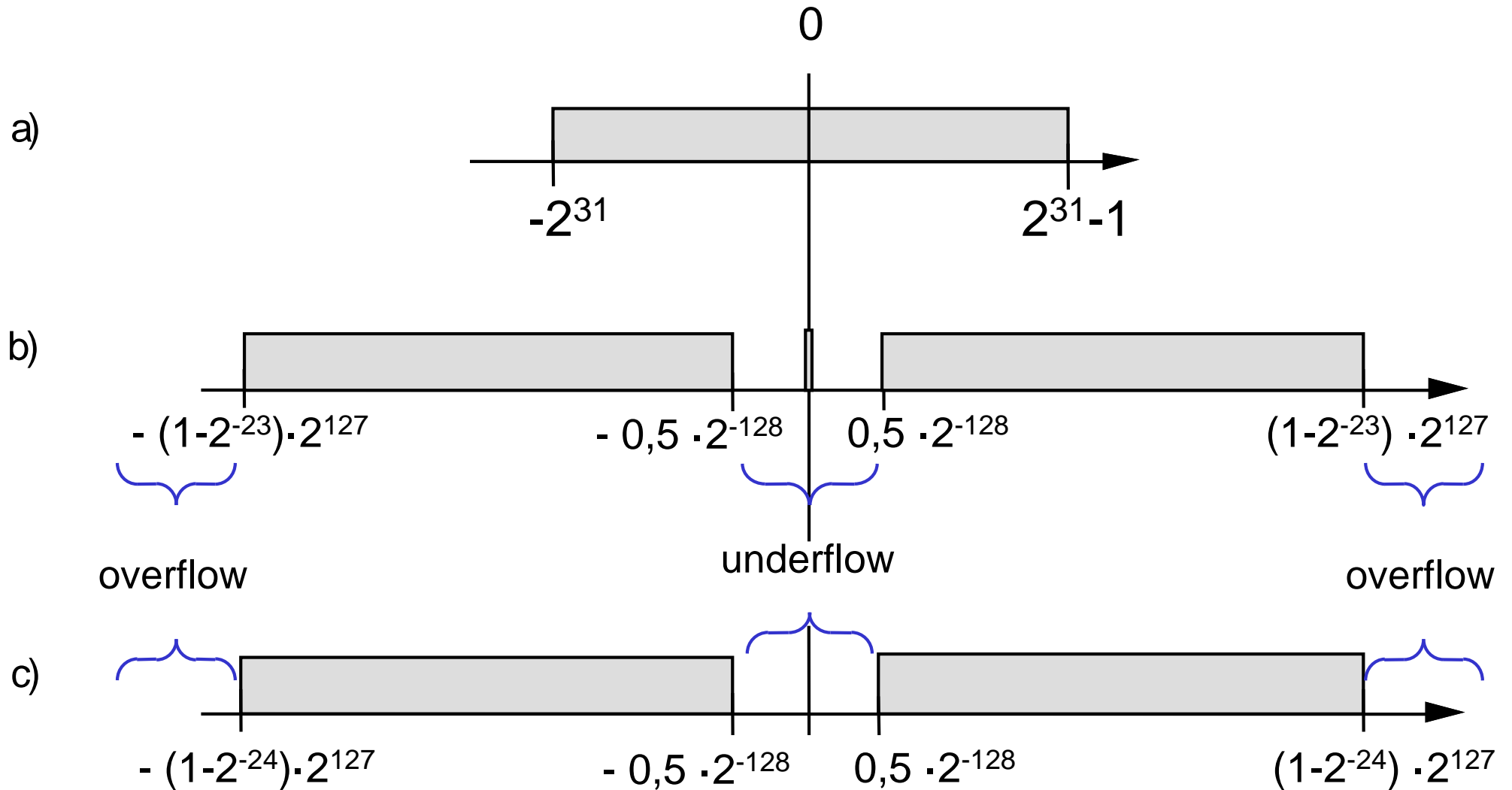


negative Zahlen: $-(1-2^{-24}) \cdot 2^{127} \dots -0,5 \cdot 2^{-128}$

positive Zahlen $0,5 \cdot 2^{-128} \dots (1-2^{-24}) \cdot 2^{127}$

Die Null kann nicht dargestellt werden!

Darstellbarer Zahlenbereich (4)

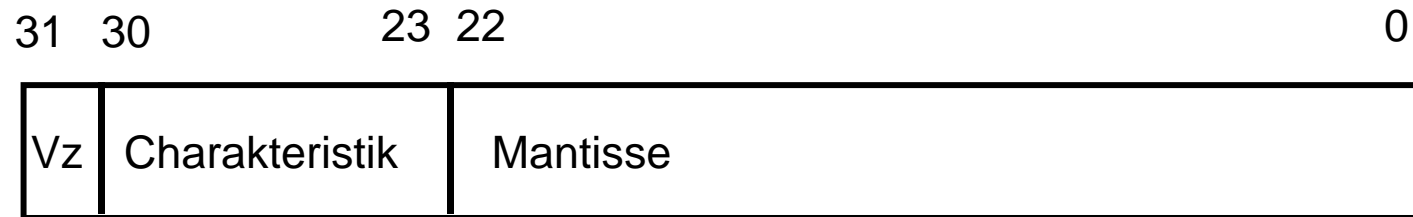


Charakteristische Zahlen

- Um verschiedene Gleitkommadarstellungen miteinander vergleichen zu können, definiert man drei charakteristische Zahlen:
- **maxreal** ist die größte darstellbare normalisierte positive Zahl
- **minreal** ist die kleinste darstellbare normalisierte positive Zahl
- **smallreal** ist die kleinste Zahl, die man zu 1 addieren kann, um einen von 1 verschiedenen Wert zu erhalten.

Beispiel

- In Format b) im letzten Beispiel



- $\text{maxreal} = (1 - 2^{-23}) \cdot 2^{127}$ $\text{minreal} = 0,5 \cdot 2^{-128}$

- Die Zahl 1 wird normalisiert als $0,5 \cdot 2^1$ dargestellt.

- Die nächstgrößere darstellbare Zahl hat in der Mantisse zusätzlich zur 1 in Bit 22 eine 1 in Bit 0.

- $\text{smallreal} = 0,000000000000000000000000000000001_2 \cdot 2^1$,
also $\text{smallreal} = 2^{-23} \cdot 2^1 = 2^{-22}$

Ungenauigkeiten

- Die Differenz zwischen zwei aufeinanderfolgenden Zahlen wächst bei Gleitkomma-Zahlen exponentiell mit der Größe der Zahlen, während sie bei Festkomma-Zahlen konstant ist.
- Bei der Darstellung großer Zahlen ergibt sich damit auch eine hohe Ungenauigkeit.
- Die Gesetzmäßigkeiten, die für reelle Zahlen gelten, werden für Maschinendarstellungen verletzt!

Dies gilt insbesondere auch wenn diese Zahlen in einer höheren Programmiersprache oft `real` heißen.

Beispiel

- Das Assoziativgesetz $(x + y) + z = x + (y + z)$ gilt selbst dann nicht unbedingt, wenn kein overflow oder underflow auftritt.

z.B.: $x = 1; y = z = \text{smallreal}/2$

$$\begin{aligned}(x + y) + z &= (1 + \text{smallreal}/2) + \text{smallreal}/2 \\ &= 1 + \text{smallreal}/2 \\ &= 1\end{aligned}$$

$$\begin{aligned}x + (y + z) &= 1 + (\text{smallreal}/2 + \text{smallreal}/2) \\ &= 1 + \text{smallreal} \\ &\neq 1\end{aligned}$$

Hinweis: `smallreal` ist die kleinste Zahl, die man zu 1 addieren kann, um einen von 1 verschiedenen Wert zu erhalten!

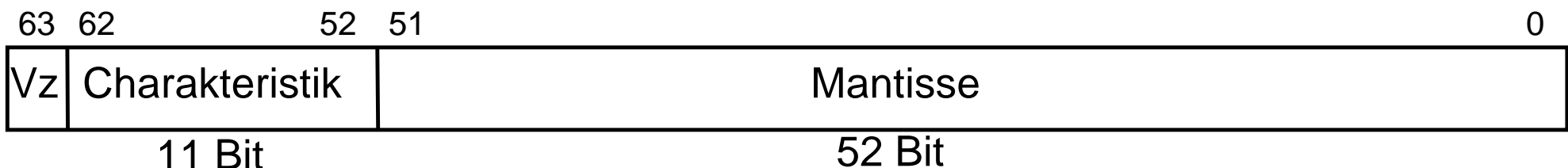
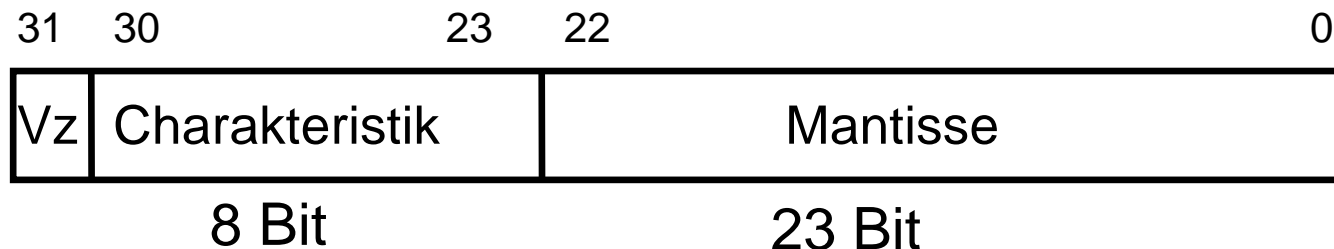
Problematik unterschiedlicher Definitionen

- Es existieren beliebig viele Möglichkeiten, selbst mit einer festen Wortbreite unterschiedliche Gleitkommaformate zu definieren (unterschiedliche Basis b , Darstellung der Null, Anzahl der Stellen für Charakteristik und Mantisse).
- Es existierten (bis Mitte der 80er Jahre) viele verschiedene, herstellerabhängige Formate
- Man konnte mit dem gleichen Programm auf unterschiedlichen Rechnern sehr unterschiedliche Ergebnisse erhalten!
- Normierung erforderlich

Normierung: IEEE-Standard (1)

IEEE: Institute of Electrical and Electronics Engineers

- IEEE-P 754-Floating-Point-Standard
- In vielen Programmiersprachen lassen sich Gleitkomma-Zahlen mit verschiedener Genauigkeit darstellen
 - z.B. in C: float, double, long double
- Der IEEE Standard definiert mehrere Darstellungsformen
 - IEEE single (einfach): 32 Bit
 - IEEE double (doppelt): 64 Bit
 - IEEE extended (erweitert): 80 Bit



Normierung: IEEE-Standard (2)

	einfach	doppelt
Vorzeichen-Bits	1	1
Exponenten-Bits	8	11
Mantissen-Bits	23	52
Bits insgesamt	32	64
BIAS	127	1023
Exponentenbereich	$[-126, 127]$	$[-1022, 1023]$

Biased Exponent	Mantisse	Bedeutung
111..111(= 255 bzw. = 2047)	$\neq 0$	<i>not a number</i> (keine gültige Zahl)
111..111(= 255 bzw. = 2047)	000..000(= 0)	$\pm\infty$
000..000(= 0)	000..000(= 0)	± 0

Modul 2: Rechnerarithmetik (1)

- ❑ Zahlensysteme
- ❑ Zahlendarstellung
- ❑ Grundrechenarten
- ❑ Zeichendarstellung

Addition von Dualzahlen

- Für die duale Addition gilt allgemein:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0 \text{ Übertrag } 1$$

$$1 + 1 + 1 \text{ (vom Übertrag)} = 1 \text{ Übertrag } 1$$

$$0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 = 45$$

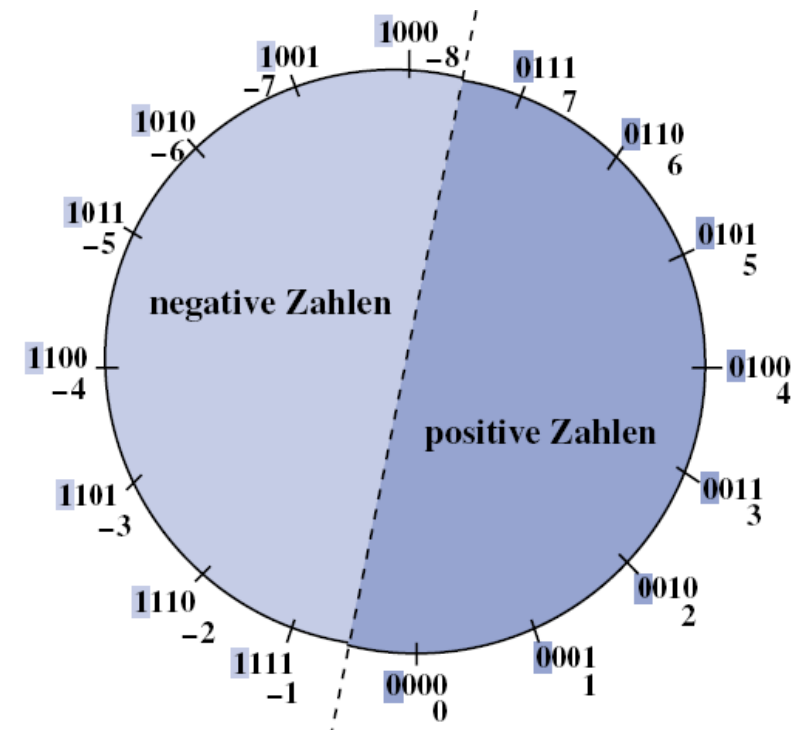
$$0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 = 54$$

$$1 \ 1 \ 1 \ 1 = \text{Übertrag}$$

$$1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 = 99$$

Addition von Dualzahlen, negative Zahlen

- Negative Zahlen werden üblicherweise durch ihren Betrag mit vorangestelltem Minuszeichen dargestellt.
 - Diese Darstellung wäre auch rechnerintern denkbar, hat jedoch den Nachteil, daß man eine gesonderte Vorzeichenrechnung durchführen müßte und man ein Rechenwerk benötigt, das sowohl addieren als auch subtrahieren kann.
 - Man kann die Subtraktion auf eine Addition durch das Verfahren der Komplementbildung zurückzuführen
 - Zuordnung der Bitkombinationen zu positiven und negativen Zahlen
 - Zahlenring für vier Bits, erstes Bit ist Vorzeichenbit
 - Bei der Verwendung der Komplementdarstellung muß eine Maschine nicht subtrahieren können, sondern kann jede Subtraktion $a - b$ durch eine Addition $a + -b$ realisieren



Überlauferkennung

- Allgemeine Überlauferkennung bei dualer Addition:
 - **Korrekte Addition:** beide Überträge sind gleich.
 - **Überlauf:** beide Überträge sind ungleich.
- Technische Realisierung z.B. durch ein Antivalenzgatter (siehe später)

Subtraktion von Dualzahlen

- Basiert auf der Addition von Festkomma-Dualzahlen:
 - Grundlage für die Durchführung aller arithmetischen Verknüpfungen

- Denn:
 - Subtraktion \triangleq Addition der negativen Zahl
 - $X - Y = X + (-Y)$

Multiplikation und Division (1)

- Zweierkomplementzahlen erwiesen sich für Addition und Subtraktion als besonders günstig, weil bei dieser Darstellung das Vorzeichen nicht explizit betrachtet werden mußte.

- Bei der Multiplikation existiert dieser Vorteil nicht.

- Lösungen:
 - **Zweierkomplementzahlen** zunächst in eine Form mit Betrag und Vorzeichen umwandeln. Zahlen dann miteinander zu multiplizieren und das Ergebnis schließlich wieder in die Zweierkomplementdarstellung umzusetzen.

 - **Spezielle Multiplikationsalgorithmen** für Zweierkomplementzahlen verwenden (Booth-Algorithmus, hier nicht behandelt).

Multiplikation und Division (2)

- Die ganzzahlige Multiplikation bzw. Division wird in einem Rechner allgemein mittels wiederholter Addition durchgeführt.
- In den Sonderfällen des Multiplikators bzw. Divisors von 2, 4, ... kann die Multiplikation bzw. Division aber einfacher und schneller durch eine Verschiebung von entsprechend vielen Bits nach links bzw. rechts erfolgen:
Bei 2 (21) um 1 Bit, bei 4 (22) um 2 Bit, bei 8 um 3 (23) Bit.

$$\text{dezimal : } (20)_{10} \times (8)_{10} = 160_{10}$$

$$\text{dual : } (10100)_2 \times (1000)_2 = (10100000)_2 \quad [10100 \mid 000]$$

$$\text{dezimal : } (20)_{10} : (4)_{10} = 5_{10}$$

$$\text{dual : } (10100)_2 : (100)_2 = (101)_2 \quad [101 \mid 00]$$

Multiplikation

□ Papier und Bleistift Methode:

- Analog zur Multiplikation mit Papier und Bleistift im Dezimalsystem kann man auch im Dualsystem vorgehen.

□ Beispiele:

$$\begin{array}{r} 13 \cdot 11 \\ \hline 13 \\ + \quad 13 \\ \hline 143 \end{array}$$

$$\begin{array}{r} 1101 \cdot 1011 \\ \hline 1101 \\ \quad 0000 \\ \quad \quad 1101 \\ + \quad \quad \quad 1101 \\ \hline 10001111 \end{array}$$

Multiplikation von Vorzeichen-Betrags-Zahlen

- Bei Zahlen, die mit Betrag und Vorzeichen dargestellt sind, ergeben sich keine Probleme.
- Die Beträge der Zahlen werden wie positive Zahlen miteinander multipliziert.
- Das Vorzeichen des Ergebnisses ergibt sich aus der Antivalenzverknüpfung der Vorzeichen der beiden Faktoren.

Vorzeichenbehaftete Multiplikation

- Vorzeichenbehaftete Zahlen können grundsätzlich in die Darstellung mit Vorzeichen und Betrag gebracht werden.
- Das Vorzeichen des Produkts wird dann nach der Regel

$$\text{sign}(X \cdot Y) = \text{sign}(X) \text{ XOR } \text{sign}(Y)$$

aus den beiden Faktorenvorzeichen durch die Exklusiv-ODER-Verknüpfung gewonnen.

Multiplikation von Gleitkommazahlen

- Zur Multiplikation von Gleitkommazahlen muß man Mantissen beider Zahlen multiplizieren und ihre Exponenten addieren:

$$m_1 b^{e_1} \cdot m_2 b^{e_2} = (m_1 \cdot m_2) b^{e_1 + e_2}$$

- Ist die Mantisse mit Hilfe von Betrag und Vorzeichen dargestellt, → ist der übliche Multiplikationsalgorithmus anwendbar.
- Das Ergebnis muß nach der Multiplikation unter Umständen noch normalisiert werden.
- Bei Addition der Charakteristiken $c_1 = e_1 + o$ und $c_2 = e_2 + o$ muß die Summe außerdem um den Offset o korrigiert werden, um die richtige Ergebnischarakteristik $c = (e_1 + e_2) + o$ zu erhalten.

Anmerkungen

- ❑ Bei vielen Anwendungen jedoch wechseln Addition und Multiplikation einander ständig ab.
- ❑ Umwandlung zwischen verschiedenen Zahlendarstellungen kann viel Zeit in Anspruch nehmen.
- ❑ Es wäre günstiger, wenn durchgängig (also auch bei der Multiplikation) im Zweierkomplement gerechnet wird, um die Vorteile bei der Addition nutzen zu können (z.B. Booth-Algorithmus).

Division

- Die Division von Dualzahlen folgt denselben Prinzipien wie die Multiplikation.
- Auch hier stellt die Papier-und-Bleistift-Methode die Basis für verschiedene Algorithmen dar.

□ Beispiel:

$$\begin{array}{r} 1224 : 12 = 102 \\ - \underline{12} \\ 02 \\ \underline{24} \\ \underline{24} \\ 0 \end{array}$$

Manuelle Division

- Im Dualsystem sind als Ergebnis einer Teildivision nur die Werte
 - 0 → Divisor > augenblicklicher Dividend
 - 1 → Divisor ≤ augenblicklicher Dividendmöglich.

- Bei manueller Division erkennt man sofort, ob das Ergebnis 0 ist und „eine weitere Stelle gebraucht wird“.

Maschinelle Division

- **Drei Möglichkeiten:**
 - 1. Komparatorschaltung, um den Divisor mit augenblicklichem Dividenden zu vergleichen.
 - 2. Subtraktion: Ergibt sich ein negatives Ergebnis, lädt man nochmals den alten Wert des Dividenden.
 - 3. Subtraktion: Bei einem negativen Ergebnis, addiert man den Divisor wieder (Rückaddition).

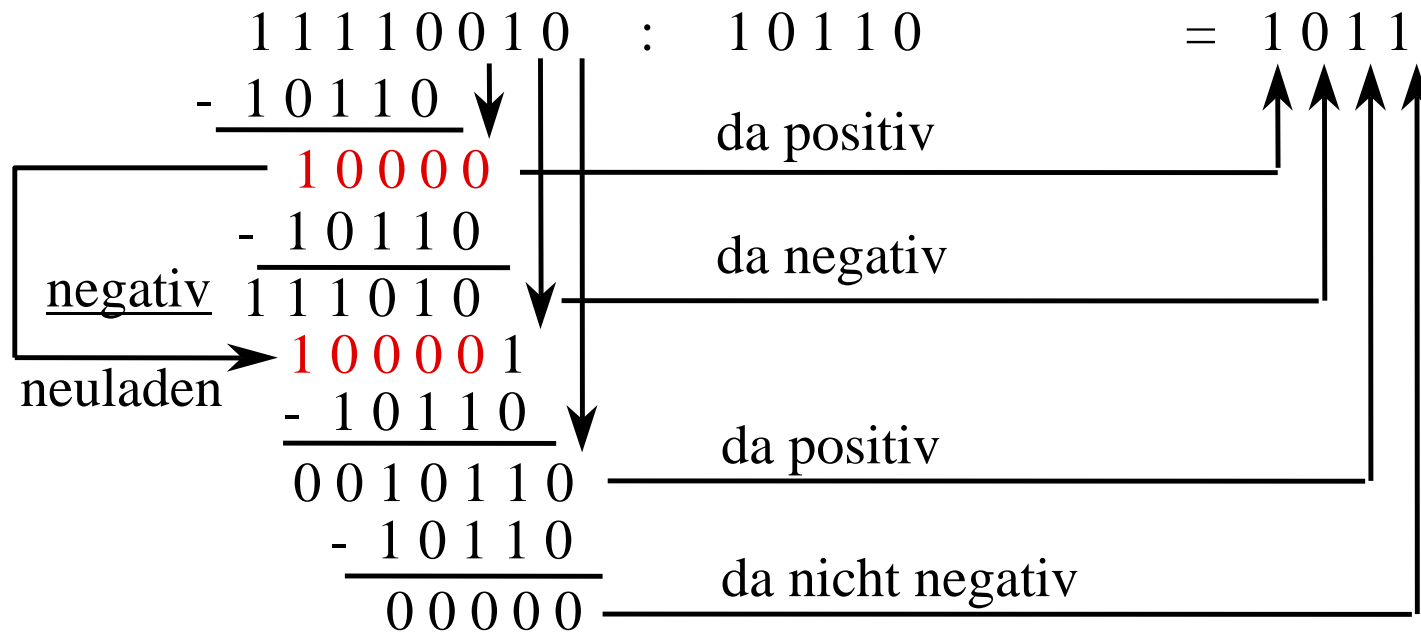
- **Verkürzte Division:** Rückaddition und Subtraktion des um eins nach rechts verschobenen Divisors zusammenziehen. Man addiert gleich den um eins verschobenen Divisor:

$$+ \text{ Divisor} - \frac{1}{2} \text{ Divisor} = + \frac{1}{2} \text{ Divisor}$$

Division

- Division durch Durchführung von Subtraktionen:
 - Direkte Subtraktion des Divisors
 - Addition des Divisor-Zweierkomplements.

Direkte Subtraktion:



Bemerkungen

- ❑ Bei Division durch 0 muß ein Ausnahmezustand erkannt werden und an die Steuereinheit (Prozessor) weitergemeldet werden.
- ❑ Die Division muß abgebrochen werden, wenn die vorhandene Bitzahl des Ergebnisregisters ausgeschöpft ist (periodische Dualbrüche).
- ❑ Die Schaltungen für die Multiplikation können nach Modifikation auch für den Grundalgorithmus der Division eingesetzt werden:
 - Linksschieben des Dividenden (statt Rechtsschieben des Multiplikanden)
 - Subtraktion des Divisors (statt Addition des Multiplikators)

Modul 2: Rechnerarithmetik (1)

- ❑ Zahlensysteme
- ❑ Zahlendarstellung
- ❑ Grundrechenarten
- ❑ Zeichendarstellung

ASCII Code zur Darstellung von Zeichen (1)

- ASCII-Code (American Standard for Coded Information Interchange) ist eine festgelegte Abbildungsvorschrift (Norm) zur binären Kodierung von Zeichen.
 - Der ASCII-Code umfasst Klein-/Großbuchstaben des lateinischen Alphabets, (arabische) Ziffern und viele Sonderzeichen.
 - Die Kodierung erfolgt in einem Byte (8 Bits), so daß mit dem ASCII-Code 256 verschiedene Zeichen dargestellt werden können.
 - Da das erste Bit nicht vom Standard-ASCII-Code genutzt wird, können im Standard-ASCII-Code nur 128 Zeichen dargestellt werden.
 - Unterschiedliche, speziell normierte, ASCII-Code-Erweiterungen nutzen das erste Bit, um weitere 128 Zeichen darstellen zu können.

ASCII Code zur Darstellung von Zeichen (2)

ASCII-Code zu den darstellbaren Zeichen (Auszug)

Zeichen	Dez.	Binär	Hexa	Oktal	Zeichen	Dez.	Binär	Hexa	Oktal
!	33	0010 0001	21	041	P	80	0101 0000	50	120
"	34	0010 0010	22	042	Q	81	0101 0001	51	121
#	35	0010 0011	23	043	R	82	0101 0010	52	122
\$	36	0010 0100	24	044	S	83	0101 0011	53	123
%	37	0010 0101	25	045	T	84	0101 0100	54	124
&	38	0010 0110	26	046	U	85	0101 0101	55	125
'	39	0010 0111	27	047	V	86	0101 0110	56	126
(40	0010 1000	28	050	W	87	0101 0111	57	127
)	41	0010 1001	29	051	X	88	0101 1000	58	130

ASCII Code zur Darstellung von Zeichen (3)

- Zur Speicherung von Texten werden einzelne Bytes, die jeweils immer ein Zeichen kodieren, einfach hintereinander abgespeichert, so daß man eine Zeichenkette (*String*) erhält.

- Um das Ende der Zeichenkette zu identifizieren, werden (in den Programmiersprachen) unterschiedliche Verfahren verwendet.
 - Die Länge der Zeichenkette wird im ersten bzw. in den ersten Bytes vor der eigentlichen Zeichenkette gespeichert.
Dieses Verfahren benutzt z. B. die Programmiersprache PASCAL.

 - Das Ende der Zeichenkette wird durch ein besonderes, nicht darzustellendes Zeichen gekennzeichnet.
So verwendet z.B. die Programmiersprache C/C++ ein 0-Byte (Byte, in dem alle Bits 0 sind), um das Ende einer Zeichenkette zu kennzeichnen.

Unterscheidung von Ziffern und Zeichen

- *Ziffer als ASCII-Code → Angabe des Zeichens (Ziffer) in Hochkomma:*

'0': 00110000 (Dezimal 48)
'4': 00110100 (Dezimal 52)
'5': 00110101 (Dezimal 53)
'8': 00111000 (Dezimal 56)


- *Ziffer als numerischer Wert → Angabe einer Ziffer (ohne Hochkomma):*

0: 00000000 (Dezimal 0)
4: 00000100 (Dezimal 4)
5: 00000101 (Dezimal 5)
8: 00001000 (Dezimal 8)

Beispiel: Speicherung von Ziffern und Zeichen

Angabe	Dezimaler ASCII-Wert	Dualdarstellung im Rechner
'a'	97	01100001
'W'	87	01010111
'*'	42	00101010
'9'	57	00111001

Unicode zur Darstellung von Zeichen (1)

- ❑ Der ASCII-Code mit 256 (128) Zeichen ist sehr begrenzt.
- ❑ Unicode für Zeichen oder Elemente praktisch aller bekannten Schriftkulturen und Zeichensysteme kodierbar. 
- ❑ Zeichenwerte der Zeichen bis Unicode Version 3.0 (September 1999) wurden ausschließlich durch eine zwei Byte lange Zahl ausgedrückt.
 - Auf diese Weise lassen sich bis zu 65 536 verschiedene Zeichen unterbringen (2 Byte = 16 Bit = 2^{16} Kombinationsmöglichkeiten).
 - *Bezeichnung des Zwei-Byte-Schemas: Basic Multilingual Plane (BMP)*
- ❑ In Version 3.1 (März 2001) sind 94.140 Zeichen aufgenommen, wobei die Zwei-Byte-Grenze durchbrochen wurde.
 - Das Zwei-Byte-Schema wird deshalb von einem Vier-Byte-Schema abgelöst.

Unicode zur Darstellung von Zeichen (2)

- In Version 6.0 (Oktober 2010) sind 109.449 Zeichen enthalten.
 - Unicode in 17 Bereiche (planes) gegliedert, jeweils á 65.536 Zeichen
 - Basic Multilingual Plane: hauptsächlich Schriftsysteme
 - Supplementary Multilingual Plane: historische Schriftsysteme
 - Supplementary Ideographic Plane: Chinesische, Japanische und Koreanische Schrift
 - Supplementary Special-purpose Plane: Private Use Area-A und -B

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

- Lateinische Schriften und Symbole
- Lautschriften
- Andere europäische Schriften
- Nahost- und Südwestasiatische Schriften
- Afrikanische Schriften
- Südasiatische Schriften
- Südostasiatische Schriften
- Ostasiatische Schriften
- CJK-Ideogramme
- Kanadische Silben
- Symbole
- Diakritika
- UTF-16-Surrogates und privater Nutzungsbereich
- Verschiedene Zeichen
- Nicht belegte Codebereiche

Unicode zur Darstellung von Zeichen (3)

- Die Notation lautet
 - � für dezimale Notation bzw.
 - � für die hexadezimale Notation,wobei das 0000 die Unicode-Nummer des Zeichens darstellt.
- Formate für Speicherung und Übertragung unterschiedlich:
 - UTF-8 (Unicode Transformation Format) – Internet und in Betriebssystemen
 - UTF-16 – Zeichencodierung in Java
- Unicode wird in ostasiatischen Ländern kritisiert.
 - Schriftzeichen verschiedener nicht verwandter Sprachen sind vereinigt
 - Vor allem in Japan konnte sich der Unicode kaum durchsetzen
 - Japan mit zahlreichen Alternativen zu Unicode wie etwa der Mojiky -Standard
 - Antike Texte in Unicode aufgrund der Vereinheitlichung ähnlicher CJK-Schriftzeichen (chinesisch, japanisch, koreanisch) nicht originalgetreu wiederzugeben

BCD-Code zur Darstellung von Zeichen

- BCD (Binary Coded Decimals) kodieren binär Zahlen Ziffern.
- Für jede Dezimalziffer werden mindestens vier, manchmal auch acht Bits verwendet.
 - Die Ziffern werden nacheinander immer durch ihren Dualwert angegeben.
 - Diese speicherplatzverschwendende Art der Speicherung von Dezimalzahlen erleichtert aber manche Anwendungen.
 - Anwendungsbereiche:
 - Rechnen im Dezimalsystem
 - Speichern von Dezimalzahlen (Telefonnummern u.ä.)
 - Ansteuerung von LCD-Anzeigen, um Dezimalziffern einzeln anzuzeigen.

Dezimalzahl	Dualzahl	Duale BCD-Darstellung
294	100100110	0010.1001.0100 2 9 4
16289	11111110100001	0001.0110.0010.1000.1001 1 6 2 8 9

Duale Größenangaben (1)

Maßeinheiten für Bytes

Maßeinheit		Anzahl von Bytes	KBytes	MBytes
Byte		1		
Kilobyte (KByte)	2^{10}	1024	1	
Megabyte (MByte)	2^{20}	1.048.576	1024	1
Gigabyte (GByte)	2^{30}	1.073.741.824	1.048.576	1024
Terabyte (TByte)	2^{40}	1.099.511.627.776	1.073.741.824	1.048.576
Petabyte (PByte)	2^{50}	1.125.899.906.842.624	1.099.511.627.776	1.073.741.824
Exabyte (EByte)	2^{60}	1.152.921.504.606.846.976	1.125.899.906.842.624	1.099.511.627.776

Duale Größenangaben (2)

- ❑ Kilo entspricht dem Faktor $2^{10} = 1024$
- ❑ Mega entspricht dem Faktor $2^{10} * 2^{10} = 1024 * 1024 = 1.048.576$
- ❑ Giga entspricht dem Faktor $2^{10} * 2^{10} * 2^{10} = 2^{30}$

- ❑ Beispiel Festplattenherstellerangaben zur Speicherkapazität

- ❑ MB MByte
 - Faktor hier $10^3 = 1000$

- ❑ Folge:
 - 1 GB 1.073.741.824 Byte sondern 1.000.000.000 Byte
 - D.h., es „fehlen“ real 73 MegaByte!
 - 200 GB entsprechen damit nur 186 Gbyte!