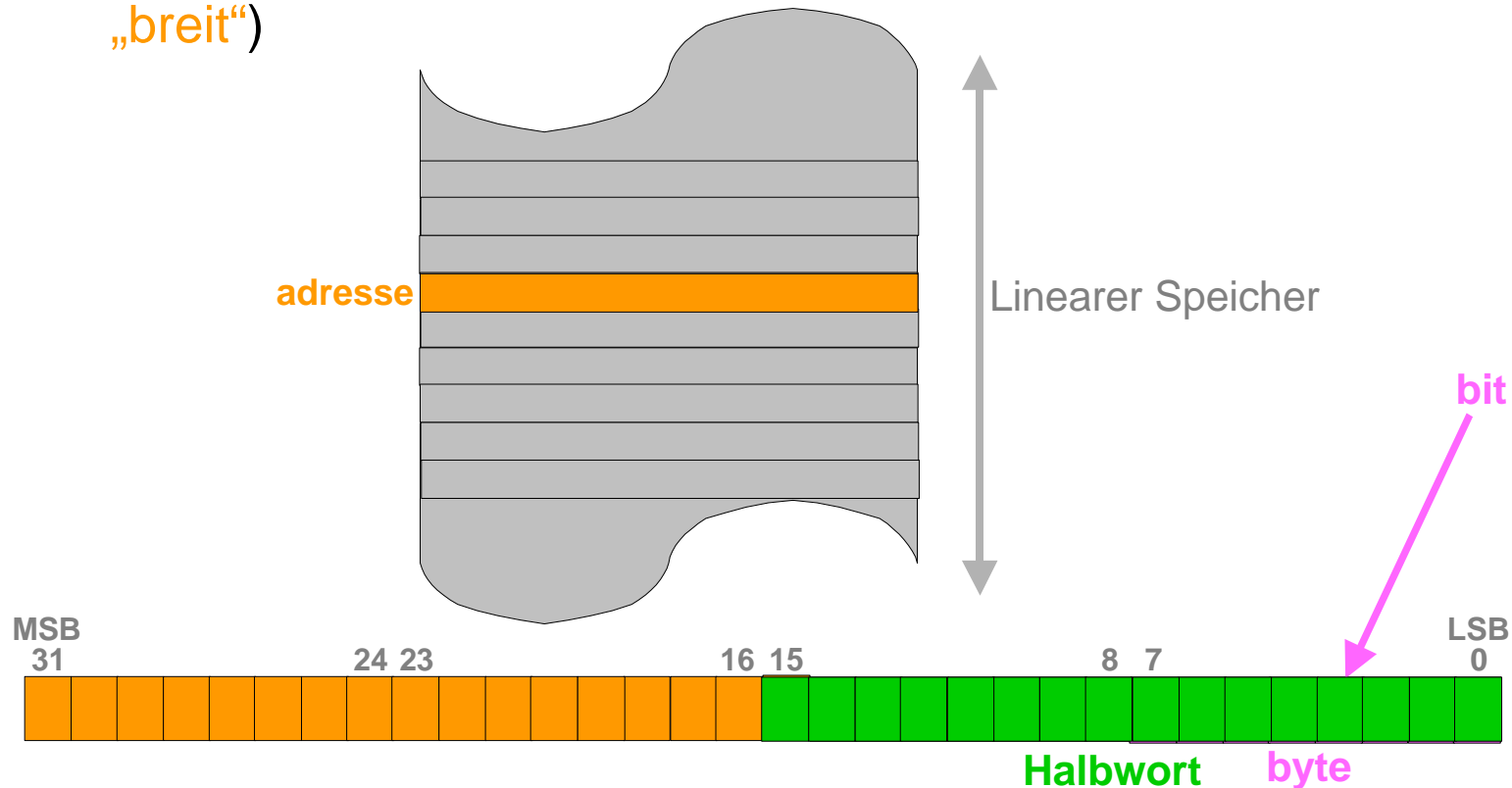


Zahlendarstellung

Zahlen und ihre Darstellung in Digitalrechnern

Grundstrukturen: Speicherorganisation und Zahlenmengen

- Linear organisierter Speicher
 - zu einer Adresse gehört ein Speicher mit 32 Bit-Zellen (1 Wort „breit“)



Grundstrukturen: Speicherorganisation und Zahlenmengen

- Zahlenmengen ($\mathbb{N}_0 \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$) | I
- natürliche Zahlen
 - ganze Zahlen = natürliche Zahlen und negative ganze Zahlen
 - rationale Zahlen = ganze Zahlen und gebrochene Zahlen
 - reelle Zahlen = rationale Zahlen und irrationale Zahlen
 - komplexe Zahlen = reelle Zahlen und „echt imaginäre“ Zahlen

Ganzzahldarstellung

➤ Basis-Zahldarstellung:

$$zahl = \sum_{i=0}^n a_i \cdot b^i$$

- die Basis $b \geq 2$ ist aus den natürlichen Zahlen
- die Ziffer a_i ist aus den natürlichen Zahlen
- $0 \leq a_i \leq b-1$
- die Darstellung ist eindeutig
- **Schreibweise:**

$$zahl = (a_n \dots a_0)_b$$

Beispiel: $(1024)_{10} = 1 \cdot 10^3 + 0 \cdot 10^2 + 2 \cdot 10^1 + 4 \cdot 10^0$

- gebräuchliche Zahlenbasen:
 - * 2 (Binär-System)
 - * 8 (Oktal-System)
 - * 10 (Dezimal-System)
 - * 16 (Hexadezimal-System)
- **Multiplikation/Division** mit b : shift der Zahl-Ziffernfolge um 1 Stelle nach **links/rechts**

Ganzzahldarstellung

➤ Konvertierung zwischen zwei Basen:

$$zahl = \sum_{i=0}^n a_i \cdot b^i = a_n b^n + a_{n-1} b^{n-1} + \dots + a_1 b + a_0$$

$$zahl = (\dots (a_n b + a_{n-1}) \cdot b + \dots) \cdot b + a_0$$

(→ Horner-Schema)

- Basis $b \rightarrow$ Basis 10

- * Eingabe: b , Feld $a[0..k]$
- * Ausgabe: Dezimalzahl
- * „vorwärts: von links nach rechts“

```
zahl := 0;  
FOR i:=n TO 0 BY -1 DO;  
    zahl := zahl*b + a[i]  
END;
```

- Basis 10 \rightarrow Basis b

- * Eingabe: Dezimalzahl, neue Basis b
- * Ausgabe: Feld $a[0..k]$ (neue Ziffernfolge)
- * „rückwärts: von rechts nach links“

```
i := 0;  
a[i] := 0;  
WHILE zahl > 0 DO  
    a[i] := zahl MOD b;  
    zahl := zahl DIV b;  
    i := i+1  
END;
```

Ganzzahldarstellung

➤ Konvertierung zwischen zwei Basen: (cont'd)

- Beispiel: $(236)_8 \rightarrow (?)_3$

- * Schritt 1: $(236)_8 \rightarrow$ Dezimalzahl

$n = 2:$

$$(236)_8 = ((0 \cdot 8 + 2) \cdot 8 + 3) \cdot 8 + 6 = 158 = (158)_{10}$$

- * Schritt 2: Dezimalzahl $\rightarrow (?)_3$

$b = 3$

$$(158)_{10} : \quad \begin{array}{rcl} 158 / 3 & = & 52 \quad \text{Rest } 2 \\ 52 / 3 & = & 17 \quad \text{Rest } 1 \\ 17 / 3 & = & 5 \quad \text{Rest } 2 \\ 5 / 3 & = & 1 \quad \text{Rest } 2 \\ 1 / 3 & = & 0 \quad \text{Rest } 1 \end{array}$$

also: $(158)_{10} = (((((0 \cdot 3 + 1) \cdot 3 + 2) \cdot 3 + 2) \cdot 3 + 1) \cdot 3 + 2) = (12212)_3$

wobei: $(\dots) = 52, (\dots) = 17, (\dots) = 5, (\dots) = 1,$

Ganzzahldarstellung

- Spezialfall: Konvertierung zwischen Basen $2 \rightarrow 2^k$:
 - von rechts nach links: Zusammenfassen von jeweils k benachbarten Ziffern
 - Beispiel mit der Zahl $(300)_{10}$: Die Darstellung zur Basis $b=2$ soll in die zur Basis $b=8$ umgewandelt werden, d.h. $k=3$:

$$(100101100)_2 = 1 \cdot 2^8 + 0 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$$

$$\mathbf{100} \mathbf{101} \mathbf{100} = (1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0) \cdot 2^6 + (1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0) \cdot 2^3 + (1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0) \cdot 2^0$$

$$\mathbf{100} \mathbf{101} \mathbf{100} = 4 \cdot 8^2 + 5 \cdot 8^1 + 4 \cdot 8^0 = (454)_8$$

- Spezialfall: Konvertierung zwischen Basen $2^k \rightarrow 2$:
 - selbst nachdenken!

Negative Zahlen im Binärsystem (Dualsystem)

➤ Allgemeines

- **Die Anzahl der darstellbaren Zahlen ist beschränkt!**
(abhängig von der Wortlänge)
 - * Wortlänge = 8bit $\rightarrow N = 2^8 = 256$ versch. 0/1-Kombinationen
 $\rightarrow 256$ verschiedene Zahlen darstellbar, z.B. 0 ... 255 (= $N - 1$)
- **Unser Rechner kann nur addieren**, besitzt lediglich ein Addierwerk, kein Subtrahierwerk; letztlich heisst das, auch Multiplikationen und Divisionen muss das Addierwerk erledigen!
- Frage: Kann man sich im Bereich der Dezimalzahlen einen Algorithmus vorstellen, der mittels Addition eine Subtraktion durchführt?
(unter der Nebenbedingung, dass die Anzahl der darstellbaren Zahlen beschränkt ist)
 - * Idee: falls $x > 0$, dann nix: $x \rightarrow x$
falls $x < 0$, dann: $x \rightarrow N - |x|$

Negative Zahlen im Binärsystem (Dualsystem)

➤ 4 Möglichkeiten der Darstellung

- Vorzeichen und Betrag („signed magnitude“)

- * das fällt einem sofort ein:

- nehme das Bit ganz links als Vorzeichen: $0 \leftrightarrow +$; $1 \leftrightarrow -$
 - die restlichen Bits stellen den Betrag der Zahl dar.

- * Beispiel: Wortlänge $n = 4$ Bit $\rightarrow N = 2^4 = 16$ versch. 0/1-Kombinationen
 $\rightarrow 16$ verschiedene Zahlen darstellbar, bisher (kardinal) $0 \dots 15 (= N - 1)$,
jetzt (integer) $-7 \dots +7$, also $-(2^{n-1} - 1) \dots +(2^{n-1} - 1)$

- * Problem 1:

- es gibt zwei Nullen: $+0 \leftrightarrow 0000$; $-0 \leftrightarrow 1000$

- also: **Eine** Zahl aber **zwei** unterscheidbare(!) Bitfolgen

- * Problem 2:

- Bei dieser Darstellung ist eine Addierwerk und ein Subtrahierwerk notwendig; es gibt keinen Algorithmus der Subtraktion per Addition erledigt für diese Darstellung.

- * Problem 3:

- Es ist eine Logik erforderlich zur Entscheidung ob Addition oder Subtraktion auszuführen (4 Vorzeichenfälle)

Negative Zahlen im Binärsystem (Dualsystem)

➤ 4 Möglichkeiten der Darstellung (cont'd)

- Einer-Komplement (One's Complement)

- * gebildet durch stellenweises **Invertieren** der **Originalzahl**: $0 \rightarrow 1, 1 \rightarrow 0$
- * addiert man zur **Originalzahl** ihr Einer-Komplement (= **Invertierte**) so ergibt sich immer eine Folge von Einsen.
- * Eine Folge von Einsen ist nichts anderes als (die Invertierte der) Null, also -0 ($+0 \leftrightarrow$ Folge von Nullen), d.h. man hat zur **Originalzahl** deren „**Negatives** addiert“.
- * Beispiel: Wortlänge $n = 4$ Bit $\rightarrow N = 2^4 = 16$ versch. 0/1-Kombinationen $\rightarrow 16$ verschiedene Zahlen darstellbar, bisher (kardinal) $0 \dots 15$ ($= N - 1$), jetzt (integer) $-7 \dots +7$, also $-(2^{n-1} - 1) \dots +(2^{n-1} - 1)$
- * Problem 1 besteht noch:
es gibt zwei Nullen: $+0 \leftrightarrow 0000$; $-0 \leftrightarrow 1111$
also: **Eine** Zahl aber **zwei** unterscheidbare(!) Bitfolgen
- * Problem 2 ist gelöst:
Bei dieser Darstellung genügt ein Addierwerk; Subtraktion bedeutet Addition des Negativen.
- * Problem 3 (Logik) stellt sich nicht mehr.
- * Problem 4: $1011 = -4$ oder $+11$?? durch beschränkten Zahlenbereich $-7 \dots +7$ gelöst. \Rightarrow Bit ganz links: $1 \leftrightarrow$ negative Zahl, $0 \leftrightarrow$ positive Zahl

Negative Zahlen im Binärsystem (Dualsystem)

➤ 4 Möglichkeiten der Darstellung (cont'd)

- Zweier-Komplement (Two's Complement)

- * gebildet durch das Einer-Komplement mit nachfolgender Addition von 1
- * addiert man zur **Originalzahl** ihr Zweier-Komplement (= **Invertierte + 1**) so ergibt sich immer eine **1** mit nachfolgenden Nullen; **die Anzahl der Stellen ist um eine gewachsen.**
- * **Streicht man die führende 1**, so sind die nachfolgenden Nullen nichts anderes als Null, man hat zur **Originalzahl** deren „**Negatives** addiert“.
- * Beispiel: Wortlänge $n = 4$ Bit $\rightarrow N = 2^4 = 16$ versch. 0/1-Kombinationen $\rightarrow 16$ verschiedene Zahlen darstellbar, bisher (kardinal) $0 \dots 15 (= N - 1)$, jetzt (integer) $-8 \dots +7$, also $-(2^{n-1}) \dots +(2^{n-1} - 1)$
- * Problem 1 besteht nicht mehr: $0000 \leftrightarrow 0$; $1111 \leftrightarrow -1$
- * Problem 2 ist gelöst:
Bei dieser Darstellung genügt ein Addierwerk; Subtraktion bedeutet Addition des Negativen.
- * Problem 3 (Logik) stellt sich nicht mehr.
- * Problem 4: **1011 = -5 oder +11 ??** durch beschränkten Zahlenbereich $-8 \dots +7$ gelöst. \Rightarrow **Bit ganz links: 1 \leftrightarrow negative Zahl, 0 \leftrightarrow positive Zahl**

Negative Zahlen im Binärsystem (Dualsystem)

➤ 4 Möglichkeiten der Darstellung (cont'd)

- Zweier-Komplement (cont'd)

- * übrigens: Im Zweier-Komplement stimmt die Dualdarstellung von -5 mit der von $2^4 - 5 = 16 - 5 = 11$ überein:

$$(5)_{10} = (0101)_2$$

$$\text{Zweier-Komplement von } (5)_{10}: (1010)_2 + 1 = (1011)_2 = (11)_{10}$$

- * Beispiel mit Dezimalzahlen ($b = 10$):

- Es sei $n = 2 \rightarrow N = 10^2 = 100$ verschiedene Dezimalzahlen, entweder kardinal $0 \dots 99 (= N - 1)$ oder (integer) $-50 \dots +49$, also $-5 \cdot 10^{n-1} \dots + 5 \cdot 10^{n-1} - 1$
- **Originalzahl** sei (z.B.): 23
Ihr Zehner-Komplement: $10^2 - 23 = 77$. 77 ist nicht im Zahlenbereich $-50 \dots +49$; 77 ist die Darstellung von -23 im Zehner-Komplement ($x < 0: x \rightarrow N - |x|$).
- addiert man zur **Originalzahl** ihr Zehner-Komplement, so ergibt sich immer eine **1** mit nachfolgenden Nullen (hier 100); **die Anzahl der Stellen ist um eine gewachsen**.
- **Streicht man die führende 1**, so sind die nachfolgenden Nullen nichts anderes als Null, man hat zur **Originalzahl** deren „**Negatives** addiert“, also eine Darstellung von -23.
- \Rightarrow Statt (z.B) $36 - 23 = 13$ kann man auch rechnen: $36 + 77 = 113$
 \Rightarrow Statt (z.B) $14 - 23 = -9$ kann man auch rechnen: $14 + 77 = 91 = 100 - 9$

Negative Zahlen im Binärsystem (Dualsystem)

➤ 4 Möglichkeiten der Darstellung (cont'd)

- Zweier-Komplement (cont'd)

- * häufigst genutzte rechnerinterne Darstellung negativer ganzer Zahlen.

- * Beispiel mit Dualzahlen ($b = 2$):

- Dual zu berechnen: $85 - 103 = -18$

$$(85)_{10} = (01010101)_2$$

- Es sei $n = 8 \rightarrow N = 2^8 = 256$ verschiedene Dezimalzahlen: $-128 \dots +127$

- Aus der Subtraktion -103 soll eine Addition werden:

Originalzahl ist: $(103)_{10} = (01100111)_2$

Ihr Zweier-Komplement: $10011000 + 1 = 10011001$.

- Subtraktion \rightarrow Addition: $01010101 + 10011001 = 11101110$

- Anzahl der Stellen nicht gewachsen

\Rightarrow Keine Streichung der führenden 1, die 1 ganz links zeigt ein negatives Ergebnis, d.h. Ergebnis liegt als Zweier-Komplement vor

\Rightarrow Übersetzung = Bildung Zweier-Komplement:

$$11101110 \rightarrow 00010001 + 1 = 00010010 = (18)_{10}$$

\Rightarrow Das negative Ergebnis lautet -18

Negative Zahlen im Binärsystem (Dualsystem)

➤ 4 Möglichkeiten der Darstellung (cont'd)

- Exzess 2^{n-1}

- * der **gesamte** darstellbare Zahlenbereich (positive und negative Halbachse) wird auf die positive Halbachse abgebildet
exzess-zahl = zahl + 2^{n-1}
- * Beispiel mit $n = 8$, d.h. Exzess = „Shift“ = $2^{8-1} = 2^7$
– $(-3)_{10} \rightarrow -3 + 128 = (125)_{10} = (01111101)_2$
- * Die Darstellungen sind mit denjenigen der Zweier-Komplement-Darstellung bis auf das invertierte linke Bit („Vorzeichen“) identisch

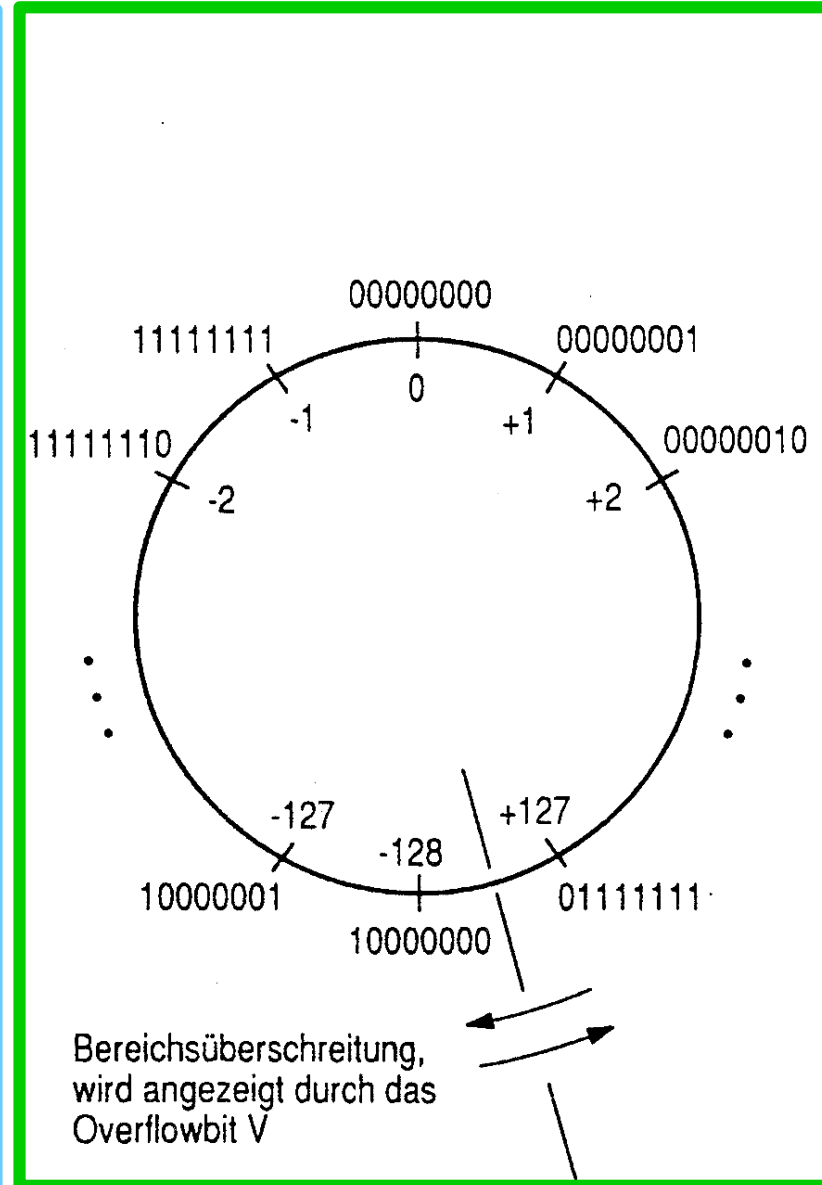
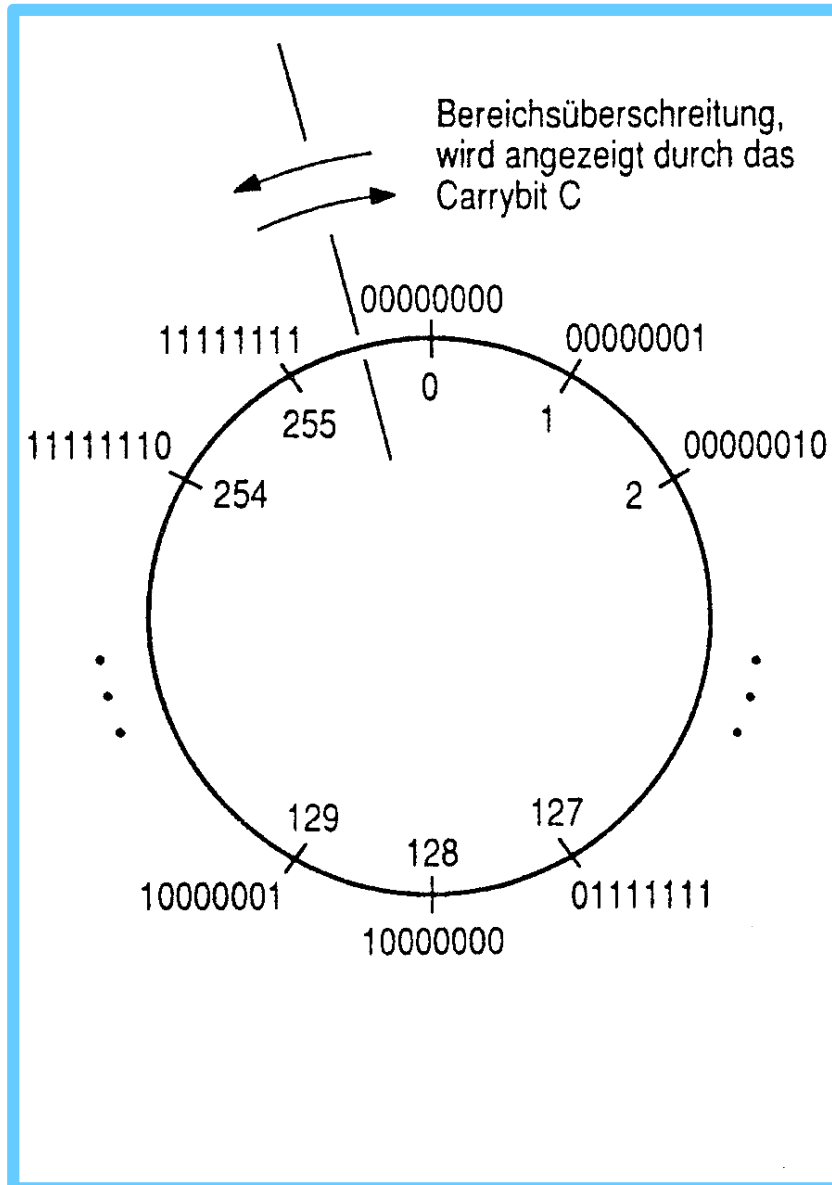
Negative Zahlen im Binärsystem (Dualsystem)

➤ Vergleich der 4 Systeme (n=4)

Dual	Vorz./Betrag	Einer-Kompl.	Zweier-Kompl.	Exzess
0000	+ 0	+ 0	0	- 8
0001	+ 1	+ 1	+ 1	- 7
0010	+ 2	+ 2	+ 2	- 6
0011	+ 3	+ 3	+ 3	- 5
0100	+ 4	+ 4	+ 4	- 4
0101	+ 5	+ 5	+ 5	- 3
0110	+ 6	+ 6	+ 6	- 2
0111	+ 7	+ 7	+ 7	- 1
1000	- 0	- 7	- 8	0
1001	- 1	- 6	- 7	+ 1
1010	- 2	- 5	- 6	+ 2
1011	- 3	- 4	- 5	+ 3
1100	- 4	- 3	- 4	+ 4
1101	- 5	- 2	- 3	+ 5
1110	- 6	- 1	- 2	+ 6
1111	- 7	- 0	- 1	+ 7

Negative Zahlen im Binärsystem (Dualsystem)

- Zahlenring (n=8) für Dualzahlen und Zweier-Komplement

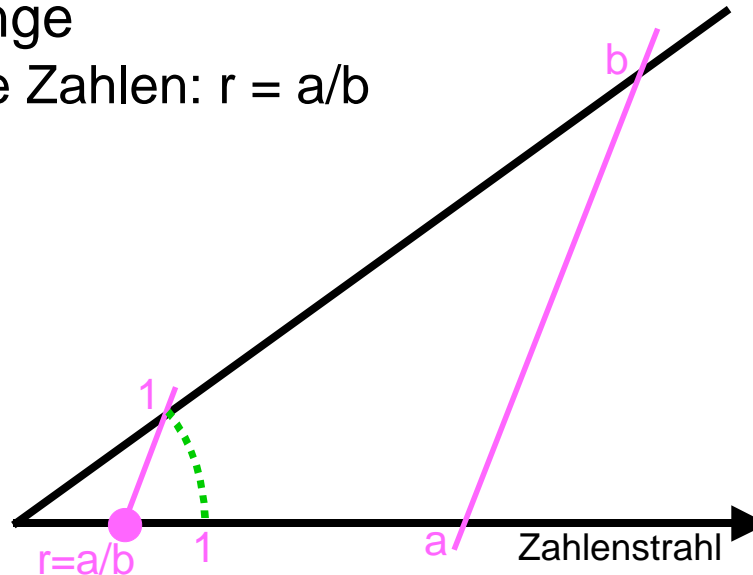


Gleitkommazahlen: Darstellung und Arithmetik

Gleitkommazahlen

➤ Zahlenmenge

- rationale Zahlen: $r = a/b$



- reelle Zahlen: Hinzunahme von nicht-rationalen Zahlen: π , e , $\sqrt{2}$
Die reellen Zahlen in ihrer mathematischen Bedeutung stellen ein **Kontinuum** dar (jedes beliebig große Intervall auf dem Zahlenstrahl enthält unendlich viele Werte)

Gleitkommazahlen

➤ Zahlenmenge (cont'd)

- Wertebereich REAL (Gleitkommazahlen) im Rechner stellt eine **endliche Menge von Repräsentanten** von Intervallen des Kontinuums dar. ⇒ **Diskretisierung**
 - * numerische Darstellung → Verarbeitung von Daten des Typs REAL **nicht** exakt (→ numerische Mathematik)



Gleitkommazahlen ~~↔~~ mathematisch reelle Zahlen
~~↔~~ mathematisch rationale Zahlen



Gleitkommazahlen

➤ Zahlendarstellung (Konrad Zuse, 1937)

$$zahl = m \cdot b^e$$

- m : Mantisse , $-M < m < +M$

Normalform: $1/b \leq |m| < 1$ oder $m = 0$

- b : Basis, z.B. 10, aber auch kleine Potenz von 2: 2, 4, 8, 16
- e : Exponent, $-E \leq e \leq +E$, auch $-E_1 \leq e \leq +E_2$
- alle Werte M , b , E sind rechnerabhängig
- Aufbau:

$$zahl = \pm 0.a_1a_2\dots a_\mu \cdot b^e \quad \text{mit } a_1 \neq 0 \text{ (normalisierte Darstellung)} \\ \text{und } 0 \leq a_i \leq b-1$$

- Beispiel: $b = 10$, Mantisse m : 3 Ziffern, Exponent e : 2 Ziffern

* Mantisse: $1/10 \leq |m| < 1$ oder 0

* Exponent: $-99 \leq e \leq +99$

* darstellbarer Bereich: $-0.999 \cdot 10^{99} \dots -0.100 \cdot 10^{-99}$
 $+0.100 \cdot 10^{-99} \dots +0.999 \cdot 10^{99}$

- **Multiplikation/Division** mit b : shift der Zahl-Ziffernfolge um 1 Stelle nach links/rechts oder $e \rightarrow e+1$ / $e \rightarrow e-1$

0.1 = 1/10 = 1/b ist kleinstmöglicher **Mantissen**betrag!

Warum?

Der nächstkleinere **Zahlen**betrag wäre: $0.099 \cdot 10^{-99}$

wegen normierter Darstellung: $0.099 \cdot 10^{-99} \rightarrow 0.99 \cdot 10^{-100}$

Das ist unmöglich wegen zwei-ziffrigen Exponent!

Gleitkommazahlen

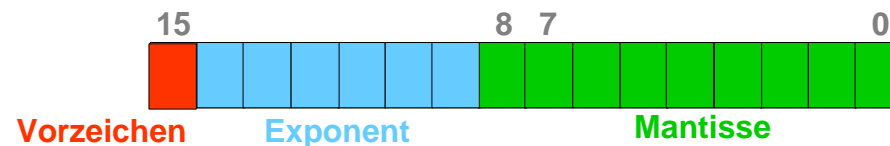
➤ Normalisierung

- Darstellung der Mantisse in Normalform: $1/b \leq |m| < 1$
- eine Mantisse mit gesetztem Führungsbit a_1 heisst normalisiert
zahl = $\pm 0.a_1a_2\dots a_\mu \cdot b^e$ mit $a_1 = 1$
- durch die Normalisierung wird die Gleitpunktdarstellung eindeutig

Die Mantisse ist somit um 1Bit länger als gedacht, weil a_1 nicht gespeichert werden muss!

➤ Rechnerinterne Repräsentation

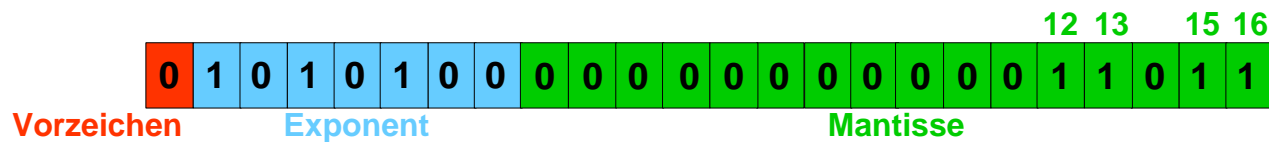
- der verfügbare Platz (hier 2 Byte) wird in Felder aufgeteilt



- Für arithmetische Operationen muss bei hardwaremäßiger Realisierung hoher Aufwand betrieben werden, daher
 - * Software-Realisierung
 - * Spezialprozessoren
 - * Leistungsdaten: MIPS, FLOPS

Gleitkommazahlen

➤ Beispiel einer 3 Byte breiten Zahlendarstellung (Basis $b = 2$)



* **Vorzeichen:** 0, also **+**

* **Exponent e :** Breite: 7 Bit
 Exzess **64**(= 2^7-1) - Darstellung
 $(1010100)_2 = (84)_{10} \rightarrow 84 - \mathbf{64} = \mathbf{20}$
 $b^e = 2^{20} = 1048576$

* **Mantisse m :** Breite: 16 Bit
 Darstellung der Mantisse:

$$\begin{aligned}
 m &= \sum_{j=1}^{16} a_j \cdot 2^{-j} = 1 \cdot 2^{-12} + 1 \cdot 2^{-13} + 1 \cdot 2^{-15} + 1 \cdot 2^{-16} \\
 &= (1 \cdot 2^{+4} + 1 \cdot 2^{+3} + 1 \cdot 2^{+1} + 1 \cdot 2^0) \cdot 2^{-16} \\
 &= 27 \cdot 2^{-16}
 \end{aligned}$$

* **Zahl:** $\text{zahl} = \mathbf{+} 27 \cdot 2^{-16} \cdot 2^{20} = \mathbf{+} 432$

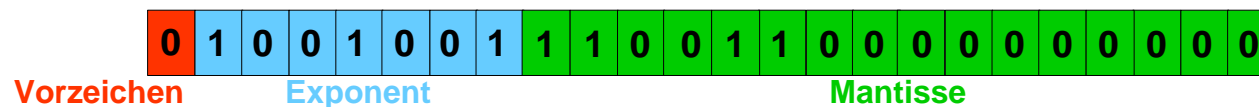
Gleitkommazahlen

➤ Beispiel (cont'd)

- Normalisierung:

$$\begin{aligned}zahl &= 2^{20} \cdot (1 \cdot 2^{-12} + 1 \cdot 2^{-13} + 0 \cdot 2^{-14} + 1 \cdot 2^{-15} + 1 \cdot 2^{-16}) \\ &= 2^{20} \cdot (2^{-11} \cdot (1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} + 1 \cdot 2^{-5})) \\ &= 2^9 \cdot (1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} + 1 \cdot 2^{-5})\end{aligned}$$

- * **Vorzeichen:** bleibt 0, also +
- * **Exponent e:** Breite: 7 Bit
Exzess 64(=2⁷-1) - Darstellung
9 + 64 = 73 → (73)₁₀ = (1001001)₂
Exponent e um 11 dekrementiert
- * **Mantisse m:** Breite: 16 Bit
Mantisse um 11 Bit nach links geschoben

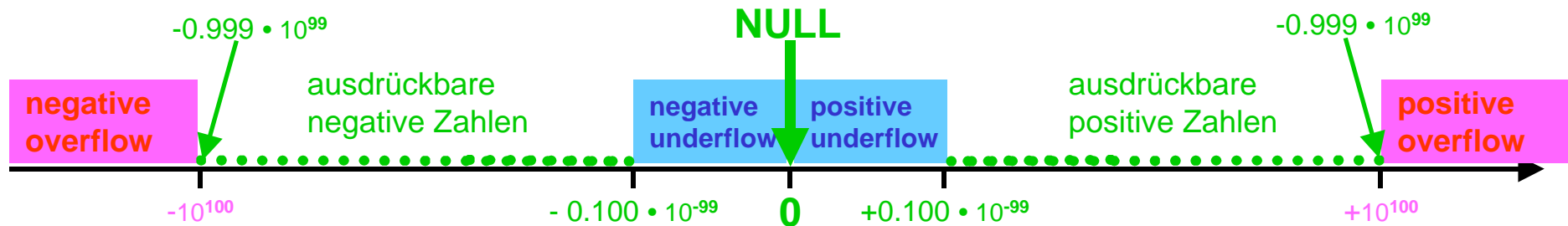


- * Zahl bleibt erhalten: $zahl = + 27 \cdot 2^{-5} \cdot 2^9 = + 432$

Gleitkommazahlen

➤ REAL-Zahlen auf dem Zahlenstrahl

- Beispiel: $b = 10$, Mantisse m : 3 Ziffern, Exponent e : 2 Ziffern



- jede REAL-Zahlen repräsentiert ein Intervall der reellen Zahlen; das Intervall wächst mit zunehmendem Betrag der Zahl, d.h. die **Dichte der Repräsentation** nimmt mit zunehmendem Betrag der Zahl ab.
- Eine Abschätzung des Einflusses der Ungleichverteilung der Repräsentanten auf die Rechenoperationen ist nicht trivial.
- Behandlung von **overflow/underflow**, **Null**, „undefiniert“?
→ IEEE Floating-Point Standard 754 (1985) (siehe A.S. Tanenbaum)

Gleitkommazahlen

➤ Probleme

- Test: **Assoziativgesetz** (Beispiel mit 4-stelliger Arithmetik)

$$x = 9.900, y = 1.000, z = -0.999$$

$$(x+y) + z = 10.90 + (-0.999) = 9.910$$

$$x + (y+z) = 9.900 + 0.001 = 9.901$$

- Test: **Distributivgesetz** (Beispiel mit 4-stelliger Arithmetik)

$$x = 1100., y = -5.000, z = 5.001$$

$$(x \cdot y) + (x \cdot z) = (-5500) + 5501 = 1.000$$

$$x \cdot (y+z) = 1100. \cdot 0.001 = 1.100$$

- Auslöschung

Bei der Subtraktion zweier fast gleich großer Werte heben sich die signifikanten Ziffern auf und die Differenz verliert dadurch an Signifikanz (z.B. Differenzenquotient)

- Überlaufgefahr

... bei Division durch kleine Werte

Gleitkommazahlen

➤ Rechnerarithmetik

$$x = m_x \cdot 2^{e_x}, \quad y = m_y \cdot 2^{e_y}$$

- Addition

$$x + y = (m_x \cdot 2^{e_x - e_y} + m_y) \cdot 2^{e_y} \quad \text{falls } e_x \leq e_y$$

- Subtraktion

$$x - y = (m_x \cdot 2^{e_x - e_y} - m_y) \cdot 2^{e_y} \quad \text{falls } e_x \leq e_y$$

- Multiplikation

$$x \cdot y = (m_x \cdot m_y) \cdot 2^{e_x + e_y}$$

- Division

$$x \div y = (m_x \div m_y) \cdot 2^{e_x - e_y}$$