

Übung 1

- ❑ Darstellung negativer Zahlen
 - Addition von Zahlen in Zweierkomplement-Form
- ❑ Gleitkommazahlen
 - Zahlen im IEEE-754 Floating-Point-Standard
 - Addition von GK-Zahlen
 - Multiplikation von GK-Zahlen
- ❑ Hammingkode
 - 1-Bit-Fehlererkennung und –korrektur
 - 2-Bit-Fehlererkennung



Wichtig

Sei $z = b_{n-1} \dots b_0$ eine n-stellige Dualzahl

□ Das Zweierkomplement einer n-stelligen Zahl z ist $2^n - z$

□ Vorgegebene Wortlänge $n \rightarrow N = 2^n$ Zahlen

➤ Darstellung einer positiven Zahl z : $+z = z$

➤ Darstellung einer negativen Zahl $-z$: $-z = N - z$

➤ Beispiel: $n = 4 \rightarrow N = 2^4 = 16$, Darstellung von -5 ?

$n = 4$

1010

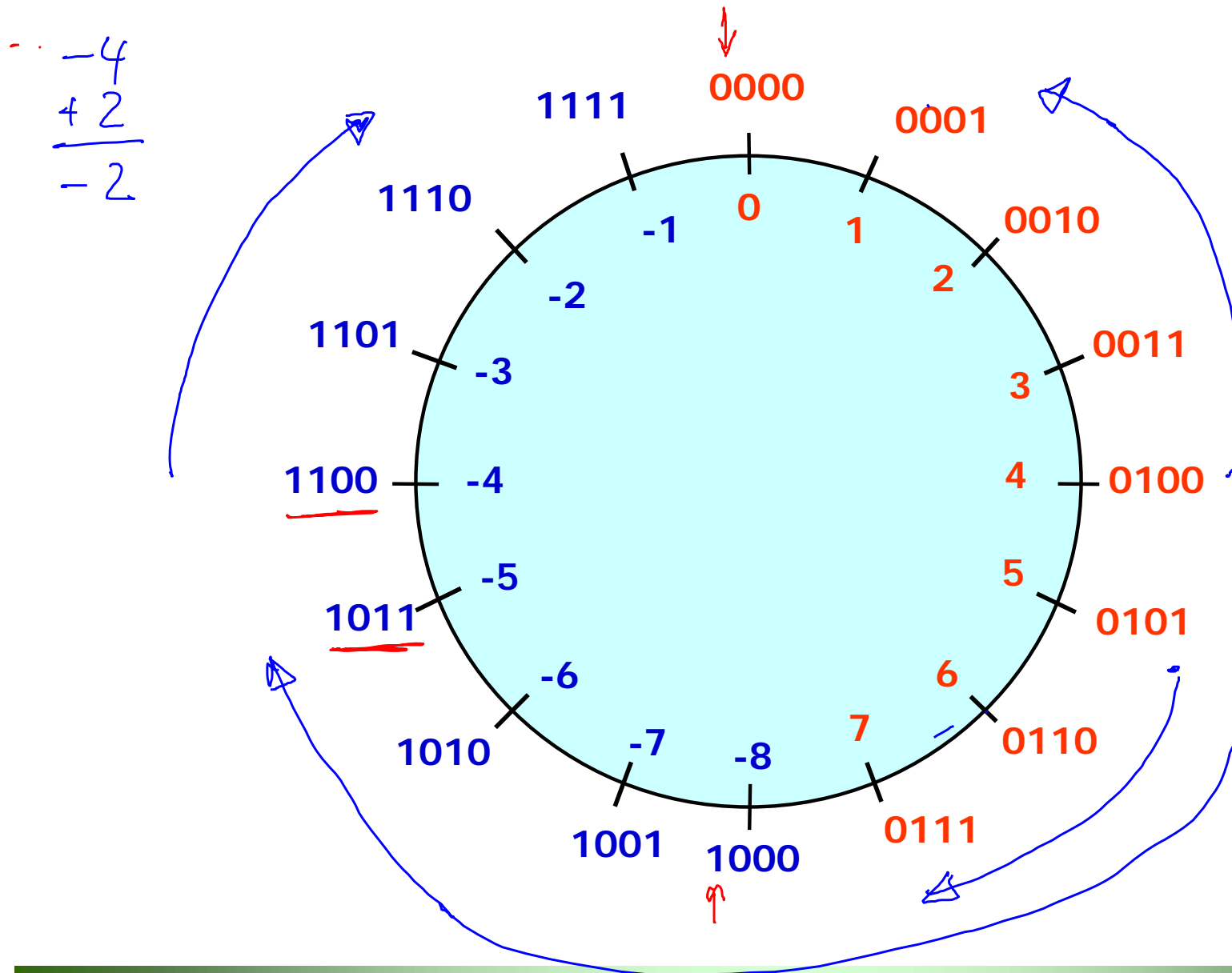
1001

$\frac{16 - 5}{11}$

~~101~~
~~010~~
~~1~~
~~011~~
0101
1010
1
1011



Zweierkomplement-Darstellung



$a = 4$
 $-8 \leq z \leq 7$
 $+4$
 -3

 5
 $+6$

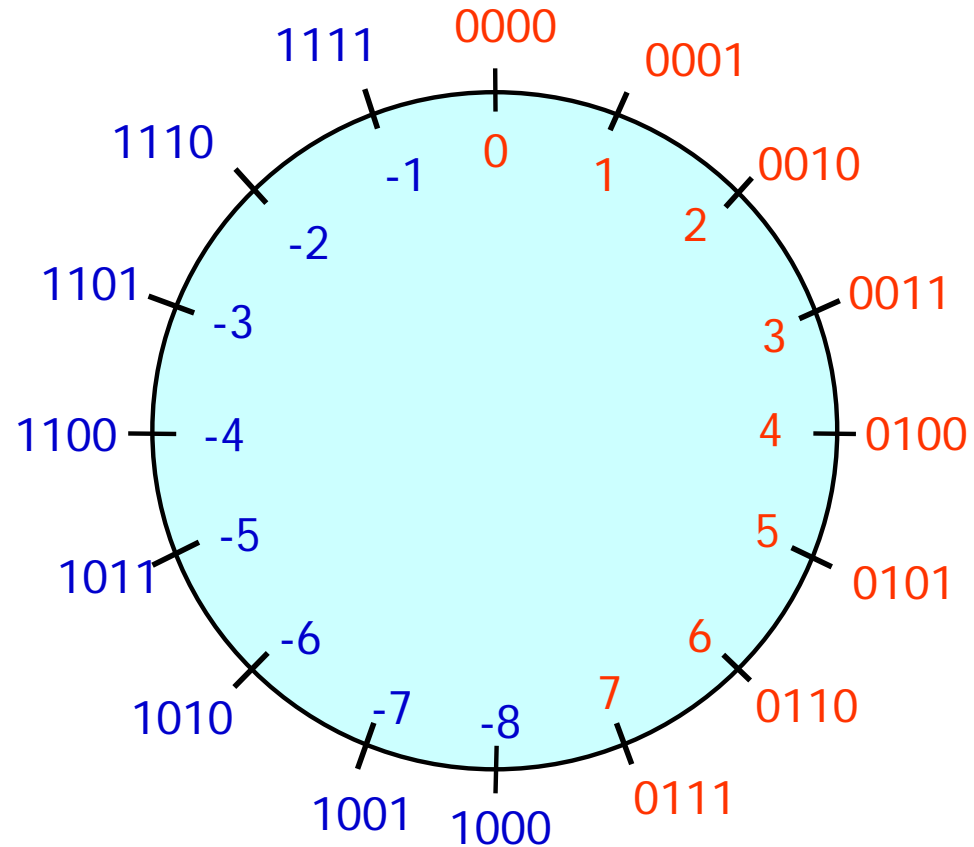
 $+1011$
 0101
 0110

 1011
 $-2^3 + 2^1 + 2^0$

Addition von ZK-Zahlen

$$\begin{array}{r} 0000 \\ 1100 - 4 \\ 0010 + 2 \\ \hline 1110 - 2 \end{array}$$

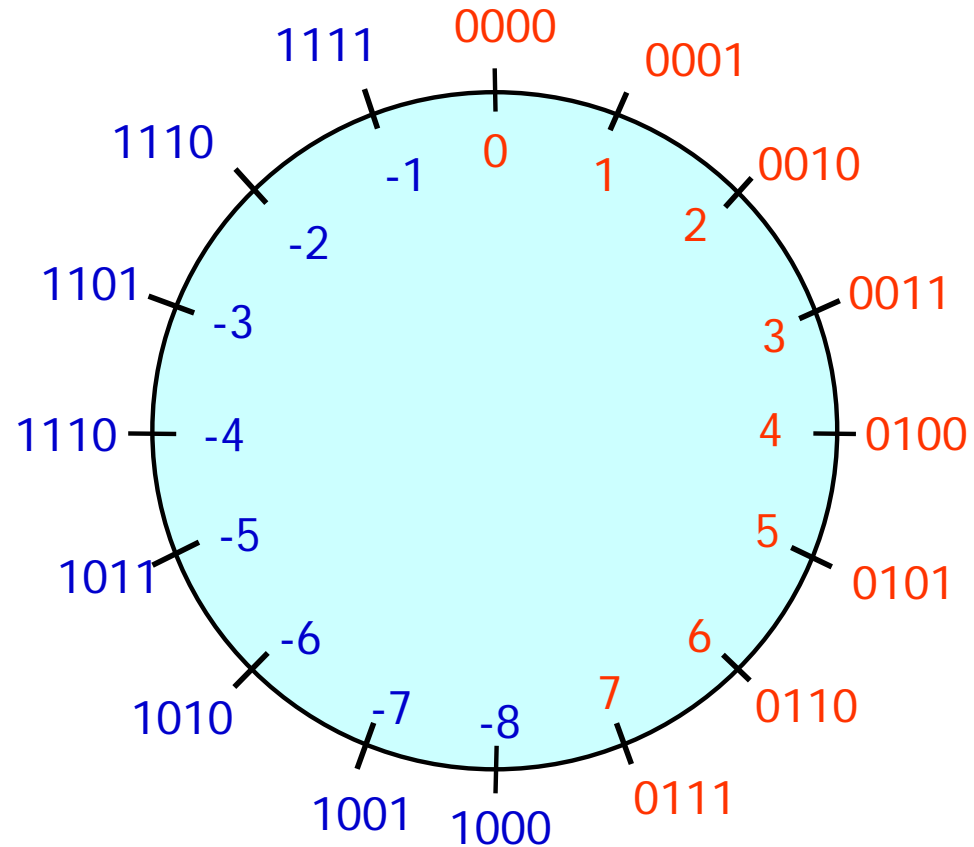
$$\begin{array}{r} 1100 \\ 1110 - 2 \\ 0100 + 4 \\ \hline 10010 + 2 \end{array}$$



Subtraktion von ZK-Zahlen

$$\begin{array}{r} 0000 \\ 0001 + 1 \\ 1100 - 4 \\ \hline 1101 - 3 \end{array}$$

$$\begin{array}{r} 1100 \\ 1111 - 1 \\ 1100 - 4 \\ \hline 11011 - 5 \end{array}$$



$y - x$ kann zurückgeführt werden auf:

$$y + 2^n - x$$

- $x < y$:

$$y + 2^n - x = \underline{2^n} + (y - x)$$

zuviel

$$\begin{array}{r} 0011 \\ 1110 \\ \hline 0001 \end{array} \quad \begin{array}{r} +3 \\ -2 \\ +1 \end{array}$$

- $y < x$:

$$y + 2^n - x = 2^n - |y - x|$$

$$\begin{array}{r} 0010 \\ 1101 \\ \hline 1111 \end{array} \quad \begin{array}{r} +2 \\ -3 \\ -1 \end{array}$$

- $(-y) + (-x) = (2^n - y) + (2^n - x) = 2^n + \underline{2^n - (y + x)}$

*-1
-3*



Addition von Zweierkomplement-Zahlen (1)

Bei der Addition lassen sich 3 Fälle unterscheiden:

1) Beide Summanden sind positiv

- die Vorzeichenbits beider Zahlen sind 0.
- das Ergebnis muss positiv sein
- Das Ergebnis ist nur dann korrekt, wenn sein Vorzeichenbit gleich 0 ist, ansonsten wurde der Zahlenbereich überschritten.

Man kann sich diese Situation anhand des Zahlenkreises klarmachen.



Beispiel 1

$$\begin{array}{r} 5 \\ + 2 \\ \hline = 7 \end{array}$$

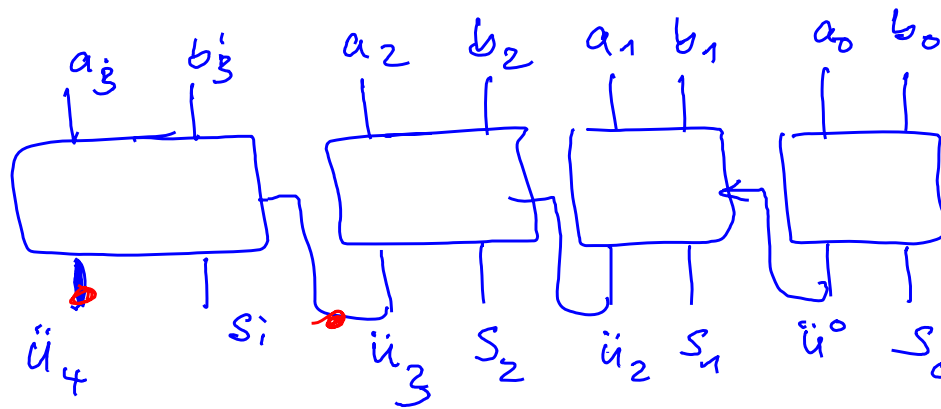
$$\begin{array}{r} 0101 \\ + 0010 \\ \hline 0000 \\ \hline 0111 \end{array}$$

Überträge gleich
Ergebnis korrekt

$$\begin{array}{r} 5 \\ + 6 \\ \hline = 11 \end{array}$$

$$\begin{array}{r} 0101 \\ + 0110 \\ \hline 0100 \\ \hline 1011 \end{array}$$

Überträge ungleich
Überlauf
Ergebnis falsch



Addition von Zweierkomplement-Zahlen (2)

2) Beide Summanden sind negativ

- die Vorzeichenbits beider Zahlen haben den Wert 1.
- Das Ergebnis muss negativ sein.
- Das Ergebnis ist nur dann korrekt, wenn das Vorzeichenbit des Ergebnisses 1 ist.
- Die beiden vordersten Überträge müssen den gleichen Wert haben.



Beispiel 2

$$\begin{array}{r} -5 \\ + (-2) \\ \hline = -7 \end{array}$$

$$\begin{array}{r} + \quad 1011 \\ \quad 1110 \\ \quad \boxed{11}10 \\ \hline \quad 1001 \end{array}$$

Überträge gleich
Ergebnis korrekt

$$\begin{array}{r} -5 \\ + (-6) \\ \hline = -11 \end{array}$$

$$\begin{array}{r} \quad 1011 \\ \quad 1010 \\ \quad \boxed{10}10 \\ \hline \quad 0101 \end{array}$$

Überträge ungleich
Überlauf
Ergebnis falsch



Addition von Zweierkomplement-Zahlen (3)

3) Summanden haben unterschiedliche Vorzeichen:

- Das Ergebnis ist auf jeden Fall korrekt, das Vorzeichen hängt davon ab, ob Subtrahend oder Minuend betragsmäßig größer ist
- Der Übertrag aus der vordersten Stelle ist zu streichen



Beispiel 3

$$\begin{array}{r} 5 \\ + (-6) \\ \hline = -1 \end{array}$$

$$\begin{array}{r} 0101 \\ 1010 \\ \boxed{00}00 \\ \hline 1111 \end{array}$$

Überträge gleich
Ergebnis korrekt

$$\begin{array}{r} -5 \\ + 6 \\ \hline = 1 \end{array}$$

$$\begin{array}{r} 1011 \\ 0110 \\ \boxed{11}10 \\ \hline 0001 \end{array}$$

Überträge gleich
Ergebnis korrekt



Überlauferkennung

Allgemeine Überlauferkennung bei dualer Addition:

□ **korrekte Addition:** beide Überträge sind gleich.

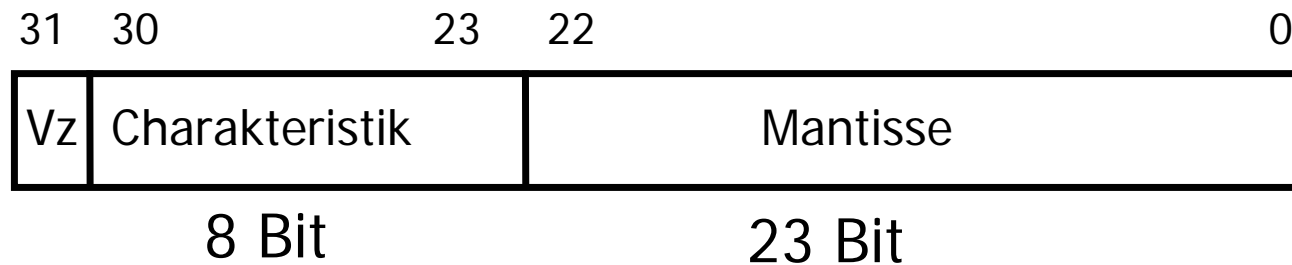
□ **Überlauf:** beide Überträge sind ungleich.

Realisierung z. B. durch ein Antivalenzgatter

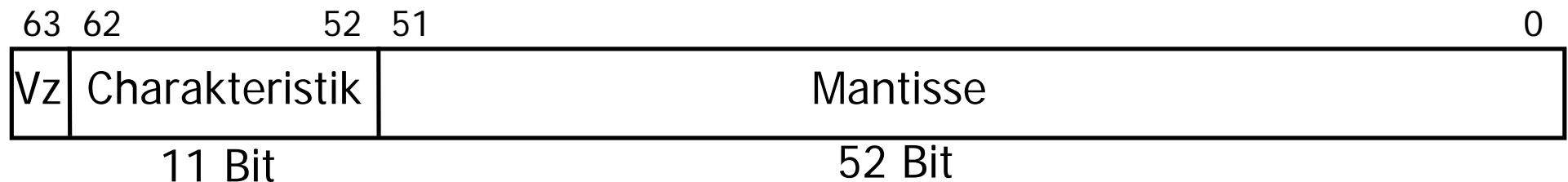


IEEE-P 754-Floating-Point-Standard

32-Bit Maschinenformate des IEEE-Standards:



64-Bit Maschinenformate des IEEE-Standards:



Zusammenfassung des 32-Bit-IEEE-Formats

Charakteristik	Zahlenwert
0	$(-1)^{Vz} 0, \text{Mantisse} \cdot 2^{-126}$
1	$(-1)^{Vz} 1, \text{Mantisse} \cdot 2^{-126}$
...	$(-1)^{Vz} 1, \text{Mantisse} \cdot 2^{\text{Charakteristik}-127}$
254	$(-1)^{Vz} 1, \text{Mantisse} \cdot 2^{127}$
255	<i>Mantisse = 0: $(-1)^{Vz} \infty$ overflow</i>
255	<i>Mantisse $\neq 0$: NaN (not a number)</i>

$$Exp_{min} \leq Exp \leq Exp_{max}$$

$$-126 \leq Exp \leq 127$$



IEEE-P 754 Format

$$\mathbf{Exp = Exp_{min} = -126 \Rightarrow Char = Exp + 127 = 1}$$

$$0000\ 0000\ 1000\ 0000\ \dots\ 0000 = (1,0\dots00)_2 \cdot 2^{-126} = \mathbf{minreal}$$

$$0000\ 0000\ 1000\ 0000\ \dots\ 0001 = (1,0\dots01)_2 \cdot 2^{-126} = (1+2^{-23}) \cdot 2^{-126}$$

$$0000\ 0000\ 1000\ 0000\ \dots\ 0010 = (1,0\dots10)_2 \cdot 2^{-126} = (1+2^{-22}) \cdot 2^{-126}$$

$$0000\ 0000\ 1000\ 0000\ \dots\ 0011 = (1,0\dots11)_2 \cdot 2^{-126} = (1+2^{-22} + 2^{-23}) \cdot 2^{-126}$$

$$\mathbf{Exp = -125 \Rightarrow Char = 2}$$

$$0000\ 0001\ 0000\ 0000\ \dots\ 0000 = (1,0\dots00)_2 \cdot 2^{-125}$$

$$0000\ 0001\ 0000\ 0000\ \dots\ 0001 = (1,0\dots01)_2 \cdot 2^{-125} = (1 + 2^{-23}) \cdot 2^{-125}$$



IEEE-P 754 Format

$Exp = 0 \Rightarrow Char = 127$

$$0011\ 1111\ 1000\ 0000\ \dots\ 0000 = (1,0 \dots 00)_2 \cdot 2^0 = 1$$

$$0011\ 1111\ 1000\ 0000\ \dots\ 0001 = (1,0 \dots 01)_2 \cdot 2^0 = 1 + 2^{-23}$$

$Exp = 127 \Rightarrow Char = 254$

$$0111\ 1111\ 0000\ 0000\ \dots\ 0000 = (1,0 \dots 00)_2 \cdot 2^{127} = 2^{127}$$

$$0111\ 1111\ 0000\ 0000\ \dots\ 0001 = (1,0 \dots 01)_2 \cdot 2^{127} = (1 + 2^{-23}) \cdot 2^{127}$$

$$0111\ 1111\ 0111\ 1111\ \dots\ 1111 = (1,11 \dots 11)_2 \cdot 2^{127}$$
$$= (1 + 2^{-1} + 2^{-2} + \dots + 2^{-22} + 2^{-23}) \cdot 2^{127}$$

$$= (1 + (1 - 2^{-24})) \cdot 2^{127} = 2^{128} - 2^{104} \quad = \textit{maxreal}$$



IEEE-P 754 Format

$$Exp = E_{min} - 1 = -127 \Rightarrow Char = 0$$

➤ Darstellung von ± 0

$$0000\ 0000\ 0000\ 0000\ \dots\ 0000^{\uparrow} = +0$$

$$1000\ 0000\ 0000\ 0000\ \dots\ 0000 = -0$$

➤ Darstellung denormalisierter Zahlen

$$0000\ 0000\ 0000\ 0000\ \dots\ 0001 = (0,0 \dots 01)_2 \cdot 2^{-126} = 2^{-23} \cdot 2^{-126} = 2^{-149}$$

$$0000\ 0000\ 0000\ 0000\ \dots\ 0010 = (0,0 \dots 10)_2 \cdot 2^{-126} = 2^{-22} \cdot 2^{-126} = 2^{-148}$$

$$\begin{aligned} 0000\ 0000\ 0111\ 1111\ \dots\ 1111 &= (0,1 \dots 11)_2 \cdot 2^{-126} \\ &= (2^{-1} + 2^{-2} + \dots + 2^{-23}) \cdot 2^{-126} \\ &= (1 - 2^{-24}) \cdot 2^{-126} \end{aligned}$$

Kleinste denormalisierte Zahl

Größte denormalisierte Zahl



IEEE-P 754 Format

$$Exp = E_{max} + 1 = 128 \Rightarrow Char = 255$$

- Darstellung von $\pm\infty$

$$0111\ 1111\ 1000\ 0000\ \dots\ 0000 = +\infty$$

$$1111\ 1111\ 1000\ 0000\ \dots\ 0000 = -\infty$$

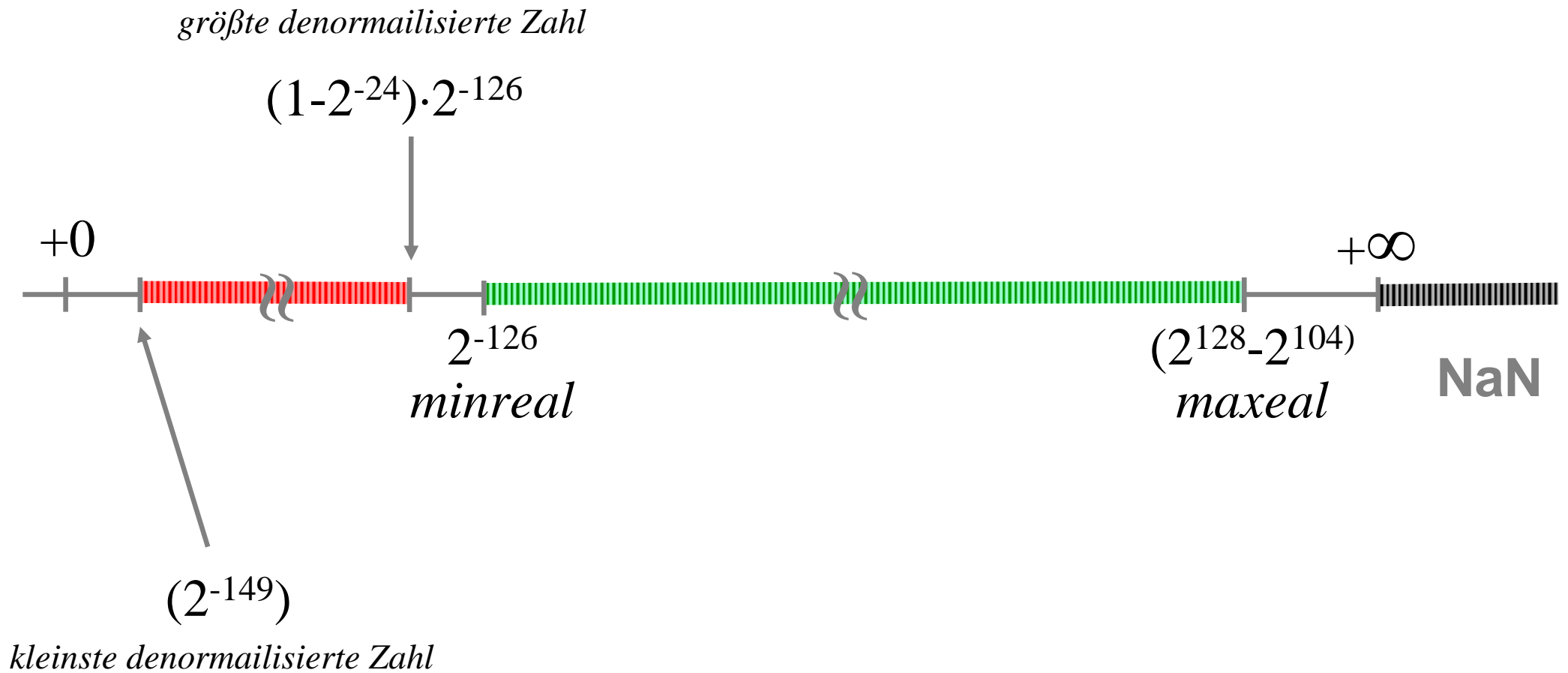
- Darstellung von NaN:

$$0111\ 1111\ 1000\ 0000\ \dots\ 0001 \Rightarrow \mathbf{NaN}$$

$$M \neq 0$$



Zusammenfassung der 32-Bit Format



Zusammenfassung der 32-Bit Format

- **Normalisierte GK-Zahlen: 254 verschiedene Exponenten**

⇒ $2 \cdot 254 \cdot 2^{23} = 127 \cdot 2^{25}$ normalisierte Zahlen

- **Denormalisierte GK-Zahlen (Char = 0)**

⇒ $2 \cdot 2^{23} = 2^{24}$ denormalisierte Zahlen (inkl. ± 0)

- **NaN (Char = 255)**

⇒ $2 \cdot 2^{23} = 2^{24}$ NaN (inkl $\pm\infty$)

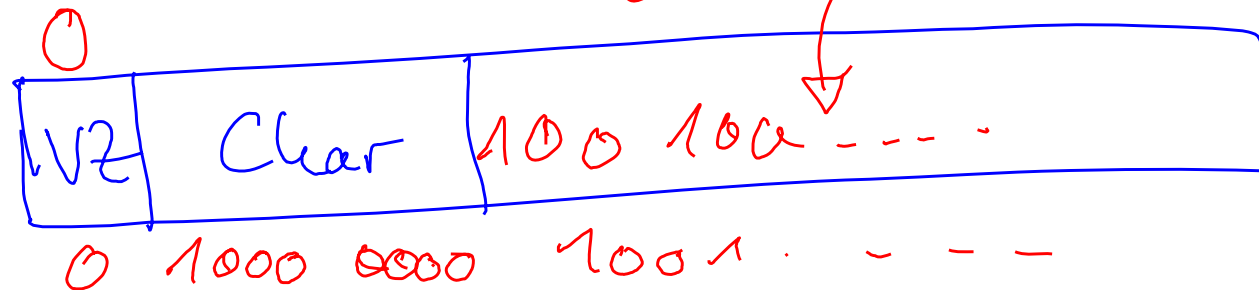


Aufgabe 1

Die Zahl π mit 7 Dezimalstellen hinter dem Komma im 32-Bit-IEEE-Format.

$$\begin{aligned}\pi \approx 3,1415926_{10} &= \underline{11},0010010000111111011010_2 \\ &= 1,1001001000011111101101 \cdot 2^1 \\ &= 1,17 \cdot 2^1\end{aligned}$$

$$\begin{aligned}\text{Char} &= \text{Exp} + 127 \\ &= 1 + 127 = 128_{10}\end{aligned}$$



Aufgabe 1

Die Zahl π mit 7 Dezimalstellen hinter dem Komma im 32-Bit-IEEE-Format.

$$\begin{aligned}\pi &\approx 3,1415926_{10} = 11,0010010000111111011010_2 \\ &= (-1)^0 \cdot 2^{(128-127)} \cdot 1,10010010000111111011010_2 \\ &= (-1)^0 \cdot 2^1 \cdot \mathbf{1,10010010000111111011010}_2\end{aligned}$$

$$VZ = 0$$

$$Exp = 1 \Rightarrow Char = 127+1 = 128_{10} = \mathbf{1000\ 0000}_2$$

$$\begin{aligned}\pi &= \mathbf{0100\ 0000\ 0100\ 1001\ 0000\ 1111\ 1101\ 1010} \\ &= \mathbf{4\ 0\ 4\ 9\ 0\ F\ D\ A}_{16}\end{aligned}$$

Aufgabe 2

Gegeben sei eine Gleitkomma-Zahl im 32-Bit-IEEE-Format.

$Z = 01000001111000000000000000000000$

Zahl in Dezimal-Darstellung?

0100 0001 1110 0000 0000 0000 0000 0000

$$\text{Char. } 128 + 2 + 1 = 131 \quad Z = \pm 1,11 \cdot 2^{\text{Exp}}$$

$$\text{Exp} = \text{Char} - 127 = 131 - 127 = \underline{\underline{4}}$$

$$\begin{aligned} Z &= + (1,110\dots 0) \cdot 2^4 = \left(1 + \frac{1}{2} + \frac{1}{4}\right) \cdot 2^4 \\ &= \left(1 + \frac{3}{4}\right) \cdot 2^4 = 28_{10} \end{aligned}$$



Aufgabe 2

Gegeben sei eine Gleitkomma-Zahl im 32-Bit-IEEE-Format.

$Z = 01000001111000000000000000000000$

Zahl in Dezimal-Darstellung?

0100 0001 1110 0000 0000 0000 0000 0000

$VZ = 0$

$Char = 128 + 2 + 1 = 131 \Rightarrow Exp = 131 - 127 = 4_{10}$

$Mantisse = 110\ 0000\ 0000\ 0000\ 0000\ 0000_2$

$$\begin{aligned} Z &= (-1)^0 \cdot 2^4 \cdot (1,1100\ 0000 \dots 0000)_2 \\ &= + (1 + 0,5 + 0,25)_{10} \cdot 2^4 \\ &= + 1,75 \cdot 2^4 = 1,75 \cdot 16 = 28 \end{aligned}$$



Aufgabe

Gegeben seien zwei 32-Bit Gleitkommazahlen x und y im IEEE Standard

$x = 0 \underbrace{0000\ 0001}_{\text{Mantel}}\ 000\ 0000\ 0000\ 0000\ 0000\ 0001$


$y = 0 \underbrace{0000\ 0001}_{\text{Mantel}}\ 000\ 0000\ 0000\ 0000\ 0000\ 0000$

Gesucht: $1/(x-y)$



Lösung

□ Implementierung im Rechner:

- 1. Vorschlag: $z := 1 / (x - y)$
- 2. Vorschlag: $\text{if } (x \neq y) \text{ then } z := 1 / (x - y)$ 

$x =$ 0 0000 0001 000 0000 0000 0000 0000 0001

$y =$ 0 0000 0001 000 0000 0000 0000 0000 0000

$$\left. \begin{aligned} x &= (1 + 2^{-23}) \cdot 2^{-126} \\ y &= 1 \cdot 2^{-126} \end{aligned} \right\} x - y = 2^{-149}$$



Addition von Gleitkommazahlen

Addiere die beiden GK-Zahlen (im IEEE-Standard)

0100 0011 1110 0000 ... 0000 und

0100 0010 0101 0000 ... 0000

$$\pm m_x \cdot 2^{e_x}$$

$$+ m_y \cdot 2^{e_y}$$



Multiplikation von Gleitkommazahlen

Multipliziere die beiden GK-Zahlen (im IEEE-Standard)

3FE0 0000 und

3E40 0000



Aufgaben

- Hamming-Code
 - Erkennen und korrigieren von 1-Bit-Fehlern
 - Erkennen von 2-Bit-Fehlern



Ein-Bit-Fehlerkorrektur

Notwendiger Aufwand für die sog. *Hammingcodes*:

$$2^k \geq m + k + 1$$

k: Zahl der zusätzlichen Korrekturbits

m: Zahl der Datenbits



Mindestaufwand: Ein-Bit Fehlerkorrekturcodes

Zahl der Datenbits m	Zahl der Prüfbits k	Bits insgesamt: $m + k$
1	2	$m + 2$
2 bis 4	3	$m + 3$
5 bis 11	4	$m + 4$
12 bis 26	5	$m + 5$
27 bis 57	6	$m + 6$
6.4	... 7	...



17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Position
	4								3				2		1	0	i
m_{12}	k_5	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	k_4	m_4	m_3	m_2	k_3	m_1	k_2	k_1	Stellen

Datenwort: 1 0 1 0 1 0 1 0 $m = 8$

Datenwort ausspreizen: $k = 4$

12	11	10	9	8	7	6	5	4	3	2	1
1	0	1	0	-	1	0	1	-	0	-	-
m_8	m_7	m_6	m_5	k_4	m_4	m_3	m_2	k_3	m_1	k_2	k_1

- 0001
- 0010
- 0011
- 0100
- 0101
- 0110
- 0111
- 1000
- 1001
- 1010
- 1011
- 1100
- 1101
- 1110
- 1111
- ...

Bestimmung der Prüfbits:

$$\begin{aligned}
 k_1 &= m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 = 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 0 \\
 k_2 &= m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0 \\
 k_3 &= m_2 \oplus m_3 \oplus m_4 \oplus m_8 = 1 \oplus 0 \oplus 1 \oplus 1 = 1 \\
 k_4 &= m_5 \oplus m_6 \oplus m_7 \oplus m_8 = 0 \oplus 1 \oplus 0 \oplus 1 = 0
 \end{aligned}$$

→ Codewort: 1 0 1 0 0 1 0 1 1 0 0 0



Aufgabe 2

Der Hauptspeicher eines Rechners mit 8-Bit Datenwortbreite unterstützt eine Einzelbitfehler-Korrektur. Aus dem Speicher erhält man die beiden Codewörter

- Codewort 1: **101001000110**
- Codewort 2: **100000000000**

Prüfen Sie beide Codewörter auf Fehler, die beim Übertragen oder Speichern entstanden sein könnten und korrigieren Sie diese (falls notwendig). Geben Sie die zugehörigen Datenwörter an.



17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Position
	4								3				2		1	0	i
m_{12}	k_5	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	k_4	m_4	m_3	m_2	k_3	m_1	k_2	k_1	Stellen

Hammingcode mit 8 Datenbits und 4 Prüfbits

12 11 10 9 8 7 6 5 4 3 2 1

Codewort 1 : 1 0 1 0 0 1 0 0 0 1 1 0

m_8 m_7 m_6 m_5 k_4 m_4 m_3 m_2 k_3 m_1 k_2 k_1

Bestimmung der Prüfbits:

$$k_1 = k_1 \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 = 0$$

$$k_2 = \bar{k}_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$k_3 = k_3 \oplus m_2 \oplus m_3 \oplus m_4 \oplus m_8 = 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = 0$$

$$k_4 = k_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_8 = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

$k_4 k_3 k_2 k_1 = 0000 \rightarrow$ Codewort ist fehlerfrei

\rightarrow Datenwort: 1 0 1 0 1 0 0 1



17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Position
	4								3				2		1	0	i
m_{12}	k_5	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	k_4	m_4	m_3	m_2	k_3	m_1	k_2	k_1	Stellen

12 11 10 9 8 7 6 5 4 3 2 1

Codewort 2 : 1 0 0 0 0 0 0 0 0 0 0 0

m_8 m_7 m_6 m_5 k_4 m_4 m_3 m_2 k_3 m_1 k_2 k_1

Bestimmung der Prüfbits:

$$\begin{aligned}
 k_1 &= k_1 \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 0 \\
 k_2 &= k_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 0 \\
 k_3 &= k_3 \oplus m_2 \oplus m_3 \oplus m_4 \oplus m_8 = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 1 \\
 k_4 &= k_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_8 = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 1
 \end{aligned}$$

$k_4 k_3 k_2 k_1 = 1100 = 12_{10} \rightarrow$ **Fehler in der 12. Position**

\rightarrow **Datenwort:** 0 0 0 0 0 0 0 0



Hammingkode: Beispiel 2-Bitfehler

- Fall 1: Kein Fehler

m_5 k_4 m_4 m_3 m_2 k_3 m_1 k_2 k_1 **PB**
0 0 1 1 0 0 1 1 0 0

$$k_4k_3k_2k_1 = 0000$$

- Fall 2: 1-Bit-Fehler

0 0 1 0 0 0 1 1 0 1

$$k_4k_3k_2k_1 = 0110$$

Fehler an 6. Position

- Fall 3: 2-Bitfehler

0 0 1 0 0 0 0 1 0 0

$$k_4k_3k_2k_1 = 0101$$

2-Bit-Fehler



Hammingcode-Applet

- TI-Homepage

ti.itec.uka.de

