

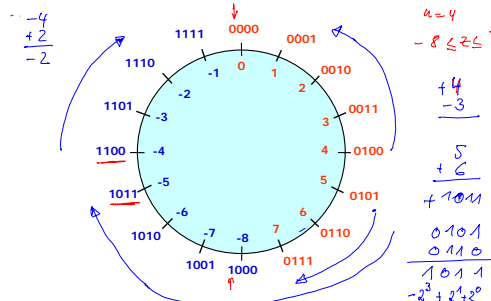
# Übung 1

- 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
- Hammingcode
  - 1-Bit-Fehlererkennung und -korrektur
  - 2-Bit-Fehlererkennung

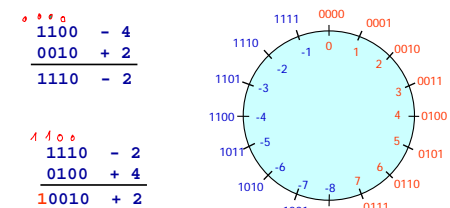
# Wichtig

- 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  $-5$ ?
- Handwritten notes:*  $n=4, N=16, 16-5=11, 1011$

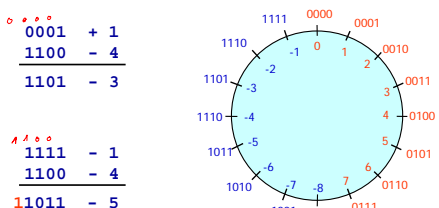
# Zweierkomplement-Darstellung



# Addition von ZK-Zahlen



# Subtraktion von ZK-Zahlen



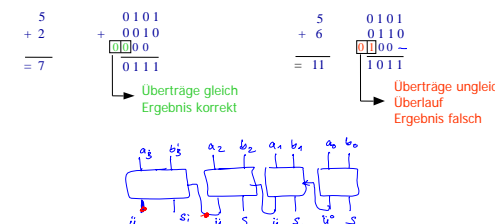
# Addition von Zweierkomplement-Zahlen (1)

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

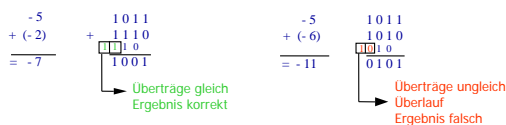


# Addition von Zweierkomplement-Zahlen (2)

## 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

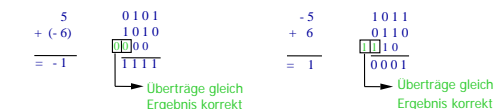


# Addition von Zweierkomplement-Zahlen (3)

## 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



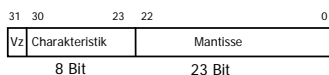
# Überlauferkennung

## Allgemeine Überlauferkennung bei dualer Addition:

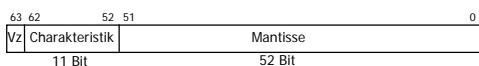
- **korrekte Addition:** beide Überträge sind gleich.
- **Überlauf:** beide Überträge sind ungleich. Realisierung z. B. durch ein Antivalenzgatter

# 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	$(-1)^z \cdot 0, \text{Mantisse} \cdot 2^{-126}$
1	$(-1)^z \cdot 1, \text{Mantisse} \cdot 2^{-126}$
...	$(-1)^z \cdot 1, \text{Mantisse} \cdot 2^{\text{Charakteristik}-127}$
254	$(-1)^z \cdot 1, \text{Mantisse} \cdot 2^{127}$
255	Mantisse = 0: $(-1)^z \cdot \infty$ overflow
255	Mantisse $\neq 0$ : NaN (not a number)

$$Exp_{min} \leq Exp \leq Exp_{max} \quad -126 \leq Exp \leq 127$$

# IEEE-P 754 Format

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

- 0000 0000 1000 0000 ... 0000 =  $(1,0 \dots 00)_2 \cdot 2^{-126} = \text{minreal}$
- 0000 0000 1000 0000 ... 0001 =  $(1,0 \dots 01)_2 \cdot 2^{-126} = (1+2^{-23}) \cdot 2^{-126}$
- 0000 0000 1000 0000 ... 0010 =  $(1,0 \dots 10)_2 \cdot 2^{-126} = (1+2^{-22}) \cdot 2^{-126}$
- 0000 0000 1000 0000 ... 0011 =  $(1,0 \dots 11)_2 \cdot 2^{-126} = (1+2^{-22} + 2^{-23}) \cdot 2^{-126}$

$$Exp = -125 \Rightarrow Char = 2$$

- 0000 0001 0000 0000 ... 0000 =  $(1,0 \dots 00)_2 \cdot 2^{-125}$
- 0000 0001 0000 0000 ... 0001 =  $(1,0 \dots 01)_2 \cdot 2^{-125} = (1 + 2^{-23}) \cdot 2^{-125}$

$Exp = 0 \Rightarrow Char = 127$

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

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

$Exp = 127 \Rightarrow Char = 254$

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

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

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

$= (1 + 2^{-1} + 2^{-2} + \dots + 2^{-22} + 2^{-23}) \cdot 2^{127}$

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

### Zusammenfassung der 32-Bit Format

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

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

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

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

**NaN (Char = 255)**

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

### Aufgabe 2

Gegeben sei eine Gleitkomma-Zahl im 32-Bit-IEEE-Format.  $Z = 01000001111000000000000000000000$

Zahl in Dezimal-Darstellung?

$0100\ 0001\ 1110\ 0000\ 0000\ 0000\ 0000\ 0000$

$VZ = 0$

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

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

$Z = (-1)^0 \cdot 2^4 \cdot (1,1100\ 0000 \dots 0000)_2$

$= + (1 + 0,5 + 0,25)_{10} \cdot 2^4$

$= + 1,75 \cdot 2^4 = 1,75 \cdot 16 = 28$

### Multiplikation von Gleitkommazahlen

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

$3FE0\ 0000$  und

$3E40\ 0000$

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

Darstellung von  $\pm 0$

$0000\ 0000\ 0000\ 0000 \dots 0000 = +0$

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

Darstellung denormalisierter Zahlen

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

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

$0000\ 0000\ 0111\ 1111 \dots 1111 = (0,1 \dots 11)_2 \cdot 2^{-126}$

$= (2^{-1} + 2^{-2} + \dots + 2^{-23}) \cdot 2^{-126}$

$= (1 - 2^{-24}) \cdot 2^{-126} = \text{Größte denormalisierte Zahl}$

### Aufgabe 1

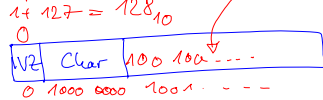
Die Zahl  $\pi$  mit 7 Dezimalstellen hinter dem Komma im 32-Bit-IEEE-Format.

$\pi \approx 3,1415926_{10} = 11,00100100000111111011010_2$

$= 1,100100100001111110101 \cdot 2^1$

$= 1,17 \cdot 2^1$

$Char = Exp + 127 = 1 + 127 = 128_{10}$



### Aufgabe

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

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

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

Gesucht:  $1/(x-y)$

### Aufgaben

**Hamming-Code**

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

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

Darstellung von  $\pm \infty$

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

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

Darstellung von NaN:

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

$M \neq 0$

### Aufgabe 1

Die Zahl  $\pi$  mit 7 Dezimalstellen hinter dem Komma im 32-Bit-IEEE-Format.

$\pi \approx 3,1415926_{10} = 11,00100100000111111011010_2$

$= (-1)^0 \cdot 2^{(128-127)} \cdot 1,10010010000111111011010_2$

$= (-1)^0 \cdot 2^1 \cdot 1,10010010000111111011010_2$

$VZ = 0$

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

$\pi = 0100\ 0000\ 0100\ 1001\ 0000\ 1111\ 1101\ 1010$

$= 4\ 0\ 4\ 9\ 0\ F\ D\ A_{16}$

### Lösung

Implementierung im Rechner:

1. Vorschlag:  $z := 1/(x-y)$

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

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

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

$x = (1 + 2^{-23}) \cdot 2^{-126}$

$y = 1 \cdot 2^{-126}$

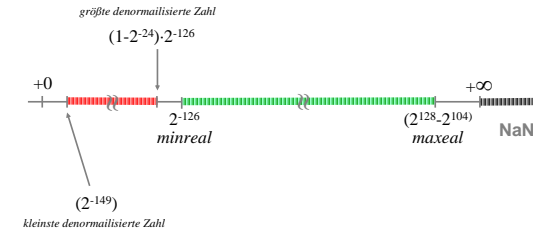
### Ein-Bit-Fehlerkorrektur

Notwendiger Aufwand für die sog. **Hammingcodes**:

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

$k$ : Zahl der zusätzlichen Korrekturbits

$m$ : Zahl der Datenbits



### Aufgabe 2

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

$Z = 01000001111000000000000000000000$

Zahl in Dezimal-Darstellung?

$0100\ 0001\ 1110\ 0000\ 0000\ 0000\ 0000\ 0000$

$Char = 128 + 2 + 1 = 131$

$Exp = Char - 127 = 131 - 127 = 4$

$Z = + (1,110 \dots 0) \cdot 2^4 = (1 + \frac{1}{2} + \frac{1}{4}) \cdot 2^4$

$= (1 + \frac{3}{4}) \cdot 2^4 = 28_{10}$

### Addition von Gleitkommazahlen

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

$0100\ 0011\ 1110\ 0000 \dots 0000$  und

$0100\ 0010\ 0101\ 0000 \dots 0000$

$+ m_x \cdot 2^{ex} \quad + m_y \cdot 2^{ey}$

Mindestaufwand: Ein-Bit Fehlerkorrekturcodes

Zahl der Datenbits	Zahl der Prüfbits	Bits insgesamt:
m	k	$m + k$
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	$m + 6$
64	...	...

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 <sub>12</sub>	k <sub>3</sub>	m <sub>11</sub>	m <sub>10</sub>	m <sub>9</sub>	m <sub>8</sub>	m <sub>7</sub>	m <sub>6</sub>	m <sub>5</sub>	k <sub>4</sub>	m <sub>4</sub>	m <sub>3</sub>	m <sub>2</sub>	k <sub>5</sub>	m <sub>1</sub>	k <sub>2</sub>	k <sub>1</sub>	Stellen

Datenwort: 1 0 1 0 1 0 1 0  $w = 8$

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<sub>8</sub> m<sub>7</sub> m<sub>6</sub> m<sub>5</sub> k<sub>4</sub> m<sub>4</sub> m<sub>3</sub> m<sub>2</sub> k<sub>3</sub> m<sub>1</sub> k<sub>2</sub> k<sub>1</sub>

Bestimmung der Prüfbits:

$$k_1 = m_1 \oplus m_2 \oplus m_7 \oplus m_8 = 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

$$k_2 = m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

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

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

→ Codewort: 1 0 1 0 0 1 0 1 1 0 0 0

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

## Hammingcode: Beispiel 2-Bitfehler

Fall 1: Kein Fehler

$$m_3 k_4 m_4 m_3 m_2 k_3 m_1 k_2 k_1 \text{ PB}$$

$$0 0 1 1 0 0 1 1 0 0 \quad k_4 k_3 k_2 k_1 = 0000$$

Fall 2: 1-Bit-Fehler

$$0 0 1 0 0 0 1 1 0 1 \quad k_4 k_3 k_2 k_1 = 0110$$

Fehler an 6. Position

Fall 3: 2-Bitfehler

$$0 0 1 0 0 0 0 1 0 0 \quad k_4 k_3 k_2 k_1 = 0101$$

2-Bit-Fehler

## 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.

## Hammingcode-Applet

TI-Homepage

ti.itec.uka.de

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 <sub>12</sub>	k <sub>3</sub>	m <sub>11</sub>	m <sub>10</sub>	m <sub>9</sub>	m <sub>8</sub>	m <sub>7</sub>	m <sub>6</sub>	m <sub>5</sub>	k <sub>4</sub>	m <sub>4</sub>	m <sub>3</sub>	m <sub>2</sub>	k <sub>5</sub>	m <sub>1</sub>	k <sub>2</sub>	k <sub>1</sub>	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 0 1 1 0  
 m<sub>8</sub> m<sub>7</sub> m<sub>6</sub> m<sub>5</sub> k<sub>4</sub> m<sub>4</sub> m<sub>3</sub> m<sub>2</sub> k<sub>3</sub> m<sub>1</sub> k<sub>2</sub> k<sub>1</sub>

Bestimmung der Prüfbits:

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

$$k_2 = k_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

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

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

$k_4 k_3 k_2 k_1 = 0000$  → Codewort ist fehlerfrei  
 → 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 <sub>12</sub>	k <sub>3</sub>	m <sub>11</sub>	m <sub>10</sub>	m <sub>9</sub>	m <sub>8</sub>	m <sub>7</sub>	m <sub>6</sub>	m <sub>5</sub>	k <sub>4</sub>	m <sub>4</sub>	m <sub>3</sub>	m <sub>2</sub>	k <sub>5</sub>	m <sub>1</sub>	k <sub>2</sub>	k <sub>1</sub>	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 0 0 0 0  
 m<sub>8</sub> m<sub>7</sub> m<sub>6</sub> m<sub>5</sub> k<sub>4</sub> m<sub>4</sub> m<sub>3</sub> m<sub>2</sub> k<sub>3</sub> m<sub>1</sub> k<sub>2</sub> k<sub>1</sub>

Bestimmung der Prüfbits:

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

$$k_2 = k_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 0$$

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

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

$k_4 k_3 k_2 k_1 = 1100 = 12_{10}$  → Fehler in der 12. Position  
 → Datenwort: 0 0 0 0 0 0 0 0