

# 02 – Zahlendarstellung

Grundzüge digitaler Systeme

Vortrag von: Stefan Neumann

Bevor es losgeht...

# Wie löst man Probleme mit Hilfe von Computern?

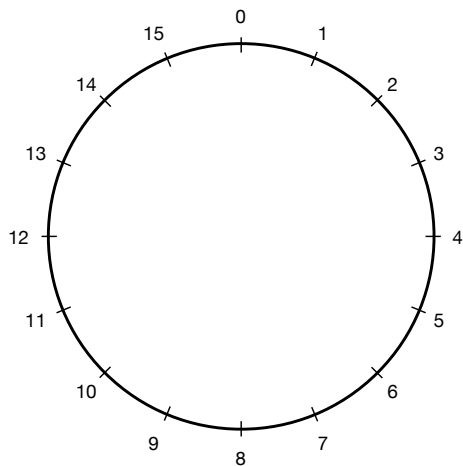
---

- Eine der wesentlichen Fragen des Informatikstudiums ist:
  - Wie löst man Probleme mit Computern?
  - Wie stellt ein Computer Zahlen dar, wie findet man kürzeste Wege von A nach B, ...
- Es gibt kein Patentrezept
  - Oft es ist sehr problemabhängig, wie man zu einer Lösung kommt
- Aber es gibt Herangehensweisen, die man lernen kann und die oft funktionieren
  - **Variante 1:** Zuerst löst man das Problem “ohne” Computer und überträgt diesen Ansatz dann
  - **Variante 2:** Man löst zuerst ein “leichteres” Problem und führt das “schwierige” Problem später auf das leichte zurück
    - **Alternativ:** Manchmal ist es ebenfalls hilfreich erst einen “schlechten” Algorithmus zu entwerfen und diesen dann entsprechend zu verbessern
- Wir werden beide Techniken in den nächsten Tagen und Wochen mehrfach anwenden
- Diese Problemlöse-Fähigkeiten zu erlernen ist ein wesentlicher Teil des Studiums, nicht bloß Algorithmen auswendig zu lernen, sondern zu verstehen, wo die Ideen herkommen

...und noch eine weitere Sache...

# Rechnen mit Rest: Modulo

- Wir führen eine neue Rechenoperation ein: **modulo**
- Bezeichnet den Rest bei Division
- Für natürliche Zahlen  $a, b \in \mathbb{N}$  schreiben wir  $a \bmod b$  für den Rest, wenn man  $a$  durch  $b$  teilt
  - $\bmod$  ist die Abkürzung für “modulo”
- Beispiele für  $b = 16$ :
  - $37 \bmod 16 = 5$ 
    - Da  $\lfloor 37/16 \rfloor = 2$ , Rest 5
  - $3 \bmod 16 = 3$ 
    - Da  $\lfloor 3/16 \rfloor = 0$ , Rest 3
  - $80 \bmod 16 = 0$ 
    - Da  $\lfloor 80/16 \rfloor = 5$ , Rest 0
- $\lfloor \cdot \rfloor$  bezeichnet Abrunden
- **Intuitive Vorstellung:** Man hat einen Zahlenkreis von 0 bis  $b - 1$ . Zahlen, die größer als  $b - 1$  sind, “gehen im Kreis”.



# Rechnen mit Rest: Modulo

- Wie funktioniert modulo-Rechnen mit negativen Zahlen  $a$ ?

- Also  $a < 0$  und  $b \in \mathbb{N}$ ,  
wollen  $a \bmod b$  ausrechnen

- So lange Vielfache von  $b$  addieren  
bis man Wert zwischen 0 und  $b - 1$  erhält

- Intuition:**

“Im Kreis gehen bis man eine positive Zahl hat”

- Beispiele für  $b = 16$ :

- $37 \bmod 16 = 5$

- Da  $\lfloor 37/16 \rfloor = 2$ , Rest 5

- Ohne Runden:  $37 = 2 \cdot 16 + 5$

- $-1 \bmod 16 = 15$

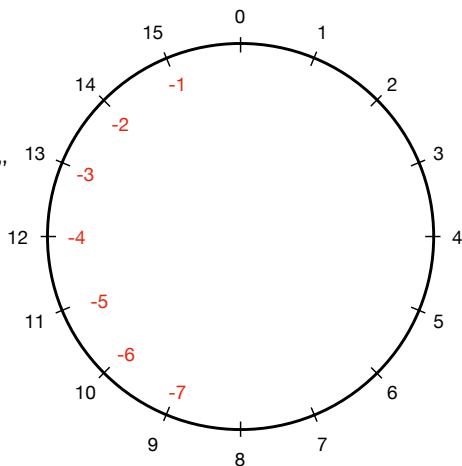
- Da  $-1 = (-1) \cdot 16 + 15$

- $-31 \bmod 16 = 1$

- Da  $-31 = (-2) \cdot 16 + 1$

- $-80 \bmod 16 = 0$

- Da  $-80 = (-5) \cdot 16 + 0$



# Zahlendarstellung — Übersicht

---

- 1 Bits und Bytes
- 2 Zahlenumwandlungen — Ganzzahlen binär und dezimal
- 3 Zahlen mit Nachkommateil
- 4 Zahlensysteme allgemein
- 5 Rechnen im Binärsystem
- 6 Darstellung negativer (Ganz-)Zahlen
  - Vorzeichenbit
  - Einerkomplementdarstellung
  - Zweierkomplementdarstellung
  - Exzessdarstellung

# Komponenten eines Rechners

---

Fünf klassische Komponenten:

- Rechenwerk (Arithmetic Logic Unit, ALU)
  - Steuer- oder Leitwerk (Control Unit, CU)
  - Hauptspeicher (Memory)
  - Eingabe (Input)
  - Ausgabe (Output)
- Historischer Ursprung: John von Neumann (Mitte der 1940er)
- Programme und Daten werden gemeinsam im Speicher abgelegt

} Prozessor





- Binärzeichen oder **Bits** (Symbole des binären Alphabets)
  - Ein binäres Alphabet hat genau zwei Symbole
  - Typischerweise sind die zwei Symbole **0** und **1**
  - Teilweise wird auch L (low) und H (high) genutzt
- Bit von **B**inary digit = Binärziffer
- Computer speichern Bits zum Beispiel mit sogenannten Flipflops (wird später bei Digitalschaltungen behandelt)
  - Für die nächsten Vorlesungen nehmen wir einfach an, dass wir Bits speichern können

# Byte, Megabyte, ...

- Byte: Entspricht 8 Bits
- Beispiel für Bytes
  - 10001010 10110110
- Maßeinheit für z. B. Hauptspeichergrößen oder Festplattengrößen
- Präfixe zur kürzeren Schreibweise
  - Präfixe für eine große Anzahl von Bytes (Kilo, Mega, Giga, Tera, Peta, Exa, Zetta, Yotta, ...)
  - **Achtung:** Unterschiedliche Verwendung der Präfixe in der Literatur und Praxis!

## SI-Präfixe

Kürzel	Name	Potenz	Wert
kB	Kilobyte	$10^3$	1 000 Bytes
MB	Megabyte	$10^6$	1 000 000 Bytes
GB	Gigabyte	$10^9$	1 000 000 000 Bytes
TB	Terabyte	$10^{12}$	1 000 000 000 000 Bytes

## Binärpräfixe

Kürzel	Name	Potenz	Wert
KiB	Kibibyte	$2^{10}$	1 024 Bytes
MiB	Mebibyte	$2^{20}$	1 048 576 Bytes
GiB	Gibibyte	$2^{30}$	1 073 741 824 Bytes
TiB	Tebibyte	$2^{40}$	1 099 511 627 776 Bytes

- Natürlich wollen wir nicht bloß Nullen und Einsen darstellen, sondern mit Computern auch Texte oder Bilder speichern
- Ein **Code** ist eine Zuordnungsvorschrift zwischen zwei Alphabeten
- Beispiel (Bitmuster zu Kleinbuchstaben)
  - Alphabet 1: Binäre Zeichenketten mit 5 Bits
  - Alphabet 2: die Kleinbuchstaben  $\{a, b, c, \dots, z\}$
  - Zuordnung: 00000  $\rightarrow$  a, 00001  $\rightarrow$  b, 00010  $\rightarrow$  c, ...
- **Binärcodierung**
  - Die Codierung irgendeines Alphabets durch Folgen von Binärzeichen
- Universeller Einsatz des binären Alphabets
  - **Alle endlichen Alphabete** lassen sich durch Folgen von Binärzeichen ausdrücken
- Im vierten Teil der Vorlesung sehen wir verschiedenste Codes (bspw. ASCII und UTF Codes, Barcodes, etc.)

# Binäre Signale

Angenommen wir wollen ein Alphabet mit  $x$  Zeichen darstellen. Wie viele Bits brauchen wir?

- Wie viele verschiedene Kombinationen können wir mit  $n$  Bits darstellen?
- Beispiel: für  $n = 2$  haben wir  $4 = 2^2$  Kombinationen
  - 00, 01, 10, 11
- Beispiel: für  $n = 3$  haben wir  $8 = 2^3$  Kombinationen
  - 000, 001, 010, 011, 100, 101, 110, 111
- Mit  $n$  Binärziffern können  $2^n$  Kombinationen gebildet werden
  - Jedes Bit hat zwei mögliche Zustände, daher mit  $n$  Bits  $2^n = \underbrace{2 \cdot 2 \cdots 2 \cdot 2}_{n \text{ mal}}$  Kombinationen
  - ⇒ Jedes zusätzliche Bit verdoppelt unsere möglichen Kombinationen
- Um  $x$  Kombinationen darzustellen, benötigt man  $\lceil \log_2(x) \rceil$  Bits
  - Um die Kleinbuchstaben  $\{a, b, \dots, z\}$  darzustellen brauchen wir  $\lceil \log_2(26) \rceil = \lceil 4.701 \rceil = 5$  Bits
  - Um die Buchstaben  $\{a, b, \dots, z, A, B, \dots, Z\}$  darzustellen brauchen wir  $\lceil \log_2(2 \cdot 26) \rceil = \lceil 5.701 \rceil = 6$  Bits

# Zahlendarstellung — Übersicht

---

- 1 Bits und Bytes
- 2 **Zahenumwandlungen — Ganzzahlen binär und dezimal**
- 3 Zahlen mit Nachkommateil
- 4 Zahlensysteme allgemein
- 5 Rechnen im Binärsystem
- 6 Darstellung negativer (Ganz-)Zahlen
  - Vorzeichenbit
  - Einerkomplementdarstellung
  - Zweierkomplementdarstellung
  - Exzessdarstellung

# Dezimalsystem

Bevor wir Zahlen von dezimal zu binär umrechnen, hier eine kurze Wiederholung zum Dezimalsystem, die uns gleich helfen wird:

- Wie stellen wir Zahlen mit dem Dezimalsystem dar?
- Beispiel:

$$\begin{aligned}1815 &= 1 \cdot 1000 + 8 \cdot 100 + 1 \cdot 10 + 5 \\ &= 1 \cdot 10^3 + 8 \cdot 10^2 + 1 \cdot 10^1 + 5 \cdot 10^0.\end{aligned}$$

- Ziffern =  $\{0,1,2,3,4,5,6,7,8,9\}$
- Basis = 10
- Die Basis entspricht der Anzahl der Ziffern
- Jede Zahl ist eine Summe mit den Summanden Ziffer  $\cdot$  (Potenz der Basis):
  - Jede  $n$ -stellige Zahl kann dargestellt werden mit Ziffern  $(a_n a_{n-1} \cdots a_3 a_2 a_1 a_0)$  sodass

$$\sum_{i=0}^n a_i 10^i = \cdots + a_3 10^3 + a_2 10^2 + a_1 10^1 + a_0 10^0$$

- Für die Zahl 1815:  $a_3 = 1$ ,  $a_2 = 8$ ,  $a_1 = 1$ ,  $a_0 = 5$

# Beispiel im Binärsystem

Wie würden wir die Zahl 1815 im Binärsystem darstellen?

- Ziffern =  $\{0,1\}$
- Basis = 2
- Beispiel:

$$1815 = 1 \cdot 2^{10} + 1 \cdot 2^9 + 1 \cdot 2^8 + 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 \\ + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

- Die Binärdarstellung von 1815 lautet **111 0001 0111**
  - Die Darstellung erhält man durch das Ablesen der Koeffizienten
- Wir haben also erneut eine Ziffernfolge  $(a_n a_{n-1} \cdots a_3 a_2 a_1 a_0)$  gefunden, sodass

$$\sum_{i=0}^n a_i 2^i = \cdots + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0$$

⇒ Analog zu Dezimalsystem, aber diesmal sind alle Ziffern in  $\{0,1\}$  und die Basis ist 2

# Zahlenumwandlungen – binär und dezimal

Konversion von ganzen Zahlen

---

- **Jetzt:** Wie wandelt man Zahlen zwischen binär und dezimal um?
- Damit wir nicht durcheinander kommen, schreiben wir  $(\cdot)_2$  für binäre Zahlen und  $(\cdot)_{10}$  für Zahlen im Dezimalsystem
  - $(111\ 0001\ 0111)_2$  ist eine Binärzahl
  - $(1815)_{10}$  ist eine Dezimalzahl
- Die **Quelldarstellung** ist die Darstellung im gegebenen System
- Die **Zieldarstellung** ist die Darstellung im neuen System



# Zahlenumwandlungen – binär und dezimal

Konversion von binär zu dezimal — Stellenwerte

- Wir wollen  $(1011\ 0111)_2$  ins Dezimalsystem umwandeln

- Variante 1: Stellenwerte

- Hier setzen wir einfach in unsere Formel  $\sum_{i=0}^n a_i 2^i$  ein

Binärdarstellung	(1	0	1	1	0	1	1	1	) <sub>2</sub>
Stellenwerte	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	
Umrechnung	$1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 =$								$(183)_{10}$

# Zahlenumwandlungen – binär und dezimal

Konversion von binär zu dezimal — Hornerschema

## ■ Variante 2: Hornerschema

- In der Berechnung oft effizienter, da wir die Stellenwerte nicht einzeln ausrechnen müssen
- Beruht auf folgender Beobachtung:

$$\begin{aligned}(\cdots a_3 a_2 a_1 a_0)_2 &= \cdots + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0 \\ &= (((\cdots + a_3) \cdot 2 + a_2) \cdot 2 + a_1) \cdot 2 + a_0\end{aligned}$$

- Beispiel:

$$\begin{aligned}(1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1)_2 \\ ((((((1 \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1 = \\ ((((((2 \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1 = \\ ((((((5 \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1 = \\ ((((((11 \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1 = \\ ((((((22 \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1 = \\ ((((((45 \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1 = \\ 91 \cdot 2 + 1 = \\ = (183)_{10}\end{aligned}$$

# Zahlenumwandlungen – dezimal und binär

Konversion von dezimal zu binär

- Jetzt wollen wir Zahlen vom Dezimalsystem ins Binärsystem umrechnen
- Gegeben: Dezimalzahl  $(Z)_{10}$
- Gesucht:  $a_0, a_1, a_2, a_3, \dots$  mit  $a_i \in \{0, 1\}$  sodass

$$(Z)_{10} = (\dots a_3 a_2 a_1 a_0)_2 = \dots + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0$$

- **Beispiel:** Wir wollen  $(7)_{10}$  ins Binärsystem umwandeln
  - 7 ist eine ungerade Zahl, also “gerade Zahl + 1” ( $7 = 6 + 1$ )
  - Alle Zahlen, die wir mit den Bits  $a_1, a_2, a_3, \dots$  darstellen können sind Vielfache von 2
  - ⇒ Wir wissen also  $a_0 = 1$ , da wir die +1 nur mit  $2^0$  darstellen können
  - Mit den Bits  $a_1, a_2, \dots$  müssen wir jetzt also 6 darstellen
  - Wir wissen  $6 = 4 + 2$
  - Alle Zahlen, die wir mit den Bits  $a_2, a_3, \dots$  darstellen können sind Vielfache von 4
  - ⇒ Wir wissen also  $a_1 = 1$ , da wir die +2 nur mit  $2^1$  darstellen können
  - Mit den Bits  $a_2, \dots$  müssen wir jetzt also 4 darstellen
  - Da  $2^2 = 4$  wissen wir, dass  $a_2 = 1$
  - ⇒ Die Binärdarstellung von  $(7)_{10}$  ist  $(111)_2$

# Zahlenumwandlungen – dezimal und binär

## Konversion von dezimal zu binär

- Jetzt wollen wir Zahlen vom Dezimalsystem ins Binärsystem umrechnen
- Gegeben Dezimalzahl  $(Z)_{10}$ , gesucht  $a_0, a_1, a_2, a_3, \dots$  mit  $a_i \in \{0, 1\}$  sodass

$$\begin{aligned}(Z)_{10} &= (\dots a_3 a_2 a_1 a_0)_2 = \dots + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0 \\ &= (((\dots + a_3) \cdot 2 + a_2) \cdot 2 + a_1) \cdot 2 + a_0\end{aligned}$$

- Algorithmus zum Umrechnen von dezimal zu binär:<sup>1</sup>

- Setze  $Z_0 \leftarrow Z, i \leftarrow 0$
- Solange  $Z_i \neq 0$ :
  - $a_i \leftarrow Z_i \bmod 2$ , d. h.  $a_i$  ist der Rest wenn man  $Z_i$  durch 2 teilt
  - $Z_{i+1} \leftarrow \frac{Z_i - a_i}{2}$
  - $i \leftarrow i + 1$

- Beispiel:  $(6)_{10}$  ins Binärsystem umrechnen

- $a_0 = Z_0 \bmod 2 = 6 \bmod 2 = 0, Z_1 = \frac{6-0}{2} = 3$
- $a_1 = Z_1 \bmod 2 = 3 \bmod 2 = 1, Z_2 = \frac{3-1}{2} = 1$
- $a_2 = Z_2 \bmod 2 = 1 \bmod 2 = 1, Z_3 = \frac{1-1}{2} = 0 \implies (6)_{10} = (110)_2$

- Den Algorithmus kann man sich aus dem Hornerschema herleiten

<sup>1</sup>Im Algorithmus schreiben wir  $\leftarrow$  wenn Variablen einen neuen Wert erhalten.

# Zahlenumwandlungen – dezimal in binär

Konversion von ganzen Zahlen – Rechnen im Quellsystem

- Gegeben: Quelldarstellung  $(29)_{10}$

- Gesucht: Zieldarstellung  $(\cdot)_2$

- Berechnung:

$$a_0 = 29 \bmod 2 = 1 \quad \text{LSB (niederstwertige Stelle, least significant bit)}$$

$$a_1 = 14 \bmod 2 = 0$$

$$a_2 = 7 \bmod 2 = 1$$

$$a_3 = 3 \bmod 2 = 1$$

$$a_4 = 1 \bmod 2 = 1 \quad \text{MSB (höchstwertige Stelle, most significant bit)}$$

→ Ergebnis:  $(29)_{10} = (1\,1101)_2$

- **Wichtig:** Die Bits werden in umgekehrter Reihenfolge angegeben, in der wir sie berechnet haben (wir berechnen  $a_0$  zuerst, aber in der Binärdarstellung kommt es zuletzt, etc.)

# Zahlendarstellung — Übersicht

---

- 1 Bits und Bytes
- 2 Zahlenumwandlungen — Ganzzahlen binär und dezimal
- 3 Zahlen mit Nachkommateil
- 4 Zahlensysteme allgemein
- 5 Rechnen im Binärsystem
- 6 Darstellung negativer (Ganz-)Zahlen
  - Vorzeichenbit
  - Einerkomplementdarstellung
  - Zweierkomplementdarstellung
  - Exzessdarstellung

# Zahlen mit Nachkommateil im Dezimalsystem

Wir beginnen wieder mit einer kurzen Wiederholung zum Dezimalsystem:

- Wie stellen wir Zahlen mit Nachkommateil mit dem Dezimalsystem dar?
- Beispiel:

$$\begin{aligned}0.1815 &= 1 \cdot 0.1 + 8 \cdot 0.01 + 1 \cdot 0.001 + 5 \cdot 0.0001 \\ &= 1 \cdot 10^{-1} + 8 \cdot 10^{-2} + 1 \cdot 10^{-3} + 5 \cdot 10^{-4}.\end{aligned}$$

- Wieder hat die letzte Ziffer die kleinste Zehnerpotenz
- Wir verringern den Exponenten bei jeder Stelle um  $-1$
- Jeder Summand hat die Form **Ziffer  $\cdot$  (Potenz der Basis) $^{-1}$**
- Jede  $n$ -stellige Zahl wird dargestellt mit Ziffern  $(0.a_{-1}a_{-2}\cdots a_{-(n-1)}a_{-n})$  sodass

$$\sum_{i=1}^n a_{-i}10^{-i}$$

- Für die Zahl 0.1815:  $a_{-1} = 1$ ,  $a_{-2} = 8$ ,  $a_{-3} = 1$ ,  $a_{-4} = 5$

# Zahlen mit Nachkommateil im Binärsystem

- Wie stellt man Zahlen mit Nachkommateil im Binärsystem dar?
- Ganz analog
- Beispiel:

$$\begin{aligned}(0.1011)_2 &= 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{8} + 1 \cdot \frac{1}{16} \\ &= 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} \\ &= 0.6875.\end{aligned}$$

- Jede  $n$ -stellige Zahl wird dargestellt mit Ziffern  $(0.a_{-1}a_{-2}\cdots a_{-(n-1)}a_{-n})$  sodass

$$\sum_{i=1}^n a_{-i}2^{-i}$$

- Für die Zahl  $(0.1011)_2$ :  $a_{-1} = 1$ ,  $a_{-2} = 0$ ,  $a_{-3} = 1$ ,  $a_{-4} = 1$



# Zahlenumwandlungen – binär in dezimal

## Nachkommateil

- Gegeben: Quelldarstellung  $(0.1101)_2$
- Gesucht: Zieldarstellung  $(0.\quad)_{10}$
- Umwandeln der Ziffern  $(0)_2, (1)_2$  in Zieldarstellung  $(0)_{10}, (1)_{10}$
- Umwandeln der Quell-Basis:  $b = (10)_2 = (2)_{10}$
- Variante 1: Stellenwerte  
 $(0.1101)_2 = 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} = (0.8125)_{10}$
- Variante 2: Basierend auf Hornerschema

$a_{-4}$	$a_{-3}$	$a_{-2}$	$a_{-1}$
1	0	1	1

$$(((1/2 + 0)/2 + 1)/2 + 1)/2 =$$

$$((0.5/2 + 1)/2 + 1)/2 =$$

$$(1.25/2 + 1)/2 =$$

$$1.625/2 = (0.8125)_{10}$$

# Zahlenumwandlungen

Nachkommateil — Umwandlung allgemein

---

- Gegeben: Die Dezimalzahl  $u$  mit Quelldarstellung  $(0.u_{-1}u_{-2}\dots u_{-m})_{10}$
- Gesucht: Zieldarstellung der Binärzahl  $U = (0.U_{-1}U_{-2}\dots U_{-n})_2$

## ■ Vorgehensweise

- Umwandeln der Ziffern  $u_{-i}$  in Zieldarstellung
- Umwandeln von Stellenwerten  $10^{-i}$  in Zieldarstellung
- Berechnung von  $u_{-1}10^{-1} + u_{-2}10^{-2} + u_{-m}10^{-m}$

oder

- Umwandlung basierend auf Hornerschema
- Mehr Details auf nächstem Slide

# Zahlenumwandlungen

Nachkommateil — Umwandlung basierend auf Hornerschema

- Die Zahl  $u$  habe die Quelldarstellung  $u = (0.u_{-1}u_{-2}\dots u_{-m})_{10}$
- Gesucht: Zieldarstellung der Zahl  $U = (0.U_{-1}U_{-2}\dots U_{-n})_2$  mit höchstens  $n$  Nachkommastellen
- Umwandlung basierend auf Hornerschema
  - $x_{-1} \leftarrow u, i \leftarrow 1$
  - Solange  $x_{-i} \neq 0$  und Anzahl der Nachkommastellen nicht überschritten:
    - $U_{-i} = \lfloor x_{-i} \cdot 2 \rfloor$
    - $x_{-(i+1)} \leftarrow x_{-i} \cdot 2 - U_{-i}$
    - $i \leftarrow i + 1$
  - Hier bezeichnet  $\lfloor \cdot \rfloor$  die Abrunden-Operation
    - Beispiele:  $\lfloor 8.5 \rfloor = 8, \lfloor 8 \rfloor = 8, \lfloor 8.9999 \rfloor = 8$
  - Da wir in jedem Schritt  $x_{-(i+1)}$  setzen indem wir  $U_{-i} = \lfloor x_{-i} \cdot 2 \rfloor$  von  $x_{-i} \cdot 2$  abziehen, ist das Vorgehen analog dazu zu sagen, dass  $U_{-i}$  der Vorkommateil von  $x_{-i} \cdot 2$  ist und  $x_{-(i+1)}$  der Nachkommateil ist

# Zahlenumwandlungen – dezimal in binär

Nachkommateil

- Quelldarstellung:  $(0.8125)_{10}$
- Gesucht: Zieldarstellung  $(0. \quad )_2$
- Umwandeln der Ziel-Basis  $B = (10)_2 = (2)_{10}$
- Berechnung:

$$U_{-1} = \lfloor 0.8125 \cdot 2 \rfloor = \lfloor 1.625 \rfloor = 1 \quad \text{MSB}$$

$$U_{-2} = \lfloor 0.625 \cdot 2 \rfloor = \lfloor 1.25 \rfloor = 1$$

$$U_{-3} = \lfloor 0.25 \cdot 2 \rfloor = \lfloor 0.5 \rfloor = 0$$

$$U_{-4} = \lfloor 0.5 \cdot 2 \rfloor = \lfloor 1 \rfloor = 1$$

$$U_{-5} = \lfloor 0 \cdot 2 \rfloor = \lfloor 0 \rfloor \rightarrow \text{Abbruch} \quad \text{LSB}$$

- Ergebnis:  $(0.8125)_{10} = (0.1101)_2$

# Zahlenumwandlungen – dezimal in binär

## Nachkommateil

- Quelldarstellung:  $(0.815)_{10}$
- Gesucht: Zieldarstellung  $(0. \quad )_2$  bis zur fünften Nachkommastelle
- Umwandeln der Ziel-Basis  $B = (10)_2 = (2)_{10}$
- Berechnung:

$$U_{-1} = [0.815 \cdot 2] = [1.\textcolor{red}{63}] = 1$$

$$U_{-2} = [\textcolor{red}{0.63} \cdot 2] = [1.\textcolor{green}{26}] = 1$$

$$U_{-3} = [\textcolor{green}{0.26} \cdot 2] = [0.\textcolor{blue}{52}] = 0$$

$$U_{-4} = [\textcolor{blue}{0.52} \cdot 2] = [1.\textcolor{violet}{04}] = 1$$

$$U_{-5} = [\textcolor{violet}{0.04} \cdot 2] = [0.08] = 0$$

MSB  
↓  
LSB

- Ergebnis:  $(0.815)_{10} = (0.11010)_2$

# Zahlendarstellung — Übersicht

---

- 1 Bits und Bytes
- 2 Zahlenumwandlungen — Ganzzahlen binär und dezimal
- 3 Zahlen mit Nachkommateil
- 4 Zahlensysteme allgemein**
- 5 Rechnen im Binärsystem
- 6 Darstellung negativer (Ganz-)Zahlen
  - Vorzeichenbit
  - Einerkomplementdarstellung
  - Zweierkomplementdarstellung
  - Exzessdarstellung

# Zahlensysteme – allgemeine Stellenwertsysteme

## ■ Basis $b$

- Basis ist Anzahl der verwendeten Ziffern,  $b \geq 2, b \in \mathbb{N}$

## ■ Zahl $(\dots a_3 a_2 a_1 a_0)_b$

$(1923)_{10}$

- Ziffern: Zeichen mit dezimalem Wert  $0, 1, \dots, b-1$

Ziffer	$a_3$	$a_2$	$a_1$	$a_0$
Wert der Stelle	$b^3$	$b^2$	$b^1$	$b^0$

## ■ Wert der Zahl:

$$a_3 \cdot b^3 + a_2 \cdot b^2 + a_1 \cdot b^1 + a_0 \cdot b^0$$

$$1 \cdot 10^3 + 9 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 = 1923$$

## ■ Nachkommastellen: $(\dots a_3 a_2 a_1 a_0 . a_{-1} a_{-2} \dots)_b$

$$x^{-y} = \frac{1}{x^y}$$

$$\dots + a_3 b^3 + a_2 b^2 + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + \dots$$

# Zahlensysteme – Beispiele

- Bsp.:  $(123.12)_{10}$  ist ident mit der gewohnten Notation 123.12 für das geläufige Zehnersystem:

$$\begin{aligned}(a_2 a_1 a_0 . a_{-1} a_{-2})_b &= a_2 \cdot b^2 + a_1 \cdot b^1 + a_0 \cdot b^0 + a_{-1} \cdot b^{-1} + a_{-2} \cdot b^{-2} \\(1\ 2\ 3\ .\ 1\ 2)_{10} &= 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 1 \cdot 10^{-1} + 2 \cdot 10^{-2} \\&= 100 + 20 + 3 + 0.1 + 0.02 \\&= (123.12)_{10}\end{aligned}$$

- Bsp. Basis 6:

$$\begin{aligned}(5\ 2\ 0\ .\ 3)_6 &= 5 \cdot 6^2 + 2 \cdot 6^1 + 0 \cdot 6^0 + 3 \cdot 6^{-1} \\&= 180 + 12 + 0 + 0.5 \\&= (192.5)_{10}\end{aligned}$$



# Zahlenumwandlungen

- Von einem Quellsystem in ein Zielsystem

Quellsystem	Zielsystem
Zahl $u$	Zahl $U$
Basis $b$	Basis $B$
$u = (u_m \dots u_1 u_0 . u_{-1} u_{-2} \dots u_{-s})_b$	$U = (U_n \dots U_1 U_0 . U_{-1} U_{-2} \dots U_{-t})_B$
$0 \leq u_i < b$	$0 \leq U_i < B$

- Zahlen mit Nachkommastellen (reelle Zahlen) aufteilen in
  - Vorkommateil
  - Nachkommateil
- Getrennt umrechnen

# Zahlenumwandlungen – allgemein

Konversion Vorkommateil allgemein

- Gegeben: Zahl  $(Z)_b$  mit Quelldarstellung  $(u_m \dots u_1 u_0)_b$  mit Basis  $b$
- Gesucht: Zieldarstellung der Zahl  $U = (U_n \dots U_1 U_0)_B$  mit Basis  $B$
  
- Variante 1: Stellenwerte
  - Umwandeln der Ziffern  $u_i$  in Zieldarstellung
  - Umwandeln von  $b^i$  in Zieldarstellung
  - Berechnung von  $u_m b^m + \dots + u_1 b + u_0$
  
- Variante 2: Basierend auf Hornerschema
  - Setze  $Z_0 \leftarrow Z, i \leftarrow 0$
  - Solange  $Z_i \neq 0$ :
    - $a_i \leftarrow Z_i \bmod B$ , d. h.  $a_i$  ist der Rest wenn man  $Z_i$  durch  $B$  teilt
    - $Z_{i+1} \leftarrow \frac{Z_i - a_i}{B}$
    - $i \leftarrow i + 1$
- Bis auf die neue Basis ist der Algorithmus identisch mit dem alten

# Zahlenumwandlungen – allgemein

Konversion Nachkommateil allgemein

- Gegeben: Zahl  $u$  mit Quelldarstellung  $(0.u_{-1}u_{-2}\dots u_{-m})_b$  mit Basis  $b$
- Gesucht: Zieldarstellung der Zahl  $U = (0.U_{-1}U_{-2}\dots U_{-n})_B$  mit Basis  $B$
  
- Variante 1: Stellenwerte
  - Umwandeln der Ziffern  $u_{-i}$  in Zieldarstellung
  - Umwandeln von Stellenwerten  $b^{-i}$  in Zieldarstellung
  - Berechnung von  $u_{-1}b^{-1} + u_{-2}b^{-2} + u_{-m}b^{-m}$
- Variante 2: Basierend auf Hornerschema
  - $x_{-1} \leftarrow u, i \leftarrow 1$
  - Solange  $u_{-i} \neq 0$  und Anzahl der Nachkommastellen nicht überschritten:
    - $U_{-i} = \lfloor x_{-i} \cdot B \rfloor$
    - $x_{-(i+1)} \leftarrow x_{-i} \cdot B - U_{-i}$
    - $i \leftarrow i + 1$

# Zahlensysteme

## Spezialfall: Hexadezimalsystem

- Hexadezimals Zahlensystem: 16 Ziffern, aber im Zehnersystem nur Ziffern 0...9
- Daher erweitert man die Dezimalziffern zu 0, 1, 2, ..., 8, 9, A, B, C, D, E, F
- Beispiele von hexadezimalen Zahlen mit ihrem dezimalen Äquivalent

hexadezimal (...) <sub>16</sub>	dezimal (...) <sub>10</sub>
10	16
1A	26
1F	31
FF	255
7FFF	32767

# Zahlensysteme

binär – hexadezimal – dezimal

binär (...) <sub>2</sub>	hexadezimal (...) <sub>16</sub>	dezimal (...) <sub>10</sub>
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9

binär (...) <sub>2</sub>	hexadezimal (...) <sub>16</sub>	dezimal (...) <sub>10</sub>
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15
10000	10	16
10001	11	17
10010	12	18

# Zahlenumwandlungen – hexadezimal in dezimal

Konversion von ganzen Zahlen – Rechnen im Zielsystem

- Gegeben: Quelldarstellung  $(1C7)_{16}$
- Gesucht: Zieldarstellung  $(\cdot)_{10}$
- Umwandeln der Ziffern  $(1)_{16}, (C)_{16}, (7)_{16}$  in  $(1)_{10}, (12)_{10}, (7)_{10}$
- Umwandeln der Quell-Basis in Zieldarstellung:  $b = (10)_{16} = (16)_{10}$
- Variante 1: Stellenwerte

$$\begin{array}{ccc} (1 & C & 7)_{16} \\ 1 \cdot 16^2 + 12 \cdot 16^1 + 7 \cdot 16^0 = (455)_{10} \end{array}$$

- Variante 2: Hornerschema

$$\begin{array}{ccc} (1 & C & 7)_{16} \\ (1 \cdot 16 + 12) \cdot 16 + 7 = \\ 28 \cdot 16 + 7 = (455)_{10} \end{array}$$

# Zahlenumwandlungen – dezimal in hexadezimal

---

■ Gegeben: Quelldarstellung  $(29)_{10}$

■ Gesucht: Zieldarstellung  $(\cdot)_{16}$

■ Umwandeln der Ziel-Basis in Quelldarstellung:  $B = (10)_{16} = (16)_{10}$

■ Berechnung:

$$U_0 = 29 \bmod 16 = 13$$

$$U_1 = 1 \bmod 16 = 1 \quad \text{höchstwertige Stelle}$$

■  $(13)_{10} = (D)_{16}$

■  $(1)_{10} = (1)_{16}$

■ Ergebnis:  $(29)_{10} = (1D)_{16}$

# Zahlenumwandlungen – binär in hexadezimal

$$u = (0010 \ 1001 \ 1111 \ . \ 1100 \ 0110)_2$$

$$U = ( \ 2 \quad 9 \quad F \quad . \quad C \quad 6 \ )_{16}$$

$$u = ( \ 10 \ 1010 \ 1110 \ . \ 1111 \ 0001 \ 1 \ )_2$$

$$u = (0010 \ 1010 \ 1110 \ . \ 1111 \ 0001 \ 1000)_2$$

$$U = ( \ 2 \quad A \quad E \quad . \quad F \quad 1 \quad 8 \ )_{16}$$

bin	hex
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F



# Zahlenumwandlungen – binär in hexadezimal

- Aus den Bits Viererblöcke vor und nach dem Komma bilden (jeweils vom Komma weg)
- Viererblöcke einzeln ins hexadezimale System umwandeln
- Gegebenenfalls sind führende Nullen zu ergänzen oder Nullen am Ende der Zahl anzuhängen
- Die Konversion in umgekehrter Richtung ist genauso leicht zu realisieren
- Wegen der leichten Konversion und der kompakten Darstellung ist das hexadezimale Zahlensystem in der Informatik weit verbreitet
  - Beispielsweise in Hexdumps<sup>2</sup>

```
00000000 48 69 65 72 20 69 73 74 20 65 69 6e 20 42 65 69 |Hier ist ein Bei|
00000010 73 70 69 65 6c 74 65 78 74 2e 20 44 65 72 20 48 |spielttext. Der H|
00000020 65 78 64 75 6d 70 20 69 73 74 20 61 75 66 20 64 |exdump ist auf d|
00000030 65 72 20 6c 69 6e 6b 65 6e 20 53 65 69 74 65 20 |er linken Seite |
00000040 7a 75 20 73 65 68 65 6e 2e 0a 0a 4e 65 75 65 20 |zu sehen...Neue |
00000050 5a 65 69 6c 65 6e 20 6f 64 65 72 20 41 62 73 e4 |Zeilen oder Absä|
00000060 74 7a 65 20 73 69 6e 64 20 64 61 6e 6e 20 61 75 |tze sind dann au|
00000070 63 68 20 22 5a 65 69 63 68 65 6e 22 20 6d 69 74 |ch "Zeichen" mit|
00000080 20 65 69 6e 65 6d 20 62 65 73 74 69 6d 6d 74 65 |einem bestimmte|
00000090 6e 0a 43 6f 64 65 2e 28 30 61 29 2e 2e 2e 0a 0a |n.Code.(0a).....|
```

<sup>2</sup><https://de.wikipedia.org/wiki/Dump#Hexdump>

# Zahlendarstellung — Übersicht

---

- 1 Bits und Bytes
- 2 Zahlenumwandlungen — Ganzzahlen binär und dezimal
- 3 Zahlen mit Nachkommateil
- 4 Zahlensysteme allgemein
- 5 Rechnen im Binärsystem**
- 6 Darstellung negativer (Ganz-)Zahlen
  - Vorzeichenbit
  - Einerkomplementdarstellung
  - Zweierkomplementdarstellung
  - Exzessdarstellung

# Rechnen im Binärsystem

---

- Im binären Zahlensystem gibt es nur die Ziffern 0 und 1
- Entsprechend einfach sind die Additionsregeln

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$

- Es bleibt nur mehr ein Falls zu behandeln, nämlich:
  - $1 + 1 = ?$
- Problem ähnlich wie im dezimalen System, bei den Rechnungen wo es zu einem Übertrag kommt ( $5 + 5 = ?$  oder  $7 + 8 = ?$ )
- Daher:

$1 + 1 = 10$
--------------

weil Stellenwertsystem

# Addition im Binärsystem

---

- Beispiel im Dezimalsystem:

$$\begin{array}{r} 3 \quad 2 \quad 4 \quad 5 \\ + \quad \quad 11 \quad 9 \quad 2 \\ \hline 3 \quad 4 \quad 3 \quad 7 \end{array}$$

- Im Binärsystem sehr ähnlich:
  - Ziffern – von rechts nach links – Stelle für Stelle aufaddieren
  - Zwischenergebnis unten notieren, jedoch nur letzte Stelle
  - Ist das Zwischenergebnis mehrstellig, so entstehen Überträge, die beim Abarbeiten der jeweils nächsten Spalte berücksichtigt werden müssen

# Addition im Binärsystem

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 10$

$$\begin{array}{rcccccc} & 1 & 1 & 0 & 1 & 1 & 0 \\ + & 0 & 1 & 1 & 0 & 1 & 1 \\ \hline & & & & & & 1 \end{array} \Rightarrow \begin{array}{rcccccc} & 1 & 1 & 0 & 1 & 1 & 0 \\ + & 0 & 1 & 1 & 0 & 1 & 1 \\ \hline & & & & & 0 & 1 \end{array} \Rightarrow \begin{array}{rcccccc} & 1 & 1 & 0 & 1 & 1 & 0 \\ + & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline & & & & & 0 & 0 & 1 \end{array} \Rightarrow$$
$$\begin{array}{rcccccc} & 1 & 1 & 0 & 1 & 1 & 0 \\ + & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline & & & & & 0 & 0 & 0 & 1 \end{array} \Rightarrow \begin{array}{rcccccc} & 1 & 1 & 0 & 1 & 1 & 0 \\ + & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline & & & & & 1 & 0 & 0 & 0 & 1 \end{array} \Rightarrow \begin{array}{rcccccc} & 1 & 1 & 0 & 1 & 1 & 0 \\ + & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline & & & & & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}$$

- Im Ergebnis tritt ein Überlauf auf

# Subtraktion im Binärsystem

- Beispiel im Dezimalsystem:

	3	2	4	5	Minuend
—		1	9	2	Subtrahend
	3	0	5	3	

- Im Binärsystem funktioniert das sehr ähnlich:

Falls bei der Subtraktion zweier Ziffern die Ziffer des Minuenden kleiner ist als die des Subtrahenden:

- Von der höherwertigen Stelle eine Zahl im Wert der Basis ausborgen
- Minuend um diesen Wert erhöhen
- Zwischenergebnis anschreiben
- Höherwertige Stelle vom Subtrahend um eins erhöhen

# Subtraktion im Binärsystem

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \\ - \quad \quad 1\ 1\ 1 \\ \hline \end{array}$$



$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \\ - \quad \quad 1\ 1\ 1 \\ \hline \end{array}$$



$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \\ - \quad \quad 1\ 1\ 1 \\ \hline \end{array}$$



$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \\ - \quad \quad 1\ 1\ 1 \\ \hline \end{array}$$



$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \\ - \quad \quad 1\ 1\ 1 \\ \hline \end{array}$$

$$(1)_2 - (1)_2 = (0)_2$$

Wir „borgen“ uns eine Zahl von der nächsten Stelle und berechnen  $(10)_2 - (1)_2 = (1)_2$ ; Bei der nächsten Stelle müssen wir  $(1)_2$  zum Subtrahenden addieren

Wir „borgen“ uns eine Zahl der nächsten Stelle und erhalten daher  $(10)_2 - (10)_2 = (0)_2$

Für die letzte Stelle „borgen“ wir uns wieder eine Zahl von der nächsten Stelle und berechnen  $(10)_2 - (1)_2 = (1)_2$

$$(1)_2 - (1)_2 = (0)_2$$

# Multiplikation im Binärsystem

---

- Multiplikation für Dezimalzahlen:
  - Multiplikand ziffernweise mit dem Multiplikator multiplizieren
  - Ergebnisse jeweils um eine Stelle verschoben untereinander schreiben
  - Zwischenergebnisse addieren
- Zum Beispiel:

$$\begin{array}{r} 145 \cdot 243 \\ \hline 290 \\ 580 \\ 11435 \\ \hline 35235 \end{array}$$

- Im Binärsystem einfacher, da weniger Ziffern



# Multiplikation im Binärsystem

- Die Multiplikationsregeln lauten:

$0 \cdot 0 = 0$
$0 \cdot 1 = 0$
$1 \cdot 0 = 0$
$1 \cdot 1 = 1$

- Man kann dieselbe Methode wie bei Dezimalzahlen verwenden, z.B.:

$$\begin{array}{r} 1\ 0\ 0\ 1 \cdot 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 1 \\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 1 \\ \hline 1\ 1\ 0\ 0\ 0\ 1\ 1 \end{array}$$

*Zeilen, die durch Multiplikation mit 0 entstehen, können auch weggelassen werden.*

# Multiplikation im Binärsystem

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

- Für Implementierung im Rechner günstiger:
  - mit der am weitesten rechts stehenden Ziffer des Multiplikators beginnen
  - Teilergebnisse nach links verschieben (a).
- Denn
  - nicht alle Zwischenergebnisse müssen gespeichert werden
  - jedes Teilergebnis kann sofort zum Endergebnis addiert werden
  - ermöglicht effizientere Implementierung (b).

(a)

$$\begin{array}{r}
 1\ 0\ 0\ 1 \cdot 1\ 0\ 1\ 1 \\
 \hline
 1\ 0\ 0\ 1 \leftarrow \\
 \phantom{1\ 0\ 0\ 1} 1\ 0\ 0\ 1 \leftarrow \\
 \phantom{1\ 0\ 0\ 1} 1\ 1\ 1\ 0\ 0\ 1 \leftarrow \\
 \hline
 1\ 1\ 0\ 0\ 0\ 1\ 1
 \end{array}$$

(b)

$$\begin{array}{r}
 1\ 0\ 0\ 1 \cdot 1\ 0\ 1\ 1 \\
 \hline
 1\ 0\ 0\ 1 \leftarrow \\
 \phantom{1\ 0\ 0\ 1} 1\ 0\ 0\ 1 \leftarrow \\
 \phantom{1\ 0\ 0\ 1} 1\ 1\ 0\ 1\ 1 \leftarrow \\
 \hline
 1\ 1\ 0\ 0\ 0\ 1\ 1
 \end{array}$$

bzw. +

$$\begin{array}{r}
 1\ 0\ 0\ 1 \cdot 1\ 0\ 1\ 1 \\
 \hline
 1\ 0\ 0\ 1 \leftarrow \\
 \phantom{1\ 0\ 0\ 1} 1\ 0\ 0\ 1 \leftarrow \\
 \phantom{1\ 0\ 0\ 1} 1\ 1\ 0\ 1\ 1 \leftarrow \\
 \hline
 1\ 1\ 0\ 0\ 0\ 1\ 1
 \end{array}$$

# Multiplikation im Binärsystem

$0 \cdot 0 = 0$
$0 \cdot 1 = 0$
$1 \cdot 0 = 0$
$1 \cdot 1 = 1$

- **Wichtiger Spezialfall:** Multiplikation mit Zweierpotenz im Binärsystem

$$\begin{array}{r} 5\ 7\ 1 \cdot 1\ 0\ 0\ 0 \\ \hline 5\ 7\ 1 \\ 0\ 0\ 0 \\ 0\ 0\ 0 \\ 0\ 0\ 0 \\ \hline 5\ 7\ 1\ 0\ 0\ 0 \end{array}$$

$$\begin{array}{r} 1\ 0\ 0\ 1 \cdot 1\ 0\ 0\ 0 \\ \hline 1\ 0\ 0\ 1 \\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0 \\ \hline 1\ 0\ 0\ 1\ 0\ 0\ 0 \end{array}$$

- Verschiebung des Multiplikanden um drei Stellen nach links, weil  $(1000)_2 = (2^3)_{10}$
- Selber Effekt wie bei Multiplikation mit Zehnerpotenz im Dezimalsystem
- **Allgemein:** Multiplikation mit  $2^k$  entspricht einer Verschiebung des Multiplikanden um  $k$  Stellen nach links
  - Englisch: *shift*
  - Bei der Implementierung effizienter Algorithmen oft praktisch

# Division im Binärsystem

---

- Division im Dezimalsystem:

$$\begin{array}{r} 7 \overline{) 431} : 3 = 247 \text{ R } 7 \\ 14 \\ 23 \\ 21 \end{array}$$

- Im Dezimalsystem ist ein wesentlicher Bestandteil des Dividierens das Erraten einer Quotientenziffer.
- Betrachtet man zum Beispiel die Division  $568 : 63 = ?$   
so sieht man zwar, dass 63 nicht mehr als 9-mal in 568 enthalten ist;
- Aber ob 63 nun 9-mal oder vielleicht nur 8-mal oder etwas noch weniger oft in 568 enthalten ist, bedarf eines geschulten Blickes (oder eines Taschenrechners)
- Im Binärsystem kommen als mögliche Quotientenziffern nur 0 und 1 in Frage

# Division im Binärsystem

## ■ Beispiel:

$$\begin{array}{r} 1\ 1\ 0\ 1\ 0\ 1 : 1\ 0\ 1\ 0 = 1 \\ - \underline{1\ 0\ 1\ 0} \\ 0\ 0\ 1\ 1\ 0 \end{array}$$



$$\begin{array}{r} 1\ 1\ 0\ 1\ 0\ 1 : 1\ 0\ 1\ 0 = 1\ 0 \\ - \underline{1\ 0\ 1\ 0} \\ 0\ 0\ 1\ 1\ 0 \\ - \underline{0\ 0\ 0\ 0} \\ 1\ 1\ 0\ 1 \end{array}$$

Dividend : Divisor = Quotient



$$\begin{array}{r} 1\ 1\ 0\ 1\ 0\ 1 : 1\ 0\ 1\ 0 = 1\ 0\ 1 \\ - \underline{1\ 0\ 1\ 0} \\ 0\ 0\ 1\ 1\ 0 \\ - \underline{0\ 0\ 0\ 0} \\ 1\ 1\ 0\ 1 \\ - \underline{1\ 0\ 1\ 0} \\ 0\ 0\ 1\ 1\ \text{Rest} \end{array}$$

## ■ Entscheiden ob Divisor $\leq$ Teil des Dividenden („geht rein“)

- Ja: Quotientenziffer = 1, Nein: Quotientenziffer = 0
- Bei Ziffern mit gleicher Bitzahl ist immer diejenige größer, die von links das erste 1-Bit hat, das in der anderen 0 ist (Beispiel: 10010000 ist größer als 10001111)

# Zahlendarstellung — Übersicht

---

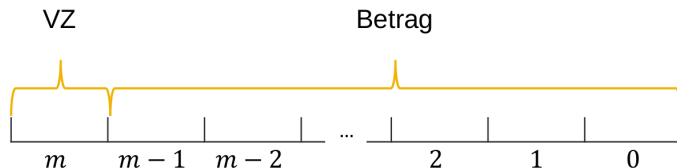
- 1 Bits und Bytes
- 2 Zahlenumwandlungen — Ganzzahlen binär und dezimal
- 3 Zahlen mit Nachkommateil
- 4 Zahlensysteme allgemein
- 5 Rechnen im Binärsystem
- 6 Darstellung negativer (Ganz-)Zahlen**
  - Vorzeichenbit
  - Einerkomplementdarstellung
  - Zweierkomplementdarstellung
  - Exzessdarstellung

# Darstellung negativer Zahlen

---

- **Bisher:** Darstellung positiver Zahlen (wie zum Beispiel 1815)
- **Jetzt:** Darstellung positiver *und negativer* Zahlen (wie zum Beispiel -209)
- Rechner-intern nur 0 und 1 zur Verfügung
  - Kodierung des Minus?
- Wir betrachten vier verschiedene Möglichkeiten (jeweils mit gewissen Vor- und Nachteilen):
  - Vorzeichen und Betrag
  - Einerkomplement
  - Zweierkomplement
  - Exzessdarstellung
- **Annahme:** Alle Zahlen, die wir darstellen, haben  $m + 1$  Bits
- **Ziel:** Benutzen der rechner-internen Addition zur Subtraktion

# Darstellung negativer Zahlen - Vorzeichenbit



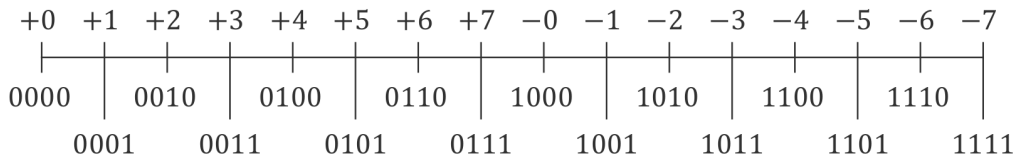
- Einfachste Möglichkeit: Verwendung von **Vorzeichenbit**, Rest wie bisher
- $m$ -te Bit: Vorzeichen (VZ)
  - positives Vorzeichen ... 0
  - negatives Vorzeichen ... 1
- Restliche  $m$  Bits: Betrag der darzustellenden Zahl
- Vorzeichen getrennt behandeln
  - bei arithmetischen Operationen und bei Vergleichsoperationen
- Bsp. für  $m = 6$ :

$$\begin{array}{r|l} +27 & 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \\ -27 & 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \end{array}$$



# Darstellung negativer Zahlen - Vorzeichenbit

- Darstellung als Zahlengerade für  $m = 3$ :



- Ordnungsrelation gilt nur innerhalb der positiven Zahlen
  - Negativen Zahlen kommen „hinter“ den positiven
- Die Zahl 0 besitzt zwei Darstellungen ( $-0$  /  $+0$ )

# Darstellung negativer Zahlen - Vorzeichenbit

## ■ Beispiel: Addition von zwei Zahlen in Vorzeichenbit Codierung

■  $x = 10\ 1001$

(entspricht -9)

■  $y = 00\ 1010$

(entspricht +10)

## ■ Bei naiver Addition:

■  $10\ 1001 + 00\ 1010 = 11\ 0011$

(entspricht -19)

■ Falsches Resultat

## ■ Korrekte Berechnung via Fallunterscheidung

### ■ Gleiches Vorzeichen von $x$ und $y$ :

■ Addition der Beträge

■ VZ behalten

### ■ Unterschiedliches Vorzeichen von $x$ und $y$ :

■ Subtraktion der Beträge

■ VZ der betragsmäßig größeren Zahl

## ■ Im Beispiel:

■ VZ von  $x$ : 1, VZ von  $y$ : 0

■ Beträge subtrahieren:  $0\ 1010 - 0\ 1001 = 0\ 0001$

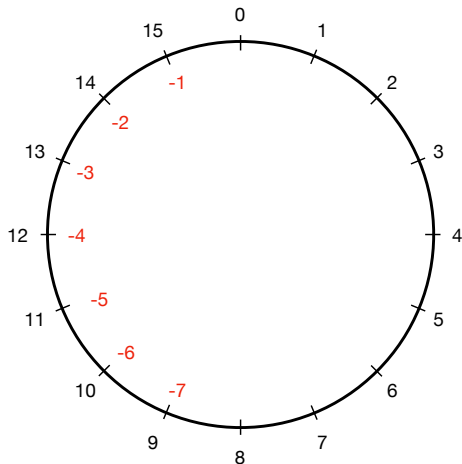
■ Ergebnis  $00\ 0001$

(entspricht +1)

⇒ Wir benötigen weiterhin eine dezidierte Subtraktion-Operation

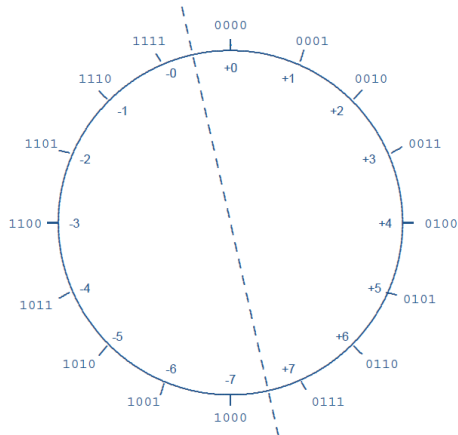
# Subtraktion beim Modulo-Rechnen

- Wie werden wir die Subtraktion-Operation los?
- Am besten wäre es, wenn wir nicht zwischen positiven und negativen Zahlen unterscheiden müssten
- Erinnerung ans Modulo-Rechnen:
  - Beim Modulo-Rechnen sind Subtraktion und Addition fast das gleiche:
    - $8 - 3 \bmod 16 = 5$
    - Alternativ können wir  $-3 \bmod 16$  auch darstellen als  $13 \bmod 16$
    - $8 - 3 \bmod 16$   
 $= 8 + 13 \bmod 16$   
 $= 21 \bmod 16 = 5$
  - ⇒ Konnte negative  $-3$  durch positive  $13$  ersetzen
  - Beim Rechnen  $\bmod k$  können also negative Zahlen  $-x$  einfach durch "große" positive Zahlen  $k - x$  ersetzen
    - Im Beispiel oben:  $-3$  durch  $13$  ersetzt



# Einerkomplementdarstellung

- Erinnerung ans Modulo-Rechnen:
  - Beim Modulo-Rechnen sind Subtraktion und Addition fast das gleiche:
    - Alternativ können wir  $-3 \bmod 16$  auch darstellen als  $13 \bmod 16$
  - Beim Rechnen  $\bmod k$  können also negative Zahlen  $-x$  einfach durch "große" positive Zahlen  $k - x$  ersetzen
    - Im Beispiel oben:  $-3$  durch  $13$  ersetzt
- Genau das ist die Idee vom **Einerkomplement**:
  - Ersetze  $-u$  durch  $(2^{m+1} - 1) - u$ 
    - Statt der negativen Zahl  $-u$  speichern wir die große (positive) Zahl  $(2^{m+1} - 1) - u$
    - Warum  $2^{m+1} - 1$ ?
      - ⇒ Die größte Zahl, die wir mit  $m + 1$  Bits darstellen können ist  $2^{m+1} - 1$
  - Rechts: Beispiel für  $m = 3$



# Einerkomplementdarstellung

Umwandeln einer Binärzahl in Einerkomplementdarstellung

**Gegeben:** Binärzahl  $u$  und Bitbreite der Einerkomplementdarstellung  $m + 1$

**Gesucht:**  $(m + 1)$ -Bit Einerkomplementdarstellung von  $u$

Positive Zahl  $u$ :

- Bei  $u$  von links 0en ergänzen bis man  $m + 1$  Stellen hat
- Beispiel:  $u = (1110\ 0011)_2$  mit  $m = 9$ 
  - EK-Darstellung: 00 1110 0011

Negative Zahl  $u$ :

- 1.Schritt: Bei  $u$  von links 0en ergänzen bis man  $m + 1$  Stellen hat
- 2.Schritt: Alle Bits komplementieren, d.h. 0en und 1en austauschen ("flippen")
  - ⇒ Das Komplementieren ist das gleiche wie zu rechnen  $\underbrace{11 \dots 11}_{m+1 \text{ mal}} - u$ , da dies der Rechnung  $(2^{m+1} - 1 - u)_{10}$  entspricht
- Beispiel:  $u = (-1110\ 0011)_2$  mit  $m = 9$ 
  - 1.Schritt: 00 1110 0011
  - 2.Schritt: 11 0001 1100 (EK-Darstellung)

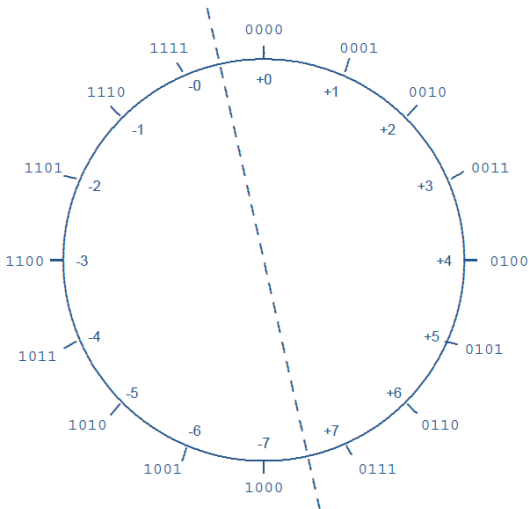
# Einerkomplementdarstellung

- Man stelle sich die Zahlen auf einen Kreis angeordnet vor (rechts für  $m = 3$ )

- Aufgrund der zweifachen Darstellung der 0 können Probleme auftreten
- Beispiel:

$$\begin{array}{r|l} 0110 & +6 \\ 1011 & -4 \\ \hline 10001 & +1 \end{array}$$

- Daher Überlauf 1 addieren!



# Addition in Einerkomplementdarstellung – Beispiel 1 ( $m = 5$ )

■ Berechnung  $-25 + 19$

$$\begin{array}{r} 1\ 0\ 0\ 1\ 1\ 0 -25 \\ +\ 0\ 1\ 0\ 0\ 1\ 1\ 19 \\ \hline \phantom{1\ 0\ 0\ 1\ 1\ 0} 1 \end{array}$$



$$\begin{array}{r} 1\ 0\ 0\ 1\ 1\ 0 -25 \\ +\ 0\ 1\ 0\ 1\ 1\ 1\ 19 \\ \hline \phantom{1\ 0\ 0\ 1\ 1\ 0} 0\ 1 \end{array}$$



$$\begin{array}{r} 1\ 0\ 0\ 1\ 1\ 0 -25 \\ +\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 19 \\ \hline \phantom{1\ 0\ 0\ 1\ 1\ 0} 0\ 0\ 1 \end{array}$$



$$\begin{array}{r} 1\ 0\ 0\ 1\ 1\ 0 -25 \\ +\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 19 \\ \hline \phantom{1\ 0\ 0\ 1\ 1\ 0} 1\ 0\ 0\ 1 \end{array}$$



$$\begin{array}{r} 1\ 0\ 0\ 1\ 1\ 0 -25 \\ +\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 19 \\ \hline \phantom{1\ 0\ 0\ 1\ 1\ 0} 1\ 1\ 0\ 0\ 1 \end{array}$$



$$\begin{array}{r} 1\ 0\ 0\ 1\ 1\ 0 -25 \\ +\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 19 \\ \hline \phantom{1\ 0\ 0\ 1\ 1\ 0} 1\ 1\ 1\ 0\ 0\ 1 -6 \end{array}$$

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 10$





### Addition in Einerkomplementdarstellung - Beispiel 3 ( $m = 5$ )

- Addition zweier negativer Zahlen:  $-6 + (-19)$ 
  - Das "Überlauf-Bit" im vorletzten Schritt impliziert, dass wir  $+1$  hinzufügen müssen

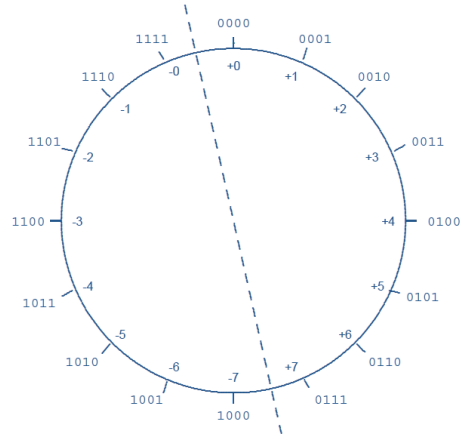
$$\begin{array}{r}
 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6 \\
 + \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ -19 \\
 \hline
 \phantom{1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6} 1
 \end{array}
 \Rightarrow
 \begin{array}{r}
 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6 \\
 + \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ -19 \\
 \hline
 \phantom{1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6} 0 \ 1
 \end{array}
 \Rightarrow
 \begin{array}{r}
 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6 \\
 + \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ -19 \\
 \hline
 \phantom{1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6} 1 \ 0 \ 1
 \end{array}
 \Rightarrow
 \begin{array}{r}
 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6 \\
 + \ 1_1 0 \ 1 \ 1 \ 0 \ 0 \ -19 \\
 \hline
 \phantom{1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6} 0 \ 1 \ 0 \ 1
 \end{array}
 \Rightarrow
 \begin{array}{r}
 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6 \\
 + \ 1_1 1_1 0 \ 1 \ 1 \ 0 \ 0 \ -19 \\
 \hline
 \phantom{1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6} 0 \ 0 \ 1 \ 0 \ 1
 \end{array}
 \Rightarrow
 \begin{array}{r}
 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6 \\
 + \ 1_1 1_1 0 \ 1 \ 1 \ 0 \ 0 \ -19 \\
 \hline
 \phantom{1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6} 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1
 \end{array}
 \Rightarrow
 \begin{array}{r}
 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6 \\
 + \ 1_1 1_1 0 \ 1 \ 1 \ 0 \ 0 \ -19 \\
 \hline
 \phantom{1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6} 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \\
 + \phantom{1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6} \phantom{1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1} 1 \ 1 \\
 \hline
 \phantom{1 \ 1 \ 1 \ 0 \ 0 \ 1 \ -6} 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ -25
 \end{array}$$

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 10$



# Einerkomplementdarstellung

- Zusammenfassung der Einerkomplementdarstellung
  - Negative Zahlen werden durch ihre (binären) Komplemente dargestellt
  - Wir konnten Subtraktion durch Addition ersetzen (gut!)
  - Um korrekt zu addieren, brauchen wir weiterhin diverse Fallunterscheidungen wegen Doppelung +0/-0 (schlecht!)
- Können wir jetzt noch die Fallunterscheidungen loswerden?
  - Das Problem ergibt sich durch die “doppelte 0”
    - ⇒ Alle negativen Zahlen um eins weiterschieben
      - Das ist die Idee vom Zweierkomplement



# Zweierkomplementdarstellung

- Negative Zahlen im Zweierkomplement durch Ergänzung auf  $2^{m+1}$  dargestellt
- Ausgangspunkt: Einerkomplement der Zahl
  - Berechnung negativer (!) Zahlen: 1 dazu addieren

	+27	0 0 0 1 1 0 1 1	
Einerkomplementdarst.:	-27	1 1 1 0 0 1 0 0	
Zweierkomplementdarst.:	-27	1 1 1 0 0 1 0 1	<i>1 dazu addieren</i>

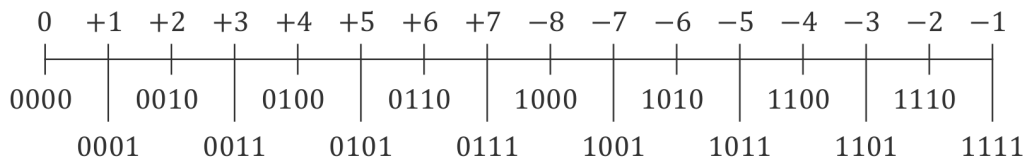
- Alternativ: die binären Ziffern der positiven Zahl von rechts nach links bis inkl. zur ersten 1 kopieren und die restlichen Ziffern komplementieren.

+27	0 0 0 1 1 0 1 1
-27	1 1 1 0 0 1 0 1

*restliche Ziffern      von rechts bis zur ersten 1 kopiert  
komplementiert*

# Zweierkomplementdarstellung - Ordnungsrelation

- Bsp.:  $m = 3$



- Ordnungsrelation wie beim Einerkomplement
  - Ordnung (getrennt) innerhalb der positiven und negativen Zahlen
  - Die negativen Zahlen kommen „hinter“ den positiven
- Die Zahl 0 hat eine eindeutige Darstellung

# Zweierkomplementdarstellung

- Positive und negative Zahlen können am führenden Bit unterschieden werden.
- Überläufe bei Rechnungen kann man ignorieren, da die 0 eine eindeutige Darstellung besitzt

dezimale Zahl	dezimale Kodierung	binäre Kodierung
0	0	000 ... 00
1	1	000 ... 01
$\vdots$	$\vdots$	$\vdots$
$2^m - 1$	$2^m - 1$	011 ... 11
$-2^m$	$2^m$	100 ... 00
$-2^m + 1$	$2^m + 1$	100 ... 01
$\vdots$	$\vdots$	$\vdots$
-1	$2^{m+1} - 1$	111 ... 11

# Addition in Zweierkomplementdarstellung – Beispiel 1 ( $m = 5$ )

$$\begin{array}{r} 0\ 1\ 1\ 0\ 0\ \mathbf{1}\ 25 \\ +\ 1\ 0\ 1\ 1\ \mathbf{0}\ \mathbf{1}\ -19 \\ \hline \mathbf{0} \end{array}$$



$$\begin{array}{r} 0\ 1\ 1\ 0\ \mathbf{0}\ 1\ 25 \\ +\ 1\ 0\ 1\ 1\ \mathbf{0}\ 1\ -19 \\ \hline \mathbf{1}\ 0 \end{array}$$



$$\begin{array}{r} 0\ 1\ 1\ \mathbf{0}\ 0\ 1\ 25 \\ +\ 1\ 0\ 1\ \mathbf{1}\ \mathbf{0}\ 1\ -19 \\ \hline \mathbf{1}\ 1\ 0 \end{array}$$



$$\begin{array}{r} 0\ 1\ \mathbf{1}\ 0\ 0\ 1\ 25 \\ +\ \mathbf{1}\ \mathbf{0}\ \mathbf{1}\ 1\ \mathbf{0}\ 1\ -19 \\ \hline \mathbf{0}\ 1\ 1\ 0 \end{array}$$



$$\begin{array}{r} 0\ \mathbf{1}\ 1\ 0\ 0\ 1\ 25 \\ +\ \mathbf{1}\ \mathbf{1}\ \mathbf{0}\ 1\ 1\ \mathbf{0}\ 1\ -19 \\ \hline \mathbf{0}\ 0\ 1\ 1\ 0 \end{array}$$



$$\begin{array}{r} \mathbf{0}\ 1\ 1\ 0\ 0\ 1\ 25 \\ +\ \mathbf{1}\ \mathbf{1}\ \mathbf{0}\ 1\ 1\ \mathbf{0}\ 1\ -19 \\ \hline \mathbf{1}\ \mathbf{0}\ 0\ 0\ \mathbf{1}\ 1\ 0\ 6 \end{array}$$

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0$

# Addition in Zweierkomplementdarstellung – Beispiel 2 ( $m = 5$ )

$$\begin{array}{r} 1\ 0\ 0\ 1\ 1\ \textcolor{red}{1}\ -25 \\ +\ 0\ 1\ 0\ 0\ \textcolor{red}{1}\ \textcolor{red}{1}\ 19 \\ \hline \textcolor{red}{0} \end{array}$$



$$\begin{array}{r} 1\ 0\ 0\ 1\ \textcolor{red}{1}\ 1\ -25 \\ +\ 0\ 1\ 0\ \textcolor{red}{1}\ \textcolor{red}{1}\ 1\ 19 \\ \hline \textcolor{red}{1}\ 0 \end{array}$$



$$\begin{array}{r} 1\ 0\ 0\ \textcolor{red}{1}\ 1\ 1\ -25 \\ \textcolor{red}{\Rightarrow} +\ 0\ 1\ \textcolor{red}{1}\ \textcolor{red}{0}\ \textcolor{red}{1}\ 1\ 19 \\ \hline \textcolor{red}{0}\ 1\ 0 \end{array}$$



$$\begin{array}{r} 1\ 0\ \textcolor{red}{0}\ 1\ 1\ 1\ -25 \\ +\ 0\ 1\ \textcolor{red}{1}\ \textcolor{red}{0}\ \textcolor{red}{1}\ 1\ 19 \\ \hline \textcolor{red}{1}\ 0\ 1\ 0 \end{array}$$



$$\begin{array}{r} 1\ \textcolor{red}{0}\ 0\ 1\ 1\ 1\ -25 \\ \textcolor{red}{\Rightarrow} +\ 0\ \textcolor{red}{1}\ \textcolor{red}{1}\ \textcolor{red}{0}\ \textcolor{red}{1}\ 1\ 19 \\ \hline \textcolor{red}{1}\ 1\ 0\ 1\ 0 \end{array}$$



$$\begin{array}{r} \textcolor{red}{1}\ 0\ 0\ 1\ 1\ 1\ -25 \\ +\ \textcolor{red}{0}\ 1\ \textcolor{red}{1}\ \textcolor{red}{0}\ \textcolor{red}{1}\ 1\ 19 \\ \hline \textcolor{red}{1}\ 1\ 1\ 0\ 1\ 0\ -6 \end{array}$$

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 10$



### Addition in Zweierkomplementdarstellung – Beispiel 3 ( $m = 5$ )

$$\begin{array}{rrrrrrr} 1 & 1 & 1 & 0 & 1 & 0 & -6 \\ + & 1 & 0 & 1 & 1 & 1 & -19 \\ \hline & & & & & 1 & \end{array}$$



$$\begin{array}{rrrrrrr}
 1 & 1 & 1 & 0 & \color{red}{1} & 0 & -6 \\
 + & 1 & 0 & 1 & 1 & \color{red}{0} & -19 \\
 \hline
 & & & & \color{red}{1} & 1 & 
 \end{array}
 \rightarrow$$



$$\begin{array}{rcccccc} & 1 & 1 & 1 & 0 & 1 & 0 & -6 \\ \Rightarrow + & 1 & 0 & 1 & 1 & 0 & 1 & -19 \\ \hline & & & & 1 & 1 & 1 & \end{array}$$



$$\begin{array}{rrrrrrr}
 1 & 1 & 1 & 0 & 1 & 0 & -6 \\
 + & 1 & 0 & 1 & 0 & 1 & -19 \\
 \hline
 & 0 & 1 & 1 & 1 & & 
 \end{array}
 \rightarrow$$



$$\begin{array}{r} 1 \text{ } 1 \text{ } 1 \text{ } 0 \text{ } 1 \text{ } 0 \text{ } -6 \\ + \text{ } 1 \text{ } 1 \text{ } 0 \text{ } 1 \text{ } 1 \text{ } 0 \text{ } 1 \text{ } -19 \\ \hline 0 \text{ } 0 \text{ } 1 \text{ } 1 \text{ } 1 \end{array}$$



$$\begin{array}{rrrrrrr} 1 & 1 & 1 & 0 & 1 & 0 & -6 \\ + & 1 & 0 & 1 & 1 & 0 & 1 & -19 \\ \hline 1 & 0 & 0 & 1 & 1 & 1 & -25 \end{array}$$

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 10$

# Überschreitung des Zahlenbereichs

- Eine tatsächliche Überschreitung des Zahlenbereiches kann wieder durch einen Plausibilitätstest des Vorzeichens erkannt werden, z.B.:

$$\begin{array}{r} 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ -25 \\ + \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ -19 \\ \hline \textcolor{red}{1} \ \textcolor{red}{0} \ \textcolor{red}{1} \ \textcolor{red}{0} \ \textcolor{red}{1} \ \textcolor{red}{0} \ \textcolor{red}{0} \ +20 \end{array}$$

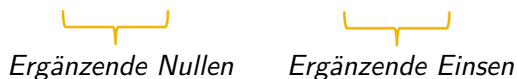
- Da die Summe zweier negativer Zahlen nicht positiv sein kann, ist bei der Addition ein wirklicher Überlauf entstanden.

# Multiplikation in Zweierkomplementdarstellung (1 von 2)

- Da die Multiplikation zweier  $m$ -stelliger Zahlen ein  $2m$ -stelliges produzieren kann, müssen die Faktoren vor der Durchführung der Rechenoperation auf  $2m$  Stellen erweitert werden
- Dabei ist zu beachten, dass positive Zahlen mit Nullern, negative Zahlen mit Einsern ergänzt werden müssen, um das Vorzeichen nicht zu zerstören

- Bsp. 1:

$$\begin{array}{rcl} (6)_{10} & \cdot & (-3)_{10} = (-18)_{10} \\ \hline (000000000110)_2 & \cdot & (111111111101)_2 = (101110)_2 \end{array}$$



## Multiplikation in Zweierkomplementdarstellung (2 von 2)

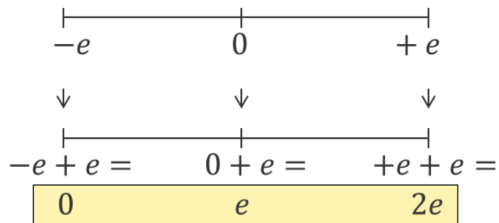
### ■ Bsp. 2: $m = 3$

1 1 1 1 1 0 0 1	$(-7)_{10}$ mit Vorzeichenerweiterung
· 1 1 1 1 1 1 0 1	$(-3)_{10}$ mit Vorzeichenerweiterung
<hr/>	
1 1 1 1 1 0 0 1	$(11111001 \cdot 1, \text{um null Stellen nach links verschoben})$
0 0 0 0 0 0 0 0	$(11111001 \cdot 0, \text{um eine Stelle nach links verschoben})$
1 1 1 0 0 1 0 0	$(11111001 \cdot 1, \text{um zwei Stellen nach links verschoben})$
1 1 0 0 1 0 0 0	$(11111001 \cdot 1, \text{um drei Stellen nach links verschoben})$
1 0 0 1 0 0 0 0	$(11111001 \cdot 1, \text{um vier Stellen nach links verschoben})$
0 0 1 0 0 0 0 0	$(11111001 \cdot 1, \text{um fünf Stellen nach links verschoben})$
0 1 0 0 0 0 0 0	$(11111001 \cdot 1, \text{um sechs Stellen nach links verschoben})$
+ 1 0 0 0 0 0 0 0	$(11111001 \cdot 1, \text{um sieben Stellen nach links verschoben})$
<hr/>	
0 0 0 1 0 1 0 1	$(+21)_{10}, \text{in 4 Bit nicht darstellbar}$

- Die Bits werden über den linken Rand hinausgeschoben, Überlauf kann vernachlässigt werden.

# Exzessdarstellung

- Weitere Möglichkeit, negative Ganzzahlen darzustellen: [Exzessdarstellung](#)
- Exzess  $e$  addieren, sodass Zahlen nicht mehr negativ sind
- Symmetrischer Exzess



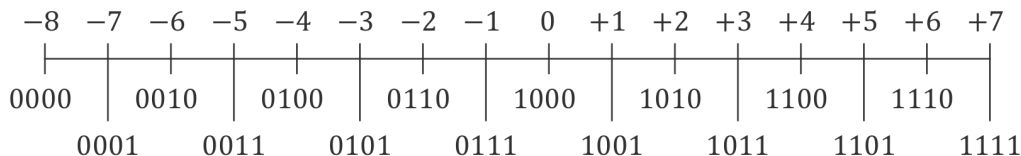
Exzess  $e$  addieren

Zahlen in Exzessdarstellung sind um  $e$  größer als sie eigentlich sind

- Exzess ergibt sich aus kleinster darstellbarer negativer Zahl
- $0_e = e$
- Welcher Zahlenbereich binär darstellbar?

# Exzessdarstellung - Ordnungsrelation

- Bsp.: Anzahl Binärstellen  $m + 1 = 4$ , Exzess  $e = 2^m = 2^3 = 8$



- Weiterer symmetrischer Exzess:  $e = 2^m - 1$ 
  - Bei  $m + 1 = 4$ , Exzess  $e = 2^m - 1 = 7$  also  $(-7, \dots, +8)$  darstellbar
- Ordnungserhaltend
  - Kleinere Zahlen in Exzessdarstellung auch kleiner
- Zahl 0 hat eine eindeutige Darstellung

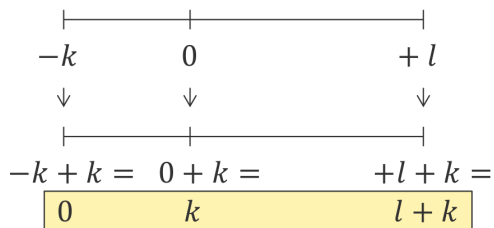
# Exzessdarstellung und Zweierkomplement

- Für  $e = 2^m$  ist Exzessdarstellung für negative Zahlen identisch mit Zweierkomplement, wenn man das Bit ganz links invertiert
  - Addition und Subtraktion von  $e$  durch invertieren des Bits ganz links

Dezimal	Exzessdarstellung, $e = 8$	Zweierkomplement
7	1111	0111
6	1110	0110
5	1101	0101
4	1100	0100
3	1011	0011
2	1010	0010
1	1001	0001
0	1000	0000
-1	0111	1111
-2	0110	1110
-3	0101	1101
-4	0100	1100
-5	0011	1011
-6	0010	1010
-7	0001	1001
-8	0000	1000

# Exzessdarstellung

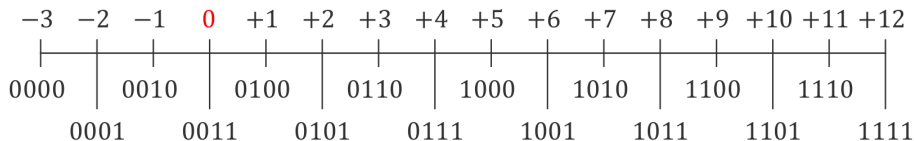
## ■ Nicht-symmetrischer Exzess



Exzess  $k$  addieren

Zahlen in Exzessdarstellung sind um  $k$  größer als sie eigentlich sind

- Kleinste darstellbare negative Zahl  $-k \rightarrow$  Exzess  $k$
- Exzess  $k$  addieren, sodass Zahlen nicht mehr negativ sind
- Bsp.:  $k = 3$ ,  $m + 1 = 4$





# Exzessdarstellung – Rechenoperationen

---

- Exzess bei arithmetischen Operationen berücksichtigen!
  - Addition: Exzess von der Summe subtrahieren
  - Subtraktion: Exzess zur Summe addieren
- Beispiel im Dezimalsystem:
  - Exzess  $e = 12$ ,  $A = -3$ ,  $B = 4$ ,  $A + B = ?$
  - $A_e = A + e = -3 + 12 = 9$
  - $B_e = B + e = 4 + 12 = 16$
  - $A_e + B_e = A + e + B + e = A + B + 2e = -3 + 4 + 2 \cdot 12 = 25$
  - $(A + B)_e = A + B + e = -3 + 4 + 12 = 13$
  - $(A + B)_e = A_e + B_e - e = 9 + 16 - 12 = 13$

# Exzessdarstellung – Beispiel

■ Bsp.:  $-(A + B)$

■  $A_e = 011001$ ,  $B_e = 110101$ ,  $e = 100101$

1. Schritt:  $(A + B)_e = A_e + B_e - e$

2. Schritt:  $-(A + B)_e = 0_e - (A + B)_e + e$

	0	1	1	0	0	1	-12
+	1	1	0	1	0	1	16
<hr/>							
	1	0	0	1	1	1	0
-	1	0	0	1	0	1	$e$
<hr/>							
	1	0	1	0	0	1	4
<hr/>							

# Exzessdarstellung – Beispiel

■ Bsp.:  $-(A + B)$

■  $A_e = 011001$ ,  $B_e = 110101$ ,  $e = 100101$

1. Schritt:  $(A + B)_e = A_e + B_e - e$

2. Schritt:  $-(A + B)_e = 0_e - (A + B)_e + e$

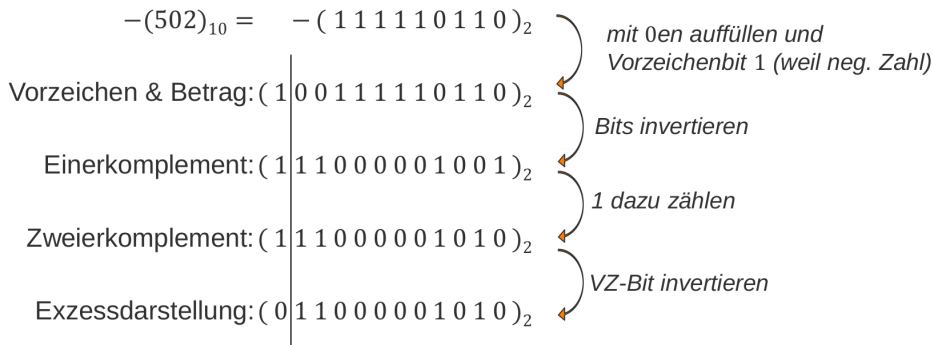
$$\begin{array}{r} 1 \ 0 \ 0 \ 1 \ 0 \ 1 \quad 0_e \\ - \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \quad 4 \\ \hline \end{array}$$

Problem, da Subtrahend größer

$$\begin{array}{r} 1 \ 0 \ 0 \ 1 \ 0 \ 1 \quad 0_e \\ + \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \quad e \\ \hline 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \\ - \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \quad 4 \\ \hline 1 \ 0 \ 0 \ 0 \ 0 \ 1 \quad -4 \\ \hline \end{array}$$

# Vergleich der verschiedenen Zahlendarstellungen 1

- Dezimale Zahl  $-502$  umwandeln auf 12 Bit-Kodierung ( $m = 11$ ) in
  - Vorzeichen mit Betrag
  - Einerkomplement
  - Zweierkomplement
  - Exzessdarstellung mit symmetrischem Exzess  $e = 2^{11}$  (Spezialfall!!)



# Vergleich der verschiedenen Zahlendarstellungen 2

---

- Bitmuster 1101 interpretieren als Zahl  $u$  in der Darstellung
  - Vorzeichen mit Betrag
    - $VZ = 1$ , also negativ, Betrag =  $(101)_2$ , daher  $u = -5$
  - Einerkomplement
    - Erste Stelle 1, also negativ, daher Betrag invertieren
    - Betrag =  $(010)_2$ , daher  $u = -2$
  - Zweierkomplement
    - Erste Stelle 1, also negativ, daher Betrag invertieren und 1 addieren
    - Betrag =  $(011)_2$ , daher  $u = -3$
  - Exzessdarstellung mit  $e = (101)_2$ 
    - binär:  $u = u_e - e = (1101)_2 - (101)_2 = (1000)_2 = 8$
    - dezimal:  $u_e = 13$ ,  $u = u_e - e = 13 - 5 = 8$