

# 1 Problem: Zeichendarstellung im Computer

Eine der am meisten genutzten Möglichkeiten eines Computers ist das Schreiben von Texten. Wie ist es eigentlich möglich, dass ein Computer die ganzen Zeichen eines Textes verarbeiten, darstellen und speichern kann? Denn ein Computer arbeitet im Normalfall mit elektrischem Strom und der kennt im digitalen Zeitalter nur zwei mögliche Zustände:

1. Es liegt eine Spannung an. (1 oder High)
2. Es liegt keine Spannung an. (0 oder Low)

Diese Information ist die kleinste Informationseinheit der Informatik und wird als **Bit** (Binary Digit) bezeichnet. Allerdings kennt ein Bit nur zwei Zustände und damit ist es nur möglich **zwei** Zeichen zu codieren. Für das deutsche Alphabet bräuchten wir aber mindestens 26 Zustände.

Wenn eine Leitung nicht ausreicht, dann nehmen wir doch einfach zwei Leitungen. Damit haben wir nun schon vier verschiedene Zustände.

Leitung 1	Leitung 2
0	0
0	1
1	0
1	1

**A 1.1.** Erstelle eine Tabelle wie oben für den Einsatz von drei und vier Leitungen. Bestimme die Anzahl der Zustände, die jeweils damit dargestellt werden können.

**A 1.2.** Bestimme, wieviele Zustände mit fünf Leitungen dargestellt werden können.

Mit fünf Leitungen können wir das deutsche Alphabet (ohne Groß- und Kleinschreibung) darstellen. Mit jeder weiteren Leitung verdoppelt sich die Anzahl der möglichen Zustände. Mit acht Leitungen können also  $2^8 = 256$  verschiedene Zustände dargestellt werden. Diese Zusammenfassung von acht Leitungen bezeichnet man als **Byte**. Das Byte ist die Grundeinheit um Speichergrößen zu beschreiben.

Die Anzahl der in einem Rechner gebündelten Leitungen bezeichnet man als Busbreite. Die ersten Taschenrechner haben mit vier Bit Busbreite gearbeitet. Die Homecomputer verwendeten dann einen acht Bit breiten Bus. Die Welt des PC startete mit acht Bit breiten Bussen. Mit der Entwicklung des AT (286er) stieg die Busbreite auf 16 Bit und mit dem 386er auf 32 Bit. Heute sind Busbreiten von 64 Bit in Rechnern üblich.

**A 1.3.** Bestimme die Anzahl der Zustände, die mit den oben genannten Busbreiten dargestellt werden können.

## 2 Das binäre Zahlensystem

Ein Rechner heißt Rechner, weil er hauptsächlich rechnet. Selbst wenn man nichts davon sieht, beruhen die meisten Aktionen eines Computers auf Berechnungen. Und für Berechnungen braucht man Zahlen. Wie kann man also mit unseren gebündelten Leitungen natürliche Zahlen darstellen?

Wir rechnen im sogenannten Dezimalsystem (Zehnersystem). Dies liegt daran, dass wir mit den Fingern an unseren Händen bis 10 zählen können. Mit einer Leitung können wir nur zwei Zustände darstellen. Also arbeiten Computer mit dem sogenannten Binärsystem (Dualsystem, Zweiersystem). Bei beiden Systemen handelt es sich um ein Stellenwertsystem. Das heißt die Position der Ziffer in der Zahl sagt etwas über ihren Wert aus.

Die Stellen nummerieren wir von der kleinsten (rechts) zur größten Stelle (links) durch und beginnen dabei mit 0. Bei einer Dezimalzahl kann die Stelle  $z_i$  die Ziffern von 0 bis 9 annehmen. Die Binärzahl hat Stellen  $z_i$ , die nur die Ziffern 0 und 1 annehmen können.

Es gibt eine schöne Aussage über das Binärsystem.

*Im Bezug auf das Binärsystem gibt 10 Gruppen von Menschen.*

*Die Menschen, die es verstehen, und die Menschen, die es nicht verstehen.*

Autsch. Reingelegt. Aber wie sollen wir auch wissen, dass mit 10 die Binärzahl und nicht die Dezimalzahl gemeint ist. Daher schreibt man an die Zahl einen Index, um das Zahlensystem festzulegen:  $10100111_2 = 167_{10}$ . Den Index kann man weglassen, wenn durch den Zusammenhang klar ist, welches System verwendet werden soll.

Dezimalsystem					Binärsystem							
3	2	1	0	Stelle $i$	7	6	5	4	3	2	1	0
T	H	Z	E	Stellenwert	128er	64er	32er	16er	8er	4er	2er	1er
$10^3$	$10^2$	$10^1$	$10^0$		$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
1000	100	10	1		128	64	32	16	8	4	2	1
<b>2</b>	<b>0</b>	<b>4</b>	<b>8</b>	$z_i$	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>
					$  \begin{array}{r}  1 \cdot 128 \\  + 0 \cdot 64 \\  + 1 \cdot 32 \\  + 0 \cdot 16 \\  + 0 \cdot 8 \\  + 1 \cdot 4 \\  + 1 \cdot 2 \\  + 1 \cdot 1 \\  = 167  \end{array}  $							
$  \begin{array}{r}  2 \cdot 1000 \\  + 0 \cdot 100 \\  + 4 \cdot 10 \\  + 8 \cdot 1 \\  = 2048  \end{array}  $												

Tabelle 2.1: Beispiel für die Stellenwerte einer Dezimalzahl und einer Binärzahl

Um die Übersicht über große Zahlen nicht zu verlieren, ist es üblich die Zahl in Blöcke aufzuteilen, die durch einen kleinen Leerraum getrennt werden. Im Dezimalsystem ist ein Block drei Stellen groß (Tausenderblock). Im Binärsystem ist ein Block vier Stellen groß. Dies nennt man einen **Nibble**.

In Tabelle 2.1 sind die Dezimalzahl 2048 und die Binärzahl 10100111 dargestellt. Bei der Dezimalzahl 2048 ist die kleinste Stelle  $z_0 = 8$  und die größte Stelle  $z_3 = 2$ . Bei der Binärzahl 10100111 ist die kleinste Stelle  $z_0 = 1$  und die größte Stelle  $z_7 = 1$ .

Jede Stelle besitzt einen Wert  $w_i$ , der größer ist, je weiter sie links liegt. Bei Dezimalzahlen sind die ersten Werte  $w_0 = 1$ ,  $w_1 = 10$  und  $w_2 = 100$ . Allgemein gilt für den Wert an der Stelle  $i$ :  $w_i = 10^i$ . Bei Binärzahlen sind die ersten Werte  $w_0 = 1$ ,  $w_1 = 2$  und  $w_2 = 4$ . Allgemein gilt für den Wert an der Stelle  $i$ :  $w_i = 2^i$ .

## 2.1 Binärzahl in Dezimalzahl umwandeln

Um den dezimalen Wert einer Binärzahl zu ermitteln, multipliziert man für jede Stelle die Ziffer  $z_i$  mit dem Stellenwert  $w_i$  und zählt alle Ergebnisse zusammen. Ein Beispiel kannst Du in Tabelle 2.1 sehen.

**A 2.1.** Wandle die folgenden Binärzahlen in Dezimalzahlen um.

- a) 00001111<sub>2</sub>                      b) 10101010<sub>2</sub>                      c) 01010101<sub>2</sub>                      d) 10010110<sub>2</sub>

## 2.2 Arbeiten wie ein Rechner

Wie Du eine Binärzahl in eine Dezimalzahl umwandeln kannst, hast Du im letzten Abschnitt erfahren. Diese Arbeitsanweisung bezeichnet man in der Informatik als Algorithmus. Ein solcher Algorithmus kann in Textform beschrieben werden. Weitere Möglichkeiten einen Algorithmus zu beschreiben sind der Programmablaufplan (PAP) und das Struktogramm.

Abbildung 2.1 zeigt den Algorithmus der Umwandlung einer 8-Bit-Binärzahl in eine Dezimalzahl als Programmablaufplan und als Struktogramm.

Dieser Algorithmus sieht komplizierter aus als die Textform. Dies liegt daran, dass er für einen "dümmeren" Ausführer gedacht ist. Um solche Algorithmen besser zu verstehen, ist es sinnvoll selber mal einen solchen Ausführer bzw. Rechner zu spielen. Dazu müssen wir uns an ein paar Spielregeln halten.

- Wir arbeiten mit natürlichen Zahlen.
- Wir können zwei Zahlen addieren, subtrahieren, multiplizieren und ganzzahlig dividieren.
- Wir können den Rest einer Division zweier Zahlen berechnen. (Modulo, %)
- Wir können die Potenz einer Zahl berechnen.
- Wir können zwei Zahlen miteinander vergleichen mit den Operatoren  $<$ ,  $\leq$ ,  $=$ ,  $\geq$  und  $>$  und die Aussage treffen "Wahr" oder "Falsch".
- Werte merken wir uns auf Karteikarten, die einen eindeutigen Namen besitzen. (Variablen)
- Listen sind eine Sammlung von Karteikarten. Die Liste hat einen Namen und die Karteikarten sind von 0 beginnend durchnummeriert.

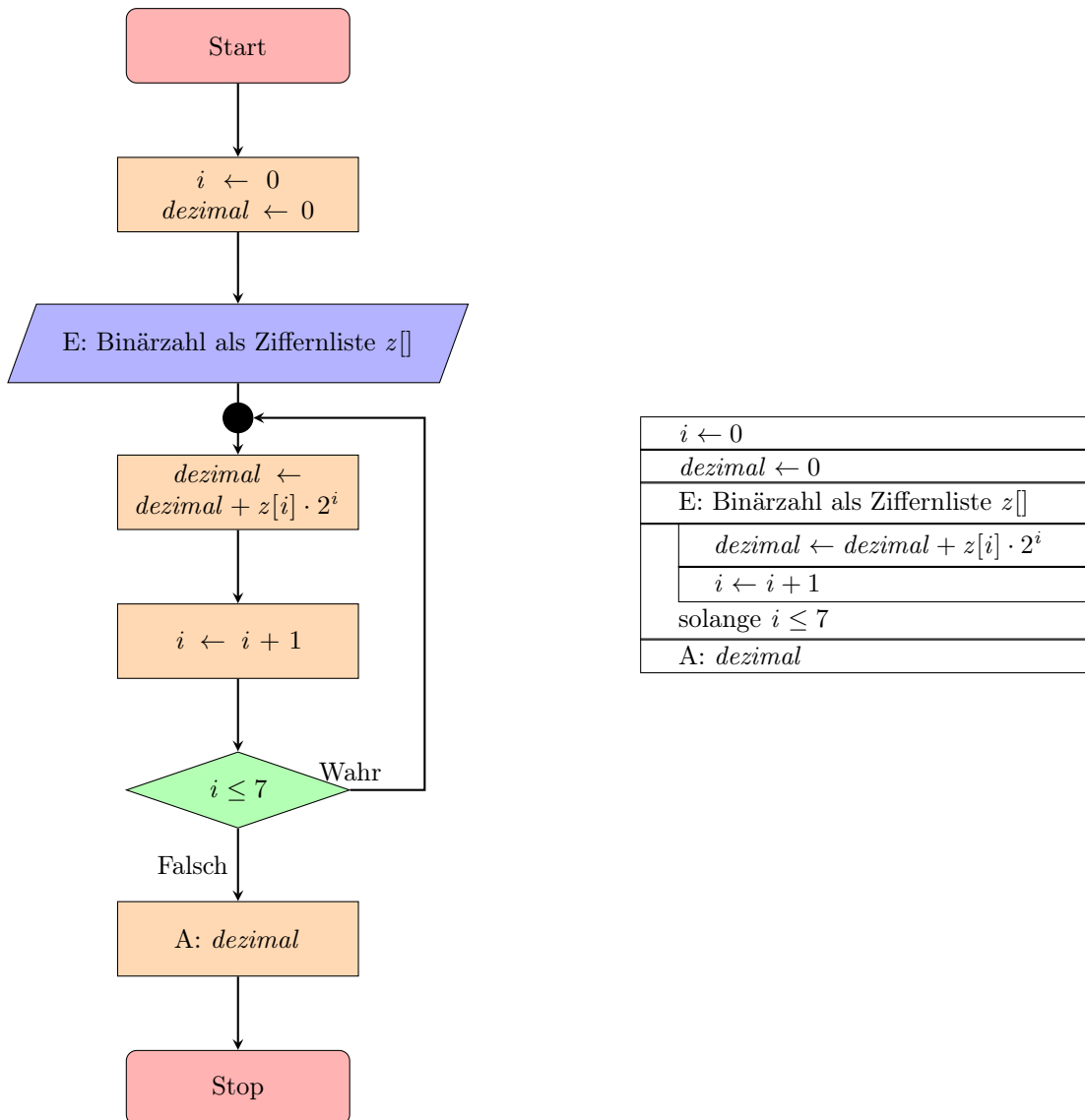


Abbildung 2.1: Umwandlung einer 8-Bit-Binärzahl in eine Dezimalzahl: Links als Programmablaufplan. Rechts als Struktogramm. Hinweis zur Reihenfolge der Ziffernliste:  $z[0]$  ist die rechte Stelle.

- Eingaben machen wir, in dem wir die benannte Karteikarte beschriften.
- Ausgaben machen wir, in dem wir sprechen.

Um den Algorithmus in Abbildung 2.1 durchzuspielen, brauchen wir zwei Karteikarten, die oben mit  $i$  und  $dezimal$  beschriftet werden. Dazu 8 Karteikarten, die von 0 bis 7 durchnummeriert werden. Diese werden zusammengeheftet und mit  $z$  benannt. Beschriftet werden die Karteikarten mit einem Bleistift. Mit einem Radiergummi werden die Werte wieder gelöscht.

**A 2.2.** Spiele die Berechnung der Werte aus Aufgabe 2.1 mit Hilfe von Programmablaufplan und Struktogramm aus Abbildung 2.1 durch.

### 2.3 Dezimalzahl in Binärzahl: Subtraktionsmethode

Die Umwandlung einer Binärzahl in eine Dezimalzahl ist einfach. Die Überführung einer Dezimalzahl in eine Binärzahl ist etwas komplizierter. Schauen wir uns erst einmal die Subtraktionsmethode an, die wir verwenden können, wenn die Anzahl der Stellen der Binärzahl bekannt ist.

1. Wir beginnen beim höchsten Stellenwert der Binärzahl.
2. Wir prüfen, ob der Stellenwert in der Dezimalzahl enthalten ist.
  - Wenn er enthalten ist, dann setzen wir an der Stelle der Binärzahl eine 1 und ziehen den Stellenwert von der Dezimalzahl ab.
  - Wenn er nicht enthalten ist, dann setzen wir an der Stelle der Binärzahl eine 0.
3. Wir nehmen die nächstkleinere Stelle und gehen zu Schritt 2, bis die letzte Stelle der Binärzahl bearbeitet wurde.

Beispiel: Umwandlung der Zahl  $123_{10}$  in eine achtstellige Binärzahl.

Achter Stellenwert (128) ist nicht in 123 enthalten	$\implies$ 8. Stelle 0	123
Siebenter Stellenwert (64) ist in 123 enthalten	$\implies$ 7. Stelle 1	$123 - 64 = 59$
Sechster Stellenwert (32) ist in 59 enthalten	$\implies$ 6. Stelle 1	$59 - 32 = 27$
Fünfter Stellenwert (16) ist in 27 enthalten	$\implies$ 5. Stelle 1	$27 - 16 = 11$
Vierter Stellenwert (8) ist in 11 enthalten	$\implies$ 4. Stelle 1	$11 - 8 = 3$
Dritter Stellenwert (4) ist nicht in 3 enthalten	$\implies$ 3. Stelle 0	3
Zweiter Stellenwert (2) ist in 3 enthalten	$\implies$ 2. Stelle 1	$3 - 2 = 1$
Erster Stellenwert (1) ist in 1 enthalten	$\implies$ 1. Stelle 1	$1 - 1 = 0$

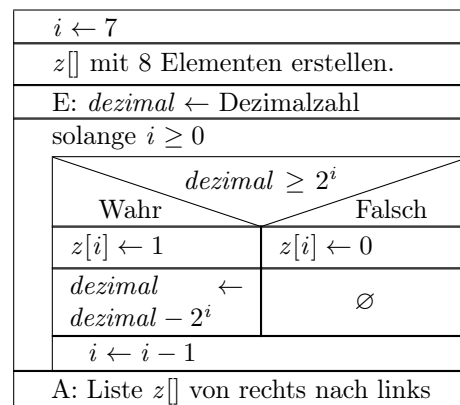
Also ist  $123_{10} = 0111\ 1011_2$ .

**A 2.3.** Wandle die folgenden Dezimalzahlen mit der obigen Methode in achtstellige Binärzahlen um. Notiere die Binärzahlen in Nibble-Block-Schreibweise.

- a)  $240_{10}$                       b)  $105_{10}$                       c)  $37_{10}$                       d)  $42_{10}$

**A 2.4.** Rechts ist das Struktogramm für den Algorithmus abgebildet. Spiele den Algorithmus für die Werte aus Aufgabe 2.3 durch.

**A 2.5.** Erstelle den passenden Programmablaufplan für das rechts stehende Struktogramm.



## 2.4 Dezimalzahl in Binärzahl: Divisionsmethode

Wenn die Anzahl der Stellen der Binärzahl vorher nicht bekannt ist, dann kann man auch folgende Methode verwenden.

1. Beginne bei der kleinsten Stelle der Binärzahl.
2. Teile die Dezimalzahl ganzzahlig durch 2 und bestimme den Rest.
  - Wenn der Rest 1 ist, dann setze die Stelle auf 1.
  - Wenn der Rest 0 ist, dann setze die Stelle auf 0.
3. Wechsle zur nächstgrößeren Stelle der Binärzahl und arbeite mit der ganzzahlig geteilten Dezimalzahl in Schritt 2 weiter, bis die Dezimalzahl 0 ist.

Beispiel: Umwandlung der Zahl  $899_{10}$  in eine Binärzahl.

$899 \div 2 = 449$	R 1	$\implies$	1. Stelle 1
$449 \div 2 = 224$	R 1	$\implies$	2. Stelle 1
$224 \div 2 = 112$	R 0	$\implies$	3. Stelle 0
$112 \div 2 = 56$	R 0	$\implies$	4. Stelle 0
$56 \div 2 = 28$	R 0	$\implies$	5. Stelle 0
$28 \div 2 = 14$	R 0	$\implies$	6. Stelle 0
$14 \div 2 = 7$	R 0	$\implies$	7. Stelle 0
$7 \div 2 = 3$	R 1	$\implies$	8. Stelle 1
$3 \div 2 = 1$	R 1	$\implies$	9. Stelle 1
$1 \div 2 = 0$	R 1	$\implies$	10. Stelle 1

Also ist  $899_{10} = 11\,1000\,0011_2$ .

**A 2.6.** Wandle die folgenden Dezimalzahlen mit der obigen Methode in Binärzahlen um. Notiere die Binärzahlen in Nibble-Block-Schreibweise.

a)  $206_{10}$

b)  $496_{10}$

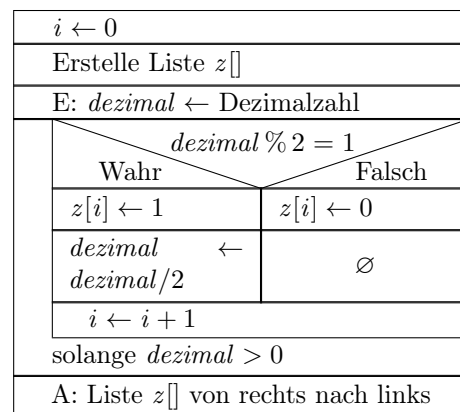
c)  $53_{10}$

d)  $2730_{10}$

Das Zeichen für die Operation *Rest einer Division* (Modulo) ist das Prozentzeichen %. Im Struktogramm rechts ist die Variable *dezimal* eine Ganzzahl. Daher erfolgt die Division auch ganzzahlig. Ein vorhandener Rest wird verworfen. Also ist z.B.  $5/2 = 2$ , wenn wir ganzzahlig teilen.

**A 2.7.** Rechts ist das Struktogramm für den Algorithmus abgebildet. Spiele den Algorithmus für die Werte aus Aufgabe 2.3 durch.

**A 2.8.** Erstelle den passenden Programmablaufplan für das rechts stehende Struktogramm.



### 3 Arbeiten im Binärsystem

Wie gesagt: Rechner kommt von Rechnen. Also wie geht das im Binärsystem? Eigentlich genau wie im Dezimalsystem. Fangen wir mit der Addition an.

#### 3.1 Addieren im Binärsystem

Üben wir doch erst einmal das schriftliche Addieren im Dezimalsystem.

**A 3.1.** Addiere die folgenden Zahlen schriftlich.

	$\begin{array}{r} 483 \\ + 294 \\ \hline \end{array}$	$\begin{array}{r} 889 \\ + 233 \\ \hline \end{array}$	$\begin{array}{r} 201 \\ + 572 \\ \hline \end{array}$
Übertrag			

**A 3.2.** Beschreibe in Worten, wie man zwei ganze dezimale Zahlen addiert. Gehe davon aus, dass der Ausführende in der Lage ist im Zahlenraum bis 20 zu addieren.

Das schriftliche Addieren von Binärzahlen funktioniert im Prinzip genau so, wie bei den Dezimalzahlen. Allerdings muss ein Übertrag schon gemacht werden, wenn die Ziffer 1 überschritten wird und nicht erst beim Überschreiten der Ziffer 9.

Dezimalsystem	Binärsystem
$\begin{array}{r} 103 \\ + 47 \\ \hline 150 \end{array}$	$\begin{array}{r} 01100111 \\ + 00101111 \\ \hline 10010110 \end{array}$

**A 3.3.** Berechne die folgenden Terme, indem Du die Dezimalzahlen ins Binärsystem überführst, dort addierst und dann wieder ins Dezimalsystem zurückführst. Überprüfe Dein Ergebnis, in dem Du die Additionen ebenfalls im Dezimalsystem durchführst.

- a)  $85_{10} + 23_{10}$       b)  $77_{10} + 88_{10}$       c)  $42_{10} + 63_{10}$       d)  $31_{10} + 31_{10} + 1_{10}$

- **A 3.4.** Erstelle einen Programmablaufplan und ein Struktogramm für das Addieren von zwei 8-Bit-Binärzahlen, deren Stellen in den Listen  $a[]$  und  $b[]$  gespeichert sind. Verwende eine dritte Liste  $u[]$  für den Übertrag und die Liste  $e[]$  für das Ergebnis.

### 3.2 AND, OR und XOR

Für Binärzahlen gibt es die Operatoren AND, OR und XOR. Dabei werden die Binärzahlen bitweise verarbeitet. Wenn zwei Binärzahlen mit AND verbunden werden, dann hat das Ergebnis an jeder Stelle eine 1, an der die erste Zahl **und** die zweite Zahl eine 1 hatten. Alle anderen Stellen sind 0.

$$\begin{array}{r} \phantom{\text{AND}} \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{0} \phantom{1} \\ \text{AND} \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{1} \phantom{1} \phantom{0} \phantom{0} \\ \hline \phantom{\text{AND}} \phantom{1} \phantom{0} \phantom{0} \phantom{0} \phantom{1} \phantom{0} \phantom{0} \phantom{0} \end{array}$$

Wenn zwei Binärzahlen mit OR verbunden werden, dann hat das Ergebnis an jeder Stelle eine 1, an der die erste Zahl **oder** die zweite Zahl eine 1 hatten. Nur wenn bei beiden Zahlen eine 0 stand, dann steht auch im Ergebnis eine 0.

$$\begin{array}{r} \phantom{\text{OR}} \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{1} \phantom{1} \phantom{0} \phantom{0} \\ \text{OR} \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{1} \phantom{1} \phantom{0} \phantom{0} \\ \hline \phantom{\text{OR}} \phantom{1} \phantom{1} \phantom{1} \phantom{0} \phantom{1} \phantom{1} \phantom{0} \phantom{1} \end{array}$$

Wenn zwei Binärzahlen mit XOR verbunden werden, dann hat das Ergebnis an jeder Stelle eine 1, an der **entweder** die erste Zahl **oder** die zweite Zahl eine 1 hatten. Wenn beide Zahlen den selben Stellenwert haben, dann steht im Ergebnis eine 0.

$$\begin{array}{r} \phantom{\text{XOR}} \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{1} \phantom{1} \phantom{0} \phantom{0} \\ \text{XOR} \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{1} \phantom{1} \phantom{0} \phantom{0} \\ \hline \phantom{\text{XOR}} \phantom{0} \phantom{1} \phantom{1} \phantom{0} \phantom{0} \phantom{1} \phantom{0} \phantom{1} \end{array}$$

Alle drei Operatoren sind kommutativ. Es gilt:

$$a \text{ AND } b = b \text{ AND } a \quad a \text{ OR } b = b \text{ OR } a \quad a \text{ XOR } b = b \text{ XOR } a$$

**A 3.5.** Berechne die folgenden Terme.

- a)  $11000011_2 \text{ AND } 00111100_2$       b)  $11100011_2 \text{ OR } 10111000_2$       c)  $11010011_2 \text{ XOR } 10111101_2$

**A 3.6.** Berechne die folgenden Terme, indem Du die Dezimalzahlen ins Binärsystem überführst, dort die angegebene Operation ausführst und dann wieder ins Dezimalsystem zurückführst.

- a)  $85_{10} \text{ AND } 23_{10}$       b)  $77_{10} \text{ OR } 88_{10}$       c)  $42_{10} \text{ XOR } 63_{10}$

**A 3.7.** Auf eine unbekannte Zahl  $x$  wird die Operation  $x \text{ AND } 64_{10}$  angewendet. Erläutere, welche Werte als Ergebnis herauskommen können.

**A 3.8.** Eine bestimmte Operation an Binärzahlen bezeichnet man als Kippen.

- a) Berechne  $11010011_2 \text{ XOR } 11111111_2$ .  
 b) Berechne  $00101010_2 \text{ XOR } 11111111_2$ .  
 c) Erläutere, warum man eine XOR-Operation mit einer Binärzahl, deren Stellen alle 1 sind, als Kippen bezeichnet.

**A 3.9.** Die Operation XOR hat eine weitere wichtige Aufgabe in der IT.

- a) Berechne  $(11010011_2 \text{ XOR } 00101010_2) \text{ XOR } 01001001_2$ .  
 b) Berechne  $(00101010_2 \text{ XOR } 01001001_2) \text{ XOR } 01001001_2$ .  
 c) Erläutere was passiert, wenn man zweimal XOR mit der gleichen Zahl hintereinander ausführt.  
 d) Gebe eine Anwendungsmöglichkeit für dieses Verfahren an.

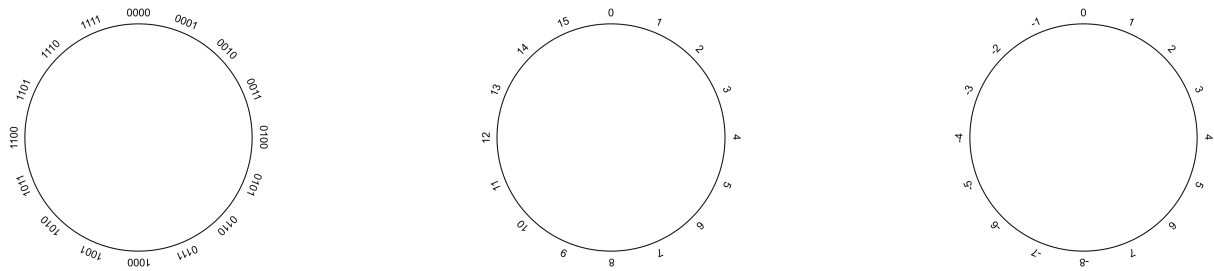


Abbildung 3.1: Zahlenkreis eines Nibbles. Links: Binärdarstellung; Mitte: Dezimaldarstellung der natürlichen Zahlen; Rechts: Dezimaldarstellung der ganzen Zahlen

### 3.3 Überlauf

In der Anwendung ist die Busbreite, also die maximale Anzahl der Stellen einer Binärzahl, begrenzt. Schauen wir uns doch mal eine vierstellige Binärzahl (Nibble) an. Mit einem Nibble können 16 verschiedene Zahlen dargestellt werden: 0, 1, 2, ..., 14, 15.

Was passiert also, wenn man fortlaufend den Wert des Nibbles um 1 erhöht.

$$0 \xrightarrow{+1} 1 \xrightarrow{+1} 2 \xrightarrow{+1} 3 \xrightarrow{+1} 4 \xrightarrow{+1} 5 \xrightarrow{+1} \dots \xrightarrow{+1} 13 \xrightarrow{+1} 14 \xrightarrow{+1} 15 \xrightarrow{+1} ???$$

Was passiert also, wenn wir zu  $1111_2$  eine  $1_2$  hinzuaddieren? Der Nibble geht auf  $0000_2$  und im Übertrag steht eine  $1_2$ . Da der Übertrag aber kein Bestandteil des Nibbles ist, ist das Ergebnis  $0000_2$ . Diesen Vorgang bezeichnet man als **Überlauf**. Dieser Fall tritt dann auf, wenn die Größe einer Zahl beschränkt ist, und ist eine der häufigsten Ursachen für Fehler in Programmen.

In der Mathematik ist eine Binärzahl in ihrer Stellenzahl natürlich nicht begrenzt. Dort ist der Zahlenstrahl unbegrenzt. Weil man bei einem Überlauf aber wieder von Vorne beginnt, ist das Modell des Zahlenstrahls für begrenzte Binärzahlen ungeeignet. Hier bietet sich das Modell des Zahlenkreises an. Dabei befindet sich die 0 oben. Um zur nächstgrößeren Zahl zu gehen, bewegen wir uns um 1 im Uhrzeigersinn (rechts rum) und wenn wir zu nächstkleineren Zahl gehen wollen, bewegen wir uns um 1 gegen den Uhrzeigersinn (links rum). Abbildung 3.1 zeigt den Zahlenkreis eines Nibbles.

### 3.4 Negative Zahlen

Wir haben bis jetzt die Binärzahlen für die Darstellung von natürlichen Zahlen verwendet. Wie müssen wir vorgehen, wenn wir ganze Zahlen darstellen wollen? Wie sieht eine negative Zahl in Binärdarstellung aus?

Schauen wir doch einfach mal auf den Zahlenkreis. Ganz oben steht die 0 ( $0000_2$ ). Wenn ich einen Schritt nach rechts gehe, dann bin ich bei der 1 ( $0001_2$ ). Ein Schritt von der 0 nach links führt mich zur -1. Beim Vergleich mit dem Zahlenkreis zeigt sich, dass im Nibble die -1 als  $1111_2$  dargestellt wird.

**A 3.10.** Die Tabelle rechts zeigt den Zusammenhang zwischen Dezimalzahl und Binärzahl mit Vorzeichen.

- a) Ergänze die Tabelle sinnvoll. Verwende dazu beide Zahlenkreise aus Abbildung 3.1.
- b) Erläutere, wie man eine negative Zahl erkennen kann.

Dezimal	Binär	Dezimal	Binär
-8	1000	0	0000
-7		1	0001
-6		2	
-5		3	
-4		4	
-3		5	
-2		6	
-1	1111	7	

Wir können also mit einem Nibble entweder die Zahlen von 0 bis 15 darstellen, wenn wir ohne Vorzeichen arbeiten. Mit Vorzeichen lassen sich mit einem Nibble die Zahlen von -8 bis 7 darstellen. Die negative Zahl erkennen wir am höchsten (linken) Bit. Ist dieses Bit auf 1 gesetzt, dann ist die Zahl negativ.

Als nächstes müssen wir überprüfen, ob diese Darstellung der negativen Zahlen auch mit der bereits definierten Addition zusammenpasst.



**A 3.15.** Dazu schauen wir uns eine besondere Eigenschaft des Zweierkomplements an.

- Denke Dir eine positive Byte-Binärzahl aus und bilde das Zweierkomplement. Bilde vom Ergebnis ebenfalls wieder das Zweierkomplement.
- Stelle eine Hypothese auf, zu welchem Ergebnis die doppelte Ausführung eines Zweierkomplements führt.
- Überprüfe an zwei weiteren Zahlen Deine Hypothese.

Jetzt ist es ganz einfach den Betrag einer negativen Binärzahl zu bestimmen. Wir bilden einfach das Zweierkomplement der negativen Binärzahl und erhalten damit den Betrag.

Beispiel: Aus  $1100\ 1000$  wird  
 das Einerkomplement  $0011\ 0111$  und es wird  
 1 dazuaddiert  $+ 0000\ 0001$   
 $= \overline{0011\ 1000} = 32 + 16 + 8 = 56$

**A 3.16.** Bestimme die dezimale Darstellung der folgenden Binärzahlen.

- $1111\ 0000_2$
- $1010\ 1010_2$
- $1100\ 1001_2$
- $1001\ 0110_2$

### 3.5 Subtrahieren im Binärsystem

Mit diesen neuen Informationen ist es jetzt leicht Binärzahlen voneinander zu subtrahieren. Zum Beispiel wollen wir die Zahl 21 von der Zahl 111 subtrahieren:  $111 - 21$ . Dazu nutzen wir aus, dass eine Subtraktion nichts anderes ist, als eine Addition mit der negativen Zahl des Subtrahenden. Also ist  $111 - 21 = 111 + (-21)$ . Wir bilden also die Binärdarstellung von 111 und -21 und addieren dann die beiden Binärzahlen mit dem bekannten Verfahren.

$$\begin{array}{r} 111 \\ + (-21) \\ \hline = 90 \end{array} \quad \begin{array}{r} 0110\ 1111 \\ + 1110\ 1011 \\ \quad 11\ 1111 \\ \hline = (1)0101\ 1010 \end{array}$$

**A 3.17.** Berechne die folgenden Aufgaben im Binärsystem und führe dann im Dezimalsystem die Probe durch.

- $56_{10} - 42_{10}$
- $127_{10} - 63_{10}$
- $88_{10} - 89_{10}$
- $63_{10} - 127_{10}$

## 4 Das oktale Zahlensystem

Ein weiteres gebräuchliches Zahlensystem in der Informatik ist das Oktalsystem, das acht Ziffern (0, 1, 2, 3, 4, 5, 6, 7) verwendet. Die Stellenwerte entsprechen daher analog zum Binär- und Dezimalsystem den Potenzen der Zahl 8:  $8^0 = 1$ ,  $8^1 = 8$ ,  $8^2 = 64$ ,  $8^3 = 256$ , usw.

Dezimalsystem					Oktalsystem			
T	H	Z	E	Stellenwert	512er	64er	8er	1er
$10^3$	$10^2$	$10^1$	$10^0$		$8^3$	$8^2$	$8^1$	$8^0$
1000	100	10	1		512	64	8	1
<b>2</b>	<b>2</b>	<b>5</b>	<b>7</b>		<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>
$2 \cdot 1000$ $+ 2 \cdot 100$ $+ 5 \cdot 10$ $+ 7 \cdot 1$ $= 2257$					$4 \cdot 512$ $+ 3 \cdot 64$ $+ 2 \cdot 8$ $+ 1 \cdot 1$ $= 2257$			

Damit ist die Umrechnung von Oktalzahlen in Dezimalzahlen definiert.

**A 4.1.** Wandle die folgenden Oktalzahlen in Dezimalzahlen um.

- a)  $3147_8$                       b)  $4711_8$                       c)  $7777_8$                       d)  $1234_8$

Bei der Umwandlung von Dezimalzahlen in Oktalzahlen verwenden wir analog die Verfahren, die wir schon bei den Binärzahlen kennengelernt haben.

### 4.1 Dezimalzahl in Oktalzahl: Von links nach rechts

- Wir beginnen beim höchsten Stellenwert der Oktalzahl.
- Wir dividieren die Oktalzahl mit dem Stellenwert und bestimmen den ganzzahligen Quotienten sowie den Rest.
- Wir setzen den ganzzahligen Quotienten an die Stelle der Oktalzahl. Der Rest ist die neue Oktalzahl.
- Wir nehmen die nächstkleinere Stelle und gehen zu Schritt 2, bis der Rest 0 ergibt.

Beispiel: Umwandlung der Zahl  $777_{10}$  in eine vierstellige Oktalzahl.

Vierter Stellenwert (512) ist einmal in 777 enthalten.  $\implies$  4. Stelle 1    Rest 265  
 Dritter Stellenwert (64) ist viermal in 265 enthalten  $\implies$  3. Stelle 4    Rest 9  
 Zweiter Stellenwert (8) ist einam in 9 enthalten  $\implies$  2. Stelle 1    Rest 1  
 Erster Stellenwert (1) ist einmal in 1 enthalten  $\implies$  1. Stelle 1    Rest 0

Also ist  $777_{10} = 1411_8$ .

**A 4.2.** Wandle die folgenden Dezimalzahlen mit der obigen Methode in Oktalzahlen um.

- a)  $2401_{10}$                       b)  $1052_{10}$                       c)  $373_{10}$                       d)  $424_{10}$

### 4.2 Dezimalzahl in Oktalzahl: von rechts nach links

- Beginne bei der kleinsten Stelle der Oktalzahl.
- Teile die Dezimalzahl ganzzahlig durch 8 und bestimme den Rest.
- Setzen die Stelle der Oktalzahl auf den Rest.
- Wechsle zur nächstgrößeren Stelle der Oktalzahl und arbeite mit der ganzzahlig geteilten Dezimalzahl in Schritt 2 weiter, bis die Dezimalzahl 0 ist.

Beispiel: Umwandlung der Zahl  $9999_{10}$  in eine Oktalzahl.

$$\begin{aligned}
 9999 \div 8 &= 1249 \text{ R } 7 \implies 1. \text{ Stelle } 7 \\
 1249 \div 8 &= 156 \text{ R } 1 \implies 2. \text{ Stelle } 1 \\
 156 \div 8 &= 19 \text{ R } 4 \implies 3. \text{ Stelle } 4 \\
 19 \div 8 &= 2 \text{ R } 3 \implies 4. \text{ Stelle } 3 \\
 2 \div 8 &= 0 \text{ R } 2 \implies 5. \text{ Stelle } 2
 \end{aligned}$$

Also ist  $9999_{10} = 23417_8$ .

**A 4.3.** Wandle die folgenden Dezimalzahlen mit der Divisionsmethode in Oktalzahlen um.

- a)  $3147_{10}$                       b)  $4711_{10}$                       c)  $7777_{10}$                       d)  $1234_{10}$

## 5 Das hexadezimale Zahlensystem

An dem Hexadezimalsystem, das 16 Ziffern (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F) verwendet, kommt man in der Informatik auf keinen Fall vorbei. Da wir nur 10 Ziffern aus dem Dezimalsystem haben, wurden hier einfach Buchstaben für die weiteren Ziffern benutzt. Für die Werte 10, 11, 12, 13, 14 und 15 verwendet man die Buchstaben A, B, C, D, E und F. Die Stellenwerte entsprechen den Potenzen der Zahl 16:  $16^0 = 1$ ,  $16^1 = 16$ ,  $16^2 = 256$ ,  $16^3 = 4096$ , usw.

Dezimalsystem					Hexaldezimalsystem			
T	H	Z	E	Stellenwert	4096er	256er	16er	1er
$10^3$	$10^2$	$10^1$	$10^0$		$16^3$	$16^2$	$16^1$	$8^0$
1000	100	10	1		4096	256	16	1
<b>8</b>	<b>3</b>	<b>6</b>	<b>7</b>		<b>2</b>	<b>0</b>	<b>A</b>	<b>F</b>
$8 \cdot 1000$ $+ 3 \cdot 100$ $+ 6 \cdot 10$ $+ 7 \cdot 1$ $= 8367$					$2 \cdot 4906$ $+ 0 \cdot 256$ $+ 10 \cdot 16$ $+ 15 \cdot 1$ $= 8367$			

Nach den ganzen Erfahrungen mit den bisherigen Stellenwertsystemen sollten die folgenden Aufgaben kein Problem sein.

**A 5.1.** Wandle die folgenden Hexadezimalzahlen in Dezimalzahlen um.

- a)  $3A4C_{16}$                       b)  $4711_{16}$                       c)  $D1EE_{16}$                       d)  $EBBE_{16}$

**A 5.2.** Wandle die folgenden Dezimalzahlen in Hexadezimalzahlen um.

- a)  $2401_{10}$                       b)  $1052_{10}$                       c)  $373_{10}$                       d)  $42424_{10}$

## 6 Vermischte Aufgaben

**A 6.1.** Um für das Datum möglichst wenige Ziffern zu benötigen ist Herr Phisigma auf die Idee gekommen, die Daten in einem Zahlensystem auf der Basis der Zahl 32 zu codieren. Ähnlich wie beim hexadezimalen Zahlensystem verwendet er Buchstaben für die großen Ziffern.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v

Tabelle 6.1: Übersicht über die Ziffern des 32er-Systems.

a) Wandle die folgenden 32er-Zahlen in Dezimalzahlen um.

(a)  $1a_{32}$ (b)  $111_{32}$ (c)  $abc_{32}$ (d)  $ole_{32}$ 

b) Wandle die folgenden Dezimalzahlen in 32er-Zahlen um.

(a)  $2401_{10}$ (b)  $1052_{10}$ (c)  $373_{10}$ (d)  $42424_{10}$ 

- c) Der Geburtstag von Carl Friedrich Gauß wird in Herrn Phisigmas System durch  $1nh4u$  dargestellt. Die letzte Ziffer steht für den Tag ( $u : 30$ ). Die vorletzte Ziffer steht für den Monat (4) und die übrigen Ziffern für das Jahr. Bestimme das Geburtsjahr von Gauß aus dieser Angabe.
- d) Carl Friedrich Gauß starb am 23. Februar 1855 in Göttingen. Bestimme die Darstellung dieses Datum nach dem System von Herrn Phisigma.

