GF Informatik: Zahlensysteme

Andreas Schärer & Moritz Küng

1M

2021

Inhaltsverzeichnis

LP Info

Links zum Umwandeln von Zahlensystemen:

- https://www.arndt-bruenner.de/mathe/scripts/Zahlensysteme.htm
- https://www.calculator.net/binary-calculator.html
- https://www.matheretter.de/rechner/zahlenkonverter

Links zu Info zu Gleitkommazahlen:

- https://www.elektronik-kompendium.de/sites/dig/1807231.htm
- https://de.wikipedia.org/wiki/IEEE_754
- Umrechnen: https://www.arndt-bruenner.de/mathe/scripts/Zahlensysteme.htm
- Visualisierung von float, berechnet jeweils Ungenauigkeit: https://www.h-schmidt.net/FloatConverter/IEEE754.html

1 Einführung

Wir sind es uns gewohnt, im Dezimalsystem, also dem 10er-System, zu arbeiten. Ein Computer hingegen arbeitet im Dualsystem (Binärsystem), also dem Zahlensystem, welches nur aus Nullen und Einsen besteht. Zum Beispiel hat die Dezimalzahl 37 im Binärsystem die Form 100101.

Doch warum rechnet ein Computer nur mit 0 und 1? Ein Computer besteht aus vielen elektronischen Bauteilen, in denen entweder Strom fliessen kann oder eben nicht. Die 1 steht dabei für 'es fliesst Strom' und die 0 für 'es fliesst kein Strom'. Diese beiden Zustände können mit einem **Bit** dargestellt werden. Ein Bit ist die kleinste mögliche Informationseinheit. Sie hat zwei Möglichkeiten: 0 oder 1. Die Binärzahl 100101 besteht also aus 6 Bits.

In diesem Dossier geht es darum, wie man Zahlen in Zahlensystemen darstellen und interpretieren kann. Betrachten wir die Zahl 100. Wahrscheinlich denkst du sofort 'Hundert'! Dies ist aber nur der Fall, wenn wir diese Zahl im uns bekanntesten Zahlensystem, dem Dezimalsystem, betrachten. Betrachtet man diese Zahl hingegen im Binärsystem, so hat es den Wert 4 im Dezimalsystem. Damit man eine Zahl sinnvoll interpretieren kann, muss man also immer wissen, in welchem Zahlensystem sie dargestellt wird.

Wir werden uns hier hauptsächlich mit dem Dualsystem beschäftigen und schauen, wie dort Grundoperationen wie das Addieren und Subtrahieren funktionieren.

Beispiel: Mit den Fingern zählen. Wie weit kann man mit einer Hand zählen? Mache für die ersten paar Beispiele eine Skizze?

1 (Daumen), 2 (Zeigefinger), 3 (Daumen + Zeigefinger), ...

Beachte: gibt entsprechende Grafik weiter hinten

insgesamt:

$$2^5 - 1 = 31$$

Wie weit kann man mit zwei Händen zählen?

$$2^{10} - 1 = 1023$$

2 Zahlensysteme

Definition 2.1

Ein **Zahlensystem** ist ein System, mit dem Zahlen dargestellt werden. Es wird durch seine **Basis** und seine **Nennwerte** festgelegt.

Beispiele für Zahlensysteme sind das uns sehr vertraute Dezimalsystem, das Binärsystem (Basis 2) oder das Hexadezimalsystem (Basis 16).

Im **Dezimalsystem** (auch **Zehnersystem**) ist die Basis 10 und die Nennwerte sind $0, 1, 2, 3, 4, 5, 6, 7, 8$ und 9. Die Zahl 1903_{10} ist dann wie folgt zu interpretieren:

$$1903_{10} = 1 \times 10^3 + 9 \times 10^2 + 0 \times 10^1 + 3 \times 10^0 \tag{1}$$

Mit der *kleinen Zahl unten rechts* (10) deuten wir an, dass die Zahl im Dezimalsystem zu betrachten ist. Lässt man diese Zahl weg, schreibt man also z.B. 576, so bedeutet dies (meistens), dass die Zahl im Dezimalsystem steht.

Aufgabe 2.1

a) Teile die Zahl $41\,086_{10}$ auf wie im Beispiel oben. Was sind ihre Nennwerte? Was ist die Basis?

b) Was ist die Basis im Dualsystem? Was sind die Nennwerte?

3 Binärsystem

"I told my friend 10 jokes about the binary system - He didn't get either of them!"

Definition 3.1

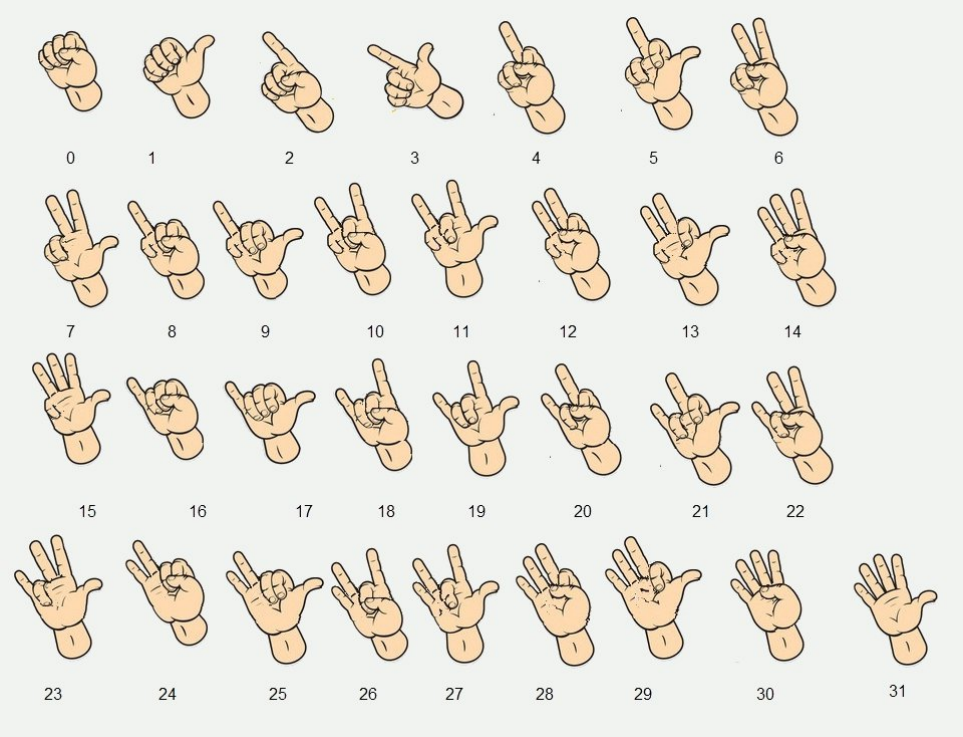
Die kleinste Informationseinheit ist das **Bit**, es hat zwei Möglichkeiten: es kann entweder 0 oder 1 sein. In der Welt der Elektrotechnik hat diese eine besondere Relevanz, da diese den beiden Zuständen 'es fliesst kein Strom (0)' oder 'es fliesst Strom (1)' entsprechen.

Deshalb ist da das **Binärsystem** (auch **Dualsystem** oder **Zweiersystem**) wichtig: Die Basis ist 2 und die Nennwerte sind 0 und 1. Eine Binärzahl besteht also aus mehreren Bits.

Das **Umrechnen einer Binärzahl in eine Dezimalzahl** geht ganz einfach. Für die Zahl 100101_2 geht man wie folgt vor:

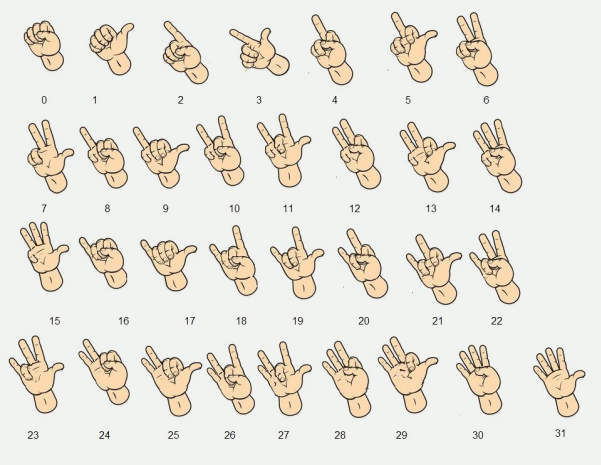
$$\begin{aligned} 100101_2 &= 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 32_{10} + 4_{10} + 1_{10} \\ &= 37_{10} \end{aligned} \tag{2}$$

Im Bild unten siehst du, wie man mit den Fingern einer Hand binär bis auf 31 zählen kann.



Aufgabe 3.1

Verwende die **Grafik** von der letzten Seite zum Lösen der Aufgabe:



a) Wandle die Binärzahl 10001_2 ins Dezimalsystem um. Tue dies zuerst mithilfe der Grafik oben. Überprüfe danach dein Resultat rechnerisch (wie im Beispiel auf der letzten Seite).

b) Wandle die Dezimalzahl 28_{10} mithilfe der Grafik oben ins Binärsystem um. Hier soll nichts gerechnet werden!

c) Warum ist die Binärzahl 10010_2 die Lieblingszahl von jedem Heavy Metal Fan?

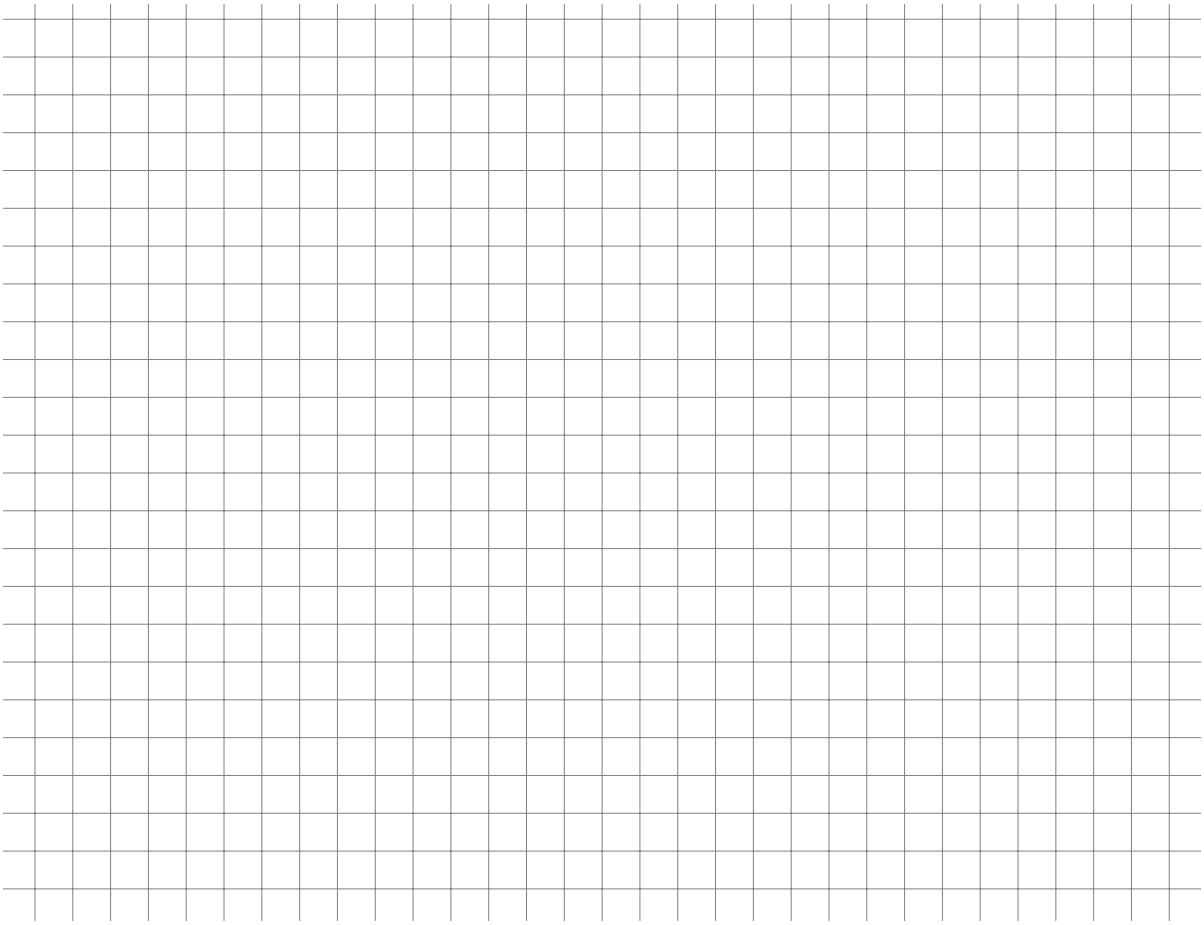
Aufgabe 3.2
Wandle die Binärzahlen ins Dezimalsystem um. Gehe dabei *rechnerisch* vor. Jeder Rechenschritt muss dabei klar ersichtlich sein.

a) 111_2

b) 1000011_2

c) 1101010_2

d) 100010001000_2



Aufgabe 3.3
Schreibe in **TigerJython** ein Programm, welches **Binärzahlen in Dezimalzahlen** umwandelt. Definiere im Programm eine Funktion `binaer_zu_dezimal(b)`, welche eine Binärzahl `b` als Argument entgegen nimmt, in eine Dezimalzahl umrechnet und mit return zurückgibt. Überprüfe deinen Code, indem du diesen auf die Beispiele aus der letzten Aufgabe anwendest.

Dazu einige Tipps:

- Schreibe die Binärzahl als String, z.B. `b = '100101'`.
- Ein String kann dann behandelt werden wie ein Array. Mit `b[2]` kann man den dritten Buchstaben des Strings `b` auslesen. Auch kann man genau gleich mit einer for-Schleife (`for x in b: ...`) durch alle Buchstaben des Strings durchgehen.

Aufgabe 3.4

Nimmt man 8 Bits zusammen, so erhält man ein **Byte**, zum Beispiel $1001\,1010_2$. Löse alle Aufgaben in dieser Aufgabe *rechnerisch*. Verwende deinen Code aus der letzten Aufgabe nur, um deine Resultate zu überprüfen.

a) Wieviele verschiedenen Zustände kann man mit einem einzigen Byte ausdrücken und warum?

b) Mit einem Byte sollen positive ganze Zahlen inklusive 0 ausgerückt werden. Was ist die grösste mögliche Zahl?

c) Es solle nun nicht nur 1 Byte, sondern x Bytes zur Verfügung stellen. Was ist nun die grösste mögliche Zahl, die man damit darstellen kann.

d) Wie viele Bytes benötigt man, um den Kontostand von Elon Musk speichern zu können? (Stand Anfangs 2021: 162 Mia.$)

.

Wir wissen nun, wie man Binärzahlen in Dezimalzahlen umwandelt. Ziel dieser Aufgabe ist herauszufinden, wie der umgekehrte Schritt funktioniert: Das **Umwandeln von Dezimalzahlen in Binärzahlen.**

Dazu gibt es einen *Algorithmus*, welcher wie folgt funktioniert: Starte mit der gegebenen Dezimalzahl und dividiere sie durch 2 und notiere das Resultat sowie den Rest. Wiederhole nun diesen Schritt, allerdings startest du nun mit dem Resultat des letzten Schritts. Dies wiederholst du solange, bis als Resultat 0 herauskommt. Die Binärzahl ist dann gegeben durch die Reste der Divisionen und zwar in umgekehrter Reihenfolge. Dieser Algorithmus wird der **Restwert-Algorithmus** genannt.

Schauen wir uns diesen Algorithmus für die Zahl 42_{10} an:

Schritt 1:	42	/	2	=	21	Rest:	0
Schritt 2:	21	/	2	=	10	Rest:	1
Schritt 3:	10	/	2	=	5	Rest:	0
Schritt 4:	5	/	2	=	2	Rest:	1
Schritt 5:	2	/	2	=	1	Rest:	0
Schritt 6:	1	/	2	=	5	Rest:	1

Das Resultat ist also

$$42_{10} = 101010_2$$

Damit man diese Rechnung auch etwas kürzer - aber doch übersichtlich - darstellen kann, bietet sich folgende Darstellung an:

42	
21	0
10	1
5	0
2	1
1	0
0	1

Aufgabe 3.5

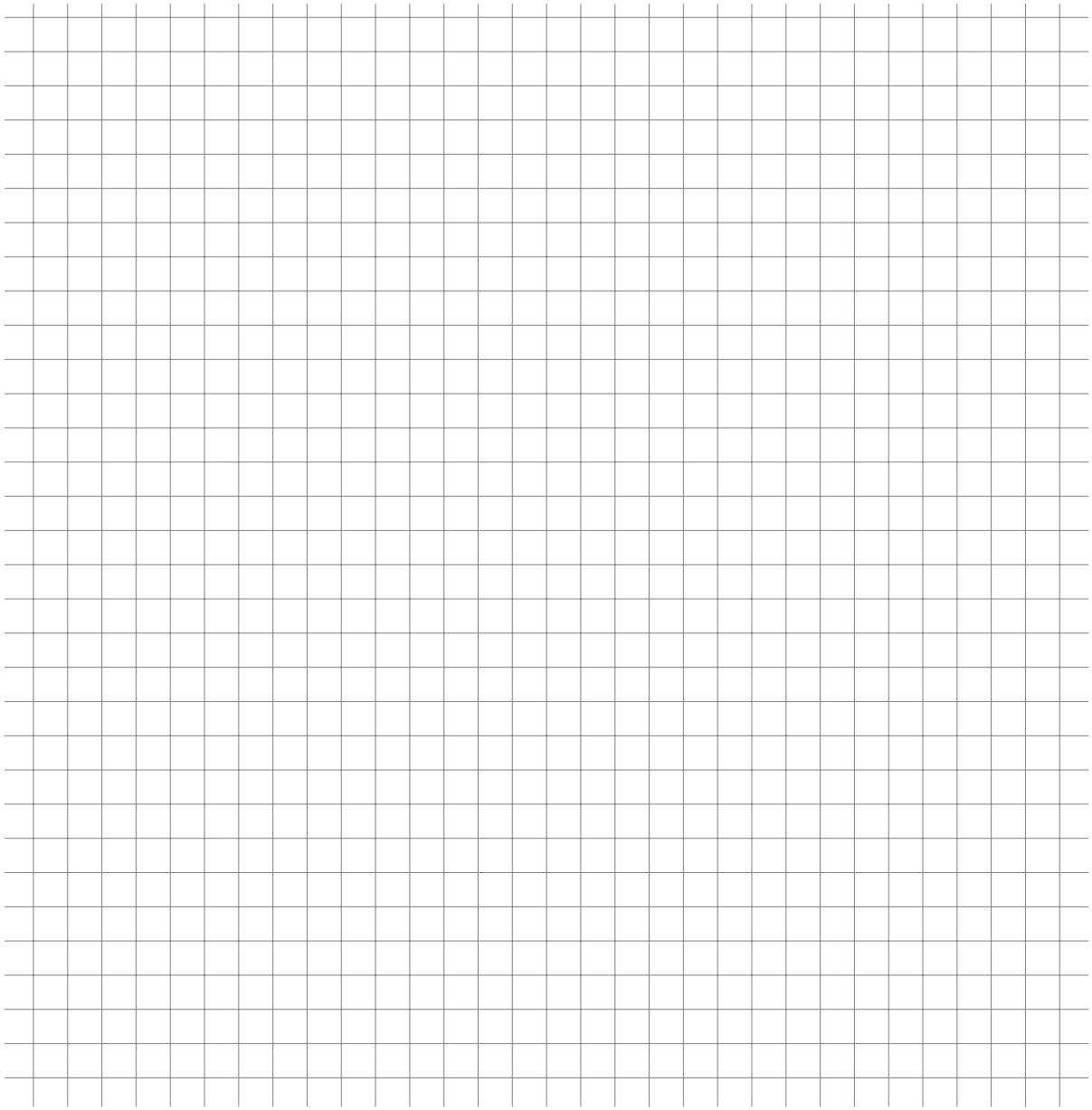
Wandle folgende Dezimalzahlen mit Hilfe des Restwert-Algorithmus in Binärzahlen um. Achte auf eine saubere und klare Darstellung.

a) 13_{10}

b) 19_{10}

c) 217_{10}

d) $56\,379_{10}$



Aufgabe 3.6

Implementiere den Restwert-Algorithmus in Python. Definiere im Programm eine Funktion `dezimal_zu_binaer(d)`, welche eine Dezimalzahl `d` als Argument entgegen nimmt, in eine Binärzahl umrechnet und mit return zurückgibt. Überprüfe deinen Code, indem du diesen auf die Beispiele aus der letzten Aufgabe anwendest.