1 Bits and Bytes: Wie funktioniert ein Computer?

1.0.1. Jegliche Information (Texte, Musik, Videos) wird in Computern als Folge von **Bits** (= binary digits = Binärziffern) gespeichert, also durch eine Folge von Nullen und Einsen. Der Grund dafür ist, dass sich zwei Zustände (AN/AUS = ON/OFF = Spannung vorhanden/nicht vorhanden) elektronisch leicht realisieren lassen. Ein **Byte** ist eine Folge von 8 Bits.

1.1 Stellenwertsysteme allgemein

1.1.1. Es gibt viele Möglichkeiten, eine natürliche Zahl zu notieren. Hier sind drei Schreibweisen derselben Zahl:

- steinzeitlich (auch unäre Notation genannt): ●●●●●●●●●●●●●●●●●●●●●●●●
- römisch: XXIV
- heutzutage, im Dezimalsystem = Zehnersystem: 24

Unterschiedliche Notationen sind unterschiedlich gut zum Rechnen geeignet, weshalb es geschichtlich durchaus einen Unterschied gemacht hat (Effizienz in Handel und Verwaltung), welche Zahlenschreibweisen verwendet wurden bzw. überhaupt bekannt waren.

Heutzutage schreiben wir Zahlen meistens im Dezimalsystem, das heisst im *Stellenwertsystem zur Basis 10*. Die *indisch-arabischen Ziffern* 0, 1, . . . , 9, die wir verwenden, kann man bereits um etwa 300 v. Chr. in Indien nachweisen. Fibonacci (= Leonardo da Pisa) lernte das Dezimalsystem in Algerien kennen und verbreitete es mit seinem Buch «Liber abaci» («Buch des Rechnens»[1], vollendet 1202) in Italien und Europa. Erst im 15. Jahrhundert setzte es sich im deutschen Sprachraum gegen die römische Zahlschreibweise durch.

Statt der Basis 10 kann man jede beliebige andere natürlich Zahl ≥ 2 als Basis nehmen.

Erklärung 1.1.2 Stellenwertsysteme, erste Beispiele

Stellenwertsysteme sind gewisse Notationssysteme für Zahlen, in denen Zahlen durch Folgen von Ziffern dargestellt werden. Die *Stelle* (= Position) einer jeden Ziffer entscheidet über ihren *Wert*. Die Stellen werden von rechts her durchnummeriert, wobei die Stelle ganz rechts die Nummer 0 bekommt. Als **Basis** eines Stellenwertsystems kann man jede natürliche Zahl $b \geq 2$ nehmen. Die erlaubten Ziffern sind dann Symbole für die Zahlen von Null bis $b-1$.
Beispiele:

- **Stellenwertsystem zur Basis 10 = Dezimalsystem = Zehnersystem = 10er-System**
  Die erlaubten Ziffern sind 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (beachte: $9 = 10-1$). Die Ziffer an der i-ten Stelle gibt an, wie oft die Zahl 10^i genommen wird.

Stelle 4	Stelle 3	Stelle 2	Stelle 1	Stelle 0
10000er= 10^4er	1000er= 10^3er	100er= 10^2er	10er= 10^1er	1er= 10^0er

nur Ziffern 0 bis 4 verwenden, dann selbe Ziffernfolge im 5er System

- **Stellenwertsystem zur Basis 5 = Fünfersystem = 5er-System**
  Die erlaubten Ziffern sind 0, 1, 2, 3, 4 (beachte: $4 = 5-1$). Die Ziffer an der i-ten Stelle gibt an, wie oft die Zahl 5^i genommen wird.

Stelle 4	Stelle 3	Stelle 2	Stelle 1	Stelle 0
625er= 5^4er	125er= 5^3er	25er= 5^2er	5er= 5^1er	1er= 5^0er

dieselbe Ziffernfolge

Die dargestellte Zahl ist die Summe der Produkte aus Ziffer und Wert der Stelle. ✎
Berechne im Zehnersystem:
$?\cdot 5^4 + ?\cdot 5^3 + ?\cdot 5^2 + ?\cdot 5^1 + ?\cdot 5^0 = ?\cdot 625 + \ldots$ (im Zehnersystem)

1.1.3. Man beachte, dass die Null eine enorm wichtige Funktion in Stellenwertsystemen hat. Beispielsweise sind die Zahlen 2024 und 224 verschieden. Die Entstehung der Null als Zeichen für nichts ist historisch interessant.

[1] Das Wort «abacus» bezog sich damals allgemein auf jede Rechenmethode und nicht nur auf den Abakus.

5er-System (als erstes Beispiel eines Stellenwertsystems)

⚒ **Aufgabe A1** Kindergarten im Fünferland (im Fünferland wird das Fünfersystem verwendet)

(a) Schreibe alle zweiunddreissig Zahlen von Null bis Einunddreissig im Fünfersystem auf (untereinander).

(b) Schreibe die aktuelle Jahreszahl im Fünfersystem.

⚒ **Aufgabe A2** Primarschule im Fünferland

Notiere in der Tabelle links das «Kleine Eins-plus-Eins» und in der Tabelle rechts das «Kleine Ein-mal-Eins» im Fünferland.

+	0	1	2	3	4	10
0						
1						
2						
3						
4						
10						

·	0	1	2	3	4	10
0						
1						
2						
3						
4						
10						

Bemerkung: Ob man hier die mit 10 benannten Zeilen und Spalten hinschreibt oder weglässt, ist Geschmackssache. In der Primarschule lässt man in der Regel die mit 0 benannten Zeilen und Spalten weg.

⚒ **Aufgabe A3** Schriftliches Addieren funktioniert im 5er-System «genauso» wie im Zehnersystem. Man muss eigentlich nur aufpassen, dass man niemals eine der Ziffern 5, 6, 7, 8, 9 verwendet. Wer mag, kann die obige Additionstabelle «Kleines Eins-plus-Eins im 5er-System» verwenden.

Addiere schriftlich im 5er-System (die Zahlen sind im 5er-System angegeben); die beiden Summanden sind zunächst rechtsbündig untereinanderzuschreiben. Kontrolliere deine Rechnung anschliessend im Dezimalsystem.

a) (ohne Übertrag) $1423 + 2011$ b) (mit Überträgen) $1443 + 243$

⚒ **Aufgabe A4** Schriftliches Multiplizieren funktioniert im 5er-System «genauso» wie im Zehnersystem. Wer mag, kann die obige Multiplikationstabelle «Kleines Ein-mal-Eins im 5er-System» verwenden.

Multipliziere schriftlich im 5er-System (die Zahlen sind im 5er-System angegeben). Kontrolliere deine Rechnung anschliessend im Dezimalsystem.

a) $1402 \cdot 3$ b) $432 \cdot 123$

Notation 1.1.4. Wenn man Zahlen in verschiedenen Stellenwertsystemen notiert, sollte man angeben, welches Stellenwertsystem wo verwendet wird. Dies geschieht häufig durch Angabe der *im Dezimalsystem geschriebenen* Basis des Stellenwertsystems als rechtem unterem Index. Zum Beispiel gilt

$$2010_{10} = 31020_5$$

Lässt man bei einer Ziffernfolge den Index weg, so handelt es sich in der Regel um die Darstellung einer Zahl im Zehnersystem; es kann aber auch sein, dass aus dem Kontext klar ist, in welchem Stellenwertsystem man arbeitet.

1.1.5. Wenn man eine Zahl im Zehnersystem schreibt, ist die Ziffer ganz rechts ihr Rest bei Division durch 10 und die verbleibenden Ziffern sind das Ergebnis der Ganzzahldivision. Zum Beispiel gilt

$2024 : 10 = 202$ Rest 4 oder gleichbedeutend $2024 = 202 \cdot 10 + 4$

Die analoge Aussge gilt im Fünfersystem. Beispielsweise gilt (Zahlen sind im Zehnersystem notiert) ✎

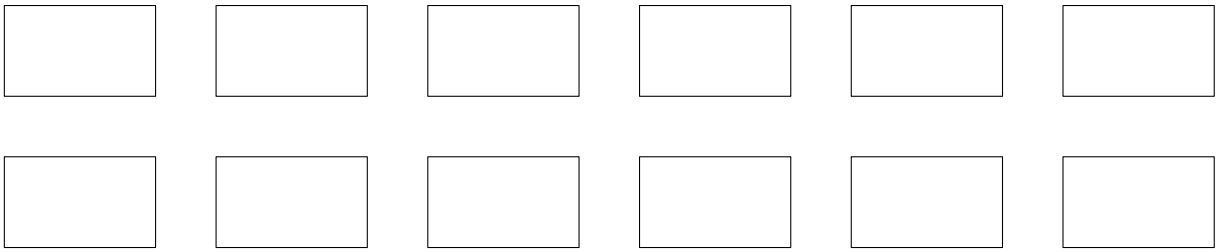
$2024 : 5 = 404$ Rest 4 oder gleichbedeutend $2024 = 404 \cdot 5 + 4$

Daraus folgt: Schreibt man $2024 = 2024_{10}$ im 5er-System, so ist 4 die Einerziffer.

Dies motiviert den folgenden Algorithmus: Iterierte Division mit Rest durch 5 liefert alle Ziffern im 5er-System.

Algorithmus 1.1.6 Divisionsalgorithmus: Umrechnen einer Zahl ins 5er-System (erklärt an einem Beispiel)

Schreibe die umzurechnende Zahl in die Box rechts oben (etwa die aktuelle Jahreszahl).
Dividiere diese Zahl (mit Rest) durch 5. Schreibe das Ergebnis (den sogenannten Ganzzahlquotienten) in die Box links daneben und den Rest in die Box darunter.
Dividiere dann die neue Zahl in der ersten Zeile durch 5. Schreibe das Ergebnis in die Box links daneben und den Rest in die Box darunter. Wiederhole dies, bis beim Dividieren Null als Ergebnis (nicht als Rest) herauskommt.



Als Ergebnis erhält man in der unteren Zeile die gesuchte Darstellung im 5er-System.

In unserem Beispiel gilt: ✎

$$\begin{aligned} 2026 &= 405 \cdot 5 + 1 & &= (81 \cdot 5 + 0) \cdot 5 + 1 \\ &= ((16 \cdot 5 + 1) \cdot 5 + 0) \cdot 5 + 1 & &= (((3 \cdot 5 + 1) \cdot 5 + 1) \cdot 5 + 0) \cdot 5 + 1 \\ &= 3 \cdot 5^4 + 1 \cdot 5^3 + 1 \cdot 5^2 + 0 \cdot 5 + 1 \end{aligned}$$

Dieses aussagekräftige Beispiel beweist im Wesentlichen, dass der Algorithmus korrekt ist.
Mit «demselben» Algorithmus kann man die Darstellung jeder natürlichen Zahl in jedem Stellenwertsystem ermitteln (Division mit Rest durch b, wenn b die Basis des betrachteten Stellenwertsystems ist).

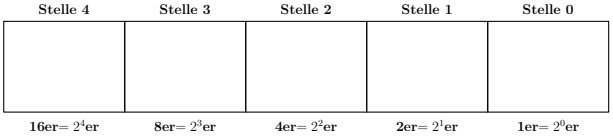
Aufgabe A5 Verwende das Divisionsverfahren (Algorithmus 1.1.6), um die folgenden Dezimalzahlen im 5er-System darzustellen.
Hinweis: Wer im Kopf durch 5 dividieren möchte: Division durch 5 ist dasselbe wie Multiplikation mit 2 und Division durch 10, denn $\frac{1}{5} = \frac{2}{10}$.

a) 194_{10} b) 678_{10} c) 2930_{10}

Binär- und Hexadezimalsystem: Die beiden wichtigsten Stellenwertsysteme in der Informatik

Erklärung 1.1.7 Stellenwertsystem zur Basis 2 = Binärsystem = Dualsystem = 2er-System

Binärsystem:
Die erlaubten Ziffern sind 0, 1. Die Ziffer an der i-ten Stelle gibt an, wie oft die Zahl 2^i genommen wird.



Schreibweise: Der besseren Lesbarkeit wegen werden längere Binärzahlen meist in Vierer-Gruppen notiert:

z. B. ✎

$$1976_{10} = 111'1011'1000_2$$

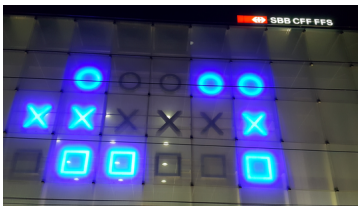
Grund dafür ist einerseits die übliche Gruppierung von 8 Bits in ein Byte, andererseits die einfache Umrechnung ins 16er-System (siehe unten).

Aufgabe A6 Umrechnen vom Binärsystem ins Zehnersystem.

Bei der St. Galler Bahnhofsuhr wird die aktuelle Uhrzeit zeilenweise (Stunden, Minuten, Sekunden) im Binärsystem angegeben. Licht an = Ziffer 1; Licht aus = Ziffer 0.

Die Form der Symbole (Kreis, Kreuz, Quadrat) dient nur der Unterscheidung von Stunden, Minuten und Sekunden.

Welche Uhrzeit wird jeweils angezeigt?

a) 

b) 

Aufgabe A7

(a) Kindergarten im 2erland: Zähle binär von 0 bis 33

(b) Primarschule im 2erland: Fülle die Eins-plus-Eins- und Einmal-Eins-Tabelle rechts aus. Als Zeilen- und Spaltenbeschriftungen sind die Zahlen 0 und 1 zu verwenden (die Zahl 10 lassen wir hier weg).

+		

·		

Aufgabe A8 Addiere schriftlich im Binärsystem. Die Zahlen sind untereinander zu schreiben. Kontrolliere deine Rechnung anschliessend im Dezimalsystem.

a) $10011 + 1111$ b) $11111 + 11111$

Aufgabe A9 Schriftliches Multiplizieren im Binärsystem ist relativ einfach, denn man muss jeweils entweder mit der Ziffer 1 oder der Ziffer 0 multiplizieren (am Ende muss man zugegebenermassen schriftlich addieren)). Multipliziere schriftlich und kontrolliere deine Rechnung anschliessend im Dezimalsystem.

a) $10011 \cdot 11001$ b) $11111 \cdot 11111$

Aufgabe A10 (Das Zahnradsymbol bedeutet «Bonus-Aufgabe»)

Schriftliches Dividieren (mit Rest oder mit Nachkommastellen) funktioniert in jedem Stellenwertsystem.

- Dividiere schriftlich im Binärsystem und kontrolliere dein Resultat im Dezimalsystem:
  a) $11110000 : 10$ b) $111100 : 101$ c) (mit Rest) $111111 : 101$
- Es gibt auch binäre Kommazahlen. Die Nachkommastellen im Binärsystem stehen für $2^{-1} = \frac{1}{2}$, $2^{-2} = \frac{1}{4}$, etc. Berechne $1011 : 101 = \frac{1011}{101}$ als Kommazahl.

Erklärung 1.1.8 Stellenwertsystem zur Basis 16 = Hexadezimalsystem = 16er-System

Hexadezimalsystem: Die erlaubten Ziffern sind 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Dabei stehen

«A» für zehn, «B» für elf, «C» für zwölf, «D» für dreizehn, «E» für vierzehn, «F» für fünfzehn.

Die Ziffer an der i-ten Stelle gibt an, wie oft die Zahl 16^i genommen wird.

Stelle 4	Stelle 3	Stelle 2	Stelle 1	Stelle 0
65536er= 16^4**er**	**4096er**= 16^3**er**	**256er**= 16^2**er**	**16er**= 16^1**er**	**1er**= 16^0**er**

✎obiges Beispiel ins Zehnersystem umrechnen

1.1.9. Der Leser überlege sich, warum man A schreibt statt 10, B statt 11 etc.

Aufgabe A11 Kindergarten im 16erland: Zähle hexadezimal von 0 bis 33.

Aufgabe A12 Stelle die folgenden (im Hexadezimalsystem angegebenen Zahlen) im Dezimalsystem dar.

a) FF_{16} b) AFFE_{16} c) $\mathrm{CAFE07}_{16}$

1.1.10. Das Divisionsverfahren (Algorithmus 1.1.6) funktioniert sinngemäss für jedes Stellenwertsystem. Beispielsweise muss man beim Umrechnen ins Binärsystem Division durch 2 und beim Umrechnen ins Hexadezimalsystem Division durch 16 verwenden.

Aufgabe A13 Wandle die folgenden Dezimalzahlen mit Hilfe des Divisionsverfahrens (Algorithmus 1.1.6) sowohl ins Binär- als auch ins Hexadezimalsystem um.

a) 2024_{10} b) 3386_{10}

Algorithmus 1.1.11 Umwandlung zwischen Binär- und Hexadezimalsystem und umgekehrt

Will man eine Binärzahl in eine Hexadezimalzahl umwandeln, so schreibt man ihre Ziffern in die Quadrate in der oberen Zeile der folgenden «Tabelle» (so, dass die Ziffer ganz rechts im Quadrat ganz rechts steht). Nun wandelt man jeden «Vierer-Block» von Binärziffern in die zugehörige (einstellige) Hexadezimalzahl um und schreibt diese in das Rechteck darunter. Dann steht in der unteren Zeile die gesuchte Hexadezimalzahl.

In unserem Beispiel gilt also

Die Umwandlung vom Hexadezimalsystem ins Binärsystem geht «umgekehrt»: Aus jeder Hexadezimalziffer wird ein Block von vier Binärziffern.
Dieses Verfahren funktioniert wegen $2^4 = 16$. Die folgende Rechnung erklärt dies in einem Beispiel:

$$\begin{aligned}
\mathrm{AC7}_{16} &= 10 \cdot 16^2 + 12 \cdot 16 + 7 \cdot 1 \\
&= (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2 + 0) \cdot (2^4)^2 + (1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2 + 0 \cdot 1) \cdot 2^4 + (0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2 + 1) \cdot 1 \\
&= 1010'1100'0111_2
\end{aligned}$$

Aufgabe A14 Verwende den Algorithmus 1.1.11, um die folgenden Zahlen vom Binärsystem ins Hexadezimalsystem bzw. vom Hexadezimalsystem ins Binärsystem umzuwandenln.

a) $1111\ 0000\ 0101_2$ b) $11\ 0011\ 1101_2$

c) FF_{16} d) $\mathrm{CAFE07}_{16}$

1.1.12. Computer rechnen mit Binärzahlen. Ein Nachteil von Binärzahlen ist, dass relativ kleine Zahlen bereits relativ viele Stellen benötigen; zum Beispiel hat die Dezimalzahl 2048 binär bereits zwölf Stellen,

$$2048_{10} = 1000'0000'0000_2$$

Hexadezimalzahlen benötigen hingegen nur etwa ein Viertel der Stellen, z. B. gilt

$$1000'0000'0000_2 = 800_{16}$$

Da die Umrechnung zwischen Binär- und Hexadezimalzahlen sehr einfach ist (und deutlich einfacher als die Umrechnung ins Dezimalsystem), werden Hexadezimalzahlen im informatischen Kontext oft verwendet (etwa bei der Kodierung von Farben, siehe später).

Bits und Bytes

Definition 1.1.13 Bit, Byte

- **Bit** = **b**inary digi**t** = Binärziffer, also 0 oder 1.
- **Byte** = Folge von 8 Bit = 8-stellige Binärzahl, z. B. 0100 1101

Im Kontext von Speichermedien, etwa Festplatten, meint man mit einem Byte meist die Möglichkeit, ein Byte zu speichern.

Aufgabe A15

(a) Wie viele verschieden Werte kann man in einem Byte speichern? Welchen Dezimalzahlen entsprechen diese achtstelligen Binärzahlen? Welchen Hexadezimalzahlen entsprechen diese achtstelligen Binärzahlen?

(b) Bei einer Binärzahl:

- Wie kann man sofort erkennen, ob sie gerade bzw. ungerade ist?
- Woran kann man sofort erkennen, ob sie durch 2 bzw. durch 4 bzw. durch 8 teilbar ist?
- Wie verdoppelt bzw. vervierfacht man eine Binärzahl?

(c) Kann man mit Hilfe der Quersumme ein Aussage über die Teilbarkeit einer Binärzahl treffen? (So wie die Quersumme einer Dezimalzahl genau dann durch 3 teilbar ist, wenn die Dezimalzahl durch 3 teilbar ist.)

Stellenwertsysteme in Python

Beispiel 1.1.14. Das folgende Programm wandelt eine beliebig vorgegebene natürliche Zahl `zahl` zwischen 0 und 4095 mit Hilfe des Divisionsalgorithmus 1.1.6 in eine Hexadezimalzahl um.

```
# beliebige (Dezimal-)Zahl zwischen 1 und 4095:
zahl = 2026

ziffernsymbole = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
x = zahl
ergebnis = ""

ziffer = x % 16
x = x // 16
ergebnis = ziffernsymbole[ziffer] + ergebnis
print(x, ziffer)

ziffer = x % 16
x = x // 16
ergebnis = ziffernsymbole[ziffer] + ergebnis
print(x, ziffer)

ziffer = x % 16
x = x // 16
ergebnis = ziffernsymbole[ziffer] + ergebnis
print(x, ziffer)

print(f'Die Dezimalzahl {zahl} ist {ergebnis} als Hexadezimalzahl.')
```

Aufgabe A16 Divisionsalgorithmus 1.1.6 (für beliebige Basis) in Python implementieren

(a) Verstehe das Programm in Beispiel 1.1.14.

(b) Schreibe eine Funktion `konvertiere16(zahl)` in Python, die eine (Dezimal-)Zahl `zahl` ins Hexadezimalsystem umwandelt und das Ergebnis als String zurückliefert.

Orientiere dich dabei an dem obigen Beispiel und verwende das folgende Programmgerüst.

```
ziffernsymbole = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"

def konvertiere16(zahl):
    if zahl == 0:
        return "0"
    x = zahl
    ergebnis = ""
    while # Abbruchbedingung zu ergänzen
        # Code zu ergänzen
    return ergebnis

# Tests:
print(konvertiere16(2026))    # 7EA
print(konvertiere16(4095))    # FFF
print(konvertiere16(45054))   # AFFE
```

(c) Warum mögen Vegetarier die Zahl $3'735'928'559_{10}$ im Hexadezimalsystem nicht?

(d) Ändere deine Funktion in eine Funktion `konvertiere(zahl, basis)` in Python, die eine Zahl `zahl` ins Stellenwertsystem zur Basis `basis` umwandelt und das Ergebnis als String zurückliefert. Teste deine Funktion mindestens mit den folgenden Aufrufen:

```
print(konvertiere(42, 2))      # 101010
print(konvertiere(3267, 7))    # 12345
```

Aufgabe A17

(a) Lies das untige Merke 1.1.15 und ermittle damit die Ausgabe des folgenden Programms.

```
d = 42
e = 0b1000011
f = 0b11_1111
g = 0x2A
h = 0xaffe
print(d, e, f, g, h, type(h))
```

(b) Prüfe deine Antwort am Computer.

Merke 1.1.15 Notation für Binär- und Hexadezimalzahlen in Python

In Python (und vielen anderen Programmiersprachen) können Zahlen direkt im Binär- bzw. im Hexadezimalsystem geschrieben werden (sowohl im Programmtext als auch bei der Eingabe). Die Kennzeichnung geschieht mit den Präfixen («Vorsilben»)

- `0b` für das Binärsystem und
- `0x` für das Hexadezimalsystem (das erste Zeichen ist jeweils eine Null, kein grosses O).

Als Gruppierungszeichen können Bodenstriche (= Unterstriche) verwendet werden.
Die Hexadezimalziffern A, B, C, D, E, F dürfen auch klein geschrieben werden.

Aufgabe A18

Lies Merke 1.1.16 und ermittle die Ausgabe des folgenden Programms zuerst ohne Computer, dann mit Computer.

```
a = hex(51)
b = hex(67)
c = bin(51)
```

```
d = bin(67)
print(a, b, c, d, type(d))
```

Merke 1.1.16

In Python gibt es die vordefinierten Funktionen

- `bin(n)` zum Umwandeln einer Zahl `n` in eine Binärzahl (Rückgabewert ist ein String, der mit dem Präfix `0b` beginnt);
- `hex(n)` zum Umwandeln einer Zahl `n` in eine Hexadezimalzahl (Rückgabewert ist ein String, der mit dem Präfix `0x` beginnt).

Aufgabe A19 Schreibe eine Funktion `dezimal_aus_16er(s)`, die aus einer als String `s` gegebenen Hexadezimalzahl die zugehörige (Dezimal-)Zahl berechnet oder allgemeiner eine Funktion `dezimal_aus(s, basis)`, die im Stellenwertsystem zur Basis `basis` (als String) gegebene Zahlen berechnet, d. h. als Dezimalzahl ausgibt.

1.1.17. Bonuswissen:

- Die Konvertierungsfunktion in ein Stellenwertsystem zu einer beliebigen Basis, die du in A16 geschrieben hast, gibt es in Python in der Bibliothek `numpy`:

  ```
  import numpy as np
  print(np.base_repr(42, 5)       # Ausgabe 132, Rückgabewert ist String
  ```

- Man kann auch Python beauftragen, eine Zeichenkette, die für eine Zahl in einem gewissen Stellenwertsystem steht, als Zahl aufzufassen (was du in Aufgabe A19 selbst programmiert hast).

  Beispielsweise «entschlüsselt» der Befehl `int('31100', 5)` die (als String im 5er-System gegebene) Zahl $(31100)_5 = (2025)_{10}$.

  Der Befehl `int('cafe', 16)` liefert dasselbe wie `0xcafe`.

- Zahlen im 8er-System = Oktalsystem kann man mit dem Präfix `0o` eingeben (etwa `0o71`). Die Funktion `oct(n)` wandelt eine Zahl `n` in das Oktalsystem um (etwa `oct(57)`).

1.2 Logischer Entwurf digitaler Systeme bzw. Einführung in die Digitaltechnik

1.2.1. Ziel dieses Abschnitt ist, zu verstehen, wie ein Computer zwei Binärzahlen addiert. Konkret bedeutet dies, dass wir dem Computer das schriftliche Addieren beibringen werden.

Als Grundbausteine werden wir die sogenannten «logischen Verknüpfungen» UND, ODER, NICHT verwenden, die sich relativ einfach als elektronische Schaltungen realisieren lassen.

Sobald die nötige Theorie verstanden ist, werden wir einen Binär-Addierer mit Hilfe einer geeigneten Simulations-Software für elektronische Schaltungen bauen.

Boolesche Algebra bzw. Logik: Rechnen mit Wahrheitswerten

1.2.2. In der «normalen» Algebra rechnet man mit Zahlen. In der **Booleschen Algebra** rechnet man mit den beiden Wahrheitswerten «wahr» und «falsch».

- Statt «wahr» schreibt man meist «1».
- Statt «falsch» schreibt man meist «0».

Der Name «Boolesche Algebra» geht auf den englischen Mathematiker George Boole (1815 – 1864) zurück. Statt von Wahrheitswerten spricht man auch von «booleschen Werten».

1.2.3. In der «normalen» Algebra verwendet man die Rechenzeichen + für die Addition, · für die Multiplikation und − als Vorzeichen.

In der booleschen Algebra verwendet man die Rechenzeichen $\vee$ für das **logische Oder** und $\wedge$ für das **logische Und** (Definition folgt unten). Ausserdem gibt es die **logische Verneinung**, die mit Hilfe eines «Strichs über dem zu verneinden Ausdruck» geschrieben wird.

Definition 1.2.4 Logische Verknüpfungen

Die Rechenoperationen **logisches Oder**, **logisches Und** und **logische Verneinung** sind wie folgt durch sogenannte **Wahrheitstafeln** (= **Wahrheitstabellen**) definiert.

- **logisches Oder** (Disjunktion), Rechenzeichen $\vee$:
  Sprechweise: $a \vee b$ wird als «a oder b» gelesen.
  Die zweite Zeile der Tabelle besagt beispielsweise:
  $0 \vee 1 = 1$ oder in Wahrheitswerten: «falsch oder wahr ist wahr».
  Merke: $a \vee b$ hat genau dann den Wert 1 (= wahr), wenn mindestens einer der beiden «Inputs» a und b den Wert 1 (= wahr) hat.

a	b	$a \vee b = a$ OR b
0	0	$0 = 0 \vee 0$
0	1	1
1	0	1
1	1	1

- **logisches Und** (Konjunktion), Rechenzeichen $\wedge$:
  Sprechweise: $a \wedge b$ wird als «a und b» gelesen.
  Die zweite Zeile der Tabelle besagt beispielsweise:
  $0 \wedge 1 = 0$ oder in Wahrheitswerten: «falsch und wahr ist falsch».
  Merke: $a \wedge b$ hat nur dann den Wert 1 (= wahr), wenn sowohl a als auch b den Wert 1 (= wahr) haben.

a	b	$a \wedge b = a$ AND b
0	0	$0 = 0 \vee 0$
0	1	0
1	0	0
1	1	1

- **logische Verneinung** (Negation), Rechenzeichen $\overline{\ }$ («Überstrich»):
  Sprechweise: $\overline{a}$ wird als «nicht a» gelesen.
  Merke: Die Verneinung von 1 (= wahr) ist 0 (= falsch) und umgekehrt.

a	$\overline{a} = \text{NOT}(a)$
0	1
1	0

1.2.5. Man könnte logisches Oder und logisches Und auch durch eine Verknüpfungstafel angeben, also in derselben Art, wie das «Kleine-Ein-mal-Eins» die Verknüpfungstafel für die Multiplikation der natürlichen Zahlen von 1 bis 10 ist.

⚒ Aufgabe A20

(a) Vervollständige die folgende Wahrheitstabelle! In jede der vier Ergebnisspalten sind also die richtigen Werte einzutragen. Beispiel: Der rote Eintrag ist das Ergebnis der Rechnung $\overline{0 \vee 1} = \overline{1} = 0$.

a	b	$\overline{a \vee b}$	$\overline{a} \vee \overline{b}$	$\overline{a \wedge b}$	$\overline{a} \wedge \overline{b}$
0	0				
0	1	0			
1	0				
1	1				

(b) Welche Rechengesetze zeigt die gerade ausgefüllte Tabelle? (Diese heissen de-Morgansche Gesetze.)

✎

de Morgansche Gesetze:
$$\overline{a \vee b} = \overline{a} \wedge \overline{b}$$
$$\overline{a \wedge b} = \overline{a} \vee \overline{b}$$

Beachte, dass es sich wirklich um einen mathematischen Beweis dieser Gesetze handelt, denn wir haben alle möglichen Fälle durchgespielt.

(c) Vervollständige die folgende Wahrheitstabelle!
Bemerkung: In einer Wahrheitstabelle sind links des senkrechten Doppelstrichs **alle** Belegungen der Variablen anzugeben. Meist sind diese Belegungen «aufsteigend als Binärzahlen» aufgeschreiben.

a	b	c	$a \wedge (b \vee c)$	$(a \wedge b) \vee (a \wedge c)$	$a \vee (b \wedge c)$	$(a \vee b) \wedge (a \vee c)$
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

(d) Welche Rechengesetze zeigt die gerade ausgefüllte Tabelle? ✎

Distributivgesetze:
$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$$
$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$$

Vergleiche: In der Arithmetik gibt es nur das Distributivgesetz (Verteilungsgesetz) für die Multiplikation über die Addition:

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

aber nicht für die Addition über die Multiplikation:

$$a + (b \cdot c) \neq (a + b) \cdot (a + c)$$

(e) Überlege dir, dass die folgenden beiden Assoziativgesetze für alle Belegungen der Variablen a, b und c gelten.
Hinweis: Mögliche Lösungswege:
- Wann genau werden die jeweiligen Ausdrücke 1 (= wahr)?
- Verwende eine Wahrheitstafel.

$$a \vee (b \vee c) = (a \vee b) \vee c$$
$$a \wedge (b \wedge c) = (a \wedge b) \wedge c$$

(f) Klammern sind wichtig: Überlege dir, dass die folgenden beiden Ausdrücke **nicht** für alle Belegungen der Variablen a, b und c das gleiche Ergebnis liefern. Hinweis: Ein «Gegenbeispiel» genügt.
- $(a \vee b) \wedge c$
- $a \vee (b \wedge c)$

Aufgabe A21 Finde für jede der «Ergebnis-Spalten» der folgenden Wahrheitstabelle einen logischen Ausdruck, der die angegebenen Werte liefert. Beispiel: Für die erste Ergebnis-Spalte ist eine Lösung bereits in rot eingetragen (auch $\overline{\overline{a} \vee b}$ wäre eine Lösung).

Ein logischer Ausdruck ist beispielsweise $\overline{a} \wedge (\overline{b \vee a})$. Die gesuchten logischen Ausdrücke sollen neben den Variablen a und b nur die drei logischen Verknüpfungen «Und», «Oder» und «Nicht» enthalten.

a	b	$a \wedge \overline{b}$			
0	0	0	0	1	0
0	1	0	1	0	1
1	0	1	0	0	1
1	1	0	0	0	0

Bemerkung: Die letzte Spalte ist die Wahrheitstabelle des **logischen Entweder-Oders** = **Exklusiv-Oders** = **exclusive Or** = XOR: Sind a und b Wahrheitswerte, so ist a XOR b genau dann 1 (= wahr), wenn genau einer der beiden Inputs 1 (= wahr) ist (aber nicht beide).

Definition 1.2.6 XOR = eXclusive OR = exklusives Oder = logisches Entweder-Oder

XOR: Das logische XOR (Entweder-Oder) ist definiert durch die rechts angegeben Wahrheitstafel.

Merke: a XOR b hat genau dann den Wert 1 (= wahr), wenn genau einer der beiden «Inputs» a, b den Wert 1 (= wahr) hat.

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

1.2.7. Video zeigen, wie logisches Und (= Konjunktion), Oder (= Disjunktion) und Nicht (= Negation) elektronisch realisiert werden können: Sebastian Lague: Exploring How Computers Work, https://www.youtube.com/watch?v=QZwneRb-zqA bis 6:16.

Disjunktive Normalform (DNF)

Definition 1.2.8

Eine **Wahrheitswertefunktion** oder **Boolesche Funktion** ist eine Funktion, deren Inputs endliche viele Wahrheitswerte sind und die als Output einen Wahrheitswert liefert.

1.2.9. Jede Wahrheitstafel stellt in offensichtlicher Weise eine boolesche Funktion dar.
Umgekehrt ist eine boolesche Funktion eindeutig durch ihre Wahrheitstafel gegeben.
Jeder logische (= boolesche) Term (etwa $a \wedge (b \vee \overline{a})$ stellt eine boolesche Funktion dar.

Satz 1.2.10 Disjunktive Normalform für boolesche Funktionen

Jede boolesche Funktion kann durch einen logischen (= booleschen) Term beschrieben werden, ja sogar durch einen logischen Term in **disjunktiver Normalform** (**DNF**), d. h. durch eine «Ver-Oder-ung (= Disjunktion) von Ver-Und-ungen der Inputs bzw. deren Verneinungen».

Folgerung: Insbesondere ist jeder boolesche Term zu einem Term in disjunktiver Normalform äquivalent.

Beweis. Wir erklären das allgemeine Verfahren «Bilden der **disjunktiven Normalform**» an einem aussagekräftigen Beispiel (mit drei Inputs).

a	b	c	
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Der folgende logische Ausdruck löst also unser Problem:

□

Schaltungen bauen mit Logik-Simulations-Software (etwa Logisim)

1.2.11. Installiere Logisim über den folgenden Link (vermutlich wirst du ausserdem Java installieren müssen - während der Logisim-Installation wirst du hoffentlich auf die entsprechende Java-Installations-Webseite geleitet): https://sourceforge.net/projects/circuit/

1.2.12. Demonstration in der Lektion:

- erste Erklärungen zur Bedienung von Logisim
- Bau des gerade konstruierten logischen Ausdrucks als logische Schaltung in Logisim.
- Test der Schaltung, indem man alle möglichen Variablenbelegungen eingibt.
- Wie man in Logisim die Wahrheitstafel zu einer Schaltung anzeigt.
- Eventuell bereits hier erklären, wie man eine solche Schaltung zu einem neuen Bauteil macht.

⚒ **Aufgabe A22** Betrachte die Wahrheitswertefunktion $f = f(a, b, c)$ mit drei Inputs a, b und c, die genau dann 1 liefert, wenn genau zwei der drei Inputs 1 sind.

(a) Schreibe die zugehörige Wahrheitstafel auf.

(b) Finde mit Hilfe des im Beweis von Satz 1.2.10 erklärten Verfahrens einen booleschen Ausdruck in disjunktiver Normalform für die Funktion $f = f(a, b, c)$.

(c) Baue die entsprechende Schaltung mit Logisim.

(d) Stimmt die von Logisim berechnete Wahrheitstafel?

Aufgabe A23 Baue mit Logisim das Bauteil «XOR» (= exklusiver Oder = Entweder-Oder). Es hat zwei Inputs x und y und einen Output. Der Output wird genau dann 1, wenn genau einer der Inputs 1 ist.

Empfohlenes Vorgehen:

- Erstelle zunächst die gewünschte Wahrheitstabelle (zwei Spalten für die Inputs, eine Output-Spalte).
- Finde einen geeigneten logischen Ausdruck für den Output (etwa per disjunktiver Normalform).
- Realisiere das Bauteil in Logisim und nenne es XOR.
- Teste dein Bauteil (von Hand oder per automatisch berechneter Wahrheitstafel).

Aufgabe A24 Halbaddierer: Addition zweier Bits = einstelliger Binärzahlen

Baue mit Logisim das Bauteil «HA» = «Halbaddierer»; was dieses Bauteil macht, wird sogleich erklärt; du wirst dieses Bauteil in den nachfolgenden Aufgaben benötigen.

Erklärung: Ein «Halbaddierer» hat zwei Inputs x und y und zwei Outputs s und c.

Wenn man die beiden Inputs x und y als einstellige Binärzahlen interpretiert, so sollen die beiden Output-Bits c und s nebeneinandergeschrieben die Summe $x + y$ darstellen. Beispielsweise soll bei den Inputs $x = 1$ und $y = 1$ als Output $c = 1$ und $s = 0$ herauskommen, denn binär gilt $1 + 1 = 10$. Allgemein soll also die folgende «binäre Gleichung»

$$x + y = cs \qquad \text{bzw. gleichbedeutend} \qquad \begin{array}{r} x \\ +\ y \\ \hline cs \end{array}$$

gelten, wobei cs als zweistellige Binärzahl aufzufassen ist.

Bemerkung: s steht für *sum*=Summe, c für carry (bit)=Übertrag.

Empfohlenes Vorgehen:

- Erstelle zunächst die gewünschte Wahrheitstabelle (zwei Spalten für die Inputs, zwei Spalten für die Outputs).
- Finde geeignete logische Ausdrücke für die beiden Outputs.
- Realisiere das Bauteil in Logisim und beschrifte die beiden Outputs mit «s» und «c» oder «sum» und «carry».
- Teste das Bauteil.

Aufgabe A25 Volladdierer: Addition dreier Bits = einstelliger Binärzahlen

Baue mit Logisim das Bauteil «VA» = «Volladdierer». Es hat drei Eingänge/Inputs x, y, z und zwei Ausgänge s und c.

Wenn man die drei Inputs x, y, z als einstellige Binärzahlen interpretiert, so sollen die beiden Output-Bits c und s nebeneinandergeschrieben die Summe $x + y + z$ darstellen. Beispielsweise soll bei den Inputs $x = 1$ und $y = 1$ und $z = 1$ als Output $c = 1$ und $s = 1$ herauskommen, denn binär gilt $1 + 1 + 1 = 11$. Allgemein soll also gelten:

$$x + y + z = cs \qquad \text{bzw. gleichbedeutend} \qquad \begin{array}{r} x \\ +\ y \\ +\ z \\ \hline cs \end{array}$$

Empfohlenes Vorgehen:

- Erstelle zunächst die gewünschte Wahrheitstabelle (drei Input-Spalten, zwei Output-Spalten).
- Realisiere das Bauteil in Logisim.
  Im folgenden werden drei sinnvolle Lösungswege angedeutet. Du solltest mindestens zwei dieser Lösungswege verstehen und die zugehörigen Schaltungen in Logisim bauen.
  (1) Statt drei Bits auf einmal zu addieren, addiere man zuerst zwei Bits (mit einem Halbaddierer) und addiere zum Resultat das dritte Bit (mit einem Halbaddierer und einem weiteren Bauteil).
  (2) Finde einen logischen Ausdruck für jeden Output durch folgende Überlegungen:
    – s wird genau dann 1, wenn «ungeradzahlig viele» der Inputs 1 sind (dies klingt nach XOR).
    – c wird genau dann 1, wenn mindestens zwei der Inputs 1 sind.
  (3) Finde einen logischen Ausdruck für jeden Output per disjunktiver Normalform.

1.2.13. Wenn man an einen Voll-Addierer an einen der drei Eingänge konstant den Wert 0 legt, erhält man einen Halbaddierer.

⚒ **Aufgabe A26** Baue nun einen 4-Bit-Addierer mit Logisim. Ein 4-Bit-Addierer addiert zwei 4-stellige Binärzahlen und hat

- 8 Inputs x_3, x_2, x_1, x_0 und y_3, y_2, y_1, y_0
- 5 Outputs s_4, s_3, s_2, s_1, s_0

Der Zusammenhang zwischen Inputs und Outputs ist durch die folgende «binäre Gleichung» gegeben

$$x_3x_2x_1x_0 + y_3y_2y_1y_0 = s_4s_3s_2s_1s_0 \qquad \text{bzw. gleichbedeutend} \qquad \begin{array}{rr} & x_3\,x_2\,x_1\,x_0 \\ + & y_3\,y_2\,y_1\,y_0 \\ \hline & s_4\,s_3\,s_2\,s_1\,s_0 \end{array}$$

wobei der «x-Input» ebenso wie der «y-Input» als 4-stellige Binärzahl und der «s-Output» als 5-stellige Binärzahl aufzufassen sind.

Hinweis: In der schriftlichen Addition sind ein Halbaddierer und drei Volladdierer «versteckt».

Bemerkung: Wer mag, kann die zwei addierten Zahlen und ihre Summe mit Hilfe einer Hexadezimalanzeige anzeigen lassen. Diese findet man in Logisim unter dem Menüpunkt «Input/Output». Zusätzlich benötigt man dafür eine Splitter, den man unter dem Menüpunkt «Wiring» findet; beim Splitter muss man noch «Fan Out» und «Bit Width In» auf 4 setzen.

1.3 Lösungen

Hinweise zu den Symbolen:

⚒ Diese Aufgaben könnten (mit kleinen Anpassungen) an einer Prüfung vorkommen. Für die Prüfungsvorbereitung gilt: "If you want to nail it, you'll need it".

✸ Diese Aufgaben sind wichtig, um das Verständnis des Prüfungsstoffs zu vertiefen. Die Aufgaben sind in der Form aber eher nicht geeignet für eine Prüfung (zu grosser Umfang, nötige «Tricks», zu offene Aufgabenstellung, etc.). **Teile solcher Aufgaben können aber durchaus in einer Prüfung vorkommen!**.

Diese Aufgaben sind dazu da, über den Tellerrand hinaus zu schauen und/oder die Theorie in einen grösseren Kontext zu stellen.

⚒ Lösung zu A1 ex-kindergarten

(a) 0, 1, 2, 3, 4, 10, 11, 12, 13, 14, 20, 21, 22, 23, 24, 30, 31, 32, 33, 34, 40, 41, 42, 43, 44, 100, 101, 102, 103, 104, 110, 111

(b) $2025_{10} = 31100_5$

⚒ Lösung zu A2 ex-primarschule-5erland

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	10
2	2	3	4	10	11
3	3	4	10	11	12
4	4	10	11	12	13

·					
	0	0	0	0	0
	0	1	2	3	4
	0	2	4	11	13
	0	3	11	14	22
	0	4	13	22	31

⚒ Lösung zu A3 ex-schriftlich-addieren-5er

a)

	5er	*10er*
	1423	238
+	2011	256
	3434	494

b)

	5er	*10er*
	1443	248
+	243	73
	2241	321

⚒ Lösung zu A4 ex-schriftlich-multiplizieren-5er

⚒ Lösung zu A5 ex-umrechnen

⚒ Lösung zu A6 ex-bahnhofsuhr-ablesen

⚒ Lösung zu A7 ex-kindergarten-2

⚒ Lösung zu A8 ex-schriftlich-addieren

⚒ Lösung zu A9 ex-schriftlich-multiplizieren

Lösung zu A10 ex-schriftlich-dividieren

Lösung zu A11 ex-kindergarten-16

Lösung zu A12 ex-umrechnen-hexadezimal-dezimal

Lösung zu A13 ex-umrechnen-nach-binaer-und-hexadezimal

Lösung zu A14 ex-binaer-hexadezimal

a) $F05_{16}$
b) $33D_{16}$
c) $1111\ 1111_2$
d) $1100\ 1010\ 1111\ 1110\ 0000\ 0111_2$

Lösung zu A15 ex-byte-zahlentheorie

(a) $2^8 = 256$ Werte
Diese Werte sind alle Binärzahlen von 0 = 0000'0000 bis 1111'1111, also alle Dezimalzahlen von 0 bis 255, also alle Hexadezimalzahlen von 0 = 00 bis FF.

(b)
- gerade = letzte Ziffer 0; ungerade = letzte Ziffer 1;
- teilbar durch 2: letzte Ziffer 0; teilbar durch 4: letzte beiden Ziffern 0; teilbar durch 8: letzte drei Ziffern 0.
- Verdoppeln: eine Null rechts dranhängen; Vervierfache = zweimal Verdoppeln: zwei Nullen rechts dranhängen.

(c) Vermutlich geht das nicht (ausser bei Teilbarkeit durch 1, je nachdem, ob die Quersumme 0 ist oder positiv): Die Quersumme ändert sich nicht, wenn man Nullen einfügt oder beseitigt. (Einen genauen Beweis müsste ich mir noch überlegen.)

Lösung zu A16 ex-umrechnen-python

(a) selbst
(b)
```
ziffernsymbole = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"

def konvertiere16(zahl):
    if zahl == 0:
        return "0"
    x = zahl
    ergebnis = ""
    while x > 0:
        ziffer = x % 16
        x = x // 16
        ergebnis = ziffernsymbole[ziffer] + ergebnis
    return ergebnis

# Tests:
print(konvertiere16(2026))   # 7EA
print(konvertiere16(4095))   # FFF
print(konvertiere16(45054))  # AFFE
```
(c) DEADBEAF (vgl. Lösung der nächsten Teilaufgabe)

(d)
```
ziffernsymbole = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"

def konvertiere(zahl, basis):
    if zahl == 0:
        return "0"
    x = zahl
    ergebnis = ""
    while x > 0:
        ziffer = x % basis
        x = x // basis
        ergebnis = ziffernsymbole[ziffer] + ergebnis
    return ergebnis

# Tests:
print(konvertiere(42, 2))            # 101010
print(konvertiere(3267, 7))          # 12345
print(konvertiere(3735928559, 16))   # DEADBEEF
```

Lösung zu A17 ex-notation-fuer-binaer-und-hexadezimalzahlen-in-python

```
42 67 63 42 45054 <class 'int'>
```

Lösung zu A18 ex-zahlkonvertierung-python

```
0x33 0x43 0b110011 0b1000011 <class 'str'>
```

Lösung zu A19 ex-ziffernfolge-in-stellenwertsystem-interpretieren

```
ziffernsymbole = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"

def dezimal_aus_16er(s):
    ergebnis = 0
    position = len(s)-1
    exponent = 0
    while position >= 0:
        zeichen = s[position]
        wert = ziffernsymbole.find(zeichen)
        ergebnis = ergebnis + wert*16**exponent
        position = position-1
        exponent = exponent+1
    return ergebnis

def dezimal_aus(s, basis):
    ergebnis = 0
    position = len(s)-1
    exponent = 0
    while position >= 0:
        zeichen = s[position]
        wert = ziffernsymbole.find(zeichen)
        ergebnis = ergebnis + wert*basis**exponent
        position = position-1
        exponent = exponent+1
    return ergebnis
```

```
print(dezimal_aus_16er('1'))
print(dezimal_aus_16er('10'))
print(dezimal_aus_16er('100'))
print(dezimal_aus_16er('FF'))

print(dezimal_aus('1', 5))
print(dezimal_aus('10', 5))
print(dezimal_aus('100', 5))
print(dezimal_aus('44', 5))
print(dezimal_aus('31100', 5))
```

Lösung zu A20 ex-wahrheitstabelle

(a)

a	b	$\overline{a \vee b}$	$\overline{a} \vee \overline{b}$	$\overline{a \wedge b}$	$\overline{a} \wedge \overline{b}$
0	0	1	1	1	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	0	0	0	0

(b) Die beiden **de-Morganschen Gesetze**

$$\overline{a \vee b} = \overline{a} \wedge \overline{b}$$
$$\overline{a \wedge b} = \overline{a} \vee \overline{b}$$

(c)

a	b	c	$a \wedge (b \vee c)$	$(a \wedge b) \vee (a \wedge c)$	$a \vee (b \wedge c)$	$(a \vee b) \wedge (a \vee c)$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	0	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

(d) Die beiden **Distributivgesetze**

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$$
$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$$

Man beachte, dass es hier im Gegensatz zur normalen Arithmetik zwei Distributivgesetze gibt (sowohl $\wedge$ verteilt «über» $\vee$ als auch andersherum). In der normalen Arithmetik gilt nur ein Assoziativgesetz $a \cdot (b + c) = a \cdot b + a \cdot c$ (Multiplikation verteilt «über» Addition), jedoch ist $a + (b \cdot c) = (a + b) \cdot (a + c)$ im Allgemeinen falsch.

(e) Die beiden Ausdrücke in der ersten Zeile werden genau dann wahr, wenn mindestens eine der drei Variablen 1 ist. Die beiden Ausdrücke in der zweiten Zeile werden genau dann wahr, wenn alle drei Variablen 1 sind.

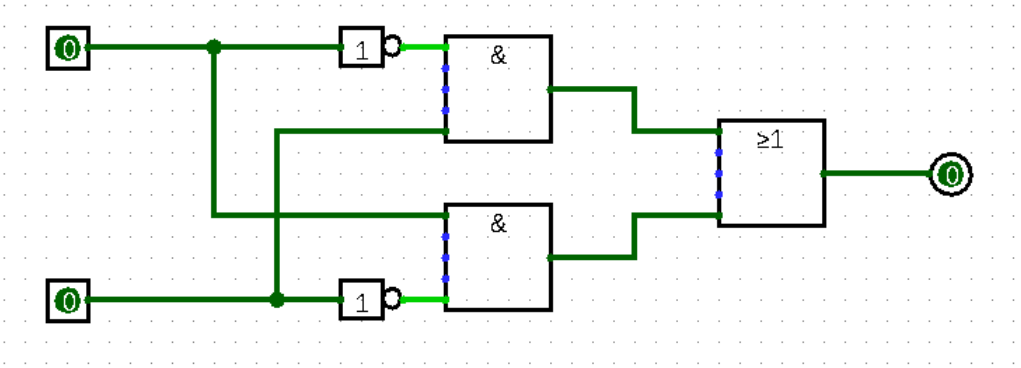
(f) Für $a = 1$, b beliebig und $c = 0$ ist der erste Ausdruck 0, der zweite aber 1.

Lösung zu A21 ex-formel-zu-wahrheitstabelle

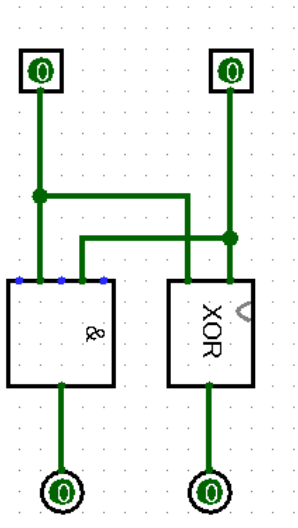
a	b	$a \wedge \overline{b}$	$\overline{a} \wedge b$	$\overline{a} \wedge \overline{b}$	$(\overline{a} \wedge b) \vee (a \wedge \overline{b})$
0	0	0	0	1	0
0	1	0	1	0	1
1	0	1	0	0	1
1	1	0	0	0	0

Lösung zu A22 ex-disjunktive-normalform

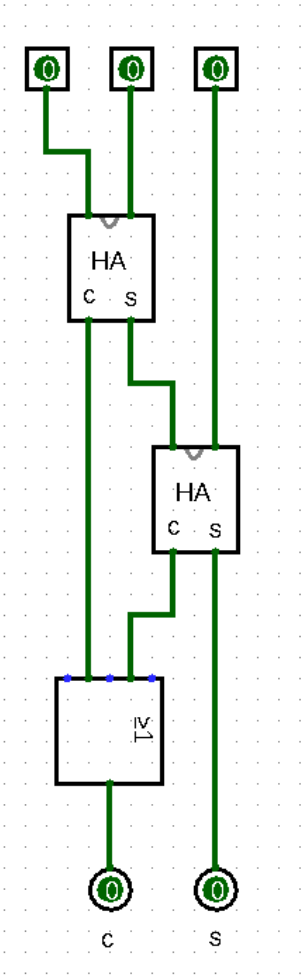
Lösung zu A23 ex-logisim-xor



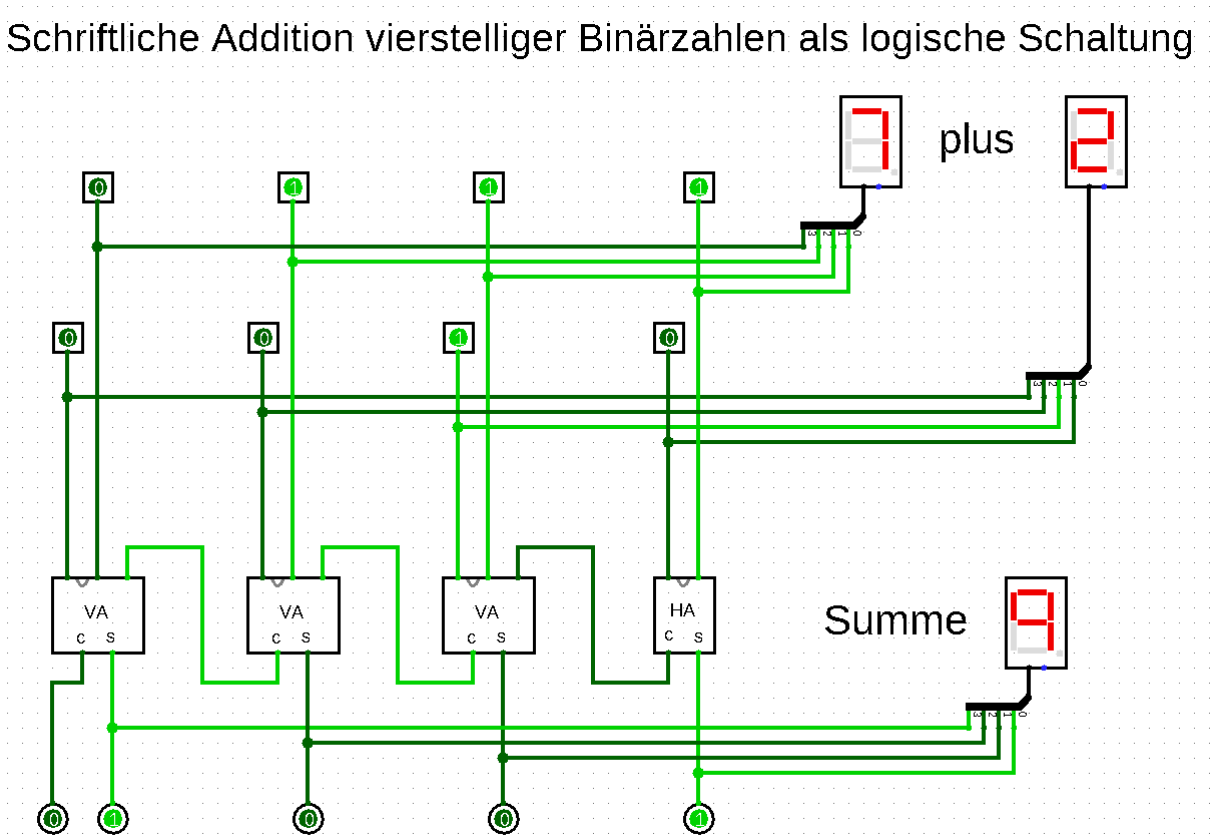
Lösung zu A24 ex-logisim-halbaddierer



Lösung zu A25 ex-logisim-volladdierer



⚒ Lösung zu A26 ex-logisim-addierer


Schriftliche Addition vierstelliger Binärzahlen als logische Schaltung
plus
VA
C S
VA
C S
VA
C S
HA
C S
Summe