1.2 Speicherung von Daten am Computer allgemein

1.2.1. Computer können nur Nullen und Einsen abspeichern. Alle Daten (Zahlen, Texte, Bilder, Musik, Videos, Programme, Computerspiele) müssen deshalb in geeignete Folgen von Nullen und Einsen umgewandelt werden.

Merke 1.2.2

Jede Datei bzw. auf dem Computer gespeicherte Information ist eine Folge von Nullen und Einsen, also eine Binärzahl!

1.3 Speicherung von Text

1.3.1. Jeder Text, der am Computer abgespeichert werden soll, muss in eine Folge von Nullen und Einsen (= eine Binärzahl) umgewandelt werden. Naheliegend ist die Idee, jedem Zeichen eine Zahl zuzuordnen und diese als Binärzahl abzuspeichern.

Allgemein nennt man eine Zuordnung von gewissen Zeichen (aus einem «Alphabet») zu gewissen anderen Dingen (etwa Zahlen) eine «Kodierung» oder kurz einen «Code».

Ein wichtiges Beispiel einer Kodierung ist der Morse-Code; bekannt ist vermutlich die Morse-Gruppe «drei kurz, drei lang, drei kurz» für «SOS». Beim Morsen wird jedem Buchstaben eine Folge langer und kurzer Töne zugeordnet; unterschiedliche lange Pausen dienen der Abgrenzung von Tönen, übermittelten Buchstaben und Wörtern.

ASCII-Code

1.3.2. Zur Speicherung von (englischsprachigen) Texten auf Computern hat sich der sogenannte ASCII-Code durchgesetzt (*American Standard Code for Information Interchange*; erste Version von 1963, aktuelle Version von 1968). Er ordnet jedem Zeichen eine 7-stellige Binärzahl zu. Die Codierung geht aus der folgenden vierspaltigen ASCII-Tabelle.

Quelle: https://github.com/stevenlinx/Four-Column-ASCII

00	01	10	11	
NUL	Spc	@	`	00000
SOH	!	A	a	00001
STX	"	B	b	00010
ETX	#	C	c	00011
EOT	$	D	d	00100
ENQ	%	E	e	00101
ACK	&	F	f	00110
BEL	'	G	g	00111
BS	(	H	h	01000
TAB	)	I	i	01001
LF	*	J	j	01010
VT	+	K	k	01011
FF	,	L	L	01100
CR	-	M	m	01101
SO	.	N	n	01110
SI	/	O	o	01111
DLE	0	P	p	10000
DC1	1	Q	q	10001
DC2	2	R	r	10010
DC3	3	S	s	10011
DC4	4	T	t	10100
NAK	5	U	u	10101
SYN	6	V	v	10110
ETB	7	W	w	10111
CAN	8	X	x	11000
EM	9	Y	y	11001
SUB	:	Z	z	11010
ESC	;	[	{	11011
FS	<	\	|	11100
GS	=	]	}	11101
RS	>	^	~	11110
US	?	_	DEL	11111

Vom Zeichen zum ASCII-Code: Das Zeichen F findet sich in der Spalte 10 und der Zeile 00110; hintereinandergeschrieben ergibt sich der zugehörige ASCII-Code 10 00110 $= (1000110)_2 = (46)_{16} = (70)_{10}$.

Der ASCII-Code eines Zeichens ist eine Zahl, die man in jedem beliebigen Stellenwertsystem angeben kann.

Vom ASCII-Code zum Zeichen: Gegeben ist der ASCII-Code $(43)_{10} = (2B)_{16}$ (als Dezimal- bzw. Hexadezimalzahl). Wandle diese Dezimalzahl in eine **siebenstellige** (mit führenden Nullen) Binärzahl um: $(43)_{10} = (2B)_{16} = (0101011)_2$. Ihre ersten beiden Stellen 01 liefern die Spalte und die hinteren fünf Stellen 01011 ihre Zeile in der Tabelle rechts: Das zugehörige Zeichen ist das Pluszeichen.

Steuerzeichen: Die Zeichen in der Zeile 00 sind sogenannte Steuerzeichen (control characters); beispielsweise steht LF für *line feed* (Zeilenvorschub; neue Zeile); CR steht für *carriage return* (Wagenrücklauf, etwa zum Steuern von Druckern). Auch DEL (ganz rechts unten) ist ein Steuerzeichen, es steht für *delete*.

Druckbare Zeichen: Die anderen 128 - 32 - 1 = 95 Zeichen sind druckbare Zeichen (Spc steht für *space* (Leerzeichen)).

ASCII ist (ursprünglich) eine 7-Bit-Zeichenkodierung: Der ASCII-Code ist eine 7-Bit-Zeichenkodierung, der $2^7 = 128$ Zeichen (davon 95 druckbar) codiert, die sogenannten ASCII-Zeichen. Da Byte (= 8 Bit) die übliche Speichereinheit ist, wird meist ein Byte zum Speichern eines mit ASCII kodierten Zeichens verwendet.
Vielleicht wurde bemerkt, dass im klassischen 7-Bit-ASCII einige Zeichen fehlen (etwa die Umlaute ä, ö, ü, das Euro-Zeichen). Einige dieser Zeichen wurden später in gewissen 8-Bit-Erweiterungen von ASCII aufgenommen.

⚒ Aufgabe A16

(a) Welche Zeichenfolge (die im Wesentlichen so auf der Festplatt eines Computers stehen könnte) codiert die folgende Folge von (8-stelligen) Binärzahlen (= Bytes)?

01000001 01101100 01101100 01100101 01110011 00100000 01101001 01110011 01110100
00100000 01000010 01101001 01101110 01100001 01100101 01110010 01111010 01100001
01101000 01101100 00100001

Hinweis: Die erste Ziffer jeder Binärzahl ist Null und kann ignoriert werden. Die verbleibende 7-stellige Binärzahl ist mit der ASCII-Tabelle zu dekodieren.

(b) Welche Zeichenfolge codiert die folgende Folge von (zweistelligen) Hexadezimalzahlen (= Bytes)?

47 75 74 20 67 65 6D 61 63 68 74 21

Hinweis: Jede zweistellige Hexadezimalzahl ist in eine 8-stellige Binärzahl umzuwandeln, welche dann per ASCII-Tabelle für ein Zeichen steht.

(c) (freiwillig) Wer mag, kann etwa in Visual Studio Code einen Hex-Editor (= Hexadezimal-Editor) als Erweiterung/Extension installieren und sich eine beliebige Datei oder die gerade entschlüsselten Texte damit anschauen (Rechtsklick auf Datei, «Open/Reopen Editor With ...»).

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded Text
00000000	57	69	65	20	73	70	65	69	63	68	65	72	74	20	65	69	W i e s p e i c h e r t e i
00000010	6E	20	43	6F	6D	70	75	74	65	72	20	64	69	65	73	65	n C o m p u t e r d i e s e
00000020	6E	20	54	65	78	74	20	61	6C	73	20	46	6F	6C	67	65	n T e x t a l s F o l g e
00000030	20	76	6F	6E	20	48	65	78	61	64	65	7A	69	6D	61	6C	v o n H e x a d e z i m a l
00000040	2D	5A	61	68	6C	65	6E	20	3D	20	42	79	74	65	73	3F	- Z a h l e n = B y t e s ?

Abbildung 1: Text-Datei im Hex-Editor; die Hexadezimalzahlen links entsprechen genau den Zeichen rechts. Die Tabelle links ist im Wesentlichen ein «Blick auf die Festplatte» wo die Hexadezimalzahlen als Binärzahlen stehen.

Speicherplatzbedarf

1.3.3 (Erinnerung: Bits and Bytes)**.**

- **Bit** = **b**inary digi**t** = Binärziffer, also 0 oder 1.
- **Byte** = Folge von 8 Bit = 8-stellige Binärzahl, z. B. 0100 1101 bzw. (im Kontext von Speichern, etwa Festplatten) die Möglichkeit, eine solche Zahl zu speichern.

Weil man an jeder der 8 Positionen zwei mögliche Ziffern hat, kann ein Byte $2^8 = 256$ verschiedene Werte annehmen, nämlich alle Binärzahlen von 00000000 bis 11111111.

1.3.4. Vermutlich ist bekannt, dass

- 1 Kilobyte, 1 kB, für $1'000 = 10^3$ Byte steht;
- 1 Megabyte, 1 MB, für $1'000'000 = 10^6$ Byte steht;
- 1 Gigabyte, 1 GB, für $1'000'000'000 = 10^9$ Byte steht;
- 1 Terabyte, 1 TB, für $1'000'000'000'000 = 10^{12}$ Byte steht.

Es gibt neben Dezimalpräfixen auch auch «Binärpräfixe». Beispiele: 1 Kibi-Byte = 1 KiB = 2^{10} = 1024 Byte; 1 Mebi-Byte = 1 MiB = $2^{20} = 1'048'576$ Byte.

⚒ Aufgabe A17

(a) Auf eine Seite DIN-A4-Papier passen ca. 2000 Text-Zeichen (in normaler Grösse). Wenn man jedes Zeichen per ASCII durch ein Byte codiert, wieviel Speicherplatz benötigt man, um den Inhalt von 500 Seiten Text abzuspeichern?

(b) Wie viele solche 500-seitigen Bücher (die nur aus Text bestehen) kann man auf einer handelsüblichen 400 Gigabyte-Festplatte abspeichern?

(c) Die grösste Biblothek der Welt ist die British Library mit ca. 14 Millionen Büchern (unter etwa 200 Millionen Medieneinheiten, laut englischer Wikipedia vom 17.01.2022). Wenn man vereinfachend annimmt, dass ein Buch durchschnittlich 500 Seiten hat und nur aus Text besteht, wie viele handelsübliche Laptops benötigt man in etwa, um diese 14 Millionen Bücher abzuspeichern?

1.4 Logischer Entwurf digitaler Systeme bzw. Einführung in die Digitaltechnik

1.4.1. 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

1.4.2. 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».

1.4.3. In der «normalen» Algebra verwendet man die Rechenzeichen $+$ für die Addition, $\cdot$ 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 sofort). Ausserdem gibt es die **logische Verneinung**, die mit Hilfe eines «Strichs über dem zu verneinden Ausdruck» geschrieben wird.

Definition 1.4.4 Logische Verknüpfungen

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

- **logisches Oder**, 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	1	
1	0	
1	1	

- **logisches Und**, 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	1	
1	0	
1	1	

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

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

⚒ Aufgabe A18

(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?

(c) Vervollständige die folgende Wahrheitstabelle! 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?

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

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

(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 A19

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