Binäre Interpretationen

Binäre Daten können auf verschiedenste Arten interpretiert werden.
Einige dieser Interpretationen müssen Sie kennen.

Zweierkomplement

(https://de.wikipedia.org/wiki/Zweierkomplement)

Bei der Codierung in der **Zweierkomplementdarstellung sind negative Zahlen** daran zu erkennen, dass das **höchstwertige Bit den Wert 1** hat.

Bei 0 liegt eine positive Zahl oder der Wert 0 vor.

Der Vorteil dieses Zahlenformates besteht darin, dass für Verarbeitung in digitalen Schaltungen keine zusätzlichen Steuerlogiken notwendig sind.

```
bei 8 Bit:                    −128(10)  bis                  +127(10)
bei 16 Bit:                 −32768(10)  bis                +32767(10)
bei 32 Bit:           −2147483648(10)  bis           +2147483647(10)
bei 64 Bit: −9223372036854775808(10)  bis +9223372036854775807(10)
```

Binärwert	Hex-Wert	Interpretation als Zweierkomplement	Interpretation als vorzeichenlose Zahl
00000000	00	0	0
00000001	01	1	1
...	...	...	...
01111110	7E	126	126
01111111	7F	127	127
10000000	80	−128	128
10000001	81	−127	129
10000010	82	−126	130
...	...	...	...
11111110	FE	−2	254
11111111	FF	−1	255

Aufgabe:

Im **Dokument 21_Zweierkomplement.pdf** finden Sie Beispiele für das Umrechnen des Zweierkomplements.

Sowie Erklärungen, wie **Computer Zahlen addiert, subtrahiert, multipliziert und dividiert.**

- Lesen Sie das Dokument aufmerksam durch.
- Rechnen Sie mindestens drei verschiedene Zweierkomplemente selbst aus.
- Rechnen Sie mindestens zwei Additionen und Subtraktionen.
- Versuchen Sie mindestens eine Multiplikation und Division.

⇨ Kontrollieren Sie Ihre Resultate selbst mit dem Taschenrechner

⇨ Tauschen Sie Ihre Rechnungen mit dem Nachbarn aus.
Erklären Sie sich gegenseitig, was Sie gemacht haben.

Von links nach rechts oder umgekehrt? LSB versus MSB

Ausgangslage sei die Zahl 83:

83 = $0101'0011_{bin}$

Bit Nr	7	6	5	4	3	2	1	0
2er Potenz	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
83	0	1	0	1	0	0	1	1

Doch nicht jedes System ordnet die Bits dabei genauso an!

Wenn Sie die **Daten bitweise übertragen** (**Serielle Datenübertragung**), dann ist es deshalb sehr wichtig, genau zu wissen, welches Bit mit welcher 2er Potenz zuerst übermittelt wird.

Unterscheiden Sie die beiden Fälle LSB und MSB!

LSB Bitnummerierung: (least significant bit)

Bit 0 hat niedrigsten Stellenwert, also 2^0

Bit Nr	7	6	5	4	3	2	1	0
2er Potenz	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
83	0	1	0	1	0	0	1	1

MSB Bitnummerierung: (most significant bit)

Bit 0 hat den höchsten Stellenwert, also 2^{n-1}

Bit Nr	7	6	5	4	3	2	1	0
2er Potenz	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
202	0	1	0	1	0	0	1	1

Merke: In der Codierung einer Datenübertragung zwischen zwei Systemen muss man abmachen, ob man LSB oder MSB anwendet.

Aufgabe:

Sie haben diese Bitfolge über eine serielle Übertragung erhalten:

Bit Nr	0	1	2	3	4	5	6	7
Bit Wert	1	0	1	0	1	1	0	0

a) Berechnen Sie die dezimale Zahl einmal mit LSB und einmal mit MSB (Ohne Zweierkomplement)

b) Tauschen Sie mit Ihrem Banknachbarn eine eigene Bitfolge aus. Berechnen Sie LSB und MSB und vergleichen Sie Ihre Resultate

Der Grosse und der kleine Indianer

Ausgangslage sei die Zahl 27888 = $6CF0_{Hex}$

Die Zahl besteht aus zwei Bytes:

$$6CF0_{Hex} = \mathbf{6C_{Hex}} \times 2^8 + \mathbf{F0_{Hex}}$$

In dieser Darstellung steht das **höherwertige Byte $\mathbf{6C_{Hex}}$ links** und das **niederwertige Byte $\mathbf{F0_{Hex}}$ rechts**.

Nehmen wir an, zwei Geräte übertrage Daten, **Bytes um Bytes**.

Es ist dabei nicht selbstverständlich, **welches Byte zuerst übertragen** wird.

Variante A) $6C_{Hex}$ und dann $F0_{Hex}$

Variante B) $F0_{Hex}$ und dann $6C_{Hex}$

Für den Empfänger ist es also wichtig zu wissen, ob das erste empfange Byte, das höherwertige oder das niederwertige ist.

⇨ Die **Reihenfolge der Bytes** muss für eine Übertragung **definiert werden**.

Die Reihenfolge spielt auch beim Speichern im Memory oder in einer Datei eine Rolle. Liegt das höherwertige Byte vor oder nach dem niederwertigen Byte im Speicher?

Diese Byte-Reihenfolge nennt man *endianness*.

Big-Endian:

Das höchstwertige Byte wird zuerst übertragen, respektive zuerst gespeichert.

Beispiel: Zuerst $6C_{Hex}$ und dann $F0_{Hex}$

Little-Endian:

Das kleinstwertige Byte wird zuerst übertragen, respektive zuerst gespeichert.

Beispiel: Zuerst $F0_{Hex}$ und dann $6C_{Hex}$

In der Computerwelt hat man lange von den zwei Welten *Motorola versus Intel* gesprochen. Der Hersteller Motorola verwendete Big-Endian. Intel setzte hingegen auf Little-Endian.

Merke: Beim hardwarenahen Übertragen, respektive Speichern von Daten muss man wissen, ob Big-Endian oder Little-Endian verwendet wird.

Aufgabe:

In einem Speicher stehen diese Bytes hintereinander:

Speicher	Wert
0000	A9 Hex
0001	B3 Hex

- Berechnen Sie die dezimale Zahl einmal mit Big-Endian und einmal mit Little Endian

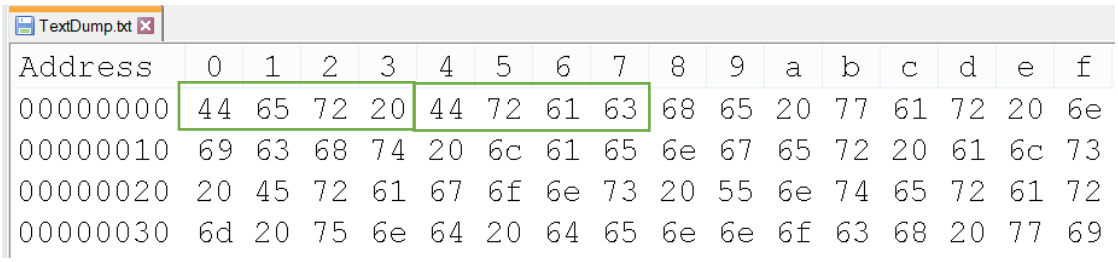
Aufgabe:

In einem Memory-Dump stehen viele Bytes hintereinander. Was bedeuten diese Bytes?

In diesem Beispiel müssen immer vier Bytes hintereinander als eine Zahl vom Typ Integer mit 32 Bit also mit 4 Bytes verstanden werden.

Doch wie sollen die vier aufeinanderfolgenden Bytes interpretiert werden? Berechnen Sie einmal das Ergebnis mit Big-Endian und einmal mit Little Endian.

Und machen Sie es nochmals vier die nächsten vier Bytes.


TextDump.txt

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00000000	44	65	72	20	44	72	61	63	68	65	20	77	61	72	20	6e
00000010	69	63	68	74	20	6c	61	65	6e	67	65	72	20	61	6c	73
00000020	20	45	72	61	67	6f	6e	73	20	55	6e	74	65	72	61	72
00000030	6d	20	75	6e	64	20	64	65	6e	6e	6f	63	68	20	77	69

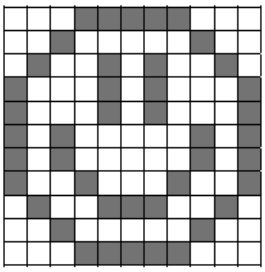
IEEE 754

Aufgabe:

- Lesen Sie das Dokument 40_IEEE 754.pdf
- Halten Sie für sich fest wie mit der Norm eine Zahl definiert wird:

Bilder

Bei Schwarz-Weiss-Zeichnungen ist jeder Bildpunkt entweder schwarz oder weiss und kann unmittelbar durch die Symbole 0 oder 1 codiert werden.



0	0	0	1	1	1	1	0	0	0
0	0	1	0	0	0	0	1	0	0
0	1	0	0	1	0	1	0	0	1	0
1	0	0	0	1	0	1	0	0	0	1
1	0	0	0	1	0	1	0	0	0	1
1	0	1	0	0	0	0	0	1	0	1
1	0	1	0	0	0	0	0	1	0	1
1	0	0	1	0	0	0	1	0	0	1
0	1	0	0	1	1	1	0	0	1	0
0	0	1	0	0	0	0	0	1	0	0
0	0	0	1	1	1	1	1	0	0	0

Der Hexdump stammt von einer schwarz/weiss Bitmap mit einer Auflösung von 32x40 Pixel. Wandeln Sie die Hex Zahlen in Binär um und übertragen Sie die "Bits" in das Raster rechts.

Aufgabe:

Malen Sie die Kästchen, wenn die Bits gesetzt sind und lassen Sie die Kästchen leer wenn die Bits nicht gesetzt sind.

00	00	00	00
00	00	00	00
00	0F	E0	00
00	1F	F0	00
00	1F	F0	00
00	3F	F8	00
00	33	38	00
00	33	38	00
00	73	3C	00
00	7F	FC	00
00	70	3C	00
00	60	3C	00
00	30	DC	00
00	39	DC	00
00	2F	1E	00
00	26	0F	00
00	60	0F	00
00	C0	07	80
00	C0	07	C0
01	80	07	C0
01	80	03	E0
03	80	03	E0
03	80	03	F0
03	80	01	F0
03	80	01	F8
07	80	01	F8
0E	00	01	F8
0A	00	01	F8
39	80	01	F8
20	C0	01	CC
30	60	01	86
30	70	03	02
60	30	0E	03
40	18	1C	01
60	0C	78	03
18	1F	F8	0E
0F	1F	F8	78
00	F0	1F	C0
00	00	00	00
00	00	00	00

ASCII

Folgend sehen Sie einen "Hex-Dump" einer Text Datei.

Aufgabe:

- Übersetzen Sie mit Hilfe der ASCII-Tabelle 22_ASCII_Tabelle.pdf die Hexadezimal-Zahlen in lesbaren ASCII Text.

TextDump.txt

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00000000	44	65	72	20	44	72	61	63	68	65	20	77	61	72	20	6e
00000010	69	63	68	74	20	6c	61	65	6e	67	65	72	20	61	6c	73
00000020	20	45	72	61	67	6f	6e	73	20	55	6e	74	65	72	61	72
00000030	6d	20	75	6e	64	20	64	65	6e	6e	6f	63	68	20	77	69
00000040	72	6b	74	65	20	65	72	20	77	75	65	72	64	65	76	6f
00000050	6c	6c	20	75	6e	64	20	61	6e	6d	75	74	69	67	2e	20
00000060	53	65	69	6e	65	20	53	63	68	75	70	70	65	6e	20	77
00000070	61	72	65	6e	20	73	61	70	68	69	72	62	6c	61	75	2c
00000080	20	64	69	65	73	65	6c	62	65	20	46	61	72	62	65	20
00000090	77	69	65	20	64	65	72	20	53	74	65	69	6e	2e	20	20

Unicode UTF-8

Aufgabe:

- Lesen Sie das Dokument 30_Unicode.pdf.
- Halten Sie für sich fest, worin Unterscheidet sich ASCII von Unicode UTF-8 unterscheidet.