Zahlensysteme.

Ein **Zahlensystem ist ein Code**, um Zahlenwerte nach einem bestimmten Schlüssel darzustellen. Die Darstellung der Zahlen erfolgt bei den hier vorgestellten Zahlensystemen allgemein in der Form:

$b_n b_{n-1} ... b_i ... b_1 b_0$ (beliebige Zahl, bestehend aus den Ziffern b_i)

Die **Decodierung** bedeutet, eine Zahl aus einem Zahlensystem mit einer anderen Basis B als 10 in das Dezimalsystem um zurechnen. Ist B die Basis des Zahlensystems, lautet allgemein die Vorschrift zum Decodieren zur Dezimalzahl D:

$$D = \sum_{i=0}^{n} b_i \cdot B^i = b_0 \cdot B^0 + b_1 \cdot B^1 + ... + b_{n-1} \cdot B^{n-1} + b_n \cdot B^n$$

(Potenzschreibweise)

Dezimalsystem.

Das **Dezimalsystem** besitzt die **Basis** B=**10**, die 10 **Dezimalziffern** b_i lauten: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**.

$$125 = 5 \cdot \mathbf{10}^0 + 2 \cdot \mathbf{10}^1 + 1 \cdot \mathbf{10}^2$$
$$= 5 \cdot 1 + 2 \cdot 10 + 1 \cdot 100$$
$$= 125_D$$ *Suffix D für Dezimal*

Dualsystem.

Das **Dualsystem** gehört zu der Gruppe der Binärsysteme (auch der BCD Zahlencode arbeitet mit Binärziffern - zum Beispiel bei Registrierkassen) und besitzt die **Basis** B=**2**, die beiden **Binärziffern** b_i lauten **0, 1**.

Im Englischen heißt Binärziffer **Binary Digit**, aus dem sich die **Abkürzung Bit** ableitet. Computer, besser gesagt Digitalrechner (es gibt auch Analogrechner), arbeiten mit Binärziffern.

Beispiel Decodierung der Dualzahl 1101:

$$1101_B = 1 \cdot \mathbf{2}^0 + 0 \cdot \mathbf{2}^1 + 1 \cdot \mathbf{2}^2 + 1 \cdot \mathbf{2}^3$$
$$= 1 \cdot 1 + 0 \cdot 2 + 1 \cdot 4 + 1 \cdot 8$$
$$= 13_D$$ *Suffix B für Binär*

Die Dualzahl 1101 hat also im Dezimalsystem den Wert 13. Für den gleichen Zahlenwert benötigt eine Dualzahl mehr Stellen als eine Dezimalzahl.

Hexadezimalsystem.

Das **Hexadezimalsystem** besitzt die **Basis** B=**16**, die 16 Hexadezimalziffern b_i lauten:
0, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **A** (10), **B** (11), **C** (12), **D** (13), **E** (14), **F** (15).
Hexadezimalzahlen dienen dem Menschen zur einfacheren Darstellung von Dualzahlen und werden in der Praxis vereinzelt benötigt (zum Beispiel um auf Webseiten Farben darzustellen oder bei der Programmierung).

Beispiel Decodierung der Hexadezimalzahl AFFE:

$$AFFE_H = E \cdot \mathbf{16}^0 + F \cdot \mathbf{16}^1 + F \cdot \mathbf{16}^2 + A \cdot \mathbf{16}^3$$
$$= 14 \cdot 1 + 15 \cdot 16 + 15 \cdot 256 + 10 \cdot 4096$$
$$= 45054_D$$ *Suffix H für Hexadezimal*

Codierung.

Die **Codierung wandelt eine Dezimalzahl** in eine Zahl eines anderen Zahlensystems der Basis B um. Der allgemeine Algorithmus zur **Codierung besteht in der fortlaufenden Ganzzahldivision** durch die Basis B des Zahlensystems, in das die Dezimalzahl D gewandelt werden soll. Der **Rest ist eine codierte Ziffer b** und das Ganzzahlergebnis der Division ist die Ausgangszahl für eine nächste Division. Dies wird solange fortgesetzt, bis das Ganzzahlergebnis Null ist. Mathematisch liegt der Codierung das **Horner-Schema** zugrunde.

Allgemeine Erklärung des Verfahrens (**Horner-Schema**), hier für B=2:

$$D = b_n \cdot 2^n + b_{n-1} \cdot 2^{n-1} + \ldots + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0$$
$$= (b_n \cdot 2^{n-1} + b_{n-1} \cdot 2^{n-2} + \ldots + b_2 \cdot 2^1 + b_1) \cdot 2 + b_0$$
$$= ((b_n \cdot 2^{n-2} + b_{n-1} \cdot 2^{n-3} + \ldots + b_2) \cdot 2 + b_1) \cdot 2 + b_0$$
$$\ldots \text{ usw. } \ldots$$
$$D = (\ldots(b_n \cdot 2 + b_{n-1}) \cdot 2 + b_{n-2}) \cdot 2 + \ldots + b_2) \cdot 2 + b_1) \cdot 2 + b_0$$

Das heißt, durch Umformung der Potenzschreibweise nach dem Horner-Schema lassen sich die Binärziffern einer Zahl direkt entnehmen. Zur Veranschaulichung - wenn die Dezimalzahl 25 der Dualzahl 11001 entspricht, läßt sich die Dualzahl wie folgt nach dem Horner-Schema umformen (überzeugen Sie sich, jede Zeile ergibt die Dezimalzahl 25):

$$11001_B = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 = 25_D$$
$$= (1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0) \cdot 2 + 1 = 25_D$$
$$= ((1 \cdot 2^2 + 1 \cdot 2^1 + 0) \cdot 2 + 0) \cdot 2 + 1 = 25_D$$
$$= (((1 \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 0) \cdot 2 + 1 = 25_D$$
$$= ((((1) \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 0) \cdot 2 + 1 = 25_D$$

Bei einer fortlaufender Ganzzahldivision durch 2 einer Dezimalzahl entsprechen die Binärziffern des Horner-Schema immer den Resten:

25 : 2 = 12	Rest **1**	*Berechnung entspricht* { ((((**1**)·2 + **1**)·2 + **0**)·2 + **0**)·2 + **1** } / 2
12 : 2 = 6	Rest **0**	*Berechnung entspricht* { (((**1**)·2 + **1**)·2 + **0**)·2 + **0** } / 2
6 : 2 = 3	Rest **0**	*Berechnung entspricht* { ((**1**)·2 + **1**)·2 + **0** } / 2
3 : 2 = 1	Rest **1**	*Berechnung entspricht* { (**1**)·2 + **1** } / 2
1 : 2 = 0	Rest **1**	*Berechnung entspricht* { **1** } / 2

In der **praktischen Ausführung** erfolgt die Codierung der **Dezimalzahl 25 in eine Dualzahl** also wie folgt:

- Fortlaufenden Ganzzahldivision durch Basis B des Zielsystems (hier B=2).
- Der Rest der Ganzzahldivision ist eine codierte Ziffer b.
- Ganzzahlergebnis der Division ist Ausgangszahl für eine nächste Ganzzahldivision.
- Wird solange fortgesetzt, bis das Ganzzahlergebnis Null ist.

25 : 2 = 12	Rest **1**	
12 : 2 = 6	Rest **0**	
6 : 2 = 3	Rest **0**	
3 : 2 = 1	Rest **1**	
1 : 2 = 0	Rest **1**	**1 1 0 0 1** B

Die Dezimalzahl 25 entspricht also der Dualzahl 11001.

Codierung der **Dezimalzahl 167 in eine Hexadezimalzahl**:

167 : 16 = 10	Rest **7**	
10 : 16 = 0	Rest 10 (**A**)	**A 7** H

Die Dezimalzahl 167 entspricht also der Hexadezimalzahl A7.

Dez	Dual	Hex	Dez	Dual	Hex
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

Die Umwandlung zwischen Dual- und Hexadezimalsystem erfolgt per **Tetradendarstellung**, **jede Hexadezimalziffer codiert 4 Bit**. Dadurch wird es dem Menschen erleichtert sich Dualzahlen zu merken, indem er sich den hexadezimalen Wert dafür merkt. Zur Rückwandlung muß er nur die Dualzahlen für die 15 Hexadezimalziffern auswendig gelernt haben.

b_i				HEX
b_3	b_2	b_1	b_0	DUAL

A				F				F				E				HEX
1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	0	DUAL

Angaben der Speicherkapazität.

Die **Einheiten für Angaben zu Speicherkapazitäten** sind nicht immer einheitlich. Eindeutig ist die **Einheit Byte**. Sie resultiert aus dem Aufbau des RAM Speichers eines Rechners (der Arbeitsspeicher, der den Inhalt im stromlosen Zustand seinen Inhalt verliert) bei dem immer 8 binäre Ziffern organisatorisch zusammengefasst sind:

1 Byte (1 B) = 8 Bit (8 b)

Weil Menschen im Alltag im **Dezimalsystem** rechnen, sind die Präfixe Kilo, Mega usw. auch dem Dezimalsystem angepasst, entsprechend den **SI Einheiten:** zum Beispiel: 1000 g = 10^3 g = 1 Kg bei Massen oder 84,3 MHz = 84,3 10^6 Hz bei Frequenzen.

In der EDV basieren die Zahlen jedoch auf dem **Dualsystem mit der Basis 2**. Daher werden für die gleichen Präfixe Werte angenommen, die deren dezimalen Werten am nächsten kommen:

1024 Byte = 2^{10} Byte = 1 Kilobyte 1024 Kilobyte = 1 Megabyte
1024 Megabyte = 1 Gigabyte 1024 Gigabyte = 1 Terabyte

Das Problem besteht nun darin, das viele Hersteller von Speichermedien (Festplatten, USB Sticks, ...) bei der Kapazitätsangabe nicht die Einheiten auf das Dualsystem beziehen, sondern auf das Dezimalsystem - dadurch ergibt sich ein größerer Zahlenwert.

Beispiel: Wenn Sie einen DVD Rohling käuflich erwerben, ist auf der Hülle in der Regel eine Kapazität von 4,7 GB angegeben. Ihr Brennprogramm dagegen zeigt aber nur eine max. Kapazität von 4,38 GB an.

***4,7 GB auf der Basis von 10** = 4 700 000 000 B*
4 700 000 000 B / (1024 x 1024 x 1024) = ***4,377 GB auf der Basis von 2***

Im Prinzip ist allerdings das Problem seit 2000 durch den **Standard 60027-2** der **IEC (International Electronical Commission)** gelöst. Danach sollten die Einheiten mit den Präfixen KB (Kilobyte), MB (Megabyte), GB (Gigabyte), TB (Terabyte) als SI Einheiten auf der Zahlenbasis von 10 verwendet werden und für die **Präfixe der Einheiten auf der Basis von 2** sollten folgende Einheiten verwendet werden:

1024 Byte = 2^{10} Byte = 1 KiB 1024 KiB = 1 MiB
1024 MiB = 1 GiB 1024 GiB = 1 TiB

Die Einheiten werden ausgesprochen als:

- **KiB** **Kibibyte** (Kibi von engl. Kilobinary)
- **MiB** **Mebibyte** (Mebi von engl. Megabinary)
- **GiB** **Gibibyte** (Gigi von engl. Gigabinary)
- **TiB** **Tebibyte** (Tebi von engl. Terabinary)

Allerdings werden diese Einheiten noch nicht allzu oft genutzt.