# Hexadezimal

>> Die meisten westlichen Kulturen verwenden das Dezimalsystem (Basis 10) zur Darstellung von Zahlen:

0 1 2 3 4 5 6 7 8 9

Computer verwenden das Binärsystem (Basis 2) zur Darstellung aller Daten:

0 1

>> In der Informatik ist es wichtig, Daten so darzustellen, wie der Computer sie verarbeitet.

Problem: Lange Ketten von Nullen und Einsen sind für Menschen schwer zu erfassen.

## Das Hexadezimalsystem

Das Hexadezimalsystem (Basis 16) ist eine effiziente Methode zur Darstellung von Computerdaten.

Das Hexadezimalsystem ermöglicht eine einfache Abbildung, da es bei vier binären Stellen (= Halbbyte, „Nibble“) genau 16 Kombinationen gibt, die sich genau mit einer hexadezimalen Ziffer darstellen lassen:

`0 1 2 3 4 5 6 7 8 9 a b c d e f`

`0 1 2 3 4 5 6 7 8 9 A B C D E F`

| Dez | Binär | Hex |
|---|---|---|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |

| Dez | Binär | Hex |
|---|---|---|
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |

| Dez | Binär | Hex |
|---|---|---|
| 0 | 0000 | 0x0 |
| 1 | 0001 | 0x1 |
| 2 | 0010 | 0x2 |
| 3 | 0011 | 0x3 |
| 4 | 0100 | 0x4 |
| 5 | 0101 | 0x5 |
| 6 | 0110 | 0x6 |
| 7 | 0111 | 0x7 |

| Dez | Binär | Hex |
|---|---|---|
| 8 | 1000 | 0x8 |
| 9 | 1001 | 0x9 |
| 10 | 1010 | 0xA |
| 11 | 1011 | 0xB |
| 12 | 1100 | 0xC |
| 13 | 1101 | 0xD |
| 14 | 1110 | 0xE |
| 15 | 1111 | 0xF |

**Stellen-wert-systeme**

Wie im Binär- und Dezimalsystem gibt es auch im Hexadezimalsystem Stellenwerte. Dabei entspricht jede Position einer Potenz von 16.

Binär:
Potenzen von 2 (1, 2, 4, 8, ...)

Dezimal:
Potenzen von 10 (1, 10, 100, 1000, ...)

Hexadezimal:
Potenzen von 16 (1, 16, 256, 4096, ...)

## Beispiel

3 9 7

## Hexadezimalzahl → Dezimalzahl

```
0x    3    9    7
```

**Hexadezimalzahl → Dezimalzahl**

| | $16^2$ | $16^1$ | $16^0$ |
|---|---|---|---|
| 0x | 3 | 9 | 7 |

**Hexadezimalzahl → Dezimalzahl**

| | $16^2$ | $16^1$ | $16^0$ |
|---|---|---|---|
| 0x | 3 | 9 | 7 |
| | $3 \cdot 256$ | $9 \cdot 16$ | $7 \cdot 1$ |

Also 919 dezimal.

**Hexadezimalzahl → Dezimalzahl**

| | $16^2$ | $16^1$ | $16^0$ |
|---|---|---|---|
| 0x | A | D | C |
| | 10·256 | 13·16 | 12·1 |

10·256 + 13·16 + 12·1 =
2560 + 208 + 12 = 2780

**Dezimalzahl → Hexadezimalzahl**

Wiederholte Division durch 16.
Reste ergeben von unten nach
oben die Hexadezimalstellen.

Beispiel: Dezimalzahl **1234**

```
1234 / 16 = 77  Rest 2
  77 / 16 =  4  Rest 13 (= d)
   4 / 16 =  0  Rest 4

Ergebnis: 0x4d2
```

## Binär → Hexadezimalzahl

Von rechts kommend immer 4 Bits gruppieren. Die Gruppe ganz links bei Bedarf mit zusätzlichen 0-Bits vorne auffüllen. Dann die Stellenwerte der auf 1 gesetzten Bits (also 8, 4, 2, 1) summieren, um Nibble einzeln umzuwandeln.

Beispiel:

`01000110101000101011100100111101`

## Binär → Hexadezimalzahl

Von rechts kommend immer 4 Bits gruppieren. Die Gruppe ganz links bei Bedarf mit zusätzlichen 0-Bits vorne auffüllen. Dann die Stellenwerte der auf 1 gesetzten Bits (also 8, 4, 2, 1) summieren, um Nibble einzeln umzuwandeln.

Beispiel:

```
01000110101000101011100100111101

0100 0110 1010 0010 1011 1001 0011 1101

   4
```

## Binär → Hexadezimalzahl

Von rechts kommend immer 4 Bits gruppieren. Die Gruppe ganz links bei Bedarf mit zusätzlichen 0-Bits vorne auffüllen. Dann die Stellenwerte der auf 1 gesetzten Bits (also 8, 4, 2, 1) summieren, um Nibble einzeln umzuwandeln.

Beispiel:

```
01000110101000101011100100111101

0100 0110 1010 0010 1011 1001 0011 1101

   4    6
```

## Binär → Hexadezimalzahl

Von rechts kommend immer 4 Bits gruppieren. Die Gruppe ganz links bei Bedarf mit zusätzlichen 0-Bits vorne auffüllen. Dann die Stellenwerte der auf 1 gesetzten Bits (also 8, 4, 2, 1) summieren, um Nibble einzeln umzuwandeln.

Beispiel:

```
01000110101000101011100100111101

0100 0110 1010 0010 1011 1001 0011 1101

   4    6   10
```

## Binär → Hexadezimalzahl

Von rechts kommend immer 4 Bits gruppieren. Die Gruppe ganz links bei Bedarf mit zusätzlichen 0-Bits vorne auffüllen. Dann die Stellenwerte der auf 1 gesetzten Bits (also 8, 4, 2, 1) summieren, um Nibble einzeln umzuwandeln.

Beispiel:

```
01000110101000101011100100111101

0100 0110 1010 0010 1011 1001 0011 1101

   4    6    A
```

## Binär → Hexadezimalzahl

Von rechts kommend immer 4 Bits gruppieren. Die Gruppe ganz links bei Bedarf mit zusätzlichen 0-Bits vorne auffüllen. Dann die Stellenwerte der auf 1 gesetzten Bits (also 8, 4, 2, 1) summieren, um Nibble einzeln umzuwandeln.

Beispiel:

```
01000110101000101011100100111101

0100 0110 1010 0010 1011 1001 0011 1101

   4    6    A    2    B    9    3    D
```

**Binär → Hexadezimalzahl**

Von rechts kommend immer 4 Bits gruppieren. Die Gruppe ganz links bei Bedarf mit zusätzlichen 0-Bits vorne auffüllen. Dann die Stellenwerte der auf 1 gesetzten Bits (also 8, 4, 2, 1) summieren, um Nibble einzeln umzuwandeln.

Beispiel:

```
01000110101000101011100100111101

0100 0110 1010 0010 1011 1001 0011 1101

   4    6    A    2    B    9    3    D

Ergebnis: 0x46A2B93D
```

**Verwendung**

Beim Debugging werden Speicheradressen üblicherweise als Hexadezimalzahlen dargestellt.

Auch beim Programmieren von systemnahen Programmen werden sie benötigt.

Darüber hinaus viele Anwendungen, z.B. RGB-Farbcodes in HTML/CSS.

# EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

## Stellenwertsysteme

Bei Basis *b* gibt es *b* verschiedene Ziffern. Die Positionen entsprechen den Potenzen der Basis. Bekannte Beispiele:

Binär (*b* = 2)

Dezimal (*b* = 10)

Hexadezimal (*b* = 16)

Stellenwertsysteme können dazu genutzt werden, eine abgeschlossene Menge von Daten, die einer Ordnung unterliegen, zu kodieren.

**Stellenwertsysteme: Kodierung von Daten**

Beispiel:
Menge der Buchstaben von A bis Z (26)

Bei *b* = 3:

D = 10 weil

A = 0

B = 1

C = 2

D = 2 + 1 (= 3, aber bei *b* = 3 gibt es ja nur die 3 Ziffern 0, 1, 2!), daher Übertrag zur nächsten Stelle, somit 10)

## Stellenwertsysteme: Kodierung von Daten

Beispiel:
Menge der Buchstaben von A bis Z (26)

A = 0, B = 1, … Z = 25

Kompakteres Kodieren als ASCII

Beispiel. INFORMATIK (10 Stellen)

I = 8, N = 13, F = 5, O = …

Als Dezimalzahl:

$8 \cdot 26^9 + 13 \cdot 26^8 + 5 \cdot 26^7 + \ldots$

$= 46.195.472.623.206$

**Stellenwertsysteme: Kodierung von Daten**

Menge der Buchstaben von A bis Z (26)
INFORMATIK (10 Stellen)
Als Dezimalzahl: 46.195.472.623.206

Benötigter Platz zur Speicherung: 6 Bytes.

*Warum?*
10 Buchstaben in Basis 26
Anzahl Möglichkeiten: $26^{10}$
$\log_2(26^{10}) \approx 47{,}004$
→ 6 Bytes (48 Bits) reichen aus
Vergleich: Als ASCII-String 10 Bytes benötigt.

## Stellenwertsysteme: Kodierung von Daten

Menge der Buchstaben von A bis Z (26)
INFORMATIK (10 Stellen)
Als Dezimalzahl: 46.195.472.623.206

Benötigter Platz zur Speicherung: 6 Bytes.

*Warum?*
10 Buchstaben in Basis 26
Anzahl Möglichkeiten: $26^{10}$
$\log_2(26^{10}) \approx 47{,}004$ **$= \log_{10}(26^{10}) / \log_{10}(2)$**
→ 6 Bytes (48 Bits) reichen aus
Vergleich: Als ASCII-String 10 Bytes benötigt.

**Wie dekodiert man die Zahl zu ihrem String?**

Beispiel: **1234**

So wie vorhin gezeigt: Wiederholte Division durch die Basis (nun 26) und die erhaltenen Reste aufschreiben.

Leere Stellen links ggf. mit 0-en auffüllen (Padding) so dass die bei der Kodierung vereinbarte Anzahl an Stellen erreicht wird.

```
Erster Schritt:
1234 / 26 = 47 Rest 12 (= M)

Ergebnis (10-stellig):
A A A A A A A B V M
```

Hexadezimal verbindet Binär- und Dezimalsystem (4 Bits = 1 Hex-Ziffer)

0x-Präfix kennzeichnet Hexadezimalzahlen eindeutig

Konvertierung Dez zu Hex: Division bzw. Multiplikation mit Stellenwerten

Anwendungen: Speicheradressen, Debugging, RGB-Farben (#FF0000)

Konvertierung Binär zu Hex: 4er-Gruppen direkt umwandeln

Stellenwertsysteme funktionieren mit beliebiger Basis (z.B. „Base26“ für Text)