## 1.2 Einführung und Zahlensysteme/Stellenwertsysteme

### 1.2.1 Darstellung der Dualzahlen im Computer

In weiten Bereichen der Technik hat sich das Dualzahlensystem zur Speicherung und Verarbeitung von Zahlen durchgesetzt. Ausnahmen sind Taschenrechner und kaufmännische Programme, die in der Mehrheit BCD-Zahlen benutzen.

Allerdings steht in einem Computer für einen Zahlenwert in der Regel nur ein (vor der Benutzung) eng begrenzter Speicherplatz zur Verfügung.

Dieser Speicherplatz wird in der Anzahl der gespeicherten Dualziffern (*binary digit*=Bit) angegeben. Vielfache dieser Einheit sind in Tabelle 1 dargestellt.

| Anzahl | Anzahl | Bezeichnung | weitere Bez. |
|---:|---:|---|---|
| $2^2$ Bit | 4 Bit | 1 Nibble | 1 Tetrade |
| $2^3$ Bit | 8 Bit | 1 Byte | 1 Oktett |
| $2^{10}$ Bit | 1024 Bit | 1 KibiBit | 1 Kib, 1 Kb |
| $2^{13}$ Bit | 1024 Byte | 1 KibiByte | 1 KiB, 1 KB |
| $2^{20}$ Bit | 1048576 Bit | 1 MebiBit | 1 Mib |
| $2^{23}$ Bit | 1048576 Byte | 1 MebiByte | 1 MiB |
| $2^{30}$ Bit | 1073741824 Bit | 1 GibiBit | 1 Gib |
| $2^{33}$ Bit | 1073741824 Byte | 1 GibiByte | 1 GiB |
| $2^{40}$ Bit | 1099511627776 Bit | 1 TebiBit | 1 Tib |
| $2^{43}$ Bit | 1099511627776 Byte | 1 TebiByte | 1 TiB |
| $1 \cdot 10^3$ Bit | 1000 Bit | 1 KiloBit | 1 kb |
| $8 \cdot 10^3$ Bit | 1000 Byte | 1 KiloByte | 1 kB |
| $1 \cdot 10^6$ Bit | 1000000 Bit | 1 MegaBit | 1 Mb |
| $8 \cdot 10^6$ Bit | 1000000 Byte | 1 MegaByte | 1 MB |
| $1 \cdot 10^9$ Bit | 1000000000 Bit | 1 GigaBit | 1 Gb |
| $8 \cdot 10^9$ Bit | 1000000000 Byte | 1 GigaByte | 1 GB |
| $1 \cdot 10^{12}$ Bit | 1000000000000 Bit | 1 TeraBit | 1 Tb |
| $8 \cdot 10^{12}$ Bit | 1000000000000 Byte | 1 TeraByte | 1 TB |

Tabelle 1: Vielfache der Einheit Bit

### 1.2.2 Datentypen

In Programmen werden Zahlen für Eingabe, Berechnung, Ausgabe in *Variablen* gespeichert. Je nach Verwendungszweck (große, kleine Zahlen, genaue oder ungenaue Operationen) kann man Zahlen in *verschiedenen numerischen Datentypen* speichern. Jeder numerische Datentyp hat einen Wertebereich, innerhalb dessen die zu speichernde Zahl liegen darf, z.B. 0..255 (256 Werte). Jeder numerische Datentyp belegt im Speicher eine bestimmte Anzahl von Bits (z.B. 8 Bits). Außerdem ist für jeden Datentyp eine Zuordnung festgelegt, mit der die Bitmuster den Werten zugeordnet werden können (z. B. sogenannter natürlicher Binärcode) sowie eine Menge erlaubter Rechenoperationen (z. B. plus, minus, ...), die man mit Variablen dieses Datentyps ausführen darf. Die Anzahl der möglichen unterschiedlichen Werte ist:

$$AUW = 2^{Wortbreite}$$

Damit liegt der Zahlenbereich, soweit man sich auf nicht-negative Zahlen beschränken kann, zwischen null und $2^{Wortbreite} - 1$. Tabelle 2 zeigt das am Beispiel des GNU-C-Compilers.

### 1.2.3 Umwandlung zwischen Datentypen

Wie kann man positive ganze Zahlen um zusätzliche Binärstellen (Bits) erweitern?

- Beispiel (4 auf 16): $7 = 0111 \mapsto 0000.0000.0000.0111 = 7$

| Datentyp | Wortbreite in Bit | Wertebereich |
|---|---|---|
| unsigned char | 8 | 0 . . . 255 |
| unsigned short int | 16 | 0 . . . 65.535 |
| unsigned int | 32 | 0 . . . 4.294.967.295 |
| unsigned long int | 32 | 0 . . . 4.294.967.295 |
| unsigned long long int | 64 | 0 . . . 18.446.744.073.709.551.615 |

Tabelle 2: Auswahl von Datentypen in GNU-C (x86/32)

- Ergebnis: Es können stets beliebig viele Stellen links angesetzt werden; sie sind mit Nullen zu füllen.

Wann und wie kann man positive ganze Zahlen um Binärstellen reduzieren?

- Beispiel (16 auf 4): $7 = 0000.0000.0000.0111 \mapsto 0111 = 7$
- Beispiel (16 auf 4): $23 = 0000.0000.0001.0111 \mapsto 0111 = 7$ — Fehler!
- Ergebnis: Reduktion ist nur erlaubt, wenn alle zu reduzierenden führenden Stellen null sind. Dann können diese Stellen weggelassen werden.

### 1.2.4 Hilfsmittel zur Darstellung von Dualzahlen für Menschen

Die Zahlendarstellung mit dem Dualsystem ist für Computer sehr geschickt, für Menschen sind lange Zahlenkolonnen (Beispiel: $12345_{(10)} = 11000000111001_{(2)}$) sehr unübersichtlich; der Umgang mit dem Dualsystem ist zudem ungewohnt. Passt man die Programme so an, dass sie zur Ein- und Ausgabe das Dezimalsystem benutzen, werden aufwendige Umwandlungen gebraucht.

Zur Abhilfe wurden das Oktalsystem und das Hexadezimalsystem erfunden.

### 1.2.5 Das Oktalsystem

Das Oktalsystem ist ein Stellenwertsystem mit der Basis 8. Jede Stelle hat also achtmal so viel Gewicht wie die folgende Stelle. Die Ziffern sind 0, 1, 2, 3, 4, 5, 6 und 7. Welche Vorteile hat dieses System?

a) Einfache Umwandlung dual → oktal: Je drei Dualziffern bilden eine Oktalziffer (Tabelle 3).

b) Einfache Umwandlung oktal → dual: Eine Oktalziffer wird in drei Dual-Ziffern zerlegt (ebenfalls Tabelle 3).

| Oktal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Dual | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |

Tabelle 3: Umwandlung oktal nach dual und zurück

Der Grund liegt darin, dass 8 selbst eine Zweierpotenz ist ($2^3 = 8$). Daher haben beide Zahlensysteme (dual und oktal) Stellenwerttafeln mit gemeinsamen Grenzen. Hier ein Beispiel, die Zahl $1011101111_2$:

| Stellenwerte, dual → | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| Ziffern, dual → | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| Gemeinsame Grenzen → | ‖ | | | | ‖ | | | ‖ | | ‖ |

| Stellenwerte, oktal → | 512 | 64 | 8 | 1 |
|---|---|---|---|---|
| Ziffern, oktal → | 1 | 3 | 5 | 7 |

Dual findet man in den letzten drei Stellen $1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1$, oktal ist das 7. Die $111_{(2)}$ wird so zur $7_{(8)}$.

In den drei Dualstellen davor findet man $1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8$, also $32 + 8$, man könnte auch sagen $4 \cdot 8 + 1 \cdot 8$. Oktal sind das $5 \cdot 8$. Die $101_{(2)}$ wird zur $5_{(8)}$.

Noch weiter vorn findet man $0 \cdot 256 + 1 \cdot 128 + 1 \cdot 64$, also $128 + 64$, man könnte auch sagen $2 \cdot 64 + 1 \cdot 64$. Oktal sind das $3 \cdot 64$. Die $011_{(2)}$ wird zur $3_{(8)}$.

Und ganz vorn ist in beiden Systemen $1 \cdot 512$. $1_{(2)}$ wird zur $1_{(8)}$:
**Immer drei Ziffern der Dualzahl entsprechen einer Oktalziffer.**

### 1.2.6 Umwandlung einer Zahl vom Dual- in das Oktalsystem

Hier soll die Zahl $1110101010011_{(2)}$ ins Oktalsystem umgewandelt werden. Zuerst fasst man immer drei Bits, beginnend von rechts (!), zusammen und trennt die Dreiergruppen durch Punkte:

```
1110101010011 -> 001.110.101.010.011
```

Falls wie hier links nicht genug Bits für eine Dreiergruppe vorhanden sind, füllt man mit Nullen auf. Nun fasst man jede Dreiergruppe als einzelne Dualzahl auf und wandelt sie (notfalls mit Hilfe der Tabelle 3) in eine Oktalziffer (oder Dezimalziffer) um:

```
421|421|421|421|421            011 = 2+1 = 3
-------------------            010 = 2
001.110.101.010.011            101 = 4+1 = 5
-------------------            110 = 4+2 = 6
 1   6   5   2   3             001 = 1
```

Das Ergebnis ist $16523_{(8)}$.

### 1.2.7 Umwandlung einer Zahl vom Oktal- in das Dualsystem

Bei der Rückumwandlung nimmt man jede Oktalziffer einzeln und wandelt sie so, als wenn sie eine eigenen Dezimalzahl wäre, in eine dreistellige Dualzahl um. Wieder ist es am einfachsten, wenn man Tabelle 3) benutzt

```
 1   6   5   2   3 =
001.110.101.010.011
```

Führenden Nullen kann man natürlich weglassen.

### 1.2.8 Das Hexadezimalsystem (Sedezimalsystem)

Bei diesem Stellenwertsystem ist die Basis 16. Jede Stelle hat also 16-mal so viel Gewicht wie die folgende Stelle. Als Ziffern benutzt man 0 bis 9 und A bis F. Die Vorteile sind:

a) Einfache Umwandlung dual $\rightarrow$ hexadezimal: Je vier Dualziffern bilden eine Hexadezimalziffer (Tabelle 4).

b) Einfache Umwandlung hexadezimal $\rightarrow$ dual: Eine Hexadezimalziffer wird in vier Dual-Ziffern zerlegt (ebenfalls Tabelle 4).

| Hexadezimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Dual | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |
| Hexadezimal | 8 | 9 | a | b | c | d | e | f |
| Dual | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |

Tabelle 4: Umwandlung hexadezimal nach dual und zurück

### 1.2.9 Umwandlung einer Zahl vom Dual- in das Hexadezimalsystem

Nun soll die Zahl $1110101010011_{()2}$ ins Hexadezimalsystem umgewandelt werden. Dazu fasst man hier immer vier Bits (beginnend von rechts) zusammen und trennt die Vierergruppen durch Punkte:

```
1110101010011 -> 0001.1101.0101.0011
```

Wieder füllt man von links mit Nullen auf. Ebenso fasst man jede Vierergruppe als einzelne Dualzahl auf und wandelt sie in eine Zahl (Dezimalsystem) um:

```
1|8421|8421|8421            0011 = 2+1 = 3
----------------            0101 = 4+1 = 5
1.1101.0101.0011            1101 = 8+4+1 = 13
----------------            0001 = 1
1. 13 .  5 .  3
```

Die Zahlen 10 bis 15 werden endlich durch die Ziffern A bis F ersetzt:

```
1.13.5.3 = 1D53 (Hex.)
```

Das Ergebnis ist jetzt also $1D53_{(16)}$.

### 1.2.10 Umwandlung einer Zahl vom Hexadezimal- in das Dualsystem

Bei der Rückumwandlung nimmt man jede Hexadezimalziffer einzeln und wandelt sie so, als wenn sie eine eigene Dezimalzahl wäre, in eine vierstellige Dualzahl um. Am einfachsten ist es, wenn man die Tabelle 4 benutzt.

```
  1    D    5    3  =
0001.1101.0101.0011
```

Führende Nullen kann man auch hier weglassen.

### 1.2.11 Umwandlung von einem beliebigen Stellenwertsystem ins Dezimalsystem

Ab und zu braucht man auch andere Zahlensysteme, zum Beispiel dann, wenn Daten gepackt werden sollen. Im folgenden Beispiel soll die Zahl $13240_{(5)}$ ins Dezimalsystem umgewandelt werden. Dazu wird zuerst die Stellenwerttabelle gebildet:

| Stellenwerte, dual → | 625 | 125 | 25 | 5 | 1 |
|---|---|---|---|---|---|
| Ziffern, dual → | | | | | |

Der Stellenwert ganz rechts ist immer eins. Der Stellenwert links davon ist 5, weil die Basis 5 ist. Der Stellenwert links davon ist wieder um den Faktor 5 größer, also $5 \cdot 5 = 25$. Der Stellenwert links davon ist wieder um den Faktor 5 größer, also $5 \cdot 25 = 125$.

Im zweiten Schritt werden die Ziffern in die Stellenwerttabelle eingetragen:

| Stellenwerte, dual → | 625 | 125 | 25 | 5 | 1 |
|---|---|---|---|---|---|
| Ziffern, dual → | 1 | 3 | 2 | 4 | 0 |

Zum Schluss errechnet man mit der Tabelle den Wert: $x = 1 \cdot 625 + 3 \cdot 125 + 2 \cdot 25 + 1 \cdot 5 + 0 \cdot 1 = 1070$

### 1.2.12 Umwandlung vom Dezimalsystem in ein beliebiges Stellenwertsystem

Nun soll die Zahl 1070 zurück in das Zahlensystem der Basis 5 umgewandelt werden. Dazu dient dasselbe Verfahren wie bei der Umwandlung ins Dualsystem, nämlich das fortlaufende Dividieren durch die Basis, nur eben hier mit der Basis 5 (anstelle der Basis 2 beim Dualsystem):

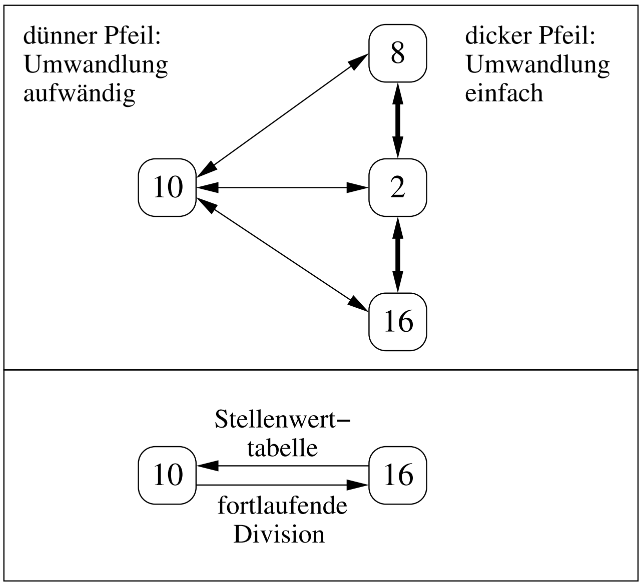


Abbildung 1: Übersicht Umwandlungen

```
1070:5 = 214R0 -----> 13240(zur Basis 5)
 214:5 = 42R4 |
  42:5 =  8R2 |
   8:5 =  1R3 |
   1:5 =  0R1 |
```

Auch hier wird das Ergebnis gebildet, indem man die Reste von unten nach oben einsammelt und aufschreibt. Das Ergebnis ist also $13240_{(5)}$.

### 1.2.13 Übersicht der Umwandlungen zwischen Stellenwertsystemen

Der obere Teil der Abbildung 1 zeigt, welche Umwandlungen aufwändig und welche einfach sind. Einfach sind Umwandlungen zwischen verwandten Systemen, z. B. zwischen Dual- und Oktal- oder Hexadezimalsystem: Man muss nur Bits gruppieren und in Ziffern wandeln und umgekehrt. Diese einfachen Umwandlungen sind durch dicke Pfeile gekennzeichnet.

Alle anderen Umwandlungen sind wesentlich aufwändiger und durch dünne Pfeile gekennzeichet. Bei der Umwandlung ins eigene Zahlensystem (Mensch: Basis 10) muss man eine Stellenwerttabelle bauen. Bei der Umwandlung in fremde Zahlensysteme (z. B. Basis 16) muss man fortlaufend dividieren (unten in Abbildung 1).