5 Zahlenformate und deren Grenzen

5.1 Erinnerung B-adische Zahlendarstellung

Stellenwertsystem: Jede Ziffer hat ihren Wert, und die Stelle der Ziffer in der Zahl modifiziert den Wert.

$$745 = 7*100+4*10+5 = (7*10+4)*10+5$$

Allgemeiner und formal:

$$(d_{-m}\dots d_{-1}d_0, d_1d_2d_3\dots)_{\mathfrak{B}}$$

$$= \sum_{k=-m}^{\infty} d_k\mathfrak{B}^{-k} = d_{-m}\mathfrak{B}^m+\dots+d_1\mathfrak{B}+d_0+\frac{1}{\mathfrak{B}}d_1+\frac{1}{\mathfrak{B}^2}d_2+\dots$$

Wie im Dezimalsystem haben ganze Zahlen keine Nachkommastellen, rationale Zahlen werden nach dem Komma periodisch oder brechen ab (Nullenperiode). Es gibt keine einfache Periode mit Ziffer $(\mathfrak{B}-1)$.

b

- $\mathfrak{B}$ ist **Basis** der Zahldarstellung, z.B. $\mathfrak{B} = 2, 8, 10, 12, 16, 1000, 1024$.
- $d_{-m}, \dots d_{-1}, d_0, d_1, d_2, \dots$ sind **Ziffern**, d.h., bewertete *Symbole* mit Werten $0, 1, \dots, \mathfrak{B}-1$.
- **Ziffernsymbole** sind $0, 1, \dots, 9$ wie üblich, dann $A = 10$, $B = 11$, …, $F = 15$;

  für $\mathfrak{B} \gg 16$ geklammerte Dezimalzahl als Symbol, $(1234)_{10} = ((12)(34))_{100}$.

5.1.1 1234 zur Basis 7

Fortlaufende Division mit Rest bis Quotient Null. Reste in umgekehrter Reihenfolge ergeben die Ziffern

$1234 \div 7 =$	176	Rest $d_0 = 2$	$1234 =$	$176*7+2$
$176 \div 7 =$	25	Rest $d_{-1} = 1$	$1234 =$	$(25*7+1)*7+2$
$25 \div 7 =$	3	Rest $d_{-2} = 4$	$1234 =$	$((3*7+4)*7+1)*7+2$
$3 \div 7 =$	0	Rest $d_{-3} = 3$		

Nach Ablesen der Reste von unten nach oben

$$(1234)_{10} = 3*7^3+4*7^2+1*7+2 = (3412)_7$$

5.1.2 5/7 zur Basis 3

$$\frac{5}{7} = \frac{1}{3} * \frac{15}{7} \qquad = \frac{1}{3} * \left(2 + \frac{1}{7}\right)$$
$$= \frac{1}{3} * \left(2 + \frac{1}{3^2} * \frac{9}{7}\right) = \frac{1}{3} * \left(2 + \frac{1}{3^2} * \left(1 + \frac{2}{7}\right)\right)$$
$$\ldots$$

Zähler	mal 3	div 7	Rest = neuer Zähler
5	15	2	1
1	3	0	3
3	9	1	2
2	6	0	6
6	18	2	4
4	12	1	5
5	15	2	1

$$\Longrightarrow \frac{5}{7} = (0, \overline{201021})_3$$

5.2 Binärdarstellung

Basis $\mathfrak{B} = 2$, Ziffern=Symbole=Bit $\{0, 1\}$

Rechenoperationen sind auf Bitebene durch Logikschaltungen realisierbar.

Wegen der festen Verdrahtung muss es eine fixierte maximale Stellenanzahl geben.

Beispiel mit 4 Bit, real sind es

8Bit für **char**,

16Bit für **short**,

32Bit für **int** und

64 Bit für **long**.

5.3 Binärdarstellung – ohne Vorzeichen

```
  0    1    2    3    4    5    6    7
---------------------------------------
0000 0001 0010 0011 0100 0101 0110 0111

1000 1001 1010 1011 1100 1101 1110 1111
---------------------------------------
  8    9   10   11   12   13   14   15
```

Rechenoperationen werden mod 2^n durchgeführt, auf Überlauf hat der Programmierer zu achten (Raketenexplosion).

5.4 Binärdarstellung – Vorzeichenbit

Reserviere das führende Bit zur Speicherung des Vorzeichens, $(n-1)$ Bit für den Betrag.

```
  0    1    2    3    4    5    6    7
---------------------------------------
0000 0001 0010 0011 0100 0101 0110 0111

1000 1001 1010 1011 1100 1101 1110 1111
---------------------------------------
  0   -1   -2   -3   -4   -5   -6   -7
```

Ergibt zwei Darstellungen der Null, aufwendige Rechenoperationen mit Abspalten des Vorzeichens und Fallunterscheidung.

5.5 Binärdarstellung – Einerkomplement

Vorzeichenwechsel erfolgt durch bitweises XOR, d.h. exklusives Oder oder Differenz (Komplement) zur Eins.

Das erste Bit signalisiert wieder das Vorzeichen, was folgende Zuordnung von Werten zu Bitmustern ergibt:.

```
  0    1    2    3    4    5    6    7
---------------------------------------
0000 0001 0010 0011 0100 0101 0110 0111

1111 1110 1101 1100 1011 1010 1001 1000
---------------------------------------
  0   -1   -2   -3   -4   -5   -6   -7
```

Wieder hat die Null zwei Darstellungen. Aber binäres Inkrement zählt überwiegend richtig mit Sprung von -15 bei der 7, $7+1 = 8 = -7+15$

Operationen mod $15 = 2^4 - 1$. So ist $16 \mod 15 = 1$, daher muss jeder Übertrag am Ende dazuaddiert werden.

```
Bsp.: -2*3 = (1101)*(0011)

       11010                              0111
     +  1101                            +   10
     --------   hat Übertrag (10)=2    --------
      100111                             1001
```

5.6 Binärdarstellung – Zweierkomplement

Eliminiere die doppelte Null durch Verschiebung der negativen Zahlen.

Vorzeichenwechsel durch bitweises XOR und nachfolgendes Inkrement, d.h. Addition von 1.

```
  0    1    2    3    4    5    6    7
-------------------------------------------
0000 0001 0010 0011 0100 0101 0110 0111

     1111 1110 1101 1100 1011 1010 1001 1000
-------------------------------------------
      -1   -2   -3   -4   -5   -6   -7   -8
```

0 und -8 haben sich selbst als Zweierkomplement.

Rechenoperationen wie mit Restklassen mod $16 = 2^n$, d.h. anfallende Überträge werden „vergessen“.

```
Bsp.: -2*-3 = (1110)*(1101)

        1110000
      +  111000
      +    1110          Kürzen auf letzte vier Stellen ergibt
     -----------
        1010110             (0110)=6
```

5.7 Gleitkommazahlen

Darstellung nicht–ganzer Zahlen. Es gibt höchstens 2^n verschiedene, wenn n die Bitbreite des Datentyps ist.

Wissenschaftliche Notation

Im Dezimalsystem $1234,5678$ für moderat große Zahlen

$1,234e56 = 1,234 * 10^{56}$ oder $9,68328e-23 = 9,68328 * 10^{-23}$ für sehr große oder kleine Zahlen.

Man nennt den Dezimalbruchfaktor **Mantisse** und die Zehnerpotenz **Exponent**.

Im Computer wird wieder die Basis $\mathfrak{B} = 2$ statt 10 verwendet.

5.8 Micro–Floats

Zur Verdeutlichung ein 6-Bit-Format, real 32 Bit für **single** und 64 für **double**

```
s aaa bb
```

- s ist Vorzeichenbit,
- aaa 3 Bit Exponent, Wert ist $(a_{-2}a_{-1}a_0)_2 - 3$,
- Die Exponenten 000 und 111 sind für Sonderfunktionen reserviert.
- bb ist normalisierte Mantisse mit Wert $(1,b_1b_2)_2$.
- zusammen ergibt dies den Wert $(-1)^s \cdot 2^{(aaa)_2-3}(1.bb)_2$

Von der größten zur kleinsten normalen Zahl:

s aaa bb	Wert
0 110 11	$+2^{6-3}\cdot\frac{7}{4}=$ 14
0 110 10	$+2^{6-3}\cdot\frac{3}{2}=$ 12
0 110 01	$+2^{6-3}\cdot\frac{5}{4}=$ 10
0 110 00	$+2^{6-3}\cdot 1=$ 8

s aaa bb	Wert
0 101 11	$+2^{5-3}\cdot\frac{7}{4}=$ 7
0 101 10	$+2^{5-3}\cdot\frac{3}{2}=$ 6
0 101 01	$+2^{5-3}\cdot\frac{5}{4}=$ 5
0 101 00	$+2^{5-3}\cdot 1=$ 4
...	

s aaa bb	Wert
...	
0 001 11	$+2^{1-3}\cdot\frac{7}{4}=\frac{7}{16}$
0 001 10	$+2^{1-3}\cdot\frac{3}{2}=\frac{3}{8}$
0 001 01	$+2^{1-3}\cdot\frac{5}{4}=\frac{5}{16}$
0 001 00	$+2^{1-3}\cdot 1=\frac{1}{4}$

Es gibt eine Lücke von 0 bis $\frac{1}{4}$, unmittelbar danach ist die Auflösung $\frac{1}{16}$. Füllen der Lücke mit nichtnormalisierten oder subnormalen Zahlen, der Exponent ist dabei 000, wird aber als 001 interpretiert. Mantisse wird als $(0,b_1b_2)_2$ interpretiert, also zusammen als $(-1)^s\cdot 2^{(001)_2-3}(0.bb)_2=(-1)^s\cdot 2^{(000)_2-3}(b.b)_2$. Der Exponent (111) wird für Fehlermeldungen verwendet.

0 000 11	$+2^{-2}\cdot\frac{3}{4}=\frac{3}{16}$
0 000 10	$+2^{-2}\cdot\frac{1}{2}=\frac{1}{8}$
0 000 01	$+2^{-2}\cdot\frac{1}{4}=\frac{1}{16}$
0 000 00	$+2^{-2}\cdot 0=+0$

0 111 00	$+\infty$
0 111 01	signalisierter Überlauf
0 111 1x	stiller Überlauf

5.9 Übliche IEEE-Formate

	Sign	Exponent	Mantisse	Bias	Dezimal	größte
nano-float	1	3	2	$2^2-1=3$	0-1	14
single, float	1	8	23	$2^7-1=127$	6-7	$\approx 10^{38}$
double	1	11	52	$2^{10}-1=1023$	15-16	$\approx 10^{308}$
extended	1	15	64	$2^{14}-1=4095$	18-19	$\approx 10^{1}232$

single und double sind exakt spezifiziert und funktionieren in vielen Prozessoren auf gleiche Weise.

extended ist ein Vorschlag für die prozessorinterne Realisierung der Gleitkommaarithmetik.