Zusatz-Scriptum zur Vorlesung Computersysteme WS 2018-2019

geschrieben von Prof. Dr. Manfred Schimmler
überarbeitet und gehalten von Prof. Dr.-Ing. Reinhard Koch

Multimediale Informationsverarbeitung

Institut für Informatik

Technische Fakultät

Universität Kiel

Inhaltsverzeichnis

1 Zahlendarstellung in Computern

Empfohlene Literatur: Klar, R.: Digitale Rechenautomaten, de Gruyter ISBN 3 11 0041944

1.1 Darstellung natürlicher Zahlen im B-adischen Zahlensystem

Im normalen Leben werden Zahlen in der Regel im Dezimalsystem dargestellt. Die natürliche Zahl 201 steht für die Summe

$$\mathbf{2}*10^2 + \mathbf{0}*10^1 + \mathbf{1}*10^0$$

Die Ziffern geben also die Koeffizienten eines Polynoms in 10 an, wobei jede Stelle für eine Potenz des Basiswerts 10 steht.

Das Dezimalsystem ist nicht besonders gut geeignet für elektronische Rechenanlagen, weil es technisch sehr schwierig ist, die zehn unterschiedlichen Ziffern in zehn Wertigkeiten einer physikalischen Größe zu codieren, so daß dieser Code

1. eindeutig ist (d.h. daß man immer weiß, welche Ziffer gemeint ist) und
2. nicht durch Störeinflüsse wie Übersprechen auf Leitungen verfälscht werden kann.

Daher bedient man sich in Computern in der Regel des dualen (binären) oder 2-adischen Zahlensystems. Hier werden die natürlichen Zahlen als Polynome in 2 betrachtet, wobei als Koeffizienten nur die Ziffern 0 und 1 erforderlich sind. Die Dezimalzahl 201 hat dann die Repräsentation 11001001, die zu verstehen ist als

$$\mathbf{1}*2^7 + \mathbf{1}*2^6 + \mathbf{0}*2^5 + \mathbf{0}*2^4 + \mathbf{1}*2^3 + \mathbf{0}*2^2 + \mathbf{0}*2^1 + \mathbf{1}*2^0.$$

Die Beschränkung auf zwei Ziffern erlaubt es, die Koeffizienten durch eine Größe zu repräsentieren, von der nur zwei Werte unterschieden werden müssen, z.B. eine Spannung, wobei 0V für die 0 und 3,3V für die 1 steht. Oder durch eine Lampe, bei der eine 1 durch Leuchten der Lampe und eine 0 durch Nicht-Leuchten codiert ist.

Nun gibt es aber nicht nur diese beiden Systeme zur Repräsentation von Zahlen. So wie wir bisher 10 und 2 als Basis des Zahlensystems verwendet haben, kann man jede positive ganze Zahl B≥2 als Basis eines sogenannten B-adischen Zahlensystems wählen. Eine beliebige Zahl n kann dann eindeutig repräsentiert werden als

$$n = \sum_{i=0}^{N-1} b_i B^i = b_0 B^0 + b_1 B^1 + \cdots + b_{N-1} B^{N-1}$$

Dabei sind die Ziffern $b_i \in \{0,1,2,...,B-1\}$. N ist die Anzahl der Stellen für die Darstellung. Da wir in Computern in der Regel ein festes „Wortformat“ zur Verfügung haben, ist damit für N eine obere Grenze gegeben. Wenn wir z.B. von einer „32-Bit-Architektur“ reden, bedeutet, das, dass die Zahlen mit 32 (binären, d.h. B = 2) Stellen dargestellt werden, also N = 32.

Mit B = 10 erhält man das Dezimalsystem, mit B = 2 das Binär- oder Dualsystem.

Zur Vereinfachung ist folgende Konvention in der Schreibweise üblich:

$$n = (\boldsymbol{b_{N-1}\, b_{N-2} \ldots b_1\, b_0})_B$$

wobei führende Ziffern weggelassen werden, wenn sie 0 sind und die Kennzeichnung durch das tiefgestellte B am Ende wegfällt, wenn man weiß, über welche Basis man redet.

Beispiele:

Wir hatten die Zahl $(201)_{10}$ bereits als $(11001001)_2$ kennengelernt. Mit der Basis 8 hat sie folgende Repräsentation:

$$\mathbf{0}*8^{N-1} + ... + \mathbf{0}*8^4 + \mathbf{0}*8^3 + \mathbf{3}*8^2 + \mathbf{1}*8^1 + \mathbf{1}*8^0 = (311)_8.$$

Das 8-adische Zahlensystem wird auch als Oktalsystem bezeichnet.

Wählen wir als Basis die 16, so begeben wir uns ins Hexadezimalsystem. Hier wird aus der Dezimalzahl 201

$$\mathbf{0}*16^{N-1} + ... + \mathbf{0}*16^3 + \mathbf{0}*16^2 + \mathbf{12}*16^1 + \mathbf{9}*16^0 = (C9)_{16}$$

Da wir für die Zahlen „zehn“, „elf“, „zwölf“, „dreizehn“, „vierzehn“ und „fünfzehn“ keine Ziffern zur Verfügung haben, behelfen wir uns mit den Großbuchstaben A bis F, wobei

A für 10

B für 11

C für 12

D für 13

E für 14

F für 15

steht. Die Zahl $(45054)_{10}$ beispielsweise hat die hexadezimale Repräsentation AFFE.

Merkmale eines B-adischen Zahlensystems:

- Es gibt B Ziffern 0, 1, ..., (B-1).
- Jede Stelle hat ein Gewicht einer Potenz von B.
- Das Gewicht der Stelle i ist das B-Fache des Gewichts der Stelle i-1.

Satz:

Die N-stellige B-adische Darstellung ermöglicht es, jede ganze Zahl aus $\{0,1,...,B^N-1\}$ auf genau eine Weise darzustellen.

Beweis:

„Jede Zahl kann dargestellt werden“ wird bewiesen mit vollständiger Induktion nach N.

Induktionsanfang: Sei N=1. Die Zahlen 0, 1, ..., B-1 sind genau durch die B-adischen Ziffern als 1-stellige Zahlen darstellbar.

Sei die Aussage des Satzes richtig für N = k.

Dann können die Zahlen 0, 1, ..., B^k-1 dargestellt werden als k-stellige B-adische Zahlen. Die k+1-stelligen Zahlen, die mit einer 0 beginnen, decken also diesen Bereich ab. Die k+1-stellige Zahl mit einer Ziffer z am Anfang gefolgt von Nullen hat den Wert zB^k.

Sei m eine Zahl aus $\{0, 1, ..., B^{k+1}-1\}$. Sei $m = q * B^k + r$, mit ganzzahligen und positiven Werten q und r, wobei $r<B^k$ sein soll. Dann ist q darstellbar als die Ziffer q gefolgt von k Nullen und r ist nach Induktionsvoraussetzung darstellbar als eine k+1-stellige Zahl, die mit einer 0 beginnt. Das Ersetzen der ersten 0 von r durch q liefert die Darstellung von m.

Zu zeigen bleibt, daß die Darstellung eindeutig ist. Nun hat die Menge $\{0, 1, ..., B^N-1\}$ genau B^N Elemente. Andererseits stehen B^N N-stellige Zeichenreihen mit den Ziffern {0, 1, ..., B-1}

zur Verfügung. Da jede Zahl dargestellt werden kann ist also keine Zeichenreihe übrig, um eine Zahl doppelt darzustellen. Also ist die Darstellung eindeutig. □

Beispiele für B-adische Zahlen (B = 2, 3, 8, 10, 16):

Binär 2-adisch	Ternär 3-adisch	Oktal 8-adisch	Dezimal 10-adisch	Hexadezimal 16-adisch
0	0	0	0	0
1	1	1	1	1
10	2	2	2	2
11	10	3	3	3
100	11	4	4	4
101	12	5	5	5
110	20	6	6	6
111	21	7	7	7
1000	22	10	8	8
1001	100	11	9	9
1010	101	12	10	A
1011	102	13	11	B
1100	110	14	12	C
1101	111	15	13	D
1110	112	16	14	E
1111	120	17	15	F
10000	121	20	16	10

1.2 Rechnen im B-adischen Zahlensystem

In jedem B-adischen Zahlensystem können wir ähnlich wie im Dezimalsystem rechnen. Man muß aber dabei aufpassen daß man nicht (implizit) Ziffern benutzt, die in dem jeweiligen System nicht vorhanden sind.

Beispiele:

Im Oktalsystem sind die Zahlen 7351 und 642 zu addieren:

```
          7351
           642
Übertrag  11100
         10213
```

Im Dualsystem sollen 1101000101 und 10011010111 addiert werden.

```
            1101000101
           10011010111
Übertrag  111110001110
         100000011100
```

Im Hexadezimalsystem sollen CAD von 1234 subtrahiert werden:

```
  1 2 3 4      4 – D => Übertrag: 14 - D =  7 (Übertrag 1, Ziffer 7)
    CAD        3 – (A+1) = 3 - B => Übertrag: 13 - B = 8 (Übertrag 1, Ziffer 8)
Übertrag 1 1 1 0   2 - (C+1) = 2 - D => Übertrag 12 – D = 5 (Übertrag 1, Ziffer 5)
    5 8 7      1 – (0+1) = 0, Ende der Rechnung
```

Merke: Subtraktion ist schwieriger als Addition, immer zum nächsten Übertrag rechnen

Multiplikation der Zahl $21D5_{16}$ mit der Zahl 6_{16}:

```
  21 D5  *  6
  CAFE
```

Denn 6*5 = 1E (E schreib hin, 1 im Sinn), 6*D = 4E (+1 macht F, F schreib hin, 4 im Sinn), 6*1 = 6 (+4 macht A, A schreib hin, 0 im Sinn) und 6*2 = C.

Division der Zahl $6402_8 : 12_8$

Aufstellen einer Tabelle der Vielfachen des Divisors $12_8 = 10_{10}$ mit b=8 ist sehr nützlich:

i_8	$i*12_8$	Dezimal
1	12	10
2	24	20
3	36	30
4	50	40
5	62	50
6	74	60
7	106	70
10	120	80

Berechne durch schriftliche Division mit b=8: 6402:12=515

62	5*12=62
20	Rest 2, hole nächste Ziffer 0
12	1*12
62	Rest 6, hole nächste Ziffer 2
62	5*12=62
0	Rest 0, Ende der Division

Darstellung negativer Zahlen

Wir sind gewohnt, Zahlen durch Betrag und Vorzeichen anzugeben. In Rechenanlagen hat diese Technik zwei wesentliche Nachteile:

1. Die Subtraktion muss durch eine eigenständige Einheit ausgeführt werden.
2. Die Entscheidung, ob ein Ergebnis größer, gleich oder kleiner als 0 ist, ist für den Rechner schwierig.

Eine wesentlich elegantere Lösung ist die Darstellung von negativen Zahlen im Komplement. Bei der B-adischen Darstellung unterscheidet man das B-Komplement und das (B-1)-Komplement.

1.2.1 B-Komplement

Definition:

Sei n eine natürliche Zahl, dargestellt als ***N-stellige B-adische Zahl***. Das B-Komplement von n ist die Zahl B^N-n. Das B-Komplement von n wird interpretiert als -n.

Auf diese Weise werden bei ungradzahligem B alle Zahlen von 000...01 bis zzz...z, wobei z die Ziffer des Wertes (B-1)/2 ist zu positiven Zahlen und alle Zahlen von zzz...(z+1) bis fff...f, wobei f die Ziffer (B-1) ist zu negativen Zahlen. Die 0 ist dargestellt als 000...0.

Bei gradzahligem B sind 000...01 bis (B/2-1)FFF...F positiv und B/2 00...0 bis FFF...F negativ.

Beispiele:

Im Dezimalsystem mit Zahlen der Länge 5 können die Zahlen von -50000 bis +49999 unter Benutzung des 10-Komplements dargestellt werden:

Die Zahlen von 0 (00000) bis 49999 haben Ihre normale Interpretation, so wie wir sie gewöhnt sind. Die -1 wird jedoch dargestellt als 99999, die -2 als 99998, die -35971 als 64029 und die -50000 als 50000.

Wir betrachten die 8-stelligen Dualzahlen mit negativen Zahlen im 2-Komplement. Mit diesen können wir den Zahlenbereich von -128 bis +127 darstellen.

$$00000000 = (0)_{10}$$

$$01010101 = (85)_{10}$$

$$01111111 = (127)_{10}$$

$$10000000 = (-128)_{10}$$

$$11100110 = (-26)_{10}$$

$$11111111 = (-1)_{10}$$

Dezimal	4-stellige 2-Komplementzahlen
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Was ist der Vorteil vom B-Komplement?

Innerhalb unseres vorgegebenen Bereichs darstellbarer Zahlen können wir jetzt eine Subtraktion der Zahl Z_2 von der Zahl Z_1 einfach als Addition von Z_1 mit dem B-Komplement von Z_2 durchführen.

Beispiele:

Alle folgenden Additionen und Subtraktionen werden mit 8-stelligen Dualzahlen durchgeführt, wobei negative Zahlen im 2-Komplement dargestellt werden:

$103_{10} - 70_{10}$:

$$\begin{array}{lll} & 01100111 & (103)_{10} \\ - & 01000110 & -(70)_{10} \\ \hline = & 01100111 & (103)_{10} \\ + & 10111010 & +(-70)_{10} \\ \hline & 1|00100001 & (33)_{10} \end{array}$$

15_{10} - 70_{10} :

		00001111	$(15)_{10}$
	-	01000110	$-(70)_{10}$
=		00001111	$(15)_{10}$
	+	10111010	$+(-70)_{10}$
		0|11001001	$(-55)_{10}$

-15_{10} - (-70_{10}) :

		11110001	$(-15)_{10}$
	-	10111010	$-(-70)_{10}$
=		11110001	$(-15)_{10}$
	+	01000110	$+(+70)_{10}$
		1|00110111	$(+55)_{10}$

Wie konvertiere ich eine Zahl in ihr B-Komplement?

Alle Ziffern z werden in (B-1-z) umgewandelt und dann wird auf die ganze Zahl 1 addiert.

Begründung:

$$-(b_{N-1}*B^{N-1} + b_{N-2}*B^{N-2} + b_{N-3}*B^{N-3} + ... + b_1*B^1 + b_0*B^0)$$

$$= B^N - (b_{N-1}*B^{N-1} + b_{N-2}*B^{N-2} + b_{N-3}*B^{N-3} + ... + b_1*B^1 + b_0*B^0)$$

$$= B^N - B^{N-1} - b_{N-1}*B^{N-1} + B^{N-1} - B^{N-2} - b_{N-2}*B^{N-2} + B^{N-2} - B^{N-3} - b_{N-3}*B^{N-3} + B^{N-3} - ... - B^1 - b_1*B^1 + B^1 - 1 - b_0*B^0 + 1$$

$$= (B - 1 - b_{N-1})*B^{N-1} + (B - 1 - b_{N-2})*B^{N-2} + (B - 1 - b_{N-3})*B^{N-3} + ... + (B - 1 - b_1)*B^1 + (B^1 - 1 - b_0)*B^0 + 1$$

Dies ist gerade die Darstellung der B-Komplement-Zahl in der oben angegebenen Form: jede Ziffer $\mathbf{b_i}$ ist durch (B -1 - $\mathbf{b_i}$) ersetzt worden und am Schluss wird zu allem 1 addiert.

Beispiele:

$-(107)_{10}$ als zweistellige Hexadezimalzahl:

$$(+107)_{10} = (6B)_{16}$$

$$(-107)_{10} = (94)_{16} + (1)_{16} = (95)_{16}$$

-107_{10} als Achtstellige Dualzahl:

$$(+107)_{10} = (01101011)_2$$

$$(-107)_{10} = (10010100)_2 + (1)_2 = (10010101)_2$$

Vorteile des B-Komplements:

- Addition und Subtraktion können mit derselben Hardware gemacht werden.
- Konversion ist einfach.
- Positive und negative Zahlen können gleich behandelt werden.
- Das Vorzeichen ist bei geradem B an der ersten Stelle (most significant digit) zu erkennen.

Nachteil: Überläufe werden nicht automatisch erkannt. Konvertierung erfordert ein „carry“

Beispiel:

$(11)_{10} + (10)_{10}$ im System der fünfstelligen Dualzahlen:

```
    01011
+   01010
=   0|10101
```

Das Ergebnis wird interpretiert als -11_{10}. Das ist ja falsch.

Allerdings kann man solche Überläufe erkennen, wenn man eine weitere (führende) Stelle vor der signifikantesten Stelle des Ergebnisses einführt.

1. Fall: Wenn eine positive und eine negative Zahl addiert werden, kann nie ein Übertrag auftreten.
2. Wenn zwei positive oder zwei negative Zahlen addiert werden, und die zusätzliche führende Stelle nach der Addition (Subtraktion) mit der signifikantesten Stelle übereinstimmt, ist kein Überlauf aufgetreten. Unterscheiden sie sich, ist ein Überlauf aufgetreten. Und zwar wenn die Stellen 01 sind, eine nicht darstellbare positive Zahl und wenn sie 10 sind ein betragsmäßig zu großes negatives Ergebnis.

Beispiele:

Das Beispiel von eben: $(11)_{10} + (10)_{10}$ im System der fünfstelligen Dualzahlen:

```
Sicherungsstelle   ↓     (Kopie der signifikantesten Ziffer)
                   001011
+                  001010
=                  0|10101
```

Sicherungsstelle der Summe = 0, MSB (most signifikant Bit) der Summe = 1, also Überlauf: nicht darstellbare positive Zahl als Ergebnis.

$10_{10} - 11_{10}$ im System der fünfstelligen Dualzahlen:

Sicherungsstelle ↓ (Kopie der signifikantesten Ziffer)

```
    001010
+   110101
=   1|11111
```

Sicherungsstelle der Summe = 1, MSB (most signifikant Bit) der Summe = 1, also kein Überlauf und kein Unterlauf: darstellbare (negative) Zahl als Ergebnis (-1).

Satz:

Genau dann ist bei Addition zweier N-stelliger 2-adischer Zahlen das Ergebnis wieder im (mit N Ziffern) darstellbaren Bereich, wenn nach der Addition die Vorzeichenstelle (Stelle N-1) mit der Sicherungsstelle (Stelle N) übereinstimmt.

Beweis:

Zu unterscheiden sind folgende 6 Fälle:

1. $n_1 \geq 0$, $n_2 \geq 0$, $n_1+n_2 \geq 2^{N-1}$
2. $n_1 \geq 0$, $n_2 \geq 0$, $n_1+n_2 < 2^{N-1}$
3. $n_1 < 0$, $n_2 \geq 0$
4. $n_1 \geq 0$, $n_2 < 0$
5. $n_1 < 0$, $n_2 < 0$, $n_1+n_2 \geq -2^{N-1}$
6. $n_1 < 0$, $n_2 < 0$, $n_1+n_2 < -2^{N-1}$

Im Fall 1. beginnen beide Zahlen mit einer 0; das MSB sowie die Sicherungsstelle sind also 0. Da $n_1+n_2 \geq 2^{N-1}$ ist, entsteht bei der Addition an der Stelle N eine 1 als Übertrag:

```
00XXX.....XXX   n1
00XXX.....XXX   n2
01XXX.....XXX
↑ Stelle N
```

Also ist die Sicherungsstelle der Summe = 0, die Stelle N = 1, d.h. ein Überlauf ist aufgetreten.

Betrachten wir noch den Fall 3. Hier ist $n_1 < 0$, $n_2 \geq 0$, also

```
11XXX.....XXX   n1
00XXX.....XXX   n2
  XXX.....XXX
↑ Stelle N
```

An der Stelle N kann ein Übertrag auftreten oder nicht. Im erste Fall sind die beiden ersten Stellen der Summe = 0, im zweiten sind sie beide = 1. Die Stelle vor der Sicherungsstelle wird

nicht betrachtet. In beiden Fällen sind die beiden höchsten Stellen gleich, d.h. es ist kein Überlauf und kein Unterlauf aufgetreten.

Der Leser möge sich selbst die anderen vier Fälle überlegen.

1.3 Konvertierung natürlicher Zahlen aus dem B-adischen Zahlensystemen ins B'-adische Zahlensystem mit unterschiedlichen Basen B und B'

Eine im B-adischen Zahlensystem dargestellte Zahl

$$n = \boldsymbol{b_N\, b_{N-1}\, b_{N-2} \ldots b_1\, b_0} = \boldsymbol{b_N} * B^N + \boldsymbol{b_{N-1}} * B^{N-1} + \ldots + \boldsymbol{b_1} * B^1 + \boldsymbol{b_0} * B^0$$

kann mit dem Hornerschema auch in folgender Weise geschrieben werden.

$$n = ((\ldots((\boldsymbol{b_N} * B + \boldsymbol{b_{N-1}}) * B + \boldsymbol{b_{N-2}}) * B + \ldots) * B + \boldsymbol{b_1}) * B + \boldsymbol{b_0}.$$

Ebenso hat n eine gleichartige Repräsentation im System mit der Basis B':

$$n = ((\ldots((\boldsymbol{b_N'} * B' + \boldsymbol{b_{N-1}'}) * B' + \boldsymbol{b_{N-2}'}) * B' + \ldots) * B' + \boldsymbol{b_1'}) * B' + \boldsymbol{b_0'}.$$

Um nun die b_i' aus der ersten Repräsentation zu berechnen, können wir durch wiederholte Division durch B' die Reste ermitteln, die dann genau den Ziffern im B'-adischen System entsprechen. Diese Division muß im B-adischen System durchgeführt werden, da wir ja anfangs nur die B-adische Darstellung von n kennen. Daher brauchen wir die B-adische Darstellung von B' für die Division.

Das Divisionsschema sieht dann so aus:

n	:	B'	=	q_0	Rest	$\boldsymbol{b_0'}$
q_0	:	B'	=	q_1	Rest	$\boldsymbol{b_1'}$
q_1	:	B'	=	q_2	Rest	$\boldsymbol{b_2'}$
		.	.	.		
q_{N-1}	:	B'	=	q_N	Rest	$\boldsymbol{b_N'}$

Und das Ergebnis wird durch die Folge der Reste $\boldsymbol{b_N'\ b_{N-1}'\ b_{N-2}\ \ldots\ b_0'}$ geliefert.

Beispiele:

Die Zahl 556_7 soll ins 3-adische System (Ternärsystem) umgewandelt werden.

556_7 : 3_7 = 164_7 Rest 1
3
25
24
16
15
1

164_7 : 3_7 = 43_7 Rest 2
15
14
12
2

43_7 : 3_7 = 13_7 Rest 1
3
13
12
1

13_7 : 3_7 = 3_7 Rest 1
$\underline{12}$
1

3_7 : 3_7 = 1_7 Rest 0
$\underline{3}$
0

1_7 : 3_7 = 0_7 Rest 1

Ergebnis $556_7 = 101121_3$

Die Zahl 556_7 soll ins Oktalsystem umgewandelt werden.

556_7 : 11_7 = 50_7 Rest 6
$\underline{55}$
06
$\underline{00}$
6

50_7 : 11_7 = 4_7 Rest 3
$\underline{44}$
3

4_7 : 11_7 = 0_7 Rest 4

Ergebnis $556_7 = 436_8$

Eine zweite Möglichkeit der Konvertierung ist die Abarbeitung des Hornerschemas

$$n = ((...((\mathbf{b}_N*B + \mathbf{b}_{N-1})*B + \mathbf{b}_{N-2})*B + ...)*B + \mathbf{b}_1)*B + \mathbf{b}_0$$

von links nach rechts. In diesem Falle müssen alle $\mathbf{b}_N$ sowie die Basis B in der Formel zunächst ins B'-System umgewandelt werden. Sodann kann die Formel wie üblich nach den Rechenregeln für + und * im B'-adischen System von links nach rechts ausgerechnet werden.

Beispiele:

Umwandlung der Zahl 110010101_2 ins Dezimalsystem:

n = (((((((1*2+1)*2+0)*2+0)*2+1)*2+0)*2+1)*2+0)*2+1 = 405

Umwandlung von 365_{10} ins Oktalsystem:

n = (3*12+6)*12+5 = (36+6)*12+5 = 44*12+5 = 550+5 = 555

Man beachte: die 10 ist im Oktalsystem als 12 dargestellt.

Bemerkung:

Die Konvertierung durch sukzessive Division empfiehlt sich, wenn die Arithmetik in dem Ausgangssystem einfacher ist und die Konvertierung durch Multiplikation und Addition ist einfacher, wenn die Arithmetik im Zielsystem besser beherrscht wird.

1.4 Brüche in Festkommadarstellung

Bisher haben wir uns nur mit ganzen Zahlen beschäftigt. Aber alle Ergebnisse, insbesondere die über die B-adische Zahlendarstellung und über die Komplementbildung können auch auf gebrochene Zahlen angewendet werden. Wir ergänzen dafür die Definition der B-adischen Darstellung in folgender Weise:

$$n = \sum_{i=M}^{N-1} b_i B^i$$

Zahlendarstellungen dieser Art nennt man Festkommadarstellungen, weil fest steht, dassdas Komma nach der N-ten Ziffer kommt. Für die Addition und Subtraktion bleibt alles bisher Gelernte gültig.

Bei der Multiplikation von ganzen Zahlen gilt: Das Produkt zweier N-stelligen ergibt eine 2N-stellige Zahl. Da die Größe der darstellbaren Zahlen in Computern in der Regel vorgegeben ist (typisch 16, 32 oder 64 Stellen) wählen wir als Ergebnis nur wieder die niederwertigsten N Stellen. Wenn eine der höherwertigen Stellen eine Ziffer ≠ 0 enthält, ist ein Überlauf aufgetreten, der über einen gesonderten Mechanismus abzufangen ist.

Für Festkommazahlen wählt man als Bezugspunkt die Stelle, an der das Komma steht. Man wählt also vom doppeltlangen Ergebnis wiederum N Stellen vor und M Stellen nach dem Komma. Alle anderen Stellen werden verworfen. Das kann genauso zu Überläufen führen wie bei der ganzzahligen Multiplikation. Zusätzlich können aber auch Nachkommastellen ausgelöscht werden, die zu weit „hinter“ dem Komma stehen, wodurch ein Teil der Genauigkeit verloren gehen kann.

1.4.1.1 Konversion von Festkommazahlen

Die Konversion von Festkommazahlen in einem B-adischen System in ein B’-adisches System kann wieder durch Auflösen des Horner-Schemas geschehen.

$$n = (b_{-1}b_{-2}\ldots b_{-M})_B = ((\ldots(b_{-M}*B^{-1}+b_{-M+1})*B^{-1}+b_{-M+2})*B^{-1}+\ldots)*B^{-1}+b_{-1})*B^{-1}$$

Wir betrachten jetzt nur den gebrochenen Teil, denn wir wissen bereits wie wir den ganzzahligen Teil konvertieren können. Wenn man im B-adischen System leichter rechnen kann als im B’-adischen, empfiehlt sich folgendes Verfahren, das man „Wiederholte Multiplikation mit abschneiden“ nennt:

Man multipliziert die zu konvertierende Zahl mit der Basis B’. Die Stelle, die dabei vor das Komma gerät, ist die erste Ziffer der B’-adischen Darstellung. Diese subtrahiert man vom Ergebnis und bekommt wieder einen B-adischen Bruch mit einer 0 vor dem Komma. Mit diesem verfährt man genauso, usw.

Beispiel:

0,75625 soll ins Binärsystem gewandelt werden:

0,75625 * 2 = 1,5125	⇒	erste Ziffer ist 1. Diese abziehen ergibt 0,5125
0,51250 * 2 = 1,025	⇒	nächste Ziffer ist 1. Diese abziehen ergibt 0,025
0,02500 * 2 = 0,05	⇒	nächste Ziffer ist 0. Diese abziehen ergibt 0,05
0,05 * 2 = 0,1	⇒	nächste Ziffer ist 0. Diese abziehen ergibt 0,1
0,1 * 2 = 0,2	⇒	nächste Ziffer ist 0. Diese abziehen ergibt 0,2
0,2 * 2 = 0,4	⇒	nächste Ziffer ist 0. Diese abziehen ergibt 0,4

0,4 * 2 = 0,8 ⇒ nächste Ziffer ist 0. Diese abziehen ergibt 0,8

0,8 * 2 = 1,6 ⇒ nächste Ziffer ist 1. Diese abziehen ergibt 0,6

0,6 * 2 = 1,2 ⇒ nächste Ziffer ist 1. Diese abziehen ergibt 0,2

0,2 * 2 = 0,4 ⇒ nächste Ziffer ist 0 usw.

Die Binärdarstellung lautet also 0,11000001100110011….

Die zweite Möglichkeit besteht in der rechnerischen Auflösung des Horner-Schemas. Diese empfiehlt sich, wenn man im Zielsystem rechnen möchte. In diesem Falle konvertiert man die Basis B sowie alle Ziffern in der Quelldarstellung zunächst ins B'-adische System und rechnet dann die oben angegebene Form der Zahl durch die angegebenen Additionen und Multiplikationen im B'-adischen System aus.

Beispiele:

1. Die Hexadezimalzahl 0,3D5 soll ins Dezimalsystem gewandelt werden:

$$((5*16^{-1}+13)*16^{-1}+3)*16^{-1} = 0{,}239501953125$$

2. Die Dezimalzahl 100,92 soll ins Binärsystem gewandelt werden:

   100,92 (mit 7 Stellen vor und 5 Stellen nach dem Komma):

   a) 100:2=50 Rest 0
      50:2=25 Rest 0
      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
      100=1100100

   b) 0,92:
      0,92*2 = 1,84
      0,84*2 = 1,68
      0,68*2 = 1,36
      0,36*2 = 0,72
      0,72*2 = 1,44
      0,44*2 = 0,88 (nur erforderlich für die Rundung)
      0,92=0,11101+0=0,11101
      100,92=1100100,11101

Wenn B eine Potenz von B' ist, ist es einfach, zwischen B-adisch und B'-adisch zu konvertieren. Sei z.B. B = 16 und B' = 2. Da $16 = 2^4$ ist, bestehen alle Ziffern im B'-adischen System aus vier B'-adischen Ziffern, z.B. 4 = 0100 oder C = 1100. Auf diese Weise kann man einfach Ziffernweise konvertieren, wobei man im B'-adischen System immer Viererblöcke aus Ziffern zusammenfassen muss.

Übung: Wie kann man einfach vom Oktalsystem ins Hexadezimalsystem konvertieren?

1.5 Gleitkommadarstellung

Im technischen und wissenschaftlichen Bereich bedient man sich neben der Festkommadarstellung von Brüchen auch der Exponentenschreibweise.

Beispiel

$$3456{,}5 = 0{,}34565*10^4 = 0{,}34565E{+}4$$

Der Vorteil ist, dass man auf kurze und übersichtliche Weise einen sehr großen Zahlenbereich darstellen kann. Diese Darstellungsweise hat auch enorme Vorteile für die Zahlendarstellung in Computern: Hier ist man durch die Hardware auf eine oder wenige Wortgrößen (Anzahl von Bits) festgelegt, in denen ein Operand untergebracht werden muss. Würde nun nur die normale Dualdarstellung verwendet, könnte man bereits die Zahl 10 Milliarden in einem 32-Bit Wort nicht mehr darstellen. Wenn man andererseits auch gebrochene Zahlen in Festkommadarstellung zulassen möchte und jeweils 16 Bit vor und 16 Bit nach dem Komma verwendet, scheitert man bereits an der Zahl 100000. Der Bereich darstellbarer Zahlen ist in diesem Falle $[-2^{15} .. +2^{15}-2^{-16}]$.

In Computern werden binäre Gleitkommaformate verwendet. In einem Register werden drei Teile einer Zahl nacheinander in einer festen Anzahl von Bits gespeichert: Das Vorzeichen V der Zahl, die Mantisse und der Exponent.

V	Exponent	Mantisse

Der Wert der Gleitkommazahl ist im einfachsten Falle

$$n = Vorzeichen * 0{,}Mantisse * 2^{Exponent}$$

wobei das Vorzeichen +1 ist, wenn V=0 ist und -1, wenn V=1 ist. Dabei wird die Mantisse als positive Dualzahl und der Exponent als Dualzahl in Zweierkomplementdarstellung interpretiert.

Beispiel:

Wir definieren ein fiktives binäres Gleitkommaformat mit 16 Bit: 1 Bit Vorzeichen, 10 Bit Mantisse und 5 Bit Exponent. Die Zahl

1000110110000000 bedeutet $-0{,}011_2*10_2^{11_2} = -0{,}375_{10}*2_{10}^3 = -3$

0111111110000000 bedeutet $+0{,}111_2*10_2^{-1} = +0{,}875_{10}*2_{10}^{-1}=+0{,}4375$

0000011000000000 bedeutet $+0{,}1_2*10_2^1 = 1$

0000000000000000 bedeutet 0

Der darstellbare Zahlenbereich für dieses Format ist bereits

$$-2^{2^4-1}+2^5...+2^{2^4-1}-2^5 \approx -2^{15}...+2^{15}$$

was mit einem Festkommaformat von 16 Bit nur dann erreicht werden kann, wenn man keine Nachkommastellen zulässt.

1.5.1 Arithmetik mit Gleitkommazahlen

Die Multiplikation von Gleitkommazahlen ist einfach, denn

$$m_1 * 2^{e_1} * m_2 * 2^{e_2} = m_1 * m_2 * 2^{e_1+e_2}$$

d. h. die Mantissen können als normale Dualzahlen multipliziert werden und die Exponenten addiert. Das gleiche gilt für die Division. Schwieriger ist dagegen die Addition: Hier muss zunächst dafür gesorgt werden, dass die Exponenten angeglichen werden, denn eine Addition der Mantissen kann nur dann richtig ausgeführt werden, wenn die Exponenten gleich sind:

$$m_1*2^{e_1}+m_2*2^{e_1}=(m_1+m_2)*2^{e_1}$$

Zu diesem Zweck muss zunächst ermittelt werden, welcher Exponent der größere ist und die Exponentendifferenz d wird mittels einer Subtraktion gebildet. Sodann wird die Mantisse der Zahl mit dem kleineren Exponenten um d Stellen nach rechts verschoben, wobei von links Nullen nachgezogen werden. Dies entspricht einer Division der Mantisse durch 2^d bei gleichzeitiger Vergrößerung des Exponenten um d. Nun kann die Addition auf den angepassten Mantissen durchgeführt werden, wobei als Exponent des Ergebnisses der größere Exponent der Operanden wird. Durch die Addition kann es passieren, dass im Ergebnis eine Folge von führenden Nullen entsteht. Um aber für nachfolgende Operationen die Genauigkeit nicht einzuschränken, wird die Mantisse des Ergebnisses nun wieder nach links verschoben, bis die erste signifikante Stelle eine 1 ist. Wenn die Verschiebedistanz d' ist, muss schließlich der Exponent noch um d' vermindert werden, damit der Wert des Ergebnisses nicht verändert wird.

Beispiele:

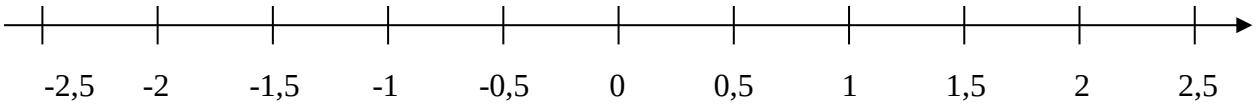
1. In unserem obigen Gleitkommaformat wollen wir 4*0,2 rechnen:
   0000111000000000 * 0111101100110011
   Mantissenmultiplikation: 0,1*0,1100110011=0,01100110011
   Exponentenaddition: 00011+11110=100001, wird interpretiert als 00001
   Normalisierung macht daraus 0,1100110011 * 2^0.
   Ergebnis: 0000001100110011 (= 0,799804...)

2. Was ist 1+0,1?
   0000011000000000 + 0111011100110011
   Die Exponentendifferenz ist 00001 - 11101 = 00100 $(4)_{10}$
   Die Mantisse mit dem größeren Exponenten ist 1000000000
   Die andere Mantisse um 4 Stellen verschoben ist 0000110011
   Die Mantissensumme ist 1000110011
   und wir bekommen als Ergebnis 0000011000110011 (=1,099609...)

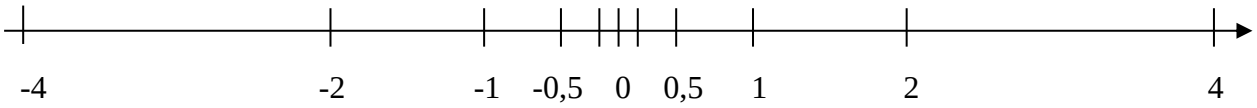
Bemerkungen:

Mit Gleitkommazahlen kann man einen größeren Zahlenbereich abdecken als mit Festkommazahlen mit gleich vielen Stellen. Während beim Festkommaformat zwischen je zwei benachbarten Zahlen der gleiche Abstand liegt, variiert dieser bei Gleitkommazahlen. In der Gegend der Null liegen sie sehr dicht, während sie zu den Grenzen des darstellbaren Zahlenbereichs hin immer weiter auseinanderliegen:

Festkommazahlen:



Gleitkommazahlen:



Multiplikation von Gleitkommazahlen ist aufwendiger als von Festkommazahlen, Addition von Gleitkommazahlen ist viel aufwendiger als bei Festkommazahlen.

1.5.2 Gebräuchliche Gleitkommaformate

Die IEEE hat für Rechenanlagen drei Gleitkommaformate spezifiziert, die wir in den gängigen Programmiersprachen wiederfinden:

32 Bit (float, single):

1 Vorzeichenbit

8 Exponentenbits (MSB first)

23 Mantissenbits (MSB first)

Der Wert w einer in diesem Format dargestellten Zahl berechnet sich als:

$w=(-1)^V*(1,M)*2^{E-127}$, falls E>0 und E<255

$w=(-1)^V*(0,M)*2^{-126}$, falls E=0 und M≠0

$w=(-1)^V*0$, falls E=0 und M=0

$w=(-1)^v$*Infinity (∞), falls E=255 und M=0

w=NaN (not a number), falls E=255 und M≠0

Darstellbarer Bereich ca. $[-10^{38}...10^{38}]$

64 Bit (double):

1 Vorzeichenbit

11 Exponentenbits (MSB first)

52 Mantissenbits (MSB first)

Der Wert w einer in diesem Format dargestellten Zahl berechnet sich als:

$w=(-1)^V*(1,M)*2^{E-1023}$, falls E>0 und E<2047

$w=(-1)^V*(0,M)*2^{-1022}$, falls E=0 und M≠0

$w=(-1)^V*0$, falls E=0 und M=0

$w=(-1)^V*$Infinity (∞), falls E=2047 und M=0

w=NaN (not a number), falls E=2047 und M≠0

Darstellbarer Bereich ca. $[-10^{300}...10^{300}]$

80 Bit (extended):

1 Vorzeichenbit

15 Exponentenbits (MSB first)

64 Mantissenbits (MSB first)

Der Wert w einer in diesem Format dargestellten Zahl berechnet sich als:

$w=(-1)^V*(0,M)*2^{E-16383}$, falls E>0 und E<32767

$w=(-1)^v*$Infinity (∞), falls E=32767 und M=0

w=NaN (not a number), falls E=32767 und M≠0

Darstellbarer Bereich ca. $[-10^{5000}...10^{5000}]$

Dabei ist zu beachten:

Die Zahlen sind in normalisierter Form, d.h. vor dem Komma steht eine 1. Diese führende 1 wird nicht mit dargestellt (hidden one).

Der Exponent ist mit einem Offset versehen. Bei Single-Zahlen 127, bei Double-Zahlen 1023 und bei Extended-Zahlen 32767. D. h. die dargestellten Exponenten sind um diesen Offset zu vermindern um ihre „normalen" Dualdarstellungen zu bekommen. Man erreicht dadurch, dass keine negativen Exponenten auftauchen können.

Folgende Sonderfälle werden unterschieden:

Eine 0 wird dargestellt durch Vorzeichen=0, Mantisse = 000...0, Exponent = 000...0.

Eine ∞ wird dargestellt durch Mantisse = 000...0, Exponent = 111...1.

Eine nicht darstellbare Zahl (NaN, not a number) wird dargestellt als Mantisse ≠ 0 und Exponent = 111...1.

Wenn der Exponent 000...0 ist und die Mantisse ≠ 000...0, so wird die versteckte 1 nicht einkopiert.

1.6 Codierung von Zeichen in Binärdarstellung

Bisher haben wir uns mit der Codierung von Zahlen (z.B. Typ Integer oder Typ float) beschäftigt. Es werden aber auch andere Codierungen benutzt. Die folgende Tabelle zeigt verschiedene Codes für die Ziffern von 0 bis 9, die in Rechenanlagen genutzt werden:

Dezimalziffer	**Dual (8421)**	**Aiken**	**3-Exzess**	**2aus5Code**
0	0000	0000	0011	11000
1	0001	0001	0100	00011
2	0010	0010	0101	00101
3	0011	0011	0110	00110
4	0100	0100	0111	01001
5	0101	1011	1000	01010
6	0110	1100	1001	01100
7	0111	1101	1010	10001
8	1000	1110	1011	10010
9	1001	1111	1100	10100
Gewichte	8421	2421	keine	74210

Auch mit dieser Codierung kann man aber nur numerische Werte verarbeiten. Um nun alle Zeichen, also auch die Buchstaben und Sonderzeichen in Rechenanlagen verarbeiten zu können, haben sich zwei Standards durchgesetzt, EBCDIC und ASCII. Diese sind auf den folgenden Tabellen dargestellt.

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000																
1	0001																
2	0010																
3	0011																
4	0100	blank										§	.	<	(	+	|
5	0101	&										!	$	•	)	;	
6	0110	-	/									^	,	%		>	?
7	0111											:	#	@	´	*	“
8	1000		a	b	c	d	e	f	g	h	i						
9	1001		j	k	l	m	n	o	p	q	r						
A	1010			s	t	u	v	w	x	y	z						
B	1011																
C	1100		A	B	C	D	E	F	G	H	I						
D	1101		J	K	L	M	N	O	P	Q	R						
E	1110			S	T	U	V	W	X	Y	Z						
F	1111	0	1	2	3	4	5	6	7	8	9						

EBCDIC **(Extended Binary Coded Decimal Interchange Code)**

ASCII

(American Standard Code for Information Interchange)

	Hex				0	1	2	3	4	5	6	7
Hex					**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
0	**0**	**0**	**0**	**0**	NUL	DLE	SP	0	@	P	`	p
1	**0**	**0**	**0**	**1**	SOH	DC1	!	1	A	Q	a	q
2	**0**	**0**	**1**	**0**	STX	DC2	“	2	B	R	b	r
3	**0**	**0**	**1**	**1**	ETX	DC3	#	3	C	S	c	s
4	**0**	**1**	**0**	**0**	EOT	DC4	$	4	D	T	d	t
5	**0**	**1**	**0**	**1**	ENQ	NAK	%	5	E	U	e	u
6	**0**	**1**	**1**	**0**	ACK	SYN	&	6	F	V	f	v
7	**0**	**1**	**1**	**1**	BEL	ETB	´	7	G	W	g	w
8	**1**	**0**	**0**	**0**	BS	CAN	(	8	H	X	h	x
9	**1**	**0**	**0**	**1**	SKIP	EM	)	9	I	Y	i	y
A	**1**	**0**	**1**	**0**	LF	SUB	*	:	J	Z	j	z
B	**1**	**0**	**1**	**1**	VT	ESC	+	;	K	[	k	{
C	**1**	**1**	**0**	**0**	FF	FS	,	<	L	\	l	|
D	**1**	**1**	**0**	**1**	CR	GS	-	=	M	]	m	}
E	**1**	**1**	**1**	**0**	SO	HOME	.	>	N	^	n	~
F	**1**	**1**	**1**	**1**	SI	NL	/	?	O	_	o	DEL

ASCII verwendet nur 7 Bits für die eigentliche Codierung. Das 8-te Bit eines Bytes (in dem jedes Zeichen codiert wird) wird als Paritätsstelle zur Fehlererkennung benutzt. Dieses achte Bit wird immer so gesetzt, dass die Anzahl der 1en im Byte gerade ist. Auf diese Weise ist bei einer Übertragung jeder Fehler zu erkennen, bei dem genau ein Bit (oder eine ungerade Anzahl) innerhalb des Bytes verfälscht wurde.