Einführung in die Informatik

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- http://www.miwilhelm.de
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt

Rechnen in den Zahlensystemen

In allen Zahlensystemen lassen sich die Grundoperationen Addition, Subtraktion, Multiplikation und Division mit dem vom Dezimalsystem bekannten Verfahren ausführen.

Addition	0	1
0	0	1
1	1	10
Multiplikation	0	1
0	0	0
1	0	1

Rechnen in den Zahlensystemen

Addition von Oktalziffern

Addition	0	1	2	3	4	5	6	7	8
0									
1									
2									
3									
4									
5									
6									
7									
8									

Rechnen in den Zahlensystemen

Addition von Hexadezimalziffern

Add	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																

Rechnen in den Zahlensystemen

Addition von Hexadezimalziffern

Add	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	11	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Rechenbeispiele

Addition:

Die Addition wird von rechts nach links durchgeführt. Folgendes Beispiel soll dabei die Grundlage bilden:

Beispiel: Addition von Dezimalzahlen: 1996 + 4711 = ?

	1996
+	4711
Übertrag	01100
=	6707

$1996_{10} + 4711_{10} = 6707_{10}$

Rechenbeispiele

Beispiel:
Addition von Dualzahlen: $111\ 1100\ 1100_2 + 1\ 0010\ 0110\ 0111_2 = ?$

	11111001100
+	1001001100111
Übertrag	0111110011000
=	1101000110011

$$11111001100_2 + 1001001100111_2 = 1101000110011_2$$

Rechenbeispiele

Beispiel: Addition von Oktalzahlen: $3714_8 + 11147_8 = ?$

	3714
+	11147
Übertrag	01010
=	15063

$3714_8 + 11147_8 = 15063_8$

Rechenbeispiele

Beispiel:
Addition von Hexadezimalzahlen: $7CC_{16} + 1267_{16} = ?$

	7CC
+	1267
Übertrag	0110
=	1A33

$$7CC_{16} + 1267_{16} = 1A33_{16}$$

Rechenbeispiele

Auch die Multiplikation kann mit Hilfe der Tabellen in gleicher Art und Weise, wie im Dezimalsystem auch in den anderen Zahlensystemen durchgeführt werden:

Beispiel: Multiplikation von Dezimalzahlen: $1996_{10} \cdot 12_{10} = ?$

		1996 x 12
		1996
	+	03992
Übertrag		111
	=	23952

$$1996_{10} \cdot 12_{10} = 23952_{10}$$

Rechenbeispiele

Beispiel:

Multiplikation von Dualzahlen: $1100_2 \cdot 1011_2 = ?$

									1	1	0	0	•	1	0	1	1		
											1	1	0	0					
															0				
													1	1	0	0			
														1	1	0	0		
Übertrag											1	1	1						
	Σ									1	0	0	0	0	1	0	0		

Rechenbeispiele

Beispiel:

Multiplikation von Dualzahlen: $1100_2 \cdot 11111001100_2 = ?$

		1	1	1	1	1	0	0	1	1	0	0	•	1	1	0	0	
				1	1	1	1	1	0	0	1	1	0	0				
					1	1	1	1	1	0	0	1	1	0	0			
						0	0	0	0	0	0	0	0	0	0	0		
							0	0	0	0	0	0	0	0	0	0	0	
Übertrag				1	1	1	1			1	1							
Σ			1	0	1	1	1	0	1	1	0	0	1	0	0	0	0	

Rechenbeispiele

Beispiel:

Multiplikation von Dualzahlen: $11111001100_2 \cdot 1100_2 = ?$

		1	1	1	1	1	0	0	1	1	0	0	•	1	1	0	0	
				1	1	1	1	1	0	0	1	1	0	0				
					1	1	1	1	1	0	0	1	1	0	0			
						0	0	0	0	0	0	0	0	0	0	0		
							0	0	0	0	0	0	0	0	0	0	0	
Übertrag				1	1	1	1			1	1							
			1	0	1	1	1	0	1	1	0	0	1	0	0	0	0	
Σ																		

Rechenbeispiele

Beispiel:

Multiplikation von Oktalzahlen: $3714_8 \cdot 14_8 = ?$

		3	7	1	4	•	1	4	
				3	7	1	4		
					4	4	4	0	
Überträge				1	3	0	2		
				1		1			
Σ				5	6	6	2	0	

Rechnen in den Zahlensystemen

Multiplikation von Oktalziffern

Mult.	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5								
6								
7								

Rechnen in den Zahlensystemen

Multiplikation von Oktalziffern

Mult.	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

Rechnen in den Zahlensystemen

Multiplikation von Hexadezimalziffern

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	AE	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

Rechenbeispiele mit einem 8-Bit-Computer

Addition von Binärzahlen

8 + 10 =

15 + 15 =

126+3 =

Rechenbeispiele mit einem 4-Bit-Computer

Addition von Binärzahlen

4 + 8 =

1+13 =

3+13 = Problem

5+13 = Problem

Zahlenformate

Für die Bearbeitung von Zahlen in Computern ist es erforderlich Vereinbarungen über die Formate zur Darstellung und Speicherung von Zahlen zu treffen.

- Computerzahlenformate begrenzen die darstellbaren Zahlenmengen auf Mengen mit nur endlich vielen Zahlen.
- Daneben steht für die Darstellung einer Zahl selbst nur ein beschränktes Format zur Verfügung (z.B. nur endlich viele Ziffern für die Darstellung einer irrationalen Zahl oder einer rationalen Zahl mit periodischer Dezimalbruchentwicklung)
- Beim Rechnen mit diesen Zahlen gelten nicht alle mathematischen Gesetze:
  - 0.1+0.1+0.1 != 0.3

Zahlenformate

Für die Darstellung von Daten als Zahlen lassen sich entsprechend der unterschiedlichen Aufgabenstellungen auch unterschiedliche Zahlenformate unterteilen. Beispielsweise müssen die Forderungen:

- großer Zahlenbereich oder
- gleichbleibende Genauigkeit im gesamten Zahlenbereich

durch verschiedene Zahlenformate realisieren lassen. Dabei ist ein Kompromiss zwischen den beiden Forderungen zu finden.

Verwendete Zahlenformate:

- Festkommazahlen,
- Vorzeichenbetragszahlen
- Komplementdarstellung und
- Gleitkommazahlen.

Festkommazahlen

Es ist ein Zahlenformat mit fester Zuordnung der Kommastelle, die damit nicht dargestellt werden muss. Denkbar sind folgende Festlegungen bezüglich der Kommastelle:

- Komma links der Zahl (linksbündig). Damit lassen sich nur gebrochene Zahlen darstellen.
- Komma rechts der Zahl (rechtsbündig). Damit lassen sich nur ganze Zahlen darstellen.
- Fünf Stellen vor und zwei Stellen nach dem Komma

Alle diese Möglichkeit werden selten benutzt, da die Gleitkommazahlen deren Anwendungsgebiet voll abdecken. (Eingeschränktes Numerikmodell)

Festkommazahlen

In Stellenwertsystemen zur Basis b gilt in der Zifferndarstellung (ohne Vorzeichen):

$10...0_b = 0Y...Y_b + 1$, wobei Y=b-1 höchste Ziffer ist.

↑ ↑

Stelle von b^n Stelle von b^n (n+1-te Stelle von hinten !)

„....Beim sukzessiven Aufaddieren von 1 wird also die Stelle von b^n (d.h. die Ziffer vor b^n) erst dann mit 1 besetzt und alle anderen Stellen vorher auf Null gesetzt, wenn im vorherigen Additionsschritt in der Zifferndar-stellung mindestens alle Stellen vorher mit der höchsten Ziffer (b-1) besetzt worden sind....“.

Beispiel (Dualzahlen): 111 + 1 = 1000 wenn 3 Stellen verfügbar

welches ist die Lösung?

Festkommazahlen

Dezimalsystem mit zwei Stellen

Daten-Bereich 0,1,2,3,4,5 ... 99

Addition: 12+87 = 99

20+81 = ?

Binärsystem mit 8 Stellen

Daten-Bereich 0,1,2,3,4,5 ... ?

Addition: 250+5 = ?

250+6 = ?

$250_{10} = 1111\ 1010_2$

Negative Zahlen: Vorzeichenbetragzeichen

Die erste Stelle des Zahlenformats ist das Vorzeichen

- „0“ für „+“
- „1“ für „-“
- danach der „Absolutwert“

Beispiel n=4

0	111	+7
1	111	-7

Negative Zahlen: Vorzeichenbetragzeichen

Beispiel n=4, Basis=2

0 000	+0	1 000
0 001	+1	1 001
0 010	+2	1 010
0 011		1 011
0 100		1 100
0 101		1 101
0 110		1 110
0 111		1 111

Negative Zahlen: Vorzeichenbetragzeichen

Addition:

Dezimal		Binär
-2		1010
+5		0101
+3	≠	1111

Vorteile/Nachteile:

- Zwei verschiedene Darstellungen für Null existieren.
- Das schriftliche Addieren ist bei negativen Zahlen nicht anwendbar (-2 + 5).

Festkommazahlen

Addiert man also in einem Stellenwertsystem, speziell hier mit der Basis 2 (Dualzahlen), zu der Zifferndarstellung einer Zahl a (bis Stelle 2^n einschließlich) jetzt die Zahl a mit der „invertierten Zifferndarstellung“ (d.h. alle Einsen durch Null, alle Nullen durch Eins ersetzt), so erhält man die Zahl bei der alle Stellen mit 1 besetzt sind, also $2^{(n+1)} - 1$.

$170_{10} = 10101010_2$

```
  10101010
  01010101      // Vertauschung  (Einerkomplement)
-----------
  11111111      // Summe
```

Negative Zahlen: Einerkomplement

Binärsystem mit 4 Bit Länge

positiver Zahlenbereich: von 0 bis 15

Möglicher Zahlenbereich: von -8 bis +8
von -7 bis +7
von -8 bis +7
von -7 bis +8

Welche Auswählen?

Negative Zahlen: Einerkomplement

Binärsystem mit 4 Bit Länge

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

Vollständiger Code?

Negative Zahlen

Binärsystem mit 8 Bit Länge

positiver Zahlenbereich: von 0 bis 255

Möglicher Zahlenbereich: von -128 bis +128
von -127 bis +127
von -128 bis +127
von -127 bis +128

Negative Zahlen: Einerkomplement

Binärsystem mit 8 Bit Länge

Bit	7-4	3-0
0	0000	0000
1	0000	0001
2	0000	0010
3	0000	0011
4	0000	0100
5	0000	0101
6	0000	0110
7	0000	0111
...		
127	0111	1111

Negative Zahlen: Einerkomplement

Binärsystem mit 8 Bit Länge

Bit	7-4	3-0
-0	1111	1111
-1	1111	1110
-2	1111	1101
-3	1111	1100
-4	1111	1011
-5	1111	1010
-6	1111	1001
-7	1111	1000
...		
-127	1000	0000

Negative Zahlen: Einerkomplement

Regeln:

- Die Subtraktion einer positiven Zahl wird auf die Addition des Komplements zurückgeführt.
- Das Komplement einer Zahl ist die bitweise Vertauschung von Nullen und Einsen.
- Falls ein Überlauf stattfindet, muss diese „Eins“ zum Ergebnis hinzuaddiert werden.

Negative Zahlen: Einerkomplement

Beispiele:

	+5	0101_2
+	–5	1010_2
Σ	**-0**	**1111_2**

	+5	0101_2
–	+3	0011_2
	+5	0101_2
+	–3	1100_2
+	1	0001
+		1
Σ		10_2

Negative Zahlen: Einerkomplement

Beispiele:

	+3	0011_2
–	5	0101_2
	+3	0011_2
+	–5	1010_2
Σ		1101_2 = -2

	–2	0010_2
	–3	0011_2
+	–2	1101_2
+	–3	1100_2
+	1	1001
		1
Σ		1010_2 = -5

Negative Zahlen: Einerkomplement

Beispiele:

	+43	0101011_2		+27	00011011_2
–	27	0011011_2	+	–43	11010100_2
	+43	0101011_2	**Σ**	**–16**	$\mathbf{11101111_2}$
+	-27	1100100_2			
+		**1 0001111**			
		1			
Σ	**16**	$\mathbf{0010000_2}$			

Negative Zahlen: Zweierkomplement

Binärsystem mit 4 Bit Länge

positiver Zahlenbereich: von 1 bis 15

Möglicher Zahlenbereich:
- von -8 bis +8
- von -7 bis +7
- **von -8 bis +7**
- von -7 bis +8

Negative Zahlen: Zweierkomplement

Binärsystem mit 4 Bit Länge

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

Negative Zahlen: Zweierkomplement

Regeln:

- Die Subtraktion einer positiven Zahl wird auf die Addition des 2-Komplements zurückgeführt.
- Das Komplement einer Zahl ist die bitweise Vertauschung von Nullen und Einsen. Danach die Addition von Eins.
- Ein Überlauf wird ignoriert.

Negative Zahlen: Zweierkomplement

Vorteile:

- MSB zeigt Vorzeichen
- Die Zahlen sind modulo 16 (2^n) in der richtigen Reihenfolge.

Beispiele:

+5	–	21	=	16
+7	–	39	=	32
–4	–	12	=	–16
–1	–	15	=	–16
–8	–	8	=	–16

Negative Zahlen: Zweierkomplement

Herleitung der Eigenschaften für 4 Bits pro Zahl:

1111 entspricht -1

0101 entspricht 5

1010 entspricht dem 1-Komplement von $5 = \overline{5}$

1111 entspricht der Summe $(a + \overline{a})$

$$\Rightarrow a + \overline{a} = -1$$

$$\Rightarrow a + \overline{a} + 1 = 0$$

$$\Rightarrow a + (\overline{a} + 1) = 0$$

$$\Rightarrow a + -a = 0$$

$$\Rightarrow -a = \overline{a} + 1$$

Negative Zahlen: Zweierkomplement

Beispiele:

	+5		0101_2		+5		0101_2
–	5		0101_2	–	3		0011_2
	+5		0101		+5		0101_2
+	¬5		1011	+	¬ 3		1101_2
Σ	**0**	**1**	**0000_2**	**Σ**		1	0010

Negative Zahlen: Zweierkomplement

Beispiele:

	+3	0011_2
–	5	0101_2
	+3	0011_2
+	–5	1011_2
Σ		$1110_2 = -2$

	–2	0010_2
	–3	0011_2
+	–2	1110_2
+	–3	1101_2
Σ	1	$1011 = -5$

Negative Zahlen: Zweierkomplement

Beispiele:

	+43	0101011_2
–	+27	0011011_2
	+43	0101011_2
+	–27	1100101_2
+	**1**	**0010000**
Σ	**16**	**0010000_2**

–	+43	0101011_2
+	+27	0011011_2
+	–43	1010101_2
+	+27	0011011_2
Σ	**–16**	**1110000_2**
		0001111_2
		1_2
		0010000_2

Addition: Zweierkomplement

Beispiele:

	+43	0101011_2
–	+27	0011011_2
	+43	0101011_2
+	–27	1100101_2
+	**1**	**0010000**
Σ	**16**	**0010000_2**

–	+43	0101011_2
–	+27	0011011_2
+	–43	1010101_2
+	+27	0011011_2
Σ	**–16**	**1110000_2**
		0001111_2
		1_2
		0010000_2

Zweierkomplement

Es macht also Sinn, "$\bar{a}+1$" als "-a" zu betrachten

In Computern kann für feste Zahlenformate damit das “Subtrahieren“ auf “Invertieren” und “Addieren von 1” zurückgeführt werden.

Bezeichnungen:

$\bar{a}$ heißt "invertierte Darstellung von a" oder „Einerkomplement von a bzw. B-1-Komplement".

$\bar{a}+1$ heißt "Zweierkomplement von a bzw. B-Komplement"

Mathematischer Befehlssatz einer CPU

Aus diesen Beispielen kann abgeleitet werden, daß die Einführung von Komplementzahlendarstellungen eine deutliche Vereinfachung der arithmetischen Operationen in einem Computer erfolgen kann. Für die vier Grundrechenarten werden nur wenige Befehle aus dem Befehlssatz der CPU benötigt:

Addition mit Übertragsrechnung

- bitweise Verschiebung nach links und nach rechts
- bitweise Negation

Abschließend sei erwähnt, dass alle Mikrorechner auf der Basis des B-Komplementes Festkommazahlen verarbeiten.

Gleitkommazahlen / Floating Points

1. Variante:

Ganzzahliges Format mit festen Nachkomastellen
Ganzzahliges Format, die letzten Ziffern sind Nachkommastellen

Beispiel: 2 Nachkomastellen

64 Bit Datenlänge:

0 .. 18446744073709551616

–9223372036854775808 ... +9223372036854775808

⇒ **± 1.000.000.000.000.000,00**

Gleitkommazahlen

1. Variante:

Eigenschaften:

- Einfache und schnelle Berechnung
- Feste Nachkommastellen
- Vorkommastellen abhängig von den Nachkommastellen
- Kein einheitliches Format (Datentransfer), da die Nachkommastellen unterschiedlich sein können

Gleitkommazahlen

2. Variante:

Dieses Zahlenformat gestattet die Beherrschung praktisch unbegrenzter Wertebereiche mit befriedigender Genauigkeit.

Grundlage der Darstellung ist, dass eine reelle Zahl im Zahlensystem mit der Basis wie folgt darstellt werden kann:

$$X = \pm m \cdot B^{\pm e}$$

e ist ganzzahlig

m ist ganzzahlig

$$e = \pm |e| \qquad\qquad m = \pm |m|$$

Das Vorzeichen der Mantisse m ist gleich dem der Zahl .

Gleitkommazahlen

Normierung:
Bei Punktverschiebung (Kommaverschiebung) in der Mantisse m um eine Stelle nach rechts (links) bleibt der Wert von erhalten, wenn gleichzeitig der Exponent e um 1 erniedrigt (erhöht) wird.

Beispiel:
$1996 = 19960 \cdot 10^{-1} = 199{,}6 \cdot 10^{1} = 1{,}996 \cdot 10^{3} = 0{,}1996 \cdot 10^{4}$

Steht, wie in den letzten beiden Darstellungsformen, die erste von Null verschiedene Mantissenziffer stets unmittelbar vor bzw. hinter dem Punkt (Komma), bezeichnet man als normiert. Damit gilt für die Gleitkommadarstellung folgende Festlegung:

$1_B \leq m < 10_B$ mit B=2,8,10,16

Gleitkommazahlen

Dabei ist zu beachten, dass die Zahl Null nicht normiert werden kann und damit eine Sonderbehandlung erfährt.

Gründe ?

Gleitkommazahlen / Floating Points

Aus dieser Darstellung folgt eine weitere Festlegung:

m = 1,*f* **Mantisse = 1,Fraction**

Für die weitere Betrachtung wird nur noch *f, die Fraktion* verwendet. *f* ist die um 1, reduzierte Mantisse m, mit der Festlegung, dass der Punkt (Komma) stets links von der Mantisse, dargestellt als *f* , definiert wird, aber nicht geschrieben wird.

Charakteristik:

Für den Exponenten genügt ein relativ kleiner Wertebereich (z.B.: bei B = 10), um extrem kleine bis extrem große Zahlenbeträge darstellen zu können.

Gleitkommazahlen

Die Vorzeichenbehandlung des Exponenten wird umgangen, indem mittels **Verschiebekonstante *K*** zur **Charakteristik *Ch*** übergewechselt wird und damit der Wertebereich an den Anfang des positiven Zahlenbereichs verschoben wird:

$$Ch = e + K$$

Mit dieser letzten Maßnahme wird das Vorzeichen des Exponenten in den positiven Bereich verschoben und wird nicht mehr (als negative Zahl) dargestellt.

Gleitkommazahlen

Für das Vorzeichen der Mantisse gelten die gleichen Vereinbarungen, wie für das Vorzeichen der Vorzeichenbetragzahlen:

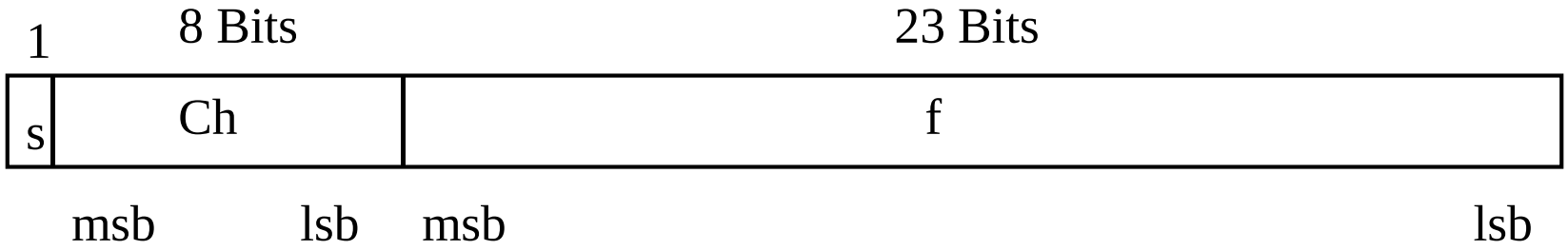
$s = 0 \quad \Rightarrow \quad z \geq 0$
$s = 1 \quad \Rightarrow \quad z < 0$

Damit sind alle Komponenten zur Zahlendarstellung eingeführt:

- das Vorzeichen der Mantisse s,
- die Abspaltung der Mantisse *f*
- die Charakteristik *Ch*

Gleitkommazahl Single:

k = 127
k = ?F



Single-Zahl hat 4 Byte =32 Bit

Falls 0 < Ch < 255, e = Ch - k Ch = e+k

m = 1.f (Hidden-Bit)

s = 0 ⇔ z ≥ 0

s = 1 ⇔ z < 0

s=1; e = 0 ; f = 0, Zahl = -0.0 (signed zero)

s=0; e = 0 ; f = 0, Zahl = 0.0 (unsigned zero)

e = 0 ; f = 0, Zahl = $(-1)^s \cdot 2^{-126} \cdot 0.f$ (subnormal numbers)

s={0,1} ; Ch =255 ; f = 0, Zahl = ±INF

s=undef ; Ch =255 ; f ≠ 0, Zahl = NAN (Not any Number)

msb = most significant bit lsb = least significant bit

Beispiel:

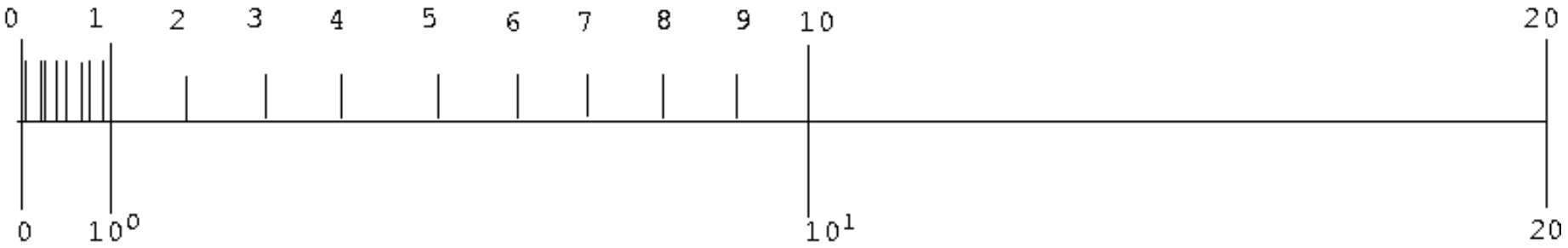
k = 127
Ch = 8 Bit

$d = -12{,}75_{10}$

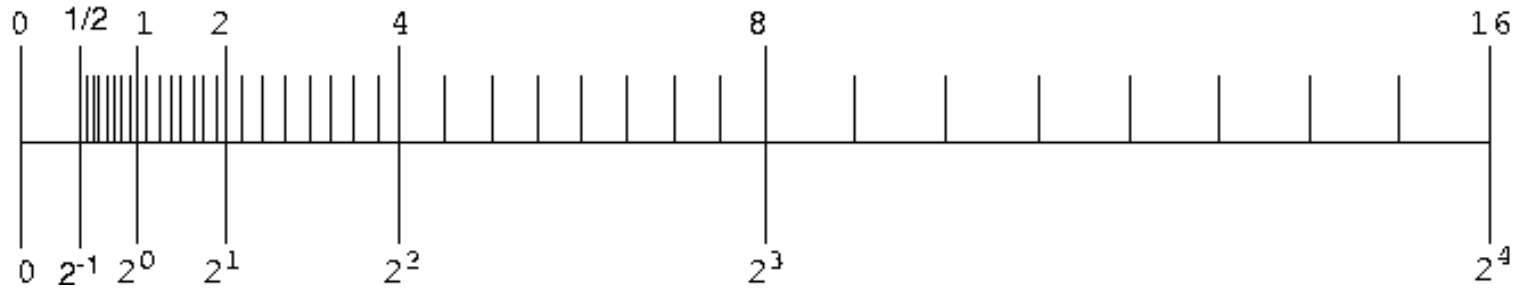
1) Umwandlung ins Binärsystem: $-1100{,}11_2$
2) Normierung auf $1{,}m \cdot 2^x$ $-1{,}100\ 11\ \ 2^3$
3) Bestimmen von s, Abspalten von f und e, Ch
   s = 1
   m=1,f=1,10011
   e=3 => Ch = 127+3=130=82_{16}
4) Zusammenfassen: s Ch f
   1 1000 0010 1011 0000 0000 0000 0000 000
5) Hexadezimale Fassung
   1100 0001 0101 1000 0000 0000 0000 0000
   C 1 5 8 0 0 0 0

Zahlendarstellung

Decimal Representation:



Binary Representation:



Min. Single: 1.17549435e-38
00800000

Max. Single: 3.40282347e+38
7F7FFFFF

Zahlendarstellungen im Singleformat

x	Nächste Zahl (x, +)	Lücke
0.0	$1.4012985 \cdot 10^{-45}$	$1.4012985 \cdot 10^{-45}$
$1.1754944 \cdot 10^{-38}$	$1.1754945 \cdot 10^{-38}$	$1.4012985 \cdot 10^{-45}$
1.0	1.0000001	$1.1920929 \cdot 10^{-07}$
2.0	2.0000002	$2.3841858 \cdot 10^{-07}$
16.000000	16.000002	$1.9073486 \cdot 10^{-06}$
128.00000	128.00002	$1.5258789 \cdot 10^{-05}$
$1.0000000 \cdot 10^{+20}$	$1.0000001 \cdot 10^{+20}$	$8.7960930 \cdot 10^{+12}$
$9.9999997 \cdot 10^{+37}$	$1.0000001 \cdot 10^{+38}$	$1.0141205 \cdot 10^{+31}$

Beispiel Single: 1,0

x=1,0

Hexadezimal Darstellung: 3F80 0000

0011 1111 1000 0000 0000 0000 0000 0000

x=1,0 + x

Hexadezimal Darstellung: 3F80 0001

0011 1111 1000 0000 0000 0000 0000 0001

s=0

Ch=0111 1111 Ch=127 e=0

m=1, 000 0000 0000 0000 0000 0001

Differenz: $\Delta = 2^{-23} = 0{,}00000011920928955078125$

Beispiel Single: 2,0

x=2,0
Hexadezimal Darstellung: 4000 0000
0100 0000 0000 0000 0000 0000 0000 0000

x=2,0 + x

Hexadezimal Darstellung: 4000 0001

0 10000000 000 0000 0000 0000 0000 0001

s=0

Ch=1000 0000 Ch=128 e=1

m=1, 000 0000 0000 0000 0000 0001

X=10,00 0000 0000 0000 0000 0001
Differenz: $\Delta = 2^{-22} = 0{,}00000023841857910156 25$

Beispiele Single:

k = 127
Ch = 8 Bit

$d = 4711_{10} = 1267_{16}$ = 45 93 38 00
s = 0, Ch = 8B, f = 267

$d = 1_{10} = 1_{16}$ = 3F 80 00 00
s = 0, Ch = 7F, f = 0

$d = -1_{10} = 1_{16}$ = BF 80 00 00
s = 1, Ch = 7F, f = 0

$d = 2_{10} = 2_{16}$ = 40 00 00 00
s = 0, Ch = 80, f = 0

Gleitkommazahl Double:

$k = 1023$

$k = 3FF_{16}$

$e = Ch - k$ $\quad Ch = e+k$ $\quad m = 1.f$

Double-Zahl hat 8 Byte =64 Bit

$0 < e < 2047$, $\quad (-1)^s \cdot 2^{-1023} \cdot 1.f$ (normal numbers)

$e = 0 \;\; ; f \neq 0$, $\quad (-1)^s \cdot 2^{-1022} \cdot 0.f$ (denormalized numbers)

$s=0 ; e = 0 \;\; ; f = 0$, $\quad$ Zahl = +0.0

$s=\{0,1\} ; Ch = 2047 ; f = 0$, $\quad$ Zahl = ±Infimum

$s=$undef $; Ch = 2047 ; f \neq 0$, $\quad$ Zahl = NaN (Not any Number).

msb = most significant bit lsb = least significant bit

Beispielwerte im Doubleformat

Bezeichung	Bit Pattern (Hex)	Dezimal Wert
+ 0	00000000 00000000	0.0
- 0	80000000 00000000	-0.0
1	3FF00000 00000000	1.0
2	40000000 00000000	2.0
Max normal Zahl	7FFFFFFF FFFFFFFF	1.7976931348623157e+308
Min positive normal Zahl	00100000 00000000	2.2250738585072014e-308
Max subnormal Zahl	000FFFFF FFFFFFFF	2.2250738585072009e-308
min positive subnormal Zahl	00000000 00000001	4.9406564584124654e-324
+∞	7FF00000 00000000	+Infinity
-∞	FFF00000 00000000	-Infinity
Not-any-Number	7FF80000 00000000	NaN

Beispiele Double:

k = 1023
Ch = 11 Bit

$d = 4711_{10} = 1267_{16}$ = 40B26700 00000000
s = 0, Ch = 40B, f = 267

$d = 1_{10} = 1_{16}$ = 3FF00000 00000000
s = 0, Ch = 3FF, f = 0

$d = -1_{10} = 1_{16}$ = BFF00000 00000000
s = 1, Ch = 3FF, f = 0

$d = 2_{10} = 2_{16}$ = 40000000 00000000
s = 0, Ch = 400, f = 0

Gleitkommazahl Extended:

k = 16383
k = 3FFF

Extended-Zahl hat 10 Byte =80 Bit

-23,5625

e = Ch – k m = **j**.f

j=1 ; 0 < e < 32767, f ≠ 0	$(-1)^s \cdot 2^{-16383} \cdot 1.f$ (normal numbers)
j=0 ; e=0, f ≠ 0	$(-1)^s \cdot 2^{-16382} \cdot 0.f$ (denormalized)
s=0 ; e = 0 ; f = 0,	Zahl = +0.0
s={0,1} ; Ch = 32767 ; f = 0,	Zahl = ± Infimum
s=undef ; Ch = 32767 und f ≠ 0,	Zahl = NaN.

msb = most significant byte lsb = least significant byte

Gleitkommazahl Extended:

Double-Extended Bit Pattern (x86)	**Value**
j = 0, 0 < e <32767	Unsupported
j = 1, 0 < e < 32767	$(-1)^s \cdot 2^{-16383} \cdot 1.f$ (normal numbers)
j = 0, e = 0; f ≠ 0 (at least one bit in f is nonzero)	$(-1)^s \cdot 2^{-16382} \cdot 0.f$ (subnormal numbers)
j = 1, e = 0	$(-1)^s \cdot 2^{-16382} \cdot 1.f$ (pseudo-denormal numbers)
j = 0, e = 0, f = 0 (all bits in f are zero)	$(-1)^s \cdot 0.0$ (signed zero)
j = 1; s = 0; e = 32767; f = 0 (all bits in f are zero)	+INF (positive infinity)
j = 1; s = 1; e = 32767; f = 0 (all bits in f are zero)	-INF (negative infinity)
j = 1; s = u; e = 32767; f = .1uuu -- uu	QNaN (quiet NaNs)
j = 1; s = u; e = 32767; f = .0uuu -- uu ≠ 0 (at least one of the u in f is nonzero)	SNaN (signaling NaNs)

Beispielwerte im Extendedformat

Bit Pattern (Hex)	**Dezimal Wert**
j = 0, 0 < e <32767	Unsupported
j = 1, 0 < e < 32767	$(-1)^s \cdot 2^{-16383} \cdot 1.f$ (normal numbers)
j = 0, e = 0; f ≠0	$(-1)^s \cdot 2^{-16382} \cdot 0.f$ (subnormal numbers)
j = 1, e = 0	$(-1)^s \cdot 2^{-16382} \cdot 1.f$ (pseudo-denormal numbers)
j = 0, e = 0, f = 0	$(-1)^s \cdot 0.0$ (signed zero)
j = 1; s = 0; e = 32767; f = 0	+INF (positive infinity)
j = 1; s = 1; e = 32767; f = 0	-INF (negative infinity)

j = 1; s = u; e = 32767; f = .1uuu -- uu	QNaN	(quiet NaNs)
j = 1; s = u; e = 32767; f = .0uuu -- uu 0		SNaN (signaling NaNs)
quiet NaN with greatest fraction	7FFF FFFFFFFF FFFFFFFF	QNaN
quiet NaN with least fraction	7FFF C0000000 00000000	QNaN
signaling NaN with greatest fraction	7FFF BFFFFFFF FFFFFFFF	SNaN
signaling NaN with least fraction	7FFF 80000000 00000001	SNaN

Beispiele Extended:

k = 16383=3FFF
Ch = 15 Bit

$d = 4711_{10} = 1267_{16} = 1001001100111_2$

$s = 0,$

$1{,}001001100111_2 \cdot 2^{12}$

$e = 12$

$Ch = k + e = 16383+12 = 16395 = 400B_{16}$

Tetrade f wird von links mit der 1, berechnet !!

$jf = 1001\ \ 0011\ 0011\ 1_2$

$jf = 9338_{16}$

$4711_{10} = 40\ 0B\ 93\ 38\ 00\ 00\ 00\ 00\ 00\ 00$

Beispiele Extended:

k = 16383=3FFF
Ch = 15 Bit

$d = 4711_{10} = 1267_{16}$ = 400B 9338 0000 0000 0000
s = 0, Ch = 400B, f = 9338 0000 0000 0000

$d = 1_{10} = 1_{16}$ = 3FFF 8000 0000 0000 0000
s = 0, Ch = 3FFF, f = 8000 0000 0000 0000

$d = -1_{10} = 1_{16}$ = BFFF 8000 0000 0000 0000
s = 1, Ch = 3FFF, f = 8000 0000 0000 0000

$d = 2_{10} = 2_{16}$ = 4000 8000 0000 0000 0000
s = 0, Ch = 4000, f = 8000 0000 0000 0000

Beispiele Extended:

k = 16383=3FFF
Ch = 15 Bit

$d = ??_{10} =$ 4009 9A50 0000 0000 0000

$s = 0$

$Ch = 4009_{16} = 16393$

$\Rightarrow \quad e = Ch-k = 16393-16383 = 10$

$f = 9A50\ 0000\ 0000\ 0000_{16}$

$f = 1001101001010000_2 \qquad (0000\ 0000\ 0000)_{16}$

$m = 1,f = 1,001101001010000_2$

$Zahl = s*m*2^e = 1,00110100101*2^{10} =$

Zahl = 10011010010,1 = 0100 1101 0010,1

Zahl = 4D2 + 0,5 = 1234,5

Rechnen mit Floating-Points

Das Rechnen geschieht nach folgenden Regeln:

- Umwandeln der Operanden ins interne Rechnerformat
- Normalisieren, Komma verschieben
- Operation ausführen
- Umwandeln des Ergebnisses in Ausgabeformat

Mögliche Rundungen:

- Round to nearest
- Round to even
- Truncate

Rechnen mit Floating-Points

Eigenes Floating-Point-System:

- Signed (Vorzeichen)
- Exponent mit 4 Bit Länge
- Mantisse mit 8 Bit Länge, ohne Hidden-Bit

- 1. Aufgabe:

$1{,}5 + 125{,}5 = 127{,}0$

$1{,}5 \equiv 1{,}1_2$

$125{,}5 \equiv 1111101{,}1_2$

$1{,}1 + 1111101{,}1 = 1111111{,}0$

Rechnen mit Floating-Points

- 2. Aufgabe:
- 1,125 + 15,5 =

- 3. Aufgabe:
- 0,5625 + 127,0 =

- 4. Aufgabe:
- 11,03125 + 120,625 =

Integer-Zahlenformate in Visual C / Delphi

Integer-Zahlen sind ganze Zahlen. Sie werden durch Festkomma-zahlen mit rechtsbündigen Komma dargestellt. Dabei wird ein symmetrischer Zahlenbereich, d.h. annähernd gleiche Bereiche positiver und negativer ganzer Zahlen zu Grunde gelegt.

Negative Zahlen werden dabei als B-Komplement bzw. 2k-Zahlen intern verarbeitet.

Integer-Zahlenformate in Visual C / Delphi

Delphi	C++	Bereich	Format
shortint	short	-128..127	8 Bit
integer	int	-32768..32767	16 Bit
longint	long	-2.147.483.648 .. 2.147.483.647	32 Bit
int64	long long	-9223372036854775809 ... +9223372036854775808	64 Bit
byte	char	0..255	8 Bit
word	unsigned int	0..65535	16 Bit
DWord	unsigned long	0.. 4294967295	32 Bit
longword	dword	0..18446744073709551616	64 Bit

Integer-Zahlenformate in Java

Java	**Bereich**	**Format**
byte	-128...127	8 Bit
short	-32768..32767	16 Bit
int	-2.147.483.648 .. 2.147.483.647	32 Bit
long	-9223372036854775809 ... +9223372036854775808	64 Bit
char	0..65535	16 Bit
_	0..65535	16 Bit
_	0.. 4294967295	32 Bit
_	0..18446744073709551616	64 Bit

Real-Zahlenformate in Visual C / Delphi

Delphi	**C++**	**Bereich**	**Stellen**	**Format**
Real	-	$2.9 \cdot 10^{-39}$ .. $1.7 \cdot 10^{38}$	11-12	6 Byte
Single	float	$1.175 \cdot 10^{-45}$ .. $3.402 \cdot 10^{38}$	7-8	4 Byte
Double	double	$2.225 \cdot 10^{-308}$ .. $1.797 \cdot 10^{308}$	16-17	8 Byte
Extended	long double	$3.362 \cdot 10^{-4932}$ .. $1.189 \cdot 10^{4932}$	19-20	10 Byte

Weitere Angaben:

Delphi	**C++**	**Länge Ch**	**Wert K**
Real	-	8	81_{16}
Single	single	8	$7F_{16}$
Double	double	11	$3FF_{16}$
Extended	long double	15	$3F\ FF_{16}$

Real-Zahlenformate in Java

Java	**Bereich**	**Stellen**	**Format**
_	$2.9*10^{-39}$ .. $1.7*10^{38}$	11-12	6 Byte
float	$1.5*10^{-45}$ .. $3.4*10^{38}$	7-8	4 Byte
double	$5.0*10^{-324}$ .. $1.7*10^{308}$	16-17	8 Byte
_	$3.4*10^{-4932}$ . . $1.1*10^{4932}$	19-20	10 Byte

Weitere Angaben:

```
float f;
f = 234.55f;          // Initialisierung mit einer float-Zahl
                         sonst wäre es eine double-Zahl
```