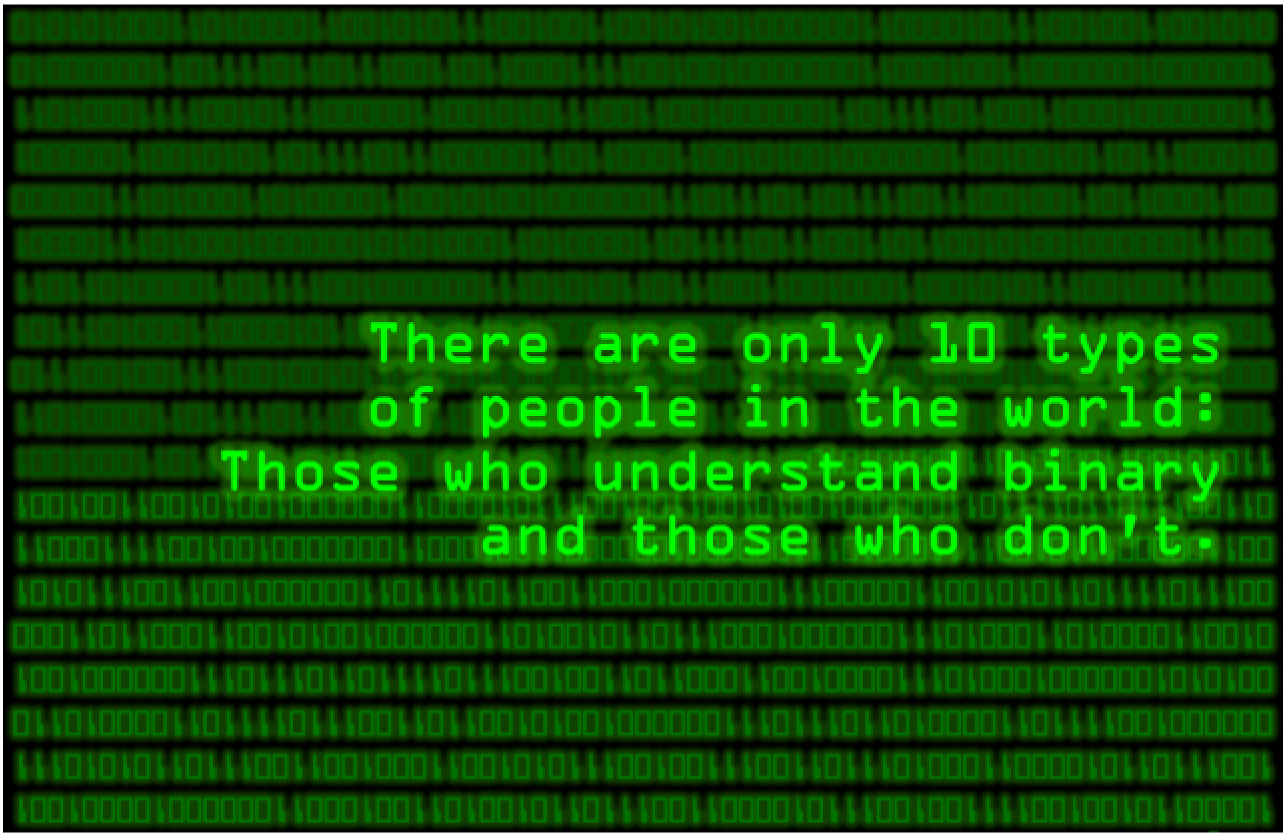
Digitale Systeme und Computersysteme

Kompetenzbereich: Embedded Systems



https://theshpitz.wordpress.com/tag/binary-code/

Auszug Lehrplan

Die Schülerinnen und Schüler können im III. Jahrgang im Bereich Embedded Systems…

die Vor- und Nachteile der verschiedenen Zahlendarstellungen im Dualsystem und die Basisalgorithmen der Dualarithmetik erklären

1 Zahlensysteme

Die Darstellung von Größen (ganz allgemein) kann prinzipiell auf 2 Weisen erfolgen:

Digitale Informationsverarbeitung: **zweiwertige (binäre) Signale**

Einsatz verschiedener Zahlensysteme:

römische Zahlen: I, II, III, IV, V, …

dezimales Zahlensystem: 0, 1, …, 9

1.1 Darstellung von Zahlen in Zahlensystemen

Eigenschaften des dezimalen Zahlensystems:

- ____
- ____
- ____

Allgemein: Z =

$$Z = \sum_{i=-m}^{n} a_i \cdot b^i$$

m, n: ganze, positive Zahlen

$0 \leq a_i < b$

a_i … Ziffern der Zahl
b … Basis des Zahlensystems; allgemein: $b \geq 2$

b = 10 :	Dezimalsystem
b = 2 :	Dualsystem
b = 8 :	Oktalsystem
b = 16 :	Hexadezimalsystem (Hexadekadisch, Sedezimalsystem)

Beispiele :

b=2: $Z = 1010_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 8 + 2 = 10_{10}$

$Z = 1001{,}11_2 =$

b=8: $Z = 1010_8 =$

$Z = 765_8 =$

b=16: $Z = 1010_{16} =$

$Z = 456_{16} =$

b = 16	b = 10	b = 8	b = 2
0	0	0	
1	1	1	
2	2	2	
3	3	3	
4	4	4	
5	5	5	
6	6	6	
7	7	7	
8	8		
9	9		

→

1.2 Umrechnung zwischen Zahlensystemen

Rationale Zahl: in 2 Schritten

1) Umrechnung des Ganzzahlanteils

2) Umrechnung des echt gebrochenen Anteils

Aufgaben:

a) $Z = 14_{10} \rightarrow ?_2$

b) $Z = 466_{10} \rightarrow ?_4$

c) $Z = 75{,}71875_{10} \rightarrow ?_2$

… wie sieht es mit der Umrechnung von z.B.

$Z = 3E4A_{16} \rightarrow ?_6$

aus?

… die Umrechnung ist extrem schwierig, da die Division im hexadezimalem Zahlensystem erfolgen muss [→ Division immer zur Basis des Quellsystems !].

Daher:

also: $Z = 3E4A_{16} \rightarrow ?_6$

Aufgabe: $0{,}49_{10} \rightarrow ?_2$

… also: $0{,}49_{10}$ =

Eine Hexadezimalzahl lässt sich als Dualzahl darstellen, indem man jede einzelne Hexadezimalziffer durch die entsprechenden 4 Dualziffern ersetzt [und vice versa].

Aufgaben: a) $F39C{,}0A_{16} \rightarrow ?_2$

b) $100101101{,}1001010_2 \rightarrow ?_{16}$

Im Oktalsystem entspricht jede Oktalziffer ↔ 3 Dualziffern.

1.3 Arithmetik mit Dualzahlen

Damit ein Rechner arithmetische Operationen mit Dualzahlen ausführen kann, müssen diese eine einheitliche, dem Rechenwerk bekannte Form haben.

Dabei ist die Wortlänge fest / begrenzt (z.B. 8 / 16 / 32 / 64 Bit); man möchte jedoch (muss)

- vorzeichenbehaftete
- gebrochenen

Zahlen verarbeiten.

1.3.1 Festkomma-Darstellung

a) Format:

Das Format ist definiert und steht somit fest; daher braucht auch kein Komma-Zeichen zwischen der „I-ten“ und „F-ten“ Stelle abgespeichert werden.

→ größte darstellbare vorzeichenlose Dualzahl:

→ kleinste darstellbare Zahl (≠ 0):

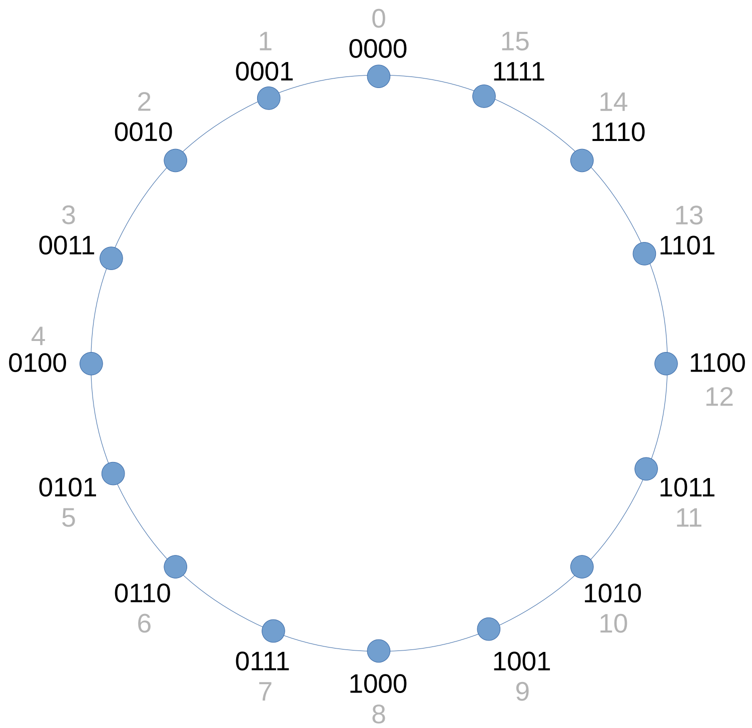
b) Darstellung vorzeichenbehafteter Dualzahlen

… <u>Vorzeichenbit:</u>

… <u>Komplementdarstellung:</u>

Idee: Der darstellbare Zahlenbereich wird in 2 Mengen aufgeteilt (mit unterschiedlichem Vorzeichen), wobei beide Mengen gleich viele Zahlen beinhalten. Da die 0 als positive Zahl gewertet wird ist somit die betragsmäßig größte Zahl um 1 kleiner als die betragsmäßig kleinste Zahl.

Beispiel: N = 4, I = 4, F = 0



<u>Bildungsregel (für Dualzahlen):</u>

1. Stellenweise Negation (→ Bildung des sog. 1er – Komplement)
2. Addition von +1 (an der niedrigsten Stelle)

Aufgaben: Es gelte stets *N=I=4*

a) $-5_{10} \rightarrow ?_{2}$

b) $1101_2 \rightarrow ?_{10}$

Dualzahlen für N = I = 8 (8bit Ganzzahlen):

+ 127	0111 1111
+ 126	0111 1110
…	
+ 2	0000 0010
+ 1	0000 0001
0	0000 0000
- 1	1111 1111
- 2	1111 1110
…	
- 127	1000 0001
- 128	1000 0000

Darstellbarer Zahlenbereich:

→ Zahlenbereich von Ganzzahlen mit n Bit Länge:

Aufgabe: $-25{,}375_{10} \rightarrow \ ?_{2}$ mit I = 8, F = 4

c) Arithmetik

→ Addition: Im Dualsystem gibt es nur die Ziffern 0 und 1, d.h.

$0 + 0 =$

$0 + 1 =$

$1 + 0 =$

$1 + 1 =$

Beispiel:

Als Überlauf bezeichnet man die sog. Zahlenbereichsüberschreitung, wobei die entstehende (N+1)te Stelle verloren geht [bei fester Wortbreite N].

→ Subtraktion: Wird auf eine Addition mit dem Zweierkomplement der neg. Zahl zurückgeführt.

→ Multiplikation: Hier gelten die 4 Grundregeln:

$0 \bullet 0 =$ $1 \bullet 0 =$

$0 \bullet 1 =$ $1 \bullet 1 =$

Händische Multiplikation im Dezimalsystem (a la Volksschule):

… identisches Vorgehen im Dualsystem ...

Aufgabe: Führe die Berechnung $5{,}625_{10} \bullet 6{,}125_{10}$ Schritt-für-Schritt im Dualsystem aus:

→ eine eigentliche Multiplikation wie im Dezimalsystem entfällt. Daher kann diese Rechenart durch Addition und Verschieben der Teilsummen implementiert werden.

Bei Multiplikation mit negativen Operanden bringt die 2er-Komplementdarstellung einen erheblichen Korrekturaufwand mit sich. Deshalb werden i.d.R. die Absolutbeträge der Operanden miteinander multipliziert und das Vorzeichen des Produkts gesondert ermittelt.

Bei der Multiplikation zweier N-langer Wörter hat das Ergebnis die doppelte Wortlänge 2N, da sich die Zahl der Vor- und Nachkommastellen verdoppeln.

In der Regel bleibt beim Festkommaformat die Stellenanzahl F des gebrochenen Anteils immer gleich, so dass es hier zu Rundungsfehler kommt.

→ Division: ist vermutlich die „komplizierteste Grundrechenart"; wobei es wie meist eine Frage der Übung ist ;-)

Händische Division im Dezimalsystem (a la Volksschule):

… identisches Vorgehen im Dualsystem ...

Aufgabe: Führe die Berechnung $34{,}453125_{10} : 5{,}625_{10}$ sowohl im Dezimal- als auch im Binärsystem Schritt-für-Schritt aus:

1.3.2 Gleitkomma-Darstellung

Festkommazahlen stoßen oft -vor allem in mathematisch-naturwissenschaftlichen Bereichen- an die Grenzen vernünftiger Verwendbarkeit, da in der Regel große Zahlenbereiche abgedeckt werden müssen.

Taschenrechner stellen Dezimalzahlen meist im Bereich von 10^{-99} bis 10^{99} mit einer Genauigkeit von 8 … 10 Dezimalstellen dar.

Da die Basis bekannt ist, braucht intern nur M und E abgespeichert werden (einfaches Speicherlayout):

Damit auch negative Zahlen und Zahlen mit kleinen Beträgen (<< 1) dargestellt werden können, müssen M und E vorzeichenbehaftet sein. Hier kann die Zweierkomplementdarstellung verwendet werden. Um beim Exponent ohne Vorzeichen arbeiten zu müssen (sofern gewünscht), wurde der Bias-Wert B (auch als Offset OS oder Charakteristik C bezeichnet) eingeführt. Dabei wird zu E ein Konstanter Bias-Wert hinzuaddiert, so dass die Summe immer ≥ 0 ist:

Beispiel: $B = 127$ $-126 \leq E \leq 127$

Wie bereits bekannt, kann in der Gleitkomma-Darstellung eine Zahl auf verschiedene Arten dargestellt werden:

$12{,}3456 =$

In der Gleitkomma-Darstellung kann eine Zahl auf verschiedene Arten dargestellt werden. Bei der Normalisierung wird die Mantisse in eine normierte Form gebracht, wobei in der Praxis die folgenden zwei Varianten sehr oft anzutreffen sind:

Variante 1:

Normierungsvorschrift: 1. Ziffer nach dem Komma $\neq 0$, Ziffer(n) vor dem Komma $= 0$

z.B. $12{,}3456 \rightarrow 0{,}123456 \cdot 10^2$

Wertebereich der Mantisse: $\frac{1}{b} \leq M < 1$ (z.B. Basis 10: $0{,}1 \leq M < 1$)

Die Zahl 0 wird durch 0,00… mit beliebigem Exponenten dargestellt. Werden negative Zahlen (bei der Mantisse) im Zweierkomplement dargestellt, so gilt:

1. Binärstelle vor dem Komma = 0 → positive Zahl

1. Binärstelle vor dem Komma = 1 → negative Zahl

Diese Normierungsvorschrift führt zu einer <u>optimalen Ausnutzung der Mantisse</u>.

Variante 2:

Normierungsvorschrift: 1.Ziffer vor dem Komma ≥ 1.

z.B. 12,3456 →

Wertebereich der Mantisse: $1 \leq M < b$ (z.B. Basis 10: $1 \leq M < 10$)

Diese Normierungsvariante wird sehr oft in der Informatik (und damit auch in der µP-Technik) eingesetzt und ist in der IEEE-Norm 754-2008 beschrieben. Diese Norm sieht (unter Anderem) Gleitkommazahlen mit einer Länge von

16 Bit : Minifloat
32 Bit : single precision
64 Bit : double precision
128 Bit

vor. *(Anmerkung: für 80 Bit wurde der Begriff „extended precision" verwendet)*

Bei binären Gleitkommazahlen ist die Basis b=2 und der Zeichenvorrat auf {0,1} begrenzt. Aufgrund der Normierungsvorschrift ist die erste Stelle der Mantisse immer eine „1" - daher wird dieses Bit nicht zusätzlich abgespeichert. Das IEEE-Format erlaubt auch die Darstellung von Sonderfällen. Auswahl einiger Sonderfälle:

+∞, -∞ … bei einem arithmetischen Überlauf; d.h. beim Über-/Unterschreiten des Wertebereichs
+0, -0
NaN … Not a Number (keine Zahl); z.B. als Ergebnis bei 0/0 oder ∞-∞

Ebenfalls spezifiziert die IEEE-Norm Bias-Werte, um negative Exponentendarstellungen zu vermeiden.

Darstellung einer Zahl Z im IEEE 754-Format:

Vorzeichen S (1Bit) *Basis b = 2*

Typ	Länge	Exponent	Mantisse	Bias-Wert	E-Bereich
single	32 bit	8 bit	23 bit	127	-126 … 127
double	64 bit	11 bit	52 bit	1023	-1022 … 1023

Im µP-Umfeld unterstützen viele Compiler nur das single-Format, auch wenn der Datentyp

„double“ zu verwenden ist.

Im Speicher wird die IEEE codierte Gleitkommazahl in der Regel im folgenden Format abgelegt:

wobei entweder 4 Byte (= 32 bit, single precision) oder 8 Byte (= 64 bit, double precision) benötigt werden. Anstelle des binären Musters wird oft die Darstellung im Hexadezimalsystem vorgenommen.

Aufgaben:

a) Im Speicher steht folgendes, 32bit-stelliges Bitmuster

11101011101101010000000000000000

Um welche IEEE-codierte Gleitkommazahl handelt es sich hier?

b) Welche IEEE single precision Gleitkommazahl stellt das folgende Bitmuster dar?

01000001010101010010000000000000

… und wie würde das Bitmuster bei double precision aussehen?

c) Bei der C-Codezeile

```
float a = 148.75;
```

generiert der Compiler folgendes Ergebnis:

4314C000h

Überprüfe dieses Ergebnis auf Richtigkeit.

d) Ermittle die (betragsmäßig) größte und die kleinste darstellbare Gleitkommazahl bei IEEE single precision.

Bei der Gleitkomma-Darstellung gilt...

Bei der Konvertierung entstehende Wandlungsfehler (aufgrund der begrenzten Stellenzahl der Mantisse bzw. so lässt sich z.B. der Dezimalbruch 0.1 nicht exakt im Binärsystem darstellen).

Beispiel Excel: Ein Zellenwert wird jeweils um 0.01 verringert

991	0,1
992	0,09
993	0,08
994	0,07
995	0,06
996	0,05
997	0,04
998	0,03
999	0,02
1000	0,01
1001	2,0961E-13
1002	-0,01
1003	-0,02
1004	-0,03
1005	-0,04
1006	-0,05
1007	-0,06
1008	-0,07
1009	-0,08
1010	-0,09
1011	-0,1
1012	-0,11

Bei der Programmierung ist daher besonders aufzupassen, wenn Gleitkommazahlen auf Gleichheit (==) oder Ungleichheit (!=) überprüft werden müssen (→ Fehlerquelle, die meist aufwändig zu lokalisieren ist !).

Dieser Wandlungsfehler ist im Gleitkommaformat nicht mehr proportional zur abgeschnittenen Stellenanzahl, sondern auch von der Größenordnung des Maßstabsfaktors (Exponent E bzw. e) abhängig.

Unterhalb der kleinsten, positiven darstellbaren Zahl kann kein Wert mehr dargestellt werden. In der Regel wird dann der Wert 0 verwendet. In diesem Fall spricht man vom sog. **Unterlauf**.

Arithmetik im Gleitkommaformat:

… es müssen beide Teile (M, Exponent) getrennt verarbeitet werden → aufwendigere Operationsabläufe.

→ Addition und Subtraktion: Nur die Mantissen zweier Zahlen mit gleichem Exponent/Bias dürfen / können addiert bzw. subtrahiert werden.

Vorgang:

a) Durch Linksverschiebung des Kommas wird der Operand mit der kleineren Charakteristik so umgeformt, dass beide Operanden die gleiche (= größere) Charakteristik besitzen

b) Durchführen der Addition / Subtraktion (Vorzeichenbit → 2er Komplement) wobei die Charakteristik unverändert bleibt

c) Ergebnis erhält man durch Verschieben des Kommas, d.h. nach der Normalisierung.

Aufgaben: Addiere $Z_1 + Z_2$ im Gleitkommaformat; der Einfachheit gelte ein IEEE 754 - Format mit 4bit Mantisse, 3bit Exponent und Bias=0 (vereinfachtes Speicherlayout).

Rechne das Ergebnis wieder zurück ins Dezimalsystem um zu überprüfen, ob Wandlungsfehler entstanden sind.

a) $Z_1 = 2{,}5_{10}$

$Z_2 = 1{,}25_{10}$

b) $Z_1 = 2{,}5_{10}$

$Z_2 = 1{,}0625_{10}$

Daher: bei der Addition / Subtraktion einer betragsmäßig viel kleineren Zahl ändert sich die größere Zahl nicht!

→ Multiplikation und Division:

… erfolgt ebenfalls in mehreren Schritten:

a) Die Mantissen werden multipliziert (dividiert). Bei negativen Mantissen sind die Beträge zu multiplizieren (dividieren) und eine getrennte Vorzeichenbetrachtung ist durchzuführen.

b) Die Charakteristiken werden addiert (subtrahiert). Da dabei jedoch ein B zu viel addiert (subtrahiert) wird, ist dies zu korrigieren. Wird ohne Bias sondern direkt mit Exponent gearbeitet, so sind diese zu addieren (subtrahieren).

c) Normalisierung (d.h. Verschiebung des Kommas und Anpassen der Charakteristik / des Exponenten)

Im Allgemeinen gelten bei Addition (und damit auch bei der Subtraktion) sowie der Multiplikation (und damit auch bei der Division) von Gleitkommazahlen…

… nicht mehr die Assoziativgesetze:

$$(x+y)+z \neq x+(y+z)$$
$$(x\cdot y)\cdot z \neq x\cdot(y\cdot z)$$

… nicht mehr die Distributivgesetze:

$$x\cdot(y+z) \neq (x\cdot y)+(x\cdot z)$$
$$(x+y)\cdot z \neq (x\cdot z)+(y\cdot z)$$

Anhang – Codes

Dezimalzahl	Dualcode	BCD-Code	Gray-Code	ASCII - Code
0	0000	0000	0000	011 0000 ≡ 30h
1	0001	0001	0001	011 0001 ≡ 31h
2	0010	0010	0011	011 0010 ≡ 32h
3	0011	0011	0010	011 0011 ≡ 33h
4	0100	0100	0110	011 0100 ≡ 34h
5	0101	0101	0111	011 0101 ≡ 35h
6	0110	0110	0101	011 0110 ≡ 36h
7	0111	0111	0100	011 0111 ≡ 37h
8	1000	1000	1100	011 1000 ≡ 38h
9	1001	1001	1101	011 1001 ≡ 39h
10	1010	0001 0000	1111	0110001 0110000 ≡ 31 30h
11	1011	0001 0001	1110	0110001 0110001 ≡ 31 31h
12	1100	0001 0010	1010	0110001 0110010 ≡ 31 32h
13	1101	0001 0011	1011	0110001 0110011 ≡ 31 33h
14	1110	0001 0100	1001	0110001 0110100 ≡ 31 34h
15	1111	0001 0101	1000	0110001 0110101 ≡ 31 35h