Zahlensysteme

Die Hauptaufgabe einer Datenverarbeitungsanlage besteht in der Übertragung von Daten. Diese Aufgabe übernimmt der `Datenbus`, welcher als Minimum 8 Leitungen besitzt. Dadurch stehen 8 Bit (1 Byte) für die binäre Zahldarstellung zur Verfügung.
In den heutigen Rechnern verwendet man in der Regel 32 bzw. 64 Bit. Die der Busbreite entsprechende Bitanzahl kann also je nach Recheneinheit variieren. Ein Datum aus 16 Bit bezeichnet man als `Wort`, eines aus 32 Bit als Langwort.
Das herkömmliche dekadische Zahlsystem ist für die Datenübertragung unzureichend. Die Übertragung beinhaltet im Wesentlichen die Zustände { an | aus }, weshalb das `Dualsystem` verwendet wird, welches auf einer Repräsentation durch $'0'$ und $'1'$ basiert.

Stellenwertsysteme

Die Zahlsysteme, welche im Wesentlichen in der Informationsverarbeitung Verwendung finden, sind neben dem Dualsystem das Oktal - (Achtersystem) und das Hexadezimalsystem (Sechzehnersystem).

Die Menge der Grundziffern stellt sich wie folgt dar.

Dualsystem	0,1
Oktalsystem	0,1,2,3,4,5,6,7
Hexadezimalsystem	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Umrechnung und Darstellung

Die Umrechnung einer Binaerzahl in eine Dezimalzahl erfolgt im Dualsystem wie folgt .

$$1101_{bin} = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 13_{dez}$$

Die Umwandlung von Zahlen des Oktal - bzw. Hexadezimalsystems in Dezimalzahlen erfolgt abhängig von der Basis auf vergleichbare Weise. Eine elegantere Methode der Umwandlung stellt das `Hornerschema (Restwertmethode)` dar. Danach ist es möglich, die Binärdarstellung der Zahl 13_{dez}

$$1101_{bin} = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

auch durch fortgesetztes Ausklammern der Basis `2` in folgende Form zu bringen

$$1101_{bin} = (1 \cdot 2^2 + 1 \cdot 2^1 + 0) \cdot 2 + 1$$

$$1101_{bin} = ((1 \cdot 2^1) + 1) \cdot 2 + 0) \cdot 2 + 1$$

Dadurch wird die Umwandlung erheblich einfacher und 100111_{bin} berechnet sich als

$$100111 = (((((1 \cdot 2) + 0) \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1 = 39_{dez}$$

Gleichzeitig steckt in diesem Ansatz die Möglichkeit der Umwandlung einer Dezimalzahl in eine Binärzahl. Da die Umkehrung der fortlaufenden Ausklammerung die fortlaufende Division durch die Basis darstellt, ergibt sich die Umwandlung von 39_{dez} in eine Binärzahl durch folgendes Vorgehen.

39	:	2	=	19	Rest 1
19	:	2	=	9	Rest 1
9	:	2	=	4	Rest 1
4	:	2	=	2	Rest 0
2	:	2	=	1	Rest 0
1	:	2	=	0	Rest 1

Als Ergebnis entsteht die Zahl 100111_{bin} von unten gelesen.

Da die Darstellung der Nachkommastellen einer Dezimalzahl durch die Basis 2^{-1} realisiert wird, liefert die Multiplikation mit 2 die gewünschte Binärdarstellung.

Ein Beispiel sei die Zahl 0.75_{dez}.

0.75	·	2	=	1.5 abgespalten 1
0.5	·	2	=	1.0 abgespalten 1

Als Ergebnis entsteht die Zahl 0.11_{bin} von oben gelesen.

Die Umwandlung von Dezimalzahlen in Oktal - oder Hexadezimalzahlen kann ebenfalls durch fortgesetzte Division bewerkstelligt werden. Man kann hierbei jedoch auch das Binärsystem verwenden. Diese Vorgehensweise sei im Folgenden für die Zahl 643_{dez} sowohl für das Oktal -als auch das Hexadezimalsystem beschrieben.

In einem ersten Schritt wird die Zahl in eine Binärzahl überführt. Daraus ergibt sich $643_{dez} = 1010000011_{bin}$.

Jetzt ist entscheidend, ob die Zahl ins Oktalsystem `oder` in das Hexadezimalsystem überführt werden soll. Im ersten Fall muss die Länge der Binärdarstellung eine durch 3 teilbare Zahl sein (Basis $2^3 = 8$) im zweiten Fall eine durch 4 teilbare Zahl ($2^4 = 16$). Im Anschluss daran erfolgt eine Umwandlung der einzelnen Binärblöcke entsprechend der Grundziffern der Darstellung (siehe oben).

Oktalsystem	: 001 010 000 011	$= 1203_{okt}$
Hexadezimalsystem	: 0010 1000 0011	$= 283_{hex}$

In beiden Fällen bestand die Notwendigkeit des Aufüllens von `Nullen` zur Herstellung der Blöcke. Eine Überprüfung der Richtigkeit der Ergebnisse ist über das `Hornerschema` möglich.

Aufgabe 1 Zahlenkonvertierungen

(a) Wandeln Sie folgende Zahlen um

- $1100111101_{bin} \mapsto$ okt, hex, dez
- $1D312_{hex} \mapsto$ okt, bin, dez
- $12.675_{dez} \mapsto$ bin

(b) Berechnen Sie den Dezimalwert von 723_{okt} mittels des Hornerschemas.

(c) Die nachfolgende Methode wurde der Klasse `binaer` entnommen. Beschreiben Sie deren Funktionalität und führen Sie einen Trockendurchlauf zum Test durch.

Zusatz : Erweitern Sie die Funktionalität durch weitere Methoden.

```
4  def dez2bin (a):
5
6        if a == 0 : return ""
7        else    : return dez2bin (a / 2) + str (a % 2)

16 def bin2dez (a) :
17       i = 0
18       x = 0
19       y = len (a) -1
20       while i < len (a) :
21             x = x + int (a [i]) * (2**(y))
22             i += 1
23             y -= 1
24       return str(x)
```

Binäres Rechnen

- Binäre Addition

  Die binäre Addition genügt den folgenden Rechenregeln.

0	+	0	=	0	
1	+	0	=	1	
0	+	1	=	0	
1	+	1	=	0	Übertrag 1

  Gerechnet wird wie im normalen Dezimalsystem.

	1	1	1	0	0
+	1	1	1	1	0
Ü	1	1			
1	1	1	0	1	0

- Binäre Subtraktion

  Die binäre Subtraktion genügt den folgenden Rechenregeln.

0	-	0	=	0	
1	-	0	=	1	
1	-	1	=	0	
0	-	1	=	0	Übertrag -1

  Auch hier wird vergleichbar gerechnet. Allerdings bestimmt man den Rest zu 2 also zu 10_{bin}

	1	1	1	0	0
-				1	0
Ü			1		
	1	1	0	1	0

  In der Praxis verwendet man diese Methode der Multiplikation nicht, sondern versucht die Subtraktion auf die Addition zurückzuführen. Dazu ist es als erstes notwendig, negative Binärzahlen einzuführen. Die Unterscheidung, ob eine Binärzahl positiv oder negativ ist erfolgt über das erste Bit.

  Hat dieses Bit den Wert $'1'$ ist die Zahl negativ, ist der Wert des Bits $'0'$ ist die Zahl positiv. Da es sich um das Bit mit dem höchsten Stellenwert handelt trägt das Bit die Bezeichnung `msb` (most signifikant bit). Geht man von einer festen Busbreite, z. B. $n = 8$ Bit aus stehen also für die Darstellung der Zahl noch 7 Bit zur Verfügung, also ein Bereich von $0 - 2^{n-1}$, im Fall von 8 Bit also 0 - 127.

  Um die Menge der negativen Zahlen zu erhalten, dient die Zahl $127 = 1111111_{bin}$
  oder allgemein

  $$k = 2^n - 1$$

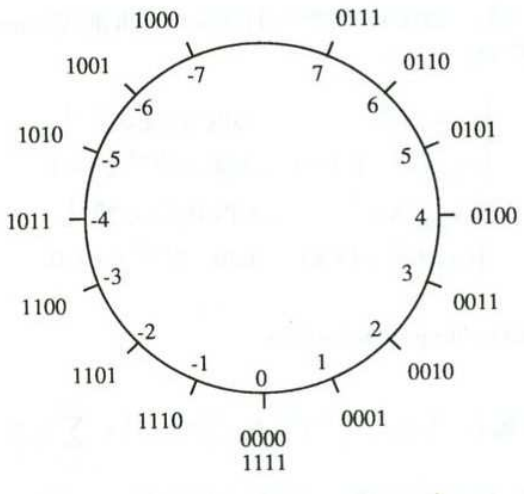
  Um nun negative Zahlen darzustellen, bildet man das `Stellenkomplement` oder Einerkomplement der Zahl

  $$\overline{z} = k - z$$

Als Beispiel 5 und -5

	1	1	1	1	1	1	1
-	0	0	0	0	1	0	1
					0		
	1	1	1	1	0	1	0

Um das `Stellenkomplement` zu bilden, ist es notwendig alle Bits zu 'kippen '.
Die folgende Darstellung verdeutlicht die Komplementbildung graphisch.



In der Darstellung fällt auf, das es 2 Darstellungen für die Zahl Null gibt. Die Eins wäre in der Stellenkomplementdarstellung mit $n = 8$ die Zahl 00000001_{bin} und die -1 die Zahl 11111110_{bin} Die Summe müsste Null ergeben, macht Sie aber nicht. Damit ergeben sich 2 differente Darstellungen für die Zahl Null.

Man behilft sich durch die Einführung des `Zweierkomplementes`. Dieses Zweierkomplement beinhaltet die Bildung des Einerkomplementes und die anschließende Addition von 1.

Beispiel für 6 mit n = 4

Binaerzahl	1100
Einerkomplement	0011
Zweierkomplement	0100

Für das oben stehende Problem bedeutet dies für $n = 4$ ist die Zahl $1_{10} = 0001_{bin}$, deren Zweierkomplement 1111. Als Summe von -1 und $+1$ ergibt sich 10000. Da wir aber die Bitbreite mit $n = 4$ festgelegt haben, also eine feste Stellenzahl , führt das erste Bit zu einem Überlauf und es bleibt die Null stehen.

Exkurs

Die Problematik des Überlaufs bewirkt generell, das die Addition in einem festen Bereich k bei fortlaufender Addition von 1 bei Erreichen von k wieder bei 1 beginnt.

Im Folgenden nun Beispiele für das Rechnen mit dem Zweierkomplement für $n = 8$.

Beispiel 1 $25 - 13$

		0	0	0	1	1	0	0	1
Zweierkomplement von 13 $\rightsquigarrow 1101_{bin}$	+	1	1	1	1	0	0	1	1
		1	1	1			1	1	
Ergebnis (msb = 0)	1	0	0	0	0	1	1	0	0

Die ‘übergelaufene ‘1 entfällt, das Ergebnis ist +12 , da das `msb = 0`

Beispiel 2 $13 - 25$

		0	0	0	0	1	1	0	1
Zweierkomplement von 25 $\rightsquigarrow 11001_{bin}$	+	1	1	1	0	0	1	1	1
					1	1	1	1	
Zwischenergebnis (msb = 1)		1	1	1	1	0	1	0	0
Zweierkomplement Zwischenergebnis		0	0	0	0	1	1	0	0

Das Ergebnis beträgt -12.

Tritt bei der Addition des Zweierkomplementes ein Überlauf auf, ist das msb = 0, das Ergebnis also positiv. Im anderen Fall ist das msb =1, die Berechnung ist also noch nicht beendet, es muss das Zweierkomplement des Zwischenergebnisses berechnet werden.

- Binäre Multiplikation

$$\begin{array}{lcll} 0 & \cdot & 0 & = & 0 \\ 1 & \cdot & 0 & = & 0 \\ 1 & \cdot & 1 & = & 1 \\ 0 & \cdot & 1 & = & 0 \end{array}$$

Die Multiplikation für Binärzahlen entspricht im wesentlichen dem Vorgehen in anderen Stellenwertsystemen.

$$\begin{array}{cccccc} 1 & 1 & * & 1 & 0 & \\ 1 & 1 & & & & \\ & 0 & 0 & & & \\ \hline 1 & 1 & 0 & & & \end{array}$$

In der Schaltechnik wird die Multiplikation durch die Verbindung von Addition und Verschieben (`shift`) realisiert. Im oberen Beispiel wurde die Zahl 11_{bin} um 1 Stelle (Multiplikation mit 2^1) nach rechts verschoben und eine Null angefügt. Eine Multiplikation mit 2^n bewirkt also im ein n-faches Shiften nach rechts.

Aufgabe 2 Binäre Arithmetik

(a) Bestimmen Sie das Einer und Zweierkomplement der Zahlen $23, -12$ für $n = 8$

(b) Begründen Sie weshalb eine Unterscheidung zwischen negativen und positiven Zahlen allein ausgehend vom Wert des `msb` unzureichend ist.

(c) Berechen Sie $234 - 546$ und $1234 - 34$ mittels der Zweierkomplementbildung.

(d) Zusatz

Implementieren Sie die binäre Addition und Subtraktion mittels Zweierkomplement in Python.

Gleitpunktzahlen - Gleitkommazahlen

Der Nachteil der Festkommadarstellung, wie sie in den oberen Beispielen ersichtlich, besteht darin, dass in der Realität sehr große bzw. sehr kleine Zahlen mit dieser Technik nicht dargestellt werden können. Die Zahl 1200000 würde zum Beispiel 7 Bits benötigen.

Eine andere bekannte Darstellung $1.2 \cdot 10^6$ benötigt weitaus weniger Bits. Darstellungen dieser Art bezeichnet man als Gleitpunktdarstellungen. Der Multiplikator bezeichnet dabei die sogenannte `Mantisse`, 6 den Exponenten der Darstellung und 10 die `Basis`. Trotz der erforderlichen Rundung ist die darstellbare Menge der Zahlen der Grund für die Verbreitung. Zur Darstellung von Gleitpunktzahlen wurde ein Standard entwickelt, den man als IEEE 754 bezeichnet.Die grundlegende Darstellung einer Gleitpunktzahl ist Folgende,

$x = m \cdot b^e$

m .. Mantisse
b .. Basis
e .. Exponent

Eine Gleitkommazahl nach IEEE 754 weist dazu einen vorgegebenen Aufbau auf. Sie besteht aus einem Vorzeichen, e Bit zur Darstellung des Exponenten, m Bit zur Darstellung der sogenannten normierten Mantisse. Normierung bedeuted nach diesem Standard, das die Mantisse so normalisiert wird, dass sie eine führende 1 vor dem Komma aufweist, die dann nicht kodiert wird. Da auch negative Exponenten auftreten können, beinhaltet die Konvertierung einer Dezimalzahl in eine Gleitkommazahl die Verschiebung des Nullpunktes in die Mitte des möglichen Bereiches.
Insgesamt stehen bei 32 Bit folgende Bereiche zur Verfügung.

	Vorzeichen	Exponent	Mantisse
Bit	31	30 - 23	22 - 0

Im Wesentlichen stellt sich die Konvertierung wie folgt dar.

Beispiel 166.125_{dez}

1.Schritt : Umwandlung in eine Binärzahl

$$1234.625_{dez} = 10011010010,101$$

2.Schritt : Normierung der Mantisse

$$10011010010,101_{bin} = 1.0011010010 \cdot 2^{10}$$
Shifting um 10 Stellen nach links

3.Schritt : Verschiebung des Exponenten (+127)

$$137_{dez} = 10001001_{bin}$$

4.Schritt :

Auffüllen der Mantisse mit Nullen und Angabe der Gleitkommazahl im IEEE Standard

	Vorzeichen	Exponent	Mantisse
Gkz	0	10001001	00110100100000000000

Aufgabe 3

Berechnen Sie die Gleitpunktzahl für 166.25 für n = 32 nach IEEE 754 Standard und überprüfen Sie die Richtigkeit ihrer Berechnung durch Probe.

Literatur

Hartmut Ernst	Grundlagen und Konzepte der Informatik (2000)
Stucky	Der Rechner als System (1997)
Oberschelp,Vossen	Rechnerstrukturen (1998)