Dr. Jürg M. Stettbacher
Neugutstrasse 54
CH-8600 Dübendorf

Telefon: +41 43 299 57 23
Fax: +41 43 299 57 25
E-Mail: dsp@stettbacher.ch

Zahlensysteme und Codes

Lösungen zu den Übungsaufgaben

Version 2.22
2015-09-01

Zusammenfassung: Dieses Dokument enthält die vollständigen Lösungen zu den Übungsaufgaben im Skript Zahlensysteme und Codes - Konzept und Verwendung in der Informatik.

1 Übungsaufgaben

1.1 Rechnen in Zahlensystemen

(a) Operanden im Binärformat:
Den Betrag der Dezimalzahlen kann man mit Hilfe des Hornerschemas ins Binärformat umwandeln. Das geht beispielsweise für die Zahl 21_d so:

$$\begin{array}{rcrcrcl} 21_d & \div & 2 & = & 10 & \text{Rest} & 1 \\ 10_d & \div & 2 & = & 5 & \text{Rest} & 0 \\ 5_d & \div & 2 & = & 2 & \text{Rest} & 1 \\ 2_d & \div & 2 & = & 1 & \text{Rest} & 0 \\ 1_d & \div & 2 & = & 0 & \text{Rest} & 1 \end{array}$$

Auslesen von unten nach oben ergibt $21_d = 10101_b$. Weiter erhalten wir:

$$\begin{array}{lrcl} & 9_d & = & \ldots \mathbf{0\ 0\ 0\ 1\ 0\ 0\ 1}_b \\ \text{1-er Komplement} & & & \ldots \mathbf{1\ 1\ 1\ 0\ 1\ 1\ 0}_b \\ \text{2-er Komplement} & -9_d & = & \ldots \mathbf{1\ 1\ 1\ 0\ 1\ 1\ 1}_b \end{array}$$

Nun führen wie die Addition aus:

$$\begin{array}{rcccccccc} & \ldots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1}_b \\ + & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1}_b \\ \hline & \ldots & 1 & 1 & & 1 & 1 & 1 & \\ & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0}_b \end{array}$$

Das Resultat können wir leicht kontrollieren. Wir erhalten, wie erwartet:

$$1100_b = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 12_d$$

(b) Zuerst müssen wir die Binärzahlen in Hexadezimalzahlen umwandeln. Das können wir in Vierergruppen tun, denn jede Hexadezimalziffer umfasst gerade vier Bit:

1101_b	0010_b	1001_b	und	0010_b	0101_b	1000_b
D_h	2_h	9_h		2_h	5_h	8_h

Damit folgt die Subtraktion:

$$\begin{array}{rccccccc}
 & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{D} & \mathbf{2} & \mathbf{9} & {}_h \\
\mathbf{-} & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{2} & \mathbf{5} & \mathbf{8} & {}_h \\
\hline
 & \ldots & & & 1 & & & \\
 & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{A} & \mathbf{D} & \mathbf{1} & {}_h
\end{array}$$

Das Resultat der Subtraktion ist die positive Zahl $AD1_h$.

(c) Wir finden: $7_d = \ldots 00111_b$ und $11_d = \ldots 001011_b$. Die Zahl -11_d erhalten wir mit dem 2-er Komplement:

$$\begin{array}{lrcccccccc}
 & 11_d & = & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & {}_b \\
\text{1-er Komplement} & & & \ldots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & {}_b \\
\text{2-er Komplement} & -11_d & = & \ldots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & {}_b
\end{array}$$

Beim schriftlichen Berechnen der Multiplikation müssen wir darauf achten, dass wir insbesondere die negativen Operatoren in der Breite des Resultats schreiben. Hier haben wir, inkl. von mindestens einem Vorzeichenbit, 5 Bit in den Operatoren. Das Resultat kann rund doppelt[1] so breit werden, also wählen wir 10 Bit.

$$\begin{array}{lccccccccccccc}
\mathbf{\ldots} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & {}_b & \mathbf{x} & \mathbf{\ldots} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & {}_b \\
\hline
 & & & & & & & & \mathbf{\ldots} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \\
 & & & & & & & \mathbf{+} & \mathbf{\ldots} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & & \\
 & & & & & & & \mathbf{+} & \mathbf{\ldots} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & & & \\
\hline
\text{Übertrag} & & & & & & & & & 2 & 0 & 2 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & \\
\text{Resultat} & & & & & & & & & \mathbf{\ldots} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & {}_b
\end{array}$$

Das Resultat ist negativ. Um heraus zu finden, welchen Wert es hat, bilden wir wieder das Komplement:

[1] Genau genommen hat das Resultat maximal die Breite $2N-1$, wenn die vorzeichenbehafteten Operanden die Breite N haben.

	...	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**1** $_b$
1-er Komplement	...	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0** $_b$
2-er Komplement	...	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1** $_b$

Damit erhalten wir $\ldots 01001101_b = 2^6 + 2^3 + 2^2 + 2^0 = 77_d$. Das Resultat der Multiplikation ist demnach -77_d, was wir schon gedacht hatten.

(d) Wir verwenden die Komplementdarstellung im gegebenen Zahlensystem:
- Binärsystem, 2-er Komplement: $-11'0111'1000_b = \ldots 11'1100'1000'1000_b$
- Zehnersystem, 10-er Komplement:$-87.25_d = \ldots 99912.75_d$

Anhand des zweiten Beispiels wollen wir zeigen, wie man das Resultat verifizieren kann. Dazu addieren wir zur negativen Zahl beispielsweise den Wert 100_d. Das Resultat müsste $100.00 - 87.25 = +12.75$ geben.

	...	**0**	**0**	**0**	**1**	**0**	**0.**	**0**	**0** $_d$
+	...	**9**	**9**	**9**	**9**	**1**	**2.**	**7**	**5** $_d$
	...	1	1	1					
	...	**0**	**0**	**0**	**0**	**1**	**2.**	**7**	**5** $_d$

1.2 Endliche Zahlen

(a) Die Zahl $1100'1001_b$ kann zwei Dinge bedeuten, je nachdem, ob im entsprechenden Kontext der Inhalt des Registers als vorzeichenlos oder vorzeichenbehaftet interpretiert wird. Im vorzeichenlosen Fall bedeutet die Zahl:

$$11001001_b = 2^7 + 2^6 + 2^3 + 2^0 = 201_d$$

Wird der Registerinhalt als vorzeichenbehaftete Zahl angesehen, so ist sie negativ. Ihren Wert finden wir über das Komplement.

	...	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1** $_b$
1-er Komplement	...	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0** $_b$
2-er Komplement	...	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**1** $_b$

Der Betrag ist dann $110111_b = 2^5 + 2^4 + 2^2 + 2^1 + 2^0 = 55_d$. Im Register steht also der Wert -55_d.

(b) Falls das Register vorzeichenbehaftete Zahlen enthält, so käme es bei der Addition von $100_d + 100_d$ zu einem Überlauf, denn in 8 Bit lassten sich nur Zahlen von -128_d bis $+127_d$ darstellen, resp. von $1000'0000_b$ bis $0111'1111_b$.

(c) Die vorzeichenlose Subtraktion von $0-1$ in 8 Bit ergibt $1111'1111_b = 255_d$. Das Resultat ist demnach falsch. Der Prozessor wird allerdings das *Borrow*-Flag setzten, das auf den Über- oder besser Unterlauf hinweist. In vielen Prozessoren heisst das Flag *Carry* und wird für Überläufe bei Additionen und Unterläufe bei Subtraktionen verwenden.

1.3 Allgemeine Codes

(a) Die binäre Zahl 110011_b entspricht 51_d. Wir benötigen also die zwei BCD-Ziffern 5 und 1. Folglich ist:

$$110011_b = 0101'0001_{BCD}$$

(b) Die BCD-Zahl $0010'1001_{BCD}$ besteht aus 2 Zehnern und 9 Einern:

$$\begin{aligned} 0010'1001_{BCD} &= 2_d \cdot 10_d + 9_d \cdot 1_d \\ &= 29_d \\ &= 11101_b \end{aligned}$$

Man kann das auch direkt binär ausrechnen:

$$\begin{aligned} 0010'1001_{BCD} &= 0010_b \cdot 1010_b + 1001_b \cdot 0001_b \\ &= 10100_b + 1001_b \\ &= 11101_b \end{aligned}$$

(c) Wenn das Code-Rad vorwärts dreht: $011 \implies 010$.
Wenn das Code-Rad rückwärts dreht: $011 \implies 001$.

(d) Den hexadezimalen Wert zum ASCII Code 100_d finden wir zum Beispiel mit dem Horner Schema:

$$\begin{aligned} 100_d &\div 16 = 6 \text{ Rest } 4 \\ 6_d &\div 16 = 0 \text{ Rest } 6 \end{aligned}$$

Also $100_d = 64_h$. Damit gehen wir in die ASCII-Tabelle und finden das Zeichen:

$$\text{ASCII}(64_h) = \text{d}$$