1.4 Einführung und Zahlensysteme/Darstellung negativer Zahlen

1.4.1 Wozu negative Zahlen?

Negative Zahlen kommen in Programmen häufig in zwei Zusammenhängen vor:

- Man will Größen darstellen, die positive oder negative Werte annehmen können: Kontostände, Spannungen, Koordinaten usw.
- Negative Zahlen ersparen das Subtrahieren: $20 - 5 = 20 + (-5) = 15$, damit kann man sich ein Subtrahierwerk in der CPU ersparen (siehe Abbildung 1).

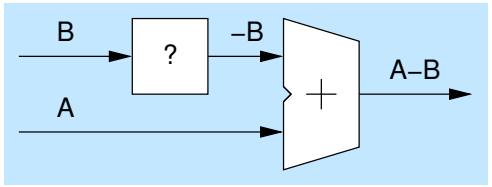

Abbildung 1: Addition statt Subtraktion

1.4.2 Betrag-und-Vorzeichen-Darstellung

Die erste Darstellungsmöglichkeit für negative Zahlen, die den meisten Menschen einfällt, ist die getrennte Darstellung mit Betrag und Vorzeichen wie in Tabelle 1 gezeigt. Hier wird das vordere

b3	b2	b2	Wert
0	0	0	+0
0	0	1	+1
0	1	0	+2
0	1	1	+3
1	0	0	-0
1	0	1	-1
1	1	0	-2
1	1	1	-3

Tabelle 1: Betrag-und-Vorzeichen-Darstellung

Bit vom höchstwertigen Bit zum Vorzeichen gemacht. Die Bildung einer negativen Zahl ist damit sehr einfach: 010 ist 2, 110 ist -2. Wie kann man nun hiermit subtrahieren? Am Beispiel 3+(-2) soll das -versucht werden:

	0	1	1
+	1	1	0
U	1	0	1

Der erste Versuch einer Addition (wie in Abbildung 1 gefordert) schlägt fehl. Man muss also diese Addition auf eine Subtraktion zurückführen:

	0	1	1
-	0	1	0
	0	0	1

Nun braucht man also für die Addition von -2 einen Subtrahierer. Dabei sollte das Subtrahierwerk doch eigentlich eingespart werden.

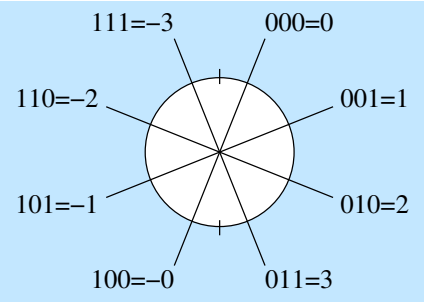

Abbildung 2: Betrag-und-Vorzeichen-Darstellung im Zahlenkreis

Als nächsten Test für diese Darstellung kann man versuchen, von -3 nach +3 hochzuzählen. Eine Faustregel besagt: Wenn das Zählen einfach ist bei einem Zahlensystem, dann ist auch das Rechnen einfach (und damit schnell). Beim Zählen von -3 nach +3 hat man aber bei 0 einen Sprung und eine Änderung der Zählrichtung. Außerdem ist die 0 doppelt. In Abbildung 2 wird dies mit Hilfe des so genannten *Zahlenkreises* gezeigt.

Fazit: Es ist ein (für den Rechner) ungünstiges System. Es wird deshalb fast nur bei Gleitkommazahlen benutzt.

1.4.3 Offset-Darstellung

Eine wesentlich bessere Methode ist die Offset-Darstellung wie in Tabelle 2 gezeigt. Hier gibt

b3	b2	b2	Wert	
0	0	0	-4	(=0-4)
0	0	1	-3	(=1-4)
0	1	0	-2	(=2-4)
0	1	1	-1	(=3-4)
1	0	0	0	(=4-4)
1	0	1	1	(=5-4)
1	1	0	2	(=6-4)
1	1	1	3	(=7-4)

Tabelle 2: Offset-Darstellung

es kein Vorzeichen. Der Wertebereich der 8 Zahlen wird um die Hälfte (also um 4) nach unten verschoben. Diese Verschiebung nennt man Offset. Damit ist die Hälfte der Zahlen negativ. Wenn man von -4 hochzählt, gibt es keinen Sprung mehr in der Mitte des Wertebereichs und auch die Zählrichtung ist immer gleich. Es gibt nur noch eine 0. Auch hier soll wieder das Beispiel 3+(-2) gerechnet werden:

	1	1	1
+	0	1	0
U	0	0	1

Nach dieser (jeder) Addition ist eine Korrektur nötig: Aus 001 muss 101 gemacht werden. Man muss nämlich das höchstwertige Bit negieren, weil das Ergebnis wieder um den Offset 4 berichtigt werden muss. Ansonsten stimmt dieses Resultat. Es handelt sich also um eine ganz gute Darstellung, und tatsächlich wird sie in AD- und DA-Umsetzern benutzt.

1.4.4 Zweierkomplement-Darstellung

Die am häufigsten benutzte Darstellung für negative Zahlen ist die Zweierkomplement-Darstellung wie in Tabelle 3 gezeigt.

b3	b2	b2	Wert	
0	0	0	0	
0	0	1	1	
0	1	0	2	
0	1	1	3	
1	0	0	-4	(=4-8)
1	0	1	-3	(=5-8)
1	1	0	-2	(=6-8)
1	1	1	-1	(=7-8)

Tabelle 3: Zweierkomplement-Darstellung

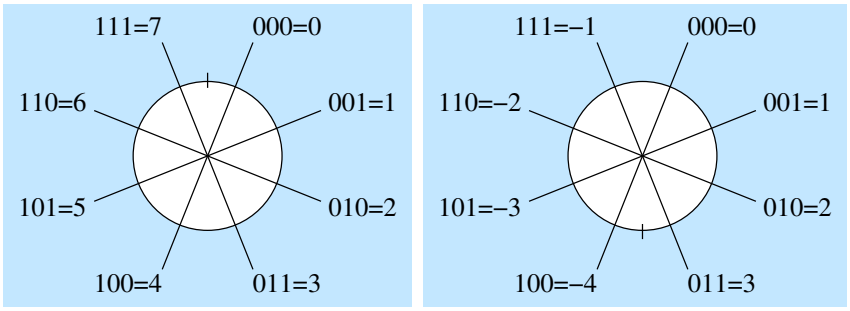

Abbildung 3: Zahlenkreise; links: bisherige nur-positive Zahlen, rechts: Zweierkomplement-Darstellung

Auch hier gibt es kein Vorzeichen. Aber hier wird der Wertebereich der oberen vier Zahlen um den gesamten Bereich (also 8) nach unten verschoben. Damit ist jetzt nicht die erste, sondern die zweite Hälfte der Zahlen negativ.

Wenn man von -4 hochzählt, gibt es auch hier keinen Sprung mehr in der Mitte des Zahlenbereichs — man muss allerdings den Überlauf von 111 nach 000 mit einbeziehen. Abbildung 3 zeigt dies mit Hilfe des so genannten *Zahlenkreises*. Und auch hier ist die Zählrichtung immer gleich.

Auch hier soll wieder das Beispiel 3+(-2) gerechnet werden:

	0	1	1
+	1	1	0
U	0	0	1

Das Ergebnis ist hier also schon ohne Korrektur richtig.[1] Hier handelt es sich also um eine Darstellung, die die Forderung nach Einfachheit am besten berücksichtigt. Für kleine Zahlen (unterhalb der Mitte des Zahlenbereichs) ist sie sogar zu 100% kompatibel zu den bisher bekannten Zahlen im natürlichen Binärcode bzw. im Stellenwertsystem zur Basis 2. Daher wird diese Darstellung in den allermeisten neuzeitlichen Systemen verwendet.

1.4.5 Vorzeichenumkehr in Zweierkomplement-Darstellung

Man hat jetzt eine Möglichkeit zur Subtraktion: A-B=A+(-B). Aber wie kommt man von B nach -B (Wie bildet man das so genannte *Zweierkomplement* von B)? In der Betrag-und-Vorzeichen-Darstellung ist das kein Problem, dort wird einfach das Vorzeichenbit negiert. Bei Zweierkomplement-Darstellung muss es nun eine ähnlich einfache Lösung geben, sonst wäre der Vorteil der einfachen Subtraktion wieder verloren. – Die Lösung sieht so aus:

a) Man negiert jedes Bit der Zahl: 010 wird zu 101.

[1] Das Überlauf-Bit wird hier nicht berücksichtigt. Bei Zweierkomplement-Darstellung werden Übertrag- und Überlauf-Bits anders als bei normalen Zahlen behandelt; die Prozessoren beherrschen normalerweise beides.

b) Man addiert zum Ergebnis +1: 101 wird zu 110.

Man kann sich dies am Zahlenkreis vorstellen (siehe Abbildung 4):

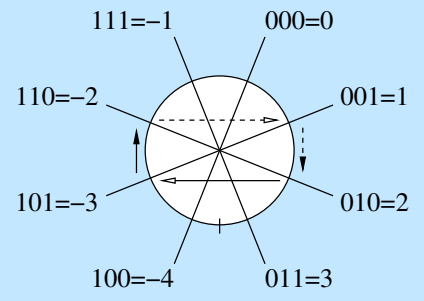

Abbildung 4: Vorzeichenumkehr in Zweierkomplement-Darstellung

a) Schritt 1: Invertieren aller Bits spiegelt den Code am Zahlenkreis (P → N-1),

b) Schritt 2: Addition von eins korrigiert den erhaltenen Wert (N-1 → N).

1.4.6 Zweierkomplement und Wortbreite

Wie kann man Zahlen im Zweierkomplement um zusätzliche Binärstellen (Bits) erweitern?

- `7 =0111→0000.0000.0000.0111= 7` – o.k.
- `-7=1001→0000.0000.0000.1001= 9` – Fehler
- `-7=1001→1111.1111.1111.1001=-7` – o.k.

Ergebnis: Es können stets beliebig viele Stellen links angesetzt werden; sie sind bei null oder positiver Zahl mit Nullen, sonst mit Einsen zu füllen. Anders gesagt: Steht ganz links eine Null, muss man mit Nullen auffüllen, steht dort eine Eins, muss man mit Einsen auffüllen.

Wann und wie kann man Zahlen im Zweierkomplement um Binärstellen reduzieren?

- `7=0000.0000.0000.0111 →0111= 7` – o.k.
- `14=0000.0000.0000.1110 →1110=-2` – Fehler
- `21=0000.0000.0001.0101 →0101= 5` – Fehler
- `-7=1111.1111.1111.1001 →1001=-7` – o.k.
- `-14=1111.1111.1111.0010 →0010=-2` – Fehler
- `-21=1111.1111.1110.1011 →1011=-5` – Fehler

Ergebnis: Reduktion ist nur erlaubt, wenn alle zu reduzierenden führenden Stellen gleich der ersten darauf folgenden Stelle (hier: der viertletzten Stelle) sind. Dann (nur dann) können diese Stellen weggelassen werden.