# Übungsblatt 5

Abgabe: 04.12.2006

## Addition und Subtraktion ganzer Zahlen vom Typ `int`

Die Addition und Subtraktion ganzer Zahlen (wir beziehen uns hier nur auf 32Bit-lange `int`-Repräsentationen) wird heutzutage bei den meisten Rechnern nach folgendem Verfahren realisiert:

**Dualdarstellung für nicht-negative Zahlen:** Ein `int`-Wert $x$ mit Bit-Repräsentation $bits(x) = (b_{31} \dots b_0)$ hat folgende Interpretation als ganze Zahl $x \in \mathbb{Z}$: Ist das *Vorzeichenbit* $b_{31} = 0$, repräsentiert $bits(x) = (b_{31} \dots b_0)$ die nicht-negative Zahl

$$x = \sum_{i=0}^{31} 2^i \cdot b_i \geq 0 \qquad (1)$$

Da $b_{31} = 0$, hat der Summand mit Faktor $2^{31}$ natürlich immer den Wert 0.

**Zweierkomplementdarstellung für negative Zahlen:** Ist für einen `int`-Wert $x$ mit Bit-Repräsentation $bits(x) = (b_{31} \dots b_0)$ das Vorzeichenbit $b_{31} = 1$, repräsentiert $bits(x) = (b_{31} \dots b_0)$ die negative Zahl

$$x = \sum_{i=0}^{31} 2^i \cdot b_i - 2^{32} < 0 \qquad (2)$$

$bits(x)$ heisst die *Darstellung von $x < 0$ im Zweierkomplement.*

Mit den beiden Formeln (1) und (2) haben wir eine *Abstraktionsfunktion*

$$\alpha : \texttt{int} \longrightarrow \mathbb{Z}$$

definiert: Zu gegebenem `int`-Wert wird abhängig vom Vorzeichenbit die zugehörige ganze Zahl $x$ durch Formel (1) oder (2), angewandt auf die Bitrepräsentation des `int`-Wertes berechnet.

**Vorzeichenwechsel:** Für $x \in \texttt{int}$ mit $bits(x) = (b_{31} \dots b_0)$ wird der Vorzeichenwechsel durch den Übergang zur Darstellung

$$K_2(bits(x)) = ((1 - b_{31}) \dots (1 - b_0)) +_2 (\underbrace{0 \dots 0}_{31} 1)$$

realisiert. $K_2(bits(x))$ heisst das *Zweierkomplement* von $bits(x)$. Ist beispielsweise $x = 5$, folgt

$$bits(x) = 00000000\ 00000000\ 00000000\ 00000101$$

und

$$K_2(bits(x)) = 11111111\ 11111111\ 11111111\ 11111011$$

Man rechnet aus (naja, da habe ich kcalc verwendet), dass

$$\alpha(11111111\ 11111111\ 11111111\ 11111011) = -5$$

und das ist natürlich ein Lichtblick an einem dunklen Novembertag. Weiterhin stellen wir beruhigt fest, dass

$$\begin{aligned}
K_2(bits(-5)) &= K_2(11111111\ 11111111\ 11111111\ 11111011) \\
&= (00000000\ 00000000\ 00000000\ 00000100) +_2 (\underbrace{0\ldots0}_{31}1) \\
&= (00000000\ 00000000\ 00000000\ 00000101) \\
&= bits(5)
\end{aligned}$$

Der zweifache Vorzeichenwechsel führt also auf die ursprüngliche Zahl zurück.

**Addierer in der ALU:** Die arithmetische Einheit enthält üblicherweise nur einen Addierer und *keinen* separaten Mechanismus zur Subtraktion. Dieser Addierer realisiert auf 32-Bit Worten $bits(x) = (b_{31} \ldots b_0), bits(y) = (c_{31} \ldots c_0)$ eine Operation $bits(x) +_2 bits(y)$ nach dem Vorbild der in der Schule gelernten Addition mit Übertrag. Beispielsweise wird

$$(00000000\ 00000000\ 00000000\ 10010011) +_2 (00000000\ 00000000\ 00000000\ 10011111)$$

nach folgendem Schema berechnet:

$$\begin{array}{cl}
 & 00000000\ 00000000\ 00000000\ 10010011 \\
+_2 & 00000000\ 00000000\ 00000000\ 10011111 \\
\text{Ü} & 00000000\ 00000000\ 00000001\ 00111110 \\
\hline
 & 00000000\ 00000000\ 00000001\ 00110010
\end{array}$$

Zeile "Ü" stellt die Überträge der bitweisen Addition dar. Überträge, die über das Bit 31 hinaus führen würden, werden ignoriert.

Man sieht sofort, dass $+_2$ für nicht-negative Zahlen in Dualdarstellung gerade die "echte" Addition mit Ergebnisrepräsentation in Dualdarstellung realisiert: In Dualdarstellung ist im obigen Beispiel $x = 147$, $y = 159$ und $x + y = 306$, und $bits(306) =$ 00000000 00000000 00000001 00110010.

**Subtraktion:** Subtraktion $x - y$ wird als Addition $x + (-y)$ realisiert: In der ALU werden dazu die Bitrepräsentationen von $x$ und $y$ gemäß

$$bits(x) +_2 K_2(bits(y))$$

verknüpft.

## Aufgabe 1: Komli(e)ment, sehr verehrte Damen und Herren! (70%)

Implementieren Sie die interne Addition und Subtraktion mittels Zweierkomplement in Java. Verwenden Sie hierzu den Programmrahmen, der zur Aufgabe ins Netz gestellt wurde und für die Implementierung die Funktionen

```
public static int plus2(int n1, int n2);
public static int chgsign(int n);
public static int minus(int n1, int n2);
```

vorgibt.

Testen Sie Ihre Implementierung mit sinnvollen Additions- und Subtraktionsausführungen, die Sie jeweils mit den vordefinierten + und - Operationen von Java vergleichen. Programmieren Sie Ihre Testfälle nach dem Schema, welches im Programmrahmen in der `main()`-Methode anhand von 2 Beispielen vorgegeben wurde.

Begründen Sie, warum Ihre Testfälle ausreichend sind – beachten Sie beim Testfallentwurf auch auch Aufgabe 2!

**Hinweis:** In Ihrer Implementierung der Funktionen darf keine einzige Verwendung der Java-Operatoren + oder - vorkommen!

## Aufgabe 2: Out of Range? Overflow ? (30%)

Die interne Realisierung von Addition und Subtraktion mittels $+_2$ und $K_2$ ist natürlich nur auf beschränkten Wertebereichen eine korrekte Verfeinerung der Addition auf $\mathbb{Z}$.

1. Welches ist der Wert der größten darstellbaren Zahl $z_{max}$ aus `int` und wie lautet die Bitrepräsentation $bits(z_{max})$ ?
2. Welches ist der Wert der kleinsten darstellbaren Zahl $z_{min}$ aus `int` und wie lautet die Bitrepräsentation $bits(z_{min})$ ?
3. Welchen Wert aus $\mathbb{Z}$ repräsentiert

   11111111 11111111 11111111 11111111

4. Auf welcher Teilmenge $I \subseteq$ `int` $\times$ `int` ist $+_2$ eine korrekte Verfeinerung der Addition auf $\mathbb{Z}$, so dass also gilt

$$\forall (n_1, n_2) \in I : \alpha(n_1) + \alpha(n_2) = \alpha(n_1 +_2 n_2)$$

5. Auf welcher Teilmenge $J \subseteq$ `int` $\times$ `int` ist $n_1 +_2 K_2(n_2)$ eine korrekte Verfeinerung der Subtraktion auf $\mathbb{Z}$, so dass also gilt

$$\forall (n_1, n_2) \in J : \alpha(n_1) - \alpha(n_2) = \alpha(n_1 +_2 K_2(n_2))$$