LU01i - Binär codierte Ganzahlen

Siehe http://www.ulthryvasse.de/index.html

Auch wenn es oft geschrieben wird: Es gibt keine binären Zahlen; Genauso wenig wie es dezimale Zahlen gibt. Korrekt ausgedrückt handelt es sich um die binäre Darstellung oder binäre Codierung von Zahlen.

Und damit wünsche ich viel Erfolg und Spass mit den binären Zahlen in diesem Kapitel. 🤣

Binär codierte Ganzzahlen

Diese Codierung wird unter anderem für Variablen vom Typ `int`, `short` und `long` verwendet. Die Zahlen werden einfach im binären Zahlensystem (siehe auch zahlensysteme) gespeichert und verarbeitet. Auf das Speichern und Verarbeiten von Brüchen wird bewusst verzichtet. Die binär codierten Ganzzahlen können nicht bloss positive Zahlen darstellen, sondern haben den Wertebereich der "Ganzen Zahlen". Negative Zahlen werden dabei durch das Vorzeichen-Bit identifiziert.

- Wenn das vorderste Bit `0` ist, handelt es sich um eine positive Zahl.
- Ist es `1`, so wird die Zahl als negativ interpretiert.

Zahlenkreis statt Zahlenstrahl

In der Mathematik stellen wir Zahlen häufig auf einem unendlichen Zahlenstrahl dar, der in beide Richtungen fortläuft. In der Informatik hingegen arbeiten wir mit einer festgelegten Anzahl von Bits zur Darstellung von Ganzzahlen. Dadurch ist der Wertebereich endlich und es entsteht kein linearer Zahlenstrahl, sondern ein Zahlenkreis.

Wenn beispielsweise ein 8-Bit-Datentyp wie `byte` verwendet wird, liegt der Wertebereich bei -128 bis 127. Wird der höchste Wert 127 um 1 erhöht, erfolgt ein Überlauf (Wrap around) und es wird wieder beim kleinsten Wert -128 angesetzt. Somit folgt auf den maximalen positiven Wert unmittelbar der minimal negative Wert – als ob die Zahlenwerte auf einem Kreis angeordnet wären.

Dieses kreisförmige Modell erklärt, warum bei binär codierten Ganzzahlen herkömmliche Konzepte wie ein unendlicher Zahlenstrahl nicht mehr sinnvoll sind. Alle arithmetischen Operationen müssen den Wrap-around-Effekt berücksichtigen, da das Rechnen im fest definierten, kreisförmigen Wertebereich erfolgt.

Positive Zahlen

Im Speicher sind alle Zahlen als binäre Codes abgelegt. Um diesen binären Code als Dezimalzahl zu lesen, übertragen Sie den Wert einfach ins 10er System.

$0011\ 0010_2$ = 50

Wie das geht, erfahren Sie im Kapitel umrechnentheorie.

Negative Zahlen als Zweierkomplement

Siehe auch http://de.wikipedia.org/wiki/Zweierkomplement

In der Informatik wird für negative binäre Ganzzahlen das Zweierkomplement verwendet. Mit dieser Technik können Rechenoperationen ohne spezielle Regeln für positive und negative Zahlen eingesetzt werden. Mehr dazu erfahren Sie im Kapitel binaermath.

Für positive Zahlen: Im Zweierkomplement ist das höchstwertige Bit das Vorzeichenbit. Ist es 0, handelt es sich um eine positive Zahl, deren Wert sich direkt aus der normalen Binärdarstellung ergibt.

Für negative Zahlen gehen Sie wie folgt vor:

1. Addieren Sie 1 zur Zahl: $-13_{10} + 1_{10} = -12_{10}$
2. Entfernen Sie das Vorzeichen der Zahl: $-12_{10} \Rightarrow 12_{10}$
3. Übertragen Sie die Zahl ins Binärsystem: $12_{10} = 0000\ 1100_2$
4. Invertieren Sie alle Bits: $0000\ 1100_2 \Rightarrow \mathbf{1111\ 0011_2}$

Negative Zahlen haben immer Bit '1' an erster Stelle.

Um eine negative binäre Ganzzahl ins Dezimalsystem zu übertragen, kehren Sie einfach das oben stehende Vorgehen um:

1. Invertieren Sie alle Bits: $1111\ 0011_2 \Rightarrow 0000\ 1100_2$
2. Übertragen Sie die Zahl ins Dezimalsystem: $0000\ 1100_2 = 12_{10}$
3. Fügen Sie das Vorzeichen hinzu: $12_{10} \Rightarrow -12_{10}$
4. Subtrahieren Sie 1 von der Zahl: $-12_{10} - 1_{10} = \mathbf{-13_{10}}$

Umwandlung ins Zweierkomplement von Hand

Trick zur schnelleren Umwandlung (einer negativen in eine positive Binärzahl oder umgekehrt) von Hand: Von rechts angefangen, alle Nullen und die erste Eins abschreiben und alle nachfolgenden Stellen invertieren.

Dieser Programmablaufplan zeigt, wie Sie das Zweierkomplement einer binären Zahl ohne zu rechnen erhalten. Dies stellt eine Alternative zum oben beschriebenen Vorgehen dar:

Wrap around

Mit Wrap around bezeichnen wir das Verhalten, wenn aus einer positiven Zahl plötzlich eine negative Zahl wird. Am Besten lässt sich das Verhalten an einem Beispiel erläutern.

Beispiel

Für unser Java-Beispiel verwenden wir den Datentyp `byte` mit 8 Bit Speicherplatz. In einer Endlosschleife (`while (true)`) addieren wir immer 1 zur Zahl.

```
byte kleineZahl = 124;
while (true) {
    kleineZahl = kleineZahl + 1;
    System.out.println(kleineZahl);
}
```

Ausgabe

```
125
126
127
-128
-127
...
```

Wieso ergibt 127 + 1 = -128? Ein Blick in die binäre Codierung der Zahl lüftet das Geheimnis.

Addiert man zur Zahl 127_{10} (binär $0111\ 1111_2$) Eins dazu, so erhält man $1000\ 0000_2$. Wie Sie gelernt haben, bedeutet die binäre Ziffer 1 an der ersten Stelle, dass es sich um eine negative Zahl handelt. Wir erhalten also -128_{10}.

Im Gegensatz zur Mathematik haben wir also keinen Zahlenstrahl, sondern einen Zahlenkreis.

---

m114-A1G, m114-A1F