Binäre Darstellung ganzer Zahlen

Binärdarstellung natürlicher Zahlen

Ganze Zahlen im Einerkomplement

Ganze Zahlen im Zweierkomplement

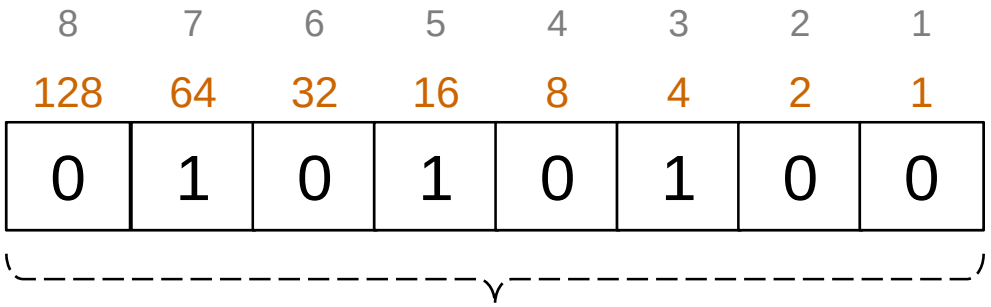
universitätbonn

Elementare Datentypen in Java

`byte`	8-bit-Zahl in Zweierkomplement-Darstellung
`short`	16-bit-Zahl in Zweierkomplement-Darstellung
`int`	32-bit-Zahl in Zweierkomplement-Darstellung
`long`	64-bit-Zahl in Zweierkomplement-Darstellung
`float`	32-bit IEEE 754-1985 Gleitkommazahl
`double`	64-bit IEEE 754-1985 Gleitkommazahl
`char`	16-bit Unicode 2.0 Zeichen
`boolean`	Wahrheitswert, `true` oder `false`

- N-bit-Zahl in Zweierkomplement??
- N-bit IEEE 754-1985 Gleitkommazahl???
- Unicode???

Exkurs: Binäre Darstellung natürlicher Zahlen



- Darstellbar mit N Bit: 2^N Zahlen
  - byte: 8 Bit → 2^8 = 256 Zahlen
  - short: 16 Bit → 2^{16} = 65.536 Zahlen
  - int: 32 Bit → 2^{32} = 4.294.967.296 Zahlen
  - long: 64 Bit → 2^{64} = 18.446.744.073.709.551.616 Zahlen
- Alles klar, aber was ist mit negativen Zahlen?

Exkurs: Binäre Darstellung ganzer Zahlen

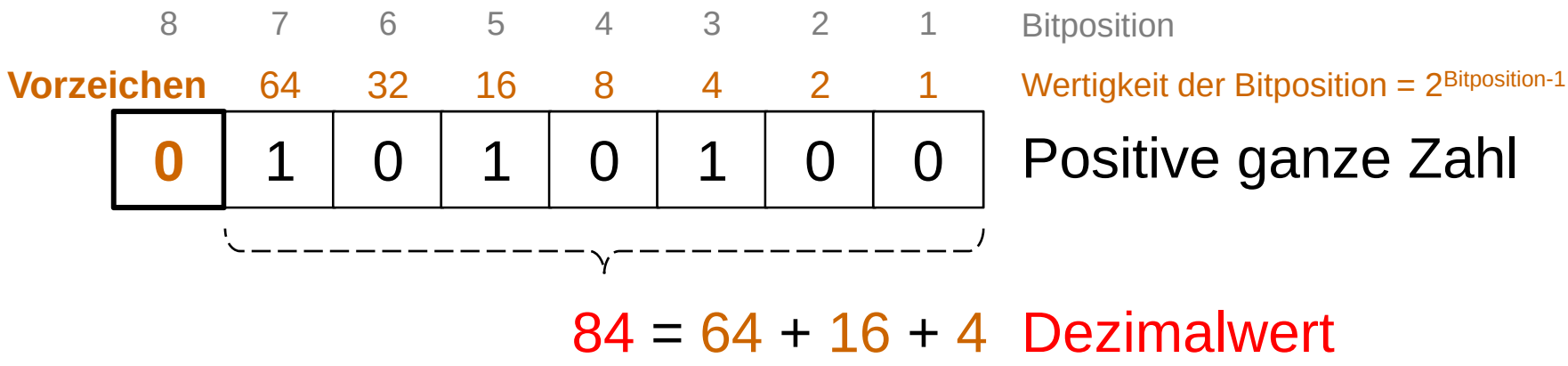
- Erste Idee: „Einerkomplement“
  - Ein Bit für das Vorzeichen, der Rest wie vorhin

	64	32	16	8	4	2	1	Wertigkeit
0	1	0	1	0	1	0	0	Positive Binärzahl: 84_{10}
1	1	0	1	0	1	0	0	Negative Binärzahl: -84_{10}
	64	32	16	8	4	2	1	Wertigkeit

- Problem
  - Zwei Nullen: **0**000 0000 und **1**000 0000 !!!
- Lösung
  - „Zweierkomplementdarstellung“ ganzer Zahlen

Exkurs: „Zweierkomplementdarstellung“: 1. Positive Zahlen und Null

- Werden mit einer führenden **0** (Vorzeichenbit) versehen



- Somit darstellbar mit N Bit
  - 2^{N-1} Zahlen: 0 bis $2^{N-1} - 1$ wegen der 0
  - Beispiel 8 Bit: 0 bis $2^{8-1} - 1 = 127$

Exkurs: „Zweierkomplementdarstellung“: 2. Negative Zahlen

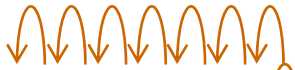
- Werden mit führender **1** (Vorzeichenbit) dargestellt
- Konstruktion der negativen Darstellung
  - Ziffern der entsprechenden positiven Zahl werden negiert.
  - Zum Ergebnis wird 1 addiert
- Beispiel: Darstellung der -84 in 8-Bit-Zweierkomplement
  - Positive Zahl: 84_{10} → $0101\ 0100_2$
  - Negiert: $1010\ 1011_2$
  - Plus 1: $1010\ 1100_2$
- Somit darstellbar
  - -1 bis -2^{N-1} (z.B. mit 8 Bit: -1 bis -128)

Rechnen im Zweierkomplement

- Addition im Zweierkomplement: „Normales“ Rechnen mit Übertrag
  - $+84_{10} = 0101\ 0100_2$ (Erster Summand)
  - $-84_{10} = 1010\ 1100_2$ (Zeiter Summand)
  - $0_{10} = 0000\ 0000_2$ (Summe)
- Subtraktion ist Addition der negierten Zahl
  - Beispiel: 90 – 84 = 90 + (–84) = 6
  - $+90_{10} = 0101\ 1010_2$ (Erster Summand)
  - $-84_{10} = 1010\ 1100_2$ (Zeiter Summand)
  - $6_{10} = 0000\ 0110_2$ (Summe)

Arithmetik modulo dem Darstellungsbereich

- Es werden keine Überläufe über den Darstellungs-bereich hinaus erzeugt
  - Nicht darstellbare Bits werden einfach abgeschnitten!
  - Dadurch wird der Darstellungsbereich zu einem „Ring“ geschlossen
    - ⇨ Zählt man über das Ende des positiven Bereichs hinaus, kommt man zur kleinsten negativen Zahl und umgekehrt
- Beispiel



  - $+127_{10} = 0111\ 1111_2$ (Maximale 8-Bit-Zahl)
  - $+1_{10} = 0000\ 0001_2$ (1 hinzuaddieren)
  - $-128_{10} = 1000\ 0000_2$ (Summe ist die minimale 8-Bit-Zahl)
  - Alles immer noch bezogen auf Zweierkomplement!

Weitere Details

- Sehr gute Übersicht bei Wikipedia
  - http://de.wikipedia.org/wiki/Zweierkomplement

Gleitkommazahlen

Bei Wikipedia nachzuschlagen

(sehr gute, detaillierte Erklärung: http://de.wikipedia.org/wiki/Gleitkommazahlen)

Unicode

Bei Wikipedia nachzuschlagen

(sehr gute, detaillierte Erklärung!) http://de.wikipedia.org/wiki/Unicode)