Negative Zahlen: Vorzeichen-Betrag-Darstellung

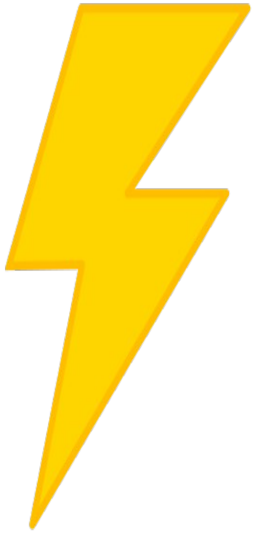
Erste Idee: Vorderstes Bit wird „geopfert“ und als Vorzeichen-Bit interpretiert

–	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	0	1	0	0	1	1

Im Beispiel: $10010011_2 = -19_{10}$

„Leider“ funktionieren die bekannten Rechenregeln damit nicht:

$-18_{10} = -19_{10} + 1_{10} = 10010011_2 + 00000001_2 = 10010100_2 = -20_{10}$



Negative Zahlen: Einer-Komplement

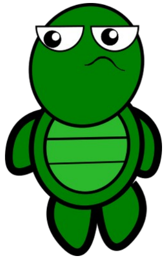
Idee: Von allen Bits wird das **Komplement** gebildet. Dabei werden aus 1en 0en und andersrum.

Beispiel: $00001101_2 = 13_{10}$
111100102 = -13_{10}

Problem: $00000000_2 = 11111111_2$ (Warum?)

Mit dem **Einerkomplement** funktionieren die bekannten Rechenregeln wie bisher, wenn man dabei nicht „über die Null“ kommt (siehe Aufgabe im Wiki).

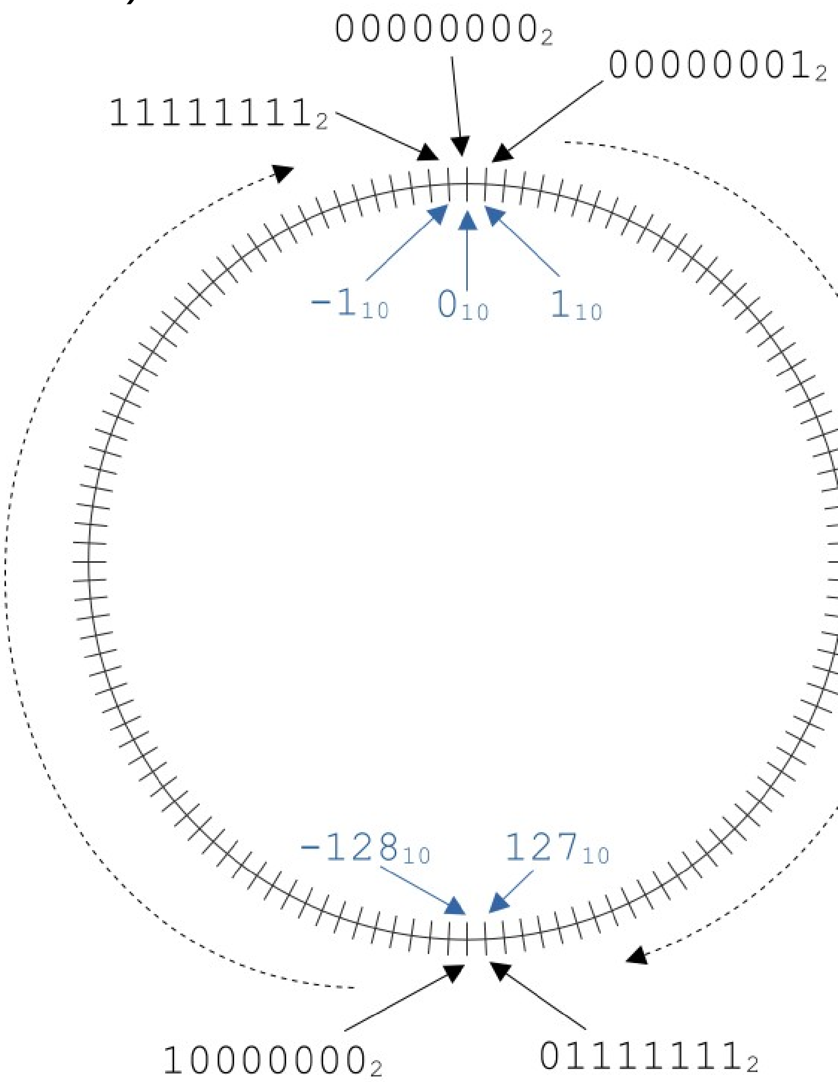
Fazit: Besser als das Vorzeichenbit, aber immer noch nicht cool...



Idee: Die Wertigkeit des höchsten Bits (ganz links) wird negiert.

Beispiele:

01111111_2	=	127_{10}
10000000_2	=	-128_{10}
10000001_2	=	-127_{10}
11111111_2	=	-1_{10}



Mit dem Zweierkomplement funktionieren die bekannten Rechenregeln!