Weitere Nachteile:

- erfordert separates Subtrahierwerk
- erfordert zusätzliche Logik, um zu entscheiden, welches Vorzeichen das Ergebnis der Operation hat

2. Die Komplement - Darstellung

Das höchstgewichtete Bit wird weiterhin (aber nicht exklusiv) für die Angabe des Vorzeichens genutzt, d.h. das Vorzeichenbit ist Teil des Summanden und wird in eine arithmetische Operation mit eingeschlossen

- → Subtraktion wird auf die Addition zurückgeführt
- → Keine Notwendigkeit für ein zusätzliches Subtrahierwerk
- → Keine zusätzliche Logik zur Bestimmung des Vorzeichens

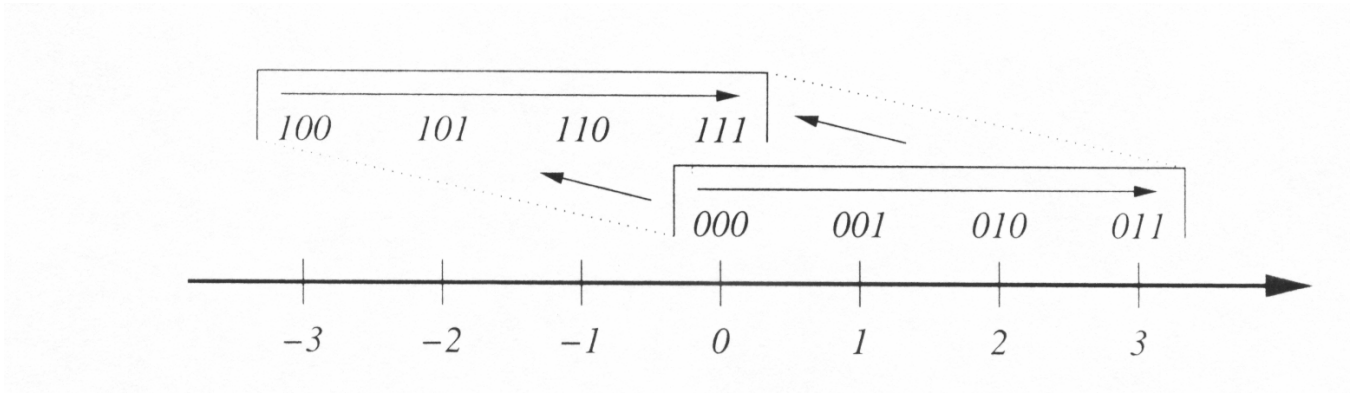
2a. Einer - Komplement

Sei N der Betrag (Größe) einer negativen ganzen Zahl Z (d.h. d_n=1). Dann gilt:
$-N = N - (2^n - 1)$

Das Einer - Komplement - N einer positiven binären Zahl N aus $[0 , 2^{n-1}-1]$ erreicht man durch bitweises Invertieren von N ---> -N aus $[-0, -2^{n-1}-1]$

Computerarithmetik (7)

Beispiel für n=3:



Subtraktion:= Addition + end-around-carry, d.h. zu der Summe wird das Übertragsbit aufaddiert.

Vorteil: zusätzliches Subtrahierwerk überflüssig

Nachteile:
- keine eindeutige Darstellung der Null
- kein echtes Komplement, da -x + x $\neq$ 0 sondern = - 0, also 111....1

Computerarithmetik (7b)

2b. Zweier - Komplement

Sei N der Betrag (Größe) einer negativen ganzen Zahl Z (d.h. d_n=1). Dann gilt:

$-N = N - 2^n$

→ -N = Einer - Komplement + 1

→ -N = (bitweises Invertieren von N) + 1

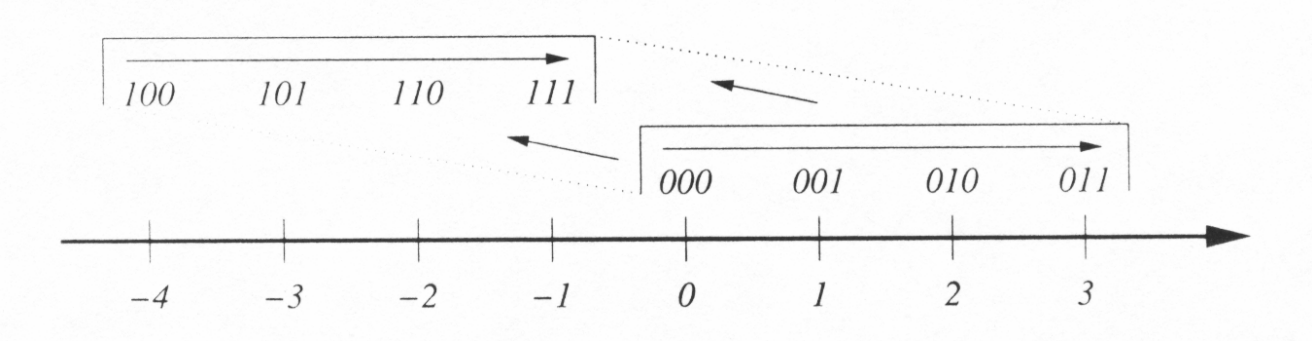
→ Es gibt eine eindeutige Darstellung der Null (0.....0)

→ -N aus $[-1, -2^{n-1}]$

→ Der Wertebereich des Zweier - Komplements ist $[-2^{n-1}, 2^{n-1}-1]$

→ Das Zweier - Komplement ist ein echtes Komplement: $N+(-N) = 2^n = (0.....0)$

Beispiel für n=3:



Computerarithmetik (8a)

Einfache Additions (Subtraktions-) Regeln

x+y: Addition der entsprechenden 2er - Komplemente ergibt korrekte Summe im 2er - Komplement, solange der Wertebereich nicht überschritten wird.

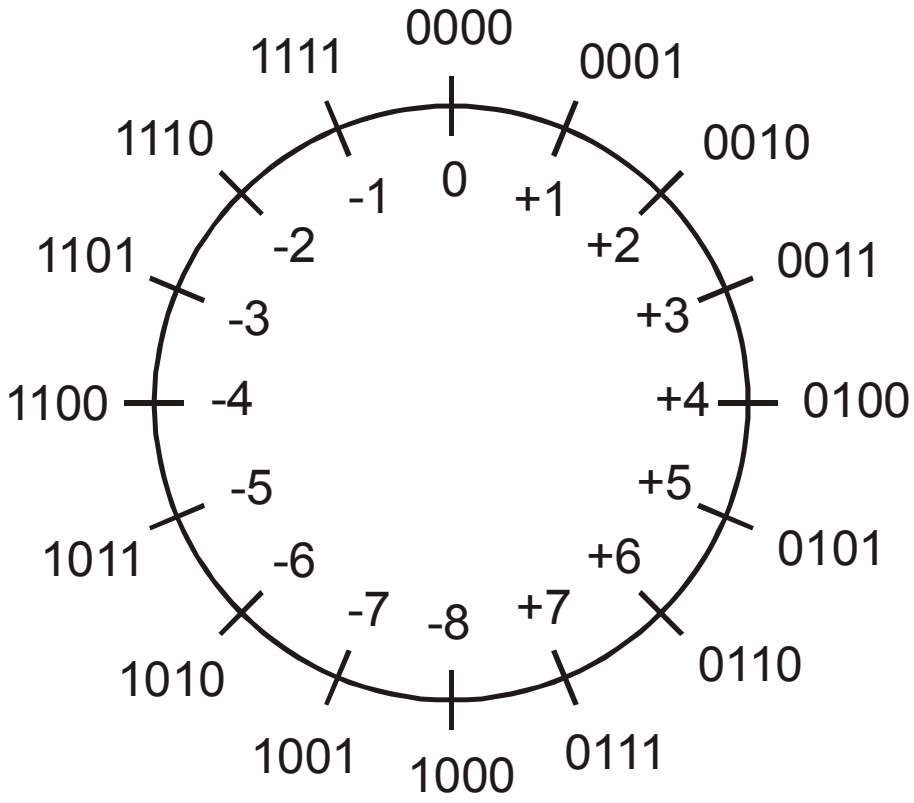
x-y = x+(-y): Bilde das 2er - Komplement von y und führe Addition wie oben aus.

Konsequenz:

Die logische Einfachheit und die daraus resultierende Geschwindigkeit (arithmetische Operation erfolgt immer in einem Schritt) führt dazu, dass das Zweier - Komplement in den ALU´s moderner Rechner eingesetzt wird.

Computerarithmetik (8)

Visualisierung des Zweier - Komplements sowie der Addition



Computerarithmetik (9)

Overflow (Summe liegt außerhalb des Wertebereiches):

Wichtig: Erkennung des Overflows

- Bei Integer-Addition dient das carry-out-Bit als Overflow-Indikator.
- Bei Addition ganzer Zahlen (signed numbers) gilt dies nicht
- Addition von Summanden mit unterschiedlichem Vorzeichen ergibt **nie** einen Overflow (Absoluter Wert ihrer Summe ist immer kleiner als der absolute Wert von einem der beiden Summanden)

Folgerung: Overflow nur möglich, wenn beide Summanden das gleiche Vorzeichen haben

Prüfung auf Overflow:

$$O = a_{n-1}\, b_{n-1}\, \overline{s_{n-1}} + \overline{a_{n-1}}\, \overline{b_{n-1}}\, s_{n-1}$$

(Die Faktoren repräsentieren die Vorzeichenbits der Summanden a und b sowie der Summe s)

Gilt O = 1 $\longrightarrow$ Es existiert ein Overflow!

Computerarithmetik (27)

Kriterien für die Qualität der Zahlendarstellung:

- Größe des darstellbaren Zahlenbereichs (**range**)
- Genauigkeit (**precision**) der Zahlendarstellung

Diese beiden Kriterien sind prinzipiell **unabhängig** voneinander.

Wissenschaftliche Notation: $d = a \times r^E$

a Mantisse (Argument), r Radix (Basis), E Exponent (Charakteristik)

Parameter für mögliche Darstellungen von Floating point - Zahlen :

- Anzahl der insgesamt verfügbaren Bits (Wortlänge bzw. Worte)
- Anzahl der verfügbaren Bits jeweils für Mantisse bzw. Exponent (Trade-off!)
- Darstellung von Mantisse und Exponent
- Lokalisierung (Mantisse vor Exponent oder umgekehrt)

Mantissendarstellung in normierter Form:

$d = (-1)^s \times a \times 2^E$ mit s als Vorzeichenbit und $1 \leq a < 2$

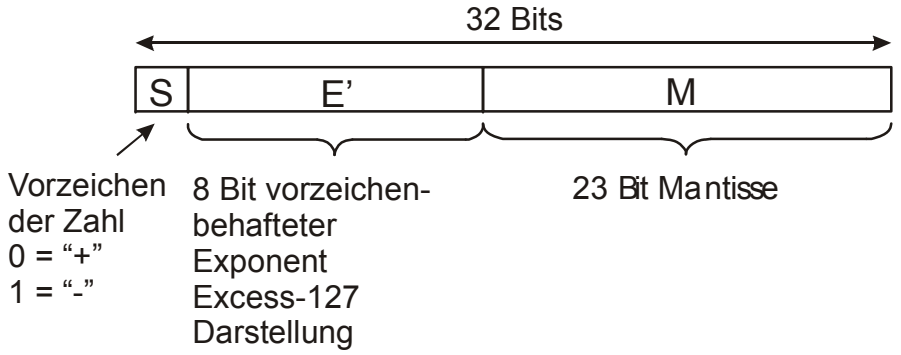
Exponentendarstellung mit Bias:

$d = (-1)^s \times a \times 2^{E'}$ mit $E' := E + 127$

Computerarithmetik (27a)

Darstellung im IEEE Standard 754:

Einfache Genauigkeit:

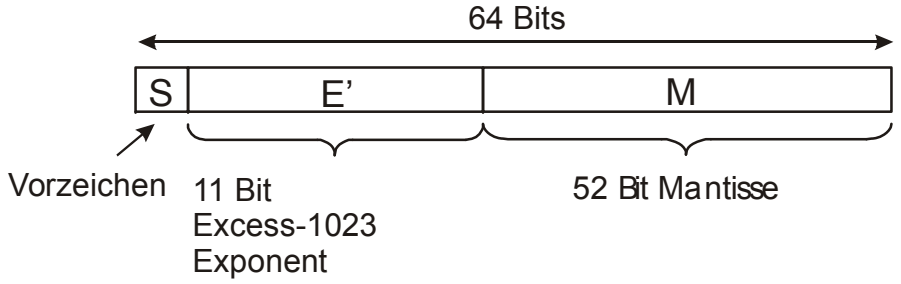


Darstellung entspricht: $\pm 1{,}M \cdot 2^{E'-127}$

Beispiel mit
Einfacher Genauigkeit:

Darstellung entspricht: $1{,}001010...0 \cdot 2^{-87}$

Doppelte Genauigkeit:



Darstellung entspricht: $\pm 1{,}M \cdot 2^{E'-1023}$

Computerarithmetik (28)

Verallgemeinerter Additions/Subtraktions - Algorithmus:

- Rechtsshift auf der Mantisse des kleineren Operanden zur Angleichung der Exponenten
  ----> Exponent der Summe/Differenz := Exponent des größten Operanden
- Addition/Subtraktion der Mantissen und Bestimmung des Vorzeichens
- Wenn nötig, Normalisierung des Ergebnisses

Verallgemeinerter Multiplikations/Divisions- Algorithmus:

- Multipliziere/Dividiere die Mantissen und bestimme das Vorzeichen
- Wenn nötig, normalisiere das Ergebnis
- Addiere/Subtrahiere die Exponenten und subtrahiere/addiere 127_{10}

Rundung: Kappung überzähliger Bits durch

- Abspalten (chopping)
- von Neuman - runden
- runden

Computerarithmetik (28a)

(vorläufige) Zusammenfassung:

- Computerarithmetik ist endlich und kann folglich **nicht** übereinstimmen mit der natürlichen Arithmetik

- Selbst der IEEE 754 - Standard für die Fließkomma - Darstellung, wie jede andere auch, ist fast immer eine **Approximation** der realen Zahlen.

- Rechnersysteme müssen dafür sorgen, den daraus resultierenden Unterschied zwischen Computerarithmetik und Arithmetik in der realen Welt möglichst zu minimieren.

- Informatiker sollten sich dieser Zusammenhänge bewusst sein.