2.1.2 Gleitkommazahlen

Überblick: Gleitkommazahlen

- Gleitkommadarstellung
- Arithmetische Operationen auf Gleitkommazahlen mit fester Anzahl von Mantissen- und Exponentenbits
- Insbesondere Rundungsproblematik:
  - Ergebnisse von arithmetischen Operationen sind evtl. nicht (exakt) als Gleitkommazahl darstellbar (auch wenn sie im Bereich zwischen größter und kleinster darstellbarer Zahl liegen).
  - Man will zumindest den Fehler möglichst gering halten: Rundung
  - Verschiedene Rundungsmodi

Gleitkommadarstellung

Idee: Repräsentiere Zahl durch Vorzeichen, Exponent und Mantisse,
Position des Kommas liegt (im Gegensatz zu Festkommazahlen)
also nicht fest!
Abdeckung eines größeren Zahlenbereichs bei gegebener Stellenanzahl

Gleitkommadarstellung einfacher Genauigkeit: $(-1)^{S} \cdot M \cdot 2^{E}$

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse M

Gleitkommadarstellung doppelter Genauigkeit: $(-1)^{S} \cdot M \cdot 2^{E}$

63	62 61 . . . 53 52	51 50 . . . 3 2 1 0
S	Exponent E	Mantisse M

Es bleibt noch festzulegen, wie die Mantissenbits bzw.
Exponentenbits als Zahlen M bzw. E interpretiert werden sollen.

Normalisierte Gleitkommadarstellungen

Beobachtung

Gleitkommadarstellung einer Zahl ist nicht eindeutig !

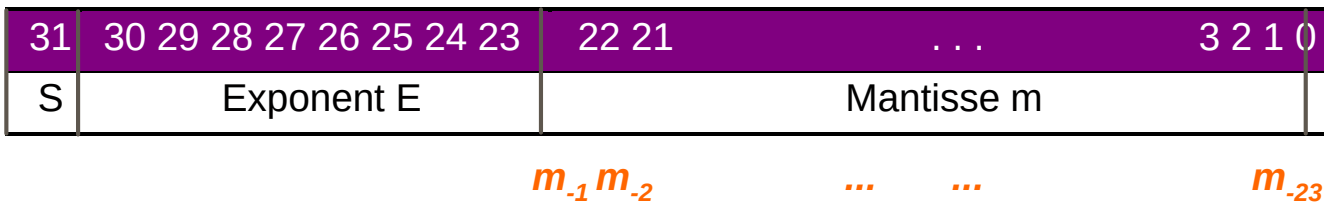
$0.111 \cdot 2^3 = 0.0111 \cdot 2^4$

Definition

Eine Gleitkommazahl (S,M,E) heißt **normalisiert**,

wenn $1 \leq M < 2$ d.h. wenn M von der Form $1.m_{-1} \ldots m_{-k}$ ist.

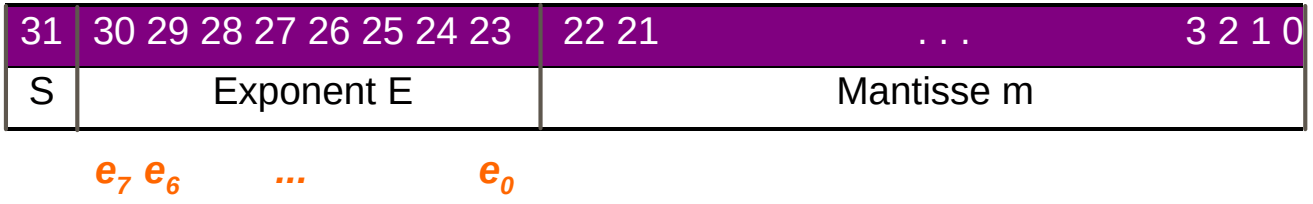
Die 1 vor dem Komma braucht nicht abgespeichert zu werden (→ „**hidden bit**“)



Für eine normalisierte Gleitkommazahl ergibt sich der Mantissenwert M als $M = 1 + \sum_{i=-1,\ldots,-k} m_i 2^i$.

Þ **Die Zahl 0 muss als Spezialfall behandelt werden !**

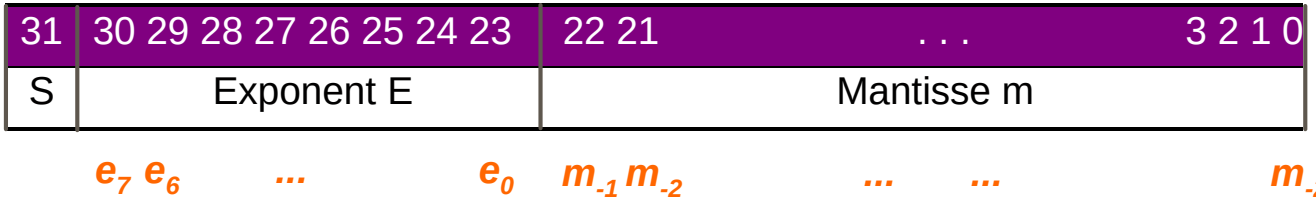
Gleitkommadarstellung - IEEE 754 Standard



- Gemäß IEEE 754-Standard werden die Exponentenbits als vorzeichenlose Zahl interpretiert. Diese Zahl wird als Charakteristik C bezeichnet.
- Um auch negative Exponenten darstellen zu können, wird von der Interpretation als vorzeichenlose Zahl eine Konstante, der sogenannte **Bias**, subtrahiert.
- Bei *n* Exponentenbits wird der Bias gewählt als $\mathbf{BIAS = 2^{n-1}-1}$, also bei einfacher Genauigkeit BIAS = 127, bei doppelter Genauigkeit BIAS = 1023.
- Bei *n* Exponentenbits ergibt sich also für E:

$$E = C\text{-}BIAS = \sum_{i=0,\ldots,n-1} e_i\, 2^i - BIAS$$

Gleitkommadarstellung - IEEE 754 Standard



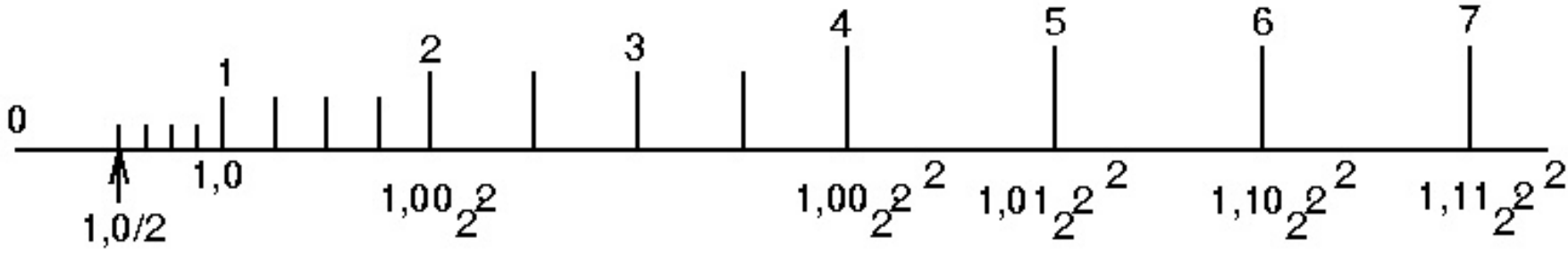
- Insgesamt ergibt sich also bei *k* Nachkommastellen der Mantisse und *n* Exponentenbits der folgende Wert für die Gleitkommazahl:

$$(-1)^S \cdot M \cdot 2^E = (-1)^S \cdot (1 + \sum_{i=-1}^{-k} m_i 2^i) \cdot 2^{\left(\sum_{i=0}^{n-1} e_i 2^i\right) - BIAS}$$

- Die Charakteristiken <1…1>=2^n-1 und <0…0>=0 sind für spezielle Zwecke reserviert (siehe später) und werden für normalisierte Gleitkommazahlen nicht verwendet.
- Damit ist die größte darstellbare Zahl $(2 - 2^{-k}) \cdot (2^{2^{n-1}-1})$
- Und die betragsmäßig kleinste darstellbare Zahl $2^{-2^{n-1}+2}$

Gleitkommadarstellung - IEEE 754 Standard

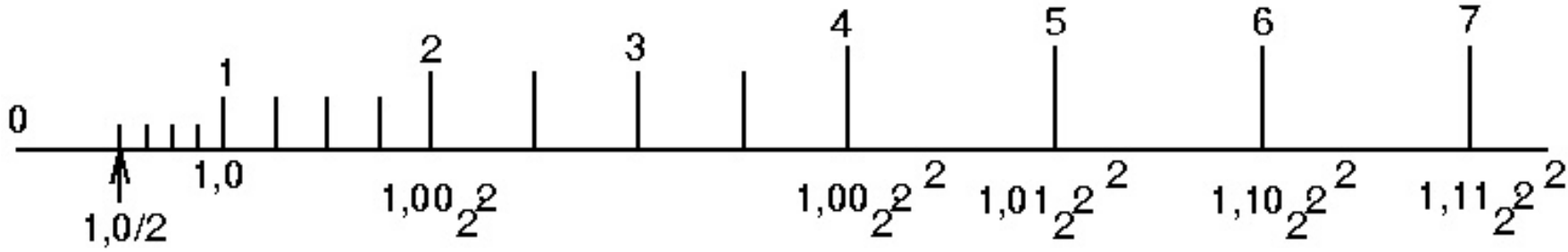
- Natürlich sind nicht alle reellen Zahlen zwischen $(2-2^{-k})\cdot(2^{2^{n-1}-1})$ und $2^{-2^{n-1}+2}$ darstellbar.
- Darstellbar sind nur Zahlen auf einem bestimmten „Raster“. Das „Raster“ wird eng bei kleinen Zahlen und weit bei großen Zahlen.
- Beispiel:
  Sei $k=2$ und zur Illustration der Exponent aus {-1, 0, 1, 2}.



- Problem:
  Ganz dicht bei 0 gibt es eine „Lücke“.
  Grund ist das „hidden bit“ 1.m_{-1} … m_{-k}.

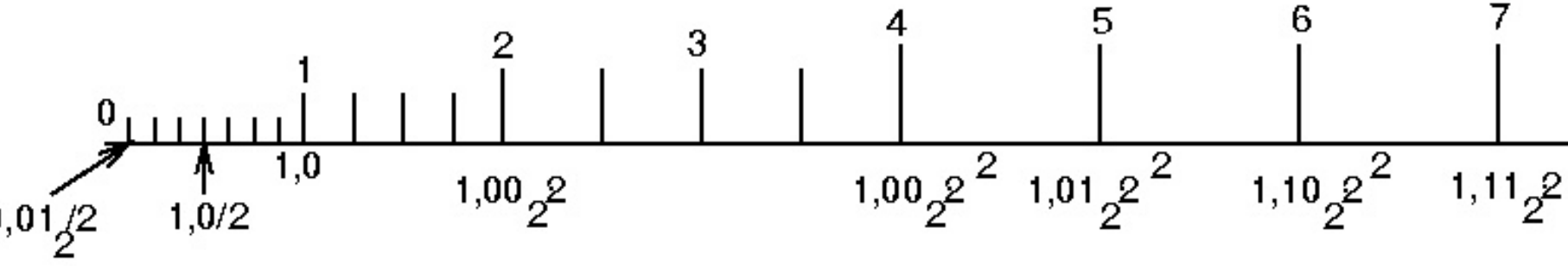
Gleitkommadarstellung

⌘ Beispiel:
Sei $k=2$ und zur Illustration der Exponent aus {-1, 0, 1, 2}.



⌘ Problem: „Lücke“ bei 0.

⌘ Abhilfe: Erlaube dicht bei 0 (wenn Exponent minimal, hier also -1) auch Zahlen ohne hidden bit, sog. denormalisierte Zahlen der Form $0.m_{-1} \ldots m_{-k}$.



Sonderfälle IEEE 754 Standard

- **Der Exponent 0 spielt beim IEEE 754-Standard eine Sonderrolle**:
  Sind alle Exponentenbits 0, so wird **ausnahmsweise** das „hidden bit“ der Mantissendarstellung weggelassen, so dass die Zahl

$$(S_{i=-1,\ldots,-k}\, m_i\, 2^i)\; 2^{-126}$$

  dargestellt wird.

- Auf diese Weise können „**denormalisierte Zahlen**“ dargestellt werden, die kleiner als die kleinste darstellbare normalisierte Zahl sind.

- Die **Null** wird folgendermaßen dargestellt: Sämtliche Mantissenbits und Exponentenbits sind 0.

- **Die Charakteristik 2^n-1 spielt ebenfalls eine Sonderrolle**:
  Sind alle Exponentenbits 1 und alle Mantissenbits 0, so wird der Wert ∞ dargestellt.

IEEE 754 Standard - Spezialfälle

Normalisierte Zahl	±	$0 < C < 255\ (2047)$	m beliebig
Denormalisierte Zahl	±	0	$m \neq 0$ beliebig
Null	±	0	0
Unendlich	±	255 (2047)	0
Not a Number	±	255 (2047)	$m \neq 0$ beliebig

Darstellbare normalisierte Gleitkommazahlen

	single precision	double precision
Vorzeichenstellen	1	1
Exponentenstellen	8	11
Mantissenstellen (ohne hidden Bit)	23	52
Bitstellen insgesamt	32	64
Bias	127	1023
Exponentenbereich	-126 bis 127	-1022 bis 1023
Darstellbare normalisierte Zahl mit kleinstem Absolutbetrag	2^{-126}	2^{-1022}
Darstellbare normalisierte Zahl mit größtem Absolutbetrag	$(1-2^{-24})\ 2^{128}$	$(1-2^{-53})\ 2^{1024}$
Darstellbare denormalisierte Zahl mit kleinstem Absolutbetrag	2^{-149}	2^{-1074}
Darstellbare denormalisierte Zahl mit größtem Absolutbetrag	$(1-2^{-23})\ 2^{-126}$	$(1-2^{-52})\ 2^{-1022}$

IEEE 754 Standard - Eigenschaften

- Eindeutige Zahlendarstellung, falls auf normalisierte Darstellungen beschränkt
- Nicht alle Zahlen zwischen der kleinsten und größten darstellbaren Zahl sind darstellbar.
- Je näher bei der Null, desto dichter liegen die darstellbaren Zahlen.
- Arithmetische Operationen sind nicht abgeschlossen!
- Assoziativgesetz und Distributivgesetz gelten nicht, da bei Anwendung der Gesetze evtl. der darstellbare Zahlenbereich verlassen wird!

Multiplikation von Gleitkommazahlen

$$\left((-1)^{VZ_1} \cdot M_1 \cdot 2^{E_1}\right) \times \left((-1)^{VZ_2} \cdot M_2 \cdot 2^{E_2}\right) = (-1)^{VZ_1 \oplus VZ_2} \cdot M_1 \cdot M_2 \cdot 2^{E_1+E_2}$$

Rechenvorschrift

- Multipliziere die Vorzeichen
- Multipliziere die beiden Mantissen
- Addiere die beiden Charakteristiken und subtrahiere (einmal) den Bias-Wert
- Normalisierung und/oder Rundung (falls erforderlich)

Beispiel

$+(1.000)_2 \cdot 2^{1\text{-BIAS}} \times -(1.110)_2 \cdot 2^{2\text{-BIAS}}$

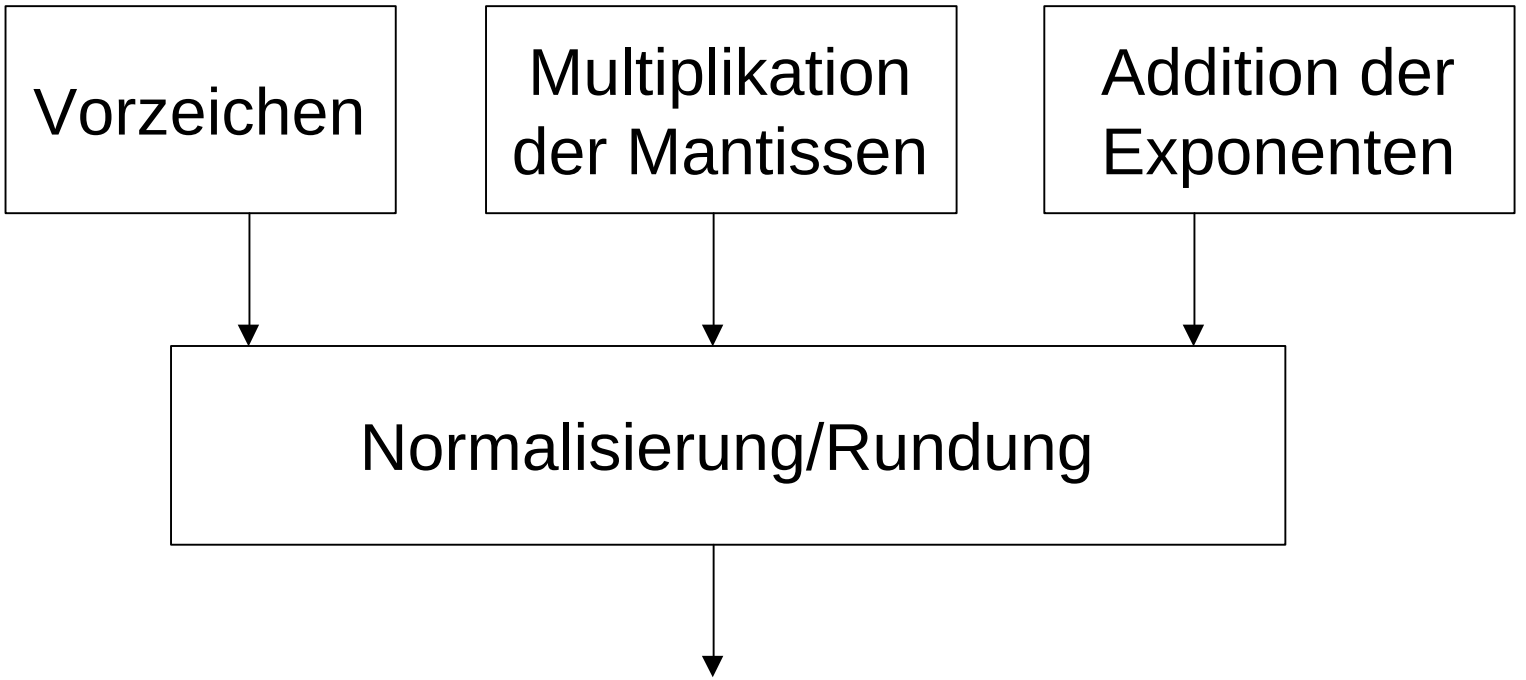
Multiplikation der Vorzeichen: $0 \oplus 1 = 1$

Multiplikation der Mantissen: $(1.000)_2 \times (1.110)_2 = (1.110)_2$

Addition der Exponenten: $(1\text{-BIAS})+(2\text{-BIAS})=(1+2\text{-BIAS})\text{-BIAS} = (3\text{-BIAS})\text{-BIAS}$

Resultat: $-(1.110)_2 \cdot 2^{(3\text{-BIAS})\text{-BIAS}}$

Multiplikation

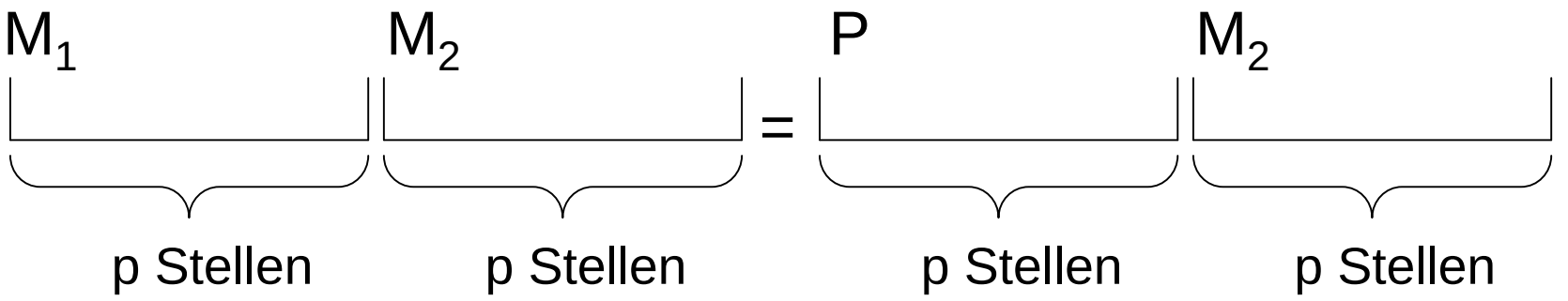


- Normalisierung und Rundung sind interessant!

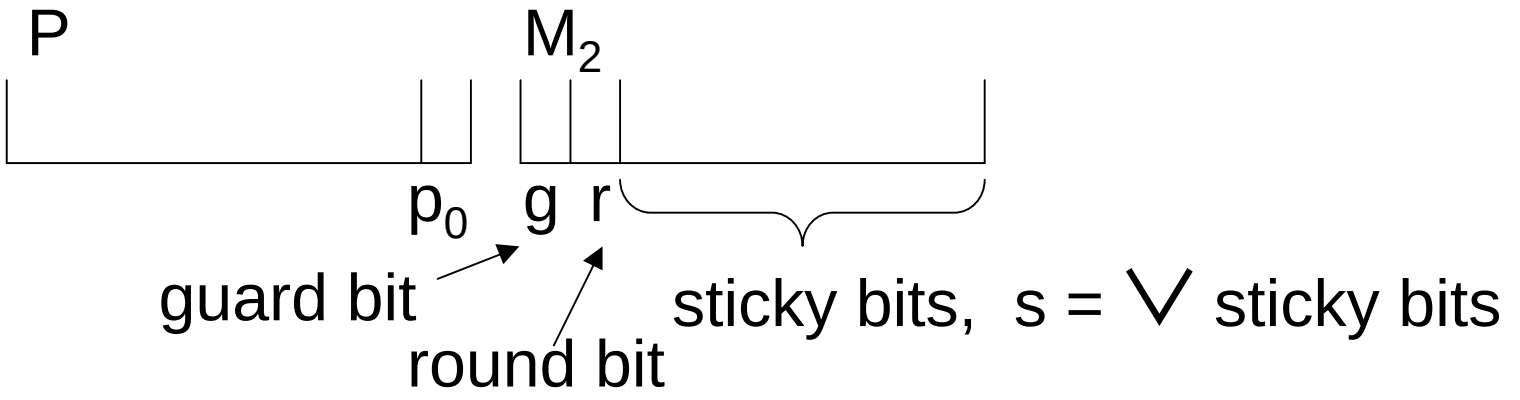
  Am Beispiel der Multiplikation einige Details

- Hier nur Multiplikation normalisierter Gleitkommazahlen.

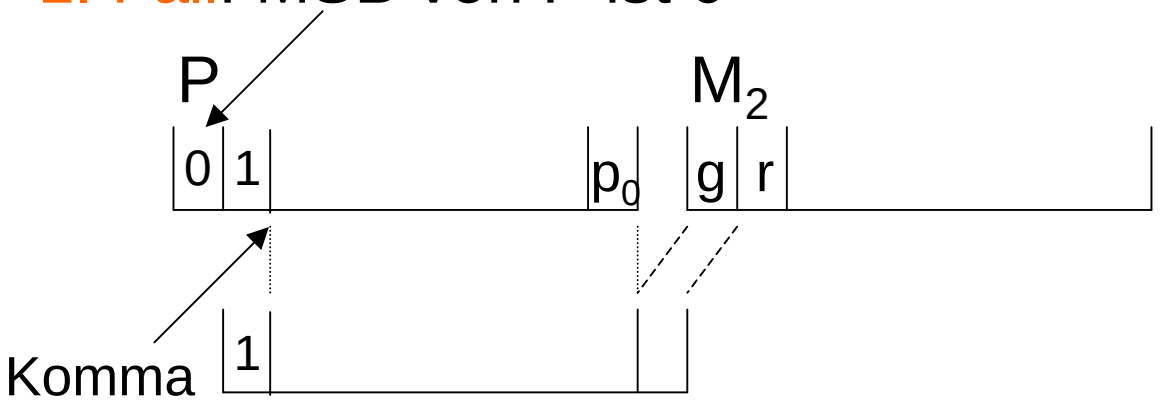
Normalisierung nach Multiplikation



Multiplikation der Mantissen M_1 und M_2 (mit p Stellen - p enthält auch das hidden bit!) liefere Ergebnis der Länge 2p in P und M_2;
M_2 enthält die LSB (Least Significant Bits) des Resultats

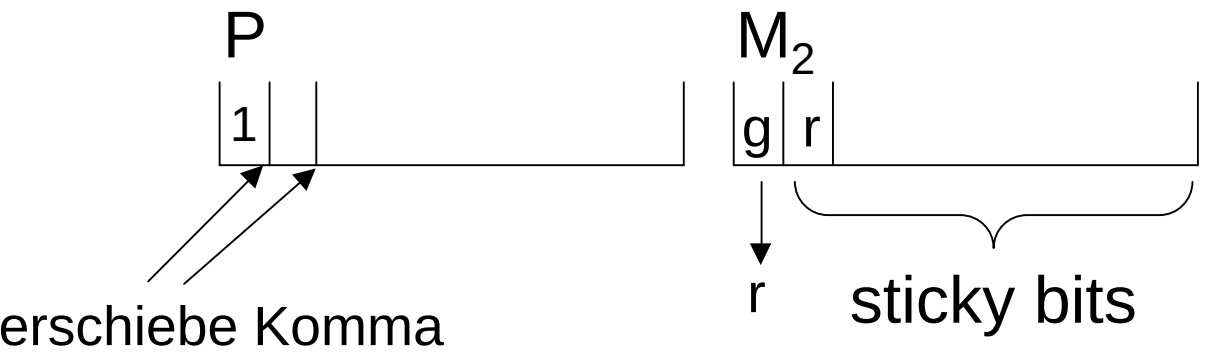


1. Fall: MSB von P ist 0



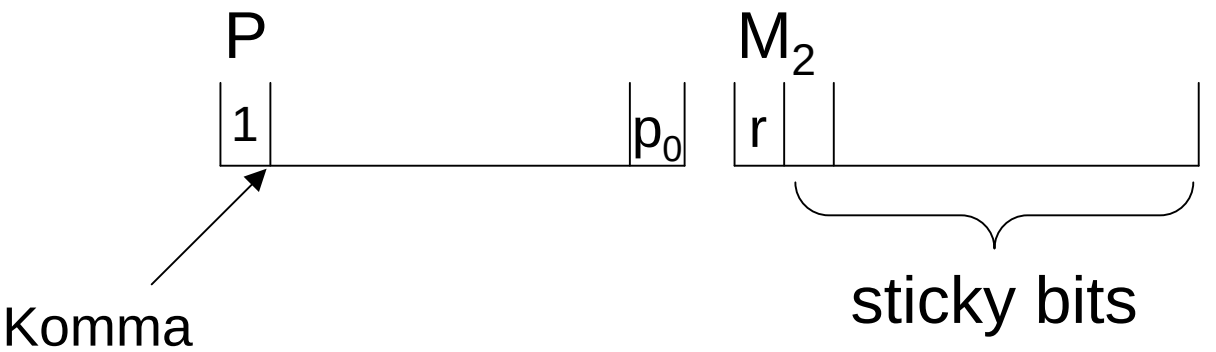
Shifte P um 1 Bit nach links und nehme g in P auf

2. Fall: MSB von P ist 1



Passe Position des Binärpunkts an durch
- Division durch 2 und
- Addition des Exponenten um 1
- Das alte guard-Bit wird Rundungsbit

Situation bei beiden Fällen:



Führe nun Rundung durch!

Runden abhängig von r und s

- r = 0 $\rightarrow$ P korrekt
- r = 1, s = 1 $\rightarrow$ P+1 ist Ergebnis
- r = 1, s = 0 $\rightarrow$ genau in der Mitte zwischen zwei darstellbaren Zahlen
  - $p_0 = 0 \rightarrow$ P korrekt
  - $p_0 = 1 \rightarrow$ P+1 ist Ergebnis

} “round to even”

Andere Rundungsarten

Rundungsmodus	Ergebnis ≥ 0	Ergebnis < 0
nearest (round to even)	+1 falls $(r \wedge s) \vee (r \wedge p_0)$	
0	--	--
$+\infty$	+1 falls $(r \vee s)$	--
$-\infty$	--	+1 falls $(r \vee s)$

Beispiel

- -5 $-1.010 \cdot 2^2$
  10 $1.010 \cdot 2^3$
- Multipliziere und normalisiere/runde nach allen Rundungsarten. Wie groß ist der Unterschied?

Overflow

- Entsteht, falls nach dem Runden die Zahl zu groß ist.

$$\begin{pmatrix} E_1 + 127 \\ E_2 + 127 \end{pmatrix} \xrightarrow{Add} (E_1 + E_2 + 2 \cdot 127 - 127)$$

Welche Werte kommen in Frage?

Beh.:Für die Charakteristik können Werte zwischen -125 und 381 herauskommen!

Berücksichtigt man Fall 2 bei der Multiplikation der Mantissen, so können Werte zwischen -125 und 382 herauskommen.

Ist die Charakteristik größer 254, so ergibt sich ein **Overflow**.

Underflow

- Ein “kleines” Ergebnis ist eventuell als denormalisierte Zahl darstellbar.
- Wenn beim Shiften nach rechts 0 entsteht, dann underflow.
- Beispiel:

$$1.000 \cdot 2^{-64} \cdot 1.000 \cdot 2^{-65} = 1.000 \cdot 2^{-129}$$
$$= 0.001 \cdot 2^{-126}$$

Je nach Größe der Mantisse kann es einen Underflow geben.

Weitere Spezialfälle

- ± 0 Ob ein Operand 0 ist, wird vorher überprüft
- $\pm \infty$ Vorsicht: $0 \cdot \infty = \text{NaN}$ (Definition)

- Die Details sind komplexer und umfassen weitere Spezialfälle
- Runden entfällt, falls ein Ergebnis mit doppelter Genauigkeit verlangt

Addition von Gleitkommazahlen

$$\left(M_1 \cdot 2^{E_1}\right) + \left(M_2 \cdot 2^{E_2}\right) = 2^{E_1}\left(M_1 + 2^{E_2 - E_1} \cdot M_2\right) \qquad (\text{O.E.}: E_1 \geq E_2)$$

Rechenvorschrift

- Angleichung des kleineren an den größeren Exponenten
- Addition der Mantissen
- Normalisierung (falls erforderlich)
- Rundung (falls erforderlich)

Beispiel

$+(1.000)_2 \cdot 2^{-1} + -(1.110)_2 \cdot 2^{-2} = +(1.000)_2 \cdot 2^{-1} + -(0.111)_2 \cdot 2^{-1}$

$= +(0.001)_2 \cdot 2^{-1}$

$= +(1.000)_2 \cdot 2^{-4}$

- Rundung nach der Normalisierung analog zur Multiplikation!

Division von Gleitkommazahlen

$$\left(M_1 \cdot 2^{E_1}\right) / \left(M_2 \cdot 2^{E_2}\right) = M_1 / M_2 \cdot 2^{E_1 - E_2}$$

Rechenvorschrift

- Division der Mantissen mit integer bzw. Festkomma-Dividierer
- Rundung analog Multiplikation

Bezug zu höheren Programmiersprachen

- Es gibt keinen Standard über die Handhabung von fp-Operationen in höheren Programmiersprachen
- Beispiel:

  p sei einfach genau,

  q sei doppelt genau.

```
p = q;          /* Zuweisung, Rundung */
if (p == q)     /* Bedingung nicht erfüllt?!? */
```

Bezug zu höheren Programmiersprachen (2)

- Optimierungen, die “offensichtlich” sind, können Ergebnisse verfälschen.
- Distributiv- und Assoziativgesetze gelten nicht nur wegen Bereichsüberschreitungen bei Anwendung der Gesetze nicht, sondern auch nicht wegen Rundungsfehlern!

- Alternative: Intervall-Arithmetik (Kulisch)