Binäre Division

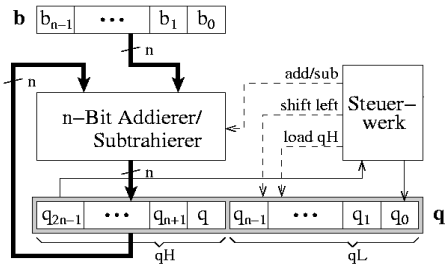
- Umkehrung der Multiplikation: Berechnung von q = a/b durch wiederholte bedingte Subtraktionen und Schiebeoperationen
- in jedem Schritt wird Divisor b testweise vom Dividenden a subtrahiert: $q_i = 1$, falls $a–b > 0$
  $q_i = 0$ und Korrektur durch $a = a + b$, falls $a–b < 0$
- Beispiel: $103_{10} / 9_{10} = 11_{10}$ mit Rest 4_{10}

```
  01100111   / 01001   = 1011
- 01001                   ↑
  00111                Quotient
- 01001
  11110
+ 01001   ←—— Korrektur
  001111
- 01001
  001101
- 01001
  00100   ←—— Rest
```


Binäre Division (Forts.)

- serieller Algorithmus zur Division zweier n-Bit Zahlen a und b:
- mit einem n-Bit Register b, einem 2n-Bit Register q, einem n-Bit Addierer/Subtrahierer direkt in Hardware implementierbar
- nach n Schritten befindet sich der Quotient q in qL, der Rest in qH
- in aktuellen Prozessorarchitekturen eingesetzte Divisionsverfahren:
  - iterative Approximation (durch Multiplikation und Addition)
  - SRT Algorithmus (simultane tabellenbasierte Generierung mehrerer Quotientenbits)

```
q = (qH,qL) = (0,a)
for i = 0 to n-1 {
   shift left (qH,qL) by 1
   qH = qH - b
   if (q 2n-1 = 0)
      q0 = 1
   else
      q0 = 0
      qH = qH + b
}
```



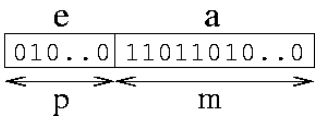
Gleitkommazahlen

- in vielen technischen und wissenschaftlichen Anwendungen erforderlich:
  - hohe Präzision und Genauigkeit
  - große Dynamik

  möglich durch Verwendung von Gleitkommazahlen
- allgemeine Gleitkommazahl zur Basis r („*radix*“) definiert durch $x = a \times r^e$ mit
  Argument oder Mantisse a
  Exponent oder Charakteristik e
- eine Gleitkommazahl $x \neq 0$ zur Basis r heißt normalisiert, wenn für die Mantisse a gilt: $1/r \leq a < 1$

Binäre Gleitkommazahlen

- Verwendung der Basis 2, d.h. eine binäre Gleitkommazahl x ist definiert durch $x = a \times 2^e$
  mit m-stelliger Mantisse a
  und p-stelligem Exponent e



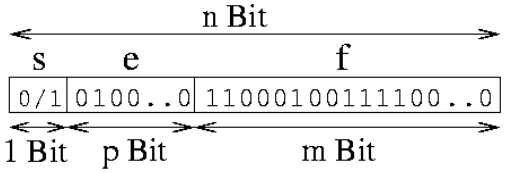
- eine binäre Gleitkommazahl $x \neq 0$ heißt normalisiert, wenn das höchstwertige Mantissenbit den Wert 1 hat
  $\Rightarrow$ zwei Interpretationen: `1.XXXXXXX` und `[0].1XXXXXX`
- häufig Darstellung des Exponenten mit Bias b: $x = a \times 2^{e-b}$
  Wahl von $b = 2^{p-1} - 1$ bewirkt Transformation des Bereiches für den Exponenten e von $0 \ldots 2^p - 1$ in $-(2^{p-1}-1) \ldots 2^{p-1}$
  $\Rightarrow$ einfache Kodierung positiver und negativer Exponenten

Binäre Gleitkommazahlen (Forts.)

- Mantisse und Exponent können positiv und negativ sein
- viele Variationsmöglichkeiten bei der Definition eines Formates zur Kodierung binärer Gleitkommazahlen:
  1) Wahl der Gesamtwortbreite n
  2) Wahl vom m und p = n–m
  3) Wahl einer Reihenfolge von a und e
  4) Darstellung der Mantisse im Einerkomplement, im Zweierkomplement oder mittels Vorzeichen und Betrag
  5) Darstellung des Exponenten im Einer- oder Zweierkomplement, mittels Vorzeichen und Betrag oder durch Subtraktion eines Bias
- früher unterschiedliches Gleitkommaformat in jedem Prozessor, heute überwiegend Verwendung des IEEE 754 Standard

IEEE 754 Standard

- allgemeine Definition: $x = (-1)^s \times 1.f \times 2^{e-b}$
- Mantisse aus Vorzeichen s und normalisiertem Betrag a = 1.f im Bereich `1.00..00` bis `1.11..11`
  (1 vor dem Komma wird nicht kodiert $\Rightarrow$ erhöhte Präzision)
- Aufbau einer n-Bit IEEE Gleitkommazahl:



- p-stelliger Exponent mit Bias $b = 2^{p-1}-1$, gültiger Exponent e nur im Bereich $e_{min} = 0 < e < e_{max} = 2^p-1 = 2b+1$

  $\Rightarrow$ darstellbarer Zahlenbereich: $\pm 2^{1-b}$ ... $(2-2^{-m}) \times 2^b$

IEEE 754 Standard (Forts.)

- 3 verschiedene Formate spezifiziert:

	single precision	*double precision*	*quad precision*
n	32	64	128
m	23	52	112
s	1	1	1
p	8	11	15
e_{min}	0	0	0
e_{max}	255	2047	32767
b	127	1023	16383
$\lvert x_{min} \rvert$	$2^{-126} \approx 10^{-38}$	$2^{-1022} \approx 10^{-308}$	$2^{-16382} \approx 10^{-4932}$
$\lvert x_{max} \rvert$	$(2-2^{-23})\times 2^{127} \approx 10^{38}$	$(2-2^{-52})\times 2^{1023} \approx 10^{308}$	$(2-2^{-112})\times 2^{16383} \approx 10^{4932}$

IEEE 754 Standard (Forts.)

- $e = e_{min} = (\texttt{00..00})_2$ und $e = e_{max} = (\texttt{11..11})_2$ werden zur Kodierung besonderer Zahlen verwendet:

  $x = +0$ („*positive Zero*“): $e = 0,\ f = 0,\ s = 0$

  $x = -0$ („*negative Zero*“): $e = 0,\ f = 0,\ s = 1$

  $x = +\infty$ („*positive Infinity*“): $e = e_{max},\ f = 0,\ s = 0$

  $x = -\infty$ („*negative Infinity*“): $e = e_{max},\ f = 0,\ s = 1$

  $x =$ NaN („*Not a Number*“): $e = e_{max},\ f \neq 0,$ s beliebig

  $x = (-1)^s \times 0.f \times 2^{1-b}$ („*Denormalized Number*“): $e = 0,\ f \neq 0$

- Denormalisierte Gleitkommazahlen ermöglichen die Darstellung sehr kleiner Werte im Bereich 2^{1-b-m} ... 2^{1-b}

Multiplikation von Gleitkommazahlen

Algorithmus zur Multiplikation zweier IEEE-Gleitkommazahlen
$x = (-1)^s \times a \times 2^{\alpha-bias}$ und $y = (-1)^t \times b \times 2^{\beta-bias}$:

1) Multipliziere Mantissen: $c = a \times b$
   $a=1.f_a$ und $b=1.f_b$ haben m+1 Stellen $\Rightarrow$ c hat 2m+2 Stellen !
2) Addiere Exponenten: $\gamma = \alpha + \beta - bias$
3) Berechne Vorzeichen des Produktes: $u = s \oplus t$
4) Normalisiere Ergebnis $z = (-1)^u \times c \times 2^{\gamma\text{-}bias}$
   a) Falls $c \geq 2$, schiebe c um 1 nach rechts und inkrementiere γ
   b) Schiebe c um 1 nach links
   c) Setze $c = 1.f_c = (c_{2m+1}\ c_{2m}\ c_{2m-1}\ \ldots\ c_{m+1})$, ggf. mit Rundung
5) Behandlung von Sonderfällen:
   a) Überlauf, falls $\gamma \geq 2^p - 1 \Rightarrow z := +\infty$ oder $z := -\infty$ (bei u = 0 bzw. 1)
   b) Unterlauf, falls $\gamma < 1 \Rightarrow$ Denormalisierung durchführen
   c) Zero, falls $c = 0 \Rightarrow z := 0$

Addition/Subtraktion von Gleitkommazahlen

Algorithmus zur Addition/Subtraktion zweier Gleitkommazahlen
$x = (-1)^s \times a \times 2^{\alpha-bias}$ und $y = (-1)^t \times b \times 2^{\beta-bias}$ im IEEE Format:

1) Sortiere x und y derart, daß x die Zahl mit kleinerem Exponenten ist
2) Anpassung der Exponenten: Transformiere x in die Gleitkommazahl
   $x' = (-1)^s \times a' \times 2^{\beta-bias}$ durch Rechtsschieben von a um $\beta-\alpha$ Bitstellen
3) Addiere/Subtrahiere Mantissen:
   a) Falls nötig, bilde Zweierkomplement von a´ oder b
   b) Berechne $c = a' + b$ bzw. $c = a' + (\ b)$
   c) Falls $c < 0$, setze Vorzeichenbit $u = 1$ und bilde Zweierkomplement
4) Normalisiere Ergebnis $z = (-1)^u \times c \times 2^{\beta-bias}$
   a) Falls $c \geq 2$, schiebe c nach rechts (ggf. Rundung) und inkrementiere β
   b) Falls $c < 1$, schiebe c nach links und dekrementiere β
   c) Wiederhole a) bzw. b), bis $1 \leq c < 2$ oder $c = 0$
5) Behandlung von Sonderfällen (Überlauf ? , Unterlauf ? , c = 0 ?)