technische universität
dortmund

Rechnerstrukturen WS 2011/12

- Boolesche Funktionen und Schaltnetze
  - Gleitkommazahlen-Arithmetik
    - Multiplikation von Gleitkommazahlen
    - Addition von Gleitkommazahlen

Hinweis: *Folien teilweise a. d. Basis von Materialien von Thomas Jansen*

29. August 2011

Gleitkommazahlen-Arithmetik

Darstellung gemäß IEEE 754-1985

$$x = (-1)^{s_x} \cdot m_x \cdot 2^{e_x}$$
$$y = (-1)^{s_y} \cdot m_y \cdot 2^{e_y}$$

- s Vorzeichenbit
- m Mantisse (Binärdarstellung, inklusive impliziter 1)
- e Exponent (Exzessdarstellung, $b = 2^{l-1} - 1$)

Ergebnis $z = (-1)^{s_z} \cdot m_z \cdot 2^{e_z}$

Vereinfachung Wir ignorieren das Runden.

Aber: Wichtiger Teil des IEEE 754 Standards!

Weitere Details z.B. in:

David Goldberg (1991): What every computer scientist should know about floating-point arithmetic. ACM Computing Serveys 23(1):5–48.

Multiplikation von Gleitkommazahlen

$$\begin{aligned} x &= (-1)^{s_x} \cdot m_x \cdot 2^{e_x} \\ y &= (-1)^{s_y} \cdot m_y \cdot 2^{e_y} \\ z = x \cdot y &= (-1)^{s_z} \cdot m_z \cdot 2^{e_z} \end{aligned}$$

Beobachtung $z = (-1)^{s_x \oplus s_y} \cdot (m_x \cdot m_y) \cdot 2^{e_x + e_y}$

also

1. $s_z := s_x \oplus s_y$ (trivial)
2. $m_z := m_x \cdot m_y$ (Multiplikation von Betragszahlen wie gesehen, implizite Einsen nicht vergessen!)
3. $e_z := e_x + e_y$ (Addition, wegen Exzessdarstellung $e_x + e_y - b$ berechnen)

Beispiel Multiplikation Gleitkommazahlen

x	1	1000	0101	101	0000	0000	0000	0000	0000
y	1	1000	0111	110	1000	0000	0000	0000	0000

Vorzeichen $s_z = 1 \oplus 1 = 0$

Exponent Bias ist $2^{l-1} - 1 = (1000\,0000)_2 - 1$

$(1000\,0111)_2 - ((1000\,0000)_2 - 1) = (111)_2 + 1 = (1000)_2$

$(1000\,0101)_2 + (1000)_2 = (1000\,1101)_2$

$(1000\,1101)_2$ ist vorläufiger Exponent

Mantisse

	1,	1	0	1	·	1,	1	1	0	1
	1	1	0	1					
		1	1	0	1				
			1	1	0	1			
				0	0	0	0		
+					1	1	0	1	
1	0,	1	1	1	1	0	0	1	

Normalisieren:

Komma 1 Stelle nach links

Exponent zum Ausgleich +1

implizite Eins streichen

z	0	1000	1110	011	1100	1000	0000	0000	0000

Addition von Gleitkommazahlen

$$\begin{aligned} x &= (-1)^{s_x} \cdot m_x \cdot 2^{e_x} \\ y &= (-1)^{s_y} \cdot m_y \cdot 2^{e_y} \\ z = x + y &= (-1)^{s_z} \cdot m_z \cdot 2^{e_z} \end{aligned}$$

Beobachtung einfach, wenn $e_x = e_y$
dann $m_1 \cdot 2^e + m_2 \cdot 2^e = (m_1 + m_2) \cdot 2^e$

Plan

1. Ergebnis wird „so ähnlich“ wie Zahl mit größerem Exponenten, darum Mantisse der Zahl mit kleinerem Exponenten anpassen
2. Mantissen auf jeden Fall addieren, bei unterschiedlichen Vorzeichen dazu eine Mantisse negieren (Zweierkomplement)
3. anschließend normalisieren

Algorithmus zur Addition

1. Falls $e_x < e_y$, dann x und y komplett vertauschen.
2. Falls Vorzeichen ungleich, dann Vorzeichen von s_y invertieren und Übergang von y zu $-y$ im Zweierkomplement („$\overline{y} + 1$“). $s_z := s_x$
3. Mantisse m_y um $e_x - e_y$ Stellen nach rechts verschieben (Exponenten „virtuell“ jetzt angeglichen)
   Achtung Kann zum “Verlust” signifikanter Stellen führen!
4. $m_z := m_x + m_y$
   Falls $e_x = e_y$, Vorzeichenwechsel möglich. Dann s_z invertieren.
5. $e_z := e_x$. Ergebnis normalisieren

Achtung Bei Mantissen an implizite Einsen denken!

klar Keine separate Subtraktion erforderlich.
Vorzeichenwechsel trivial.
Addition negativer Zahlen enthalten durch Zweierkomplement.

Beispiel Addition Gleitkommazahlen

x	1	1001	0101	111	0010	0000	0000	0000	0000
y	1	1001	0100	110	0001	1000	0000	0000	0000

$e_x > e_y$, Vorzeichen gleich, also zunächst nur $s_z := 1$

Mantisse m_y um $e_x - e_y = 1$ Stelle nach rechts verschieben

$\rightsquigarrow$ 0,111000011

Mantissen addieren

		1,	1	1	1	0	0	1		
	+	0,	1	1	1	0	0	0	0	1	1
	1	0,	1	1	0	0	0	1	0	1	1

Normalisieren

- Komma um 1 Stelle nach links verschieben $\rightsquigarrow$ 1,0110001011
- Exponent um 1 vergrößern $\rightsquigarrow$ 1001 0110

z	1	1001	0110	011	0001	0110	0000	0000	0000

Noch ein Beispiel zur Addition

x	1	1000	0101	010	0000	0000	0000	0000	0000
y	0	1000	0100	101	1010	0000	0000	0000	0000

klar $e_x > e_y$ ✓

weil $s_x \neq s_y$ s_y invertieren
Vorzeichenwechsel bei m_y
in Zweierkomplementdarstellung

x	1	1000	0101	1	,	010	0000	0000	0000	0000	0000
aus	0	1000	0100	01	,	101	1010	0000	0000	0000	0000
wird y	1	1000	0100	10	,	010	0110	0000	0000	0000	0000

Noch ein Beispiel zur Addition (2)

x	1	1000	0101	1	,	010	0000	0000	0000	0000	0000
y	1	1000	0100	10	,	010	0110	0000	0000	0000	0000

jetzt e_y an e_x anpassen, m_y verschieben

x	1	1000	0101	1	,	010	0000	0000	0000	0000	0000
y	1	1000	0101	11	,	001	0011	0000	0000	0000	0000
z	1	1000	0101	100	,	011	0011	0000	0000	0000	0000

Erinnerung „überfließende“ 1 einfach ignorieren

z	1	1000	0101	0	,	011	0011	0000	0000	0000	0000

Normalisieren Komma um zwei Stellen nach rechts verschieben
Exponent zum Ausgleich um zwei verkleinern

z	1	1000	0011	100	1100	0000	0000	0000	0000

Fehlerquellen bei der Gleitkommaarithmetik

- Rundung, wenn Berechnungsergebnis zur korrekten Darstellung *mehr* signifikante Bits (d.h. i.d. Mantisse) erfordert als verfügbar
  (kann bei Multiplikation *und* Addition auftreten)
- "Verlust" niederwertiger Bits durch Angleich der Exponenten während der Addition
  Worst Case: $x \gg y$ und $y \neq 0$ aber $x + y = x$ 

Und nicht zu vergessen: Darstellung nur einer *extrem kleinen* Auswahl der rationalen Zahlen möglich, variiert mit der Größenordnung der repräsentierten Zahlen!

Probleme bei der Addition: Ein Szenario

Gegeben: Folge von n Gleitkommazahlen $[x_i]$ mit $0 \leq i \leq n$
(z.B. gespeichert in einem Feld/Array `x[i]`)

Aufgabe: Berechne Summe S ... möglichst exakt
(d.h. mit den Möglichkeiten der Gleitkommaarithmetik)

Naive Lösung: Direkte Summation, d.h. berechne:

$$S\uparrow = \sum_{i=0}^{n-1} x_i \quad \text{oder lieber} \quad S\downarrow = \sum_{i=n-1}^{0} x_i$$

~~Theorie/Intuition: Beide Summationen liefern dasselbe Ergebnis!~~

Praxis: $S\uparrow$ und $S\downarrow$ sind i.a. nicht gleich! $\longrightarrow$ Beispielprogramm (in C)

Mögliche Abhilfe: Erhöhung der Genauigkeit (i.d.R. schwierig)
... oder "schlauere" Berechnung ;-)

Fehlerreduktion: Kahan-Summation

Algorithmus zur numerisch stabileren Berechnung von $S = \sum_{i=0}^{n-1} x_i$:

```
S = 0;                          /* Summe */
E = 0;                          /* geschaetzter Fehler */
for i = 0 to n-1 {
        Y = x[i] - E;           /* bish. Fehler beruecksichtigen */
        Z = S + Y;              /* neues Summationsergebnis */
        E = (Z - S) - Y; /* neue Fehlerschaetzung */
        S = Z;
        }
```

⟶ Beispielprogramm (in C)

Fehlerreduktion: Kahan-Summation (2)

Veranschaulichung des fehlerkompensierenden Berechnungsablaufs: