Computerarithmetik

Vortrag im Rahmen des Arbeitsseminars der Mitarbeiter der Arbeitsgruppe Technomathematik

Zentrum für Technomathematik

Universität Bremen veranstaltet in Uttendorf

Henning Thielemann

2003.02.11

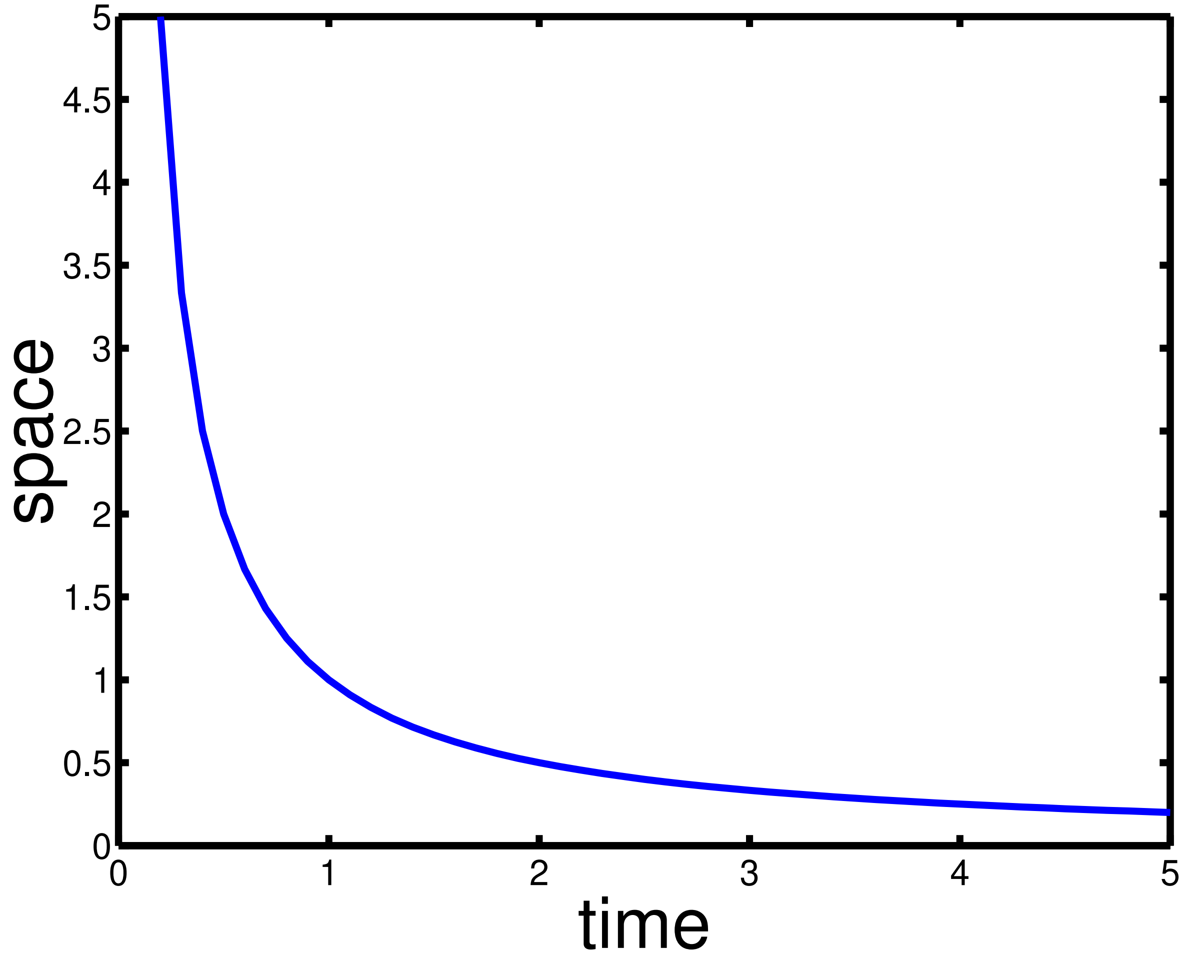
Übelblick

- Zielstellung
- Wurzel
- Logarithmus
- Exponentialfunktion
- Winkelfunktionen
- Division

Zentrum für
Technomathematik

Zielstellung

Algorithmen müssen Balance zwischen Speicherverbrauch und Zeitaufwand herstellen.



Zentrum für Technomathematik

Methodenvergleich

- Schnell und speicherhungrig: Wertetabelle
- Langsam und platzsparend: Taylor-Entwicklung, Iterationsverfahren
- Gutes Mittel: Pseudomultiplikation und -division mit wenigen Konstanten und Beschränkung auf Addition und einfache Skalierung

Zentrum für Technomathematik

Wurzel

- Taylor für $|x| < 1$

$$
\begin{aligned}
(1+x)^\alpha &= \sum_{k=0}^{\infty} \binom{\alpha}{k} x^k \\
&= \binom{\alpha}{0} + \binom{\alpha}{1} \cdot x + \binom{\alpha}{2} \cdot x^2 + \ldots
\end{aligned}
$$

- Newton

$$
\begin{aligned}
y_{k+1} &= \frac{1}{n} \cdot \left((n-1) y_k + \frac{x}{y_k^{n-1}} \right) \\
\lim_{k \to \infty} y_k &= \sqrt[n]{x}
\end{aligned}
$$

- Potenzgesetz

$$
x^\alpha = \mathrm{e}^{\alpha \cdot \ln x}
$$

Quadratwurzel als Umkehrung des Quadrierens

Es sei x der Radikand $(0 \leq x < 1)$ und y eine Näherung für die Wurzel mit $y \leq \sqrt{x}$.

Wähle ε mit $y + \varepsilon \leq \sqrt{x}$ oder gleichbedeutend $(y + \varepsilon)^2 \leq x$, dann ist $y + \varepsilon$ neue Näherung.

Teste für ε nacheinander die Wertigkeiten der Binärstellen, also $2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}, \ldots$

Beispiel: Wurzel von $x = 0.1_2$

y	y^2	$y^2 \; ? \; x$
0.1	0.01	$<$
0.11	0.1001	$>$
0.101	0.011001	$<$
0.1011	0.01111001	$<$
0.10111	0.1000010001	$>$
0.101101	0.011111101001	$<$
0.1011011	0.10000001011001	$>$
0.10110101	0.0111111111111001	$<$

Zentrum für Technomathematik

Wurzelbehandlung mit Binomen

Bislang wird für $(y+\varepsilon)^2$ eine Multiplikation benötigt. Diese lässt sich umgehen, denn es gilt:

$$(y+\varepsilon)^2 = y^2 + 2y\varepsilon + \varepsilon^2$$

y^2	aus dem letzten Rechenschritt bekannt
$2y\varepsilon$	Verschiebung von y um einige Binärstellen
ε^2	einzelnes Bit irgendwo in den Nachkommastellen

y	y^2	$2y\varepsilon + \varepsilon^2$
0.1	0.01	0.01
0.11	0.1001	0.0101
0.101	0.011001	0.001001
0.1011	0.01111001	0.00010101
0.10111	0.1000010001	0.0000101101
0.101101	0.011111101001	0.000001011001
0.1011011	0.10000001011001	0.00000010110101
0.10110101	0.0111111111111001	0.0000000101101001

$$\sqrt{0.1_2} \approx 0.10110101_2$$
$$\sqrt{0.5} \approx 0.70703$$

Zentrum für Technomathematik

Logarithmus

- Taylor für $|x| < 1$

$$\ln(1+x) = \sum_{k=1}^{\infty} (-1)^{k+1} \cdot \frac{x^k}{k}$$
$$= x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - + \ldots$$

- Logarithmengesetz

$$\ln x^2 = 2 \cdot \ln x$$

- binärer Logarithmus

$$\operatorname{lb} x^2 = 2 \cdot \operatorname{lb} x$$
$$\operatorname{lb} 2x = 1 + \operatorname{lb} x$$

Logarithmus-Algorithmus

Berechnung von $\operatorname{lb} x$

Es sei $1 \leq x < 2$, was sich durch Skalierung mit Zweiterpotenzen immer erreichen lässt. Dann hat der Wert des Logarithmus die Binärdarstellung

$$\operatorname{lb} x = 0.y_1 y_2 y_3 y_4 y_5 \ldots$$

Bestimme zuerst y_1:

$$\begin{aligned} \operatorname{lb} x^2 &= 2 \cdot \operatorname{lb} x \\ &= y_1.y_2 y_3 y_4 y_5 \ldots \end{aligned}$$

Teste den Wert von y_1:

$$\begin{aligned} y_1 = 0 \quad &\Leftrightarrow \quad \operatorname{lb} x^2 < 1 \\ &\Leftrightarrow \quad x^2 < 2 \end{aligned}$$

Wiederhole den Test mit $x' = x^2/2^{y_1}$ usw.

Zentrum für Technomathematik

Beispiel

Berechne $\operatorname{lb} 1.1_2$

Operation	x	$\operatorname{lb} x$	Ergebnis
	1.1	$0.y_1y_2y_3y_4y_5$	
$x \mapsto x^2$	10.01	$y_1.y_2y_3y_4y_5$	$y_1 = 1$
$x \mapsto \frac{x}{2}$	1.001	$0.y_2y_3y_4y_5$	
$x \mapsto x^2$	1.010001	$y_2.y_3y_4y_5$	$y_2 = 0$
$x \mapsto x^2$	1.1011010	$y_3.y_4y_5$	$y_3 = 0$
$x \mapsto x^2$	10.10010000	$y_4.y_5$	$y_4 = 1$
$x \mapsto \frac{x}{2}$	1.01001000	$0.y_5$	
$x \mapsto x^2$	1.10100101	$y_5.$	$y_5 = 0$

Ergebnis:

$$\begin{aligned}
\operatorname{lb} 1.1_2 &\approx 0.10010_2 \\
\operatorname{lb} 1.5 &\approx 0.5625 \\
\operatorname{lb} 1.5 &= 0.58496250\ldots
\end{aligned}$$

Zentrum für Technomathematik

Auswertung

Quelle	Ziel
$0.y_1y_2y_3y_4y_5$	0
$0.y_2y_3y_4y_5$	$0.y_1$
$0.y_3y_4y_5$	$0.y_1y_2$
$0.y_4y_5$	$0.y_1y_2y_3$
$0.y_5$	$0.y_1y_2y_3y_4$
$0.$	$0.y_1y_2y_3y_4y_5$

- Vorteil: Keine vorberechneten Konstanten nötig
- Nachteil: x^2 aufwändig zu berechnen (Multiplikation)

Hinweis: Eine Multiplikation $c \cdot x$ mit einer Konstanten c ist oft schneller als eine Multiplikation $x \cdot y$ zweier Variablen x, y, denn die Multiplikation mit einer Konstanten lässt sich in eine feste Anzahl Additionen zerlegen.

Beispiel:

$$1.001_2 \cdot x = x + 0.001_2 \cdot x$$

Zentrum für Technomathematik

Neuer Ansatz

Speichere Satz von Konstanten

$$2^{2^{-1}}, 2^{2^{-2}}, 2^{2^{-3}}, 2^{2^{-4}}, \ldots$$

oder anders geschrieben

$$\sqrt{2},\ \sqrt[2^2]{2},\ \sqrt[2^3]{2},\ \sqrt[2^4]{2}, \ldots$$

Es ist

$$\begin{aligned} \operatorname{lb} x &= \sum_{k=1}^{\infty} y_k \cdot 2^{-k} \\ x &= \prod_{k=1}^{\infty} 2^{y_k 2^{-k}} \end{aligned}$$

Algorithmus: Vergleiche der Reihe nach jede Konstante $2^{2^{-k}}$ mit x. Falls $x \geq 2^{2^{-k}}$ dann teile $x/2^{2^{-k}}$ und setze $y_k = 1$.

Zentrum für Technomathematik

Auswertung

Quelle	Ziel
$0.y_1y_2y_3y_4y_5$	0
$0.0\ y_2y_3y_4y_5$	$0.y_1$
$0.0\ 0\ y_3y_4y_5$	$0.y_1y_2$
$0.0\ 0\ 0\ y_4y_5$	$0.y_1y_2y_3$
$0.0\ 0\ 0\ 0\ y_5$	$0.y_1y_2y_3y_4$
$0.0\ 0\ 0\ 0\ 0$	$0.y_1y_2y_3y_4y_5$

- Vorteil: $x/2^{2^{-k}}$ nicht ganz so aufwändig wie x^2
- Nachteil: Vorberechnete Konstanten nötig

Kann man die verbleibende Multiplikation noch weiter vereinfachen?

Entwicklung des Verfahrens:

	Operationen für x	Operationen für y
1.	$x^2, x/2$	$y + 2^{-k}$
2.	$x/2^{2^{-k}}$	$y + 2^{-k}$
3.	x/c_k	$y + d_k$

Zentrum für Technomathematik

Pseudodivision und -multiplikation

Wähle

$$c_k = 1 + 2^{-k} = \underbrace{1.00\ldots00}_{k}1$$

und

$$d_k = \mathrm{lb}\left(1 + 2^{-k}\right) = \mathrm{lb}\, c_k$$

dann ist

$$z \cdot c_k = z + 2^{-k} z$$

durch Skalierung und Addition berechenbar und statt $x/c_k < z$ wird $x < c_k \cdot z$ getestet.

Das entspricht der Darstellung:

$$\begin{aligned} x &= 1.1_2^{y_1} \cdot 1.01_2^{y_2} \cdot 1.001_2^{y_3} \cdot \ldots \\ \mathrm{lb}\, x &= y_1 \cdot \mathrm{lb}\, 1.1_2 + y_2 \cdot \mathrm{lb}\, 1.01_2 \\ &\quad + y_3 \cdot \mathrm{lb}\, 1.001_2 + \ldots \end{aligned}$$

Es gilt $y_k \leq 1$, weil

$$\begin{aligned} \left(1 + 2^{-k}\right)^2 &= 1 + 2^{1-k} + 2^{-2k} \\ &> 1 + 2^{1-k} \end{aligned}$$

Exponentialfunktion

Berechnung von e^y als Umkehrung vom Logarithmus. Zerlege y in eine Linearkombination von Logarithmen durch fortgesetzte Subtraktion von $\mathrm{lb}\left(1+2^{-k}\right)$ und Vergleiche gegen 0

$$\begin{aligned} y &= y_1 \cdot \mathrm{lb}\, 1.1_2 + y_2 \cdot \mathrm{lb}\, 1.01_2 \\ &\quad + y_3 \cdot \mathrm{lb}\, 1.001_2 + \ldots \end{aligned}$$

und multipliziere die zugehörigen Faktoren

$$\mathrm{e}^y = 1.1_2^{y_1} \cdot 1.01_2^{y_2} \cdot 1.001_2^{y_3} \cdot \ldots$$

Winkelfunktionen

- Reihen für Sinus und Cosinus konvergieren sehr schnell

$$\begin{aligned}
\sin x &= \sum_{k=0}^{\infty} (-1)^k \cdot \frac{x^{2k+1}}{(2k+1)!} \\
&= x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + - \ldots
\end{aligned}$$

$$\begin{aligned}
\cos x &= \sum_{k=0}^{\infty} (-1)^k \cdot \frac{x^{2k}}{(2k)!} \\
&= 1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + - \ldots
\end{aligned}$$

- Reihe für Arkustangens konvergiert sehr gemächlich, vergleiche mit Logarithmus-Reihe

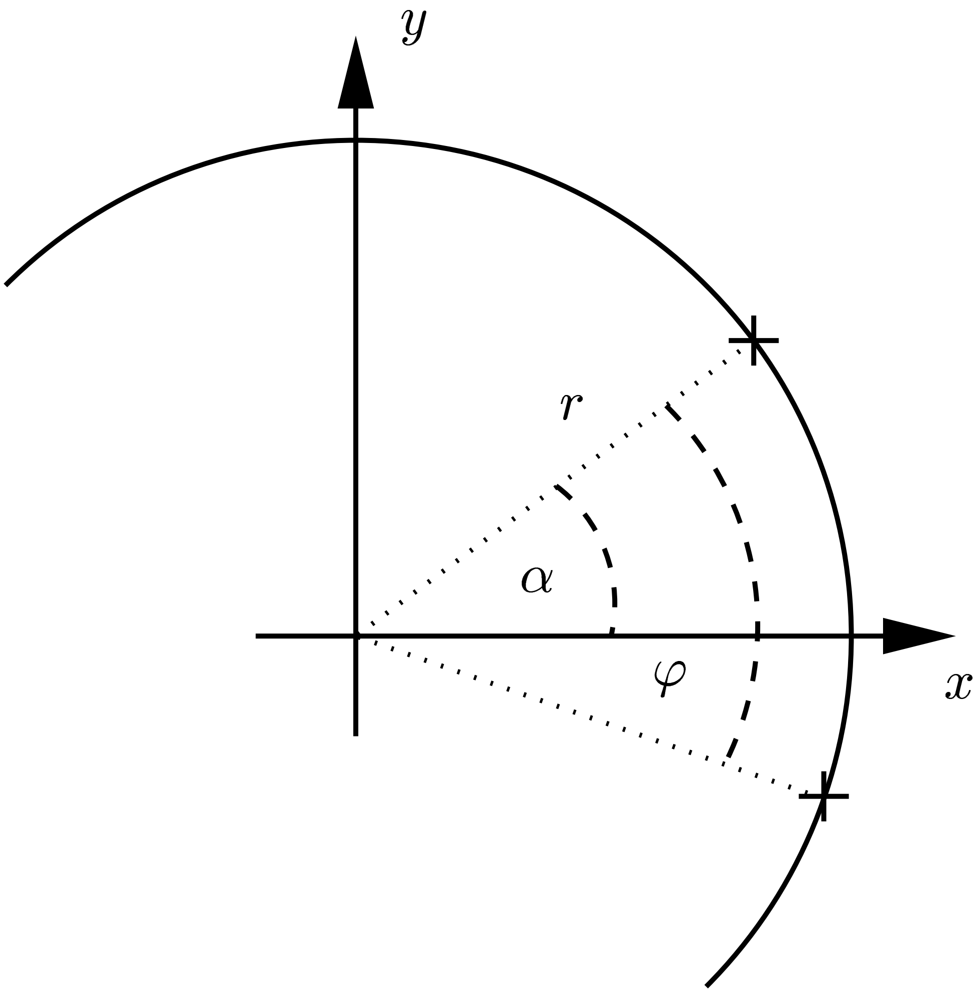
$$\begin{aligned}\arctan x &= \sum_{k=0}^{\infty} (-1)^k \cdot \frac{x^{2k+1}}{(2k+1)} \\ &= x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + - \ldots\end{aligned}$$

Winkelbestimmung

Eine Aufgabe, viele Formulierungen

- $\alpha = \arctan \frac{y}{x}$
- $\alpha = \arg(x + \mathrm{i}\, y)$
- Umwandlung kartesische Koordinaten (x, y) in Polarkoordinaten (r, α)

Versuchsaufbau



$$
\begin{aligned}
x' &= x \cdot \cos\varphi + y \cdot \sin\varphi \\
y' &= -x \cdot \sin\varphi + y \cdot \cos\varphi
\end{aligned}
$$

Dann ist

$$\alpha < \varphi \quad \Leftrightarrow \quad y' < 0$$

Vereinfachung

$\cos\varphi$ und $\sin\varphi$ sind in der Regel „krumme“ Zahlen. Betrag der Zahl $x + \mathrm{i}\,y$ darf ruhig verändert werden. Für $\varphi < \frac{\pi}{2}$ rechne mit

$$\begin{array}{rcrcl} x' \;/\; \cos\varphi & = & x & + & y \cdot \tan\varphi \\ y' \;/\; \cos\varphi & = & -x \cdot \tan\varphi & + & y \end{array}$$

Wähle einfache Werte für $\tan\varphi$, nämlich 2^{-k} und lege Tabelle für die Werte

$$\arctan\frac{1}{2}, \quad \arctan\frac{1}{4}, \quad \arctan\frac{1}{8}, \quad \arctan\frac{1}{16}, \quad \ldots$$

an.

Division

Die schnelle Division zur Basis 4 nach Sweeney, Robertson and Tocher (Radix-4 SRT division)

oder

Warum der Kreis um das Intel-Logo nach einer Umrundung nicht wieder den Anfang trifft.

- Gegeben: X, Y
- Gesucht: Q, R mit $X = Q \cdot Y + R$ und $|R| < |Y|$
- Entspricht Division X/Y mit Quotient Q und Rest R

Schulalgorithmus

Schuldivision mit positiven Binärzahlen $X < Y$:

Bezeichnungen:

- Q_k - k. Nachkommastelle des Quotienten
- R_k - Rest der Division nach $(k-1)$. Iteration

Algorithmus:

1. $R_0 := X$
2. k. Iteration
   Wenn $R_k > Y$
   - dann $Q_k := 1, \Delta := R_k - Y$
   - sonst $Q_k := 0, \Delta := R_k$

   $R_{k+1} := 2 \cdot \Delta$

Beschleunigung

Addition, Subtraktion, Multiplikation lassen sich gut parallelisieren, Division nicht. Was kann man dennoch tun, um Hardwareimplementation der Division zu beschleunigen?

- mit Basen größer als 2 rechnen, um die Anzahl der Iterationen zu verringern
- Vergleiche und Subtraktionen beschleunigen

Der SRT-Algorithmus nutzt beide Strategien:

- Basis 4 statt Basis 2
- anderer und größerer Ziffernvorrat als üblich: $\{-2, -1, 0, 1, 2\}$
- Darstellung des Quotienten mehrdeutig
- Deshalb Näherung bei Subtraktion tolerierbar

Basis 4

Schreibe $-m$ als Ziffer $\bar{m}$.

Betrachte zunächst Division zur Basis 4 mit Ziffernvorrat $\{\bar{1}, 0, 1, 2\}$

Vorteile

- negative Zahlen ohne Vorzeichen darstellbar
- Division funktioniert automatisch für negative Zahlen

k. Iteration

bereits sichergestellt:
$$Y \cdot \bar{1}.\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\ldots_4 \quad < \quad R_k \quad < \quad Y \cdot 2.22222\ldots_4$$
noch sicherzustellen:
$$Y \cdot 0.\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\ldots_4 \quad < \quad R_{k+1} \quad < \quad Y \cdot 0.22222\ldots_4$$

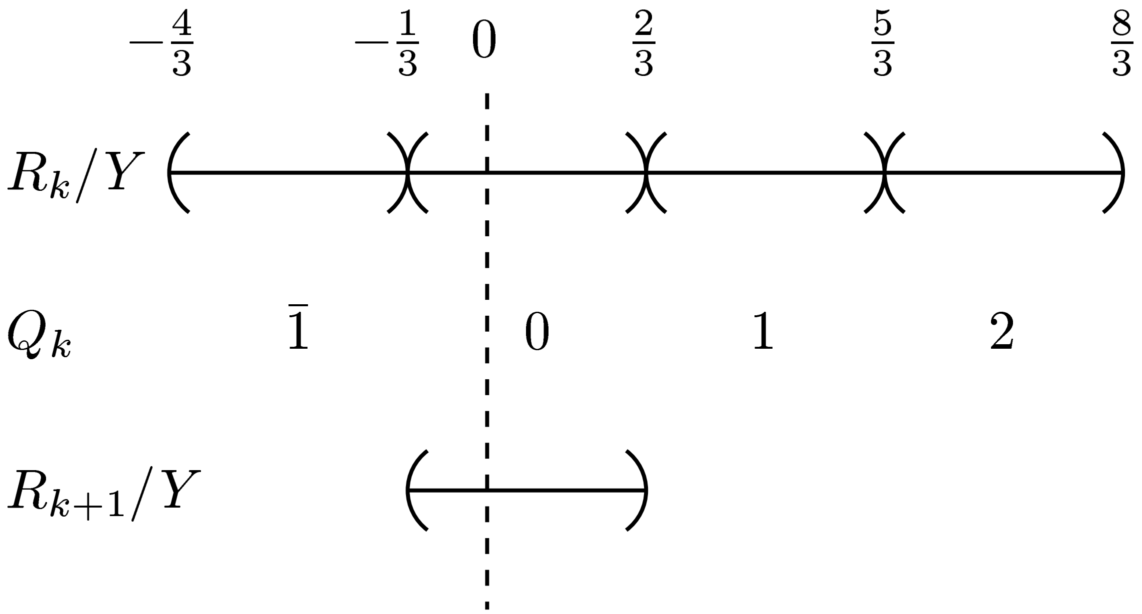
Zentrum für Technomathematik

Dabei ist

$$
\begin{aligned}
1.11111\ldots_4 &= \sum_{j=0}^{\infty} \left(\frac{1}{4}\right)^j \\
&= \frac{1}{1-\frac{1}{4}} \\
&= \frac{4}{3}
\end{aligned}
$$

$$
\begin{array}{ccccc}
-\frac{4}{3}\cdot Y & < & R_k & < & \frac{8}{3}\cdot Y \\
-\frac{1}{3}\cdot Y & < & R_{k+1} & < & \frac{2}{3}\cdot Y
\end{array}
$$

$$R_k = Y \cdot Q_k + R_{k+1}$$



Basis 4 mit Redundanz

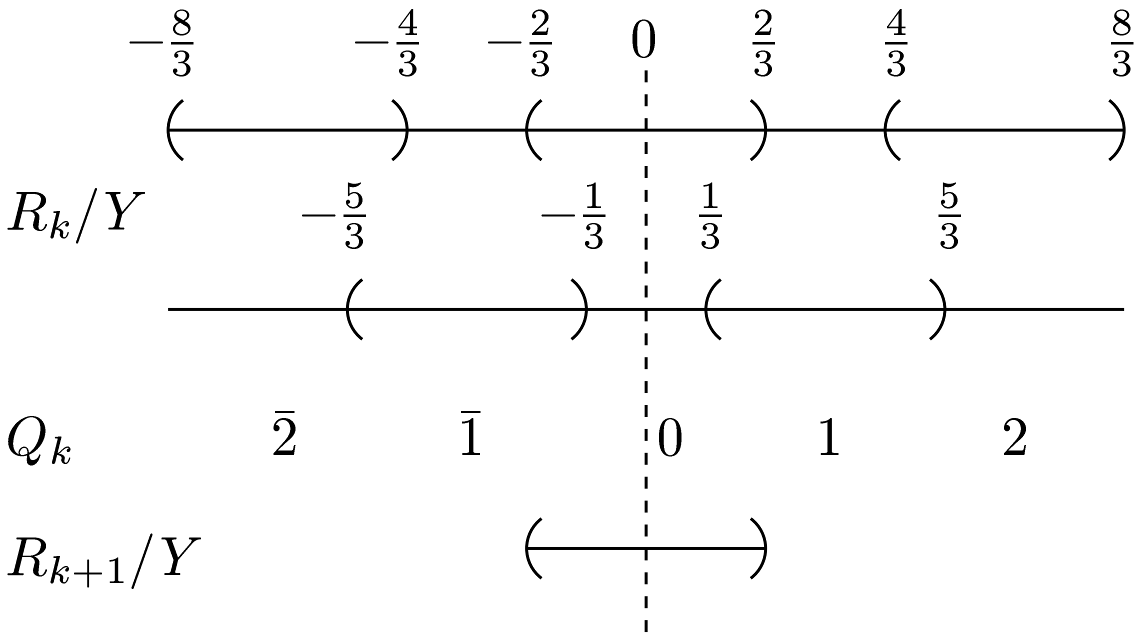
Ziffernvorrat $\{\bar{2}, \bar{1}, 0, 1, 2\}$

Beispiele:

$$2_{10} = 2_4 = 1\overline{2}_4$$

$$6_{10} = 12_4 = 2\overline{2}_4 = 1\overline{22}_4$$

$$R_k = Y \cdot Q_k + R_{k+1}$$



P-D-Diagramm

- P-D-Diagramm bildet den temporären Rest R_k (partial remainder) und den Divisor Y auf den Quotienten (Farbe) ab.

- P-D-Diagramm wird als Wertetabelle im Chip gespeichert ($16 \cdot 86 = 1376$ Werte)

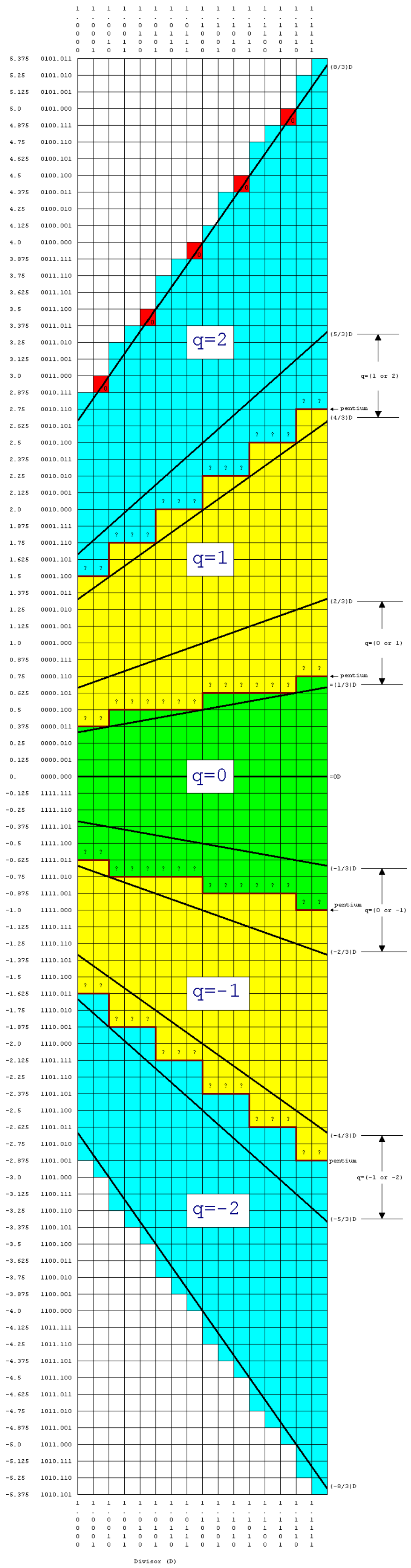
- Benötigte Auflösung:
  7 Bits für R_k und 4 Bits für Y

- Pentium: rot gekennzeichnete Felder mit Quotient 0 statt 2 besetzt

- Deshalb

$$\frac{5506153}{294911} = \begin{cases} 18.67055823621364 & : \text{exakt} \\ 18.66990719233938 & : \text{Pentium} \end{cases}$$

Zentrum für Technomathematik



q=2
q=1
q=0
q=-1
q=-2
(8/3)D
(5/3)D
(4/3)D
(2/3)D
(1/3)D
0D
(-1/3)D
(-2/3)D
(-4/3)D
(-5/3)D
(-8/3)D
pentium
Divisor (D)

Schnelle Addition

Schuladdition in Hardware recht langsam, wegen Durchreichen des Übertrages:

$$\begin{array}{rr} & 11111111 \\ + & 00000001 \\ \hline = & 100000000 \end{array}$$

Schneller lässt sich die Operation „Reduziere drei Summanden zu zwei Summanden“ ausführen. Schreibe Summe und Übertrag getrennt als neue Summanden auf.

$$\begin{array}{rrl} & 11010000 & \\ + & 00110011 & \\ + & 10011010 & \\ \hline = & 01111001 & \text{Summe} \\ + & 10010010 & \text{Übertrag} \end{array}$$

Beispiel

$$
\begin{aligned}
5506153 &= 1.01010000000100011010010_2 \cdot 2^{22} \\
294911 &= 1.00011111111111111100000_2 \cdot 2^{18}
\end{aligned}
$$

- Schreibe Reste und Divisor im Binärsystem und den Quotienten zur Basis 4.

- vorzeichenbehaftete Zahlen - Binärzahlen im Zweierkomplement

- Spalte Reste auf in Summe und Überträge:

$$R_k = R_{\mathsf{s},k} + R_{\mathsf{c},k}$$

Zentrum für Technomathematik

1. Iteration

$$
\begin{aligned}
R_{\mathsf{s},0} &= 0001.0101000000010001101001000000 \\
R_{\mathsf{c},0} &= 0000.0000000000000000000000000000 \\
Y &= 0001.0001111111111111110000000000
\end{aligned}
$$

Aus Wertetabelle: $Q_0 = 1$

$$
\begin{aligned}
R_{\mathsf{s},0} &= 0001.0101000000010001101001000000 \\
R_{\mathsf{c},0} &= 0000.0000000000000000000000000001 \\
-Y &\approx 1110.1110000000000000001111111111 \\
\hline
R_{\mathsf{s},1}/4 &= 1111.1011000000010001100110111110 \\
R_{\mathsf{c},1}/4 &= 0000.100000000000000001001000001
\end{aligned}
$$

Zentrum für Technomathematik

2. Iteration
Benötige erste sieben Stellen von R_1

$$
\begin{array}{rcl}
R_{\mathsf{s},1} & = & 1110.110\ 0000001000110011011111000 \\
R_{\mathsf{c},1} & = & 0010.000\ 0000000000001001000001000 \\
\hline
R_1 & \approx & 0000.110
\end{array}
$$

Aus Wertetabelle: $Q_1 = 1$
Fehler beim Abrunden von R_1 wird dadurch aufgefangen, dass sich die Bereiche für jede Quotientenziffer weit genug überlappen!

$$
\begin{array}{rcl}
R_{\mathsf{s},1} & = & 1110.1100000001000110011011111000 \\
R_{\mathsf{c},1} & = & 0010.0000000000000001001000001001 \\
-Y & \approx & 1110.1110000000000000001111111111 \\
\hline
R_{\mathsf{s},2}/4 & = & 0010.0010000001000111011100001110 \\
R_{\mathsf{c},2}/4 & = & 1101.100000000000000001011111001
\end{array}
$$

Zentrum für Technomathematik

3. Iteration

$$\begin{array}{rcl} R_{\mathsf{s},2} & = & 1000.100\ 000010001110111000011110 \\ R_{\mathsf{c},2} & = & 0110.000\ 000000000000101111111001 \\ \hline R_2 & \approx & 1110.100 \end{array}$$

Aus Wertetabelle: $Q_2 = \bar{1}$

usw. usf. ...

Endergebnis:

$$\begin{array}{cccccccccccc} Q & = & 1. & 1 & \bar{1} & \bar{1} & \bar{1} & \bar{1} & \bar{1} & 2 & 2 & _4 \\ \\ & + & 1. & 01 & 00 & 00 & 00 & 00 & 00 & 10 & 10 & \\ & - & 0. & 00 & 01 & 01 & 01 & 01 & 01 & 00 & 00 & \\ \hline & & 1. & 00 & 10 & 10 & 10 & 10 & 11 & 10 & 10 & \end{array}$$

$$\begin{array}{rcl} Q & = & 1.1669_{10} \\ 2^4 \cdot Q & = & 18.67041_{10} \end{array}$$

Zentrum für Technomathematik

Wie wurde der Fehler im Pentium entdeckt?

- erster Pentium Mai 1993 auf dem Markt, 2 Millionen verkauft

- Dr. Thomas R. Nicely: Reziproke von großen Primzahlzwillingen

- meiste Berechnungen auf Intels 80486, seit März 1994 ein Pentium

- 13.06.1994: Probleme bemerkt mit 824633702441 und 824633702443

- 30.10.1994: Dr. Nicely berichtet einer Hand voll Leute von dem Fehler, nach dem Intel nichts zur Klärung beigetragen hatte

- Fehler wird im Netz heiß diskutiert

- 10.11.1994: Andreas Kaiser zeigt 23 Fälle, in denen $\frac{1}{x}$ falsch berechnet wird

- 16.11.1994: Tim Coe verbreitet sein Programm, das den Pentium-Fehler nachbildet

- 20.12.1994: Intel muss Fehler öffentlich zugeben und bietet Umtausch an

Literatur

[1] Günter Jorke, Bernhard Lampe, Norbert Wengel: „Arithmetische Algorithmen der Mikrorechentechnik“, Berlin 1989

[2] David Deley: „The Pentium division flaw“, `http://members.cox.net/srice1/pentbug/introduction.html`