Lehrstuhl für Rechnerarchitektur und Parallele Systeme
Fakultätfür Informatik
Technische Universität München

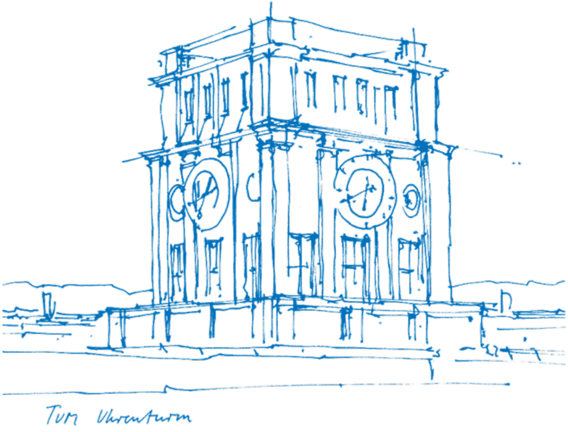
Grundlagenpraktikum Rechnerarchitektur

Gruppe 108 – Vortrag zu Aufgabe A329

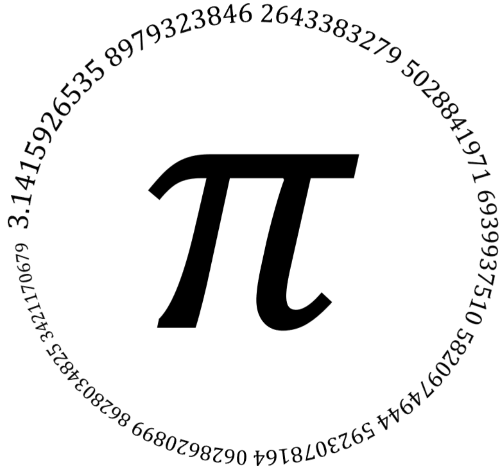
Berechnung der Konstante π

Cara Dickmann **Thua Duc Nguyen**
Eslam Nasrallah

Wintersemester 2022



Outline


3.1415926535 8979323846 2643383279 5028841971 6939937510 5820974944 5923078164 0628620899 8628034825 3421170679

Outline

Fließkommazahlen nach IEEE-754

nicht geeignet, warum?

- begrenzte Anzahl von Bits zur Darstellung des Exponenten und der Signifikanten
- verliert Genauigkeit bei der Berechnung von großen Zahlen
- Rundungsfehler können auftreten

Aufbau von Bignum

- Array von 32 Bit unsigned Integern
- Anzahl der belegten Blöcke
- Anzahl der Nachkommablöcke

```
struct bignum
{
    uint32_t *numbers;
    size_t length;
    size_t subone;
};
```

Aufbau von Bignum (Little Endian)

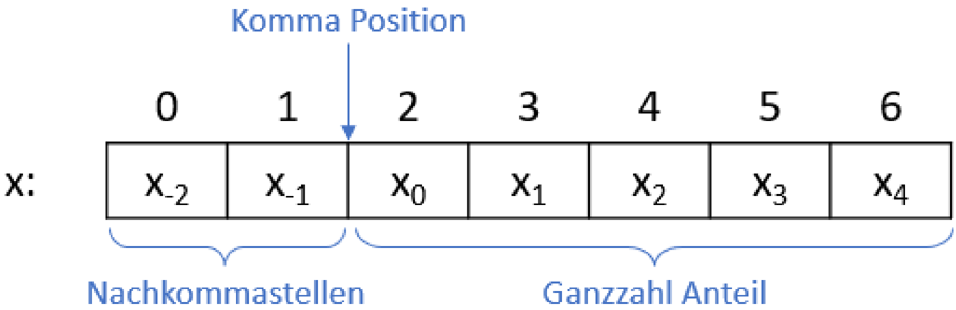
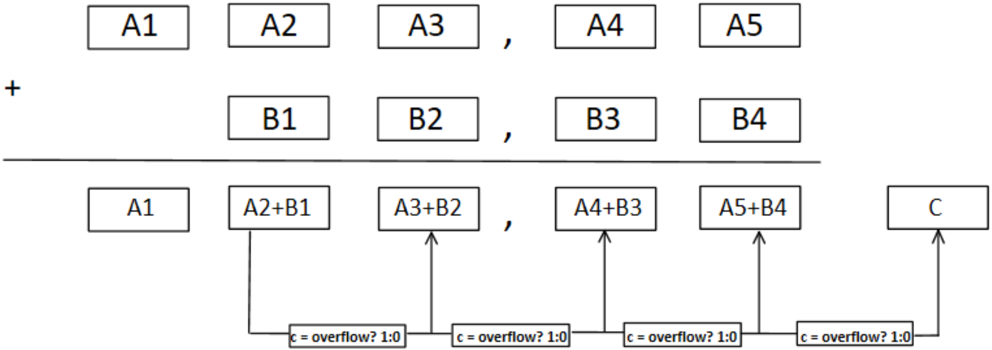


Abbildung 1 Bsp. für bignum Array

Addition / Subtraktion



Karazuba Multiplikation (1)

1. Zu multiplizierenden Zahlen in zwei Hälften teilen
2. Produkte der Hälften rekursiv zu berechnen
3. Ergebnisse durch Addition und Subtraktion zu kombinieren

Karazuba Multiplikation (2)
Bignums halbieren

- Teilen der Bignums in 2 Hälften der Länge $m = \lceil \frac{n}{2} \rceil$:

  $$x \cdot y = (x_h \cdot B^m + x_l) \cdot (y_h \cdot B^m + y_l)$$

- Füllen der Lücken mit 0

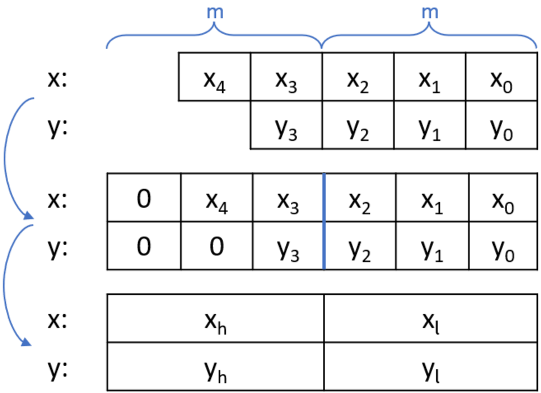


Abbildung 2 Bsp. für bignum base-splitten

Karazuba Multiplikation (3)
Karazuba Formel

- Normale Multiplikation: 4 Multiplikationen, $\Theta(n^2)$

$$x \cdot y = \boxed{x_l y_l} + B^m \cdot (\boxed{x_l y_h} + \boxed{x_h y_l}) + B^{2m} \cdot \boxed{x_h y_h}$$

- Karazuba Multiplikation: 3 Multiplikationen, $O(n^{1{,}59})$

$$x \cdot y = \boxed{x_l y_l} + B^m \cdot (\boxed{(x_l + x_h)(y_l + y_h)} - \boxed{x_l y_l} - \boxed{x_h y_h}) + B^{2m} \cdot \boxed{x_h y_h}$$

- Multiplikation mit Vielfachen der Basis B $\Rightarrow$ Mit Links-Shift ersetzbar

Newton-Raphson Division (1)

Newton-Raphson Division ermitteln den Kehrwert vom Zähler und multipliziert diesen Kehrwert mit dem Nenner N.

1. $N' = nomalize(N)$: Nenner N wird zwischen 0,5 und 1 normalisiert
2. $Z' = nomalize(Z)$: Gleich viele Shifts auf den Zähler anwenden
3. $\frac{1}{N'} \approx X = \frac{48}{17} - \frac{32}{17} \cdot N'$: Kehrwert approximieren
4. $I = \left\lceil \frac{\log_2(P+1)}{\log_2 17} \right\rceil$: Anzahl der Iterationen, um eine Genauigkeit von P Bit sicherzustellen
5. $X \cdot (2 - N' \cdot X)$: sukzessive genauere Schätzungen I Mal durchführen
6. $Q = X \cdot Z'$: Multiplikation ergibt das Endergebnis

Newton-Raphson Division (2)

Die Approximations-Funktion ergibt das Ergebnis mit Fehler ϵ:

$$|\epsilon| = |1 - N' \cdot (\frac{48}{17} - \frac{32}{17} \cdot N')|$$

Der maximale Fehlerwert beträgt:

$$|\epsilon| \leq \frac{1}{17} \qquad \forall N' \in (0.5, 1)$$

Iterationsschritte um eine Genauigkeit von P Bit sicherzustellen:

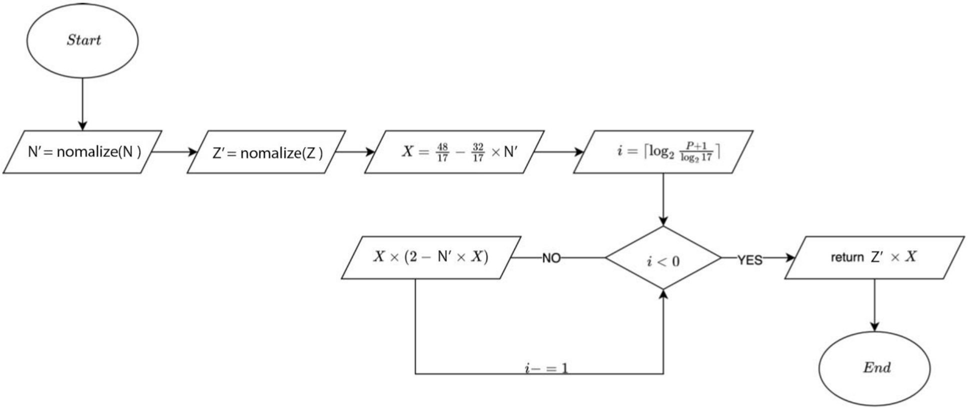
$$\lceil \log_2 \frac{P+1}{\log_2 17} \rceil$$

Newton-Raphson Division (3)

Newton-Raphson Division ermitteln den Kehrwert vom Zähler und multipliziert diesen Kehrwert mit dem Nenner N.

1. $N' = nomalize(N)$: Nenner N wird zwischen 0,5 und 1 normalisiert
2. $Z' = nomalize(Z)$: Gleich viele Shifts auf den Zähler anwenden
3. $\frac{1}{N'} \approx X = \frac{48}{17} - \frac{32}{17} \cdot N'$: Kehrwert approximieren
4. $I = \left\lceil \frac{\log_2(P+1)}{\log_2 17} \right\rceil$: Anzahl der Iterationen, um eine Genauigkeit von P Bit sicherzustellen
5. $X \cdot (2 - N' \cdot X)$: sukzessive genauere Schätzungen I Mal durchführen
6. $Q = X \cdot Z'$: Multiplikation ergibt das Endergebnis

Newton-Raphson Division (4)



Outline

Binary Splitting (1)

- Die Formel:
  Sei S_{n_1,n_2} für $n_1, n_2 \in N$ mit $n_1 < n_2$ eine Folge der folgenden Form und seien $a(n), b(n), p(n)$ und $q(n)$ Polynome mit ganzzahligen Koeffizienten:

$$S_{n=n1,n2} = \sum_{n1}^{n2-1} \frac{a(n)}{b(n)} \cdot \prod_{k=n1}^{n} \frac{p(k)}{q(k)} = \frac{T_{n1,n2}}{B_{n1,n2}Q_{n1,n2}}$$

- Annäherung der Konstante π:

$$\pi = 2 + \sum_{n=1}^{\infty} \frac{a(n)}{b(n)} \cdot \prod_{k=1}^{n} \frac{p(k)}{q(k)}$$

mit $a(n) = 2$, $b(n) = 1$, $p(k) = n$, $q(k) = 2n + 1$

Binary Splitting (2)

Zur Berechnung von $S_{n=n1,n2}$ werden folgende Hilfsterme rekursiv definiert

$$P_{n1,n2} = p(n1)...p(n2-1)$$

$$Q_{n1,n2} = q(n1)...q(n2-1)$$

$$B_{n1,n2} = b(n1)...b(n2-1)$$

$$T_{n1,n2} = P_{n1,n2}Q_{n1,n2}B_{n1,n2}$$

Daraus folgt, dass:

$$\pi = 2 + \frac{T(1,n)}{B(1,n)Q(1,n)} = 2 + \frac{T(1,n)}{Q(1,n)}$$

Outline

Ergebnisse (von 1000 bis 100000)

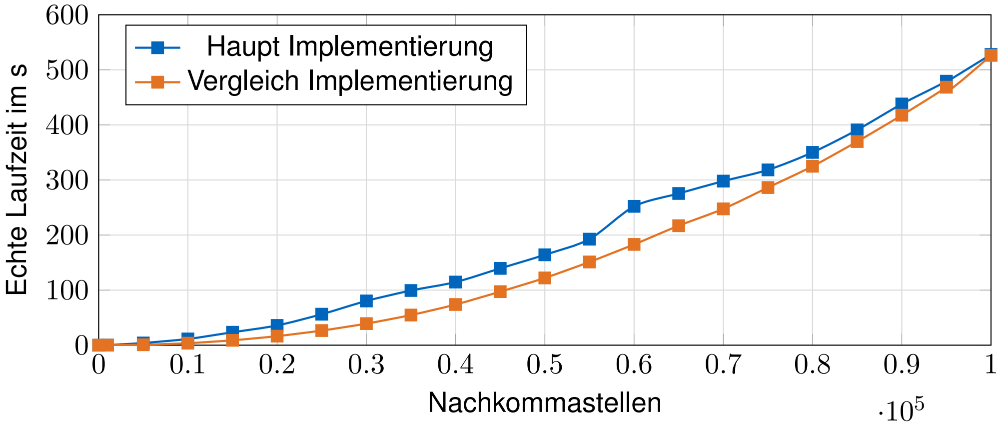


Abbildung 3 Durchschnittliche Laufzeit bei zehn Wiederholungen auf Intel i7-1165G7

Ergebnisse (von 100000 bis 500000)

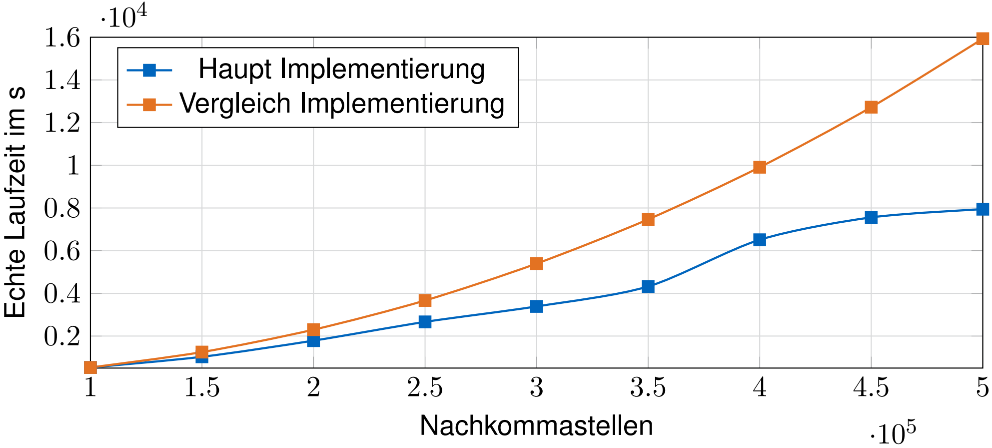


Abbildung 4 Durchschnittliche Laufzeit bei drei Wiederholungen auf Intel i7-1165G7

Outline

Zusammenfassung

- Bignum-Struktur 32 Bit Basis
- Arithmetik: Addition/Subtraktion, Karazuba, Newton Raphson
- Binary Splitting
- Ergebnisse

Ausblick– weitere Optimierungsideen

- Verwendung von 64 Bit Basis
- Verwendung von SIMD

Vielen Dank für Ihre Aufmerksamkeit!