Implementierung des Sinh

in x86-Assembly

Kevin Holm, Deniz Candas, Jakob Mezger

09.08.2017

1. Aufgabe
2. Verwendete Umgebung
3. Verwendeter Ansatz
4. Code Review
   - Pseudo Code
   - Markante Stellen im Code

Aufgabe

- Implementierung des Sinus Hyperbolicus
- Erlaubte Befehle
  - x87 FPU-Befehle für Grundrechenarten, Negation
  - Speicherverwaltungsbefehle
- C-Rahmenprogramm
  - Validierung
  - Leistungsmessung
  - Vergleich mit Funktion der Standardbibliothek

Verwendete Umgebung

- Betriebssystem: Linux Ubuntu 64-Bit LTS 16.04
- Assembly-Syntax: nasm

Verwendeter Ansatz

Und dessen Vor- und Nachteile

- Reihenentwicklung: $sinh(x) = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!}$
- Gesteigerte Effizienz durch Rekursion:
  $term_0 = x,\ term_{n+1}(x) = term_n(x) \times \frac{x}{n+1} \times \frac{x}{n+2}$

Vorteile	**Nachteile**
Beliebige Genauigkeit	Viele Schleifendurchläufe für genaue Werte
Rekursion $\rightarrow$ Einfache Implementierung	Rekursion $\rightarrow$ Große Stack-Auslastung
Wenige Speicherzugriffe	Rechenintensiv

Code Review

Pseudo Code

Sinh in Pseudocode

```
1   sinh(double x):
2   double i = 1
3   double result = x
4   double term = x
5
6   loop:
7   i = i + 1;
8   term = term / i
9   term = term * x
10
11  i = i + 1;
12  term = term / i
13  term = term * x
14
15  result = result + term;
16  if result is not precise enough jmp loop
17
18  return result
```

$$term_0 = x,\ term_{n+1}(x) = term_n(x) \times \frac{x}{n+1} \times \frac{x}{n+2}$$

Code Review

Markante Stellen im Code

sinh.asm vor Aufruf der Hauptschleife

```
22  fld  st0                    ;st0 = x, st1 = x
23
24  fabs                        ;st0 = |x|
25
26  fdiv st1, st0               ;st1 = x/|x|, either 1 or -1
```

$\forall x \in \mathbb{R} : sinh(-x) = -sinh(x)$. Also kann man den sinh nur für positive Werte berechnen...

sinh.asm nach Verlassen der Hauptschleife

```
82  fincstp
83  fincstp                     ;st6= previous, st7 = i, st0 = result,
84                              ;st1 = 1, st2 = x, st3 = sign
85  fmul st0, st3               ;st0 = result * sign
```

...und nach der Berechnung das Vorzeichen anpassen.