LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Praktikum Rechnerarchitektur

Numerische Quadratur (A305)

Projektaufgabe – Aufgabenbereich Algorithmik

1 Organisatorisches

Auf den folgenden Seiten finden Sie die Aufgabenstellung zu Ihrer Projektaufgabe für das Praktikum. Die Rahmenbedingungen für die Bearbeitung werden in der Praktikumsordnung festgesetzt, die Sie über die Praktikumshomepage [1] aufrufen können.

Wie in der Praktikumsordnung beschrieben, sind die Aufgaben relativ offen gestellt. Besprechen Sie diese innerhalb Ihrer Gruppe und konkretisieren Sie die Aufgabenstellung. Die Teile der Aufgabe, in denen Assembler-Code anzufertigen ist, sind für die 64-Bit x86-Architektur (x86-64) unter Verwendung der SSE/SSE2-Erweiterungen zu schreiben.

Der **Abgabetermin** ist der **3. Februar 2019, 23:59 Uhr (MEZ)**. Die Abgabe erfolgt per Git in für Ihre Gruppe eingerichtete Projektrepository. Bitte beachten Sie die in der Praktikumsordnung angebene Liste von abzugebenden Dateien.

Der **erste Teil Ihrer Projektpräsentation** ist eine kurze Vorstellung Ihrer Aufgabe im Umfang von ca. 2 Minuten in einer Tutorübung in der Woche **10.12.2018 – 14.12.2018**. Erscheinen Sie bitte **mit allen Team-Mitgliedern** und wählen Sie bitte eine Übung, in der mindestens ein Team-Mitglied angemeldet ist.

Die **Abschlusspräsentationen** finden in der Woche vom **25.02.2019 – 01.03.2019** statt. Weitere Informationen zum Ablauf und die Zuteilung der einzelnen Präsentationstermine werden noch bekannt gegeben. Beachten Sie, dass die Folien für die Präsentation am obigen Abgabetermin im PDF-Format abzugeben sind.

Bei Fragen/Unklarheiten in Bezug auf den Ablauf und die Aufgabenstellung wenden Sie sich bitte an Ihren Tutor.

Wir wünschen Ihnen viel Erfolg und Freude bei der Bearbeitung Ihrer Aufgabe!

Mit freundlichen Grüßen
Die Praktikumsleitung

PS: Vergessen Sie nicht, sich rechtzeitig in TUMonline zur Prüfung anzumelden. Dies ist Voraussetzung für eine erfolgreiche Teilnahme am Praktikum im laufenden Semester.

[1] `https://www.caps.in.tum.de/lehre/ws18/praktika/rechnerarchitektur/`

2 Numerische Quadratur

2.1 Überblick

Die Algorithmik ist ein Teilgebiet der Theoretischen Informatik, welches wir hier unter verschiedenen praktischen Aspekten beleuchten: Meist geht es um eine konkrete Frage- oder Problemstellung, welche durch mathematische Methoden beantwortet oder gelöst werden kann. Sie werden im Zuge Ihrer Projektaufgabe ein Problem lösen und die Güte Ihrer Lösung wissenschaftlich bewerten.

2.2 Funktionsweise

Als Numerische Quadratur bezeichnet man den Vorgang des Annäherns von Integralen durch Summation von Flächen. Wir betrachten dazu die Familie der Newton-Cotes Formeln, welche zur Quadratur von Polynomen verwendet werden.
Sei $p(x)$ ein zu integrierendes Polynom. Wir suchen ein bestimmtes Integral $I_n[f]$ mit Grenzen a und b, für welches nach Newton-Cotes gilt:

$$I_n[f] = \int_a^b p(x)\,\mathrm{d}x = (b-a) \cdot \sum_{i=0}^{n} \alpha_{in} \cdot p(x_i) \tag{1}$$

Dabei bezeichne x_i die i-te von $n+1$ Stützstellen, mit Formel

$$x_i = a + \frac{i \cdot (b-a)}{n}\,. \tag{2}$$

Für die Gewichte α_{in} gilt wiederum:

$$\alpha_{in} = \frac{1}{n} \int_0^n \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x-j}{i-j}\,\mathrm{d}x \tag{3}$$

In allen Formeln steht der Buchstabe n für die n-te Entwicklung der Newton-Cotes Formel. In Ihrer Projektaufgaben werden Sie ein explizites n wählen.

2.3 Aufgabenstellungen

Ihre Aufgaben lassen sich in die Bereiche Konzeption (theoretisch) und Implementierung (praktisch) aufteilen. Sie können (müssen aber nicht) dies bei der Verteilung der Aufgaben innerhalb Ihrer Arbeitsgruppe ausnutzen. Alle Antworten auf konzeptionelle Fragen sollten in Ihrer Ausarbeitung erscheinen. Besprechen Sie nach eigenem Ermessen außerdem im Zuge Ihres Vortrags einige der konzeptionellen Fragen. Die Antworten auf die Implementierungsaufgaben werden durch Ihrem Code reflektiert.

Wichtig: Sie dürfen im Assembler-Code nur grundlegende arithmetische Operationen (im Zweifel: die vier Grundrechenarten) verwenden, nicht jedoch Instruktionen, die komplexere Berechnungen durchführen (z.B. Wurzel, Logarithmus, Exponentiation).

2.3.1 Theoretischer Teil

- Stellen Sie die Newton-Cotes Formel für $n = 1$ und $n = 2$ auf. Vereinfachen Sie das Ergebnis so weit, bis Sie eine Formel für $I_1[f]$ bzw. $I_2[f]$ erhalten, welche nur noch die vier Grundrechenarten, die Integrationsgrenzen a und b und die Funktionswerte des Polynoms p (die Stützstellen) enthält.
- Schlagen Sie in einem C-Standardwerk nach, wie in C korrekt mit Funktionspointern umgegangen wird. Die zu integrierende Funktion f soll der Benutzer dem fertigen Programm als dynamische Bibliothek übergeben können. Ihr Rahmenprogramm führt die benötigten Operationen zum Laden der Bibliothek in C durch und übergibt dann einen Zeiger auf die Funktion an Ihre Implementierung, welche die übergebene Funktion dann aufruft.
- Untersuchen Sie die Güte der Näherung, die Sie mit Ihrer Implementierung für verschiedene Polynome erreichen. Stellen Sie Ihre Erkenntnisse grafisch dar.
- Gibt es Security-Bedenken bezüglich des angestrebten Programmdesigns?

2.3.2 Praktischer Teil

- Implementieren Sie in der Datei mit dem Assemblercode eine Funktion

  ```
  float *newton_cotes_1(float ( *p)(float), int a, int b)
  ```

  welche vom Benutzer spezifizierbare Integrationsgrenzen a und b sowie die zu integrierende Funktion p übergeben bekommt und das Ergebnis der numerischen Quadratur mithilfe von Formel $I_1[f]$ berechnet und als Fließkommazahl zurückgibt.
- Implementieren Sie in der Datei mit dem Assemblercode eine Funktion

  ```
  double *newton_cotes_2(double ( *p)(double), int a, int b)
  ```

  welche vom Benutzer spezifizierbare Integrationsgrenzen a und b, sowie die zu integrierende Funktion p übergeben bekommt und das Ergebnis der numerischen Quadratur mithilfe von Formel $I_2[f]$ berechnet und als Fließkommazahl zurückgibt.
- Beachten Sie, dass der Benutzer das zu integrierende Polynom p frei wählen können soll. Daher wurde f als Funktionspointer deklariert. Das Programm soll vom Benutzer eine Bibliothek mit der Implementierung einer bestimmten Funktion übergeben bekommen, welche das Programm zur Laufzeit ausführen soll. (Sie finden Beispieldateien, welche auf `.so` enden, anbei.) Die Funktion $p(x)$ entspricht einer Funktion

  ```
  double f(double x)
  ```

welche ein x (eine Stützstelle) übergeben bekommt und ein $y = f(x)$ zurückliefert. Sie müssen die Funktion von Ihrem Assemblerprogramm aus aufrufen, um den Funktionswert an den Stützstellen zu bestimmen. Denken Sie hierbei an die Calling Conventions. Wie werden letztere auf Fließkommazahlen erweitert?

2.4 Checkliste

Die folgende Liste soll Ihnen als Gedächtnisstütze beim Bearbeiten der Aufgaben dienen. Beachten Sie ebenfalls die in der Praktikumsordnung angegebenen Hinweise. Sollten Sie eine Reverse-Engineering-Aufgabe erhalten haben, sind diese Punkte für Sie weitestgehend hinfällig.

- Verwenden Sie keinen Inline-Assembler.
- I/O-Operationen dürfen grundsätzlich in C implementiert werden.
- Sie dürfen die Signatur der in Assembler zu implementierenden Funktion nur dann ändern, wenn Sie dies (in Ihrer Ausarbeitung) rechtfertigen können.
- Denken Sie daran, das Laufzeitverhalten Ihres Codes zu testen (Sichere Programmierung, Performanz) und behandeln Sie alle möglichen Eingaben, auch Randfälle. Ziehen Sie ggf. alternative Implementierungen als Vergleich heran.
- Verwenden Sie SIMD-Befehle, wenn möglich.
- Fügen Sie Ihrem Projekt ein funktionierendes `Makefile` hinzu, welches durch den Aufruf von `make` Ihr Projekt kompiliert.
- Eingabedateien, welche Sie generieren, um Ihre Implementierungen zu testen, sollten mit abgegeben werden.
- Verwenden für die Ausarbeitung Sie die bereitgestellte TEX-Vorlage und legen Sie sowohl die PDF-Datei als auch sämtliche TEX-Quellen in das Repository.
- Stellen Sie Performanz-Ergebnisse grafisch oder tabellarisch dar.
- Vermeiden Sie unscharfe Grafiken und Screenshots von Code.
- Bonusaufgaben (sofern vorhanden) müssen nicht implementiert werden.
- Geben Sie die Folien für Ihre Abschlusspräsentation im PDF-Format ab. Achten Sie auf hinreichenden Kontrast (schwarzer Text auf weißem Grund!) und eine angemessene Schriftgröße. Verwenden Sie 4:3 als Folien-Format.