LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Praktikum Rechnerarchitektur

Tricorn-Fraktal (A212)

Projektaufgabe – Aufgabenbereich Bildverarbeitung

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-Erweiterungen zu schreiben.

Der **Abgabetermin** ist der **29. Januar 2020, 11:59 Uhr (MEZ)**. Die Abgabe erfolgt per Git in das 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 **09.12.2019 – 13.12.2019**. 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 Zeit vom **24.02.2020 – 28.02.2020** 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/ws19/praktika/era/`

2 Tricorn-Fraktal

2.1 Überblick

Im Zuge Ihrer Projektaufgabe werden Sie theoretisches Wissen aus der Mathematik im Anwendungszusammenhang verwenden, um einen Algorithmus in Assembler zu implementieren. Sie konzentrieren sich dabei auf das Feld des *Image Processing*, in welchem Pixelbilder, wie sie typischerweise Digitalkameras produzieren, als Eingabe für bestimmte Algorithmen verwendet werden und mathematische Überlegungen dadurch sichtbar gemacht werden.

2.2 Funktionsweise

In dieser Aufgabe befassen Sie sich mit dem Multicorn-Fraktal. Diese ist gegeben durch iterative Anwendung der komplexen Gleichung

$$z_{i+1} = \overline{z_i}^m + c \; (i \geq 0)$$

und Betrachtung des Wertes z_i innerhalb der komplexen Ebene. Dabei wählt man $z_0 = 0$ und Real- sowie Imaginärteil von c gemäß der Koordinaten in der komplexen Ebene. Durch Variation von a und b in $c = (a + bi)$ für $a \in [-2; 1]$ und $b \in [-1; 1]$ und Betrachtung von z_n für große n lässt sich so das Fraktal in der komplexen Ebene darstellen. Im Rahmen der Aufgabe betrachten wir nur den Fall $m = 2$. Der Term $\overline{z_i}$ beschreibt die komplex konjugierte Zahl.

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. Antworten auf konzeptionelle Fragen sollten an den passenden Stellen in Ihrer Ausarbeitung in angemessenem Umfang erscheinen. Entscheiden Sie nach eigenem Ermessen, ob Sie im Rahmen Ihres Abschlussvortrags auch auf konzeptionelle Fragen eingehen. Die Antworten auf die Implementierungsaufgaben werden durch Ihrem Code reflektiert.

2.3.1 Theoretischer Teil

- Berechnen Sie Real- sowie Imaginärteil von z_{i+1} in Abhängigkeit von Real- sowie Imaginärteil von z_i explizit für $m = 2$. Dies ist die Iterationsvorschrift, die Ihr Algorithmus verwenden soll.
- Sofern z_i beschränkt ist, färben Sie den Pixel, der zu Real- sowie Imaginärteil von c korrespondiert schwarz, da er zur Menge des Fraktals gehört. Wie verfahren Sie in dieser Hinsicht mit Pixeln, die zu Koordinaten korrespondieren, für die z_i divergiert?

- Der Algorithmus lässt sich leicht parallelisieren. Entwerfen Sie einen parallelen Algorithmus, der die SIMD-Einheit Ihrer Ziel-Architektur verwendet.

2.3.2 Praktischer Teil

- Implementieren Sie in der Datei mit dem Assemblercode eine Funktion

```
void multicorn(float r_start, float r_end, float i_start, float
            i_end, float res, unsigned char* img)
```

  welche den Ausschnitt der MulticornMenge in der komplexen Ebene, der auf der reellen Achse durch `r_start` und `r_end` und auf der imaginären Achse durch `i_start` sowie `i_end` gekennzeichnet wird, übergeben bekommt. Als weitere Parameter erhält die Funktion die gewünschte Auflösung (Schrittweite pro Bildpixel) `res` sowie einen Zeiger auf einen Speicherbereich der groß genug ist, um die berechneten Bitmap-Daten des Ergebnisses zu speichern `img`. Legen Sie Ihrer Ausgabe eine sinnvolle Farbpalette zugrunde.

- Wie Sie wissen, dürfen Sie alle I/O-Operationen im Rahmenprogramm in C durchführen. Wir schreiben Ihnen kein Ausgabeformat vor, legen Ihnen aber nahe, mit unkomprimierten, 24-bit BMP-Dateien zu arbeiten. Das Schreiben des Headers und der Daten kann daher in C erfolgen.

2.4 Allgemeine Bewertungshinweise

Die folgende Liste soll Ihnen als Gedächtnisstütze beim Bearbeiten der Aufgaben dienen. Beachten Sie ebenfalls die in der Praktikumsordnung angegebenen Hinweise.

- Stellen Sie unbedingt sicher, dass Ihre Abgabe auf der Referenzplattform des Praktikums (`lxhalle`) kompiliert und funktionsfähig ist.
- Fügen Sie Ihrem Projekt ein funktionierendes `Makefile` hinzu, welches durch den Aufruf von `make` Ihr Projekt kompiliert.
- Verwenden Sie keinen Inline-Assembler.
- Verwenden Sie SIMD-Befehle, wenn möglich.
- Verwenden Sie keine x87-FPU- oder MMX-Instruktionen. Sie dürfen alle SSE-Erweiterungen bis SSE4.2 benutzen. AVX-Instruktionen dürfen Sie benutzen, sofern Ihre Implementierung auch auf Prozessoren ohne AVX-Erweiterungen lauffähig ist.
- Sie dürfen die Signatur der in Assembler zu implementierenden Funktion nur dann ändern, wenn Sie dies (in Ihrer Ausarbeitung) rechtfertigen können.
- I/O-Operationen dürfen grundsätzlich in C implementiert werden.

- 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.
- Eingabedateien, welche Sie generieren, um Ihre Implementierungen zu testen, sollten mit abgegeben werden.
- Verwenden Sie für die Ausarbeitung die bereitgestellte TEX-Vorlage und legen Sie sowohl die PDF-Datei als auch sämtliche TEX-Quellen in das Repository.
- Stellen Sie Performanz-Ergebnisse nach Möglichkeit grafisch dar.
- Vermeiden Sie unscharfe Grafiken und Screenshots von Code.
- 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.
- Zusatzaufgaben (sofern vorhanden) müssen nicht implementiert werden. Es gibt keine Bonuspunkte.