LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Grundlagenpraktikum: Rechnerarchitektur

Das Geburtstagsparadox (A301)
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-Erweiterungen zu schreiben; alle anderen Bestandteile der Hauptimplementierung sind in C nach dem C11-Standard anzufertigen.

Der **Abgabetermin** ist der **07. Februar 2022, 11:59 Uhr (MEZ)**. Die Abgabe erfolgt per Git in das für Ihre Gruppe eingerichtete Projektrepository. Bitte beachten Sie die in der `README.md` angegebene Liste von abzugebenden Dateien.

Die **Abschlusspräsentationen** finden in der Zeit vom **07.03.2022 – 11.03.2022** statt. Weitere Informationen werden noch bekannt gegeben. Beachten Sie, dass die Folien für die Präsentation am obigen Abgabetermin im PDF-Format abzugeben sind.

Sofern die Rahmenbedingungen (Hygiene-, Abstandsregeln etc.) für eine Präsenzprüfung nicht vorliegen, kann gemäß §13a APSO die geplante Prüfungsform auf eine virtuelle Präsentation (Videokonferenz) oder eine kurze schriftliche Fernprüfung umgestellt werden. Die Entscheidung über diesen Wechsel wird möglichst zeitnah, spätestens jedoch 14 Tage vor dem Prüfungstermin nach Abstimmung mit dem zuständigen Prüfungsausschuss bekannt gegeben.

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

[1] `https://gra.caps.in.tum.de`

2 Das Geburtstagsparadox

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

Das Geburtstagsparadox beschäftigt sich mit der Frage, wie viele Leute mindestens in einem Raum sein müssen, sodass mindestens zwei der Anwesenden mit einer Wahrscheinlichkeit von mindestens $\frac{1}{2}$ am gleichen Tag Geburtstag haben. Man legt dabei eine Gleichverteilung der Geburtstage über das Jahr zugrunde.
Weitet man die Frage aus, wie viele Leute k in einem Raum sein müssen, um mit Wahrscheinlichkeit $\frac{1}{2}$ mindestens zwei Leute zu finden, welche das gleiche Element einer Grundmenge M mit n Elementen teilen, so lässt sich die Zahl der Personen k berechnen als:

$$k \geq \frac{1 + \sqrt{1 + 8n \cdot \ln 2}}{2} \tag{1}$$

Man beachte, dass $k \approx 23$ für $n = 365$! (Intuitiv würde man den Wert k für die Wahrscheinlichkeit zweier Geburtstage an einem Tag weit höher schätzen.)

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.

Wichtig: Sie dürfen in Ihrer C-Implementierung nur solche arithmetischen Operationen verwenden, die grundlegende Berechnungen durchführen (im Zweifel: die vier Grundrechenarten), nicht jedoch Instruktionen, die komplexere Berechnungen durchführen (z.B. Wurzel, Logarithmus, Exponentiation).

2.3.1 Theoretischer Teil

- Leiten Sie mathematisch zwei Möglichkeiten her, die Wurzelfunktion in guter Näherung zu berechnen. Verwenden Sie dazu sowohl eine reine Reihendarstellung

als auch eine Methodik, die auch einen Tabellen-Lookup benutzt. Warum müssen Sie *keine* Näherung für den Logarithmus berechnen?

- Untersuchen Sie Ihre fertigen Implementierungen auf Genauigkeit und Performanz. Vergleichen Sie Ihre beiden Verfahren und ziehen Sie als weiteren Vergleich eine C-Implementierung unter Nutzung von Instruktionen, die eine Wurzelberechnung durchführen, heran.
- Beschreiben Sie, wie die Berechnung der Wurzelfunktion in einer aktuellen Version der C-Standardbibliothek *glibc* realisiert ist.
- Welcher Wert ergibt sich für k für die Situation, dass mindestens zwei Leute mit einer Wahrscheinlichkeit von mindestens $\frac{1}{2}$ die gleiche PIN für ihr Girokonto verwenden? Wie viele MD5-Hashes muss man berechnen, um im Mittel mit mindestens fünfzig Prozent Wahrscheinlichkeit eine beliebige Kollision zu finden? Erstellen Sie geeignete Diagramme für die Entwicklung von k in Abhängigkeit von n.

2.3.2 Praktischer Teil

- Implementieren Sie im Rahmenprogramm I/O-Operationen in C, mit welchen Sie vom Benutzer einen Wert für die Größe der Grundmenge abfragen und ihn anschließend an Ihre C-Funktion übergeben können.
- Implementieren Sie in der Datei mit Ihrem C-Code die Funktion:

```
float birthday_eq(unsigned long n)
```

  Die Funktion implementiert die Formel 1 und gibt für ein bestimmtes `n` den Wert k zurück.

2.3.3 Rahmenprogramm

Ihr Rahmenprogramm muss bei einem Aufruf die folgenden Optionen entgegennehmen und verarbeiten können. Wenn möglich soll das Programm sinnvolle Standardwerte definieren, sodass nicht immer alle Optionen gesetzt werden müssen.

- `-V<int>` — Die Implementierung, die verwendet werden soll. Hierbei soll mit `-V0` Ihre Hauptimplementierung verwendet werden. Wenn diese Option nicht gesetzt wird, soll ebenfalls die Hauptimplementierung ausgeführt werden.
- `-B<int>` — Falls gesetzt, wird die Laufzeit der angegebenen Implementierung gemessen und ausgegeben. Das optionale Argument dieser Option gibt die Anzahl an Wiederholungen des Funktionsaufrufs an.
- `-n<int>` — Größe der Grundmenge

- `-h` — Eine Beschreibung aller Optionen des Programms und Verwendungsbeispiele werden ausgegeben und das Programm danach beendet.
- `--help` — Eine Beschreibung aller Optionen des Programms und Verwendungsbeispiele werden ausgegeben und das Programm danach beendet.

Sie dürfen weitere Optionen implementieren, beispielsweise um vordefinierte Testfälle zu verwenden. Ihr Programm muss jedoch nur unter Verwendung der oben genannten Optionen verwendbar sein. Beachten Sie ebenfalls, dass Ihr Rahmenprogramm etwaige Randfälle korrekt abfangen muss und im Falle eines Fehlers mit einer aussagekräftigen Fehlermeldung auf `stderr` und einer kurzen Erläuterung zur Benutzung terminieren sollte.

2.4 Allgemeine Bewertungshinweise

Beachten Sie grundsätzlich alle in der Praktikumsordnung angegebenen Hinweise. Die folgende Liste konkretisiert einige der Bewertungspunkte:

- Stellen Sie unbedingt sicher, dass *sowohl* Ihre Implementierung *als auch* Ihre Ausarbeitung auf der Referenzplattform des Praktikums (`lxhalle`) kompilieren und vollständig korrekt bzw. funktionsfähig sind.
- Die Implementierung soll mit GCC/GNU as kompilieren. Achten Sie darauf, dass Ihr Programm keine x87-FPU- oder MMX-Instruktionen und SSE-Erweiterungen nur bis SSE4.2 verwendet. Andere ISA-Erweiterungen (z.B. AVX, BMI1) dürfen Sie nur benutzen, sofern Ihre Implementierung auch auf Prozessoren ohne derartige Erweiterungen lauffähig ist.
- Sie dürfen die angegebenen Funktionssignaturen (nur dann) ändern, wenn Sie dies (in Ihrer Ausarbeitung) begründen.
- Verwenden Sie die angegebenen Funktionsnamen für Ihre Hauptimplementierung. Falls Sie mehrere Implementierungen schreiben, legen wir Ihnen nahe, für die Benennung der alternativen Implementierungen mit dem Suffix „`_V1`“, „`_V2`“ etc. zu arbeiten.
- 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; größere Eingaben sollten stattdessen stark komprimiert oder (bevorzugt) über ein abgegebenes Skript generierbar sein.
- 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 16:9 als Folien-Format.