LEHRSTUHL FÜR RECHNERTECHNIK UND RECHNERORGANISATION

Aspekte der systemnahen Programmierung bei der Spieleentwicklung

Projektaufgabe – Aufgabenbereich Algorithmik (A323)

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 [1] festgesetzt, die Sie auch über die Praktikumshomepage [2] aufrufen können.

Der **Abgabetermin** ist der **4. Februar 2018 (23:59 MEZ)**. Die Abgabe erfolgt per Git in das im Gitlab für Ihre Gruppe eingerichtete Projektrepository. Bitte beachten Sie unbedingt die in der Praktikumsordnung angebene Liste von abzugebenden Dateien, **insbesondere** Ihre Folien für die Abschlusspräsentation!

Wie in der Praktikumsordnung beschrieben, sind die Aufgaben relativ offen gestellt. Besprechen Sie diese innerhalb Ihrer Gruppe und konkretisieren Sie die Aufgabenstellung. **Der erste Teil Ihrer Projektpräsentation** ist eine kurze Vorstellung Ihrer Aufgabe in einer Tutorübung in der Woche **11.12.2017 – 15.12.2017**. Wählen Sie bitte eine Übung, in der mindestens ein Team-Mitglied angemeldet ist.
Erscheinen Sie bitte **mit allen Team-Mitgliedern** und bereiten Sie einen Kurzvortrag mit folgenden Inhalten vor:

- 1 Folie: Vorstellung der Team-Mitglieder
- 2 Folien: Zusammenfassung der Aufgabenstellung

In der Übung wird eine Beamer vorhanden sein, an den Sie Ihr eigenes Notebook anschließen können. Alternativ können Sie auch Ihre Präsentation als PDF-Datei auf einem USB-Stick mitbringen. Der Kurzvortrag wird von einer Person gehalten.

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

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] `http://wwwi10.lrr.in.tum.de/~trinitic/GEP-ASP/Praktikumsordnung.pdf`
[2] `http://www.lrr.in.tum.de/lehre/`

2 Berechnen von Primzahlen

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

Primzahlen sind Zahlen, die nur von durch eins und durch sich selbst geteilt werden. Zur Berechnung der n-ten Primzahl p_n lässt sich die Wormell'sche Formel verwenden:

$$p_n = 2 + \frac{3}{2} + 2^{n-1} - \frac{1}{2}\sum_{m=2}^{2^n}(-1)^{2^{\prod_{r=1}^{n}\left(1-r+\frac{m-1}{2}+\frac{1}{2}\sum_{x=2}^{m}(-1)^{2^{\prod_{a=2}^{x}\prod_{b=2}^{x}(x-ab)^2}}\right)^2}}$$

Die Implementierung der Wormell'schen Formel zur Berechnung der n-ten Primzahl ist Ihre Projektaufgabe.

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 Assemblercode ausschließlich auf die vier Grundrechenarten zurückgreifen.

2.3.1 Theoretischer Teil

- Die Wormell'sche Formel ist in Ihrer Darstellung optimiert, um nur analytische Ausdrücke zu enthalten. Analysieren Sie die Teilkomponenten und brechen Sie die Formel in mehrere kleine Funktionen auf, wenn nötig. Sie dürfen Vereinfachungen und Umschreibungen vornehmen, sofern Sie den Wert des Terms nicht verändern.

2.3.2 Praktischer Teil

- Implementieren Sie in der Datei mit dem Assemblercode eine Funktion

```
unsigned int p(unsigned int n)
```

welche ein `n` entgegen nimmt und mithilfe der Wormell'schen Formel die `n`-te Primzahl berechnet. Die Funktion muss nur für Primzahlen $< 2^{32}$ korrekte Ergebnisse liefern.

2.4 Checkliste

Die folgende Liste soll Ihnen als Gedächtnisstütze beim Bearbeiten der Aufgaben dienen. 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). Ziehen Sie ggf. alternative Implementierungen als Vergleich heran.
- Verwenden Sie SIMD-Befehle, wenn möglich.
- Fügen Sie Ihrem fertigen Quelltext Anweisungen hinzu, wie das Projekt kompiliert werden kann, beispielsweise ein funktionierendes `Makefile`.
- Eingabedateien, welche Sie generieren, um Ihre Implementierungen zu testen sollten mit abgegeben werden.
- Bonusaufgaben (sofern vorhanden) müssen nicht implementiert werden.
- Beachten Sie sämtliche Hinweise in der Praktikumsordnung bezüglich der Struktur und Bewertung Ihrer Abgabe.