LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Grundlagenpraktikum: Rechnerarchitektur

Arithmetik in Zahlensystemen mit ganzzahliger Basis (A318)
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 **Sonntag 24. Juli 2022, 23:59 Uhr (CEST)**. 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 **24.08.2022 – 01.09.2022** statt. Weitere Informationen werden noch bekannt gegeben. Beachten Sie, dass die Folien für die Präsentation am obigen Abgabetermin im PDF-Format abzugeben sind und keine nachträglichen Änderungen akzeptiert werden können.

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 Arithmetik in Zahlensystemen mit ganzzahliger Basis

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

Im Folgenden betrachten wir Zahlensysteme beliebiger ganzzahliger Basen. Sie kennen die g-adische Schreibweise einer Zahl A als Abfolge von Ziffern a_i mit ihrer Position entsprechender Wertigkeit gemäß g:

$$A = \sum_{i=0}^{n} a_i \cdot g^i \tag{1}$$

Im Alltag setzen wir $g = 10$ und schreiben beispielsweise:

$$123_{(10)} = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$$

Es kann aber auch $g = -2$ gelten, dort gilt beispielswiese:

$$11_{(-2)} = 1 \cdot (-2)^1 + 1 \cdot (-2)^0 = -1_{(10)}$$

In Ihrer Projektaufgabe soll für eine wählbare Basis $g \in \mathbb{Z} \setminus \{-1, 0, 1\}$ eine einfache arithmetische Operation (Addition, Subtraktion, Multiplikation; aber keine Division) auf zwei vom Nutzer vorgegebenen Zahlen durchgeführt und das Ergebnis in der gewählten Basis dargestellt werden. Bei positiven Basen sollen auch negative Zahlen behandelt werden können. Bei $|g| > 10$ soll die Repräsentation der möglichen Ziffernwerte wählbar sein.

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 Ihren 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). Der Zahlenbereich, in dem Ihr Programm einwandfrei funktionieren muss, ist nicht limitiert, es müssen ganzzahlige Werte beliebiger Größe unterstützt werden.

2.3.1 Theoretischer Teil

- Vollziehen Sie anhand der Gleichung für g-adische Zahlensysteme die beiden Programmdurchläufe nach.
- In welchem Zahlensystem arbeiten moderne Prozessoren? Ist dieses Wissen hilfreich bei der Bearbeitung der Aufgabe?
- Überlegen Sie, wie Sie eine Multiplikation und Addition von ganzen Zahlen beliebiger Genauigkeit möglichst effizient realisieren können.
- Vergleichen Sie eine Implementierung, welche die Eingaben vor der Berechnung in das Binärsystem umrechnet, mit einer anderen Implementierung, welche ohne Konvertierung arbeitet.

2.3.2 Praktischer Teil

- Welche Fehler können bei der Eingabe durch den Nutzer auftreten? Wie hängt die Länge des Alphabets mit der Basis zusammen? Reagieren Sie im Rahmenprogramm entsprechend auf fehlerhafte Nutzereingaben.
- Implementieren Sie in der Datei mit Ihrem C-Code die Funktion:

```
void arith_op_any_base(int base, const char* alph, const char* z1,
                       const char* z2, char op, char* result)
```

  Die Funktion nimmt vom Benutzer spezifizierte Werte für die Eingabezahlen `z1`, `z2` in der durch das Alphabet `alph` spezifizierten Basis `base`, sowie eine Operation `op` entgegen und schreibt das Ergebnis als String in den Speicherbereich, auf den `result` zeigt. Allokieren Sie hierzu in Ihrem Rahmenprogramm einen Buffer passender Größe. Denken Sie daran, dass alle Strings in C null-terminiert sind, was das zusätzliche Übergeben der Längen der Strings vermeidet.

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.
- `-b<int>` — Die Basis, in welcher gerechnet wird
- `-a<str>` — Das zu verwendende Alphabet
- `<str>` — Positional Argument: Die Zahl z_1
- `<str>` — Positional Argument: Die Zahl z_2
- `-o<char>` — Die zu verwendende Rechenoperation (also entweder $*$, $+$ oder $-$)
- `-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.