LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Grundlagenpraktikum: Rechnerarchitektur

Arithmetik in Zahlensystemen mit ungewöhnlicher Basis (A319)
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 C-Code anzufertigen ist, sind in C nach dem C17-Standard zu schreiben.

Der **Abgabetermin** ist **Sonntag 16. Juli 2023, 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 **21.08.2023 – 01.09.2023** 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 ungewöhnlicher 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

Wir betrachten zunächst Zahlensysteme beliebiger natürlicher 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 als Kurzform für den Wert der Zahl $3 \cdot 10^0 + 2 \cdot 10^1 + 1 \cdot 10^2$. Es folgt intuitiv, dass auf diese Weise beliebige natürliche Zahlen dargestellt werden können.
Wählt man $g = -1 + i$ mit $i^2 = -1$, so können alle komplexen Zahlen mit ganzen Real- und Imaginärteilen dargestellt werden, und das sogar, wenn man a_i als Elemente der Menge $\{0, 1\}$ festsetzt.
In Ihrer Projektaufgabe entwerfen Sie einen Rechner, welcher zwischen der üblichen kartesischen Darstellung einer komplexen Zahl $a + b \cdot i$ und ihrer Darstellung in Basis $-1 + i$ konvertieren kann.

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).

2.3.1 Theoretischer Teil

- Vollziehen Sie nach, warum die Darstellung der Zahl $(3 + 2i)$ zur Basis $-1 + i$ dem Wert 1001 entspricht. Entwickeln Sie darauf aufbauend Ihre Algorithmen zur Konversion in beide Richtungen. Ziehen Sie wenn nötig geeignete Literatur zu Rate.
- Zeigen Sie für $a \in \{1, 0, -1\}$ und $b \in \{1, 0, -1\}$, dass die komplexen Zahlen $(a + bi)$ zur Basis $-1 + i$ mit binären Koeffizienten dargestellt werden können.

2.3.2 Praktischer Teil

- Implementieren Sie in der Datei mit Ihrem C-Code die Funktion:

  ```
  void to_carthesian(unsigned __int128 bm1pi, __int128* real, __int128*
                                    imag)
  ```

  Die Funktion nimmt eine vom Benutzer spezifizierte Zahl (`bm1pi`), deren Bits eine Zahl in Darstellung zur Basis $-1 + i$ repräsentieren, entgegen und überführt diese in ihre kartesische Darstellung mit Realteil `real` und Imaginärteil `imag`.

- Implementieren Sie in der Datei mit Ihrem C-Code die Funktion:

  ```
  unsigned __int128 to_bm1pi(__int128 real, __int128 imag)
  ```

  Die Funktion bekommt eine komplexe Zahl in kartesischer Darstellung mit Realteil `real` und Imaginärteil `imag` übergeben und gibt einen Integer zurück, dessen Bits die Ziffern der Eingabezahl zur Basis $-1 + i$ darstellen.

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<Zahl>` — Die Implementierung, die verwendet werden soll. Hierbei soll mit `-V 0` Ihre Hauptimplementierung verwendet werden. Wenn diese Option nicht gesetzt wird, soll ebenfalls die Hauptimplementierung ausgeführt werden.
- `-B<Zahl>` — Falls gesetzt, wird die Laufzeit der angegebenen Implementierung gemessen und ausgegeben. Das *optionale* Argument dieser Option gibt die Anzahl an Wiederholungen des Funktionsaufrufs an.
- `<Zahl>` — Positional Argument: Die Zahl wird zur Basis $-1 + i$ interpretiert und *to_carthesian* wird aufgerufen
- `<Zahl>,<Zahl>` — Positional Argument: Die Zahl wird als komplexe Zahl interpretiert und *to_bm1pi* wird aufgerufen

- `-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. Verwenden Sie keinen Inline-Assembler. 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.