LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Grundlagenpraktikum: Rechnerarchitektur

CRC32 (A400)

Projektaufgabe – Aufgabenbereich Theorie der Informationsverarbeitung

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 CRC32

2.1 Überblick

Die Informationstheorie ist ein Feld der Mathematik, in welchem man sich allgemein mit dem Kodieren von Nachrichten aufgrund von statistischen Gegebenheiten beschäftigt. Sie werden in Ihrer Projektaufgabe einen bestimmten Teilbereich näher beleuchten und einen bestimmten Algorithmus in C implementieren.

2.2 Funktionsweise

Die Zyklische Redundanzprüfung ist ein fehlererkennender Algorithmus, welcher oft zum Erstellen einfacher Prüfsummen von Dateien verwendet wird.
Die Berechnung der CRC32 beruht auf Polynomdivision. Dabei wird die Bitfolge einer zu prüfenden Nachricht N der Länge l als Polynom mit binären Koeffizienten K_i aufgefasst:

$$N = K_{l-1}K_{l-2}\dots K_1K_0 \quad \hat{=} \quad \sum_{i=0}^{l-1} K_i \cdot x^i =: N(x)$$

Als Dividend für die Polynomdivision kommt ein sogenanntes Generatorpolynom $G(x)$ zum Einsatz. Die zu berechnende Prüfsumme P ist dann der Rest der Polynomdivision von $N(x)/G(x)$. Formal gilt:

$$\mathrm{CRC32}(N, G) = P(x) \equiv N(x) \mod G(x)$$

Die Prüfsumme P wird nach der Berechnung einfach an die ursprüngliche Nachricht N angehängt: $N' = (N||P)$
Der Empfänger führt nach Erhalten der Nachricht N' die gleiche Polynomdivision mit dem gleichen $G(x)$ aus. Ergibt sich als Rest 0, so kann davon ausgegangen werden, dass die Nachricht mit großer Wahrscheinlichkeit korrekt übermittelt wurde.

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.

2.3.1 Theoretischer Teil

- Erarbeiten Sie anhand geeigneter Sekundärliteratur die genaue Funktionsweise des CRC32-Algorithmus. Rechnen Sie zunächst auf Papier nach, dass das oben skizzierte Verfahren funktionieren kann. Überlegen Sie sich auch, wie Addition und Subtraktion innerhalb eines binären Feldes funktionieren.
- Welche Arten von Fehlern erkennt CRC32? Zeigen Sie, dass es mindestens eine Klasse von Fehlern gibt, für die CRC32 nicht funktioniert. Geben Sie dazu geeignete Beispiele für zwei Nachrichten N und M mit $N \neq M$ an, für die aber $CRC32(N) = CRC32(M)$ gilt.
- Suchen Sie nach alternativen Implementierungen von CRC32 und vergleichen Sie sie mit der Ausgabe Ihres fertigen Programms. Vergleichen Sie auch die Geschwindigkeit Ihrer Implementierung und einer alternativen (beispielsweise C) Implementierung.

2.3.2 Praktischer Teil

- Implementieren Sie im Rahmenprogramm I/O-Operationen in C, mit welchen Sie Ihrer C-Funktion die Inhalte einer eingelesenen Datei als Pointer übergeben können.
- Implementieren Sie im Rahmenprogramm I/O-Operationen in C, mit welchen Sie Ihrer C-Funktion ein beliebig von Benutzer spezifizierbares Generatorpolynom G übergeben können. Sie dürfen die Länge von G auf maximal 32-Bit beschränken.
- Welches ist das vom IEEE802.3 (Ethernet) Standard spezifizierte Generatorpolynom? Verwenden Sie dieses Generatorpolynom als Standardwert, wenn der Benutzer G nicht explizit angegeben hat.
- Implementieren Sie in der Datei mit Ihrem C-Code die Funktion:

  ```
  uint32_t CRC32(size_t len, const char msg[len], uint32_t generator)
  ```

  Die Funktion bekommt einen Pointer auf eine Nachricht `msg` der Länge `len` sowie ein Generatorpolynom `generator` übergeben und gibt die Prüfsumme P als Integer 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<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.
- `<Dateiname>` — Positional Argument: Eingabedatei
- `-G<Zahl>` — Generatorpolynom
- `-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.