LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Aspekte der systemnahen Programmierung bei der Spieleentwicklung

Lauflängencodierung (A403)

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 Assembler-Code anzufertigen ist, sind für die 64-Bit ARMv8-Architektur (AArch64) zu schreiben.

Der **Abgabetermin** ist der **29. Januar 2020, 11:59 Uhr (MEZ)**. Die Abgabe erfolgt per Git in das für Ihre Gruppe eingerichtete Projektrepository. Bitte beachten Sie die in der Praktikumsordnung angebene Liste von abzugebenden Dateien.

Der **erste Teil Ihrer Projektpräsentation** ist eine kurze Vorstellung Ihrer Aufgabe im Umfang von ca. 2 Minuten in einer Tutorübung in der Woche **09.12.2019 – 13.12.2019**. Erscheinen Sie bitte **mit allen Team-Mitgliedern** und wählen Sie bitte eine Übung, in der mindestens ein Team-Mitglied angemeldet ist.

Die **Abschlusspräsentationen** finden in der Zeit vom **02.03.2020 – 06.03.2020** statt. Weitere Informationen zum Ablauf und die Zuteilung der einzelnen Präsentationstermine werden noch bekannt gegeben. Beachten Sie, dass die Folien für die Präsentation am obigen Abgabetermin im PDF-Format abzugeben sind.

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

PS: Vergessen Sie nicht, sich rechtzeitig in TUMonline zur Prüfung anzumelden. Dies ist Voraussetzung für eine erfolgreiche Teilnahme am Praktikum im laufenden Semester.

[1] `https://www.caps.in.tum.de/lehre/ws19/praktika/asp/`

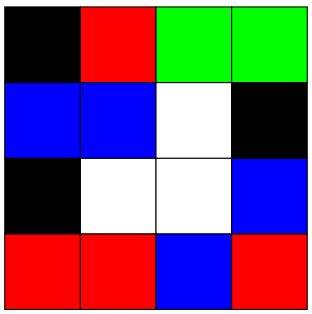
2 Lauflängencodierung

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 Assembler implementieren.

2.2 Funktionsweise

Die Lauflängencodierung (Run-Length-Encoding) ist ein einfaches Verfahren zur Datenkompression. Betrachten Sie beispielsweise die 16 Pixel des folgenden Bildes:



Anstatt die Pixel einzeln abzuspeichern, kann man auch *Folgen* von gleichen Pixeln speichern. Im konkreten Fall scannt man das Bild zeilenweise von links oben nach rechts unten und speichert Tupel $T = (L, V)$ mit L hintereinander auftretenden Werten $V \in \{R, G, B, W, S\}$:

$$RLE = ((1,S),(1,R),(2,G),(2,B),(1,W),(2,S),(2,W),(1,B),(2,R),(1,B),(1,R)) \tag{1}$$

Die Werte lassen sich dann wiefolgt lesen:

- $(1, S)$ Ein Pixel Schwarz
- $(1, R)$ Ein Pixel Rot
- $(2, G)$ Zwei Pixel Grün
- ...

So lässt sich aus den komprimierten Daten das Bild Stück für Stück wieder rekonstruieren.

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.

2.3.1 Theoretischer Teil

- Erarbeiten Sie anhand geeigneter Sekundärliteratur die genaue Funktionsweise der Lauflängenkodierung.
- Erarbeiten Sie ein geeignetes Format zum Speichern der Liste RLE von Tupeln T, welches Sie als Ausgabeformat Ihrer Implementierung verwenden werden. Versuchen Sie dabei, so kompakt wie möglich zu arbeiten, das heißt sie sollten in jedem Fall eine binäre Darstellung Ihrer Daten wählen.
- Untersuchen Sie Ihre fertige Implementierung auf Performanz. Errechnen Sie auch die durchschnittliche Kompressionsrate für die Datei `tick.gepasp` und für repräsentative Eingaben ihrer Wahl.

2.3.2 Praktischer Teil

- Implementieren Sie im Rahmenprogramm I/O-Operationen in C, mit welchen Sie Ihrem Assemblerprogramm die Inhalte einer eingelesenen Datei als Pointer übergeben können.

  Ein unkomprimiertes einzulesendes Bild liegt hexadezimal in folgendem Format vor:

  - Der Header reserviert je 16 Bit für die Höhe und die Breite des Bildes im Little-Endian-Format.
  - Danach folgt ein 8-Bit-Wert für die Zahl der Farben in der Palette. Es wird auf Basis 0 gezählt, das heißt, eine Farbe entspricht der Null, zwei Farben der Zahl eins, usw …
  - Anschließend folgen die Palettendaten im Format RRGGBB für den Rot-, Grün-, und Blaukanal der Farbe. Es folgen so viele Farben wie vorher als Farbanzahl spezifiziert.
  - Als Letztes folgen die Bilddaten selbst, wobei ein Pixel jeweils einem Byte entspricht. Der Wert des Bytes ist der Index innerhalb der Palette. (Basis 0)

  Das Beispielbild aus der Angabe sähe im Hexeditor beispielsweise so aus:

```
0000: 04 00 04 00 04 00 00 00
0008: FF 00 00 00 FF 00 00 00
0010: FF FF FF FF 00 01 02 02
0018: 03 03 04 00 00 04 04 03
0020: 01 01 03 01
```

- Implementieren Sie in der Datei mit dem Assemblercode eine Funktion

  ```
  int run_length_encode(char *data, char *result, unsigned int result_size)
  ```

  welche einen Pointer auf die unkomprimierten Eingabedaten `data` übergeben bekommt und die lauflängenkodierten Daten in das Array `result` speichert. Der Parameter `result_size` gibt die maximale Größe des für `result` reservierten Speicherbereichs an. Der Rückgabewert soll im Erfolgsfall die Länge der kodierten Daten sein. Wenn die Funktion fehlschlägt (beispielsweise, weil das Ergebnisarray nicht groß genug gewählt wurde) soll -1 zurückgegeben werden. Speichern Sie das komprimierte Bild im von Ihnen gewählten Format in einer vom Benutzer wählbaren Datei im Rahmenprogramm (d.h. in C).

- Implementieren Sie in der Datei mit dem Assemblercode eine Funktion

  ```
  int run_length_decode(char *data, char *result, unsigned int result_size)
  ```

  welche einen Pointer auf die lauflängenkodierten Eingabedaten `data` übergeben bekommt und die unkomprimierten Daten in das Array `result` speichert. Der Parameter `result_size` gibt die maximale Größe des für `result` reservierten Speicherbereichs an. Der Rückgabewert soll im Erfolgsfall die Länge der unkomprimierten Daten sein. Wenn die Funktion fehlschlägt (beispielsweise, weil das Ergebnisarray nicht groß genug gewählt wurde) soll -1 zurückgegeben werden.

2.3.3 Zusatzaufgabe

Implementieren Sie mithilfe von gefärbten ASCII/ANSI-Zeichen einen Bildbetrachter, der das dekomprimierte Bild auf der Konsole ausgibt.

2.4 Allgemeine Bewertungshinweise

Die folgende Liste soll Ihnen als Gedächtnisstütze beim Bearbeiten der Aufgaben dienen. Beachten Sie ebenfalls die in der Praktikumsordnung angegebenen Hinweise.

- Stellen Sie unbedingt sicher, dass Ihre Abgabe auf der Referenzplattform des Praktikums (HimMUC-Cluster) kompiliert und funktionsfähig ist.
- Fügen Sie Ihrem Projekt ein funktionierendes `Makefile` hinzu, welches durch den Aufruf von `make` Ihr Projekt kompiliert.
- Verwenden Sie keinen Inline-Assembler.

- Verwenden Sie SIMD-Befehle, wenn möglich.
- Sie dürfen die Signatur der in Assembler zu implementierenden Funktion nur dann ändern, wenn Sie dies (in Ihrer Ausarbeitung) rechtfertigen können.
- I/O-Operationen dürfen grundsätzlich in C implementiert werden.
- 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.
- Verwenden Sie für die Ausarbeitung die bereitgestellte TEX-Vorlage und legen Sie sowohl die PDF-Datei als auch sämtliche TEX-Quellen in das Repository.
- 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 4:3 als Folien-Format.
- Zusatzaufgaben (sofern vorhanden) müssen nicht implementiert werden. Es gibt keine Bonuspunkte.