ERA Praktikum SS 2017

Gleitkomma-Arithmetik

Realisierung von sin x

Inhaltsverzeichnis

Ablauf des Projekts, Probleme und Lösungen

Im ersten Treffen wurde ein grober Überblick über die Aufgabenstellung geschafft und die Rollen wurden verteilt. Zuständig für die Dokumentation war Florian Müller, Thea Kramer für den Vortrag und Berzan Yildiz für die Projektleitung.

Im nächsten Treffen haben wir das Pflichtenheft fertiggestellt und uns im Rahmen dessen mit den zu erwägenden Lösungsmethoden (Reihenentwicklung und Lookup-Table) und Werkzeugen zur Implementierung (nasm, C-Rahmenprogramm und zu der Zeit noch voraussichtlich GNU) auseinandergesetzt.

Anschließend haben wir die Spezifikation fertiggestellt, wobei wir uns gegen die Reihenentwicklung und für die effizientere Lookup-Table entschieden haben, und auch direkt in Pseudocode festgehalten haben, wie wir diese verwirklichen würden. Unser Prinzip, zunächst den Eingabewert x auf $\frac{\pi}{2}$ abzubilden, dann Binär mithilfe dieser Abbildung in der Lookup-Table zu suchen und zuletzt zu interpolieren, hat sich tatsächlich bis zur endgültigen Version gehalten.

Wir haben im nächsten Treffen direkt schon damit angefangen, den Pseudocode aus der Spezifikation in Assemblercode umzuschreiben, wobei vor allem das Übersetzen der Binären Suche von Pseudocode in Assemblercode am kompliziertesten war. Wir sind an einem Tag bis zu der Implementierung der Interpolation gekommen.

Zum nächsten Treffen ist uns jedoch aufgefallen, dass wir alles mit den regulären Registern implementiert hatten, und eigentlich die FPU-Register und ihre Befehle nutzen mussten. Somit haben wir einen ganzen Tag an Zeit verloren. Die Implementation mithilfe der FPU war dadurch schwierig für uns, dass wir im ersten Semester in der Vorlesung “Einführung in die Rechnerarchitektur” nicht in dieser Weise gearbeitet hatten und uns somit komplett in die ungewohnte Arbeitsweise von FPU-Assembler einarbeiten mussten. Zusätzlich dazu haben wir auch noch kleinere Denkfehler wie verschätzte Speicherplatzgrößen, oder falsche Bedingungen für Sprünge behoben.

Im nächsten Treffen haben wir uns an die Kompilierung gesetzt. Die Einarbeitung in die Tools und Arbeitsweise mit Assembler und C hat uns sehr viel Zeit und Aufwand gekostet. Wir wussten zunächst nicht genau, wie das Kompilieren funktionierte und mussten uns auch mit einigen ungewohnten Fehlermeldungen auseinandersetzen.

Wir haben das Projekt relativ früh ausschließlich auf MacOS implementiert, da uns schnell aufgefallen ist, dass die Arbeit mit einem Unix-System erheblich einfacher ist und bei den Windows-Systemen oft Fehler auftraten.

Um Fehler im Code zu beheben haben wir immer wieder alles bis auf einzelne Abschnitte herausgeschnitten, dann kompiliert und auf entstehende Fehler geachtet und uns somit von Abschnittsweise durch den Code vorangetastet, bis keine Fehler mehr auftraten. Dies war definitiv ineffizient, aber wir hatten keine Alternative, da es uns nicht gelang, den Code im Terminal zu debuggen.

Bei dem Schreiben des C-Rahmenprogramms mussten wir einige MacOS spezifische Dinge beachten, die beim Kompilieren aufgefallen sind, wie beispielsweise die Nutzung von “Clang” statt GNU oder unterschiedlichen Funktionsnamen in C (z.B “`_floatSin`” bei MacOS statt regulär “`floatSin`”).

Bei dem C-Rahmenprogramm war der schwierigste Teil definitiv die Übergabe eines Wertes von C nach Assembler, da uns zunächst unklar war, wie ein Wert von C an eine Assemblerfunktion übergeben wird und in welchem Register dieser in Assembler anschließend vorzufinden ist. Dies war jedoch mit etwas Recherche lösbar.

Allgemein lag die größte Hürde bei der Implementierung definitiv dabei, dass wir nicht normal debuggen konnten und die Fehlerfindung sich somit als sehr kompliziert gestaltet hat, zumindest im Vergleich zu den umfangreichen Java IDEs.

Auch war es generell nicht so leicht Antworten auf jegliche Fragen und Fehlermeldungen im Internet zu finden, da es sehr wenig C und Assembler bezogene Beiträge und Dokumentation gibt.

Das Schreiben des C-Testprogramms war nicht weiter schwierig, da C sich relativ ähnlich zu Java verhält und auch die Einbindung der Standardbibliothek für die Sinusfunktion unkompliziert war. Lediglich die gewöhnungsbedürftige Funktionsweisen von “`scanf`” und “`printf`” waren verwirrend, aber auch dies war nicht weiter schwierig.

Das Erstellen der Makefile war nicht weiter kompliziert, da wir uns schon zuvor reichlich mit den benötigten Terminal Befehlen auseinandergesetzt hatten und diese nur noch in die Makefile einbinden mussten.

Rückblickende Analyse für Umsetzung der Zeitplanung

Rückblickend war die insgesamt eingeplante Zeit nicht schlecht eingeschätzt, denn bei 8 Treffen mit im Schnitt 7 Stunden Arbeit kamen wir auf ca. 56 Stunden pro Person, was etwas mehr als die ursprünglich geplanten 50 Stunden war. Dies lag hauptsächlich daran, dass die Implementierung mehr Zeit in Anspruch nahm, als zunächst angenommen. Vortrag und Dokumentation wurden hierbei nicht mit einberechnet, da sie zur Zeit der Fertigstellung dieses Berichtes noch nicht erarbeitet worden sind.

Die Organisation hat nicht mehr als 1 Stunde in Anspruch genommen und hat reibungslos über WhatsApp funktioniert. Auch die Besprechungen und die Aufgabenanalyse waren gut eingeplant, hier sind wir nicht stark von abgekommen. Da diese Punkte fließend in die übrige Teamarbeit übergingen, sind diese auch schwierig separat zu bewerten.

Die reine Schreibarbeit am Pflichtenheft hat auch nicht wirklich mehr als 4 Stunden gedauert. Insgesamt war die theoretische Arbeit und Zeit zum Nachdenken an dem Pflichtenheft gut geplant, hier waren 6 Stunden gut veranschlagt.

Für die Spezifikation haben wir etwas mehr als für das Pflichtenheft gebraucht, auch hier war die Zeitplanung relativ akkurat. Da wir immer Assembler vor VHDL bearbeitet haben, ist hier auch ein Großteil der Vorarbeit für die Strukturierung der Dokumente geschehen. Dadurch hat das Assemblerprojekt in diesen Phasen mehr Zeit eingenommen, als das VHDL Projekt.

Die Implementierung insgesamt hat uns definitiv mehr als 11 Stunden gekostet, eher näher an 14. Sie war der aufwendigste und auch anspruchsvollste Teil der Projektarbeit.

4 Stunden für die Protokollierung war rückblickend zu viel eingeplant, denn die Protokolle fertigzustellen hat viel weniger Zeit in Anspruch genommen, als zunächst erwartet.

Im Großen und Ganzen also haben wir unsere Zeitplanung gut umgesetzt. Vor allem haben wir uns als Ziel gesetzt, nicht mehr als zwei Treffen pro Arbeitsschritt (Pflichtenheft, Spezifikation etc.) zu brauchen. Dies hat auch gut geklappt, wobei manche

Projektabschnitte, wie zum Beispiel das Pflichtenheft oder die Spezifikation weniger Zeit brauchten, als andere. Hierdurch konnte man aber in den anspruchsvolleren Abschnitten, wie beispielsweise der Implementierung, mehr Zeit investieren und dies somit gut ausgleichen.

Von anfänglich einem Treffen pro Woche sind wir schnell auf zwei gewechselt, da wir ohne Arbeitsteilung gearbeitet haben, wodurch der Zeitaufwand etwas höher war als zu Beginn geplant wurde. Dies hatte jedoch gewisse Vorteile, auf die ich im nächsten Teil noch eingehen werde.

Allgemeine Bewertung der Teamarbeit

Die Teamarbeit hat bei uns sehr gut funktioniert, da wir stets in Kontakt waren und alle Teammitglieder gleich stark motiviert waren. Es wurde ein bisschen diskutiert, ob eine Arbeitsteilung nicht sinnvoller bzw. effizienter wäre. Letztendlich haben wir uns jedoch dagegen entschieden, da wir bevorzugt haben, dass auch jeder im Team von jedem Projektabschnitt etwas mitbekommt und von jedem Aspekt sowohl des Assembler-, als auch des VHDL-Projektes eine Ahnung hatte. Dadurch, dass jedes Teammitglied sich bei jedem Projekt gut auskannte, wurde die Kooperation erleichtert und jeder hat gleich viel zu den Projekten beigetragen. Auch wurde hierdurch Vorarbeit für die späteren Phasen der Dokumentation und des Vortrags geleistet, da jeder nun das Fachwissen hatte, in diesen Phasen gute Arbeit zu leisten. Ich denke, die Vorteile unserer Herangehensweise haben ihre Nachteile klar überwogen.

Einzelne Schwierigkeiten in der Terminfindung aufgrund von unterschiedlichen Wochenplanungen der einzelnen Teammitglieder konnten durch gute Zeitplanung, vorausschauende Raumreservierung und etwas Kompromissfähigkeit souverän bewältigt werden.