Albert Ludwigs Universität Freiburg

Technische Fakultät

PicoC-Compiler

Übersetzung einer Untermenge von C in den Befehlssatz der RETI-CPU

Bachelorarbeit

Abgabedatum: 28th April 2022

Author:
Jürgen Mattheis

Gutachter:
Prof. Dr. Scholl

Betreung:
M.Sc. Seufert

Eine Bachelorarbeit am Lehrstuhl für
Betriebssysteme

ERKLÄRUNG

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Inhaltsverzeichnis

Abbildungsverzeichnis

Codeverzeichnis

Tabellenverzeichnis

Definitionsverzeichnis

Grammatikverzeichnis

1 Motivation

Als Programmierer kommt man nicht drumherum einen **Compiler** zu nutzen, er ist geradezu **essentiel** für den Beruf oder das Hobby des Programmierens. Selbst in der Programmiersprachen **Python**, welche als interpretierte Sprache bekannt ist, wird das in der Programmiersprache **Python** geschriebene Programm vorher zu **Bytecode** kompiliert, bevor dieser von der **Python Virtual Machine (PVM)** interpretiert wird.

Compiler, wie der **GCC**[1] oder **Clang**[2] werden üblicherweise über eine **Commandline-Schnittstelle** verwendet, welche es für den Benutzer **unkompliziert** macht ein Programm, dass in der Programmiersprache geschrieben ist, die der Compiler kompiliert[3] zu **Maschinencode** zu kompilieren.

Meist funktioniert das über schlichtes und einfaches **Angeben der Datei**, die das Programm enthält, welches kompiliert werden soll, z.B. im Fall des **GCC** über `> gcc file.c -o machine_code`[4]. Der Benutzer muss dazu **nichts** über die **Theoretischen Grundlagen des Compilerbau** wissen, noch wie der Compiler **intern** umgesetzt ist. Als Ergebnis erhält man im Fall des **GCC** die mit der Option `-o` selbst benannte Datei `machine_code`, welche dann zumindest unter **Unix** über `> ./machine_code` **ausgeführt** werden kann, wenn das **Ausführungsrecht** gesetzt ist.

Der ganze Kompiliervorgang kann, wie es in Abbildung 1.1 dargestellt ist zu einer Box abstrahiert werden, der Benutzer gibt ein Programm in der Sprache des Compilers rein und erhält **Maschinencode**, denn er dann im besten Fall in eine andere Box hineingeben kann, mit der passenden **Maschine** oder mit einem **Interpreter** in Form einer **Virtuellen Maschine** ausführen kann.

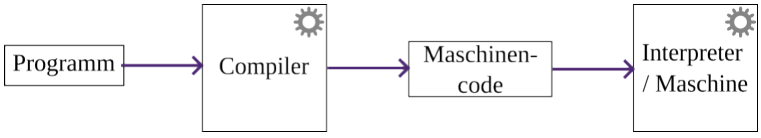

Abbildung 1.1: *Stark vereinfachte Schritte zum Ausführen eines Programmes*

1.1 RETI-Architektur

Die RETI ... basiert auf ... der Vorlesung C. Scholl, „Betriebssysteme“.

Definition 1.1: Caller-save Register

a

[a] G. Siek, *Course Webpage for Compilers (P423, P523, E313, and E513)*.

[1] *GCC, the GNU Compiler Collection - GNU Project*.
[2] *clang: C++ Compiler*.
[3] Im Fall des **GCC** und **Clang** ist es die Programmiersprache **C**.
[4] Bei mehreren Dateien... allderdings.

Definition 1.2: Callee-save Register

[a]

[a] G. Siek, *Course Webpage for Compilers (P423, P523, E313, and E513)*.

1.2 PicoC

Die Sprache

1.3 Eigenheiten der Sprache C

Definition 1.3: Deklaration

[a]

[a] P. Scholl, „Einführung in Embedded Systems“.

Definition 1.4: Definition

[a]

[a] P. Scholl, „Einführung in Embedded Systems“.

Definition 1.5: Allokation

[a]

[a] Thiemann, „Einführung in die Programmierung“.

Definition 1.6: Initialisierung

[a]

[a] Thiemann, „Einführung in die Programmierung“.

Definition 1.7: Scope

[a]

[a] Thiemann, „Einführung in die Programmierung“.

Definition 1.8: Call by value

[a]

[a] Bast, „Programmieren in C“.

Definition 1.9: Call by reference

[a]

[a] Bast, „Programmieren in C“.

1.4 Gesetzte Schwerpunkte

Die **Laufzeit** ist bei Compilern zwar vor allem in der Industrie **nicht unwichtig**, aber bei **Compilern**, verglichen mit **Interpretern** weniger zu gewichten, da ein Compiler bei einem fertig implementierten Programm nur **einmal** Maschinencode generieren muss und dieser Maschinencode danach fortan ausgeführt wird. Beim einem **Compiler** ist daher eher zu priorisieren, dass der kompilierte **Maschinencode** möglichst **effizient** ist.

Beim **PicoC-Compiler** wurde eher darauf Wert gelegt **sauberen, strukturierten Code** zu schreiben, den die Studenten sogar selber verstehen könnten und eine **unkomplizierte Bibliothek** mit **guter Dokumentation**[5], nämlich das **Lark Parsing Toolkit**[6] für das **Parsen** zu verwenden. Vor allem, da zu erwarten ist, dass der **PicoC-Compiler** vielleicht in einigen anderen Projekten eingebunden werden könnte, ist es von **Vorteil** bei der Notwendigkeit kleiner **Erweiterungen**, diese Erweiterungen **unkompliziert** durchführen zu können.

1.5 Richtlinien

1.6 Still der Arbeit

[5] *Welcome to Lark's documentation! — Lark documentation*.
[6] *Lark - a parsing toolkit for Python*.

Literatur

Online

- *clang: C++ Compiler*. URL: http://clang.org/ (besucht am 29. 07. 2022).
- *GCC, the GNU Compiler Collection - GNU Project*. URL: https://gcc.gnu.org/ (besucht am 13. 07. 2022).
- *Welcome to Lark's documentation! — Lark documentation*. URL: https://lark-parser.readthedocs.io/en/latest/ (besucht am 31. 07. 2022).

Bücher

- G. Siek, Jeremy. *Course Webpage for Compilers (P423, P523, E313, and E513)*. 28. Jan. 2022. URL: https://iucompilercourse.github.io/IU-Fall-2021/ (besucht am 28. 01. 2022).

Vorlesungen

- Bast, Hannah. „Programmieren in C“. Vorlesung. Vorlesung. Universität Freiburg, 2020. URL: https://ad-wiki.informatik.uni-freiburg.de/teaching/ProgrammierenCplusplusSS2020 (besucht am 09. 07. 2022).
- Scholl, Christoph. „Betriebssysteme“. Vorlesung. Vorlesung. Universität Freiburg, 2020. URL: https://abs.informatik.uni-freiburg.de/src/teach_main.php?id=157 (besucht am 09. 07. 2022).
- Scholl, Philipp. „Einführung in Embedded Systems“. Vorlesung. Vorlesung. Universität Freiburg, 2021. URL: https://earth.informatik.uni-freiburg.de/uploads/es-2122/ (besucht am 09. 07. 2022).
- Thiemann, Peter. „Einführung in die Programmierung“. Vorlesung. Vorlesung. Universität Freiburg, 2018. URL: http://proglang.informatik.uni-freiburg.de/teaching/info1/2018/ (besucht am 09. 07. 2022).

Sonstige Quellen

- *Lark - a parsing toolkit for Python*. 26. Apr. 2022. URL: https://github.com/lark-parser/lark (besucht am 28. 04. 2022).