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 Ergebnisse und Ausblick

Zum Schluss soll ein **Überblick** über das gegeben werden, was im Kapitel **??** implementiert wurde. In Unterkapitel 1.1 wird mithilfe **kurzer Anleitungen** ein grober Einblick in die **wichtigsten Funktionalitäten** des implementierten **PicoC-Compilers** und **anderer mitimplementierter Tools** gegeben. Im Unterkapitel 1.2 wird aufgezeigt, was zur **Qualitätssicherung** implementiert wurde, um zu gewährleisten, dass der **PicoC-Compiler** die Kompilierung der **Programmiersprache** L_{PicoC} in **Syntax** und **Semantik identisch** zur entsprechenden **Untermenge** der Programmiersprache L_C umsetzt. Als allerletztes wird im Unterkapitel 1.3 ein Ausblick gegeben, wie der PicoC-Compiler **erweitert** werden könnte.

1.1 Funktionsumfang

Bei der Implementierung des **PicoC-Compilers** wurden verschiedene **Kommandozeilenoptionen** und **Modes** implemenetiert. Diese werden in den folgenden Kapiteln 1.1.1, 1.1.2 und 1.1.3 mithilfe kleiner **Anleitungen** erklärt werden. Eine **Dokumentation** die **ausführlicher** ist, als sie durch die **Anleitungen** gegeben ist, kann unter Link[1] gefunden werden.

1.1.1 Kommandozeilenoptionen

Will man einfach nur ein **Programm** `program.picoc` kompilieren ist das mit dem **PicoC-Compiler** genauso **unkompliziert** wie mit dem **GCC** durch einfaches **Angeben der Datei**, die kompiliert werden soll: `> picoc_compiler program.picoc`. Als Ergebnis des Kompiliervorgangs wird eine Datei `program.reti` mit dem entsprechenden **RETI-Code** erstellt, wobei für die **Benennung der Datei** einfach nur der **Basisname** der Datei `program` an eine neue **Dateiendung** `.reti` angehängt wird[2].

Daneben gibt es allerdings auch die Möglichkeit **Kommandozeilenoptionen** `<cli-options>` in der Form `> picoc_compiler <cli-options> program.picoc` mitanzugeben, von denen die **wichtigsten** in Tabelle 1.1 erklärt sind. Alle weiteren **Kommandozeilenoptionen** können in der **Dokumenation** unter Link[1] nachgelesen werden.

[1] https://github.com/matthejue/PicoC-Compiler/blob/new_architecture/doc/help-page.txt
[2] Beim **GCC** wird bei **Nicht-Angabe** eines **Dateinamen** mit der `-o` Option dagegen eine Datei mit der festen Namen `a.out` erstellt.

Kommandozeilenoption	Beschreibung	Standardwert
`-i`, `--intermediate_stages`	Gibt **Zwischenschritte** der Kompilierung in Form der verschiedenen **Tokens**, **Ableitungsbäume**, **Abstrakten Syntaxbäume** der verschiedenen **Passes** in Dateien mit **entsprechenden Dateiendungen** aber gleichem **Basinamen** aus. Im **Shell-Mode** erfolgt **keine** Ausgabe in Dateien, sondern nur im **Terminal**.	`false`
`-p`, `--print`	Gibt alle **Dateiausgaben** auch im **Terminal** aus. Diese Option ist im **Shell-Mode** dauerhaft aktiviert.	`false` (`true` im **Shell-Mode**)
`-v`, `--verbose`	Fügt den verschiedenen **Zwischenschritten der Kompilierung**, unter anderem auch dem finalen RETI-Code **Kommentare** hinzu, welche ein **Statement** oder **Befehl** aus einem **vorherigen Pass** beinhalten, der durch die darunterliegenden Statements oder Befehle **ersetzt** wurde. Wenn die `--run`-Option aktivert ist, wird der **Zustand** der virtuellen RETI-CPU **vor** und **nach jedem Befehl** angezeigt.	`false`
`-vv`, `--double_verbose`	Hat **dieselben Effekte**, wie die –**verbose**-Option, aber bewirkt zusätzlich **weitere Effekte**. **PicoC-Knoten** erhalten bei der Ausgabe in den Abstrakten Syntaxbäumen zustätzliche **runde Klammern**, sodass direkter abgelesen werden kann, wo ein Knoten anfängt und wo einer aufhört. In **Fehlermeldungen** werden **mehr Tokens** angezeigt, die an der Stelle der Fehlermeldung erwartet worden wären. Bei Aktivierung der `--intermediate_stages`-Option werden in den dadurch ausgegebenen **Abstrakten Syntaxbäumen** ebenfalls versteckte Attribute, die **Informationen** zu **Datentypen** und für **Fehlermeldungen** beinhalten angezeigt.	`false`
`-h`, `--help`	Zeigt die **Dokumentation**, welche ebenfalls unter Link[1] gefunden werden kann **im Terminal** an.	`false`
`-R`, `--run`	Führt die **RETI-Befehle**, die das Ergebnis des Kompilierung sind mit einer **virtuellen RETI-CPU** aus. Wenn die `--intermediate_stages`-Option aktiviert ist, wird eine Datei `<basename>.reti_states` erstellt, welche den **Zustsand der RETI-CPU** nach dem **letzten** ausgeführten RETI-Befehl enthält. Wenn die `--verbose`- oder `--double_verbose`-Option aktiviert ist, wird der Zustand der RETI-CPU **vor** und **nach** jedem Befehl auch noch zusätlich in die Datei `<basename>.reti_states` ausgegeben.	`false`
`-B`, `--process_begin`	Setzt die **relative Adresse**, wo der **Prozess** bzw. das **Codesegment** für das ausgeführte Programm beginnt.	`3`
`-D`, `--datasegment_size`	Setzt die Größe des **Datensegments**. Diese **Option** muss mit **Vorsicht** gesetzt werden, denn wenn der Wert zu niedrig gesetzt wird, dann können die **Globalen Statischen Daten** und der **Stack** miteinander kollidieren.	`32`

Tabelle 1.1: *Kommandozeilenoptionen*

1.1.2 Shell-Mode

Will man z.B. **Folgen von Statements** in der Programmiersprache L_{PicoC} **schnell** kompilieren ohne eine Datei erstellen zu müssen, so kann der **PicoC-Compiler** im sogenannten **Shell-Mode** aufgerufen werden. Hierzu wird der PicoC-Compiler **ohne Argumente** `picoc_compiler` aufgerufen, wie es in Code 1.1 zu sehen ist.

Mit dem `compile <cli-options> <filename>`-Befehl (oder der Abkürzung `cpl <cli-options> <filename>`) kann **PicoC-Code** zu **RETI-Code** kompiliert werden. Die Kommandozeilenoptionen `<cli-options>` sind dieselben, wie wenn der Compiler **direkt** mit Kommandozeilenoptionen aufgerufen wird. Die wichtigsten Kommandozeilenoptionen sind in Tabelle **??** angegeben.

icoC Code can be compiled into RETI Code with the ‘compile ¡cli options¿ ≪code¿”;‘ command (shortcut ‘cpl‘). The cli options are the same as for calling the compiler from outside, except for the ‘infile‘ argument which is interpreted as string with PicoC Code and which will be compiled as if it was enclosed in a main function.

```
> picoc_compiler
PicoC Shell. Enter `help` (shortcut `?`) to see the manual.
PicoC> cpl "6 * 7;";
--------------------------- RETI ---------------------------
SUBI SP 1;
LOADI ACC 6;
STOREIN SP ACC 1;
SUBI SP 1;
LOADI ACC 7;
STOREIN SP ACC 1;
LOADIN SP ACC 2;
LOADIN SP IN2 1;
MULT ACC IN2;
STOREIN SP ACC 2;
ADDI SP 1;
LOADIN BAF PC -1;

Compilation successfull

PicoC> quit
```

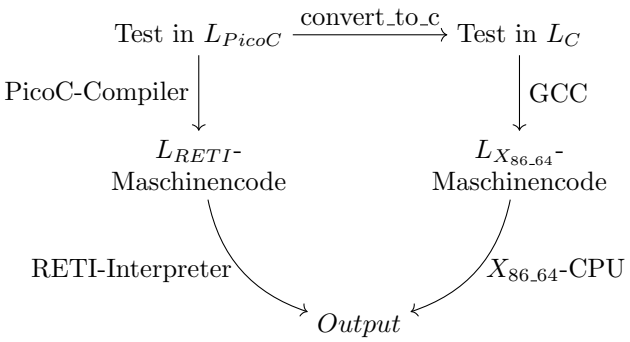
Code 1.1: *Shellaufruf*

Wenn man möglichst alle nützlichen **Kommandozeilenoptionen** angegeben haben will, bei denen es **keinen** Grund gibt, sie nicht mitanzugeben, kann der Befehl `most_used` (oder)

1.1.3 Show-Mode

Der **Show-Mode** ist ein Nebenprodukt der Implementierung des **PicoC-Compilers**. Dieser **Mode** wurde eigentlich nur implementiert, um beim **Testen** des PicoC-Compilers **Bugs** bei der Generierung des **RETI-Code** zu finden, indem im Terminal eine **virtuelle RETI-CPU** angezeigt wird, welches den **kompletten Zustand** einer virtuell ausgeführten RETI mit allen **Registern**, **SRAM**, **UART**, **EPROM** und einigen **weiteren Informationen** anzeigt. Allerdings bringt die Möglichkeit des **Show-Mode** die **RETI-Befehle** des übersetzten Programmes in Ausführung zu sehen auch einen großen **Lerneffekt** mit sich, weshalb der **Show-Mode** noch **weiterentwickelt** wurde, sodass auch **Studenten** ihn auf unkomplizierte Weise nutzen können.

1.2 Qualitätssicherung



(1.0.1)

1.3 Erweiterungsideen

Literatur