universität innsbruck

ARM Simulator, Interpreter und Debugger als Webanwendung

Leopold-Franzens-Universität Innsbruck
Institut für Informatik
Security and Privacy Lab

Dominik Zangerl
Betreuer: Alexander Schlögl

Innsbruck, 8. Juli 2021

1 Motivation

Das Ziel meiner Bachelorarbeit ist es eine Webanwendung zu entwickeln, mit der die ARMv5 Entwicklungsumgebung simuliert wird. ARMv5 [2] wird im ersten Semester als Beispiel für eine Befehlssatzarchitektur unterrichtet. Studierende sollen ihre eigenen Programme in Assembler [3] schreiben und diese dann auf einer ARMv5 Architektur ausführen. Diese Entwicklungsumgebung wird zurzeit mit verschiedenen Linux-Programmen simuliert. Die GNU Toolchain für die ARM Cortex-A Architektur [1] wird für das Kompilieren und Linken der Assembler-Dateien verwendet. Das kompilierte ARM-Programm läuft dann nicht auf der Architektur des Hostrechners, sondern wird mit Hilfe des QEMU User-Space-Emulators [6] ausgeführt. Dieser Prozess kann vereinfacht werden, indem man die Toolchain in einer virtuellen Maschine oder dem Windows Subsystem for Linux [4] installiert und sich für die Befehlskette ein Skript schreibt.

Das größere Problem bei dieser Toolchain ist die Fehlersuche und das Debugging des Programms. Den Fehler auf eine bestimmte Instruktion oder ein Registers zurückzuführen nimmt oft die größte Zeit in Anspruch. Der ARM-Emulator kann zusammen mit dem GNU Debugger [5] verwendet werden, welcher auch die Inhalte der Register anzeigen kann. Dies bedeutet jedoch häufig einen großen Zeitaufwand um alles aufzusetzen. Auch das Arbeiten mit Debuggern, besonders auf der Kommandozeile, könnte vielen noch nicht geläufig sein.

An dieser Stelle greift dieses Bachelorprojekt ein und versucht die ARMv5 Entwicklungsumgebung inklusive Debugging mit einer Webanwendung zu simulieren. Benutzer:innen schreiben den ARM-Code direkt in die Webanwendung, welcher dann auf einer simulierten CPU und simuliertem Hauptspeicher direkt im Browser ausgeführt wird. Die Inhalte der Register, des Stacks und Teile des Hauptspeichers werden dauerhaft angezeigt und helfen Benutzer:innen bei der Fehlerbehebung, da sie sofort sehen, an welcher Stelle ein ungewünschter Wert in ein Register geschrieben wird. Zusammen mit den Funktionen eines Debuggers, wie zeilenweise Abarbeitung des Codes oder setzen von Breakpoints, wird den Studierenden die zeitaufwändigste Arbeit abgenommen und sie können sich auf den wichtigen Teil konzentrieren, nämlich das Schreiben und Verstehen von ARM-Assembler Code.

2 Theorie

2.1 ARMv5

2.2 Parsing Expression Grammatik und tsPEG

3 Implementation

3.1 Implementierte Instruktionen

ADD

ADC

...

3.2 Codeausführung und Debugger

3.3 Parser

4 Evaluation

5 Zusammenfassung und Ausblick

Literatur

[1] ARM Limited. GNU Toolchain for ARM processors. Zugegriffen am: 04.03.2021. `https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain`.

[2] ARM Limited. ARMv5 Architecture Reference Manual - Issue I, 2005.

[3] P. Knaggs. ARM Assembly Language Programming, 2016.

[4] Microsoft. Windows Subsystem for Linux. Zugegriffen am: 04.03.2021. `https://docs.microsoft.com/en-us/windows/wsl/install-win10`.

[5] The GNU Project. GDB: The GNU Project Debugger. Zugegriffen am: 04.03.2021. `https://www.gnu.org/software/gdb/`.

[6] The QEMU Project Developers. QEMU User Mode Emulation. Zugegriffen am: 04.03.2021. `https://qemu.readthedocs.io/en/latest/user/index.html`.