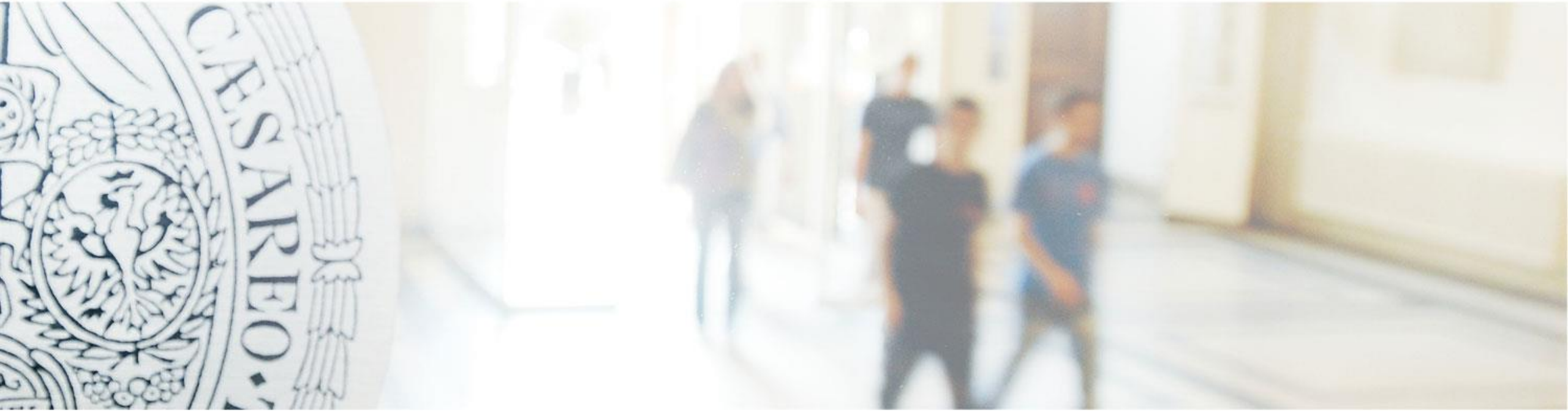
universität innsbruck



ARM Simulator, Interpreter und Debugger als Webanwendung

Initialpräsentation

Zangerl Dominik
Betreuer: Alexander Schlögl

Gliederung

- Motivation
- Implementation und Technologien
  - Typescript
  - Parser
  - Simulator und Debugger
  - Frontend/React
- Vorgehensweise und Zeitplan
- Voraussetzungen für finale Implementierung
- Referenzen

ARMv5 im ersten Semester

- ARMv5 [2] als Beispiel einer Befehlssatzarchitektur

ARMv5 im ersten Semester

- ARMv5 [2] als Beispiel einer Befehlssatzarchitektur
- Schreiben von Assembler-Programme und Ausführung auf einer ARMv5 Architektur

ARMv5 im ersten Semester

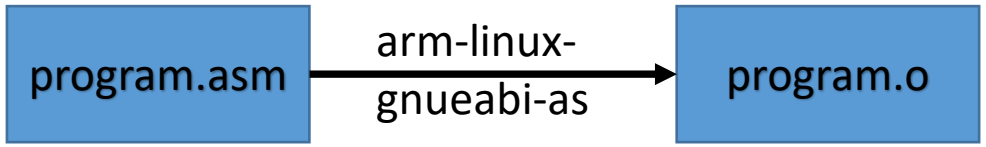
- ARMv5 [2] als Beispiel einer Befehlssatzarchitektur
- Schreiben von Assembler-Programme und Ausführung auf einer ARMv5 Architektur
- Simulation mit GNU Toolchain [1]:

ARMv5 im ersten Semester

- ARMv5 [2] als Beispiel einer Befehlssatzarchitektur
- Schreiben von Assembler-Programme und Ausführung auf einer ARMv5 Architektur
- Simulation mit GNU Toolchain [1]:

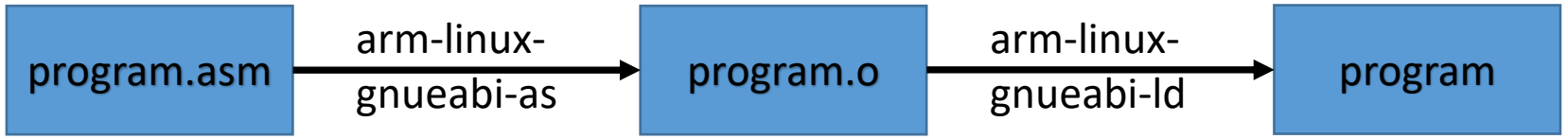
ARMv5 im ersten Semester

- ARMv5 [2] als Beispiel einer Befehlssatzarchitektur
- Schreiben von Assembler-Programme und Ausführung auf einer ARMv5 Architektur
- Simulation mit GNU Toolchain [1]:



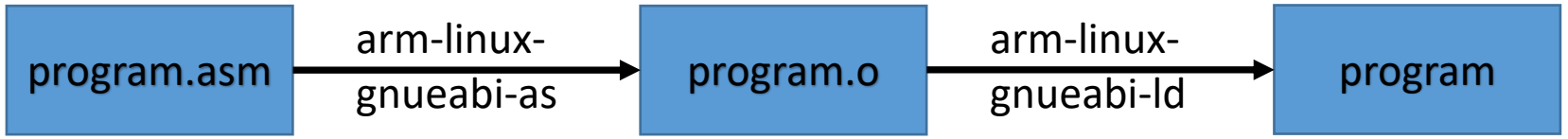
ARMv5 im ersten Semester

- ARMv5 [2] als Beispiel einer Befehlssatzarchitektur
- Schreiben von Assembler-Programme und Ausführung auf einer ARMv5 Architektur
- Simulation mit GNU Toolchain [1]:



ARMv5 im ersten Semester

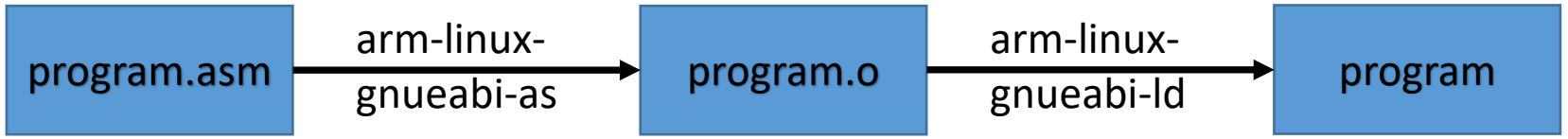
- ARMv5 [2] als Beispiel einer Befehlssatzarchitektur
- Schreiben von Assembler-Programme und Ausführung auf einer ARMv5 Architektur
- Simulation mit GNU Toolchain [1]:



- Ausführen mit QEMU User-Space-Emulator [9]

ARMv5 im ersten Semester

- ARMv5 [2] als Beispiel einer Befehlssatzarchitektur
- Schreiben von Assembler-Programme und Ausführung auf einer ARMv5 Architektur
- Simulation mit GNU Toolchain [1]:



- Ausführen mit QEMU User-Space-Emulator [9]
- Vereinfachung mit Skript und Ausführung über virtuelle Maschine oder WSL [6]

Debugging

- Größter Zeitaufwand bei Fehlersuche im Programm

Debugging

- Größter Zeitaufwand bei Fehlersuche im Programm
- Kann zusammen mit dem Gnu Debugger [8] verwendet werden:

Debugging

- Größter Zeitaufwand bei Fehlersuche im Programm
- Kann zusammen mit dem Gnu Debugger [8] verwendet werden:

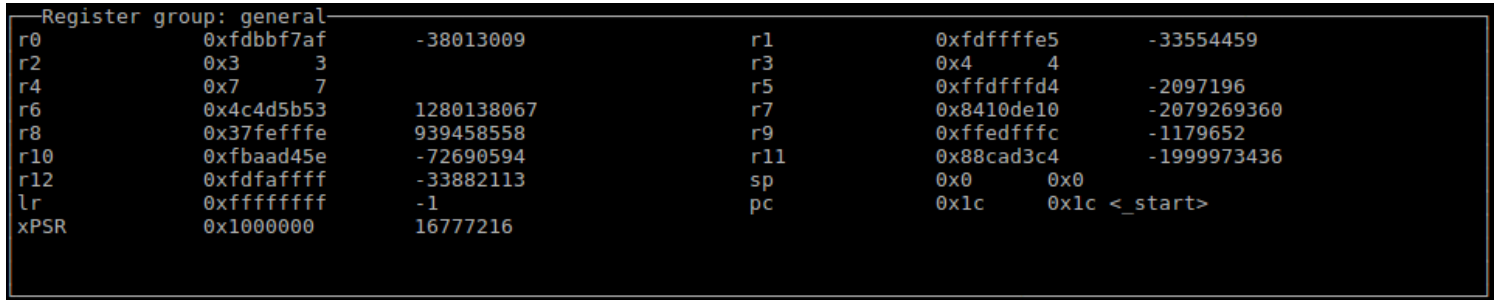


Bild: Use GDB on an ARM assembly program [7]

Debugging

- Größter Zeitaufwand bei Fehlersuche im Programm
- Kann zusammen mit dem Gnu Debugger [8] verwendet werden:

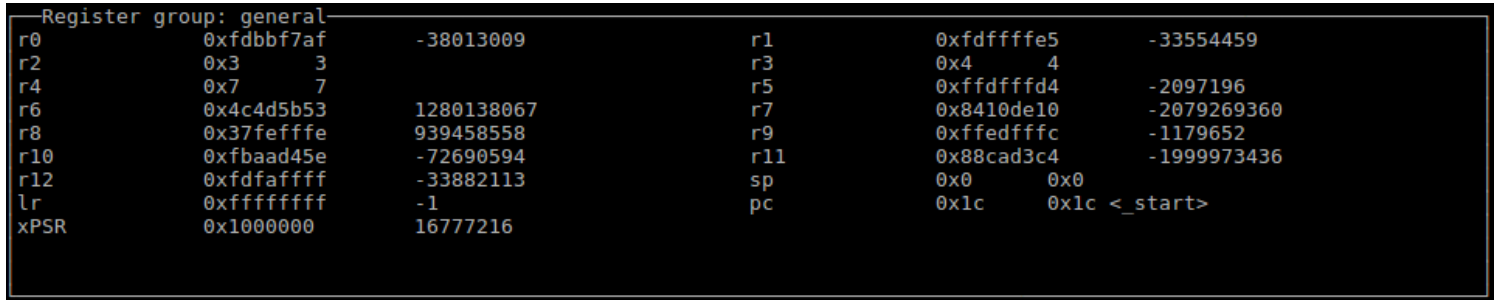


Bild: Use GDB on an ARM assembly program [7]

- Arbeiten mit Debuggern im ersten Semester oft schwierig

Debugging

- Größter Zeitaufwand bei Fehlersuche im Programm
- Kann zusammen mit dem Gnu Debugger [8] verwendet werden:

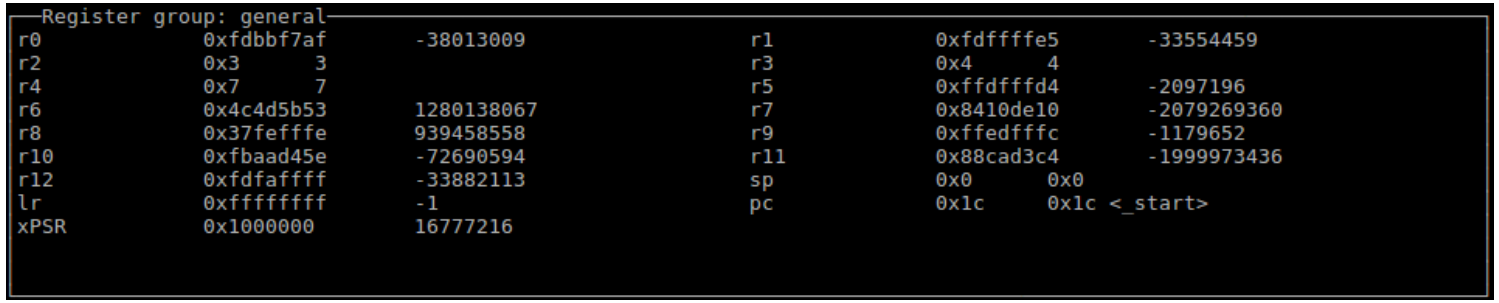


Bild: Use GDB on an ARM assembly program [7]

- Arbeiten mit Debuggern im ersten Semester oft schwierig
- Großer Zeitaufwand zusammen mit Aufsetzen der Toolchain

ARMv5 Umgebung und Debugging als Webanwendung

- Bachelorprojekt: Simuliere ARMv5 Entwicklungsumgebung und Debugger als Webanwendung

ARMv5 Umgebung und Debugging als Webanwendung

- Bachelorprojekt: Simuliere ARMv5 Entwicklungsumgebung und Debugger als Webanwendung
- ARMv5 Entwicklungsumgebung

ARMv5 Umgebung und Debugging als Webanwendung

- Bachelorprojekt: Simuliere ARMv5 Entwicklungsumgebung und Debugger als Webanwendung
- ARMv5 Entwicklungsumgebung
  - Simulierte CPU und Hauptspeicher

ARMv5 Umgebung und Debugging als Webanwendung

- Bachelorprojekt: Simuliere ARMv5 Entwicklungsumgebung und Debugger als Webanwendung
- ARMv5 Entwicklungsumgebung
  - Simulierte CPU und Hauptspeicher
  - Assembler-Code direkt in Anwendung schreiben und ausführen→ ersetzt Toolchain

ARMv5 Umgebung und Debugging als Webanwendung

- Bachelorprojekt: Simuliere ARMv5 Entwicklungsumgebung und Debugger als Webanwendung
- ARMv5 Entwicklungsumgebung
  - Simulierte CPU und Hauptspeicher
  - Assembler-Code direkt in Anwendung schreiben und ausführen→ ersetzt Toolchain
  - Dauerhafte Anzeige von Registern und Stacks

ARMv5 Umgebung und Debugging als Webanwendung

- Bachelorprojekt: Simuliere ARMv5 Entwicklungsumgebung und Debugger als Webanwendung
- ARMv5 Entwicklungsumgebung
  - Simulierte CPU und Hauptspeicher
  - Assembler-Code direkt in Anwendung schreiben und ausführen→ ersetzt Toolchain
  - Dauerhafte Anzeige von Registern und Stacks
- Debugger

ARMv5 Umgebung und Debugging als Webanwendung

- Bachelorprojekt: Simuliere ARMv5 Entwicklungsumgebung und Debugger als Webanwendung
- ARMv5 Entwicklungsumgebung
  - Simulierte CPU und Hauptspeicher
  - Assembler-Code direkt in Anwendung schreiben und ausführen→ ersetzt Toolchain
  - Dauerhafte Anzeige von Registern und Stacks
- Debugger
  - Breakpoints

ARMv5 Umgebung und Debugging als Webanwendung

- Bachelorprojekt: Simuliere ARMv5 Entwicklungsumgebung und Debugger als Webanwendung
- ARMv5 Entwicklungsumgebung
  - Simulierte CPU und Hauptspeicher
  - Assembler-Code direkt in Anwendung schreiben und ausführen→ ersetzt Toolchain
  - Dauerhafte Anzeige von Registern und Stacks
- Debugger
  - Breakpoints
  - Zeilenweise Abarbeitung

TypeScript

- TypeScript [5] ist eine Sprache von Microsoft, die auf JavaScript aufbaut

TypeScript

- TypeScript [5] ist eine Sprache von Microsoft, die auf JavaScript aufbaut
- JavaScript überprüft nicht, ob Typen korrekt zugewiesen werden

TypeScript

- TypeScript [5] ist eine Sprache von Microsoft, die auf JavaScript aufbaut
- JavaScript überprüft nicht, ob Typen korrekt zugewiesen werden
  - TypeScript fügt statische Typisierung und Klassen hinzu

TypeScript

- TypeScript [5] ist eine Sprache von Microsoft, die auf JavaScript aufbaut
- JavaScript überprüft nicht, ob Typen korrekt zugewiesen werden
  - TypeScript fügt statische Typisierung und Klassen hinzu
- Fertiger Code wird zu einem ausführbaren JavaScript Programm kompiliert

TypeScript

- TypeScript [5] ist eine Sprache von Microsoft, die auf JavaScript aufbaut
- JavaScript überprüft nicht, ob Typen korrekt zugewiesen werden
  - TypeScript fügt statische Typisierung und Klassen hinzu
- Fertiger Code wird zu einem ausführbaren JavaScript Programm kompiliert
- Backend:

TypeScript

- TypeScript [5] ist eine Sprache von Microsoft, die auf JavaScript aufbaut
- JavaScript überprüft nicht, ob Typen korrekt zugewiesen werden
  - TypeScript fügt statische Typisierung und Klassen hinzu
- Fertiger Code wird zu einem ausführbaren JavaScript Programm kompiliert
- Backend:
  - Simulierte CPU
  - Parser
  - Debugger

Parser

- Erzeugen eines Parsers auf einer vordefinierten Grammatik mit tsPEG [3]

Parser

- Erzeugen eines Parsers auf einer vordefinierten Grammatik mit tsPEG [3]

```
start := instruction | data
instruction := instruction1 | instruction2

instruction1 := inst='MOV' '[ \t]+' reg='r[0-9]+' ', #' immediate='[0-9]+'
instruction2 := inst='CMP' '[ \t]+' reg1='r[0-9]+' ', ' reg2='r[0-9]+'

data := '.data\n' label='.[a-zA-Z]+' '[ \t]+' '\"' data='[a-zA-Z0-9\n]*' '\"'
```

- Beispielgrammatik, die 2 Instruktionen und einem Datenbereich erkennt

Parser

- Erzeugen eines Parsers auf einer vordefinierten Grammatik mit tsPEG [3]

```
start := instruction | data
instruction := instruction1 | instruction2

instruction1 := inst='MOV' '[ \t]+' reg='r[0-9]+' ', #' immediate='[0-9]+'
instruction2 := inst='CMP' '[ \t]+' reg1='r[0-9]+' ', ' reg2='r[0-9]+'

data := '.data\n' label='.[a-zA-Z]+' '[ \t]+' '\"' data='[a-zA-Z0-9\n]*' '\"'
```

- Beispielgrammatik, die 2 Instruktionen und einem Datenbereich erkennt
- Speichern der wichtigen Werte mit **inst='MOV'** oder **reg='r[0-9]+'**

Parser

- Erzeugen eines Parsers auf einer vordefinierten Grammatik mit tsPEG [3]

```
start := instruction | data
instruction := instruction1 | instruction2

instruction1 := inst='MOV' '[ \t]+' reg='r[0-9]+' ', #' immediate='[0-9]+'
instruction2 := inst='CMP' '[ \t]+' reg1='r[0-9]+' ', ' reg2='r[0-9]+'

data := '.data\n' label='.[a-zA-Z]+' '[ \t]+' '\"' data='[a-zA-Z0-9\n]*' '\"'
```

- Beispielgrammatik, die 2 Instruktionen und einem Datenbereich erkennt
- Speichern der wichtigen Werte mit **inst='MOV'** oder **reg='r[0-9]+'**
- Weitergabe an CPU, die Instruktionen ausführt

Debugger

- Anzeige von Registern, Stack und Teilen des Hauptspeichers

Debugger

- Anzeige von Registern, Stack und Teilen des Hauptspeichers
- Zeilenweise Abarbeitung und Setzen von Breakpoints

Debugger

- Anzeige von Registern, Stack und Teilen des Hauptspeichers
- Zeilenweise Abarbeitung und Setzen von Breakpoints
- Funktionen des Debuggers:
  - *Step Into* – Nächste Zeile + Springen in eine mögl. Subroutine

Debugger

- Anzeige von Registern, Stack und Teilen des Hauptspeichers
- Zeilenweise Abarbeitung und Setzen von Breakpoints
- Funktionen des Debuggers:
  - *Step Into* – Nächste Zeile + Springen in eine mögl. Subroutine
  - *Step Over* – Nächste Zeile + Ausführen einer mögl. Subroutine

Debugger

- Anzeige von Registern, Stack und Teilen des Hauptspeichers
- Zeilenweise Abarbeitung und Setzen von Breakpoints
- Funktionen des Debuggers:
  - *Step Into* – Nächste Zeile + Springen in eine mögl. Subroutine
  - *Step Over* – Nächste Zeile + Ausführen einer mögl. Subroutine
  - *Continue* – Ausführen bis zum nächsten Breakpoint

Debugger

- Anzeige von Registern, Stack und Teilen des Hauptspeichers
- Zeilenweise Abarbeitung und Setzen von Breakpoints
- Funktionen des Debuggers:
  - *Step Into* – Nächste Zeile + Springen in eine mögl. Subroutine
  - *Step Over* – Nächste Zeile + Ausführen einer mögl. Subroutine
  - *Continue* – Ausführen bis zum nächsten Breakpoint
  - *Step Return* – Ausführen bis zum Ende der Subroutine

Debugger

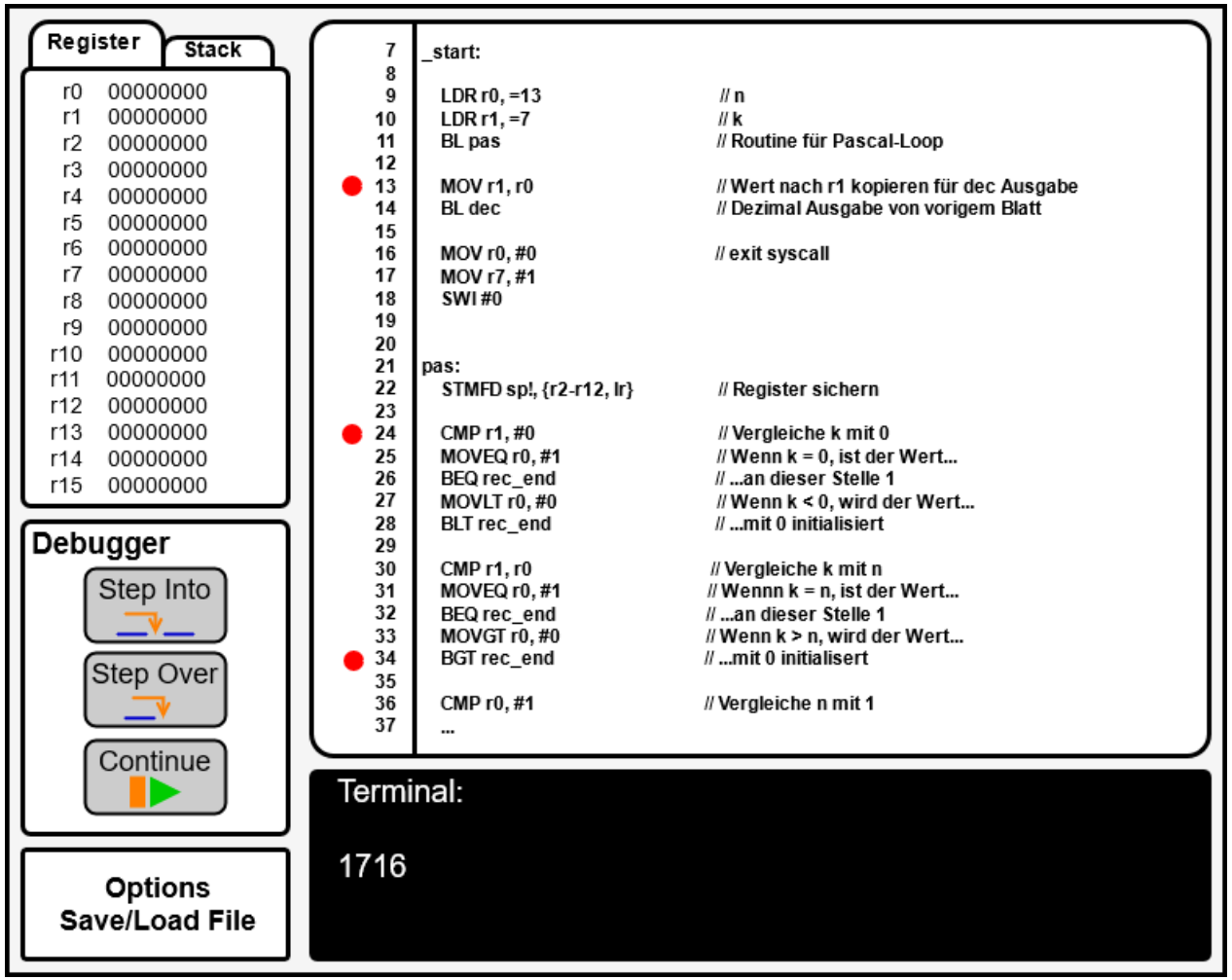
- Anzeige von Registern, Stack und Teilen des Hauptspeichers
- Zeilenweise Abarbeitung und Setzen von Breakpoints
- Funktionen des Debuggers:
  - *Step Into* – Nächste Zeile + Springen in eine mögl. Subroutine
  - *Step Over* – Nächste Zeile + Ausführen einer mögl. Subroutine
  - *Continue* – Ausführen bis zum nächsten Breakpoint
  - *Step Return* – Ausführen bis zum Ende der Subroutine
  - *Stop* – Beenden der Ausführung

React

- React [4] ist ein Webframework von Facebook um Benutzeroberflächen in JavaScript zu erstellen

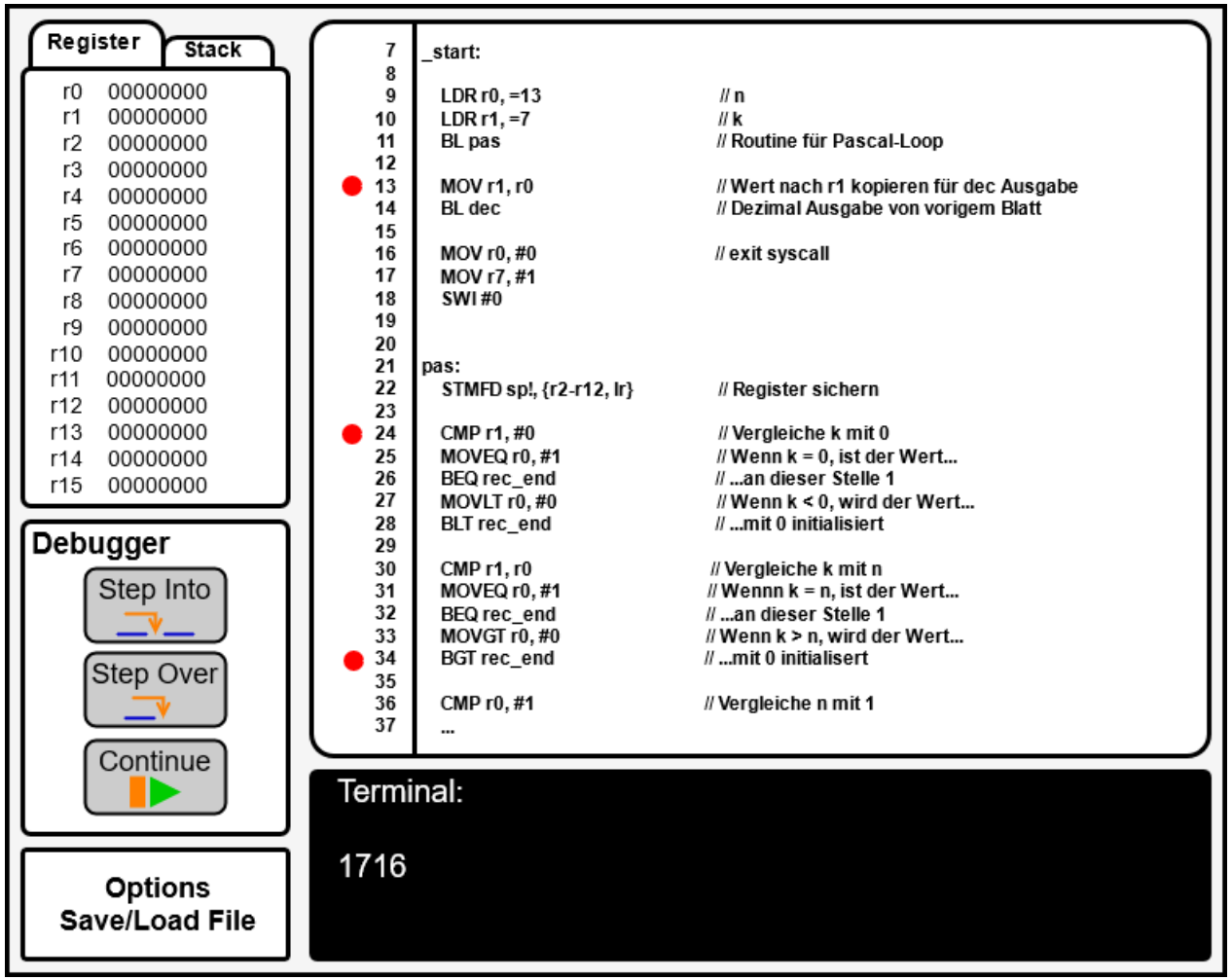
React

- React [4] ist ein Webframework von Facebook um Benutzeroberflächen in JavaScript zu erstellen
- Frontend der Webanwendung



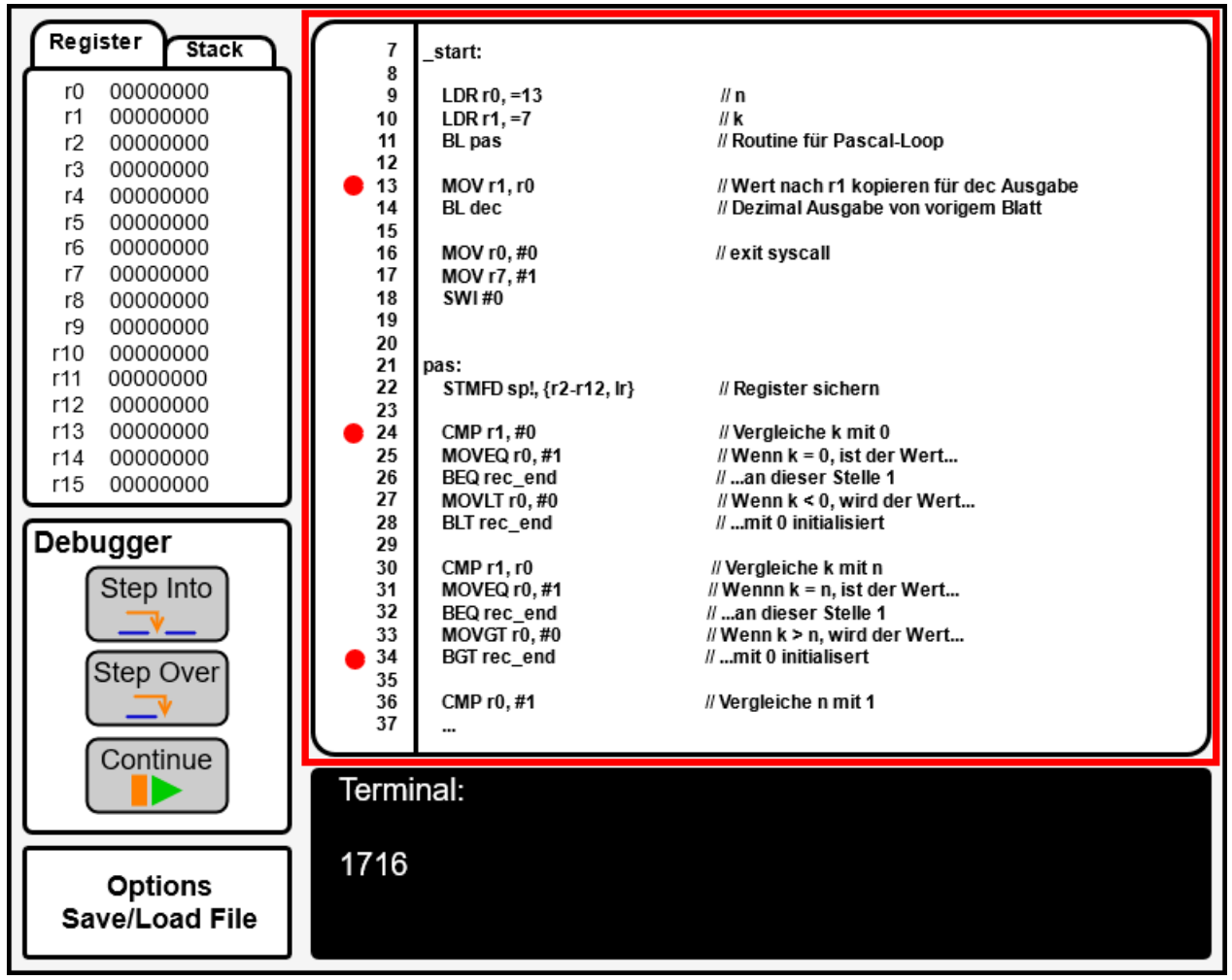
React

- React [4] ist ein Webframework von Facebook um Benutzeroberflächen in JavaScript zu erstellen
- Frontend der Webanwendung
- Visualisierung der einzelnen Komponenten:



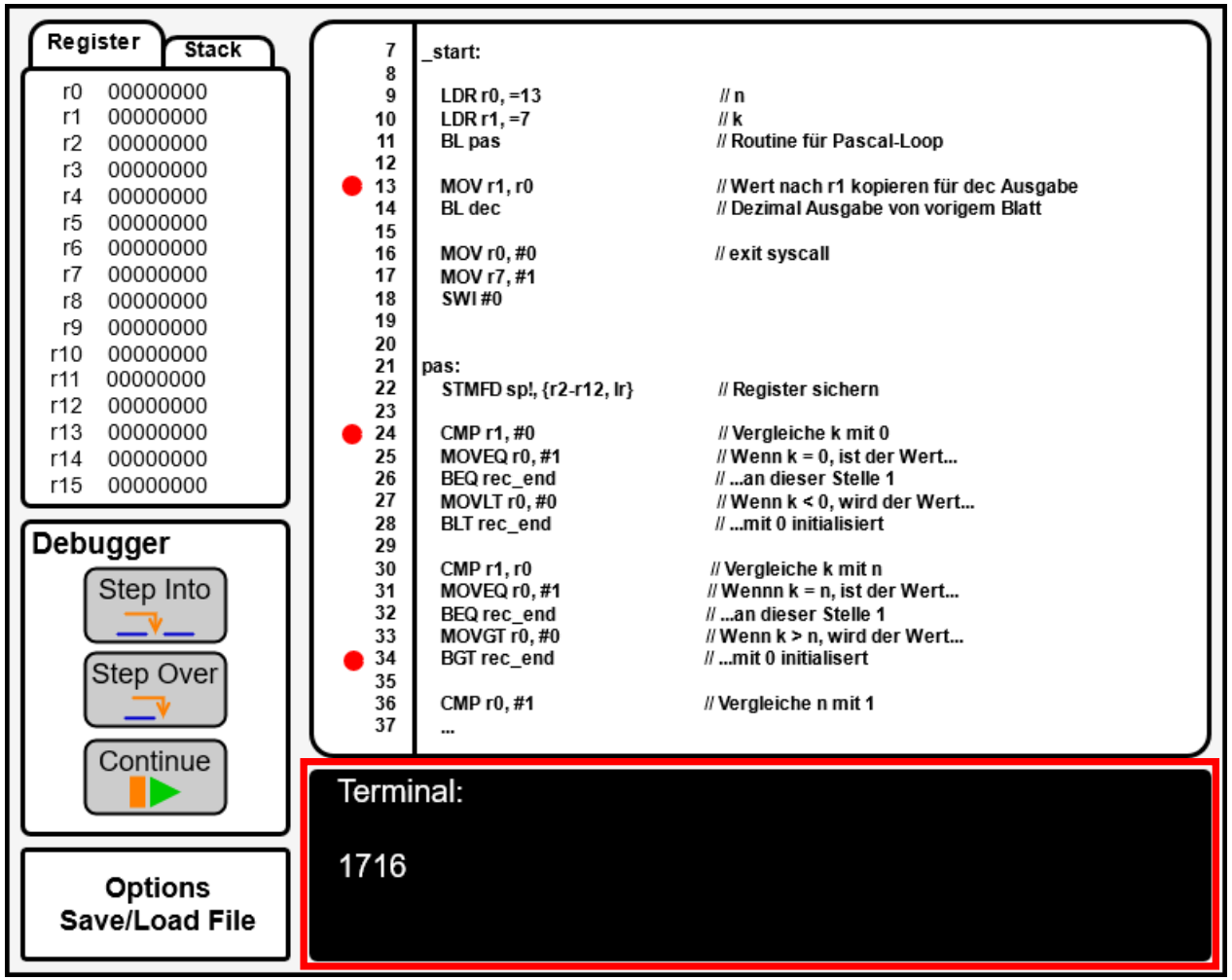
React

- React [4] ist ein Webframework von Facebook um Benutzeroberflächen in JavaScript zu erstellen
- Frontend der Webanwendung
- Visualisierung der einzelnen Komponenten:
  - Textfeld für Benutzereingabe und Setzen von Breakpoints



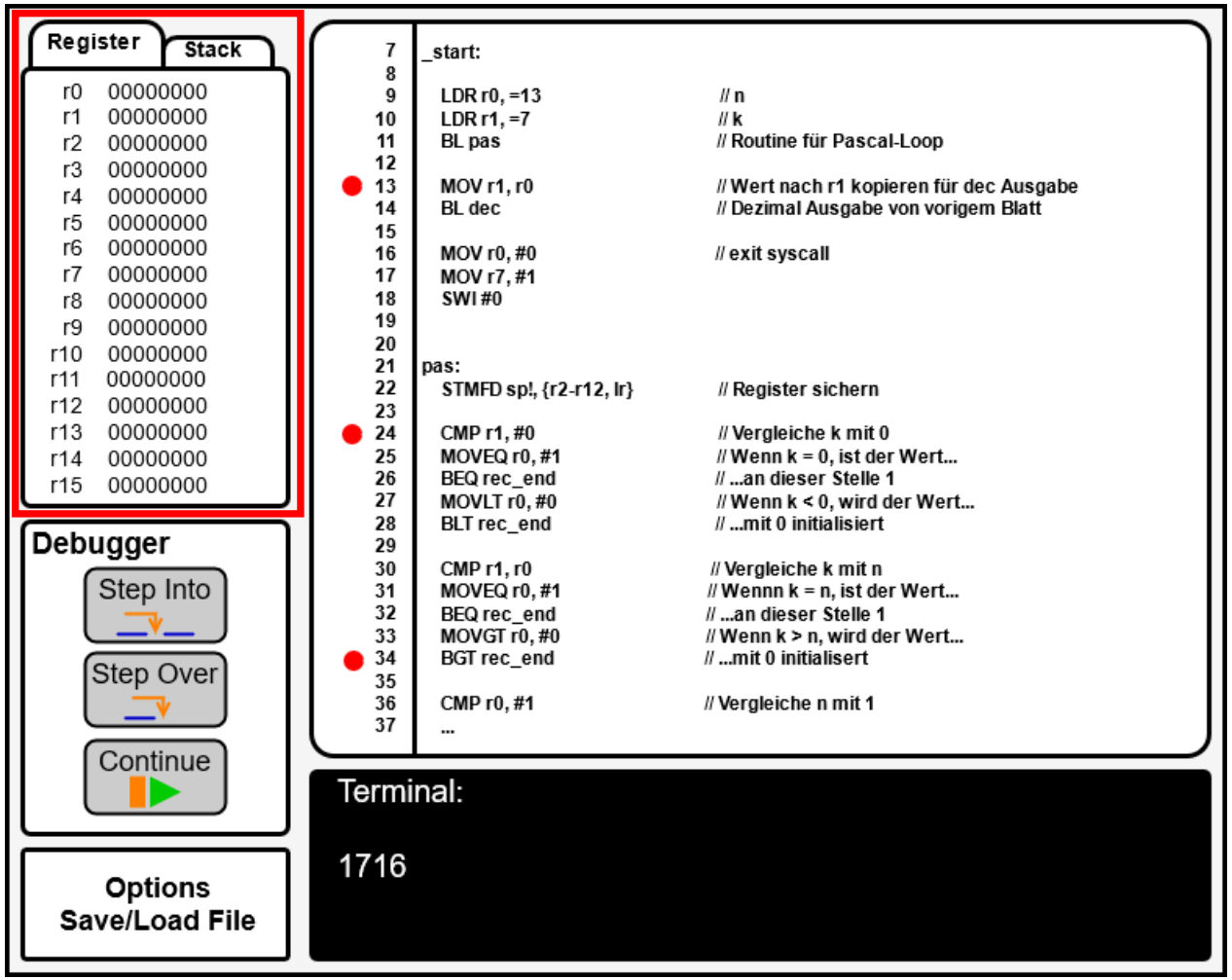
React

- React [4] ist ein Webframework von Facebook um Benutzeroberflächen in JavaScript zu erstellen
- Frontend der Webanwendung
- Visualisierung der einzelnen Komponenten:
  - Textfeld für Benutzereingabe und Setzen von Breakpoints
  - Terminal für Ausgabe von Ergebnissen und Fehlern/Warnungen



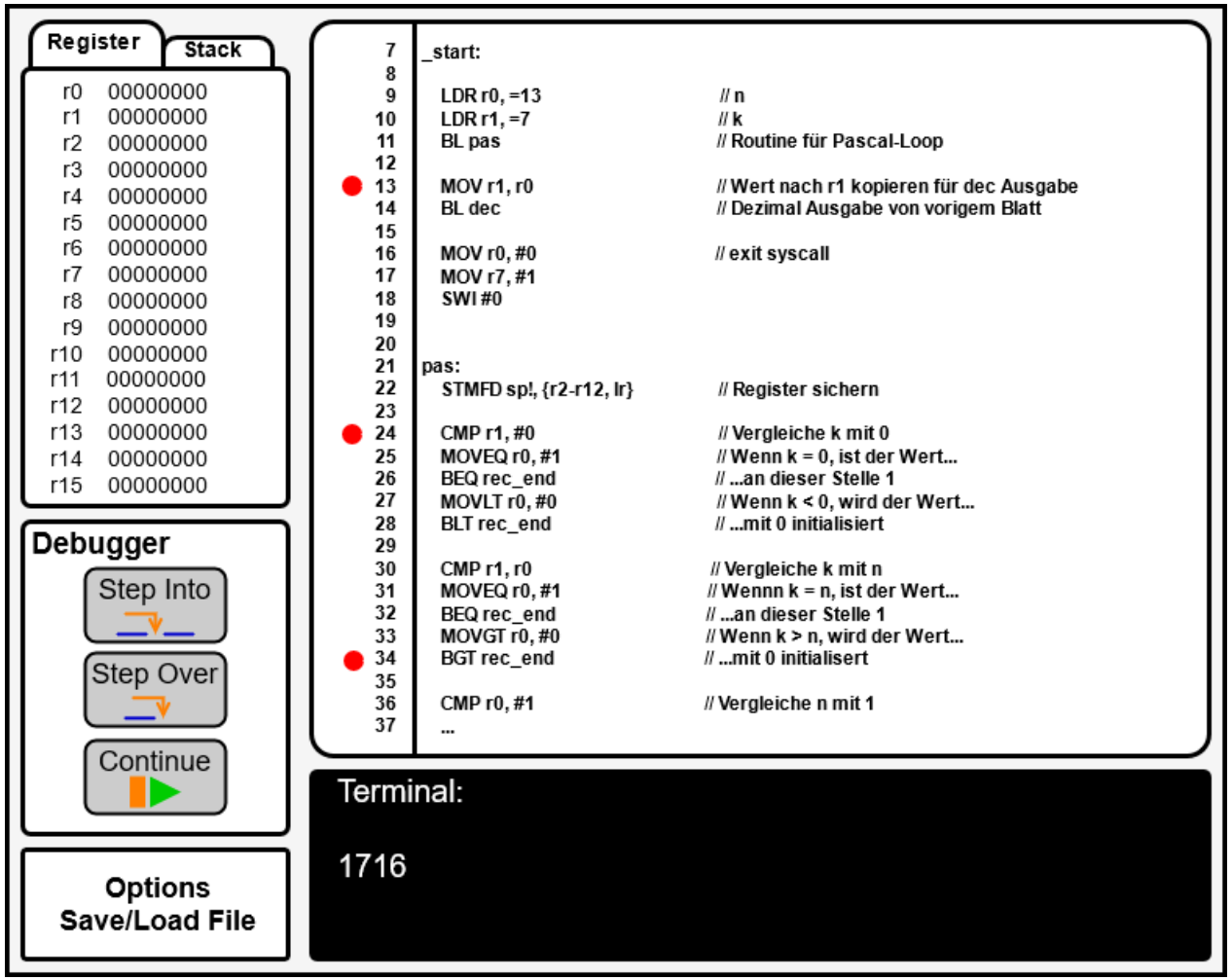
React

- React [4] ist ein Webframework von Facebook um Benutzeroberflächen in JavaScript zu erstellen
- Frontend der Webanwendung
- Visualisierung der einzelnen Komponenten:
  - Textfeld für Benutzereingabe und Setzen von Breakpoints
  - Terminal für Ausgabe von Ergebnissen und Fehlern/Warnungen
  - Zustand des Programms, wie Inhalt der Register und des Stack



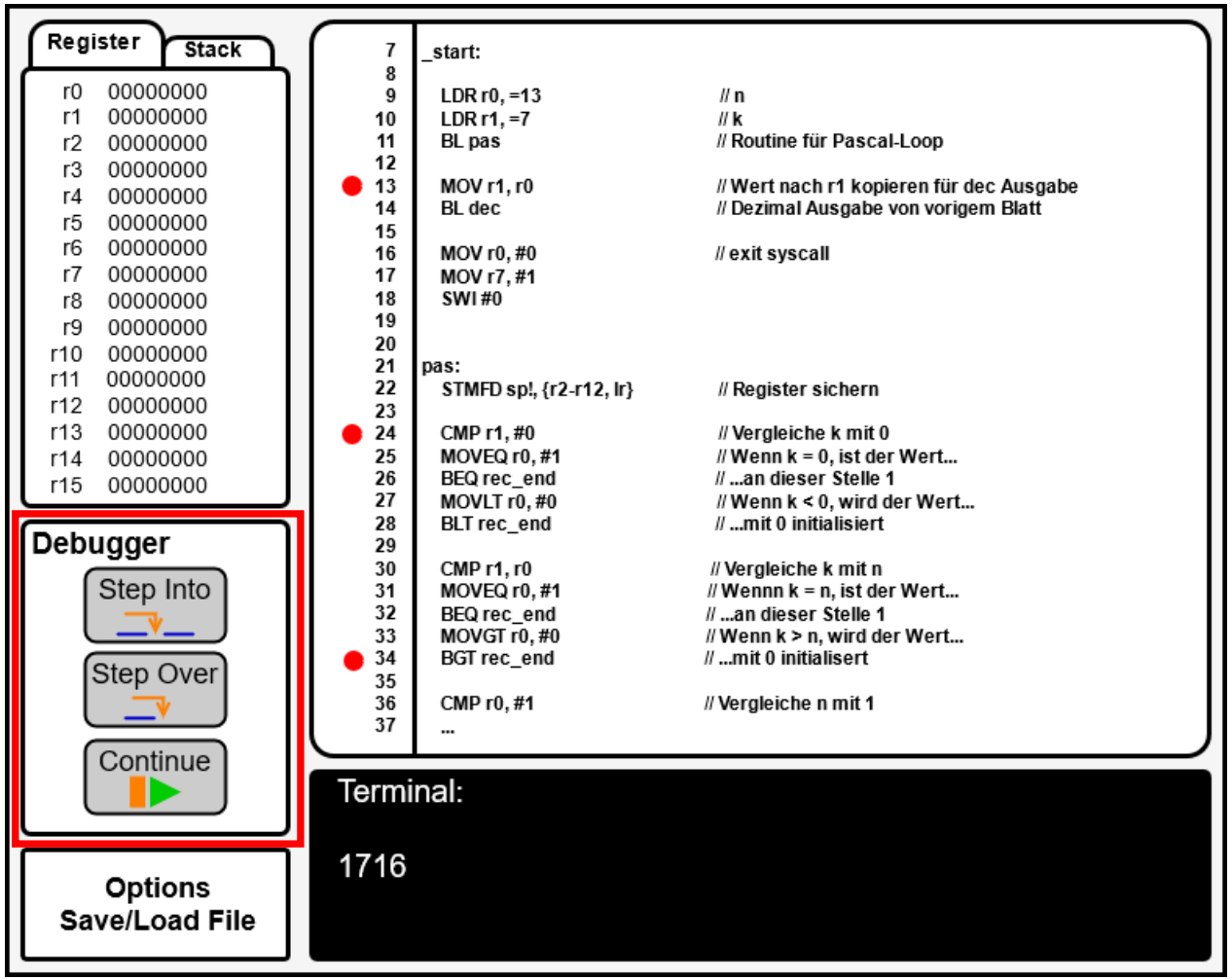
React

- Visualisierung der einzelnen Komponenten:
  - Textfeld für Benutzereingabe und Setzen von Breakpoints
  - Terminal für Ausgabe von Ergebnissen und Fehlern/Warnungen
  - Zustand des Programms, wie Inhalt der Register und des Stack



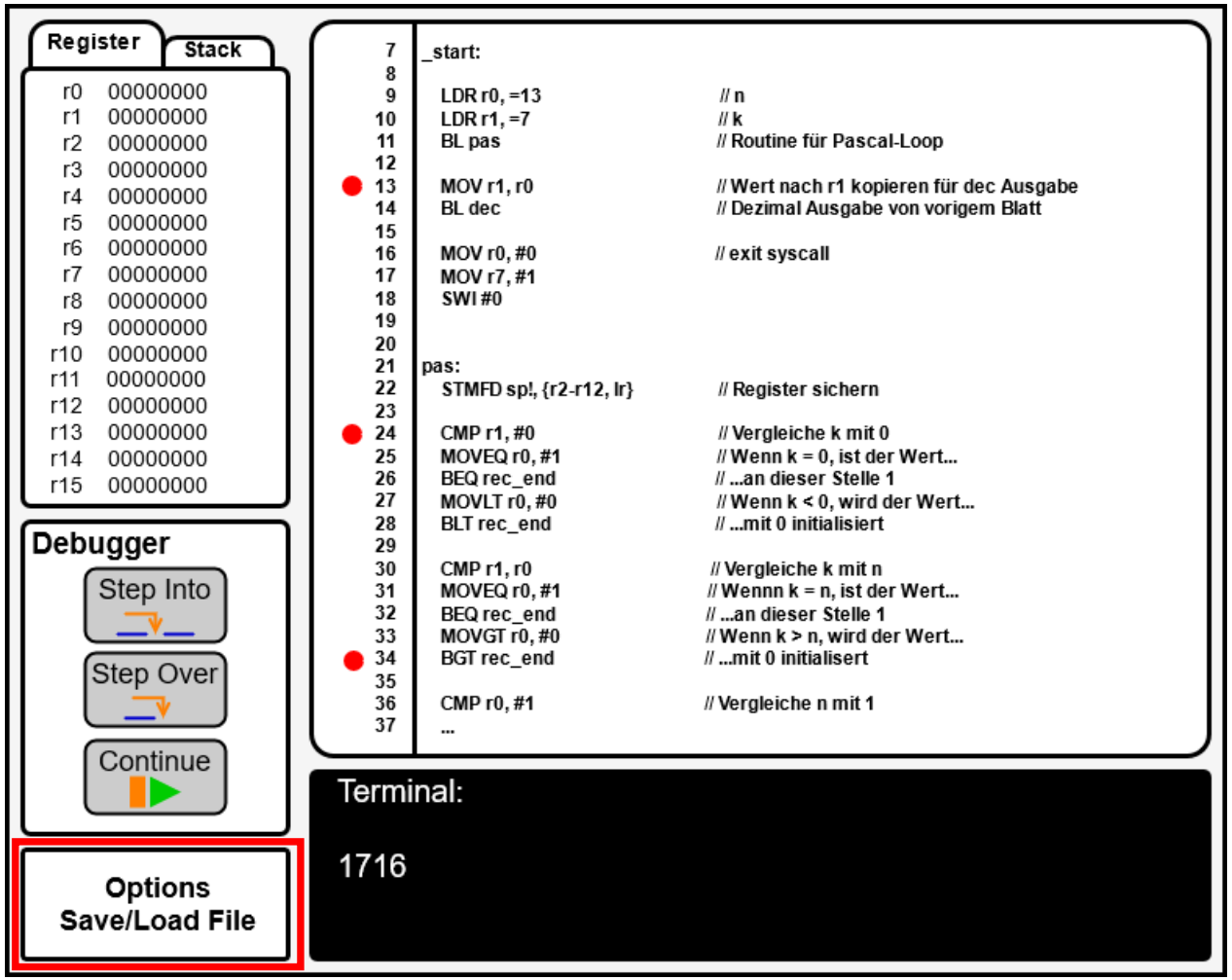
React

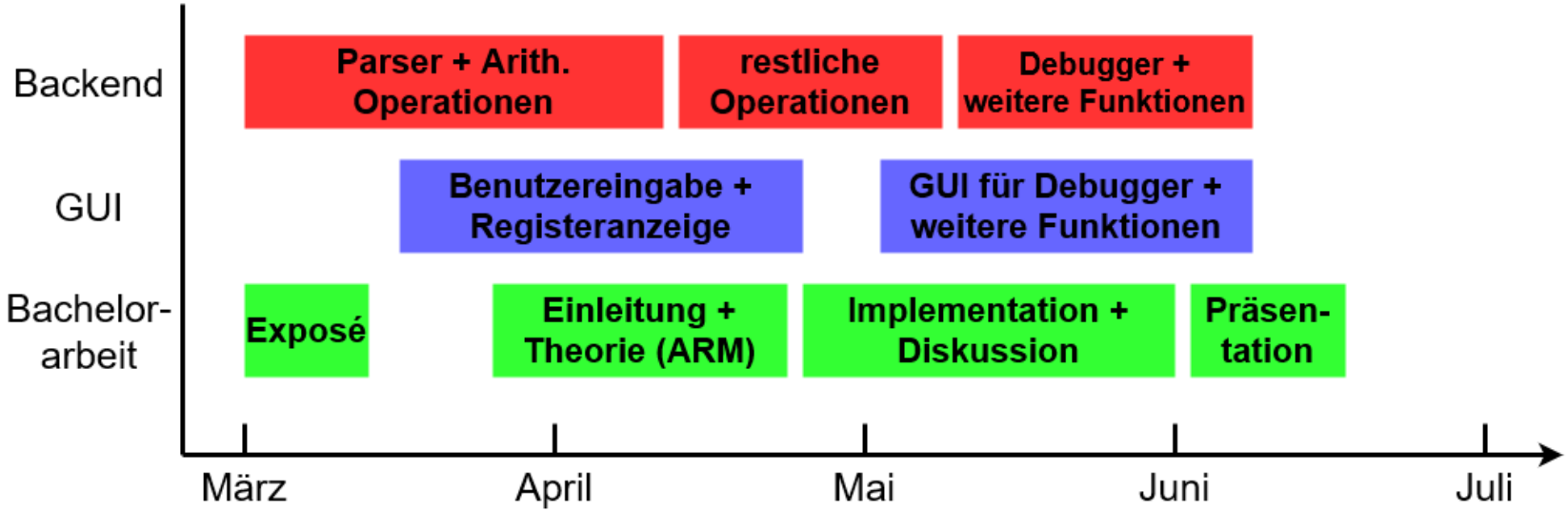
- Visualisierung der einzelnen Komponenten:
  - Textfeld für Benutzereingabe und Setzen von Breakpoints
  - Terminal für Ausgabe von Ergebnissen und Fehlern/Warnungen
  - Zustand des Programms, wie Inhalt der Register und des Stack
  - Funktionen des Debuggers



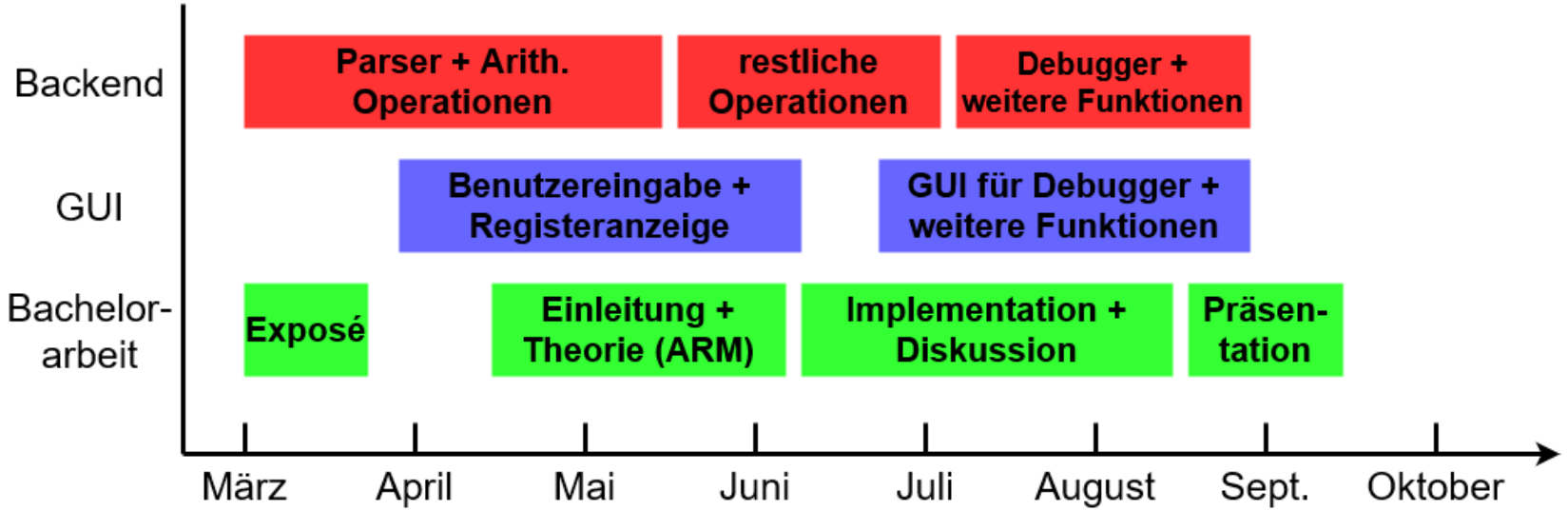
React

- Visualisierung der einzelnen Komponenten:
  - Textfeld für Benutzereingabe und Setzen von Breakpoints
  - Terminal für Ausgabe von Ergebnissen und Fehlern/Warnungen
  - Zustand des Programms, wie Inhalt der Register und des Stack
  - Funktionen des Debuggers
  - Weitere Optionen

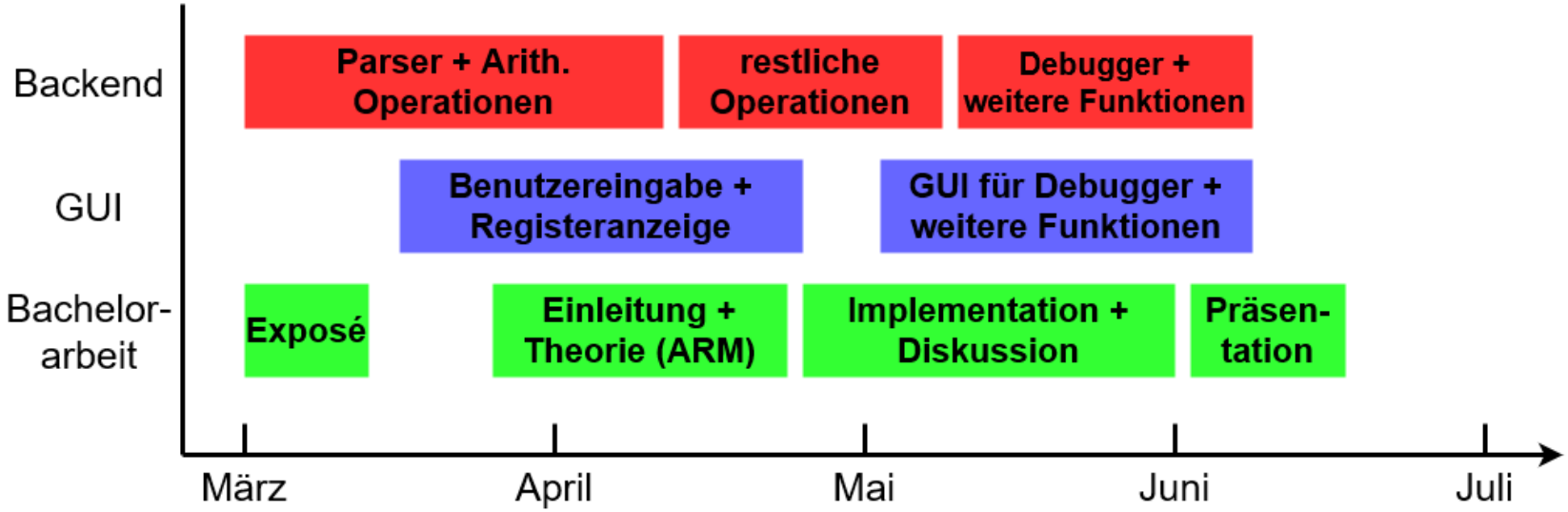




- Gesamte Zeit für Bachelorarbeit zur Verfügung – Zeitplan für Präsentation dieses Semester

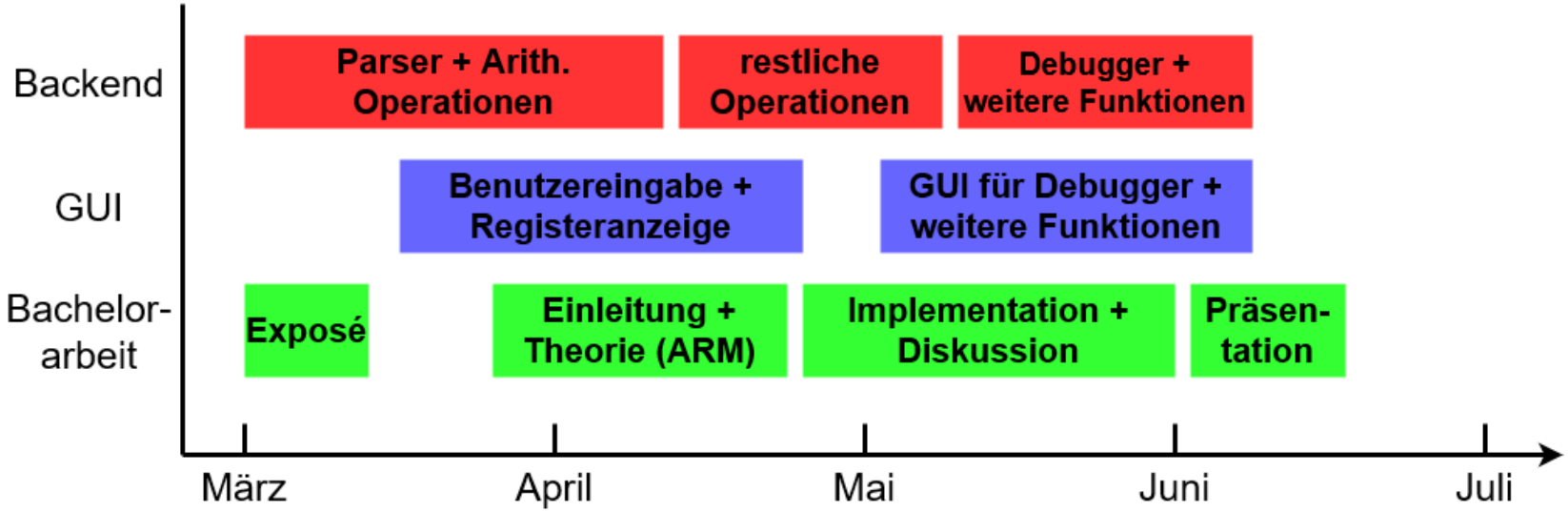


- Gesamte Zeit für Bachelorarbeit zur Verfügung – Zeitplan für Präsentation dieses Semester
  - Falls trotzdem zu viel Arbeit – Präsentation Anfang des nächsten Semesters



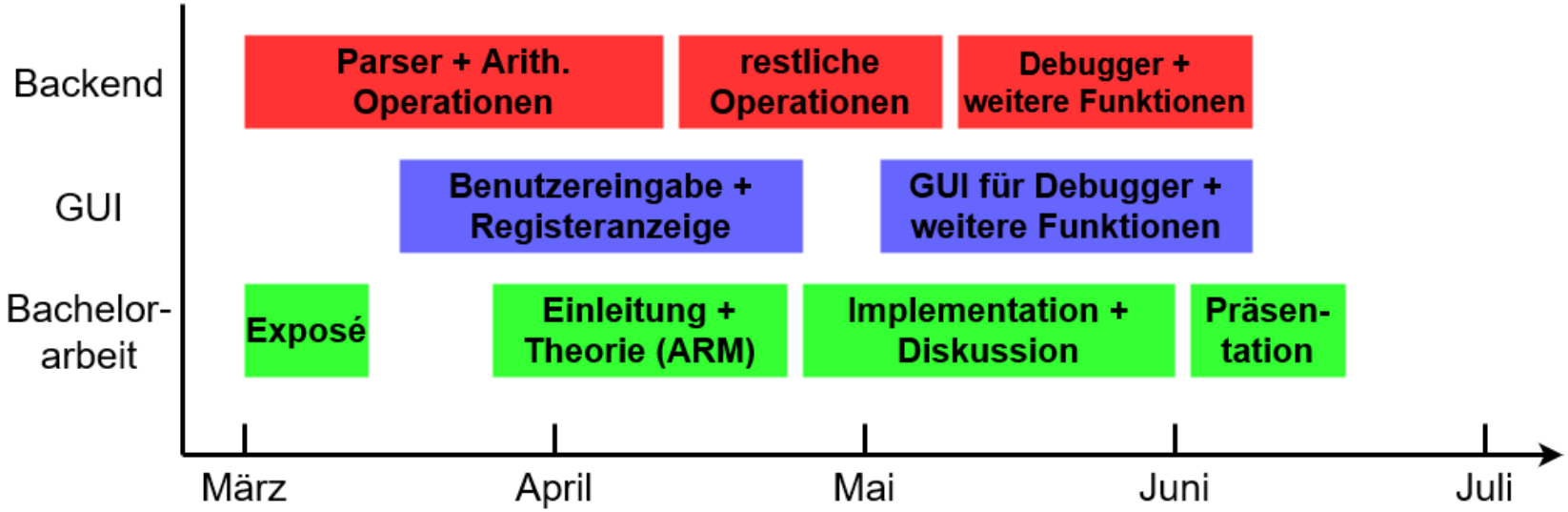
- Gesamte Zeit für Bachelorarbeit zur Verfügung – Zeitplan für Präsentation dieses Semester
  - Falls trotzdem zu viel Arbeit – Präsentation Anfang des nächsten Semesters

1. Beginn mit Parser und arithmetischen Operation + Visualisierung zum Testen



- Gesamte Zeit für Bachelorarbeit zur Verfügung – Zeitplan für Präsentation dieses Semester
  - Falls trotzdem zu viel Arbeit – Präsentation Anfang des nächsten Semesters

1. Beginn mit Parser und arithmetischen Operation + Visualisierung zum Testen
2. Restliche ARMv5 Instruktionen und Beginn Theorie der Bachelorarbeit



- Gesamte Zeit für Bachelorarbeit zur Verfügung – Zeitplan für Präsentation dieses Semester
  - Falls trotzdem zu viel Arbeit – Präsentation Anfang des nächsten Semesters

1. Beginn mit Parser und arithmetischen Operation + Visualisierung zum Testen
2. Restliche ARMv5 Instruktionen und Beginn Theorie der Bachelorarbeit
3. Debugger und weitere Funktionen (Speicher/Laden von Dateien, ...)

Voraussetzungen:

- Die in der Vorlesung vorgestellten bzw. für das Proseminar benötigten ARMv5-Instruktionen sind implementiert.

Voraussetzungen:

- Die in der Vorlesung vorgestellten bzw. für das Proseminar benötigten ARMv5-Instruktionen sind implementiert.
- Die Webanwendung weißt eine Benutzeroberfläche (ähnlich Folie 10) mit Anzeige von Registern, Stack und Teilen des Hauptspeichers auf.

Voraussetzungen:

- Die in der Vorlesung vorgestellten bzw. für das Proseminar benötigten ARMv5-Instruktionen sind implementiert.
- Die Webanwendung weißt eine Benutzeroberfläche (ähnlich Folie 10) mit Anzeige von Registern, Stack und Teilen des Hauptspeichers auf.
- Der Debugger implementiert die auf Folie 8 beschriebenen Funktionen.

Voraussetzungen:

- Die in der Vorlesung vorgestellten bzw. für das Proseminar benötigten ARMv5-Instruktionen sind implementiert.
- Die Webanwendung weißt eine Benutzeroberfläche (ähnlich Folie 10) mit Anzeige von Registern, Stack und Teilen des Hauptspeichers auf.
- Der Debugger implementiert die auf Folie 8 beschriebenen Funktionen.
- Die korrekte Funktionsweise wird mit den Musterlösungen der Beispiele aus dem Proseminar getestet.

Referenzen

[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] E. Davey. tsPEG: A PEG Parser Generator for TypeScript. Zugegriffen am: 04.03.2021. https://github.com/EoinDavey/tsPEG.

[4] Facebook. React. Zugegriffen am: 04.03.2021. https://reactjs.org/.

[5] Microsoft. Typescript. Zugegriffen am: 04.03.2021. https://www.typescriptlang.org/.

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

[7] J. Mossberg. Use GDB on an ARM assembly program. Zugegriffen am: 04.03.2021. https://jacobmossberg.se/posts/2017/01/17/use-gdb-on-arm-assembly-program.html

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

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

www.uibk.ac.at