Diploma Thesis

Höhere Technische Bundeslehranstalt Leonding

Abteilung für Informatik

Visulation of the NoBeard Virtual Machine

Submitted by:	**Egon Manya, 5AHIF**
Date:	**April 4, 2018**
Supervisor:	**Peter Bauer**

Declaration of Academic Honesty

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.

This paper has neither been previously submitted to another authority nor has it been published yet.

Leonding, April 4, 2018

Egon Manya

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorgelegte Diplomarbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Gedanken, die aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Leonding, am 4. April 2018

Egon Manya

Abstract

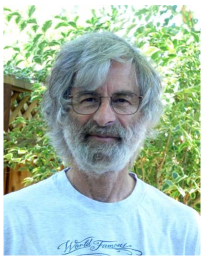
Here it is described what the thesis is all about. The abstract shall be brief and concise and its size shall not go beyond one page. Furthermore it has no chapters, sections etc. Paragraphs can be used to structure the abstract. If necessary one can also use bullet point lists but care must be taken that also in this part of the text full sentences and a clearly readable structure are required.

Concerning the content the following points shall be covered.

- *Definition of the project:* What do we currently know about the topic or on which results can the work be based? What is the goal of the project? Who can use the results of the project?
- *Implementation:* What are the tools and methods used to implement the project?
- *Results:* What is the final result of the project?

This list does not mean that the abstract must strictly follow this structure. Rather it should be understood in that way that these points shall be described such that the reader is animated to dig further into the thesis.

Finally it is required to add a representative image which describes your project best. The image here shows Leslie Lamport the inventor of LaTeX.



Zusammenfassung

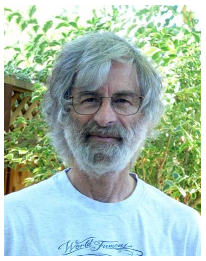
An dieser Stelle wird beschrieben, worum es in der Diplomarbeit geht. Die Zusammenfassung soll kurz und prägnant sein und den Umfang einer Seite nicht übersteigen. Weiters ist zu beachten, dass hier keine Kapitel oder Abschnitte zur Strukturierung verwendet werden. Die Verwendung von Absätzen ist zulässig. Wenn notwendig, können auch Aufzählungslisten verwendet werden. Dabei ist aber zu beachten, dass auch in der Zusammenfassung vollständige Sätze gefordert sind.

Bezüglich des Inhalts sollen folgende Punkte in der Zusammenfassung vorkommen:

- *Aufgabenstellung:* Von welchem Wissenstand kann man im Umfeld der Aufgabenstellung ausgehen? Was ist das Ziel des Projekts? Wer kann die Ergebnisse der Arbeit benutzen?
- *Umsetzung:* Welche fachtheoretischen oder -praktischen Methoden wurden bei der Umsetzung verwendet?
- *Ergebnisse:* Was ist das endgültige Ergebnis der Arbeit?

Diese Liste soll als Sammlung von inhaltlichen Punkten für die Zusammenfassung verstanden werden. Die konkrete Gliederung und Reihung der Punkte ist den Autoren überlassen. Zu beachten ist, dass der/die LeserIn beim Lesen dieses Teils Lust bekommt, diese Arbeit weiter zu lesen.

Abschließend soll die Zusammenfassung noch ein Foto zeigen, das das beschriebene Projekt am besten repräsentiert. Das folgende Bild zeigt Leslie Lamport, den Erfinder von LaTeX.



Acknowledgments

If you feel like saying thanks to your grandma and/or other relatives.

Contents

Chapter 1

Introduction

1.1 Initial Situation

Common word processors do not prepare print-like documents in so far as these programs do not reflect the rules of professional printing which have been grown over centuries. These rules contain clear requirements for balancing page layouts, the amount of white space on pages, font-handling, etc. Donald Knuth's TeX package (see [?]) is a word processor which conforms to these printing rules. This package was enhanced by Leslie Lamport by providing more text structuring commands. He called his package LaTeX [?].

When preparing a thesis, we want not only to have our content on a top level, we also want to commit to a high level of formal criteria. Therefore, we request our students to use one of these professional printing production environments like TeX or LaTeX.

Furthermore students should train their scientific writing skills. This includes a clear and structured break-down of their ideas, a high-level and clear wording, and the training of transparent citations of ideas from other sources than from theirs. A good source for more information concerning technical and scientific writing can be found in [?].

1.2 Goals

The general goals and objectives of the project are described here. Care must be taken that the goals documented here are purely project goals and have nothing to do with individual goals of the team members. If individual goals should be part of the thesis they are listed in appendix B.

1.3 Overview

Details of the diploma thesis have to be aligned between student and supervisor. This should be a basic structure to facilitate the first steps when students start to write their theses.

Figure 1.1: Don Knuth, the inventor of TEX

Never forget to add some illustrative images. Images must not be messed up with your normal text. They are encapsulated in floating bodies and referenced in your text. An example can be seen in figure 1.1. As you can see, figures are placed by default on top of the page nearby the place where they are referenced the first time. Furthermore you can see that a list of figures is maintained automatically which can be included easily by typing the command `\listoffigures` into your document.

1.4 Basic Terminology

As usual the very basic terminology is briefly explained here. Most probably the explanations here only scratch a surface level. More detailed explanations of terminology goes into chapter **??**.

1.5 Related Work and Projects

Here a survey of other work in and around the area of the thesis is given. The reader shall see that the authors of the thesis know their field well and understand the developments there. Furthermore here is a good place to show what relevance the thesis in its field has.

1.6 Structure of the Thesis

Finally the reader is given a brief description what (s)he can expect in the thesis. Each chapter is introduced with a paragraph roughly describing its content.

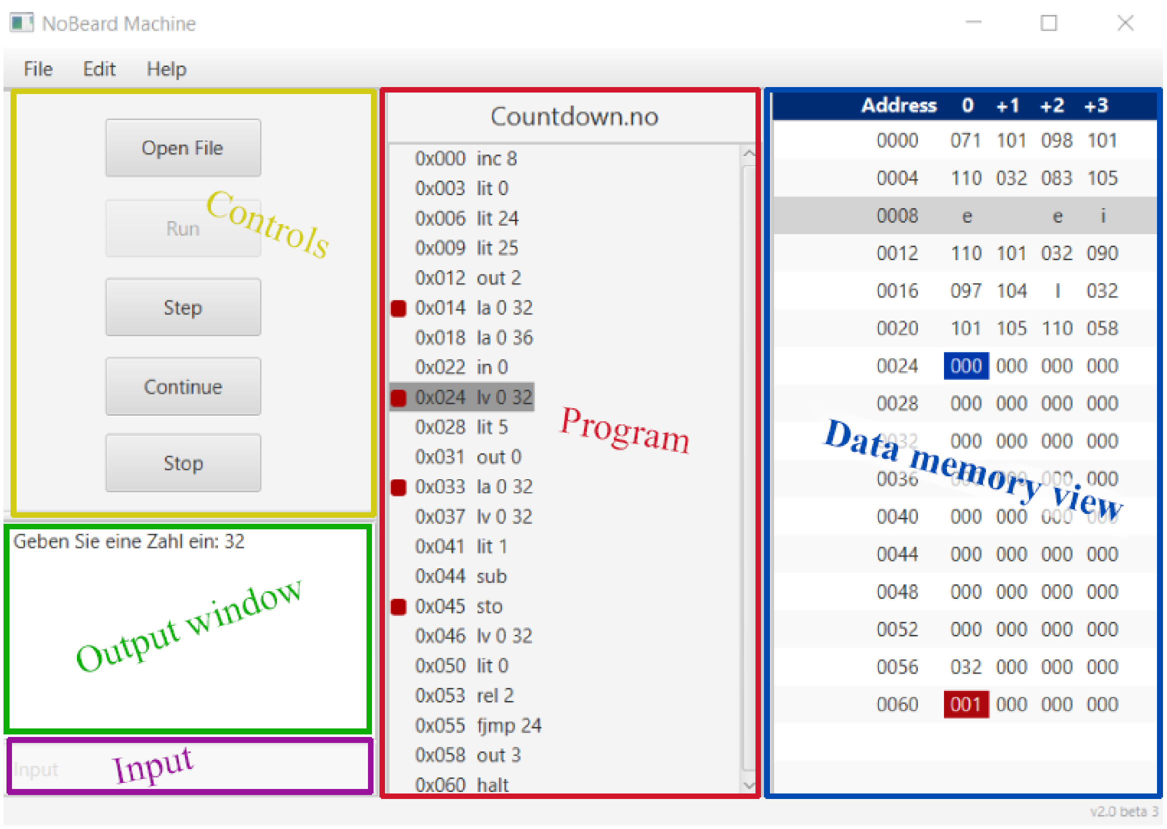
Chapter 2

Formal languages & Compiler construction

Chapter 3

Description of the GUI

This chapter describes how to use the graphical user interface of the NoBeard virtual machine and serves at the same time as a user manual. The application is developed as simple as possible to use. With the three main tools the user can load and run NoBeard object files, debugging running programs and get a whole view of the data memory.



Geben Sie eine Zahl ein: 32

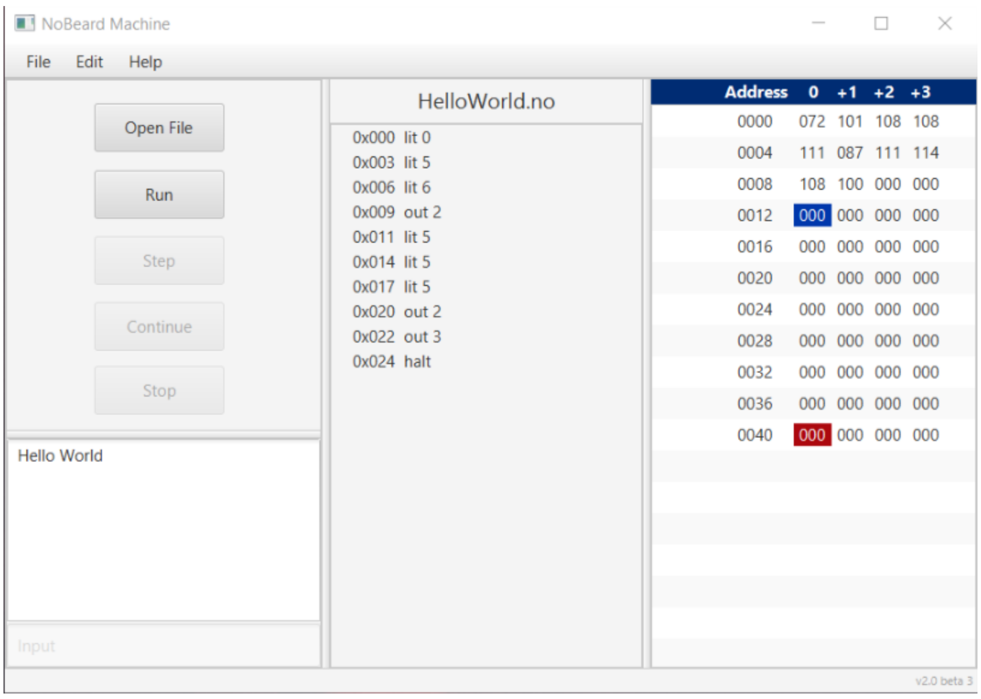
Countdown.no

```
  0x000  inc 8
  0x003  lit 0
  0x006  lit 24
  0x009  lit 25
  0x012  out 2
● 0x014  la 0 32
  0x018  la 0 36
  0x022  in 0
● 0x024  lv 0 32
  0x028  lit 5
  0x031  out 0
● 0x033  la 0 32
  0x037  lv 0 32
  0x041  lit 1
  0x044  sub
● 0x045  sto
  0x046  lv 0 32
  0x050  lit 0
  0x053  rel 2
  0x055  fjmp 24
  0x058  out 3
  0x060  halt
```

Address	0	+1	+2	+3
0000	071	101	098	101
0004	110	032	083	105
0008	e		e	i
0012	110	101	032	090
0016	097	104	l	032
0020	101	105	110	058
0024	000	000	000	000
0028	000	000	000	000
0032	000	000	000	000
0036	000	000	000	000
0040	000	000	000	000
0044	000	000	000	000
0048	000	000	000	000
0052	000	000	000	000
0056	032	000	000	000
0060	001	000	000	000

3.1 Loading & Running a program

By starting the user interface, a NoBeard object file has to be loaded by a click on the "open file" button. Than the user can choose the desired file. Afterward, the window shows the content and the title of the opened program which is now executable.



HelloWorld.no

```
0x000  lit 0
0x003  lit 5
0x006  lit 6
0x009  out 2
0x011  lit 5
0x014  lit 5
0x017  lit 5
0x020  out 2
0x022  out 3
0x024  halt
```

```
Address   0   +1   +2   +3
0000     072  101  108  108
0004     111  087  111  114
0008     108  100  000  000
0012     000  000  000  000
0016     000  000  000  000
0020     000  000  000  000
0024     000  000  000  000
0028     000  000  000  000
0032     000  000  000  000
0036     000  000  000  000
0040     000  000  000  000
```

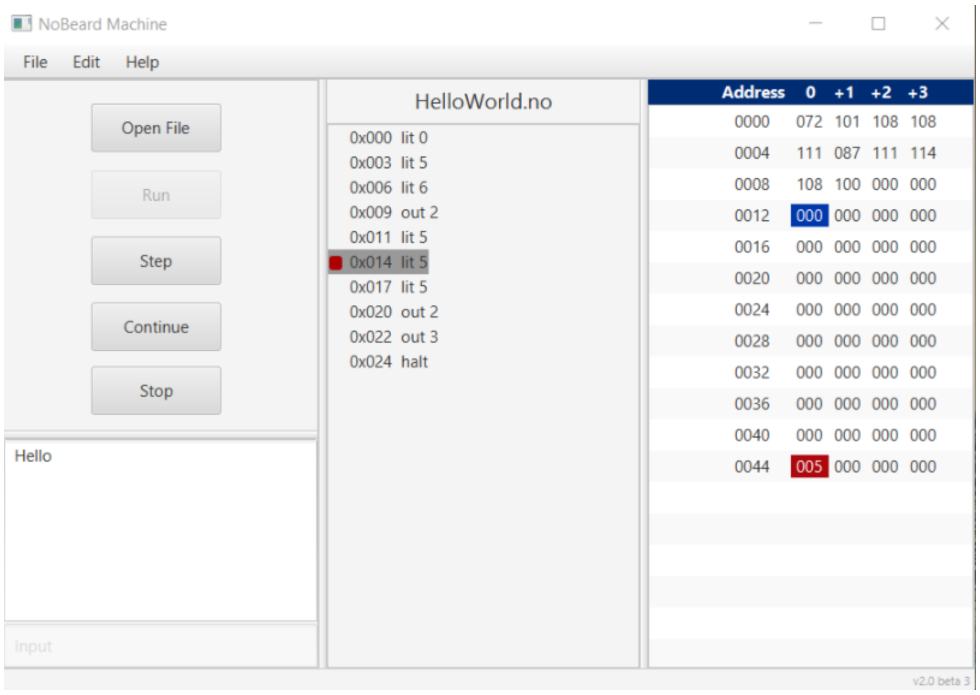
```
Hello World
```

Figure 3.1: Executed "Hello World" program

The content lists instructions with the belonging addresses and operators. By hitting the "Run" button, the machine executes the loaded program. Under the control buttons there is a window which simulates a terminal where the user can see the outputs of the running program. If the program runs against an input instruction, the machine stops and requests the user for an input which can be done under the output window. To submit an input, the Enter key has to be pressed.

3.2 Debugging

To debug a program the user has to set breakpoints. These breakpoints can be placed by clicking on the studied address. If a breakpoint has been placed and the program is about to start, it runs only until the line where the breakpoint has been set.



```
HelloWorld.no
0x000  lit 0
0x003  lit 5
0x006  lit 6
0x009  out 2
0x011  lit 5
0x014  lit 5
0x017  lit 5
0x020  out 2
0x022  out 3
0x024  halt
```

```
Hello
```

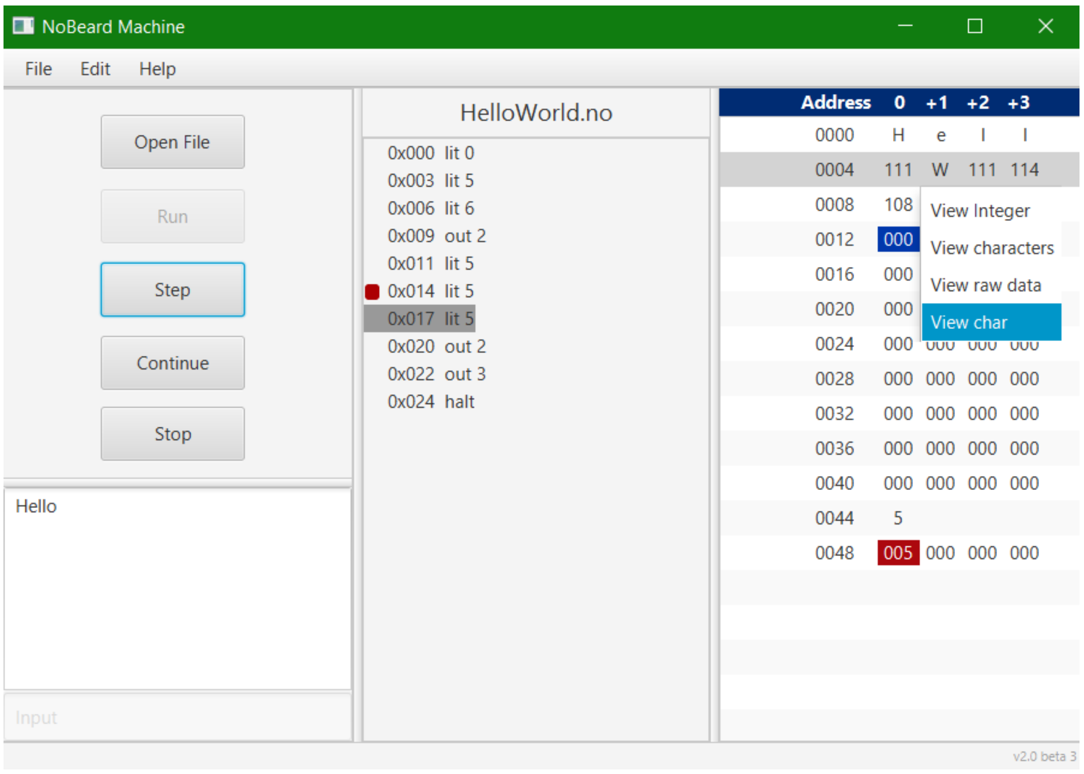
```
Address   0   +1   +2   +3
0000     072  101  108  108
0004     111  087  111  114
0008     108  100  000  000
0012     000  000  000  000
0016     000  000  000  000
0020     000  000  000  000
0024     000  000  000  000
0028     000  000  000  000
0032     000  000  000  000
0036     000  000  000  000
0040     000  000  000  000
0044     005  000  000  000
```

Figure 3.2: Debugging the "Hello World" program

Than the user is able to step one line further or jump to the next breakpoint. Stepping is handled by the "Step" button as shown in the following picture. By clicking on the "Continue" button, the program runs from the current line until the next breakpoint. If there is not any breakpoint left from the current line, it runs until the end of file. Optionally, the user is able to stop the program during the execution. This could be achieved by the "Stop" button.

3.3 Data Visualization

On the right side of the window is the visualisation of the data memory view in a list form. This view lists data byte wise from the data memory of the machine. Each line of the list view has a content of an address given in decimal notation following by four bytes raw data. The memory is separated in two parts. The list starts on the top with the string constants followed by stack frames of the currently running functions. While the frame pointer is highlighted with a blue background, the stack pointer is signed with a red background. The user has also the possibility to convert confused raw data to characters or integers.



```
Hello
```

```
HelloWorld.no
0x000  lit 0
0x003  lit 5
0x006  lit 6
0x009  out 2
0x011  lit 5
0x014  lit 5
0x017  lit 5
0x020  out 2
0x022  out 3
0x024  halt
```

```
Address   0   +1   +2   +3
0000      H    e    l    l
0004    111    W  111  114
0008    108
0012    000
0016    000
0020    000
0024    000  000  000  000
0028    000  000  000  000
0032    000  000  000  000
0036    000  000  000  000
0040    000  000  000  000
0044      5
0048    005  000  000  000
```

Figure 3.3: Converting data to characters

With a right click on the selected line of the list, a context menu could be opened. This menu includes following functions: showing integer, characters of the whole line, a single character or converting back the line to raw data. To make the conversion from raw data to alphanumeric characters or integers more easy and fast, a multi selection function is developed for the list of the data memory.

Chapter 4

Implementation

The Implementation of the project is written in Java on the integrated development environment IntelliJ. This IDE is one of the best options for coding because of the huge variety of benefits just like maven. This is a software management and build automation tool which is based on the concept of a project object modem (POM). One of the biggest feature in Maven is the dependency management. Maven automatically downloads in POM file declared libraries and plug-ins from a repository and stores them local. The local repository is a folder structure that is used as a centralized storage place for locally built artifacts and as a cache for downloaded dependencies. The Maven command mvn install builds a project and places its binaries in the local repository. Then other projects can utilize this project by specifying its coordinates in their POMs.

The NoBeard GUI depends on some existing external package libraries. These libraries include a wide amount off packages. One of the most important package is the "machine". This is defined as the core of the whole project because it has the most significantly parts including:

- **NoBeardMachine:** a stack based machine with instructions defined as co-operator
- **ControllUnit:** is responsible to execute commands and instructions
- **DataMemory:** a byte addressed storage
- **CallStack:** provide functions to add and remove frames from stack and maintains the expression stack
- **InstructionSet:** a list of instructions and their implementations

The GUI is designed very similar to the Model-View-Controller pattern. However, this project is not depending on any database so it comes without any Model. The view is a single fxml file which holds the view components

of the layout. It was created with an external tool called SceneBuilder that allows developers to design UI without any coding, just drag & drop with the mouse. Every item of the view has an fx:id which is used in the controller to get easy access to the items. The controller is responsible for the coordination and operation of the view. Initially, it gets an instance of the NoBeard machine where simple program can be executed. The opening of a NoBeard object file is operated with the help of a BinaryFileHandler. After a successful opening of the object file, it has to be disassemble. This function converts binary files to primary program data where addresses, instructions and operands become visible on the UI.

By finishing the translation, the machine has to load the string storage and the program from the object file. Program data is filled in a VBox with a CheckBox and text of the data for every line. All of the CheckBoxes gets an OnAction event which add and remove breakpoints from the machine.

On starting a program, a new external thread has to be started where the machine runs separate from the UI. Its executes step by step every instruction of the program until any interruptions. It can be interrupted by a breakpoint, input request or by a halt instruction.

4.1 Interruption by breakpoint (Observer pattern)

The implementation for the maintaining of breakpoints is coded with a simple observer pattern design to achieve the best performance of the virtual machine. The machine is completely separated from the breakpoint stuff. It runs only until its state equals to “running”. So, a new instruction is introduced, called “BREAK”, which sets the machine in to a “blocked” state. The ControlUnit class is going to be an Observable. Then a new Observer class is implemented which is called debugger that is holding all breakpoints in a HashMap. This HashMap stores the address and the instruction of a breakpoint. The debugger class contains in all three major functions. Adding, removing breakpoints to the HashMap and replacing an instruction at a specified address from the program memory to a new one. The selection of a breakpoint on the UI calls the set or remove function from the debugger, as the case may be. As soon as a breakpoint is selected, it will be stored to the HashMap with its original instruction and at same time replaces the original instruction by the newly added break instruction. This break instruction set the machine to a blocked state and notify the observer to change the break instruction back to the original one which is stored in the HashMap. Now the machine is blocked at a specified breakpoint and the user can take a look to the current stack frames and go one step further in the program or continue the execution cycle to a next breakpoint. However, when the user wants to step further to execute the current instruction where the breakpoint is, a switch back to the break instruction is needed again after the original instruction completed.

4.2 Interruption by input request (Threading)

As already mentioned, the virtual machine runs on a separate thread so every time the user has to operate an input, a switch to the UI thread is needed. Otherwise it would cause a critical section between the threads. The synchronization of these two threads is implemented with the semaphore construction. The NoBeardMachine contains two interfaces to optimize outputs and inputs on the used device. As soon as the machine executes an input instruction (IN), it calls firstly a function from the input interface either hasNextInt() or hasNext(). Both do the same thing, checks whether there is an input by the user or not. The only different is that the one of them also checks if the string is numeric. These functions are overridden in the FxInputDevice class where also an instance of the controller is loaded by the constructor. So, by calling one of these hasNext function the machine thread should be paused. To avoid the deadlock, the semaphore from the controller is acquired at this position. Now the user can make an input on the FX thread and submit it. To get back to the machine thread, the semaphore hast to be released after the user fires the submit event by pressing the ENTER key. Then the machine continues at the same position where the semaphore was acquired, at one of the hasNext function and can analyse the provided input string.

Chapter 5

Summary

Here you give a summary of your results and experiences. You can add also some design alternatives you considered, but kicked out later. Furthermore you might have some ideas how to drive the work you accomplished in further directions.

List of Figures

List of Tables

Project Log Book

Date	Participants	Todos	Due

Appendix A

Additional Information

If needed the appendix is the place where additional information concerning your thesis goes. Examples could be:

- Source Code
- Test Protocols
- Project Proposal
- Project Plan
- Individual Goals
- ...

Again this has to be aligned with the supervisor.

Appendix B

Individual Goals

This is just another example to show what content could go into the appendix.