Master Thesis

Niklas Lundberg, inaule-6@student.ltu.se

July 6, 2021

1 Abstract

TODO

Contents

2 Introduction

TODO

2.1 Background

TODO

2.2 Motivation

TODO

2.3 Problem definition

Debugging Rust code on embedded system today is not a very good experience for many reasons. One of the big problems is debugging optimized code is often near impossible. That is because the compilers today are very good at inlining the code and removing unused code. This changes the program so drastically in many cases that when trying to debug the code all the original variable can be optimized out. Thus the user doesn't get any useful information from the debugger about the program which makes is near impossible to debug. One of the big reasons why variable gets optimized out is because unoptimized code always push the value of the variable to memory which then can be read by the debugger at anytime. This is not done for optimized code because speed is prioritized and storing the value of a variable that will not be used on the stack is costly.Thus the time that most variable exist is very short in optimized code which results in that the debugger saying that a variable is optimized out.

There are two main problem that this thesis tries to tackle to improve the experience of debugging optimized code for embedded systems. The fist problem is about the generation of the debug information, if more debug information can be generated then there is more information the debugger can retrieve an show the user. This is also were the problem starts with debugging optimized code, because debuggers needs the debug information to understand the relation between the source code and the machine code. Thus it is very important that the compiler generates as much debug information as possible, because there is nothing that can be done later to get more information. The first problem then is to look at the different options that can be set in the *llvm* compiler to improve the generations of debug information without impacting the optimisation of the code to much. Speed of the resulting code is still a big priority.

The second problem is looking at the available debug information that the *llvm* compiler generates for optimized code and creating a debugger that utilises that information to the fullest. This problem has two parts to it, the first being retrieving the needed information from the debug information. This will be the hardest part and is the most important for improving the debugging for optimized code. The second part is to display the debug information to the user in a user friendly way.

The goal of solving these to problems is to create a debugger that gives a better debugging experience for optimized rust code on embedded systems then some of the most commonly used debugger, such as *gdb* and *lldb*. And to inspire further development for debugging tools in the rust community.

2.4 Equality and ethics

TODO

2.5 Sustainability

TODO

2.6 Delimitations

TODO

2.7 Thesis structure

TODO

3 Related work

TODO

4 Theory

TODO

5 Implementation

TODO

6 Evaluation

TODO

7 Discussion

TODO

8 Conclusions and future work

TODO