Eridu – an Alternative Type System for a C-Like Language

(Eridu - et alternativt typesystem i et C-lignende sprog)

Lasse Usbeck Andresen and Frederik Gram Kortegaard

Advisor: Kim Skak Larsen

5.2.2022

A modern compiler is organized into phases, where the basic ones cover lexical and syntactic analysis, resulting in an abstract syntax tree. Subsequent phases analyze and adorn the abstract syntax tree, building a symbol table, performing type checking, and finally generating target code (assembler, for instance). This material is known from DM565 for a simple imperative language. The goal of this project is to go beyond this basic language and basic compiler.

After having defined a C-inspired general-purpose programming language and implemented the compiler, translating to 64 bit X86 Assembly, our focus will be on features not present in C.

The language will contain some basic types as well as possibilities for defining composite types. In that setting, we will allow for a type, *callable*, which will enable the programmer to store functions such that some of the code saving and polymorphic behavior of object-oriented programming styles can be utilized without enabling actual object-orientation. A main challenge here will be to implement type checking for this construction.

To the extent that time allows, we will then consider advanced error handling, as it exists in more modern programming languages than C.

For all the advanced elements, we will motivate and exemplify the constructions to demonstrate functionality and where they have their strengths and

limitations.

The report should document and discuss the developed compiler and the interesting choices made in the process. It should be organized by phases and extensions beyond a basic compiler. The project will be evaluated on language design and usability, extent of advanced additions and phases, and maintainability. For the connection between the implementation and the report, the evaluation will focus on motivation, descriptions of advanced features relative to a basic language and compiler, and documented correctness and effect.