Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Dipartimento di Elettronica, Informazione e Bioingegneria
Master Degree In Computer Science and Engineering

Thesis Title

Advisor: Prof. Giovanni AGOSTA

Thesis by:
Pietro Ghiglio Matr. 920491

Academic Year 2019–2020

To someone very special...

Acknowledgments

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Sommario

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Contents

List of Figures

List of Tables

List of Algorithms

Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Chapter 1

Background

1.1 LLVM

The LLVM Project [3] is a collection of modular and reusable compiler toolchain technologies. It is built around an intermediated representation called LLVM-IR, and provides a set of APIs to interact with it. LLVM provides an optimizer that works on the intermediate representation, and also several code generation helpers that allow to target all the main hardware architectures.

1.1.1 LLVM-IR

The LLVM-IR is a language that resembles a generic assembly language, while also providing some high level features such as unlimited registers, explicit stack memory allocation and pointer deferentation. This allows LLVM-IR to be both the ideal target for high-level language developers, that do not have to worry about architecture specific details, and also the ideal source language for compiler backend developers, that have to implement only a translator from LLVM-IR to their target architecture's assembly language, without worrying about high-level language features. The LLVM-IR is accesible in three formats: in-memory represantation, that allows manipulation through the LLVM APIs, binary format, used by many LLVM tools, and the human-readable textual format, that can also very conveniently be parsed by means of the APIs.

1.1.2 SSA and Phi nodes

The LLVM-IR is by definition in SSA (Static Single Assignment) form. The SSA form requires a variable to be assigned only once, and requires every variable to be defined before its uses. It is called static because it does not take into account dynamic (related to the program's runtime) considerations. For instance, an assignment in a loop counts always as one assignment, even if at runtime it will be

performed several times.
—-esempio
It is always clear which definition to use, unless a basic block has multiple predecessors. In that case it is necessary to add phi nodes that carry the information to disambiguate the uses at runtime.
—-esempio

1.1.3 Class hierarchy

The class hierarchy defined in the LLVM APIs consists of hundreds of classes, a complete and exhaustive view is given by the LLVM Doxygen Documentation. The main components of the hierarchy are:

- Module: the entire program/compile unit. Contains the global values of the program: mainly the global variables and the functions.

- Function: a function in the compile unit, contains mainly a set of arguments and it's control flow graph in the form of a set of basic blocks.

- Basic Block: a set of instructions with no branches between them.

- Instruction: An instruction of the IR.

Another key class in the LLVM class hierarchy is the Value class. It represents anything that has a type and can be used as an operand to an instruction: function arguments, constants, instructions, basic blocks and functions are all Values. A Value also carries information of what other Values it uses, and what other Values use it.

1.1.4 LLVM Metadata

The LLVM-IR allows metadata to be attached to Instructions, Functions, Global Variables or Modules. Metadata can convey extra information about the code to the optimizers and code generator. The main use of metadata is debug information, but they may also carry information about loop boundaries or other assumption that are useful during the optimization or code generation phases.
Metadata can either be a simple string attached to an instruction, or they can be a Metadata Node (MDNode). MDNodes can reference each other and are specified by other classes in the LLVM APIs. See section 2.1 or the LLVM Language Reference [1] for more details.

1.1.5 LLVM Passes

LLVM passes are where most of the interesting parts of the compiler exist. Passes perform the transformations and optimizations that make up the compiler, they build the analysis results that are used by these transformations, and they are, above all, a structuring technique for compiler code. In LLVM, passes may depend on other passes, for instance a pass that performs an optimization may require the results of a pass that performs a specific analysis. They are therefore handled by a Pass Manager, that schedules the passes, ensuring that all the requirements for a pass are satisfied before executing it.

1.2 Debugging

A debugger is a computer program used to test and debug other programs. It allows a programmer to run the target program in controlled conditions, pause the program's execution, check the state of variables and more.

1.2.1 Debug information

The main functionalities of a debugger, over which more advanced features can be built, are setting break points and accessing the content of a variable.
This is achieved by means of debug information: information stored by the compiler in the program's executable, with the purpose of providing a correspondence between source level entities (variable, source code locations, data types) and low level entities (assembly instructions and memory locations).
The format used to store them may vary with the compiler/operating system used, but the stored information are mainly:

- Definition of the data types employed in the program and their layout in memory, both language-defined (eg. int, float, unsigned in C) or user defined (eg. C structs or C++ classes).

- Mapping between variables defined in the source code and memory locations in which they are stored. This allows a debugger to output the value of a variable given its name.

- Mapping between source code locations and assembly instruction. This allows the debugger to pause the program's execution when a given source code location is reached.

1.2.2 DWARF format

The DWARF format [2] is a debugging file format used by many compilers and debuggers to support source-level debugging. It is designed to be extensible with re-

spect of the source language, and be architecture and operating system independent. The main data structure used to store debug information is the DIE (Debug Information Entry). DIEs are used to describe both data types and variables, and can reference each other creating a tree structure.
Another data structure that is very useful for our purposes is the Line Number Table: it contains the mapping between memory addresses of the executable code, and the source line corresponding to those addresses.
Each row of the table contains the following fields:

- Address: the program counter value of a machine instruction.
- Line: the source line number.
- Column: the column number within the line.
- File: an integer that identifies the source file.

1.3 Energy modeling

Chapter 2

State of the art

2.1 How LLVM handles debug information

2.2 Review of the literature

2.3 Visualization of compiler optimizations

2.4 Producing and energy model

Conclusions

Bibliography

[1] *LLVM Language Reference Manual*. URL: https://llvm.org/docs/LangRef.html.

[2] *The DWARF Debugging Standard*. URL: http://dwarfstd.org/.

[3] *The LLVM Compiler Infrastructure*. URL: https://llvm.org/.

Appendix A

First appendix

Appendix B

Second appendix

Appendix C

Third appendix