Where is LLVM being used today?

Tilmann Scheller
Senior LLVM Compiler Engineer
t.scheller@samsung.com

Samsung Open Source Group
Samsung Research UK

FOSDEM 2016
Brussels, Belgium, January 30 – 31, 2016

Overview

- Introduction
- LLVM Overview
- Projects
- Summary

Introduction

What is LLVM?

- Mature, production-quality compiler framework
- Modular architecture
- Heavily optimizing static and dynamic compiler
- Supports all major architectures (x86, ARM, MIPS, PowerPC, …)
- Powerful link-time optimizations (LTO)
- Permissive license (BSD-like)

Which companies are contributing?

Google

SAMSUNG

Microsoft

intel

QUALCOMM®

ARM®

IBM

Imagination

SONY

COMPUTER ENTERTAINMENT ®

NVIDIA®

CRAY

Linaro

AZUL SYSTEMS®

AMD

History

- Started as Chris Lattner's Master's Thesis at UIUC
- LLVM 1.0 released in October 2003
- LLVM 3.8 about to be released
- Today: LLVM + Clang together 2.5 million LOC of C++ code

LLVM sub-projects

- **Clang**

  C/C++/Objective C frontend and static analyzer

- **LLDB**

  Next generation debugger leveraging the LLVM libraries, e.g. the Clang expression parser

- **lld**

  Framework for creating linkers, will make Clang independent of the system linker in the future

- **Polly**

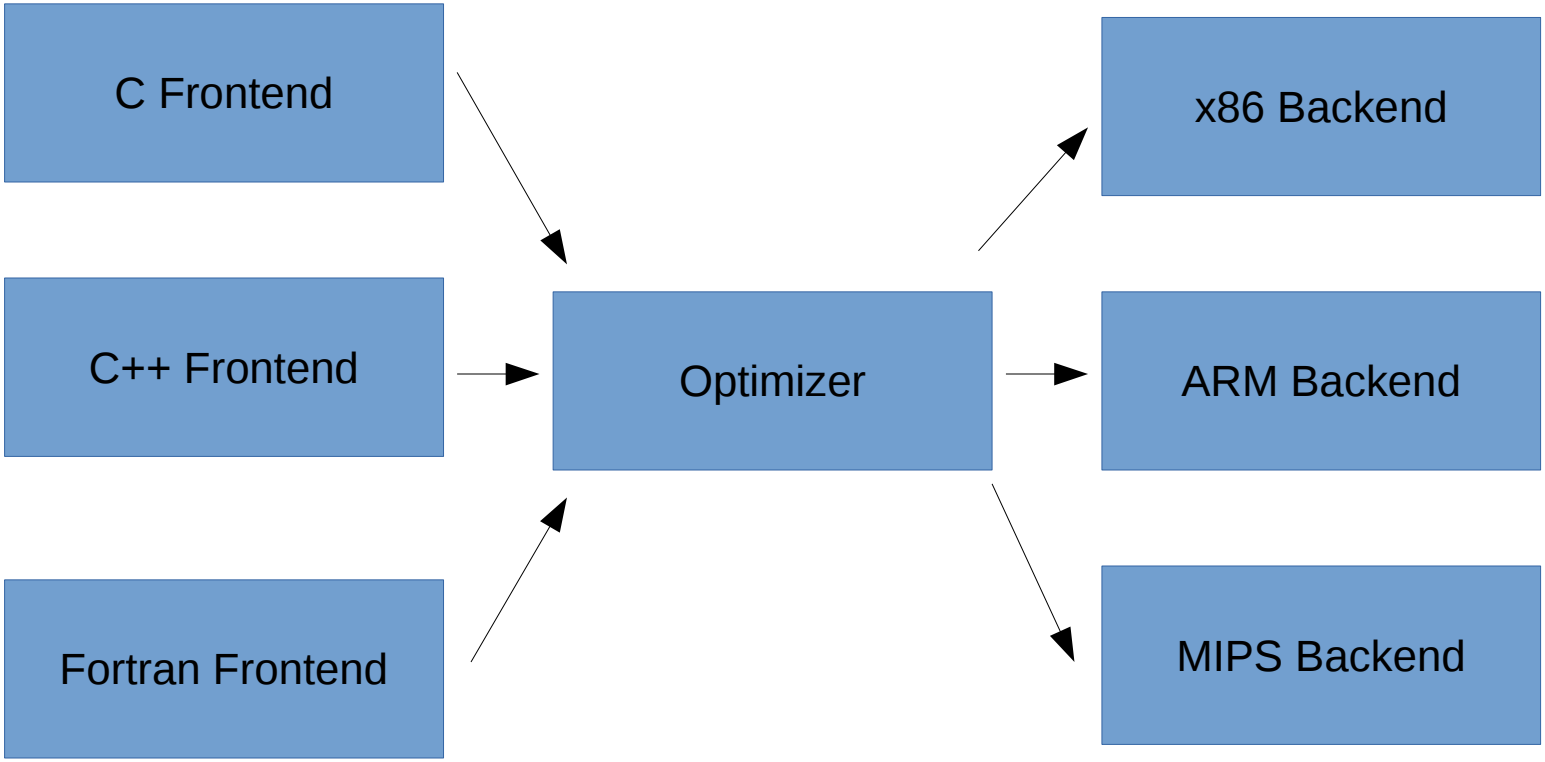
  Polyhedral optimizer for LLVM, e.g. high-level loop optimizations and data-locality optimizations

LLVM Overview

LLVM

- LLVM IR (Intermediate Representation)
- Scalar optimizations
- Interprocedural optimizations
- Auto-vectorizer (BB, Loop and SLP)
- Profile-guided optimizations

Compiler architecture



Compilation steps

- Many steps involved in the translation from C source code to machine code:
  - Frontend:
    - Lexing, Parsing, AST construction
    - Translation to LLVM IR
  - Middle-end
    - Target-independent optimizations (Analyses & Transformations)
  - Backend:
    - Translation into a DAG
    - Instruction selection: Pattern matching on the DAG
    - Instruction scheduling: Assigning an order of execution
    - Register allocation: Trying to reduce memory traffic

Clang

- Goals:
  - Fast compile time
  - Low memory usage
  - GCC compatibility
  - Expressive diagnostics
- Several tools built on top of Clang:
  - Clang static analyzer
  - clang-format, clang-modernize, clang-tidy

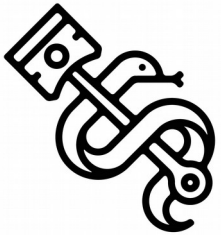
Projects

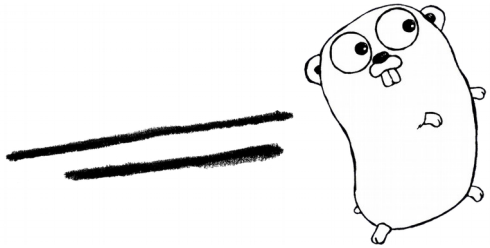
Traditional C/C++ Toolchain

- Apple iOS/OS X SDK
- Android NDK
- Tizen SDK
- Sony PS4 SDK
- Qualcomm Snapdragon LLVM Compiler for Android

Programming languages

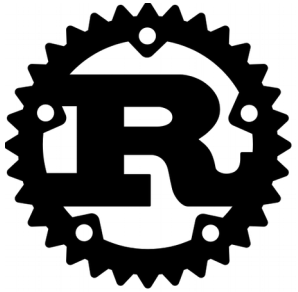
- Swift
- Haskell: GHC, LHC, UHC
- Ruby: Rubinius, RubyMotion
- Python: Pyston
- Common Lisp: Clasp
- D: LDC
- Go: llgo





Programming languages

- Standard ML: MLton, SML#, Ex-SML
- Rust
- Julia
- Pure
- Ravi



Language Runtime Systems

- VMKit (unmaintained)
- LLILC - LLVM-based .NET MSIL compiler
- Mono
- OpenJDK

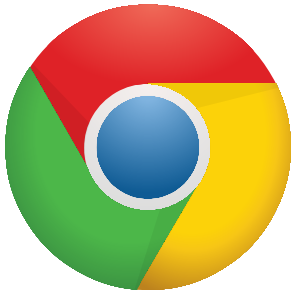
OpenJDK

GPU

- LLVMpipe (software rasterizer)
- CUDA
- GLSL (LunarGLASS)
- AMDGPU open source drivers
- SPIR
- Majority of OpenCL implementations based on Clang/LLVM

Web

- PNaCl
- WebKit FTL JIT
- Emscripten
- WebAssembly



Sanitizers

- AddressSanitizers
- MemorySanitizer
- ThreadSanitizer
- LeakSanitizer
- SAFECode

Integrated Development Environments

- Xcode
- KDevelop
- CodeLite
- Qt Creator
- Geany

Source code navigation

- Doxygen
- Woboq Code Browser
- YouCompleteMe - Code completion for Vim
- clang-tags
- clang-ctags
- clang_complete - Code completion for Vim
- rtags - Indexer for C/C++ with Emacs integration

Out of tree LLVM backends

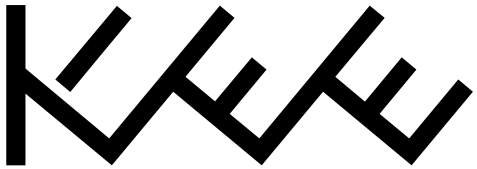
- RISC-V
- OpenRISC 1000
- VideoCore IV (VPU/QPU)
- LatticeMico32
- AAP

Binary translation

- llvm-qemu
- Dagger
- McSema
- libcpu
- Fracture
- SkyEye

Symbolic Execution/Formal Verification

- KLEE
- S2E
- K framework with formal semantics for LLVM IR



Linux/FreeBSD

- Debian experimenting with Clang as an additional compiler (94.1% of ~22k packages successfully build with Clang 3.6)
- LLVMLinux
- OpenMandriva Lx
- FreeBSD

Misc

- Emacs fork using the LLVM JIT for Elisp byte code execution :)
- Cling - C++ interpreter
- CodeChecker - Web frontend for the Clang static analyzer
- include-what-you-use
- clang-closure
- Numba

Summary

Summary

- Wide range of different projects
- New frontends being written constantly
- Great compiler infrastructure
- Fast C/C++ compiler with expressive diagnostics

Give it a try!

- Visit llvm.org
- Distributions with Clang/LLVM packages:
  - Fedora
  - Debian/Ubuntu
  - openSUSE
  - Arch Linux
  - ...and many more

Thank you.

Contact Information:

Tilmann Scheller
t.scheller@samsung.com

Samsung Open Source Group
Samsung Research UK