Xufan Lu

Email : luxufan@tecnico.ulisboa.pt

Education

- **ZheJiang University of Technology** — HangZhou, China
  Bachelor of Computer Science — *Sep. 2017 – June. 2021*

- **Instituto Superior Técnico / University of Lisbon** — Lisbon, Portugal
  PhD of Computer Science — *Seq. 2023 –*

Publications

Link-Time Optimization of Dynamic Casts in C++ Programs

Xufan Lu, Nuno Lopes. **PLDI**, 2025.

Experience

- **Institute of Software Chinese Academy Of Sciences** — Remote, China
  LLVM Compiler Engineer — *July 2021 - Present*
  - **Improve and optimize LLVM middle-end Analysis and Transforms passes**: The middle-end performs optimizations that are independent (to the degree that this is possible) from both the input language and the output target. I have contributed code to the following modules:
    - **IPSCCP** Improve implementation and increase the number of instructions deleted by IPSCCP pass
    - **MemorySSA** Relax conditions to make MemorySSA give more precise results
    - **CorrelatedValuePropagation and LazyValueInfo** Fix bugs, make LVI give more precise result so that CVP can optimize out more instructions.
    - **InstCombine and InstructionSimplify** Combine and simplify more patterns.
    - **PoisonValue** and **UndefValue** Fix bugs that caused by abusing Undef and Poison value.
    - **Loop optimization and ScalarEvolution** Improve loop optimization and fix bugs that caused by forgetting clear loop disposition and block dispositon information.
  - **Improve and optimize LLVM RISC-V backend**: The RISC-V backend performs target-specific optimizations and emits RISC-V machine code. In these two years, I have done these things:
    - Optimize FrameLowering: improve frame layout and decrease the number of mv instructions to handle stack stuff.
    - Support .option directive: .option can be used to enable code snippet.
    - Optimize instruction selection: reduce the number of machine code instructions generated for specific pattern.

    - Support RISC-V instruction set extensions: vector zvlsseg extension, zfinx, zdinx, zhinx extensions.
  - **Solve LLVM part issues arise during porting Android to RISC-V**: Currently, the latest Android NDK version only has clang/llvm compiler available. In the process of porting Android to RISC-V, there are many LLVM related toolchain issues out there. As a LLVM compiler engineer, I solved part of these issues. Here are some patches that I upstreamed:
    - Generate correct ELF EFlags when .ll file has target-abi attribute
    - Pass -mno-relax to assembler when -fno-integrated-as specified
    - Support fe_getround and fe_raise_inexact in builtins
  - **Add RISCV backend support for LLVM JITLink**: JITLink is a library for JIT Linking. It was built to support the ORC JIT APIs and is most commonly accessed via ORC's ObjectLinkingLayer API. It is depended on by many projects like BOLT, llvm-pipe, Julia. I contributed the following functional support:
    - Relocate all of the RISCV ELF relocations
    - Handle GOT and PLT table in PIC mode
    - Add platform and runtime support for riscv64
  - **Add native ELF TLS support for JITLink x86-64 ELF platform**: JITLink is a new JIT backend to replace MCJIT, and there is no TLS related support in MCJIT. I implemented the x86-64 ELF TLS support for JITLink. And improved the ELF TLS support framework in JITLink

- **Institute of Software Chinese Academy Of Sciences** Remote, China
  LLVM Compiler Engineer Intern *May 2020 - July 2021*
  - **Support RISC-V Vector extension in LLVM**: Support RISC-V Vector intrinsics in clang and code generation it to RISC-V Vector version 0.8 instructions in an internal project with other team members.
    - Write tablegen files to create all of the vector c intrinsics and make clang frontend codegen to the corresponding llvm ir intrisics.
    - Write tablegen files to define all of the RISC-V Vector extension instructions. Hack the RISC-V backend framework to make backend generate correct machine codes for program that has vector intrinsics
    - Handle vector stack object. RISC-V vector is a kind of scalable vector. Hack to RISC-V backend FrameLowering part to support such scalable vector objects in stack. I also proposed a new Stack Frame layout to support the scalable vecotr stack object. And I have contributed this part to LLVM upstream and accepted.

Skills

- **Languages**: C++, C, Python, Shell script
- **Technologies**: Compilation Principle, LLVM IR, RISC-V Instruction Set, Operating system and computer architecture

Most of the time, I program by C++. I also have previous experiences with RISC-V assembly language, Python and Shell script language, and functional programming languages. It is not a challenge for me to pick up a new language.

I am familiar with compiler optimizations and llvm ir design. I have contributed some patches to the LLVM middle-end optimizer, backend target-specific optimizations, and language frontend.

Most of my knowledge about operating system and computer architecture was learned when I was in the university.