Proposal for GSoC 2025

Participate in ClangIR Upstreaming

Project Abstract

This project focuses on participating in the process of bringing ClangIR, currently an LLVM incubator project, into the main LLVM monorepo.

Project Repo Description

Applicant Information

- **Name**: [REDACTED]
- **GitHub**: el-ev (Iris)
- **Email**: 0.0@owo.li

- **Education**:
  - 2022-Present: **Beihang University**, Computer Science and Technology
- **Contributions to ClangIR** (till April 2nd, Link):
  - #1502 [CIR][CIRGen][Builtin] add `__builtin_tan`
  - #1504 [CIR] Refactor `StructType` with TableGen
  - #1508 [CIR] Add `AddressPointAttr`
  - #1521 [CIR][CIRGen][NFC] Handle `PragmaComment` in `emitTopLevelDecl`
  - llvm/llvm-project#133385 [mlir][llvm] Add `LLVM_DependentLibrariesAttr`
  - #1532 [CIR][LowerToLLVM] Support for LinkerOptions lowering
  - #1543 [CIR][CIRGen][Builtin] Add `__builtin_elementwise_{log, log2, log10}`
- **Related Experience**:
  - Have 2 years of experience in C, modern C++, and Rust programming, and basic knowledge of several other languages.
  - Finished Compiler Principle course, with a simplfied C compiler as the final project.
  - Have some previous contributions to llvm-project. Link. Have basic understanding of LLVM IR and llvm internals.
  - Have previous experince in other open source projects, like files-community/Files (in C#) and web3infra-foundation/mega (in Rust).
  - Teaching Assistant of Operating System course in Beihang University.

Proposal

Project Name

Participate in ClangIR Upstreaming

Synopsis

The project consists of three main objectives:

1. Migrate ClangIR support for C and C++ language features into the main LLVM repository.
2. While migrating, refactor the codebase to meet the LLVM coding standards.
3. Suggestions for future improvements to ClangIR migration process.

Motivation

ClangIR is a MLIR dialect for C/C++ based languages. It is designed to express high level C/C++ semantics, enabling a class of idiomatic diagnostics and performance optimizations that are hard to explore on Clang AST or LLVM IR level.

After two years of development, ClangIR has started the process of upstreaming. This process will make ClangIR more accessible to the LLVM community, encouraging more contributions. Additionally, the upstreaming process will involve high-quality refactoring and code reviews, enhancing the overall quality of the codebase.

Concrete Deliverables

For coding parts, this project will deliver the following:

- **At least eight** feature parts (pull requests, each of about 500 lines of code) merged into the llvm-project repository.
- Enhancements to the ClangIR codebase, such as refactoring, code cleanup, and implementing NYI parts of the features.

As ClangIR is under active development and the upstreaming process is in progress, it is uncertain which features will be ready for upstreaming when the coding period officially starts. Therefore, the exact features to be migrated will be determined later in collaboration with the community. Project success will be determined by successful upstreaming of the feature or set of features agreed up by the project mentor.

Development Plan

The following plan outlines the steps to have a single feature migrated into the llvm-project repository. After one feature is successfully migrated, the process will be repeated for the next feature.

1. **Feature Identification:** Collaborate with mentors to select a specific C/C++ language feature within the ClangIR codebase for migration.
2. **Code Locating and Understanding:** Thoroughly examine the existing ClangIR codebase to identify code sections related to the chosen feature. Understand the structure, functionality and dependencies of the code.
3. **Decomposition:** Depending on the size and complexity, divide the code into smaller units that are independently reviewable and testable.
4. **Refactoring:** Choose one of the units. Refactor the code to ensure it adheres to the LLVM coding standards.
5. **Testing:** Locate and adapt existing ClangIR tests that directly exercise the migrated feature. Write new tests to improve coverage if necessary.
6. **Review and Refinement:** Submit the refactored code along with the tests to llvm-project for review. Actively participate in the review process, addressing feedback and making necessary changes, until the code is approved and merged.
7. Iteratively repeat Step 4 to 6 for the remaining units of the feature.

Expected Timeline

Date	Work
Prior to May 8th	• Set up development environment. • Get familiar with the ClangIR bodebase. • Fix issues in the ClangIR repository.
May 9th-May 31st	• Select a specific feature for migration with the mentors. • Locate and understand the relevant code sections. • Aim for at least one patch submitted (if not merged).
June 1st-July 14th	• Continue refactoring, testing, and submitting patches related to the feature. • Follow up on the review process. • Have 4 patches submitted and merged by the end of this period. • Prepare documents for midterm evaluation.
July 15th-August 25th	• Continue working on the selected feature, or, if completed, begin a new one. • Prioritize merging existing patches.

	• Have another 4 patches submitted and merged. • Prepare documents for final evaluation.

Extra Information

Working Hours

I will be based in Beijing, China during the coding period. Therefore, I will be working in the UTC+8 timezone.

I believe I can finish the project in 12 weeks, which is the size of a **large-scale project**. If I finish the original plan earlier than expected, I would like to work on upstreaming more features.

I will be taking final exams in the last two weeks of June, when I may be less available (but still progressing). Other than that, I will be able to fully commit to the project.

Reason for Participation

I am an undergraduate student majoring in Computer Science and Technology and have a strong interest in compilers and programming languages. LLVM is a well-known compiler infrastructure which provides a lot of opportunities for learning compiler techniques. Among the projects provided by LLVM for GSoC 2025, ClangIR is a relatively new and fast evolving project, which I believe will be easier for me to make significant contribution to, as well as learn a lot from.

I've completed the Compiler Principle course in my university, have concrete C++ programming experience, and my previous contributions to LLVM and ClangIR could demonstrate my ability to write clear and well-tested code.

In the future, I hope to work in the field of compilers and programming languages. I believe that participating in GSoC 2025, especially in the ClangIR project, will be a great opportunity for me to learn more about compiler design, and further help me to achieve my career goals.

If I had the opportunity to participate in GSoC with LLVM community, I would do my best to complete this project. I am looking forward to working with the LLVM community and contributing to the ClangIR project.