1. Введение

Бинарные трансляторы являются инструментами для эмуляции, бинарного анализа и профиляции, важным классом которых являются динамические бинарные трансляторы, которые преобразуют двоичный ход на лету и генерируют нужные инструкции по мере требования. Примером такого транслятора является Rosetta 2. Часто, исходный двоичный код преобразуется в какое-либо промежуточное представление, где такое преобразование осуществляет лифтер. Примерами такого представления являются TCG для QEMU. Но в качестве промежуточного представления хочется использовать LLVM IR, так как тулчейн LLVM может делать качественные оптимизации кода. В данной работе хотим рассмотреть динамические бинарные трансляторы с поднятием двоичного кода в LLVM IR для RISC-V.

2. Обзор

Для начала необходимо рассмотреть такие канонические ДБТ, как

- QEMU-user, который является стандартом эмуляции userland программ;
- Rosetta 2, который является эмулятором с закрытым исходным кодом от Apple;
- Valgrind, который является инструментом динамического бинарного анализа в состав которого входит Memcheck и Callgrind.

Также довольно популярным является Box64, который позволяет запускать программы, в частности игры, скомпилированные под linux x86/x64, на ARM64 и RISC-V, и который отличается производительностью. Из ДБТ с поднятием в LLVM IR, можно выделить

- BinRec (не обновлялся с 2022го года);
- HQEMU (не обновлялся с 2018го года);
- DBILL, LnQ, Lasagne (есть только статьи, либо код закрыт).

В результате обзора стало понятно, что, судя по бенчмаркам, Instrew + Rellume производительнее, чем QEMU и Valgrind, благодаря оптимизациям LLVM. Также, в сравнение: с QEMU,

Valgrind и Box64, является не только эмулятором, но и инструментом для динамического бинарного анализа (что позволяет менять семантику транслируемых программ), так как из коробки имеет API для добавления анализаторов, например, подсчет инструкций. Также в сравнение с другими ДБТ с поднятием в LLVM IR поддерживается контрибьютором и не заброшен, а также имеет поддержку архитектуры RISC-V, как целевой.

3. Постановка задачи

Целью работы является добавление поддержки host-архитектуры RISC-V. Для ее выполнения были поставлены следующие задачи:

- выполнить обзор архитектуры Instrew;
- реализовать минимальную функциональность для запуска Instrew на RISC-V, а именно:
  - скомпилировать, добавив процессорно-специфические патчи и проставив константы (макросы riscv, размеры функций, адреса и номера и аргументы системных вызовов);
  - реализовать функции, связанные с RISC-V ABI (эмуляция системных вызовов, трамплины PLT таблицы, релокации).
- протестировать на простых примерах.

4. Архитектура Instrew

Instrew реализует архитектуру клиент-сервер. Клиент написан на языке Си и из его основных функций можно выделить:

- отправка функций транслируемой программы серверу;
- получение от сервера ELF объекта, проведение релокаций и разрешение символов.

Также важно отметить, что для клиента написана своя реализация libc.

Сервер написан на C++ с помощью LLVM, и его основными функциями являются:

- поднятие двоичного кода в LLVM IR с помощью Rellume;
- применение оптимизаций и кодогенерация.

В данной практике была проведена работа именно над клиентской частью Instrew.