Санкт-Петербургский государственный университет

Кафедра системного программирования

Динамическая бинарная трансляция в RISC-V с помощью Instrew

Михайлов Илья Игоревич, группа 21.Б07-мм

Научный руководитель: Д.С. Косарев, ассистент кафедры системного программирования

Санкт-Петербург
2024

Введение

- Бинарные трансляторы являются популярными инструментами для анализа кода, профиляции и эмуляции, важным классом которых являются динамические бинарные трансляторы (далее — ДБТ)
- В качестве промежуточного представления для ДБТ хорошо использовать LLVM IR
  - Тулчейн LLVM позволяет проводить качественные оптимизации
- Хотим ДБТ для RISC-V
- В результате работы хотим добавить поддержку архитектуры RISC-V для ДБТ с открытым исходным кодом

Обзор

- Динамические бинарные трансляторы:
  - QEMU
  - Box64
  - Rosetta 2
  - Valgrind
- Динамические бинарные трансляторы с поднятием в LLVM IR:
  - Remill + McSema, rev.ng, llvm-mctoll[1]
  - Instrew + Rellume
- Выводы:
  - Instrew работает быстрее, чем QEMU и Valgrind, а так же имеет API для добавления инструментов динамического бинарного анализа[2]

[1] https://se.math.spbu.ru/thesis/texts/Frolov_Timofej_Sergeevich_Spring_practice_2nd

[2] https://mediatum.ub.tum.de/doc/1614897/1614897.pdf

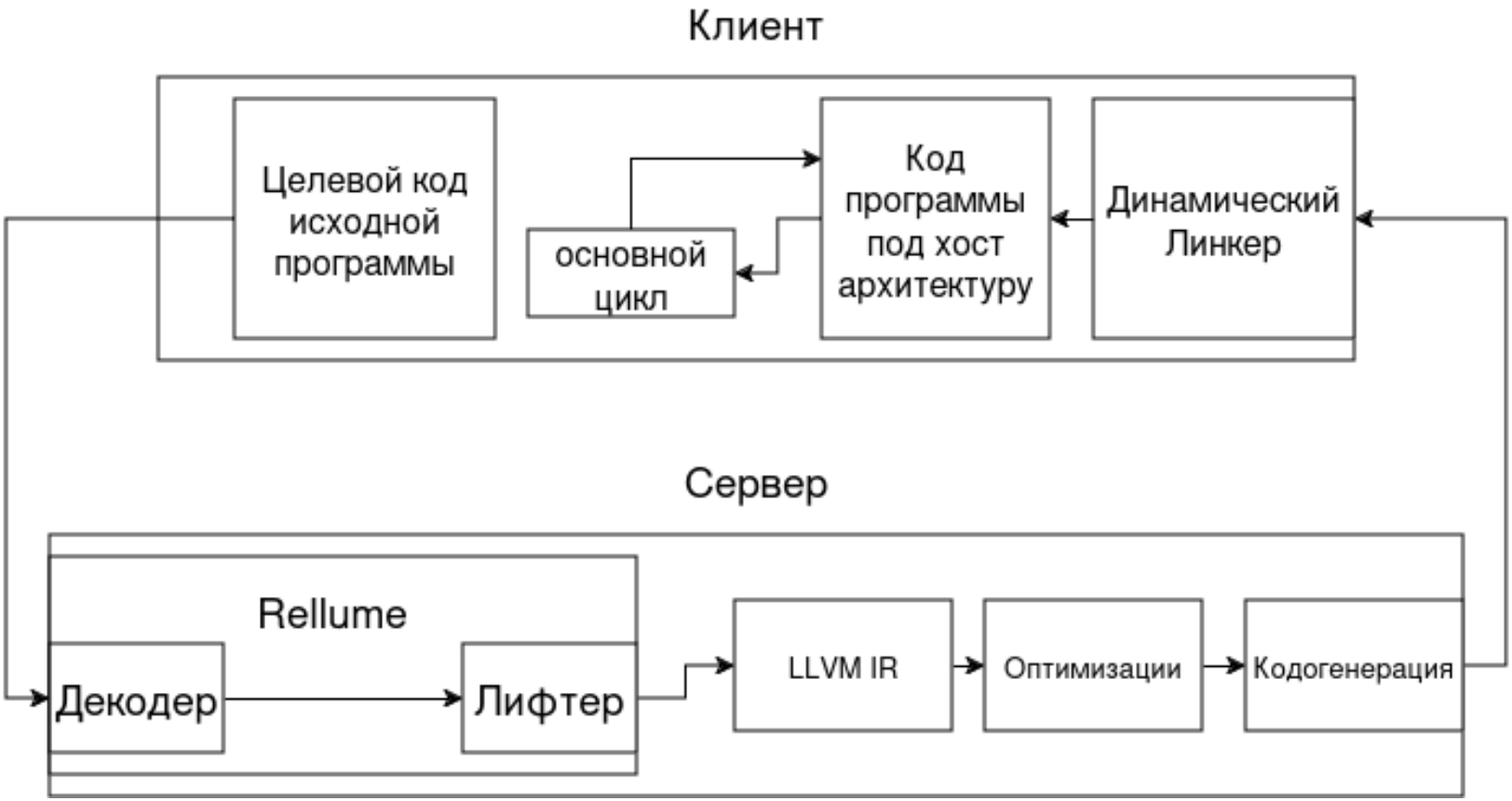
Постановка задачи

Цель: Добавить поддержку архитектуры RISC-V для ДБТ Instrew.

Задачи:

1. Выполнить обзор архитектуры Instrew
2. Реализовать минимальную функциональность для запуска Instrew на RISC-V:
   - Добавить процессорно-специфические патчи в minilibc и другие компоненты
   - Реализовать функции, связанные с RISC-V ABI
3. Протестировать на простых примерах

Архитектура Instrew



Реализация (1/2)

- Чтобы скомпилировать, необходимо:
  - Проставить константы
  - Сделать простой header на ассемблере, для входа в программу, после execve
- Из-за недостатка опыта с архитектурой RISC-V, необходимо было прочитать документацию[3][4]

[3] https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf
[4] https://github.com/riscv-non-isa/riscv-elf-psabi-doc/blob/master/riscv-elf.adoc

Особенности Instrew (грабли)

- Флаги сборки клиента Instrew:
  - В рантайме к instrew-client подключался ld-linux
  - Но такого быть не должно, так как instrew-client сам является линкером
  - Оказалось, что -static-pie флаг gcc не подразумевал собой -no-dynamic-linker флаг
- Динамические релокации:
  - Так как instrew-client собирается и линкуется без libc, ему нужно самому обработать свои релокации
  - Клиент загадочно падет на какой-то релокации
  - Оказалось, что над объявлением функции memset стоит аттрибут weak и из-за этого нужно реализовывать другую релокацию
- Некоторый вывод:
  - Из-за специфических флагов сборки и линковки, сложно диагностировать segfault
  - Asan и gprof здесь не работают, Valgrind сложен в использовании
  - Остается только printf и gdb

Реализация (2/2)

- После успешного запуска клиента Instrew нужно:
  - Реализовать эмуляцию системных вызовов для RISC-V
  - Реализовать некоторые релокации для запуска просто примера
- Оказалось, что:
  - Сервер запускает клиент Instrew через fork и fexecve
  - Возникли сложности с диагностированием с помощью gdb:
    - ★ follow-exec-mode не работал
    - ★ отдельно запуская клиент в гдб, не подгружались символы
  - Через некоторое время был найден флаг, который запускал клиент через execve
  - Некоторые функции в Instrew для диагностирования работают исправно, но не помогают для локализации ошибок
- Небольшой итог:
  - Удалось найти правильную комбинацию флагов, для диагностирования клиента с помощью gdb
  - Запустить простой пример не удалось из-за неправильного разрешения символов

Заключение

В результате работы были выполнены следующие задачи:

1. Выполнен обзор архитектуры Instrew и других ДБТ
2. Реализованы некоторые процессорно-специфические патчи
3. Реализована эмуляция системных вызовов
4. Реализованы архитектурно-специфичные ELF релокации