Министерство образования и науки Российской Федерации Московский физико-технический институт (государственный университет)

Физтех-школа радиотехники и компьютерных технологий
Кафедра Микропроцессорных технологий в интеллектуальных системах управления
Syntacore

Выпускная квалификационная работа бакалавра

Гибкий подход к подъёму LLVM MIR кода открытой архитектуры RISC-V в SSA форму LLVM IR

Автор:

Студент Б01-110 группы
Романов Александр Викторович

Научный руководитель:

Владимиров Константин Игоревич

Москва 2025

Аннотация

Гибкий подход к подъёму LLVM MIR кода открытой архитектуры RISC-V в SSA форму LLVM IR

Романов Александр Викторович

Проблема бинарной совместимости програм и их переносимости на разные архитектуры без возможности перекомпиляции часто решается при помощи бинарных трансляторов. Существует большое колличество статических и динамических бинарных трансляторов. Большинство из них работают либо за счёт прямого сопоставления инструкциям и регистрам исходной архитектуры инструкции и регистры целевой архитектуры, либо за счёт паттерн матчинга. Такие решения делают сложным поддержание новых исходных архитектур ввиду чего поддержка относительно молодой микропроцессорной архитектуры RISC-V в существующих трансляторах либо отсутсвует, либо сильно ограничена.

В данной работе рассмотрен новый инструмент для подъёма машинно зависимого представления RISC-V кода LLVM MIR в высокоуровневое машинно-независимое представление LLVM IR и его применение для простой статической трансляции бинарного RISC-V кода на любую поддержанную LLVM архитектуру.

Содержание

1 Введение

1.1 Бинарная совместимость

Бинарный код состоит из закодированных иструкций для конкретной архитектуры комманд. При компиляции программы её код на высокоуровневом языке програмирования (Например C/C++/Fortran) переводится в бинарный код целевой архитектуры и операционной системы.

Бинарной совместимостью называется возможность исполнения бинарного кода, скомпилированного под одну архитектуру команд и операционную систему на других устройствах и системах без модификации этой программы. Бинарная совместимость является одной из фундаментальных проблем в сфере компьютерных технологий в связи с постоянным развитием архитектур набора команд и операционных систем.

Основными проблемами для бинарной совместимости являются:

1. Различные архитектуры команд (ISA). Процессорные архитектуры являются главной причиной бинарной несовместимости. Процессоры каждой архитектуры исполняют свой уникальный набор команд и не работают с другими. Кроме различия в наборе инструкций архитектуры могут также отличаются размерос инструкций. Например, X86 и RISC-V поддерживают инструкции разной длины, в то время как ARM фиксирует длину всех инструкций в 4 байта. Архитектуры также отличаются набором регистров, принципами доступа к памяти а также порядком байт (например big-endian или little-endian). В то время как разница в наборе инструкций чаще всего влечёт за собой быструю остановку программы из-за невалидной инструкции, разница в порядке доступов к памяти при прочих равных может вызывать непредсказуемой поведение программы.
2. Операционные системы (ОС) также играют большую роль в бинарной несовместимости. Набор и мезханизм системных вызовов отличается на разных платформах (К примеру, `open` для Linux систем и для FreeBSD работают по разному, несмотря на общее название). Наборы системных вызовов также могут отличатся от версии к версии одной операционной системы. Например Windows не имеет фиксированного набора системных вызовов и они часто изменяются между версиями.
3. Соглашение о вызовах обычных функций (ABI) также значительно отличаются даже внутри одной архитектуры (Например программа, написанная под RISC-V процессор с LP64D не будет работать для RISC-V с LP64F).
4. Наконец, окружение запуска (Набор доступных на момент запуска динамических библиотек) также является критически важным для запуска программы и может

значительно отличаться как от машины к машине, так и на разных версиях операционной системы (Например программа, слинкованная динамически для операционной системы Ubuntu не сможет найти динамические библиотеки на устройстве с операционной системой Arch, т.к. эти библиотеки будут установлены по другим путям)

1.2 RISC-V

1.3 Компиляторы

2 Постановка Задачи

3 Обзор существующих решений

4 Requirements

4.1 Overview

4.2 Existing System

4.3 Proposed System

4.3.1 Functional Requirements

4.3.2 Quality Attributes

4.3.3 Constraints

4.4 System Models

4.4.1 Scenarios

4.4.2 Use Case Model

4.4.3 Analysis Object Model

4.4.4 Dynamic Model

4.4.5 User Interface

5 Architecture

5.1 Overview

5.2 Design Goals

5.3 Subsystem Decomposition

5.4 Hardware Software Mapping

List of Figures

Appendix A: Supplementary Material

– Supplementary Material –

Bibliography

[1] Marcus Aurelius, “Meditations.”