УДК 004.4'415

Генерация правил выбора инструкций в компиляторе на основе описания архитектуры набора команд центрального процессора

Никитин Б.П.[1], Шамшура Е.С.[1], Костяков А.С.[2]

[1] ФГАОУ ВО «Национальный исследовательский университет ИТМО», 197101, г. Санкт-Петербург, Кронверкский проспект, д.49, литер А.
[2] Lomonosov Research Institute.

Аннотация. На ранних этапах разработки новых процессоров, микроконтроллеров и другой аппаратуры используется подход кодизайна для получения готового инструментария широкого спектра: ассемблер, дизассемблер, симулятор и компилятор. В современных компиляторах правила выбора инструкций описываются вручную, увеличивая время поддержки новых или расширения существующих архитектур набора команд. В данной работе представлен способ автоматической конфигурации алгоритма выбора инструкций, использующий язык описания архитектур. В рамках исследования реализован генератор правил, который был применен для процессоров RV64IM и ARMv9. Полученный результат интегрирован в компилятор LLVM.

Введение. Ручное описание правил выбора машинных инструкций в компиляторе сталкивается с серьезными ограничениями по мере роста сложности процессоров. Современные архитектуры набора команд (НК) включают в себя более 1000 инструкций и десятки расширений, что делает их поддержку в компиляторе трудоемкой задачей. Учитывая ключевую роль компилятора в тестировании и оптимизации новых процессоров, необходимо автоматизировать процесс генерации правил выбора инструкций.

В настоящее время применяется технология программно-аппаратного кодизайна, позволяющая автоматически получать симулятор, ассемблер, дизассемблер [1] и бэкенд компилятора. Несмотря на большую роль векторных инструкций и операций с плавающей точкой в современных приложениях, существующие решения генерации правил выбора инструкций их не поддерживают [2], [3].

Таким образом, наше решение включает в себя создание современной C++ библиотеки для работы с иерархическими ациклическими направленными графами, алгоритма нормализации графа для поддержки новых типов инструкций и автоматическую интеграцию в бэкенд компилятора LLVM [4].

Основная часть. Разработанная в рамках исследования библиотека для работы с графами имеет иерархичную структуру, позволяющую ввести понятие векторной семантики, предикатов и исключений в рамках описания инструкции. Целью проектирования была поддержка расширений, содержащих векторные инструкции.

Основным алгоритмом является нормализация и фильтрация графа для приведения к единому виду, эффективному в контексте LLVM TableGen правил. Необходимо было написать развертку векторных конструкций, выделение общей семантики в двух предикативных блоках для поддержки арифметики с плавающей точкой, легализацию из операций языка описания архитектуры НК в операции на уровне SelectionDAG в LLVM.

Набор тестов был написан на LLVM IR с простыми арифметическими выражениями для проверки сгенерированных TableGen правил. В рамках тестирования была использована архитектура RV64IM [5] для целочисленных операций и ARMv9 для векторных инструкций и операций с плавающей точкой.

Выводы. Проведенное исследование демонстрирует, что конфигурация алгоритма выбора инструкций может быть произведена автоматически на основе языка описания архитектуры. Разработанные методы позволяют поддержать не только узкий набор целочисленных операций, но и вычисления с плавающей точкой и векторные расширения НК. Полученные результаты формируют основу для последующей автоматизации полной поддержки новых архитектур в компиляторе.

Список литературы.

1. S. Onder and R. Gupta, "Automatic generation of microarchitecture simulators," Proceedings of the 1998 International Conference on Computer Languages (Cat. No.98CB36225), Chicago, IL, USA, 1998, pp. 80-89, doi: 10.1109/ICCL.1998.674159.
2. Graf A. Compiler backend generation using the VADL processor description language. Diploma Thesis, Technische Universität Wien, 2021.
3. Drescher, F., & Engelke, A. (2026). Synthesizing Instruction Selection Back-Ends from ISA Specifications Made Practical. In Proceedings of the 2026 International Symposium on Code Generation and Optimization (CGO). Sydney, Australia. (To appear).
4. Lattner C., Adve V. LLVM: A compilation framework for lifelong program analysis and transformation // International Symposium on Code Generation and Optimization. 2004.
5. Cui E., Li T., Wei Q. Risc-v instruction set architecture extensions: A survey // IEEE Access. 2023. T. 11. C. 24696--24711.