УДК 004.94

СРАВНЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ БИБЛИОТЕК JIT-КОМПИЛЯЦИИ ДЛЯ РАЗРАБОТКИ ФУНКЦИОНАЛЬНОГО СИМУЛЯТОРА ЦЕНТРАЛЬНОГО ПРОЦЕССОРА

Державин А. А.[1], Шурыгин А. А.[1], Шамшура Е. С.[2]
Научный руководитель – старший преподаватель Петушков И. В.[1]
[1]МФТИ, [2]Университет ИТМО
egorshamshura@itmo.ru

Введение

Симуляция центрального процессора (ЦП) играет ключевую роль в проектировании, тестировании и оптимизации процессоров, а также в разработке программного обеспечения. Одним из базовых подходов симуляции ЦП является интерпретация, при которой каждая инструкция гостевой системы исполняется последовательно на хозяйской платформе. Эффективной альтернативой интерпретации является техника двоичной трансляции (ДТ [1]). Для целей симуляции полной системы обычно используют именно динамическую двоичную трансляцию (ДДТ). Её основная идея заключается в преобразовании блоков гостевого машинного кода в семантически эквивалентные последовательности инструкций хозяйской архитектуры “на лету”. Кэшируя трансляции “горячих” участков кода и оптимально чередуя фазы трансляции и исполнения кода, можно значительно повысить производительность модели процессора. Этот подход, известный также как JIT-компиляция [2].

В настоящее время существует множество симуляторов ЦП, использующих технику динамической двоичной трансляции, в разной степени близких к промышленной разработке. QEMU [3] является открытым высокопроизводительным эмулятором полной системы. Для ДТ в QEMU используется собственный JIT-компилятор TCG. Наиболее приближенным к научному сообществу является Pydgin [4] — это исследовательский инструмент, который использует мета-трассирующий JIT-компилятор на основе RPython для автоматической генерации высокопроизводительных симуляторов на основе ДДТ из собственного языка описания архитектуры.

Несмотря на имеющуюся потребность, в научной литературе отсутствуют актуальные исследования, проводящие широкий сравнительный анализ современных JIT-библиотек именно для целей симуляции ЦП с технологией ДДТ. Для восполнения этого пробела мы создали выборку из пяти современных библиотек: LLVM JIT [5], AsmJit [6], Xbyak [7], GNU Lightning [8], MIR [9]. В качестве объекта исследования выбрана популярная открытая RISC-архитектура RISC-V. Таким образом, наш вклад [10] включает создание модульного симулятора с перемежающимися режимами интерпретации и ДДТ, реализацию динамической трансляции поверх нескольких популярных JIT-библиотек, а также последующий сравнительный анализ производительности и удобства использования этих инструментов в современном C++ окружении.

Основная часть

Разработанная в рамках исследования модель ЦП имеет модульную архитектуру. Целью проектирования было обеспечение быстрого и гибкого процесса прототипирования различных подходов моделирования процессора с использованием техник ДДТ.

Ключевой частью движка трансляции является общая логика — адаптивный алгоритм ДДТ, определяющий режим исполнения: интерпретация, трансляция. Алгоритм основан на механизме подсчета количества исполнений базовых блоков.

Решение о трансляции принимается достижением порогового значения для счетчика выполнений. Данная логика динамической двоичной трансляции — адаптированный алгоритм из исследования, в котором детально описаны методы оптимизации высокоскоростной симуляции ЦП [2].

Набор использованных тестов производительности включает в себя как алгоритмические задачи (сортировки, рекурсивные вычисления), так и синтетические тесты (умножение). Все тесты были скомпилированы под архитектуру RV32I без стандартной библиотеки.

Выводы

Проведенное исследование демонстрирует, что выбор конкретной JIT-библиотеки оказывает критическое влияние на производительность симулятора, основанного на ДДТ. Таким образом, данное исследование предоставляет сравнительный анализ современных библиотек JIT-компиляции для задачи ДДТ при симуляции ЦП, учитывающий как производительность, так и аспекты практической интеграции.

Список литературы

1. John Aycock. 2003. A brief history of just-in-time. ACM Comput. Surv. 35, 2 (June 2003), 97–113. https://doi.org/10.1145/857076.857077.
2. Daniel Jones and Nigel Topham. 2008. High Speed CPU Simulation Using LTU Dynamic Binary Translation. In Proceedings of the 4th International Conference on High Performance Embedded Architectures and Compilers (HiPEAC '09). Springer-Verlag, Berlin, Heidelberg, 50–64. https://doi.org/10.1007/978-3-540-92990-1_6.
3. Bellard F. QEMU, a fast and portable dynamic translator // USENIX Annual Technical Conference, FREENIX Track. 2005. Т. 41. № 46. С. 10–55.
4. Lockhart D., Ilbeyi B., Batten C. Pydgin: Generating fast instruction set simulators from simple architecture descriptions with meta-tracing JIT compilers // 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE, 2015. С. 256–267.
5. Lattner C., Adve V. LLVM: A compilation framework for lifelong program analysis and transformation // International Symposium on Code Generation and Optimization. 2004.
6. asmjit. asmjit // GitHub. URL: https://github.com/asmjit/asmjit (дата обращения: 31.10.2025).
7. herumi. xbyak // GitHub. URL: https://github.com/herumi/xbyak (дата обращения: 31.10.2025).
8 GNU Project. GNU Lightning: a portable dynamic code generation system. GNU Press, 2023. URL: https://www.gnu.org/software/lightning/manual/lightning.html (дата обращения: 30.10.2025).
9. vnmakarov. mir // GitHub. URL: https://github.com/vnmakarov/mir (дата обращения: 31.10.2025).
10. ProteusLab. JitResearch // GitHub. URL: https://github.com/ProteusLab/JitResearch (дата обращения: 31.10.2025).