Арифметика и математические функции для порта OpenJDK на RISC-V

Купоросов В. В.

Научный руководитель: к.ф.-м.н. И. А. Близнец

Научный консультант: А. П. Козлов

Санкт-Петербургская школа физико-математических и компьютерных наук

НИУ ВШЭ – Санкт-Петербург

2020

Java и OpenJDK

- Программы на Java транслируются в байт-код
- Байт-код исполняется виртуальной машиной Java (JVM)
- Java Development Kit – набор средств разработки для Java
- OpenJDK – эталонная реализация JDK

OpenJDK на RISC-V

- RISC-V – открытый стандарт архитектуры процессора, разрабатываемый с 2010 года
- Используется для исследований и в реальных компьютерах
- Сложно получить доступ к физической реализации
- Качественного порта OpenJDK на RISC-V нет
  - Есть академические JVM на RISC-V (Jikes, Maxine)
  - Есть не оптимизированный Zero порт OpenJDK
  - Есть порты на другие процессоры (x86, PPC, ...)
- За основу был взят порт на PowerPC

Арифметика и математические функции

- Арифметические операции в Java представлены байт-кодами
  - Байт-коды реализуются на ассемблере
  - Реализация полностью зависит от платформы
- Математические функции могут быть реализованы разными способами:
  - Java-код в классе java.lang.Math
  - Более быстрая реализация на C++
  - Интринсик (intrinsic) – оптимизированная функция на ассемблере, вызываемая вместо Java-метода

Цель и задачи

Цель: реализовать в порте OpenJDK на RISC-V арифметические операции и математику

Задачи:

- Реализация арифметических байт-кодов на RISC-V
- Вызов статических методов из интерпретатора байт-кодов Java
- Вызов вспомогательных C++ функций из интерпретатора
- Реализация интринсиков на основе имеющихся в RISC-V инструкций
- Тестирование результата

Арифметические байт-коды

- Байт-коды реализуются в шаблонном интерпретаторе
  - Вычисления происходят на стеке, последнее значение кешируется в регистре для ускорения
- Сложение, вычитание, умножение, деление, битовые операции, сравнения… Всего 42 байт-кода
- Они работают над разными типами: int (32 бита), long (64 бита), float (32 бита), double (64 бита)
- Реализация некоторых операций получилась краткой
  - Потребовалось указывать режим округления для операций с плавающей точкой

Арифметические байт-коды

- Другие операции были существенно более сложными
  - При делении на ноль необходимо генерировать исключение
  - Необходимо отобразить семантику байт-кодов сравнения на инструкции RISC-V
  - Байт-коды для сравнения значений с плавающей точкой должны конкретным образом обрабатывать значение NaN

Вызов статических Java-методов

- Математические функции представлены статическими методами в классе java.lang.Math
- Вызов статических Java-методов кодируется байт-кодом **invokestatic**
- Процесс вызова:
  - По индексу метода ищется адрес соответствующего фрагмента кода
  - В зависимости от возвращаемого типа выбирается блок кода для возврата из метода
  - Выполняется код метода, а затем код выхода из него

Вызов C++ методов

- Некоторые Java-методы заменяются на вызов C++ реализации
- Процесс вызова:
  - Сохранение значений из nonvolatile регистров в память
  - Копирование аргументов в соответствующие регистры
  - Выделение кадра C-стека
  - Переход по адресу функции

Математические интринсики

- Доступные инструкции для интринсиков в RISC-V:
  - fabs, fsqrt, fmadd (a * b + c)
- Интринсики, не использующиеся в других портах:
  - fmin, fmax, abs
  - Для них необходимо править общий код

Тестирование

- Для отладки и тестирования используется QEMU RISC-V
- Сравниваются результаты тестов на x86 и RISC-V
- Тесты генерируются с помощью JASM tool
- Набор тестов составлен исходя из спецификации
- Fuzzing-тестирование (50000 случайных тестов) для дополнительного подтверждения

Тестирование

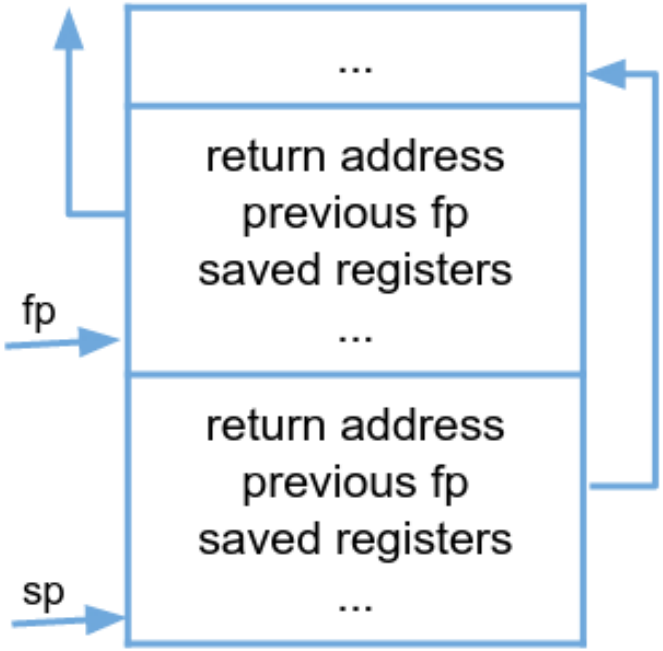
- Комбинации 18 значений для целочисленных операций
  - Переполнение, деление на ноль, отрицательные значения
- Комбинации 20 значений для вещественных операций
  - Округление, бесконечные значения, NaN, денормализованные числа
- Всего $\approx$ 7800 тестовых случаев на арифметику и $\approx$ 2500 на математику
- В процессе выполнения этих тестов было найдено и исправлено 5 ошибок
  - 2 из них не в арифметике

Результаты

- Реализованы все 42 арифметических байт-кода Java
- Реализованы статические вызовы Java-методов
- Разработан механизм вызова C++ функций для вспомогательных функций
- Реализованы 16 интринсиков, 11 из которых новые для OpenJDK
- Разработан набор тестов, покрывающий краевые случаи согласно спецификации JVM

Различия в структуре C-стека в RISC-V и PowerPC

Си-стек в RISC-V



Си-стек в PowerPC