LLCCEP: Машинно-независимая, абстрагированная от ОС, платформа для исполнения программ

Андрей Беззубцев

Цель и задачи работы

Целью настоящего проекта явилась разработка независимой от процессора и операционной системы платформы, реализующую подсистемы работы с окнами и файлами.

В частности, стояли задачи обеспечить:

- Высокоуровневые интерфейсы взаимодействия с файловой и оконной системой,
- Машинно-независимое исполнение программ,
- Абстракцию от ОС при исполнении.

Актуальность работы

Существует несколько аналогичных решений. В частности, это JVM. Ранние версии JVM исполняли всю программу на интерпретаторе, а лишь потом стал использоваться JIT-компилятор[2]. Однако, JVM вообще не поддерживает AOT-компиляцию. Очевидно, что сокрытия байт-кода от лишних глаз ~~смертных~~ при таком способе исполнения не предусмотрено. Так как некоторое время перед исполнением в JVM происходит компиляция в нативный код хост-процессора[2, 5, 6], снижается скорость работы по сравнению с максимально возможной для LLCCEP.

Второе подобное решение -- CIL, доступное только под Windows. Помимо интерпретатора и JIT-компилятора, CIL реализует и AOT-компилятор. Однако CIL не поддерживает на уровне ядра ряд интерфейсов, поддерживаемых в LLCCEP.

Третье же решение -- LLVM. Оно является кроссплатформенным, представляет такие же варианты сборки как и LLCCEP, однако отсутствует поддержка интерфейсов взаимодействия с ОС на уровне ядра, как и в CIL[8, 9].

Благодаря кроссплатформенности и высокой скорости исполнения программ, платформа LLCCEP опережает ряд существующих решений и предоставляет более гибкий инструментарий для отладки, исполнения и сборки программ.

Архитектура платформы

Платформа разработана в соответствии с модульным и конвейерным принципами.

Предусмотрено несколько сценариев действий с программами. Все они начинаются с того, что исходный код программы компилируется в байт-код виртуальной машины (ВМ). Далее возможны следующие варианты (см. рис. 1):

1) JIT-компиляция байт-кода в нативный машинный код процессора и исполнение программы на хост-процессоре[2, 5, 8].
2) AOT-компиляция[5] в нативный код процессора для конкретной операционной системы, с целью последующего исполнения программы на хост-процессоре.
3) Исполнение на ВМ в режиме интерпретации байт-кода[2].
4) Дизассемблирование и получение ассемблерного листинга программы[2].

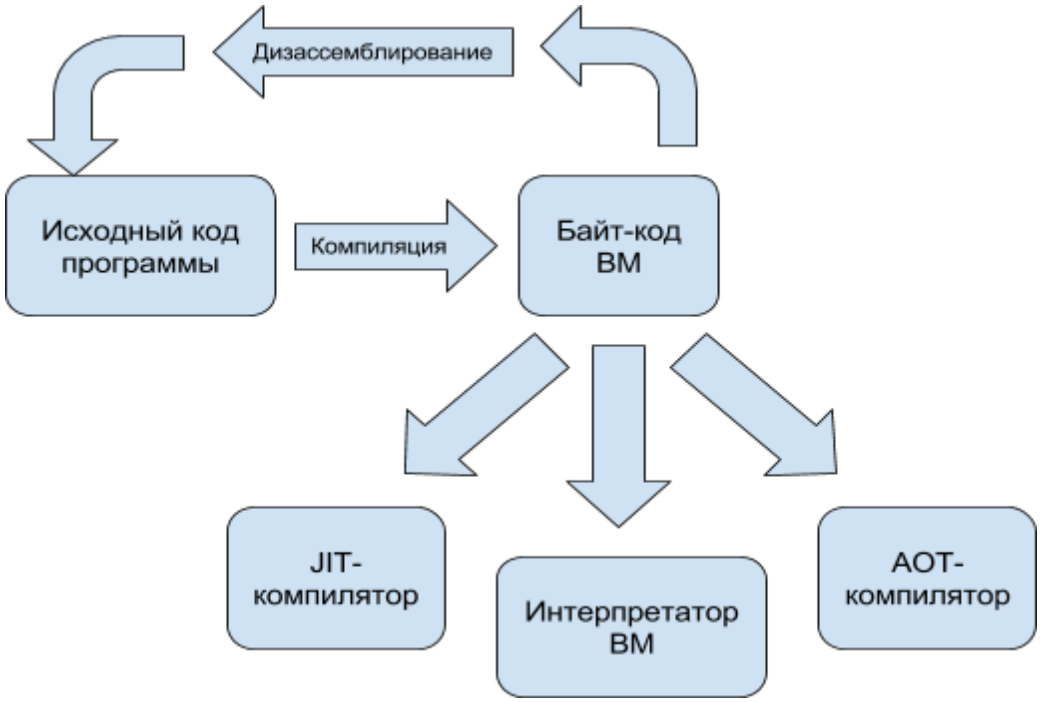

Рис. 1. Трансляция кода в LLCCEP.

Исполнение кода

JIT- и AOT-компиляторы транслируют байт-код ВМ в нативный код процессора, что позволяет в разы увеличить скорость исполнения, по сравнению с интерпретацией на ВМ[2]. Так как кодогенераторы в JIT- и AOT-компиляторах были написаны с использованием фреймворка LLVM[8, 9], который предоставляет гибкий инструментарий обработки ошибок и исключений, исполнение кода, сгенерированного ими, так же безопасно, как и на ВМ.

По-сути, AOT-компиляция является наилучшим решением для исполнения, так как по окончанию трансляции выдается исполняемый файл, что позволяет обеспечить:

- сокрытие исходного кода программы и исключение случайного его изменения,
- очень высокую скорость исполнения,

- дальнейшую независимость от модулей LLCCEP.

Результаты

- Реализованы высокоуровневые интерфейсы взаимодействия с оконной и файловой системой.
- Разработан кроссплатформенный программный комплекс, предназначенный для сборки и исполнения кода.
- Обеспечена полная абстракция от ОС при сборке и исполнении программ.

Информация о проекте

- 5000 строк кода на языках C, C++, LLVM IR в более 130 файлов.
- 6 программных модулей: ассемблер, дизассемблер, ВМ, отладчик, AOT-компилятор, JIT-компилятор.
- Репозиторий проекта - https://github.com/Andrew-Bezzubtsev/LLCCEP.

Список литературы

1. Д. Кнут. Искусство программирования для ЭВМ, Т.1: Основные алгоритмы.
2. Э. Таненбаум. Архитектура компьютера.
3. Э. Таненбаум. Операционные системы: разработка и реализация.
4. Б. Страуструп. Дизайн и эволюция C++.
5. Т. Пратт, М. Зелковиц. Языки программирования: разработка и реализация
6. А. Ахо, Р. Сети, Д. Ульман. Компиляторы: принципы, технологии и инструменты
7. AMD Inc., AMD64 Architecture Programmer’s Manual Volume 3: General-Purpose and System Instructions
8. LLVM Tutorial. Kaleidoscope: Implementing a Language with LLVM. http://llvm.org/docs/tutorial/LangImpl01.html
9. Create a working compiler with the LLVM framework. https://www.ibm.com/developerworks/library/os-createcompilerllvm1/