Университет ИТМО, факультет программной инженерии и компьютерной техники
Двухнедельная отчётная работа по «Информатике»: аннотация к статье
Дата лекции: __ 9 октября____ Дата сдачи: ____11 января____

Выполнил(а) _____Лагус М.С._____, № группы __3114___, оценка _____
Фамилия И.О. студента не заполнять

Название статьи/главы книги/видеолекции

Краткий и бодрый обзор архитектуры компиляторов

ФИО автора статьи (или e-mail)	**Дата публикации (не старше 2018 года)**	**Размер статьи (от 400 слов)**
Vk.team	"_15_" 05_2019 г.	____1270_

Прямая полная ссылка на источник и сокращённая ссылка (bit.ly, goo.gl, tr.im и т.п.)
https://habr.com/ru/company/vk/blog/451894/
https://clck.ru/ahTv2

Теги, ключевые слова или словосочетания
Компиляторы, Парсеры, Лексические и синтаксические анализаторы, LLVM, AST, архитектура компиляторов, формальные языки, атрибутные грамматики и семантики

Перечень фактов, упомянутых в статье

1. Почти все компиляторы построены по схожей архитектуре: front-end, middle-end, back-end.
2. Основная цель компилятора — трансляция языка высокого уровня абстракций в язык с более низким уровнем абстракций (например компиляция языка C в ассемблер).
3. Front-end компиляторов отвечает за построение AST по исходному коду, то есть перевод изначального кода программы в удобный для анализа и дальнейшей обработки вид. Чтобы построить синтаксическое дерево, используется сначала лексический анализатор — разбиение терминальных символов на токены, затем синтаксический анализатор или парсер — построение AST из предъявленных токенов. На этом же этапе обработки выполняется проверка соответствия исходного кода программы грамматике компилируемого языка.
4. Middle-end отвечает за семантический анализ AST, а также выведение типов, проверка на изменяемость переменных и построение графов потоков управления.
5. Back-end отвечает за, непосредственно, генерацию результирующего кода и оптимизации.

Позитивные следствия и/или достоинства описанной в статье технологии (минимум три пункта)
1. В данный момент существует несколько парсер-генераторов, которые по описанию грамматики вашего языка способны сгенерировать Front-end для вашего компилятора.
2. Существует проект LLVM, который содержит основное ядро back-enda, и который можно использовать в своём компиляторе, вместо того чтобы с нуля писать сложные оптимизации.
3. Есть популярный подход генерации байткода для виртуальной машины по типу JVM, главное преимущество в котором это портируемость скомпилированного кода на разные архитектуры

Негативные следствия и/или недостатки описанной в статье технологии (минимум три пункта)
1. Компиляторы — это очень сложно. Порог входа в разработку чего-то большего чем студенческий проект требует исключительных знаний во многих смежных областях.
2. Разрыв между индивидуальными проектами и индустриальными компиляторами по размеру и качеству кодовой базы сложно описуем. Преодолеть эту яму может помочь только опыт.
3. Несмотря на эпоху интернета, многие аспекты построение компиляторов не так сильно освещены в массах как другие отрасли IT. Как следствие, тяжелее найти единомышленников.

Ваши замечания, пожелания преподавателю *или* анекдот о программистах[1]
Данная тема представляется мне настолько же интересной и неизведанной, насколько и сложной и непонятной. Обучение на программной инженерии и знакомство с архитектурой компьютеров располагают к хотя бы поверхностному изучению затронутых аспектов построения компиляторов.

[1] Наличие этой графы не влияет на оценку