УНИВЕРСИТЕТ ИТМО

ФГАОУ ВО «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

---

# ПРОЕКТИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

**Лабораторная работа №1**

Анализ архитектуры вычислительной системы

---

Лабушев Тимофей

Группа P3402

Санкт-Петербург

2020

# Описание проекта

В работе описывается предстоящий дипломный проект: система статического анализа исполняемых файлов для графических процессоров AMD.

Решаемая проектом задача — поиск ошибок в потоке инструкций, которые могут быть совершены программистом на ассемблере или быть следствием неправильной компиляции кода из высокоуровнего языка. В качестве примера таких ошибок можно рассматривать отсутствие ожиданий (nop команд) при наличии зависимостей между инструкциями, которые не разрешаются аппаратно.

Правила, обнаружающие подобные ошибки, составляются пользователем. Рассматриваемая система отвечает за следующие этапы анализа:

1. дизассемблирование и чтение метаданных;
2. исполнение пользовательских анализаторов;
3. вывод результатов анализа.

Языком программирования для реализации системы выбран Haskell.

# Жизненный цикл

| **Этап жизненного цикла** | **Роль для системы** | **Методы и средства** |
|---|---|---|
| Составление и анализ требований | Создание технического задания, согласно которому должен проектироваться и разрабатываться проект. Нахождение правил, применимых к реальным задачам, которые могут использоваться для верификации готовой системы. | Исследование существующих средств статического анализа. Рассмотрение сценариев, в которых необходима проверка исполняемых файлов. Обсуждение с научным руководителем. |
| Проектирование системы | Определение архитектуры и внешних зависимостей. Разработка интерфейса, через который осуществляется взаимодействие с библиотекой (API). | Описание подсистем, составление UML диаграмм. |
| Разработка системы | Реализация библиотеки, включая автоматизированное тестирование для проверки функциональности. | Написание исходного кода и параллельное составление модульных, затем интеграционных тестов. |
| Создание образцов правил | Предоставление пользователям примера взаимодействия с библиотекой и решения реальных задач. | Реализация правил, найденных на этапе составления и анализа требований. |
| Применение системы | Демонстрирация применения статического анализа к исполняемому файлу для графических процессоров AMD. | Написание отчета о проделанной работе, защита. |
| Дальнейшее развитие системы | Анализ проделанной работы, исследование перспектив дальнейшего развития, в том числе поиск применения в производственных задачах. | Обсуждение результатов защиты с научным руководителем. |

# Архитектурные проблемы

1. **Обеспечение совместимости с различными наборами команд**

   На этапе дизассемблирования программе необходимо знать соответствие машинного кода инструкции ее текстовому представлению, которое различно для каждого семейства графических процессоров.

   Система должна включать в себя интеграцию с библиотеками LLVM, которые осуществляют дисассемблирование для различных семейств ГП.

2. **Предоставление выходного отчета в различных форматах**

   Найденные в ходе анализа проблемы могут потребоваться пользователю в различных представлениях: текстовой вывод для использования в интерфейсе командной строки, сериализованный формат (JSON, XML) для интеграции со средой разработки и т.д.

   Если вывод тесно связан с анализом, то добавление нового формата потребует от пользователя изменения всех анализаторов, что непрактично.

   Требуется промежуточный тип сообщения о проблеме, который возвращается анализатором и принимается на вход функцией форматирования. Он должен поддерживать не только строки, но и ссылки на инструкции, операнды и другие форматируемые элементы.

3. **Разрешение зависимостей между анализаторами**

   Результат работы одного анализатора может послужить служить входными данными для другого — это требуется для возможности создания промежуточных представлений программы (к примеру, упомянутого выше *control flow graph*).

   Необходимо предусмотреть выходной тип анализатора (помимо сообщений о проблемах), а также систему мемоизации выходных значений, чтобы не выполнять анализ повторно для одной и той же программы.

4. **Возможность параллельного выполнения анализаторов**

   При наличии большого числа анализаторов библиотека должна эффективно использовать доступные вычислительные ресурсы. Хотя входные данные для анализа иммутабельны, разрешение зависимостей между анализаторами может потребовать синхронизации.

   Необходимо предоставить пользователю интерфейс, инкапсулирующий общее состояние анализаторов (описанную в предыдущем пункте систему мемоизации) с использованием типов, поддерживающих одновременный доступ.

5. **Динамическая конфигурация анализаторов**

   Пользователю библиотеки может потребоваться передать определенные параметры каждому из анализаторов перед их запуском. Помимо этого, некоторые анализаторы могут быть неприменимы к текущему исполняемому файлу.

   Требуется интерфейс, позволяющий динамически добавлять анализаторы после того, как становятся доступными метаданные о программе, которые включают в себя семество ГП, от чего может зависеть применимость отдельных правил.

# Вывод

В ходе выполнения лабораторной работы были приобретены навыки описания жизненного цикла системы и анализа архитектурных проблем, влияющих на дальнейший процесс ее разработки.