# LLVM Snippy

МФТИ
Весна 2025

# Повторение

- Тестирование сложной системы нетривиально
- Полный перебор граничных условий может быть невозможен
- Для первичной оценки «качества» тестов используют покрытие
- **Основная часть покрытия – сгенерированные тесты**
- Референсный результат сгенерированного теста получают с эталонной модели (функциональной)

# Генератор случайных тестов

- Генерирует поток инструкций

# Генератор случайных тестов

- Генерирует поток инструкций
  - Случайная инструкция имеет риск оказаться невалидной

# Генератор случайных тестов

- Генерирует поток *корректных* инструкций

# Генератор случайных тестов

- Генерирует поток *корректных* инструкций
- Поток инструкций должен исполняться в заранее определенном окружении (execution environment)
  - Должны совпадать адреса расположения инструкций и обращений в память
  - В некоторых случаях должен соблюдаться ABI

# Генератор случайных тестов

- Генерирует поток *корректных* инструкций
- Поток инструкций должен исполняться в заранее определенном окружении (execution environment)
- Генерация должна быть конфигурироемой

# Генератор случайных тестов

- Генерирует поток *корректных* инструкций
- Поток инструкций должен исполняться в заранее определенном окружении (execution environment)
- Генерация должна быть конфигурироемой
  - «Честная» случайная генерация может давать неравномерное покрытие

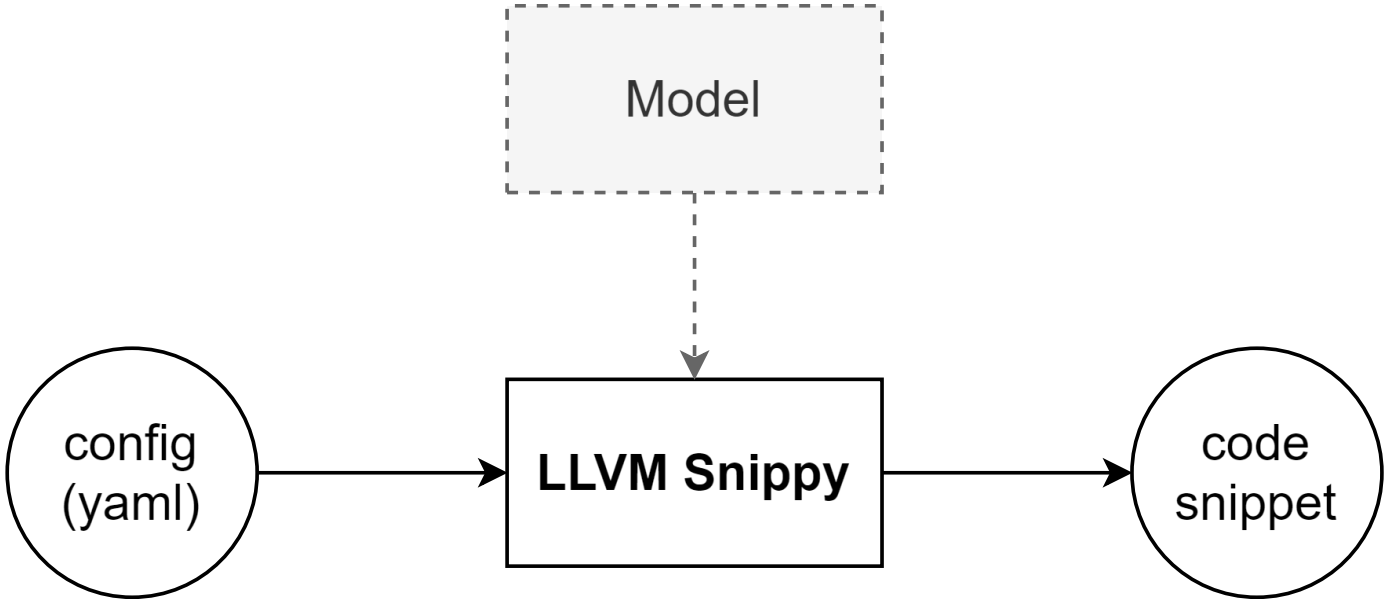
# Тестовый генератор LLVM Snippy

- Создает ассемблерные сниппеты с инструкциями с заданным случайным распределением
- Использует инфраструктуру LLVM
- Спроектирован кроссплатформенным
- Имеет встроенную возможность создавать трассы с помощью моделей
- Имеет возможность рандомизировать CF и вызовы функций
- Может вставлять код самопроверки корректности исполнения

syntacore/snippy

# Тестовый генератор LLVM Snippy

- Генератор конфигурируется с помощью yaml файла
- Модель может быть подключена прямо к генератору
- Результатом работы генератора является ассемблерный сниппет



# Минимальный конфиг LLVM Snippy

- Histogram:
  - ADD, 1.0
  - SUB, 2.0
  - ADDI, 4.0
  - LW, 3.0
  - BEQ, 5.8
- Задает набор разрешенных для генерации опкодов
- Каждому опкоду соответствует вес, соответствующий относительной вероятности его генерации

- Sections:
  - `no: 1`
    `VMA: 0x210000`
    `SIZE: 0x100000`
    `LMA: 0x210000`
    `ACCESS: rx`
  - `no: 2`
    `VMA: 0x100000`
    `SIZE: 0x100000`
    `LMA: 0x100000`
    `ACCESS: rw`
- Инструкции генерируются только в `rx`, обращения в память только в `rw`

# Структура проекта LLVM (как найти Snippy)

- Subproject – подпроект LLVM (Clang, LLVM, LLDB, …)
  - include – интерфейсные заголовочные файлы для библиотек проекта
  - lib – библиотеки, реализующие функциональность проекта
  - tools – инструменты, использующие проект как библиотеку
    - llvm-snippy
      - include – интерфейсные заголовочные файлы библиотек llvm-snippy
      - lib – библиотеки, реализующие функциональность llvm-snippy
      - llvm-snippy.cpp – вызов main

# To be continued …

На следующем занятии

- Подробнее рассмотрим как модель подключается к llvm-snippy
- Научимся запускать сниппеты в разных окружениях