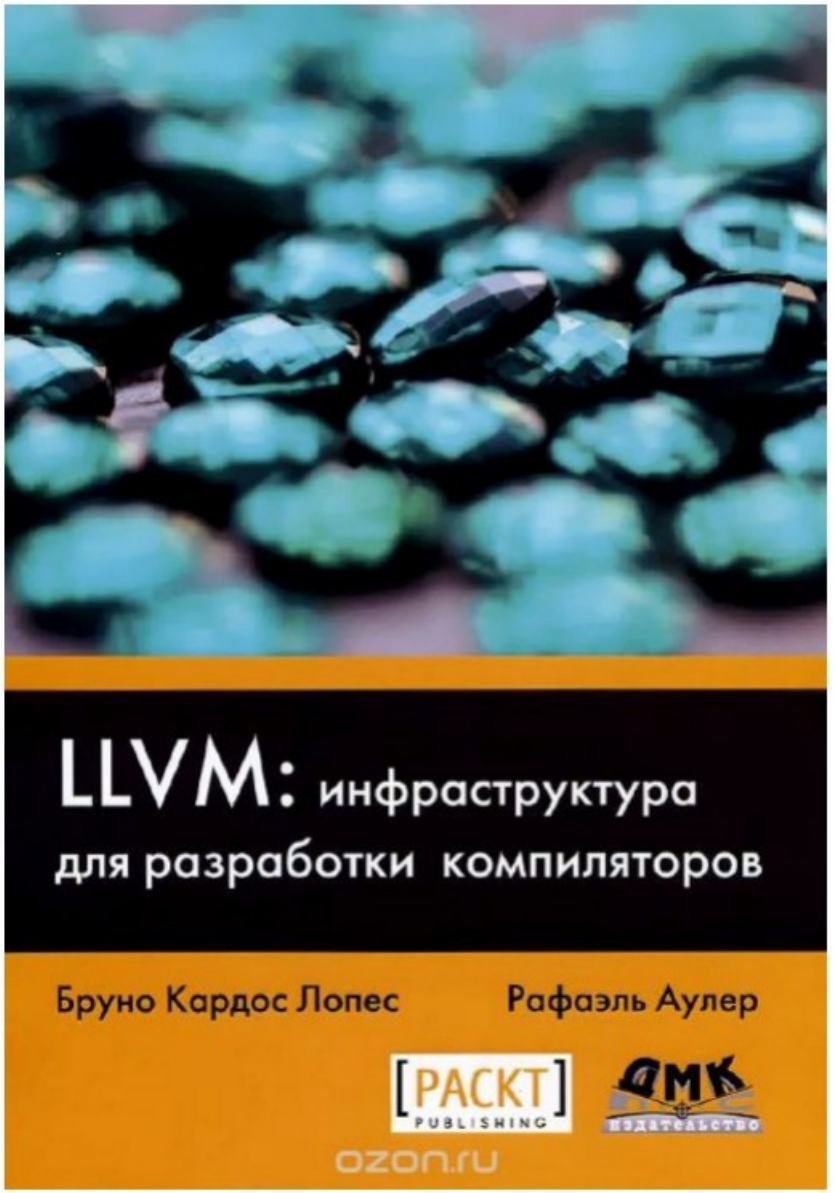
Как сделать свой компилятор на базе LLVM

Ресурсы

- http://www.llvm.org/ - сайт о LLVM
- http://www.llvm.org/docs/ - документация
- http://www.aosabook.org/en/llvm.html - введение в LLVM
- http://www.llvm.org/docs/CMake.html - как собирать LLVM
- http://www.aosabook.org/en/llvm.html - введение в LLVM

Ресурсы


LLVM: инфраструктура
для разработки компиляторов
Бруно Кардос Лопес
Рафаэль Аулер
[PACKT]
PUBLISHING
ДМК
издательство
ozon.ru

Как собрать LLVM

```
#!/usr/bin/bash

mkdir  -p  ./_build.llvm

cd ./_build.llvm


cmake \
    ..\llvm\llvm-11.0.0.src \
    -DCMAKE_C_FLAGS=" -O0 -fno-inline-functions " \
    -DCMAKE_CXX_FLAGS=" -O0 -fno-inline-functions " \
    -DLLVM_ENABLE_WERROR=1 \
    -DLLVM_CCACHE_BUILD=1 \
    -DLLVM_TARGETS_TO_BUILD="X86" \
    -DLLVM_ENABLE_PROJECTS="clang" \
    -DCLANG_ANALYZER_ENABLE_Z3_SOLVER=OFF \
    -DCMAKE_BUILD_TYPE=Debug \
    -DLLVM_ENABLE_ASSERTIONS=ON \
    -DCMAKE_VERBOSE_MAKEFILE=ON


make -j7
```

Собранный компилятор будет здесь

./_build.llvm/bin

Инициализация

```
llvm::LLVMContext& context = llvm::getGlobalContext();
llvm::Module *module = new llvm::Module("top", context);
llvm::IRBuilder<> builder(context);
llvm::FunctionType *funcType =
        llvm::FunctionType::get(builder.getInt32Ty(), false);
llvm::Function *mainFunc =
    llvm::Function::Create(
                                funcType,
                                llvm::Function::ExternalLinkage,
                                "main",
                                module);
```

Создание базового блока

```
// each Basic Block should be terminated
// by terminator instructions (br, ret, etc)
llvm::BasicBlock *entry =
        llvm::BasicBlock::Create(
                                        context,
                                        "entrypoint",
                                        mainFunc);

// current insertion point
builder.SetInsertPoint(entry);
```

Создание стековых переменных

```
llvm::Value *value_ptr_p = builder.CreateAlloca(
        Type::getDoubleTy(context), // type
        nullptr, // size
        "my_var");
llvm::Value *value_to_be_stored =
      ConstantFP::get(context, llvm::APFloat(0.5));
builder.CreateStore(value_to_be_stored, value_ptr_p);
```

=>

```
// %my_var = alloca double
// store double 0.5, double * %my_var
```

Арифметические операции

```
llvm::Value *value_1_p = ConstantFP::get(
                llvm::getGlobalContext(),
                llvm::APFloat(0.1));
llvm::Value *value_2_p = ConstantFP::get(
                llvm::getGlobalContext(),
                llvm::APFloat(0.1));
builder.CreateFAdd(
                value_1_p,
                value_2_p,
                "res");


=>


// %res = fadd f32 0.1, f32 0.2
```

Создание инструкции ветвления

```
builder.CreateBr(testbb);
        =>
// br label %label1

Value *test = builder.CreateOr(
                value_0, value_1, "testreg");
        =>
// %testreg = or i1 %test1, i1 %test1

builder->CreateCondBr(test, label1, label2);
        =>
// br i1 %testreg, label %label1, label %label2
```

Выход из функции

```
llvm::Value RetVal =
        builder.getInt32(3534855);
builder.CreateRet(RetVal);
```

=>

```
// ret i32 3534855
```

Получение LLVM кода

```
llvm::Module *module =
…
module->dump(); // распечатка в консоль
                // LLVM-кода в виде текста
```

Директории

- ./_build.llvm/include - *.h файлы
- ./_build.llvm/lib — бинарные библиотеки
- ./_build.llvm/bin — утилиты
- ./llvm-3.7.1.src/examples — здесь лежат примеры на LLVM (Kaleidoscope, HowToUseJIT и т.д.)

Утилиты

- llvm-as — LLVM ассемблер. Из текстового LLVM кода получить LLVM байт-код
- llvm-dis — LLVM дизассемблер. Из LLVM байт-кода получить текстовый LLVM код.
- llvm-opt - оптимизировать LLVM байт-код
- llc — кодогенератор. Из LLVM байт-кода получить исполняемый файл

Вопросы ?