目标代码生成详细设计

## 一、目标代码生成概述：

目标代码生成是编译的最后一个阶段。目标代码生成器把语义分析后或优化后的中间代码变换成目标代码。目标代码指计算机科学中编译器或汇编器处理源代码后所生成的代码，它一般由机器代码或接近于机器语言的代码组成。

在本次目标代码生成过程中，我们选择目标代码的格式为汇编语言代码，再通过编译链接，生成可执行文件。通过TargetMachine类返回DataLayout类，通过类提供的接口API成功生成汇编语言目标文件。

### 二、与中间代码的接口

在中间代码生成的过程中，会生成源代码的中间表示，并且在遍历AST抽象语法树的时候会完成类型检查。其生成的指令等信息会存储在context中，目标代码生成通过传入的context实例以及要输出的目标文件完成生成操作。

CodeGenContext context;

string output\_filename = argv[2];

ObjCodeGen(context, output\_filename);

CodeGenContext类实现如下

class CodeGenContext {

public:

    llvm::LLVMContext llvmContext;

    unique\_ptr<llvm::IRBuilder<>> builder;

    unique\_ptr<llvm::Module> Module;

    TypeSystem typeSystem;    //自定义类型系统

    CodeGenContext(): typeSystem(llvmContext) {

        //Module = make\_unique<llvm::Module>("Program", this->llvmContext);

        Module = make\_unique<llvm::Module>("main", this->llvmContext);

    }

    //初始化库函数

    void InitCodeGen();

};

其中Module类在中间代码生成部分，通过调用LLVM API，将函数插入指定的module。在目标代码生成阶段，通过调用module相关的API，生成目标代码。

Builder类是用于简化LLVM指令生成的辅助对象。IRBuilder类模板的实例可用于跟踪当前插入指令的位置，同时还带有用于生成新指令的方法。

## 三、各部分功能描述

###### 3.1 （根据本地运行环境）初始化llvm，设置目标机TargetMachine

    //初始化llvm，设置目标机

    llvm::InitializeAllTargetInfos();

    llvm::InitializeAllTargets();

    llvm::InitializeAllTargetMCs();

    llvm::InitializeAllAsmParsers();

    llvm::InitializeAllAsmPrinters();

###### 3.2 获取当前环境的三元组

    //获取目标三元组并设置

    auto targetTriple = llvm::sys::getDefaultTargetTriple();

    context.Module->setTargetTriple(targetTriple);

###### 3.3 获取目标机并设置DataLayout

    //获取目标机数据类型并设置

    string TargetError;

    auto target = llvm::TargetRegistry::lookupTarget(targetTriple, TargetError);

    if(!target){

        llvm::errs()<<TargetError;

        return;

    }

    auto CPU = "generic";

    auto features = "";

    llvm::TargetOptions options;

    auto relocationModel = llvm::Reloc::Model::PIC\_;

    auto theTargetMachine = target->createTargetMachine(targetTriple, CPU, features, options, relocationModel);

    context.Module->setDataLayout(theTargetMachine->createDataLayout());

###### 3.4 获取目标文件类型，确定要生成哪种类型的目标文件

    llvm::CodeGenFileType type;

    if(filename[filename.size()-1] == 's'){    //生成汇编文件

        type = llvm::CGFT\_AssemblyFile;

    }

    else if(filename[filename.size()-1] == 'o'){ //生成目标文件

        type = llvm::CGFT\_ObjectFile;

    }

    else{

        llvm::errs() << "Unsupported target file format: " + filename;

        return;

    }

###### 3.5 将目标代码输出到文件

    //将目标代码输出到文件

    error\_code ErrCode;

    llvm::raw\_fd\_ostream dest(filename.c\_str(), ErrCode, llvm::sys::fs::OF\_None);

    llvm::legacy::PassManager pass;

    if(theTargetMachine->addPassesToEmitFile(pass, dest, &dest, type)){

        llvm::errs() << "theTargetMachine can't emit a file of this type";

        return;

    }

    //cout<<"here!\n";

    pass.run(\*(context.Module));

    dest.flush();

目标代码生成总体设计并不难，只要正确调用API即可。而后按照以上顺序，先初始化llvm有关目标机的参数，接着获取本机有关目标机的三元组，数据类型，运行环境等。LLVM IR进入后端要经过pass优化，指令选择，指令调度，寄存器分配，代码布局优化以及汇编发行等过程。最后通过获取的参数生成目标机，再将目标代码输出到文件就可以啦。

## 四、数据结构说明

### 1、错误处理相关数据结构

在目标代码生成过程中，我们将输出的错误信息重定向到llvm:errs()中。最后可以通过日志输出错误信息。

###### 获取目标机失败

    string TargetError;

    auto target = llvm::TargetRegistry::lookupTarget(targetTriple, TargetError);

    if(!target){

        llvm::errs()<<TargetError;

        return;

    }

###### 输出文件类型有误

    llvm::CodeGenFileType type;

    if(filename[filename.size()-1] == 's'){    //生成汇编文件

        type = llvm::CGFT\_AssemblyFile;

    }

    else if(filename[filename.size()-1] == 'o'){ //生成目标文件

        type = llvm::CGFT\_ObjectFile;

    }

    else{

        llvm::errs() << "Unsupported target file format: " + filename;

        return;

    }

###### 目标代码输出文件失败

    error\_code ErrCode;

    llvm::raw\_fd\_ostream dest(filename.c\_str(), ErrCode, llvm::sys::fs::OF\_None);

    llvm::legacy::PassManager pass;

    if(theTargetMachine->addPassesToEmitFile(pass, dest, &dest, type)){

        llvm::errs() << "theTargetMachine can't emit a file of this type";

        return;

    }

## 五、算法描述

### 1、目标代码生成

**void ObjCodeGen(CodeGenContext& context, string& filename)**

功能：目标代码生成

参数值：CodeGenContext类型的实例，输出文件名

返回：void

详情：通过该实例以及目标机提供的API，找到本机的目标机类型，成功创建目标机，并把目标代码输出到指定文件。