**第三章-IR生成设计文档**

一．虚拟代码Codegen函数

为所有AST添加虚拟代码生成的函数，并在底部进行重写

Value \*codegen() override;

二.codegen里全局变量

static LLVMContext TheContext;

static IRBuilder<> Builder(TheContext);

static std::unique\_ptr<Module> TheModule;

static std::map<std::string, Value \*> NamedValues;

以及报错函数

Value \*LogErrorV(const char \*Str) {

LogError(Str);

return nullptr;

}

三.代码override过程中对数字类型的改变

所有的codegen函数参照万花筒，中间只需注意对INTERGER数字类型的变化.

1.整数处理

Value \*NumberExprAST::codegen() {

return ConstantInt::get(TheContext, APInt(32, Val,true));

}

2.运算符处理时

switch (Op) {

case '+':

return Builder.CreateAdd(L, R, "addtmp");

case '-':

return Builder.CreateSub(L, R, "subtmp");

case '\*':

return Builder.CreateMul(L, R, "multmp");

case '/':

L = Builder.CreateExactSDiv(L, R, "divtmp");

// Convert bool 0/1 to int 0 or 1

return Builder.CreateUIToFP(L, Type::getInt32Ty(TheContext), "booltmp");

default:

return LogErrorV("invalid binary operator");

}

3.函数代码生成

std::vector<Type\*> Integers(Args.size(),

Type::getInt32Ty(TheContext));

FunctionType \*FT =

FunctionType::get(Type::getInt32Ty(TheContext), Integers, false);