编译原理研讨课实验PR003实验报告

一、任务说明

本次任务负责完善elementWise从AST到LLVM IR的代码生成的通路。

二、成员组成

李昊宸 2017K8009929044

李颖彦 2017K8009929025

陆润宇 2017K8009929027

三、实验设计

在EmitAnyExpr函数中新增分支，跳向自定义的EmitVectorExpr()函数，这个函数负责给elementWise类型的元素生成对应LLVM IR代码。

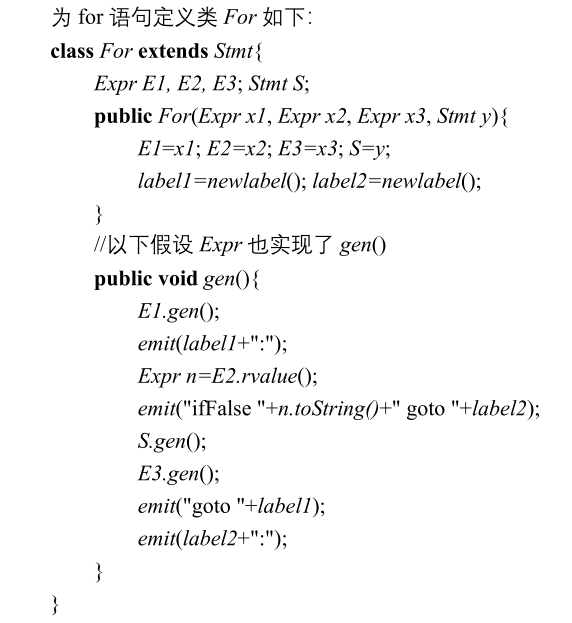
四、设计思路

将elementWise的部分转为对应的for循环来实现其功能。

例如将a = b转化为：

for (int i = 0; i < n ; i++) a[i] = b[i]

可以参考我们编译原理作业中为for循环生成中间代码的实现：

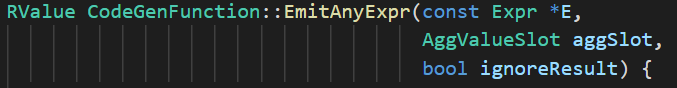


对应过来，E1为i = 0，E2为i < n，E3为i++，S为a[i] = b[i]。这样我们根据上面的代码生成片段，可以类似地完成本次实验的编写。

五、实验实现

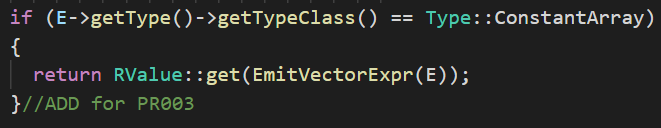
首先找到入口函数

From /home/clang7/llvm/tools/clang/lib/CodeGen/CGExpr.cpp



这个函数的作用是产生代码去计算具有任何类型的特定的表达式，并返回一个右值结构。

在这个函数最开始处，我们加入：



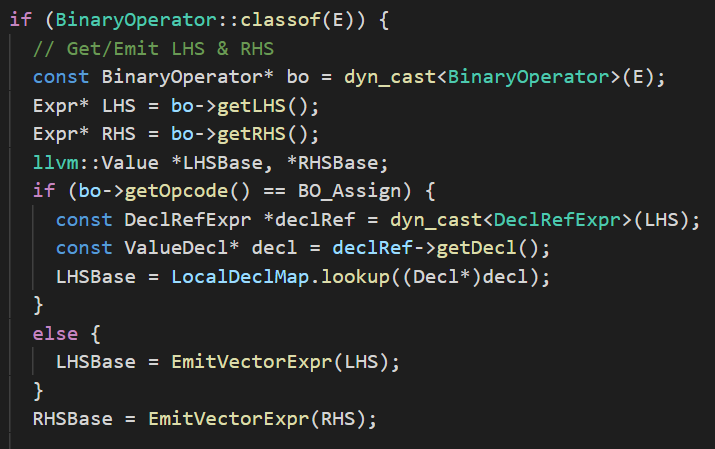
这个if分支专门用于应对形如a = b以及a + b这类数组运算的表达式，即elementWise的情况。

下面来看我们的重点，即EmitVectorExpr函数：



该函数负责为VectorExpr产生代码。

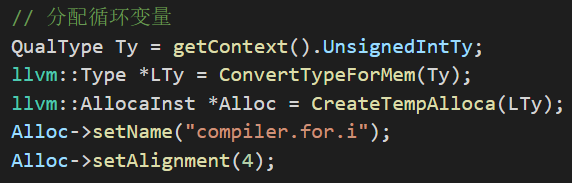
首先，表达式如果是二元运算符形式：



此时分两种情况，一种是二元运算符是赋值，那么LHSBase已经存在，需要去查找；否则，递归生成LHSBase。RHSBase始终采用递归生成。

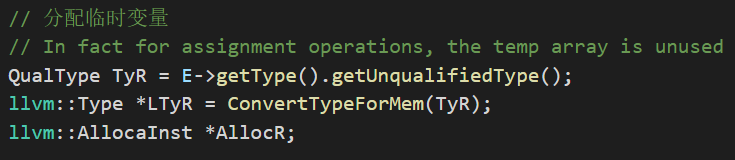
我们用for循环的形式来进行elementWise下的运算：

首先我们分配循环变量。注意变量以4字节对齐。

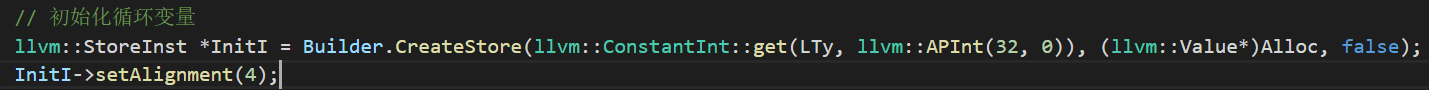


分配过程按照IR的语言格式。

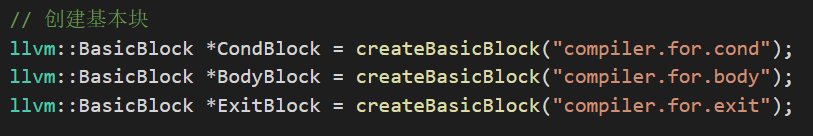
分配加法和乘法操作时所需要的临时数组。需要说明的是，如果运算符是赋值运算，临时数组还是需要被分配，虽然后续不会用到。



初始化循环变量。变量的初始化仿照老师给出的框架进行。



之后我们创建for循环所需要的三个基本块：

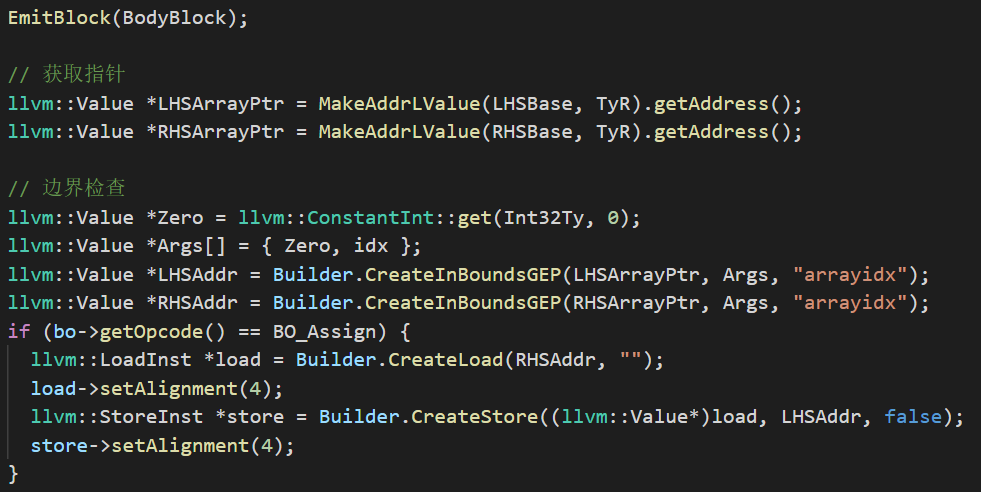


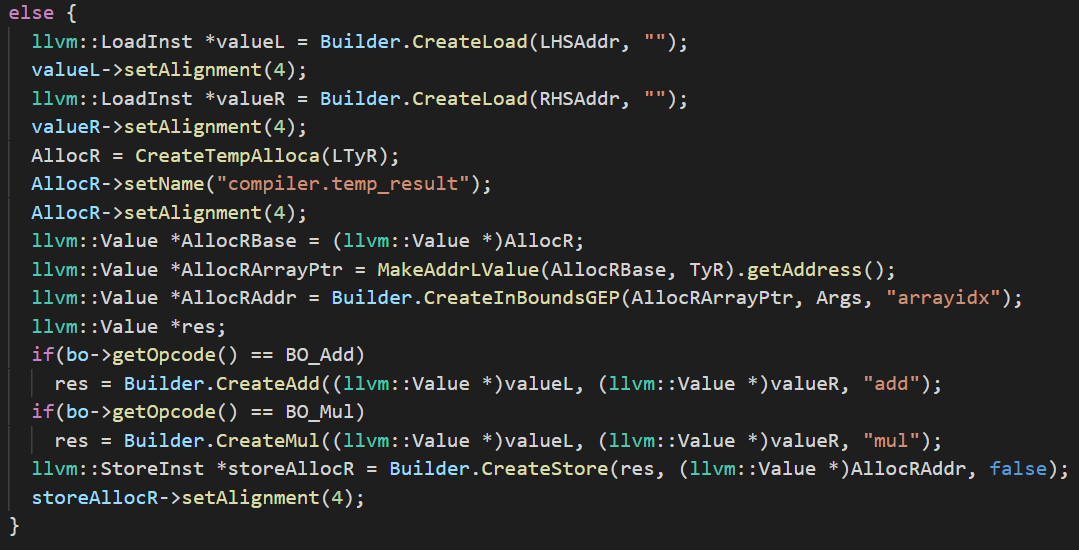
cond部分：



首先我们使用一条Load将循环变量读进来，并将该变量类型提升为Int，得到在Int类型下的值；接着我们获取数组的size所对应的值，用这两个值做Compare，并将Compare的结果作为操作数，生成一条条件跳转语句，分别跳转到body和exit对应的标号。

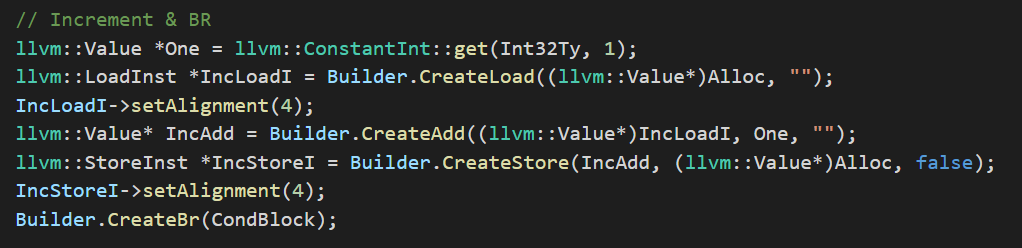
body部分：



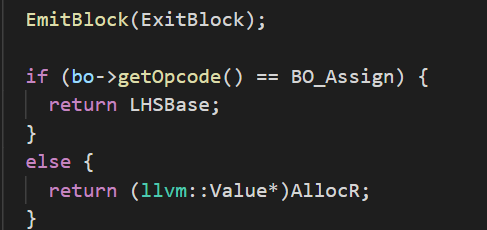


这里我们分别得到两个数组第i个元素的地址（假设i为循环变量的值）。如果是赋值，则直接用Load从右边地址位置读取元素到register中，再Store到左边地址位置中，由此实现值从右到左的传递；如果是加法或者乘法，则进行相应的运算并Store结果至临时空间中相应的位置，等到递归分析到赋值号时，将临时空间中的值传递到赋值号左端。

下面这一片段对应了循环变量值+1和跳转。

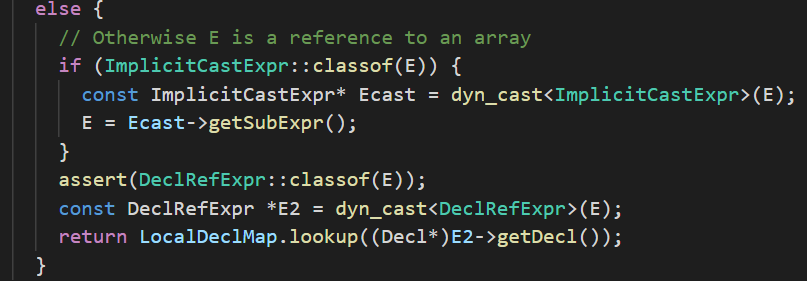


最后是exit部分：



可以看到，这个部分的IR其实只包含一个标号。

如果最初的if (BinaryOperator::classof(E))不成立，则跳转到下面的else部分，这部分实际上是代表常数组的引用（也就是递归生成RHSBase时，发现RHS只是一个数组表达式，那么执行递归函数的最底层，通过查询AST找到数组基址）：



六、实验结果总结

本次实验成功实现了任务要求，覆盖了各种要求情况。

七、总结

本次实验需要对LLVM IR的语言格式有一定的理解，还要能将for循环各部分具体实现。由于编译原理理论课上曾经也进行过中间代码生成的一些练习，因此我们可以将所学知识和代码框架的细节相结合，进而完成本次实验。

八、分成员总结

李昊宸：

本次实验负责实验代码的书写。起初在老师给出的基本框架下进行编写，但对于for循环部分的处理总是有些问题。于是经查阅资料，实现了通过建立基本块的方式，单独存放各部分变量的值。在修改主体，将参考资料给出书写基本块的形式嵌入到当前框架下后，可以进行正常的编译，实现了目的要求。主要的处理流程还是相对较为直观的，关键在于对clang下处理各种情况的函数仍然不够熟悉，导致过程较为困难。

李颖彦：

本次实验负责实验报告的书写。本次代码看上去代码量不算大，但实际上要接触很多新知识，需要对AST转LLVM IR的过程及方法有一个细致的理解，也要能对for循环进行底层实现。

陆润宇：

本次实验负责实验报告的书写和修改。可以看到，本次实验的代码结构基本与我们理论课上所学一致，而为了能与框架相结合，我们还需要进行例如类型提升这样的额外操作，并注意对齐等细节的实现。