编译大作业

第一部分

0. 补充内容

1. 输入的表达式文法：
   P为开始符号。终结符号为Id（字母、数字、下划线组成的非数字开头字符串），FloatV（仅考虑小数点表示法，不考虑科学计数法，无起始0），IntV（仅含数字，无起始0），分别表示变量名（可能是张量变量的名字或标量变量的名字），浮点数常量（32位），整数常量（32位）。
   其余符号解释:

```
S：语句
LHS：左值
RHS：右值
TRef：张量引用
SRef：标量引用
Const：常量
CList：常量列表
AList：变量列表
```

```
P ::= P S| S
S ::= LHS = RHS ;
LHS ::= TRef
RHS ::= RHS + RHS
        | RHS * RHS
        | RHS - RHS
        | RHS / RHS
        | RHS % RHS
        | RHS // RHS
        | (RHS)
        | TRef
        | SRef
        | Const
TRef ::= Id < CList > [ AList ]
SRef ::= Id < CList >
CList ::= CList , IntV | IntV
AList ::= AList , IdExpr | IdExpr
IdExpr ::= Id | IdExpr + IdExpr | IdExpr + IntV | IdExpr * IntV | IdExpr // IntV | IdExpr % IntV | (IdExpr)
Const ::= FloatV | IntV
```

另外，虽然文法允许张量名字出现在AList中，但是在语义上这是不允许的，所以在测试例子中不会出现这种输入。

1. 问题描述

在给定的代码里，你们看到的是一个简单的IR框架实现。正如之前让大家学习的TVM一样，我们要做的也是代码生成的内容，只不过我们做的事情比TVM简单了许多。

所谓代码生成，就是先要用某种IR表示出你想要生成的程序，然后扫描这个IR去打印出字符串，最终所有的字符串汇集在一起形成一个源代码。

我们已经在给定代码里展示了一个简单的IRPrinter的实现，它的作用是遍历IR打印出一种中间表示形式，虽然这个打印出来的字符串不是任何一种语言的源代码，但是只要按照这种思路去修改，可以做到生成任意语言的源代码。

在这里，我们需要同学们实现一个生成C/C++源代码的功能，你们需要学习IRPrinter的做法，然后自己动手实现一个代码生成的pass。

为了证明你们实现的代码生成是有效的，我们提供了10个例子，这10个例子描述的不同的运算，根据这些运算描述，你们可以利用你们实现的代码生成功能生成10个C/C++源代码程序，然后我们会检测你们生成的代码是否能通过编译且功能正确。

2. 测试方法

首先同学们要实现代码生成的功能，然后在 `project1/solution` 里编写接口应用，我们将会调用这里面的程序来生成所有测试例子的函数，生成的函数需要被安放在 `project1/kernels` 文件夹下，每个测试例子单独放在一个文件下，命名规则已经给出，请不要改变。为了测试同学们生成的代码，我们会运行 `project1/run.cc` 编译生成的目标程序，这个程序会检测所有 `kernels/` 下面的文件，然后逐个输入随机数据测试正确性。

3. 生成规则

你们需要生成的代码的信息被放在 `project1/cases/` 下的json文件里，比如

```
{
    "name": "kernel_example",
    "ins": ["B", "C"],
    "outs": ["A"],
    "data_type": "float",
    "kernel": "A<32, 16>[i, j] = C<32, 16>[i, j] * B<32, 16>[i, j];"
}
```

描述了你要生成一个函数名字叫做 `kernel_example` ，有三个参数，其中两个是作为输入 `"ins": ["B", "C"]` ， 一个是作为输出 `"outs": ["A"]` ，他们的数据类型是 `float` ，所以函数签名应该是

```
void kernel_example(float (&B)[32][16], float (&C)[32][16], float (&A)[32][16]);
```

事实上这些函数签名已经在 `run.h` 里给出了，不用同学们生成，同学们关注于生成函数体即可。
对于这个例子，你们需要生成的代码应该是如下：

```
// this is supposed to be generated by codegen tool
#include "../run.h"

void kernel_example(float (&B)[32][16], float (&C)[32][16], float (&A)[32][16]) {
    for (int i = 0; i < 32; ++i) {
        for (int j = 0; j < 16; ++j) {
            A[i][j] = B[i][j] * C[i][j];
        }
    }
}
```

一些固定的选项是生成一个头文件的包含，参数要用引用形式传入，输入放在输出的前面。

3.1 提示

有一些细节提示给同学们，希望注意

1. 在json文件里给出的表达式上，在 `<>` 中间标明了每个数组的大小
2. 如果数组的大小为 `<1>` ，表示这是一个标量，这时就不应该生成数组的形式了
3. 数据类型只有两种选择 `float` 和 `int`
4. 表达式可能不止一条，比如case5的表达式是两条，这两条所属的循环结构也是不同的，第一条比第二条多了一个k的循环，请注意这里的处理。
5. 有的表达式右侧出现的下标在左侧不出现时，表示一个求和，这符合爱因斯坦求和规范，比如case4里的矩阵乘法就是这样，k表示一个求和的循环，不要漏掉这个循环。
6. 输入输出可能是同一个，比如输入是A输出也是A，此时在函数签名上就不能同时出现两个A，而应该只有一个A，这里也可以看出来为什让大家传入参数是引用。另外，输入可能为空，但输出一定不为空。
7. 如何推断循环的范围呢？拿case6的卷积来说，p+r, q+s这种下标比较困惑，此时会保守地推测p和r的范围都是[0, 7)，但是如果看到在C中使用了r，就会推测它的范围是[0, 3)，我们约定，总是取最小的那个区间，并且保证所有的测试例子都符合这个约定，因此在case6里，r的范围是[0, 3)，其他例子也可以通过相似的方法推测循环边界，特别是case10也可以这样确定。

4. 评分

1. 满分100分，共10个测试例子，每通过一个得10分
2. 本部分内容占总成绩20%
3. 评分以小组为单位，每个小组2-4人，小组成员最终得分完全一致，请同学们自行组队，于4月17日前完成组队，将组队信息(姓名学号)发送到 `compiler2020spring@163.com` 。我们会给每个组一个组号。注意该分组将会共同完成第一部分和第二部分的内容，所以请谨慎选队友。
4. 目前公开6个测试例子，另外4个不公开，防止打表作弊。隐藏的4个测试例子不会比公开的例子复杂。
5. 任何抄袭、打表行为都会被记零分。
6. 编译不通过的提交记零分。
7. 提交截止日期是5月17日晚23:59:59，逾期提交一律记零分。提交需要包含一个说明文档或报告，汇报设计思路、实现方法、组内分工等，没有报告直接扣40分（到0为止）。提交一律将所有文件压缩后命名“组号+日期.zip”。仅最后一次提交为有效提交。
8. 截止日期后约一周我们会公开另外4个测试例子以及各组得分，如果得分不符合预期的，可以申诉，我们会再次运行当时提交的代码，重新检测结果，如果结果错误经助教评测属非人为造成意外（如编译器版本问题、操作系统版本问题）导致，可酌情重新定分，每组只能申诉最多一次。

5. 运行代码

一些同学可能不熟悉cmake，我们提供了方便的编译支持，只需要:

```
mkdir build
cd build
cmake ..
make -j 4
```

即可编译整个项目，编译涉及四个部分：

```
include和src下的文件
test中的文件
project1下的文件
project1/solution下的文件
```

其中你需要改动的应该是

```
include和src下的文件
project1/solution下的文件
```

在编译过后，进入 `build/project1` ，运行 `./test1` 即可测试所有例子。有兴趣的同学可以搞清楚这些编译关系，就可以清楚地知道自己要做的事情:-)

目前推荐在Ubuntu和Mac OS上进行开发，windows可以考虑使用WSL。
对于IR的使用，可以参考 `test` 目录下的两个例子。

6. 报告问题

如果你发现代码中不理解的地方，可以在微信群内讨论。
如果发现属于助教实现的代码bug，助教会及时修复。如果bug有价值，提出bug的组会被加分，每个bug加1分，最多10分（加分为折合总成绩之前的分数，最高不超过100分）。