Stage-5 实验报告

计03 王文琦 2020010915

实验内容

Step 11

`Step 11` 主要是引入了数组的定义， 包括了局部数组和全局数组。

- 引入数组需要在前端做更多的改变。在 `frontend/node/tree.py` 中新添加了 `TArray` 类型和 `IndexExpression` 类型。引入了新的数据类型意味着类型检查中需要添加更多的判断，而不是一视同仁全部视为 INT 类型变量。
- 中间代码生成过程，我们新增一条 TAC 代码 `Alloc` 代表出现局部数组的声明时，提前分配指定的空间。这里全局数组和局部数组的处理逻辑有很多地方不相同，例如在加载全局数组的值的时候只能通过其 `symbol` 获取地址的临时变量，再根据索引向内存中加载。其余指令均由之前的指令组合生成。
- 目标代码生成过程。目标代码生成部分增加了在栈空间上分配指定空间的过程。当然这些是针对局部数组的。具体操作就是将 SP 指针加减到合适的地址位置作为数组的首地址然后返回。

Step 12

`Step 12` 加入数组的初始化，以及数组函数传参。

- 前端增加了更多的语法分析项。具体可以参考代码注释。**这里需要坦白的一点是，关于类型检查的部分可能还存在着很多的瑕疵，因为时间的原因没有能够全部写完，但是现有的测例存在着一些疏漏的地方**。
- 中间代码生成过程在全局数组的赋值阶段和局部数组不相同，在这里为 `Funcvisitor` 额外设计了一个函数 `initglobal_vars()`，可以为全局函数赋初值。这里需要用到 `runtime` 的 `fill_n` 函数，调用的方法和 step 10 中相同即可。
- 目标代码生成过程没有需要改变的地方。

思考题

1. **Step 11:** C 语言规范规定，允许局部变量是可变长度的数组（Variable Length Array，VLA），在我们的实验中为了简化，选择不支持它。请你简要回答，如果我们决定支持一维的可变长度的数组(即允许类似 `int n = 5; int a[n];` 这种，但仍然不允许类似 `int n = ...; int m = ...; int a[n][m];` 这种)，而且要求数组仍然保存在栈上（即不允许用堆上的动态内存申请，如`malloc`等来实现它），应该在现有的实现基础上做出那些改动?

   答：首先语法分析支持变量出现在数组下标的位置，然后在 TAC 生成的时候，查看该变量是否有 int 数值挂载，如果有的话，那么将值取出，视作普通数组分配栈空间即可。如果是复杂类型的表达式，那么拒绝此类型的数组声明，报错即可。

2. **Step 12:** 作为函数参数的数组类型第一维可以为空。事实上，在 C/C++ 中即使标明了第一维的大小，类型检查依然会当作第一维是空的情况处理。如何理解这一设计?

   答：第一维是否为空并不影响，因为数组传参的本质实际上是传递数组的 `首地址` 和 `偏移计算方式`，当不影响这两个要素的传递方式应该都是可行的。

声明

本次 stage 作业向叶舟桐同学请教过相关问题。