stage-5 实验报告

计 93 王哲凡 2019011200

实验内容

step11：数组

在词法分析阶段，我引入了 Array 表示标识符后续可选择的下标运算（可为空）。同时为了区分左右值，我还自定义了 Reference 表示左值（Assign 的左侧），通过以下递归方式来完成左值的判定：

```
def p_ref_identifier(p):
    """
    Reference : Identifier
    """
    p[0] = Reference(p[1])


def p_ref_array(p):
    """
    Reference : Reference LBrack expression RBrack
    """
    p[0] = Reference(p[1], p[3])
```

同样，我添加了其对应的抽象语法树节点类 Reference 和基本的接口。

在符号表建立阶段，首先需要考虑全局数组的情况，我通过 size 的方式来判断和生成对应的全局变量（需要检查大小为正）。

而具体检查数组时，我此时仅判断了命名合法性，并添加了的对应的类型到符号表中，并未进行类型检查。

我将类型检查移至了 frontend/typecheck/typer.py，在其中主要对于 Reference 类型通过递归判定，检查了相关表达式的类型是否满足要求。

中间代码生成时，我添加了 A = ALLOC B 指令用于表示数组内存的申请分配，当声明变量不为内建类型时，我通过 ALLOC 分配其空间。并且在访问 Reference 时，根据其递归调用的方式，顺而计算其对应的实际偏移量（多维数组），并利用之前的 StoreMem 指令 和 LoadMem 访问与写回。

目标代码生成时，全局数组与全局变量类似，仅需根据数组大小动态调整 .space 的数即可。局部数组对应的 ALLOC 指令，我则采用了分别维护 array_size 的方式，记录当前开辟的数组总大小。通过 add 汇编实现 GetLocalAddr 指令表示获取 ALLOC 得到的数组首地址，并通过将数组划分为栈中最低位置以不影响栈帧大小的利用，而栈帧大小只须简单加上 4 倍的 array_size 即可。

实验思考题

step11：数组

1. 即对于此种带非常量的 ALLOC，我们不能维护 array_size，需要真实地动态改变 SP 的值用以分配栈空间。可以考虑使用一个新的固定寄存器表示动态分配前的 SP 地址（可称为静态 SP 地址），继续用于寄存器的存储、CallerSaved 寄存器的保存等，而真实的 SP 则可用于 ALLOC 非常量形式的分配（做减法），并将 SP 对应改变后的值作为 ALLOC 指令的返回值。而对于现有的 NativeStore/Load 方法，需要将其改为依赖于静态 SP 地址，以保证地址的不变性。