Lab3 实验报告

191870147 屈力

实现功能

完成了所有内容（必做+选做）

1、操作数的数据结构如下：

```
   struct Operand_ {
     enum { OP_VARIABLE, OP_CONSTANT, OP_ADDRESS,
     OP_ARRAY, OP_TEMP, OP_FUNCTION, OP_LABEL, OP_STRUCTURE} kind; //操作数类型
     union {
         int var_no;
         int const_value;
         int addr_no;
         int arrray_no;
         int temp_no;
         int label_no;
         char *func_name;
         int struct_no;
     } u; //不同类型对应的id
     Type type;//供OP_ARRAY使用
     int size;//供OP_ARRAY使用
     FieldList structure;//供OP_STRUCTURE使用
 };
 };
```

通过kind和u成员即可确定一个变量名，而剩余三个成员用于存储在语法树上dfs时遍历完子树所获得的信息，从而得到offset等信息来计算地址。

2、存储一句中间代码的数据结构按讲义处理，存储所有中间代码的数据结构通过双向链表实现，方便添加和删除。

3、中间代码的翻译

首先借助之前已经得到的语法树和符号表，在此基础上进行翻译。dfs的模式和lab2相近，不同之处在于有些函数可以省略（因为已经得到了符号表，所以不再需要诸如$VarDec\ LB\ BASIC_INT\ RB$，$Specifier$等反映变量类型的产生式）。大部分翻译模式讲义已经给出。
两个选做的翻译模式没有给出，因此在翻译$Exp \rightarrow ID$时首先需要对这两个情况额外处理。由于实验要求参数传递使用引用传递，无论是数组还是结构体，传参时必须先转化为地址，因此要对该ID是否

为参数进行判断，然后采取不同的处理方法（是参数的话要多一条地址赋值指令）。因此，在遇到表达式时（非结构体和数组的产生式），如果操作数类型是地址，需要生成中间代码，取出地址指向的值。

对于$Exp \rightarrow Exp_1\ LB\ Exp_2\ RB$产生式，首先递归处理$Exp_1$和$Exp_2$，然后根据两个返回操作数的$type$和$size$成员计算偏移量，同时生成地址赋值指令，并处理返回操作数的$type$和$size$。

对于$Exp \rightarrow Exp_1\ DOT\ ID$产生式。我在之前语义分析阶段新增了代码，计算每个结构体成员变量的偏移量，方便这一问题的处理。首先递归处理Exp_1，得到返回操作数的结构体类型，然后在符号表中查找ID，由于新增的代码，可以直接得到它的偏移量，同时生成地址赋值指令，并处理返回操作数的$structure$信息（如果Exp仍是结构体的话）。

4、我并没有对代码进行过多优化，只在翻译途中减少了一些冗余代码。比如，对于产生式$Exp \rightarrow ID$，就不再专门生成赋值代码，而是直接将ID信息拷贝到返回操作数供上层使用。另外我还做了常量折叠优化，在处理算术表达式和计算数组或结构体的偏移量时，若参与计算的源操作数均为常量，则不再生成计算过程代码，而是计算得到结果后直接赋值给相应的操作数。

编译方法

在当前目录下直接make（我删除了Makefile的-ly选项，因为这个选项一般没有影响，但是加上后我这里(Ubuntu 20.04)无法编译）。
在终端输入./parser ../Test/filename可以测试某一文件，或修改Makefile中的test目标然后直接输入make test以测试某一文件。