编译原理lab3实验报告

姓名：张玄逸	学号：201220194
日期：2022.11.18	邮箱：1822771416@qq.com

实现内容

- 全部必做要求
- 选做要求3.1，3.2

测试方法

代码可使用Makefile进行编译，得到可执行文件parser。

在文件根目录下输入 ./parser [文件名1] [文件名2]即可执行，如 ./parser Test/test1.cmm 1.ir，前一个作为输入文件，后一个作为输出文件，也可以忽略[文件名2]，如./parser Test/test1.cmm，此时输出到控制台。

实验思路

总体思路

采用lab2和lab3分离的方式，在完成语义分析后再次遍历语法树，进行中间代码的生成。

```
typedef struct Operand_
{
    enum Operand_kind kind; //不同的类型，如立即数，参数，函数，临时变量，label
    enum Operand_type       //是普通操作数还是取地址，取值
    {
        Normal,
        Address,
        Star
    } type;
    char name[30];
    int value;
    Type *variable; //对应的变量
} Operand;
```

和lab2不同的是，lab3中不存在“变量”的概念，因为所有的变量都是在函数中定义的临时变量，与参数相当，需要重新安排一个名称避免重复。而variable用于记录该操作数对应的真实变量类型。

基本要求

下面列举部分。

- 变量，数组，结构体的翻译

trans_exp函数这一次返回Operand*类型，表示存储该exp运算结果的变量。同时因为数组和结构体需要递归解析，所以在找到对应变量后需要将其Type类型附加到Operand上。

```
else if (strcmp(k->child->name, "ID") == 0)
        {
            //不能直接返回名称，因为函数参数名已重置
            ListNode *tmp = search_all_listnode(define,
                                                k->child->val.char_val,
false);
            now = new_para();
            now->value = tmp->para_no;
            now->variable = tmp->type; //附加变量
            ...
        }
```

- 参数与临时变量

  lab3中的参数可以认为全部是普通类型，因为函数传参时数组和结构体都传了地址，因此使用时不必再取地址。而函数内部定义的临时变量使用时需要取地址。

  所以在记录变量编号时，参数编号为负数，但二者绝对值都按照递增规律分配，以此区分。

附加要求

- 结构体变量

  在lab2已经实现了结构体的解析，因此现在只用在符号表中查找对应变量并附加在相应的Operand上即可。取成员变量时，遍历所有成员，同时记录经过的成员空间大小总和，就可以在结构体首地址的基础上加对应的空间得到某个成员。注意传参时传入地址。

- 数组

  和结构体类似，传参和使用的时候都是地址。从左往右时，将左边已计算好的地址加上当前维度所占的空间，最后一个维度时将临时变量改为Star类型，因为地址计算完后需要取值。

其它工作

- 符号表的设置

  在lab2中，考虑到散列表空间开销大，简单链表搜索时间长，我采用了AVL树来存储符号，并用AVL节点的栈来模拟嵌套的变量定义。但这带来的问题是：当语义分析完毕，栈也会随之清空，虽然所有的符号都还存在，但根节点已被从栈中清除，无法访问。

  如果要在不改动lab2代码的情况下使用符号表，就必须在语义分析的同时生成中间代码，整个实验的代码会变得冗长混乱。我的解决方法如下。

  - 修改语法树节点，增加一个指向自己对应的符号表（AVL树节点）的指针。因为每一次嵌套都是在新的Compst内部，所以将其作为符号表的保存节点，这样就可以获取到该语句块内的符号。而Program作为终结符号，自然就存储了所有的全局变量。
  - 当语义分析结束后，Compst和program节点保存了自己对应的符号表。在lab3中间代码生成时，就可以将它们的符号表头指针重新入栈，完成符号表的再次装填。

- 一些中间代码优化
  - 在return和(label或fuction)之间的语句不可能被访问

- 利用指令自然流动去掉与对应LABEL相邻的GOTO
- 清除未被引用的LABEL（由上一条产生）
- if x relop y goto label1
  goto label2
  label1 一类的语句
  改为if x !relop y goto label2
- 常数之间的运算可以直接赋值，去掉临时变量
- 去掉特殊四则运算，如自己加0，乘1
- 一些测试样例在优化前后的对比

[!] your username: 201220194,	[!] your username: 201220194, your groupType: 0
[!] testing sample 'M1'	[!] testing sample 'M1'
[pass] 'M1', instructions=28	[pass] 'M1', instructions=25
[!] testing sample 'M2'	[!] testing sample 'M2'
[pass] 'M2', instructions=55	[pass] 'M2', instructions=50
[!] testing sample 'O1'	[!] testing sample 'O1'
[pass] 'O1', instructions=22	[pass] 'O1', instructions=18
[!] testing sample 'O2'	[!] testing sample 'O2'
[pass] 'O2', instructions=99	[pass] 'O2', instructions=84
[summary-of-sample-tester] th	[summary-of-sample-tester] the code passed sample test 4/4
[!] testing 'A-1'	[!] testing 'A-1'
[pass] 'A-1', instructions=33	[pass] 'A-1', instructions=33
[!] testing 'A-2'	[!] testing 'A-2'
[pass] 'A-2', instructions=78	[pass] 'A-2', instructions=68
[!] testing 'A-3'	[!] testing 'A-3'
[pass] 'A-3', instructions=2325	[pass] 'A-3', instructions=2270
[!] testing 'A-4'	[!] testing 'A-4'
[pass] 'A-4', instructions=1484	[pass] 'A-4', instructions=1428
[!] testing 'A-5'	[!] testing 'A-5'
[pass] 'A-5', instructions=103	[pass] 'A-5', instructions=101
[!] testing 'B-1'	[!] testing 'B-1'
[pass] 'B-1', instructions=366	[pass] 'B-1', instructions=352
[!] testing 'B-2'	[!] testing 'B-2'
[pass] 'B-2', instructions=503	[pass] 'B-2', instructions=462
[!] testing 'B-3'	[!] testing 'B-3'
[pass] 'B-3', instructions=1011	[pass] 'B-3', instructions=931
[!] testing 'C-1'	[!] testing 'C-1'
[pass] 'C-1', instructions=897	[pass] 'C-1', instructions=850
[!] testing 'C-2'	[!] testing 'C-2'
[pass] 'C-2', instructions=908	[pass] 'C-2', instructions=741
[!] testing 'D-1'	[!] testing 'D-1'
[pass] 'D-1', instructions=9661	[pass] 'D-1', instructions=9530
[!] testing 'E1-1'	[!] testing 'E1-1'
[pass] 'E1-1', instructions=54	[pass] 'E1-1', instructions=44
[!] testing 'E1-2'	[!] testing 'E1-2'
[pass] 'E1-2', instructions=401	[pass] 'E1-2', instructions=377
[!] testing 'E1-3'	[!] testing 'E1-3'
[pass] 'E1-3', instructions=996	[pass] 'E1-3', instructions=922
[!] testing 'E2-1'	[!] testing 'E2-1'
[pass] 'E2-1', instructions=825	[pass] 'E2-1', instructions=809
[!] testing 'E2-2'	[!] testing 'E2-2'
[pass] 'E2-2', instructions=397	[pass] 'E2-2', instructions=367
[!] testing 'E2-3'	[!] testing 'E2-3'
[pass] 'E2-3', instructions=584	[pass] 'E2-3', instructions=553
[summary] the code passed tes	[summary] the code passed test 17/17, your score is 100
[score-log]: {'A-1': 6, 'A-2': 6, 'A-3': 6, 'A-4': 6, 'A-5': 6, 'B-1': 6, 'B-2': 6, 'B-3': 6, '(	

左侧为原始代码，右侧为优化后。