编译原理实验三实验报告

171860607

白晋斌

810594956@qq.com

任务号:14

目录

实验要求

实验三要求基于 7 种假设(假设 2,3 可能因后面的不同选做要求而有所改变).程序需要将符合以上假设的 C--源代码翻译为中间代码.

此外,我们需要完成要求 3.2,即修改假设 2,3 为: 1)一维数组类型的变量可以作为函数参数(但函数不会返回一维数组类型的值)。 2) 可以出现高维数组类型的变量(但高维数组类型的变量不会作为函数的参数或返回类值)。

测试环境

测试环境如下,值得注意的是,gcc 版本为 5.5.0,与助教测试机接近,不再是讲义所述的上古 gcc 版本.

~~gcc version 4.7.4 (Ubuntu/Linaro 4.7.4-3ubuntu12)~~(实验一)

gcc version 5.5.0 20171010 (Ubuntu 5.5.0-12ubuntu1~16.04)

flex 2.6.0

bison (GNU Bison) 3.0.4

运行方法

make clean; make; ./parser xxx.cmm out1.ir

实验步骤

1.数据结构

数据结构采用讲义所述双向链表串联 InterCode 的方法,并进行适当拓展.

2.线形 IR 打印

结合讲义 70 页中间代码的形式与操作规范,我们通过 switch 语句可以轻松打印出对应的 InterCode 和 Operand.

3.线性 IR 建立

借助实验二建立的语法分析过程以及江一中对应的语句的翻译模式,我们可以轻松的建立线性 IR,其中包括:基本表达是的翻译、语句的翻译、条件表达式的翻译、函数调用的翻译、函数参数的翻译.

值得注意的是,我们的翻译要严格按照讲义的语法来,切忌胡乱放飞自我.例如,对于语句 if(a=0),按照 c 语言语法赋值语句条件为 true,按照 cmm 语法条件为 a,如果不安讲义来,会产生很多无法意料的错误.

此外,关于条件表达式的翻译,乍一看可能不够全面,因此自作主张加了很多,但后来发现不仅是多余的,而且会干扰正常代码 other cases 的执行.因此还是要严格按照讲义所述的语法进行.

4.高维数组的实现

这里以代码文件 lab2.h 的 1246-1345 行为例说明.代码思路如下:

```
if(!(t1->kind==ARRAY &&t2->kind==ARRAY)){
        }
else{//t1,t2 都是数组
        int copyNum;//计算需要复制的元素数,以 t1,t2 中小的为准
        //...
        for(int i=0;i<copyNum;i++){
                //分别获得 t1,t2 对应+i*4 的位置 p1,p2
                //从 p2 位置获得值,赋值给 p1 位置
        }
        if(place!=NULL){
        //返回 t1
        }
}
```

5.代码优化

中间代码优化主要包括对常数进行提前四则运算的优化、对 GOTO 语句部分顺序的调转优化、删除部分无用标签.

遇到的问题

一路遇到很多问题,这里挑选一部分经典问题与对应解决方案链接列出:

[1]https://blog.csdn.net/qq_30242609/article/details/52858115
error: a label can only be part of a statement and a declaration is not a statement
在 switch 语句的每个 case 下要定义变量的话,要用大括号括起来.

[2] https://www.cnblogs.com/stoneJin/archive/2011/09/16/2179248.html
memcpy 和 strcpy 的区别

[3] https://blog.csdn.net/Rafe_ma/article/details/68944842