编译原理实验

第三次实验报告

孙楷文 学号：111100027 （5班）

1. 代码文件

实验源代码文件包含：

iropt.c	main.c	semantic.h	syntaxtree.h
iropt.h	Makefile	syntax.y	**translate.c**
lexical.l	semantic.c	syntaxtree.c	**translate.h**

其中用红色标出的四个文件为 Lab3的主要内容。

编译和使用方法：

用 Makefile 生成名为“g--”的可执行程序，运行./g-- <c--源代码文件名> <中间代码文件名>，如：

```
./g-- test.cmm test.ir
```

另外，用 Makefile 生成的其他部分文件也在提交的代码中。

2. 数据结构

中间代码采用双链表记录。为了提高链接合并代码段时的效率，同时记录了双链表的首和尾。

3. 翻译思路

翻译过程由 void generateIrCode();函数驱动，遍历语法树。对需要翻译为中间代码的语法树节点用 struct CodeSegment translate_node_xxxxx();函数翻译，返回值为该节点下翻译得到的中间代码段。值得

一提的是 translate_node_Exp();函数，为了 Exp 要在中间代码中告诉上层自己的运算结果，故由调用 Exp 的上层函数生成一个临时变量名 place，告诉 Exp，Exp 在计算的末尾通过 x:=y 格式的中间代码把计算结果赋值给上层生成的临时变量 place。但 Exp 生成的可能是一个普通变量，也可能是一个常数，甚至是一个指针。因而需要伴随着 place 由 Exp 告诉上层调用者该 place 的建议使用方法。

4. 避免临时变量名和 label 名与 c--代码中变量重名

临时变量名初始时设置为“tmp+数字”，LABEL 名初始设置为“label+数字”。普通的变量名直接取自 c--源代码中的变量名。但万一 c--代码中有名为 tmp3的变量怎么办呢？在进行中间代码翻译之前，会遍历符号表，一旦发现有名为“tmp***”的变量，就把临时变量的生成模式改为“tmpp+数字”，并重新以“tmpp***”查找符号表中的重名，直至“tmpp...pp+数字”的格式不与符号表中的任何变量名冲突。label 名的处理方式类似。

5. 中间代码的优化

中间代码的优化在 iropt.c 和 iropt.h 中实现，由 void iroptimizer();函数驱动。循环检查/执行下列6个优化字模块，直至某次循环中的6个子模块均没有对代码产生变动：

<table>
<tr><th>函数名</th><th colspan="2">函数功能概述</th></tr>
<tr><td>opt_usageCheck();</td><td colspan="2">检查并删除中间代码中生成左值后不再被用过的左值变量（为保持用户体验，READ 产生的左值除外）</td></tr>
<tr><td>opt_traceRightOpd();</td><td colspan="2">对于所有...:=...[op ...]形式的中间代码，如果等号右侧的变量 x 是之前由 s 经 x := s 赋值过的一个变量 x，则尝试在此直接使用... := s</td></tr>
<tr><td>opt_calculateFourOperatopnConst();</td><td colspan="2">把 x := #const [+-*/] #const 改写为 x = #const</td></tr>
<tr><td>opt_ignoreInvalidFourOperationConst();</td><td colspan="2">把无效的四则运算变为赋值语句，如
a := b + #0，a := #1 * x，a := b / #1等。</td></tr>
<tr><td>opt_duplexCheck();</td><td colspan="2">若由相邻连续的两行中间代码为 a := Exp 和 b := a，且 a 没再被作为右值使用过，则可把两行代码转换为一行 b := Exp</td></tr>
<tr><td rowspan="6">opt_clearJump();</td><td rowspan="6">对跳转语句进行优化，包括右侧的6种情况。</td><td>如果 goto 到的就是下一条代码，则 goto 无用。删除之</td></tr>
<tr><td>如果 ifgoto 到的就是下一条代码，则 ifgoto 无用。删除 ifgoto（暂保留目标 label）</td></tr>
<tr><td>如果目标 label 没有跳转来源，则是无效的 label，删除之</td></tr>
<tr><td>如果存在相邻的两个目标 label，则这两个 label 可以合并。这里选择保留第一个，删除第二个，同时更新全局中的相关 goto/ifgoto 语句</td></tr>
<tr><td>如果目标 label1的下一条代码就是一个 goto label2，则可以删除掉目标 label1并把所有的 label1的发起源改为跳转到 label2</td></tr>
<tr><td>把相邻的
IF(.....)GOTO label1
GOTO label2
LABEL label1:
改为
IF(not .....)GOTO label2
LABEL label1:</td></tr>
</table>