编译实习MiniC报告

罗昊 1700010686

hcc(Hao miniC Compiler) is a simple compiler for MiniC.
It contains 3 parts: eeyore, tigger and risc-v.

supported additional C rules besides MiniC

0. 支持空语句(;)
1. 支持逻辑表达式与算术表达式互相自动转换.
2. 支持无返回值调用函数.
3. 支持调用函数时使用表达式作为参数.
4. 支持在程序体内声明函数.
5. 支持C风格多行注释和C++风格单行注释.

error report

0. 报告语法错误和词法错误及其行号,并给出该处正确的token类型提示.
1. 检查标识符使用,如果使用了未定义的标识符会报错.
2. 检查标识符重复,对于重复定义的报错,函数名与变量名冲突的报错,如果在{}程序块内使用与程序块外同名的变量,则不会报错.
3. 检查函数参数表,对于重复声明但参数表不一致,或定义与声明参数表不一致,或调用时所用的参数表与声明的类型不一致时报错.
4. 检查+,-,*,/,%运算符对于数组类型变量的不合法操作给出错误提示,这些操作中除了 (int[])+(int), (int)+(int[]), (int[])-(int), 外涉及数组的运算都是不合法 的.
5. 检查对数组变量的赋值,无法将数值赋给数组变量.
6. 检查a[b]使用,如果a不是数组变量,会报错.

eeyore

0. 使用flex,bison和C++构建,将输入的MiniC代码转换为Eeyore三地址代码.
1. 使用STL的map模板制作符号表,使用链表串联内外层程序块的符号表.
2. 使用回填法构建eeyore中的标号及goto语句.

Tigger

活性分析

以语句为基本块进行自下而上的活性分析

寄存器分配

使用图着色算法进行寄存器分配.
利用活性分析的结果构造冲突图,兼用邻接表和邻接矩阵表示冲突图.
图染色有四种操作:
1.简化 对于度数小于k的传送无关的点,将其从图中删除并放入栈中.

2.合并 对于一条传送指令,对其关联的两个点在保守规则下合并.
3.冻结 冻结一条传送指令,将他当作非传送指令.
4.溢出 将图中度数最大的点放入栈中,并从图中删除..
操作按1.2.3.4的优先级进行,直到图中不再有点.
将栈中元素依次弹出,并为其分配颜色,如果无颜色可用,则将该顶点真实溢出.
若有真实溢出发生,则重写程序,为溢出的变量分配栈空间,加入load语句和store语句来读取和写回,并创建临时变量进行运算.

其他说明

Tigger不再创建符号表,因为其中所有变量名称的作用域都是全局,所以直接由词法分析器为其分配一个id(>27).
全局变量和栈数组不分配寄存器,直接存到内存中.

预着色节点的处理

Tigger有28个寄存器,给其确定id为0-27,并为其预着色
函数参数及返回值需要预着色. 进入函数体时创建变量保存参数,因此形参使用寄存器会视情况而定,不会长期占用a0-a7
每次进入函数体时保存被调用者保存的寄存器,return时恢复.
每次调用call时保存调用者保存的寄存器以及a0-a7中不用做参数的部分,call后恢复.
事实上由于寄存器分配时会合并部分传送指令,因此最后生成的程序正常情况下不会有大量预着色节点的move指令.\

合并预着色点相关传送节点的例子

MiniC代码

```
int v0;
v0 = getint();
int v1;
v1 = func(v0);
int v2;
v2 = putint(v1+1);
return v2;
```

tigger代码

```
call getint
call func
a0 = a0 + 1
call putint
return
```

riscv64代码

```
main:
        add     sp,sp,-16
        sd      ra,8(sp)
        call    getint
        call    func
        addiw   a0,a0,1
        call    putint
        ld      ra,8(sp)
        addi    sp,sp,16
        jr      ra
        .size   main, .-main
```

变量大量合并,只用了a0,没有一条move指令.