L4 中间代码生成 实验报告

181250015 陈彦泽

1 实现功能

在前三次代码的基础上，将C--源码翻译为中间代码

本次作业由于不需要完成**要求3.1**和**要求3.2**，实现较为简单，SDT在实验指导中基本已经给出，只需要自己补充包括数组、数组定义、变量初始化、 语法单元CompSt、语法单元StmtList等一小部分的SDT即可

所有实际上，本次作业的工作量都是在debug，主要是解决L3中遗留的一些历史问题

2 实现方法

1. 对先前生成的语法树进行解析，入口为`tranlate_InterCode(root,argv[2]);`
2. 生成中间代码，用链表`CodeList`进行记录
3. 输出到目标文件

3 一些实现细节

3.1 数组空间的分配

使用数组前需要分配空间，注意需要乘size，比如整数是4

`a[2]`要翻译为`DEC v1 8`而不是`DEC v1 2`

3.2 实现数组

难点在于规则，但由于假设只有一维数组，所以做起来并不困难

首先是Dec

```
CodeList translate_Dec(Node *Dec){
    if(Dec->child[2]!=NULL){
        ...// 略
    }else{
        Node* VarDec = Dec->child[0]; // VarDec: VarDec LB INT RB | ID
        if(strcmp(VarDec->child[0]->identifier,"VarDec")==0){
        // 数组初始化
            Operand var = op_from_var(VarDec->child[0]->child[0]->value); // 只可能是一维数组
            InterCode ic = new_InterCode(IR_DEC);
            ic->u.dec.x = var;
            ic->u.dec.size =  (int)strtol(VarDec->child[2]->value,NULL,10)*4;
            return new_CodeList(ic);
        }
    }
    return NULL;
}
```

然后是Exp

```
}else if(strcmp(Exp->child[1]->identifier,"LB")==0){ // Exp1 LB Exp2 RB
    // 根据假设，必是一维数组
    // 1. 获取Exp1的baseAddr
    // 2. 计算Exp2，存到t1
    // 3. 地址 = baseAddr + t1 * #4
}
```

3.3 历史遗留问题

由于之前的代码存在一些缺陷，恰巧没被测试用例发现，但在这次作业里暴露了（Hardtest-4），很大的工作量在debug上，这里记录一下

1. 语法树的子节点个数超过最大值（使用数组实现，有最大限制）

```
Reading symbols from ./parser...done.
(gdb) r ./tests/lab4/test1.cmm out1.ir
Starting program: /mnt/hgfs/share_ubuntu/compilers/Lab/parser ./tests/lab4/test1
.cmm out1.ir

Program received signal SIGSEGV, Segmentation fault.
0x0000555555558573 in addChild (parent=0x555555e0e4e0, son=0x555555e0dfa0)
    at src/Node.c:28
28          parent->child[parent->child_ptr++] = son;
(gdb) q
```

   解决：将`MAX_CHILD`调大即可

2. 自定义结构体的next指针问题

```
Program received signal SIGSEGV, Segmentation fault.
0x0000555555555243 in check (name=0x55555af70c20 "id_r") at src/hashtable.c:69
69              fprintf(stderr,"%s %s \n",node->name,name);
(gdb) p node
$1 = (HashNode *) 0x20
(gdb) p node->name
Cannot access memory at address 0x20
(gdb)
```

   这里发现`node=node->next`以后，node指向了一个非法地址而不是NULL，这是因为在node初始化的时候没有将`node->next`设置为NULL，添加后即可修复

   类似的错误出现在了FieldList上，一样要检查初始化时是否将tail指针指向NULL

3. 自定义结构体的成员变量初始化

```
    for(i = 0 ; i< root->child_ptr;++i){
        calTreeDeep(root->child[i],deep+1);
    }
```

   在这报了个错误

```
Program received signal SIGSEGV, Segmentation fault.
0x0000555555558ad2 in calTreeDeep (root=0x3000005bf, deep=400)
    at src/Node.c:134
134         root->deep = deep;
(gdb)
```

   原来是因为在初始化Node结构体时，没设置`child_ptr=0`，他变成了一个随机值。