编译原理实验四报告

周羽萱 211220074 1813156367@qq.com

一. 实现功能

所有的必做功能，具体如下：
在词法分析、语法分析、语义分析和中间代码生成程序的基础上，将 C 源代码翻译为 MIPS32 指令序列，并在 SPIM Simulator 上运行。

二. 程序编译

1. gcc -g -w main.c syntax.tab.c common.c semantic.c intermediate_code.c objectcode.c symbol_tab.c -lfl -ly -o parser
2. ./parser test.cmm

三. 代码实现

1. 数据结构
变量和寄存器结构如下：

```
struct VarStructure_{
    char* name; //变量名
    int reg; //变量存放的寄存器编号
    int offset; //变量在栈中的偏移量
    VarStructure next; //下一个变量
};

struct Register_{
    char* name; //寄存器名
    VarStructure var; //寄存器中存放的变量
    int used; //寄存器是否被使用
};
```

2. 变量空间分配
我在每个函数开始前先遍历每个语句，初始化变量，把变量加入变量表中。（在 generate_IR_func 中）

```
while(c!=NULL && c->code->kind != IC_FUNC){
    init_var(c->code,f);//初始化变量
    c = c->next;
}
```

```
//在函数开始时，初始化变量表
void init_var(InterCode ic,FILE* f){
    assert(ic!=NULL);
    switch(ic->kind){
        case IC_ASSIGN:
            op2var(ic->u.assign.left);
            op2var(ic->u.assign.right);
            break;
        case IC_PLUS:
```

首先把每个中间代码中的 operand 转换为变量并储存：如果 operand 是常量，不用处理；否则在变量表中根据这个 operand 的名字查找，如果找不到，就新建一个变量并加入变量表。

```
//把operand转换成变量，并把变量加入变量表
void op2var(Operand op){
    if(op->kind == EM_CONSTANT) return;
    VarStructure var = getVar(op);
    if(var == NULL){
        local_offset += 4;
        char* name = getopname(op);
        var = newVar(name,-1,-local_offset);
        addvar2list(var);
        printf("var->name = %s;var->offset = %d\n",var->name,var->offset);
    }
}
```

3. 寄存器分配

为了少出错，我使用了朴素的寄存器分配方式。每翻译一条中间代码之前我都把要用到的变量先加载到寄存器中，得到该代码的计算结果之后，判断结果是否有用或令原值改变，如果有用，则将结果写回内存。

4. 指针，取地址相关处理

- *x = y:

首先找到 x 这个变量（根据 x 这个 operand 的名字找到该变量，注意是 x,不是 *x），之后

```
fprintf(f, "\tlw $s0, %d($fp)\n", x->offset);//把x存储在寄存器$s0中
int y = getReg(right,f);
fprintf(f, "\tsw %s, 0($s0)\n", r[y].name);//将寄存器r[y]中的数据存储到*x处
```

- x = &y

  x = *y

对&y 和*y 的处理都在 getReg 中进行。不再赘述。

```
else if(op->kind == EM_GETADDR){
    //&x
    op->kind = EM_ARR;
    VarStructure var1 = getVar(op);
    op->kind = EM_GETADDR;
    VarStructure var2 = getVar(op);
    fprintf(f, "\taddi %s, $fp, %d\n", r[i].name, var1->offset);
    var2->reg = i;
    r[i].var = var2;
}
else if(op->kind == EM_ADDRESS){
    //*x
    op->kind = EM_TEMP;
    VarStructure var1 = getVar(op);//x
    op->kind = EM_ADDRESS;
    VarStructure var2 = getVar(op);//*x
    fprintf(f, "\tlw %s, %d($fp)\n", r[i].name, var1->offset);//把x存储在寄存器r[i]中
    fprintf(f, "\tlw %s, 0(%s)\n", r[i].name,r[i].name);//从*x处加载数据到寄存器r[i]中
    var2->reg = i;
    r[i].var = var2;
}
```

四． 总结与感悟

1. 本次实验中我认为难度最大的是指针、取地址等处理，比较难想。
2. 通过本次实验，我对函数调用时栈的变化也有了更深刻的认识，收获很多。

感谢助教哥哥批改！