实验三——中间代码生成

实验三内容是在词法分析、语法分析和语义分析程序的基础上，将 C--源代码翻译为中间代码。理论上中间代码在编译器的内部表示可以选用树形结构（抽象语法树）或者线形结构（三地址代码）等形式，为了方便检查你的程序，我们要求将中间代码输出成线性结构，从而可以使用我们提供的虚拟机小程序（附录 B）来测试中间代码的运行结果

实验特色：

我是选做内容 2，需要实现一维数组作为函数参数以及高维函数作为函数变量的情况。

```
struct Operand_{
        OperandTypeKind kind;
        union{
                int Var_Num;    // 变量定义的序号
                int Label_Num;  // 标签序号
                int Value;      // 操作数的值
                int Temp_Num;   // 临时变量序号（唯一性）
        }inform;
        Type type;      // 计算数组、结构体占用size
        int param;      // 标识函数参数
};
```

首先是对于操作数的结构体，为了实现选做功能我们需要 type 来计算数组的大小，同时需要 param 来标记该变量是否是作为函数参数被记录的。

```
char* GetVarString(Operand op)
{
        char*msg = malloc(10);
        if(op->kind == OP_CONSTANT){
                sprintf(msg, "#%d", op->inform.Value);
        }
        else if(op->kind == OP_TEMP || op->kind == OP_ADDR){
                sprintf(msg, "t%d", op->inform.Temp_Num);
        }
        else
                sprintf(msg, "v%d", op->inform.Var_Num);
        return msg;
}
```

然后为了便于打印，我将变量打印封装了一下以便于直接调用。

```
IRCode ircode;
if(op->kind == OP_ARRAY){//整个数组的赋值
        Operand addr1 = CreateTemp();
        //Operand addr2 = CreateTemp();
        addr1->kind = OP_ADDR;
        //addr2->kind = OP_ADDR;
        IRCode ir1 = CreateIRCode(IR_GET_ADDR);
        //IRCode ir2 = CreateIRCode(IR_GET_ADDR);
        ir1->inform.assign.left = addr1;
        ir1->inform.assign.right = op;
        //ir2->inform.assign.left = addr2;
        //ir2->inform.assign.right = Temp;
        code = CreateNewCodeList(ir1);
        Type array = Type_get(node->firstChild->firstChild);
        CodeList arraycopy = NULL;
        int Arraysize = Get_arraysize(array);
        for(int i = 0 ;i < Arraysize; i += 4){
                Operand array1 = CreateTemp();
                Operand array2 = CreateTemp();
                IRCode ir3 = CreateIRCode(IR_ADD);
                IRCode ir4 = CreateIRCode(IR_ADD);
                ir3->inform.binOp.result = array1;
                ir3->inform.binOp.op1 = addr1;
                ir3->inform.binOp.op2 = CreateConstant(i);
                ir4->inform.binOp.result = array2;
                 ir4->inform.binOp.op1 = Temp;
                 ir4->inform.binOp.op2 = CreateConstant(i);
                CodeList c1 = Merge_CodeList(CreateNewCodeList(ir3), CreateNewCodeList(ir4));
                //c1 = Merge_CodeList(Merge_CodeList(CreateNewCodeList(ir1), CreateNewCodeList(ir2)), c1);
                IRCode assign1 = CreateIRCode(IR_GET_ADDRVAL);
                Operand temp = CreateTemp();
                assign1->inform.assign.left = temp;
                assign1->inform.assign.right = array2;
                IRCode assign2 = CreateIRCode(IR_GET_VAL);
                 assign2->inform.assign.left = array1;
                 assign2->inform.assign.right = temp;
                arraycopy = Merge_CodeList(arraycopy, Merge_CodeList(c1, Merge_CodeList(CreateNewCodeList(assign1), CreateNewCodeList(assign2))));
        }
        code = Merge_CodeList(code, arraycopy);
}
```

对于 Exp ASSIGNOP Exp 的表达式中，若左右均为数组，则我们需要对数组中每个元素进行复制。

```
if(equal_string(Children->nodeName, "ID")){
        if(Children->bro == NULL){
                // 处理标识符
                if(LAB3_DEBUG) printf("Exp := ID\n");
                IR_VarList oplist = Findlist(Children->Valstr);
                Operand operand = oplist->Var;          //查找标识符i
                //value->kind = OP_TEMP;
                if(operand->kind == OP_ARRAY || operand->kind == OP_STRUCTURE){
                        if(oplist->ifParam){
                                IRCode ircode = CreateIRCode(IR_ASSIGN);
                                ircode->inform.assign.left = value;
                                ircode->inform.assign.right = operand;
                                return CreateNewCodeList(ircode);
                        }
                        IRCode ircode = CreateIRCode(IR_GET_ADDR);
                        ircode->inform.assign.left = value;
                        ircode->inform.assign.right = operand;
                        return CreateNewCodeList(ircode);
                }
```

然后对于作为函数参数的数组我们直接取 ASSIGN 而不是地址，因为传进来的就是地址。

实验感想：

相比实验二个人感觉实验三要复杂许多，因为实验三有许多小细节需要自己思考和实操才能发现端倪，而且也比较繁冗。