# 南京大学本科生实验报告

课程名称：**编译原理**

| 学院 | 计算机科学与技术 | 专业（方向） | 计算机科学与技术 |
| --- | --- | --- | --- |
| 学号 | **201220215** | 姓名 | **田馥雯** |
| Email | 1259067849@qq.com | 实验名称 | **实验四：目标代码生成** |

## 实验内容

**已实现全部必做及选做内容**





通过`make`生成可执行文件并通过`make test`指令输出到指定的.s文件中，并可以在SPIM Simulator中运行得到正确的结果

```
make clean
make
make test
```

## 核心代码

新建两个结构变量`Register`和`VarOffset_`，前者用于存储寄存器内容，后者用于存储变量地址偏移量

```
typedef struct Register
{
    bool used;          // 是否使用
    char *name;         // 寄存器名字
    VarOffset variable; // 所存变量的偏移值：用于save到原地址
} Register;
Register Registers[32];

typedef struct VarOffset_
{
    char name[32]; // 变量名字
    int reg;       // 寄存器号
    int offset;    // 偏移量
    VarOffset next;
} VarOffset_;
VarOffset varOffsetHead = NULL;
```

并且由此写出要用到的工具函数

```
/* 初始化所有寄存器 */
void initRegisters();
/* 寻找合适寄存器并填入操作内容 */
int findRegisters(Operand op, FILE *file);
/* 将寄存器值存回原地址 */
void SaveRegister(int index, FILE *file);
/* 获取操作数偏移量 */
VarOffset getVarOffset(Operand op);
/* 创建操作数偏移量 */
void createVarOffset(Operand op);
```

其中`findRegisters`根据操作数的不同将不同的内容存入寄存器，详见总结与感想中的翻译问题。

然后将19种中间代码一一翻译，形如

```
void generateObjectCode(FILE *file);
void generateLableObjectCode(InterCodes curInterCodes, FILE *file);
void generateFuncObjectCode(InterCodes curInterCodes, FILE *file);
void generateAssignObjectCode(InterCodes curInterCodes, FILE *file);
……
```

其中特殊处理的有`generateFuncObjectCode`（`func`和所有`param`的中间代码统一生成机器代码，并在这个函数声明时，将其中所有语句所用到的变量的偏移量算出来储存到链表中）、`generateArgObjectCode`（所有的`arg`和`call`的中间代码统一生成机器代码）

# 总结与感想

## 代码错误及解决方法

### *x = y、x = *y、x = &y、x = y、x = Constant 的翻译问题

这个将我在36分困扰了很久……主要是理解这几个指令吧

后四个在分配寄存器时处理加以区分

***y：将y搬入寄存器后，再将y对应地址的值搬入该寄存器**

```
fprintf(file, "\tlw %s, %d($fp)\n", Registers[i].name, varOffset->offset);
fprintf(file, "\tlw %s, 0(%s)\n", Registers[i].name, Registers[i].name);
```

**&y：对应偏移量地址中的值搬入寄存器**

```
fprintf(file, "\taddi %s, $fp, %d\n", Registers[i].name, varOffset->offset);
```

**y：直接将这个值载入寄存器**

```
fprintf(file, "\tlw %s, %d($fp)\n", Registers[i].name, varOffset->offset);
```

**Constant**

```
fprintf(file, "\tli %s, %d\n", Registers[i].name, op->value);
```

第一个整体进行翻译

```
fprintf(file, "lw %s, %d($fp)\n", Registers[leftIndex].name, leftVarOffset->offset);
fprintf(file, "sw %s, 0(%s)\n", Registers[rightIndex].name, Registers[leftIndex].name);
```

## lab2哈希表建立时的错误

感谢助教gg A-5的testcase，发现在lab2建立哈希表时当哈希值重复时插入节点直接将之前的结点覆盖而非生成一个链，将注释掉的内容换为下方语句就行。

```
    /*if (currentSymbolTableNode != NULL)
    {
        currentSymbolTableNode->sameHashSymbolTableNode = symbolTableNode;
    }*/
    symbolTableNode->sameHashSymbolTableNode = currentSymbolTableNode;
```

# 感悟

感谢助教gg和我亲爱的舍友，虽然我的代码生成方法非常笨拙，但还是过了……下学期再选这么多课我就是狗！！！！！！不过亲自干完这几个实验还蛮有成就感的，代码能力还是有提升的，等lab5写完就解放了~