实验四——目标代码生成

实验四内容是在词法分析、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（可以包含伪指令），并在 SPIM Simulator 上运行。

实验特色

```
typedef struct VarStructure_* VarStructure;
typedef struct Register_* Register;
typedef struct StackPointer_* StackPointer;

struct VarStructure_{
    Operand op; //变量名
    int Reg;    //变量存放的寄存器
    int VarOffset;  //变量在内存中的存储位置
    VarStructure next;  //变量链表
};


struct Register_{
    char * RegName;
    VarStructure var;
};
```

该实验我采取较为简单直接地办法，只使用变量表和寄存器两个结构来完成该实验。

```
Register Regs[32];
VarStructure AsmVarList = NULL;
FILE*file = NULL;
int Arg_Num = 0;
//int Param_Num = 0;
//int Last_clearReg = 8;
int CurOffset;
```

用 32 个寄存器，一个变量链表存储变量，一个 file 指针指向打印地文件，arg_num 记录函数参数个数以便于恢复栈帧，curoffset 用于记录将变量记录到栈上地位置。

```
if(Regs[index]->var->VarOffset == 1){
    CurOffset -= 4;
    Regs[index]->var->VarOffset = CurOffset;
}
fprintf(file, "\tsw %s, %d($fp)\n", Regs[index]->RegName, Regs[index]->var->VarOffset);
Clear_Reg(index);
break;
```

记录时我们首先查看该变量的是否已经在栈上存储过了，没有就赋予其偏移量，然后存储。最后将该变量占据的寄存器释放掉。

实验感想

本次实验总的来说并不难，但是需要充分了解栈的运作情况，并且需要熟悉中间代码的运行逻辑，但是由于我可能存在历史遗留问题导致最后 OJ 始终存在一个样例无法通过，故我认为该次实验还是非常困难的，希望老师能够放出样例来帮助检测一下 bug。