LAB 4 实验报告

裴明亮　151242033

一．实验环境及编，译方法

使用的是 Windows 系统（安装了虚拟机），在实验的根目录下创建了 Makefile 文件；进行测试时，执行Make test 即可执行相应测试样例的输出。

二．文件结构

在lab3 的基础上添加了新增了objcode.c 和objcode.h文件

相关函数：

```
void initreg();
void writeAsmHead(char *filename);
void inter2asm(char *filename);
void labelop(int labelid, FILE *fp);
void funcop(char *funcname, FILE *fp);
void li(int dstreg, int value, FILE *fp);
void move(int dstreg, int opreg, FILE *fp);
void addi(int dstreg, int op1reg, int imm, FILE *fp);
void add(int dstreg, int op1reg, int op2reg, FILE *fp);
void sub(int dstreg, int op1reg, int op2reg, FILE *fp);
void mul(int dstreg, int op1reg, int op2reg, FILE *fp);
void mdiv(int dstreg, int op1reg, int op2reg, FILE *fp);
void lw(int dstreg, int opreg, int addrbase, FILE *fp);
void sw(int dstreg, int opreg, int addrbase, FILE *fp);
void j(int labelid, FILE *fp);

void callop(int dstreg, char *funcname,int argcount, FILE *fp);
void retop(int dstreg, FILE *fp);
void beq(int op1reg, int op2reg, int labelid, FILE *fp);
void bne(int op1reg, int op2reg, int labelid, FILE *fp);
void bgt(int op1reg, int op2reg, int labelid, FILE *fp);
void blt(int op1reg, int op2reg, int labelid, FILE *fp);
void bge(int op1reg, int op2reg, int labelid, FILE *fp);
void ble(int op1reg, int op2reg, int labelid, FILE *fp);
void fwriteAllObjCode(char* filename);
```

三．数据结构

1.寄存器

```
struct reg
{
    int id;
    int inusing;
    operand *op;
};
extern REG r[REG_SIZE];
```

2. 栈指针

```
struct _stack
{
    char val[64];
    int ifarg;
    stack *next;
    stack *pre;
};
```

3. 总栈

```
struct stackstack
{
    stack *head;
    ss *next;
};
```

四．总结

基本实现必做要求，无选做部分，对于寄存器的分配，采用直接分配的方式。实验通过对中间表示生成对应的机器指令这一朴素方式进行的，没有进行过多的优化，虽然内容相对于实验三较少，却实现略为匆忙。总体的实验使我对编译原理这门课有了更为深入的理解，将理论诉诸于实验。