lab4

211275009 陈铭浩
211275009@smail.nju.edu.cn

实现功能

- 在lab1、lab2和lab3的基础上，将C--语言的源代码翻译为的中间代码进一步翻译为目标MIPS32汇编代码。

实现思路

1. 使用一个数组放置32个寄存器，并记录上次改变的寄存器的编号；记录变量存储的寄存器编号（如果不在寄存器中则记录为-1），并用链表将一个函数内的所有变量串联起来，并区分寄存器中的变量和内存中的变量。

```
typedef struct Register_* Register;
typedef struct Register_ {
    int isFree;
    char* name;
}Register_;

typedef struct Varible_* Varible;
typedef struct Varible_ {
    int regNo;
    Operand op;
    Varible next;
}Varible_;

typedef struct Registers_* Registers;
typedef struct Registers_ {
    Register regList[REG_NUM];
    int lastChangedNo;
}Registers_;
```

```
typedef struct VarList_* VarList;
typedef struct VarList_ {
    Varible head;
    Varible cur;
}VarList_;

typedef struct VarTable_* VarTable;
typedef struct VarTable_ {
    VarList varlistReg; // 寄存器中的变量表
    VarList varlistMem; // 内存中的变量表
    int inFunc;
    char* curFuncName;
}VarTable_;
```

2. 指令选择：由于lab3得到的中间代码是线形IR，因此直接逐行翻译中间代码为目标汇编代码，这里注意需要预先翻译好read和write函数
3. 寄存器分配：采用了比较朴素的寄存器分配算法，如果有空闲则直接使用，如果没有空闲则找一个最近没用过的立即数或临时变量释放掉，再存入对应寄存器中（这里只考虑分配T0~T9这些可随意使用的寄存器）
4. 栈的管理：参数传递采用寄存器与栈相结合的方式：如果参数少于 4 个，则使用 $a0 至 $a3 这四 个寄存器传递参数；如果参数多于 4 个，则前 4 个参数保存在 $a0 至 $a3 中，剩下的参数依次压到栈里。返回值直接放在$v0中。所有寄存器（T0~T9）都由调用者保存，全部压栈，调用结束后恢复

编译方式

```
在Lab4/Code文件夹下
$ make parser
然后用Code/parser替换Lab4下的parser（当然，在提交的版本中我已经替换过了）
在Lab4文件夹下执行
$ ./parser <输入文件路径> <输出文件路径> //输出到单独文件中
```