《编译器构造实验》lab4报告

20331041　徐锶达　xusd3@mail2.sysu.edu.cn

1 实现功能

1.1 完成内容

- **必做内容**：在假设1-8下，将lab3得到的中间代码转换为MIPS32汇编代码

1.2 指令选择

- 采用了线形IR
- 指令选择方式：逐条将中间代码对应（翻译）到目标代码

1.3 寄存器分配

- 采用了朴素寄存器分配算法
- 所有的变量或临时变量都放在内存里，即栈里，需要时再做取出/修改操作

1.4 栈实现与管理

- 通过数据结构`stack_node`来表示栈中元素，记录了相对栈顶的偏移的值（取负）

```
struct stack_node {
    int offset;
    int kind;
    int no;
    struct stack_node *next;
};
```

- 通过`stack_node`结构体指针的`stack_sp`，`stack_fp`来表示当前栈的栈顶和栈底

```
struct stack_node *stack_sp;
struct stack_node *stack_fp;
```

- 每一次进入新的函数时：
  - 把`stack_fp`之后的内容清空并且重新初始化`stack_sp`，`stack_fp`
  - 同时对这个函数做一次扫描，统计要开辟的占空间，一次性进行开辟
  - 并且确定好每个变量的`offset`，存入`stack_node`链表中
    - 对于函数的参数，在开辟局部变量之前确定了，因为对于一个函数，它的参数应该是调用它的对象实现压入栈的，因此只需要做好`offset`的关联工作就可以了
- 这个结构体和实际的MIPS32的栈的区别主要在于，MIPS32的栈的`fp`其实需要用一个栈来存储
- 但是因为扫描函数仅仅一遍不需要递归，因此这里实现结构体的时候不需要存储`stack_fp`，每一次都初始化到`stack_head->next`即可

2 Makefile编译

- 进入`Code`文件夹：运行：`make`
- 清理编译生成文件，运行：`make clean`

3 测试

1️⃣ **单个文件手动测试**，运行：`./parser test.cmm out1.s`

若`test.cmm`和`out1.s`与`parser`不在同一文件夹下，需自行添加路径

比如：`./parser ../Test/test.cmm ../result/out1.s`

2️⃣ **多个文件脚本测试**，运行：`./test.sh`

将多个测试文件（**只分析后缀为.cmm的文件**）统一放在`Test`文件夹下，运行**测试脚本**`test.sh`。

结果输出文件放在`result`文件夹下，文件名同测试文件名，后缀为.s。

注意默认情况是`parser`在`Code`文件夹下，`Code`文件夹与`Test`和`result`文件夹同级。不同则自行修改脚本中的`path1`和`path2`。

3️⃣ **QtSPIM测试目标代码**