实验三：中间代码生成

2019.11.26

1 小组信息

组号	学号	姓名	邮箱
1	161220022	陈哲霏	novelist.chan@gmail.com
	161220024	成威	weicheng.nju@qq.com

2 实验说明

2.1 实验要求

2.1.1 必做

- 将满足7个假设的C--语言源代码翻译成符合要求的中间代码指令序列

2.1.2 选做

- 可以出现结构体类型的变量（但不会有结构体变量之间间接赋值）
- 结构体类型的变量可以作为函数的参数

2.2 编译运行

- 在Lab/Code目录下执行make完成编译
- 执行./parser test1.cmm out1.ir， 可以将test1.cmm的翻译结果输出到out1.ir文件

3 实验内容

3.1 数据结构设计

- 在原有数据结构中添加了枚举类型OpType与ICType，用以指示操作类型和中间代码类型

```
typedef enum {
    VARIABLE, FUNCNAME,
    CONSTANT, ADDRESS,
    //...
} OpType;

typedef enum {
    LABEL, FUNCTION, ASSIGN,  ADD,
    //...
} ICType;
```

- 中间代码存储采用带头节点的双向循环链表

```
struct InterCode_ {
    ICType kind;
    union{
        struct {
            Operand op;
        } one;
        struct {
            Operand right, left;
        } assign;
        //...
    } u;
    InterCode prev;
    InterCode next;
};

InterCode head;
```

3.2 函数设计

- 添加了若干个构造函数对应各个不同类型的中间代码生成

```
InterCode new_oneOp_interCode(ICType kind, Operand op);
InterCode new_twoOp_interCode(ICType kind, Operand left, Operand right);
InterCode new_threeOp_interCode(ICType kind, Operand res, Operand op1,
Operand op2);
InterCode new_logic_goto_interCode(Operand left, Operand right, Operand dest,
char* relop);
InterCode new_dec_interCode(Operand op, int size);
```

3.3 代码优化

- 在Exp->ID | INT的语法条件下缩减了这部分的中间代码，免去了中间变量(t)的定义
- 在算术、逻辑运算中，对于所有操作数都是INT常量的情况，精简了计算逻辑
  - 例：i = 1 + 2; 此类源代码将被翻译为 i = #3 形式的中间代码
- 在数组和结构体取地址时，通过累计偏移量简化了中间代码
  - 例：a.b.c = 1; 此类源代码将取a的首地址和总的偏移量，将多次地址计算合并成一次

4 实验总结

- 本次实验的最大难点在于函数设计，由于我们将中间代码生成与语法树部分完全独立，需要在生成中间代码时也进行类似遍历语法树的操作，加深对语法树的理解
- 通过将类似代码块抽象成函数精简了代码
- 在完成编译器的同时收获了快乐