实验三 中间代码生成

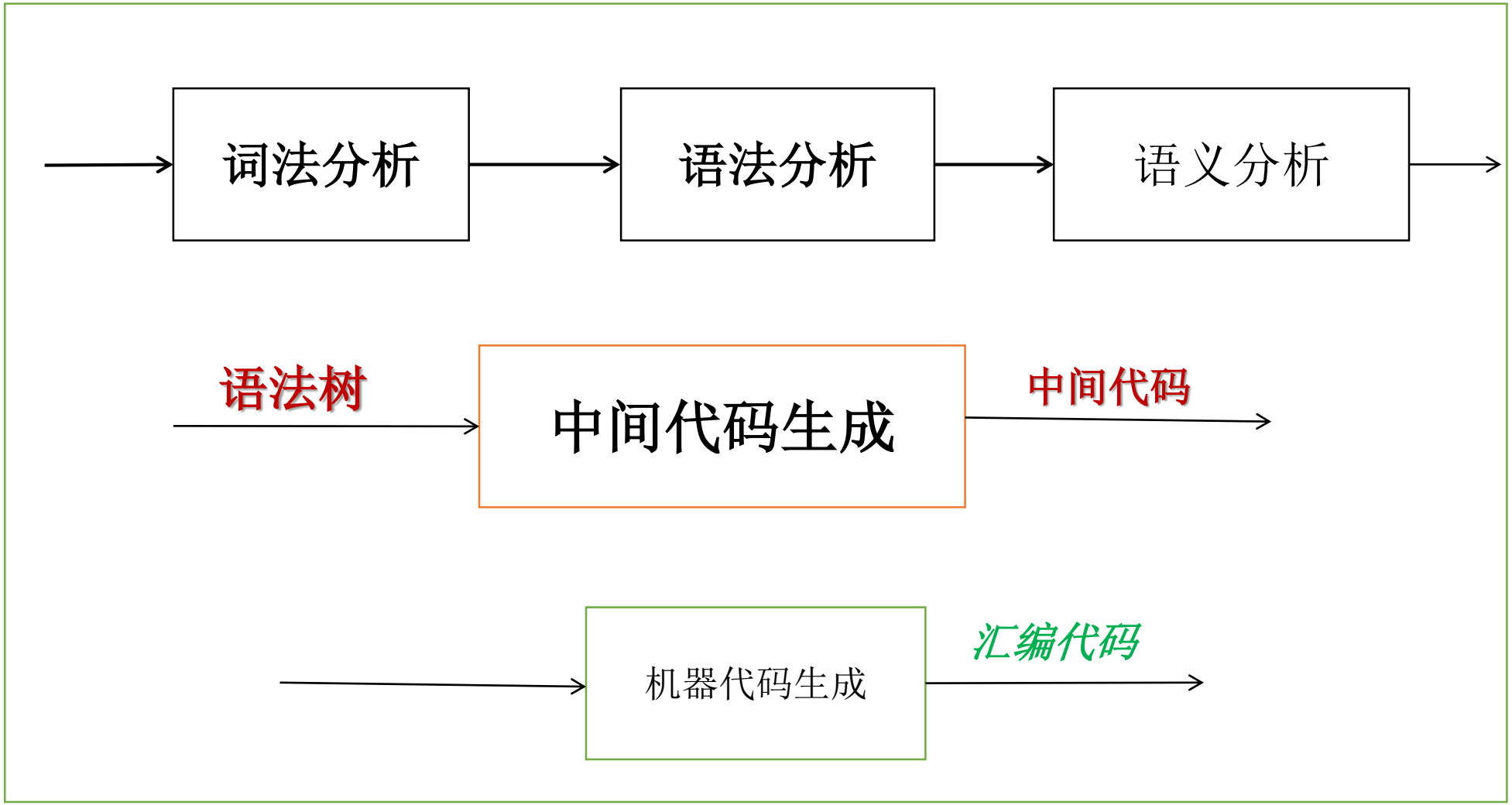
任课老师：戴新宇

助教：

尚迪(shangd@nlp.nju.edu.cn)

胡光能(hugn@nlp.nju.edu.cn)

编译器模块分解图



概要

- 实验3任务
- 实验3讲解
  - 数据结构
  - 简单Exp表达式翻译
  - 条件语句翻译
  - 循环语句翻译
- IR Simulator测试中间代码

实验3任务

- 中间代码生成
  - 根据生成的语法树产生中间代码
  - 将中间代码按照输出格式输出到文件中
- 基本方案
  - 依次递归遍历整个语法树
  - 将生成的中间代码使用链表拼接起来

一个简单的例子

```
int main()
{
   int n;
   n = read();
   if (n>0) write(1);
   else if (n<0) write(-1);
   else write(0);
   return 0;
}
```

```
1 FUNCTION main :
2 READ t1
3 v1 := t1
4 t2 := #0
5 IF v1 > t2 GOTO label1
6 GOTO label2
7 LABEL label1 :
8 t3 := #1
9 WRITE t3
10 GOTO label3
11 LABEL label2 :
12 t4 := #0
13 IF v1 < t4 GOTO label4
14 GOTO label5
15 LABEL label4 :
16 t5 := #1
17 t6 := #0 - t5
18 WRITE t6
19 GOTO label6
20 LABEL label5 :
21 t7 := #0
22 WRITE t7
23 LABEL label6 :
24 LABEL label3 :
25 t8 := #0
26 RETURN t8
```

表1. 中间代码的形式及操作规范。

语法	描述
LABEL x :	定义标号x。
FUNCTION f :	定义函数f。
x := y	赋值操作。
x := y + z	加法操作。
x := y - z	减法操作。
x := y * z	乘法操作。
x := y / z	除法操作。
x := &y	取y的地址赋给x。
x := *y	取以y值为地址的内存单元的内容赋给x。
*x := y	取y值赋给以x值为地址的内存单元。
GOTO x	无条件跳转至标号x。
IF x [relop] y GOTO z	如果x与y满足[relop]关系则跳转至标号z。
RETURN x	退出当前函数并返回x值。
DEC x [size]	内存空间申请，大小为4的倍数。
ARG x	传实参x。
x := CALL f	调用函数，并将其返回值赋给x。
PARAM x	函数参数声明。
READ x	从控制台读取x的值。
WRITE x	向控制台打印x的值。

中间代码数据结构

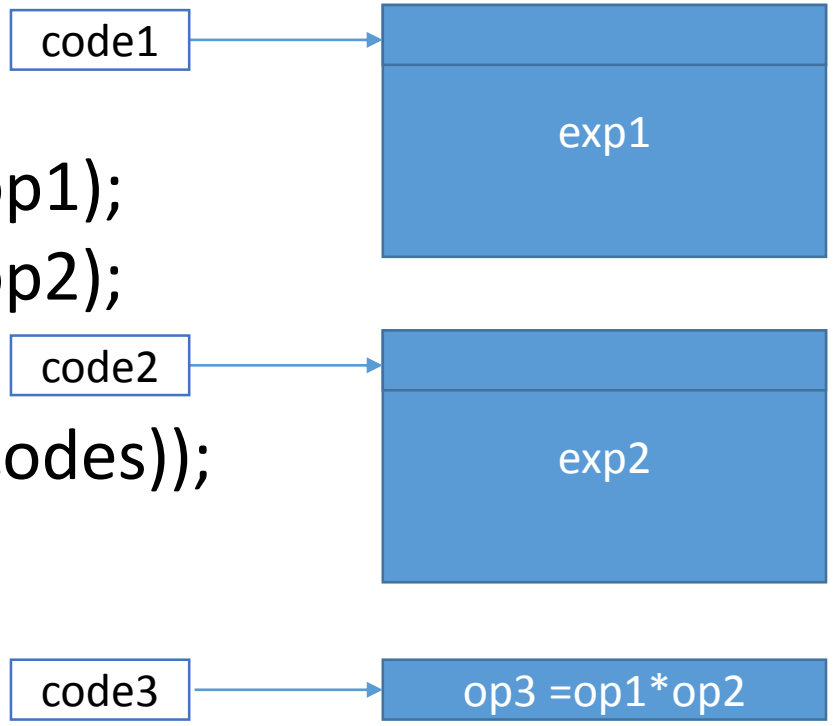
- 中间代码存储
  - 使用链表存储中间代码
  - 每行中间代码存于一个中间代码结构体中
- struct InterCode需包含内容
  - enum kind (ASSIGN, ADD, SUB, MUL, DIV_, ETURN_, LABEL_CODE, LABEL_TRUE, LABEL_GOTO,CALLFUNC, FUNCTION, ARG, PARAM, REFASSIGN, WRITE, READ)
  - union u(assign,add,sub,mul,call,func,arg…)
  - …

遍历语法树

- translate(root)
  - 开始遍历语法树
- 举例：当遇到Exp节点(treenode.type = exp)时
  - translate_exp(treenode)
  - Exp -> Exp1 STAR Exp2

简单Exp表达式翻译

- **Exp → Exp1 STAR Exp2**

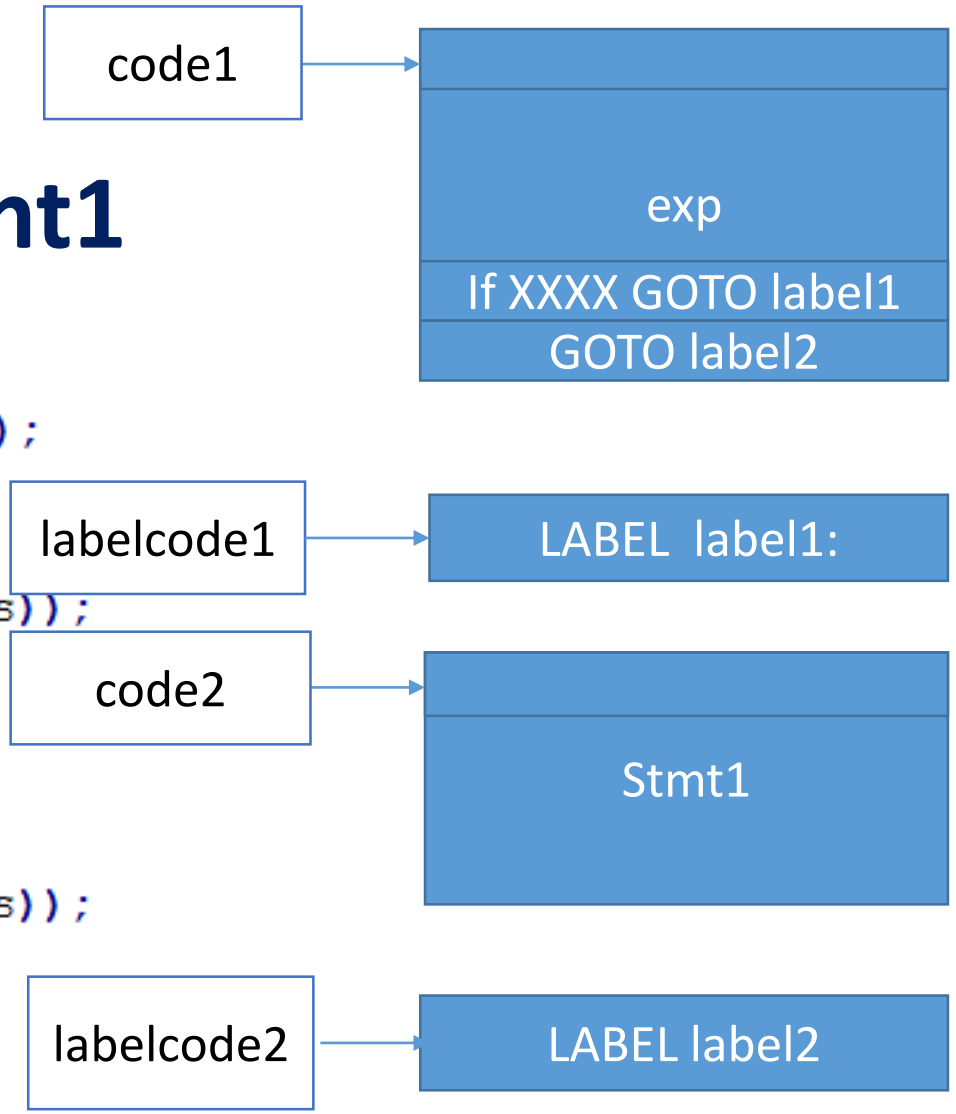



```
struct InterCodes* code1 = translate_Exp(Exp1,op1);
struct InterCodes* code2 = translate_Exp(Exp2,op2);
struct InterCodes* code3 =
   (struct InterCodes*)malloc(sizeof(struct InterCodes));
code3->code.kind = MUL;
code3->code.u.binop.op1 = op1;
code3->code.u.binop.op2 = op2;
…//初始化code3的其他信息
return bindCode(bindCode(code1,code2),code3);
```

条件语句翻译

- **Stmt → IF LP Exp RP Stmt1**

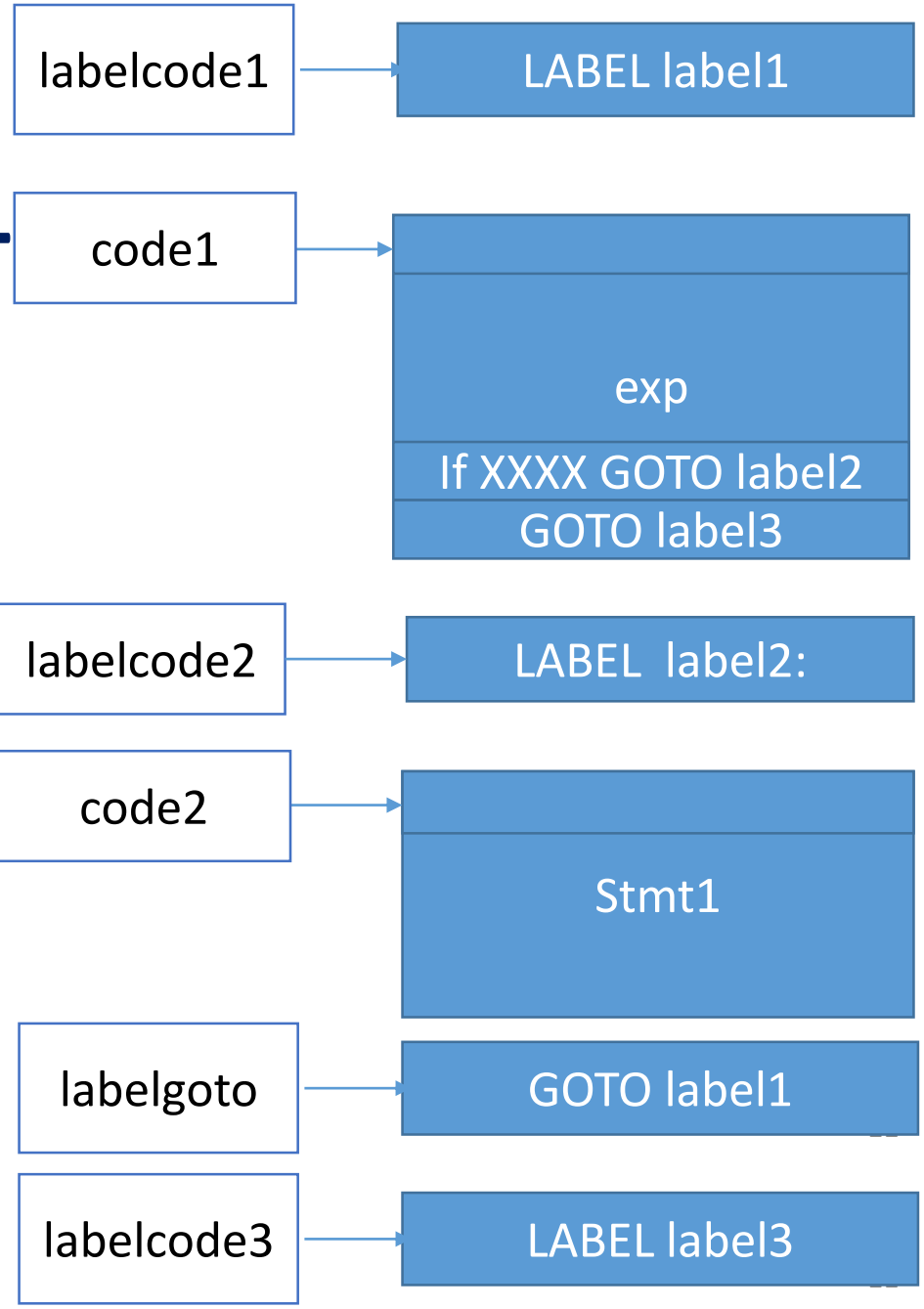
```
Operand label1 = new_label();
Operand label2 = new_label();
InterCodes* code1 = translate_Cond(Exp, label1, label2);
InterCodes* code2 = translate_Stmt(Stmt1, sTable);
InterCodes* labelcode1 =
    (struct InterCodes*)malloc(sizeof(struct InterCodes));
labelcode1->code.kind = LABEL_CODE;//初始化labelcode1
labelcode1->code.u.label_code.label = label1;
...
InterCodes* labelcode2 =
    (struct InterCodes*)malloc(sizeof(struct InterCodes));
labelcode2->code.kind = LABEL_CODE;//初始化labelcode2
labelcode2->code.u.label_code.label = label2;
...
return bindCode(
    bindCode(code1,labelcode1),
    bindCode(code2,labelcode2)
    );
```



循环语句翻译

- **Stmt → WHILE LP Exp RP Stmt1**

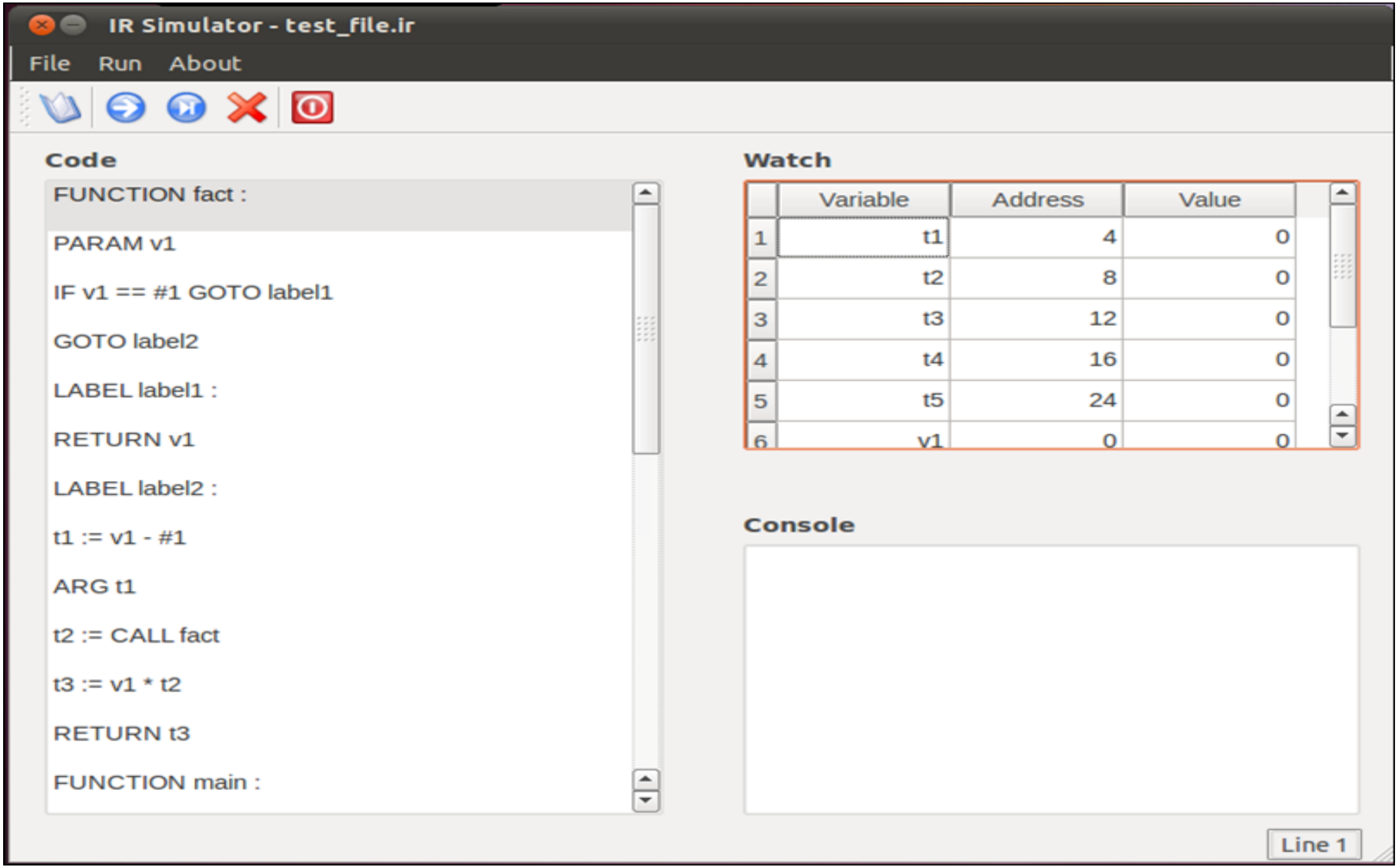
```
Operand label1 = new_label();
Operand label2 = new_label();
Operand label3 = new_label();
InterCodes* code1 = translate_Cond(Exp, label2, label3);
InterCodes* code2 = translate_Stmt(Stmt1);
InterCodes* labelcode1 = new_InterCode(LABEL_CODE,label1);
InterCodes* labelcode2 = new_InterCode(LABEL_CODE,label2);
InterCodes* labelgoto = new_InterCode(LABEL_GOTO,label1);
InterCodes* labelcode3 = new_InterCode(LABEL_CODE,label3);
return bindCode(
    bindCode(
        bindCode(labelcode1,code1),
        bindCode(labelcode2,code2)
            ),
    bindCode(labelgoto,labelcode3)
    );
```



函数read和write

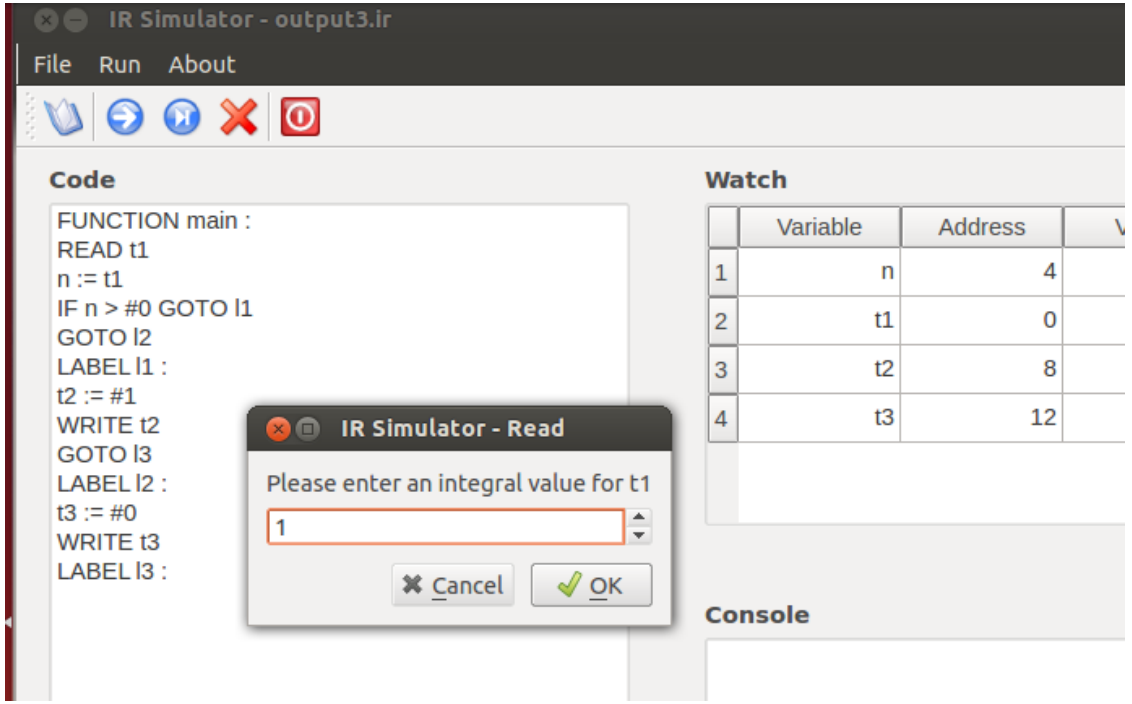
- 输入输出语句READ和WRITE用于和控制台进行交互

- READ语句可以从控制台读入一个整型变量
- WRITE语句可将一个整型变量的值写到控制台上。
- 程序初始化时，向符号表中预先添加read和write两个函数条目


IR Simulator - test_file.ir
File
Run
About
Code
FUNCTION fact :
PARAM v1
IF v1 == #1 GOTO label1
GOTO label2
LABEL label1 :
RETURN v1
LABEL label2 :
t1 := v1 - #1
ARG t1
t2 := CALL fact
t3 := v1 * t2
RETURN t3
FUNCTION main :
Watch
Variable
Address
Value
1 t1 4 0
2 t2 8 0
3 t3 12 0
4 t4 16 0
5 t5 24 0
6 v1 0 0
Console
Line 1

IR Simulator中间代码解释器

- 将三个python文件放在当前目录
- 使用命令 python irsim.py
- 载入*.ir三地址中间代码文件，即可执行



注意 .ir文件的格式要求即可

提交说明

- 地址：**ftp://114.212.190.181: *22***
- 用户名和密码：**upload**
- **输出要求：　./parser test1.c test1output.ir**
  - **把生成的中间代码逐行写入test1output.ir文件中**
- 内容：
  - 源程序(必须能通过编译)
  - 可执行程序(**必须命名为 parser**)
  - 报告PDF(*完成的功能，编译步骤，实现方法，结点的数据结构表示；不超过3页*)
- **Deadline ：　6月1日**

Thank you.
Any questions?