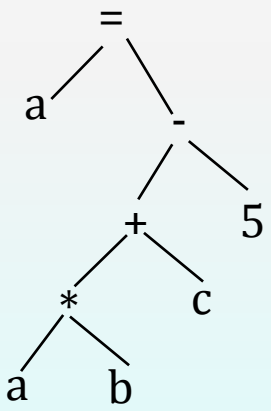
第7章 中间代码生成

- 7.1 中间代码的形式
- 7.2 表达式的四元式代码生成
- 7.3 原子语句的四元式代码生成
- 7.4 结构语句的四元式代码生成
- 7.5 声明的四元式代码生成

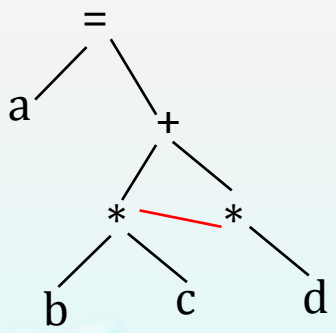
7.1 中间代码的形式

- 逆波兰式(后缀式)
  - (a*b+c)-5 的后缀式：ab*c+5-
- 三元式:编号表示计算的中间结果
  - 赋值语句a=(a*b+c)-5的三元式表示
  - (1) (*, a,b)
  - (2) (+,(1), c)
  - (3) (-, (2), 5)
  - (4) (=, (3), a)
- 语法树和DAG
- 四元式
  - (*,a,b,t1)
  - (+,t1,c,t2)
  - (-,t2, 5, t3)
  - (=, t3, -, a)

树和图的边要有箭头



a=(a*b+c)-5的语法树



a=b*c+b*c*d的DAG

7.1 中间代码的形式

- 一种通用的四元式形式:
- 表达式的运算符:
  - 算术运算: ADDI, SUBI,MULTI,DIVI, ADDF, SUBF, MULTF, DIVF, MOD
  - 逻辑运算: AND, OR, NOT
  - 关系运算: GT, GE, LT, LE, EQ, NE
  - 地址运算: AADD
- I/O语句:
  - 输入: (READI, -, -, id) 或 (READF, -, -, id)
  - 输出: (WRITE, -, -, id)
- 类型转换语句: (FLOAT, id1,-,id2) //将整型数id1转换成实型数赋值给id2
- 赋值语句: (ASSIGN, id1, n, id2) //把id1的值赋值给id2(长度为n)
- 标号语句: (LABEL, -, -, label) //定义label为标号,并且定位于当前位置
- 跳转语句:
  - (JUMP, -, -, label) //无条件跳转
  - (JUMP0, id, -, label) //如果id为假，则转向label指示的代码
  - (JUMP1, id, -, label) //如果id为真，则转向label指示的代码

7.1 中间代码的形式

- 分支语句：
  - (THEN, id, -, label)
  - (ELSE, -, -, -)
  - (ENDIF, -, -, -)
- 循环语句：
  - (WHILE, -, -, -)
  - (DO, id, -, -)
  - (ENDWH, -, -, -)
- 函数相关的语句：
  - 声明：(ENTRY, label,size, level) //函数入口
  - (RETURN, -, -,t) / (RETURN, -, -,-) //从函数调用返回
  - (ENDFUNC, -,-,-) //函数出口
  - 调用：(VALACT, id, offset, size) //传递实参变量id的值给形参
  - (VARACT, id, offset, size) //传递实参变量id的地址给形参
  - (FUNCACT, id, offset, size) //传递实参函数id的信息给形参
  - (CALL, Lf, true/false, t) //调用函数，并用t记录返回值

7.2 表达式的四元式代码生成

- 表达式的中间代码生成就是依据原表达式的语义产生出正确计算表达式值的四元式序列(必要时要进行类型转换).
- 例:有表达式a*(3.5+i*b)，其中a、b为实型变量，i为整型变量，则生成的四元式中间代码如下:
  - (FLOAT, i,-,t1)
  - (MULTF, t1,b,t2)
  - (ADDF, 3.5, t2, t3)
  - (MULTF, a, t3, t4)

7.2 表达式的四元式代码生成

- 表达式的运算分量可以很复杂:
  - 多维数组下标变量：A[i][j][k]
  - 结构体域名变量：st.address.city
  - 函数调用：f(x,y)
  - 指针引用：*(p+1)
  - ……
- 在表达式的运算中，实际参与运算的是这些复杂变量的地址中存放的数据值,要想取到这些值,首先要计算复杂变量的地址，因此对于复杂变量要给出其地址计算的四元式.

7.2 表达式的四元式代码生成

- 下标变量V[E]的四元式:
  - V是数组名或代表数组的复杂变量
  - E是下标表达式

V.code
E.code
(*, E.arg, V.ElemType.size, t1)
(AADD, V.arg, t1, t2)

```
int   A[10];
float   B[5][5];
A[5]+B[2][3]
```

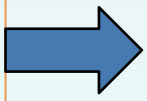
```
(MULTI, 5, 1,t1)
(AADD, A,t1,t2)
(MULTI,2,10,t3)
(AADD, B, t3)
(MULTI,3,2,t4)
(AADD,t3,t4,t5)
(FLOAT,t2,-,t6)
(ADDF, t6,t5,t7)
```

7.2 表达式的四元式代码生成

- 域名变量V.id的四元式:
  - V是结构体变量或代表代表结构体的复杂变量
  - id是结构体的某个域名

V.code
(AADD, V.arg, id.offset, t1)

```
struct student{
                    char name[10];
                    int age;}  st[100];
st[5].age  +10
```



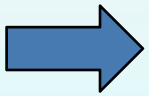
```
(MULTI, 5, 11,t1)
(AADD, st,t1,t2)
(AADD,t2,10,t3)
(ADDI, t3, 10,t4)
```

7.2 表达式的四元式代码生成

- 指针变量*V的四元式:
  - V是指针类型变量
  - *V表示取出指针类型变量V所指向的地址内存放的值
  - 因此,*V的四元式是将V的地址取出来.

V.code
(ASSIG, V.arg, 1, t1)

```
int*  p;
……
*p+10
```



```
(ASSIG, p,1,t1)
(ADDI, t1, 10,t2)
```

7.2 表达式的四元式代码生成

- 例7.1：有如下的类型声明，为表达式 **class[5].age + *ptr.age**生成四元式形式中间代码.

```
typedef struct {
        char   name[30];
        int    age;
       float   height;
       }person;

int x[10];
person  class[10];
person  *ptr;
```

(SUBI, 5, 0, t5)
(MULTI, t5, 33, t6)
(AADD, class, t6, t4)
(AADD , t4, 30, t1)
(Assig, ptr, _, t7)
(AADD , t7, 30, t2)
(ADDI, t1, t2, t3)

t1,t2,t4和t7的mode是indir;

7.3 原子语句的四元式代码生成

- 赋值语句:V = E
  - V.code
  - E.code
  - (ASSIGN, E.arg, n, V.arg)　　//n表示E结果值所占空间的大小
- 输入语句Read(V):
  - V.code
  - (ReadI, -, -, V.arg) or (ReadF, -, -, V.arg)
- 输出语句Write(E):
  - E.code
  - (Write, -, n, E.arg)　　//n表示E结果值所占空间的大小

7.3 原子语句的四元式代码生成

- 标号语句label: SL
  - (Label, -,-, l)
  - SL.code
- 跳转语句Goto label:
  - (JUMP, -,-, l)　　//l代表label对应的内部标号
  - (ReadI, -, -, V.arg) or (ReadF, -, -, V.arg)
- 函数返回语句Return E:
  - E.code
  - (Return, -, -, E.arg)
  - 或　(Return, -, -, -)

7.3 原子语句的四元式代码生成

- 函数调用语句f(E1,E2,...En):
  - Ei.code
  - ……
  - En.code
  - (VALACT/VARACT/FUNCACT, E1.arg, offset1,size)
  - ……
  - (VALACT/VARACT/FUNCACT, En.arg, offsetn,size)
  - (CALL, Lf, true/false, t)

7.3 原子语句的四元式代码生成

- 例7.2: 写出表达式x + f(H(10), g(Y))对应的四元式代码.其中:x是一般整型变量, Y是变参整型变量，f的返回值是整型，f和g是实在函数名,H是形参函数名，f的两个形参都是值参，g的形参为变参，H的形参为值参.
- 四元式代码如下:

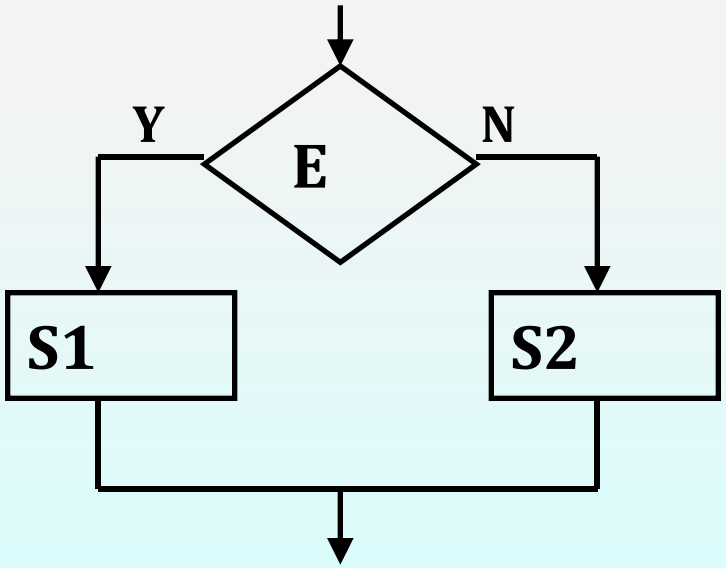
1.(VALACT, 10, 0, 1)
2.(call, H, false, t1)
3.(VARACT, Y, 0, 1)
4.(call, g, true, t2)
5.(VALACT, t1, 0, 1)
6.(VALACT, t2, off, 1)
7.(call, f, true, t3)
8.(ADDI, x, t3, t4)

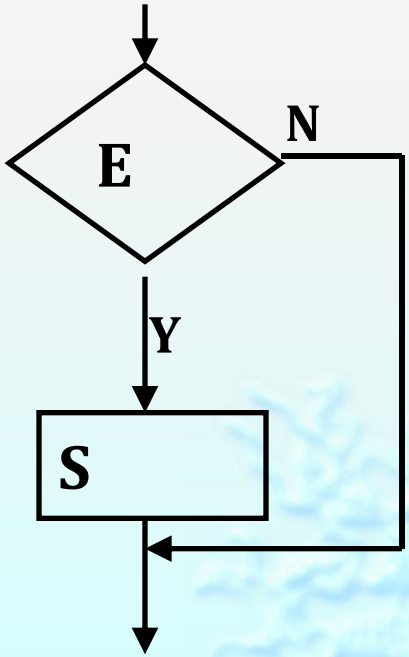
7.4 结构语句的四元式代码生成

- 7.4.1 分支语句的四元式代码生成
  - If语句
  - Switch语句*
- 7.4.2 循环语句的四元式代码生成
  - While循环
  - Repeat循环*
  - For循环*

7.4.1 分支语句的四元式代码生成

- If 语句有下面两种形式:
- If <E> then <S1> else <S2> 或
- If <E> then <S1>
- 其执行逻辑如下:





If语句的四元式(方案1)

If <E> then <S1> else <S2>

E.code
(THEN, E.arg, _, _)
S1.code
(ELSE, _, _, _)
S2.code
(ENDIF, _, _, _)

If <E> then <S1>

E.code
(THEN, E.arg, _, _)
S1.code
(ENDIF, _, _, _)

If语句的四元式(方案2)

If <E> then <S1> else <S2>

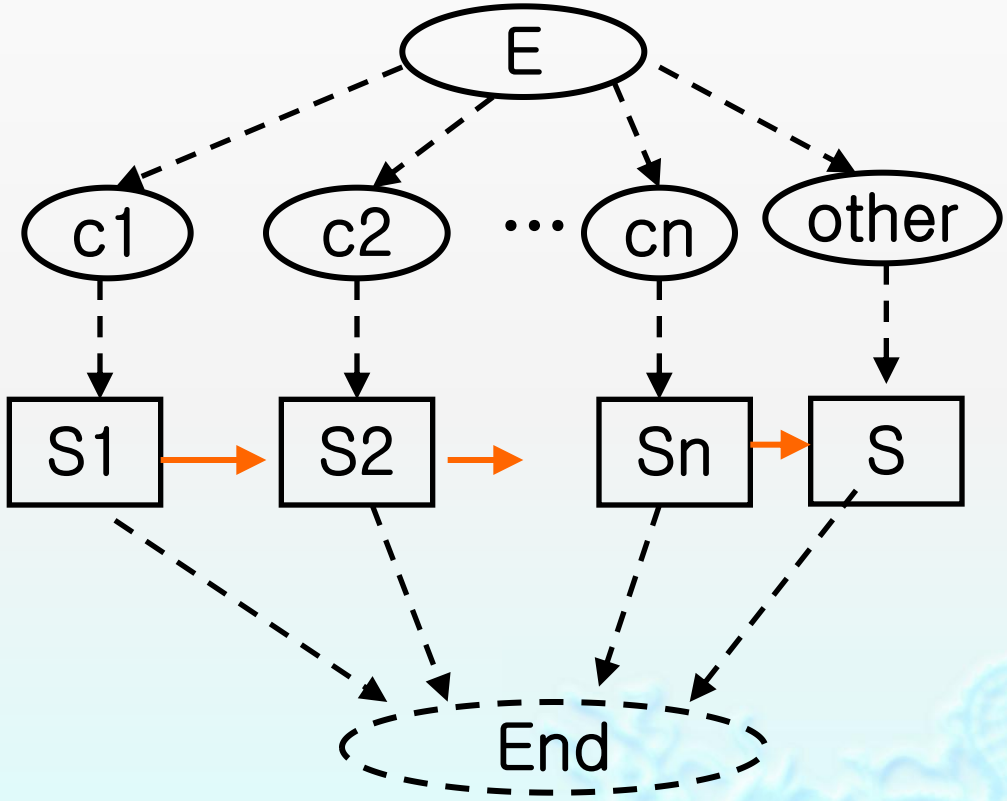
E.code
(JUMP0, E.arg, _, ElseL)
S1.code
(JUMP, _, _, ExitL)
(LABEL, _, _, ElseL)
S2.code
(LABEL, _, _, ExitL)

If <E> then <S1>

E.code
(JUMP0, E.arg, _, exitL)
S.code
(LABEL, _, _, exitL)

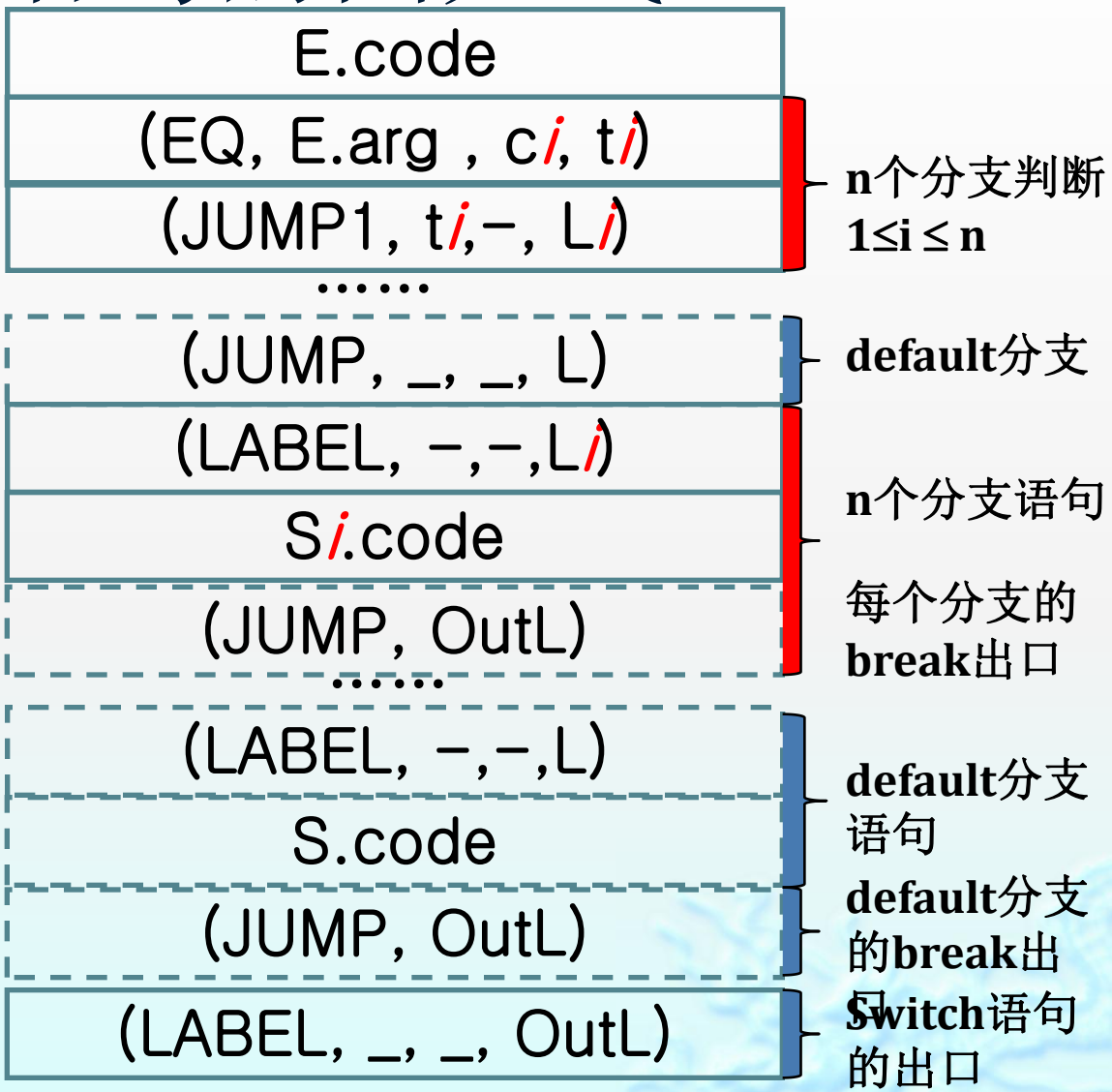
Switch语句的四元式*

```
switch E of
 {
  case c1: S1; [break;]
  ...
  case cn: Sn; [break;]
  [default: S;   [break;]]
 }
```



Switch语句的四元式

```
switch E of
{
 case c1: S1; [break;]
 ...
 case cn: Sn; [break;]
 [default: S;   [break;]]
}
```



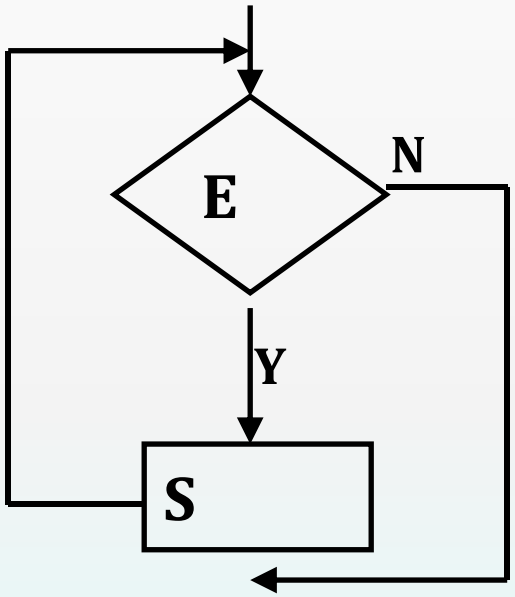
分支语句四元式代码的例子

1. **if (a>b) then {mark = true; c=a;}**
   else {mark = false; c=b;} endif

2. **switch (a+b) of**
   {
   case 1: x=1; break;
   case 3: x=3;
   case 5: x=5; break;
   default: y=4;
   }

7.4.2 循环语句的四元式代码生成

- While循环: while (E) S



方案1:

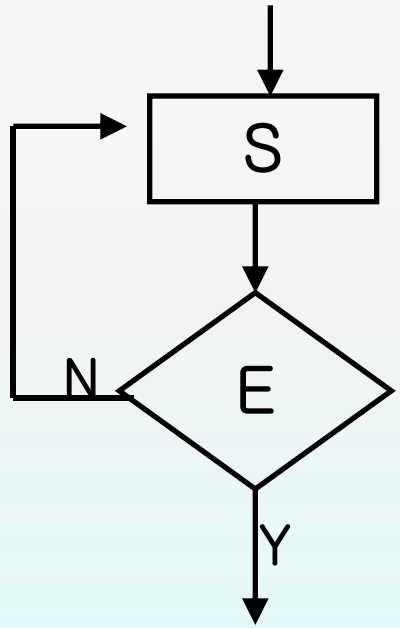
(WHILE, _, _, _)
E.code
(DO, E.arg, _, _)
S.code
(ENDWHILE, _, _,_)

方案2:

(LABEL, _, _, StartL)
E.code
(JUMP0, E.arg, _, ExitL)
S.code
(JUMP, _, _, StartL)
(LABEL, _, _, ExitL)

7.4.2 循环语句的四元式代码生成

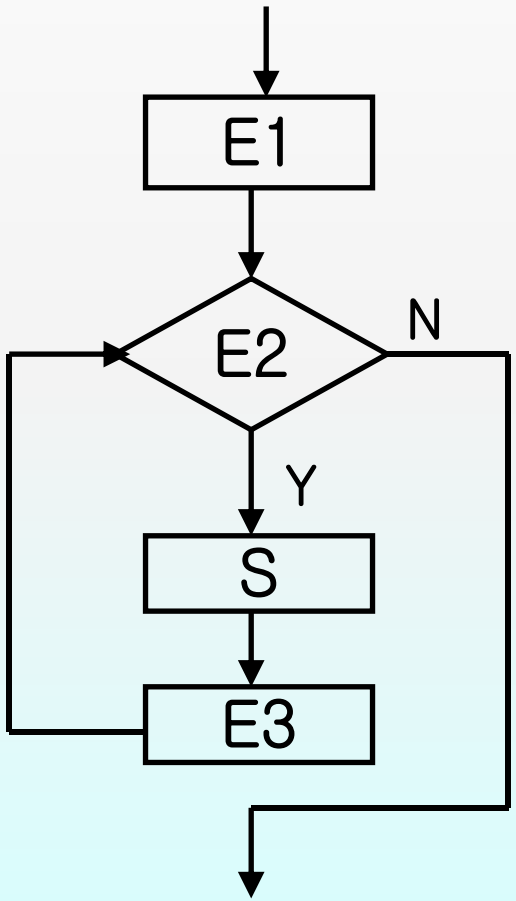
- Repeat循环: Repeat S until E



(LABEL, _, _, StartL)
S.code
E.code
(JUMP0, E.arg, _, StartL)

7.4.2 循环语句的四元式代码生成

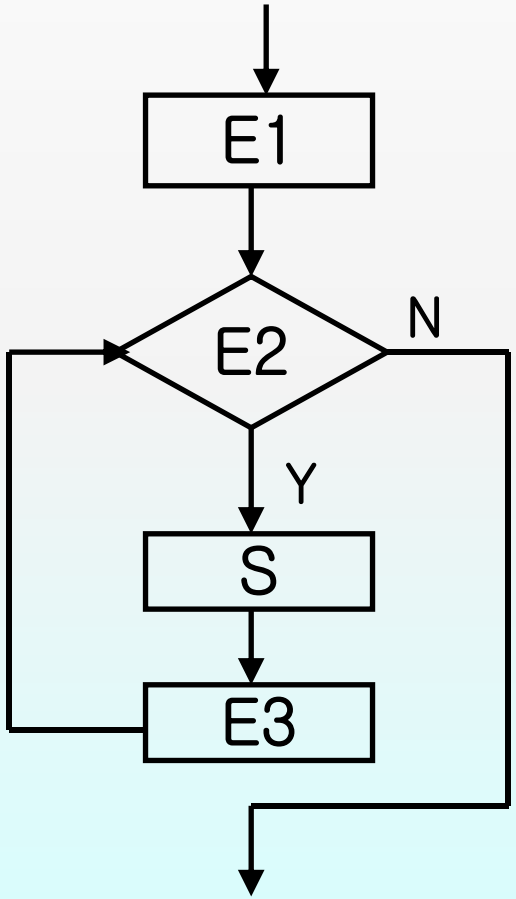
- For循环: for (E1;E2;E3) S



E1.code
(LABEL, _, _, StartL)
E2.code
(JUMP0, E2.arg, _, ExitL)
(JUMP, _, _, SL)
(LABEL, _, _, EL)
E3.code
(JUMP, _, _, StartL)
(LABEL, _, _, SL)
S.code
(JUMP, _, _, EL)
(LABEL, _, _, ExitL)

7.4.2 循环语句的四元式代码生成

- For循环: for (E1;E2;E3) S



E1.code
(LABEL, _, _, StartL)
E2.code
(JUMP0, E2.arg, _, ExitL)
S.code
E3.code
(JUMP, _, _, StartL)
(LABEL, _, _, ExitL)

7.5 声明的四元式代码生成

- 常量声明
- 类型声明
- 变量声明

不生成代码

- 函数声明 ---顺序生成下列代码
  - 函数入口(ENTRY, Lf, size,level)
  - 函数体的代码
  - 函数出口(ENDFUNC, -,-,-)

作 业

- 写出下面程序的四元式代码

```
#define  m 100
typedef  struct { char name[10]; int age; int number;} student;
student  class[100];
int  FindAge(int id)
{  int i = 0;
   while (i<100) do
    { if (class[i].number == id)  then return (class[i].age); endif;
      i = i+1;
    } endwhile
   return -1;
}
void main ()
{   FindAge(5);  }
```