三地址码指令（四元式）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| read x | rd | x |  |  |
| t1 = x > 0 | gt | x | 0 | t1 |
| if (t1 == false) go to L1 | if\_f | t1 | L1 |  |
| fact = 1 | asn | 1 | fact |  |
| label L2 | lab | L2 |  |  |
| t2 = fact \* x | mul | fact | x | t2 |
| fact = t2 | asn | t2 | fact |  |
| t3 = x - 1 | sub | x | 1 | t3 |
| x = t3 | asn | t3 | x |  |
| t4 = x == 0 | eq | x | 0 | t4 |
| if false t4 goto L2 | if\_f | t4 | L2 |  |
| write fact | wri | fact |  |  |
| label L1 | lab | L1 |  |  |
| halt | halt |  |  |  |

三地址码指令（三元式）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| read x | rd | x |  | (0) |
| t1 = x > 0 | gt | x | 0 | (1) |
| if (t1 == false) go to L1 | if\_f | (1) | (11) | (2) |
| fact = 1 | asn | 1 | fact | (3) |
| t2 = fact \* x | mul | fact | x | (4) |
| fact = t2 | asn | (4) | fact | (5) |
| t3 = x - 1 | sub | x | 1 | (6) |
| x = t3 | asn | (6) | x | (7) |
| t4 = x == 0 | eq | x | 0 | (8) |
| if false t4 goto L2 | if\_f | (8) | (4) | (9) |
| write fact | wri | fact |  | (10) |
| halt | halt |  |  | (11) |

数组引用的三地址码：a[i + 1]=a[j \* 2] + 3

t1 = j \* 2

t2 = t1 \* elem\_size(a)

t3 = &a + t2

t4 = \* t3

t5 = t4 + 3

t6 = i + 1

t7 = t6 \* elem\_size(a)

t8 = &a + t7

\*t8 = t5

条件语句的三地址码：if (E) S1 else S2

<Code E>

if false t1 goto L1

<Code S1>

goto L2

label L1

<Code S2>

label L2

循环语句的三地址码：while (E) S

label L1

<Code E>

if\_false t1 goto L2

<Code S>

goto L1

label L2

函数的三地址码：int f(int x, int y) { return x + y + 1; }

entry f

t1 = x + y

t2 = t1 + 1

return t2

参数计算和压栈的三地址码：f(2+3, 4)

begin\_args

t1 = 2 + 3

arg t1

arg 4

call f

**P-码指令**

|  |  |
| --- | --- |
| loc 2 | 加载常数 |
| lod a | 加载值 |
| lda | 加载地址 |
| grt | 大于 |
| adi | + |
| sbi | - |
| mpi | \* |
| sto | 将栈顶的值赋值到栈顶下面一个元素(地址)的值中，并全部Pop |
| stn | 与sto类似，但是会将表达式的值留在栈中 |
| rdi | 读入整数 |
| ind i | 间接装载， ind i => \*(a+i)，执行前栈中元素有a |
| ixa s | 索引地址 ixa s => a + s \* i，执行前栈中有i和a （i在栈顶） |
| fjp | false then jump |
| ujp | 无条件跳转 |
| neq | 不相等判定 |
| equ | 相等判定 |

赋值语句的P码：x : = y + 1

lda x

lod y

ldc 1

ad i

sto

赋值运算表达式的P码：(x = x + 3) + 4

lda x

lod x

ldc 3

adi

stn

ldc 4

adi

装载地址的P码：\*(x + 10) = 2

lda x

ldc 10

ixa 1

ldc 2

sto

数组引用的P码：a[i + 1]=a[j \* 2] + 3

lda a

lod i

ldc 1

adi

ixa elem\_size(a)

lda a

lod j

ldc 2

mpi

ixa elem\_size(a)

ind 0

ldc 3

adi

sto

条件语句的P码：if (E) S1 else S2

<Code E>

fjp L1

<Code S1>

ujp L2

lab L1

<Code S2>

lab L2

循环语句的P码：while (E) S

label L1

<Code E>

fjp L2

<Code S>

ujp L1

lab L2

函数的P码：int f(int x, int y) { return x + y + 1; }

ent f

lod x

lod y

adi

ldc 1

adi

ret

参数计算和压栈的P码：f(2+3, 4)

mst ;mark stack

ldc 2

ldc 3

adi

ldc 4

cup f ;call user procedure

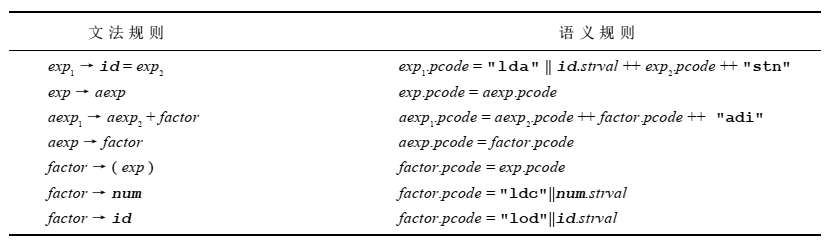
逻辑运算符短路：

a and b === if a then b else false

a or b === if a then true else b

生成代码时串的连接方式：++表示所连的串之间不能在同一行， | |表示用空格相隔

P-Code合成字符串的属性文法：



中间代码生成目标代码的技术：宏扩展(macro expansion)和静态模拟(static simulation)

宏扩展：宏扩展涉及到用一系列等效的目标代码指令代替每一种中间代码指令

静态模拟：静态模拟包括中间代码效果的直线模拟和生成匹配这些效果的目标代码