新编考研辅导丛书

Compiling Principle

编译原理辅导

胡元义 柯丽芳 编著

西安电子科技大学出版社
http://www.xdupb.com

新编考研辅导丛书

编 译 原 理 辅 导

胡元义　柯丽芳　编著

西安电子科技大学出版社

2001

内 容 简 介

本书是“ 编译原理 ”课程的辅导教材。书中的例题大多选自历届研究生入学考试试题，或是作者为帮助学生正确理解编译概念和原理，在多年教学实践中总结、设计出来的典型范例，具有一定的知识水平和代表性。本书对示例进行了细致、深入的分析和解答，这为读者熟练掌握编译技术、抓住重点、突破难点提供了有益的帮助。另外，针对某些难题，本书还提出了一些新的解题方法和思路。书中注有“ * ”的章节为选看内容。

本书既可作为考研复习辅导书，也可作为“ 编译原理 ”课程的学习指导书，此外，还可作为计算机软件人员的参考资料。

图书在版编目(CIP)数据

编译原理辅导 / 胡元义，柯丽芳编著. —西安：西安电子科技大学出版社，2001．8

(新编考研辅导丛书)

ISBN 7–5606–1046–3

Ⅰ. 编… Ⅱ. ① 胡… ②. 柯… 编译程序–研究生–入学考试–自学参考资料 Ⅳ. TP314

中国版本图书馆 CIP 数据核字(2001)第 048029 号

策　　划　李惠萍　毛红兵

责任编辑　钟宏萍

出版发行　西安电子科技大学出版社(西安市太白南路 2 号)

电　　话：(029)8227828　　邮　编　710071

http: //www.xduph.com　　E-mail: xdupfxb@pub.xaonline.com

经　　销　新华书店

印　　刷　西安兰翔印刷厂

版　　次　2001 年 8 月第 1 版　　2001 年 8 月第 1 次印刷

开　　本　787 毫米×960 毫米　1/16　印张 15.625

字　　数　310 千字

印　　数　1～4 000 册

定　　价　19.00 元

ISBN 7–5606–1046–3/TP・0521

如有印装问题可调换

本书封面贴有西安电子科技大学出版社的激光防伪标志，，无标志者不得销售。

此为试读，需要完整PDF请访问：www.ertongbook.com

前 言

计算机语言由单一的机器语言发展到现今内容迥异的数千种高级语言，就是因为有了编译技术。编译技术是计算机科学中发展得最迅速、最成熟的一个分支，它集中体现了计算机发展的成果与精华。在编译原理与技术的学习中，我们可以领略到计算机大师们那种化繁为简、点石成金的超然功力，而在编译符号的字里行间也处处闪烁着大师们智慧的火花。

本书作为编译原理的学习辅导书，其内容与要点大多通过例题给出。全书共分为八章：第一章对编译方法及高级语言进行了综述，并提出了求解值参和变参传递的新方法——动态图法；第二章重点介绍了词法分析的自动生成，即有限自动机 DFA 的构造；第三章主要涉及算符优先文法（自下而上分析）和预测分析法（自上而下分析）的语法分析；第四章重点介绍了语法分析的自动生成——各类 LR 分析器的构造；第五章重点介绍了语法制导翻译中典型语句及表达式到四元式的翻译；第六章讨论了运行空间的组织问题；第七章重点介绍了局部优化和循环优化；第八章作为选看内容简要介绍了符号表及错误处理。此外需要说明的是，本书略去了有关代码生成的内容。

在本书的编写过程中，得到了西安电子科技大学出版社李惠萍副编审的具体指导，在此表示衷心的感谢。

书中出现的不足和差错之处，敬请广大读者批评指正。

编　　者

2001 年 6 月 5 日

目 录

第一章

高级语言与编译

1.1 编译程序概论

计算机执行一个高级语言程序一般要分为两步：第一步，用一个编译程序把高级语言翻译成机器语言程序；第二步，运行所得的机器语言程序，求得计算结果。

通常所说的翻译程序是指这样的一个程序，它能够把某一种语言程序（称为源语言程序）改造成另一种语言程序（称为目标语言程序），而后者与前者在逻辑上是等价的。如源语言是 FORTRAN、PASCAL、COBOL 或 C 这样的“高级语言”，而目标语言是汇编语言这类的“低级语言”，这样的一个翻译程序就称为编译程序。

编译程序的工作贯穿于从输入源程序开始到输出目标程序为止的整个过程，是非常复杂的。一般来说，整个编译过程可以划分成五个阶段：词法分析、语法分析、中间代码生成、优化和目标代码生成。

第一阶段，词法分析。词法分析的任务是输入源程序，对构成源程序的字符串进行扫描和分解，识别出一个个单词符号，如基本字、标识符、常数、算符和界符等。在词法分析阶段的工作中所依循的是语言的构词规则。

第二阶段，语法分析。语法分析的任务是：在词法分析的基础上，根据语言的语法规则（文法规则）把单词符号串分解成各类语法单位（语法范畴），如“短语”、“子句”、“句子（语句）”、“程序段”和“程序”。通过语法分解确定整个输入串是否构成一个语法上正确的“程序”。语法分析所依循的是语言的语法规则。

第三阶段，中间代码产生。这一阶段的任务是对各类不同语法范

此为试读，需要完整PDF请访问：www.ertongbook.com

畴按语言的语义进行初步翻译的工作。把语法范畴翻译成中间代码所依循的是语言的语义规则。一般而言，中间代码是一种独立于具体硬件的记号系统，但它与现代计算机的指令形式有某种程度的接近，或者能够比较容易地把它变换成现代计算机的机器指令。常用的中间代码有四元式、三元式、间接三元式和逆波兰记号等。

第四阶段，优化。优化的任务在于对前阶段产生的中间代码进行加工变换，以期在最后阶段能产生出更为高效（节省时间和空间）的目标代码。优化的主要方面有：公共子表达式的提取、循环优化、算符归约等。优化所依循的原则是程序的等价变换规则。

第五阶段，目标代码生成。这一阶段的任务是把中间代码（或经优化处理之后）变换成特定机器上的绝对指令代码或者是可重新定位的指令代码或汇编指令代码。这一阶段实现了最后的翻译，它的工作有赖于硬件系统的结构和机器指令的含义。

如果最终得到的目标代码是绝对指令代码，则这种目标代码可以立即运行；如果目标代码是汇编指令代码，则这种目标代码需经汇编之后方可运行。目前多数实用编译程序所产生的目标代码都是一种可重定位的指令代码，这种目标代码必须通过一个连接装配程序把各个目标模块连接、装配在一起，使之成为一个可以独立运行的绝对指令代码程序。

上述编译过程的五个阶段是编译程序工作时的动态特征，编译程序的结构可以按照这五个阶段的任务分模块进行设计。编译程序的结构示意如图 1－1 所示。

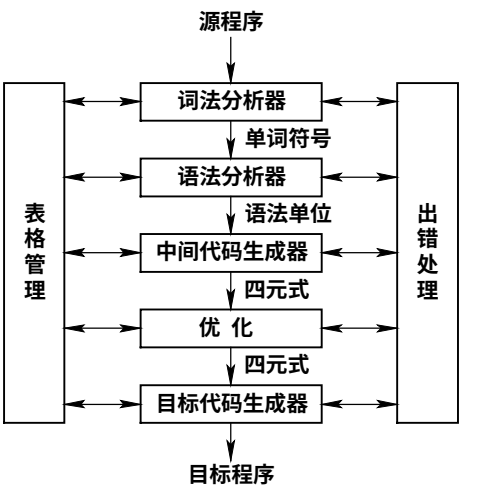

图 1－1　编译程序结构示意

编译过程中源程序的各种信息被保留在种种不同的表格里，编译各阶段的工作都涉及到构造、查找或更新有关的表格。因此，编译过程的绝大部分时间是花在造表、查表和更新表格的事务上。出错处理与编译过程的各阶段都有联系，与前三阶段的联系尤为密切。

现在已经建立了各种编制部分编译程序或整个编译程序的有效工具。有些能用于自动产生扫描器，有些可用于自动产生语法分析器，有些甚至可用来自动产生整个的编译程序。此外，有些编译程序还可通过“移植”得到，即把某一机器上的编译程序移植到另一机器上。这需要寻找某种适当的“中间语言”。但是，由于建立通用中间语言实际上办不到，因此移植也只能在几种语言和几种机种之间进行。

1.2 高级程序语言概述

高级程序语言是用来交流算法和计算机实现这两重目的的。高级语言一般较接近数学语言和工程语言，因此比较直观、自然和易于理解。并且，高级语言通常都是独立于机器的。

1.2.1 程序语言的定义

(1) 程序语言：一种程序语言是一个记号系统。如同自然语言一样，程序语言也是由语法和语义两方面定义的。任何语言程序都可看成是一定字符集 (称为字母表) 上的一个字符串，合乎语法的字符串才算是一个合式的程序。所谓一种语言的语法，是指这样一组规则，即用它可以形成和产生一个合式的程序。这些规则一部分称为词法规则，另一部分称为语法规则 (或产生规则) 。

(2) 词法规则：单词符号的形成规则。

(3) 语法规则：规定了如何从单词符号形成更大的结构 (即语法单位) 。换言之，语法规则是语法单位的形成规则，一般程序语言的语法单位有：表达式、语句、分程序、函数、过程和程序等。

语言的词法规则和语法规则定义了程序的形式结构，是判断输入字符串是否构成一个形式上正确 (即合式) 的程序的依据。

(4) 语义规则：对于一种语言，不仅要给出它的词法、语法规则，而且要定义它的单词符号和语法单位的意义，这就是语义问题。离开了语义，语言只不过是一堆符号的集合。所谓一种语言的语义，是指这样一组规则，即使用它可以定义一个程序的意义。这些规则称为语义规则。

一种程序语言的基本功能是描述数据和对数据进行运算。所谓程序，从本质上来说是描述一定数据的处理过程。一个程序大体上可视为图 1－2 所示的层次结构。

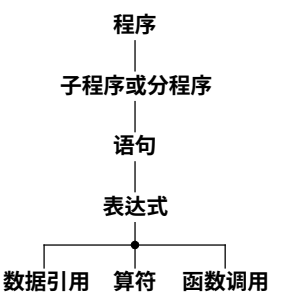

图 1－2 程序的层次结构

1.2.2 名字、类型、表达式与语句

1. 初等类型数据

一种程序语言必须提供一定的初等类型数据成分，并定义对于这些数据成分的运算。有些语言 (如 PASCAL) 还提供了由初等数据构造复杂数据的手段。常见的初等数据类型有：①数值数据；②逻辑数据；③字符数据；④指示器 (它的值指向另一些数据)。

2. 标识符和名字

在程序语言中各种名字都是用标识符表示的。所谓标识符，是指由字母或数字组成的，且只以字母为开头的字母数字串。注意，标识符是一个没有意义的字符序列，而名字却有明确的意义和属性。每个名字可看成是一个抽象的存贮单元的代表，这个单元的内容则认为是此名字 (在某一时刻) 的值。如果不指出名字的属性，它的值就无法理解。一个名字的属性包括类型和作用域。名字的类型决定了它能具有什么样的值，值在计算机内的表示方式以及对它能施加什么运算；名字的作用域则规定了它的值的存在范围。

3. 表达式

一个表达式是由运算量 (亦称操作数，即数据引用或函数调用) 和算符组成的。对多数程序语言来说，表达式的形成规则可概括为：

(1) 变量 (包括下标变量)、常数是表达式；

(2) 若 E_1、E_2 为表达式，θ 是一个二元算符，则 E_1 θ E_2 是表达式；

(3) 若 E 是表达式，θ 为一元算符，则 θ E (或 E θ) 是表达式；

(4) 若 E 是表达式，则 (E) 是表达式。

表达式中算符的运算顺序和结合性的约定大多和通常的数学习惯一致。

4. 语句

从功能上说，语句大体可分为执行语句和说明语句两大类。说明语句旨在定义各种不同数据类型的变量或运算，执行语句旨在描述程序的动作。

(1) 赋值语句：我们知道，每个名字有两方面的特征，一方面它代表一定的存贮单元，另一方面它又以该单元的内容作为值。对赋值句 A: =B，其意义是“ 把变量 B 的值送入变量 A 所代表的存贮单元。”亦即，对赋值号右边的 B 我们需要的是它的值，对左边的 A 我们需要的是它所代表的那个存贮单元 (地址) 。为了区分一个名字的这两种特征，我们把一个名字所代表的那个单元 (地址) 称为该名的左值，把一个名字的值称为该名的右值。也就是说，名字的左值指它所代表的存贮单元的地址，右值指该单元的内容。

变量 (包括下标变量) 既持有左值又持有右值。常数和带有算符的表达式一般认为只持有右值。但对指示器变量，如 P，它的右值 P↑既持有左值又持有右值。

(2) 说明语句：旨在定义名字的性质，许多说明语句没有相应的目标代码，编译程序把这些性质登记在符号表中，并检查程序中名字的引用和说明是否一致。

(3) 简单句和复合句：前者为不包含其它语句成分的基本语句，后者则指那些句中有句的语句。

1.2.3 程序段

一种语言的最高级结构是：分程序、过程和程序。对于分程序结构的语言来说，其分

程序的结构可以是嵌套的。也就是说，分程序内可以含有别的分程序。过程也可以看成是一个分程序，这个分程序可以在别的分程序中被调用。

分程序结构允许同一标识符在不同的分程序表示不同的名字。关于名字的作用域规定是：

(1) 一个在分程序 B_1 中说明的名字 X 只在 B_1 中有效 (局部于 B_1) ；

(2) 如果 B_2 是 B_1 的一个内层分程序，且 B_2 中对标识符 X 没有新的说明，则原来的名字 X 在 B_2 中仍然有效。如果 B_2 对 X 重新作了说明，那么 B_2 中对 X 的任何引用都是指重新说明过的这个 X。

换言之，标识符 X 的任一出现 (除出现在说明句的名表中外) 都意味着引用某一说明句所说明的那个 X，此说明句同所出现的 X 共处在一个最小分程序中，这个原则称为“ 最近嵌套原则 ”。

分程序结构的主要优点是可以非常有效地使用存贮器，其原因是一个分程序只有在执行时才需要数据空间，执行后所占用的空间即被释放。由于分程序结构的嵌套性，因此可用一个栈作为整个程序运行的数据空间。

1.2.4 参数传递

在 PASCAL 语言中，下面的过程段：

```
FUNCTION   MOD (X，Y) ：real；
    BEGIN
        MOD：=sqrt(x*x+y*y)
    END；
```

定义了一个称为 MOD 的函数过程。其中 X、Y 称为形式参数，简称形参。下面这个语句表示了对这个函数的一次调用：

```
A:=MOD (B,C) ;
```

其中，B、C 称为实在参数，简称实参。

下面，我们分别讨论参数传递的四种不同的途径，即传地址 (Call by reference) 、传值 (Call by value)、传结果 (Call by result) 和传名 (Call by name) 。“ 传名 ”常常也称“ 换名 ”，“ 传结果 ”也称“ 得结果 ”。

(1) 传地址：把实在参数的地址传递给相应的形式参数。在过程段中，每个形式参数都有一个对应单元，称为形式单元。形式单元将用来存放相应的实在参数的地址。当调用一个过程时，调用段必须预先把实在参数的地址传递到一个为被调用段可以拿得到的地方。如果实在参数是一个变量 (包括下标变量)，则直接传递它的地址；如果实在参数是常数或其它表达式 (如 A+B) ，那么就先把它的值计算出来并存放在某一临时单元之中，然后传

此为试读，需要完整PDF请访问：www.ertongbook.com

递这个临时单元的地址。当程序控制转入被调用段之后，被调用段首先把实参地址抄进自己相应的形式单元中，过程体对形式参数的任何引用或赋值都被处理成对形式单元的间接访问（即通过形式单元所存放的实参地址转而对实参进行操作）。这样，当被调用段工作完毕返回时，实在参数单元已经持有所期望的值。

(2) 传结果：与“传地址”相似（但不等价）的另一种参数传递方法。这种方法的实质是，每个形式参数对应有两个单元，第一个单元存放实参的地址，第二个单元存放实参的值。在过程体中对形参的任何引用或赋值都看成是对它的第二个单元的直接访问。但在过程工作完毕返回前必须把第二个单元的内容存放到第一个单元所指的那个实参单元中。

(3) 传值：一种最简单的参数传递方法。调用段把实在参数的值计算出来并存放在一个被调用段可以拿得到的地方。被调用段开始工作时，首先把这些值抄进自己的形式单元中，此后，就好像使用局部名一样使用这些形式单元，亦即形式参数如同是一种先从实参那里获得初值的局部变量，获得初值后就不再与实参发生任何联系了。这点与传地址不同，在传地址中，形式参数始终都与实参联系着（形参的值始终指向实参，而操作都据此而转向实参，即针对实参进行）。在传值方式下，如果实在参数不为指示器（即指针变量），那末，在被调用段里将永远无法改变实参的值。

(4) 传名：ALGOL 语言所定义的一种特殊的形—实参数结合方式。ALGOL 用“替换规则”解释“传名”参数的意义；过程调用的作用相当于把被调用段的过程体抄到调用出现的地方，但把其中任一出现的形式参数都替换成相应的实在参数（文字替换）。如果在替换时发现过程体中的局部名和实在参数的名字使用相同的标识符，则必须用不同的标识符来表示这些局部名。

1.3 过程与函数执行的分析方法

1.3.1 动态图描述规则

为了能够描述过程或函数调用执行的全过程，我们设计并采用被称之为动态图的方法来对程序的执行进行描述，即记录主程序和过程或函数的调用、运行及撤销各个阶段的状态，以及程序运行期间所有变量和过程、函数中值参（传值方式下的形式参数）与变参（传地址方式下的形式参数）的变化过程。动态图规则如下：

(1) 动态图纵向描述主程序、过程或函数各层之间的调用关系，横向由左至右按执行的时间顺序记录主程序、过程或函数中各变量值的变化情况。

(2) 过程或函数的值参均看作是带初值的局部变量（也可用箭头来表示实际参数传给值参的指向），其后，值参就作为局部变量参与过程或函数中的操作。对于变参，由于其作用就像指向实际参数的指针，故动态图中变参一律指向与其对应的实参变量（注意，两者

位于动态图相邻的两层上。如果实参为表达式，则用一个临时变量代表这个表达式）。此后，所有对变参的操作都是根据变参箭头所指对实参变量进行的。

(3) 函数名由于具有值的类型也看作为一个局部变量，但不同的是当此层函数调用结束时须将这个函数名所具有的值返回到上一层的调用者（可用箭头来指向）。

(4) 主程序、过程或函数按运行中的调用关系由上向下分层，各层说明的变量包括形式参数和函数都依次列于该层首列，各变量值的变化情况按时间顺序记录在与该变量对应的同一行上。

注意 动态图描述规则仅能够描述参数传递中的传值与传地址两种途径。

1.3.2 过程与函数执行描述示例

示例 1 试分析下面程序输出的结果，其中过程 silly 中的形参 x 为传值方式，而形参 y 为传地址方式。

```
PROGRAM varment(output);
  VAR   x,y,z : integer;
  PROCEDURE   silly (x,y);
    VAR z : integer;
    BEGIN  {silly}
      X:=10; y:=12; z:=14
    END;  {silly}
  BEGIN
    x:=1; y:=2; z:=3;
    silly(y,x);
    write(x,y,z)
  END.
```

分析 过程 silly 的形式参数 x 是值参，在过程调用时它接受了全局变量 y 的值 2；而形式参数 y 是变参，过程调用中它存放的是实参变量 x 的地址（图 1-3 中用箭头表示 y→x 的指向）。过程 silly 中还定义了一个局部变量 z，它与全局变量 z 同名。由名字作用域的“最近嵌套原则”知：在过程 silly 执行中局部变量 z 起作用；而在过程 silly 之外全局变量 z 起作用，局部变量的 z 并不存在。图 1－3 的动态图描述了整个程序的执行情况，其中包括过程 silly 因调用而创建、运行以及运行结束撤销的全过程，所有全局变量和局部变量的变化情况也在动态图中描述出来。动态图中“—”上的数值为主程序最后的输出结果，虚线“---”用来按

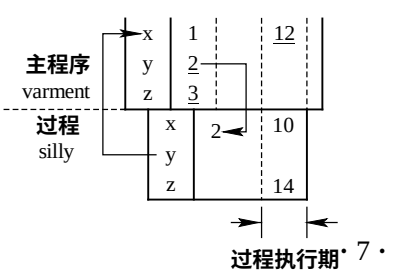

图 1－3 程序运行和过程调用动态图

时间顺序分隔各个变量值的变化情况。从动态图上可以看出：在过程 silly 中的变参 y 这一行上没有值出现，所有对变参 y 进行的操作都是根据 y 指针的指向对实参变量 x 进行的，它们分别位于相邻的两层上。由动态图可知程序的输出结果为：x=12; y=2; z=3。

示例 2　分析下面程序的输出结果，其中函数 f 中的形式参数 x 为传地址方式，n 为传值方式。

```
PROGRAM parament (output);
  VAR m,n:integer;
      x,y:real;
  FUNCTION f(x,n):real;
    BEGIN {f}
      x:=x/n;n:=n+1;m:=m-1;
      f:=x*y
    END; {f}
  BEGIN
    n:=5;m:=2;y:=10;
    x:=f(y,n);
    write(n,m,x,y)
  END.
```

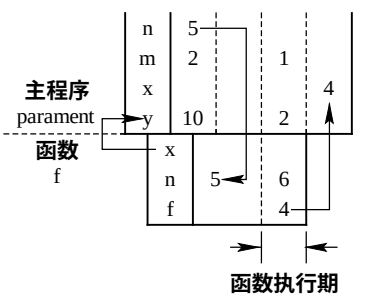

图 1－4　程序运行和函数调用动态图

分析　我们把函数名 f 看成一个局部变量，该程序执行的动态图见图 1－4。从动态图中可以看出：函数 f 执行中全局变量 m 和 y 的值都被改变。因为 x 指向实参 y，对 x 的赋值实际上是对实参 y 赋值；其次，函数 f 执行中对全局变量 m 进行了计算和赋值，故 m 值也被改变。由动态图可知，在函数 f 执行结束时函数名 f 具有值 4，这个值在函数 f 结束时被返回给主程序中的全局变量 x (动态图中函数名 f 值 4 的箭头所指) 。最终输出结果如下：

n=5;　m=1;　x=4;　y=2

1.4　典型范例解析

例 1.1　单项选择题：

(1) **编译程序是对**_____**。**

①　汇编程序的翻译　　②　高级语言的解释执行

③ 机器语言的执行　　④ 高级语言的翻译

(2) 将编译程序分成若干“遍”是为了_____。

① 提高程序的执行效率　　② 利用有限的内存，提高机器的效率

③ 使程序的结构更加清晰　　④ 利用有限的内存，降低机器的效率

(3) 指示器变量的右值_____。

① 持有右值　　② 持有左值

③ 既持有右值又持有左值　　④ 持有右值但不持有左值

(4) 中间代码生成时所依据的是_____。

① 语法规则　② 词法规则　③ 语义规则　④ 等价变换规则

(5) 使用_____可以定义一个程序的意义。

① 语义规则　② 词法规则　③ 产生规则　④ 左结合规则

解析　正确选择为： (1) ④；　(2) ③；　(3) ③；　(4) ③；　(5) ①

例 1.2　多项选择题：

(1) 中间代码主要有_____。

① 四元式　② 二元式　③ 三元式　④ 逆波兰记号　⑤ 间接三元式

(2) 编译程序各阶段的工作都涉及到_____。

① 语法分析　② 表格管理　③ 出错处理　④ 语义分析　⑤ 词法分析

(3) 编译程序工作时，通常有_____阶段。

① 词法分析　② 语法分析　③ 中间代码生成　④ 语义检查　⑤ 目标代码生成

解析　正确选择为： (1) ①③④⑤；　(2) ②③；　(3) ①②③⑤

例 1.3　对于下面的程序：

```
PROCEDURE   P (x,y,z)；
   BEGIN｛P｝
      Y:=Y+1;
      I:=z+x
   END;｛P｝
BEGIN｛main｝
   A:=2;
   B:=3;
   P(A+B,A,A);
   print A
```

END.｛main｝

若参数传递的办法分别为：(1) 传名，(2) 传地址，(3) 传结果，(4) 传值，试问当程序执行时所输出的 A 值分别是什么？

解析　(1) 传名的意义是：过程调用的作用相当于把被调用段的过程体抄到调用出现的地方，但把其中任一出现的形式参数都替换成相应的实在参数 (文字替换) 。所以，实际执行的程序为：

```
A:=2；
B:=3；
A:=A+1;          /*形参 Y 换成 A*/
A:=A+A+B;        /*形参 Z 换成 A，形参 X 换成 A+B*/
print A
```

由此得到 A=9 。

(2) 传地址，我们先用 T 代表表达式 A+B，再用动态图分析 (如图 1－5 所示)。由此得到 A=8。

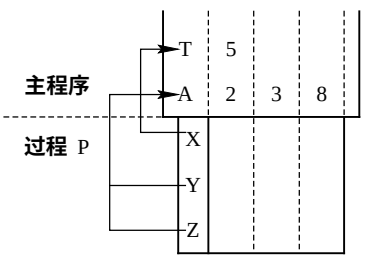

图 1－5　传地址时的动态图

(3) 传结果：传结果的方法是每个形式参数对应两个单元，第一个单元存放实参地址，第二个单元存放实参的值。在过程体中对形参的任何引用或赋值都看成是对它的第二个单元的直接访问。但在过程工作完毕返回前必须把第二个单元的内容存放到第一个单元所指的那个实参单元中。

在调用过程 P 前，A=2，B=3，设 T 代表 A+B 的临时单元，即 T=5。参数替换如图 1－6 所示。

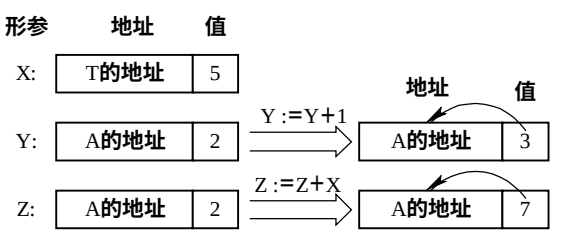

图 1－6　传结果示意图

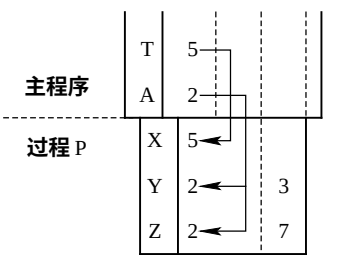

图 1－7　传值时的动态图

由图 1－6 所示，A=7 (注：先执行对应形参 Y 值返回，即将 3⇒A，然后再执行对应形参 Z 值返回，即又将 7⇒A) 。

(4) 传值：先用 T 代表 A+B 的临时变量，再用动态图分析(见图 1－7 所示)。由此得到

A=2。

例 1.4　对于下面的程序：

```
PROGRAM    main;
  a:integer;
  PROCEDURE test(x:integer);
    temp:integer;
    BEGIN｛test｝
      x:=a+1;
      temp:=a+2；
      x:=temp
    END;｛test｝
   BEGIN
    a:=2;
    test(a);
    print(a)
  END.
```

分别给出参数传递为：传地址、传结果、传值和传名时 a 的输出结果。

解析　(1) **传地址用动态图表示如图** 1－8 **所示，输出结果为：**a=5。

(2) **传结果示意如图** 1－9 **所示，输出结果为：**a=4 (**最终将** x **的值** 4 **送** a)。

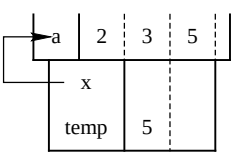

图 1－8　**传地址动态图**

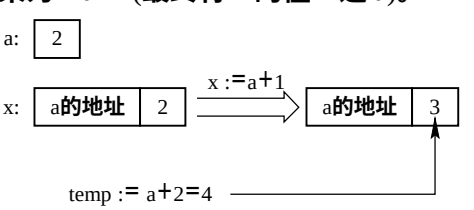

图 1－9　**传结果示意图**

(3) **传值动态图如图** 1－10 **所示，输出结果为：**a=2。

(4) **传名的执行程序为：**

```
a:=2;
a:=a+1;
temp:=a+2;
a:=temp;
print(a);
```

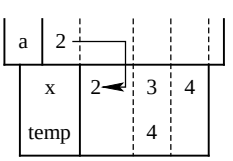

图 1－10　**传值的动态图**

此为试读，需要完整PDF请访问：www.ertongbook.com