青松

编译程序设计方法

邢玉国 张伟 编著

青岛出版社

编译程序设计方法

邢玉国　张　伟 编著

青 岛 出 版 社

鲁新登字 08 号

图书在版编目(CIP)数据

编译程序设计方法/邢玉国、张伟编著.-青岛：青岛出版社，1998.8
ISBN 7-5436-1982-2

Ⅰ. 编…
Ⅱ. ①邢…②张…
Ⅲ. 编译程序—程序设计
Ⅳ. TP314

中国版本图书馆 CIP 数据核字（98）第 20992 号

责任编辑　樊建修
装帧设计　青松美工

*

青 岛 出 版 社
（青岛市徐州路 77 号）
邮政编码：266071
新华书店北京发行所发行
青岛双星集团华信印刷厂印刷

*

1998 年 8 月第 1 版　　1998 年 8 月第 1 次印刷
16 开(787×1092 毫米)　　10.25 印张　225 千字
印数：1—3000
定价：15.00 元

前　言

这本书是根据作者多年编译系统实践和为计算机专业学生讲授编译方法课准备的一份讲稿整理而成，曾经在北京计算机学院计算机科学系、中国科大软件班、中国人大研究生班、青岛大学计算机系等讲授过。

本书以 FORTRAN 语言为主要加工对象，多遍扫描编译为讲解主线，介绍了状态矩阵法、分块优化、半目标模块的生成和装配技术。另外还简单地介绍了递归子程序法。

编译程序是计算机科学发展中的重要分支，有一套比较成熟的方法。学习编译程序的主要目的不在于将来具体搞一个编译程序，而在于学习这些方法，应用到其它实际的软件研制工作中去。软件是工程性的东西，仅限于原理性的了解远远不够，重要的在于具体实现。本书中有大量具体的框图，目的就在于此。因为无论多么好的逻辑设计思想，没有用具体的框图表示出来，是不能最后通过程序实现的。

书中八、九、十章主要内容选用北京中科院高能物理所牛永成、邵佩英、李伯民三位同志的部分工作成果。他们在 FORTRAN Ⅳ编译的研制中做出了很好的工作，曾为本书的这部分内容提供了参考材料，做过有益的讨论，得到过他们具体的帮助。

这次出版得到了青岛大学计算机系总支书记周正灵同志，主任邵峰晶教授，逯昭义教授大力支持，更得到了烟台大学计算机系主任陈守孔教授的帮助，在此对他们一并表示感谢。

学习不够、时间仓促、错误一定不少，敬请读者指正。

作者

1998 年 3 月

第一章　引　论

第一节　什么叫编译程序

一、翻译程序的提出

计算机功能强、效率高，是各个领域中不可缺少的工具，计算机科学已经发展成一门独立的学科。计算机本身只不过是一个二进制运算的机器，它所能提供人们的基本操作称为指令。其种类不外乎是下面几种：

存取型

运算型(+，－，*，/等)

逻辑判断型

转移型

输入输出型

指令的全体称为指令系统，也称为机器语言。指令的形式和数量因机器而异，但基本形式都是：

操作码	操作地址

指令在存储单元中以二进制编码存放。通常以八进制数或十六进制数格式书写。

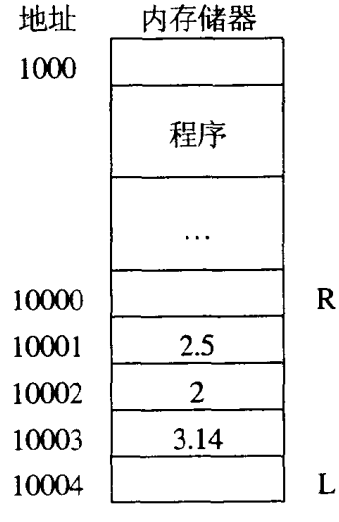
例如，已知半径求圆的周长：

R=2.5

L=2*3.14*R

用某种机器指令编写的程序是：

	程　序		注　释
1000	16	0010001	取 2.5
1	42	0010000	送 R
2	16	0010002	取 2
3	23	0010003	* 3.14
4	23	0010000	* R
5	42	0010004	送 L
6	输出打印指令)		打印 L
7	(停止指令)		程序停止



启动地址 1000，程序运行，得到运行结果。

这种用机器指令编写的程序一般称为手编，或称机器语言程序。最初人们就是这样使用计算机的。

这种程序的缺点是显而易见的：难学、易错、各机器指令不一样。因此计算机的使用和程序的编写局限于专业人员。

后来人们使用了符号化的指令编写程序，于是出现了汇编语言，用汇编语言编写的程序称为源程序，这种程序仍然不是人们习惯的形式，而格式也是多种多样。

例如，上面的程序可写成如下形式：

```
S→   2.5
→S   R
S→   2
*    3.14
*    R
S→   L
…
```

这种程序虽然省去了程序和数据(变量和常数)的地址分配，但基本上和机器指令相对应，同样难学易错。尽管这样，助记性符号究竟是前进了一步。这种程序机器并不能由计算机直接执行，于是出现了把这种源程序翻译成机器指令的汇编程序。

(汇编语言) 源程序 → 汇编程序 → 机器指令

这种机器指令的程序又称目标程序。后来这种汇编语言逐渐发展成高级语言，如 FORTRAN，ALGOL，PASCAL，COBOL，C 等等。人们用这些语言编写程序就像书写公式一样方便。例如用 FORTRAN 语言可直接写成如下形式的程序：

```
      REAL   L
      R=2.5
      L=2.0*3.14*R
      WRITE  (LP1,10)  L
10    FORMAT  (/3HL=Δ,F5.2)
      STOP
      END
```

这种程序虽然直观易学，但机器仍然不能直接执行，于是出现了把这种源程序翻译成目标程序的编译程序。

(高级语言) 源程序 → 编译程序 → 目标程序

除了汇编语言，高级语言外还有一种会话式语言，如 BASIC、APL 等，把这种源程序翻译成目标指令的程序称为解释程序。

(会话语言) 源程序 → 解释程序 → 目标程序

总括起来，翻译程序包括汇编程序、编译程序、解释程序。

而其中编译程序的工作就是把用高级语言写的源程序变成一个具体机器指令表示的目标程序。

二、编译程序和解释程序的区别

它们虽然都是把源程序变成目标程序，但它们的工作方式是有区别的，这一点又决定于源程序所使用的语言不同。

编译程序是把源程序整个的翻译成目标程序后，再去执行这种目标程序，得出结果。而解释程序则是翻译一句执行一句，人们可以随时干预机器的运行，正好满足会话式语言的要求。显然编译程序生成的目标程序质量高，运行时间短，而解释程序生成的目标程序质量差些，运行时间长，而这正是换来会话式工作方式的代价。

三、编译阶段和目标运行阶段

这是学习编译程序时经常用到的概念。用高级语言所写的源程序机器不能直接执行，必须经过两个阶段：编译阶段与运行阶段，所谓编译阶段是指编译程序在工作(动态执行中)，这时源程序是它的加工对象(作为输入数据)，而它的加工结果则是目标程序(输出数据)。所谓目标运行阶段则是生成后的目标程序处于动态执行阶段，这时编译程序则是处于毫不相干的静止状态。目标的运行则以程序员给的数据作为加工对象，输出程序员要求的结果，宣告自己工作的结束。

第二节　程序设计语言的定义

高级语言又称为程序设计语言，下面以 FORTRAN 为例直观的描述一下它的定义，在自然语言中如英语：

字母 → 词法规则 → 单词 → 语法规则 → 句子 → 语义规则 → 文章

一、字母：基本字符集

A ~ Z

0 ~ 9

+　-　*　/　=　.　,　(　)　Δ　$ 等

其中△表示空格

二、单词：具有独立意义的语法单位

(1) 专用定义符(也称关键字)

借用少数固定的英语单词，赋予程序设计语言的含义，FORTRAN 中共有 30 多个，如 DO、REAL、DIMENSION、SUBROUTINE 等。

(2) 符号名

定义为字母开头后跟字母数字串，用来标识变量名、数组名、过程名等，如 A、X1、B3X 等都可作为符号名，(不同的语言对其长度有所限制)。

(3) 常数

整常数　a (a 为 0 ~ 9 数字串)

实常数　a.　.b　a.b（基本实常数）

a.E±d　.bE±d　a.bE±d　aE±d（b为0 ~ 9数字串，d一般为两位数字）

双精度常数以D代替E，D±d表示$10^{\pm d}$

复常数(实常数，实常数)

逻辑常数　.TRUE.　.FALSE.

文字常数　如‘ABC ’或 3HABC

(4) 标号

定义性(出现句首)

使用性(出现句中)

形同整常数，表明程序的转移地址。

(5) 运算符

算术运算符　+　-　*　/　**

关系运算符 .LT.　.LE.　.EQ.　.NE.　.GE.　.GT.

逻辑运算符 .NOT.　.AND.　.OR.

(6) 其他

赋值号　=

圆点　．格式句中分隔符如F5.3(作小数点时，如3.5不把它单独作为一个单词)

圆括号　(　)

逗号　,

格式句中的格式控制符 I，F，E，D等等。

三、语句

两大类：非执行语句，一般不对应目标代码，只提供编译信息。

可执行语句，对应目标代码。

(1) 非执行语句

非执行语句又可分类型说明、种属说明、赋初值语句等。

类型说明，用以说明变量、数组、函数的类型(整、实、双、复、逻)。

例如　INTEGER A,B

COMPLEX　C1,C2

种属说明：如COMMON　/A/X,Y 指明公用量而不是局部于程序段中的量。

DIMENSION　B(10,10) 指明数组而不是变量。

EXTERNAL　S1,S2 指明作实元的S1，S2是过程。

在 FORTRAN 中数组说明也可以出现在类型说明或公用说明之中，总括起来无论是类型还是种属都是表明一个符号名的性质。为编译生成目标时的符号名的使用和存储分配提供依据。

赋初值语句指在编译阶段把程序员指出的量赋予初值。

(2) 可执行语句

赋值语句

三种转向语句

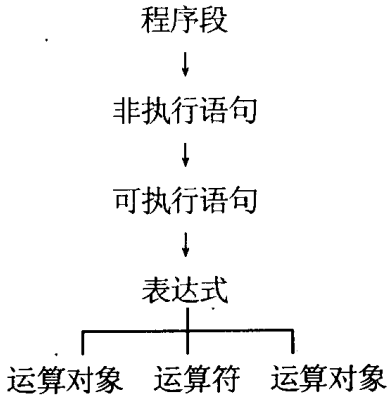
两种条件语句

循环语句
停与暂停语句
返回语句
调用语句
输入输出语句

每一种语句都有特定的语义规则，可以粗略地分为四类：用于计算的语句(如赋值语句)、用于控制转移的语句(例如转向条件等语句)、调用子程序的语句、输入输出的语句。编译可执行语句的工作就在于给出每个语句对应的具体机器指令(目标)。语句中最重要的语法成分是表达式，它在机器内的表示和目标实现成为编译程序最重要的工作之一。

四、文章：源程序

FORTRAN 程序是段结构(这和 C 语言一样)，每个程序段都具如下的层次结构。



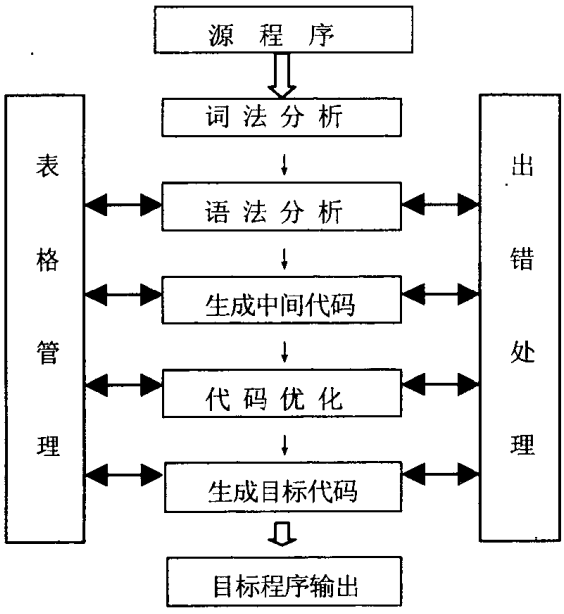
因此编译的语法分析就存在着自上而下和自下而上两种方法，其中每个语法成分都有逻辑和计算机实现两方面的意义。从数学上考察注重它的逻辑含义，从编译的角度注重它在计算机内实现的可能性和效率。

第三节　编译程序的组成

从语言翻译的逻辑上可分为五个部分，如图 1.1 所示。

从语言翻译的具体实现上分为一遍扫描和多遍扫描。

编译程序中关于“遍”的概念十分糊涂，遍的意思本来是从头到尾扫描一次，象阅读书刊一样，从头到尾看一遍。但编译程序中的“遍”应理解为一个编译阶段。在某一阶段中，对于源程序或由原程序生成的中间语言，为了取得足够的信息，完成本阶段编译的工作不只扫描一次，可能很多次。不过在习惯上对于分几个阶段的编译程序仍称为几遍。



编　译：　类比文章翻译

词法分析：　查字典识别单词

语法分析：　识别句子分析句型

生成中间代码：写成翻译初稿

代码优化：　修饰加工

生成目标代码：定稿译文

图 1.1

(1) 一遍扫描的编译程序

为了取得快速编译，不要求更高的目标质量，通常采取一遍扫描编译程序。所谓一遍一般不生成什么中间代码，将分析、优化等编译加工合在一起进行。作为一个编译阶段，一遍扫描生成的半目标程序(操作地址为相对的)经过装配代真即可生成运行的目标程序。

早期的编译较为简单，一般只有管理外部设备的所谓管理程序，没有完备的操作系统支持，也没有文件系统，单道运行。源程序从输入、换码、编译加工，到生成目标程序一般都在内存进行，不和外存打交道。一遍编译如图 1.2 所示。

有些编译系统中，源程序的输入，装配连接程序已独立出来，作为操作系统支持下的程序单独运行。

图 1.2 中把读单词(词法分析)作为一个子程序供随时调用，在多遍扫描的编译程序中有时也作为单独的一遍进行。

(2) 多遍扫描的编译程序

如不过分考虑节省编译时间，为了生成高质量的目标代码，一般在操作系统支持下，采用多遍扫描的方法完成整个编译过程。

多遍编译尽管编译时间长，但编译各阶段工作清楚、接口简单、优化充分，易于被初学者接受，本书就以多遍编译为讲授的基本思想。

一个多遍扫描编译示意图如图 1.3 所示。

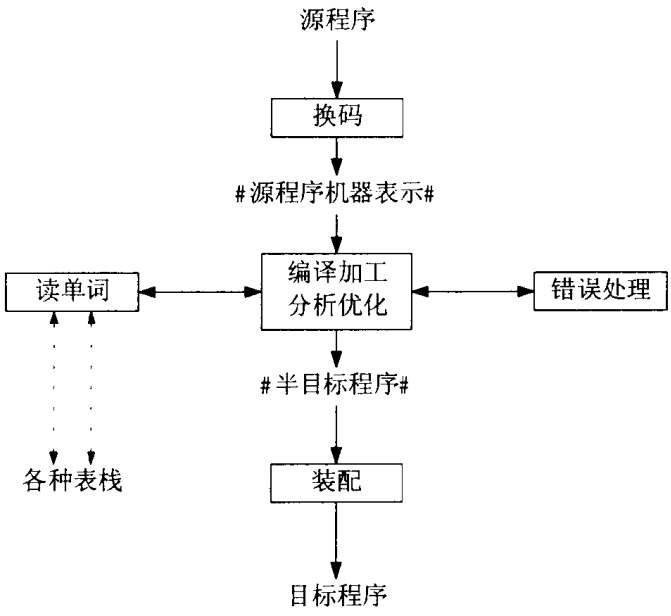
通常的作法(IBM 的 FORTRAN(H)编译)是：

① 将词法分析仍然作为一个子程序调用。

② 用户可选择优化阶段。

③ 编译的加工对象是一个源程序段。

④ 把装配程序从编译程序中分离出来，这对模块化语言十分方便。



方框表示各处理程序，实线表示程序流程，虚线表示信息流程。

图 1.2

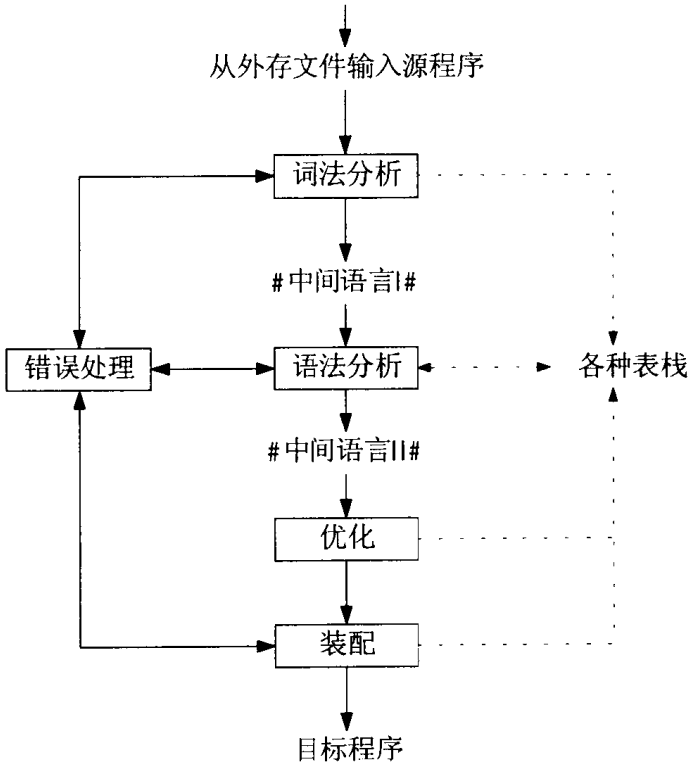


图 1.3

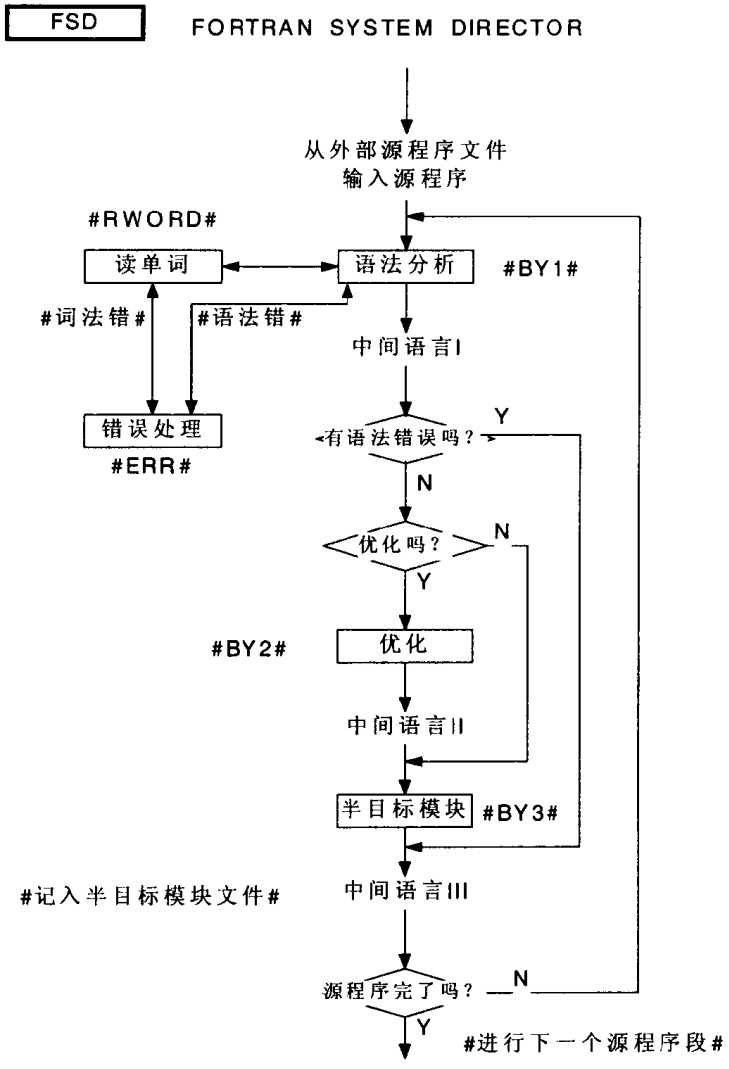

FSD
FORTRAN SYSTEM DIRECTOR
从外部源程序文件
输入源程序
#RWORD#
读单词
语法分析
#BY1#
#词法错#
#语法错#
中间语言I
错误处理
#ERR#
有语法错误吗？
Y
N
优化吗？
N
Y
#BY2#
优化
中间语言II
半目标模块
#BY3#
#记入半目标模块文件#
中间语言III
源程序完了吗？
N
Y
#进行下一个源程序段#

图 1.4

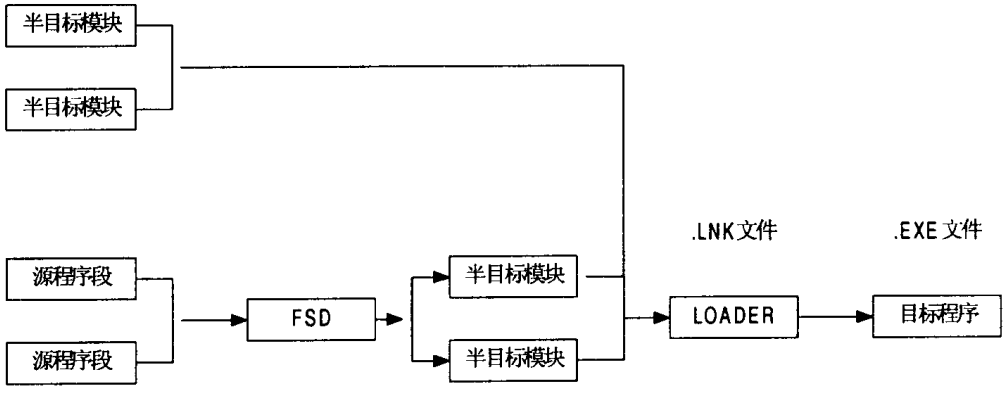

半目标模块
半目标模块
.LNK文件
.EXE文件
源程序段
源程序段
FSD
半目标模块
半目标模块
LOADER
目标程序

图 1.5

一个源程序从输入到计算机到最后输出计算结果，是在用户的界面上操作，由操作系统提供的命令(或称操作语句)实现的。

输入：纸带、卡片、键盘打入→源程序文件。

启编：启动编译程序(FSD)运行→半目标模块文件。

装配：启动装配程序(LOADER)运行→目标文件。

启目：将目标调入内存，启动运行→计算结果。

由键盘输入源程序，经过启编发现源程序有错，还可以通过命令调用操作系统提供的编辑程序修改，生成源程序的正确文件再进行编译，直到无错为止。用户的优化选择也是由命令实现的。一般在源程序输入之后，启编之前，用户打一个优化命令，通知编译程序，对该程序进行优化。一般优化是针对调试成功的程序，而且要多次进行。

第四节　三遍扫描的编译程序和装配程序工作简述

一个具体例子：

```
      SUBROUTINE  S(X，Y)
      INTEGER  X
      COMMON /Z/A，B
      LOGICAL  A
1     Y=5*2+X-B
      IF(A) GOTO  1
      RETURN
      END
```

一、语法分析遍

建立字典和中间文本，统称中间语言，中间语言的形式有多种(见第三章)，此处用三元组。

```
(1)  ( DUA   X   △ )
(2)  ( DUA   Y   △ )   } SUBROUTINE S(X, Y)
(3)  ( DEFS  S   △ )
(4)  ( LAB   △   L1 )
(5)  (  *    5   2 )
(6)  (  +   (5)  X )   } Y=5*2+X-B
(7)  (  -   (6)  B )
(8)  (  =   (7)  Y )
(9)  ( LIF   A   L2 )
(10) ( GOTO  △   L1 )  } IF (A) GOTO 1
(11) ( LAB   △   L2 )
(12) ( RETURN △ △ )    RETURN
```

SD 字典

sd→

名字	类型种属	地址
S	子程序	
X	整型、哑变量	
Y	实型、哑变量	
A	逻辑型、公用	
B	实型、公用	

CD 字典

cd→

类型	数值
整型	2
整型	5

三元组文件中的第一项是伪操作码，另外两项为操作对象。

伪操作码只是编译约定的内部编码。

操作对象部分：名字是指它在 SD 字典的地址
常数是指它在 CD 字典的地址
标号是指它在 LD 字典的地址
△表示空
(i)表示上一个三元组的结果

LD 字典

ld→

信息	标号
	L1
	L2

注：L2 编译加的内部标号

SD 字典地址一栏，第一遍结束时，根据符号名属性分配相对地址。例如 X，Y 分在一般变量区，A，B 分在公用区等，每个区头地址为 0，此时只分房号，不分运行时的内存空间。

二、优化遍

为提高生成目标的质量，对三元组进行优化工作一般是重排、删除某些三元组，而不改变中间文本项的基本结构。优化工作有很多，有效的是表达式优化和循环优化，详见第八章。这里举例说明一下，对上例 5，6，7 表达式三元组：

(5) (* 5 2)
(6) (+ (5) X)
(7) (- (6) B)

直接生成目标 →

取 5
* 2
\+ X
\- B

四条指令

优化时第 5 个三元组可以去掉，编译时可算出：

(5) (C 10 △)
(6) (+ (5) X)
(7) (- (6) B)

优化后生成目标 →

取 10
\+ X
\- B

三条指令

其中 C 是一个不产生目标指令的操作码，常数 10 由编译算出，填入 CD 字典。

C D 字典

类型	数值
整型	2
整型	5
整型	10

←优化时生成

三、生成半目标模块

(1) 字典的整理和改造，保留有用的，去掉无用的。例如从 SD 字典中保留过程项改为过程字典 PD，保留 CD 字典。

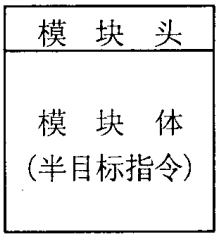
过程字典(PD)

属性	地址	名字
过程	V	S

CD 字典

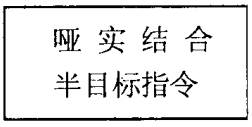
类型	数值
整型	2
整型	5
整型	10

(2) 将三元组中的操作对象由对字典指示字的操作，改为分配的相对地址的操作，生成半目标模块，最后生成如下形式：



模块头：过程字典(PD)
常数字典(CD)
公用区等

上面例题的模块体示意如下：



d1	取	10
+1	+	X
+2	-	B
+3	送	Y
+4	取	A
+5	零转	d1+7→L2
+6	转	d1
+7	返回	

详见第九章。

四、装配程序的工作

从目标文件上调入所有的半目标模块，然后进行：

① 存储分配：各段局部量的覆盖分配。
　　各段公用量的分配，在机器中指出绝对位置。

② 半目标指令操作地址代真成绝对地址，成为可运行的目标指令。

③ 进行程序段间的调用连接。

第二章　词法分析——识别单词

这一章把读单词程序设计成一个独立的子程序 RWORD，供语法分析识别一个语句时随时调用，每经过它一次就从源程序中识别出一个单词。

单词是由字符组成的，正像英语单词是由字母组成一样。因此要识别出一个单词必需从字符读起。本章介绍两种方法：直接分析法和状态矩阵法。前一种方法是从单词的第一个字符分析起，根据单词的不同构成，设计成几个不同的子程序去处理；后一种方法是根据单词的组成规则，用一张表(称为状态矩阵)把它反映出来，实际上就是一个能够解释执行的某种数据结构，通过其中的加工程序识别单词。

识别出的单词要进行分类，正像看英文单词要分成名词、动词、形容词等一样，以便为句子的语法分析作好准备。单词的分类用一种类别码(编译内部给出的编码)表示，称为单词的内码，一般放在一个固定的编译工作单元 KIND 中。在进行语义加工时，除了要知道单词的类别之外，还要知道这个单词具体是什么。比如一个常数 5.3 单词，类别是常数，而不是符号名，具体值是 5.3。因此在识别单词时除了记录单词的内码外，还要记录单词本身的一些信息。这些信息也分别放在编译所给出的一些工作单元之中，进而填入字典。

第一节　读字符

处理字符就是要设计读字符程序，它的任务是顺序地从源程序的文件中读出一个基本字符，通常放在编译所给出的一个固定工作单元 CH 中。读字符程序 RCHAR 是读单词程序经常调用的一个子程序。

处理字符的具体工作视不同语言而定。在 FORTRAN 语言中，每个语句后没有特殊的结束符(ALGOL，PASCAL，C 等语言中用分号，作为语句结束符)，可以由程序员打入一个或多个‘回车’‘换行’符，读字符程序要处理为一个，即读掉多余的，保留一个作为识别语句的结束，文字常数中的‘空格’符和格式句中文字说明符中的‘空格’以外的‘空格’符应全部读掉，因为这些字符大多是为了改善源程序书写外观而加入的，去掉它们不会改变源程序的功能。此外还要读掉注解行，因为注解行是为了增加源程序的可读性而加入的，原则上它不是程序的内容。此外还要连接续行等。总而言之，每执行读字符程序一次，顺序读出一个有效字符。FORTRAN 源程序格式，第一列字符为 C 或*时，该行为注解行，编译不处理，应在读字符程序中略过。1 至 5 列可以为标号，应在读字符程序中识别出来，作为整常数存放在固定单元 LAB 中，供语法分析程序使用。第 6 列除空格以外的非空字符作为续行处理，且在语句结束时给出回车换行符(≡)。语句从第 7 列开始，每读出一个

有效字符存放于固定单元 CH 中供读单词程序识别单词之用。

下面框图所调用的 RCHA1 程序，只从源程序中读一个字符，存放于 CH 中，不做任何处理，列号加 1，这个程序请读者自己编写。

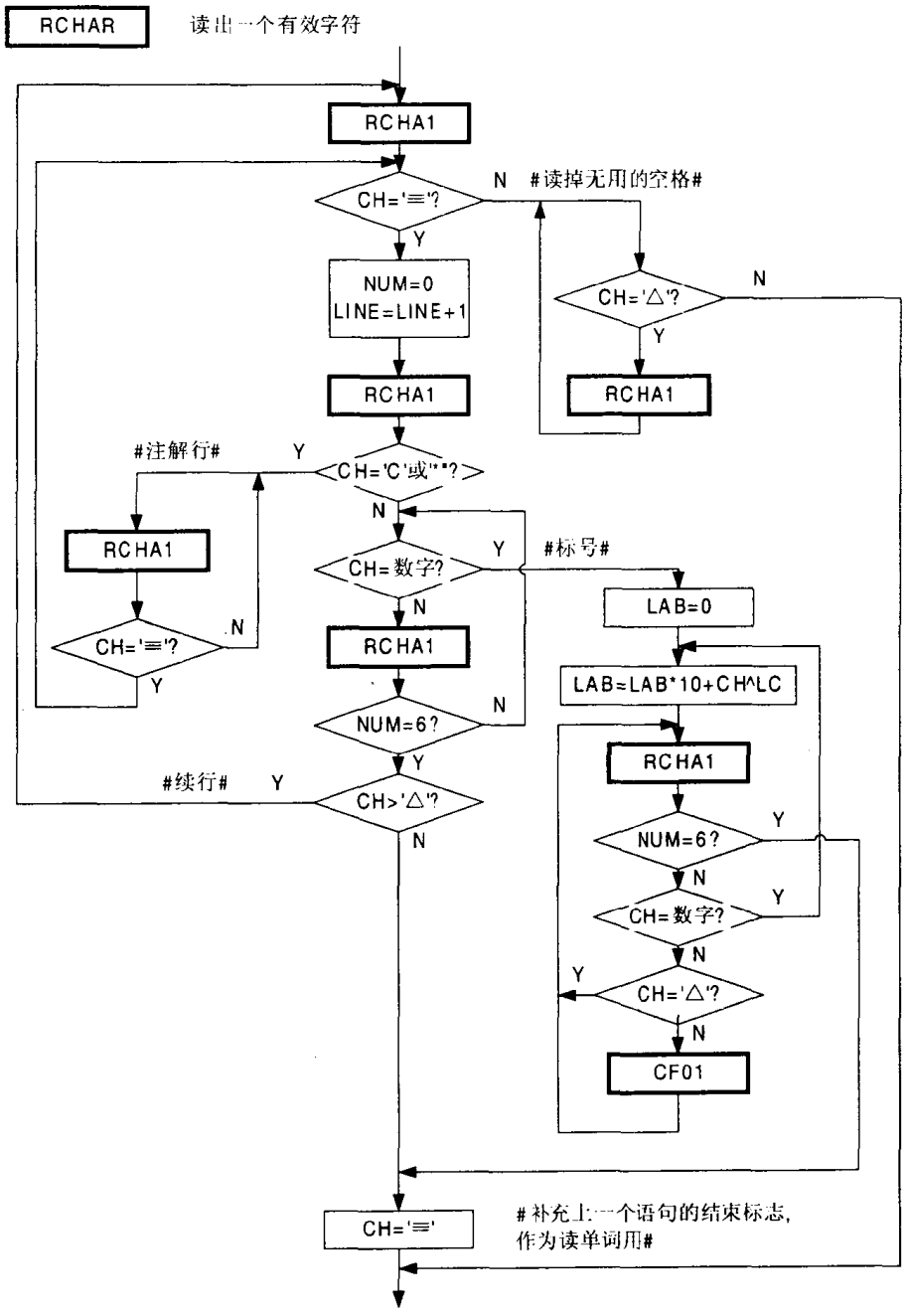


图 2.1

[注]

LINE: 行号，初值为 0

NUM: 列号

LC : 二进制数 00001111，CH^LC 将字符编码转换为数值