华中科技大学

编译原理

Compiler Principles and Techniques

主讲老师：

QQ群：

2019年11月4日星期一

编译原理课程组

课程信息

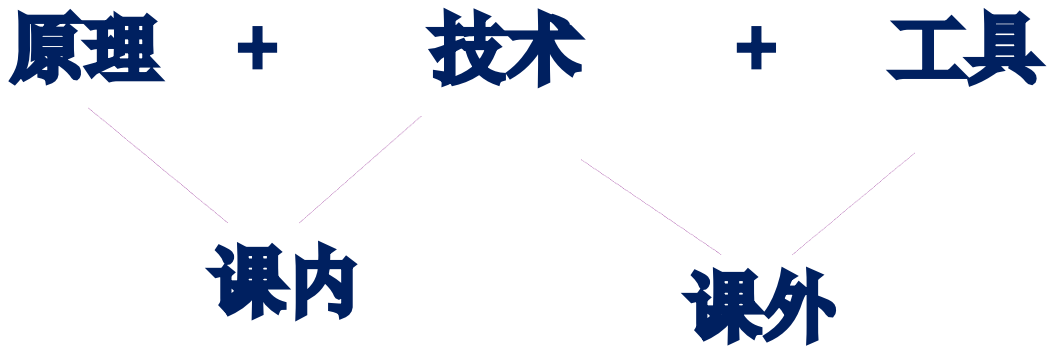
- **课名：编译原理**
- **类别：选修**
- **学时：32（课堂教学）+16（上机实验）**
- **上课时间：**
- **实验时间：**
- **考试：（）**

课程地位

- **计算机专业主干课**
  - 编译程序（系统）是计算机系统的核心支撑软件
  - 贯穿程序语言、运行时系统、体系结构
  - 联系计算机科学和计算机系统的典范
- **专业工作者必备的基本技能**
  - 编译原理的知识影响到专业人员的素质
  - 大量专业工作与编译技术相关
  - 高级语言实现，体系结构设计与优化，硬件综合，二进制翻译，智能编辑器，面向领域的语言以及业务逻辑语言的实现，软件静态分析，逆向工程，调试器，模型驱动的开发，程序验证，…

教学目的与要求

- 掌握编译程序/系统设计的基本原理
- 掌握“常见”语言机制的实现技术
- 经历开发一个小型编译程序的主要阶段
- 自学并使用自动构造工具
- 加深对计算机系统的理解
- 体会将所学知识灵活应用



先行课学习

- **《高级语言程序设计》（Java, C/C++）**
- **《数据结构》**
- **《操作系统》**
- **《计算机系统结构》**
- **《汇编语言》**
- **《计算机原理》**
- **《计算机系统联合实验》**

教材及主要参考书目

1. 王生原，董渊，张素琴 . 北京：清华大学出版
2. Alfred Aho ect.，Com Tools，北京：人民邮版集团，2002.2.
3. 许畅等.编译原理实践社.

章节内容

第1章　引论

第2章　文法和语言

第3章　词法分析

第4章　自顶向下语法分析方法

第5章　自底向上优先分析（课外自学）

第6章　LR分析

第7章　语法制导的语义计算

第8章　静态语义分析和中间代码生成

第9章　运行时存储管理

第10章　代码优化和目标代码生成

实验内容

- **要求：实现一个小型（面向对象）语言**

  （给定架构下扩展或改造）

- **分阶段进行**

  - **实验1　词法和语法分析（使用lex和bison）**
  - **实验2　　语义分析**
  - **实验3　　中间代码生成**
  - **实验4　　目标代码生成**

课程设置目的和要求——考试要求

- **题型：单选多选、判断、综合**
- **重点和难点**
  - **会在各章的开始点明**
- **考试权重**
  - **平时成绩（课后作业或出勤情况）占10%**
  - **实验占30%（其中：词法分析 语法分析40%、 语义分析30%、中间代码生成10%、目标代码10%）**
  - **期末考试占60%**
- **答疑：QQ群留言，离线答疑**

第 1 章 引 论

重点讲解

1.1 什么是编译程序

1.2 编译过程和编译程序结构

1.3 解释程序和一些软件工具

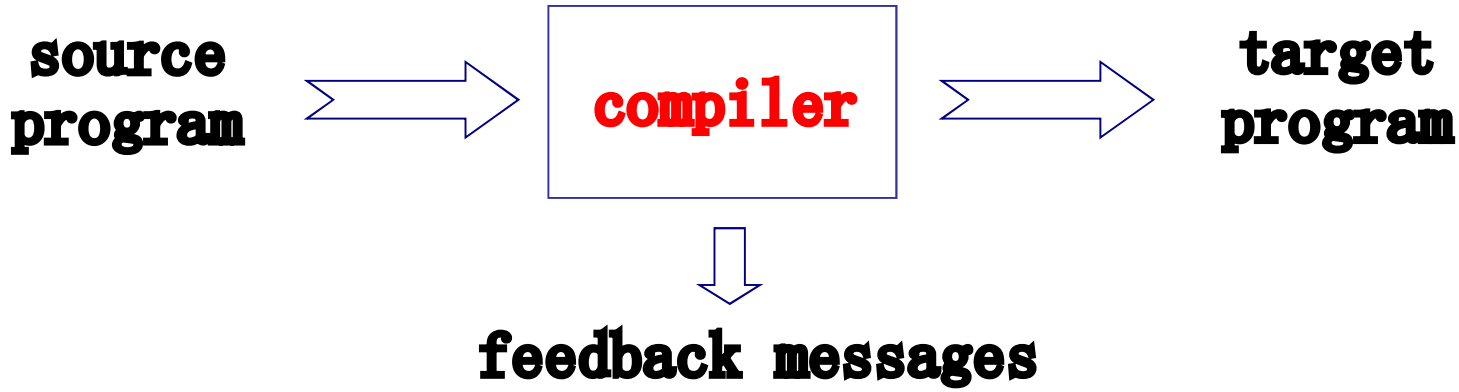
1.4 编译技术的新发展

1.1什么是编译程序(compiler)

编译程序是现代计算机系统的基本组成部分。

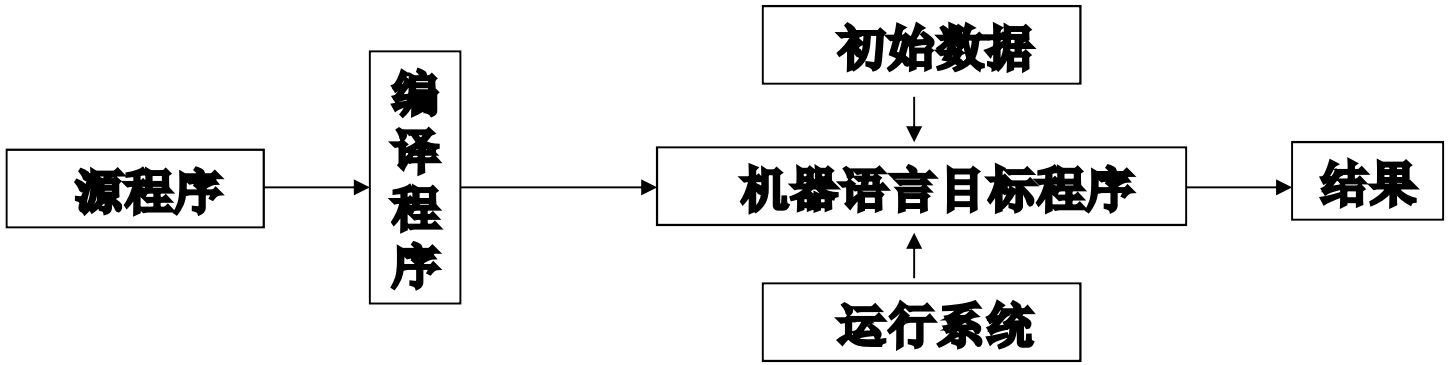
从功能上看，一个编译程序就是一个语言翻译程序，它把一种语言(称作源语言)书写的程序翻译成另一种语言(称作目标语言)的等价的程序.

目的：使得程序员不必考虑机器的细节，独立于机器。

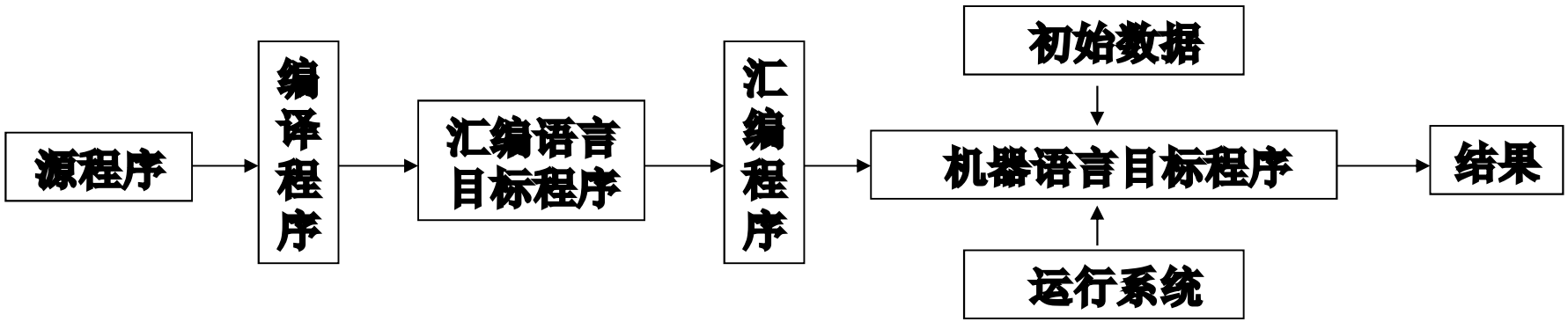


1.1什么是编译程序(compiler)

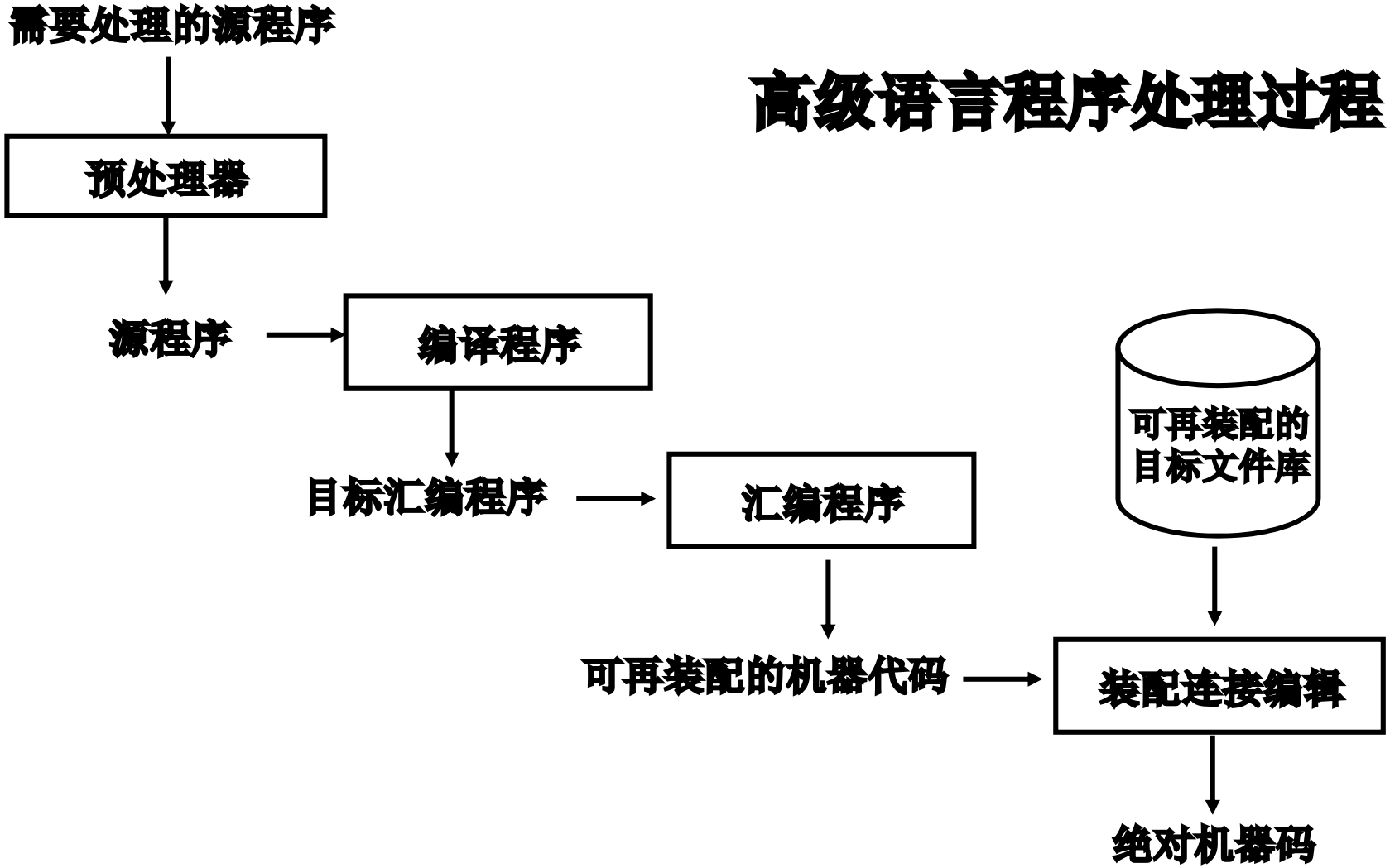
源程序的编译和运行阶段(编译的目标代码为机器语言):



源程序的编译、汇编和运行阶段（目标代码为汇编）:



1.1 什么是编译程序(compiler)



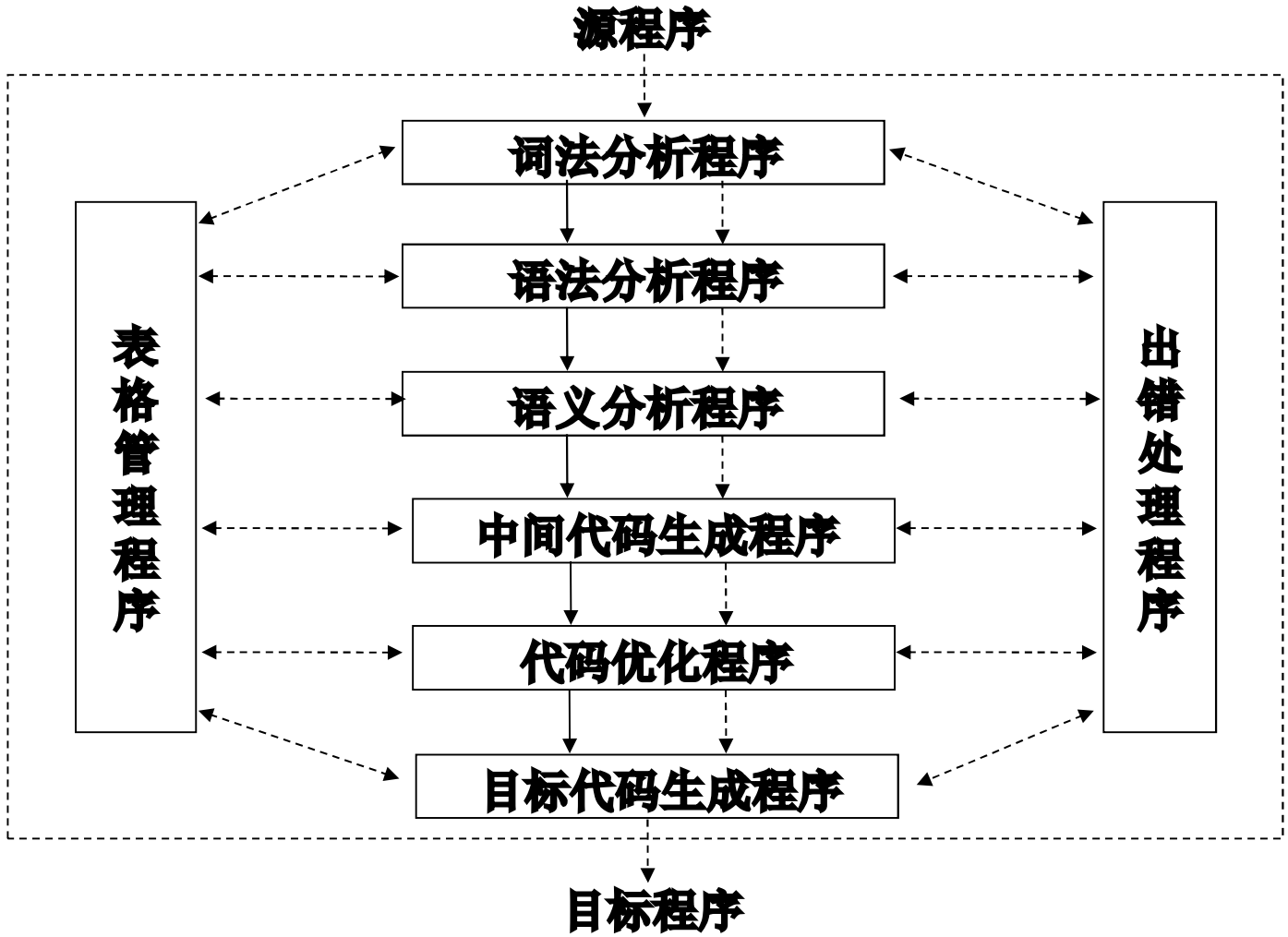
1.2 编译过程和编译程序结构

1.2.1 编译过程

计算机语言编译阶段	自然语言翻译阶段
1. 词法分析	1. 词汇学习
2. 语法分析	2. 句子结构分析
3. 语义分析	3. 句义分析
4. 中间代码生成	4. 译文草稿
5. 代码优化	5. 译文修饰
6. 目标代码生成	6. 译文定稿

表1.3 自然语言和计算机语言的翻译阶段对照表

1.2.2 编译程序结构


源程序
词法分析程序
语法分析程序
语义分析程序
中间代码生成程序
代码优化程序
目标代码生成程序
表格管理程序
出错处理程序
目标程序

1.2.3 编译阶段组合

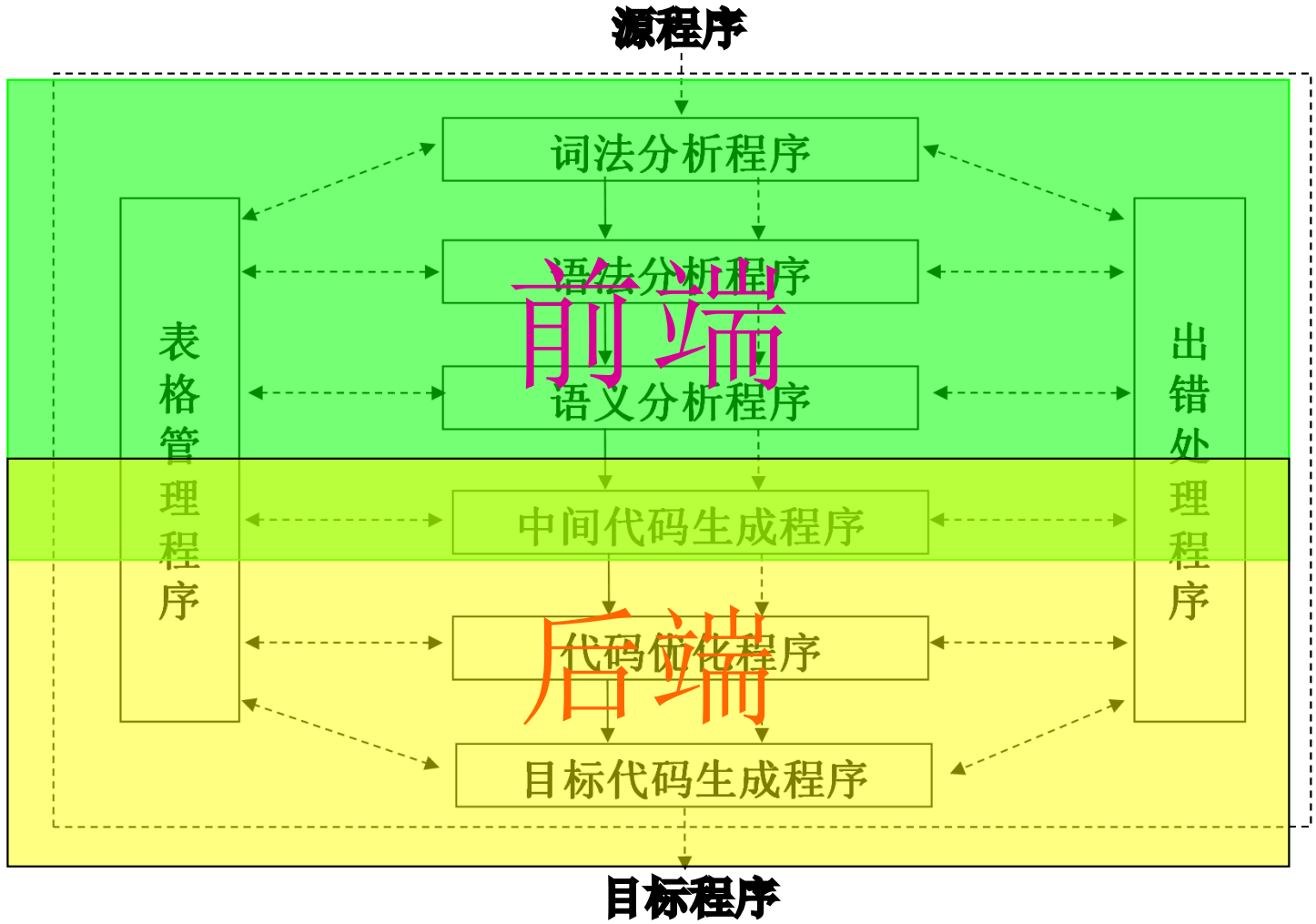
编译过程的阶段是一种逻辑上的划分。在具体设计和实现上，可以重新组织系统模块结构。

（1）划分“前端/后端”。 将与仅依赖于源程序而与目标机器（硬件）无关的阶段组合成前端，将与目标机器（硬件）相关的阶段组合成后端。

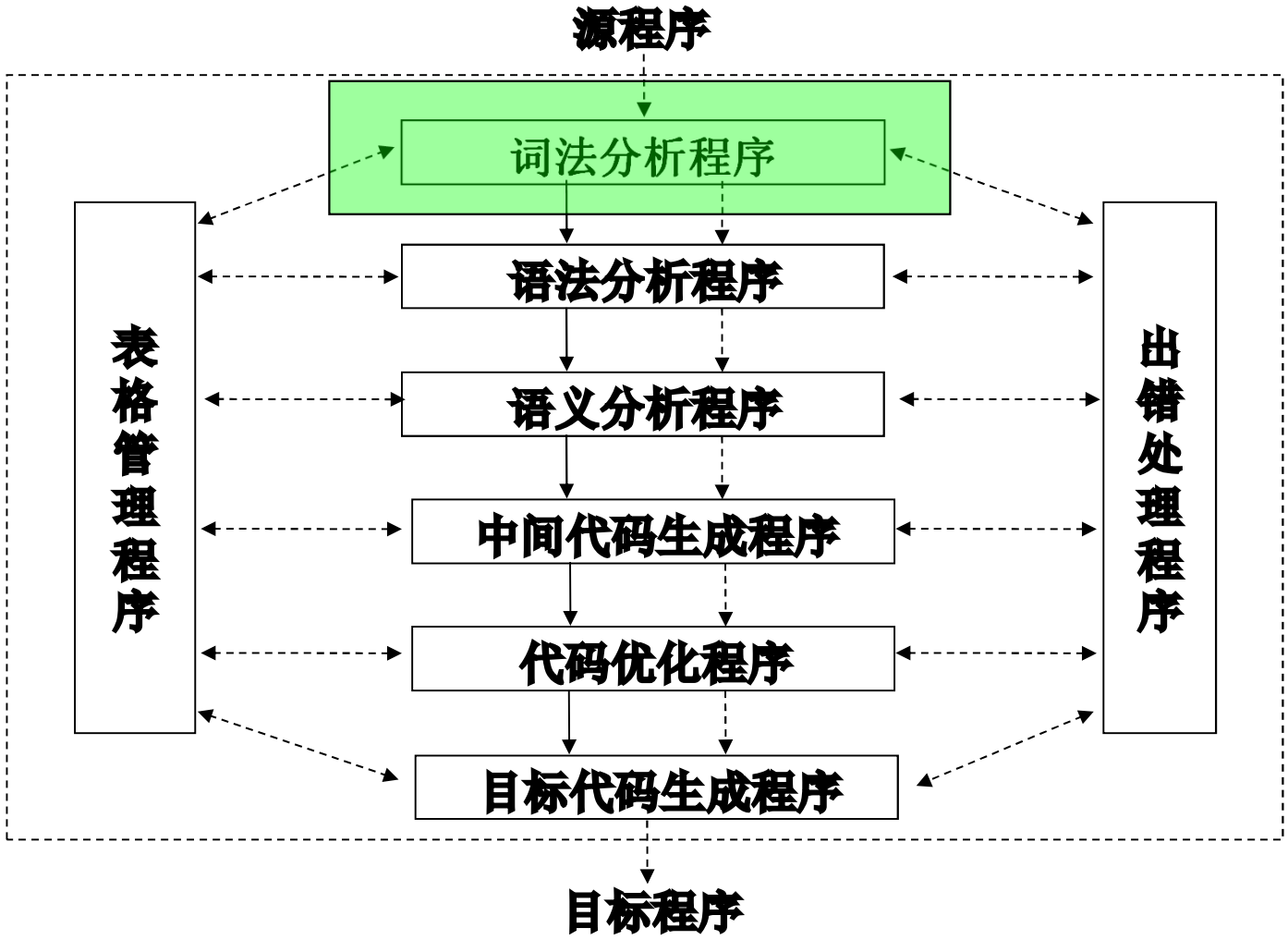
（2）划分“遍”。每遍可以完成编译的若干阶段的编译任务。

“遍”也称为“趟”。所谓“一遍”，就是对源程序或等价的中间语言程序，从头到尾扫视一次，并完成一定编译任务之过程。

1.2.3 编译阶段组合


源程序
词法分析程序
语法分析程序
语义分析程序
中间代码生成程序
代码优化程序
目标代码生成程序
表格管理程序
出错处理程序
前端
后端
目标程序

词法分析（Lexical Analysis）



词法分析（Lexical Analysis）

英文句子由单词构成

This line is a longer sentence.

（字母组成的有具体含义的最小成分）

- 单词的特性
  - 每个单词都有明确意义
  - 单词中字母的顺序是固定的（ist his linealo gerse nte nce 不是正确的单词）
  - 单词的个数是有限的
  - 在句子中空格是单词分隔符
  - 句点是句子结束标志

词法分析

从左至右扫描字符流的源程序、分解构成源程序的字符串，识别出(拼)一个个的单词（符号）

单词符号是语言中具有独立意义的最基本结构。多数程序语言中，单词符号一般包括：

①常数、

②保留字、

③标识符、

④运算符、

⑤界符等类型。

词法分析

```
double f = sqrt(-1);
```

(1) 保留字	double
(2) 标识符	f
(3) 赋值号	=
(4) 标识符	sqrt
(5) 左小括号	(
(6) 单目负号	-
(7) 整型常数	1
(8) 右小括号	)
(9) 分号	;

词法分析

```
sum   :=   first   +   count   *   10;
```

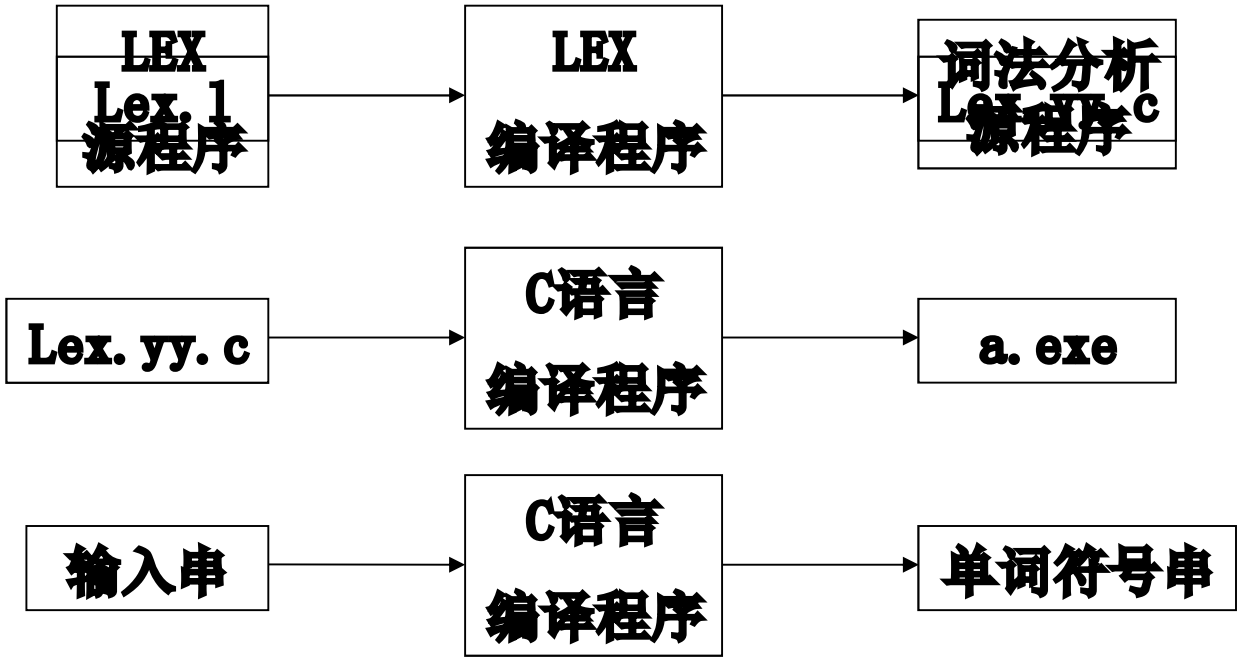
单词类型	单词值
(1) 标识符	sum
(2) 赋值号	: =
(3) 标识符	first
(4) 加号	+
(5) 标识符	count
(6) 乘号	*
(7) 整常数	10
(8) 分号	;

单词符号是由**词法规则**所确定的。词法规定了字母表中哪样的字符串是一个单词符号。经过词法分析后语句为：

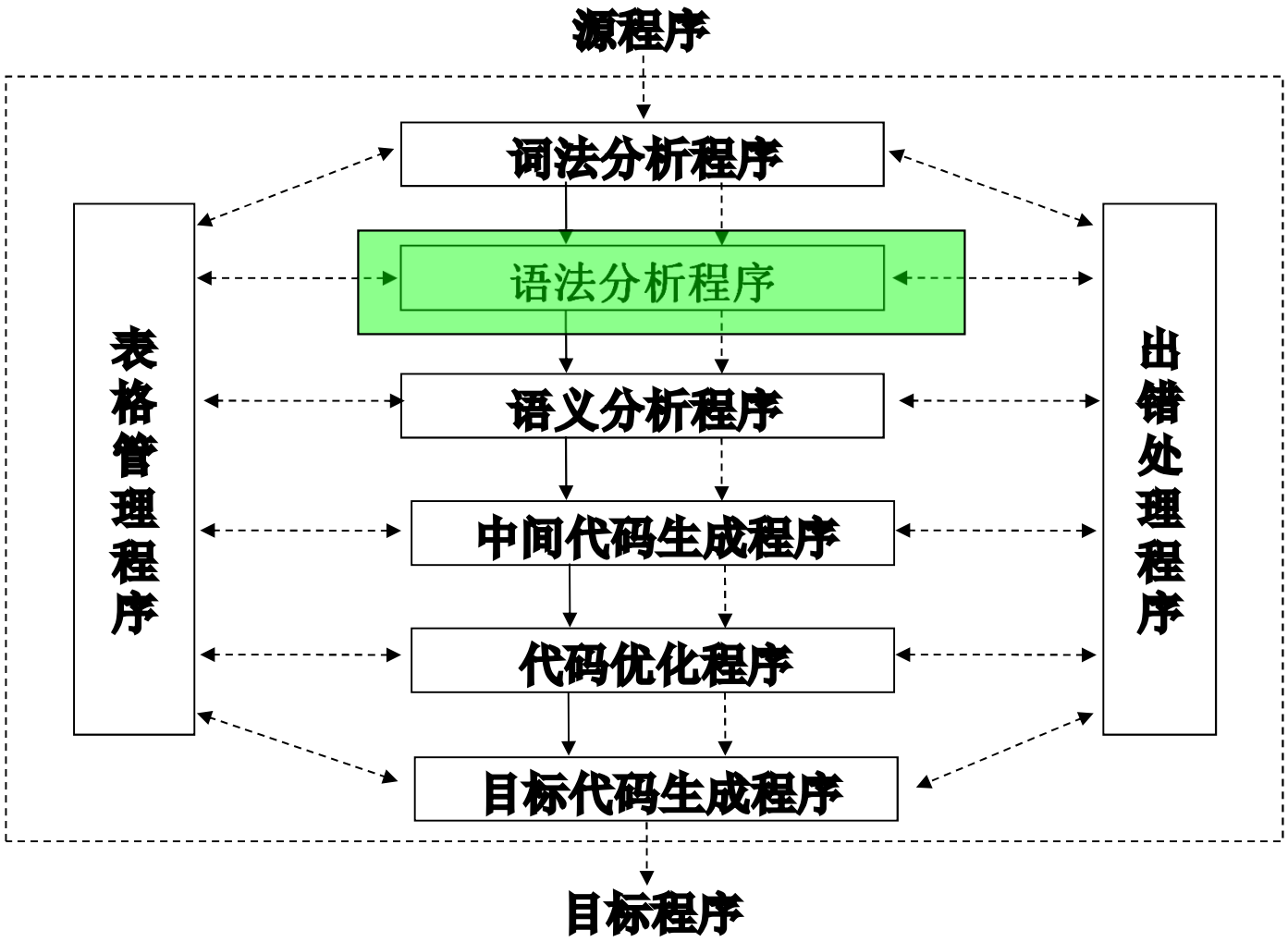
```
id1  :=   id2   +   id3   *   10;
```

词法分析程序的自动构造工具

LEX：windows中FLEX，词法分析程序生成工具，实现词法分析程序的自动生成。



语法分析（Syntax Analysis）



语法分析（Syntax ）

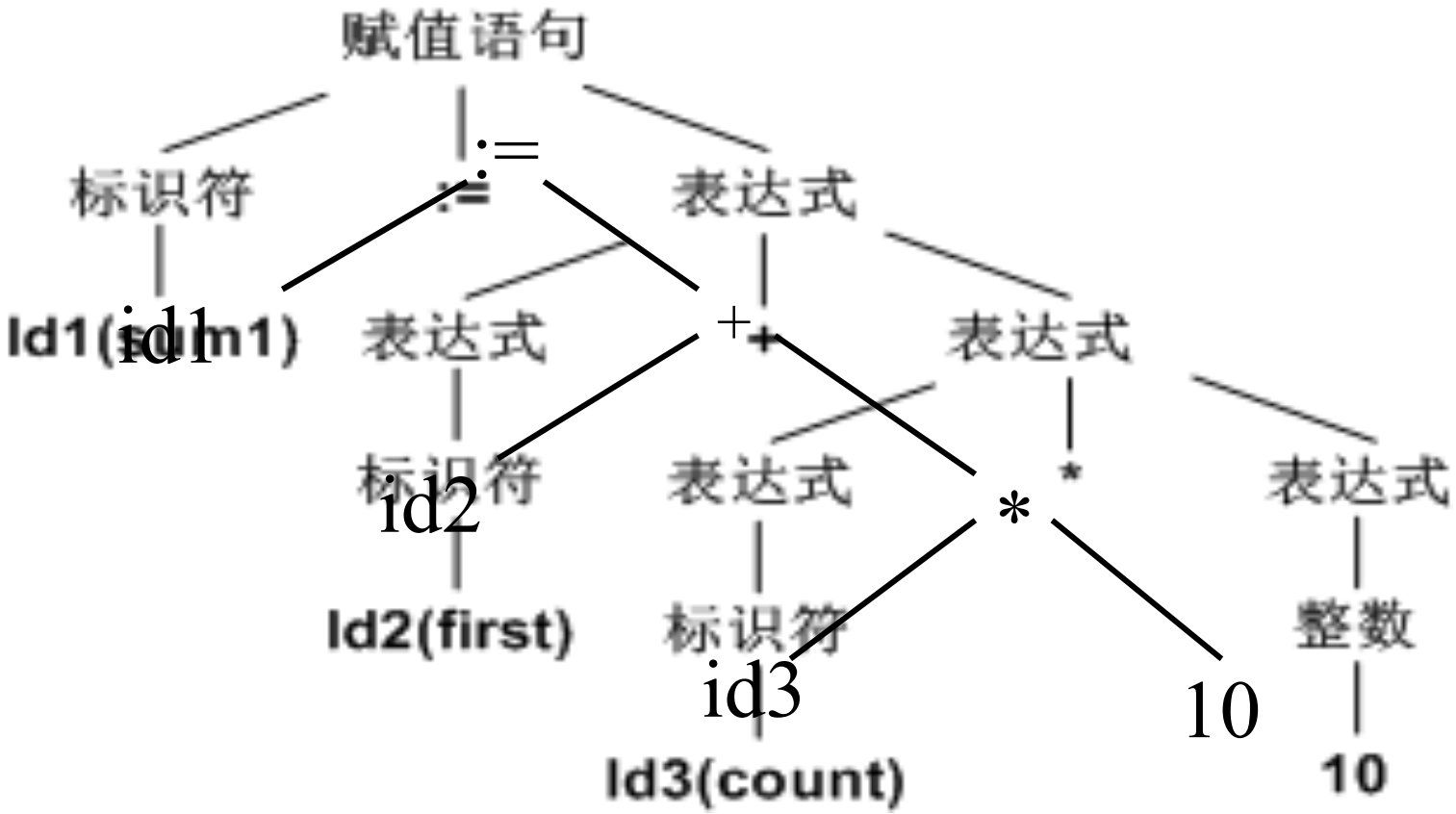
语法分析　Syntax Analysis

功能:层次分析.依据源程序的语法规则把源程序的单词序列组成语法短语(表示成语法树).

- 也称为 “parsing”
- 使用 context-free grammars
- 结构上的合法性Structural validation
  （可生成语法树或推导Creates parse tree or derivation）

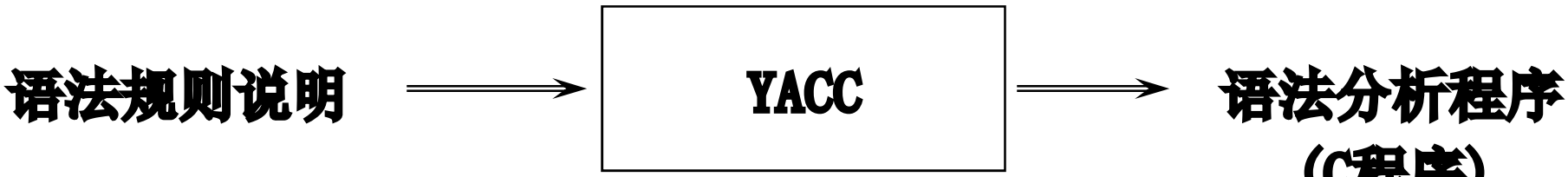
语法分析（Syntax ）

id1 := id2 + id3 * 10;



语法分析算法的语法推导树

语法分析程序的自动生成工具



输入：

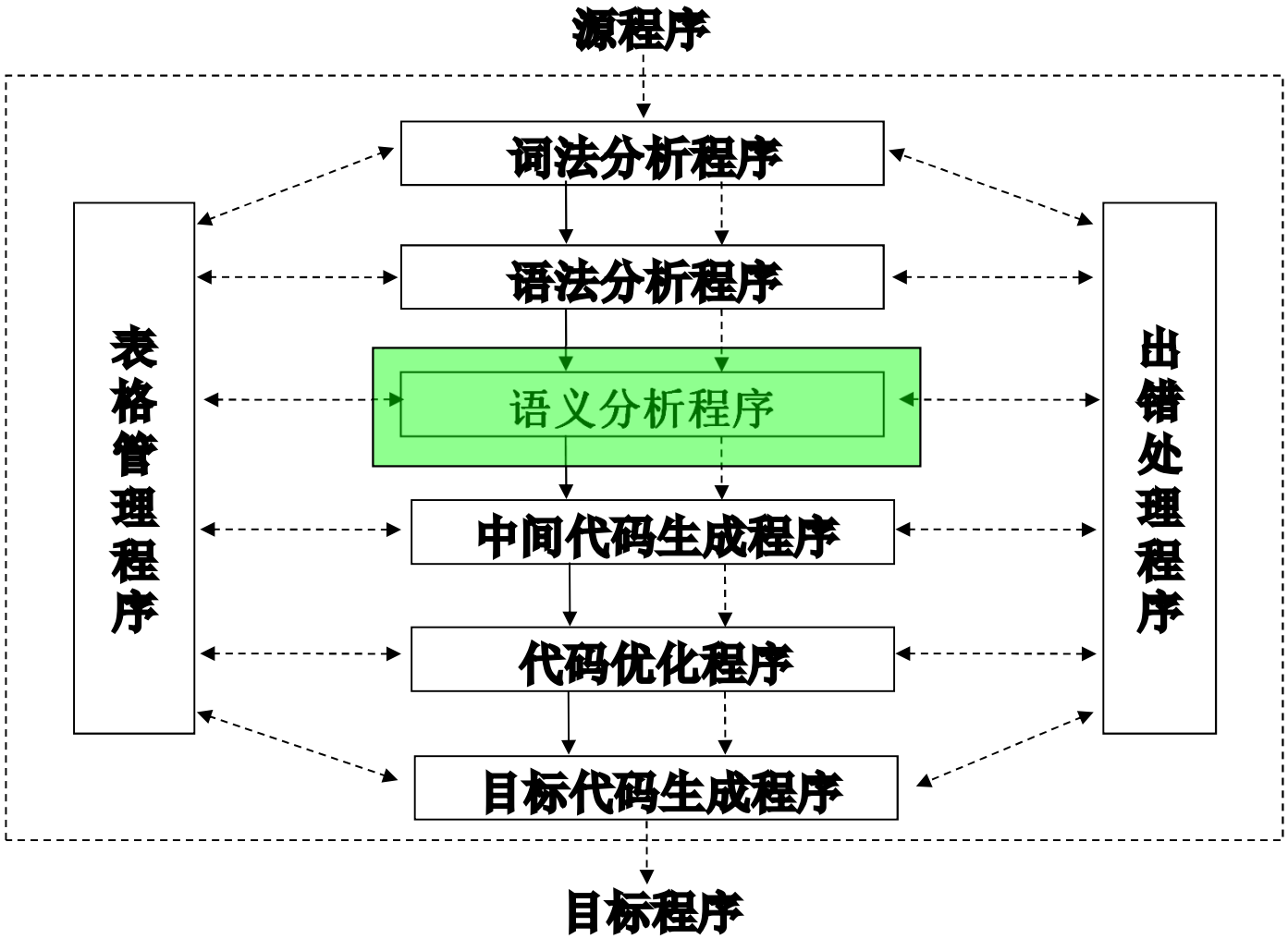
语法规则（产生式）

语义动作（C程序段）

输出：

yyparse( ) 函数

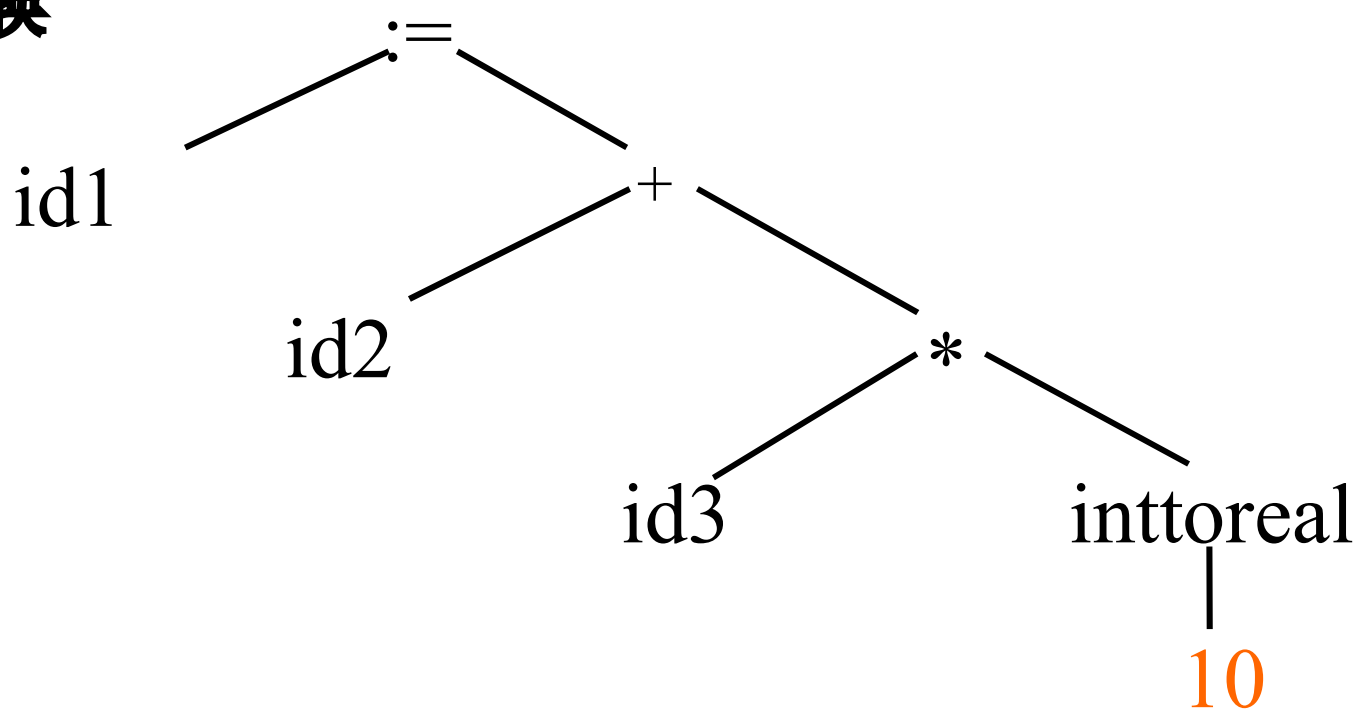
语义分析


源程序
词法分析程序
语法分析程序
语义分析程序
中间代码生成程序
代码优化程序
目标代码生成程序
表格管理程序
出错处理程序
目标程序

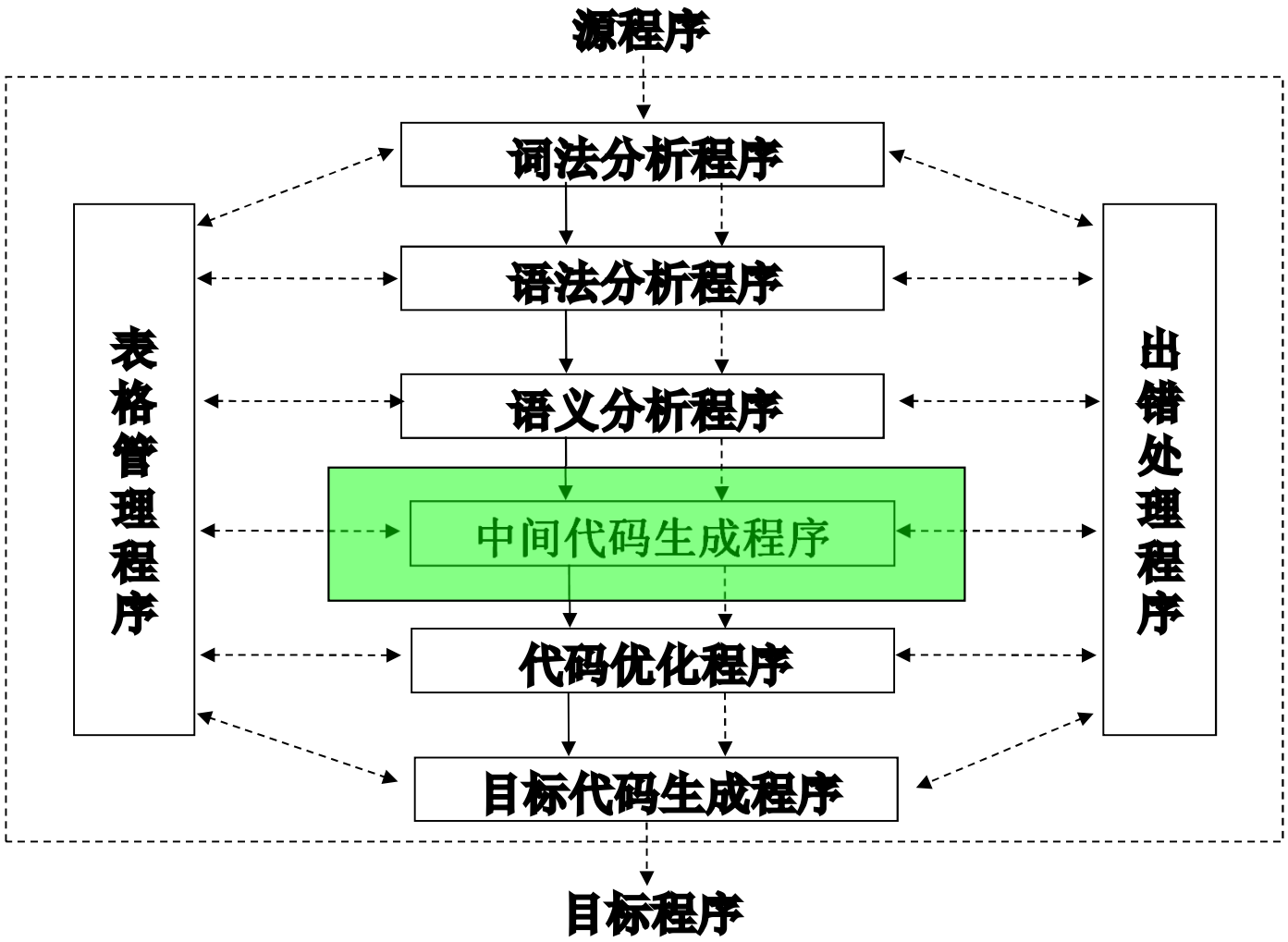
语义分析

语义审查

- **上下文相关性**
- **类型匹配**
- **类型转换**



中间代码生成


源程序
词法分析程序
语法分析程序
语义分析程序
中间代码生成程序
代码优化程序
目标代码生成程序
表格管理程序
出错处理程序
目标程序

中间代码生成

源程序的内部(中间)表示

三元式、四元式、P-Code、C-Code

、

id1:= id2 + id3 * 60

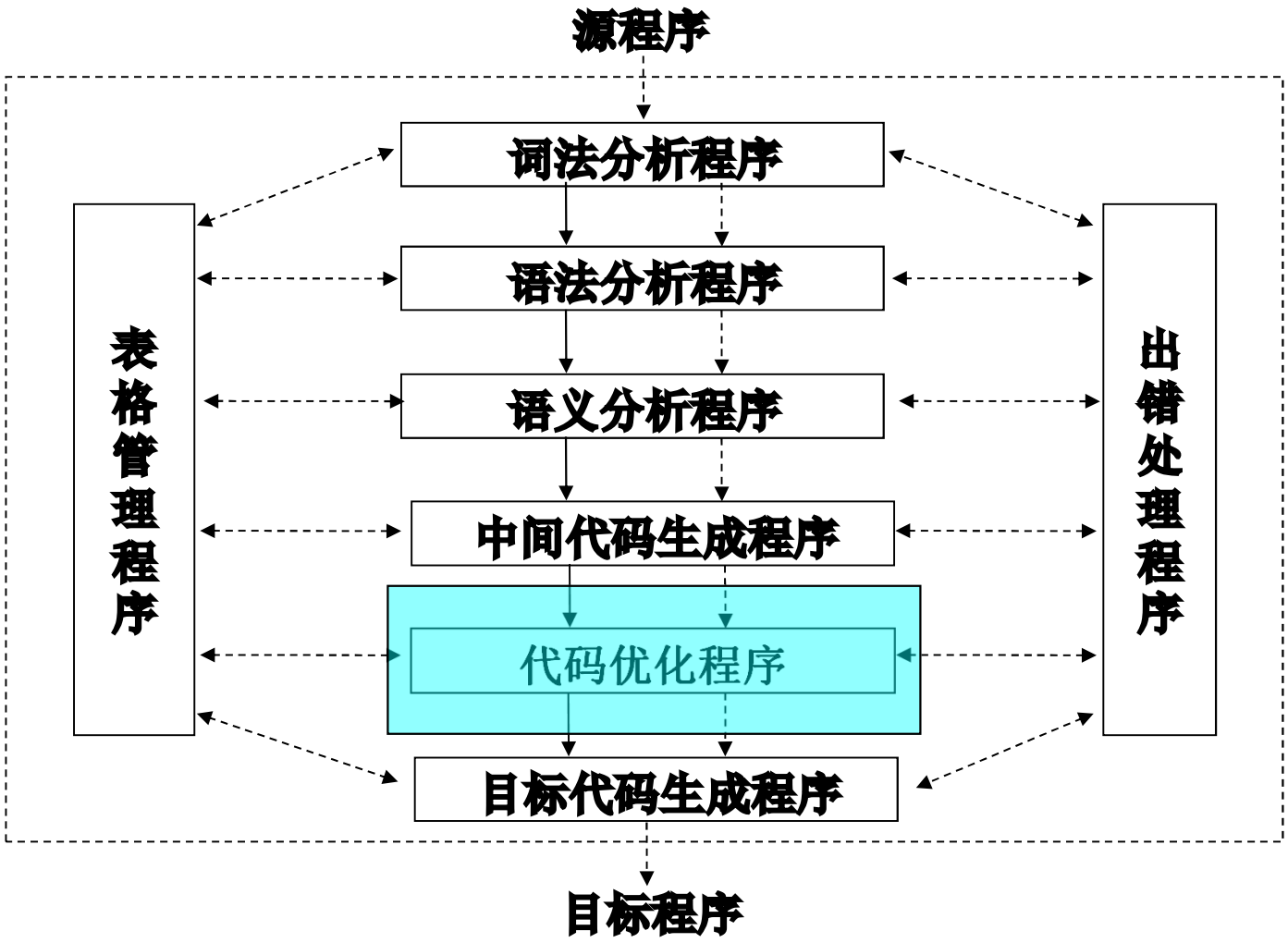
(1) (inttoreal,` 60 - t1)

(2) (* , id3 t1 t2)

(3) (+ , id2 t2 t3)

(4) (:= , t3 - id1)

代码优化


源程序
词法分析程序
语法分析程序
语义分析程序
中间代码生成程序
代码优化程序
目标代码生成程序
表格管理程序
出错处理程序
目标程序

代码优化

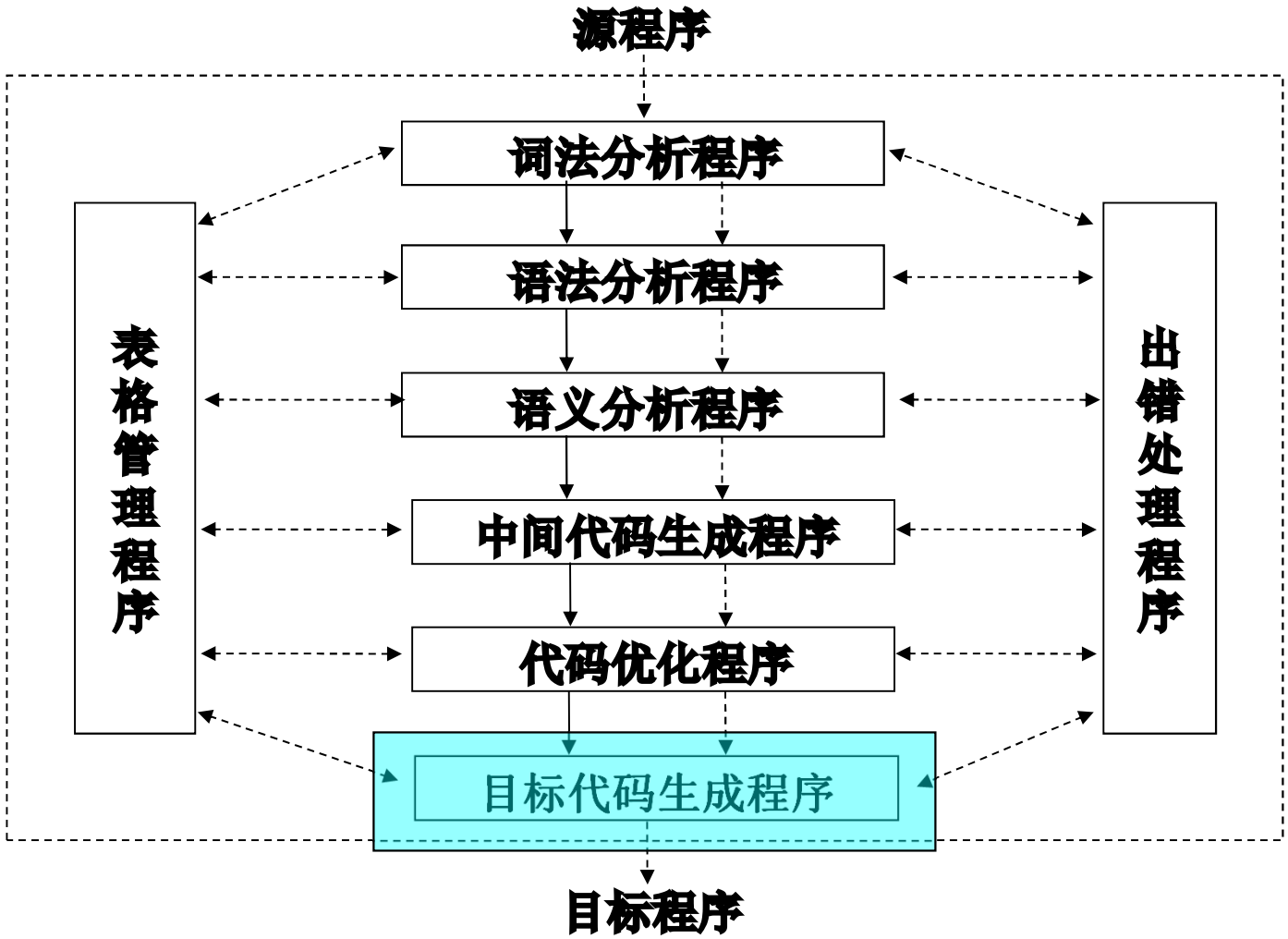
id1:= id2 + id3 * 60

(1)	(inttoreal	60	-	t1)
(2)	**(* **	**id3**	**t1**	**t2)**
(3)	**(+**	**id2**	**t2**	**t3)**
(4)	(:=	t3	-	id1)

变换 ⇒

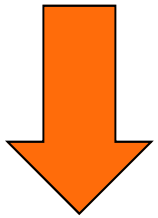
(1)	**(***	**id3**	**60.0**	**t1)**
(2)	**(+**	**id2**	**t1**	**id1)**

目标代码生成


源程序
词法分析程序
语法分析程序
语义分析程序
中间代码生成程序
代码优化程序
目标代码生成程序
表格管理程序
出错处理程序
目标程序

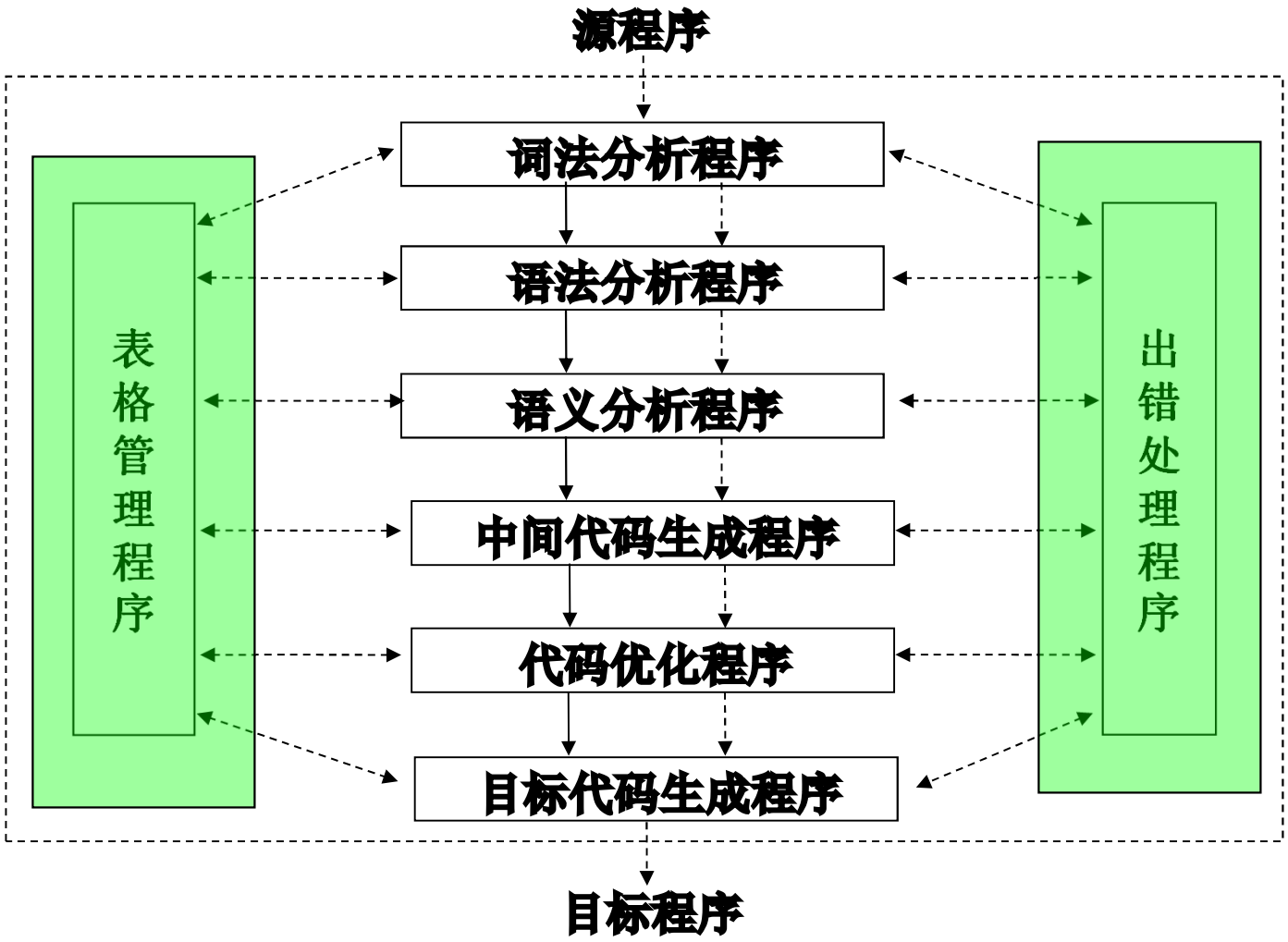
目标代码生成

```
(*     id3    60.0   t1     )
(+     id2    t1     id1    )
```



```
MOV   id3,     R2
MUL   #10.0,   R2
MOV   id2,     R1
ADD   R2,      R1
MOV   R1,      id1
```

表格管理与出错处理


源程序
词法分析程序
语法分析程序
语义分析程序
中间代码生成程序
代码优化程序
目标代码生成程序
表格管理程序
出错处理程序
目标程序

符号表管理与错误处理

符号表管理

- 记录源程序中使用的各种符号名称
- 收集每个符号的各种名的属性信息
  - 类型、作用域、分配存储信息
- 符号表管理()
  - 登录：扫描到说明语句就将标识符登记在符号表中
  - 查找：在执行语句查找标识符的属性，判断语义是否正确

错误检查

- 报告出错信息
- 排错
- 恢复编译工作

1.3 解释程序和一些软件工具

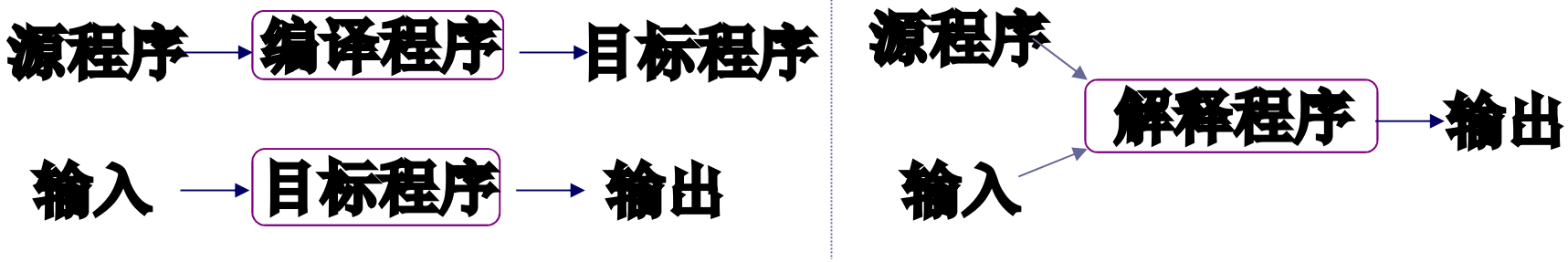
在翻译的方式上，编译程序采用了类似于自然语言翻译中使用的笔译和口译两种，分别叫做**编译方式**和**解释方式**。采用编译方式的编译程序称为编译型的编译程序，简称编译程序；采用解释方式的编译程序称为解释型的编译程序，简称解释程序。

编译方式是先翻译后执行，即将整个源程序翻译完毕，再执行目标程序，只需要保存完整的目标程序而无需保存源程序。一次翻译后无需再翻译，可多次执行。

解释方式是边翻译边执行，即翻译一句就执行一句，翻译完毕也执行完毕，只保存源程序无需保存完整的目标程序。执行一次需要翻译一次。

1.3.1解释系统

✧ 比较编译程序和解释程序

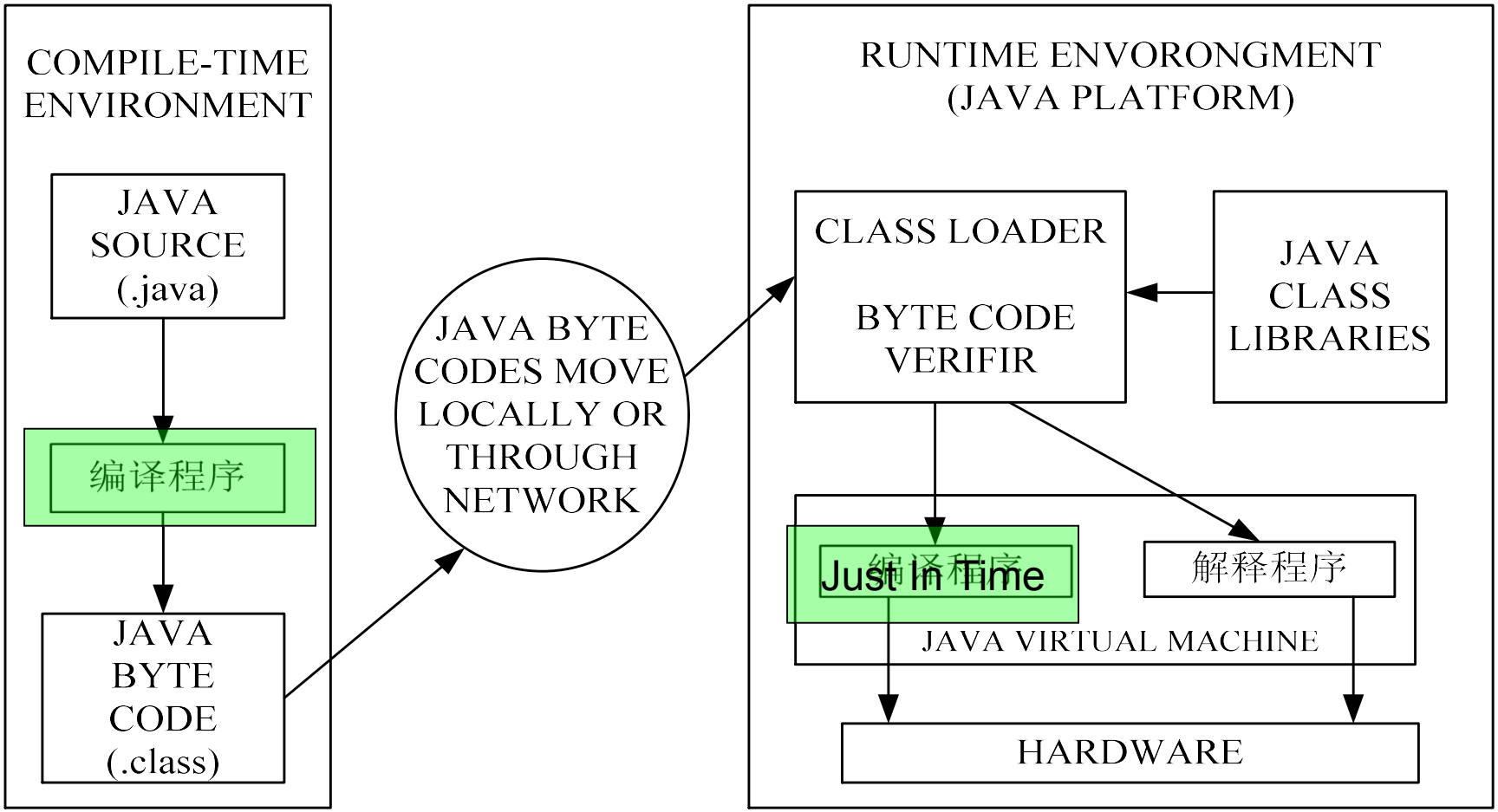


解释程序

- 不产生目标程序文件
- 不区别翻译阶段和执行阶段
- 翻译源程序的每条语句后直接执行
- 程序执行期间一直有解释程序守候
- 常用于实现虚拟机

.著名的解释程序有Basic语言解释程序 ,Lisp语言解释程序,UNIX命令语言解释程序(shell),数据库查询语言SQL解释程序以及bytecode解释程序.

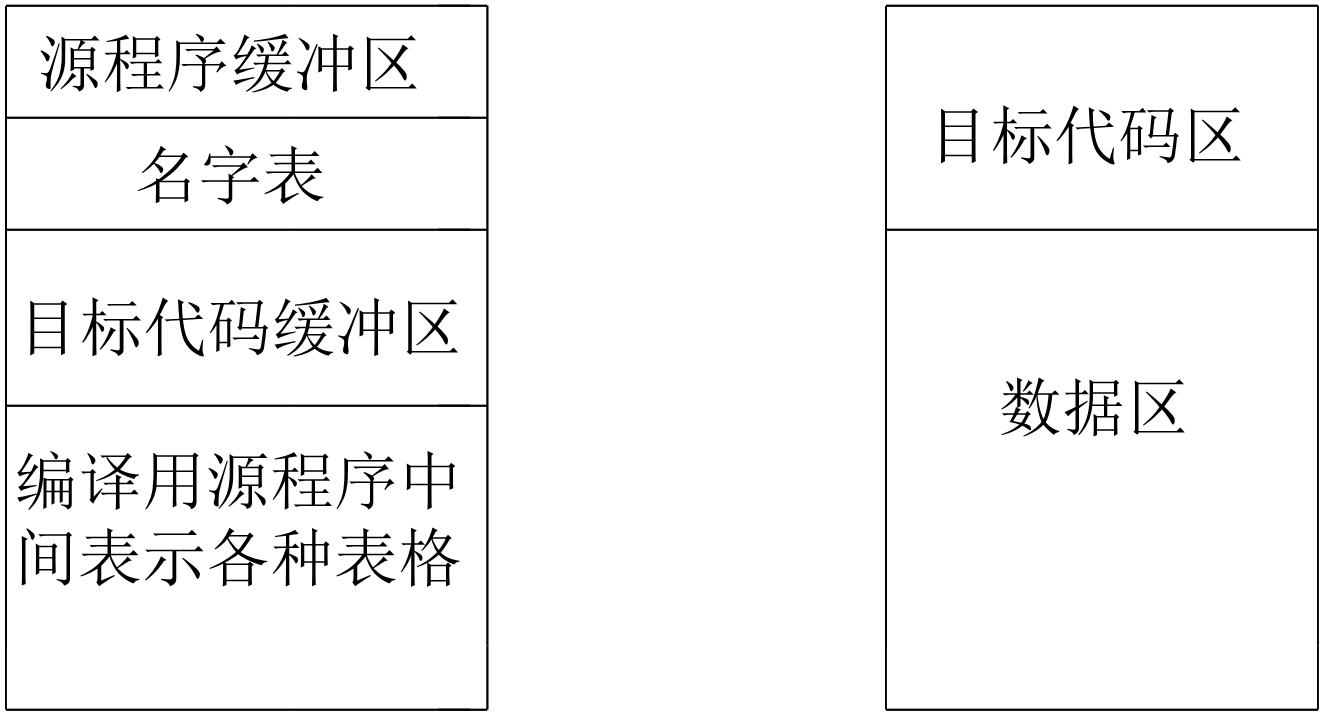
解释程序


COMPILE-TIME ENVIRONMENT
JAVA SOURCE (.java)
编译程序
JAVA BYTE CODE (.class)
JAVA BYTE CODES MOVE LOCALLY OR THROUGH NETWORK
RUNTIME ENVORONGMENT (JAVA PLATFORM)
CLASS LOADER
BYTE CODE VERIFIR
JAVA CLASS LIBRARIES
编译程序
Just In Time
解释程序
JAVA VIRTUAL MACHINE
HARDWARE

编译程序和解释程序的存储组织有很大不同

- **编译程序**处理时，在源语言程序被编译阶段，存储区中要为源程序（中间形式）和目标代码开辟空间，要存放编译用的各种各样表格，比如符号表. 在目标代码运行阶段，存储区中主要是目标代码和数据，编译所用的任何信息都不再需要.
- **解释程序**一般是把源程序一条语句一条语句的进行语法分析，转换为一种**内部表示形式**，存放在源程序区，比如BASIC解释程序，将LET和GOTO这样的关键字表示为一个字节的操作码，标识符用其在符号表的入口位置表示. 因为解释程序允许在执行用户程序时修改用户程序，这就要求源程序，符号表等内容始终存放在存储区中，并且存放格式要设计的易于使用和修改.

编译阶段和运行阶段存储结构



解释系统存储结构

源程序, 符号表等内容始终存放在存储区中, 并且存放格式要设计的易于使用和修改.

1.3.2 处理源程序的软件工具

1. **语言的结构化编辑器**
2. **语言程序的调试工具**
3. **程序格式化工具**
4. **语言程序测试工具**
5. **程序理解工具**
6. **高级语言之间的转换工具**

1.3.3 编译技术的发展

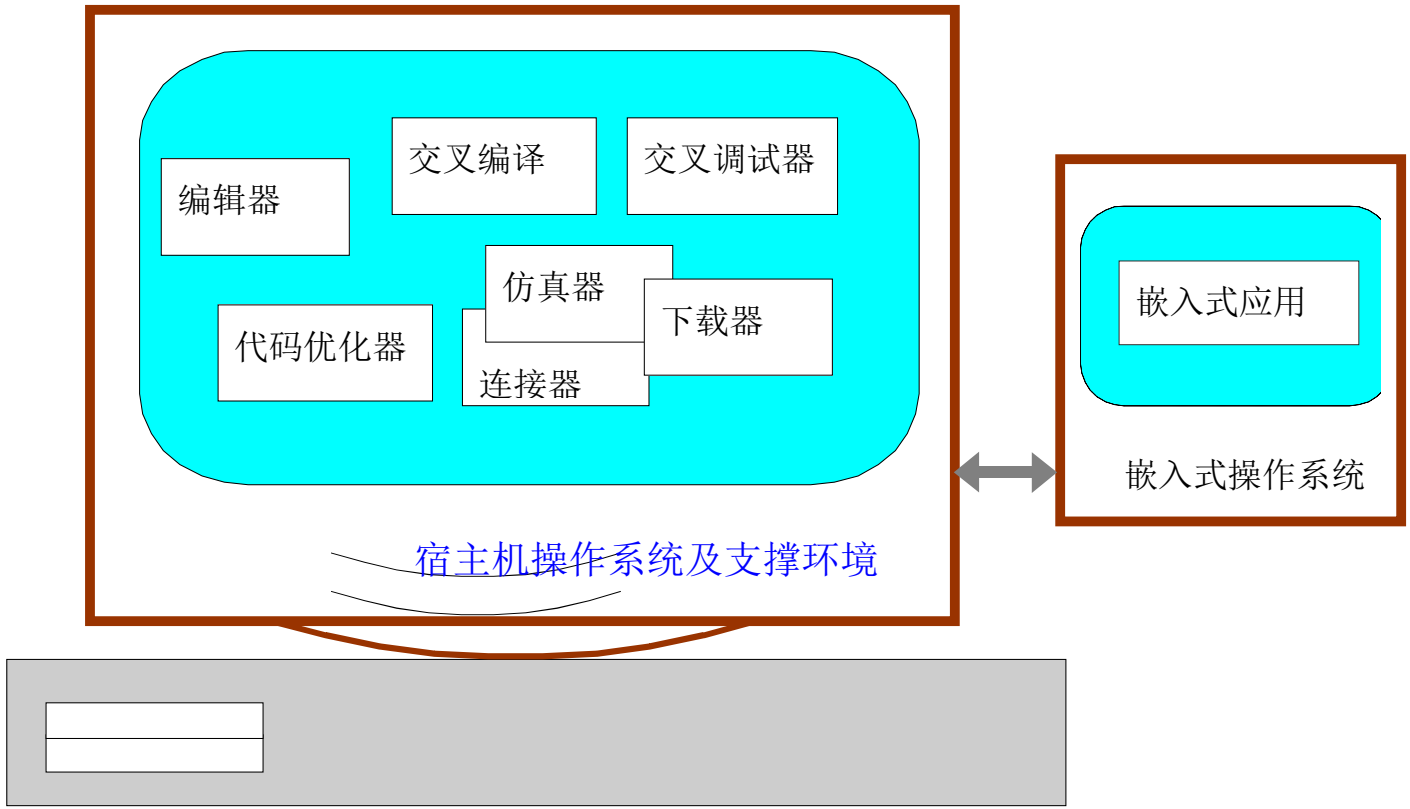
编译程序的实现方式

- 手工
  - 机器语言
  - 汇编
  - 系统程序设计语言
- 自展，交叉编译
- 自动构造工具，如 lex　yacc
- 编译基础设施（多源语言多目标机体系结构的编译程序构造和编译技术研究平台）

重要方向：

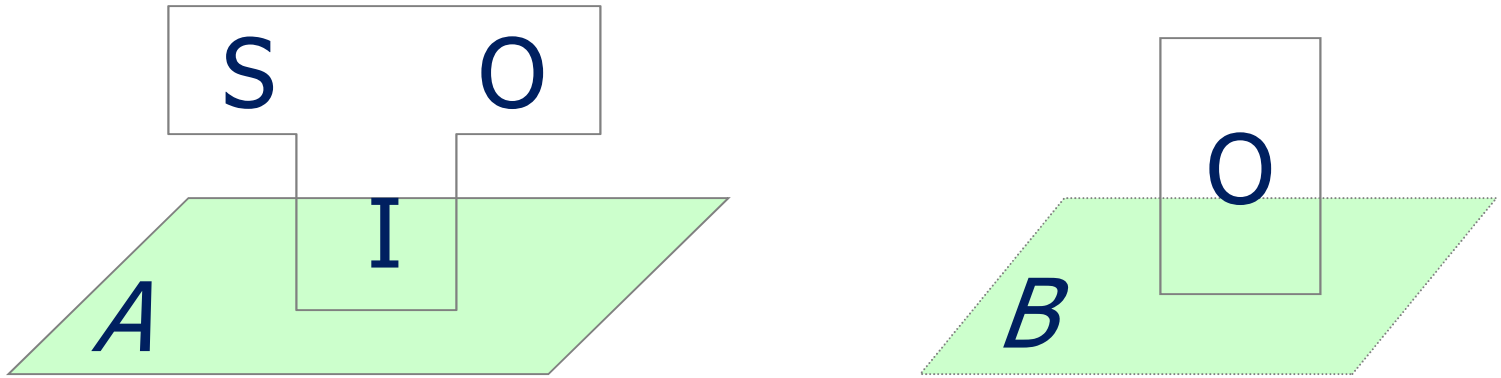
- 并行编译技术－　面向高性能计算
- 交叉编译技术－　面向嵌入式计算

嵌入式系统开发环境


编辑器
交叉编译
交叉调试器
仿真器
下载器
代码优化器
连接器
宿主机操作系统及支撑环境
嵌入式应用
嵌入式操作系统

交叉编译程序

由于目标机指令系统与宿主机的指令系统不同，编译程序在宿主机A上运行，将应用程序的源程序生成目标机B的代码，这种编译称为交叉编译。



一些编译基础设施

- 编译基础设施（Compiler Infrastructure）
  - NCI (National Compiler Infrastructure) project
    - SUIF (Stanford University)
    - Zephyr (Virginia University and Princeton University)
  - Trimaran compiler infrastructure
    - IMPACT (UIUC)
    - CAR (Hewlett Packard Laboratories)
    - ReaCT-ILP (NYU and GIT)
  - GCC
    - GNU project , everyone can get and maintain freely

1.4 PL/0语言编译系统

1.4.1 PL/0语言

图 2.1(a)　程序语法描述图

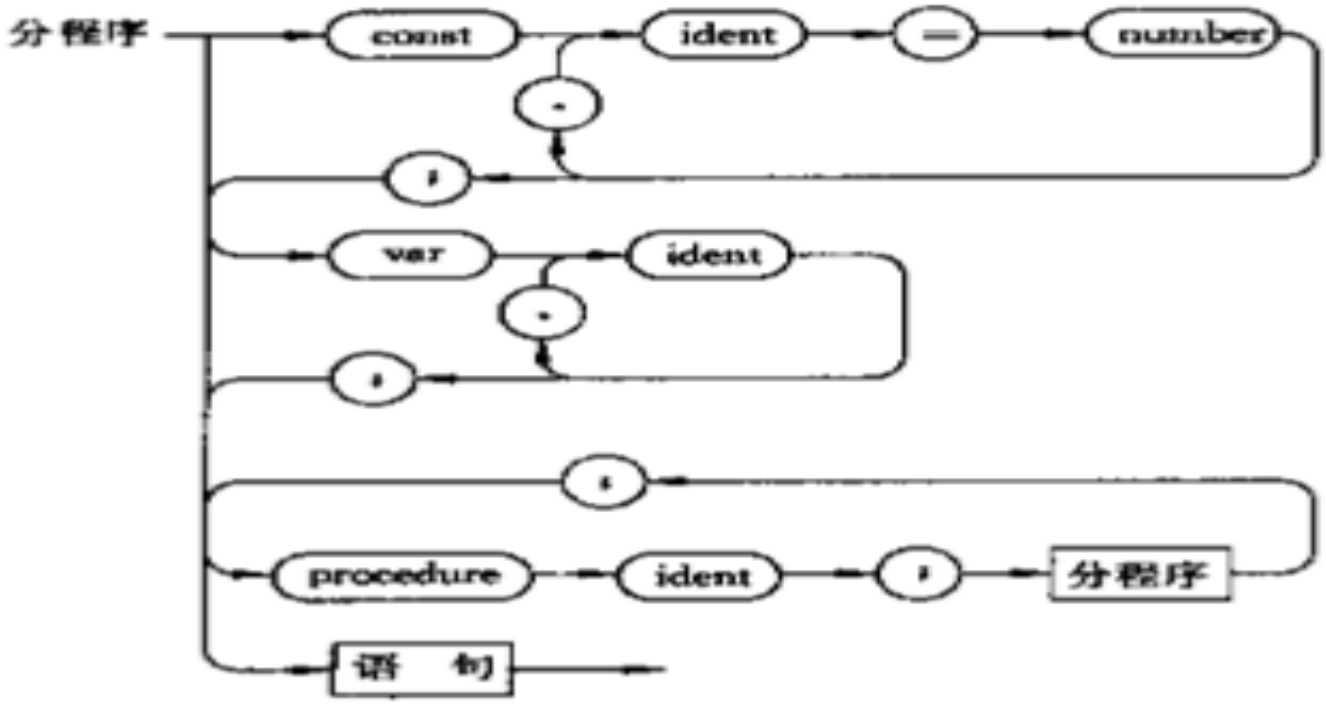


图 2.1(b)　分程序语法描述图

1.4 .1 PL/0语言

程序示例：

```
const   a=10;
var     b, c;
procedure  p;
      const  a=20；
      var c;
      begin
                  c:=5;
                  b:=a*10+c;
      end;
begin
   read(c);
   call p;
   write(b);
   write(c);
end.
```

1.4.2 PL/0语言编译系统构成

PL/0的编译程序采用一趟扫描方式，将其源程序翻译成的目标程序为假想栈式计算机的汇编语言，该汇编语言与计算机无关，再提供汇编语言的解释程序，解释执行目标程序。

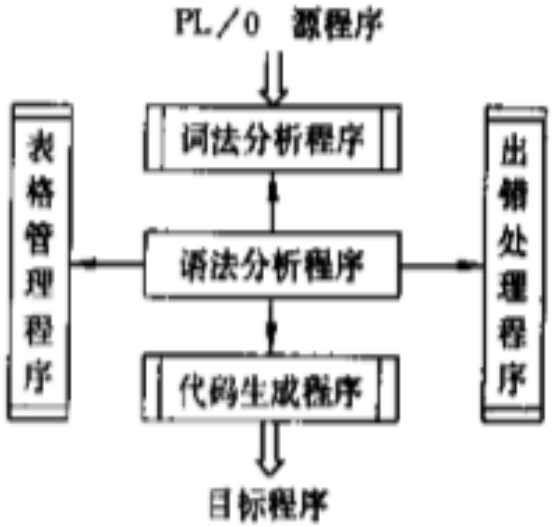


图 2.2(a) PL/0 编译程序的结构图

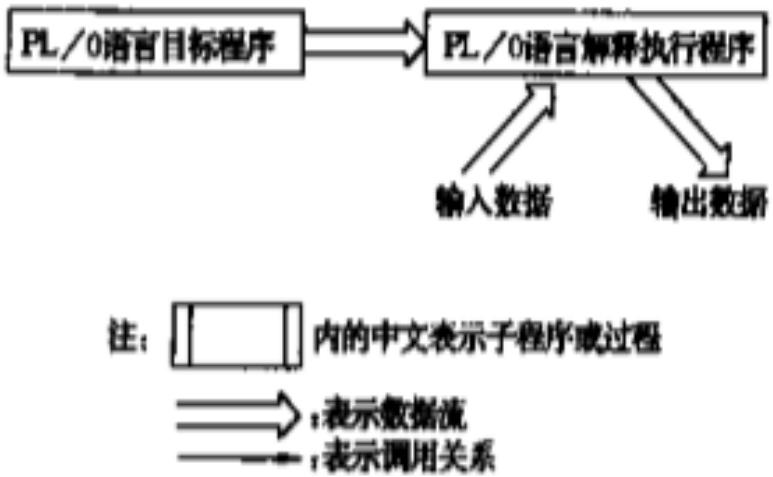


图 2.2(b) PL/0 的解释执行结构

1.4.2 PL/0语言编译系统

语法语义分析BLOCK是整个编译程序的核心。

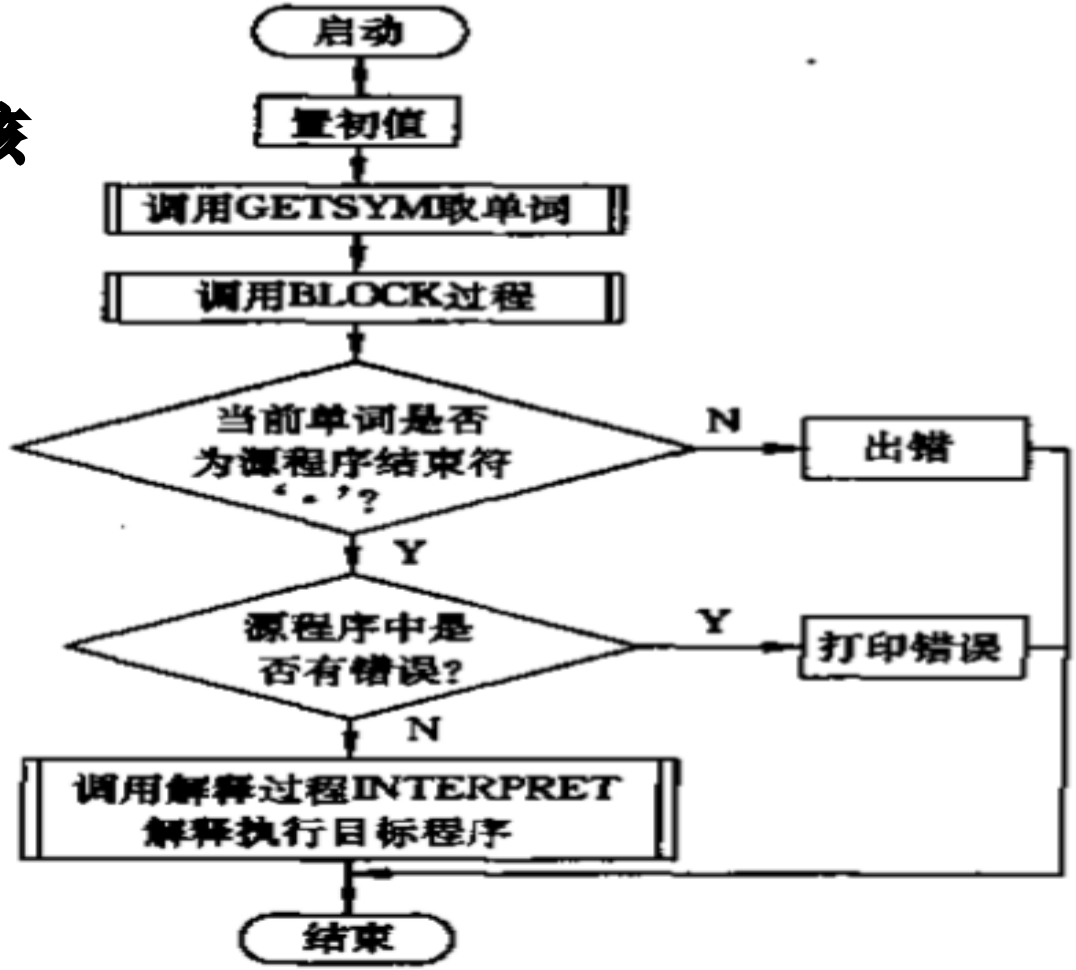


图 2-4 PL/0 编译程序总体流程图

1.4 PL/0语言词法分析

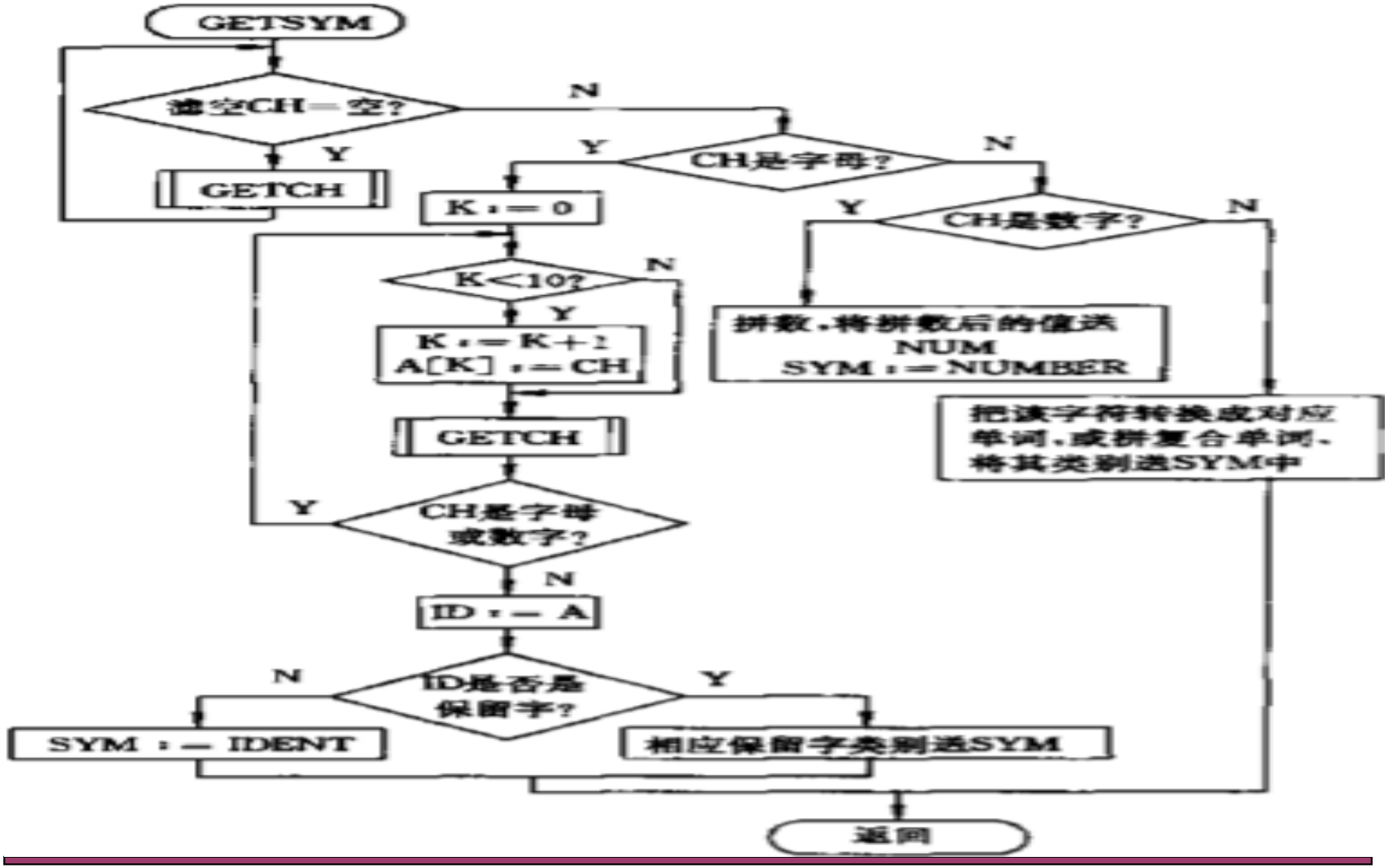
GETSYM：词法分析程序是一个独立的过程，每次调用能获取一个单词，通过3个全程量变量传递结果：

SYM	存放单词的种类；
ID	存放用户定义的标识符的值；
NUM	存放用户定义的常数。

单词种类：

基本字、运算符、标识符、常数，界符。

1.4 PL/0语言词法分析



1.4 PL/0语言语法分析

采用的是递归下降的语法分析（第5章介绍）。对应每一个非终结符定义一个子程序。

（1）说明部分的处理

使用一个结构（记录）数组TABLE记录每个常量，变量和过程的信息。规定主程序的层号为第0层，主程序中定义的过程的层号为第1层。PL/0最多允许3层。

TABLE中记录各常量和变量的属性、层号以及在对应过程中的分配单元的相对地址；

对过程的地址部分，当分析到过程的语句时，将语句的起始地址回填。

TX为索引表的指针。

1.4 PL/0语言语法分析

说明语句程序片段：

```
CONST   A =35, B=49;
  VAR       C, D, E;
 PROCEDURE    P;
          VAR G;
```

TX_0→	NAME:A	KIND:CONSTANT	VAL:35		
	NAME:B	KIND:CONSTANT	VAL:49		
	NAME:C	KIND:VARIABLE	LEVEL:LEV	ADR:DX	
	NAME:D	KIND:VARIABLE	LEVEL:LEV	ADR:DX+1	
	NAME:E	KIND:VARIABLE	LEVEL:LEV	ADR:DX+2	
	NAME:P	KIND:PROCEDUR	LEVEL:LEV	ADR:	SIZE:4
TX→	NAME:G	KIND:VARIABLE	LEVEL:LEV+1	ADR:DX	
	⋮	⋮	⋮	⋮	

变量等的地址为相对于过程数据区的起始位置的偏移量，初值 DX:=3

1.4 PL/0语言语法分析

```
const   a=10;
var     b, c;
procedure  p;
      const  a=20;
      var c;
      begin
         c:=5;
         b:=a*10+c;
      end;
begin
   read(c);
   call p;
   write(b);
   write(c);
end.
```

LEV: 0

pTX

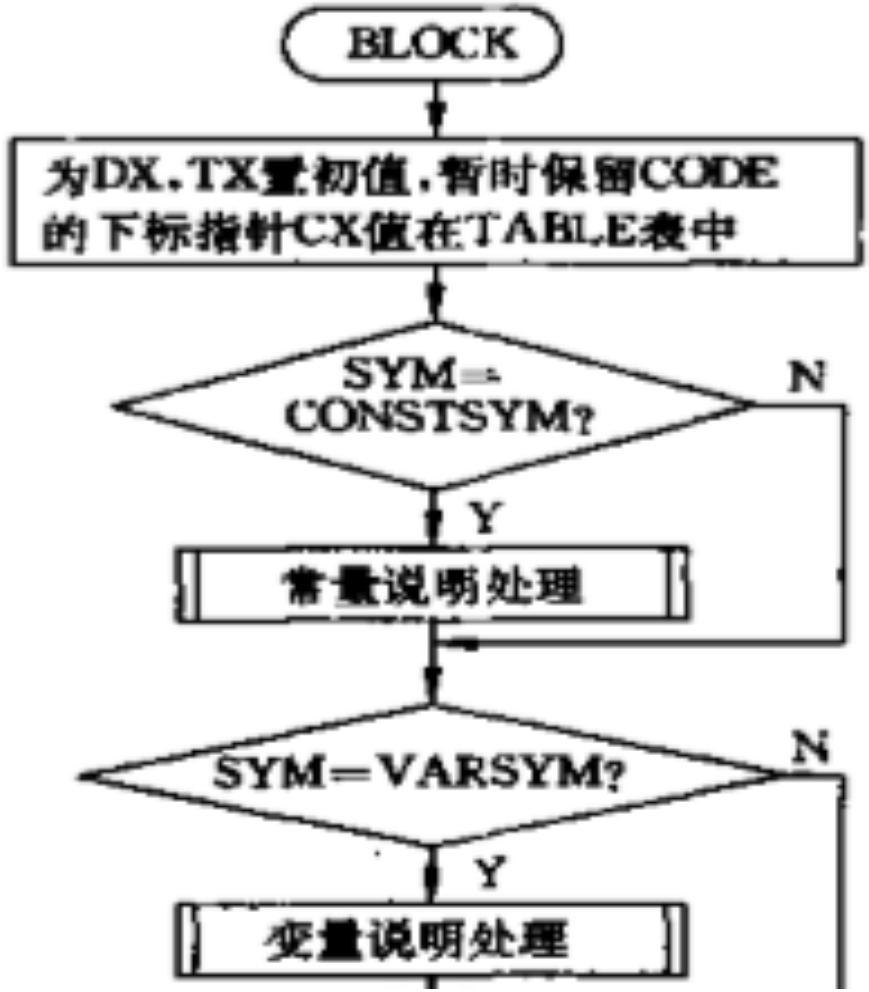
TX1

name	KIND	V/L	ADR	SIZE
a	constant	10		
b	VARIABLE	0	DX	
c	VARIABLE	0	DX+1	
p	PROCEDU	0	P的入口	4
a	constant	20		
c	VARIABLE	1	DX	

1.4 PL/0语言语法分析

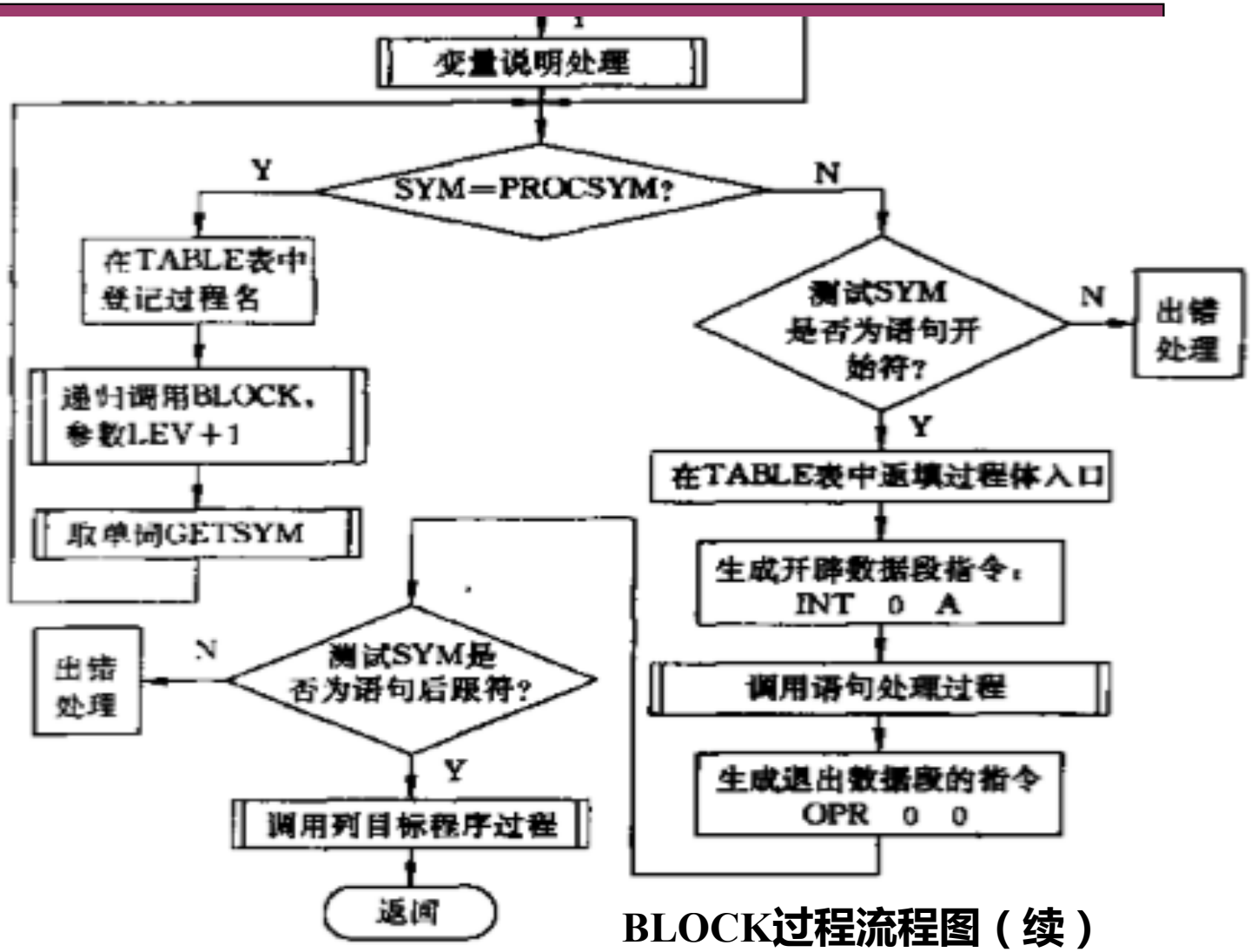
（2）过程体（分程序）的处理

语法正确时，生成目标代码。遇到标识符的引用就查TABLE表，如果有定义，就取出相应信息，无则报错。



BLOCK过程流程图

1.4 PL/0语言语法分析



BLOCK过程流程图（续）

目标代码结构和代码生成

目标代码是一个假想栈式计算机的汇编语言，可称为PCODE指令代码。格式为：

f：代表功能码，也可称为操作码；

l：表示层次差，引用层和声明层之间的层次差，否则置0

a：不同的指令含义不一样（详见教材）

代码生成由过程GEN完成，生成的代码放在CODE数组中。

目标代码结构和代码生成

```
const a=10;
var   b,c;
procedue   p;
   begin
        c:=b+a
   end.
```

部分程序的目标代码

```
0  jmp   0   n      (假定n是主程序入口)
1  jmp   0   2      (2是过程p入口)
2  int   3          (在运行栈中为SL、DL、RA开辟3+0个单元)
3  lod   1   3      (取出p的外层的b的值放在栈顶)
4  lit   0   10     (将常量10取到栈顶)
5  opr   0   2      (加运算，结果放在次栈顶)
   6  sto   1   4      (运算结果送p的外层的c的单元中)
      7  opr   0   0      (返回调用点，并释放单元)
```

PL/0编译程序的语法错误处理

采用一些措施，对源程序中的错误尽量查出，不是遇到错误马上就停止。常采用的两种方法：

（1） 对易于校正的错误，如丢失逗号，分号等，指出错误位置并校正。

（2） 对难于校正的错误，就跳过一些单词，知道读入一个符号能使编译程序恢复正常语法分析为止。

目标代码结构和代码生成

解释执行CODE[0…]中的指令序列，数据空间为栈式计算机的存储空间。定义了4个寄存器：

I 指令寄存器，存放当前正在解释的目标指令；

P 程序地址寄存器，下一条要执行的目标指令地址；

T 栈顶寄存器；

B 基地址寄存器。

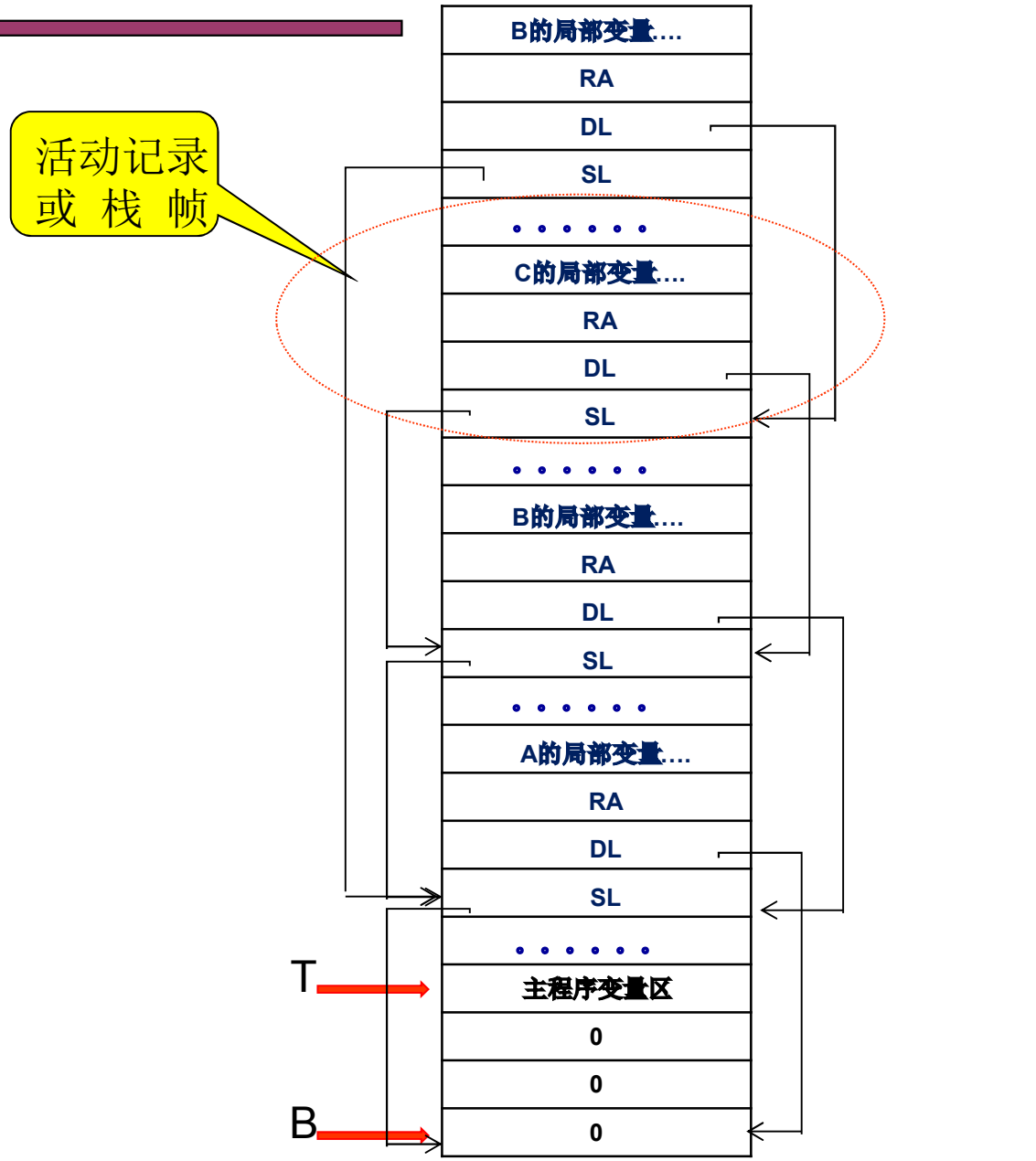
B和T指向的单元间存放的数据有：

静态部分：变量和联系单元SL（静态链指向定义该过程的直接外层最新数据段基地址）、DL（动态链指向调用者基地址）、RA（返回地址））

动态部分：临时单元和累加器用，随时分配和用完释放，

例：若有程序结构为：

```
┌ ⋮
└procedure A；
      ┌ ⋮
      └procedure B；
            ┌ ⋮
            └procedure C；
                  程序体 C
                  ┌ ⋮
                  │ call B；
                  └ ⋮
            程序体 B
            ┌ ⋮
            │ call C；
            └ ⋮
      程序体 A
      ┌ ⋮
      │ call B；
      └ ⋮
主程序
┌ ⋮
│ call A；
└ ⋮
```



65

小结

本章重点介绍了编译程序定义、编译过程、编译方式、编译程序结构和高级程序语言的分类以及相关的应用。

提出的基本概念是源语言、源程序、目标语言、目标程序、程序等价、翻译方式、遍/趟、翻译程序、汇编程序、编译程序和解释程序。

重点掌握的概念：①编译程序；②编译过程；③编译程序结构；④编译程序生成方法。