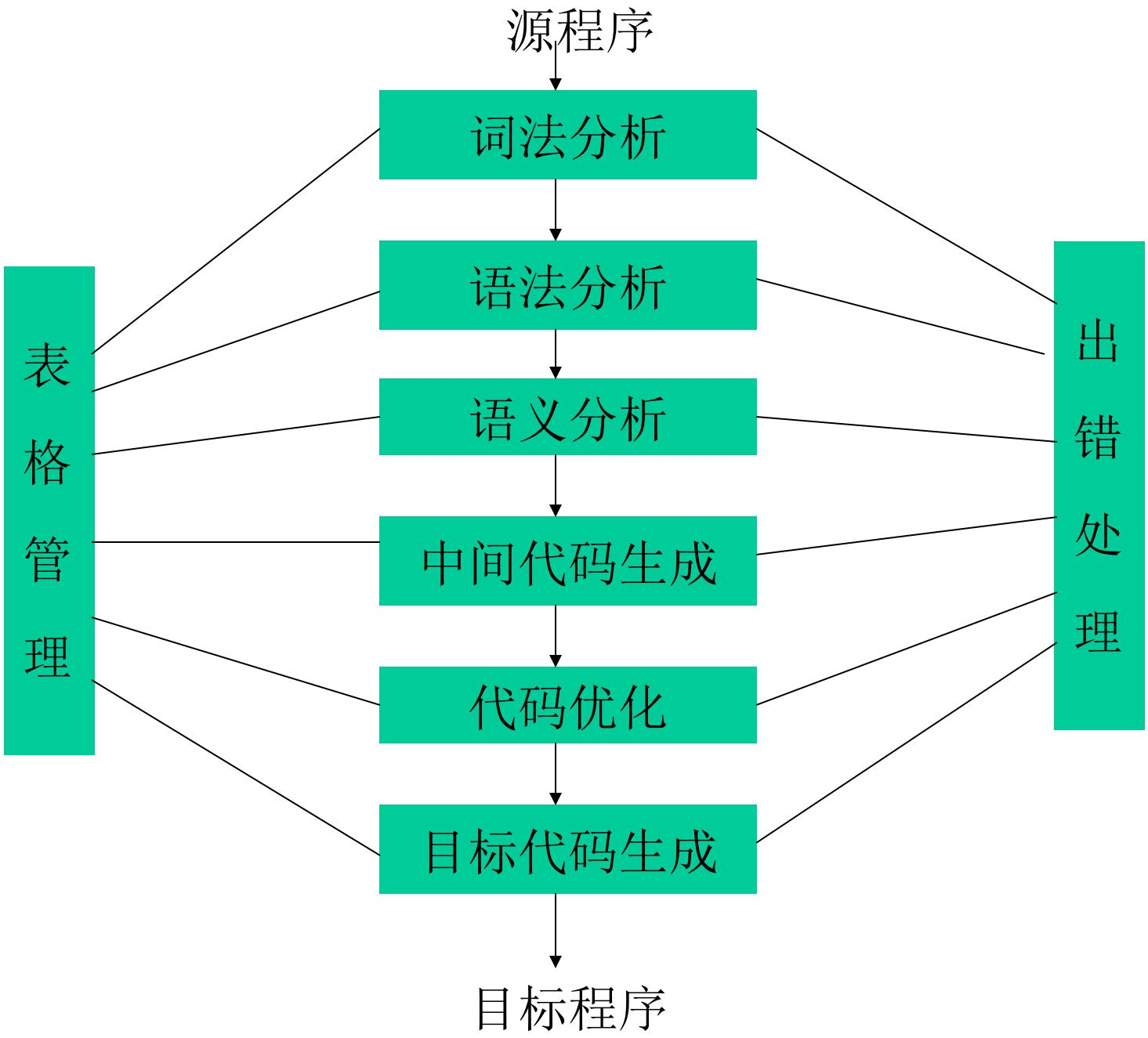

源程序
词法分析
语法分析
语义分析
中间代码生成
代码优化
目标代码生成
目标程序
表格管理
出错处理

第9章　符号表

9.1符号表的作用和地位

9.2符号的主要属性

9.3符号表的组织

符号表的作用和地位----

语义检查的依据;目标代码生成阶段地址分配的依据

- 在编译程序中符号表用来存放语言程序中出现的有关标识符的属性信息，符号表中所登记的信息在编译的不同阶段都要用到。
- 在语义分析中，符号表所登记的内容将用于语义检查（如检查一个名字的使用和原先的说明是否一致）。
- 在目标代码生成阶段，当对符号名进行地址分配时，符号表是地址分配的依据（要根据是什么类型的符号决定分配多大的地址空间）。

在整个编译期间，对于符号表的操作大致可归纳为五类：

- 对给定名字，查询名字是否已在表中；
- 往表中填入一个新的名字；
- 对给定名字，访问它的某些信息；
- 对给定名字，填写或更新它的某些信息；
- 删除一个或一组无用的项。

不同种类的表格所涉及的操作往往也是不同的。上述五个方面只是一些基本的共同操作。

符号属性(信息)

常用信息：

1、 符号名

2、 符号的类型 （整型、实型、字符串型等））

3、 符号的存储类别（公共、私有）

4、 符号的作用域及可视性 （全局、局部）

5、 符号变量的存储分配信息 （静态存储区、动态存储区）

6、 符号的其它属性 （１） 数组内情向量（数组类型、维数、各维的上下界、首地址等）

（２） 记录结构型的成员信息 （３） 函数及过程的形参

符号表的组织

总体组织

第一种： 把属性种类完全相同的那些符号组织在一起，构造出表项是分别为等长的多个符号表

第二种： 把所有语言中的符号都组织在一张符号表中。组成一张包括了所有属性的庞大的符号表

第三种折衷方式是根据符号属性相似程度分类组织成若干张表，每张表中记录的符号都有比较多的相同属性。

符号表项的排列

- 符号表作为一个多元组，表中元组的排列组织是构造符号表的重要成分。在编译程序的整个工作过程中，符号表被频繁地用来建立表项，找查表项，填充和引用表项的属性。因此表项的排列组织对该系统运行的效率起着十分重要的作用。
- 在编译程序中，符号表项的组织传统上采用三种构造方法。即线性法，排序法及散列法。

- 线性组织：按符号在程序中被扫描到的先后顺序组织；
- 排序组织：按符号从大到小或从小到大的顺序排列
- 散列组织：一个符号在散列表中的位置由对该符号进行某种函数（杂凑函数）操作所得到的函数值来决定。

第十章　代码优化

- 什么是代码优化?
- 在中间代码或目标代码生成之后对生成的代码进行优化。
- 所谓优化，就是对代码进行等价变换，使得变换后的代码运行结果与变换前代码运行结果相同，而运行速度加快或占用存储空间减少。

何谓代码优化：

宗旨： 获得较好性能的代码

等价

意图，结果，权衡

阶段：

优化分类

按阶段分:

与机器无关的优化---对中间代码进行
依赖于机器的优化--对目标代码进行

根据优化所涉及的程序范围分成:

(1)局部优化:(基本块)

(2)循环优化:对循环中的代码进行优化

(3)全局优化:整个程序范围内的优化

●优化技术简介

1.删除多余运算

2.循环不变代码外提

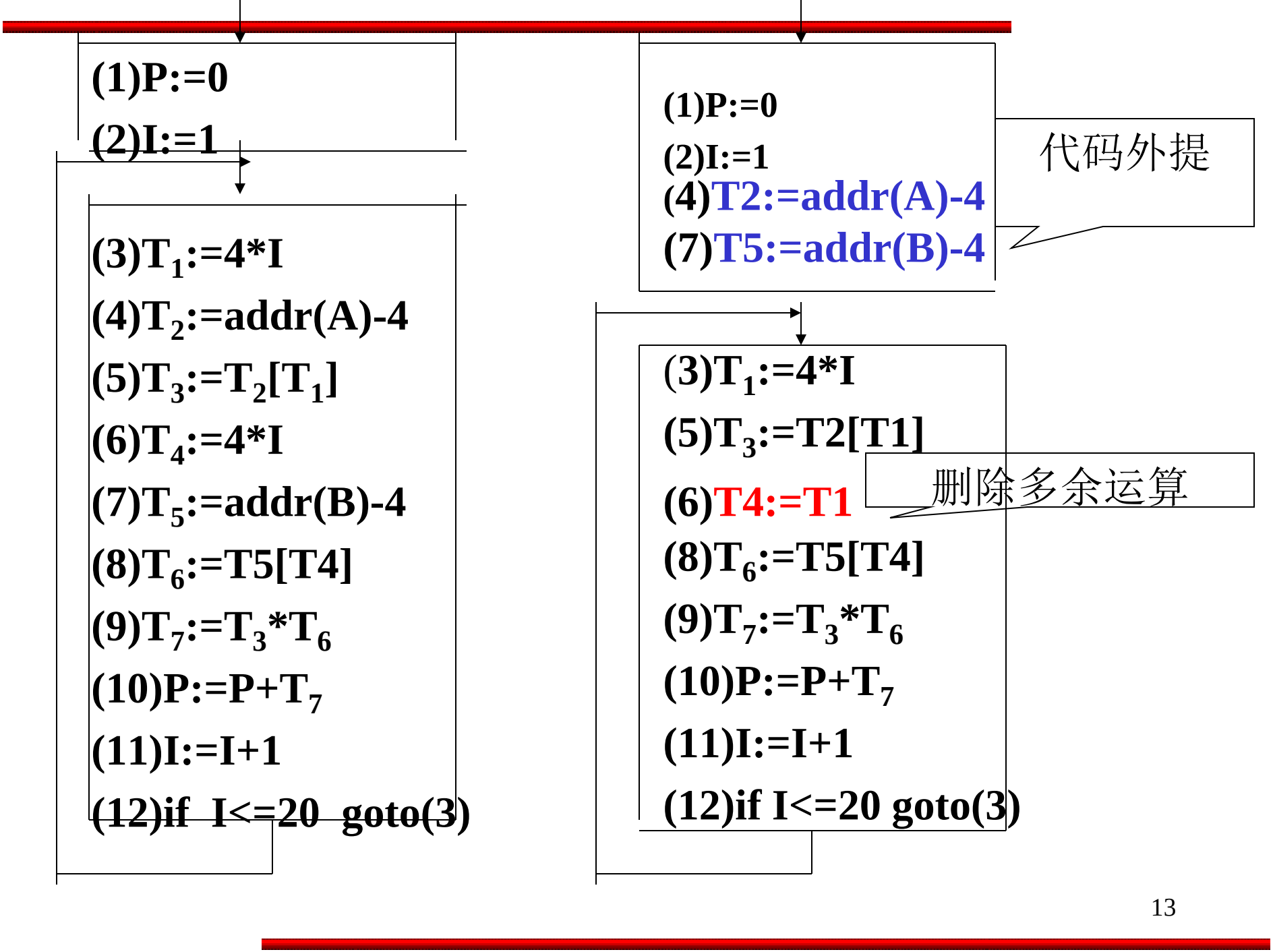
3.强度削弱

4.变换循环控制条件

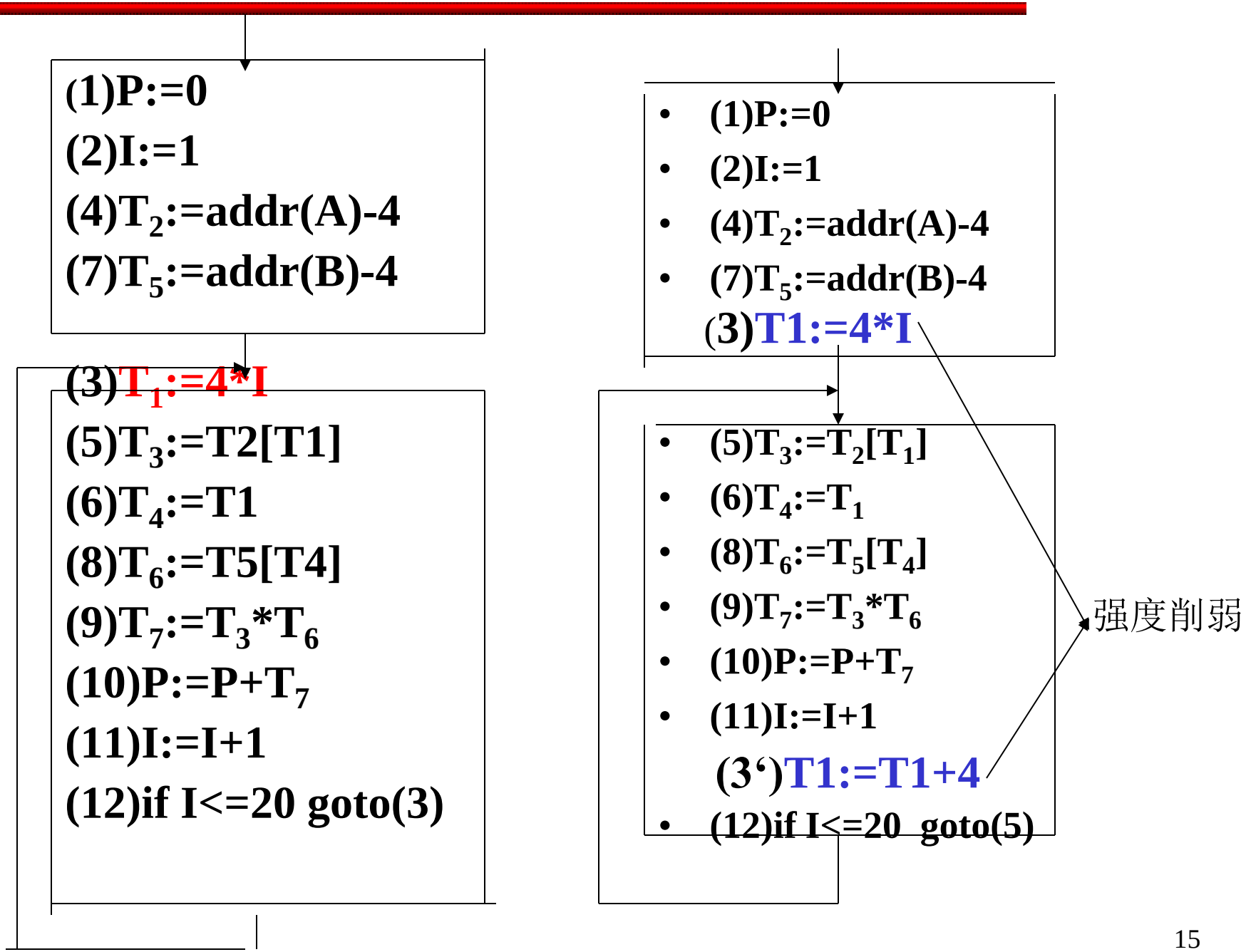
5.合并已知量与复写传播

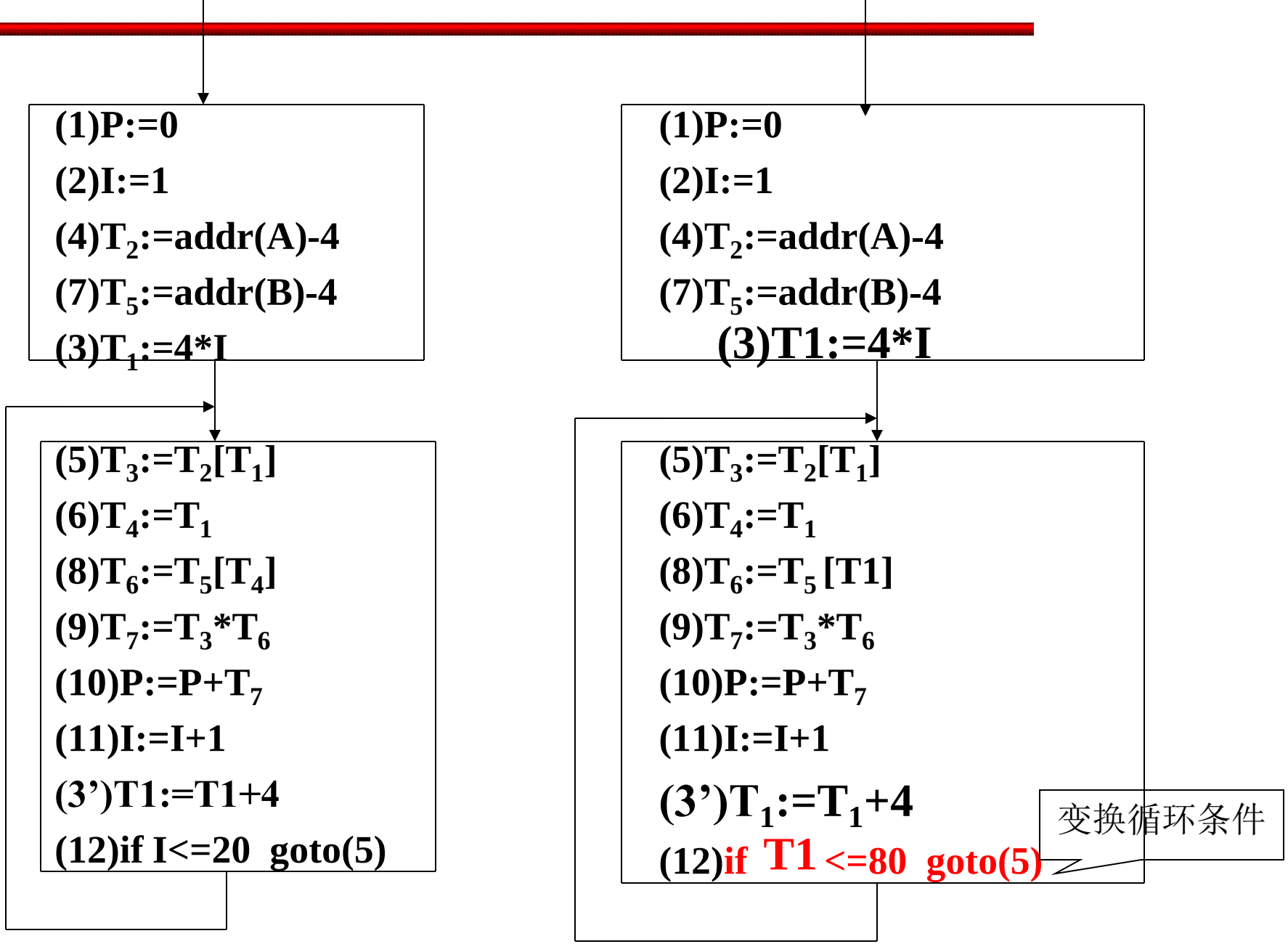
6.删除无用赋值

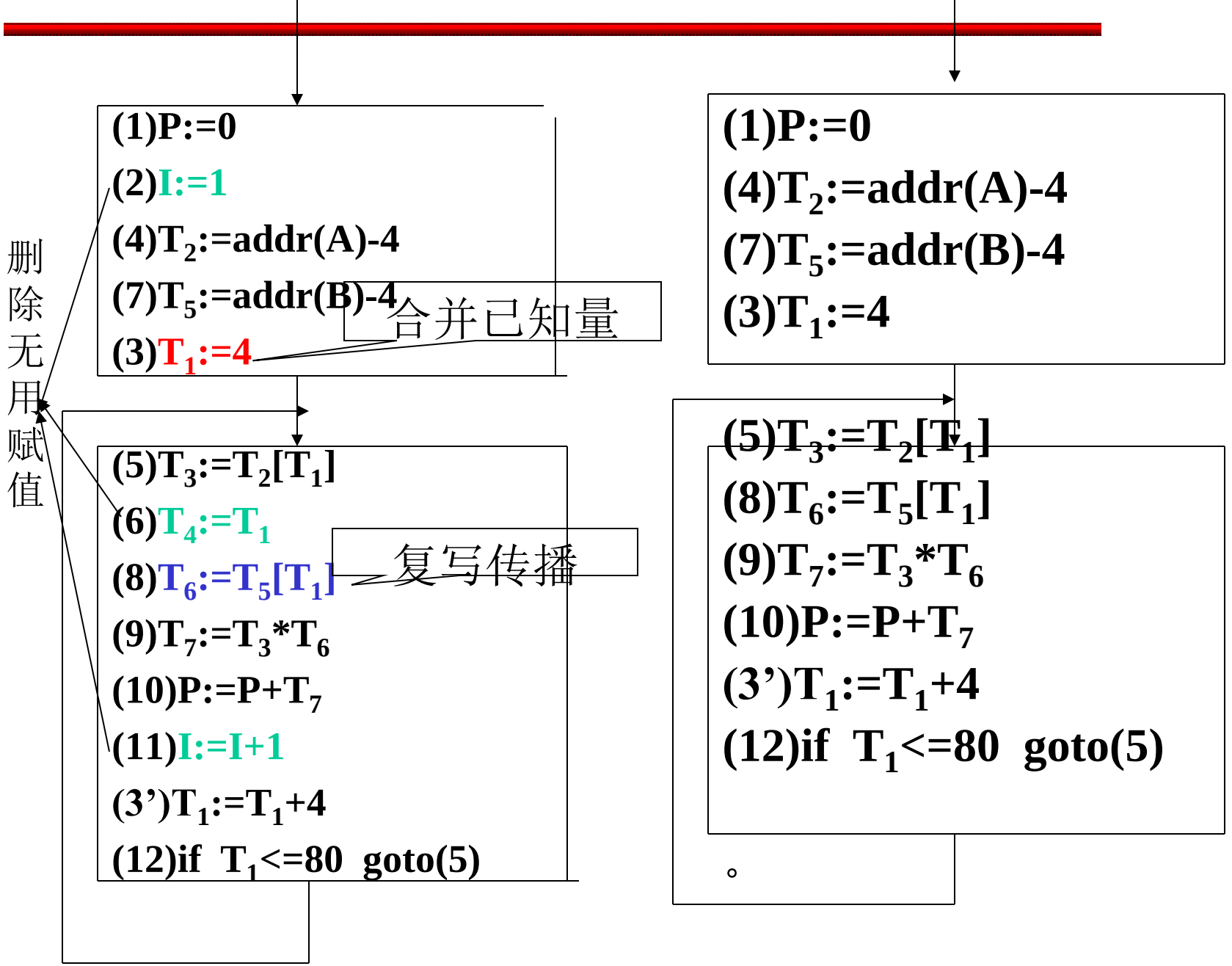
```
P:=0
for  I:=1 to 20 do
        P:=P+A[I]*B[I]
```


(1)P:=0
(2)I:=1
(3)T1:=4*I
(4)T2:=addr(A)-4
(5)T3:=T2[T1]
(6)T4:=4*I
(7)T5:=addr(B)-4
(8)T6:=T5[T4]
(9)T7:=T3*T6
(10)P:=P+T7
(11)I:=I+1
(12)if I<=20 goto(3)
(1)P:=0
(2)I:=1
(4)T2:=addr(A)-4
(7)T5:=addr(B)-4
代码外提
(3)T1:=4*I
(5)T3:=T2[T1]
(6)T4:=T1
删除多余运算
(8)T6:=T5[T4]
(9)T7:=T3*T6
(10)P:=P+T7
(11)I:=I+1
(12)if I<=20 goto(3)

- 数组元素的地址分两部分：基本地址T和变址地址T1。用T1[T]表示数组元素的地址。


(1)P:=0
(2)I:=1
(4)T2:=addr(A)-4
(7)T5:=addr(B)-4
(3)T1:=4*I
(5)T3:=T2[T1]
(6)T4:=T1
(8)T6:=T5[T4]
(9)T7:=T3*T6
(10)P:=P+T7
(11)I:=I+1
(12)if I<=20 goto(3)
(1)P:=0
(2)I:=1
(4)T2:=addr(A)-4
(7)T5:=addr(B)-4
(3)T1:=4*I
(5)T3:=T2[T1]
(6)T4:=T1
(8)T6:=T5[T4]
(9)T7:=T3*T6
(10)P:=P+T7
(11)I:=I+1
(3‘)T1:=T1+4
(12)if I<=20 goto(5)
强度削弱


(1)P:=0
(2)I:=1
(4)T2:=addr(A)-4
(7)T5:=addr(B)-4
(3)T1:=4*I
(5)T3:=T2[T1]
(6)T4:=T1
(8)T6:=T5[T4]
(9)T7:=T3*T6
(10)P:=P+T7
(11)I:=I+1
(3')T1:=T1+4
(12)if I<=20 goto(5)
(1)P:=0
(2)I:=1
(4)T2:=addr(A)-4
(7)T5:=addr(B)-4
(3)T1:=4*I
(5)T3:=T2[T1]
(6)T4:=T1
(8)T6:=T5[T1]
(9)T7:=T3*T6
(10)P:=P+T7
(11)I:=I+1
(3')T1:=T1+4
(12)if T1<=80 goto(5)
变换循环条件


(1)P:=0
(2)I:=1
(4)T2:=addr(A)-4
(7)T5:=addr(B)-4
(3)T1:=4
合并已知量
删除无用赋值
(5)T3:=T2[T1]
(6)T4:=T1
(8)T6:=T5[T1]
复写传播
(9)T7:=T3*T6
(10)P:=P+T7
(11)I:=I+1
(3')T1:=T1+4
(12)if T1<=80 goto(5)
(1)P:=0
(4)T2:=addr(A)-4
(7)T5:=addr(B)-4
(3)T1:=4
(5)T3:=T2[T1]
(8)T6:=T5[T1]
(9)T7:=T3*T6
(10)P:=P+T7
(3')T1:=T1+4
(12)if T1<=80 goto(5)
。

第十一章　代码生成

代码生成是把优化后的中间代码转换成特定目标机的机器语言或汇编语言。

代码生成程序的构造与输入的中间代码形式和目标机的指令系统及结构密切相关。

目标代码一般有三种形式：

（1）能够立即执行的机器语言代码，所有地址均已定位；

（2）待装配的机器语言模块，在执行之前，还需要把它们和某些运行程序连接起来，转换为能执行的机器语言代码；

（3）汇编语言代码，还需要经过汇编程序汇编，转换成可执行的机器语言代码。

设计代码生成程序的基本问题

- 代码生成程序的输入：中间代码表示和符号表中的信息
- 指令选择：寻找一个合适的目标机指令序列以实现给定的中间表示。
- 寄存器的分配：确定在程序的哪个点将哪些变量或中间量的值放在寄存器中。因为指令在寄存器中访问操作数的开销小于在内存中访问，应充分利用寄存器。
- 指令调度：确定程序指令的执行顺序，以充分利用指令的执行周期。