第10章(Part 2) 代码生成

1 代码生成概述

2 一个简单的代码生成程序

1 代码生成概述

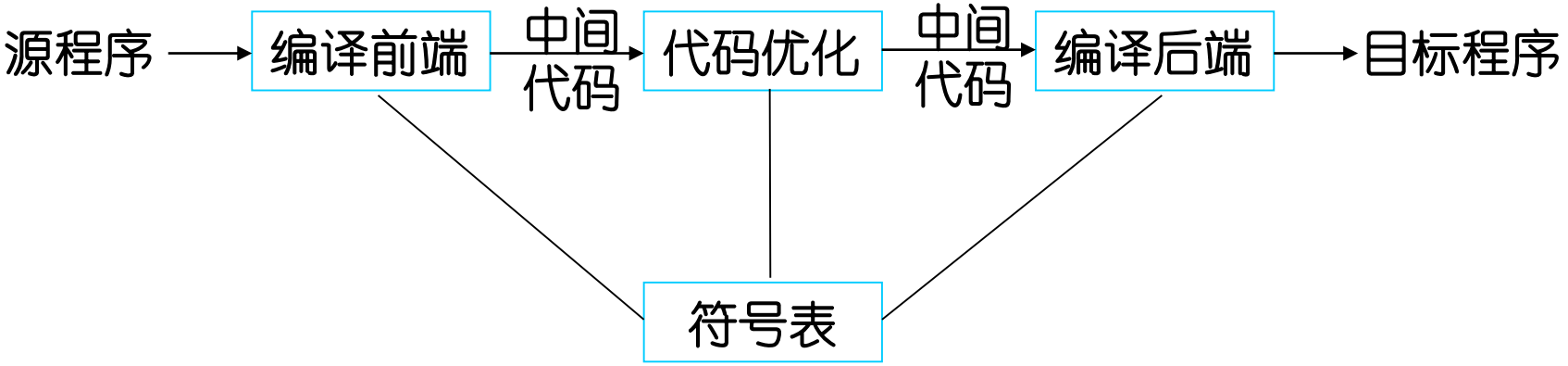
代码生成是把某种高级程序设计语言经过语法语义分析或优化后的中间代码作为输入，将其转换成特定机器的机器语言或汇编语言作为输出，这样的转换程序称为**代码生成程序**。

代码生成器的构造与输入的中间代码形式和输出的目标代码的机器结构密切相关。特别是高级程序设计语言和计算机硬件结构的多种多样性为代码生成的理论研究和实现技术带来很大的复杂性。

由于一个高级程序设计语言的目标代码需反复使用，代码生成器的设计要着重考虑**目标代码的质量问题**。

衡量目标代码的质量主要从**占用空间**和**执行效率**两个方面综合考虑。到底产生什么样的目标代码取决于具体的机器结构、指令格式、字长及寄存器的个数和种类，并与指令的语义和所用操作系统、存储管理等都密切相关。

代码生成程序在编译系统中的位置



目标代码的形式：

1 地址已定位的机器代码

2 待装配的机器代码模块

3 汇编语言

设计代码生成程序的基本问题

1 代码生成程序的输入

代码生成程序的输入由前端所产生的**中间表示**以及**符号表中的信息**组成。中间表示形式有：线形表示法（如后缀表示）、三地址表示法（如四元式）、抽象机表示法（栈式机器代码）和图形表示法（语法树）。

2 指令选择

指令选择是指寻找一个合适的目标机器指令序列来实现给定的中间表示。

生成的代码的质量取决于**它的执行速度**和**代码序列的长度**。

指令选择的原则:

(1) 减小产生代码的尺寸;

(2) 减少目标代码的执行时间;

(3) 减低目标代码的能耗。

机器指令形式

(op source, destination)

ADD s, d　　//d+s

SUB　s, d　　//d-s

MOV s, d　　//s ⇒ d

机器指令开销 (cost)

MOV　R, M　　**开销** 2

ADD　#1, R　　**开销** 2

MOV　R0, R1　　**开销** 1

目标机器的地址方式

地址方式	汇编形式	地址	增加的开销
直接地址方式	M	M	1
寄存器方式	R	R	0
间接寄存器方式	*R	contents(R)	0
索引方式	c(R)	c+contents(R)	1
间接索引方式	*c(R)	contents(c+contents(R))	1

a:=b+c

1. MOV b, R_0
   ADD c, R_0 cost=6
   MOV R_0, a
2. MOV b, a
   ADD c, a cost=6

假定R_0, R_1和R_2中分别存放了a, b和c的地址, 采用:

3. MOV *R_1, *R_0
   ADD *R_2, *R_0 cost=2

假定R_1和R_2中分别包含b和c的值, 并且b的值在这个赋值以后不再需要, 则还可有

4. ADD R_2, R_1
   MOV R_1, a cost=3

3 寄存器分配

寄存器的分配与指派：在寄存器分配期间，为程序的某一点选择驻留在寄存器中的一组变量；在随后的指派阶段，选出变量将要驻留的具体寄存器。

寄存器分配原则：

(1) 生成某变量的目标对象值时，尽量让变量值或计算结果保留在寄存器中直到寄存器不够分配为止。

(2) 当到基本块出口时，将变量的值存放在内存中。

(3) 在同一基本块内后面不再被引用的变量所占用的寄存器应尽早释放，以提高寄存器利用率。

4 指令调度

指令调度是指确定指令的执行顺序。

上述2、3、4阶段关系紧密！

2 一个简单的代码生成程序

计算机模型

假定一台M计算机具有n个通用寄存器为R0,R1,…,Rn-1,Rn。它们既可作为累加器又可作为变址器。如果用‘op’表示运算符，用‘M’表示内存单元，用变量名表示该变量所在的单元，‘C’表示常量，‘*’表示间址方式存取，指令形式可包含以下四种类型。

类型	指令形式	意义(设op是二目运算符)
直接地址型	op Ri, M	(Ri)op(M)⇒Ri
寄存器型	op Ri, Rj	(Ri)op(Rj)⇒Ri
变址型	op Ri, c(Rj)	(Ri)op((Rj)+c)⇒Ri
间接型	op Ri, *M	(Ri)op((M))⇒Ri
	op Ri, *Rj	(Ri)op((Rj))⇒Ri
	op Ri, *c(Rj)	(Ri)op(((Rj)+c))⇒Ri

指令	意义	指令	意义
LD R_1,B	把B单元的内容取到寄存器 R_1，即(B)⇒R_1。	J<X	如CT=0转X单元。
		J≤X	如CT=0或CT=1转X单元。
ST R_1,B	把寄存器R_1的内容存到B单元，即(R_1)⇒B。	J=X	如CT=1转X单元。
		J≠X	如CT≠1转X单元。
J X	无条件转向X单元。	J>X	如CT=2转X单元。
CMP A,B	把A单元和B单元的值进行比较，并根据比较情况把机器内部特征寄存器CT置成相应状态。CT占两个二进位。根据A<B或A=B或A>B分别置CT为0或1或2。	J≥X	如CT=2或CT=1转X单元。

待用信息链表法

为了在一个基本块内的目标代码中，寄存器得到充分利用，需**把基本块内还要被引用的变量值尽可能保存在寄存器中，而把基本块内不再被引用的变量所占的寄存器尽早释放**。

若在一个基本块中，变量A在四元式i中被定值，在i后面的四元式j中要引用A值，且从i到j之间没有其它对A的定值点，这时称**j是四元式i中对变量A的待用信息**或称**下次引用信息**，同时也称**A是活跃的**。若A被多处引用则可构成待用信息链与活跃信息链。

为了得到在一个基本块内每个变量的待用信息和活跃信息，可以**从基本块出口的四元式开始由后向前扫描**，对每个变量名建立相应的待用信息链和活跃变量信息链。

假定**基本块中的变量在出口处都是活跃的**，而对基本块内的**临时变量可分两种情况**处理。

a) 对于没经过数据流分析且中间代码生成的算法中临时变量不允许在基本块外引用，则临时变量在基本块出口处都认为是不活跃的。

b) 如果中间代码生成时的算法允许某些临时变量在基本块外引用时，则假定这些临时变量也是活跃的。

假设在变量的符号表的记录项中含有待用信息和活跃信息的栏目，**算法步骤**如下：

① 对各基本块的符号表中的"待用信息"栏和"活跃信息"栏置初值，即把"待用信息"栏置"非待用"，对"活跃信息"栏按在基本块出口处是否为活跃而置成"活跃"或"非活跃"。

现假定变量都是活跃的，临时变量都是非活跃的。

② 从基本块出口到基本块入口由后向前依次处理每个四元式。对每个四元式i：A:=B op C，依次执行下述步骤：

a) 把符号表中变量A的待用信息和活跃信息附加到四元式i上。

b) 把符号表中变量A的待用信息栏和活跃信息栏分别置为"非待用"和"非活跃"。

由于在i中对A的定值只能在i以后的四元式才能引用，因而对i以前的四元式来说A是不活跃也不可能是待用的。

c) 把符号表中B和C的待用信息和活跃信息附加到四元式i上。

d) 把符号表中B和C的待用信息栏置为"i"，活跃信息栏置为"活跃"。

例： 若用A，B，C，D表示变量，用T，U，V表示中间变量，有四元式如下：

(1) T:=A-B

(2) U:=A-C

(3) V:=T+U

(4) D:=V+U

待用信息和活跃信息链

变量名	待用信息					待用信息				
	初值	待用信息链				初值	活跃信息链			
A	F			(2)	(1)	L			L	L
B	F				(1)	L				L
C	F			(2)		L			L	
D	F	F				L	F			
T	F		(3)		F	F		L		F
U	F	(4)	(3)	F		F	L	L	F	
V	F	(4)	F			F	L	F		

待用信息和活跃信息在四元式上的标记如下所示。

(1) $T^{(3)L} := A^{(2)L} - B^{FL}$

(2) $U^{(3)L} := A^{FL} - C^{FL}$

(3) $V^{(4)L} := T^{FF} + U^{(4)L}$

(4) $D^{FL} := V^{FF} + U^{FF}$

代码生成算法

用一个数组RVALUE来描述(记录)每个寄存器当前的状况，是处于空闲状态还是被某个或某几个变量占用；用寄存器Ri的编号值作为数组RVALUE的下标，其数组元素值为变量名；

用数组AVALUE[M]表示变量的存放情况。一个变量的值可能存放在寄存器中或存放在内存中，也可能既在寄存器中又在内存中。

一个变量的值表示可能有：

RVALUE[Ri]={A, C}表示Ri的现行值是变量A，C的值

AVALUE[A]={A}表示A的值在内存中

AVALUE[A]={Ri, A}表示A的值既在寄存器Ri中又在内存中

AVALUE[A]={Ri}表示变量A的值在寄存器Ri中

设GETREG是一个函数过程，它的参数是一个形如i: A:=B op C的四元式，每次调用GETREG(i: A:=B op C)则返回一个寄存器R，用以存放A的结果值。

GETREG分配寄存器的算法为：

① 如果B的现行值在某寄存器Ri中，且该寄存器只包含B的值，或者B与A是同一标识符，或B在该四元式后不会再被引用，则可选取Ri作为所需的寄存器R，并转(4)。

② 如果有尚未分配的寄存器，则从中选用一个Ri为所需的寄存器R，并转(4)。

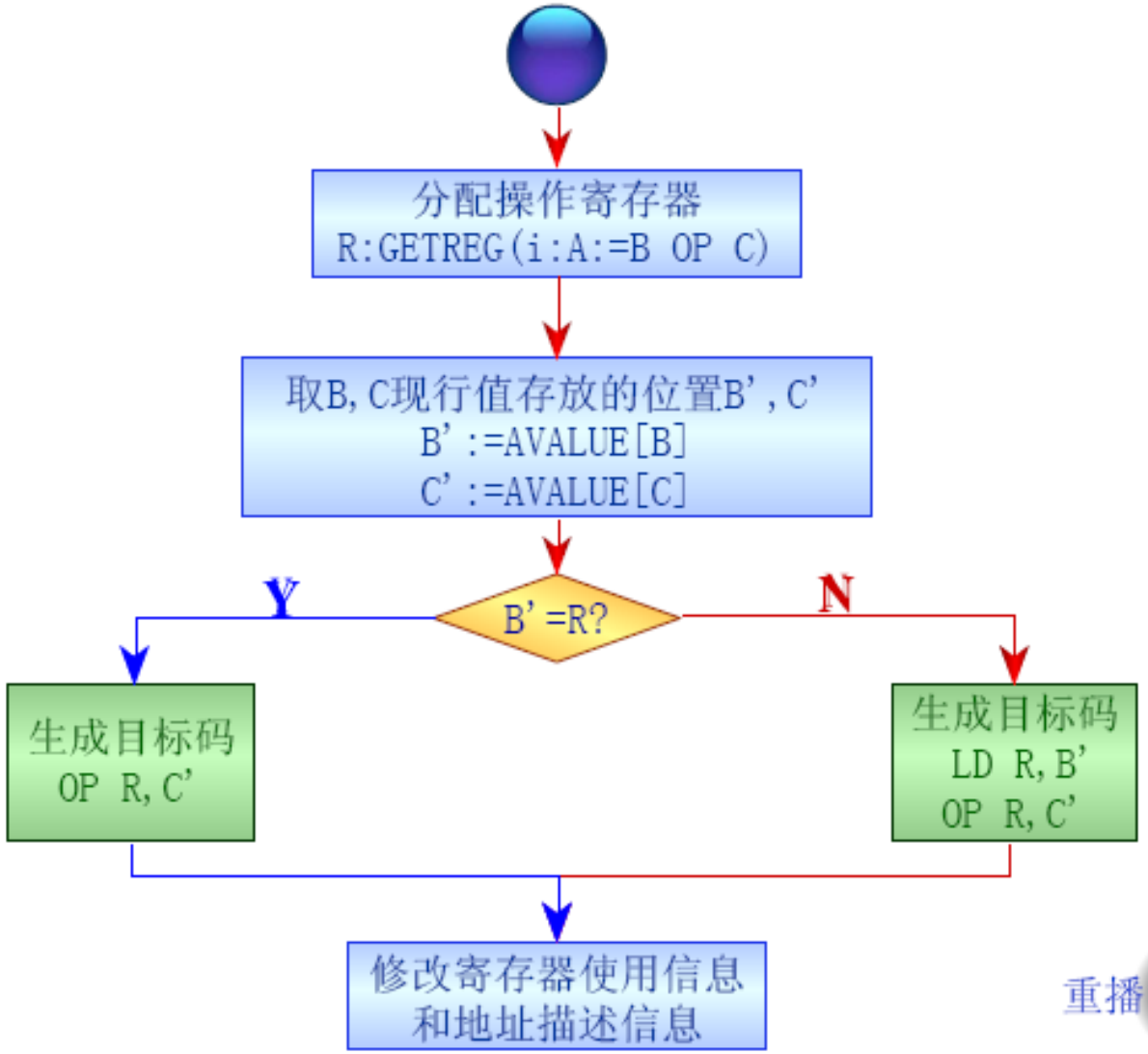
③ 从已分配的寄存器中选取一个Ri作为所需寄存器R，其选择原则为：占用该寄存器的变量值同时在主存中，或在基本块中引用的位置最远，这样对寄存器Ri所含的变量和变量在主存中的情况必须先做如下调整：即对RVALUE[Ri]中的每一变量M，如果M不是A且AVALUE[M]不包含M，则需完成以下处理。

a) 生成目标代码ST Ri,M;即把不是A的变量值由Ri中送入内存中。

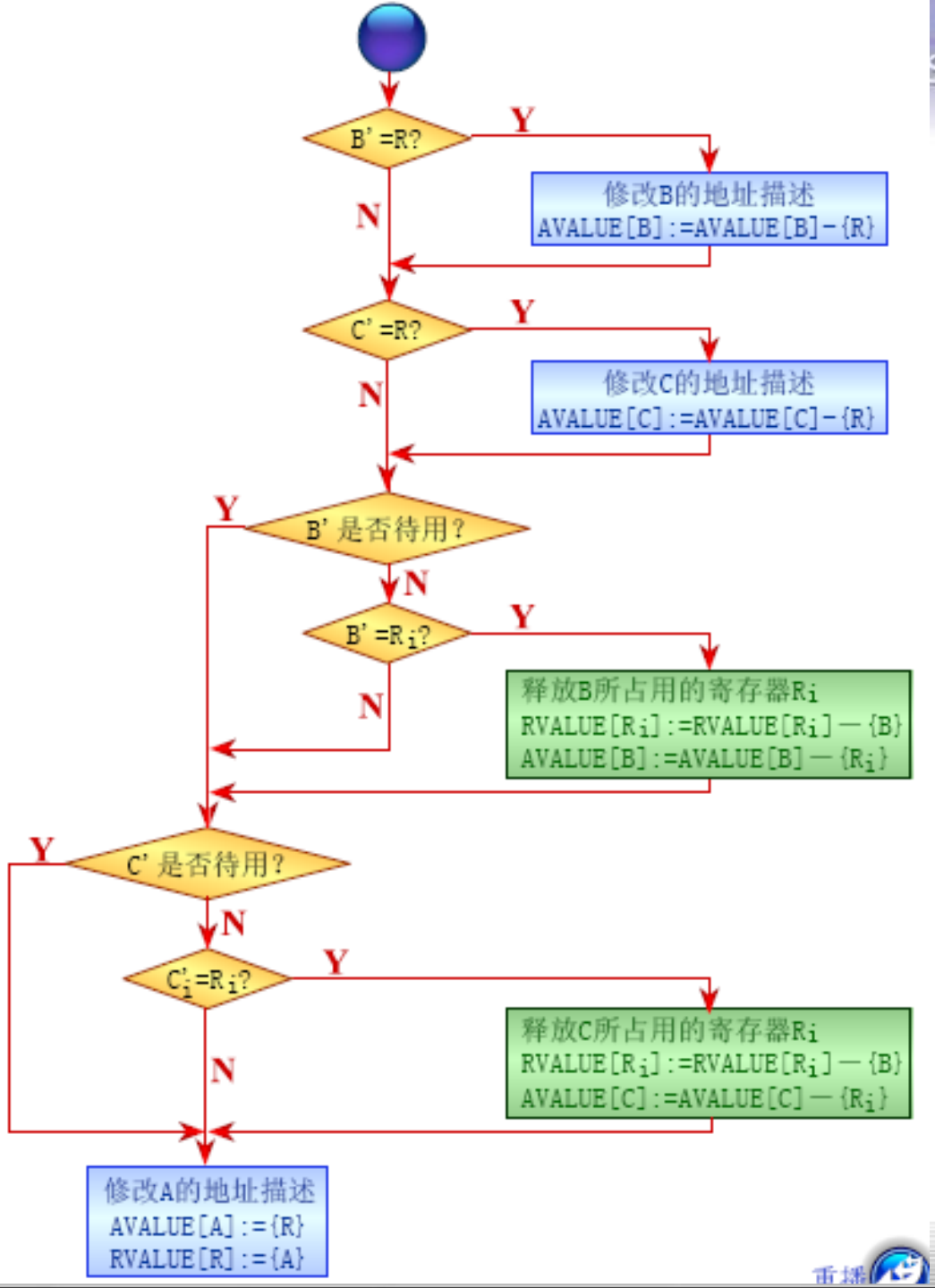
b) 如果M不是B，则令AVALUE[M]={M}，否则，令AVALUE[M]={M, Ri}。

c) 删除RVALUE[Ri]中的M。

④ 给出R，返回。


分配操作寄存器
R:GETREG(i:A:=B OP C)
取B,C现行值存放的位置B',C'
B':=AVALUE[B]
C':=AVALUE[C]
B'=R?
Y
N
生成目标码
OP R,C'
生成目标码
LD R,B'
OP R,C'
修改寄存器使用信息
和地址描述信息

重播


B'=R?
Y
修改B的地址描述
AVALUE[B]:=AVALUE[B]-{R}
N
C'=R?
Y
修改C的地址描述
AVALUE[C]:=AVALUE[C]-{R}
N
B'是否待用?
Y
N
B'=Ri?
Y
释放B所占用的寄存器Ri
RVALUE[Ri]:=RVALUE[Ri]—{B}
AVALUE[B]:=AVALUE[B]—{Ri}
N
C'是否待用?
Y
N
C'i=Ri?
Y
释放C所占用的寄存器Ri
RVALUE[Ri]:=RVALUE[Ri]—{B}
AVALUE[C]:=AVALUE[C]—{Ri}
N
修改A的地址描述
AVALUE[A]:={R}
RVALUE[R]:={A}

练习

1，

对照课件，好好看书吧！