




中国科学技术大学  
University of Science and Technology of China


代码生成  
《编译原理和技术》

张昱  
0551-63603804, yuzhang@ustc.edu.cn  
中国科学技术大学  
计算机科学与技术学院



中国科学技术大学  
University of Science and Technology of China


本章内容



本章内容

- 一个简单的代码生成算法，将中间代码IR映射成为可以在目标机器上运行的指令序列
- 涉及目标机器指令选择，寄存器分配和计算次序选择等基本问题


张昱：《编译原理和技术》代码生成2



中国科学技术大学  
University of Science and Technology of China

1. 代码生成器设计中的问题

- 目标程序
- 指令选择
- 寄存器的分配和指派
- 计算次序




中国科学技术大学  
University of Science and Technology of China

代码生成器设计中的问题

- 目标程序(target program)
  - 绝对机器语言程序(absolute machine-language ...)
    - 目标程序将装入到内存的固定地方
    - 粗略地说，相当于现在的可执行目标模块（第11章介绍）
  - 可重定位目标模块(relocatable object module)
    - 代码中含重定位信息，以适应重定位要求

张昱：《编译原理和技术》代码生成4



中国科学技术大学  
University of Science and Technology of China


代码生成器设计中的问题

- 目标程序
  - 可重定位目标模块

```
.L7:
    testl %eax,%eax
    je .L3
    testl %edx,%edx
    je .L7
    movl %edx,%eax
    jmp .L7
.L3:
    leave
    ret
```

可重定位目标模块中，
 需要有蓝色部分的重定位信息

张昱：《编译原理和技术》代码生成5




中国科学技术大学  
University of Science and Technology of China

代码生成器设计中的问题

- 目标程序
  - 绝对机器语言程序
    - 目标程序将装入到内存的固定地方
    - 粗略地说，相当于现在的可执行目标模块（第11章介绍）
  - 可重定位目标模块
    - 代码中含重定位信息，以适应重定位要求
    - 允许程序模块分别编译
    - 调用其它先前编译好的程序模块

张昱：《编译原理和技术》代码生成6



中国科学技术大学  
University of Science and Technology of China

代码生成器设计中的问题

❑ 目标程序

■ 绝对机器语言程序

■ 可重定位目标模块

❑ 代码中含重定位信息，以适应重定位要求

❑ 允许程序模块分别编译


❑ 调用其它先前编译好的程序模块

■ 汇编语言程序(assembly-language program)

❑ 免去编译器重复汇编器的工作

❑ 从教学角度，增加可读性

张昱：《编译原理和技术》代码生成7



中国科学技术大学  
University of Science and Technology of China


代码生成器设计中的问题

❑ 指令的选择(instruction selection)

■ 目标机器指令系统的性质决定了指令选择的难易程度，指令系统的统一性和完备性是重要的因素

■ 指令的速度和机器特点是另一些重要的因素

张昱：《编译原理和技术》代码生成8



中国科学技术大学  
University of Science and Technology of China

代码生成器设计中的问题

❑ 代码生成机制

逐条语句地产生代码，常常得到低质量的代码


例：三地址语句 $x = y + z$  ( $x$ ,  $y$ 和 $z$ 都静态分配)

MOV y, R0 /\* 把y装入寄存器R0 \*/

ADD z, R0 /\* 把z加到R0上 \*/

MOV R0, x /\* 把R0存入x中 \*/

张昱：《编译原理和技术》代码生成9



中国科学技术大学  
University of Science and Technology of China

代码生成器设计中的问题

语句序列 $a = b + c$   
 $d = a + e$

的一种目标代码如下：

MOV b, R0

ADD c, R0


MOV R0, a

MOV a, R0

ADD e, R0

MOV R0, d

张昱：《编译原理和技术》代码生成10



中国科学技术大学  
University of Science and Technology of China

代码生成器设计中的问题

语句序列 $a = b + c$   
 $d = a + e$

的一种目标代码如下：

MOV b, R0

ADD c, R0

MOV R0, a


MOV a, R0

ADD e, R0

MOV R0, d

由于a的值仍然存于寄存器R0中，因此该指令是冗余的。

张昱：《编译原理和技术》代码生成11



中国科学技术大学  
University of Science and Technology of China

代码生成器设计中的问题

语句序列 $a = b + c$   
 $d = a + e$

的一种目标代码如下：

MOV b, R0

ADD c, R0

MOV R0, a


MOV a, R0

ADD e, R0

MOV R0, d

如果a不再被使用，该指令也可以删除。

张昱：《编译原理和技术》代码生成12



中国科学技术大学  
University of Science and Technology of China

代码生成器设计中的问题

□ 代码生成机制

同一中间表示代码可以实现为多组指令序列  
不同实现之间的效率差别是很大

例：语句 $a = a + 1$ 可以有两种实现方式


MOV a, R0  
ADD #1, R0  
MOV R0, a

INC a

因此，生成高质量代码需要知道指令代价。

张翌：《编译原理和技术》代码生成

13



中国科学技术大学  
University of Science and Technology of China

代码生成器设计中的问题

□ 代码生成机制


考虑指令的代价和序列长度、运算对象和结果如何存储

↑  
速度越快  
成本越高  
空间越小

CPU  
Register  
Cache  
DRAM  
External  
NVM  
Large SSD  
Large hard drive

张翌：《编译原理和技术》代码生成

14



中国科学技术大学  
University of Science and Technology of China

代码生成器设计中的问题

□ 代码生成机制

考虑指令的代价和序列长度、运算对象和结果如何存储


↑  
速度越快  
成本越高  
空间越小

CPU  
Register  
Cache  
DRAM  
External  
NVM  
Large SSD  
Large hard drive

由编译器分配  
由MMU和操作系统共同管理  
由I/O处理

张翌：《编译原理和技术》代码生成

15



中国科学技术大学  
University of Science and Technology of China

代码生成器设计中的问题

□ 寄存器的合理使用

相比操作置于内存的运算对象，操作寄存器型操作数的指令要短一些，执行也快一些

■ 寄存器分配(register allocation)


- 选择驻留在寄存器中的一组变量

■ 寄存器指派(register assignment)

- 挑选变量要驻留的具体寄存器

张翌：《编译原理和技术》代码生成

16



中国科学技术大学  
University of Science and Technology of China

代码生成器设计中的问题

□ 计算次序的选择(evaluation order)


■ 程序中计算的执行次序会影响目标代码的执行效率

■ 如，对表达式的计算而言，一种计算次序可能会比其它次序需要较少的寄存器来保存中间结果

■ 选择最佳计算次序是一个NP完全问题

张翌：《编译原理和技术》代码生成

17




中国科学技术大学  
University of Science and Technology of China

2. 目标语言

□ 目标机器指令集

□ 指令代价

3



中国科学技术大学  
University of Science and Technology of China

目标语言

□ 一个简单目标机器的指令系统

■ 字节寻址，四个字节组成一个字

■ 有 $n$ 个通用寄存器 $R0, R1, \dots, Rn-1$


■ 二地址指令：op 源，目的

MOV {源传到目的}

ADD {源加到目的}

SUB {目的减去源}

张昱：《编译原理和技术》代码生成19



中国科学技术大学  
University of Science and Technology of China

目标语言

□ 例 指令实例

MOV R0, M

MOV 4(R0), M


4(R0)的值： $contents(4 + contents(R0))$

MOV \*4(R0), M

\*4(R0)的值： $contents(contents(4 + contents(R0)))$

MOV #1, R0

张昱：《编译原理和技术》代码生成20



中国科学技术大学  
University of Science and Technology of China


目标语言

□ 指令的代价(instruction costs)

在上述简单的目标机器上，指令代价简化为

1 + 指令的源和目的寻址模式(addressing mode)的附加代价

张昱：《编译原理和技术》代码生成21




中国科学技术大学  
University of Science and Technology of China

目标语言

□ 寻址模式和它们的汇编语言形式及附加代价

模式	形式	地址	附加代价
绝对地址	M	M	1
寄存器	R	R	0
变址	$c(R)$	$c + contents(R)$	1
间接寄存器	*R	$contents(R)$	0
间接变址	* $c(R)$	$contents(c + contents(R))$	1
直接量	# $c$	$c$	1

张昱：《编译原理和技术》代码生成22



中国科学技术大学  
University of Science and Technology of China


目标语言

□ 指令代价简化为

1 + 指令的源和目的地址模式的附加代价

指令	代价
MOV R0, R1	
MOV R5, M	
ADD #1, R3	
SUB 4(R0), *12(R1)	

张昱：《编译原理和技术》代码生成23



中国科学技术大学  
University of Science and Technology of China

目标语言

□ 指令代价简化为

1 + 指令的源和目的地址模式的附加代价

指令	代价	
MOV R0, R1	1	寄存器
MOV R5, M	2	寄存器+内存
ADD #1, R3	2	常量+寄存器
SUB 4(R0), *12(R1)	3	变址+间接变址

张昱：《编译原理和技术》代码生成24

中国科学技术大学

University of Science and Technology of China

目标语言

例  $a = b + c$ ,  $a$ 、 $b$ 和 $c$ 都静态分配内存单元

可生成

MOV b, R0

ADD c, R0

MOV R0, a

也可生成

MOV b, a

ADD c, a

张昱:《编译原理和技术》代码生成

25

中国科学技术大学

University of Science and Technology of China

目标语言

例  $a = b + c$ ,  $a$ 、 $b$ 和 $c$ 都静态分配内存单元

可生成

MOV b, R0

ADD c, R0

MOV R0, a

代价= 6

也可生成

MOV b, a

ADD c, a

代价= 6

张昱:《编译原理和技术》代码生成

26

中国科学技术大学

University of Science and Technology of China

目标语言

例  $a = b + c$ ,  $a$ 、 $b$ 和 $c$ 都静态分配内存单元

若R0, R1和R2分别含 $a$ ,  $b$ 和 $c$ 的地址, 则可生成

MOV \*R1, \*R0

ADD \*R2, \*R0

代价= 2

若R1和R2分别含 $b$ 和 $c$ 的值, 并且 $b$ 的值在这个赋值后不再需要, 则可生成

ADD R2, R1

MOV R1, a

代价= 3

张昱:《编译原理和技术》代码生成

27

中国科学技术大学

University of Science and Technology of China

3. 代码生成器的输入

中间代码IR的表示

基本块

流图

循环

中国科学技术大学

University of Science and Technology of China

三地址代码

三地址代码(three-address code)

一般形式:  $x = y \text{ op } z$

例 表达式 $x + y * z$ 翻译成的三地址语句序列是

$t_1 = y * z$

$t_2 = x + t_1$

张昱:《编译原理和技术》代码生成

29

中国科学技术大学

University of Science and Technology of China

基本块和流图

基本块

连续的语句序列, 控制流从它的开始进入, 并从它的末尾离开, 没有停止或分支的可能性 (末尾除外)

流图(flow graph)

用有向边表示基本块之间的控制流信息, 基本块作为结点

(1)prod = 0

(2) i = 1

(3)  $t_1 = 4 * i$

(4)  $t_2 = a[t_1]$

(5)  $t_3 = 4 * i$

(6)  $t_4 = b[t_3]$

(7)  $t_5 = t_2 * t_4$

(8)  $t_6 = \text{prod} + t_5$

(9) prod =  $t_6$

(10)  $t_7 = i + 1$

(11)  $i = t_7$

(12) if  $i <= 20$  goto (3)

$B_1$

$B_2$

张昱:《编译原理和技术》代码生成

30

5

中国科学技术大学

University of Science and Technology of China

流图上的程序点和路径

流图上的(程序)点

基本块中，两个相邻的语句之间为程序的一个点

基本块的开始点和结束点

流图上的路径

点序列 $p_1, p_2, \dots, p_n$ ，对1和 $n-1$ 间的每个 $i$ ，满足

(1)  $p_i$ 是先于一个语句的点， $p_{i+1}$ 是同一基本块中位于该语句后的点，或者

(2)  $p_i$ 是某基本块的结束点， $p_{i+1}$ 是后继块的开始点

张翌：《编译原理和技术》代码生成

31

中国科学技术大学

University of Science and Technology of China

流图上的路径

流图(flow graph)

(1)  $d_1: a = 1$

$B_1$

(2)

(3) if read( ) <= 0 goto  $B_4$

$B_2$

(4)

(5)  $d_2: b = a$

$B_3$

(6)  $d_3: a = 243$

(7) goto  $B_3$

(8)

(9)

$B_4$

(1, 2, 3, 4, 9)

(1, 2, 3, 4, 5, 6, 7, 8, 3, 4, 9)

(1, 2, 3, 4, 5, 6, 7, 8, 3, 4, 5, 6, 7, 8, 3, 4, 9)

(1, 2, 3, 4, 5, 6, 7, 8, 3, 4, 5, 6, 7, 8, 3, 4, 5, 6, 7, 8, ...)

路径长度无限

路径数无限

张翌：《编译原理和技术》代码生成

32

中国科学技术大学

University of Science and Technology of China

循环

循环

流图中的一个结点集合 $L$ 是一个循环，如果：

该集合中所有结点是强连通的

该集合有唯一的入口结点

内循环

不包含其他循环的循环

(1)  $d_1: a = 1$

$B_1$

(2)

(3) if read( ) <= 0 goto  $B_4$

$B_2$

(4)

(5)  $d_2: b = a$

$B_3$

(6)  $d_3: a = 243$

(7) goto  $B_3$

(8)

(9)

$B_4$

张翌：《编译原理和技术》代码生成

33

中国科学技术大学

University of Science and Technology of China

循环

识别循环并对循环专门处理的重要性

程序执行的大部分时间消耗在循环上，改进循环性能的优化会对程序执行产生显著影响

循环也会影响程序分析的运行时间

支配结点

$d$ 是 $n$ 的支配结点( $d \text{ dom } n$ ): 若从初始结点起，每条到达 $n$ 的路径都要经过 $d$

结点是它本身的支配结点

循环的入口是循环中所有结点的支配结点

1

2

3

4

5

6

7

8

9

10

$d$

$n$

张翌：《编译原理和技术》代码生成

34

中国科学技术大学

University of Science and Technology of China

回边和可归约性

深度优先表示

1

2

3

4

5

6

7

8

9

10

1

2

3

4

5

6

7

8

9

10

张翌：《编译原理和技术》代码生成

35

中国科学技术大学

University of Science and Technology of China

流图中的边的分类

深度优先表示

前进边(深度优先生成树的边)

$m \rightarrow n$ 是后撤边，如果 $n$ 在深度优先生成树上是 $m$ 的祖先

$4 \rightarrow 3, 7 \rightarrow 4, 10 \rightarrow 7, 8 \rightarrow 3$ 和 $9 \rightarrow 1$

$m \rightarrow n$ 是交叉边，如果 $n$ 和 $m$ 在深度优先生成树上互不为对方的祖先

$2 \rightarrow 3$ 和 $5 \rightarrow 7$

张翌：《编译原理和技术》代码生成

36

6

中国科学技术大学

University of Science and Technology of China

回边和可归约性

回边

如果有  $a \text{ dom } b$ ，那么边  $b \rightarrow a$  叫做回边

可归约性

一个流图称为可归约的，如果在它任何深度优先生成树上，所有的后撤边都是回边。

张昱：《编译原理和技术》代码生成

37

中国科学技术大学

University of Science and Technology of China

回边和可归约性

回边

如果有  $a \text{ dom } b$ ，那么边  $b \rightarrow a$  叫做回边

可归约性

一个流图称为可归约的，如果在它任何深度优先生成树上，所有的后撤边都是回边。

如果把一个流图中所有回边删掉后，剩余的图无环

张昱：《编译原理和技术》代码生成

38

中国科学技术大学

University of Science and Technology of China

不可归约流图

开始结点是1

$2 \rightarrow 3$  和  $3 \rightarrow 2$  都不是回边

该图不是无环的

从结点2和3两处都能进入由它们构成的环

张昱：《编译原理和技术》代码生成

39

中国科学技术大学

University of Science and Technology of China

流图的深度

深度是无环路径中包含的最大后撤边数

深度不大于流图中循环嵌套的层数

该例深度为3

$10 \rightarrow 7 \rightarrow 4 \rightarrow 3$

张昱：《编译原理和技术》代码生成

40

中国科学技术大学

University of Science and Technology of China

自然循环

自然循环的性质

有惟一的入口结点(首结点)。首结点支配该循环中的所有结点

至少存在一条回边进入该循环的首结点

是流图的强连通分量(SCC)中的一种类型

回边  $n \rightarrow d$  确定的自然循环

$d$  加上不经过  $d$  能到达  $n$  的所有结点

结点  $d$  是该循环的首结点

张昱：《编译原理和技术》代码生成

41

中国科学技术大学

University of Science and Technology of China

自然循环

回边  $n \rightarrow d$  确定的自然循环是  $d$  加上不经过  $d$  能到达  $n$  的所有结点

回边  $10 \rightarrow 7$  循环  $\{7, 8, 10\}$


回边  $7 \rightarrow 4$  循环  $\{4, 5, 6, 7, 8, 10\}$

回边  $4 \rightarrow 3$  和  $8 \rightarrow 3$  循环  $\{3, 4, 5, 6, 7, 8, 10\}$

回边  $9 \rightarrow 1$  循环  $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

张昱：《编译原理和技术》代码生成

42



中国科学技术大学

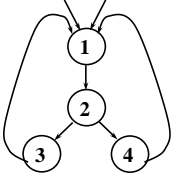
University of Science and Technology of China

内循环

内循环


若一个循环的结点集合是另一个循环的结点集合的子集

两个循环有相同的首结点，  
 但并非一个结点集是另一个  
 的子集，则看成一个循环



张翌：《编译原理和技术》代码生成

43



中国科学技术大学

University of Science and Technology of China

下次引用信息

过程中变量被使用的信息=>帮助寄存器的分配和释放

名字的引用(use)

假设三地址码语句i为x赋值，语句j将x作为运算对象，且i到j的控制流路径中无其他对x的赋值语句，则称语句j引用了语句i计算的x值

计算基本块中下次引用信息的方法

对每一个基本块，反向扫描，对语句 $x = y \text{ op } z$ ，在符号表中记录x、y、z是否活跃或会被下次引用

张翌：《编译原理和技术》代码生成

44



中国科学技术大学

University of Science and Technology of China

4.一个简单的代码生成器

寄存器和地址的描述


代码生成算法

寄存器选择函数

为特殊语句产生代码

张翌：《编译原理和技术》代码生成

45



中国科学技术大学

University of Science and Technology of China

一个简单的代码生成器

基本考虑

依次考虑基本块的每个语句，为其产生代码

假定三地址语句的每种算符都有对应的目标机器算符

假定计算结果尽可能长久地保留在寄存器中，除非：
 


该寄存器要用于其它计算，或者

到基本块结束

为此，在生成代码过程中需要记录一些信息

张翌：《编译原理和技术》代码生成

46



中国科学技术大学

University of Science and Technology of China

一个简单的代码生成器

寄存器描述和地址描述

例：对 $a = b + c$ 

如果寄存器Ri含b，Rj含c，且b此后不再活跃产生ADD Rj, Ri，结果a在Ri中

如果Ri含b，但c在内存单元，b仍然不再活跃产生ADD c, Ri，或者产生
 

MOV c, Rj

ADD Rj, Ri

若c的值以后还要用，第二种代码较有吸引力

张翌：《编译原理和技术》代码生成

47



中国科学技术大学

University of Science and Technology of China

一个简单的代码生成器

在代码生成过程中，需要跟踪寄存器的内容和名字的地址

寄存器描述记住每个寄存器当前存的是什么，即在任何一点，每个寄存器保存若干个(包括零个)名字的值

例：
 

// 语句前，R0保存变量a的值

$b = a$  // 不为该语句产生任何指令

// 语句后，R0保存变量a和b的值

张翌：《编译原理和技术》代码生成

48

8





## 一个简单的代码生成器

### □ 在代码生成过程中，需要跟踪

寄存器的内容和名字的地址

- 寄存器描述记住每个寄存器当前存的是什么，即在任何一点，每个寄存器保存若干个（包括零个）名字的值

- 名字（变量）的地址描述记住运行时每个名字的当前值可以在哪个场所找到。这个场所可以是寄存器、栈单元、内存地址、甚至是它们的某个集合

例：产生MOV c, R0后，c值可在R0和c的存储单元找到



## 一个简单的代码生成器

### □ 在代码生成过程中，需要跟踪

寄存器的内容和名字的地址

- 寄存器描述记住每个寄存器当前存的是什么，即在任何一点，每个寄存器保存若干个（包括零个）名字的值

- 名字（变量）的地址描述记住运行时每个名字的当前值可以在哪个场所找到。这个场所可以是寄存器、栈单元、内存地址、甚至是它们的某个集合

例：产生MOV c, R0后，c值可在R0和c的存储单元找到

- 名字的地址信息存于符号表，另建寄存器描述表
- 这两个描述在代码生成过程中是变化的



## 一个简单的代码生成器

### □ 寄存器选择函数

- 函数getReg返回保存 $x = y \text{ op } z$ 的x值的场所L

- 如果名字y在R中，这个R不含其它名字的值并且在执行 $x = y \text{ op } z$ 后y不再有下次引用，那么返回这个R作为L
- 否则，如果有的话，返回一个空闲寄存器
- 否则，如果x在块中有下次引用，或者op是必须用寄存器的算符，那么找一个已被占用的寄存器R（可能产生MOV R, M指令，并修改M的描述）
- 否则，如果x在基本块中不再引用，或者找不到适当的被占用寄存器，选择x的内存单元作为L



## 一个简单的代码生成器

### □ 代码生成算法

- 对每个三地址语句 $x = y \text{ op } z$

- 调用函数getReg决定放y op z计算结果的场所L
- 查看y的地址描述，确定y值当前的一个场所y'。如果y的值还不L中，产生指令MOV y', L
- 产生指令op z', L，其中z'是z的当前场所之一
- 如果y和/或z的当前值不再引用，在块的出口也不活跃，并且还在寄存器中，那么修改寄存器描述，使得不再包含y和/或z的值



## 一个简单的代码生成器

### □ 赋值语句 $d = (a - b) + (a - c) + (a - c)$


- 编译产生三地址语句序列：

$t_1 = a - b$   
 $t_2 = a - c$   
 $t_3 = t_1 + t_2$   
 $d = t_3 + t_2$



## 一个简单的代码生成器

语 句	生成的代码	寄存器描述	名字的地址描述
		寄存器空	
$t_1 = a - b$			
$t_2 = a - c$			
$t_3 = t_1 + t_2$			
$d = t_3 + t_2$			




中国科学技术大学  
University of Science and Technology of China

一个简单的代码生成器

语 句	生成的代码	寄存器描述	名字的地址描述
		寄存器空	
$t_1 = a - b$	MOV a, R0 SUB b, R0	R0含 $t_1$	$t_1$ 在R0中
$t_2 = a - c$			
$t_3 = t_1 + t_2$			
$d = t_3 + t_2$			

张翌：《编译原理和技术》代码生成55




中国科学技术大学  
University of Science and Technology of China

一个简单的代码生成器

语 句	生成的代码	寄存器描述	名字的地址描述
		寄存器空	
$t_1 = a - b$	MOV a, R0 SUB b, R0	R0含 $t_1$	$t_1$ 在R0中
$t_2 = a - c$	MOV a, R1 SUB c, R1	R0含 $t_1$ R1含 $t_2$	$t_1$ 在R0中 $t_2$ 在R1中
$t_3 = t_1 + t_2$			
$d = t_3 + t_2$			

张翌：《编译原理和技术》代码生成56




中国科学技术大学  
University of Science and Technology of China

一个简单的代码生成器

语 句	生成的代码	寄存器描述	名字的地址描述
		寄存器空	
$t_1 = a - b$	MOV a, R0 SUB b, R0	R0含 $t_1$	$t_1$ 在R0中
$t_2 = a - c$	MOV a, R1 SUB c, R1	R0含 $t_1$ R1含 $t_2$	$t_1$ 在R0中 $t_2$ 在R1中
$t_3 = t_1 + t_2$	ADD R1,R0	R0含 $t_3$ R1含 $t_2$	$t_3$ 在R0中 $t_2$ 在R1中
$d = t_3 + t_2$			

张翌：《编译原理和技术》代码生成57



中国科学技术大学  
University of Science and Technology of China

一个简单的代码生成器

语 句	生成的代码	寄存器描述	名字的地址描述
		寄存器空	
$t_1 = a - b$	MOV a, R0 SUB b, R0	R0含 $t_1$	$t_1$ 在R0中
$t_2 = a - c$	MOV a, R1 SUB c, R1	R0含 $t_1$ R1含 $t_2$	$t_1$ 在R0中 $t_2$ 在R1中
$t_3 = t_1 + t_2$	ADD R1,R0	R0含 $t_3$ R1含 $t_2$	$t_3$ 在R0中 $t_2$ 在R1中
$d = t_3 + t_2$	ADD R1,R0 MOV R0, d	R0含d	d在R0中 d在R0和内存中

张翌：《编译原理和技术》代码生成58



中国科学技术大学  
University of Science and Technology of China

一个简单的代码生成器

□ 前三条指令可以修改，使执行代价降低


修改前

MOV a, R0  
SUB b, R0  
MOV a, R1  
SUB c, R1  
...

修改后

MOV a, R0  
MOV R0, R1  
SUB b, R0  
SUB c, R1  
...

张翌：《编译原理和技术》代码生成59



中国科学技术大学  
University of Science and Technology of China

一个简单的代码生成器


□ 为特殊语句产生代码

■ 变址和指针语句

□ 变址与指针运算的三地址语句的处理和二元运算符的处理相同

语句	i在寄存器Ri中		i在内存Mi中		i在栈中	
	代码	代价	代码	代价	代码	代价
$a = b[i]$	MOV b(Ri), R	2	MOV Mi, R MOV b(R), R	4	MOV Si(Rs), R MOV b(R), R	4
$b[i] = a$	MOV a, b(Ri)	3	MOV Mi, R MOV a, b(R)	5	MOV Si(Rs), R MOV a, b(R)	5

张翌：《编译原理和技术》代码生成60



一个简单的代码生成器

中国科学技术大学  
University of Science and Technology of China

为特殊语句产生代码

变址和指针语句

变址与指针运算的三地址语句的处理和二元算符的处理相同


条件语句

根据寄存器的值是否为下面六个条件之一进行分支：负、零、正、非负、非零和非正

用条件码来表示计算的结果或装入寄存器的值是负、零还是正

张昱：《编译原理和技术》代码生成

61



一个简单的代码生成器

中国科学技术大学  
University of Science and Technology of China

1、根据寄存器的值是否为下面六个条件之一进行分支：负、零、正、非负、非零和非正


例 if  $x < y$  goto  $z$

把 $x$ 减 $y$ 的值存入寄存器 $R$

如果 $R$ 的值为负，则跳到 $z$

张昱：《编译原理和技术》代码生成

62



一个简单的代码生成器

中国科学技术大学  
University of Science and Technology of China

2、用条件码的例子

例：若if  $x < y$  goto  $z$  | 则：  $x = y + w$

的实现是： | if  $x < 0$  goto  $z$

CMP  $x, y$  | 的实现是：

CJ<  $z$  | MOV  $y, R0$


| ADD  $w, R0$

| MOV  $R0, x$

| CJ<  $z$

张昱：《编译原理和技术》代码生成

63



本章小结

中国科学技术大学  
University of Science and Technology of China

中间代码

目标程序

Tree、DAG  
基本块+流图  
+循环

考虑目标机器语言即指令集

寻址模式

合理利用寄存器  
理解指令代价  
各种优化策略

目标机器指令序列

可重定位

速度快+开销低

简单的代码生成算法

张昱：《编译原理和技术》代码生成

64