



# 编译原理与技术



李文生



北京邮电大学计算机学院(国家示范性软件学院)

SCHOOL OF COMPUTER SCIENCE(NATIONAL PILOT SOFTWARE ENGINEERING SCHOOL)BUPT



# 第9章 中间代码优化



李文生

2022年8月16日 星期二



北京邮电大学计算机学院(国家示范性软件学院)

SCHOOL OF COMPUTER SCIENCE(NATIONAL PILOT SOFTWARE ENGINEERING SCHOOL)BUPT

# 教学内容、目标与要求

## ■ 教学内容

- 代码优化概述
- 基本块与流图
- 基本块优化
- 循环优化

## ■ 教学目标与要求

- 了解代码优化程序的功能及其在编译程序中的位置；
- 掌握基本块的概念及其划分方法；  
掌握程序流图的概念及其构造方法；
- 掌握代码优化的主要种类；

## ■ 教学目标与要求（续）

- 掌握基本块优化技术，主要包括：
  - 公共表达式的删除、复制传播、常数合并及常数传播、削弱计算强度等；
- 掌握循环优化技术，主要包括：
  - 循环展开、代码外提、削弱计算强度、删除归纳变量；
- 能够针对中间代码，识别出优化需求，并进行优化分析，提出优化方案，并对中间代码进行相应的优化

# 内容目录

- 9.1 代码优化概述
- 9.2 基本块及控制流图
- 9.3 基本块优化
- 9.4 循环优化
- 小结

# 9.1 代码优化概述

## ■ 代码优化程序的任务

- 对中间代码或目标代码进行等价变换，使变换后的代码质量更高。

## ■ 对代码优化程序的要求

- 等价变换
- 提高目标代码的执行速度
- 减少目标代码占用的空间

## ■ 代码优化程序的位置

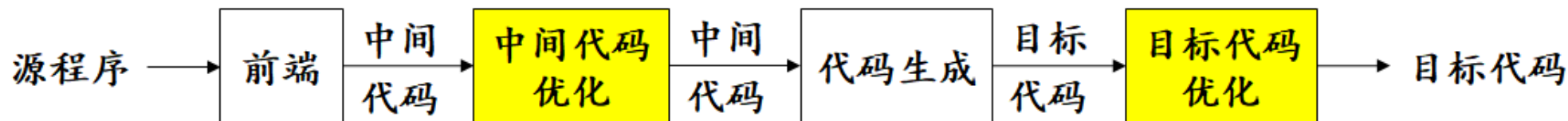
- 目标代码生成之前的中间代码优化
- 目标代码生成之后的目标代码优化

## ■ 中间代码优化

- 基本块优化
- 循环优化
- 全局优化

## ■ 目标代码优化

- 窥孔优化



## 9.2 基本块与流图

## 基本块的划分方法

### ■ 基本块

- 具有原子性的一组连续语句序列。
- 控制从第一条语句（入口语句）流入，从最后一条语句（出口语句）流出，中途没有停止或分支。

### ■ 如：

$t_1 := a * a$   
 $t_2 := b * b$   
 $t_3 := t_1 + t_2$

### ■ 基本块：

$t_1 := a * a$   
 $t_2 := a * b$   
 $t_3 := 2 * t_2$   
 $t_4 := t_1 + t_3$   
 $t_5 := b * b$   
 $t_6 := t_4 + t_5$

### ■ 确定入口语句：

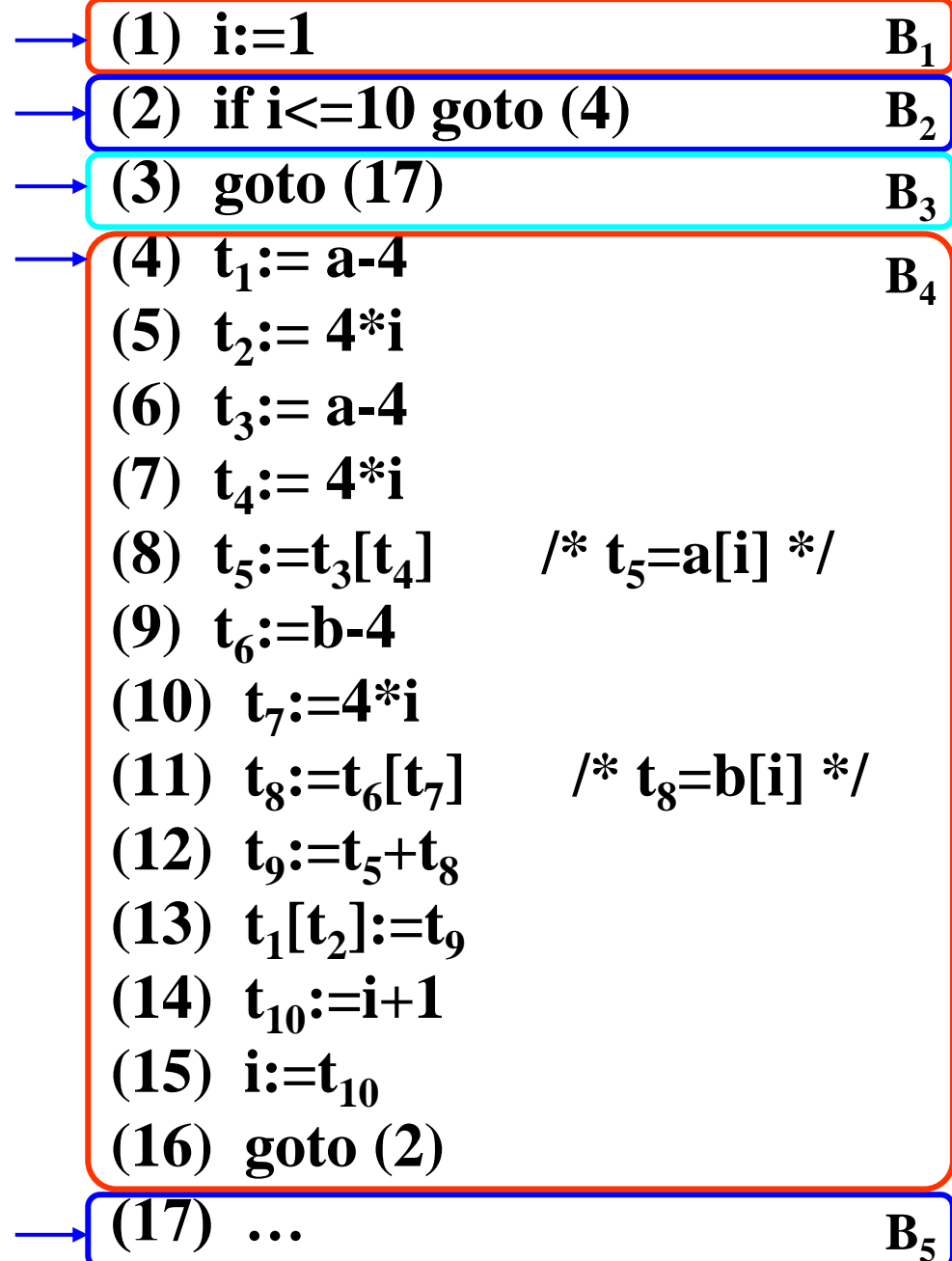
- 三地址代码的第一条语句；
- goto语句转移到的目标语句；
- 紧跟在goto语句后面的语句。

### ■ 确定基本块：

- 从一个入口语句（含该语句）到下一个入口语句（不含）之间的语句序列；
- 从一个入口语句（含该语句）到停止语句（含该语句）之间的语句序列。

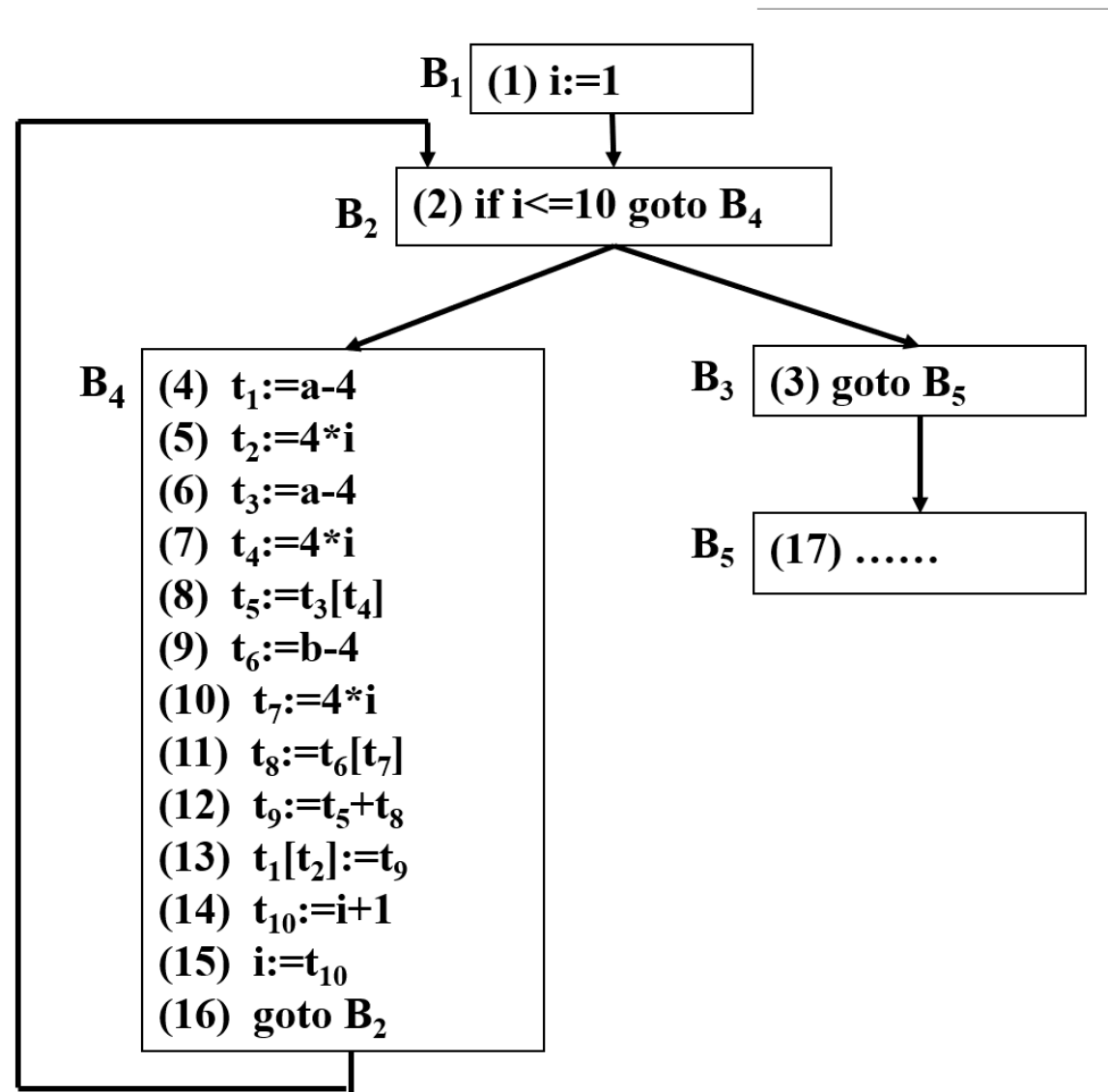
# Pascal程序片断:

```
i:=1;  
while (i<=10) do  
begin  
    a[i]:=a[i]+b[i];  
    i:=i+1  
end;
```



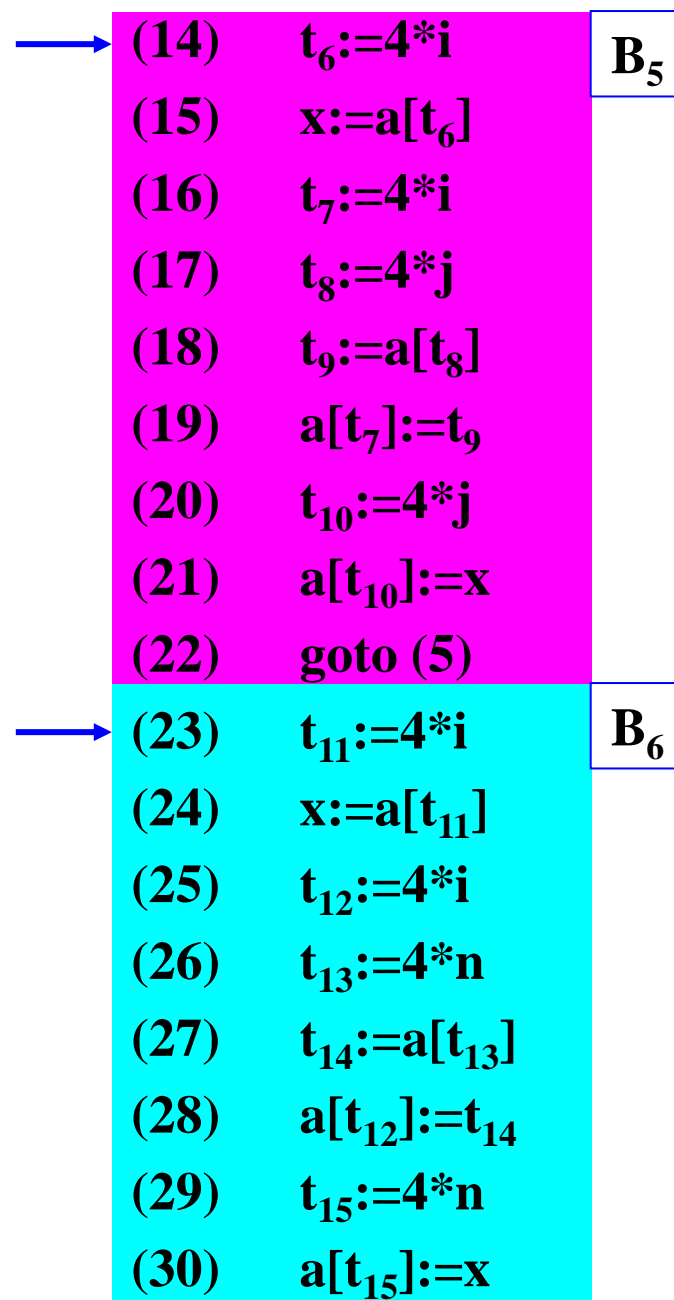
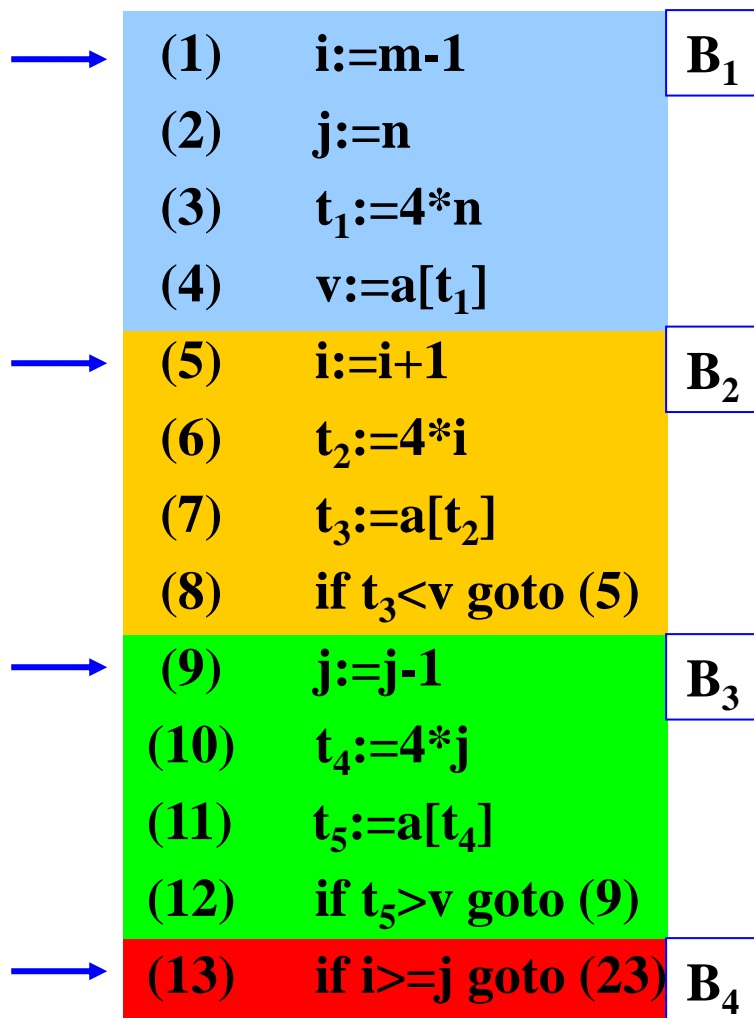
# 流图

- 把控制信息加到基本块集合中，形成程序的有向图，称为**流图**（控制流图）。
- **结点**：基本块
- **首结点**：第一条语句开始的基本块。
- 如果基本块 $B_2$ 紧跟在基本块 $B_1$ 之后执行，则从 $B_1$ 到 $B_2$ 有一条有向边， $B_1$ 是 $B_2$ 的**前驱**， $B_2$ 是 $B_1$ 的**后继**。  
即如果：
  - 有一个条件/无条件转移语句从 $B_1$ 的最后一句转移到 $B_2$ 的第一句；
  - $B_1$ 的最后一句不是转移语句，并且在程序的语句序列中， $B_2$ 紧跟在 $B_1$ 之后。

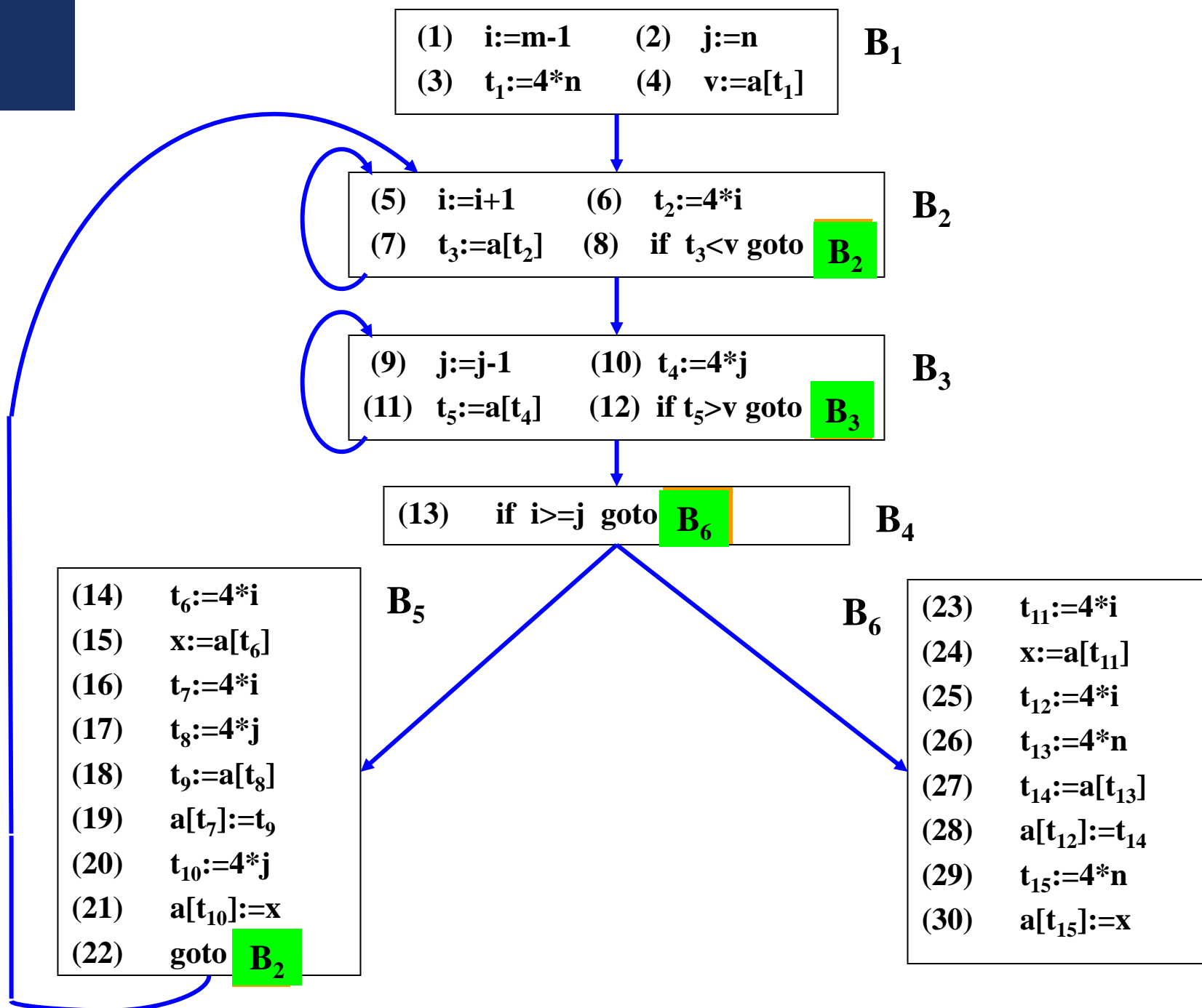




# 基本块划分示例



# 流图



## 9.3 基本块优化

1. 常数合并及常数传播
2. 删除公共表达式
3. 复制传播
4. 削弱计算强度
5. 改变计算次序

# 1. 常数合并及常数传播

- 常数合并：将在编译时可计算出值的表达式用其值替代。

$x=2+3+y$  可代之以:  $x=5+y$

- 常数传播：用在编译时已知的变量值代替程序正文中对这些变量的引用。

$PI:=3.14;$

$D\text{-}to\text{-}R:= 0.01744$

- 可否跨越基本块？

$i:=0$
10: $i:=i+1$
...
if $i<10$ goto 10



$i:=0$
10: $i:= 1$
...
if $i<10$ goto 10

...
$a[i]:=9.0$
...
$a[j]:=3.0$
$b:=a[i]$

# 常数合并的实现

- 在符号表中增加两个信息域
  - 标志域：指示当前该变量的值是否存在。
  - 常数域：如果变量值存在，则该域存放的即是该变量的当前值。
- 常数合并时，注意事项：
  - 不能将结合律与交换律用于浮点表达式。
    - 浮点运算的精度有限，这两条定律并非是恒真的。
  - 不应将任何附加的错误引入。

## 2. 删除公共表达式

- 在一个基本块中，当第一次对表达式E求值之后，如果E中的运算对象都没有改变，再次对E求值，则除E的第一次出现之外，其余的都是冗余的公共表达式。
- 删除冗余的公共表达式，用第一次出现时的求值结果代替之。

(1) **a:=b+c**

(2) **b:=a-d**

(3) **c:=b+c**

(4) **d:= b**

(4) **t<sub>1</sub>:=a-4**

(5) **t<sub>2</sub>:=4\*i**

(6') **t<sub>3</sub>:=t<sub>1</sub>**

(7') **t<sub>4</sub>:=t<sub>2</sub>**

(8) **t<sub>5</sub>:=t<sub>3</sub>[t<sub>4</sub>]**

(9) **t<sub>6</sub>:=b-4**

(10') **t<sub>7</sub>:=t<sub>2</sub>**

(11) **t<sub>8</sub>:=t<sub>6</sub>[t<sub>7</sub>]**

(12) **t<sub>9</sub>:=t<sub>5</sub>+t<sub>8</sub>**

(13) **t<sub>1</sub>[t<sub>2</sub>] := t<sub>9</sub>**

(14) **t<sub>10</sub>:=i+1**

(15) **i:=t<sub>10</sub>**

(16) **goto B2**

### 3. 复制传播

- 在复制语句  $f:=g$  之后，尽可能用  $g$  代替  $f$ 。

(4)  $t_1:=a-4$

(5)  $t_2:=4*i$

(8')  $t_5:=t_1[t_2]$

(9)  $t_6:=b-4$

(11')  $t_8:=t_6[t_2]$

(12)  $t_9:=t_5+t_8$

(13)  $t_1[t_2]:=t_9$

(15')  $i:=i+1$

(16) goto B2

删除死代码！

# 删除死代码

- 死代码：如果对一个变量  $x$  求值之后却不引用它的值，则称对  $x$  求值的代码为死代码。
- 死块：控制流不可到达的块称为死块。
  - 如果一个基本块是在某一条件为真时进入执行的，经数据流分析的结果知该条件恒为假，则此块是死块。
  - 如果一个基本块是在某个条件为假时才进入执行，而该条件却恒为真，则这个块也是死块。
- 在确定一个基本块是死块之前，需要检查转移到该块的所有转移语句的条件。
- 死块的删除，可能使其后继块成为无控制转入的块，这样的块也成为死块，同样应该删除。



## 4. 削弱计算强度

- 对基本块的代数变换：对表达式中的求值计算用代数上等价的形式替换，以便使复杂的运算变换成为简单的运算。

$x := y ** 2$

可以用代数上等价的乘式（如：  $x := y * y$ ）代替

- $x := x + 0$  和  $x := x * 1$

- 执行的运算没有任何意义
- 应将这样的语句从基本块中删除。

## 5. 改变计算次序

- 考虑语句序列：

$t_1 := b + c$

$t_2 := x + y$

- 如果这两个语句是互不依赖的，即  $x$ 、 $y$  均不为  $t_1$ ， $b$ 、 $c$  均不为  $t_2$ ，则交换这两个语句的位置不影响基本块的执行结果。
- 对基本块中的临时变量重新命名不会改变基本块的执行结果。

如：语句  $t := b + c$

改成语句  $u := b + c$

把块中出现的所有  $t$  都改成  $u$ ，不改变基本块的值。



## 9.4 循环优化

- 为循环语句生成的中间代码包括如下4部分：
  - **初始化部分**：对循环控制变量及其他变量赋初值。此部分组成的基本块位于循环体语句之前，可视为构成循环的第一个基本块。
  - **测试部分**：测试循环控制变量是否满足循环终止条件。这部分的位置依赖于循环语句的性质，若循环语句允许循环体执行0次，则在执行循环体之前进行测试；若循环语句要求循环体至少执行1次，则在执行循环体之后进行测试。
  - **循环体**：由需要重复执行的语句构成的一个或多个基本块组成。
  - **调节部分**：根据步长对循环控制变量进行调节，使其增加或减少一个特定的量。可把这部分视为构成该循环的最后一个基本块。
- 循环结构中的**调节部分**和**测试部分**也可以与**循环体**中的其他语句一起出现在基本块中。

# 循环优化的主要技术

1. 循环展开
2. 代码外提/频度削弱
3. 削弱计算强度
4. 删除归纳变量

# 1. 循环展开

- 以空间换时间的优化过程。
  - 循环次数在编译时可以确定
  - 针对每次循环生成循环体（不包括调节部分和测试部分）的一个副本。
- 进行循环展开的条件：
  - 识别出循环结构，而且编译时可以确定循环控制变量的初值、终值、以及变化步长。
  - 用空间换时间的权衡结果是可以接受的。
- 在重复产生代码时，必须确保每次重复产生时，都对循环控制变量进行了正确的合并。

# 示例:

假定: `int x[10];`  
其存储空间基址: `x`

■ 语句: `for (i=0; i<10; i++)`  
`x[i]=0;`

■ 生成三地址代码:

```
100: i:=0
101: if i<10 goto 103
102: goto 108
103: t1:=4*i
104: x[t1]:=0
105: t2:=i+1
106: i:=t2
107: goto 101
108: ...
```

■ 循环展开:

```
100: x[0]:=0
101: x[4]:=0
102: x[8]:=0
103: x[12]:=0
104: x[16]:=0
105: x[20]:=0
106: x[24]:=0
107: x[28]:=0
108: x[32]:=0
109: x[36]:=0
```

8 10

空间?  
执行时间?

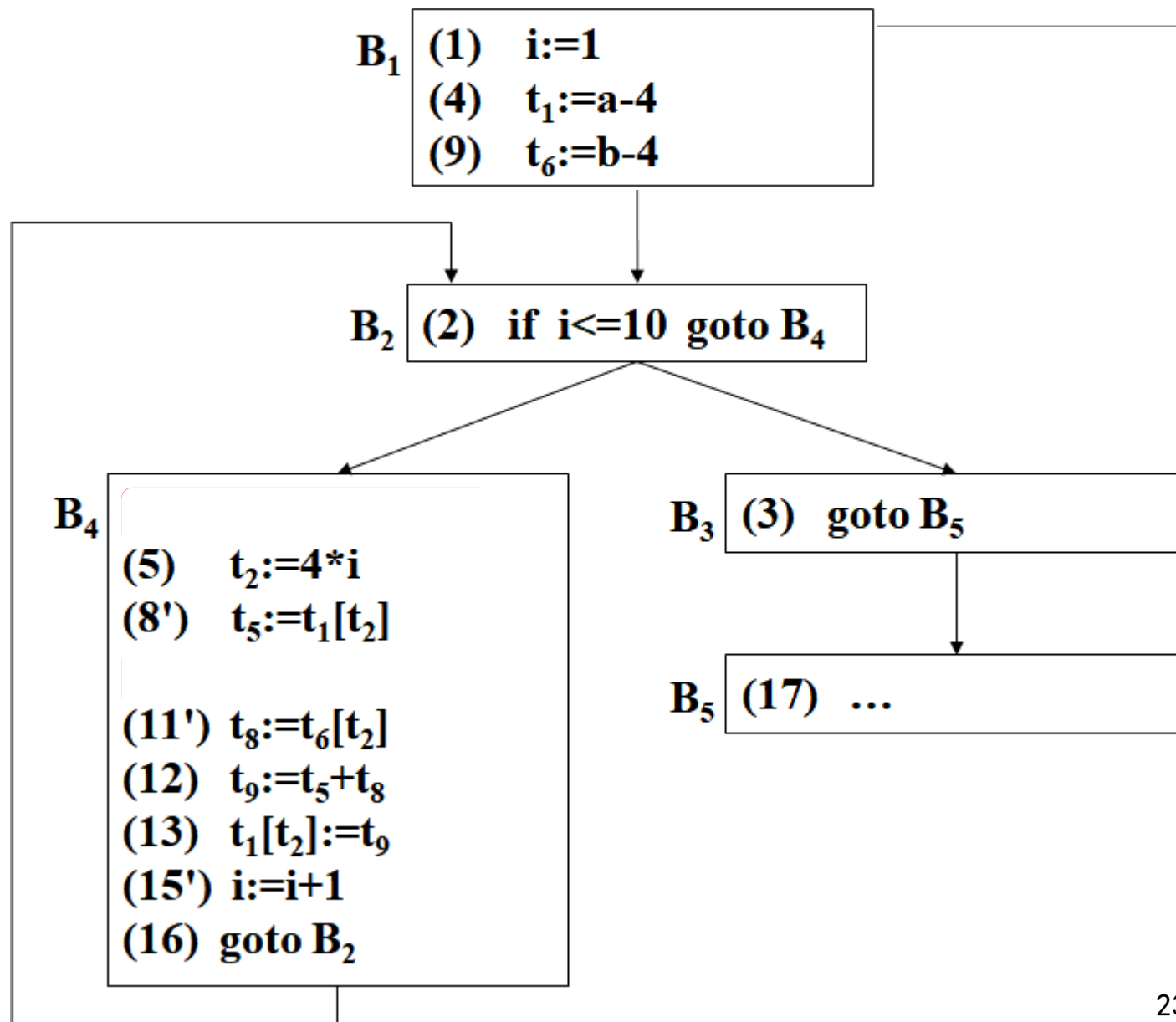
63 10

## 2. 代码外提/频度削弱

- 降低计算频度
- 将循环结构中的循环无关代码提到循环结构的前面, 减少循环中的代码总数。
- 如C语言程序中的语句:  

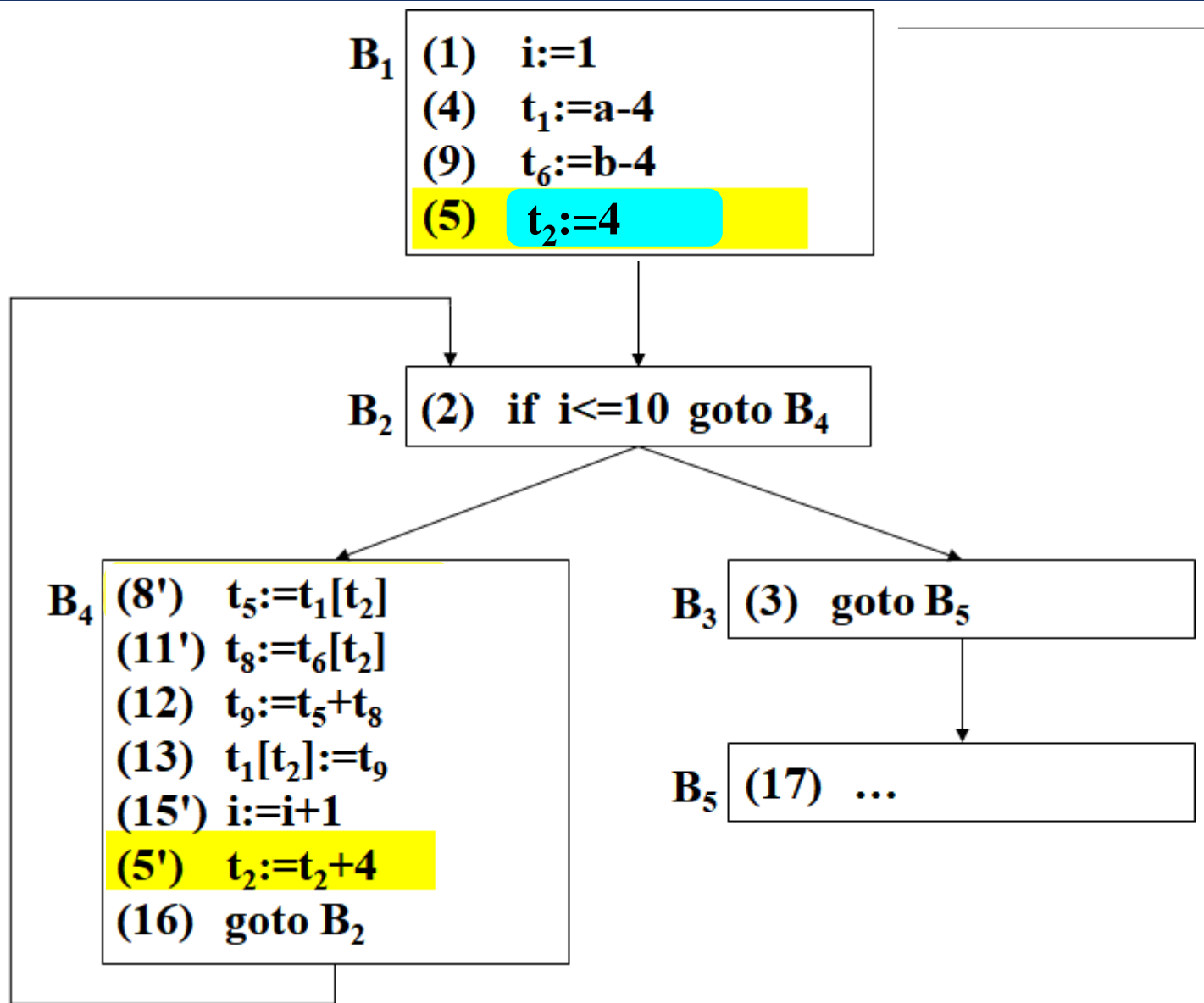
```
while (i<= limit-2) {  
    ...  
}
```

```
t:=limit-2;  
while (i<=t) {  
    ...  
}
```
- 若在循环中limit的值保持不变, 则limit-2的计算与循环无关。



### 3. 削弱计算强度

- 将当前运算类型代之以需要较少执行时间的运算类型的优化方法。
- 大多数计算机上乘法运算比加法运算需要更多的执行时间。
- 如可用 '+' 代替 '\*', 则可节省许多时间, 特别是当这种替代发生在循环中时更是如此。





## 4. 删除归纳变量

### ■ 基本归纳变量:

- 如果循环中对变量  $i$  只有唯一的形如  $i:=i+c$  的赋值, 且  $c$  为循环不变量, 则称  $i$  为循环的基本归纳变量。

### ■ 同族归纳变量:

- 如果  $i$  是循环的一个基本归纳变量,  $j$  是  $i$  的线性函数, 即  $j:=c_1*i+c_2$ , 这里  $c_1$  和  $c_2$  都是循环不变量, 则称  $j$  是归纳变量, 并称  $j$  与  $i$  同族。

$B_4$

- (8')  $t_5:=t_1[t_2]$
- (11')  $t_8:=t_6[t_2]$
- (12)  $t_9:=t_5+t_8$
- (13)  $t_1[t_2]:=t_9$
- (15')  $i:=i+1$
- (5')  $t_2:=t_2+4$
- (16) goto  $B_2$

### ■ 如: 基本块 $B_4$ 中

- $i$  是基本归纳变量
- $t_2:=4*i$
- $t_2$  是与  $i$  同族的归纳变量

# 删除归纳变量

- 通常，一个基本归纳变量除用于其自身的递归定值外，往往只用于计算其他归纳变量的值、以及用来控制循环的进行。

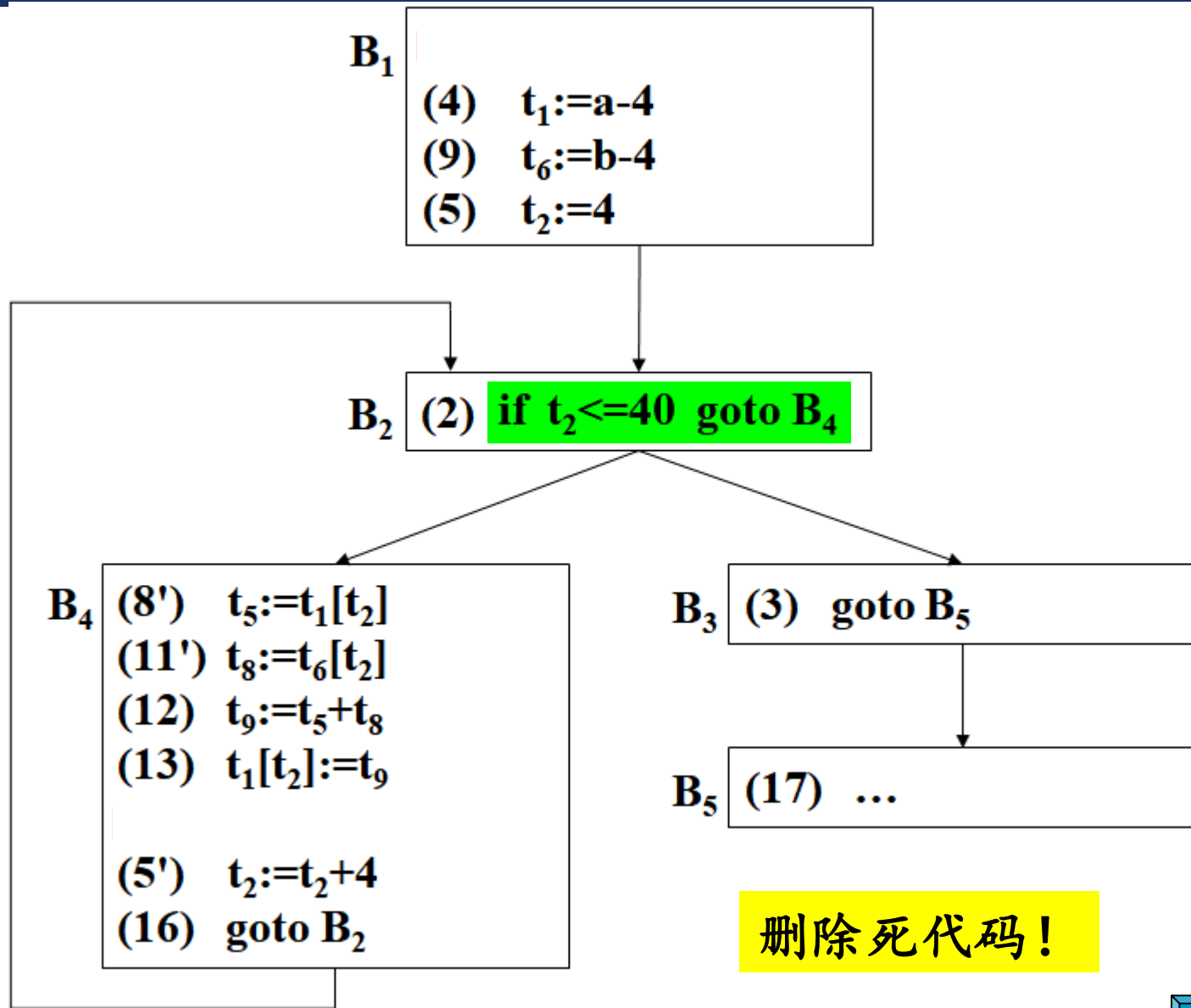
- 由于 $t_2$ 和 $i$ 之间具有线性函数关系： $t_2 = 4 * i$

$i \leq 10$  与  $t_2 \leq 40$  等价。

可用  $t_2 \leq 40$  替换  $i \leq 10$

语句(2)变换为：

if  $t_2 \leq 40$  goto  $B_4$



# 本章小结

## ■ 代码优化程序的功能

- 等价变换
- 执行时间
- 占用空间

## ■ 优化种类

- 基本块优化
- 循环优化
- 窥孔优化

## ■ 基本块与流图

- 基本块划分
- 流图构造

## ■ 基本块优化的主要技术

- 常数合并与常数传播
- 删除冗余的公共表达式
- 复制传播
- 删除死代码
- 削弱计算强度
- 改变计算次序

## ■ 循环优化的主要技术

- 循环展开
- 代码外提/频度削弱
- 削弱计算强度
- 删除归纳变量

# 学习任务

## ■ 作业要求

- 基本块划分与流图构造
- 基本块优化
- 对循环结构生成的中间代码进行优化

## ■ 研究性学习

- 控制流分析
- 数据流分析

