编译原理与技术

第9章 中间代码优化

wenshli@bupt.edu.cn

李文生

2024年2月19日 星期一

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

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

内容目录

wenshli@bupt.edu.cn

中间代码优化考题示例

六、（15 分）有如下三地址代码：

```
1   i:=1
2   if i<=15 goto 4
3   goto 14
4   t1:=a-8
5   t2:=4*i
6   t3:=t1[t2]
7   t4:=b-8
8   t5:=t4[t2]
9   t6:=t3*t5
10  t1[t2]:=t6
11  t7:=i+1
12  i:=t7
13  goto 2
14  halt
```

（1） 将它划分基本块，并构造其流图。

（2） 在这段代码上进行循环优化，给出优化后的三地址代码。

六、（15 分）有赋值语句：a = 3*2 + x/(m+n) - y + (m+n);

其中：m、n 为整数类型，a、x、y 为实数类型

（1）将该语句翻译为语法树；

（2）将该语句翻译为三地址代码；

（3）对三地址代码进行优化。

五、（15 分）有算术表达式：3*6+(a-b)/(a-b-(a-b*c))

（1）将该表达式翻译为语法树。

（2）将该表达式翻译为后缀表达式。

（3）将该表达式翻译为三地址代码，并对三地址代码进行优化。

wenshli@bupt.edu.cn

考点1：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。

wenshli@bupt.edu.cn

Pascal程序片断:

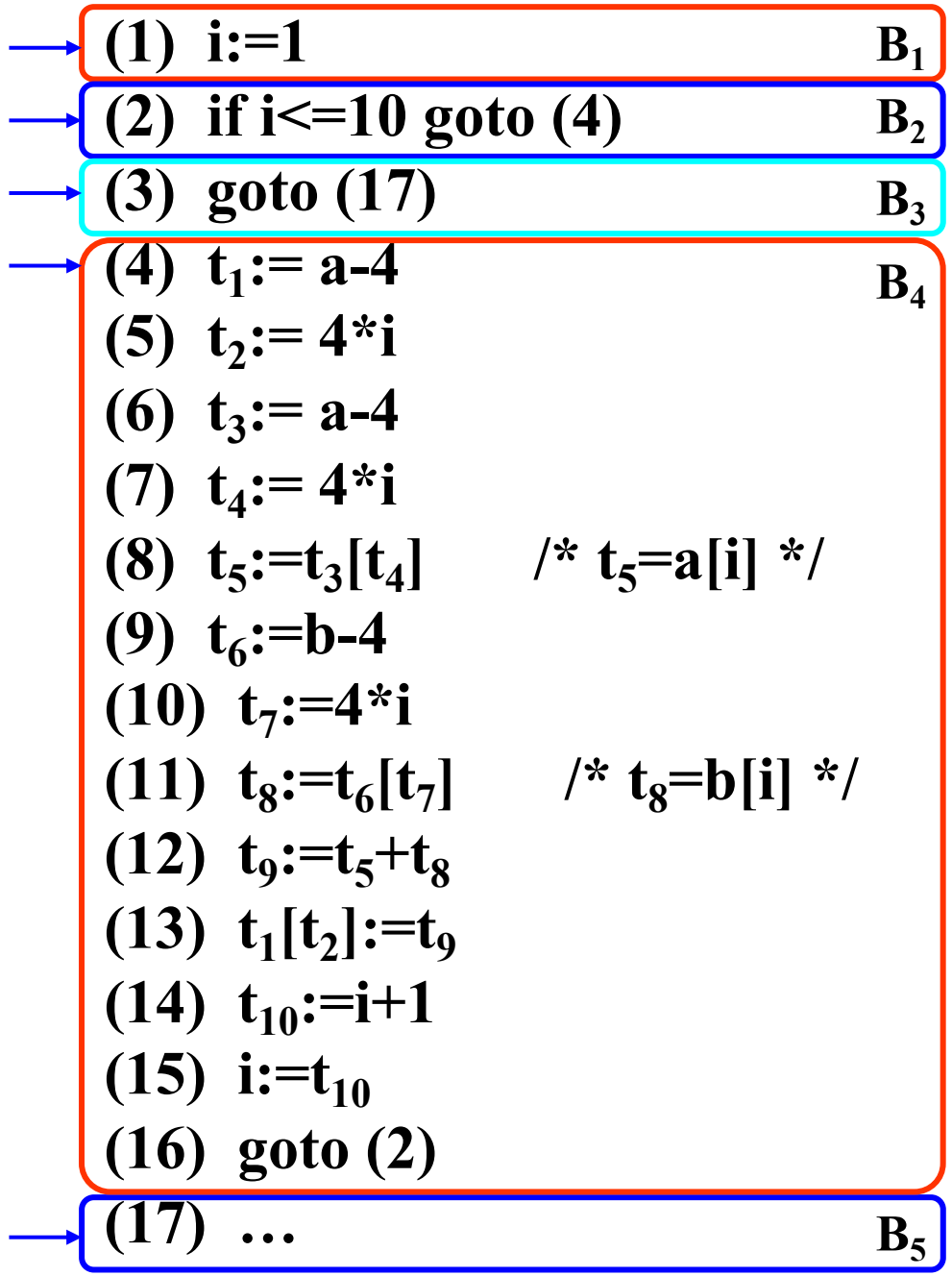
```
i:=1;
while (i<=10) do
begin
    a[i]:=a[i]+b[i];
    i:=i+1
end;
```

(1) i:=1
(2) if i<=10 goto (4)
(3) goto (17)
(4) t_1:= a-4
(5) t_2:= 4*i
(6) t_3:= a-4
(7) t_4:= 4*i
(8) t_5:=$t_3[t_4]$ /* t_5=a[i] */
(9) t_6:=b-4
(10) t_7:=4*i
(11) t_8:=$t_6[t_7]$ /* t_8=b[i] */
(12) t_9:=t_5+t_8
(13) $t_1[t_2]$:=t_9
(14) t_{10}:=i+1
(15) i:=t_{10}
(16) goto (2)
(17) …

wenshli@bupt.edu.cn

Pascal程序片断:（答案）

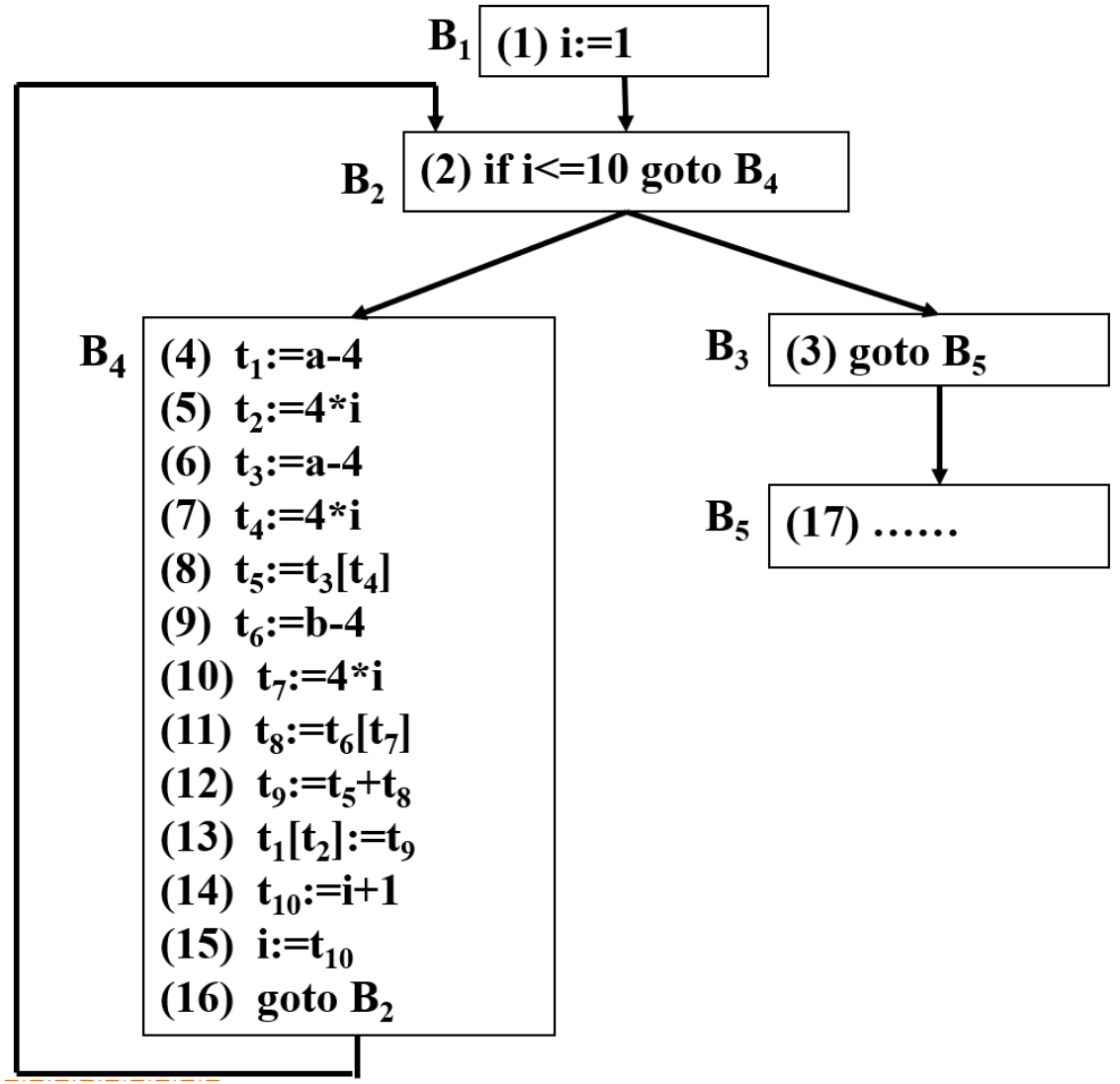
```
i:=1;
while (i<=10) do
begin
    a[i]:=a[i]+b[i];
    i:=i+1
end;
```



wenshli@bupt.edu.cn

考点2: 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的最后一条语句转移到B_2的第一条语句；
  - B_1的最后一条语句不是转移语句，并且在程序的语句序列中，B_2紧跟在B_1之后。



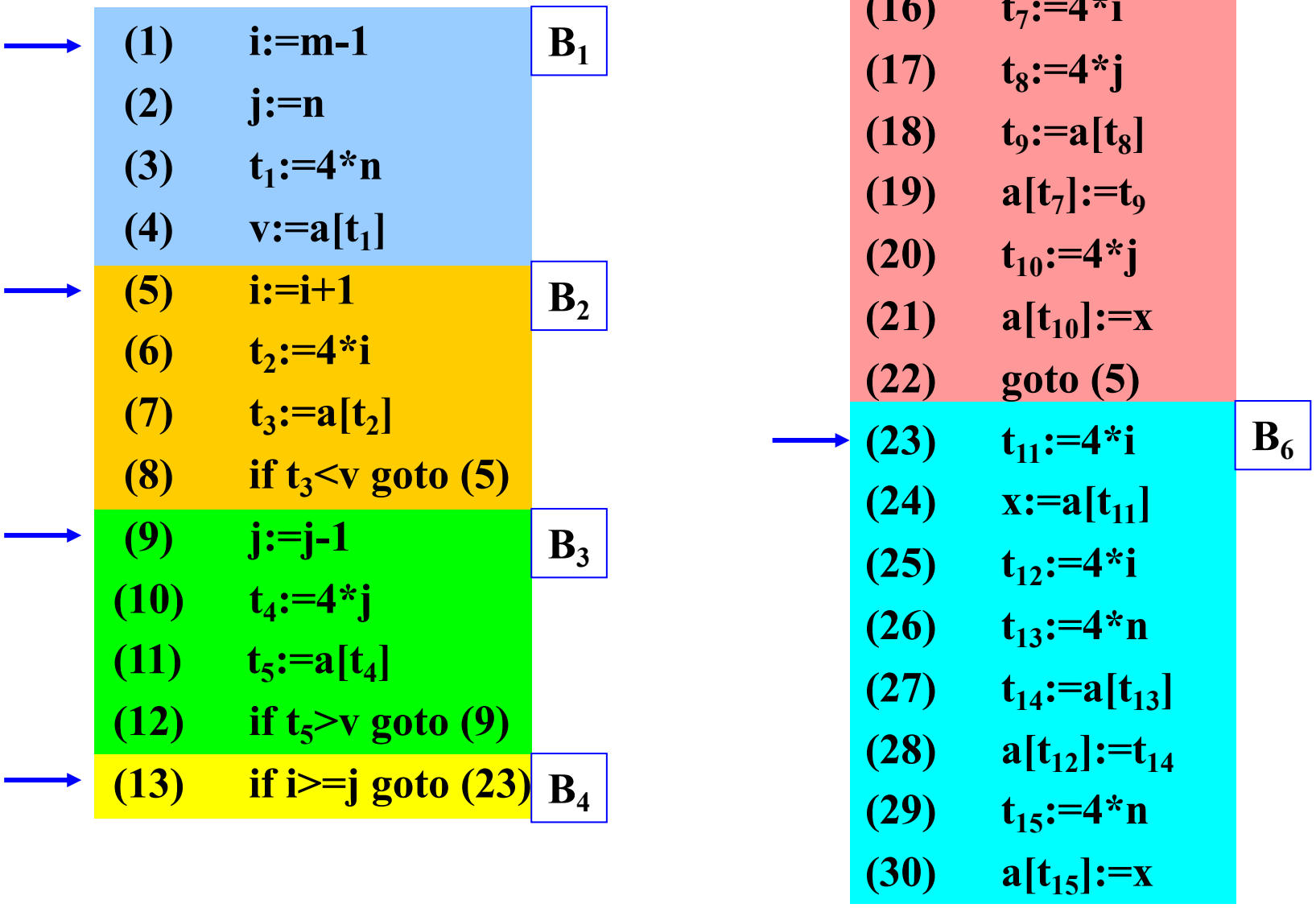
wenshli@bupt.edu.cn

基本块划分与流图示例

(1) i:=m-1
(2) j:=n
(3) t_1:=4*n
(4) v:=a[t_1]
(5) i:=i+1
(6) t_2:=4*i
(7) t_3:=a[t_2]
(8) if t_3<v goto (5)
(9) j:=j-1
(10) t_4:=4*j
(11) t_5:=a[t_4]
(12) if t_5>v goto (9)
(13) if i>=j goto (23)
(14) t_6:=4*i
(15) x:=a[t_6]
(16) t_7:=4*i
(17) t_8:=4*j
(18) t_9:=a[t_8]
(19) a[t_7]:=t_9
(20) t_{10}:=4*j
(21) a[t_{10}]:=x
(22) goto (5)
(23) t_{11}:=4*i
(24) x:=a[t_{11}]
(25) t_{12}:=4*i
(26) t_{13}:=4*n
(27) t_{14}:=a[t_{13}]
(28) a[t_{12}]:=t_{14}
(29) t_{15}:=4*n
(30) a[t_{15}]:=x

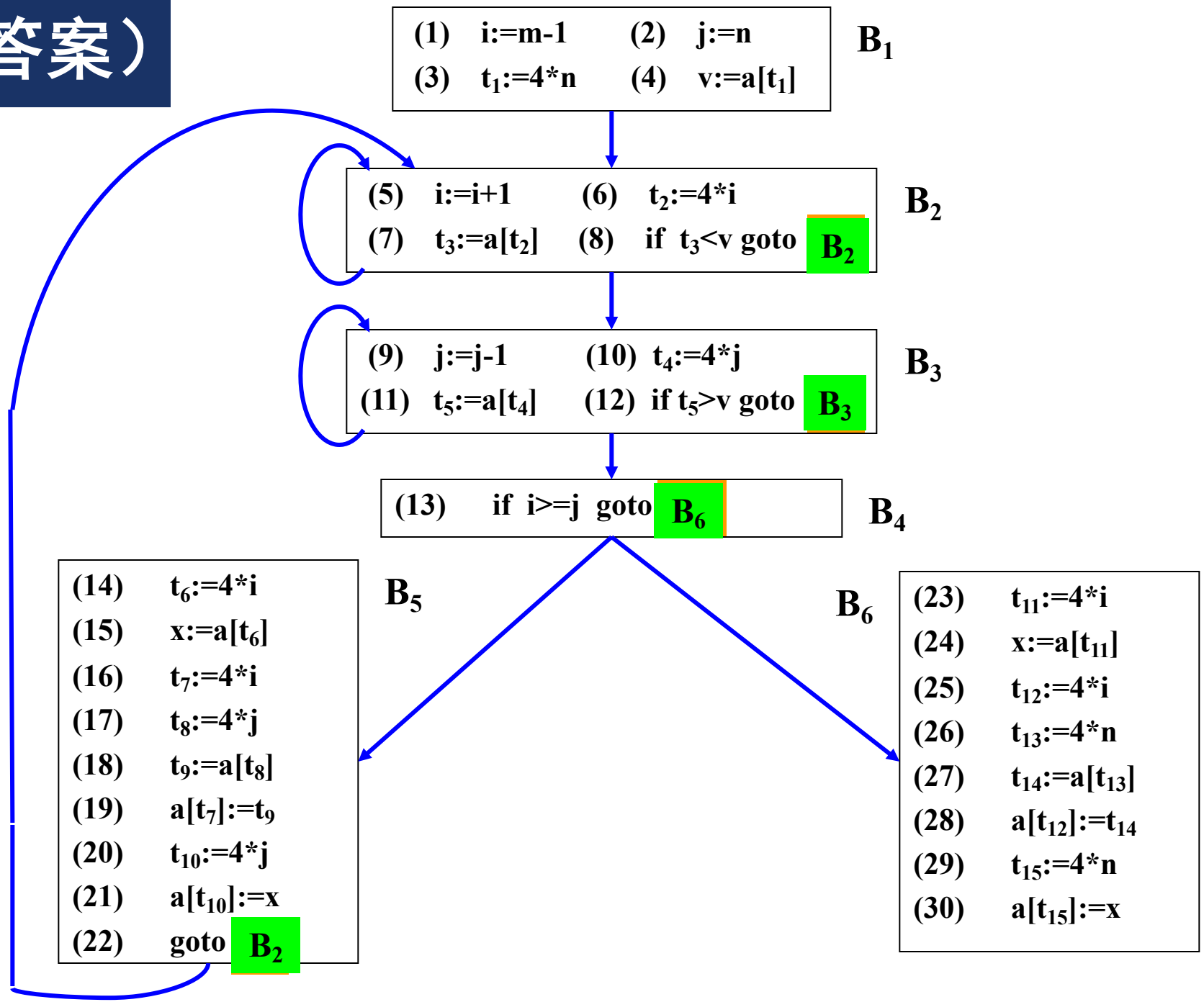
wenshli@bupt.edu.cn

基本块划分(答案)



wenshli@bupt.edu.cn

流图（答案）



wenshli@bupt.edu.cn

考点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

wenshli@bupt.edu.cn

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。

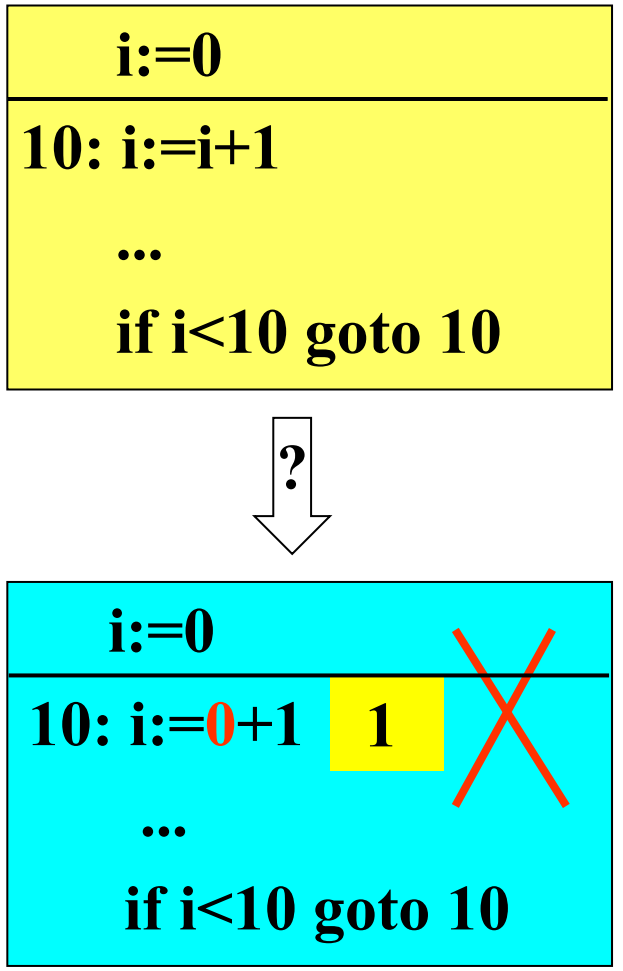
  PI:=3.14;

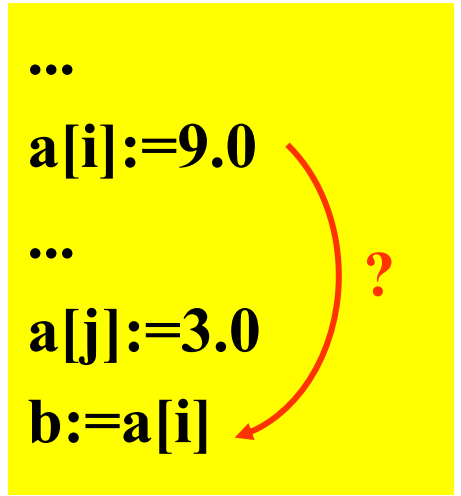
  D-to-R:= PI/180.0;

  3.14/180.0

  0.01744

- 可否跨越基本块？





wenshli@bupt.edu.cn

常数合并的实现

- 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。

wenshli@bupt.edu.cn

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:=a-d

b

(4) t_1:=a-4	
(5) t_2:=4*i	
(6) t_3:=a-4	(6') t_3:=t_1
(7) t_4:=4*i	(7') t_4:=t_2
(8) t_5:=t_3[t_4]	
(9) t_6:=b-4	
(10) t_7:=4*i	(10') t_7:=t_2
(11) t_8:=t_6[t_7]	
(12) t_9:=t_5+t_8	
(13) t_1[t_2]:=t_9	
(14) t_{10}:=i+1	
(15) i:=t_{10}	
(16) goto B2	

wenshli@bupt.edu.cn

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
(6') t_3:=t_1
(7') t_4:=t_2
(8) t_5:=t_3[t_4]
(9) t_6:=b-4
(10') t_7:=t_2
(11) t_8:=t_6[t_7]
(12) t_9:=t_5+t_8
(13) t_1[t_2]:=t_9
(14) t_{10}:=i+1
(15) i:=t_{10}
(16) goto B2

删除死代码！

(8') t_5:=t_1[t_2]

(11') t_8:=t_6[t_2]

(15') i:=i+1

wenshli@bupt.edu.cn

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。

wenshli@bupt.edu.cn

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:=y2**

  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。

wenshli@bupt.edu.cn

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。

wenshli@bupt.edu.cn

考点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。

wenshli@bupt.edu.cn

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

wenshli@bupt.edu.cn

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。

wenshli@bupt.edu.cn

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: t_1:=4*i
104: x[t_1]:=0
105: t_2:=i+1
106: i:=t_2
107: goto 101
108: …

8 10

空间？

执行时间？

63 10

- 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

wenshli@bupt.edu.cn

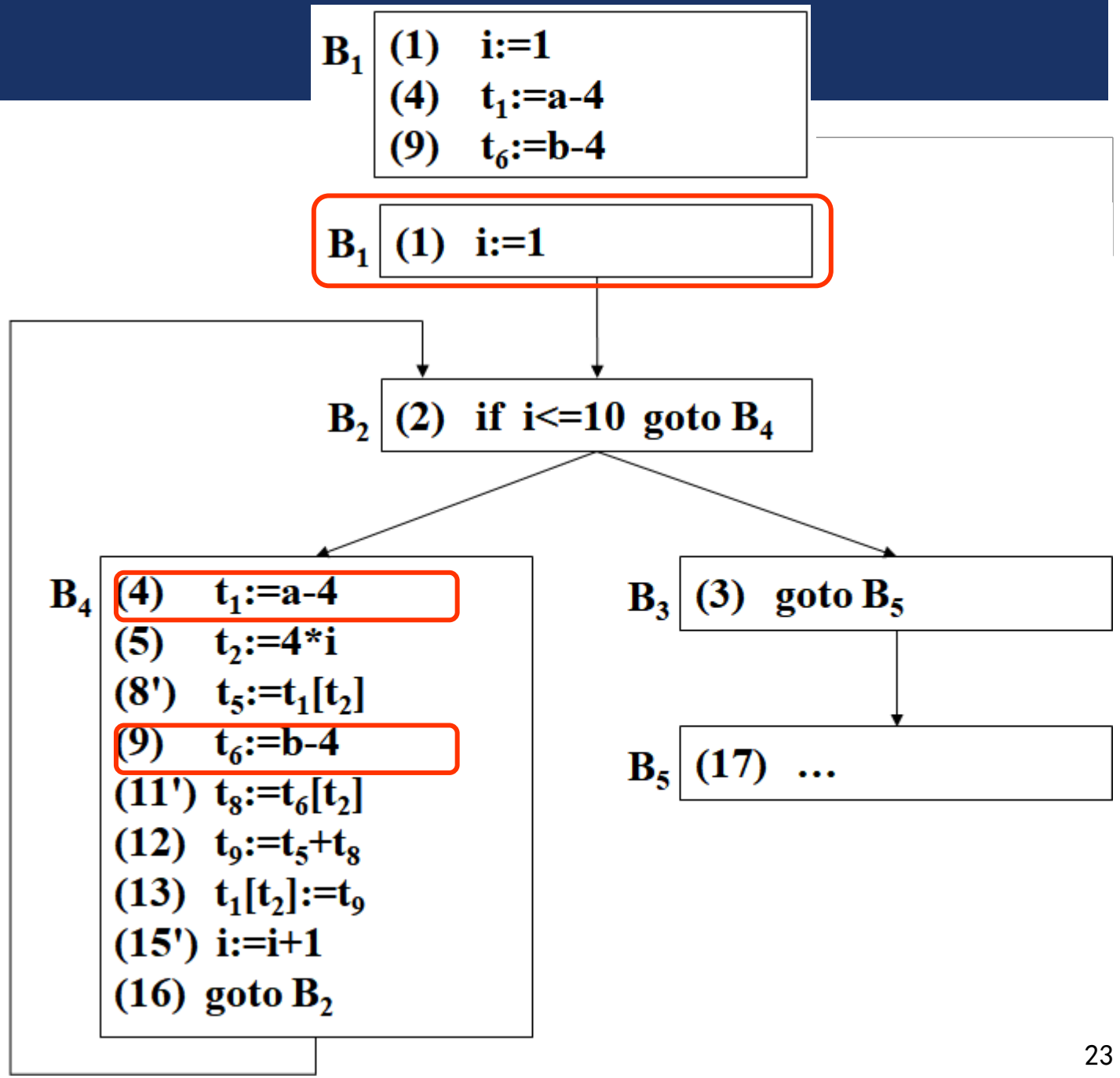
2. 代码外提/频度削弱

- 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) {
…
}
```

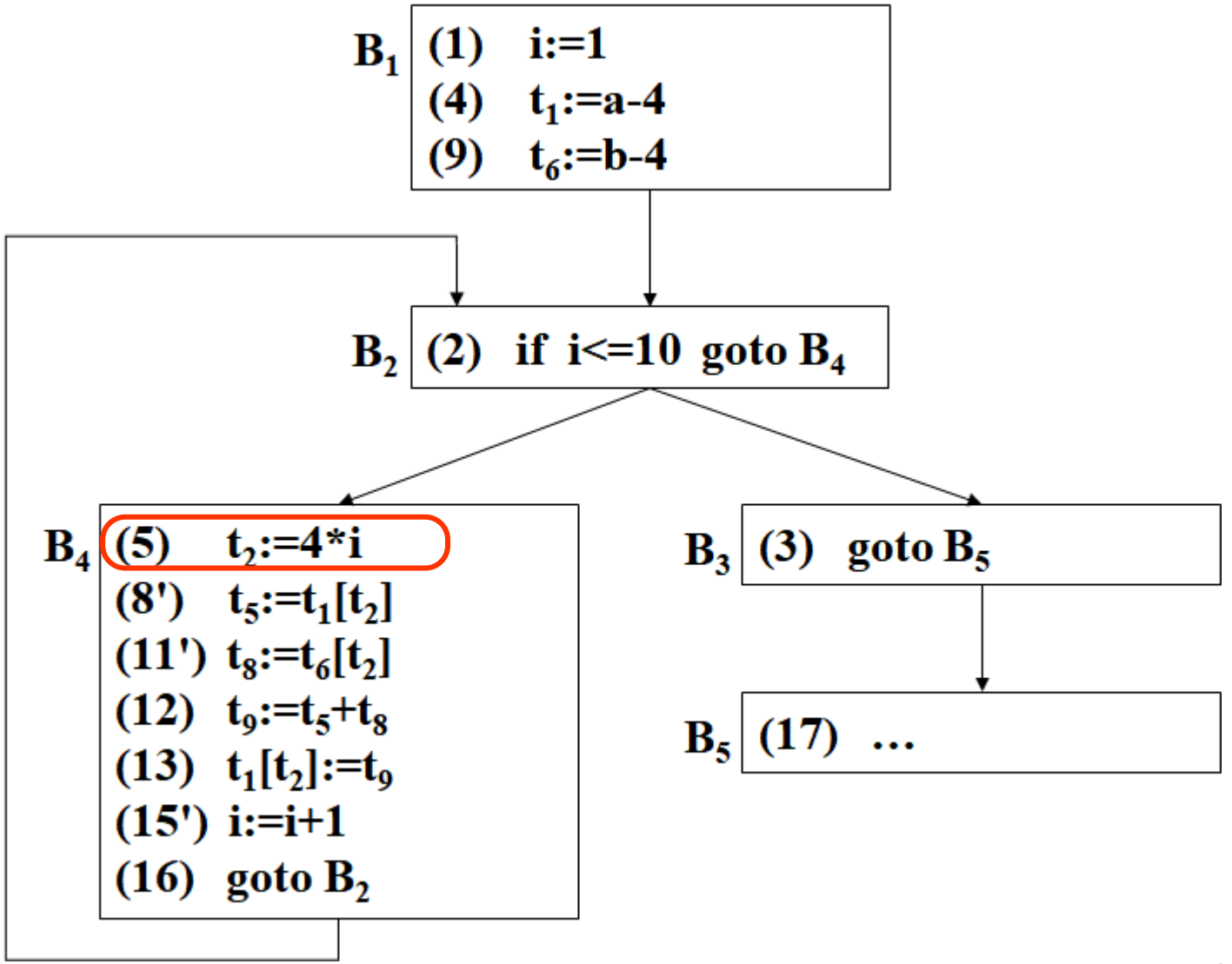
- 若在循环中limit的值保持不变，则limit-2的计算与循环无关。



wenshli@bupt.edu.cn

3. 削弱计算强度

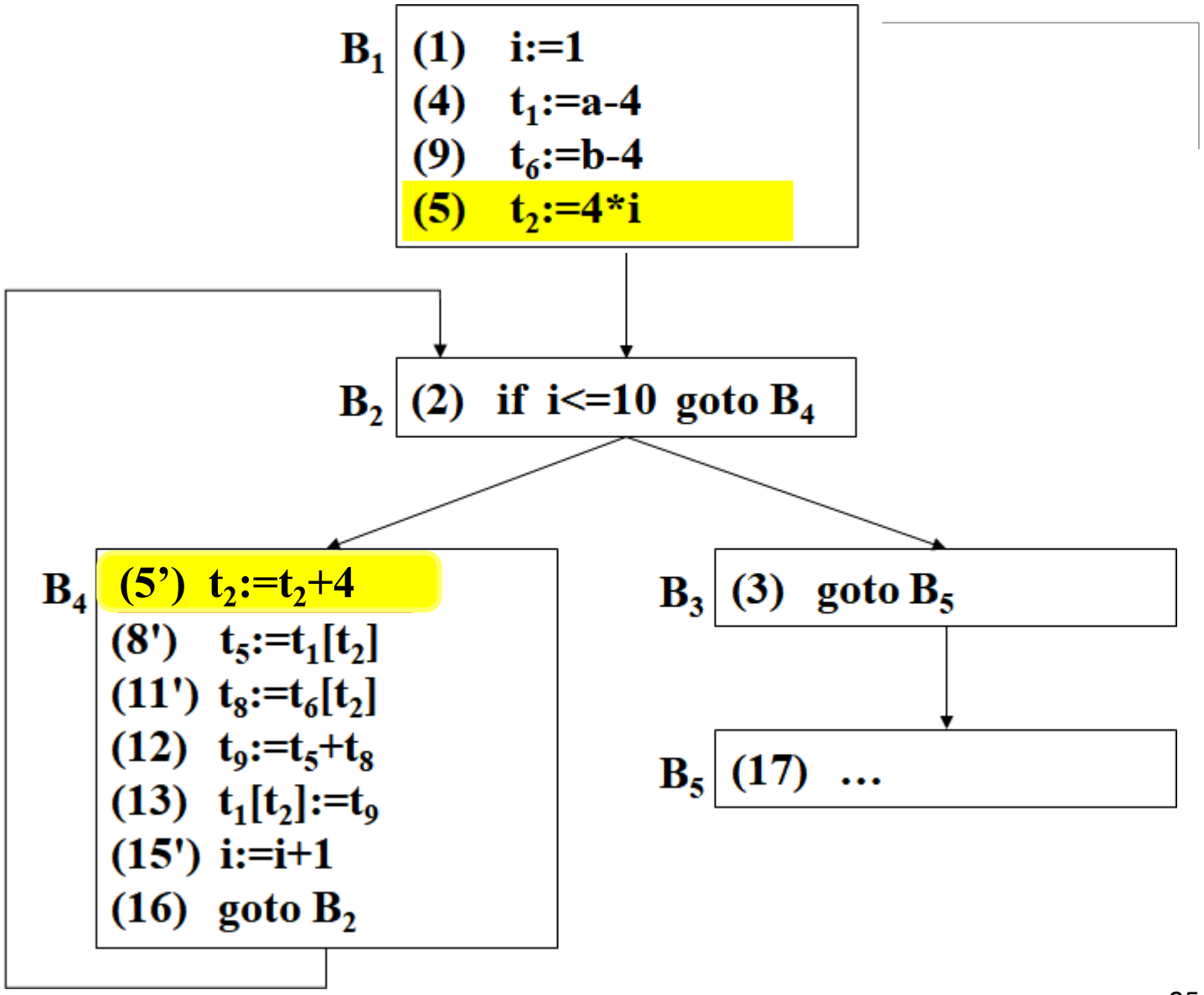
- 将当前运算类型代之以需要较少执行时间的运算类型的优化方法。
- 大多数计算机上乘法运算比加法运算需要更多的执行时间。
- 如可用'+'代替'*'，则可节省许多时间，特别是当这种替代发生在循环中时更是如此。



wenshli@bupt.edu.cn

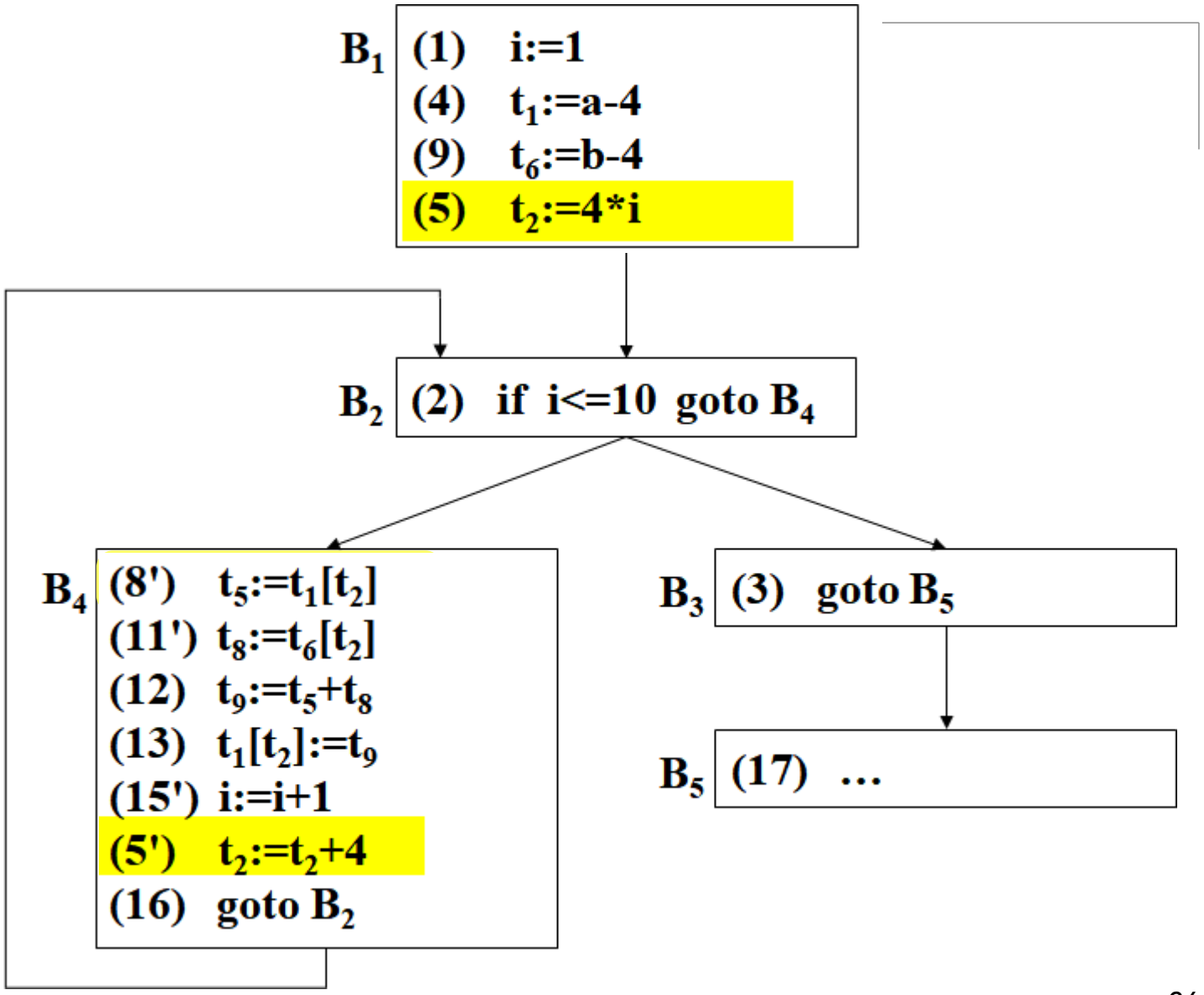
3. 削弱计算强度

- 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。



3. 削弱计算强度

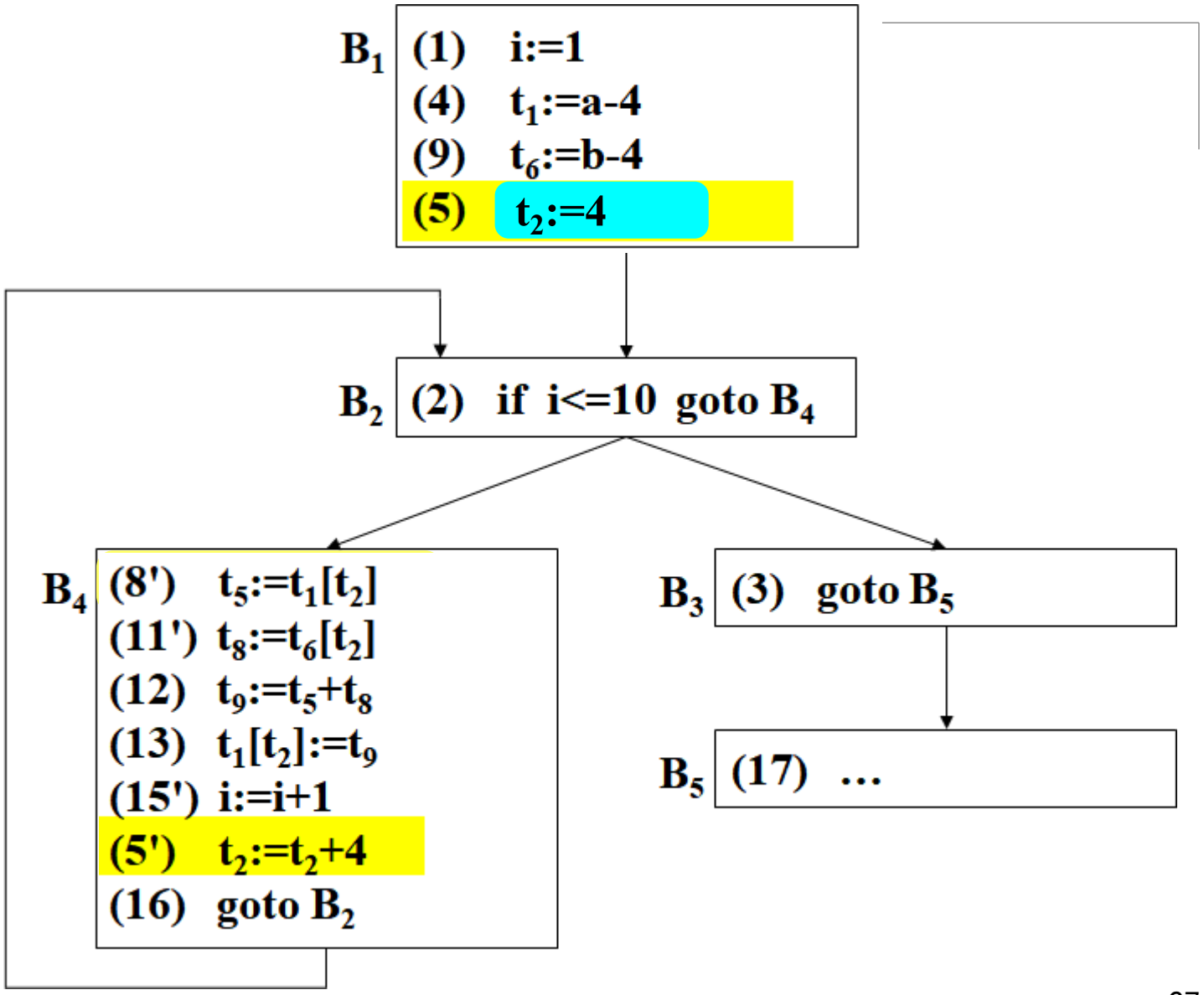
- 将当前运算类型代之以需要较少执行时间的运算类型的优化方法。
- 大多数计算机上乘法运算比加法运算需要更多的执行时间。
- 如可用'+'代替'*'，则可节省许多时间，特别是当这种替代发生在循环中时更是如此。



wenshli@bupt.edu.cn

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。



wenshli@bupt.edu.cn

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

wenshli@bupt.edu.cn

删除归纳变量

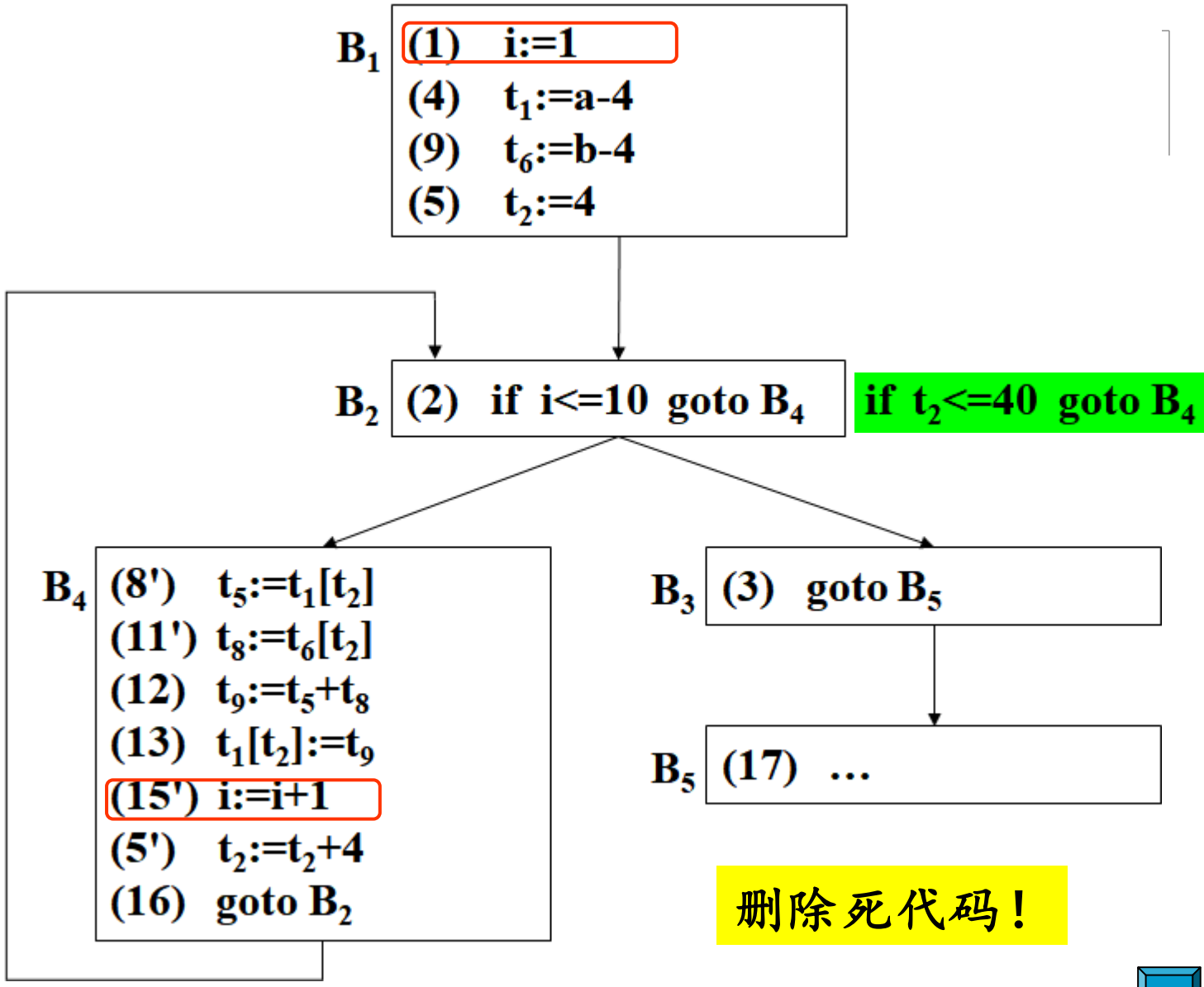
- 通常，一个基本归纳变量除用于其自身的递归定值外，往往只用于计算其他归纳变量的值、以及用来控制循环的进行。
- 由于t_2和i之间具有线性函数关系：$t_2=4*i$

  i<=10 与 t_2<=40 等价。

  可用 t_2<=40 替换 i<=10

  语句(2)变换为：

  if t_2<=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

wenshli@bupt.edu.cn

编译原理与技术

第10章 目标代码生成

wenshli@bupt.edu.cn

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

学习任务

- 作业要求
  - 目标代码生成算法应用

wenshli@bupt.edu.cn

示例：

- 赋值语句：

  x:=a+b*c-d

- 三地址代码：

  t:=b*c

  u:=a+t

  v:=u-d

  x:=v

- 假定
  - 寄存器R_0和R_1
  - 在块出口，x活跃

三地址语句	目标代码	寄存器描述器	地址描述器
		寄存器全空	a:Ma b: Mb c:Mc d:Md
t:=b*c	MOV R_0, b MUL R_0, c	R_0: t	t: R_0
u:=a+t	MOV R_1, a ADD R_1, R_0	R_0: t R_1: u	t: R_0 u: R_1
v:=u-d	SUB R_1, d	R_0: t R_1: v	t: R_0 u: v: R_1
x:=v		R_1: v, x	x: R_1
	MOV Mx, R_1		x: R_1, Mx

wenshli@bupt.edu.cn