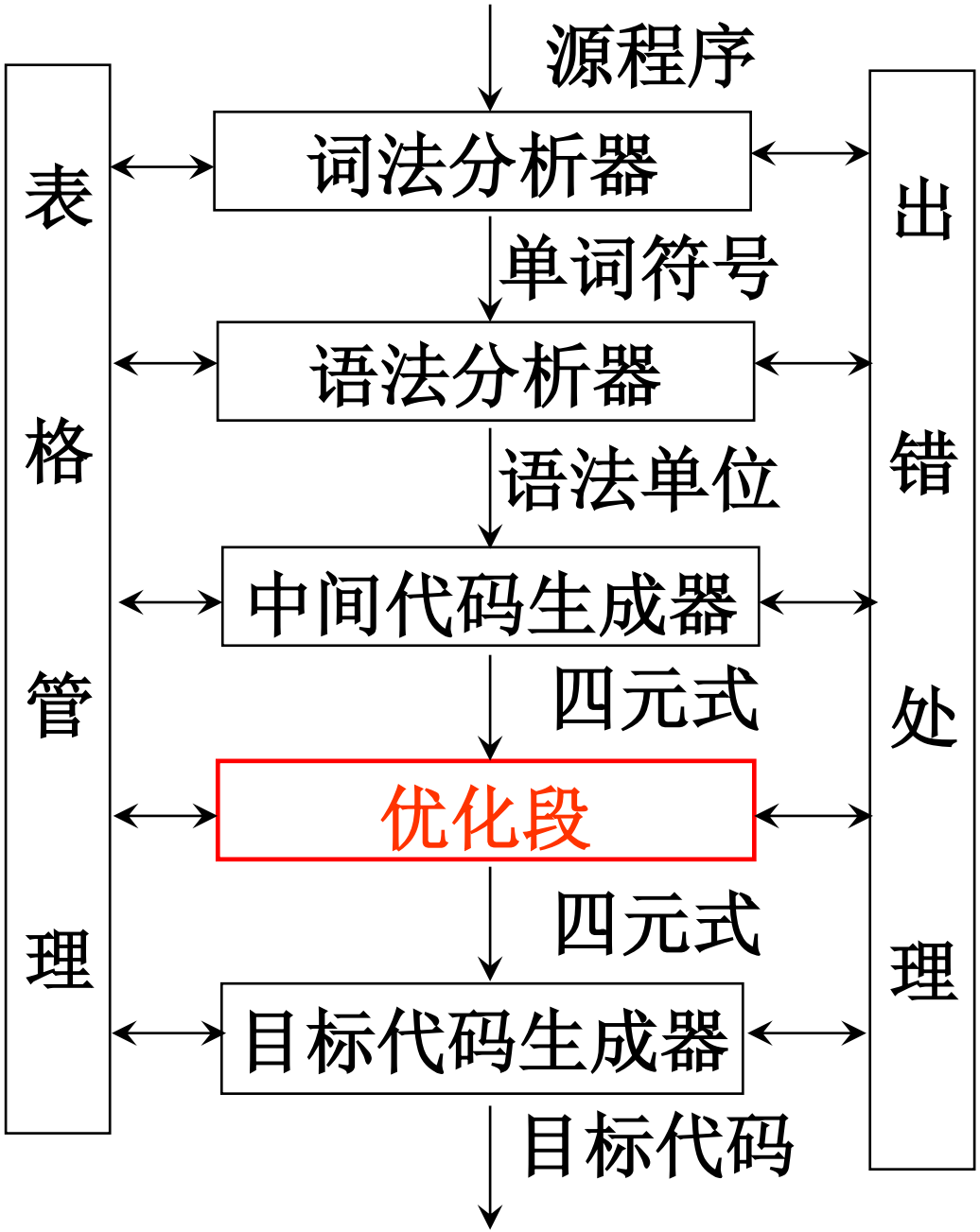
第十章 优化

同济大学计算机系

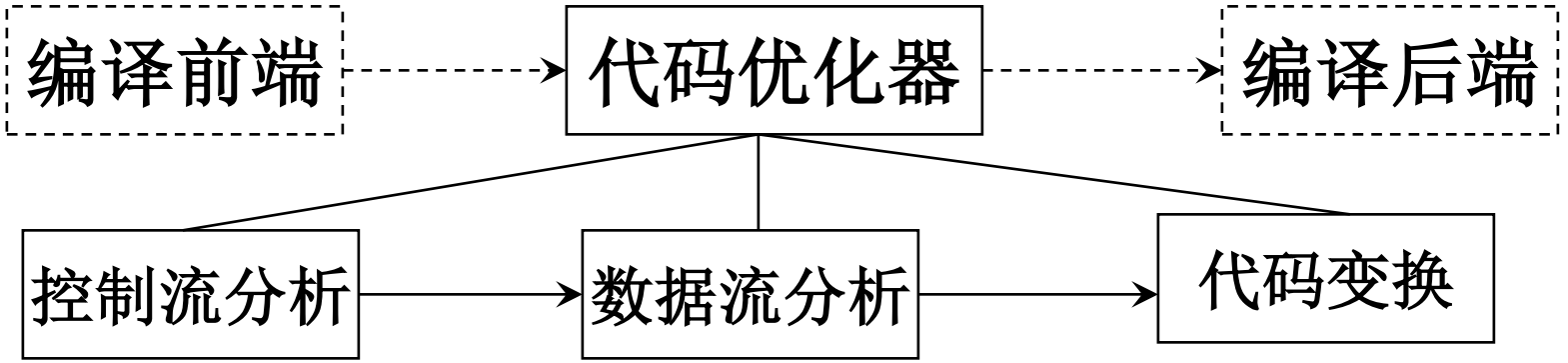

源程序
词法分析器
单词符号
语法分析器
语法单位
中间代码生成器
四元式
优化段
四元式
目标代码生成器
目标代码
表格管理
出错处理

问题的提出

- 编译程序的作用
  - 使计算机的使用方式从用机器语言编程发展到用高级语言编程。是计算机发展史上的一次飞跃。
- 早期编译程序的不足
  - 目标程序质量差
    - 占用的空间大
    - 运行的时间长

优化

- **优化：对程序进行各种等价变换，使得从变换后的程序出发，能生成更有效的目标代码。**



内容线索

- **概述**
- **局部优化**
- **循环优化**

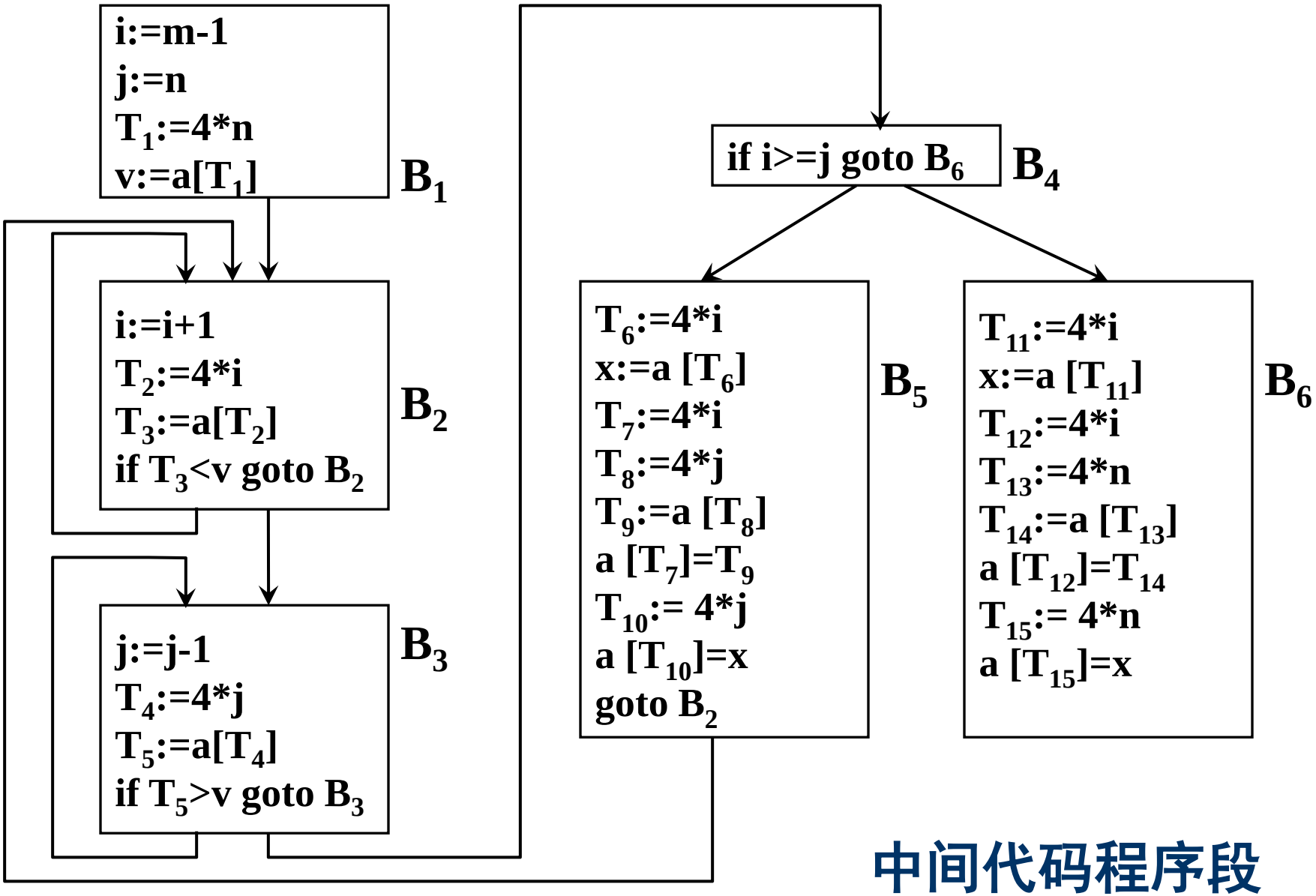
概述

- 优化的目的是为了产生更高效的代码。由优化编译程序提供的对代码的各种变换必须遵循一定的原则：
  - **等价**原则：经过优化后不应改变程序运行的结果；
  - **有效**原则：使优化后所产生的目标代码运行时间较短，占用的存储空间较小；
  - **合算**原则：应尽可能以较低的代价取得较好的优化效果。

概述

- 优化的三个不同级别：
  - 局部优化
  - 循环优化
  - 全局优化
- 优化的种类：
  - 删除多余运算（或称删除公用子表达式）
  - 代码外提
  - 强度消弱
  - 变换循环控制条件
  - 合并已知量
  - 复写传播
  - 删除无用赋值

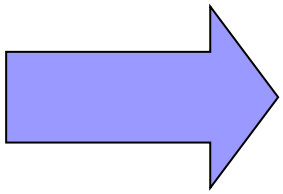
```
void quicksort (m, n);
int m, n;
    {
        int i, j;
        int v, x;
        if (n<=m) return;
        /* fragment begins here*/
        i=m-1; j=n; v=a [n];
        while (1) {
                do  i=i+1;  while (a [i]<v);
                do  j=j-1;   while (a [j]>v);
                if (i>=j) break;
                x=a [i]; a[i]=a [j]; a[j]=x;
         }
       x=a[i]; a[i]=a [n]; a [n]=x;
        /*fragment ends here*/
       quicksort (m, j); quicksort (i+1, n);
}
```


i:=m-1
j:=n
T1:=4*n
v:=a[T1]
B1
i:=i+1
T2:=4*i
T3:=a[T2]
if T3<v goto B2
B2
j:=j-1
T4:=4*j
T5:=a[T4]
if T5>v goto B3
B3
if i>=j goto B6
B4
T6:=4*i
x:=a [T6]
T7:=4*i
T8:=4*j
T9:=a [T8]
a [T7]=T9
T10:= 4*j
a [T10]=x
goto B2
B5
T11:=4*i
x:=a [T11]
T12:=4*i
T13:=4*n
T14:=a [T13]
a [T12]=T14
T15:= 4*n
a [T15]=x
B6

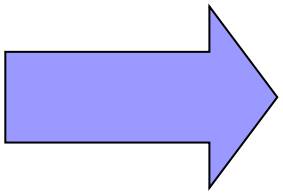
中间代码程序段

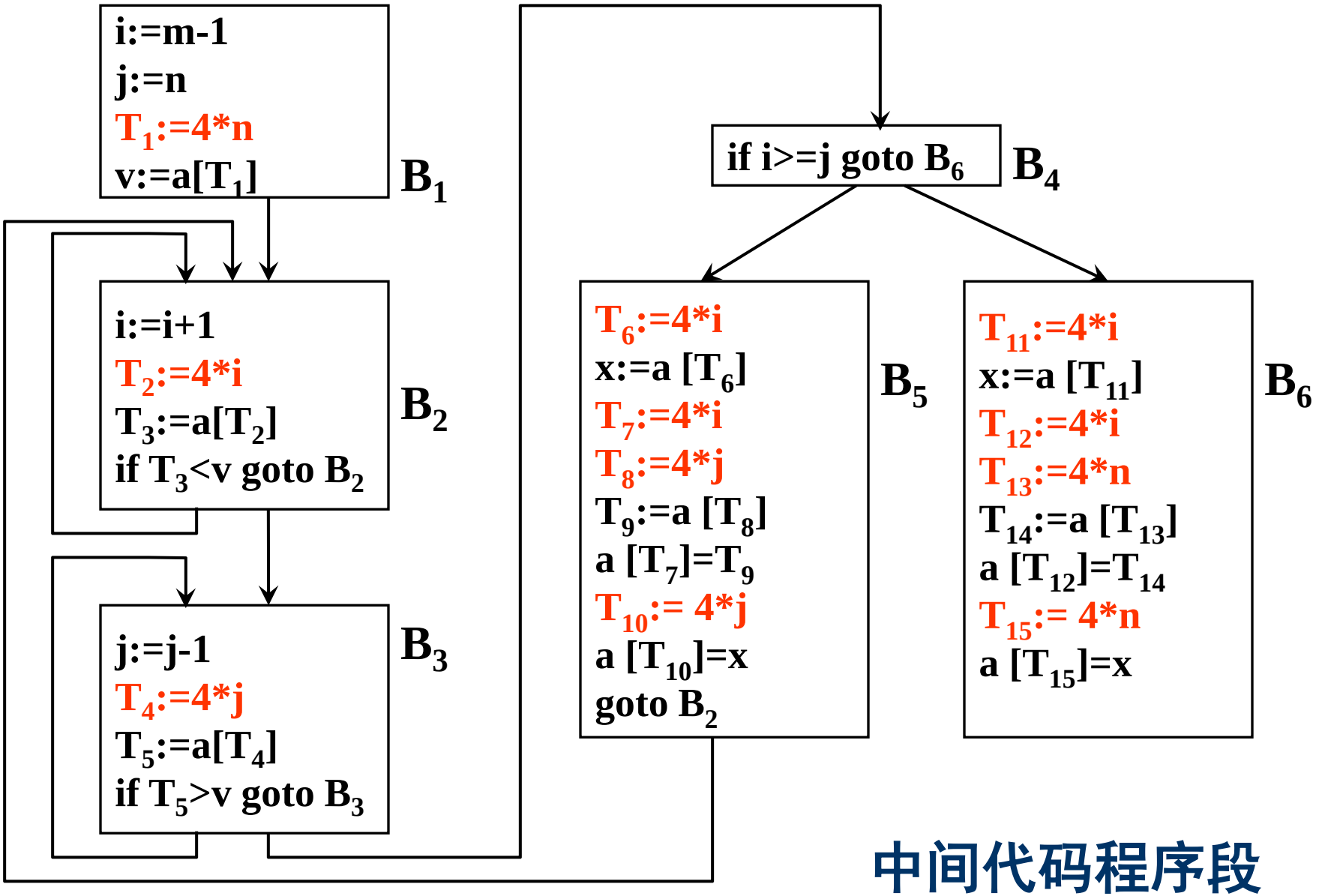
删除公共子表达式

B_5:
T_6:=4*i
x:=a [T_6]
T_7:=4*i
T_8:=4*j
T_9:=a [T_8]
a [T_7]=T_9
T_{10}:= 4*j
a [T_{10}]=x
goto B_2

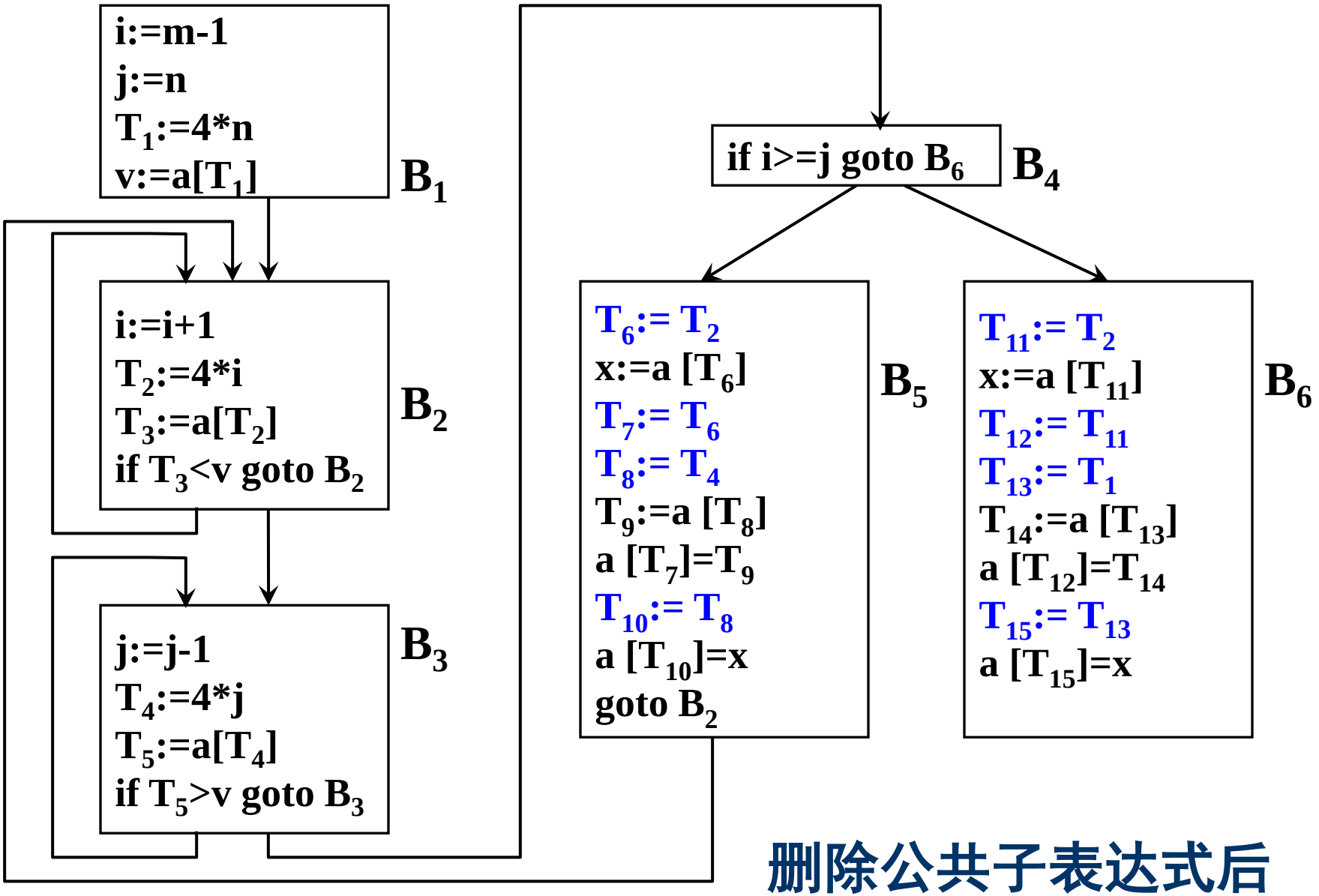


B_5:
T_6:=4*i
x:=a [T_6]
T_7:= T_6
T_8:=4*j
T_9:=a [T_8]
a [T_7]=T_9
T_{10}:= T_8
a [T_{10}]=x
goto B_2




i:=m-1
j:=n
T1:=4*n
v:=a[T1]
B1
i:=i+1
T2:=4*i
T3:=a[T2]
if T3<v goto B2
B2
j:=j-1
T4:=4*j
T5:=a[T4]
if T5>v goto B3
B3
if i>=j goto B6
B4
T6:=4*i
x:=a [T6]
T7:=4*i
T8:=4*j
T9:=a [T8]
a [T7]=T9
T10:= 4*j
a [T10]=x
goto B2
B5
T11:=4*i
x:=a [T11]
T12:=4*i
T13:=4*n
T14:=a [T13]
a [T12]=T14
T15:= 4*n
a [T15]=x
B6

中间代码程序段

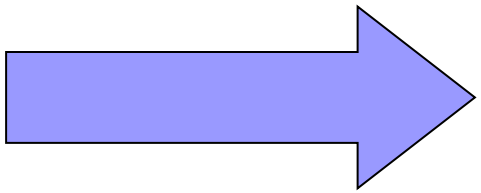

i:=m-1
j:=n
T1:=4*n
v:=a[T1]
B1
i:=i+1
T2:=4*i
T3:=a[T2]
if T3<v goto B2
B2
j:=j-1
T4:=4*j
T5:=a[T4]
if T5>v goto B3
B3
if i>=j goto B6
B4
T6:= T2
x:=a [T6]
T7:= T6
T8:= T4
T9:=a [T8]
a [T7]=T9
T10:= T8
a [T10]=x
goto B2
B5
T11:= T2
x:=a [T11]
T12:= T11
T13:= T1
T14:=a [T13]
a [T12]=T14
T15:= T13
a [T15]=x
B6

删除公共子表达式后

复写传播

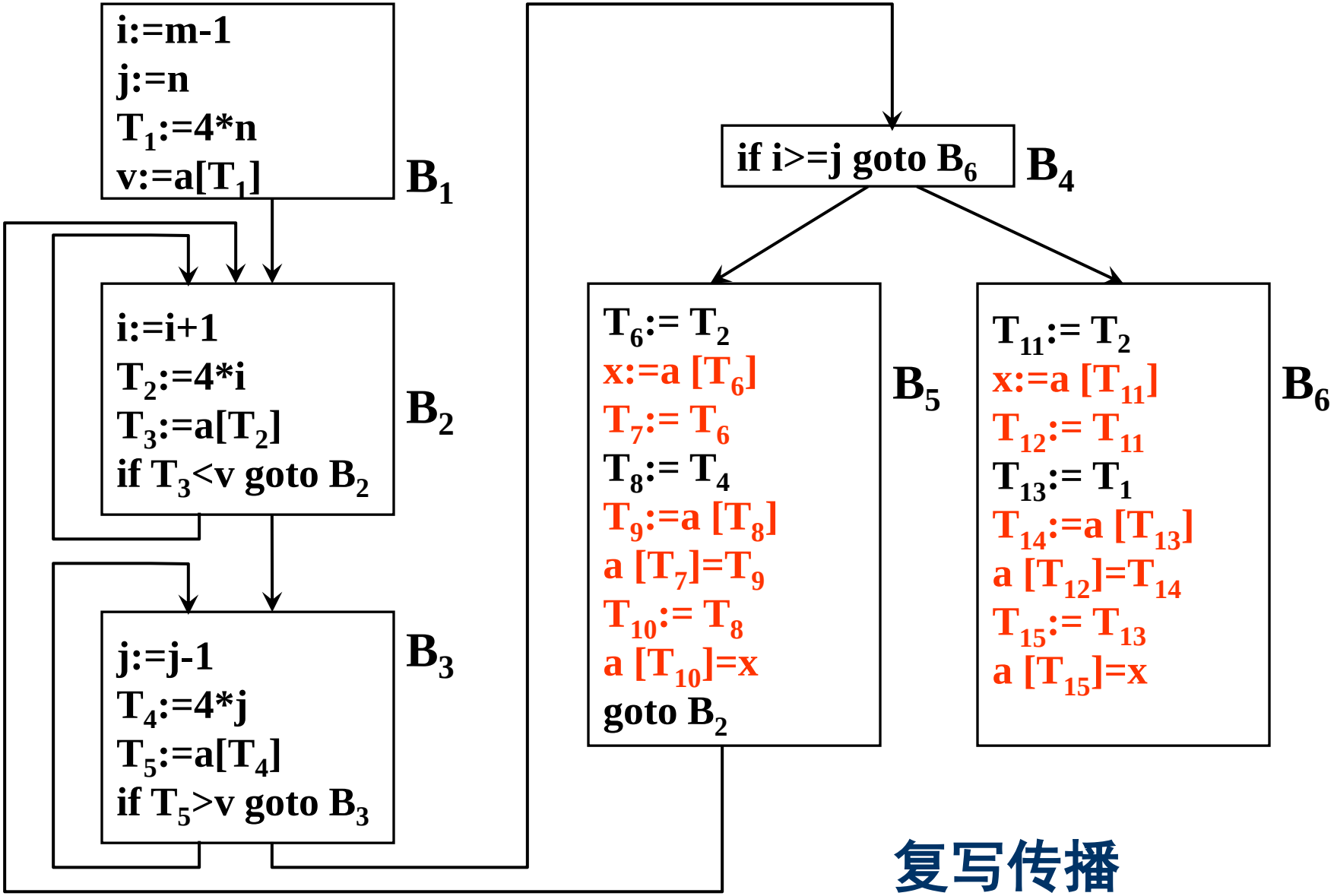
B_5:
$T_6 := T_2$
x:=a [T_6]
$T_7 := T_6$
$T_8 := T_4$
T_9:=a [T_8]
a [T_7]=T_9
$T_{10} := T_8$
a [T_{10}]=x
goto B_2

$T_6 := T_2$
x:=a [T_6]

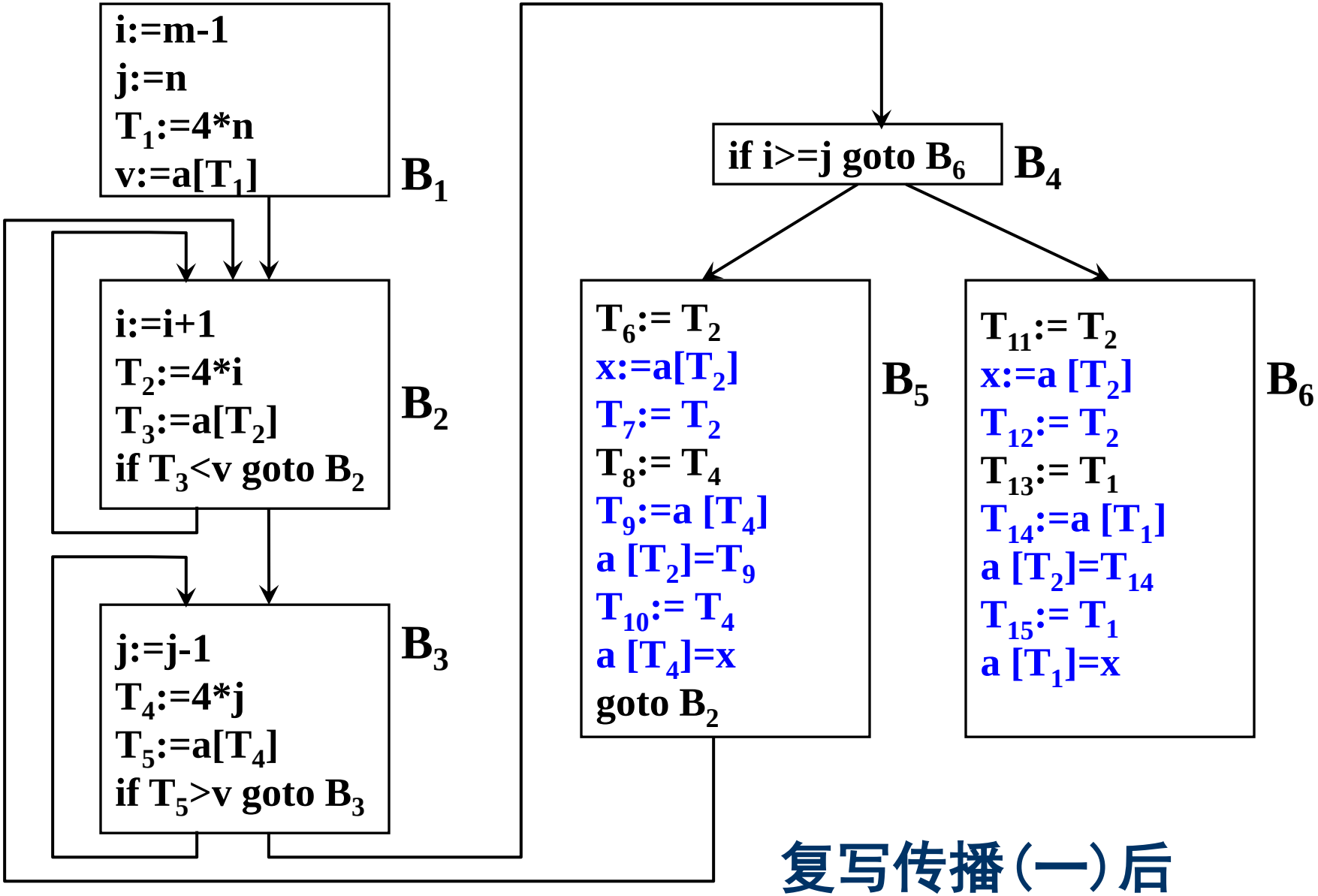


没有改变
T_6的值

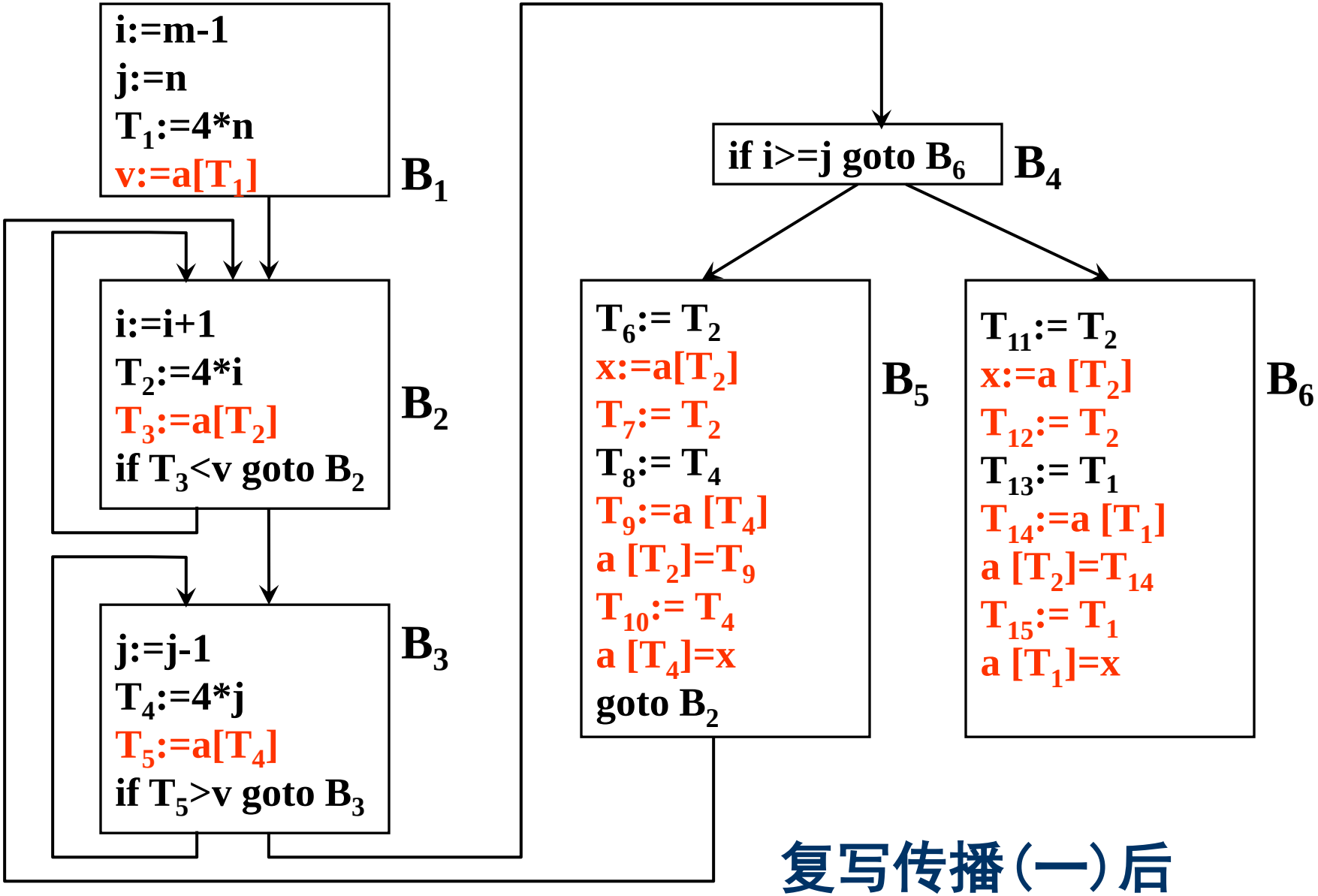
B_5:
$T_6 := T_2$
x:=a [T_2]
$T_7 := T_2$
$T_8 := T_4$
T_9:=a [T_4]
a [T_2]=T_9
$T_{10} := T_4$
a [T_4]=x
goto B_2


i:=m-1
j:=n
T1:=4*n
v:=a[T1]
B1
i:=i+1
T2:=4*i
T3:=a[T2]
if T3<v goto B2
B2
j:=j-1
T4:=4*j
T5:=a[T4]
if T5>v goto B3
B3
if i>=j goto B6
B4
T6:= T2
x:=a [T6]
T7:= T6
T8:= T4
T9:=a [T8]
a [T7]=T9
T10:= T8
a [T10]=x
goto B2
B5
T11:= T2
x:=a [T11]
T12:= T11
T13:= T1
T14:=a [T13]
a [T12]=T14
T15:= T13
a [T15]=x
B6

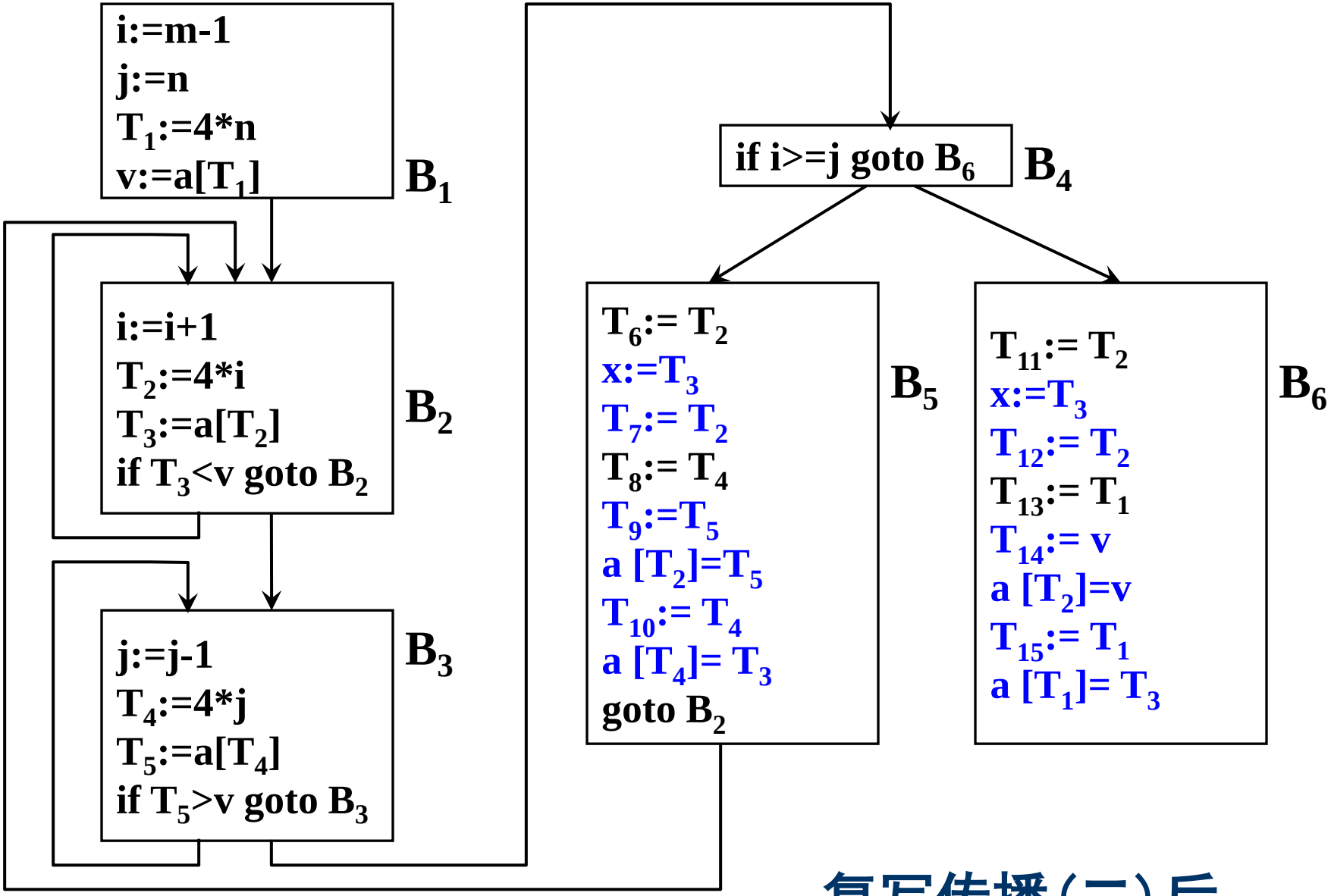
复写传播


i:=m-1
j:=n
T1:=4*n
v:=a[T1]
B1
i:=i+1
T2:=4*i
T3:=a[T2]
if T3<v goto B2
B2
j:=j-1
T4:=4*j
T5:=a[T4]
if T5>v goto B3
B3
if i>=j goto B6
B4
T6:= T2
x:=a[T2]
T7:= T2
T8:= T4
T9:=a [T4]
a [T2]=T9
T10:= T4
a [T4]=x
goto B2
B5
T11:= T2
x:=a [T2]
T12:= T2
T13:= T1
T14:=a [T1]
a [T2]=T14
T15:= T1
a [T1]=x
B6

复写传播(一)后


i:=m-1
j:=n
T1:=4*n
v:=a[T1]
B1
i:=i+1
T2:=4*i
T3:=a[T2]
if T3<v goto B2
B2
j:=j-1
T4:=4*j
T5:=a[T4]
if T5>v goto B3
B3
if i>=j goto B6
B4
T6:= T2
x:=a[T2]
T7:= T2
T8:= T4
T9:=a [T4]
a [T2]=T9
T10:= T4
a [T4]=x
goto B2
B5
T11:= T2
x:=a [T2]
T12:= T2
T13:= T1
T14:=a [T1]
a [T2]=T14
T15:= T1
a [T1]=x
B6

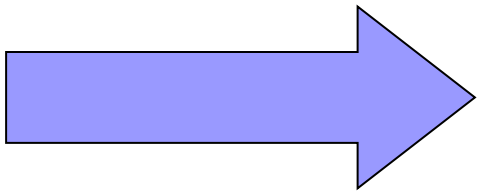
复写传播(一)后


i:=m-1
j:=n
T1:=4*n
v:=a[T1]
B1
i:=i+1
T2:=4*i
T3:=a[T2]
if T3<v goto B2
B2
j:=j-1
T4:=4*j
T5:=a[T4]
if T5>v goto B3
B3
if i>=j goto B6
B4
T6:= T2
x:=T3
T7:= T2
T8:= T4
T9:=T5
a [T2]=T5
T10:= T4
a [T4]= T3
goto B2
B5
T11:= T2
x:=T3
T12:= T2
T13:= T1
T14:= v
a [T2]=v
T15:= T1
a [T1]= T3
B6

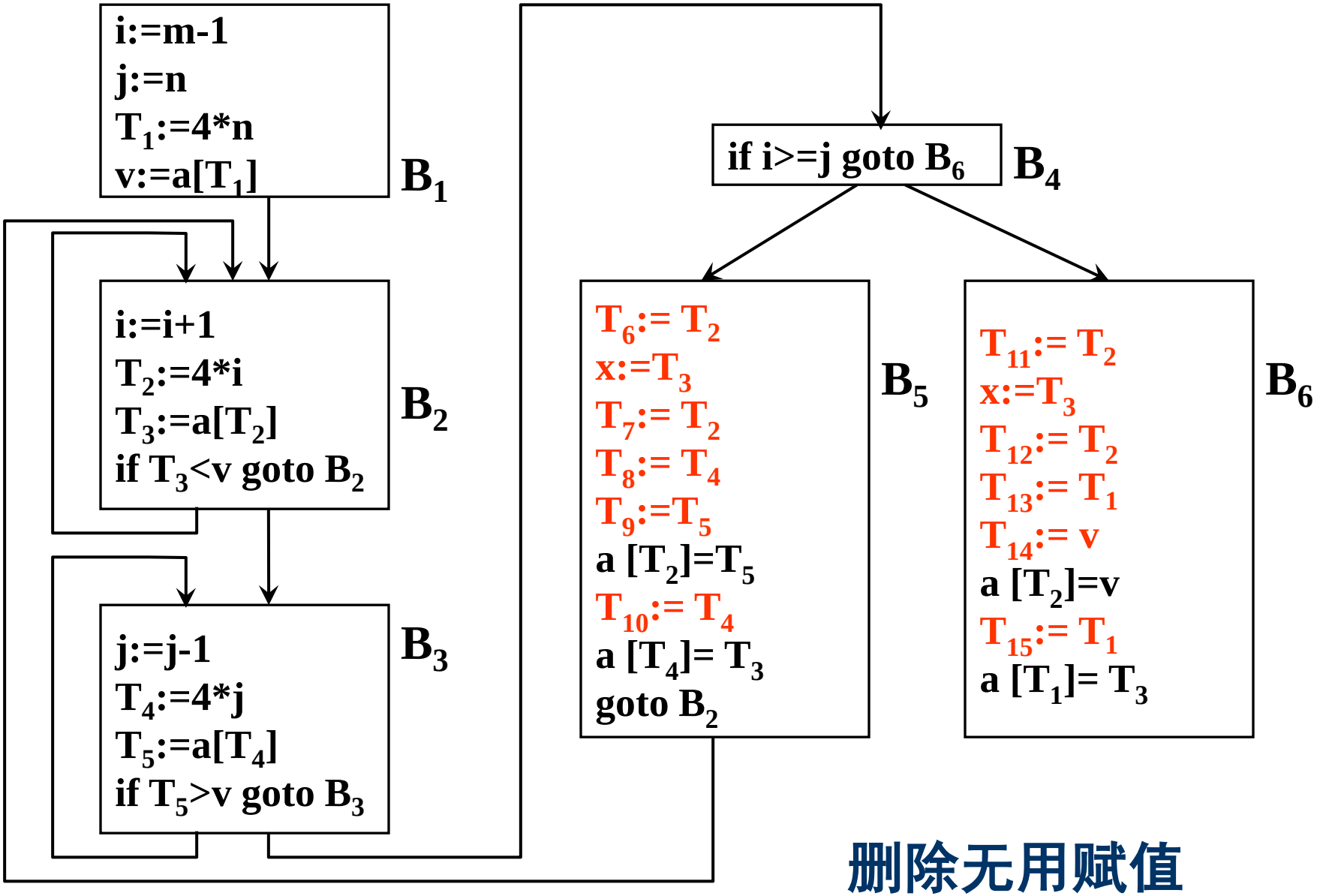
复写传播(二)后

删除无用代码

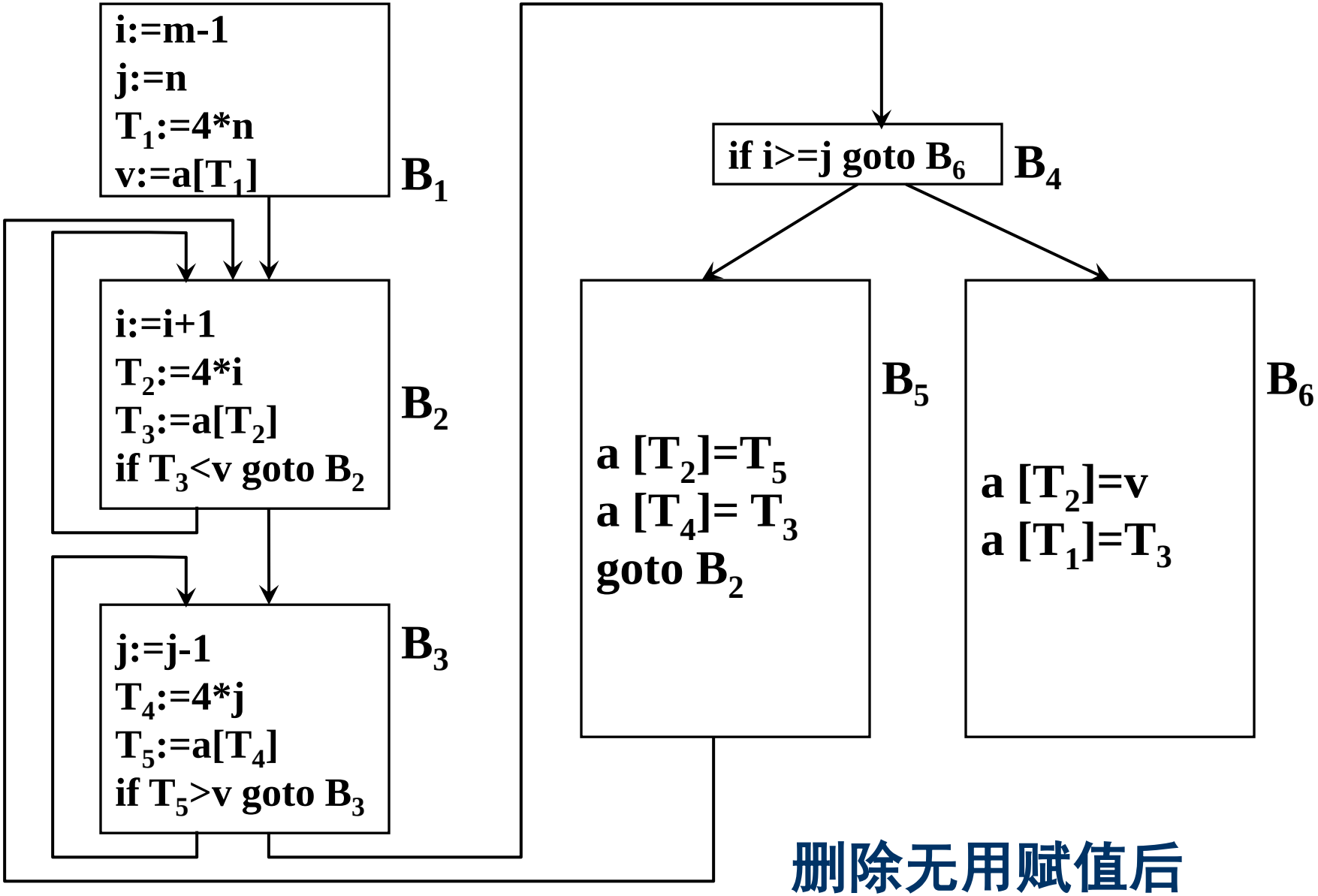
B_5:
$T_6 := T_2$
$x := T_3$
$T_7 := T_2$
$T_8 := T_4$
$T_9 := T_5$
$a[T_2] = T_5$
$T_{10} := T_4$
$a[T_4] = T_3$
goto B_2



B_5:
$a[T_2] = T_5$
$a[T_4] = T_3$
goto B_2

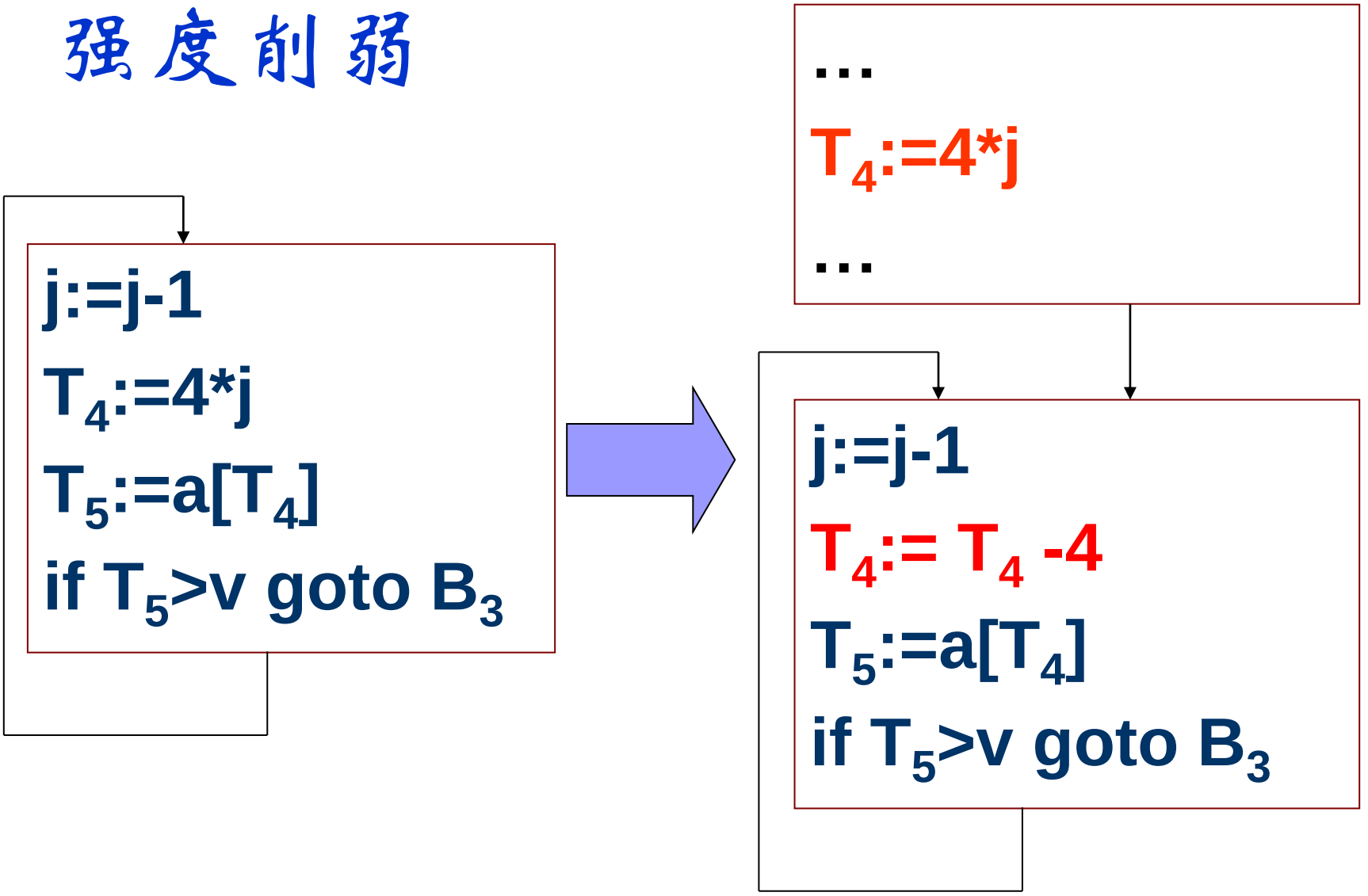

i:=m-1
j:=n
T1:=4*n
v:=a[T1]
B1
i:=i+1
T2:=4*i
T3:=a[T2]
if T3<v goto B2
B2
j:=j-1
T4:=4*j
T5:=a[T4]
if T5>v goto B3
B3
if i>=j goto B6
B4
T6:= T2
x:=T3
T7:= T2
T8:= T4
T9:=T5
a [T2]=T5
T10:= T4
a [T4]= T3
goto B2
B5
T11:= T2
x:=T3
T12:= T2
T13:= T1
T14:= v
a [T2]=v
T15:= T1
a [T1]= T3
B6

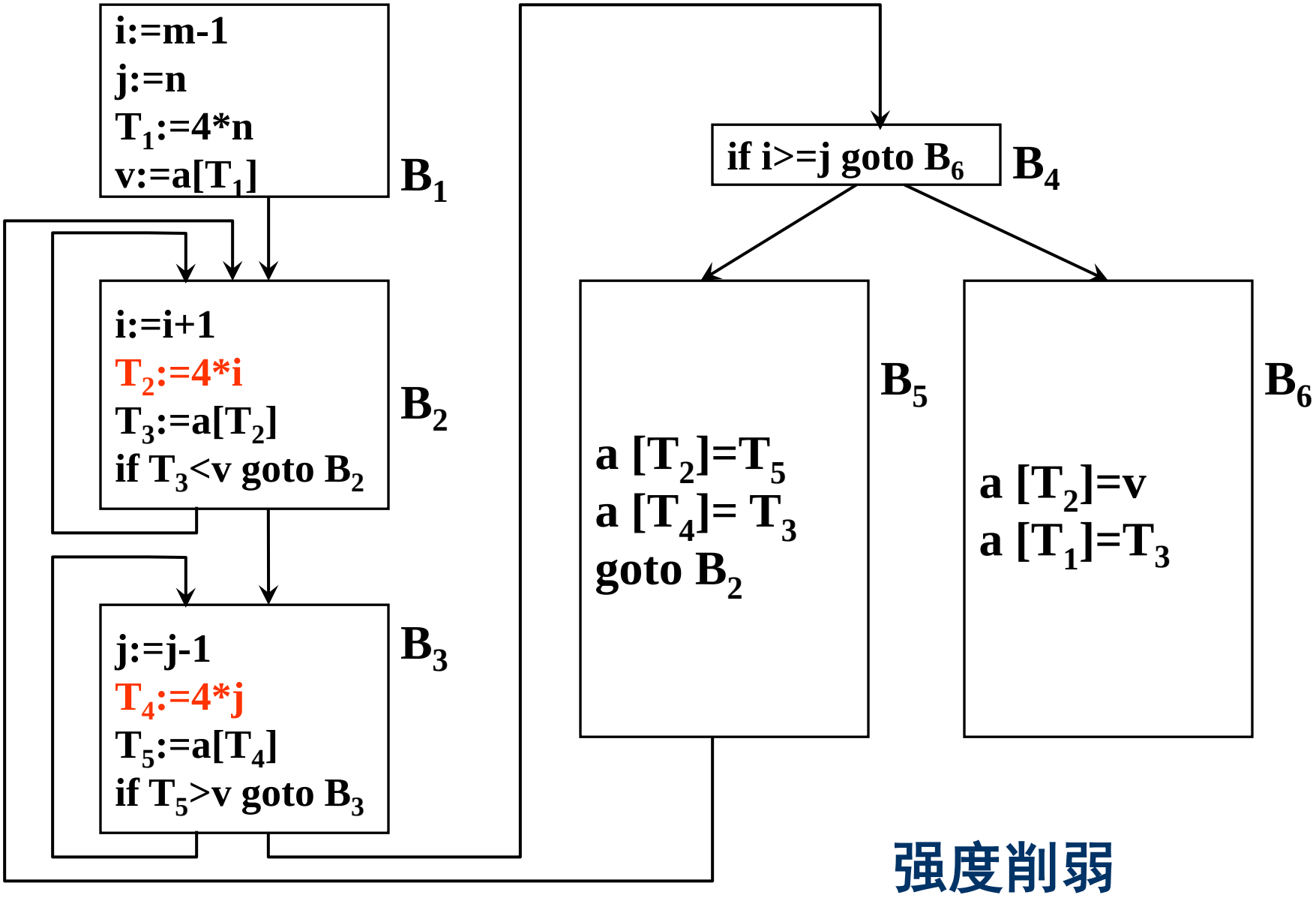
删除无用赋值


i:=m-1
j:=n
T1:=4*n
v:=a[T1]
B1
i:=i+1
T2:=4*i
T3:=a[T2]
if T3<v goto B2
B2
j:=j-1
T4:=4*j
T5:=a[T4]
if T5>v goto B3
B3
if i>=j goto B6
B4
a [T2]=T5
a [T4]= T3
goto B2
B5
a [T2]=v
a [T1]=T3
B6

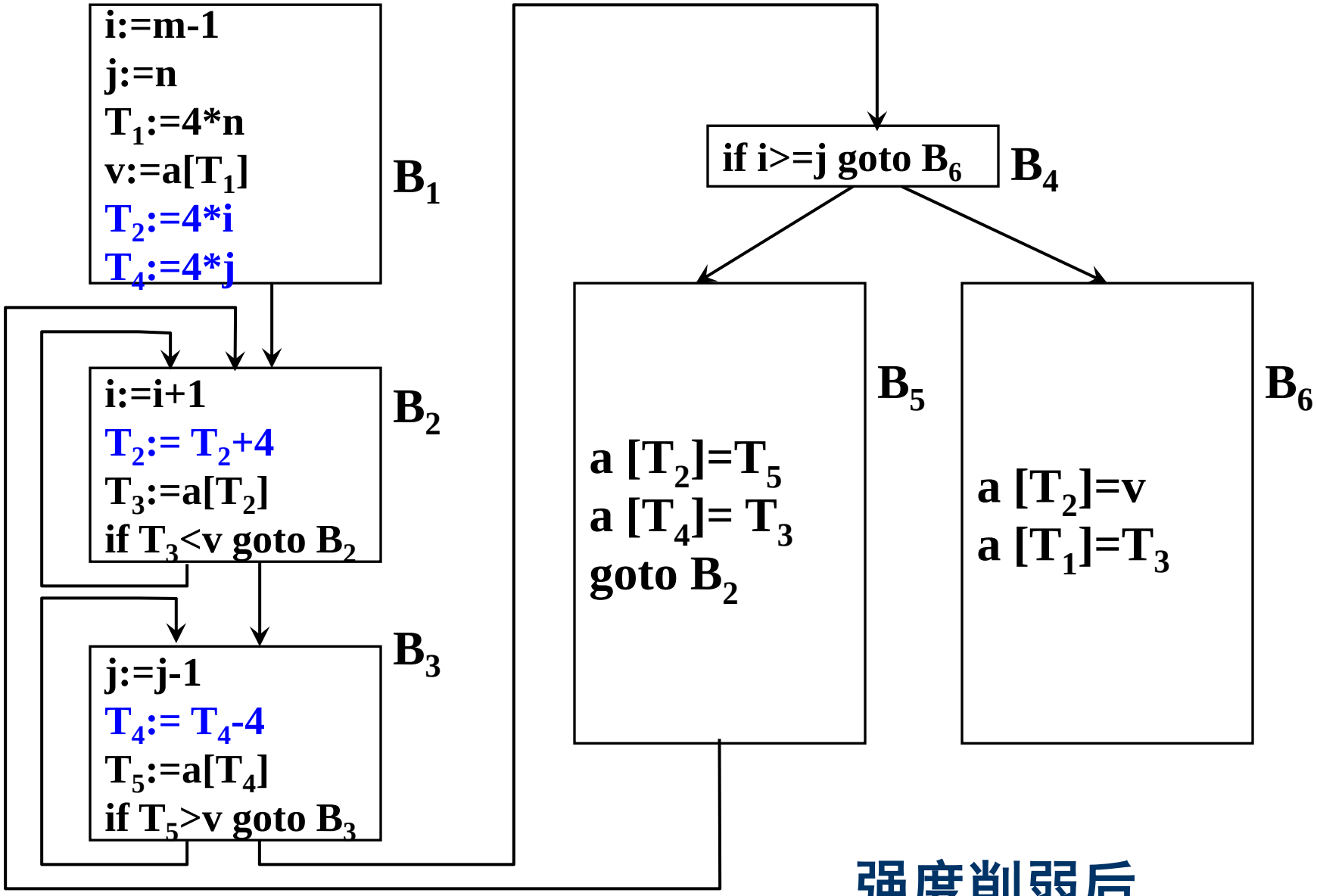
删除无用赋值后

强度削弱


j:=j-1
T4:=4*j
T5:=a[T4]
if T5>v goto B3
...
T4:=4*j
...
j:=j-1
T4:= T4 -4
T5:=a[T4]
if T5>v goto B3


i:=m-1
j:=n
T1:=4*n
v:=a[T1]
B1
i:=i+1
T2:=4*i
T3:=a[T2]
if T3<v goto B2
B2
j:=j-1
T4:=4*j
T5:=a[T4]
if T5>v goto B3
B3
if i>=j goto B6
B4
a [T2]=T5
a [T4]= T3
goto B2
B5
a [T2]=v
a [T1]=T3
B6

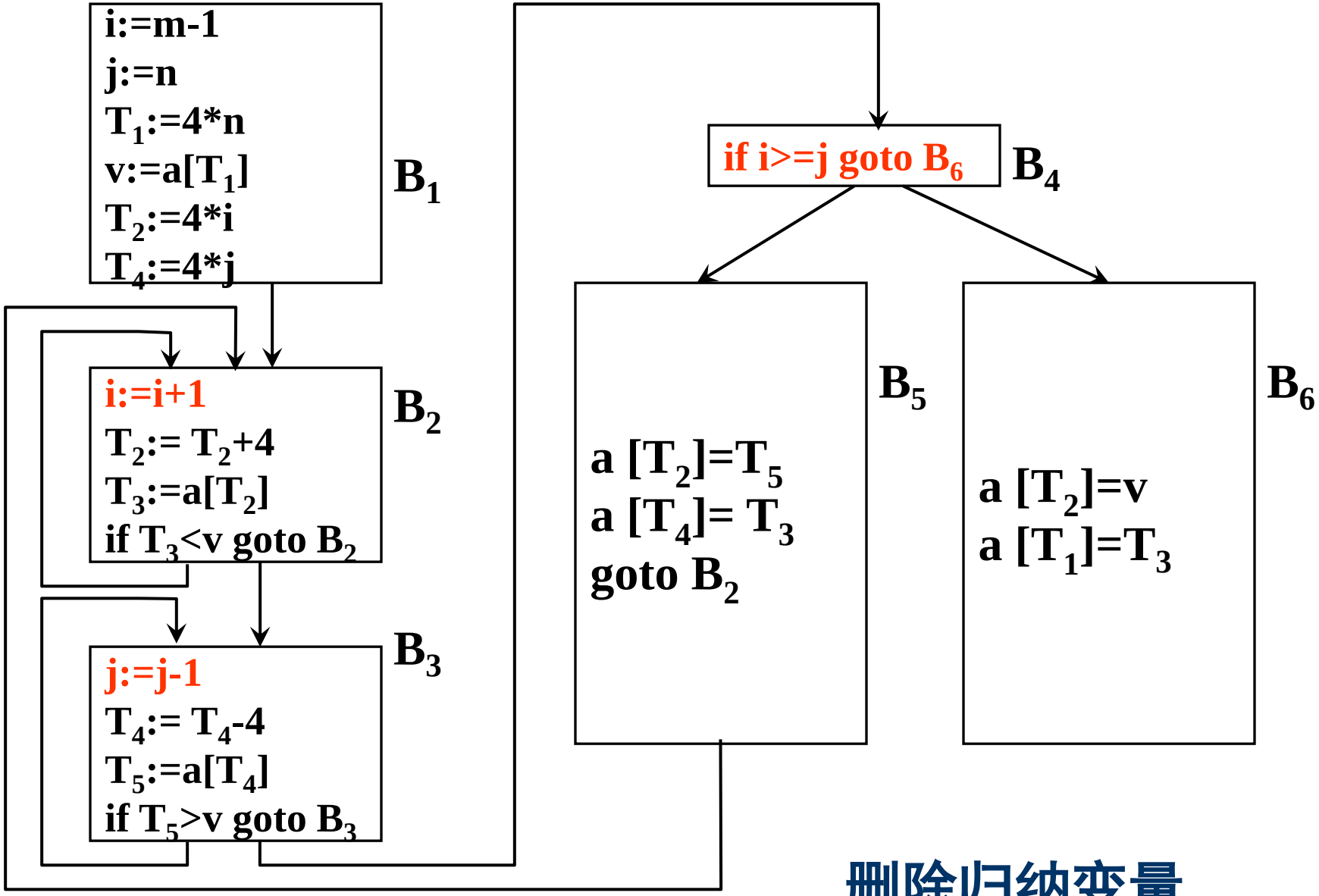
强度削弱


i:=m-1
j:=n
T1:=4*n
v:=a[T1]
T2:=4*i
T4:=4*j
B1
i:=i+1
T2:= T2+4
T3:=a[T2]
if T3<v goto B2
B2
j:=j-1
T4:= T4-4
T5:=a[T4]
if T5>v goto B3
B3
if i>=j goto B6
B4
a [T2]=T5
a [T4]= T3
goto B2
B5
a [T2]=v
a [T1]=T3
B6

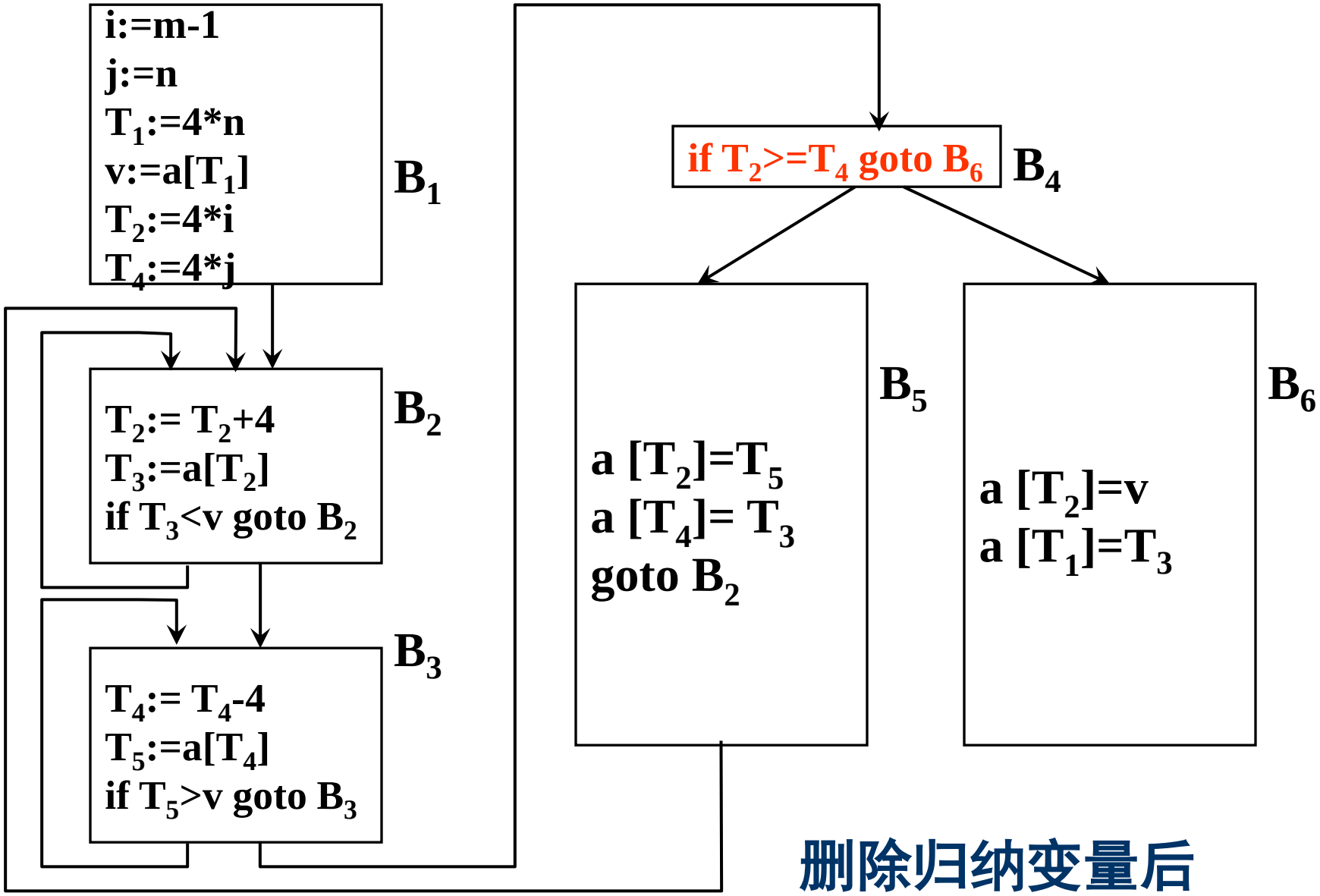
强度削弱后

删除归纳变量

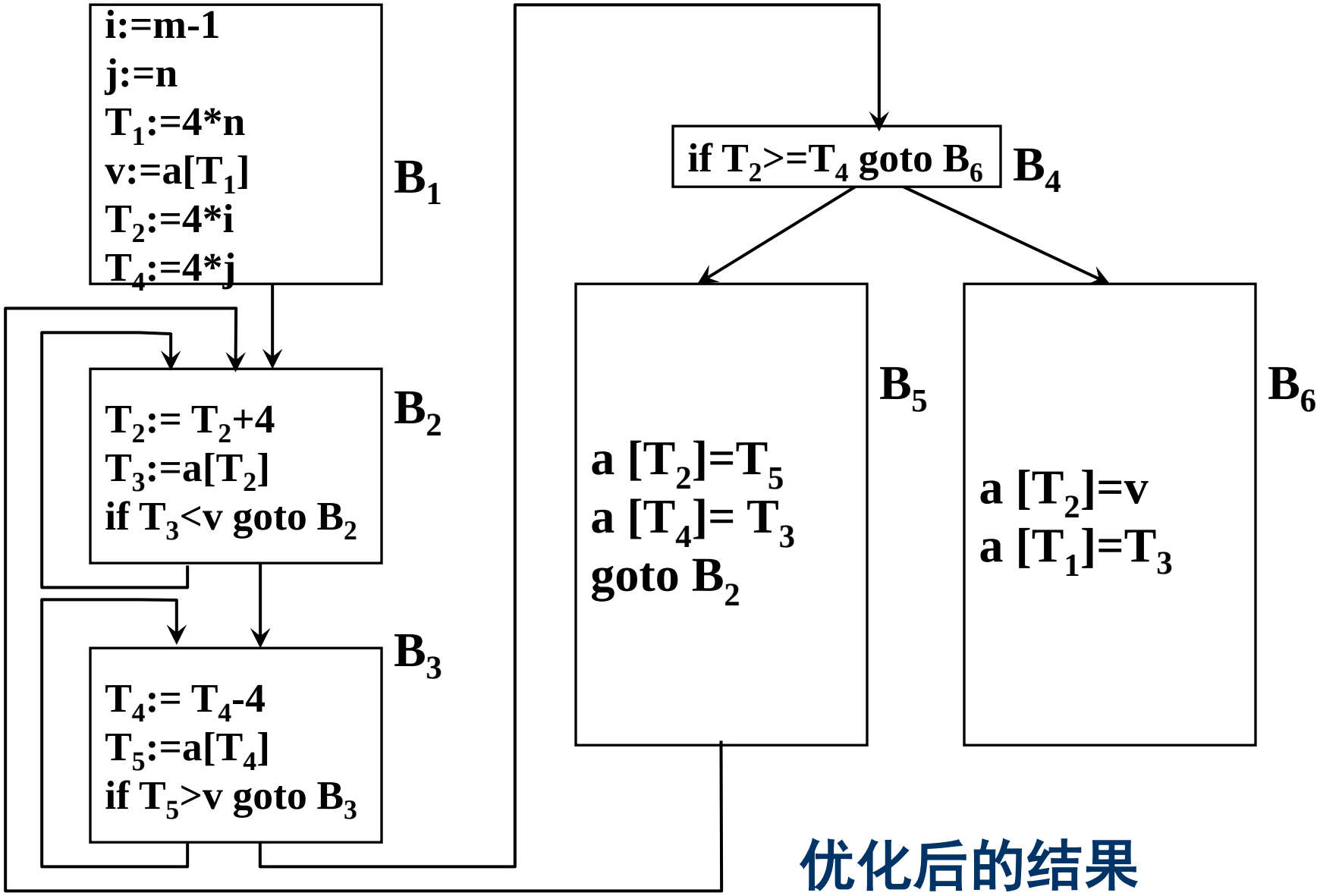
- B_2中的i与T_2的值保持着线性关系
- B_3中的j与T_4的值保持着线性关系
- 此种变量称之为归纳变量
- 对该类变量也可进行优化


i:=m-1
j:=n
T1:=4*n
v:=a[T1]
T2:=4*i
T4:=4*j
B1
i:=i+1
T2:= T2+4
T3:=a[T2]
if T3<v goto B2
B2
j:=j-1
T4:= T4-4
T5:=a[T4]
if T5>v goto B3
B3
if i>=j goto B6
B4
a [T2]=T5
a [T4]= T3
goto B2
B5
a [T2]=v
a [T1]=T3
B6

删除归纳变量


i:=m-1
j:=n
T1:=4*n
v:=a[T1]
T2:=4*i
T4:=4*j
B1
T2:= T2+4
T3:=a[T2]
if T3<v goto B2
B2
T4:= T4-4
T5:=a[T4]
if T5>v goto B3
B3
if T2>=T4 goto B6
B4
a [T2]=T5
a [T4]= T3
goto B2
B5
a [T2]=v
a [T1]=T3
B6

删除归纳变量后


i:=m-1
j:=n
T1:=4*n
v:=a[T1]
T2:=4*i
T4:=4*j
B1
T2:= T2+4
T3:=a[T2]
if T3<v goto B2
B2
T4:= T4-4
T5:=a[T4]
if T5>v goto B3
B3
if T2>=T4 goto B6
B4
a [T2]=T5
a [T4]= T3
goto B2
B5
a [T2]=v
a [T1]=T3
B6

优化后的结果

内容线索

- ✓ 概述
- ■ **局部优化**
- ■ 循环优化

问题

- 什么是基本块?
- 怎样划分基本块?
- 在基本块中可以进行哪些优化?
- 怎样进行局部优化?

基本块及相关概念

- **基本块**：指程序中一顺序执行语句序列，其中只有一个入口和一个出口。入口就是其中第一个语句，出口就是其中最后一个语句。
- 如果一条三地址语句为x:=y+z，则称对x**定值**并**引用**y和z。
- 在一个基本块中的一个名字，所谓在程序中的某个给定点是**活跃**的，是指如果在程序中(包括在本基本块或在其它基本块中)它的值在该点以后被引用。

局部优化

- 局限于基本块范围内的优化称为**基本块内的优化**，或称**局部优化**。
- 在一个基本块内通常可以实行下面的优化：
  - 删除公共子表达式
  - 删除无用赋值
  - 合并已知量
  - 临时变量改名
  - 交换语句的位置
  - 代数变换

划分基本块

- **划分四元式程序为基本块的算法：**

1. 求出四元式程序中各个基本块的入口语句：

1）**程序第一个语句，或**

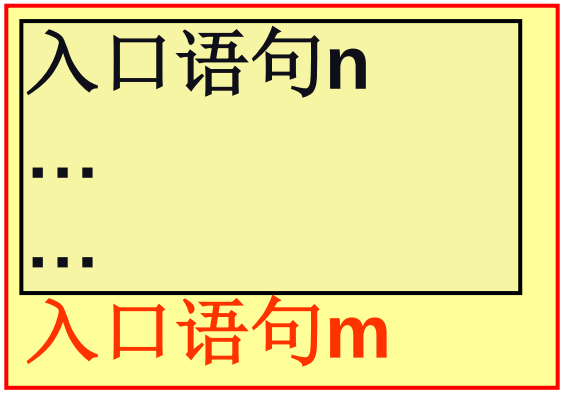
2）**能由条件转移语句或无条件转移语句转移到的语句，或**

3）**紧跟在条件转移语句后面的语句。**

划分基本块

- **划分四元式程序为基本块的算法：**

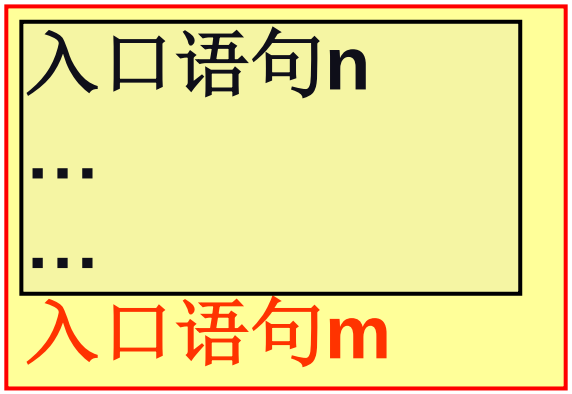
2. **对以上求出的每个入口语句，确定其所属的基本块。它是由该入口语句到下一入口语句（不包括该入口语句）之间的语句序列组成的。**

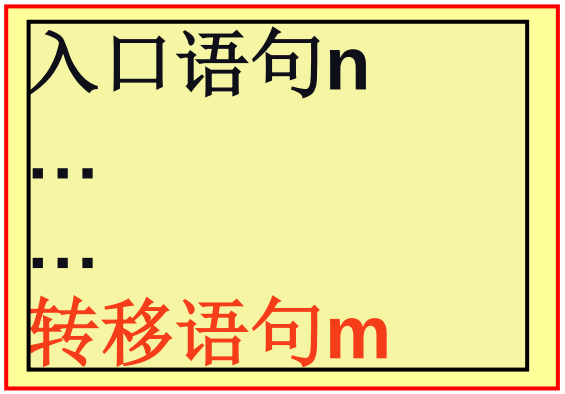


划分基本块

- **划分四元式程序为基本块的算法：**

2. **对以上求出的每个入口语句，确定其所属的基本块。它是由该入口语句到下一入口语句（不包括该入口语句）或到一转移语句（包括该转移语句）之间的语句序列组成的。**

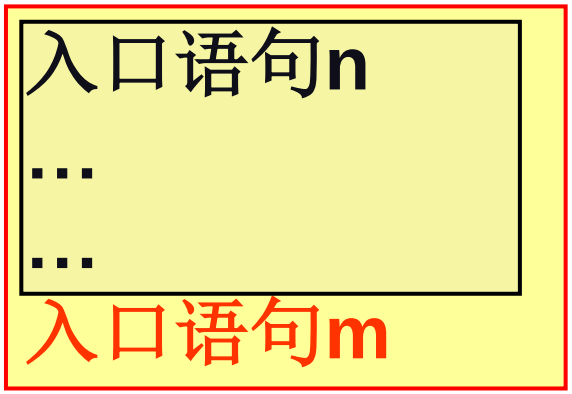


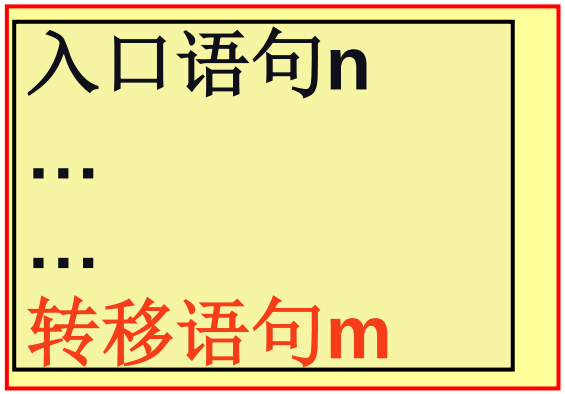


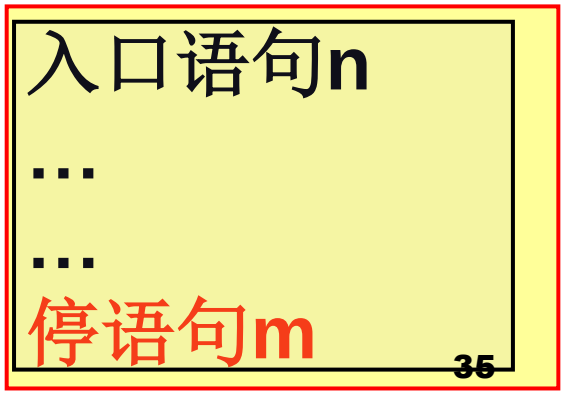
划分基本块

- 划分四元式程序为基本块的算法：

2. 对以上求出的每个入口语句，确定其所属的基本块。它是由该入口语句到**下一入口语句**（不包括该入口语句）或到**一转移语句**（包括该转移语句）或**一停语句**（包括该停语句）之间的语句序列组成的。



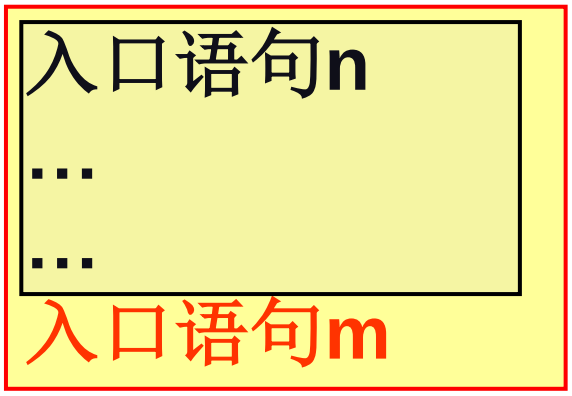


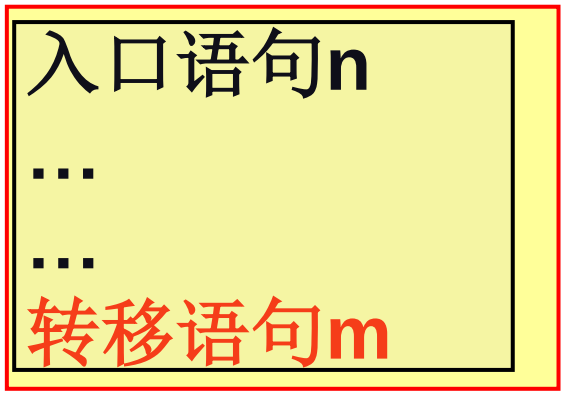


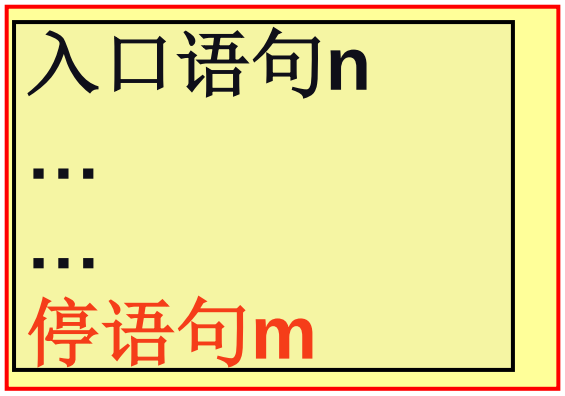
划分基本块

- **划分四元式程序为基本块的算法：**

2. **对以上求出的每个入口语句，确定其所属的基本块。它是由该入口语句到下一入口语句（不包括该入口语句）或到一转移语句（包括该转移语句）或一停语句（包括该停语句）之间的语句序列组成的。**







3. **凡未被纳入某一基本块中的语句，可以从程序中删除。**

1.求出四元式程序中各个基本块的入口语句:

1) 程序第一个语句，或

2) 能由条件转移语句或无条件转移语句转移到的语句，或

3) 紧跟在条件转移语句后面的语句。

例. 划分基本块

```
(1)    read X
(2)    read Y
(3)    R:=X mod Y
(4)    if  R=0 goto (8)
(5)    X:=Y
(6)    Y:=R
(7)    goto (3)
(8)    write Y
(9)    halt
```

1.求出四元式程序中各个基本块的入口语句:

1) 程序第一个语句，或

2) 能由条件转移语句或无条件转移语句转移到的语句，或

3) 紧跟在条件转移语句后面的语句。

例. 划分基本块

```
(1)    read X
(2)    read Y
(3)    R:=X mod Y
(4)    if  R=0 goto (8)
(5)    X:=Y
(6)    Y:=R
(7)    goto (3)
(8)    write Y
(9)    halt
```

1.求出四元式程序中各个基本块的入口语句:

1) 程序第一个语句，或

2) 能由条件转移语句或无条件转移语句转移到的语句，或

3) 紧跟在条件转移语句后面的语句。

例. 划分基本块

```
(1)    read X
(2)    read Y
(3)    R:=X mod Y
(4)    if  R=0 goto (8)
(5)    X:=Y
(6)    Y:=R
(7)    goto (3)
(8)    write Y
(9)    halt
```

1.求出四元式程序中各个基本块的入口语句:

1) 程序第一个语句，或

2) 能由条件转移语句或无条件转移语句转移到的语句，或

3) 紧跟在条件转移语句后面的语句。

例. 划分基本块

```
(1)    read X
(2)    read Y
(3)    R:=X mod Y
(4)    if  R=0 goto (8)
(5)    X:=Y
(6)    Y:=R
(7)    goto (3)
(8)    write Y
(9)    halt
```

例. 划分基本块

```
(1)    read X
(2)    read Y

(3)    R:=X mod Y
(4)    if  R=0 goto (8)

(5)    X:=Y
(6)    Y:=R
(7)    goto (3)

(8)    write Y
(9)    halt
```

2. 对以上求出的每个入口语句，确定其所属的基本块。它是由该入口语句到下一入口语句(不包括该入口语句)、或到一转移语句(包括该转移语句)、或一停语句(包括该停语句)之间的语句序列组成的。

优化措施

- 合并已知量

T_1:=2

…

T_2:=4*T_1

变换成

T_2:=8

- 临时变量改名

T:=b+c

其中T是一个临时变量名。把这个语句改成：

S:=b+c

优化措施

- 交换语句的位置

T_1:=b+c

T_2:=x+y

- 代数变换

x:=x+0

或 x:=x*1

x:=y**2

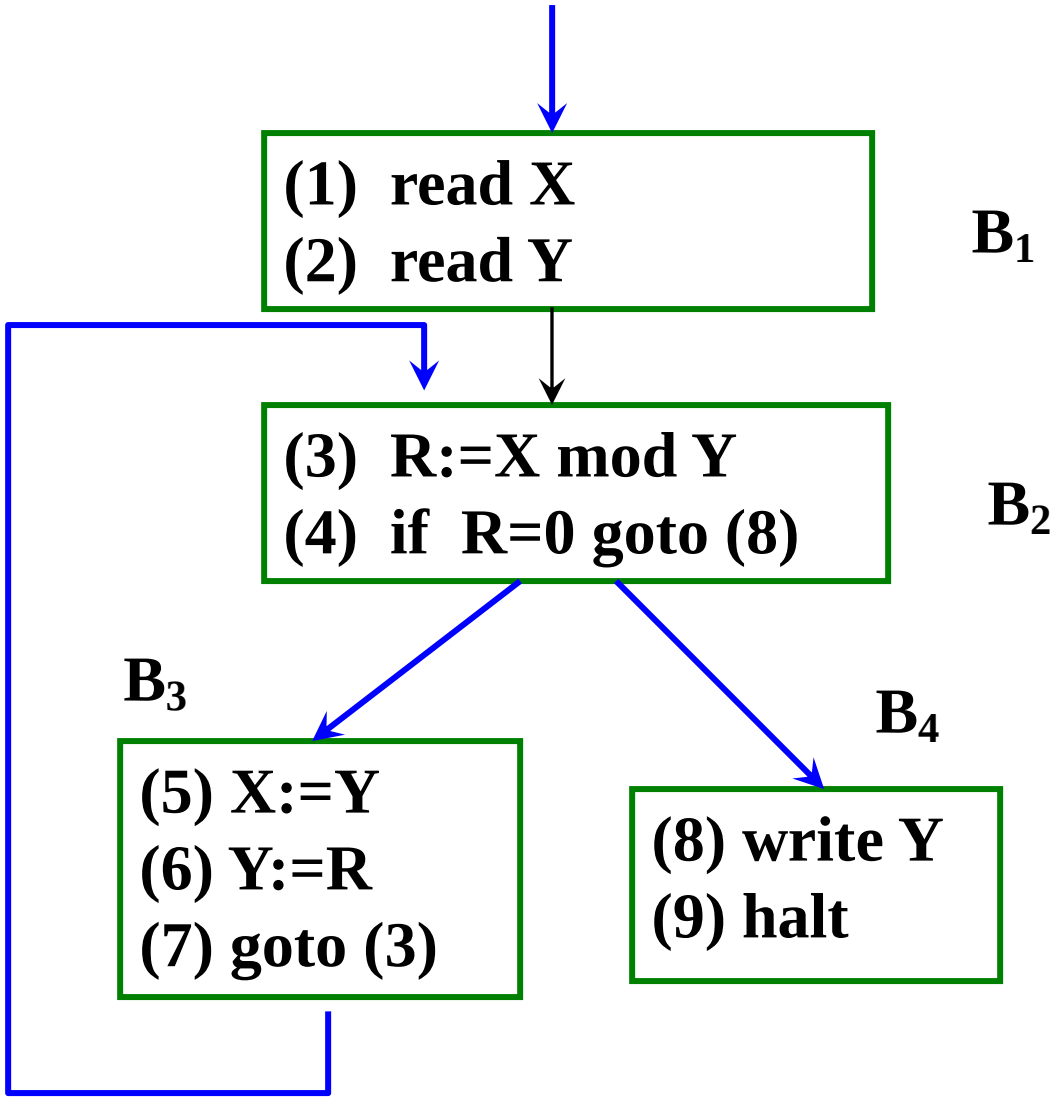
变换成

x:=y*y

流图

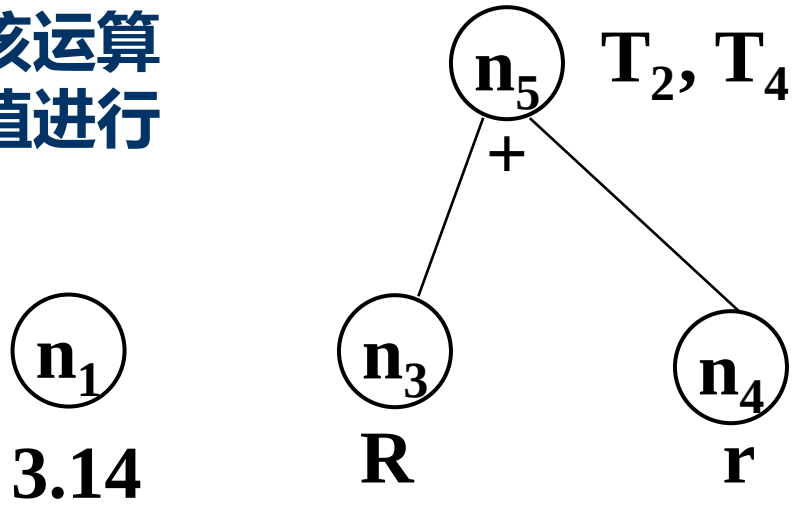
- 每个流图以基本块为**结点**。如果一个结点的基本块的入口语句是程序的第一条语句，则称此结点**为首结点**。如果在某个执行顺序中，基本块B_2紧接在基本块B_1之后执行，则从B_1到B_2有一条有向边。即，如果
  - 有一个条件或无条件转移语句从B_1的最后一条语句转移到B_2的第一条语句；或者
  - 在程序的序列中，B_2紧接在B_1的后面，并且B_1的最后一条语句不是一个无条件转移语句。我们就说B_1是B_2的**前驱**，B_2是B_1的**后继**。

(1) read X
(2) read Y
(3) R:=X mod Y
(4) if R=0 goto
(5) X:=Y
(6) Y:=R
(7) goto (3)
(8) write Y
(9) halt



基本块的DAG表示

- 基本块的DAG是一种带有下述标记或附加信息的DAG
  - 图的叶结点以一标识符或常数作为标记，表示该结点代表该变量或常数的值;
  - 图的内部结点以一运算符作为标记，表示该结点代表应用该运算符对其后继结点所代表的值进行运算的结果;
  - 图中各个结点上可能附加一个或多个标识符(称附加标识符)表示这些变量具有该结点所代表的值。

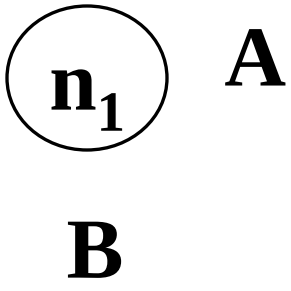


基本块的DAG表示及应用

- **与各四元式相对应的DAG结点形式:**

四元式 **DAG 图**

(0) 0型: A:=B
(:=，B，-，A)



四元式

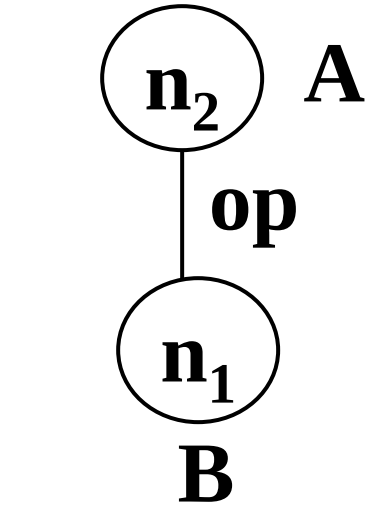
(1) 1型: A:=op B

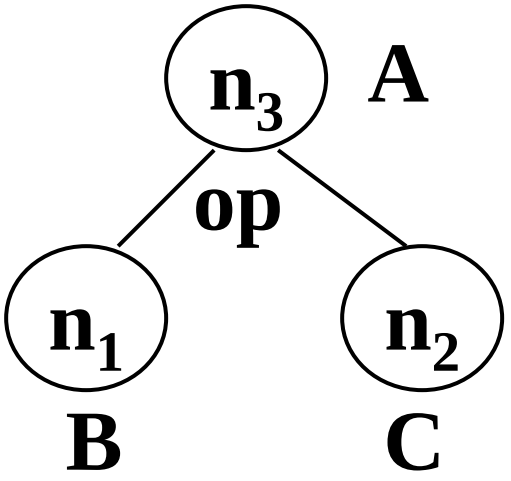
(op，B，-，A)

(2) 2型: A:=B op C

(op，B，C，A)

DAG 图



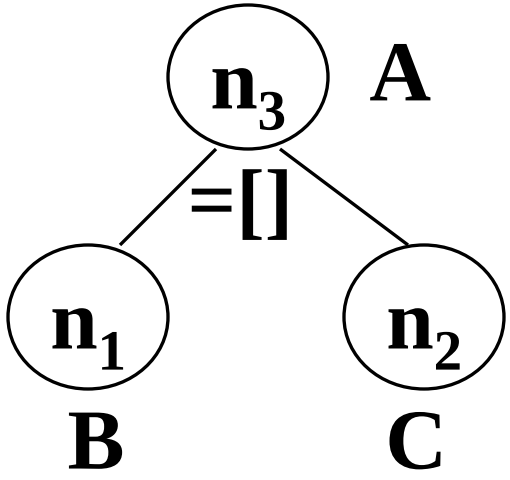


四元式

DAG 图

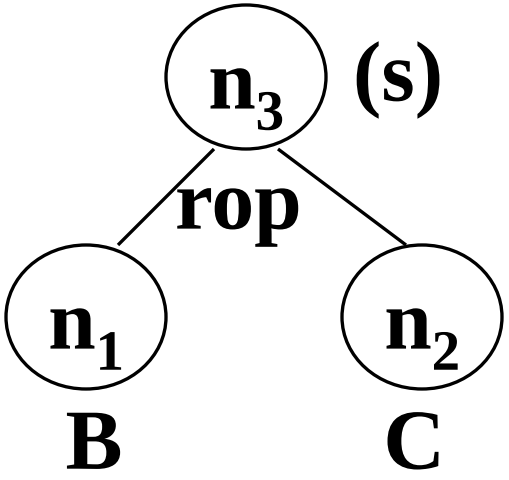
(3) 2型: A:=B[C]

(=[]，B[C]，-，A)



(4) 2型: if B rop C goto (s)

(jrop，B，C，(s))



四元式

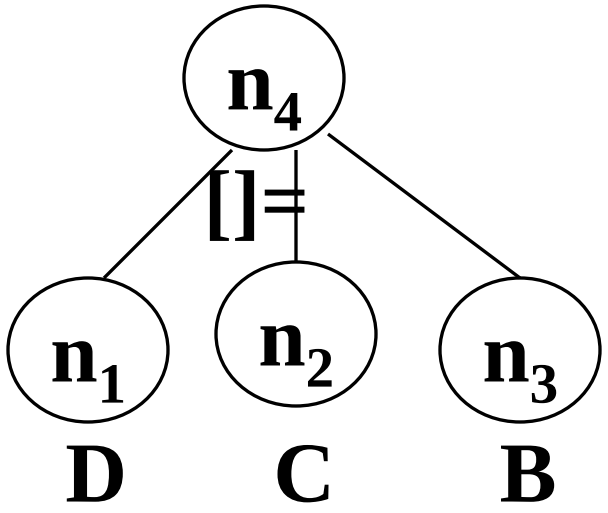
(5) 3型: D[C]:=B

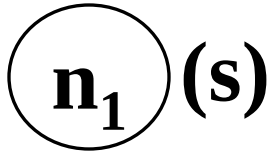
([]=，B，-，D[C])

(6) 0型: goto (s)

(j，-，-，(s))

DAG 图





基本块的DAG构造算法

（0）A:=B
（1）A:=op B
（2）A:=B op C 或A：=B[C]

- 对基本块中每一四元式，依次执行以下步骤:
  1. 准备操作数的结点
  2. 合并已知量
  3. 寻找公共子表达式
  4. 删除无用赋值

基本块的DAG构造算法

1.准备操作数的结点

- 如果NODE(B)无定义，则构造一标记为B的叶结点并定义NODE(B)为这个结点;
  - 如果当前四元式是0型，则记NODE(B)的值为n，转4。
  - 如果当前四元式是1型，则转2(1)
  - 如果当前四元式是2型，则(i)如果NODE(C)无定义，则构造一标记为C的叶结点并定义NODE(C)为这个结点;(ii)转2(2)。

基本块的DAG构造算法

2.合并已知量

(1) 如果NODE(B)是标记为常数的叶结点，则转2(3)；否则，转3(1)。

(2) 如果NODE(B)和NODE(C)都是标记为常数的叶结点，则转2(4)；否则，转3(2)。

(3) 执行op B (即合并已知量)。令得到的新常数为P。如果NODE(B)是处理当前四元式时新构造出来的结点，则删除它。如果NODE(P)无定义，则构造一用P作标记的叶结点n。置NODE(P)=n，转4。

(4)执行B op C (即合并已知量)。令得到的新常数为P。如果NODE(B)或NODE(C)是处理当前四元式时新构造出来的结点，则删除它。如果NODE(P)无定义，则构造一用P作标记的叶结点n。置NODE(P)=n，转4。

基本块的DAG构造算法

3.寻找公共子表达式

(1) 检查DAG中是否已有一结点，其唯一后继为NODE(B)且标记为op(即公共子表达式)。如果没有，则构造该结点n，否则，把已有的结点作为它的结点并设该结点为n。转4。

(2) 检查DAG中是否已有一结点，其左后继为NODE(B)，右后继为NODE(C)，且标记为op(即公共子表达式)。如果没有，则构造该结点n，否则，把已有的结点作为它的结点并设该结点为n。转4。

基本块的DAG构造算法

4. 删除无用赋值

如果NODE(A)无定义，则把A附加在结点n上并令NODE(A)=n;否则，先把A从NODE(A)结点上的附加标识符集中删除（注意，如果NODE(A)是叶结点，则其A标记不删除）。把A附加到新结点n上并置NODE(A)=n。转处理下一四元式。

例. 试构造以下基本块G的DAG

(1) T_0:=3.14
(2) T_1:=2*T_0
(3) T_2:=R+r
(4) A:=T_1*T_2
(5) B:=A
(6) T_3:=2*T_0
(7) T_4:=R+r
(8) T_5:=T_3*T_4
(9) T_6:=R-r
(10) B:=T_5*T_6

(1) T_0:=3.14

(2) T_1:=2*T_0

(3) T_2:=R+r

(4) A:=T_1*T_2

(5) B:=A

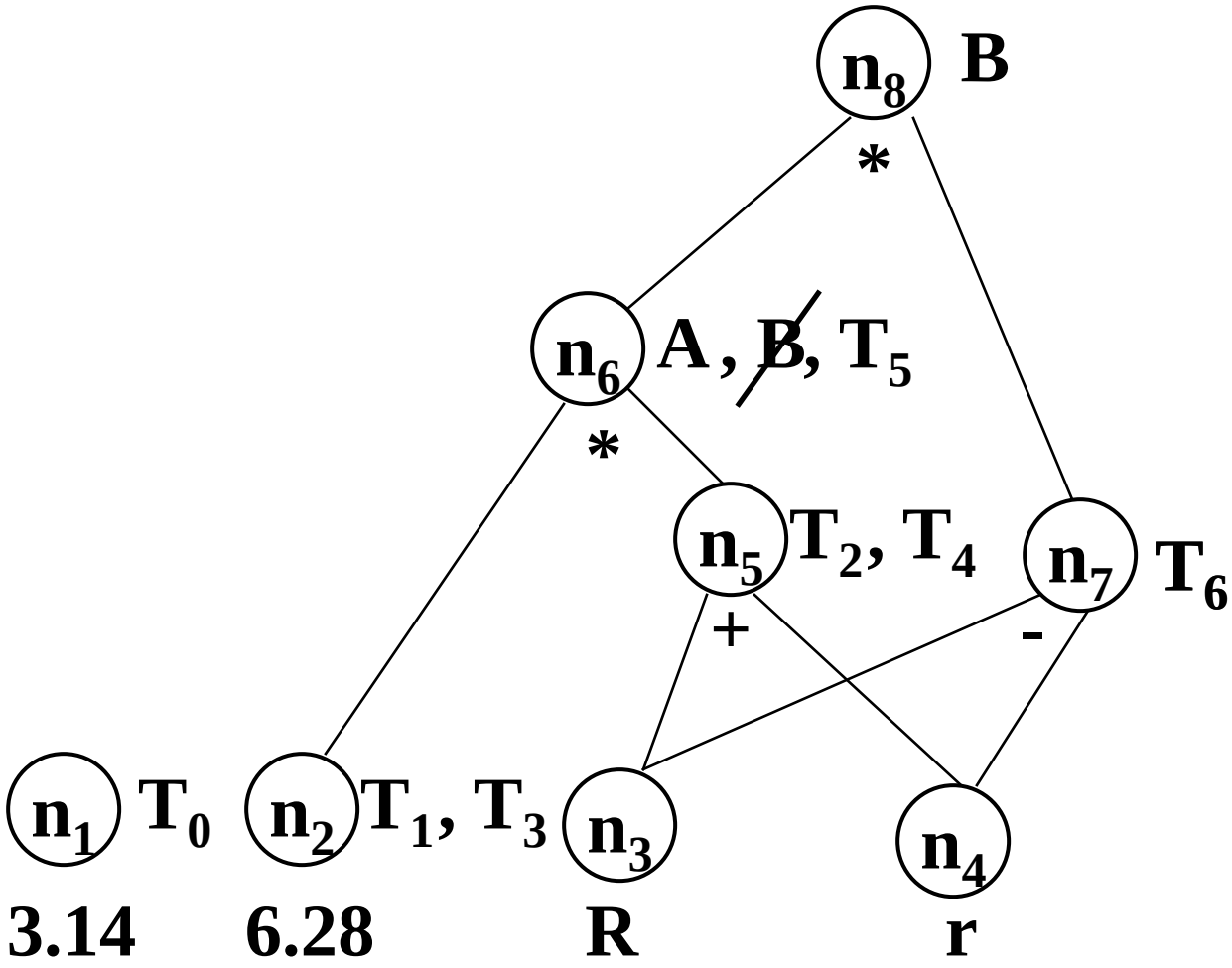
(6) T_3:=2*T_0

(7) T_4:=R+r

(8) T_5:=T_3*T_4

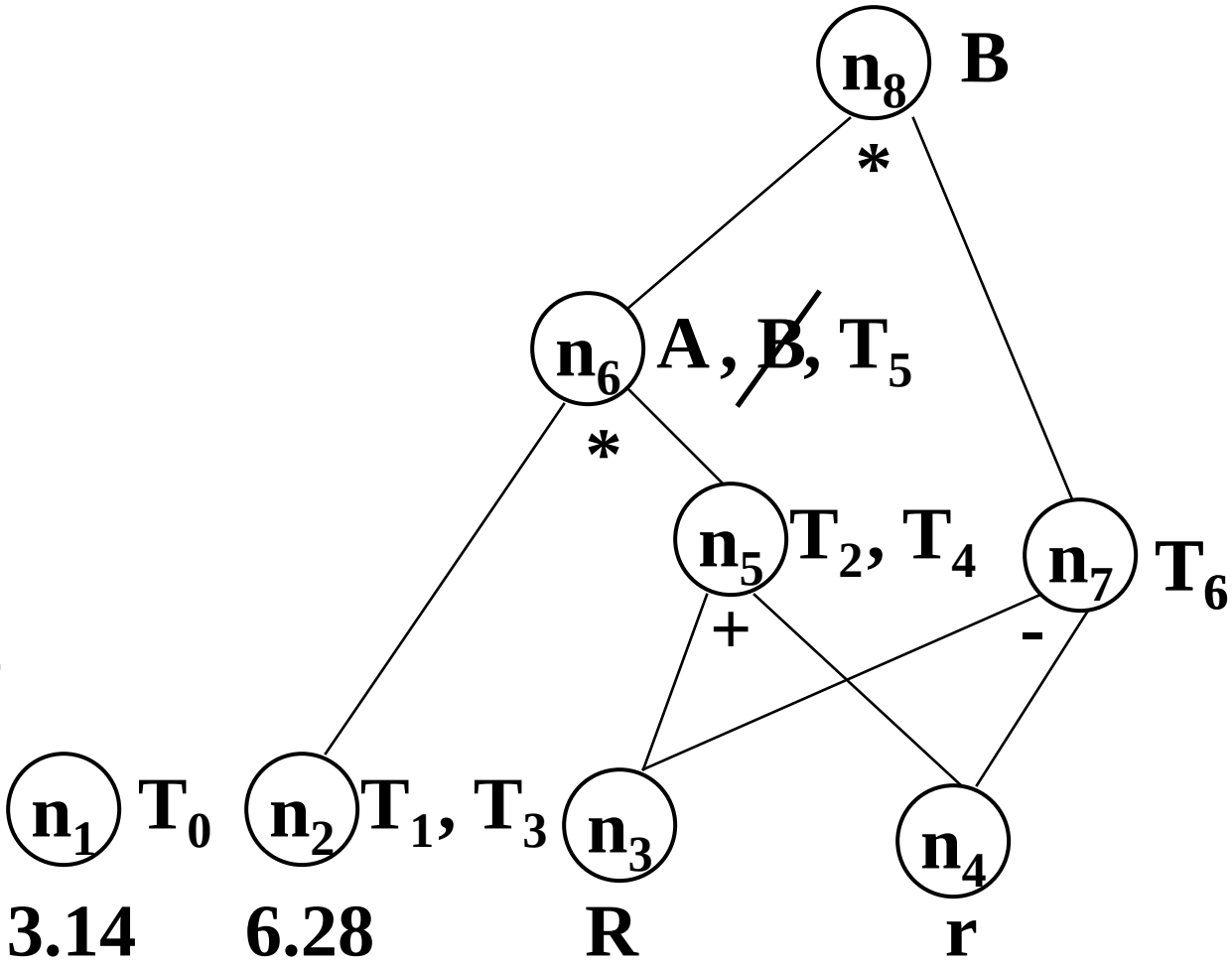
(9) T_6:=R-r

(10) B:=T_5*T_6



优化后的四元式

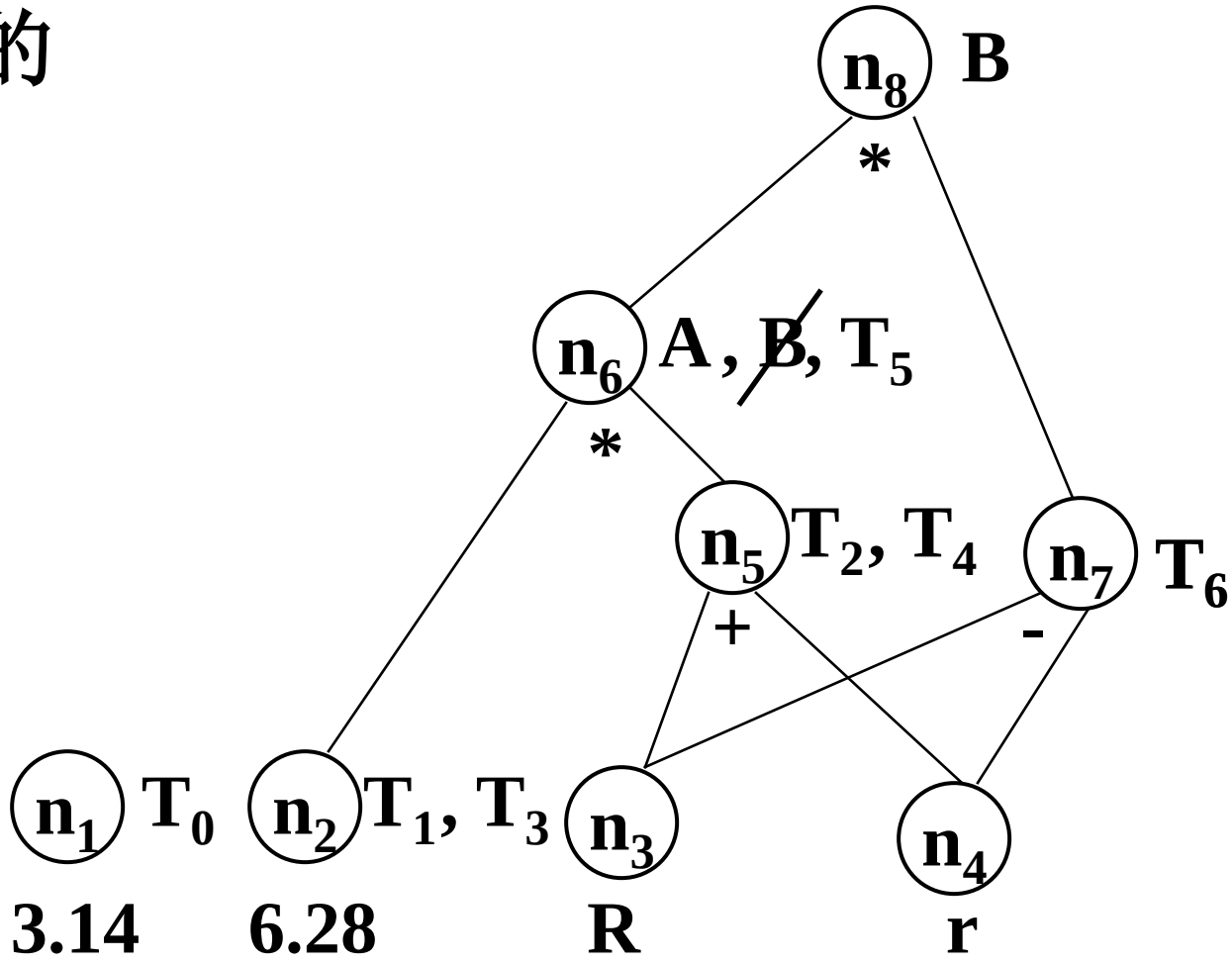
(1) T_0:=3.14
(2) T_1:=6.28
(3) T_3:=6.28
(4) T_2:=R+r
(5) T_4:=T_2
(6) A:=6.28*T_2
(7) T_5:=A
(8) T_6:=R-r
(9) B:=A*T_6



- 优化后的四元式——若只有A和B是出基本块之后活跃的

(1) T_2:=R+r
(2) A:=6.28*T_2
(3) T_6:=R-r
(4) B:=A*T_6

(1) S_1:=R+r
(2) A:=6.28*S_1
(3) S_2:=R-r
(4) B:=A*S_2



DAG的优化

- 合并已知量
- 删除公共子表达式
- 删除无用赋值
- 在基本块外被定值并在基本块内被引用的所有标识符，就是作为叶子结点上标记的那些标识符。
- 在基本块内被定值并且该值在基本块后面可以被引用的所有标识符，就是DAG各结点上的那些附加标识符。

内容线索

- ✓ 概述
- ✓ 局部优化
- ■ 循环优化

循环优化

- 对循环中的代码，可以实行：
  - 代码外提
  - 强度削弱
  - 删除归纳变量（变换循环控制条件）
  - 循环展开
  - 循环合并

代码外提

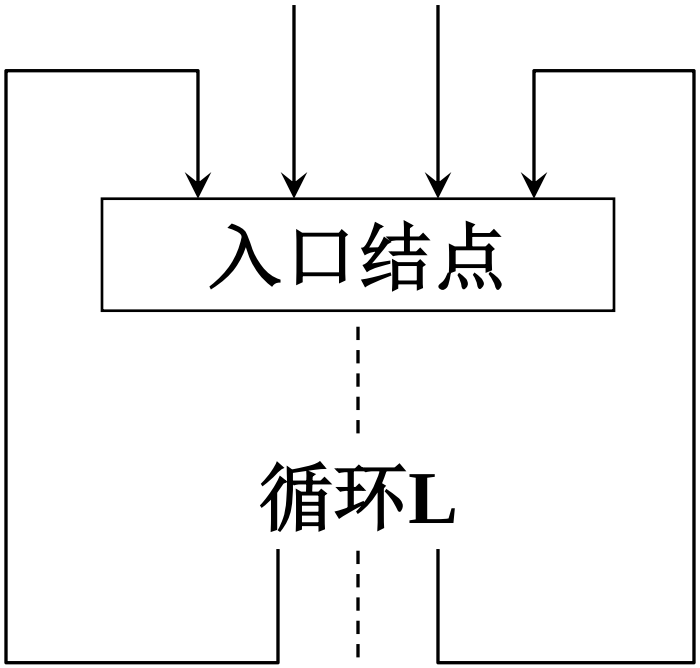
- **循环不变运算**: 对四元式A:=B op C，若B和C是常数，或者到达它们的B和C的定值点都在循环外。
- 变量A在某点d的**定值到达**另一点u（或称变量A的定值点d到达另一点u），是指流图中从d有一通路到达u且该通路上没有A的其它定值。

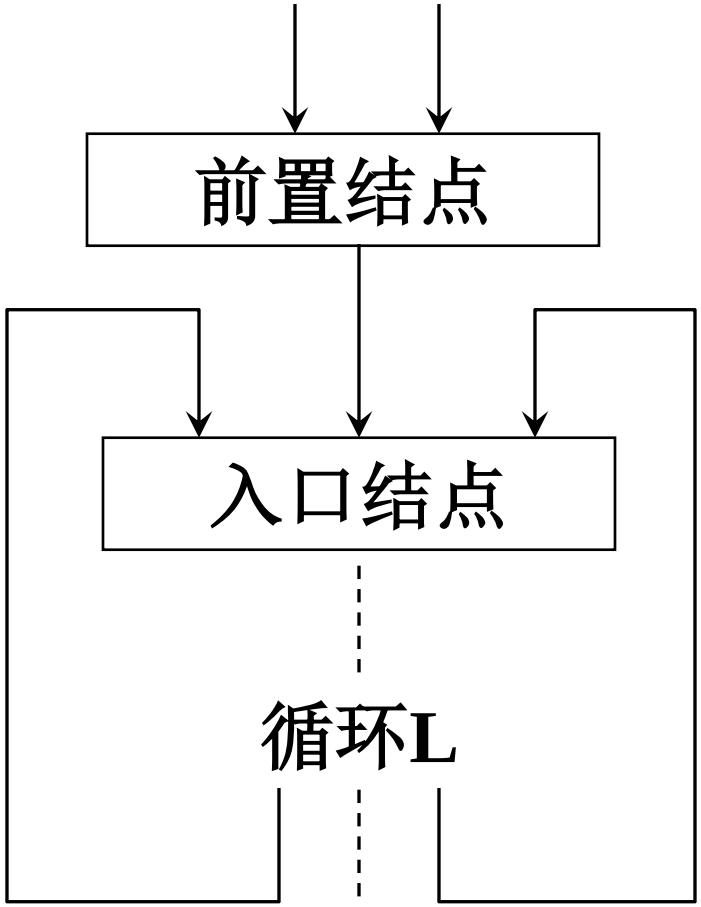
d： A:=B op C

…

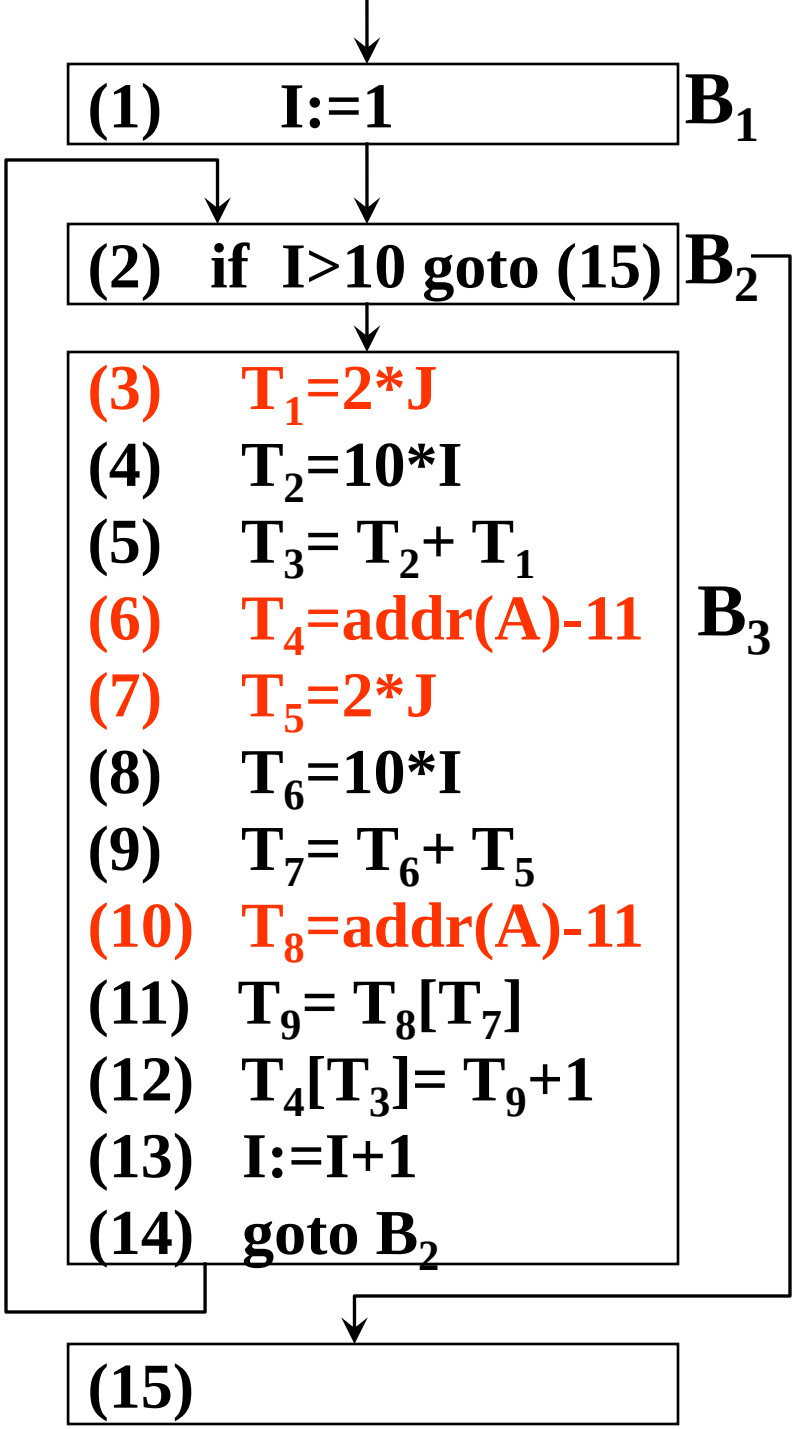
u： D:=A op E

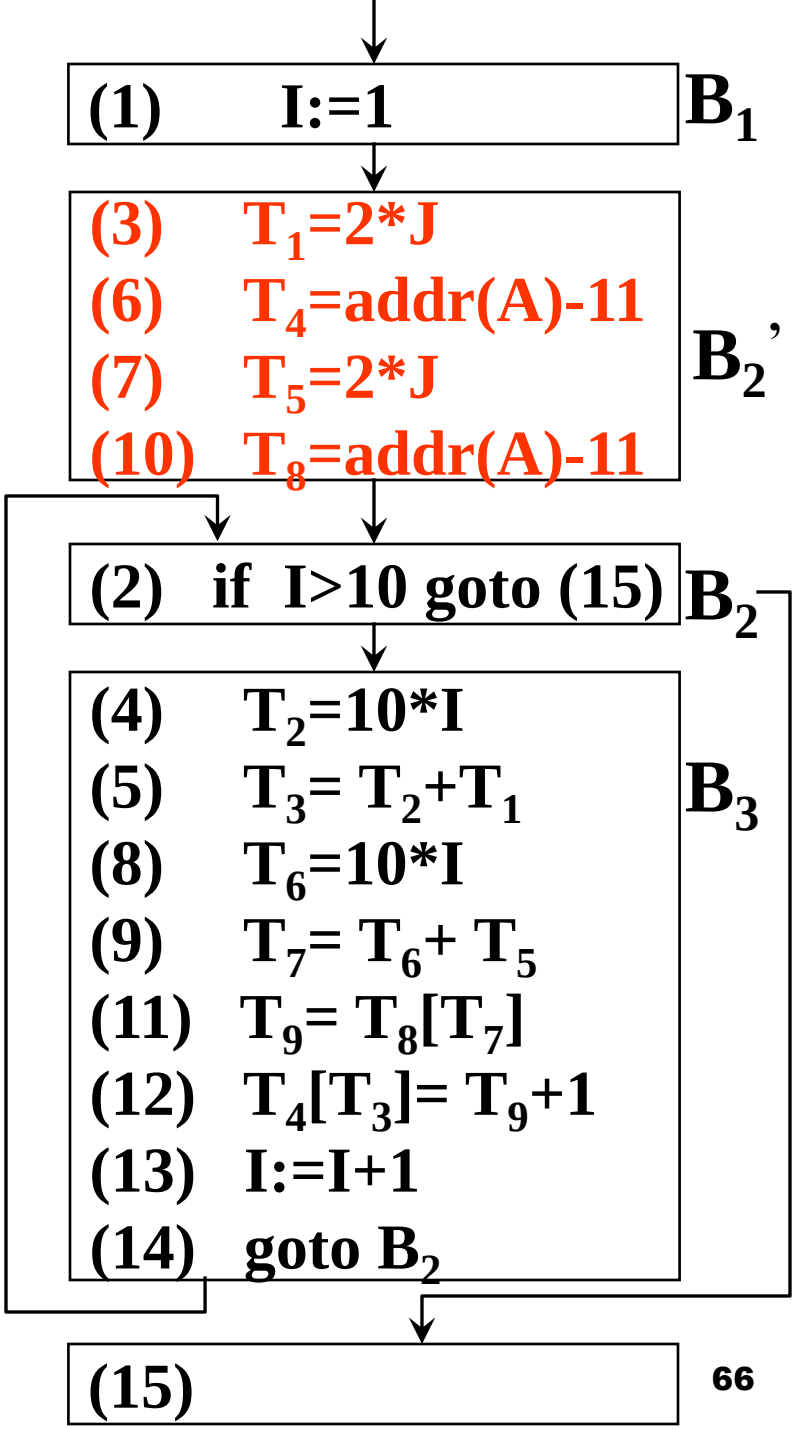
- 把循环不变运算提到循环体外。

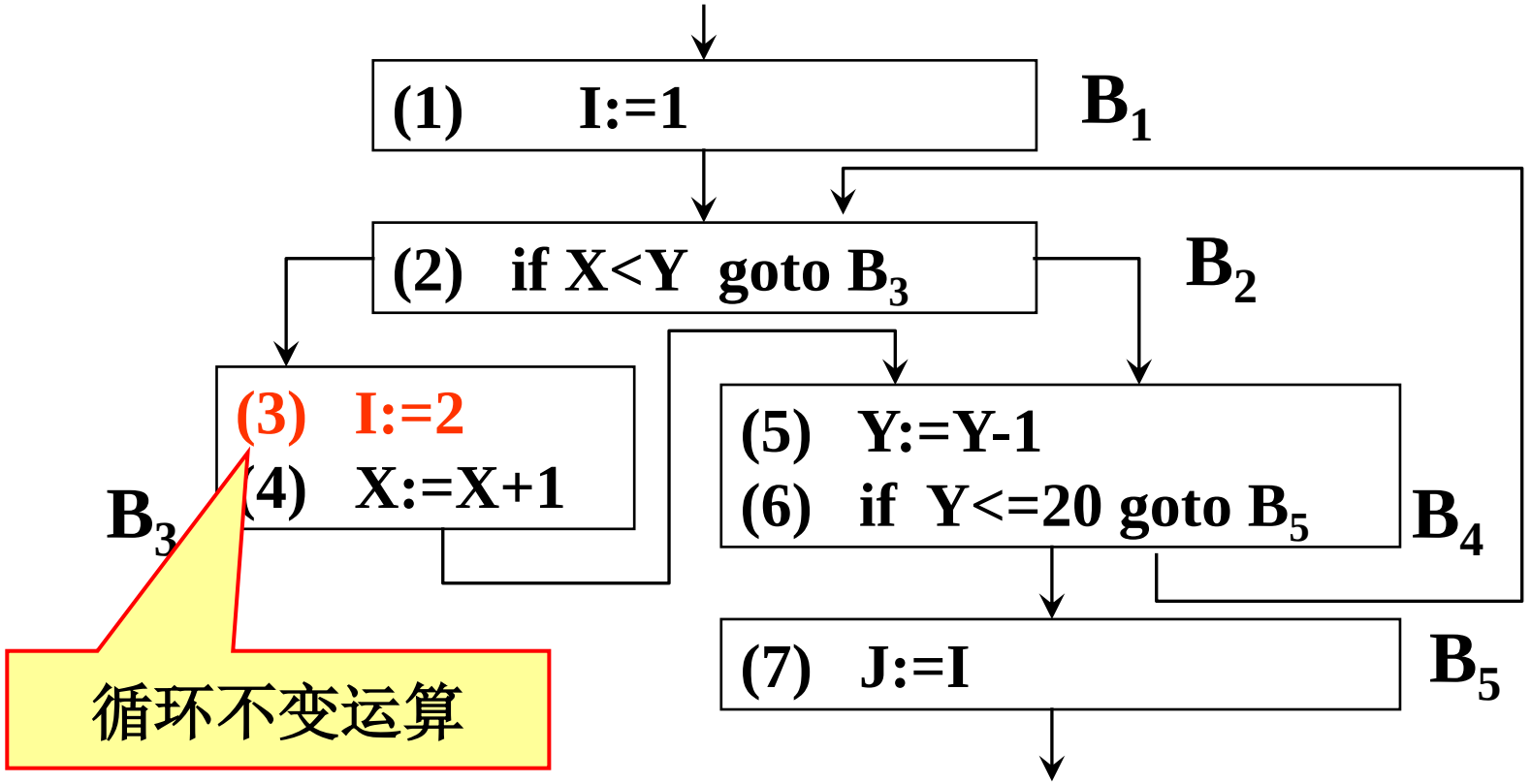

入口结点
循环L


前置结点
入口结点
循环L

```
for I:=1 to 10 do  A[I, 2*J] := A[I, 2*J] + 1
```


(1) I:=1
B1
(2) if I>10 goto (15)
B2
(3) T1=2*J
(4) T2=10*I
(5) T3= T2+ T1
(6) T4=addr(A)-11
(7) T5=2*J
(8) T6=10*I
(9) T7= T6+ T5
(10) T8=addr(A)-11
(11) T9= T8[T7]
(12) T4[T3]= T9+1
(13) I:=I+1
(14) goto B2
B3
(15)

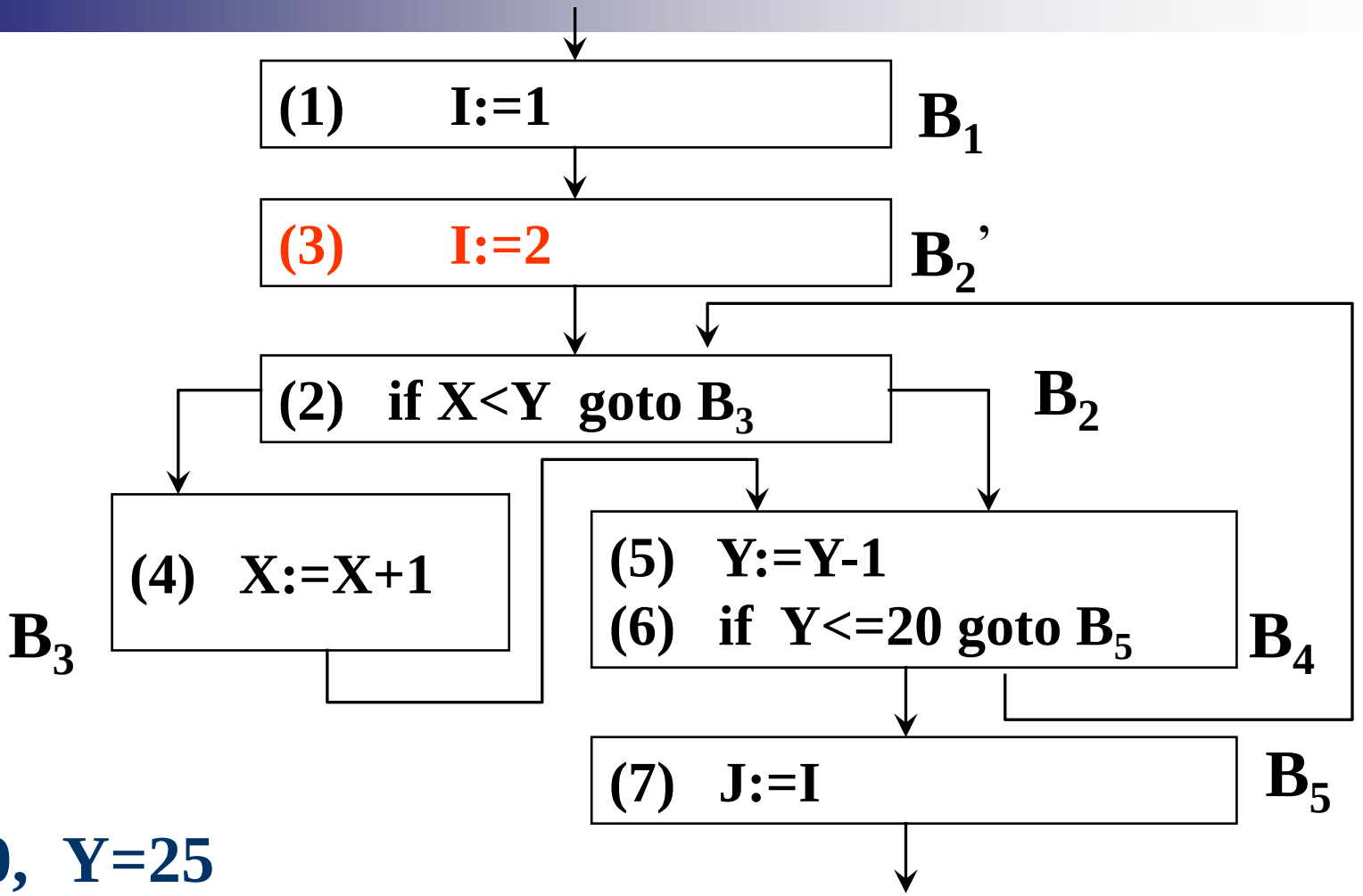

(1) I:=1
B1
(3) T1=2*J
(6) T4=addr(A)-11
(7) T5=2*J
(10) T8=addr(A)-11
B2'
(2) if I>10 goto (15)
B2
(4) T2=10*I
(5) T3= T2+T1
(8) T6=10*I
(9) T7= T6+ T5
(11) T9= T8[T7]
(12) T4[T3]= T9+1
(13) I:=I+1
(14) goto B2
B3
(15)



X=30, Y=25

$B_1 \rightarrow B_2 \rightarrow B_4 \rightarrow B_2 \rightarrow B_4 \rightarrow \cdots \rightarrow B_2 \rightarrow B_4 \rightarrow B_5$

J=1, I=1

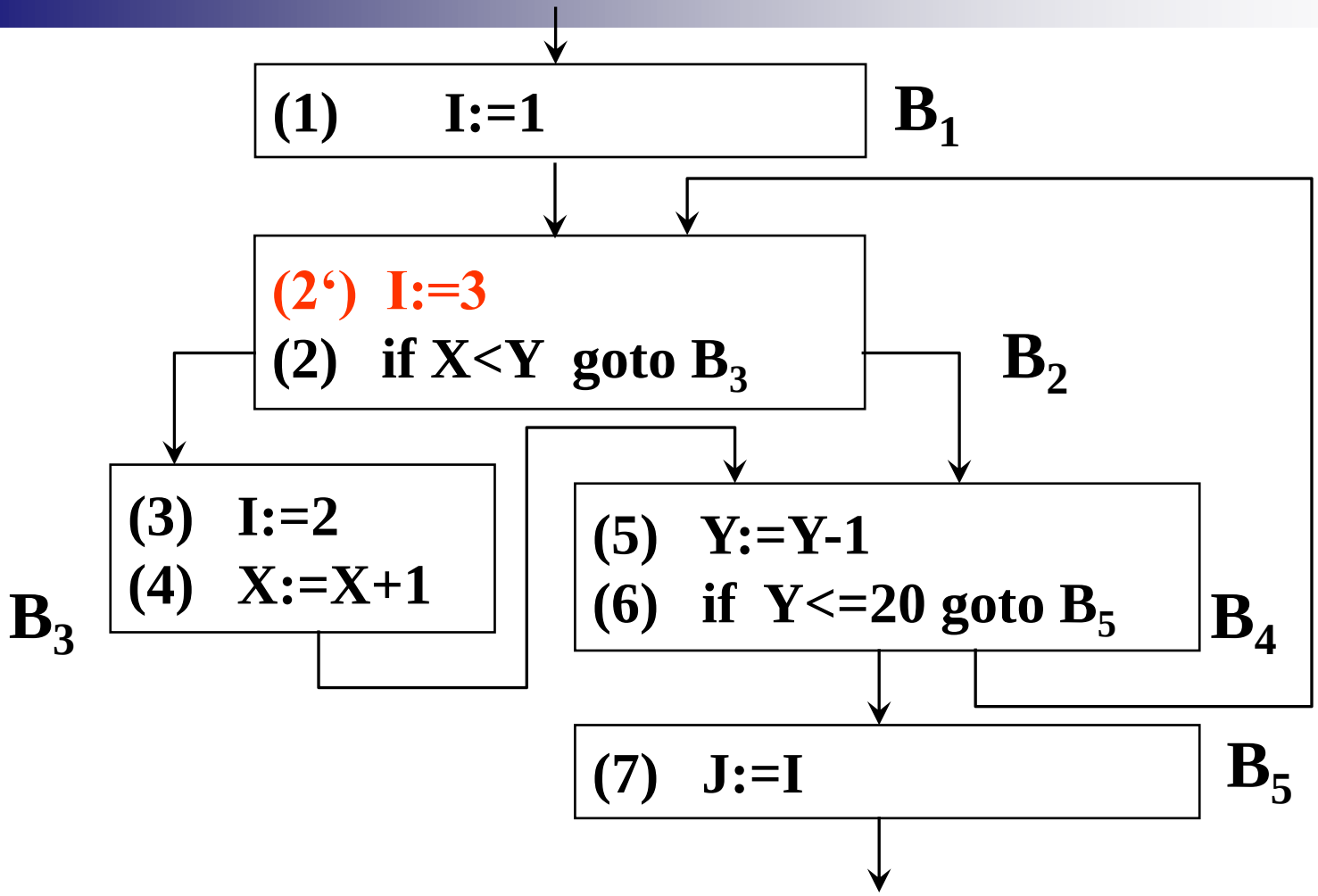


X=30, Y=25

$B_1 \rightarrow B_2 \rightarrow B_4 \rightarrow B_2 \rightarrow B_4 \rightarrow \cdots \rightarrow B_2 \rightarrow B_4 \rightarrow B_5$

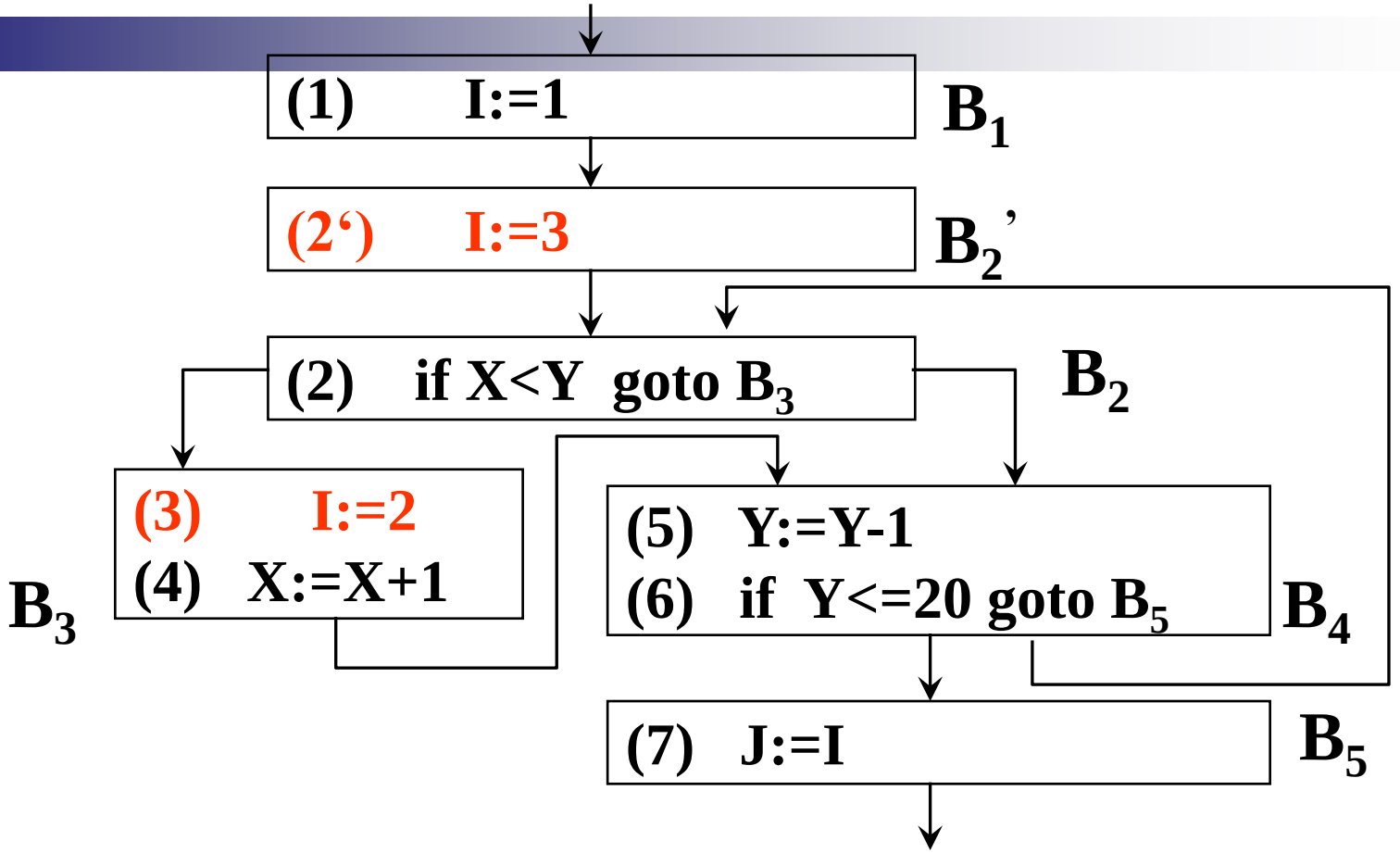
J=2, I=2

代码外提条件：不变运算所在的结点是L所有出口结点的必经结点.



考虑: $B_2 \rightarrow B_3 \rightarrow B_4 \rightarrow B_2 \rightarrow B_4 \rightarrow B_5$

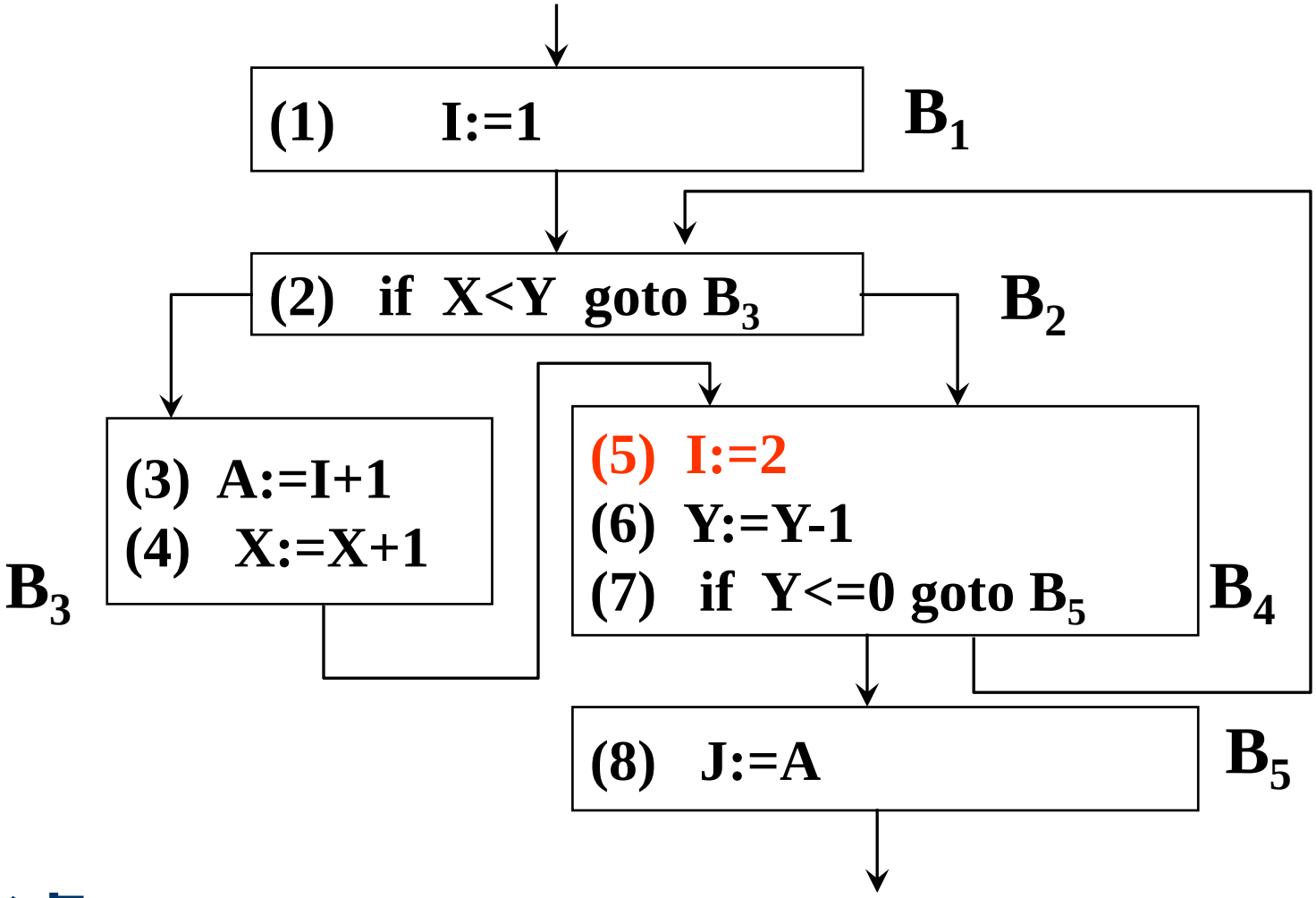
I=3, J=3



考虑: $B_2 \rightarrow B_3 \rightarrow B_4 \rightarrow B_2 \rightarrow B_4 \rightarrow B_5$

I=2, J=2

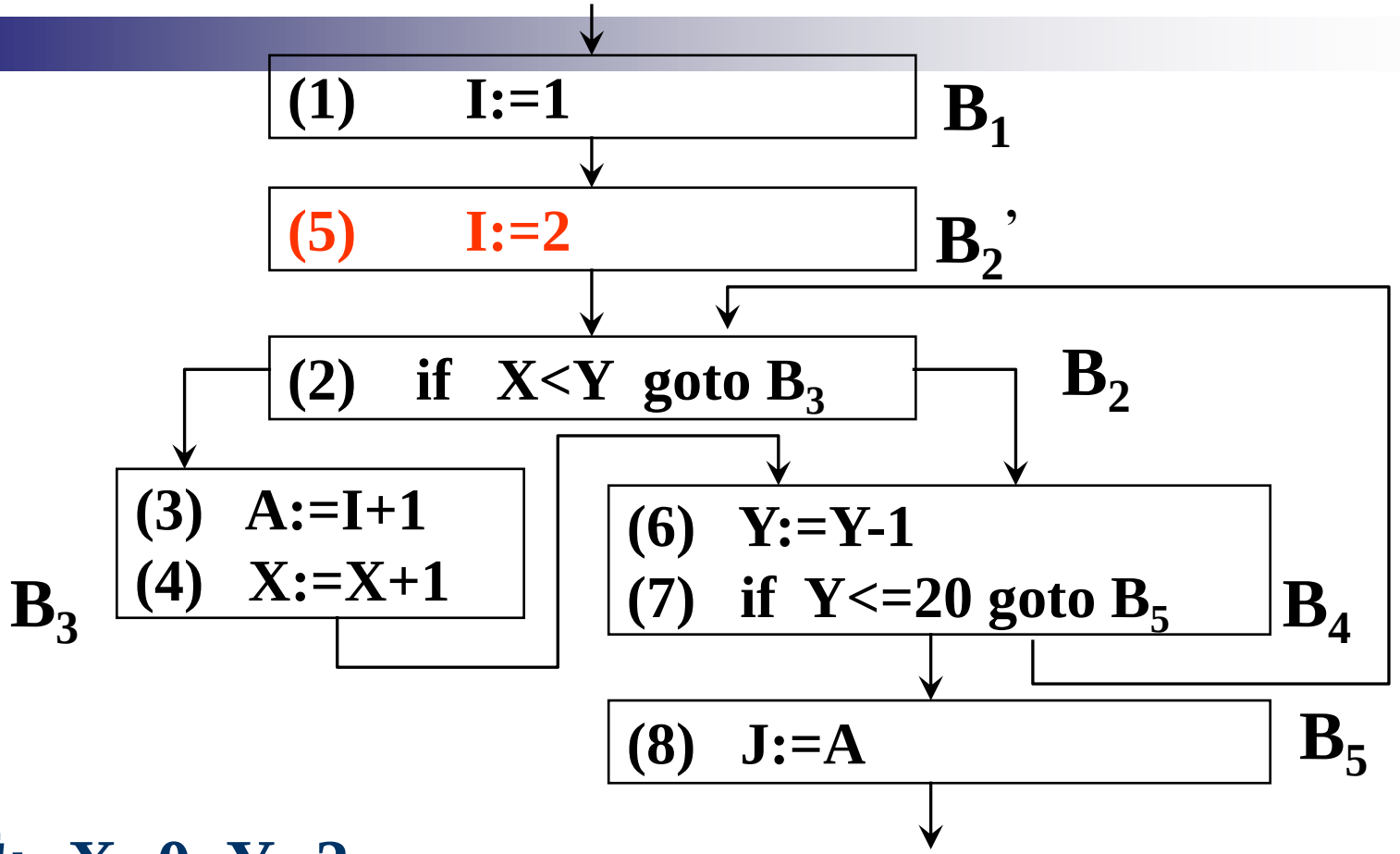
代码外提条件: A在循环中其他地方未再定值,才能把循环不变运算A:=B op C外提;



考虑: X=0, Y=2

$B_2 \rightarrow B_3 \rightarrow B_4 \rightarrow B_2 \rightarrow B_4 \rightarrow B_5$

A=2, J=2



考虑: X=0, Y=2

$B_2 \rightarrow B_3 \rightarrow B_4 \rightarrow B_2 \rightarrow B_4 \rightarrow B_5$

A=3, J=3

S(A:=B OP C)外提条件:循环中所有A的引用点只有S中的A的定值才能到达。

查找循环L的不变运算的算法

- 依次查看L中各基本块的每个四元式，如果它的每个运算对象或为常数，或者定值点在 L外，则将此四元式标记为"不变运算";
- 重复第3步直至没有新的四元式被标记为"不变运算"为止;
- 依次查看尚未被标记为"不变运算"的四元式，如果它的每个运算对象或为常数，或定值点在L之外，或只有一个到达-定值点且该点上的四元式已被标记为"不变运算"，则把被查看的四元式标记为"不变运算"。

代码外提算法

1. 求出L的所有不变运算
2. 对每个不变运算s:A:=B op C 或 A:=op B 或 A:=B检查是否满足条件(1)或(2)

   (1)

   (i) s所在的结点是L所有出口结点的必经结点;

   (ii) A在L中其他地方未再定值;

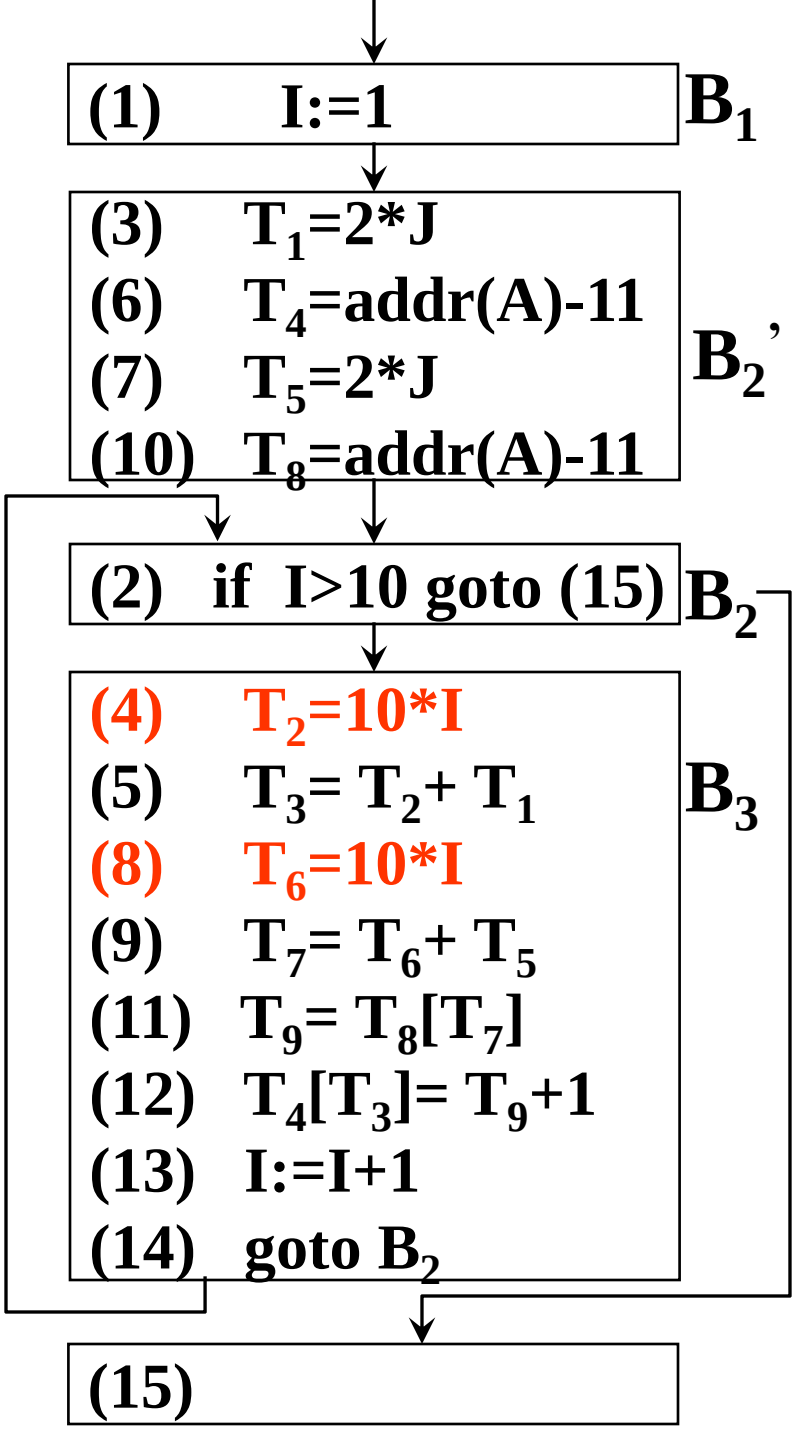
   (iii) L中所有A的引用点只有s中的A的定值才能到达。

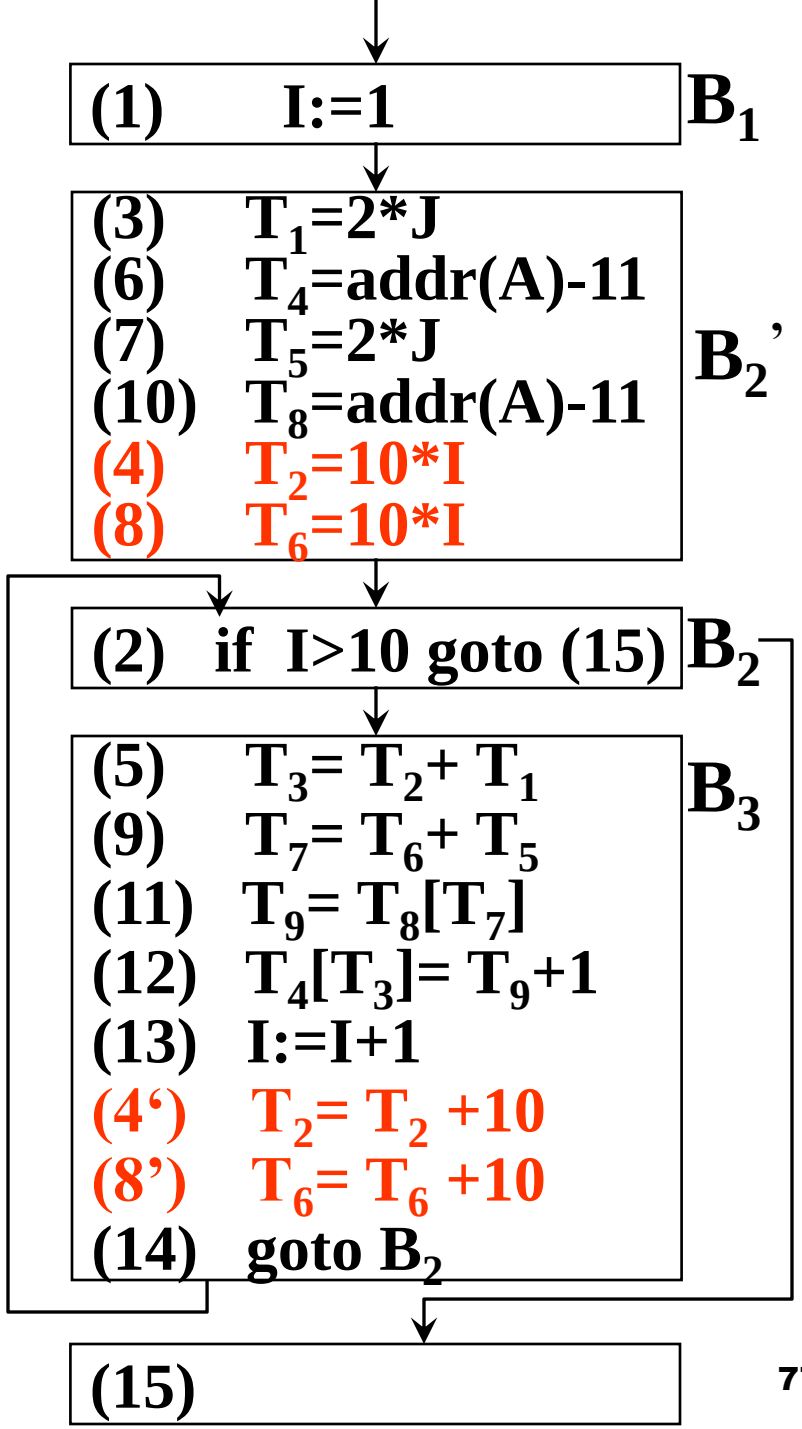
(2) A在离开L后不再是活跃的，并且条件(1)的(ii)和(iii)成立。所谓**A在离开L后不是活跃的**是指，A在L的任何出口结点的后继结点入口处不是活跃的。

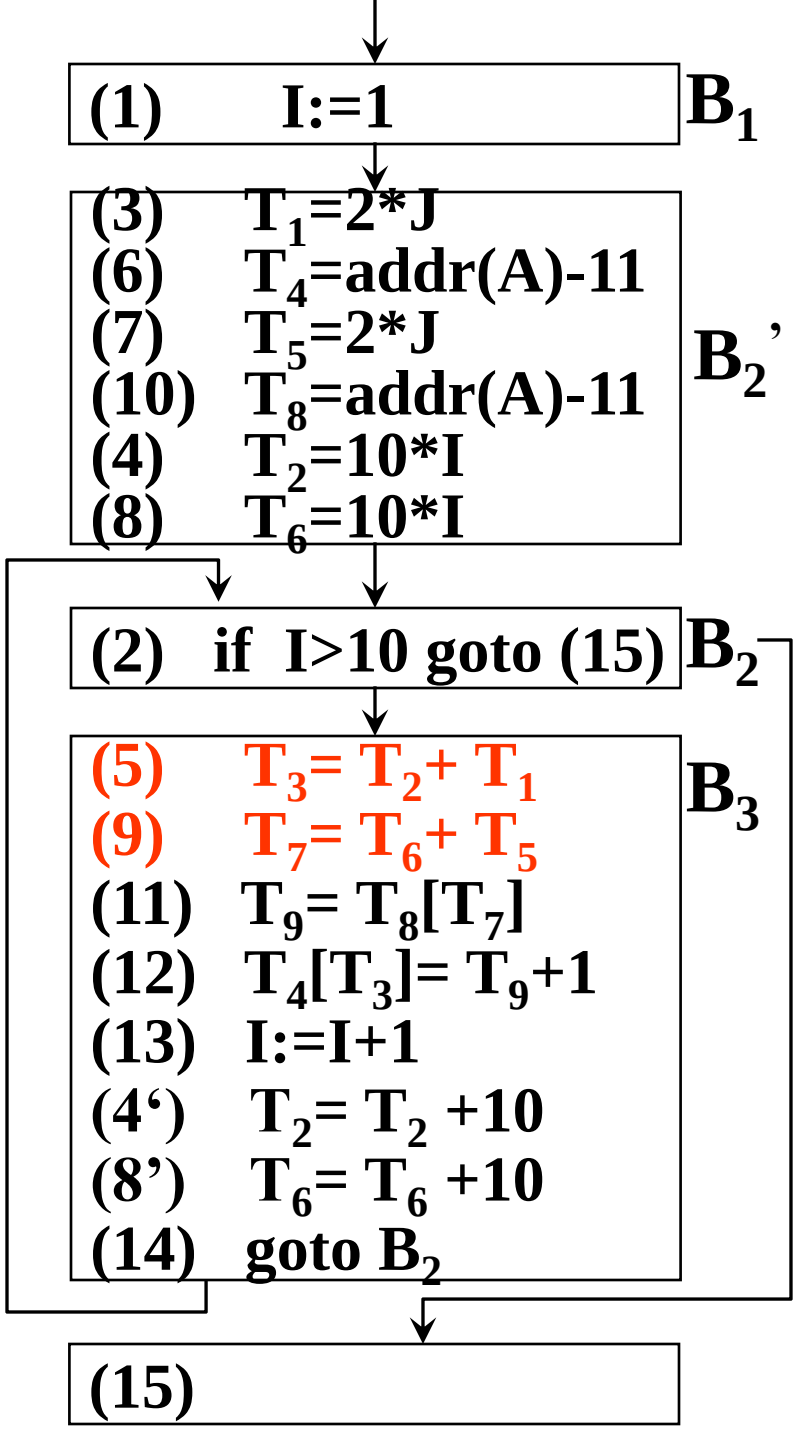
3.按步骤1所找出的不变运算的次序，依次把符合条件2的条件(1)或(2)的不变运算s外提到L的前置结点中。但是，如果s的运算对象(B或C)是在L中定值的，那么，只有当这些定值四元式都已外提到前置结点中时，才能把s也外提到前置结点中。

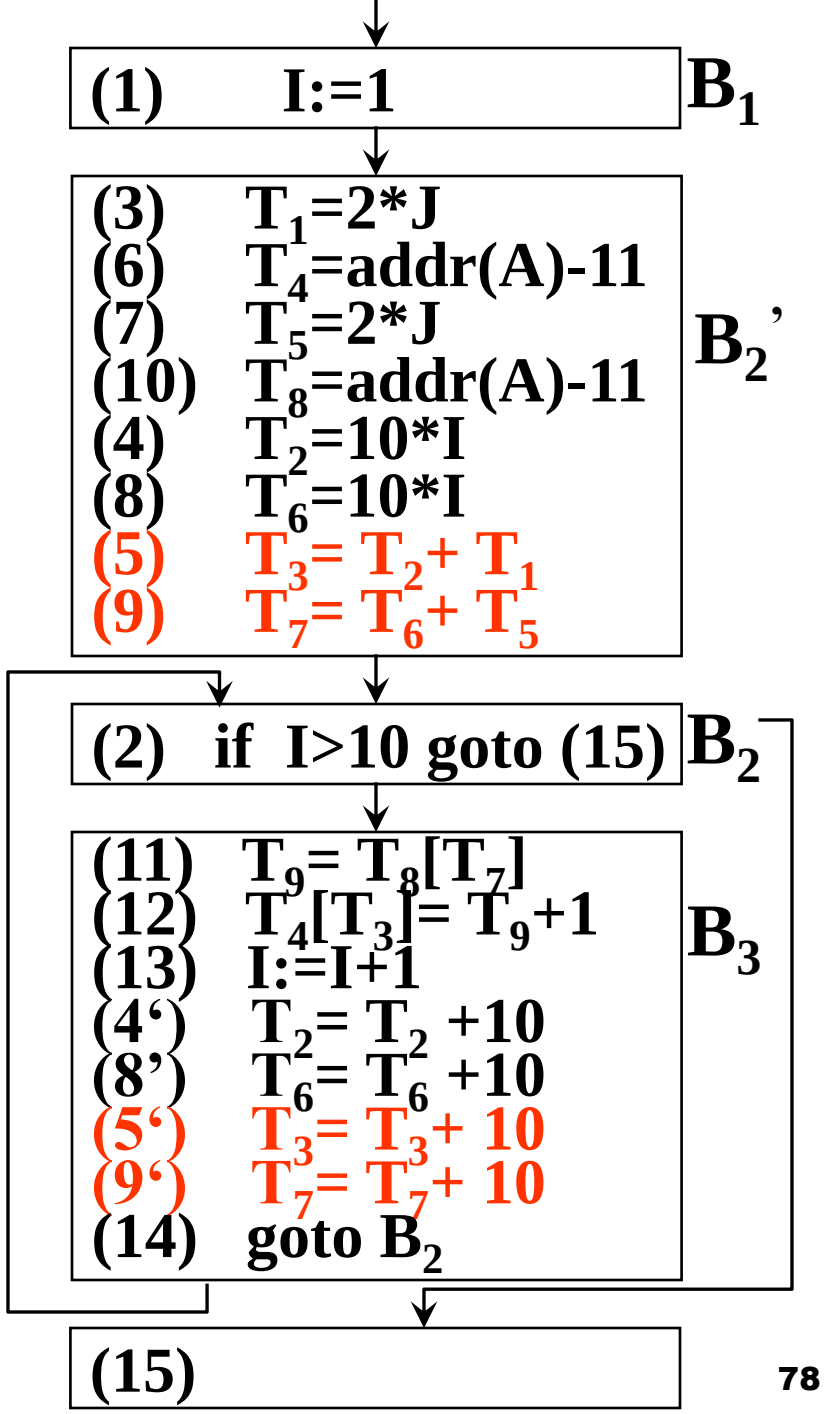
强度消弱

- 把程序中执行时间较长的运算转换为执行时间较短的运算。
  - 如把循环中的乘法运算用递归加法运算替换


(1) I:=1
B1
(3) T1=2*J
(6) T4=addr(A)-11
(7) T5=2*J
(10) T8=addr(A)-11
B2'
(2) if I>10 goto (15)
B2
(4) T2=10*I
(5) T3= T2+ T1
(8) T6=10*I
(9) T7= T6+ T5
(11) T9= T8[T7]
(12) T4[T3]= T9+1
(13) I:=I+1
(14) goto B2
B3
(15)


(1) I:=1
B1
(3) T1=2*J
(6) T4=addr(A)-11
(7) T5=2*J
(10) T8=addr(A)-11
(4) T2=10*I
(8) T6=10*I
B2'
(2) if I>10 goto (15)
B2
(5) T3= T2+ T1
(9) T7= T6+ T5
(11) T9= T8[T7]
(12) T4[T3]= T9+1
(13) I:=I+1
(4') T2= T2 +10
(8') T6= T6 +10
(14) goto B2
B3
(15)


(1) I:=1
B1
(3) T1=2*J
(6) T4=addr(A)-11
(7) T5=2*J
(10) T8=addr(A)-11
(4) T2=10*I
(8) T6=10*I
B2'
(2) if I>10 goto (15)
B2
(5) T3= T2+ T1
(9) T7= T6+ T5
(11) T9= T8[T7]
(12) T4[T3]= T9+1
(13) I:=I+1
(4‘) T2= T2 +10
(8’) T6= T6 +10
(14) goto B2
B3
(15)

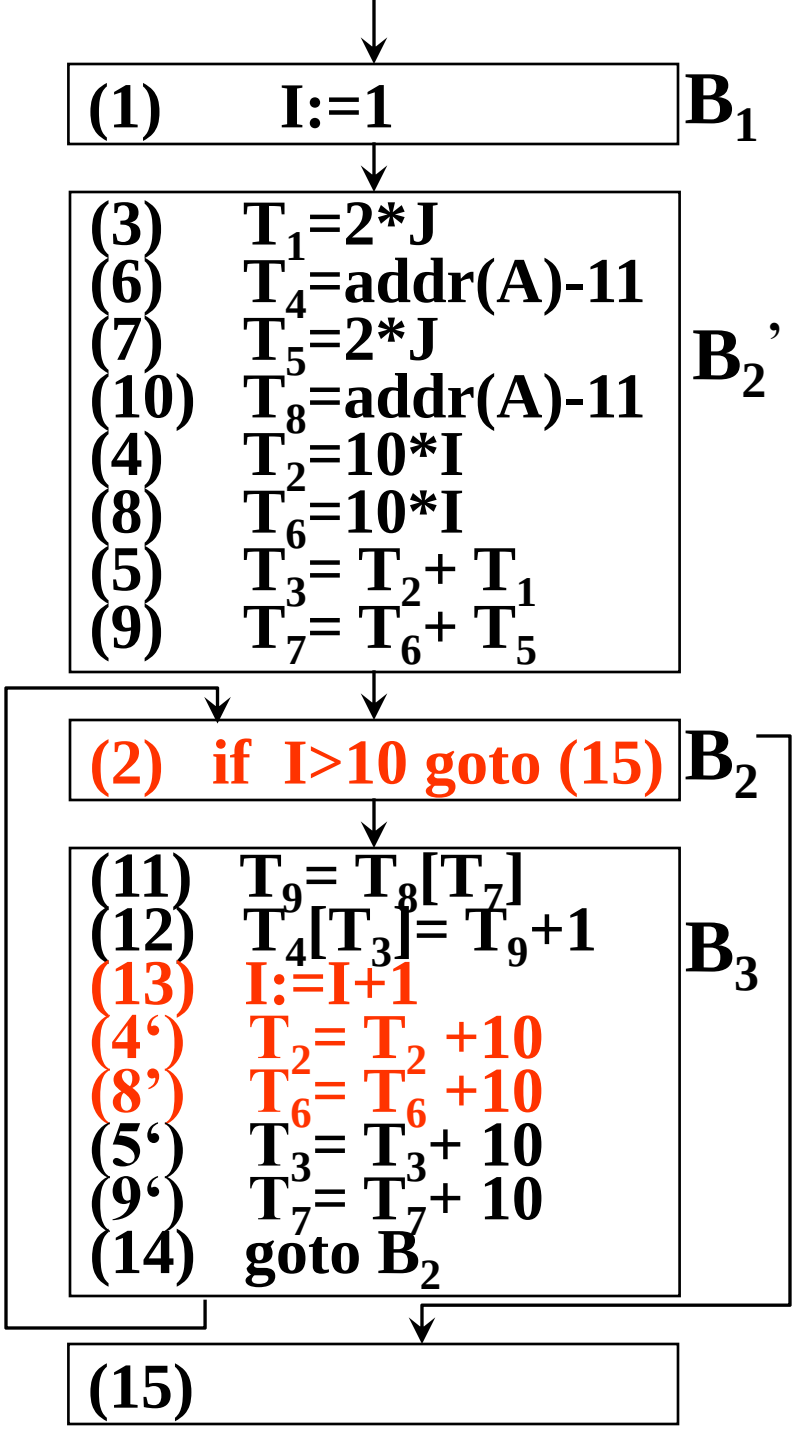

(1) I:=1
B1
(3) T1=2*J
(6) T4=addr(A)-11
(7) T5=2*J
(10) T8=addr(A)-11
(4) T2=10*I
(8) T6=10*I
(5) T3= T2+ T1
(9) T7= T6+ T5
B2'
(2) if I>10 goto (15)
B2
(11) T9= T8[T7]
(12) T4[T3]= T9+1
(13) I:=I+1
(4‘) T2= T2 +10
(8’) T6= T6 +10
(5‘) T3= T3+ 10
(9‘) T7= T7+ 10
(14) goto B2
B3
(15)

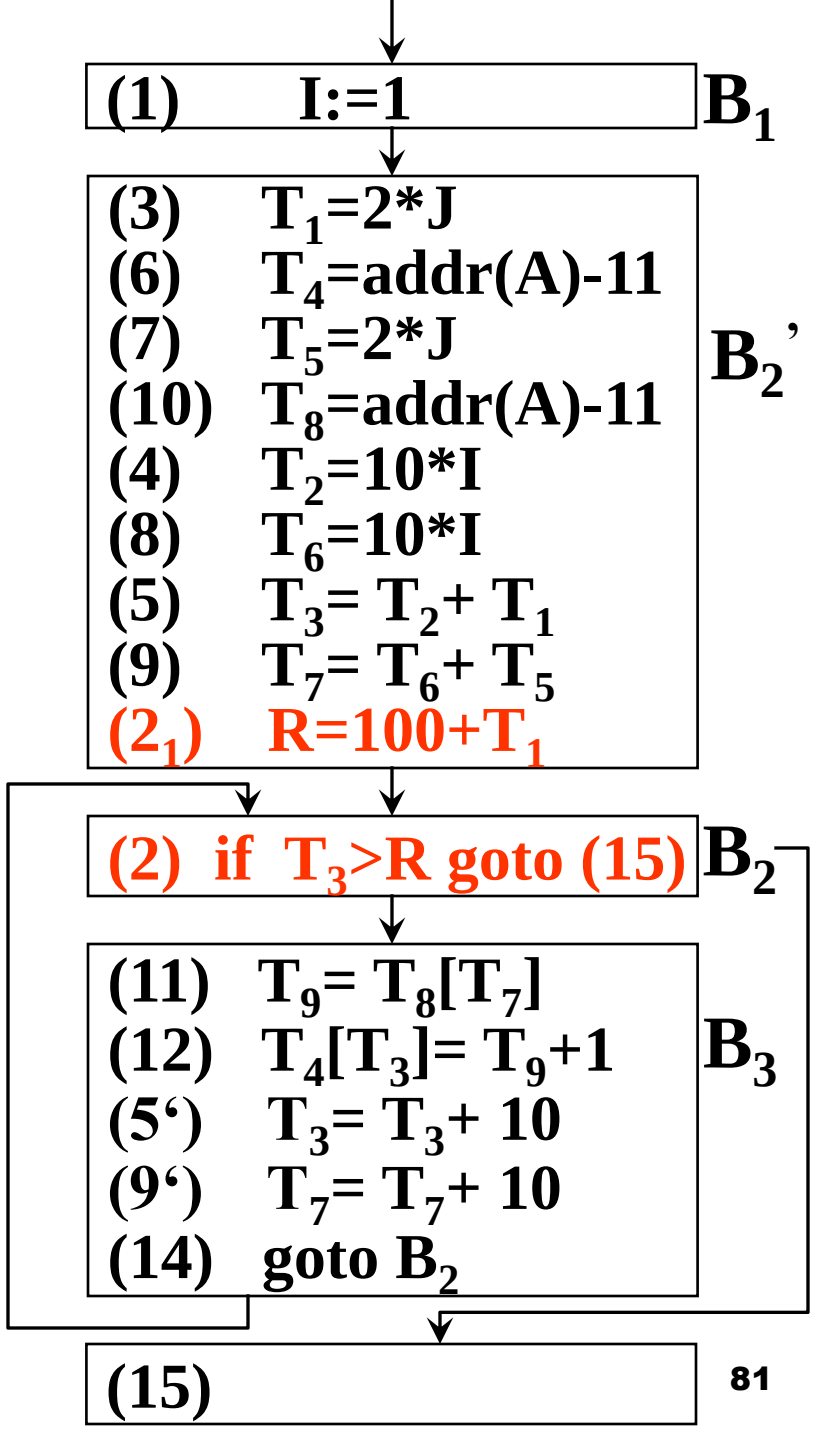
总结

- 强度削弱主要是对与归纳变量有线性关系的变量赋值进行削弱；
- 经过强度削弱后，循环中可能出现一些新的无用赋值；
- 对于削弱下标变量地址计算的强度非常有效。

删除归纳变量

- 如果循环中对变量I只有唯一的形如I:=I±C的赋值，且其中C为循环不变量，则称I为循环中的**基本归纳变量**。
- 如果I是循环中一基本归纳变量，J在循环中的定值总是可化归为I的同一线性函数，也即$J=C_1*I \pm C_2$，其中C_1和C_2都是循环不变量，则称J是**归纳变量**，并称它**与I同族**。
- 一个基本归纳变量也是一归纳变量。


(1) I:=1
B1
(3) T1=2*J
(6) T4=addr(A)-11
(7) T5=2*J
(10) T8=addr(A)-11
(4) T2=10*I
(8) T6=10*I
(5) T3= T2+ T1
(9) T7= T6+ T5
B2'
(2) if I>10 goto (15)
B2
(11) T9= T8[T7]
(12) T4[T3]= T9+1
(13) I:=I+1
(4') T2= T2 +10
(8') T6= T6 +10
(5') T3= T3+ 10
(9') T7= T7+ 10
(14) goto B2
B3
(15)


(1) I:=1
B1
(3) T1=2*J
(6) T4=addr(A)-11
(7) T5=2*J
(10) T8=addr(A)-11
(4) T2=10*I
(8) T6=10*I
(5) T3= T2+ T1
(9) T7= T6+ T5
(21) R=100+T1
B2'
(2) if T3>R goto (15)
B2
(11) T9= T8[T7]
(12) T4[T3]= T9+1
(5') T3= T3+ 10
(9') T7= T7+ 10
(14) goto B2
B3
(15)

强度削弱和删除归纳变量的统一算法

1. 利用循环不变运算信息，找出循环中所有基本归纳变量。
2. 找出所有其它归纳变量A，并找出A与已知基本归纳变量X的同族线性函数关系 $F_A(X)$。
3. 对2中找出的每一归纳变量A，进行强度削弱。
4. 删除对归纳变量的无用赋值。
5. 删除基本归纳变量。如果基本归纳变量B在循环出口之后不是活跃的，并且在循环中，除在其自身的递归赋值中被引用外，只在形如

   if B rop Y goto L

   中被引用，则可选取一与B同族的归纳变量M来替换B进行条件控制。最后删除循环中对B的递归赋值的代码。