



Bài 13

Tối ưu mã



Mở đầu

■ Yêu cầu

- ☐ Chương trình sau khi tối ưu phải tương đương
- ☐ Tốc độ thực hiện trung bình tăng
- ☐ Hiệu quả đạt được tương xứng với công sức bỏ ra

■ Có thể tối ưu mã vào lúc nào

- ☐ Mã nguồn- do người lập trình (giải thuật)
- ☐ Mã trung gian
- ☐ Mã đích



Các mức độ tối ưu mã trung gian

- Tối ưu cục bộ
- Tối ưu trong khối cơ sở
- Tối ưu trên đồ thị



Tối ưu cục bộ

1. Kỹ thuật để cải tiến mã đích một cách cục bộ.
2. Một phương pháp để cải tiến chương trình trung gian (CT đích) bằng cách xem xét một dãy lệnh trong mã TG (đích) và thay thế chúng bằng những đoạn mã ngắn hơn và hiệu quả hơn

Xu hướng chính

1. Loại bỏ lệnh dư thừa
2. Thông tin dòng điều khiển
3. Giảm lược biểu thức đại số
4. Sử dụng các đặc trưng ngôn ngữ

Tối ưu cục bộ

- Tính toán biểu thức hằng

$x := 32$ trở thành $x := 64$
 $x := x + 32$

- Mã không đến được

goto L2
 $x := x + 1$ ← Không cần

- Tối ưu dòng điều khiển

goto L1 trở thành goto L2

...

L1: goto L2 ← Không cần nếu không còn lệnh sau L2

Tối ưu cục bộ

■ Giảm lược biểu thức đại số

$x := x + 0 \leftarrow$ Không cần

■ Mã chết

$x := 32 \leftarrow$ x không được dùng trong những lệnh tiếp theo

$y := x + y \rightarrow y := y + 32$

■ Giảm chi phí tính toán


$x := x * 2 \rightarrow x := x + x$

$\rightarrow x := x \ll 1$



Tối ưu trong từng khối cơ bản

1. Loại bỏ biểu thức con chung
2. Tính giá trị hằng
3. Copy Propagation
4. Loại mã chết...



Khối cơ bản (basic block)

Chuỗi các lệnh kế tiếp nhau trong đó dòng điều khiển đi vào lệnh đầu tiên của khối và ra ở lệnh cuối cùng của khối mà không bị dừng hoặc rẽ nhánh.

Ví dụ

$t1 := a * a$

$t2 := a * b$

$t3 := 2 * t2$

$t4 := t1 + t2$

$t5 := b * b$

$t6 := t4 + t5$



Giải thuật phân chia các khối cơ bản

Input: Dãy lệnh ba địa chỉ.

Output: Danh sách các khối cơ bản với mã lệnh ba địa chỉ của từng khối

Phương pháp:

1. Xác định tập các lệnh đầu (leader), của từng khối cơ bản
 - i) Lệnh đầu tiên của chương trình là lệnh đầu.
 - ii) Bất kỳ lệnh nào là đích nhảy đến của các lệnh GOTO có hoặc không có điều kiện là lệnh đầu
 - iii) Bất kỳ lệnh nào đi sau lệnh GOTO có hoặc không có điều kiện là lệnh đầu
2. Với mỗi lệnh đầu, khối cơ bản bao gồm nó và tất cả các lệnh tiếp theo không phải là lệnh đầu hay lệnh kết thúc chương trình



Ví dụ

```
(1) prod := 0
(2) i := 1
(3) t1 := 4 * i
(4) t2 := a[t1]
(5) t3 := 4 * i
(6) t4 := b[t3]
(7) t5 := t2 * t4
(8) t6 := prod + t5
(9) prod := t6
(10) t7 := i + 1
(11) i := t7
(12) if i <= 20 goto (3)
```

- Lệnh (1) là lệnh đầu theo quy tắc i,
- Lệnh (3) là lệnh đầu theo quy tắc ii
- Lệnh sau lệnh (12) là lệnh đầu theo quy tắc iii.
- Các lệnh (1) và (2) tạo nên khối cơ bản thứ nhất.
- Lệnh (3) đến (12) tạo nên khối cơ bản thứ hai.



Loại biểu thức con chung

Ví dụ: Lệnh mã nguồn

$a[i+1] = b[i+1]$

$t1 = i + 1$

$t2 = b[t1]$

$t3 = i + 1$

$a[t3] = t2$

$t1 = i + 1$

$t2 = b[t1]$

$t3 = i + 1$ ← Không cần

$a[t1] = t2$

Truyền hằng (Constant Propagation)

i là hằng

```
i = 4  
t1 = i+1  
t2 = b[t1]  
a[t1] = t2
```

```
i = 4  
t1 = 5  
t2 = b[t1]  
a[t1] = t2
```

```
i = 4  
t1 = 5  
t2 = b[5]  
a[5] = t2
```

Mã nhận được:

```
i = 4  
t2 = b[5]  
a[5] = t2
```



Copy Propagation

- $t2 = t1 ;$
- $t3 = t2 * t1 ;$
- $t4 = t3 ;$
- $t5 = t3 * t2 ;$
- $c = t5 + t4 ;$
- $t3 = t1 * t1 ;$
- $t5 = t3 * t1 ;$
- $c = t5 + t3 ;$



Tối ưu trên CFG (Control Flow Graph)

- Vấn đề cần quan tâm
 - ☐ Loại bỏ biểu thức con chung
 - ☐ Tính các biểu thức hằng
 - ☐ Loại mã chết
 - ☐ Loại những dư thừa cục bộ...
- Ứng dụng một phương pháp tối ưu dẫn đến việc tạo ra những đoạn mã có thể ứng dụng phương pháp tối ưu khác.

Mã ba địa chỉ của Quick Sort

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 $i \geq j$ goto (23)
14	$t_6 = 4 * i$
15	$x = a[t_6]$

16	$t_7 = 4 * j$
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	$t_{11} = 4 * j$
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	$a[t_{15}] = x$



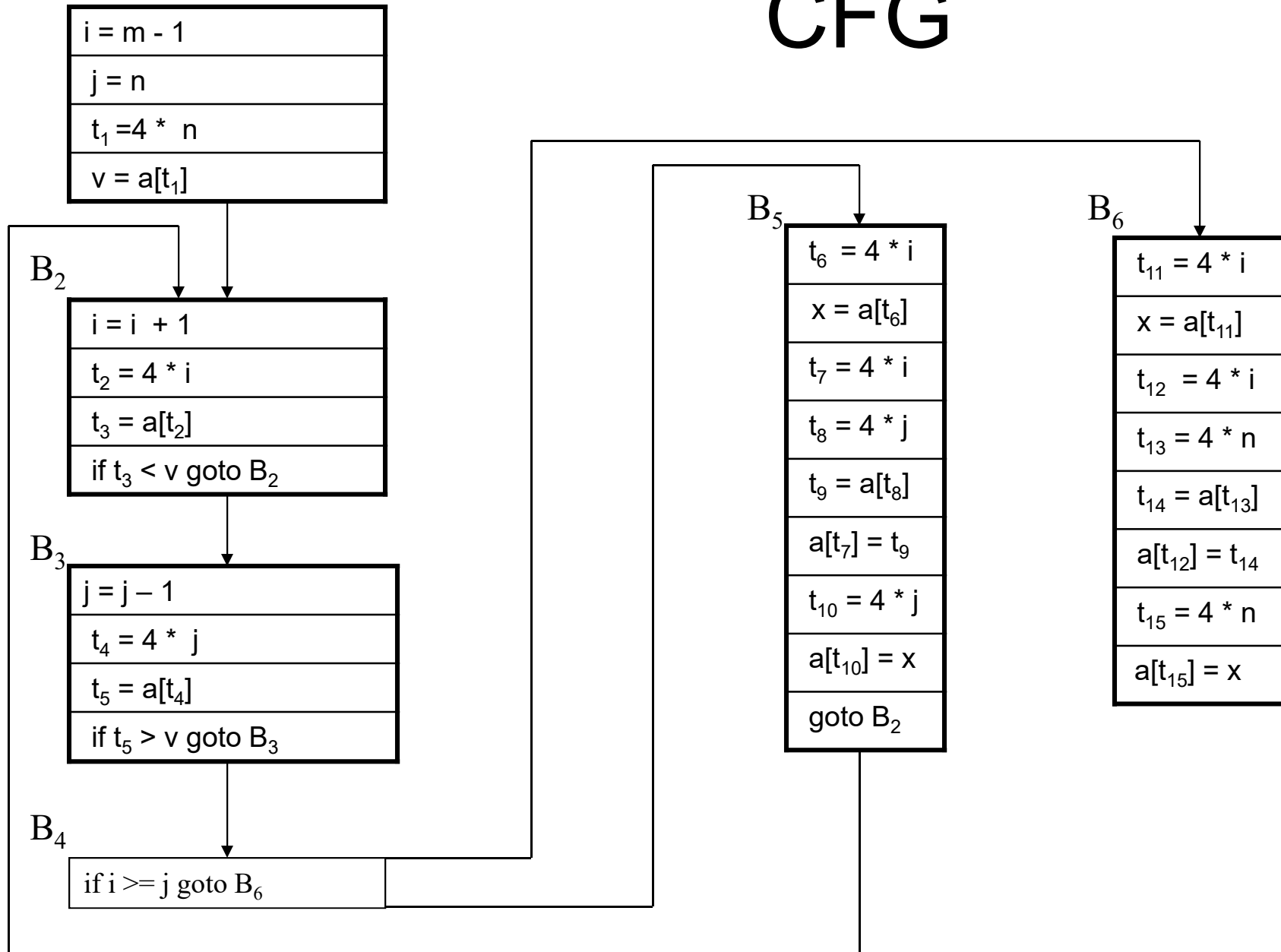
1	$i = m - 1$
2	$j = n$
3	$t_1 = 4 * n$
4	$v = a[t_1]$
5	$i = i + 1$
6	$t_2 = 4 * i$
7	$t_3 = a[t_2]$
8	if $t_3 < v$ goto (5)
9	$j = j - 1$
10	$t_4 = 4 * j$
11	$t_5 = a[t_4]$
12	if $t_5 > v$ goto (9)
13	if $i \geq j$ goto (23)
14	$t_6 = 4 * i$
15	$x = a[t_6]$

Xác định khối cơ bản

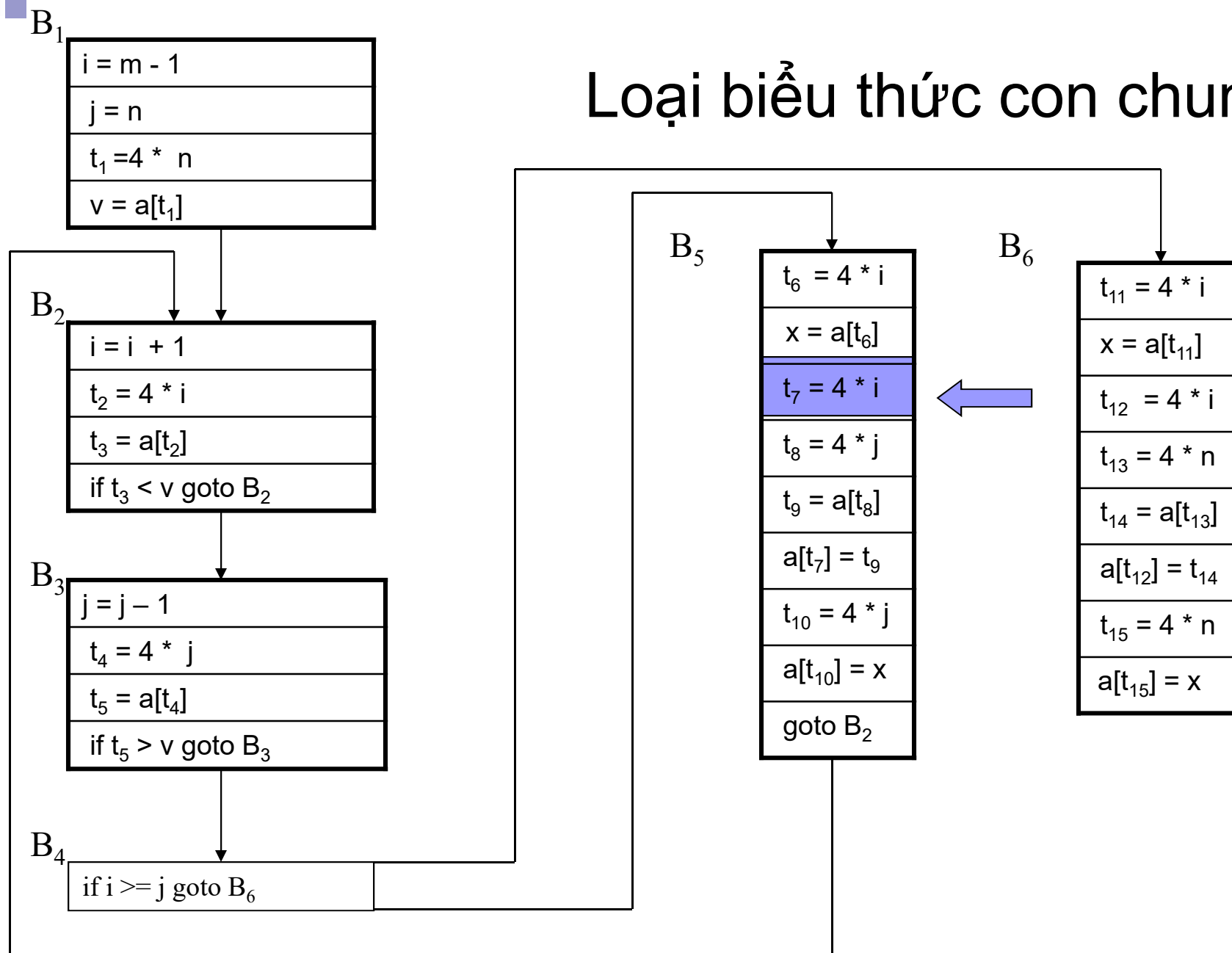
16	$t_7 = 4 * i$
17	$t_8 = 4 * j$
18	$t_9 = a[t_8]$
19	$a[t_7] = t_9$
20	$t_{10} = 4 * j$
21	$a[t_{10}] = x$
22	goto (5)
23	$t_{11} = 4 * i$
24	$x = a[t_{11}]$
25	$t_{12} = 4 * i$
26	$t_{13} = 4 * n$
27	$t_{14} = a[t_{13}]$
28	$a[t_{12}] = t_{14}$
29	$t_{15} = 4 * n$
30	$a[t_{15}] = x$

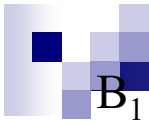


CFG



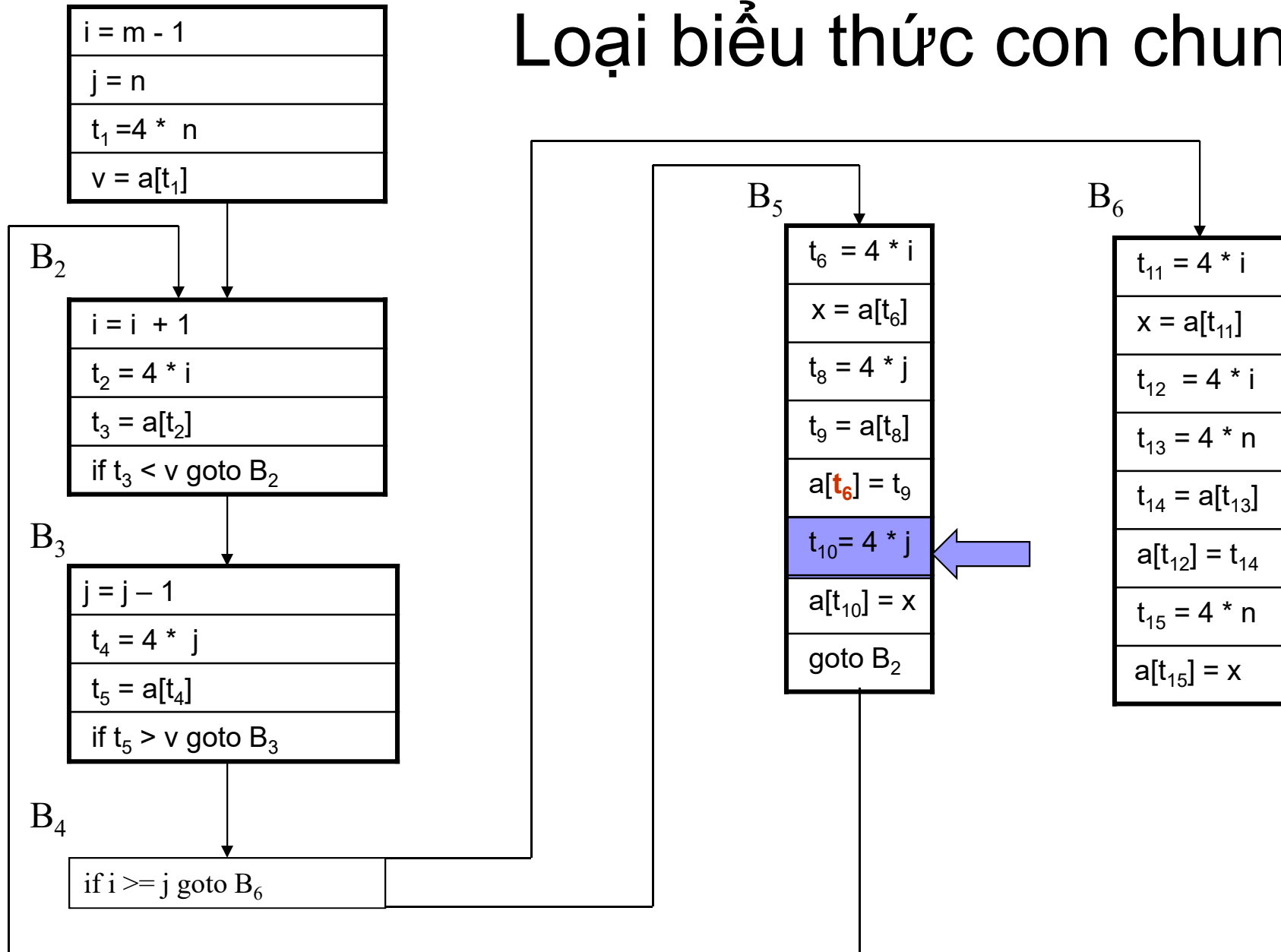
Loại biểu thức con chung





B₁

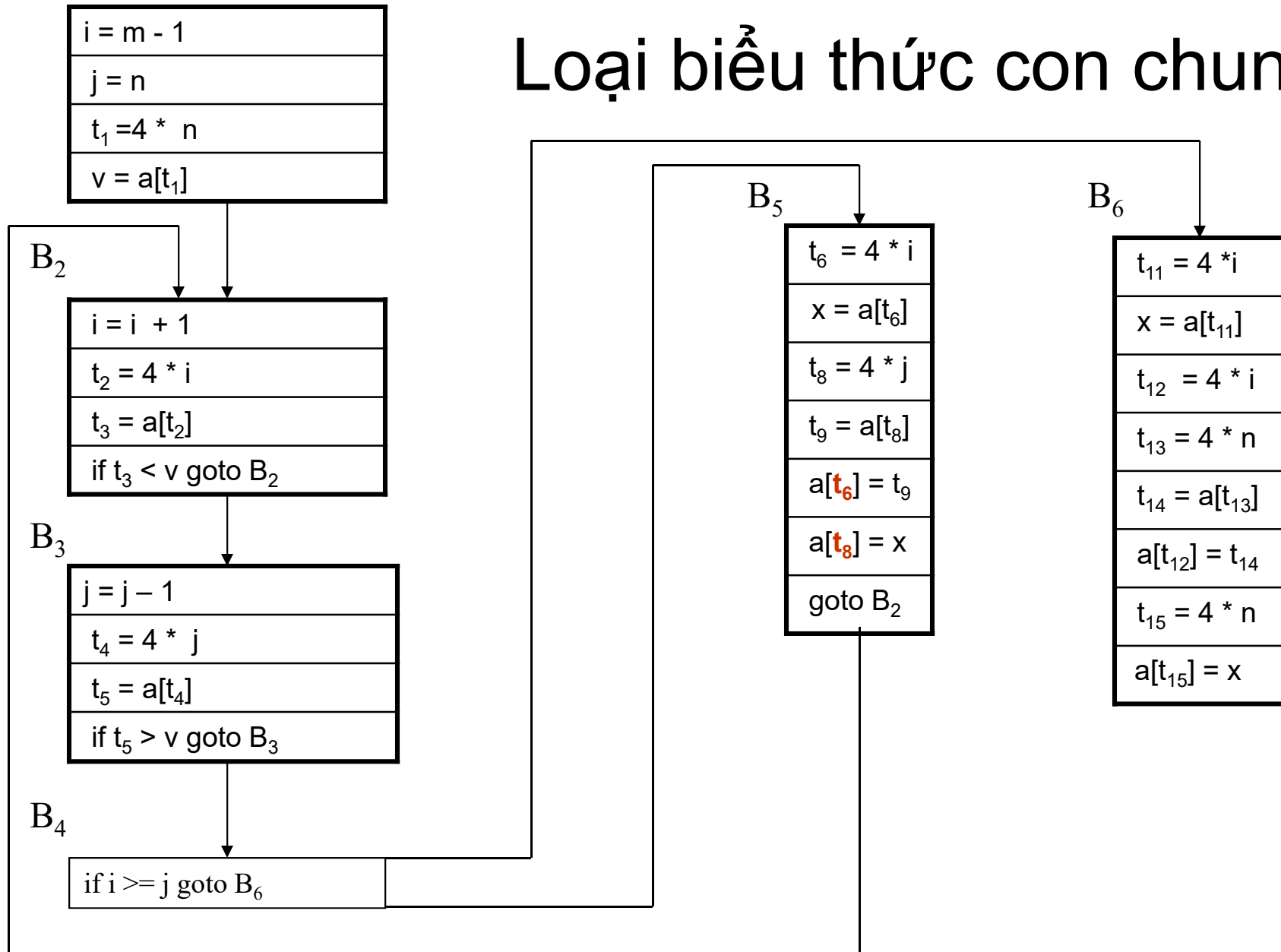
Loại biểu thức con chung

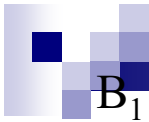




B₁

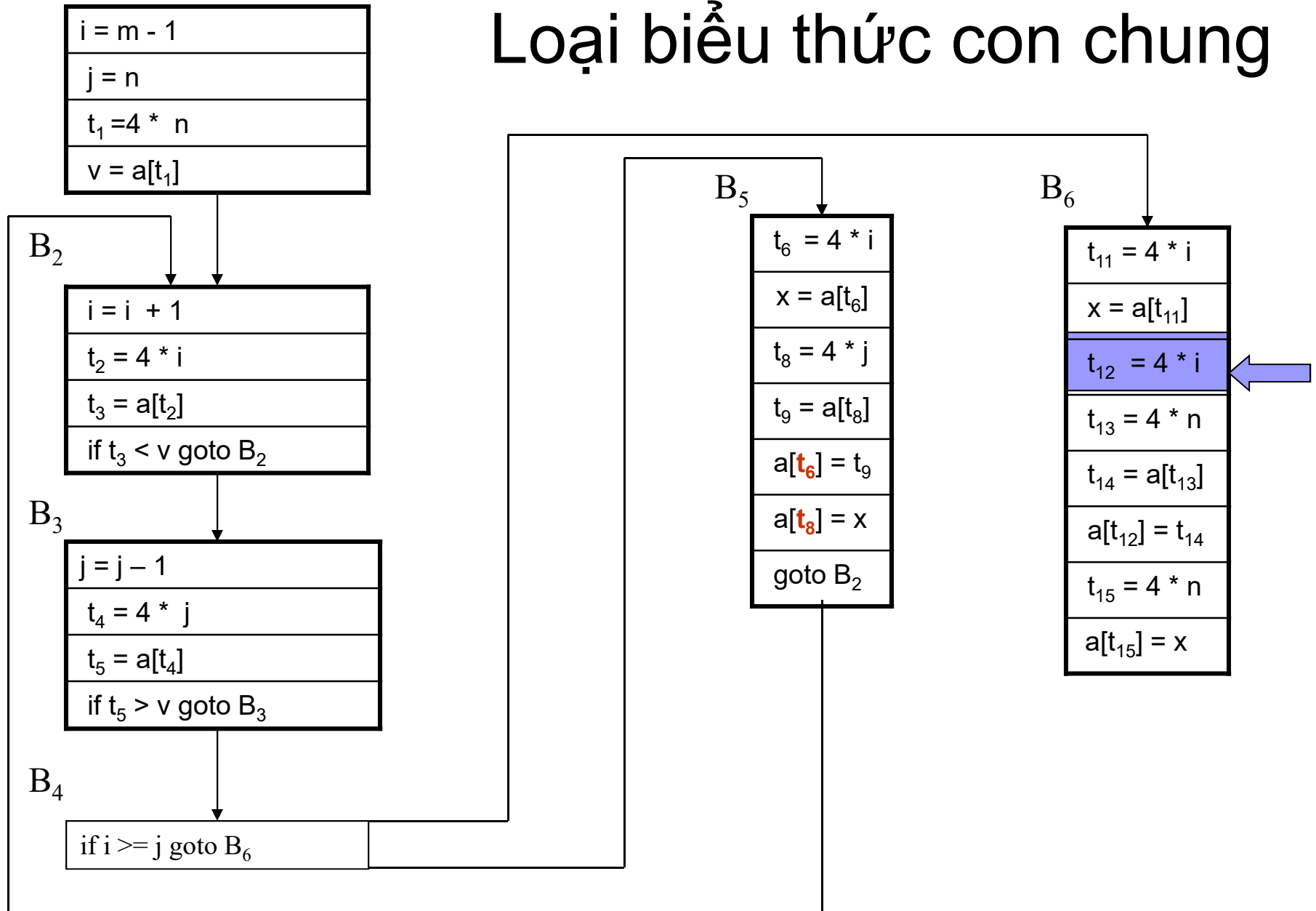
Loại biểu thức con chung





B₁

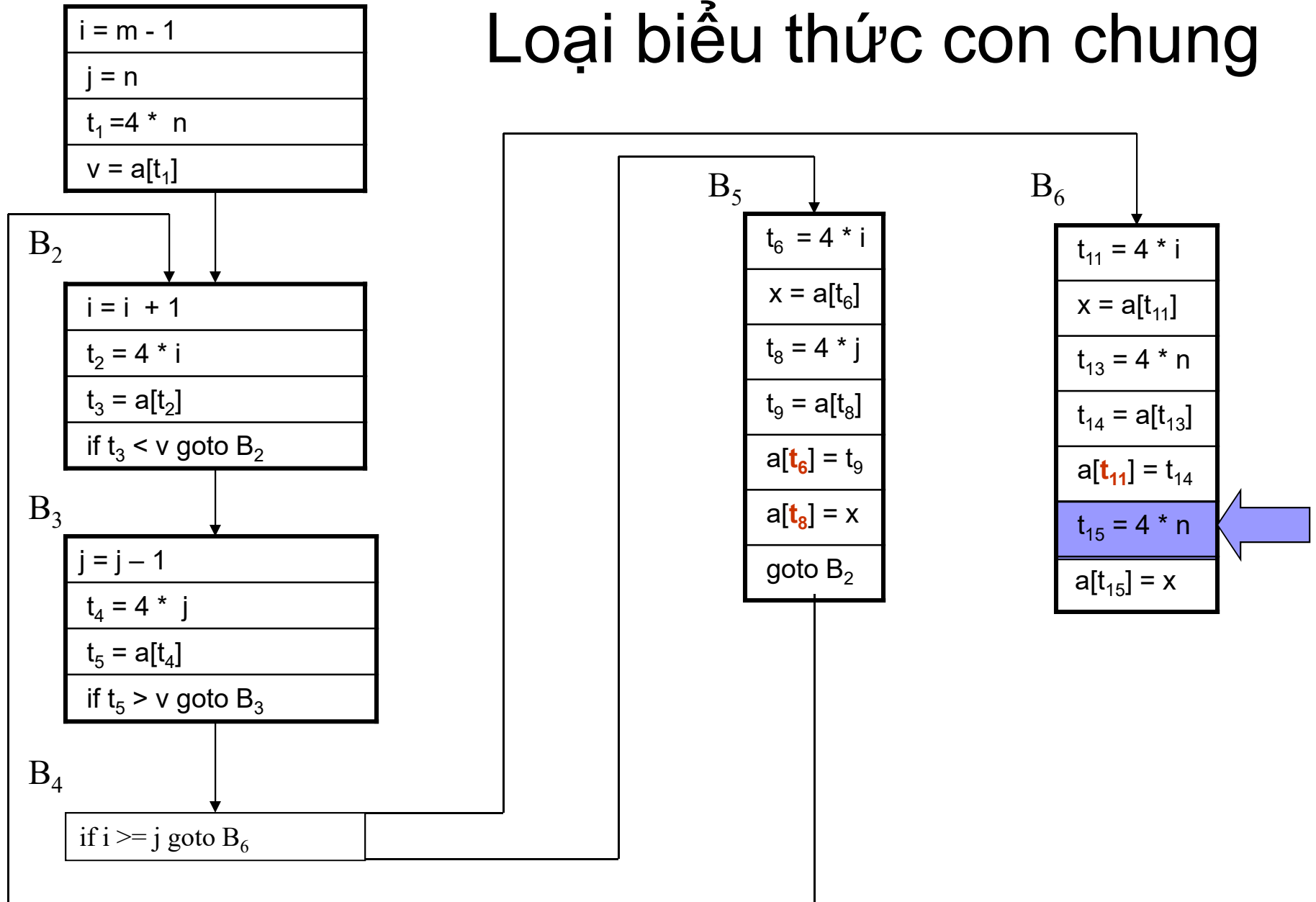
Loại biểu thức con chung

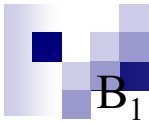




B₁

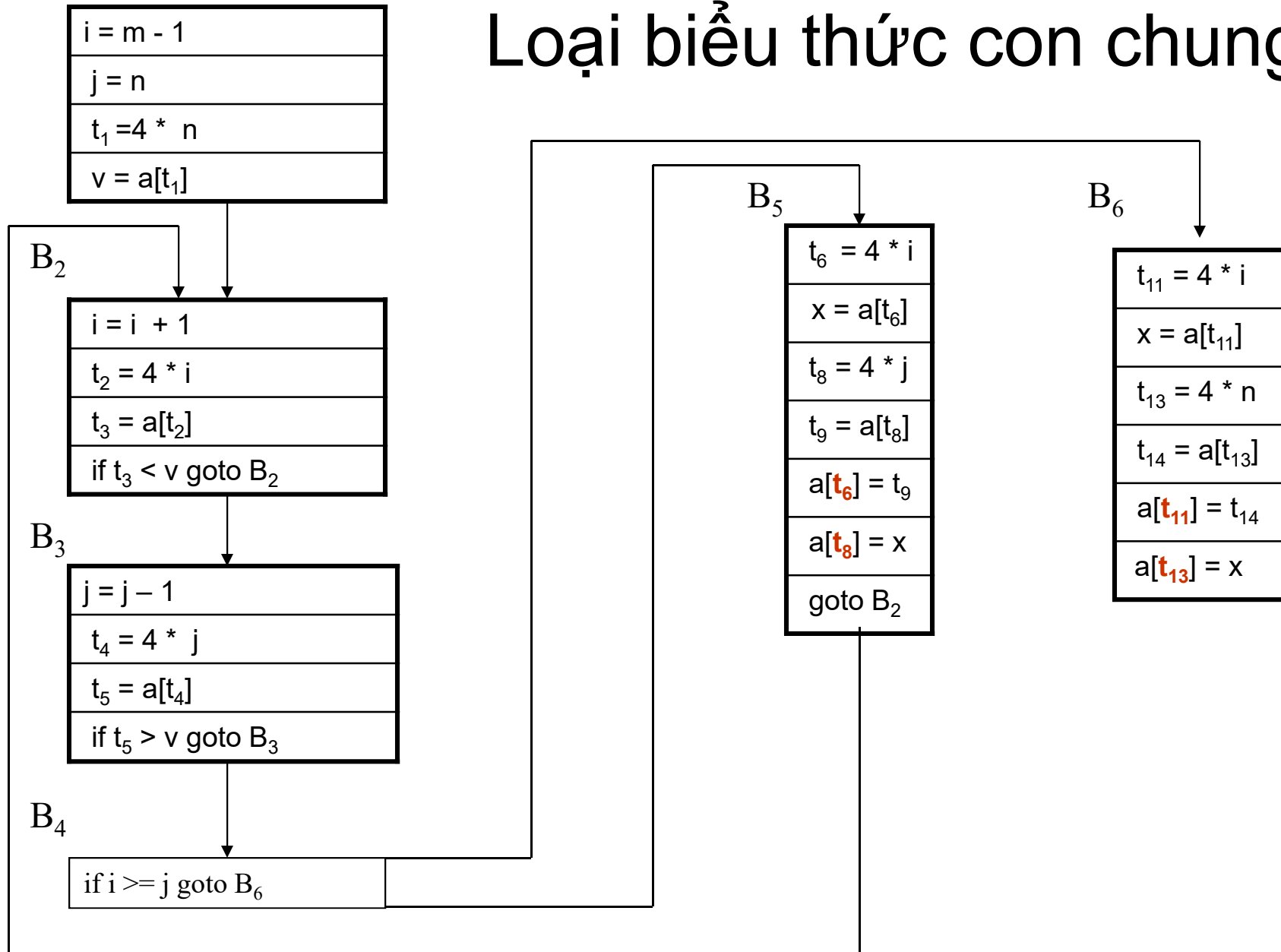
Loại biểu thức con chung





B₁

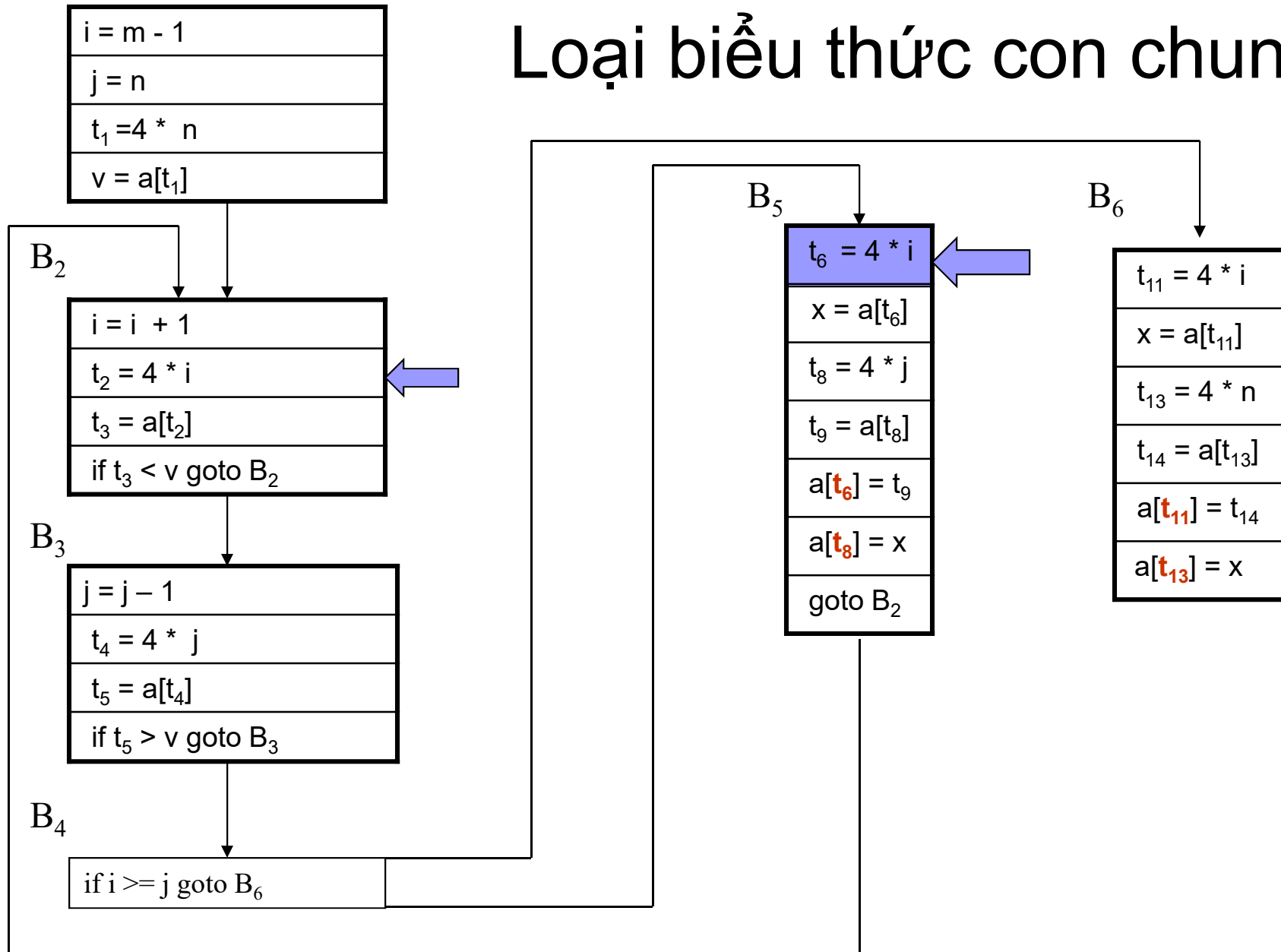
Loại biểu thức con chung

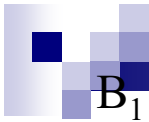




B₁

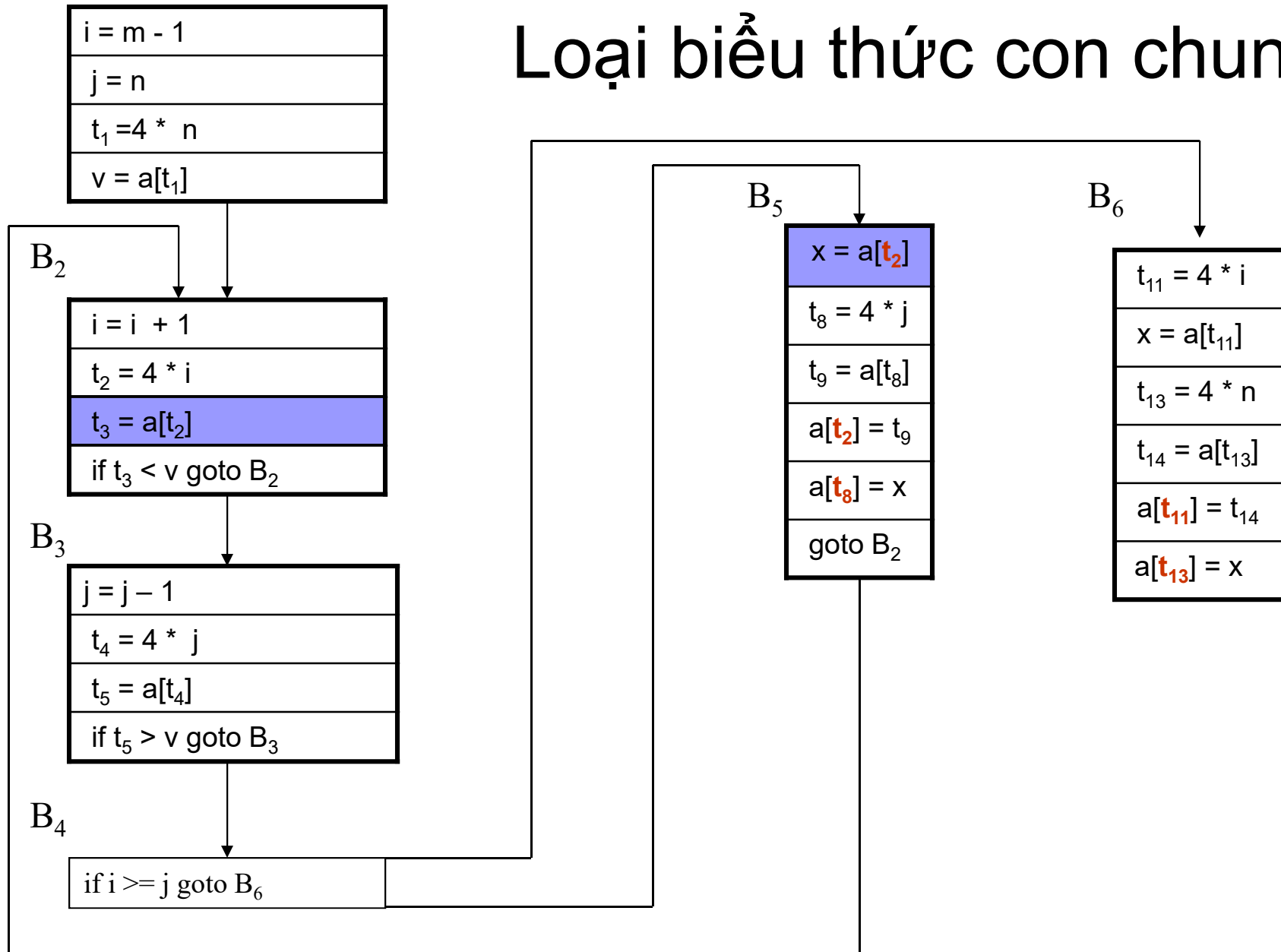
Loại biểu thức con chung





B₁

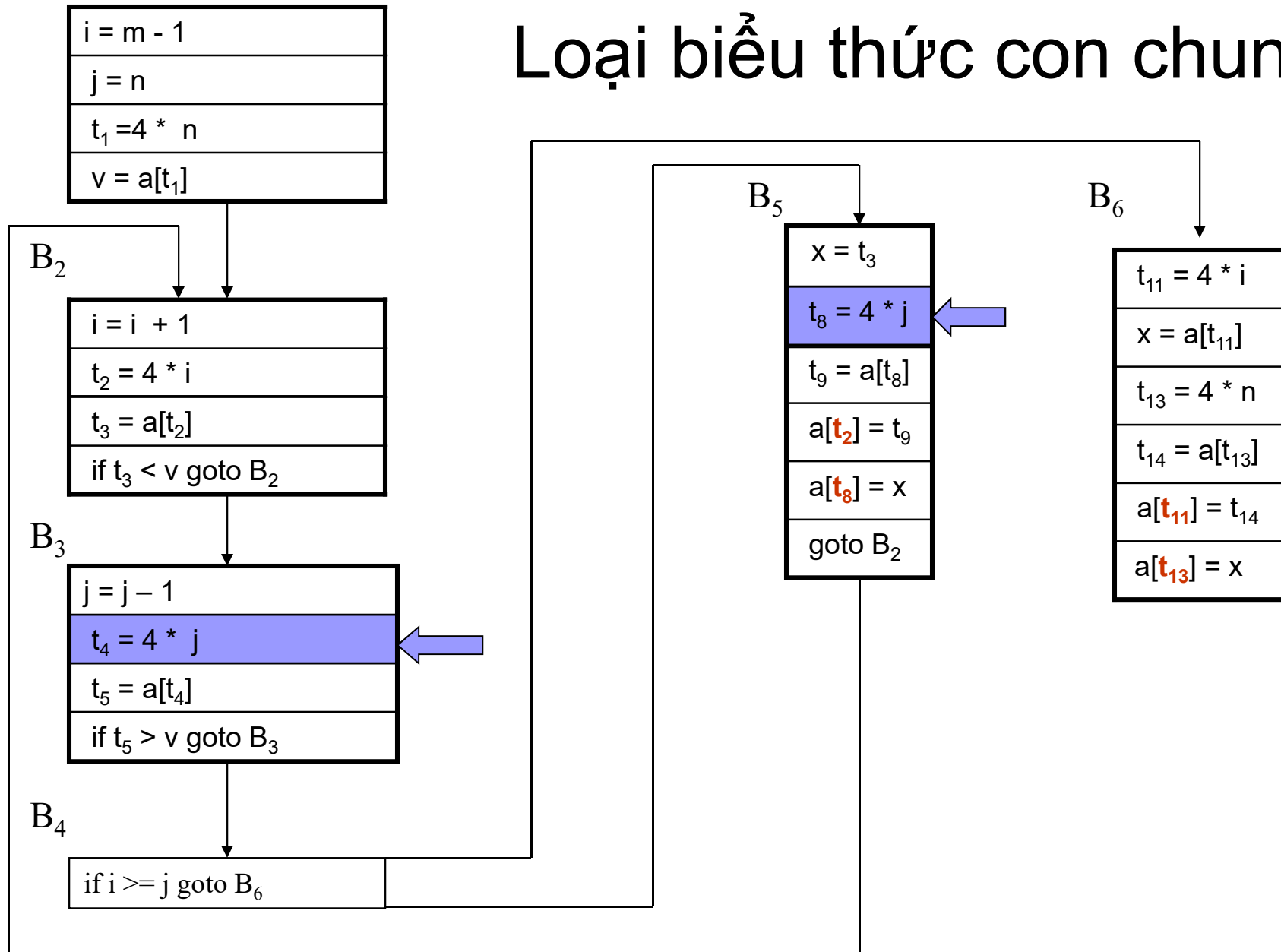
Loại biểu thức con chung





B₁

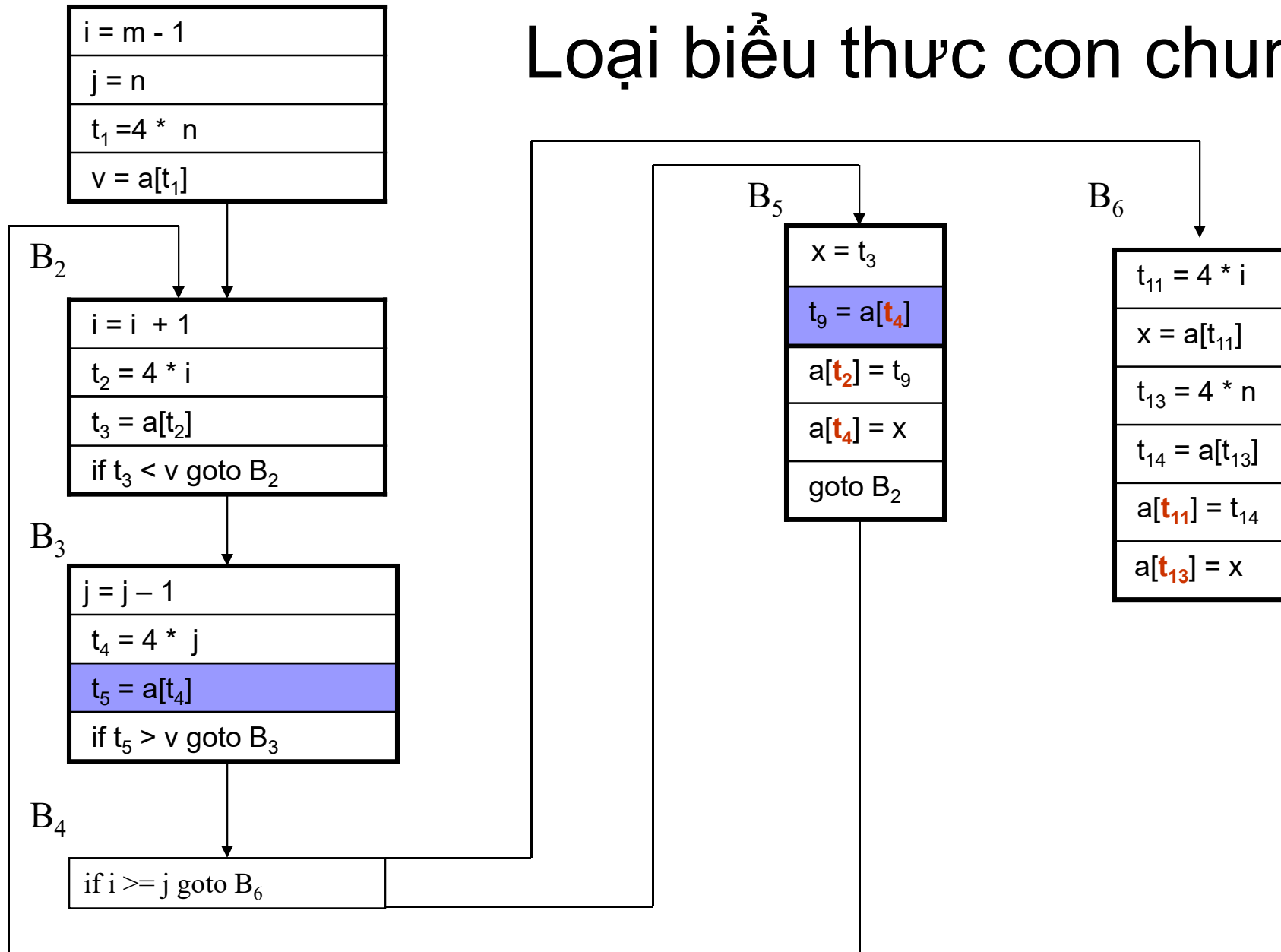
Loại biểu thức con chung





B₁

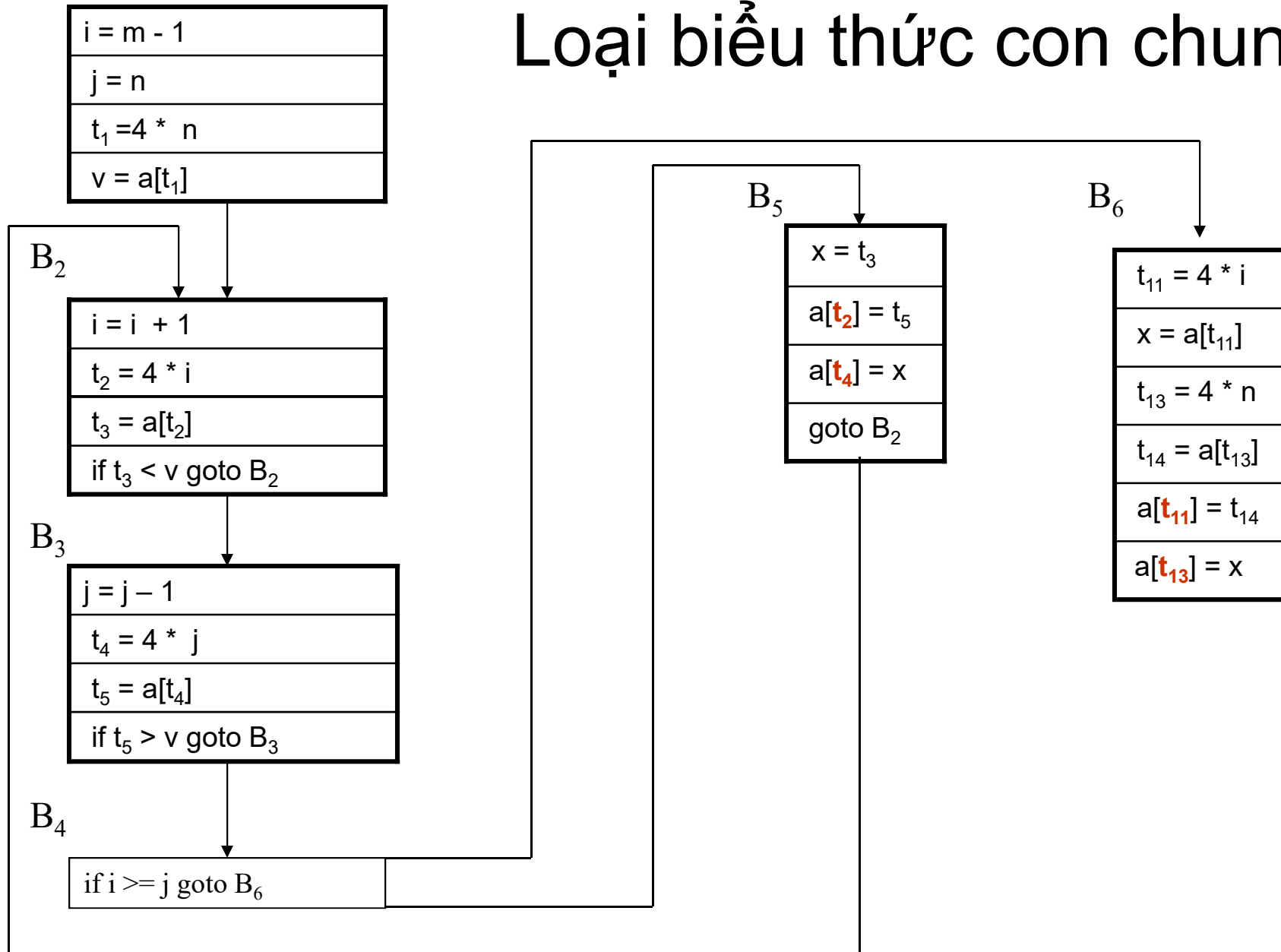
Loại biểu thức con chung





B₁

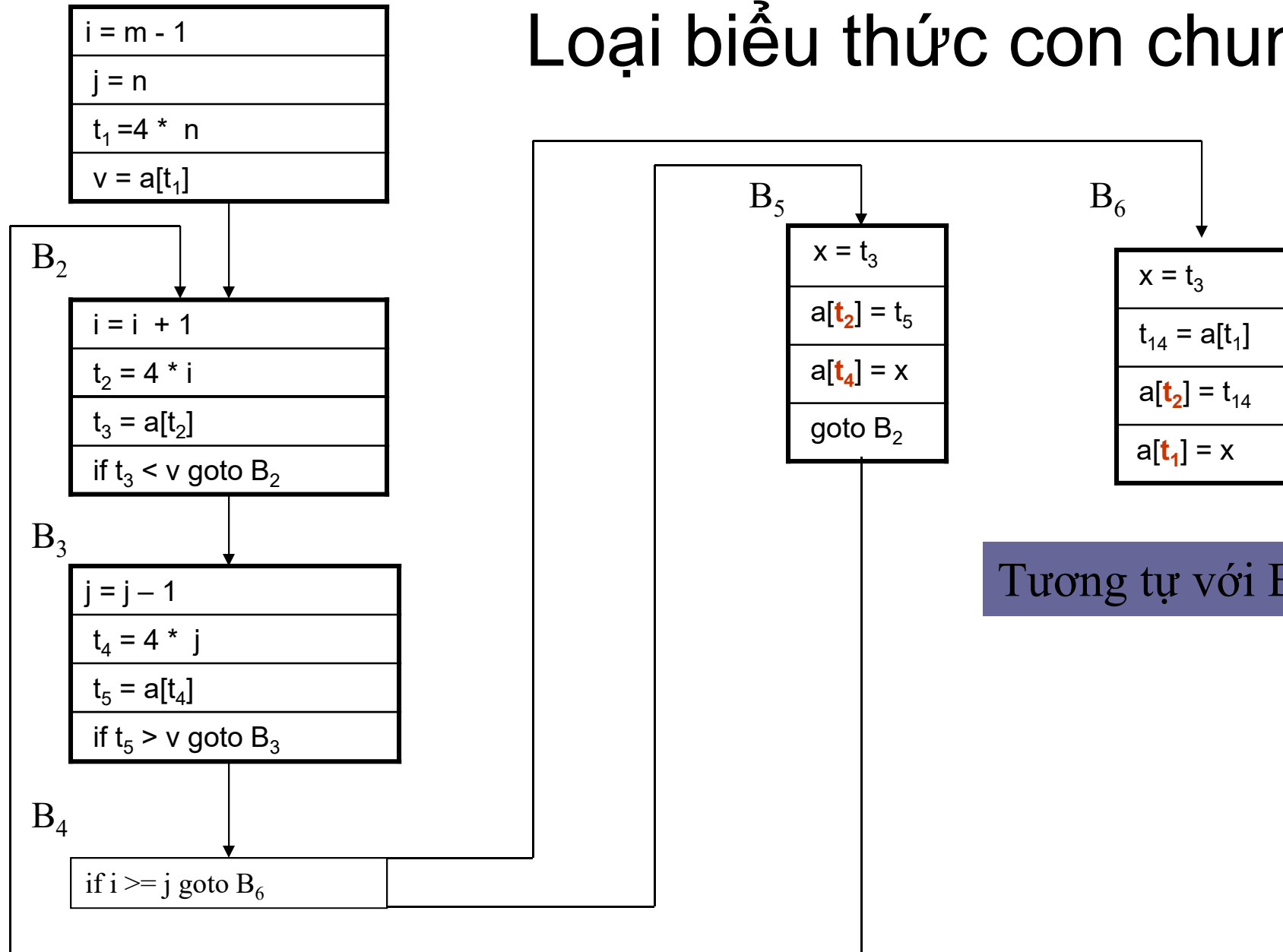
Loại biểu thức con chung

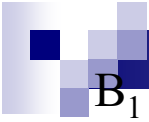




B₁

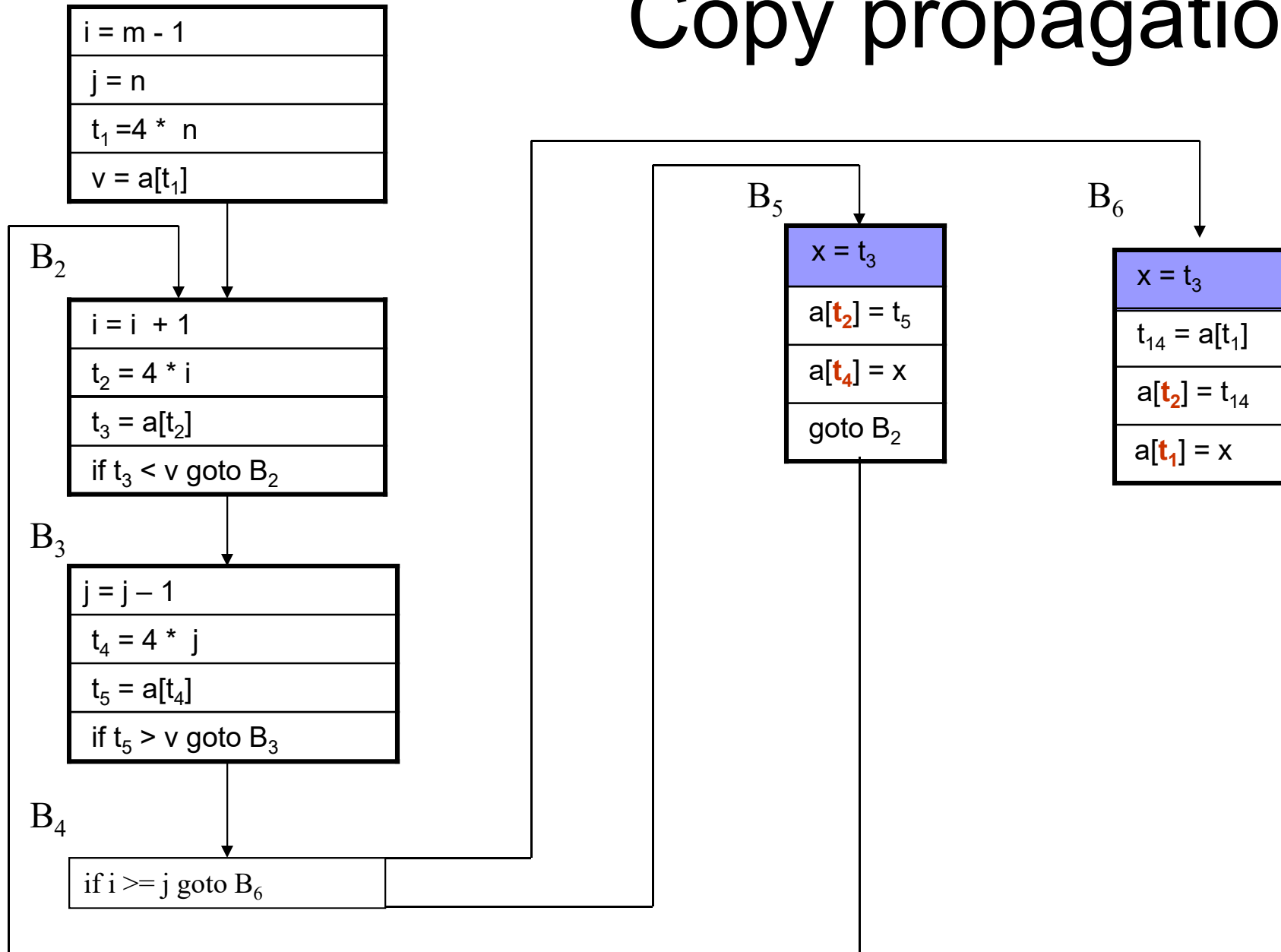
Loại biểu thức con chung



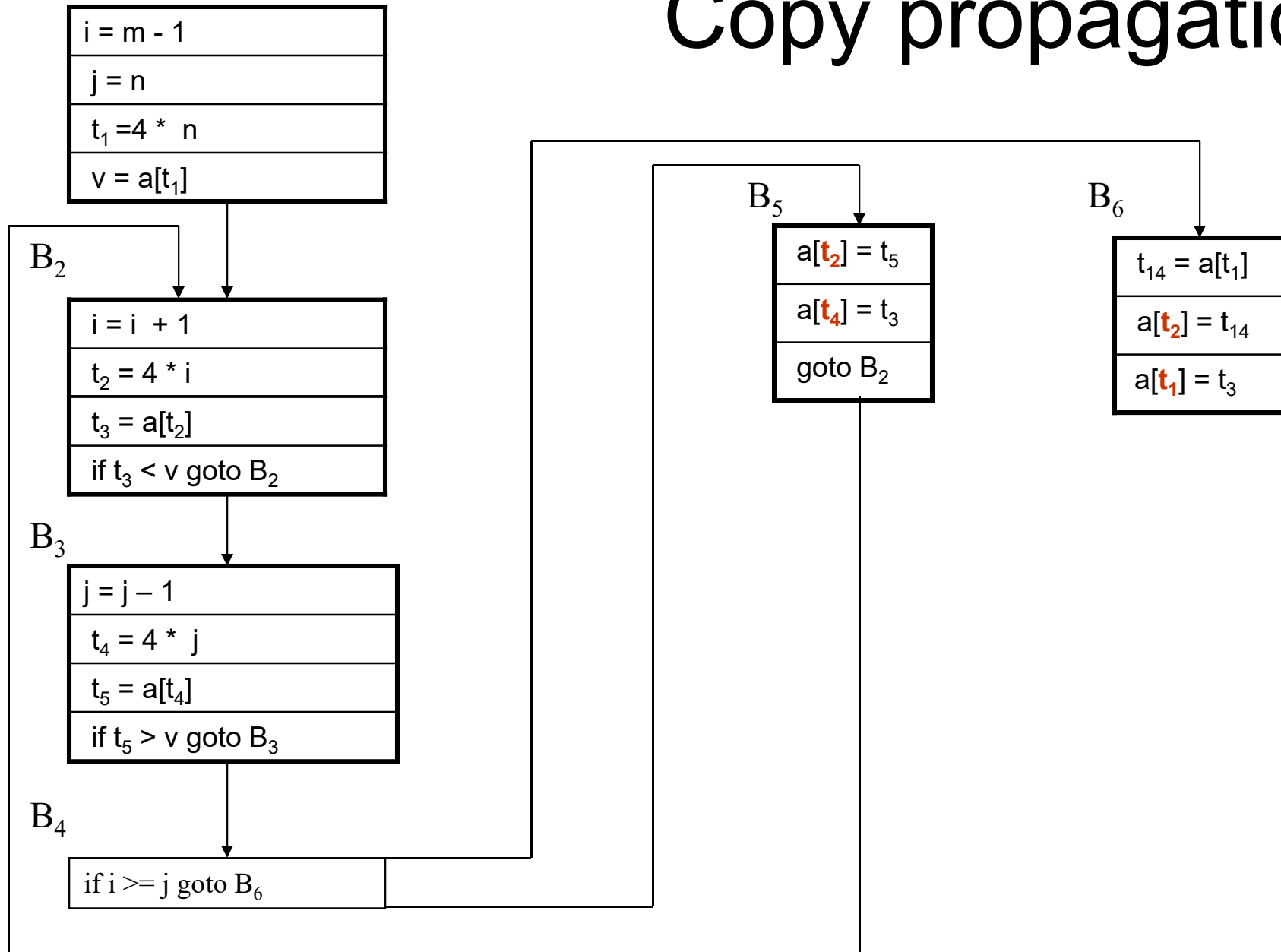


B₁

Copy propagation



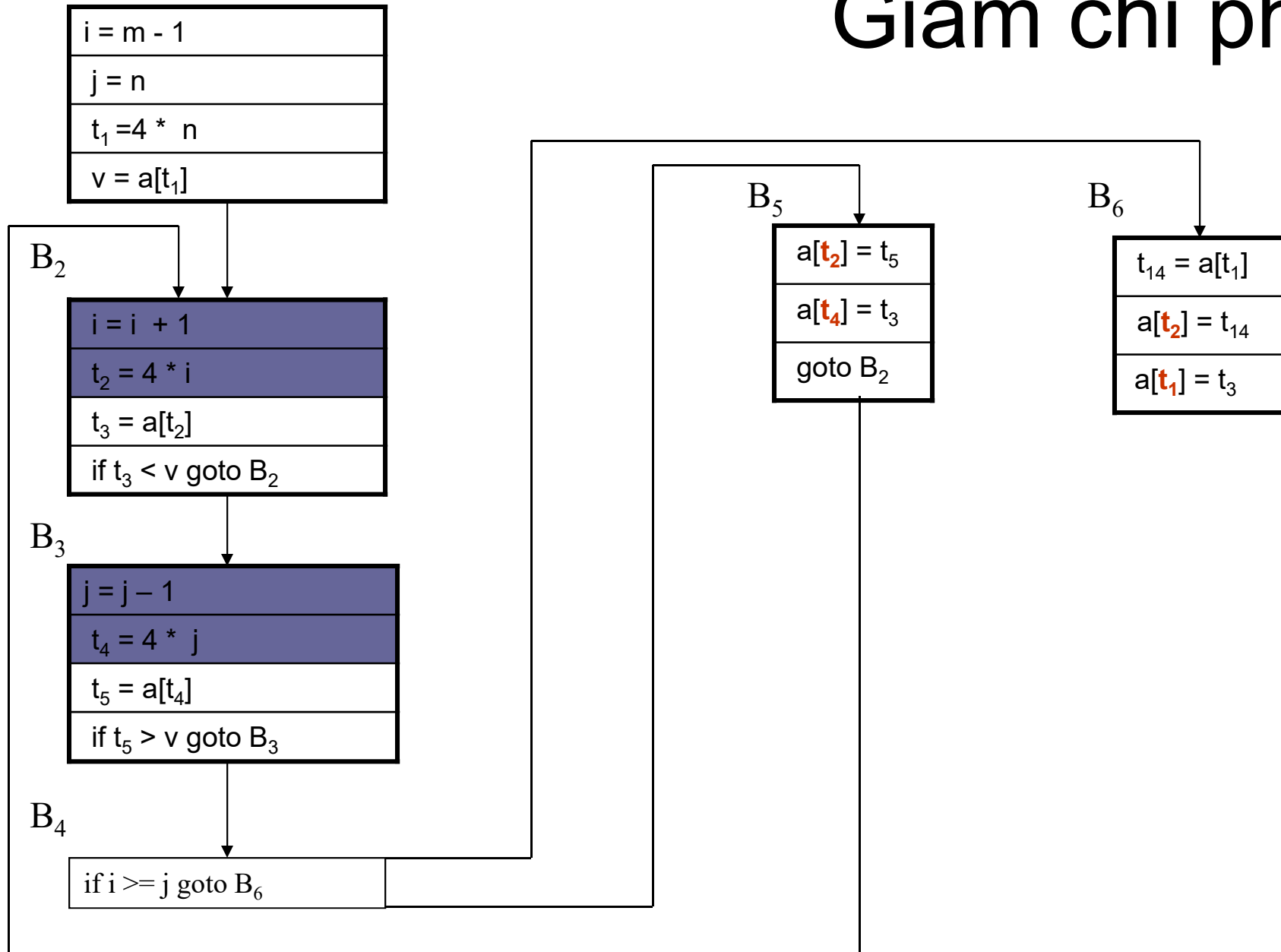
Copy propagation





B₁

Giảm chi phí



Giảm chi phí

