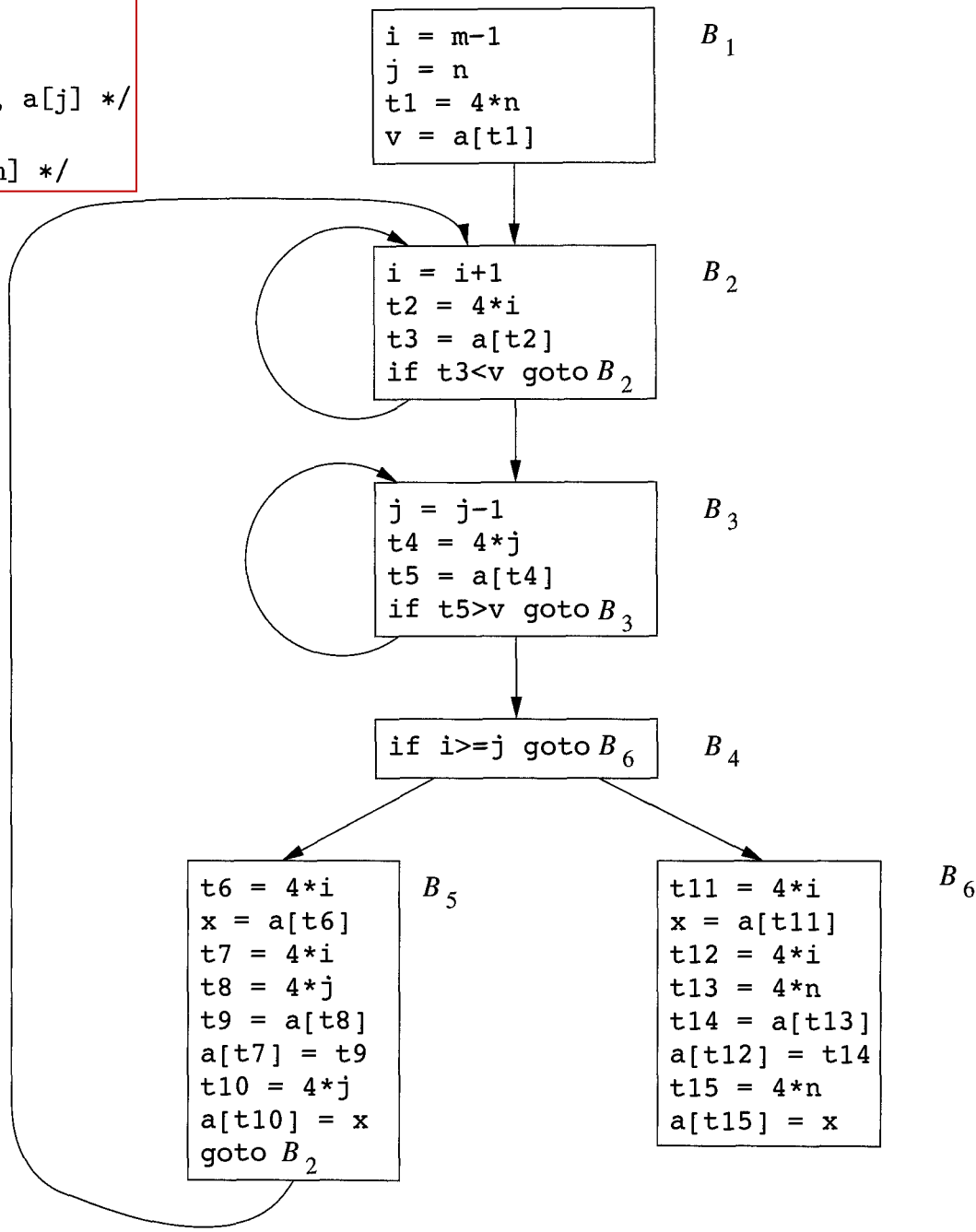
Compilers

Topic: Introduction to Machine Independent Code Optimizations

Monsoon 2010, IIIT-H, Suresh Purini

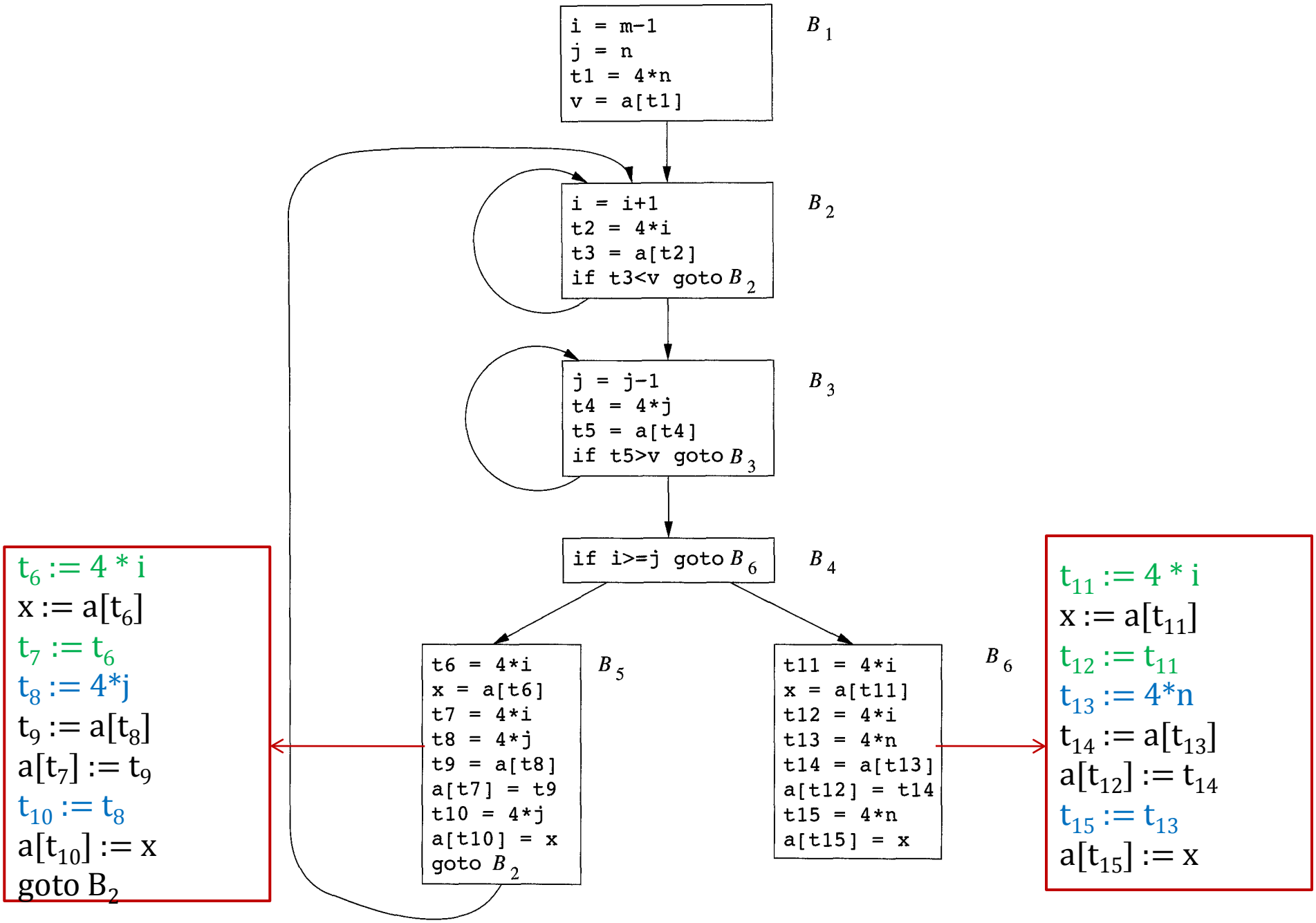
```
void quicksort(int m, int n) {          /* recursively sorts a[m] through a[n] */
    int i, j, v, x;
    if (n <= m) return;
            /* fragment begins here */
    i = m-1; j = n; v = a[n];
    while(1) {
        do i = i + 1; while (a[i] < v);
        do j = j - 1; while (a[j] > v);
      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

```
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; /* swap a[i], a[j] */
}
x = a[i]; a[i] = a[n]; a[n] = x; /* swap a[i], a[n] */
```

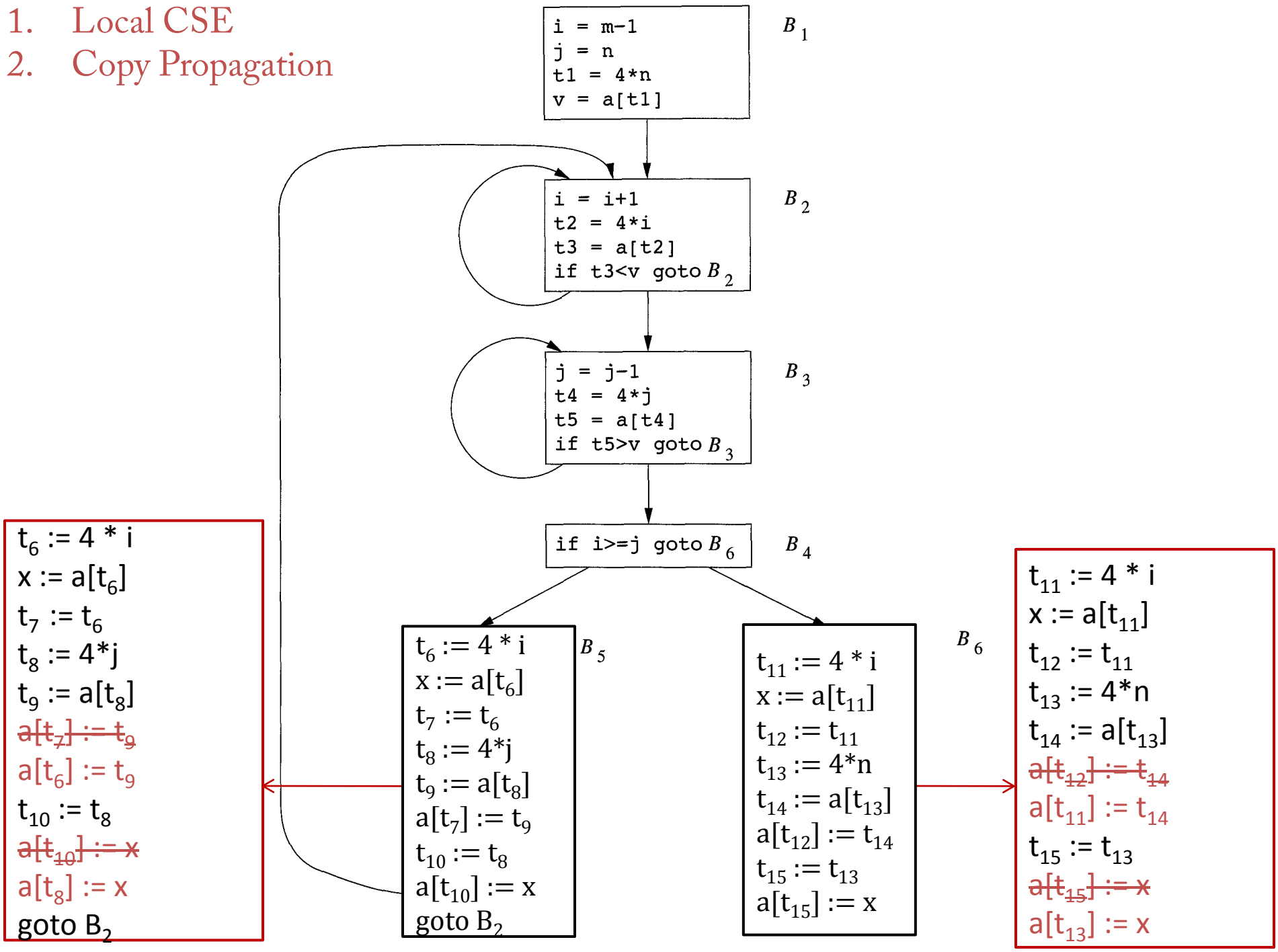


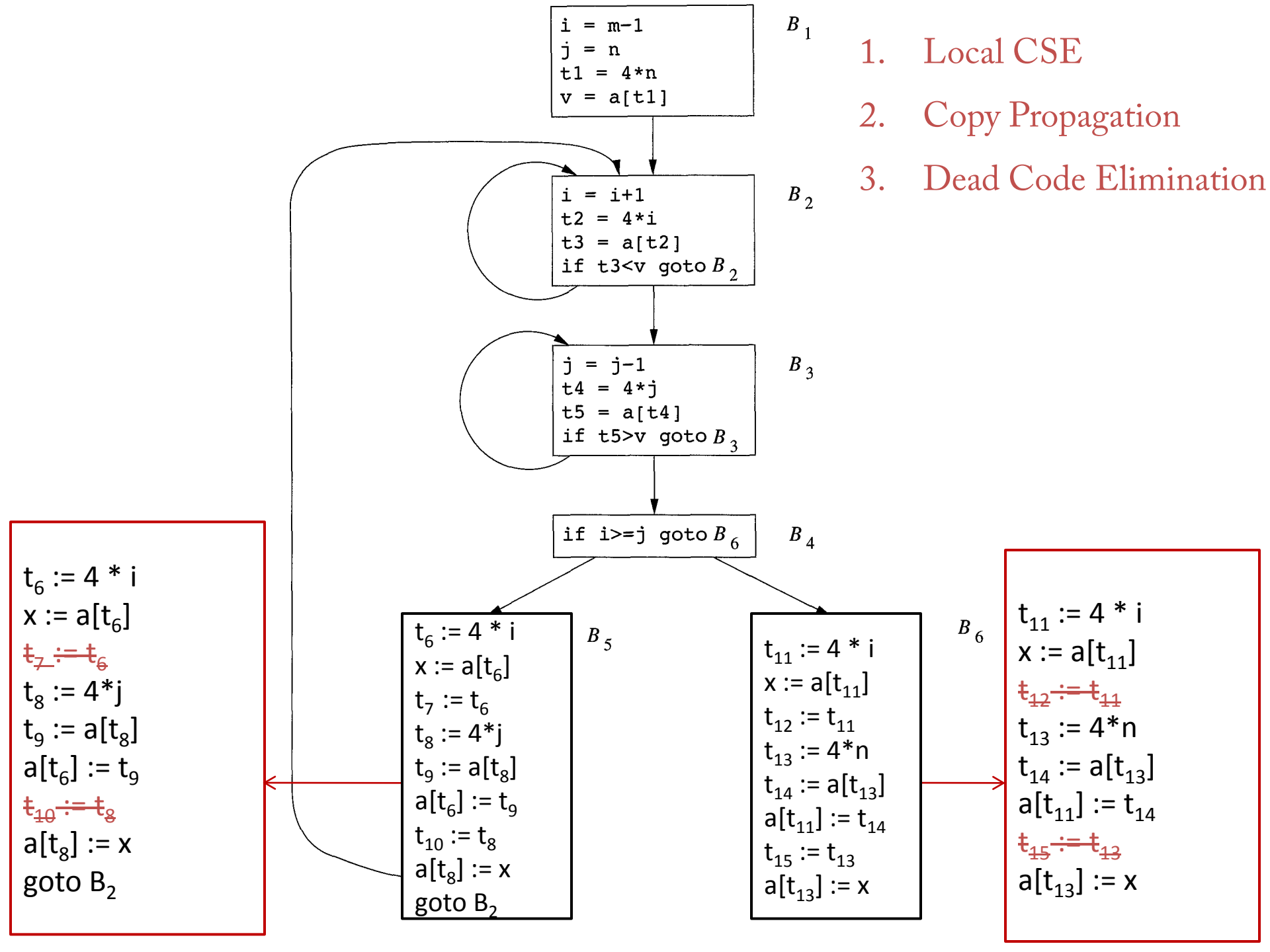
Note: We can safely assume that a CFG has a single special Entry Block and single Special Exit Block.

Local Common Sub-Expression Elimination

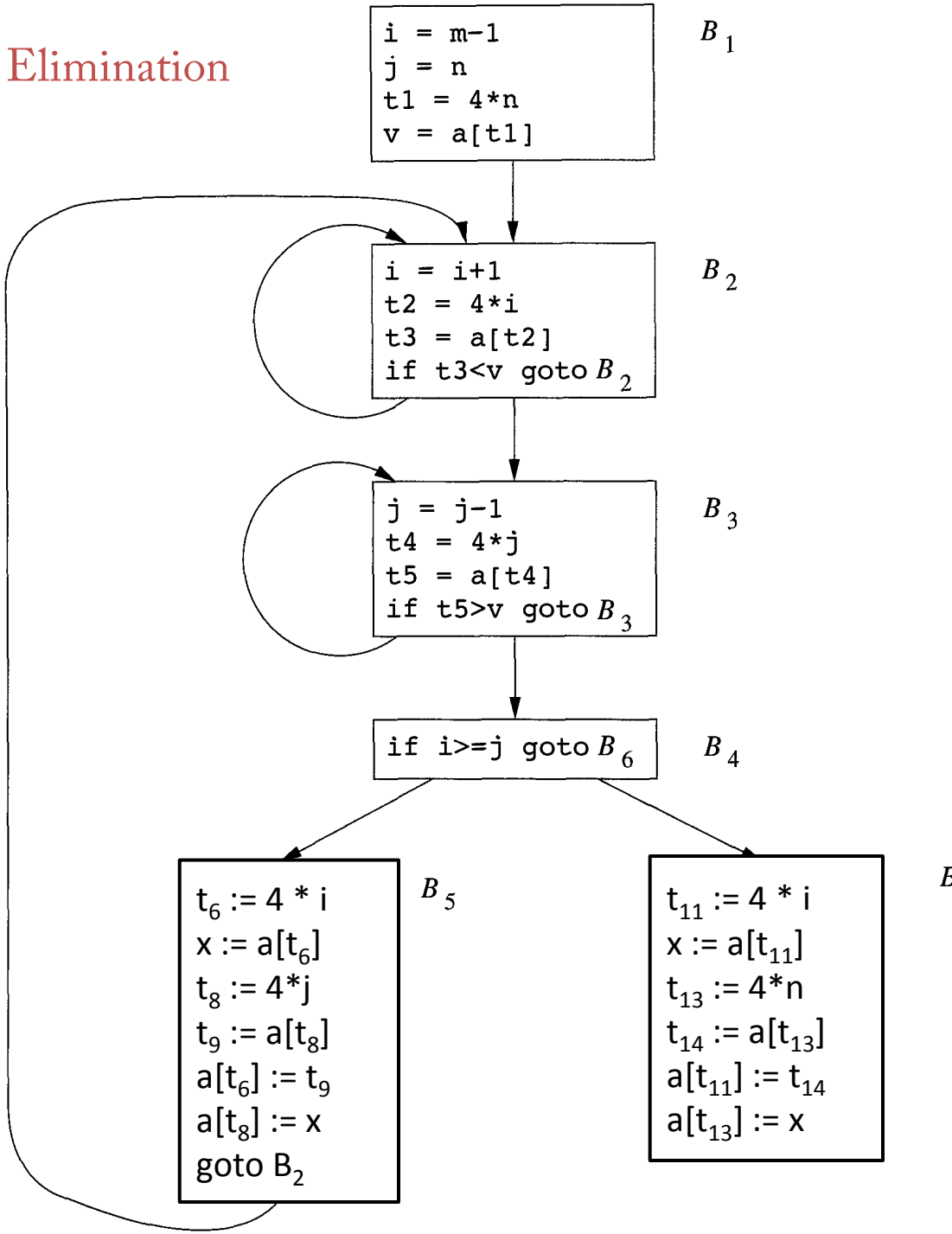


1. Local CSE
2. Copy Propagation



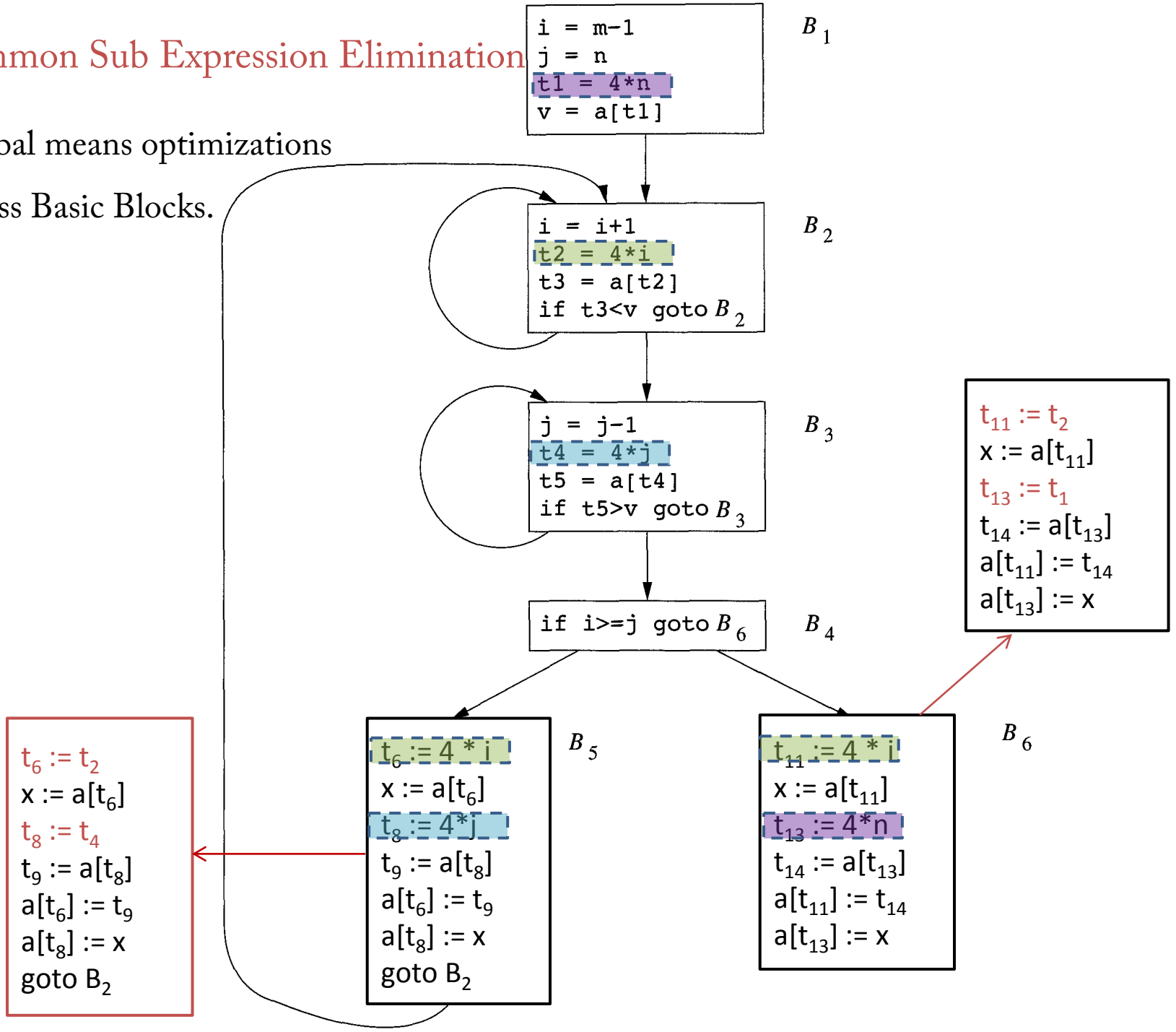

i = m-1
j = n
t1 = 4*n
v = a[t1]
B1
i = i+1
t2 = 4*i
t3 = a[t2]
if t3<v goto B2
B2
j = j-1
t4 = 4*j
t5 = a[t4]
if t5>v goto B3
B3
if i>=j goto B6
B4
t6 := 4 * i
x := a[t6]
t7 := t6
t8 := 4*j
t9 := a[t8]
a[t6] := t9
t10 := t8
a[t8] := x
goto B2
t6 := 4 * i
x := a[t6]
t7 := t6
t8 := 4*j
t9 := a[t8]
a[t6] := t9
t10 := t8
a[t8] := x
goto B2
B5
t11 := 4 * i
x := a[t11]
t12 := t11
t13 := 4*n
t14 := a[t13]
a[t11] := t14
t15 := t13
a[t13] := x
B6
t11 := 4 * i
x := a[t11]
t12 := t11
t13 := 4*n
t14 := a[t13]
a[t11] := t14
t15 := t13
a[t13] := x
1. Local CSE
2. Copy Propagation
3. Dead Code Elimination

1. Local Common Sub-Expression Elimination
2. Copy Propagation
3. Dead Code Elimination



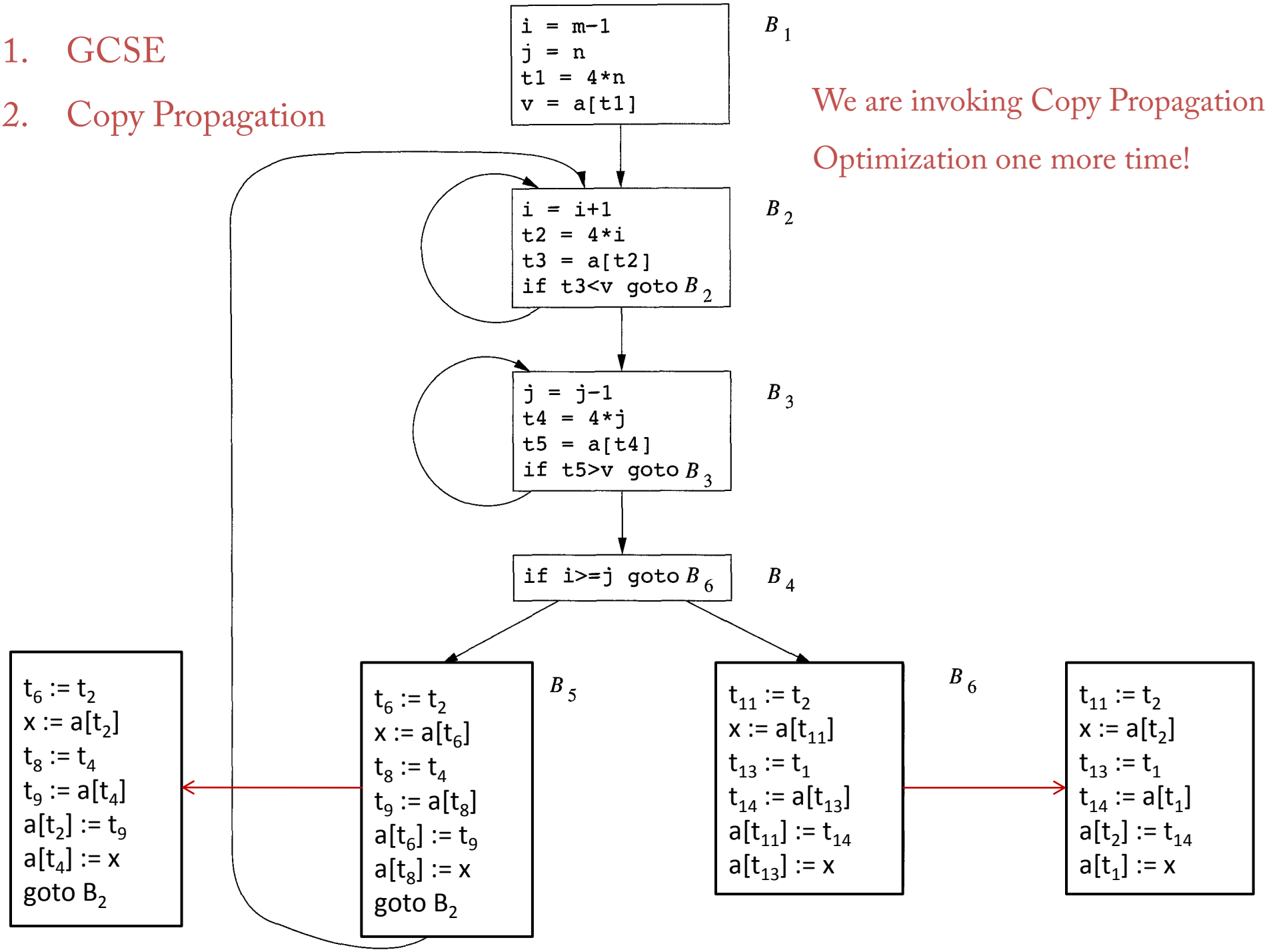
Global Common Sub Expression Elimination

Remark: Global means optimizations are done across Basic Blocks.



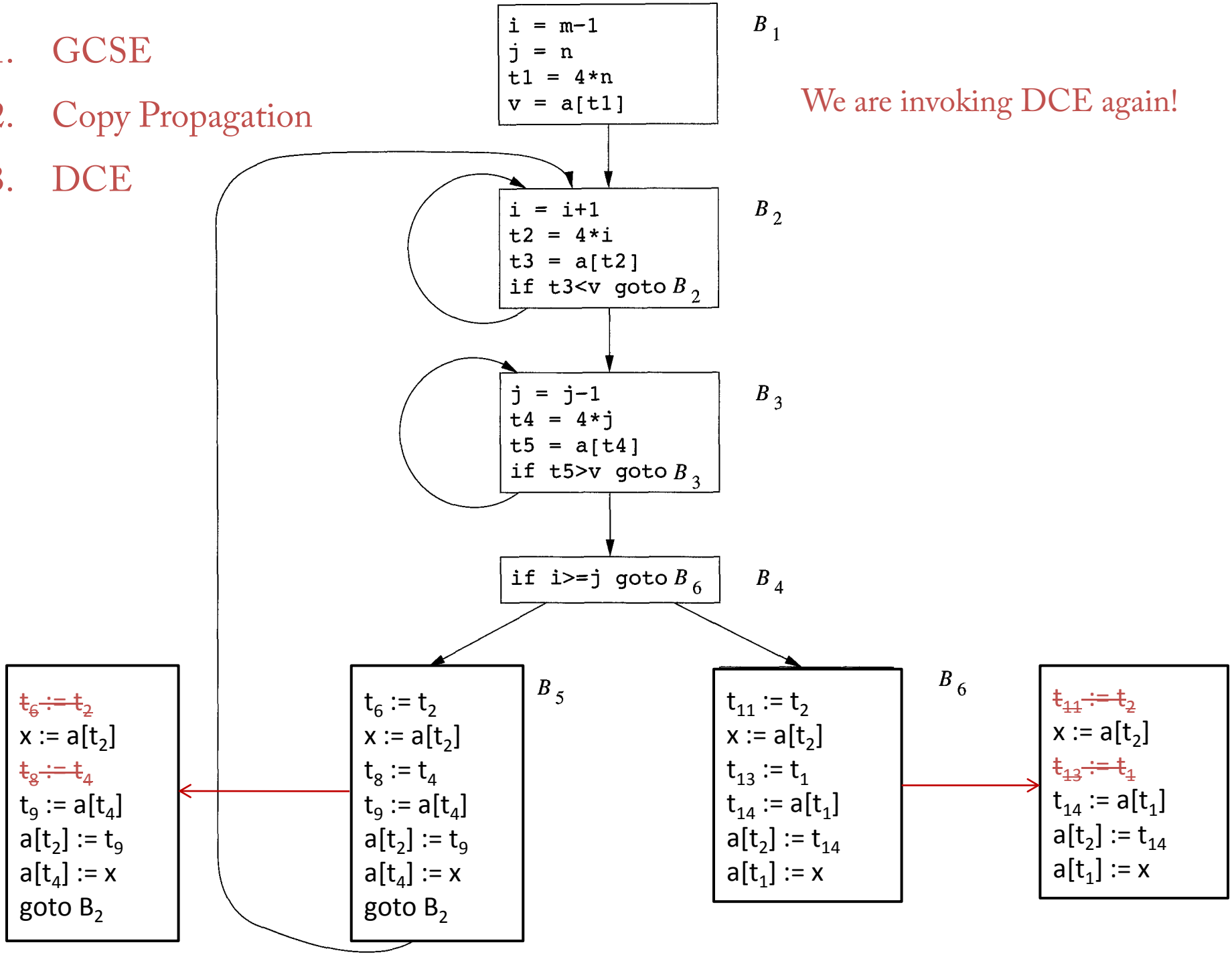
1. GCSE
2. Copy Propagation

We are invoking Copy Propagation Optimization one more time!

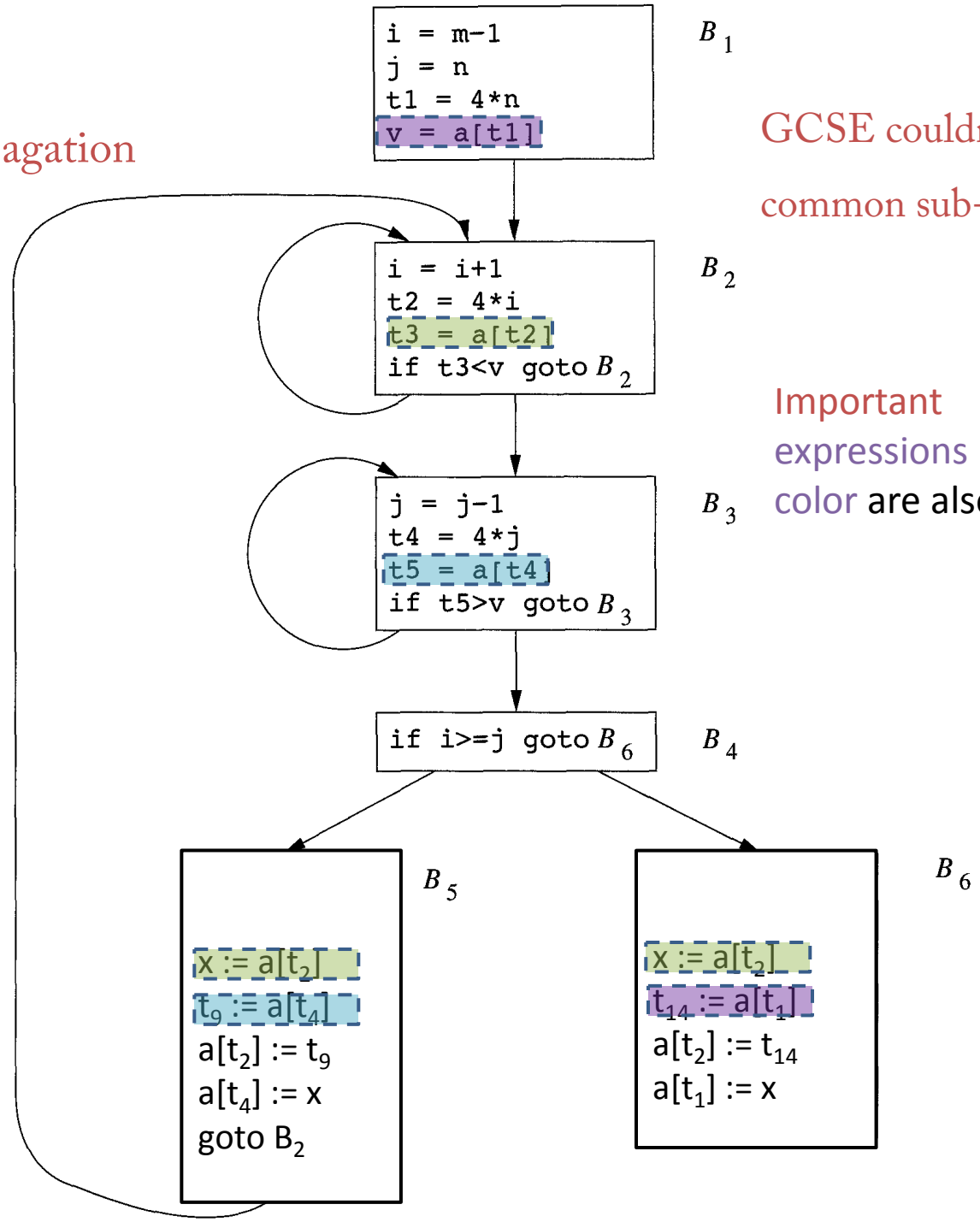


1. GCSE
2. Copy Propagation
3. DCE

We are invoking DCE again!



1. GCSE
2. Copy Propagation
3. DCE
4. GCSE



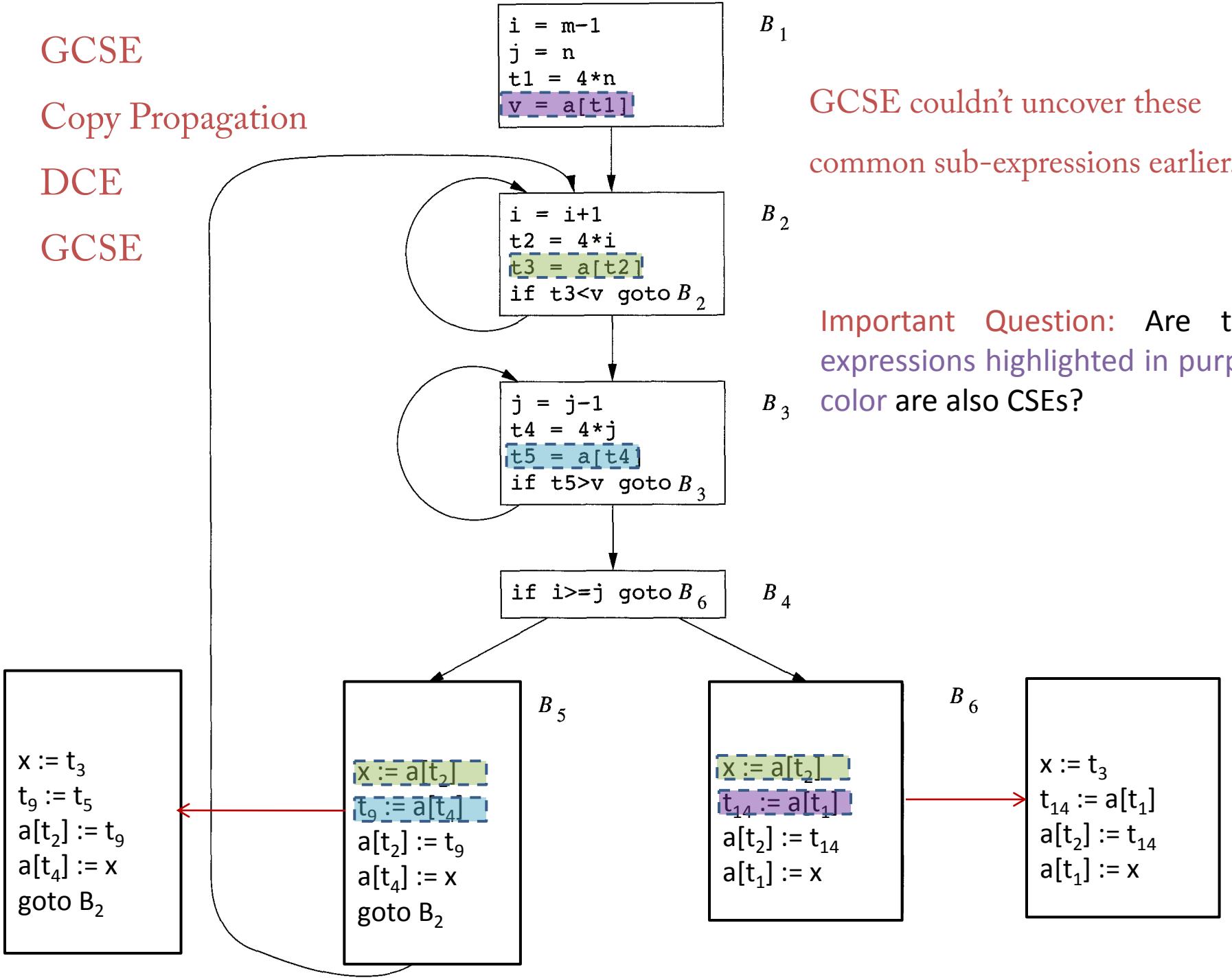
GCSE couldn't uncover these common sub-expressions earlier.

Important Question: Are the expressions highlighted in purple color are also CSEs?

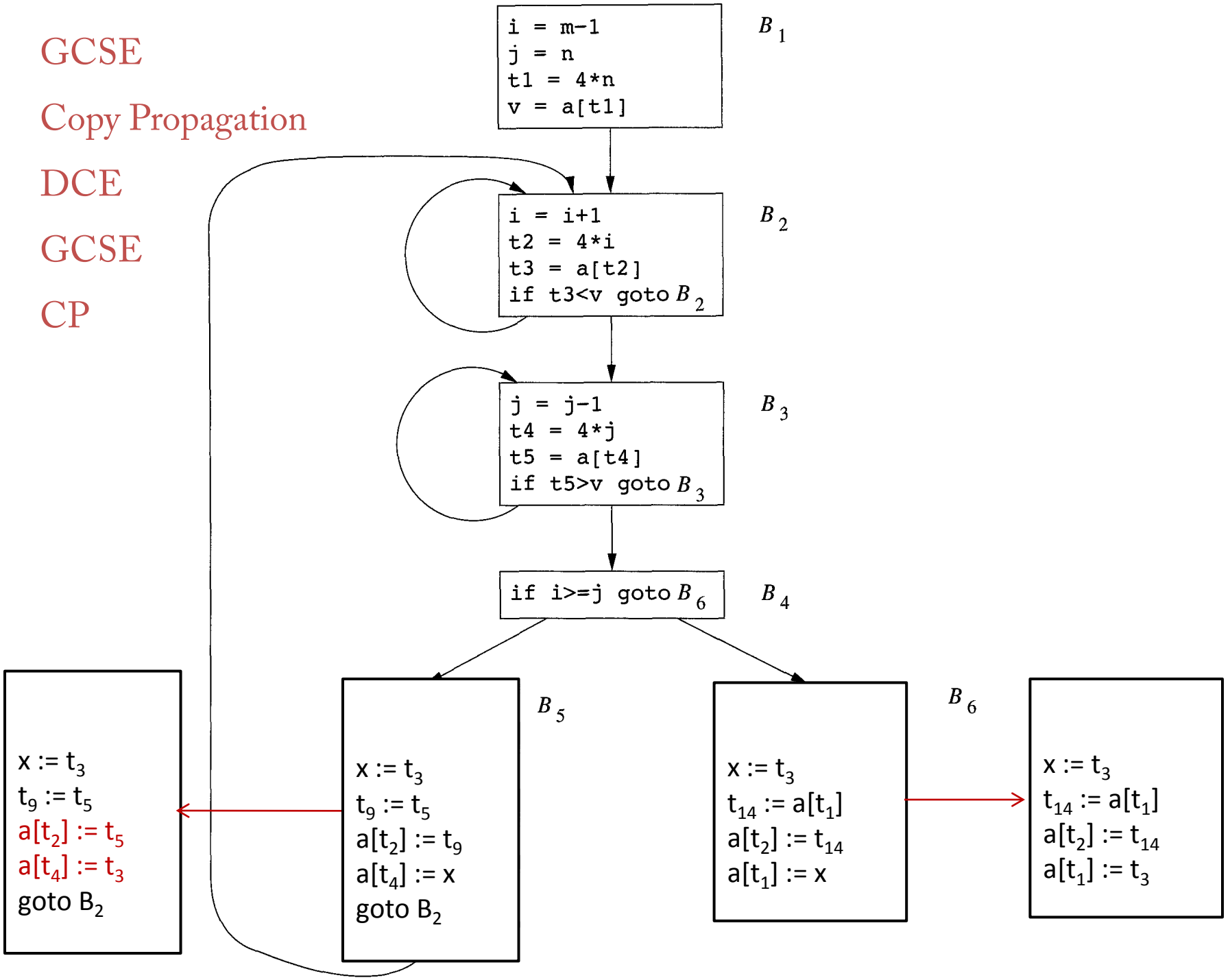
1. GCSE
2. Copy Propagation
3. DCE
4. GCSE

GCSE couldn't uncover these common sub-expressions earlier.

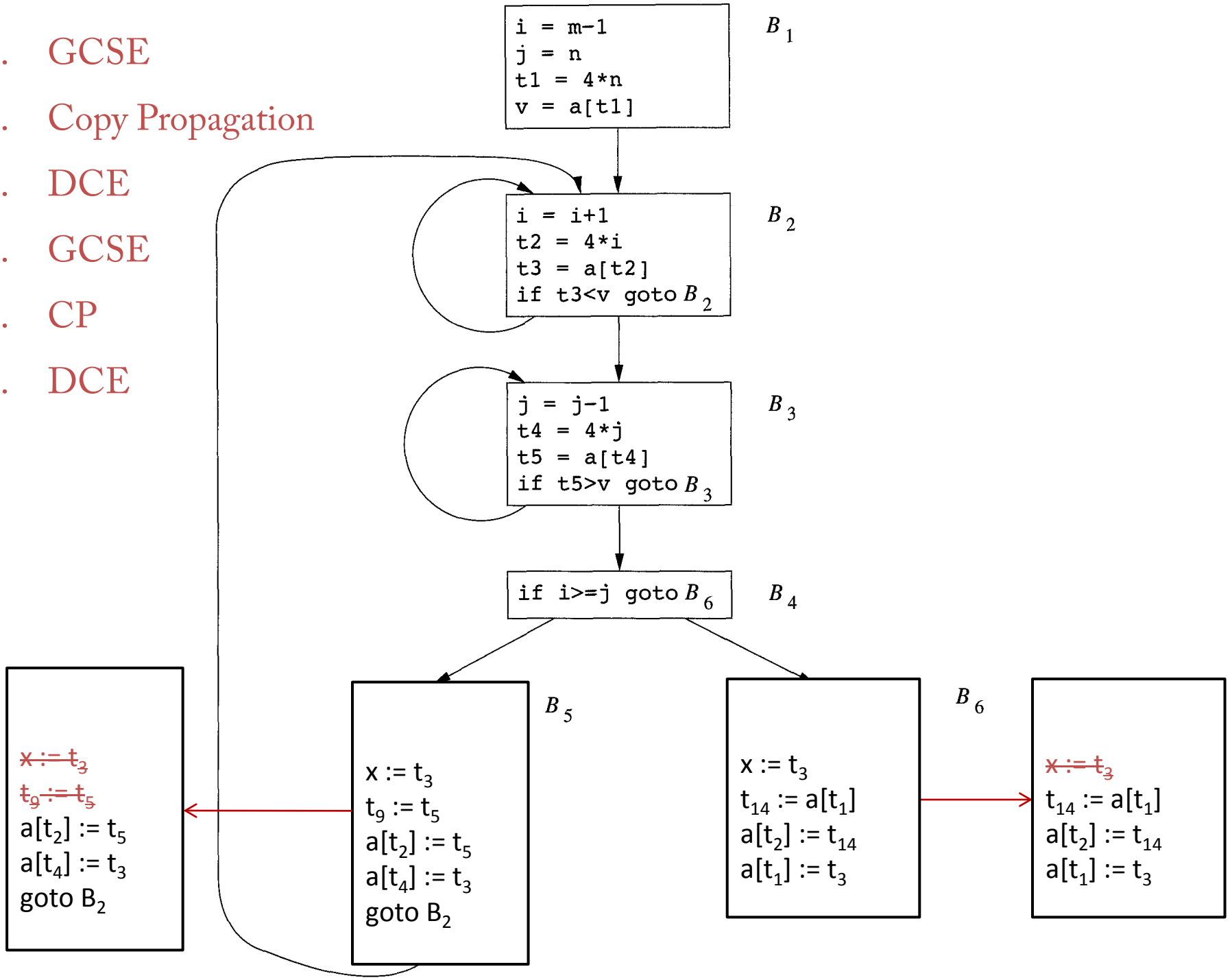
Important Question: Are the expressions highlighted in purple color are also CSEs?



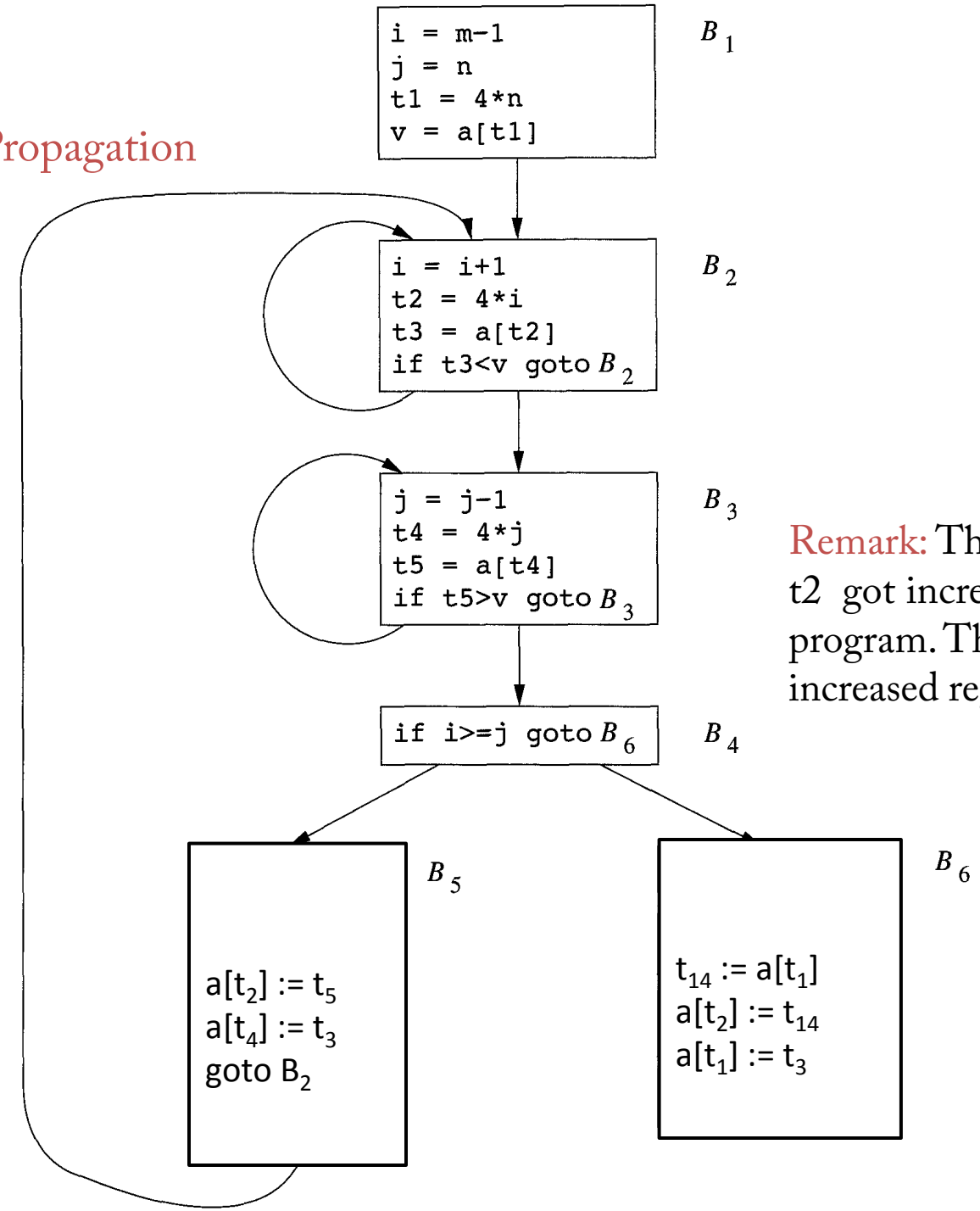
1. GCSE
2. Copy Propagation
3. DCE
4. GCSE
5. CP



1. GCSE
2. Copy Propagation
3. DCE
4. GCSE
5. CP
6. DCE



1. GCSE
2. Copy Propagation
3. DCE
4. GCSE
5. CP
6. DCE



Remark: The Live Range of variable t2 got increased in the optimized program. That could translate to increased register pressure.