C Code für Quicksort

```
void quicksort(int m, int n)
  {   int i,j; int v,x;
      if (n <= m) return;
      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;
      quicksort(m,j); quicksort(i+1,n);
  }
```

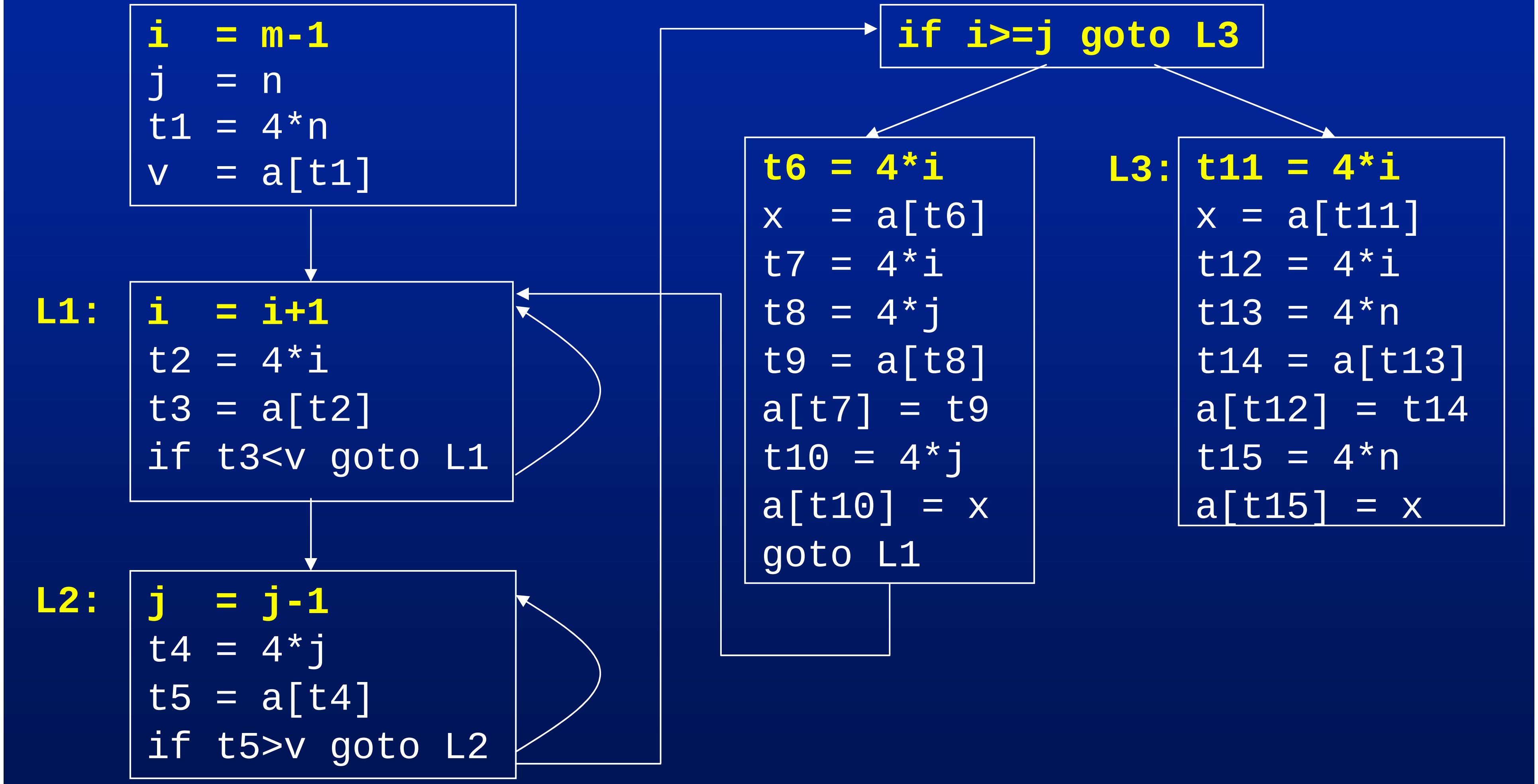
Drei-Adress-Code für markiertes Fragment

```
        i  = m-1
        j  = n
        t1 = 4*n
        v  = a[t1]
L1:     i  = i+1
        t2 = 4*i
        t3 = a[t2]
        if t3<v goto L1
L2:     j  = j-1
        t4 = 4*j
        t5 = a[t4]
        if t5>v goto L2
        if i>=j goto L3
        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 L1
L3:     t11 = 4*i
        x = a[t11]
        t12 = 4*i
        t13 = 4*n
        t14 = a[t13]
        a[t12] = t14
        t15 = 4*n
        a[t15] = x
```

„Leader“ sind gelb markiert.

Basisblockdarstellung



```
i  = m-1
j  = n
t1 = 4*n
v  = a[t1]
```

```
L1: i  = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto L1
```

```
L2: j  = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto L2
```

```
if i>=j goto L3
```

```
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 L1
```

```
L3: t11 = 4*i
    x = a[t11]
    t12 = 4*i
    t13 = 4*n
    t14 = a[t13]
    a[t12] = t14
    t15 = 4*n
    a[t15] = x
```

Elimination lokaler gemeinsamer Teilausdrücke

```
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 L1
```

⇒

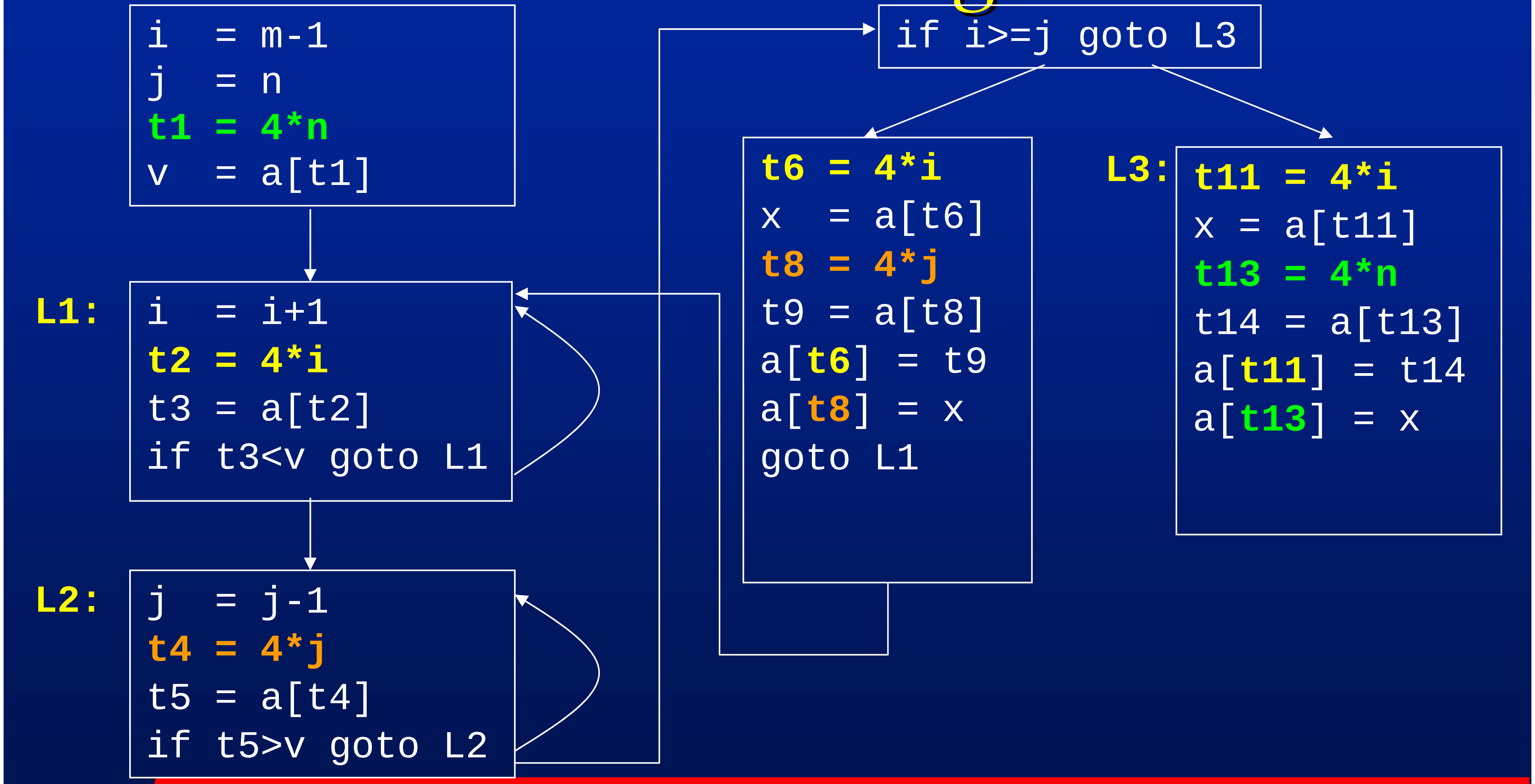
```
t6 = 4*i
x  = a[t6]

t8 = 4*j
t9 = a[t8]
a[t6] = t9

a[t8] = x
goto L1
```

```
L3:
t11 = 4*i
x = a[t11]
t12 = 4*i
t13 = 4*n
t14 = a[t13]
a[t12] = t14
t15 = 4*n
a[t15] = x
```

⇒

```
L3:
t11 = 4*i
x = a[t11]

t13 = 4*n
t14 = a[t13]
a[t11] = t14

a[t13] = x
```

Lokal optimierte Basisblockdarstellung



```
i  = m-1
j  = n
t1 = 4*n
v  = a[t1]
```

```
L1: i  = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto L1
```

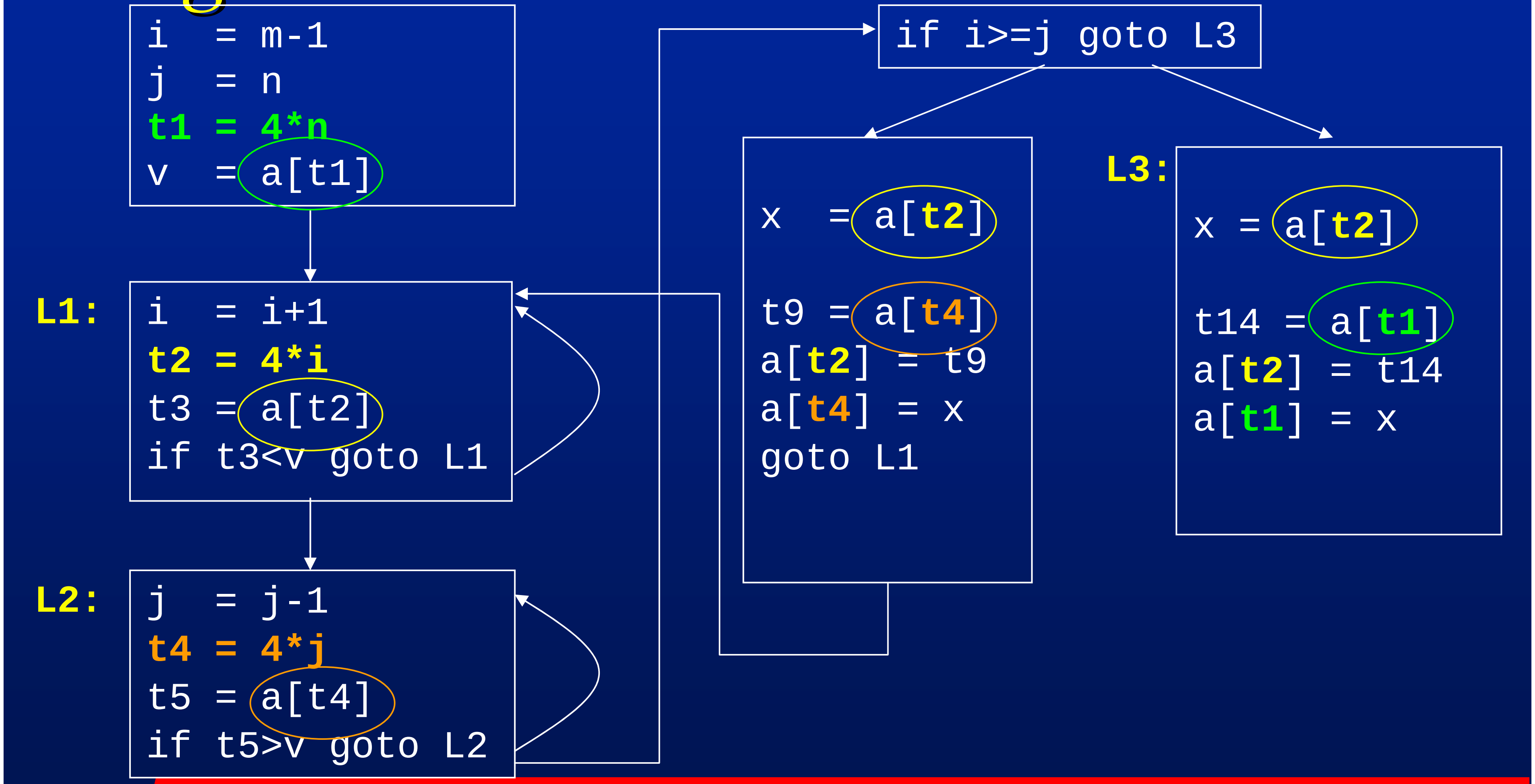
```
L2: j  = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto L2
```

```
if i>=j goto L3
```

```
t6 = 4*i
x  = a[t6]
t8 = 4*j
t9 = a[t8]
a[t6] = t9
a[t8] = x
goto L1
```

```
L3: t11 = 4*i
    x = a[t11]
    t13 = 4*n
    t14 = a[t13]
    a[t11] = t14
    a[t13] = x
```

Elimination globaler gemeinsamer Teilausdrücke



```
i  = m-1
j  = n
t1 = 4*n
v  = a[t1]
```

```
L1: i  = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto L1
```

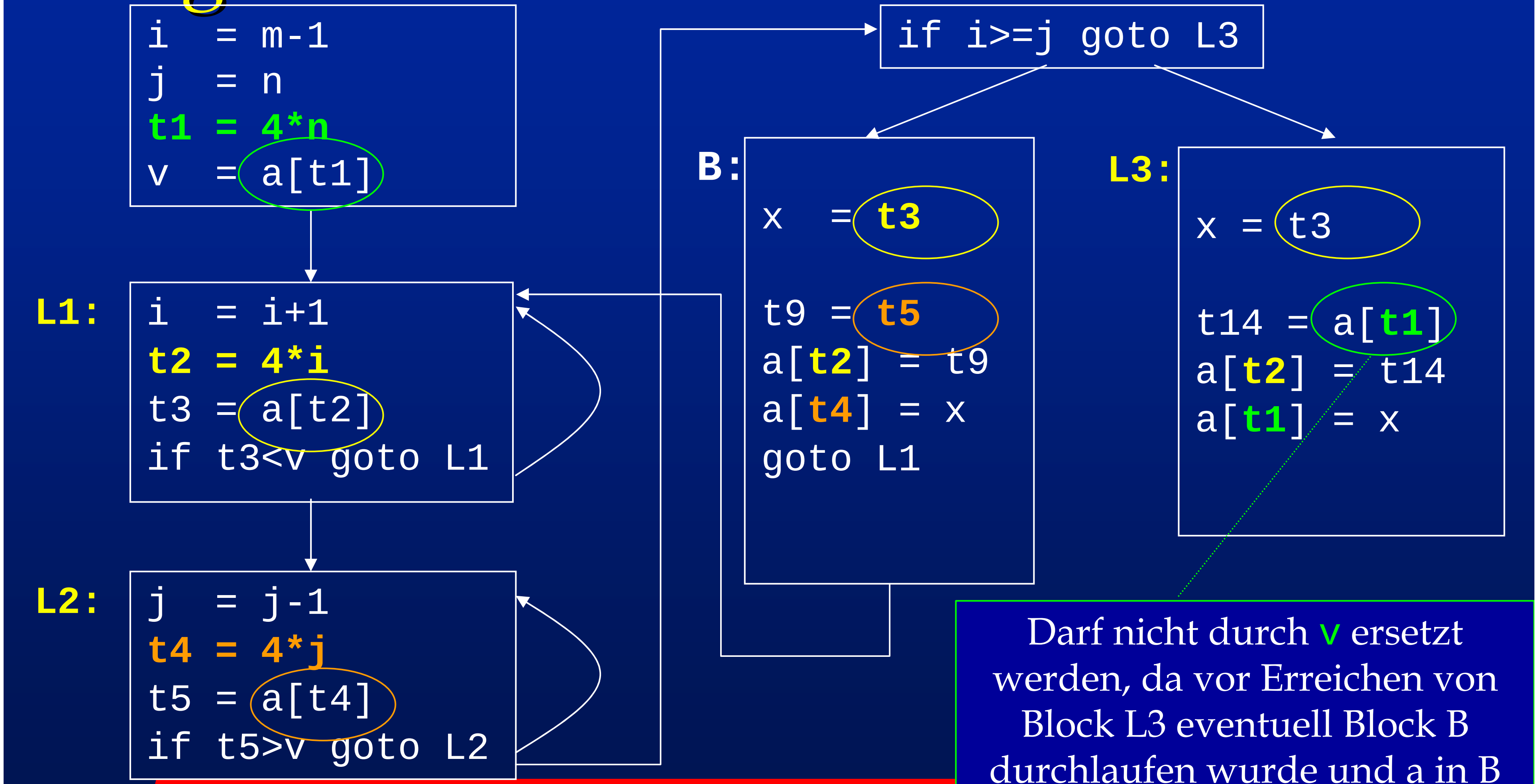
```
L2: j  = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto L2
```

```
if i>=j goto L3
```

```
x  = a[t2]

t9 = a[t4]
a[t2] = t9
a[t4] = x
goto L1
```

```
L3: x = a[t2]

    t14 = a[t1]
    a[t2] = t14
    a[t1] = x
```

Elimination globaler gemeinsamer Teilausdrücke



```
i  = m-1
j  = n
t1 = 4*n
v  = a[t1]
```

```
L1: i  = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto L1
```

```
L2: j  = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto L2
```

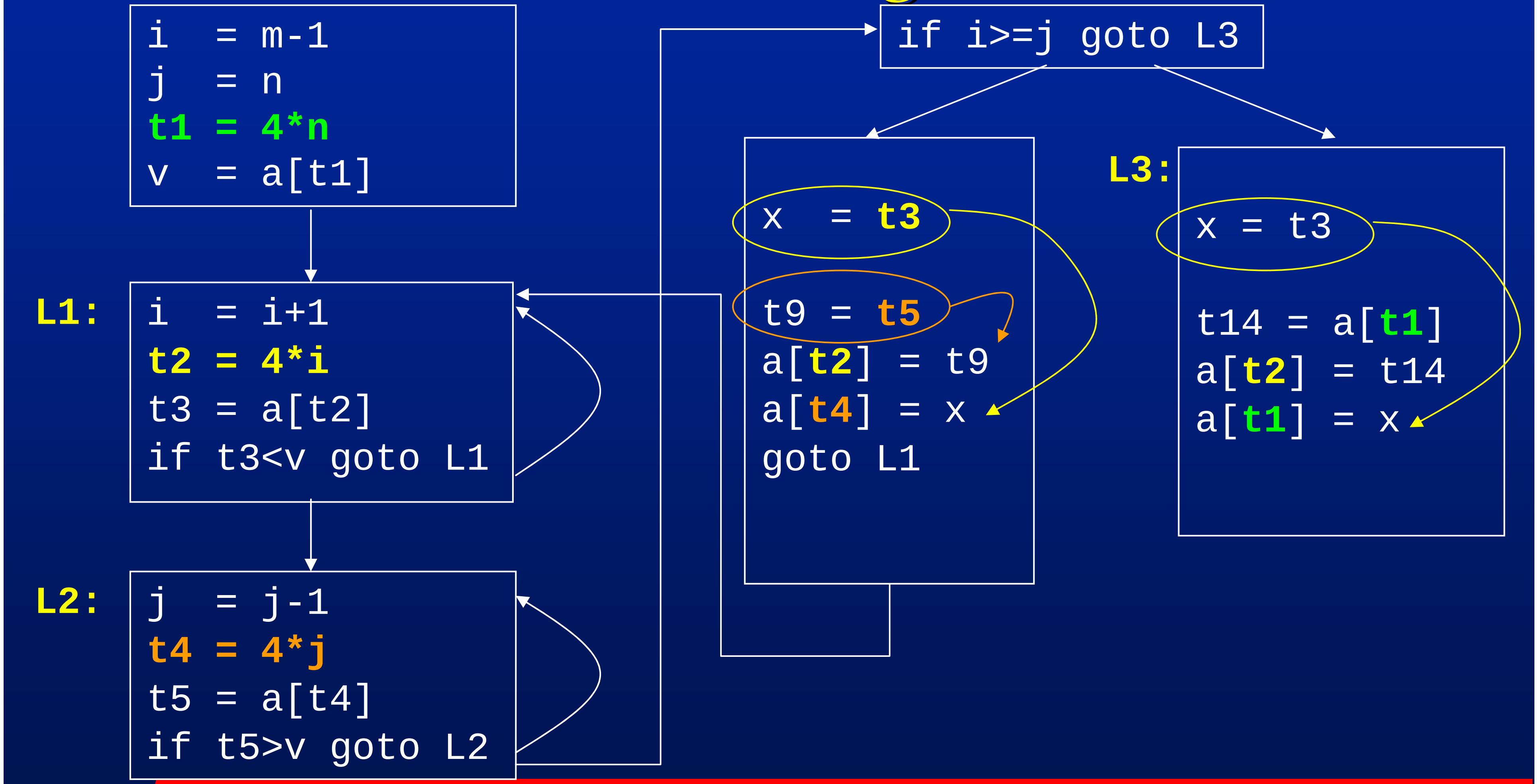
```
if i>=j goto L3
```

```
B:  x   = t3

    t9 = t5
    a[t2] = t9
    a[t4] = x
    goto L1
```

```
L3: x = t3

    t14 = a[t1]
    a[t2] = t14
    a[t1] = x
```

Darf nicht durch v ersetzt werden, da vor Erreichen von Block L3 eventuell Block B durchlaufen wurde und a in B modifiziert wird.

Kopienpropagation und Elimination nutzloser Anweisungen



```
i  = m-1
j  = n
t1 = 4*n
v  = a[t1]
```

```
L1: i  = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto L1
```

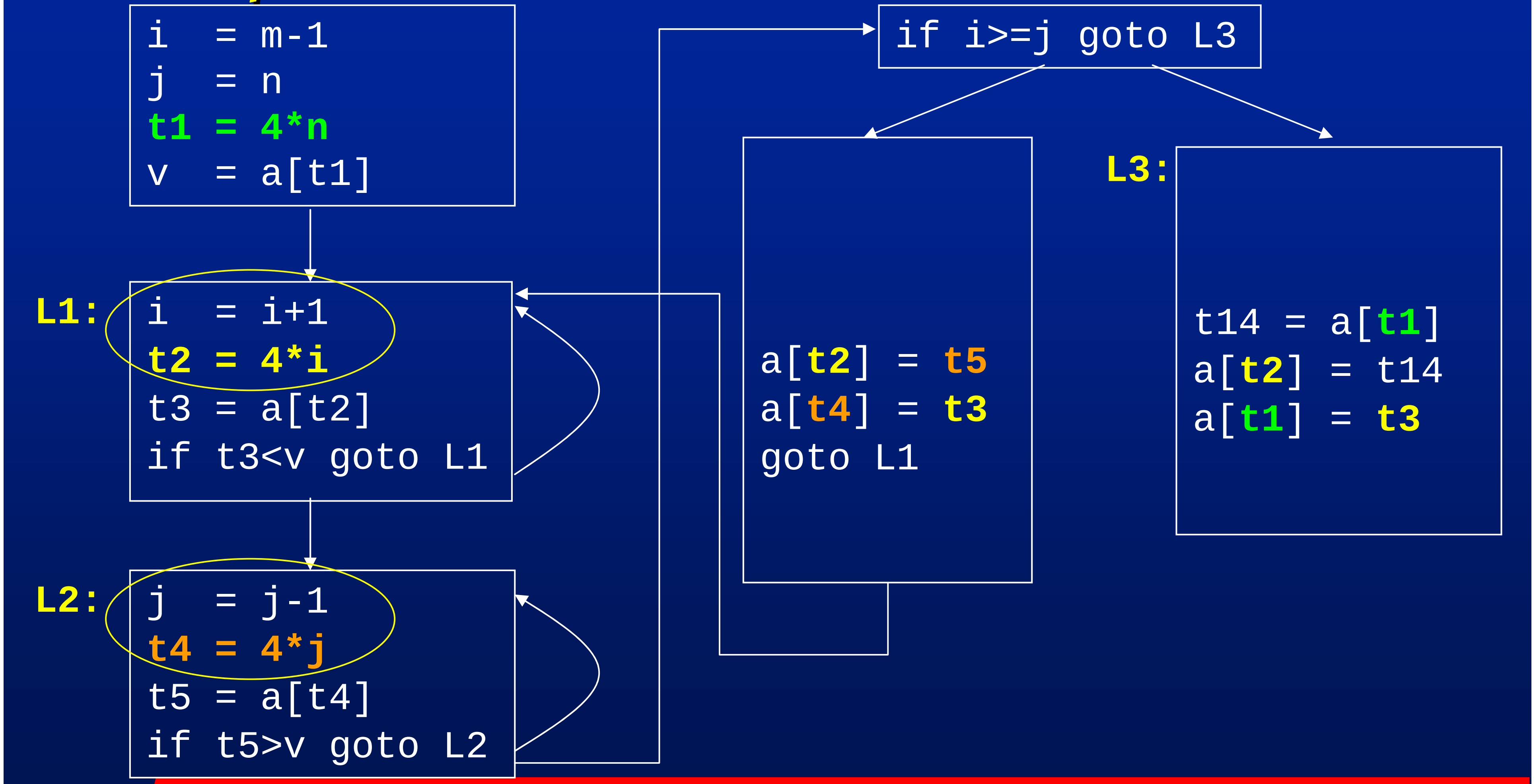
```
L2: j  = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto L2
```

```
if i>=j goto L3
```

```
x  = t3
t9 = t5
a[t2] = t9
a[t4] = x
goto L1
```

```
L3: x = t3
    t14 = a[t1]
    a[t2] = t14
    a[t1] = x
```

Induktionsvariablenerkennung und Operatorreduktion



```
i  = m-1
j  = n
t1 = 4*n
v  = a[t1]
```

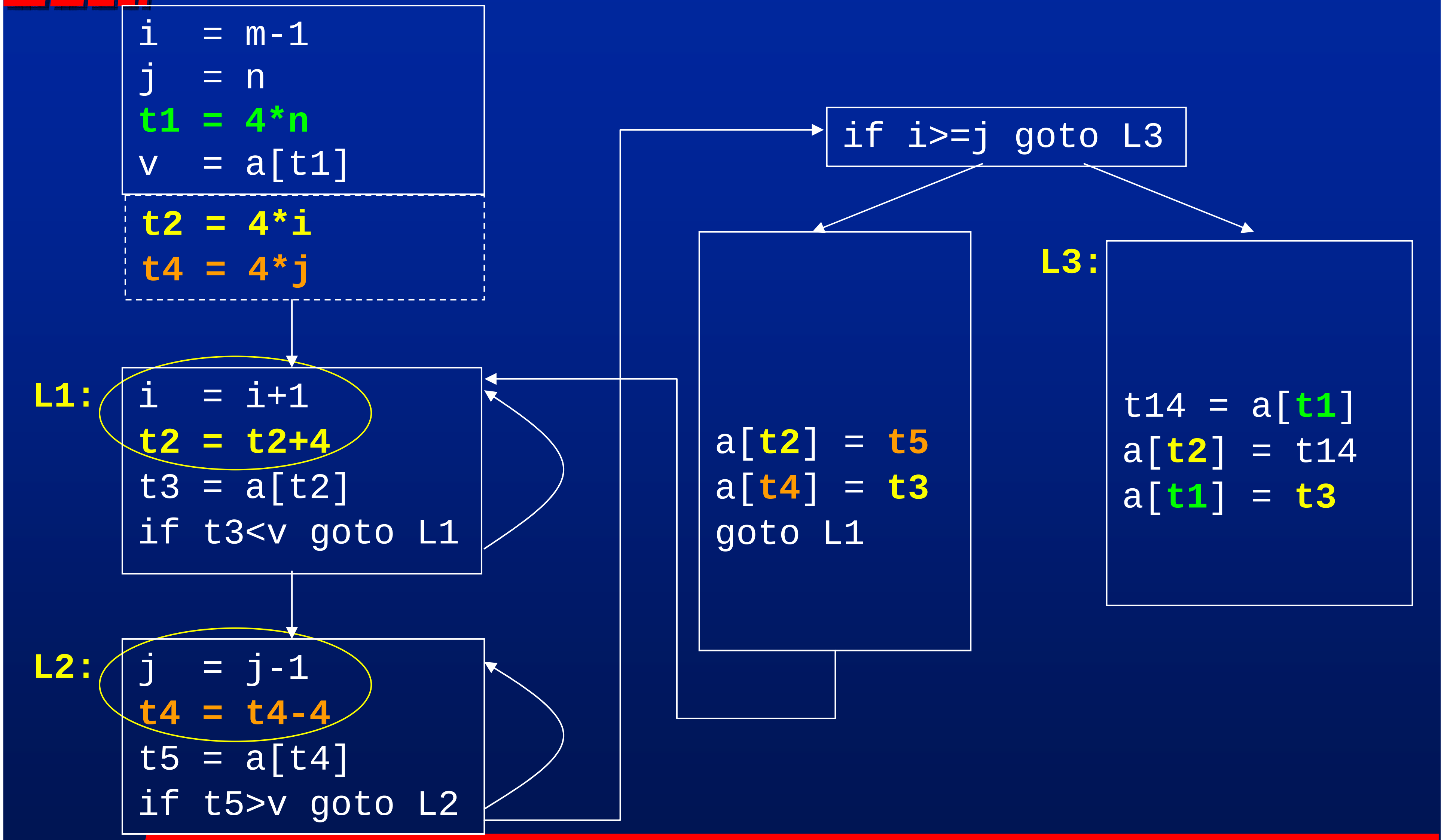
```
L1: i  = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto L1
```

```
L2: j  = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto L2
```

```
if i>=j goto L3
```

```
a[t2] = t5
a[t4] = t3
goto L1
```

```
L3: t14 = a[t1]
    a[t2] = t14
    a[t1] = t3
```



```
i  = m-1
j  = n
t1 = 4*n
v  = a[t1]
```

```
t2 = 4*i
t4 = 4*j
```

```
L1: i  = i+1
    t2 = t2+4
    t3 = a[t2]
    if t3<v goto L1
```

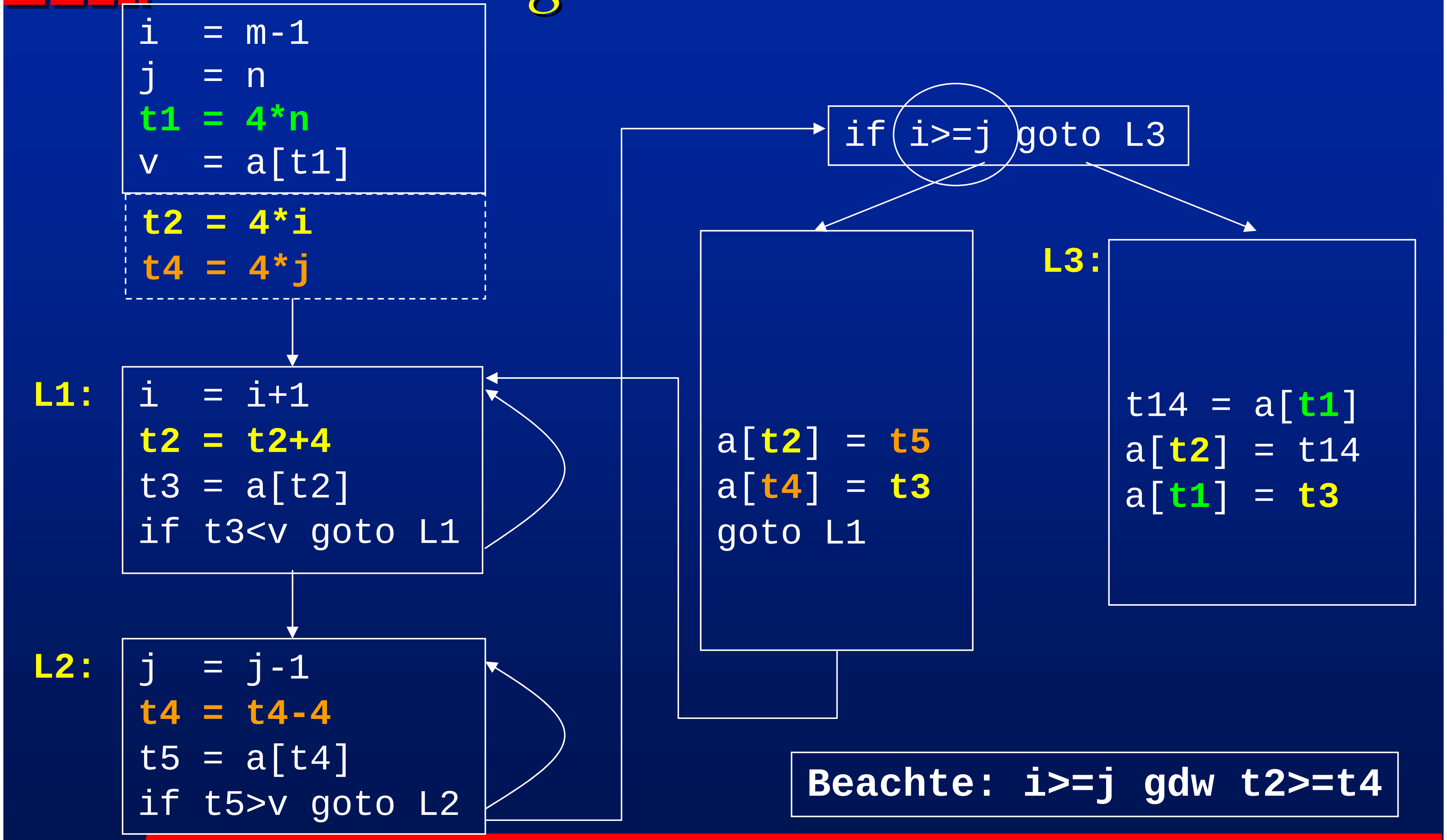
```
L2: j  = j-1
    t4 = t4-4
    t5 = a[t4]
    if t5>v goto L2
```

```
if i>=j goto L3
```

```
a[t2] = t5
a[t4] = t3
goto L1
```

```
L3: t14 = a[t1]
    a[t2] = t14
    a[t1] = t3
```

Eliminierung von Induktionsvariablen



```
i  = m-1
j  = n
t1 = 4*n
v  = a[t1]
```

```
t2 = 4*i
t4 = 4*j
```

```
L1: i  = i+1
    t2 = t2+4
    t3 = a[t2]
    if t3<v goto L1
```

```
L2: j  = j-1
    t4 = t4-4
    t5 = a[t4]
    if t5>v goto L2
```

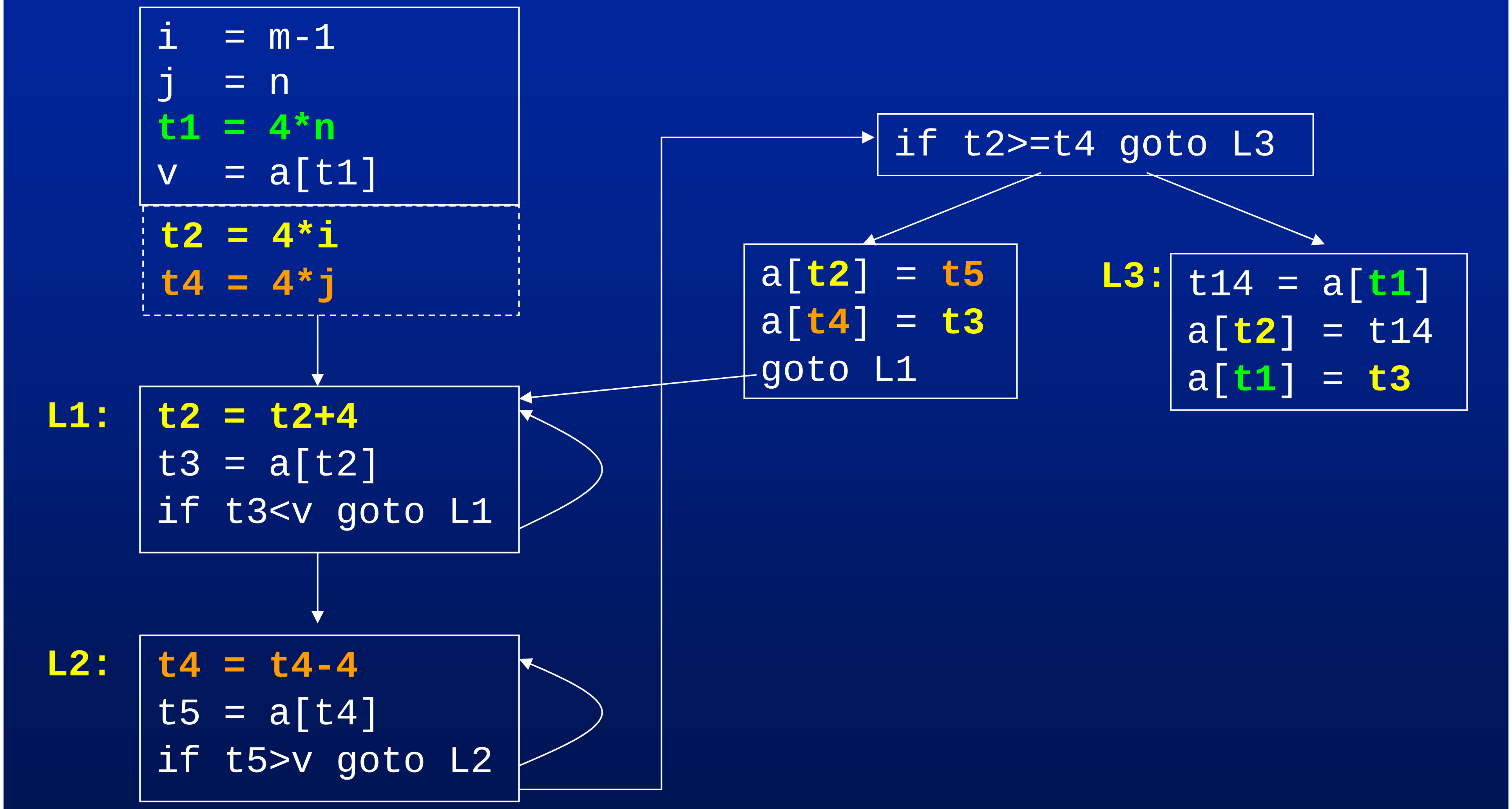
```
if i>=j goto L3
```

```
a[t2] = t5
a[t4] = t3
goto L1
```

```
L3: t14 = a[t1]
    a[t2] = t14
    a[t1] = t3
```

Beachte: i>=j gdw t2>=t4

Optimiertes Programm



```
i  = m-1
j  = n
t1 = 4*n
v  = a[t1]
```

```
t2 = 4*i
t4 = 4*j
```

```
L1: t2 = t2+4
    t3 = a[t2]
    if t3<v goto L1
```

```
L2: t4 = t4-4
    t5 = a[t4]
    if t5>v goto L2
```

```
if t2>=t4 goto L3
```

```
a[t2] = t5
a[t4] = t3
goto L1
```

```
L3: t14 = a[t1]
    a[t2] = t14
    a[t1] = t3
```