

Compiler Optimizations

Unnikrishnan C

September 15, 2019

- Examples
 - Global common sub-expression elimination
 - Copy propagation
 - Constant propagation and constant folding
 - Induction variable elimination and strength reduction
 - Loop unrolling and Function Inlining
 - Tail recursion removal.
 - Vectorization (for SIMD processor)
 - Parallelization (for Multi-core processors)
 - Loop Interchange
- LLVM Assignment: Local Optimizations (within a BasicBlock)

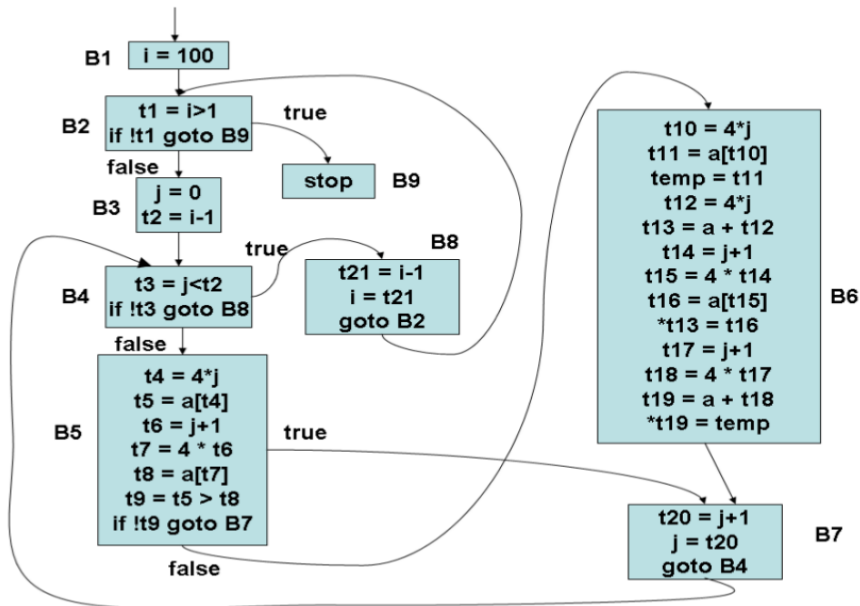
- Description
 - Copy propagation (remove a statement $x := y$, replace all uses of x by y)
 - Constant propagation (if a variable is having constant value, replace all uses with constant value)
 - Strength Reduction (e.g. Replace $x = y * 2$ by $x = y + y$)
 - Induction variable Elimination (for loops)
 - Tail recursion (removing recursion)
 - Function inlining (to remove space and time overhead of function call)
 - Loop interchange (for cache locality)
- LLVM Assignment: First three optimizations comes within local optimizations.
- LLVM Assignment: local optimizations not limited to the first three optimizations.

- Code optimization needs following information
 - Which definition reach a point.
 - Which expression are recomputed.
 - Which copies and constants can be propogated, and many more.
 - *A defintion kills value of the defined variable.*
- All the above information computed using Data Flow Analysis

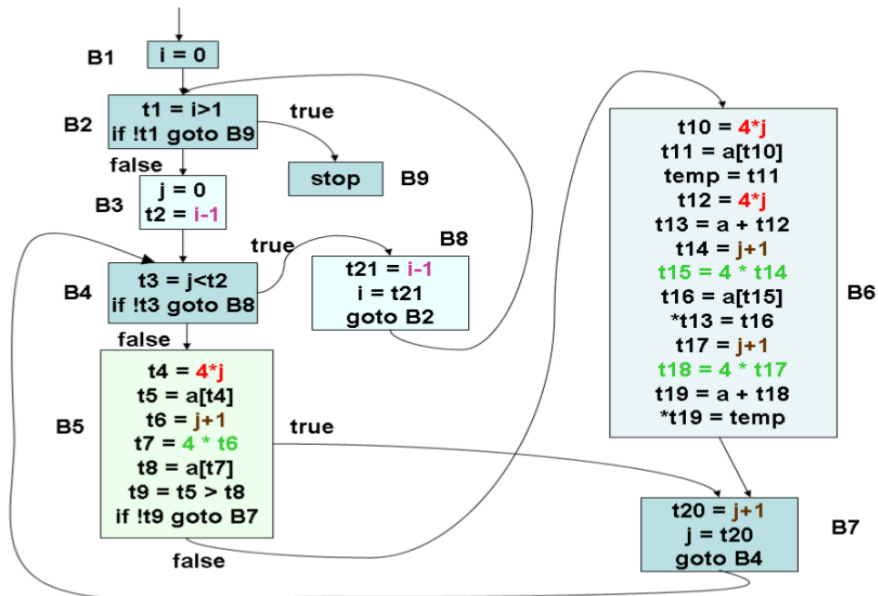
```
for (i=100; i>1; i--) {  
    for (j=0; j<i-1; j++) {  
        if (a[j] > a[j+1]) {  
            temp = a[j];  
            a[j+1] = a[j];  
            a[j] = temp;  
        }  
    }  
}
```

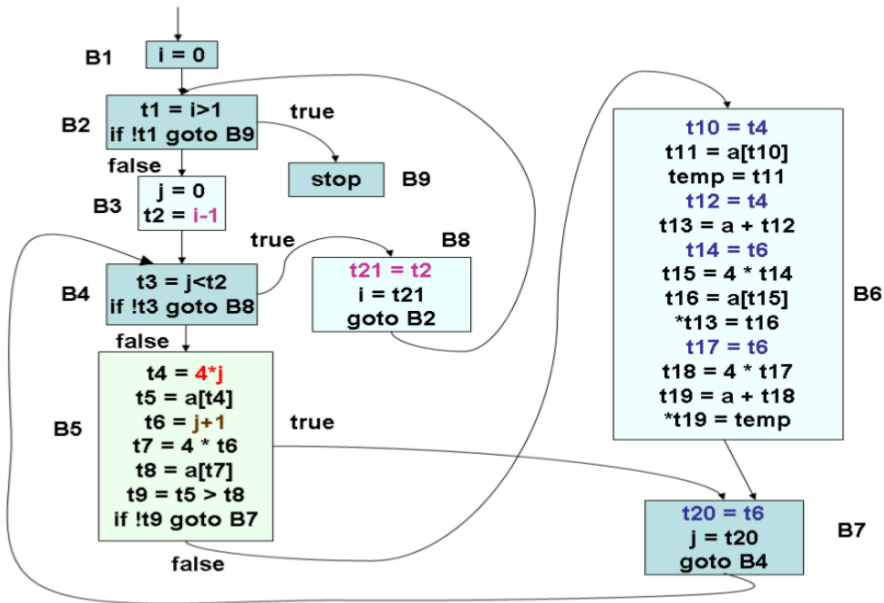
- int a[100]
- array a runs from 0 to 99
- No special jump out if array is already sorted

Control Flow Graph (CFG) of Bubble Sort

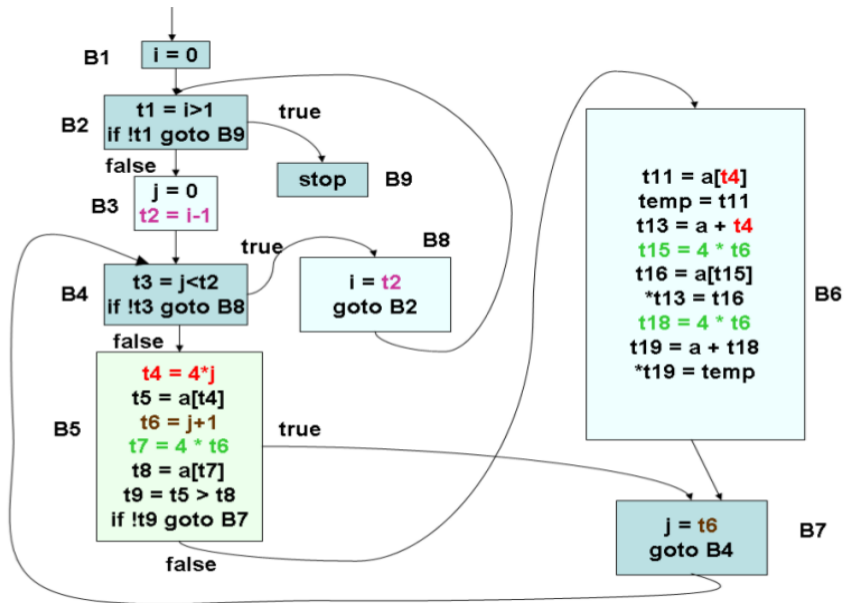


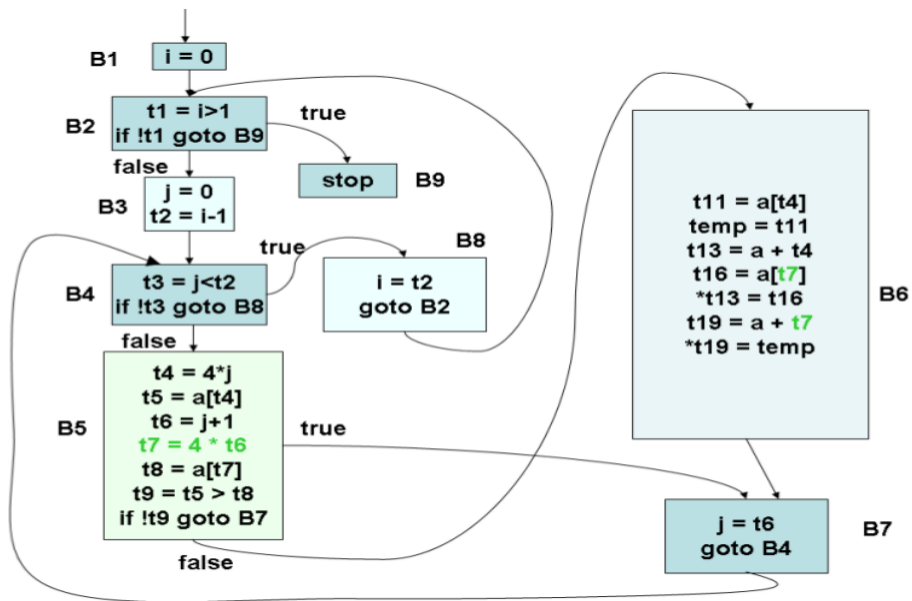
Global Common Sub Expression (GCSE)-one



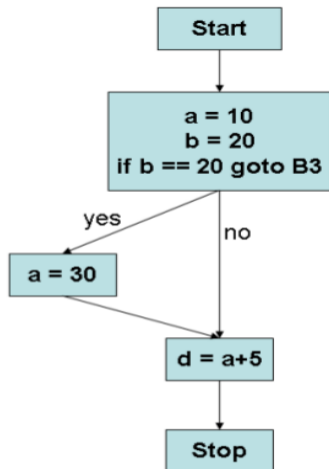


Copy Propagation

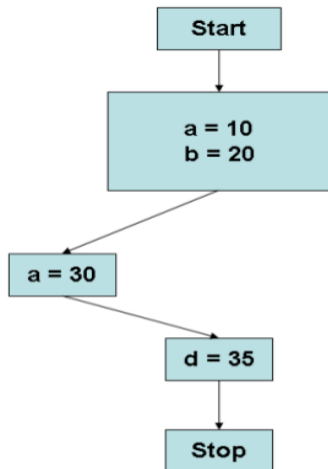




Constant Propagation and Folding Example



Before constant propagation



After constant propagation and folding

```
t1 = 202
i = 1
L1: t2 = i>100
    if t2 goto L2
    t1 = t1-2
    t3 = addr(a)
    t4 = t3 - 4
    t5 = 4*i
    t6 = t4+t5
    *t6 = t1
    i = i+1
    goto L1
L2:
```

**Before LIV
code motion**

```
t1 = 202
i = 1
    t3 = addr(a)
    t4 = t3 - 4
L1: t2 = i>100
    if t2 goto L2
    t1 = t1-2
    t5 = 4*i
    t6 = t4+t5
    *t6 = t1
    i = i+1
    goto L1
L2:
```

**After LIV
code motion**

```
t1 = 202
i = 1
t3 = addr(a)
t4 = t3 - 4
L1: t2 = i>100
    if t2 goto L2
    t1 = t1-2
    t5 = 4*i
    t6 = t4+t5
    *t6 = t1
    i = i+1
    goto L1
L2:
```

**Before strength
reduction for t5**

```
t1 = 202
i = 1
t3 = addr(a)
t4 = t3 - 4
t7 = 4
L1: t2 = i>100
    if t2 goto L2
    t1 = t1-2
    t6 = t4+t7
    *t6 = t1
    i = i+1
    t7 = t7 + 4
    goto L1
L2:
```

**After strength reduction
for t5 and copy propagation**

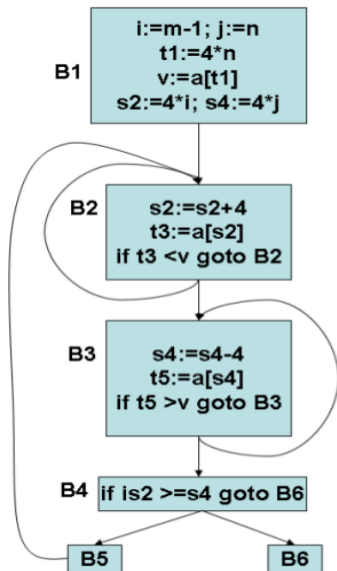
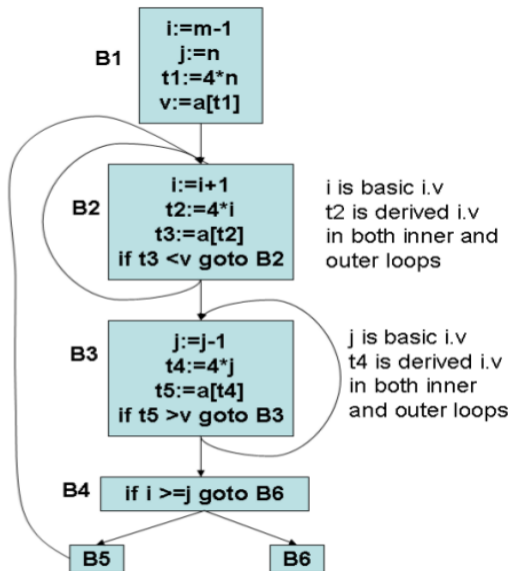
```
t1 = 202
i = 1
t3 = addr(a)
t4 = t3 - 4
t7 = 4
L1: t2 = i > 100
    if t2 goto L2
    t1 = t1 - 2
    t6 = t4 + t7
    *t6 = t1
    i = i + 1
    t7 = t7 + 4
    goto L1
L2:
```

**Before induction variable
elimination (i)**

```
t1 = 202
t3 = addr(a)
t4 = t3 - 4
t7 = 4
L1: t2 = t7 > 400
    if t2 goto L2
    t1 = t1 - 2
    t6 = t4 + t7
    *t6 = t1
    t7 = t7 + 4
    goto L1
L2:
```

**After eliminating i and
replacing it with t7**

Induction Variable Elimination and Strength Reduction



- Topics: Loop Interchange, Parallelization Vectorization will be covered in coming classes.
- Read online on Data Flow Analysis and Abstract Interpretation (upto you).