



CS 4120
Introduction to Compilers

Ross Tate
Cornell University

Lecture 33: Loop Optimizations II

Induction-Variable Strength Reduction

```

m = i * 8;
while (i < 10) {
    i = i + 2;
    m = i * 8;
    use(m + 6);
}

m = i * 8;
while (i < 10) {
    i = i + 2;
    m = m + 16;
    use(m + 6);
}
    
```

*Handwritten notes: $m = i + (i * 4)$*

2

Loop Unrolling

```

m = i * 8;
while (i < 10) {
    i = i + 2;
    m = m + 16;
    use(m + 6);
}

m = i * 8;
while (i < 10 - 2 * 1) {
    i = i + 2;
    m = m + 16;
    use(m + 6);
}
    
```

Handwritten notes: Red arrows and numbers (4, 32) indicating unrolling steps.

3

Common-Subexpression Elimination

```

a := i;
b := a * 2;
if (c)
    d := a + b;
else
    d := i + b;
e := b * 3;
f := b * 3;
i := i * 2;
return e + f + i * 2;
    
```

*Handwritten notes: Red annotations showing subexpressions like $a := i$, $b := a * 2$, $d := i + i * 2$, $e := b * 3$, $f := b * 3$. Includes a tree diagram for $e + f + i * 2$.*

```

Foo(n)
if (n > 0)
    return f(n-1) + f(n-1)
else
    return 0
    
```

5