Tour of common optimizations
Simple example

```plaintext
foo(z) {
    x := 3 + 6;
    y := x - 5
    return z * y
}
```
Simple example

```cpp
foo(z) {
    x := 3 + 6;  \text{const prop (CP)}
    y := x - 5 \quad 4 \text{ (CF)}

    return z * y
\}
```

\text{Constant folding (CF)}

\text{Arith simp}

\text{Strength reduction}
Another example

\[ x := a + b; \]

\[ \ldots \]

\[ y := a + b; \times \]
Another example

\[ x := a + b; \]
\[ \ldots \]
\[ y := a + b; \]

\{ only if \( x \), \( a \), \( b \) not modified \}
Another example

```plaintext
if (...) {
    a := read();
    x' := a + b; x := x'
    print(x);
} else { x' := a + b }

... print(y)

y := a + b; x'
```
Another example

```java
if (...) {
    a := read();
    x := a + b;
    t := a + b;
    print(x);
}
else {
    y := a + b;
}

...
Another example

\[
x := y \\
\ldots \\
z := z + x
\]
Another example

\[
x := y
\]
\[
\ldots
\]
\[
z := z + x
\]

\textit{\textcolor{red}{copy prop}}
Another example

\[
\begin{aligned}
x & := y \\
\ldots \\
z & := z + y
\end{aligned}
\]

What if we run CSE now?

\[
\begin{aligned}
x & := E \\
\ldots \\
\text{\ul{E}} \ldots \\
x
\end{aligned}
\]
Another example

\[ x := y \]
\[ \ldots \]
\[ z := z + y \]

What if we run CSE now?
Another example

\[ x := y**z \]

\[ \ldots \]

\[ x := \ldots \]
Another example

\begin{align*}
  x &:= y**z \\
  \ldots &
  \} \quad \text{if } x \text{ is not used}
  \quad \text{dead assignment elim} \\
  x &:= \ldots
\end{align*}

- Often used as a clean-up pass

\begin{align*}
  x &:= y \\
  z &:= z + x \\
  &\quad \text{Copy prop} \\
  x &:= y \\
  z &:= z + y \\
  &\quad \text{DAE}
\end{align*}
Another example

if (false) {

...

}

if (false) {
Another example

if (false) {
...}

dead code elim
( unreachable code elim)

Another common clean up opt
Another example

• In Java:

```java
a = new int [10];
for (index = 0; index < 10; index ++) {
    a[index] = 100;
}
```
Another example

• In “lowered” Java:

```java
[a = new int[10];] a.length = 10
for (index = 0; index < 10; index++) {
    if (index < 0 || index >= a.length()) {
        throw OutOfBoundsException;
    }
    a[index] = 0;
}
```
Another example

• In “lowered” Java:

```java
int a[] = new int[10];  // 1
for (int index = 0; index < 10; index++) {
    if (index < 0 || index >= a.length()) {
        throw OutOfBoundsException;
    }
    a[index] = 0;
}
```

- Index `index` iterates from 0 to 9, inclusive.
- The check `index < 0 || index >= a.length()` ensures the bounds are respected.
- unreachable code elim
Another example

```plaintext
p := &x;
*p := 5
y := x + 1;
```

5
Another example

```
p := &x;
*p := 5
y := x + 1;
x := 5;
*p := 3
y := x + 1;
```

```
x := 5;
*p := 3
y := x + 1;  
```
Another example

for \( j := 1 \) to \( N \)
  for \( i := 1 \) to \( M \)
    \( a[i] := a[i] + b[j] \)

\[
\begin{align*}
    t_1 &= a + i \times 4 \\
    t_2 &= t_1 \\
    t_3 &= a + i \times 4 \times t_1 \\
    \star t_1 &= t_2 + t
\end{align*}
\]
Another example

\begin{align*}
  &\text{for } j := 1 \text{ to } N \quad t := b[j] \\
  &\text{for } i := 1 \text{ to } M \\
  &\quad a[i] := a[i] + b[j] \\
\end{align*}
Another example

\[
\text{area}(h,w) \{ \text{return } h \times w \}
\]

\[
h := \ldots;
\]

\[
w := 4;
\]

\[
a := \text{area}(h,w)
\]

\[
h \times 4
\]

\[
h \ll 2
\]
Another example

\[
\text{area}(h,w) \{ \text{ return } h \times w \} \\
h := \ldots; \\
w := 4; \\
a := \text{area}(h,w) \\
\]

Many "killy" opts became important after inlineing
Optimization themes

• Don’t compute if you don’t have to
  – unused assignment elimination

• Compute at compile-time if possible
  – constant folding, loop unrolling, inlining

• Compute it as few times as possible
  – CSE, PRE, PDE, loop invariant code motion

• Compute it as cheaply as possible
  – strength reduction

• Enable other optimizations
  – constant and copy prop, pointer analysis

• Compute it with as little code space as possible
  – unreachable code elimination