Tour of common optimizations
Simple example

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

\[ x := a + b; \]

\[ \ldots \]

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

\[
\begin{align*}
    x & := a + b; \\
    \ldots & \\
    y & := a + b; \quad \text{\textcolor{red}{\checkmark}} \\
\end{align*}
\]

\textcolor{red}{\textit{}\{ only if } x, a, b \text{ not modified!}}
Another example

```
if (...) {
    x := a + b;
}

...

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

\[
\text{if (\ldots)} \{ \\
\quad \textcolor{red}{t := a + b} \\
\text{\quad x := a + b; t} \\
\} \quad \text{else} \quad \{ \textcolor{red}{t := a + b} \} \\
\]

\[
\ldots
\]

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

\texttt{x := y}
\texttt{\ldots}
\texttt{z := z + x}
Another example

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

\{ x, y not modified \}

\textit{copy prop}
Another example

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

What if we run CSE now?
Another example

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

What if we run CSE now?
Another example

\[ x := y^{**z} \]

\[ \ldots \]

\[ x := \ldots \]
Another example

\[ x := y^{**}z \]

... \{ if \( x \) is not used \}

\[ x := \ldots \]

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

• Often used as a clean-up pass

```
x := y
z := z + x
Copy prop
x := y
z := z + y
DAE
x := y
z := z + y
```
Another example

```java
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;
}
```
• In “lowered” Java:

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

- In “lowered” Java:

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

- `index` is in `[0..9]` by range analysis.
- This is similar to CP if we assume stmt 0 acts like `a.length := 10`.
Another example

\[
p := \&x; \\
*p := 5 \\
y := x + 1;
\]
Another example

\begin{align*}
p & := \&x; \\
\times*p & := 5 \\
y & := x + 1; \quad 6
\end{align*}

\begin{align*}
x & := 5; \\
*p & := 3 \\
y & := x + 1; \quad ???
\end{align*}

\textcolor{red}{\textit{pointer/alias analysis}}
Another example

for j := 1 to N
    for i := 1 to M
        a[i] := a[i] + b[j]
Another example

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

\[ t := b[j] \]

Loop invariant
Code motion
Another example

```plaintext
area(h,w) { return h * w }

h := ...;
w := 4;
a := area(h,w)
```
Another example

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

\[
\begin{align*}
  h & := \ldots; \\
  w & := 4; \\
  a & := \text{area}(h,w)
\end{align*}
\]

\[ h \times w \]
\[ h \times 4 \]
\[ h \ll 2 \]

Many "silly" opts became important after inlining
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