Tour of common optimizations
Simple example

```c
foo(z) {
    x := 3 + 6;  \( \text{9} \)
    y := x - 5  \( \text{4} \)
    return z * y \( \text{4} \)
}
```
Simple example

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

Constant folding (CF)

Arith simple

Strength reduction
Another example

\[ x := a + b; \]

\[ \ldots \]

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

\[
x := a + b;
\]

\[
\ldots
\]

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

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

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

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

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

```
...  
```

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

Partial Redundancy Elimination (PRE)
Another example

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

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

\( \{ \) x, y not modified \( \}

\( \) copy prop
Another example

What if we run CSE now?

\[ \mathbf{v} = \tilde{E} \]

\[ x := y \]

\[ z := z + x \]

\[ \vdots \]

\[ \ldots \tilde{E} \ldots \]
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

\[ x := y**z \]

\[
\ldots \}
\textit{if } x \textit{ is not used}
\]

\[ x := \ldots \]

\[ x := y \]

\[ z := z + x \]

\[ \text{dead assignment elim} \]

\[ \text{unused assignment elim} \]

• 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

if (false) {
    ...
}

Another example

```java
if (false) {
    ...
}
```

dead code elimination (unreachable code elimination)

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 new OutOfBoundsException;
    }
    a[index] = 0;
}
```

1. Kinda like CP if we assume stmt 0 acts like a.length := 10
2. Range analysis
3. Branch folding + unreachable code elim
Another example

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

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

```markdown
pointer/alias analysis
```
```
x := 5;
*p := 3
y := x + 1;
```

???
Another example

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

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

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

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

\[ h \times w \]
Another example

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

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

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