Tour of common optimizations

Simple example

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

Another example

```plaintext
x := a + b;
...
```

Another example

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

Another example

```plaintext
x := a + b;
... y := a + b;
```
Another example

```plaintext
if (...) {
    a := read(); t := a + b
    x := &a->b; t
    print(x);
} else { t := a + b }  // Partial Redefining
...
```
Another example

- Often used as a clean-up pass

```
x := y**z
...  
x := ...
```

Another example

```
if (false) {

}
```

Another example

```
if (false) {

dead code elim
(unreach code elim)

Another common clean up opt
```

Another example

- In "lowered" Java:

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

Another example

- In "lowered" Java:

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

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

Another example

p := &x;  \text{pointe/alias analysis}
*p := 5
y := x + 1;

Another example

x := 5;
*p := 3
y := x + 1;  \text{???}

Another example

\begin{verbatim}
for j := 1 to N
  for i := 1 to M
    a[i] := a[i] + b[j]
\end{verbatim}

Another example

\begin{verbatim}
for j := 1 to N
  for i := 1 to M
    a[i] := a[i] + b[j]
\end{verbatim}

Another example

\begin{verbatim}
area(h,w) { return h * w }
h := ...;
w := 4;
a := area(h,w)
\end{verbatim}

Another example

\begin{verbatim}
area(h,w) { return h * w }
h := ...;
w := 4;
a := area(h,w)
\end{verbatim}

Many "ill" dot become
important after inlining

h <= 2
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