Tour of common optimizations

Simple example

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

Another example

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

```plaintext
if (...) {
  x := a + b;
  t := a + b;  // Partial Redecoration
  y := a + b;  // Eliminate DRF
  ...
}
```

Another example

```plaintext
x := y
...
z := z + x
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```

Another example

```plaintext
x := y
...
z := x + y
```
Another example

- Often used as a clean-up pass

\[
x := y^{z^2}
\]

\[
\ldots
\]

\[
\text{if } x \text{ is not used}
\]

\[
\text{dead assignment elim}
\]

\[
\text{avoid assignment elim}
\]

Another example

- In 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;
    }
}
```
Another example

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

Another example

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

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

Another example

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

Another example

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

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

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

Another example

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

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

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

Another example

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

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

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

Another example

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

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

```
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]
```
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