Tour of common optimizations

How excited are you about this course?

- A. Super excited
- B. A little excited
- C. Not that excited
- D. Not at all excited

How nervous are you about this course?

- A. Super nervous
- B. A little nervous
- C. Not that nervous
- D. Not at all nervous

What is your primary reason for 231?

- A. I'm doing research and compilers and related areas, so I want to learn about compilers
- B. I'm not doing research in this area, but still want to learn about compilers
- C. A friend recommended it
- D. I want to only take AI and Machine learning courses, but the program requires me to take other classes too, so here I am. Ugh
- E. Other

Simple example

foo(z) {						
	x	:=	3	+	6;	
	У	:=	x	_	5	
	return			Z	*	У

}

Simple example

foo(z) {

$$x := 3 + 6; 9$$
 Constant folding (CF)
Court prop(CP) g
 $y := 5 + (CF)$
return $z = 4 + 4 + (CP)$
}
 $z << 2 + 4 + (CP)$
Strength reduction

x := a + b;

y := a + b;

$$x := a + b;$$

$$\dots$$

$$y := a + b; X$$

$$nodified$$

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

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

Partial Redundency Eliminatia PRE

y := a + b; t

х := у

What if we run CSE now?

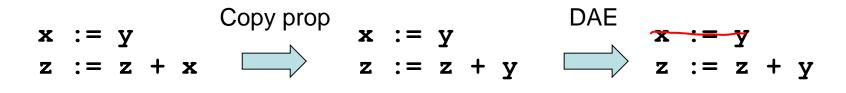
z := z + y

$$\begin{array}{l} \mathbf{x} & := \mathbf{y} \\ \dots \\ \mathbf{z} & := \mathbf{z} + \mathbf{y} \times \end{array}$$

What if we run CSE now?

x := ...

Often used as a clean-up pass



if (false) {

}

if (false) {
 ...
}

dead code elim (unreachable code elim)

Another comma clean up oft

• In Java:

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

• In "lowered" Java:

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

• In "lowered" Java:

p :=
$$\&x$$

× *p := 5
y := $x + 1; 6$

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

for j := 1 to N
$$t = b[j]$$

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

```
area(h,w) { return h * w }
h := ...;
w := 4;
a := area(h,w)
```

area
$$(h, w)$$
 { return $h * w$ }
 $h := \dots;$
 $w := 4;$
 $a := area (h, w)$ Many "hilly" of the become
 $h \notin W$ important of the inlining
 $h \ll \lambda$

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