CS 536 Announcements for Wednesday, April 24, 2024

Course evaluation – log into heliocampusac.wisc.edu using your NetID

Last Time
- optimization overview
- peephole optimization
- loop optimizations

Today
- wrap up optimization
- copy propagation

Optimization Review

Goal: Produce "better" code that does the "same thing" as the original code.
- better =
- same thing =

When?
- before code generation
- after code generation

Important considerations
- performance/profitability – want to be sure optimization is "worth it"
- safety – original source code, non-optimized target code, and optimized target code all do the "same thing" / have the same "meaning"

Look at optimizations that
- are sound transformations
- recognize a behavior in a program & replace it with a "better" version
Copy propagation

**copy statement**

\[ x = y. \]

**definition** of \( x \)

**use** of \( y \)

**Idea:** Suppose we are at **use** \( U \) of \( x \) and a **definition** \( D \) of \( x \) (of the form \( x = y \)) reaches \( U \)

- If
  1) no other definition of \( x \) reaches \( U \) and
  2) \( y \) does not change between \( D \) and \( U \)
- then we can replace the use of \( x \) at \( U \) with \( y \)

**Example**

\[
\begin{align*}
x &= 3. \\
y &= 5. \\
p &= x. \\
\text{if } w*x > 9 [ \\
  x &= 4. \\
  z &= x + w*y. \\
\]}
\]
\[
\begin{align*}
\text{else } [ \\
  z &= 2*y + x. \\
\]}
\]
\[
q &= 5*p. \\
s &= z + x. \\
t &= s + y.
\]
How is this an optimization?

- can create useless code (which can then be removed)

- can create improved code

- constant folding

- if done before other optimizations, can improve results
Copy propagation (cont.)

Recall: Suppose we are at use U of x and a definition D of x (of the form $x = y$) reaches U

- If
  - 1) no other definition of x reaches U and
  - 2) $y$ does not change between D and U
- then we can replace the use of x at U with $y$

So, to do copy propagation, we must make sure two properties hold:

Property 1) No other definition of x reaches U

Property 2) $y$ does not change between D and U

How?

Property 1) No other definition of x reaches U

- How? Do a reaching-definitions analysis
  - one way: data flow analysis
  - another way: create control flow graph (CFG)
Example

x = 3.
y = w.
p = x.

if w * x > 9 [
x = 4.

while x < 10 [
z = x + w * y.
x = x + 1.
]
]

else [
z = 2 * y + x.
]

q = 5*p.
s = z + x.
t = s + y.
Copy Propagation (cont.)

Property 2) $y$ does not change between D and U

- If $y$ is a constant, then this is trivially true.

- If on any path through the CFG from D to U there is a definition of $y$, then

  - If $y$ and $z$ are aliases and there is a definition of $z$ between D and U, then
Example (cont.)

\[ x = 3. \]
\[ y = w. \]
\[ p = x. \]
\[ \text{if } w \times x > 9 \text{ [ } \]
\[ \quad x = 4. \]
\[ \quad \text{while } x < 10 \text{ [ } \]
\[ \quad \quad z = x + w \times y. \]
\[ \quad \quad x = x + 1. \]
\[ \quad ] \]
\[ \text{else [ } \]
\[ \quad z = 2 \times y + x. \]
\[ \] \]
\[ q = 5 \times p. \]
\[ s = z + x. \]
\[ t = s + y. \]

Optimization Wrap-up