Lecture 11: Static Single Assignment

COS 598C – Advanced Compilers

Prof. David August
Department of Computer Science
Princeton University

Where are we?

- Analysis
  - Control Flow/Predicate
    - Treat basic blocks as a black box
    - Only look at branches
  - Dataflow
    - Look inside basic blocks
    - What is computed where?

Static Single Assignment (SSA) Form

- Each assignment to a variable is given a unique name
- All of the uses reached by that assignment are renamed
  - Trivial for straight line code

\[
\begin{align*}
x &= -1 \\
y &= x \\
x &= 5 \\
z &= x \\
x0 &= -1 \\
y &= x0 \\
x1 &= 5 \\
z &= x1 \\
\end{align*}
\]

- More complex with control flow – Must use Phi nodes

\[
\begin{align*}
\text{if } ( \ldots ) & \text{ x = -1 } \\
\text{else } & \text{ x = 5 } \\
y &= x \\
\text{if } ( \ldots ) & \text{ x0 = -1 } \\
\text{else } & \text{ x1 = 5 } \\
x2 &= \text{Phi}(x0, x1) \\
y &= x2
\end{align*}
\]
SSA Overview (continued)

• What about loops – no problem
  
  \[
  \begin{align*}
  i &= 0 \\
  &\text{do } \{ \\
  &\quad i = i + 1 \\
  &\} \text{ while } (i < 50) \\
  \end{align*}
  \]

  \[
  \begin{align*}
  i0 &= 0 \\
  &\text{do } \{ \\
  &\quad i1 = \text{Phi}(i0, i2) \\
  &\quad i2 = i1 + 1 \\
  &\} \text{ while } (i2 < 50) \\
  \end{align*}
  \]

• Advantages of SSA
  • Explicit DU chains – Trivial to figure out what defs reach a use
  • Each use has exactly 1 definition!!!
  • Explicit merging of values
  • Makes optimizations easier

• Disadvantages
  • When transform the code, must either recompute (slow) or incrementally update (tedious)

Phi Nodes (aka Phi Functions)

• Special kind of copy that selects one of its inputs
• Choice of input is governed by the CFG edge along which control flow reached the Phi node

\[
x0 = x1 = x2 = \text{Phi}(x0, x1)
\]

• Phi nodes are required when 2 non-null paths X→Z and Y→Z converge at node Z, and nodes X and Y contain assignments to V

SSA Construction

• High-level algorithm
  1. Insert Phi nodes
  2. Rename variables
  3. Profit 😊

• A dumb algorithm
  • Insert Phi functions at every join for every variable
  • Solve reaching definitions
  • Rename each use to the def that reaches it (will be unique)

• Problems with the dumb algorithm
  • Too many Phi functions (precision)
  • Too many Phi functions (space)
  • Too many Phi functions (time)
A definition at n forces a Phi node at m iff
- n not in DOM(m), but n in DOM(p) for some predecessors p of m

**Dominance frontier**
The dominance frontier of node X is the set of nodes Y such that
- X dominates a predecessor of Y, but
- X does not strictly dominate Y

**Dominator Tree**

First BB is the root node, each node dominates all of its descendants

**Computing Dominance Frontiers**

For each join point X in the CFG
- For each predecessor of X in the CFG
  - Run up to the IDOM(X) in the dominator tree,
  - adding X to DF(N) for each N between X and IDOM(X)
Draw the dominator tree, calculate the dominance frontier for each BB

```
BB0
  \_ BB1
  |    \_ BB2
  |    |    \_ BB3
  |    |    \_ BB4
  |    |    \_ BB5
  |    |    \_ BB6
  |    |    \_ BB7

BB0
  \_ BB1
  |    \_ BB2
  |    |    \_ BB3
  |    |    \_ BB4
  |    |    \_ BB5
  |    |    \_ BB6
  |    |    \_ BB7

BB0
  \_ BB1
  |    \_ BB2
  |    |    \_ BB3
  |    |    \_ BB4
  |    |    \_ BB5
  |    |    \_ BB6
  |    |    \_ BB7
```

**Phi Node Insertion Algorithm**

- Compute dominance frontiers
- Find global names (aka virtual registers)
  - Global if name live on entry to some block
  - For each name, build a list of blocks that define it
- Insert Phi nodes
  - For each global name \( n \)
    - For each BB \( b \) in which \( n \) is defined
      - For each BB \( d \) in \( b \)'s dominance frontier
        - Insert a Phi node for \( n \) in \( d \)
        - Add \( d \) to \( n \)'s list of defining BBs

**Phi Node Insertion – Example**

```
BB0
  \_ BB1
  |    \_ BB2
  |    |    \_ BB3
  |    |    \_ BB4
  |    |    \_ BB5
  |    |    \_ BB6
  |    |    \_ BB7

BB0
  \_ BB1
  |    \_ BB2
  |    |    \_ BB3
  |    |    \_ BB4
  |    |    \_ BB5
  |    |    \_ BB6
  |    |    \_ BB7

BB0
  \_ BB1
  |    \_ BB2
  |    |    \_ BB3
  |    |    \_ BB4
  |    |    \_ BB5
  |    |    \_ BB6
  |    |    \_ BB7

BB0
  \_ BB1
  |    \_ BB2
  |    |    \_ BB3
  |    |    \_ BB4
  |    |    \_ BB5
  |    |    \_ BB6
  |    |    \_ BB7
```

- \( a \) is defined in 0, 1, 3 need Phi in 7
  - then \( a \) is defined in 7 need Phi in 1
  - \( b \) is defined in 0, 2, 6 need Phi in 7
  - then \( b \) is defined in 7 need Phi in 1
  - \( c \) is defined in 0, 1, 2, 5 need Phi in 6, 7
  - then \( c \) is defined in 7 need Phi in 1
  - \( d \) is defined in 2, 3, 4 need Phi in 6, 7
  - then \( d \) is defined in 7 need Phi in 1
  - \( i \) is defined in BB7 need Phi in BB1
Class Problem

Insert the Phi nodes

SSA Step 2 – Renaming Variables

- Use an array of stacks, one stack per global variable (VR)
- Algorithm sketch
  - For each BB b in a preorder traversal of the dominator tree
  - Generate unique names for each Phi node
  - Rewrite each operation in the BB
    - Uses of global name: current name from stack
    - Defs of global name: create and push new name
  - Fill in Phi node parameters of successor blocks
  - Recurse on b’s children in the dominator tree
  - <on exit from b> pop names generated in b from stacks

Renaming Variables – Pseudo Code

- Main function
  - For each global name i
    - counter[i] = 0
    - stack[i] = NULL
  - call Rename(BB0)

- Rename(b)
  - For each Phi node in b, x = Phi()
    - Rename x as NewName(x)
  - For each operation, x = y op z, in b
    - y, z = top(stack[y]), top(stack[z])
    - x = NewName(x)
  - For each successor of b in the CFG
    - Set appropriate Phi parameters
  - For each successor s of b in dom tree
    - Rename(s)
  - For each operation, x = y op z, in b
    - pop(stack[x])

- NewName(n)
  - i = counter[n]
  - counter[n]++
  - push ni onto stack[n]
  - return ni
Renaming – Example (Initial State)

### Initial State

<table>
<thead>
<tr>
<th>var</th>
<th>stk</th>
<th>ctr</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>i</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Renaming – Example (After BB0)

### After BB0

<table>
<thead>
<tr>
<th>var</th>
<th>stk</th>
<th>ctr</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₀</td>
<td>a₀</td>
<td>1</td>
</tr>
<tr>
<td>b₀</td>
<td>b₀</td>
<td>1</td>
</tr>
<tr>
<td>c₀</td>
<td>c₀</td>
<td>1</td>
</tr>
<tr>
<td>d₀</td>
<td>d₀</td>
<td>1</td>
</tr>
<tr>
<td>i₀</td>
<td>i₀</td>
<td>1</td>
</tr>
</tbody>
</table>

Renaming – Example (After BB1)

### After BB1

<table>
<thead>
<tr>
<th>var</th>
<th>stk</th>
<th>ctr</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>a₁</td>
<td>3</td>
</tr>
<tr>
<td>b₁</td>
<td>b₁</td>
<td>2</td>
</tr>
<tr>
<td>c₁</td>
<td>c₁</td>
<td>3</td>
</tr>
<tr>
<td>d₁</td>
<td>d₁</td>
<td>2</td>
</tr>
<tr>
<td>i₁</td>
<td>i₁</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>var</th>
<th>stk</th>
<th>ctr</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₂</td>
<td>a₂</td>
<td>3</td>
</tr>
<tr>
<td>b₂</td>
<td>b₂</td>
<td>2</td>
</tr>
<tr>
<td>c₂</td>
<td>c₂</td>
<td>3</td>
</tr>
<tr>
<td>d₂</td>
<td>d₂</td>
<td>2</td>
</tr>
<tr>
<td>i₁</td>
<td>i₁</td>
<td>2</td>
</tr>
</tbody>
</table>
Renaming – Example (After BB2)

Renaming – Example (Before BB3)

Renaming – Example (After BB3)
Renaming – Example (After BB4)

\[
\begin{align*}
\text{BB0} & : & a_0 = a_1 = \phi_i(a_0, a) \\
& & b_0 = b_1 = \phi_i(b_0, b) \\
& & c_0 = c_1 = \phi_i(c_0, c) \\
& & d_0 = d_1 = \phi_i(d_0, d) \\
& & i_0 = \phi_i(0, i) \\
\text{BB1} & : & a_2 = a_3 = \phi_i(a_2, a) \\
& & b_2 = b_3 = \phi_i(b_2, b) \\
& & c_3 = c_4 = \phi_i(c_3, c) \\
& & d_2 = d_3 = \phi_i(d_2, d) \\
\text{BB2} & : & b_3 = c_4 = d_4 = \phi_i(b_3, c_4, d_4) \\
\text{BB3} & : & a_4 = b_4 = \phi_i(a_4, b_4) \\
& & c_4 = \phi_i(c_4, c_5) \\
& & d_4 = \phi_i(d_4, d_5) \\
\text{BB4} & : & a_5 = \phi_i(a_2, a) \\
& & b_5 = \phi_i(b_2, b) \\
& & c_5 = \phi_i(c_3, c) \\
& & d_5 = \phi_i(d_2, d) \\
\text{BB5} & : & a_6 = b_6 = \phi_i(a_6, a) \\
& & c_6 = \phi_i(c_2, c_4) \\
& & d_6 = \phi_i(d_4, d_5) \\
\text{BB6} & : & a_7 = b_7 = \phi_i(a_7, a) \\
& & c_7 = \phi_i(c_5, c_5) \\
& & d_7 = \phi_i(d_5, d_5) \\
\text{BB7} & : & a_8 = b_8 = \phi_i(a_8, a) \\
& & c_8 = \phi_i(c_2, c_4) \\
& & d_8 = \phi_i(d_4, d_5)
\end{align*}
\]

Renaming – Example (After BB5)

\[
\begin{align*}
\text{BB0} & : & a_0 = a_1 = \phi_i(a_0, a) \\
& & b_0 = b_1 = \phi_i(b_0, b) \\
& & c_0 = c_1 = \phi_i(c_0, c) \\
& & d_0 = d_1 = \phi_i(d_0, d) \\
& & i_0 = \phi_i(0, i) \\
\text{BB1} & : & a_2 = a_3 = \phi_i(a_2, a) \\
& & b_2 = b_3 = \phi_i(b_2, b) \\
& & c_3 = c_4 = \phi_i(c_3, c) \\
& & d_2 = d_3 = \phi_i(d_2, d) \\
\text{BB2} & : & b_2 = c_3 = d_2 = \phi_i(b_2, c_3, d_2) \\
\text{BB3} & : & a_4 = \phi_i(a_4, a) \\
& & b_4 = \phi_i(b_2, b_3) \\
& & c_4 = \phi_i(c_3, c_5) \\
& & d_4 = \phi_i(d_2, d_5) \\
\text{BB4} & : & a_5 = \phi_i(a_2, a) \\
& & b_5 = \phi_i(b_2, b_3) \\
& & c_5 = \phi_i(c_3, c_5) \\
& & d_5 = \phi_i(d_2, d_5) \\
\text{BB5} & : & a_6 = b_6 = \phi_i(a_6, a) \\
& & c_6 = \phi_i(c_2, c_4) \\
& & d_6 = \phi_i(d_4, d_5) \\
\text{BB6} & : & a_7 = b_7 = \phi_i(a_7, a) \\
& & c_7 = \phi_i(c_5, c_5) \\
& & d_7 = \phi_i(d_5, d_5) \\
\text{BB7} & : & a_8 = b_8 = \phi_i(a_8, a) \\
& & c_8 = \phi_i(c_2, c_4) \\
& & d_8 = \phi_i(d_4, d_5)
\end{align*}
\]

Renaming – Example (After BB6)

\[
\begin{align*}
\text{BB0} & : & a_0 = a_1 = \phi_i(a_0, a) \\
& & b_0 = b_1 = \phi_i(b_0, b) \\
& & c_0 = c_1 = \phi_i(c_0, c) \\
& & d_0 = d_1 = \phi_i(d_0, d) \\
& & i_0 = \phi_i(0, i) \\
\text{BB1} & : & a_2 = a_3 = \phi_i(a_2, a) \\
& & b_2 = b_3 = \phi_i(b_2, b) \\
& & c_3 = c_4 = \phi_i(c_3, c) \\
& & d_2 = d_3 = \phi_i(d_2, d) \\
\text{BB2} & : & b_0 = c_3 = d_2 = \phi_i(b_0, c_3, d_2) \\
\text{BB3} & : & a_4 = \phi_i(a_4, a) \\
& & b_4 = \phi_i(b_2, b_3) \\
& & c_4 = \phi_i(c_3, c_5) \\
& & d_4 = \phi_i(d_2, d_5) \\
\text{BB4} & : & a_5 = \phi_i(a_2, a) \\
& & b_5 = \phi_i(b_2, b_3) \\
& & c_5 = \phi_i(c_3, c_5) \\
& & d_5 = \phi_i(d_2, d_5) \\
\text{BB5} & : & a_6 = b_6 = \phi_i(a_6, a) \\
& & c_6 = \phi_i(c_2, c_4) \\
& & d_6 = \phi_i(d_4, d_5) \\
\text{BB6} & : & a_7 = b_7 = \phi_i(a_7, a) \\
& & c_7 = \phi_i(c_5, c_5) \\
& & d_7 = \phi_i(d_5, d_5) \\
\text{BB7} & : & a_8 = b_8 = \phi_i(a_8, a) \\
& & c_8 = \phi_i(c_2, c_4) \\
& & d_8 = \phi_i(d_4, d_5)
\end{align*}
\]
Class Problem

Rename the variables so this code is in SSA form