CS738: Advanced Compiler Optimizations

SSA Continued

Amey Karkare

`karkare@cse.iitk.ac.in`

`http://www.cse.iitk.ac.in/~karkare/cs738`
Department of CSE, IIT Kanpur

Agenda

► Properties of SSA
► SSA to Executable
► SSA for Optimizations

Complexity of Construction

► $R = \max(N, E, A, M)$
► N: nodes, E: edges in flow graph
► A: number of assignments
► M: number of uses of variables
► Computation of DF: $O(R^2)$
► Computation of SSA: $O(R^3)$
► In practice, worst case is rare.
► Practical complexity: $O(R)$
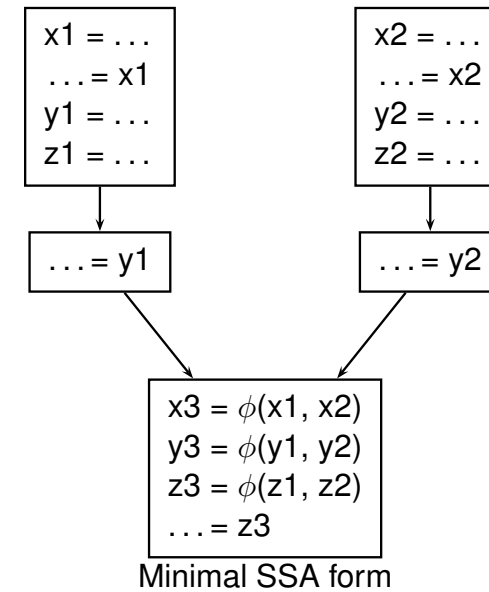
Linear Time Algorithm for ϕ-functions

► By Sreedhar and Gao, in POPL'95
► Uses a new data structure called DJ-graph
► Linear time is achieved by careful ordering of nodes in the DJ-graph
► DF for a node is computed only once an reused later if required.

Variants of SSA Form: Simple Example

Original Program
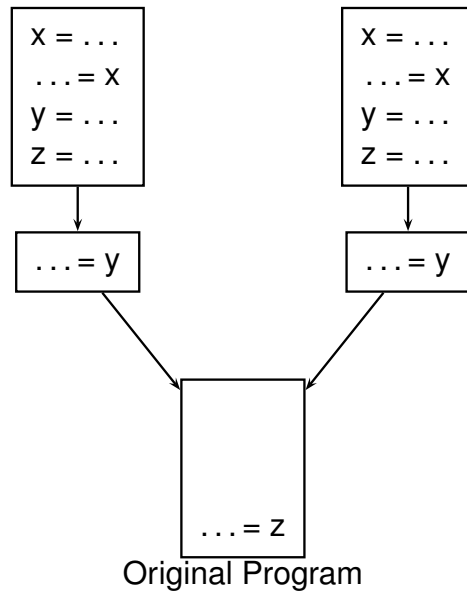
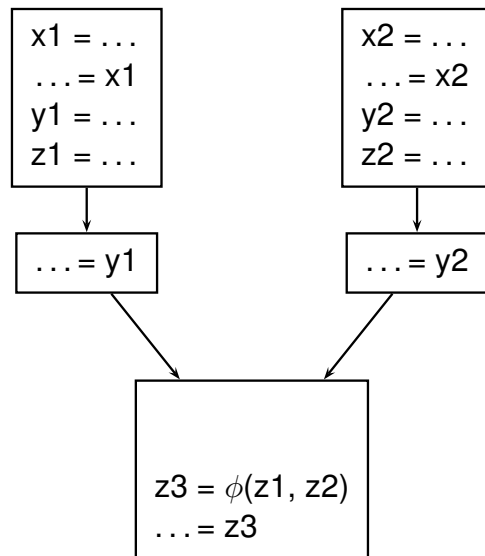## Variants of SSA Form: Simple Example

Minimal SSA form

Variants of SSA Form

- ▶ Minimal SSA still contains extraneous ϕ-functions
    - ▶ Inserts some ϕ-functions where they are dead
    - ▶ Would like to avoid inserting them
- ▶ Pruned SSA
- ▶ Semi-Pruned SSA

Pruned SSA

- ▶ Only insert ϕ-functions where their value is live
- ▶ Inserts fewer ϕ-functions
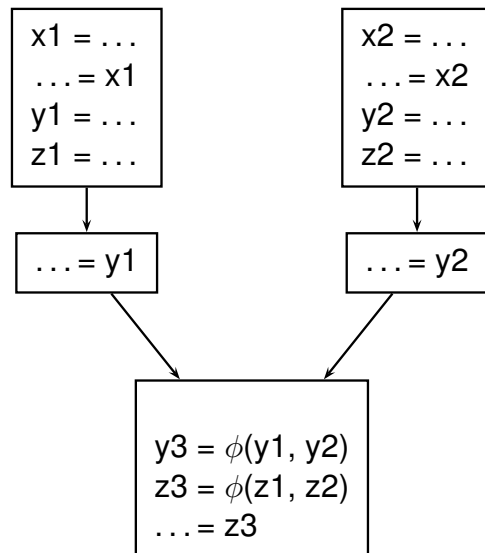- ▶ Costs more to do
- ▶ Requires global Live variable analysis

Variants of SSA Form: Pruned SSA Example

```
x1 = ...          x2 = ...
... = x1          ... = x2
y1 = ...          y2 = ...
z1 = ...          z2 = ...

... = y1          ... = y2

z3 = φ(z1, z2)
... = z3
```

Semi-Pruned SSA Form

- Discard names used in only one block
- Total number of ϕ-functions between minimal and pruned SSA
- Needs only local Live information
- Non-locals can be computed without iteration or elimination

Variants of SSA Form: Semi-pruned SSA Example

```
x1 = ...          x2 = ...
... = x1          ... = x2
y1 = ...          y2 = ...
z1 = ...          z2 = ...

... = y1          ... = y2

y3 = φ(y1, y2)
z3 = φ(z1, z2)
... = z3
```

Computing Non-locals
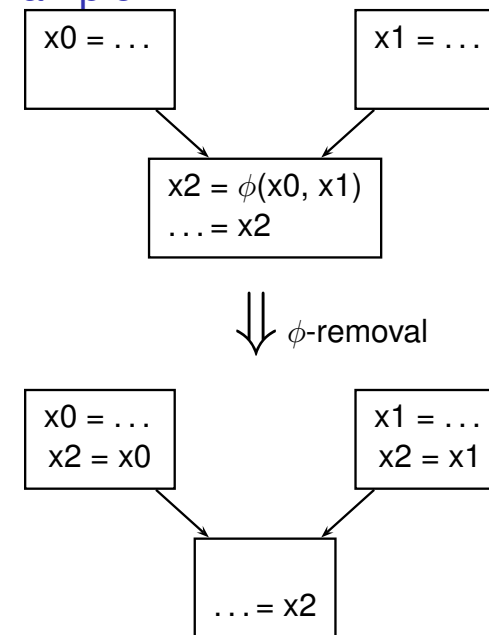
```
foreach block B {
    defined = {}
    foreach instruction v = x op y {
        if x not in defined
            non-locals = non-locals ∪ {x}
        if y not in defined
            non-locals = non-locals ∪ {y}
        defined = defined ∪ {v}
    }
}
```
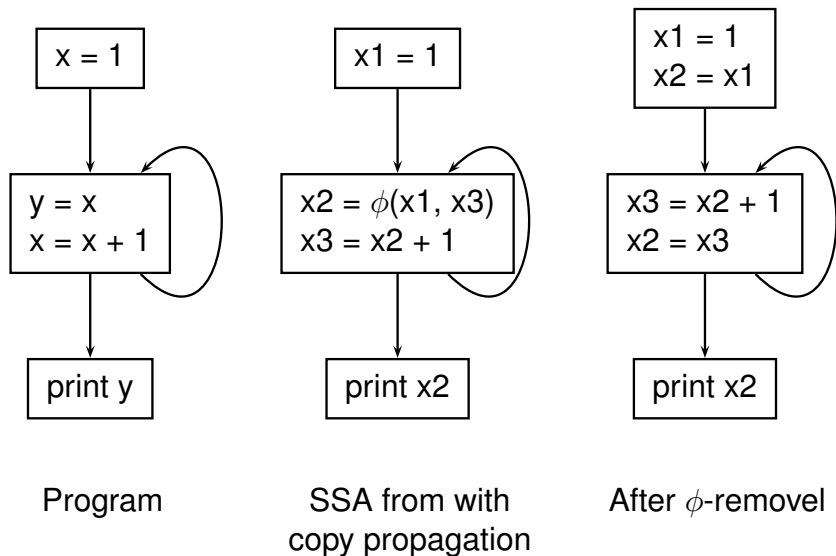
SSA to Executable

- ► At some point, we need executable code
  - ► Need to fix up the ϕ-function
- ► Basic idea
  - ► Insert copies in predecessors to mimick ϕ-function
  - ► Simple algorithm
    - ► Works in most cases, but not always
  - ► Adds lots of copies
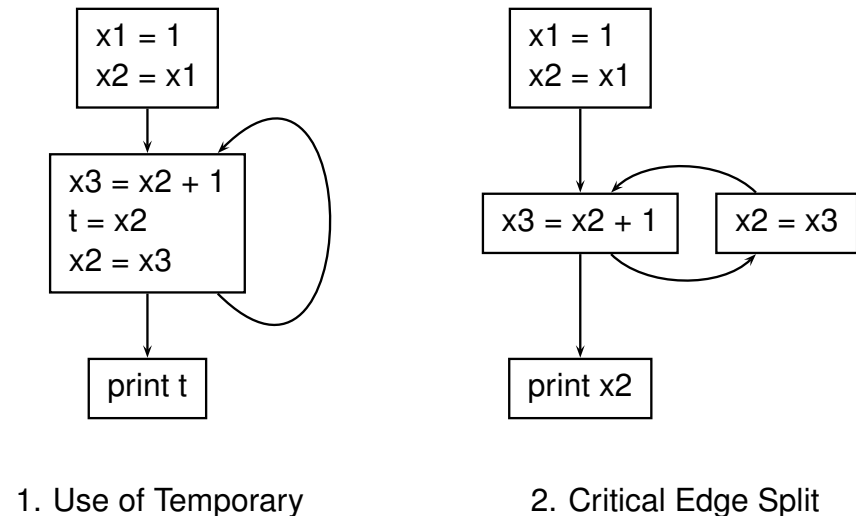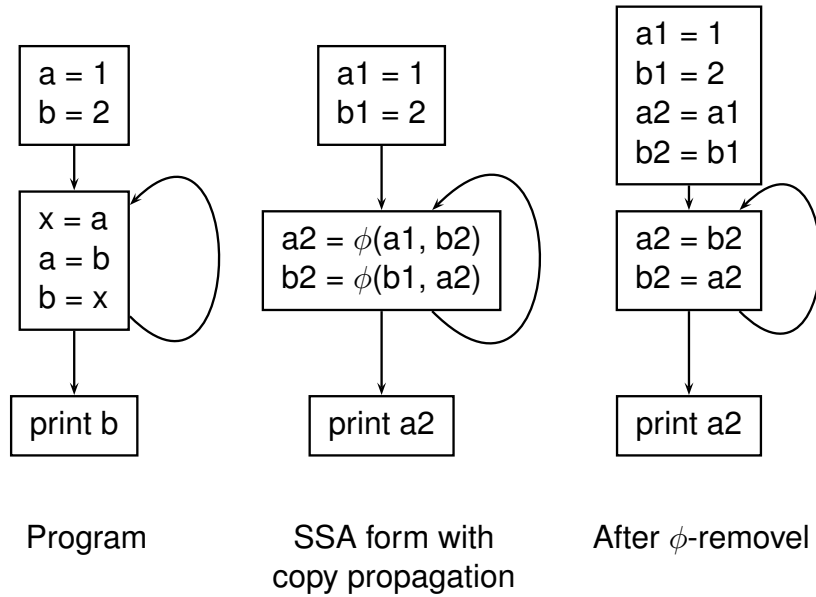   - ► Many of them will be optimized by later passes

ϕ-removal: Example

Lost Copy Problem

Program   SSA from with copy propagation   After ϕ-removel

Lost Copy Problem: Solutions

1. Use of Temporary   2. Critical Edge Split

Swap Problem

a = 1	a1 = 1	a1 = 1
b = 2	b1 = 2	b1 = 2
		a2 = a1
		b2 = b1

x = a
a = b
b = x

a2 = ϕ(a1, b2)
b2 = ϕ(b1, a2)

a2 = b2
b2 = a2

print b

print a2

print a2

Program   SSA form with   After ϕ-removel
           copy propagation

Swap Problem: Solution

► Fix requires compiler to detect and break dependency from output of one ϕ-function to input of another ϕ-function.

► May require temporary if cyclic dependency exists.

SSA Form for Optimizations

► SSA form can improve and/or speed up many analyses and optimizations
   ► (Conditional) Constant propagation
   ► Dead code elimination
   ► Value numbering
   ► PRE
   ► Loop Invariant Code Motion
   ► Strength Reduction