Program Analysis

https://www.cse.iitb.ac.in/~karkare/cs618/

Static Single Assignment (SSA)

Amey Karkare
Dept of Computer Science and Engg
IIT Kanpur
Visiting IIT Bombay
karkare@cse.iitk.ac.in
karkare@cse.iitb.ac.in

SSA Form

- Developed by Ron Cytron, Jeanne Ferrante, Barry K. Rosen, Mark N. Wegman, and F. Kenneth Zadeck,
  - in 1980s while at IBM.
- Static Single Assignment – A variable is assigned only once in program text
  - May be assigned multiple times if program is executed

2

SSA Form

- Intermediate representation
- Sparse representation
  - Definitions sites are directly associated with use sites
- Advantage
  - Directly access points where relevant data flow information is avaliable

3

SSA Form

- In SSA Form
  - Each variable has exactly one definition
    - Each use of a variable is reached by exactly one definition
  - Control flow like traditional programs
  - Some magic is needed at join nodes

4

SSA Form: Examples

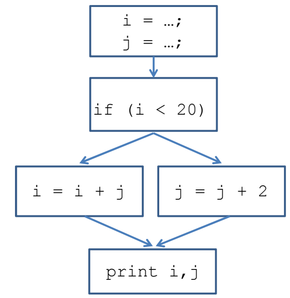
```
i = 0;            i1 = 0;
…                 …
i = i + 1;   ➡    i2 = i1 + 1;
…                 …
j = i * 5;        j1 = i2 + 1;
…                 …
```

5

SSA Form: Examples

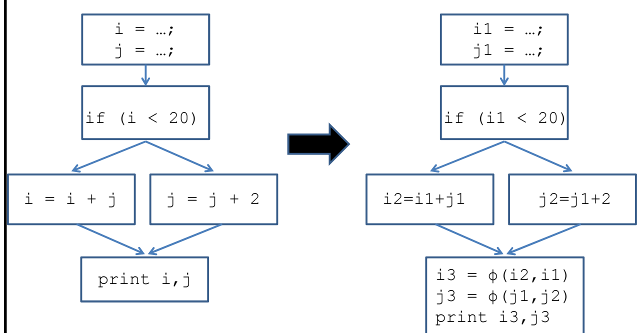
```
i = …;
j = …;
if (i < 20)
  i = i + j;
else
  j = j + 2;

print i,j;
```



6

SSA Form: Examples



7

SSA Form: Examples

```
i = …;              i1 = …;
j = …;              j1 = …;
if (i < 20)         if (i1 < 20)
  i = i + j;   ➡      i2 = i1 + j1;
else                else
  j = j + 2;          j2 = j1 + 2;
                    i3 = ϕ(i2, i1);
print i,j;          j3 = ϕ(j1, j2);
                    print i3, j3;
```

8

The “magic” : φ-function

- φ is used for selection
  - One out of multiple values at join nodes
- Not every join node needs a φ
  - Needed only if multiple definitions reach the node

9

But what does φ operation mean in a machine code?

- φ is a conceptual entity
- No direct translation to machine code
  - typically mimicked using “copy” in predecessors
  - Inefficient
  - Practically, the inefficiency is compensated by dead code elimination and register allocation passes

10

φ Properties

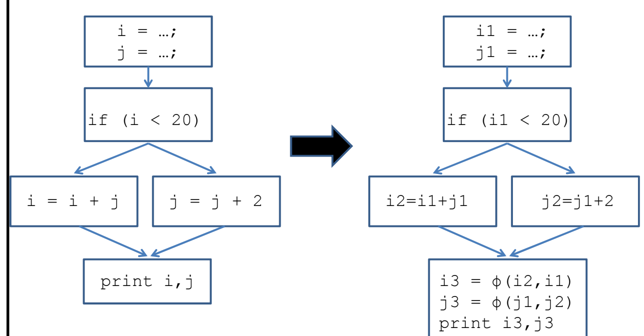
- Placed only at the entry of a join node
- Multiple φ-functions could be placed
  - for multiple variables
  - all such φ functions execute concurrently
- n-ary φ function at n-way join node

$$xm = \phi(x1, x2, \dots, xi, \dots, xn)$$

- xm gets the value of i-th argument xi if control enters through i-th edge
  - Ordering of edges is improtant

11

SSA Form: Example (revisit)



12

Construction of SSA Form

Assumptions

- Only scalar variables
  - Structures, pointers, arrays could be handled
  - Refer to publications

Dominators

- Nodes x and y in flow graph
- x dominates y if every path from ENTRY to y go through x
  - x dom y
  - partial order?
- x strictly dominates y if x dom y and x ≠ y
  - x sdom y

Computing Dominators

$$DOM(n) = \{n\} \cup \left(\bigcap_{m \in preds(n)} DOM(m) \right)$$

Initial Conditions:

$$DOM(n_0) = \{n_0\}$$
$$\forall n \neq n_0 \; DOM(n) = N$$

N is the set of all nodes, n_0 is ENTRY

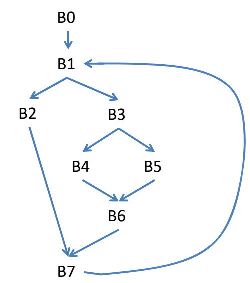
NOTE: Efficient methods exist for computing dominators

Immediate Dominators and Dominator Tree

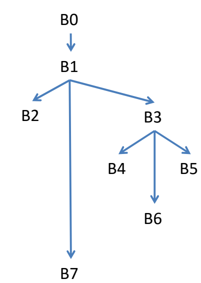
- x is immediate dominator of y if x is the *closest* strict dominator of y
  - unique, if it exists
  - denoted idom[y]
- Dominator Tree
  - A tree showing all immediate dominator relationships

17

Dominator Tree



Control Flow Graph



Dominator Tree

18

Dominance Frontier

- Dominance Frontier of x is set of all nodes y s.t.
  - x dominates a predecessor of y AND
  - x does not strictly dominate y
- Denoted DF(x)
- Why do you think DF(x) is important for any x?
  - Think about information *originated* in x

19

Computing Dominance Frontier

$$DF(x) = DF_{local}(x) \cup \bigcup_{z \in children(x)} DF_{up}(z)$$

$$DF_{local}(x) = \{y \in succ(x) | idom(y) \neq x\}$$
$$DF_{up}(z) = \{y \in DF(z) | idom(y) \neq parent(z)\}$$

* parent, children in dominator tree, succ in CFG
* parent(z) = x above

20

Iterated Dominance Frontier

- $DF^{+}(S)$: Transitive closure of Dominance frontiers on a set of nodes

$$DF(S) = \bigcup_{x \in S(x)} DF(x)$$

$$DF^{1}(S) = DF(S)$$
$$DF^{i+1}(S) = DF(S \cup DF^{i}(S))$$

21

Minimal SSA Form Construction

- Compute DF+ set for each flow graph node
- Place trivial φ-functions for each variable in the node
- Rename variables

- Why DF+? Why not only DF?

22

Inserting φ-functions

```
foreach variable v {
  S = ENTRY ∪ {n | v defined in n}
  Compute DF+(S)
  foreach n in DF+(S) {
    insert ϕ-function for v at start of n
  }
}
```

23

Renaming Variables (Pseudo Code)

- Rename from the ENTRY node recursively
  - maintain a *rename* stack of $var \rightarrow var_{version}$ mapping
- For node n
  - For each assignment (x = …) in n
    - If non-phi assignment, Rename any use of x with the Top mapping of x from the rename stack
    - Push the $x \rightarrow x_i$ on rename stack
    - i = i + 1
- For successors of n
  - Rename φ operands through succ edge index
- Recursively rename for all child nodes in the dominator tree
- For each assignment (x = …) in n
  - Pop $x \rightarrow \cdots$ from the rename stack

24