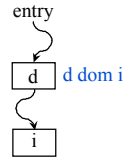


## Machinery for Placing $\phi$ -Functions

### Recall Dominators

- $d \text{ dom } i$  if all paths from entry to node  $i$  include  $d$
- $d \text{ sdom } i$  if  $d \text{ dom } i$  and  $d \neq i$



### Dominance Frontiers

- The **dominance frontier** of a node  $d$  is the set of nodes that are “just barely” not dominated by  $d$ ; i.e., the set of nodes  $n$ , such that
  - $d$  dominates a predecessor  $p$  of  $n$ , and
  - $d$  does **not** strictly dominate  $n$
- $DF(d) = \{n \mid \exists p \in \text{pred}(n), d \text{ dom } p \text{ and } d \text{ !sdom } n\}$

### Notational Convenience

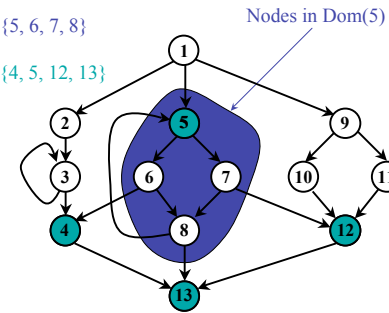
- $DF(S) = \bigcup_{s \in S} DF(s)$

## Dominance Frontier Example

$$DF(d) = \{n \mid \exists p \in \text{pred}(n), d \text{ dom } p \text{ and } d \text{ !sdom } n\}$$

$$\text{Dom}(5) = \{5, 6, 7, 8\}$$

$$DF(5) = \{4, 5, 12, 13\}$$



What's significant about the Dominance Frontier?

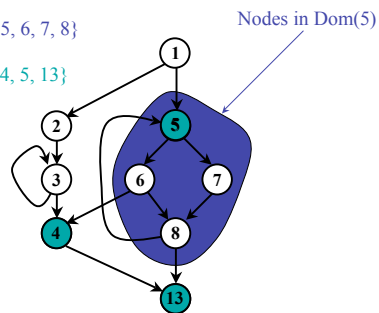
In SSA form, definitions must dominate uses

## Dominance Frontier Example II

$$DF(d) = \{n \mid \exists p \in \text{pred}(n), d \text{ dom } p \text{ and } d \text{ !sdom } n\}$$

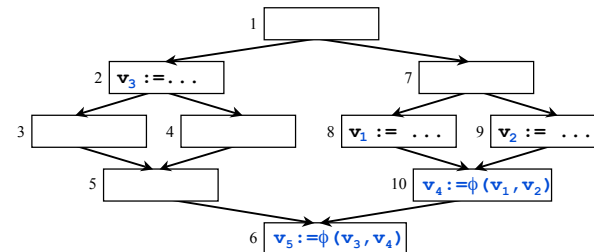
$$\text{Dom}(5) = \{5, 6, 7, 8\}$$

$$DF(5) = \{4, 5, 13\}$$



In this new graph, node 4 is the first point of convergence between the entry and node 5, so do we need a  $\phi$ -function at node 13?

## SSA Exercise



$$DF(8) = \{10\}$$

$$DF(9) = \{10\}$$

$$DF(2) = \{6\}$$

$$DF(\{8,9\}) = \{10\}$$

$$DF(10) = \{6\}$$

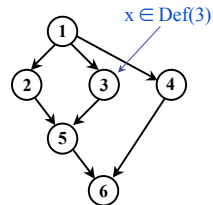
$$DF(\{2,8,9,10\}) = \{6,10\}$$

$$DF(d) = \{n \mid \exists p \in \text{pred}(n), d \text{ dom } p \text{ and } d \text{ !sdom } n\}$$

## Dominance Frontiers Revisited

Suppose that node 3 defines variable  $x$

$DF(3) = \{5\}$



Do we need to insert a  $\phi$ -function for  $x$  anywhere else?

Yes. At node 6. Why?

## Dominance Frontiers and SSA

**Let**

- $DF_1(S) = DF(S)$
- $DF_{i+1}(S) = DF(S \cup DF_i(S))$

**Iterated Dominance Frontier**

- $DF_\infty(S)$

**Theorem**

- If  $S$  is the set of CFG nodes that define variable  $v$ , then  $DF_\infty(S)$  is the set of nodes that require  $\phi$ -functions for  $v$

## Algorithm for Inserting $\phi$ -Functions

```

for each variable  $v$ 
  WorkList  $\leftarrow \emptyset$ 
  EverOnWorkList  $\leftarrow \emptyset$ 
  AlreadyHasPhiFunc  $\leftarrow \emptyset$ 
  for each node  $n$  containing an assignment to  $v$ 
    WorkList  $\leftarrow$  WorkList  $\cup \{n\}$ 
  EverOnWorkList  $\leftarrow$  WorkList
  while WorkList  $\neq \emptyset$ 
    Remove some node  $n$  from WorkList
    for each  $d \in DF(n)$ 
      if  $d \notin$  AlreadyHasPhiFunc
        Insert a  $\phi$ -function for  $v$  at  $d$ 
        AlreadyHasPhiFunc  $\leftarrow$  AlreadyHasPhiFunc  $\cup \{d\}$ 
      if  $d \notin$  EverOnWorkList
        WorkList  $\leftarrow$  WorkList  $\cup \{d\}$ 
  
```

Put all defs of  $v$  on the worklist

Insert at most one  $\phi$  function per node

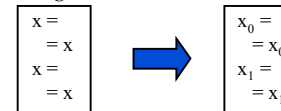
Process each node at most once

## Variable Renaming

**Basic idea**

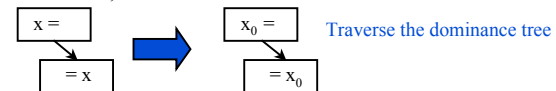
- When we see a variable on the LHS, create a new name for it
- When we see a variable on the RHS, use appropriate subscript

**Easy for straightline code**



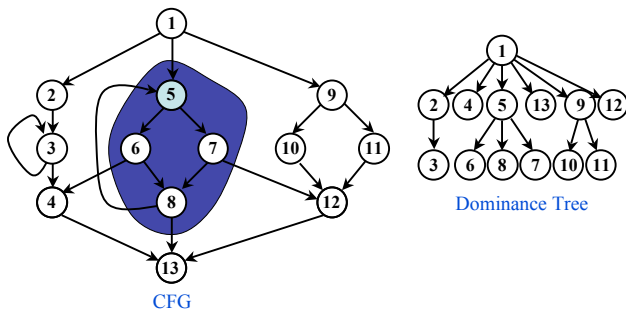
**Use a stack when there's control flow**

- For each use of  $x$ , find the definition of  $x$  that dominates it



## Dominance Tree Example

The dominance tree shows the dominance relation



CS553 Lecture

Static Single Assignment Form

15

## Variable Renaming (cont)

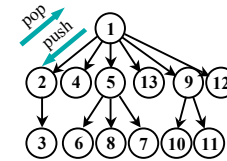
### Data Structures

- Stacks[v]  $\forall v$   
Holds the subscript of most recent definition of variable v, initially empty
- Counters[v]  $\forall v$   
Holds the current number of assignments to variable v; initially 0

### Auxiliary Routine

```

procedure GenName(variable v)
  i := Counters[v]
  push i onto Stacks[v]
  Counters[v] := i + 1
    
```



Use the Dominance Tree to remember the most recent definition of each variable

CS553 Lecture

Static Single Assignment Form

16

## Variable Renaming Algorithm

```

procedure Rename(block b)
  if b previously visited return
  for each  $\phi$ -function p in b
    GenName(LHS(p)) and replace v with  $v_i$ , where  $i = \text{Top}(\text{Stack}[v])$ 
  for each statement s in b (in order)
    for each variable  $v \in \text{RHS}(s)$ 
      replace v by  $v_i$ , where  $i = \text{Top}(\text{Stack}[v])$ 
    for each variable  $v \in \text{LHS}(s)$ 
      GenName(v) and replace v with  $v_i$ , where  $i = \text{Top}(\text{Stack}[v])$ 
  for each  $s \in \text{succ}(b)$  (in CFG)
    j  $\leftarrow$  position in s's  $\phi$ -function corresponding to block b
    for each  $\phi$ -function p in s
      replace the  $j^{\text{th}}$  operand of RHS(p) by  $v_i$ , where  $i = \text{Top}(\text{Stack}[v])$ 
  for each  $s \in \text{child}(b)$  (in DT)
    Rename(s)
  for each  $\phi$ -function or statement t in b
    for each  $v_i \in \text{LHS}(t)$ 
      Pop(Stack[v])
    
```

Call Rename(entry-node)

Recurse using Depth First Search

Unwind stack when done with this node

CS553 Lecture

Static Single Assignment Form

17

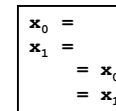
## Transformation from SSA Form

### Proposal

- Restore original variable names (*i.e.*, drop subscripts)
- Delete all  $\phi$ -functions

### Complications

- What if versions get out of order?  
(simultaneously live ranges)



### Alternative

- Perform dead code elimination (to prune  $\phi$ -functions)
- Replace  $\phi$ -functions with copies in predecessors
- Rely on register allocation coalescing to remove unnecessary copies

CS553 Lecture

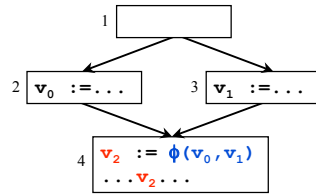
Static Single Assignment Form

18

## Backward Analyses vs. Forward Analyses

For forward data-flow analysis, at phi node apply meet function

For backward data-flow analysis?



## Static Single Information Form (SSI)

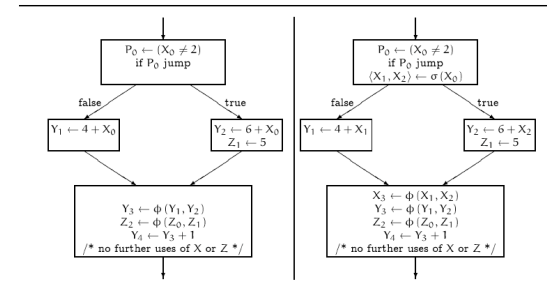


Figure 5.1: A comparison of SSA (left) and SSI (right) forms.

Ananian's Masters Thesis, 1997 MIT

## Concepts

### SSA construction

- Place phi nodes
- Variable renaming

Transformation from SSA to executable code depends on the optimizations dead-code elimination and copy propagation

Backward data-flow analyses can use SSI modification to SSA

## Next Time

### Assignments

- HW1 due

### Lecture

- Using SSA for program optimization