Static Single Assignment Form (SSA)

Unnikrishan C

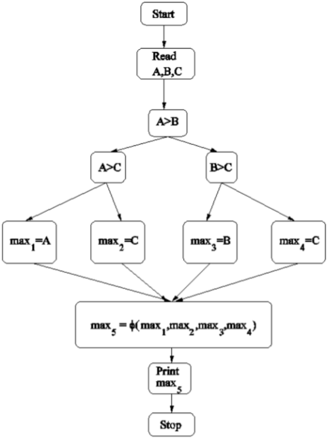
August 27, 2019

SSA Form

- A new intermediate representation.
- Incorporates **def-use** information.
- Every **variable has exactly one definition** in the program text (i.e in SSA IR).
  - This does not mean that there are no loops.
- Some compiler optimizations perform better on SSA forms.

SSA Example - program1

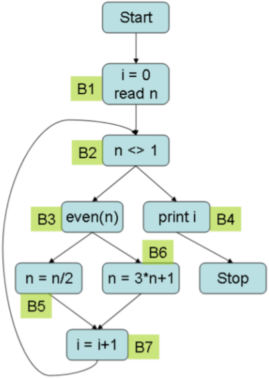
read A, B, C
if (A>B)
  if (A>C) max = A
  else max = C
else if (B>C) max = B
    else max = C
printf (max)

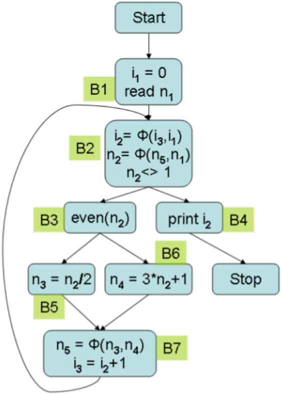


SS form- Join nodes and ϕ function

- A special merge operator, ϕ is used for selection of values in join nodes.
- The SSA form is augmented with **use-def** and **def-use** chains. to facilitate design of faster algorithms.
- Translation from SSA to machine code introduces copy operations, which may introduce some inefficiency (**will be covered in coming classes**).

SSA Example : program1 in non SSA form



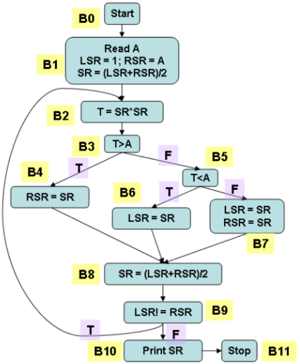


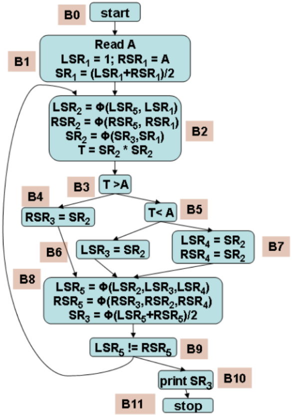
SSA Example - program2

```
{ Read A; LSR = 1; RSR = A;
  SR = (LSR+RSR)/2;
  Repeat {
       T = SR*SR;
       if (T>A) RSR = SR;
       else if (T<A) LSR = SR;
              else { LSR = SR; RSR = SR}
       SR = (LSR+RSR)/2;
  Until (LSR ≠ RSR);
  Print SR;
}
```

SSA Example : program2 in non SSA form

Just workout this example yoursel





Dominance Frontier

- If two **non-null** paths from **nodes** X and Y each having a definition of v converge at a node P, then P contains a trivial ϕ-function of the form $V_p = \phi(V_x\ ,\ V_y)$, where V_x and V_y are values coming from nodes X and Y.
- It would be wasteful to place ϕ-functions in all join nodes.
- It is possible to locate the nodes where ϕ-functions are needed.
- This is captured by the **dominance frontier**.

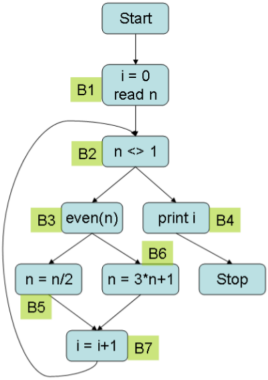
Join Set and ϕ nodes

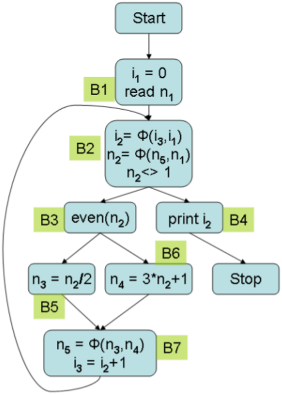
Given S: **set** of flow graph nodes, the **set** JOIN(S) is

- the set of all nodes N, such that there are **at-least two non-null paths** in the flow graph that **start at two distinct nodes** in S and converge at N.
  - The paths considered **should not have any other common nodes** apart from N.
- The **iterated join set**, $JOIN^{+}(S)$ is
  - $JOIN^{(1)}(S) = JOIN(S)$
  - $JOIN^{(i+1)}(S) = \text{JOIN}(\text{S} \cup JOIN^{(i)}\ (\text{S}))$
- If S is the set of assignment nodes for a variable v , then $JOIN^{+}(S)$ is precisely the set of flow graph nodes, where ϕ-functions are needed for v.
- $JOIN^{+}(S)$ is termed the dominance frontier (DF(S))

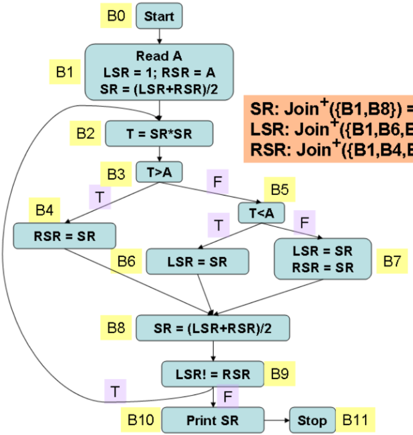
$JOIN^+$ Example-one

- Variable i: $JOIN^+\{B1, B2\} = \{B2\}$ // i modified (assigned) in B1 and B2.
- Variable n: $JOIN^+\{B1, B5, B6\} = \{B2, B7\}$ // n modified (by user-input in B1), (assigned in B5, B6).





$JOIN^+$ Example-two



SR: Join$^+$({B1,B8}) = {B2}
LSR: Join$^+$({B1,B6,B7}) = {B2,B8}
RSR: Join$^+$({B1,B4,B7}) = {B2,B8}

Dominator and Dominance Frontier

- Given two **nodes** x and y in a flow graph, x dominates y ($x \in dom(y)$) , if x appears in **all paths** from the **Start node** to y.
- The node x **strictly** dominates y , if x dominates y and $x \neq y$.
- x is the **immediate** dominator of y (denoted $idom(y)$), if x is the closest strict dominator of y.
- A **dominator tree** shows all the **immediate** dominator relationships.
- The dominance frontier of a node x, DF (x), is the set of all nodes y such that
  - x dominates a predecessor of y ($p \in preds(y)$ and $x \in dom(p)$)
  - but x does not strictly dominate y ($x \notin dom(y) \setminus \{y\}$)
- **See informal definition in next slide if got confused**.

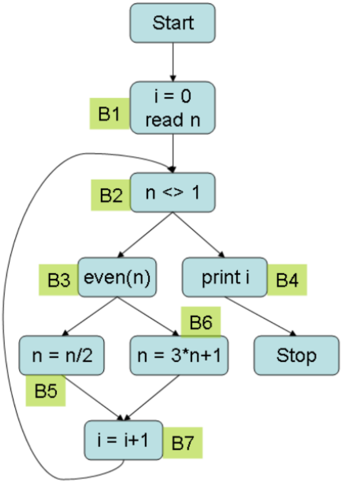
Dominance Frontier - Informal definiton

- **Informally**, DF (x) contains the **first** nodes reachable from x that **x does not dominate**, on each path leaving x.

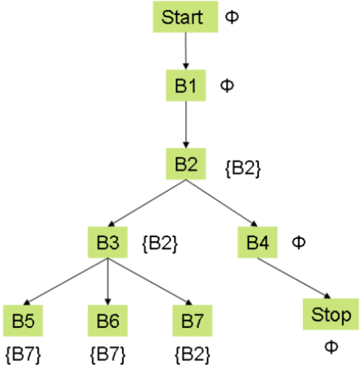
strictly

- In example 1 (**next slide**),
  - DF (B1) = ϕ since B1 dominates all nodes in the flow graph except Start and B1, and there is no path from B1 to Start or B1.
  - In the same example, DF (B2) = {B2}, since B2 dominates all nodes **except** Start, B1, and B2, and there is a path from B2 to B2 (via the back edge).
  - DF (B3)={B2}, B2 is the first node reachable from B3, which it **does not** dominate.
  - Continuing in the same example, B5, B6, and B7 **do not dominate any node** and the first reachable nodes are B7, B7, and B2 (respectively). Therefore, DF (B5) = DF (B6) = {B7} and DF (B7) = {B2}
- In example 2 (**second next slide**), B5 **dominates** B6 and B7, **but not B8**; B8 is the first reachable node from B5 that B5 **does not** dominate; therefore, DF (B5) = {B8}

DF-Example1

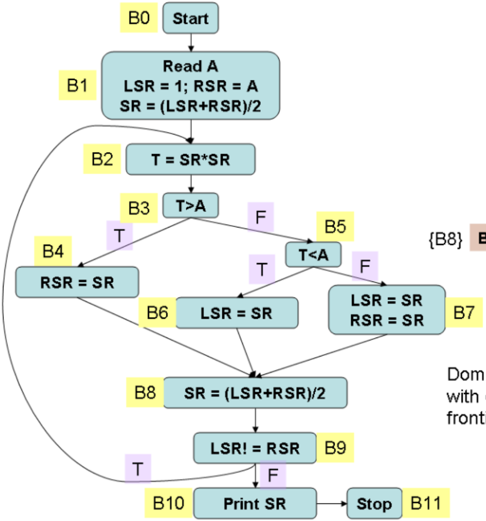


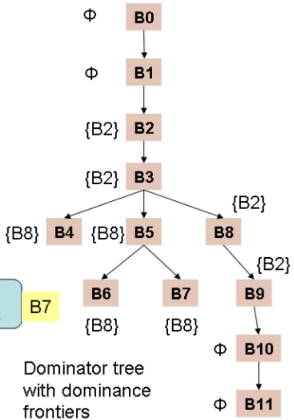
Start
i = 0
read n
B1
B2
n <> 1
B3
even(n)
print i
B4
B6
n = n/2
n = 3*n+1
Stop
B5
i = i+1
B7



Start Φ
B1 Φ
B2 {B2}
B3 {B2}
B4 Φ
B5 {B7}
B6 {B7}
B7 {B2}
Stop Φ

DF-Example2





Dominator tree with dominance frontiers