SSA: Static Single Assignment

Uses of SSA

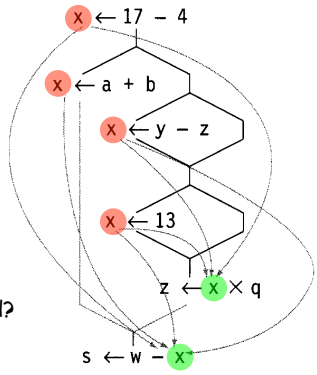
- Used in almost all modern compilers
  - gimple form in GCC (version 4 in April 2005)
  - Open 64
    - OpenUH, UPC, AMD, Loongson compiler
  - LLVM
  - Jikes RVM
  - Java HotSpot VM
  - Mono's Mini JIT compiler
  - Crankshaft for Chromium V8 JavaScript engine (Dec. 2010)
  - PyPy's JIT compiler
  - Android's Dalvik VM's JIT compiler
  - Single-assignment C (SaC)
  - Boomerang decompiler
  - ML compiler MLton (Matthew Fluet at RIT)
  - LuaJIT

History of SSA

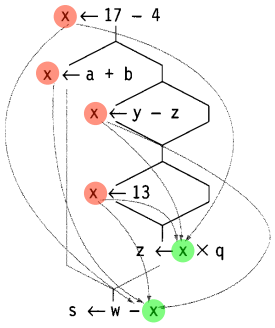
- Two IBM groups
  - Kenneth Zadeck et al. on optimization
  - Ferrante et al. on control dependence
  - info session before paper submission to POPL 1986
  - joint paper in ACM Transactions on Programming Languages and Systems in 1991
- First academic and industry implementations
  - Rice compiler, Keith Cooper
  - SGI's MIPSpro compiler, Fred Chow, Shan Sun
    - Preston Briggs' interview
    - took four years for SSAPRE in a commercial compiler
      - Kennedy et al. TOPLAS 1999

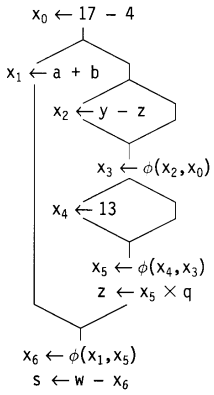
Static Single Assignment (SSA)

- Size of a def-use graph
- SSA
  - each static definition defines a new name
  - each use has a single static definition
- Meet operation
  - x <- Φ(y, z)
  - placement
- Naive SSA insertion
  - Algorithm?
  - How many Φ functions are needed?



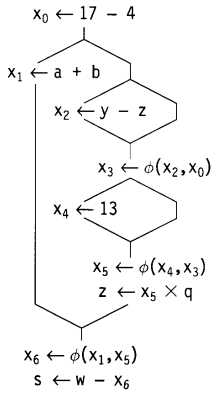
Def-Use vs. SSA





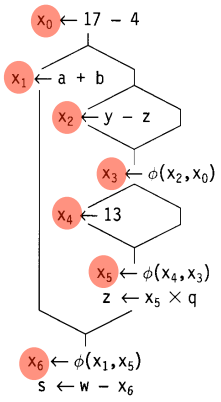
Control Flow in Data Flow

- How to identify the earliest meeting point of two values?
- Dominance frontiers DF(n)
  - a block f is in DF(n) if
    1. n dominates a predecessor of f
    2. n does not strictly dominate f
  - implications
    - f is a join point
    - one of the predecessors of f is dominated by n



Dominators

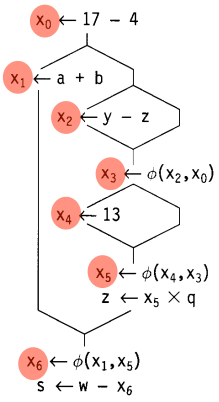
- Dominator analysis
  - Dom(n) definition
  - Data flow equation
  - Initialization
  - Convergence



Computing Dominance Frontiers

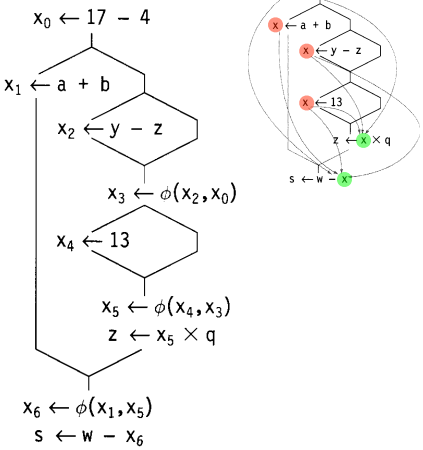
- Dom to DF
  - backward alg. [EAC, Fig. 9.10]
  - forward alg., linear time [AK, Fig. 4.9]

```
for each join j
  for pred p
    for all nodes n in dom tree
          from p up till IDOM(j)
      j is in DF(n)
```



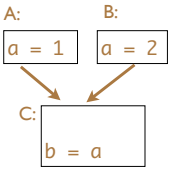
SSA Algorithm

1. CFG
2. compute Dom
3. compute DF
4. insert phi
5. rename
6. reaching def
7. "destruct" SSA



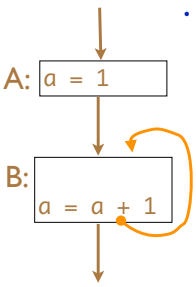
Example 1

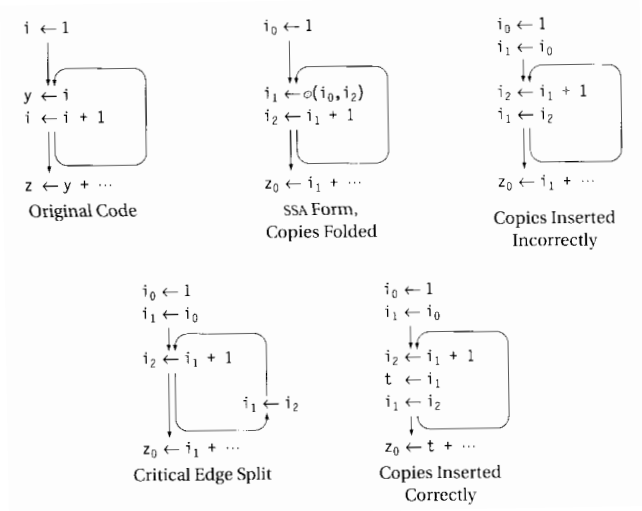
- IDOM DF phi renaming reach destruction

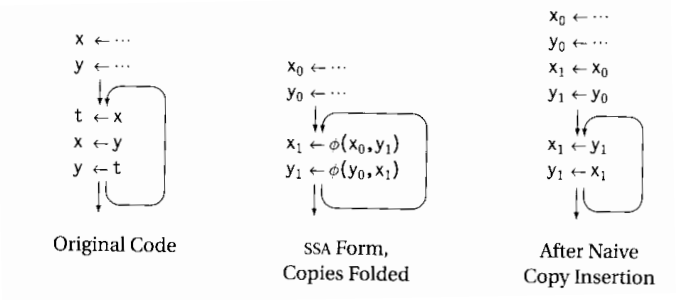


Example 2

- IDOM DF phi renaming reach destruction






x ← ···
y ← ···
t ← x
x ← y
y ← t
Original Code
x0 ← ···
y0 ← ···
x1 ← φ(x0,y1)
y1 ← φ(y0,x1)
SSA Form,
Copies Folded
x0 ← ···
y0 ← ···
x1 ← x0
y1 ← y0
x1 ← y1
y1 ← x1
After Naive
Copy Insertion