CSE 401 - Compilers
Dataflow and SSA

Nathaniel Mote

Winter 2015

FYI

- Compiler extensions due tonight
- Project report due Saturday night
- Final exam Tuesday at 2:30
  - Topic list is live
- Then you're done!

Common Dataflow Problems

- Common Subexpression Elimination
  - Solved by computing available expressions with a dataflow problem
- Live Variable Analysis – used for:
  - Register Allocation
  - Eliminating useless stores
  - Detecting uses of uninitialized variables
  - Avoid placing useless Φ functions in SSA construction
- Reaching Definitions
  - Basically computes what SSA computes
- Etc... You should know the basics for the final!

Example from Au11 Final

In most production compilers, optimization is done using intermediate code that consists of simple 3 - address instructions like the following:

r5 = r3 + r7

r6 = r5 * 8

Some of the instructions in the intermediate code may be dead code because the result s of the instruction s are never used later in the program. An important optimization in compilers is dead - code elimination , where we remove such instructions to save space and execution time.

What data flow analysis should be used to discover which instructions are dead code? Describe the appropriate data flow problem to use and explain how to use its results to identify dead instruction(s).

A: Live variable analysis. If the destination is not live, the statement can be deleted

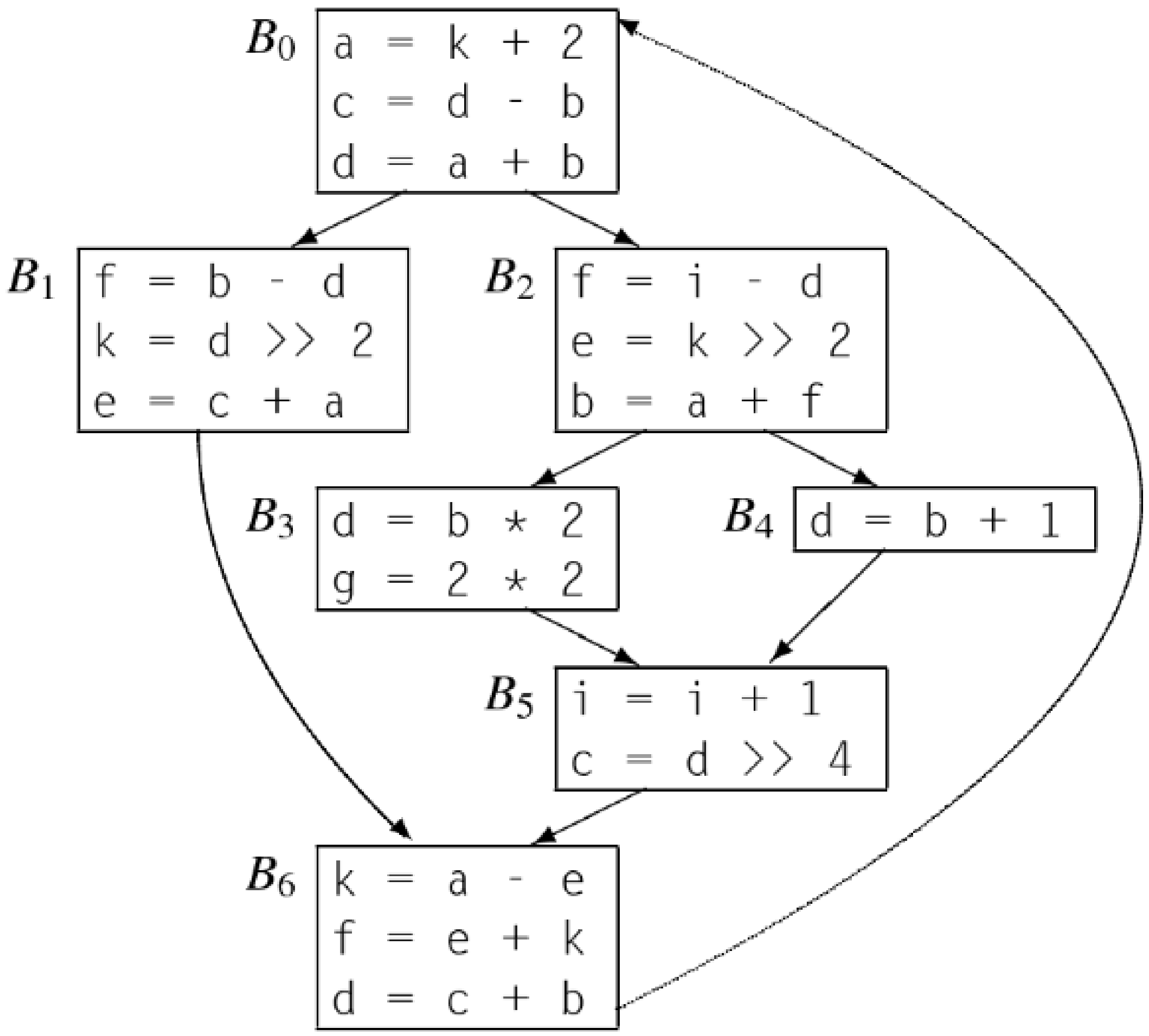
Review of Live Variable Analysis

- Two equations:
  - $in[b] = use[b] \cup (out[b] – def [b])$
  - $out[b] = \cup_{s \in succ [b]} in[s]$
  - Where use[a] and def[a] are functions of a single block
- This is how a dataflow problem is defined:
  - A system of simultaneous equations that must be solved iteratively
  - Sometimes propagates information down, sometimes up

Common Subexpression Elimination

- Compute available expressions:
  - $\mathrm{AVAIL}(b) = \bigcap_{x \in \mathrm{preds}(b)} (\mathrm{DEF}(x) \cup (\mathrm{AVAIL}(x) \cap \mathrm{NKILL}(x)))$
  - Again, where DEF(a) and NKILL(a) are functions that only need to inspect a single block.

SSA



Translate this code to SSA form (Cooper & Torczon 9.6)

Reminder: The dominance frontier of a node x is the set of all nodes w such that

- x dominates a predecessor of w
- but x does not strictly dominate w

Final Exam Topics

http://courses.cs.washington.edu/courses/cse401/15wi/exams/final-topics.html