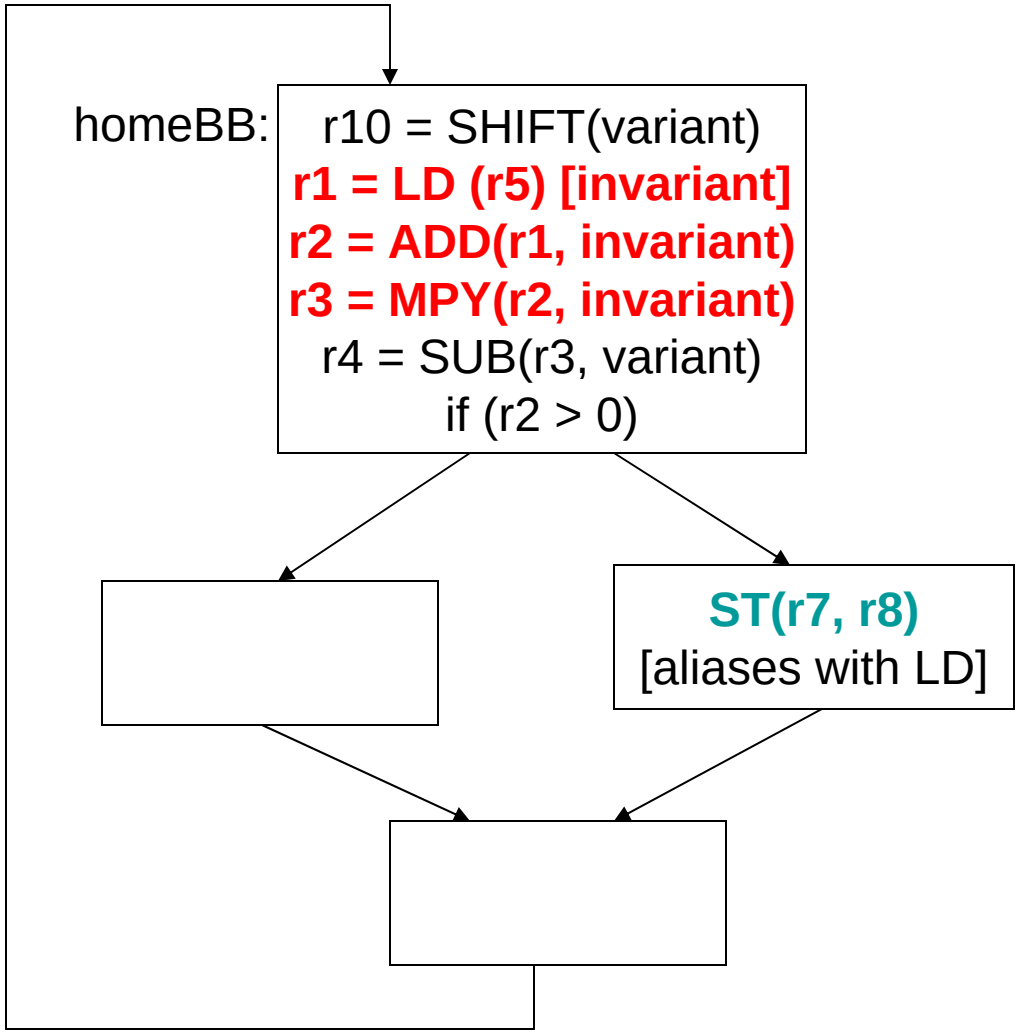
Conceptual LICM Transformation for HW2



Initial code

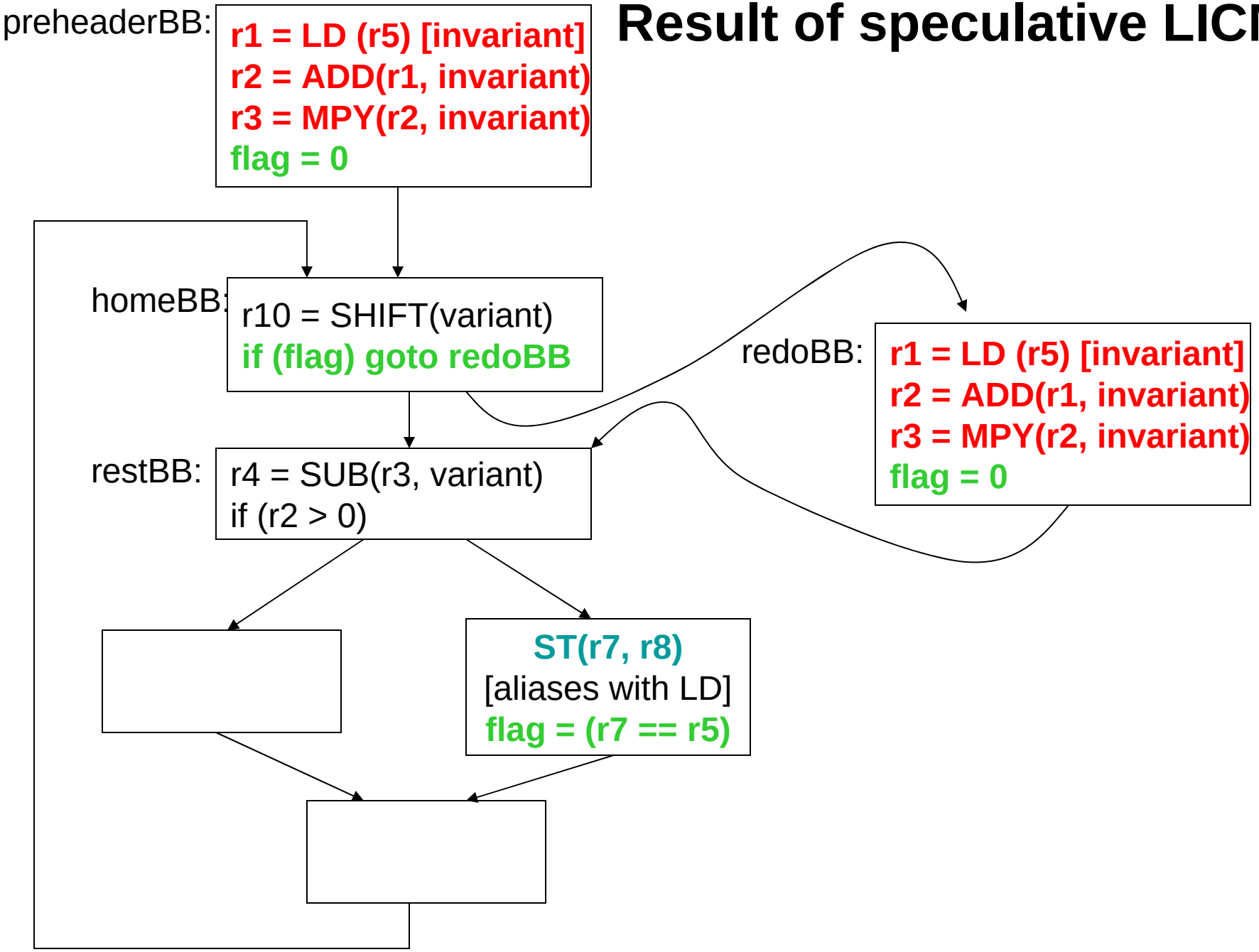
invariant load that cannot be removed due to aliasing store.

Check the memory dependence profile and see that the alias count between LD and ST is small

Hoist out the LD

LICM will then take over and hoist out any instructions that become invariant after the LD is removed. In this case, the ADD and MPY will also be moved out

Result of speculative LICM



Notes on Transformation

- Split home BB of hoisted LD
  - Everything above LD stays put in homeBB
  - Everything after LD gets put in new BB called restBB
  - new branch added to end of homeBB, that tests if flag=1 and branches to redoBB if true, and restBB if false
- redoBB
  - Place copy of hoisted LD, and copy of any other hoisted invariant instructions that directly or indirectly use the result of the LD. Note it's important to populate this list correctly. Look at the set of instrs that are hoisted to identify those that became invariant because of the LD by examining the use lists
  - Note - invariant uses of the LD will automatically get hoisted by LICM, you don't need to do anything to make this happen
  - Note2 – even if these uses that become invariant occur in other BBs, you can redo them all in redoBB (think about this one!)
  - Clear flag at the end of redoBB
- Preheader
  - Set all flags to 0 at end of preheader. Note, each LD that you hoist should have its own flag variable