Compiler Construction

Lab 5

Nov 9, FSU/CS

Optimization

Implement any optimization from the list that transforms IR to IR (but does not change AST):

- Constant propagation for integers and Booleans (`--cp`):
  - if there exist an assignment $v_i := v_j$ `op` v_k, where v_j / v_k get constants `a` / `b` (and nothing else), then precompute `c = a op b` and replace $v_i := v_j$ `op` v_k by $v_i :=$ `c`
- Simplify CFG (`--simpcfg`):
  - if a basic block has a single predecessor, merge it with the predecessor
  - If a basic block has a conditional branch instruction, and its first argument is always evaluated to true (resp. false), then replace the conditional branch by an unconditional one, and remove one of the basic blocks
- Function inliner (`--inline`):
  - Embed the CFG of all functions into their calling context (thus, potentially making the CFG cyclic)

Optimization (cont.)

Make sure that:

- All basic blocks/instructions that are removed by your optimizations are also removed from your data structures.

Write a command-line support for your optimization:

- Optimizations are by default disabled
- You can enable the optimization using the corresponding option (e.g., `--cp`)
- Your compiler should print the IR/CFG to command line/dot-file after the optimizations

- If you submit bonus lab 4*, make sure that your SSA/SMT printing works after the optimizations too – then, you will have an ability to check equivalence of the IR before/after optimizations for free!