Hints for Project Step 7

Here are a couple of ideas to help you get started with project step 7. The description of the step on the project page tells you to do liveness on a per-basic-block basis. Here, I will describe an approach that calculates liveness on a per-statement basis. You can choose either approach.

1. **Control flow graph.** In class, we discussed how to build a control flow graph for any piece of code, including code that uses arbitrary jumps and branches. Building a CFG for Micro code is easier, as the only control flow is `if` statements and `for` statements. Your current IR probably represents the three address code as a linked list of IR instructions. You can use this linked list as a basis for a CFG—each statement represents one node in the CFG. The only modifications you need to make to this are to add additional predecessor and successor information to nodes that need it.

    - **Unconditional jumps** (like at the end of for loops) have only one successor: the target of the jump statement. When you see an unconditional jump, add the target of the jump statement as a successor of the jump, and the jump statement as a *predecessor* of the target.
    - **Conditional jumps** have two successors: the "fall-through" target, which is already a successor in the original linked list, and the "taken" target. Add the branch as a predecessor of the taken target, and the taken target as an additional successor of the branch.

2. **Liveness analysis.** Once you've built the CFG, liveness analysis is easy. Compute the GEN and KILL sets for each statement. Create IN and OUT sets for each CFG node (program statement), and initialize them to empty. Then iterate over every node, updating IN and OUT according to the dataflow equations we discussed in class. You need to iterate until convergence (there are many ways to do this, but in general, when a statement's liveness information changes, you need to update its predecessors).

3. **Constructing basic blocks.** A basic block is a set of statements that starts with a "leader" statement and continues until the next leader. In this case, leader statements are any statements that a) have predecessors other than the immediately preceding instruction or b) have successors other than the immediately following instruction.

4. **Register allocation.** You may use either bottom-up register allocation, as discussed in class, or graph coloring-based register allocation.