An accurate estimate of the Worst Case Execution Time (WCET) of a task is essential in hard real-time system scheduling. Often, a function or loop has multiple execution paths, due to conditional branch instructions. Based on the nature of the comparisons, some paths may be infeasible. Eliminating infeasible paths from consideration can give a tighter bound on the WCET.

We created a tool, a Retargetable Assembly Level Hierarchical Organizer (RALPHO). Its goal is to glean control-flow facts to pass to our timing analyzer. These facts are used to create function instances and paths. RALPHO is easy to retarget and replaces the need for us to rely on a compiler to provide control-flow information.

First, read assembly definition file to understand the syntax of the target instruction set. Next, read the assembly source:
- Find functions and labels (branch targets)
- Partition instructions into basic blocks
- Find predecessors, successors, dominators of blocks.
- Find loops, calculate number of iterations, and how the loops nest. Identify which blocks comprise each loop.

For each block, RALPHO computes the presence of branch prediction or branch correlation constraints.
- Block A makes block B unknown.
- Block A makes block B fall through.
- Block A makes block B branch.
- If A falls through, then B is unknown.
- If A branches, then B is unknown.
- If A falls through, then B will branch.
- If A branches, then B will branch.
- If A falls through, then B will fall through.
- If A branches, then B will fall through.

We consider branches in pairs. If two branches in the same function compare the same register, we perform correlation analysis. For each pair, there are 4 possibilities: both taken, both fall through, first only taken, second only taken. We determine which possibilities could occur.
- There are $6 \times 3 = 18$ possible cases of branch correlation, due to 6 relational operators for each comparison, and the 3 ways that the constants compare to each other.
- The infeasible paths can be selected by scanning through the appropriate possibilities. For example, if the first branch is $a >$, the second branch is $a <$, and the two numbers being compared to the register are equal then (True, False) and (False, True) are the only feasible paths.

If a block updates a register that is later used in a branch instruction, we determine if the effect makes the branch taken or not.

Above are the results of a sample test whose accuracy improved with infeasible path detection. Alongside each test is the C code and assembly code used to test the timing analyzer.

We plan to test our approach with established WCET benchmarks. Further refinements to our approach include the following: dealing with branches that compare more than one register, and determining on which loop iterations a branch should be taken. We also plan to incorporate RALPHO’s infeasible path detection into our related work on stochastic analysis to improve its accuracy.

The work was supported in part by grants from the Furman Advantage and Francis M. Hipp Research Fellowship and the South Carolina Independent Colleges and Universities Research Program.