What We’ve Accomplished
We’ve implemented a conservative procedure analysis pass in MachSUIF and used this to generate annotations that are printed in the C code from a modified m2c MachSUIF pass. These annotations are used by gcc to optimize redundant calls of the procedures.

Results on synthetic micro-benchmarks show considerable speedups versus code with unoptimized redundant calls.

The analysis is currently conservative in that it treats all procedure calls as (possibly) having side effects. We are currently working on building a call graph of compiled procedures to recognize calls to other procedures without side-effects (since the call graph may have cycles, we have to solve it as we would a data flow problem on basic blocks.)

Changes to the Project
Because gcc implements optimization of redundant calls annotated procedures, we have been able to focus on automatically generating the annotations for the procedures (which gcc does not do).

Because gcc provides both “pure” and “const” annotations (where pure functions can read global variables), we have expanded our analysis to make this distinction and be able to optimize calls in more situations.

Because building a call graph in MachSUIF has proved difficult, we have started using SUIF to implement the interprocedural analysis.

Meeting Our Milestone
We’ve both accomplished more and less than what we set out to do for our milestone. We’ve already dealt with pointer procedure arguments (which we had later on our schedule), but we have not yet fully dealt with procedure calls. We are pleased with our progress so far, and consider our milestone met.

Revised Schedule
This is our current schedule for the remaining weeks of the project:

Week 4 (Nov 17 – Nov 23)
- Write milestone report
- Port existing analysis from MachSUIF to SUIF

Week 5 (Nov 24 – Nov 30)
- Start interprocedural analysis (call graph generation)

Week 6 (Dec 1 – Dec 7)
- Finish interprocedural analysis
- Write project summary