LSU EE 4720 **Homework 6** Solution **Due: 20 October 2010**

Links in this assignment are clickable in Adobe Reader. For the questions below refer to the gcc 4.1.2 manual, available via `http://gcc.gnu.org/onlinedocs/gcc-4.1.2/gcc/`.

Problem 1: Read the introductory text to the optimization options page, 3.10, in the GCC 4.1.2 manual, and familiarize yourself with your Web browser's search function so that you can search the rest of the page. Answer the following questions.

(*a*) When optimizing gcc tries to fill branch delay slots. What option can be used to tell gcc not to fill delay slots, without affecting other optimizations? What option can be used to control how much effort gcc makes to fill delay slots?

The option for not filling delay slots is `-fno-delayed-branch`. At least two options can be used for effort. They are `--param max-delay-slot-insn-search=N` and `--param max-delay-slot-live-search=N`, where `N` is a number of instructions, a higher number means more effort. These options indicate how many instructions to look at in a search to fill a branch delay slot.

(*b*) A reason given in class for scheduling code was to avoid stalls due to a lack of bypass paths. What reason is given in the description of the -fschedule-insn option?

The documentation attributes stalls to slow floating point instructions (which most of them are) and for memory instructions. The assumption is that bypass paths that would be frequently used are already there. (That is, there are no "missing" bypass paths to schedule around, or more precisely the missing bypass paths are used so infrequently they are not worth mentioning.)

Problem 2: The POWER and PowerPC ISAs have alot in common, but each has instructions the other lacks. Show the gcc command line switch to compile for both, start looking in section 3.17, Hardware Models and Configurations.

To compile for both use two switches, each eliminating instructions limited to one of the ISAs: `-mcpu=common` one could also use the pair of switches: `-mno-power -mno-powerpc`.

Problem 3: Read the following blog post about the use of profiling in the build of the Firefox Web browser:
`http://blog.mozilla.com/tglek/2010/04/12/squeezing-every-last-bit-of-performance-out-of-the-linux-toolchain/`.
The post compares the results of profiling optimizations provided by gcc to those obtained using other tools for optimization.

(*a*) As described in the blog post, what was the training data used for profiling?

The training data was just the "Quit" command. That is, Firefox was started and then exited. One might expect they would profile Firefox rendering some Ajaxy Web page, but they didn't.

(*b*) Suppose that a Web page with a 5000-row table performs just as sluggishly with the profile-optimized gcc build described in the blog post (firefox.static.pgo) as the ordinary Firefox build (firefox.stock). Provide a possible reason for this, and a solution.

The default startup code either did not have table, or did not have a table elaborate enough to guide optimization. The solution would be to profile on the 5000-row table.

Problem 4: SPEC recently ended a call for possible programs for their next CPU suite, cpuv6. Read the page describing the call: `http://www.spec.org/cpuv6/`.

(*a*) There is a section entitled "Criteria SPEC considers important for the next CPU benchmark suite." Evaluate the suitability of the `pi.c` program used in class based on each of these criteria.

The pi program would not be suitable. Here is how the pi program meets each of the criteria:

A good benchmark candidate is:

- Used by real users

  :-(Criterion not met: No "real" user because no one uses that program to compute π.

- Compute bound, or can have its compute bound portion excerpted

  :-) Criterion met: The program is compute bound.

- Portable or can be ported to multiple hardware architectures and operating systems with reasonable effort

  :-) Criterion met: Written in standard C, easy to port.

- Represents the state of the art for the given field

  :-(Criterion not met: Much better ways to compute π.

- Derived from a representative application

  :-(Criterion not met: Nope.

- Capable of solving problems of varying sizes. SPEC CPU2006 used 3 workloads, in various capacities, for its benchmarks.

  :-) Criterion met: Can vary the number of iterations.

- Reasonably predictable as to its code path. For example, minor differences in floating point accuracy across platforms should not cause the program/application to do wildly different work on those platforms.

  :-(Criterion not met: Not sure, but given the answers above not worth it to find out.