Lab 6

By Stephen Calabrese and Matt Bague

1. If you are building a processor and have to do static branch prediction (meaning you have to assume at compile time whether a branch is taken or not), how should you do it? You can make a different decision for branches that go forward or backward.
    a. If we were building a processor and had to do static branch prediction we would implement different decision for forward and backwards branches. For a forward branch we would predict not taken. For a backward branch we would predict taken because most loops are backward branch and the loop is more likely to continue to the next iteration than it is to terminate.

2. If you are building a 256-byte direct-mapped cache, what should you choose as your block (line) size?
    a. We would choose a block size of 32 bytes. This is based on the results of fib, shangO1 and shangO2. Even though a block size of 256 bytes worked best on fib, a block size of 32 bytes performed well in all the test files. The benefit or performing well on both a small test file and a large test file is why we would choose 32 bytes.

3. What conclusions can you draw about the differences between compiling with no optimization and -O3 optimization?
    a. The is a big difference between compiling with no optimization compare with O3. First O3 performs the same optimization as O1 and O2. This means that O3 gets the smaller and faster executables that result from O1 and O2 most of the time. O3 optimization can result in a slower and larger executable file based on the more expensive optimizations that have to be performed. This a very different from no optimization. With no optimization, gcc compiles the source code in the most straightforward way possible. This makes it a good option to use when debugging a program.