Compiler Optimizations and Debugging

The object code produced by a compiler using straightforward compiling techniques is often not very efficient. In many cases, the object code can be made to run faster or take up less space through program transformations known as optimizations. Compilers that improve performance through code transformations are known as optimizing compilers.

In an unoptimized program, there is generally a one-to-one correspondence between a source statement and a group of one or more machine instructions. Optimizing compilers must preserve the correctness of a program, but after optimization, this one-to-one correspondence no longer exists in many cases, and the program may no longer execute in the order implied by the source code. This complicates debugging. Programmers must often resort to debugging assembly code to figure out how an optimized program is behaving.

In general, users should debug the unoptimized version of a program before using the optimizer. However, there are a number reasons why the ability to debug optimized code is desirable:

- A program may run correctly when compiled without optimization and fail when compiled with optimization. This can happen even if the optimizer is working correctly. For example, reordering statements may result in arithmetic overflow or underflow. An uninitialized variable or an out-of-bounds memory reference might cause a problem in the optimized version but not in the unoptimized version.
- The time or space requirements of an unoptimized program might be too big to allow adequate testing.
- Production code is often optimized. A customer may submit a bug report with a core file produced from an optimized program. It would be desirable to have the ability to analyze the core file.
- Optimizing compilers can be written more easily with good tools for debugging optimized code.
- The compiler may not have the ability to generate unoptimized code.
- The programmer may have mistakenly supplied explicit assertions, directives, or options that caused the optimizing compiler to generate incorrect code.
- Optimizing compilers may have bugs.