What does "Optimize Code" option really do in Visual Studio?

Asked 16 years, 3 months ago Modified 5 years, 10 months ago

Viewed 59k times

84

Name of the option tells something but what Visual Studio/compiler really do and what are the real consequences?

Edit: If you search google you can find [this address](), but that is not really I am looking for. I wonder the real things happening. For example why do the loops get less time, etc.

visual-studio optimization

Share

Improve this question

Follow

edited Dec 7, 2008 at 10:34

asked Sep 22, 2008 at 8:51

spinodal

4,027 ● 3 ● 31 ● 39

3 Answers

Sorted by: Highest score (default)

67

Without optimizations the compiler produces very dumb code - each command is compiled in a very straightforward manner, so that it does the intended thing. The Debug builds have optimizations disabled by default, because without the optimizations the produced executable matches the source code in a straightforward manner.

Variables kept in registers

Once you turn on the optimizations, the compiler applies many different techniques to make the code run faster while still doing the same thing. The most obvious difference between optimized and unoptimized builds in Visual C++ is the fact the variable values are kept in registers as long as possible in optimized builds, while without optimizations they are always stored into the memory. This affects not only the code speed, but it also affects debugging. As a result of this optimization the debugger cannot reliably obtain a variable value as you are stepping through the code.

Other optimizations

There are multiple other optimizations applied by the compiler, as described in /O Options (Optimize Code) MSDN docs. For a general description of various optimizations techniques see Wikipedia Compiler Optimization article.

Share Improve this answer

Follow

edited Feb 3, 2019 at 2:21

user1991179

623 ● 3 ● 11 ● 25

answered Sep 22, 2008 at 10:00

Suma

34.3k ● 18 ● 129 ● 199

Do you know how the "Optimize Code" option affects the StackTrace class? In this question stackoverflow.com/questions/628565/…, coxymla recommends to uncheck it, but I didn't do it and it still worked. Do you know why this is? Thanks! – Carlo Mar 1, 2010 at 20:25

1 @Carlo: When inlining, you cannot see the inlined functions in the stack trace as they do not exist at all in the code (this is what inlining is all about). – Suma Mar 2, 2010 at 9:54

18

From Paul Vick's blog:

- It removes any NOP instructions that we would otherwise emit to assist in debugging. When optimizations are off (and debugging information is turned on), the compiler will emit NOP instructions for lines that don't have any actual IL associated with them but which you might want to put a breakpoint on. The most common example of something like this would be the “End If“ of an “If” statement - there's no actual IL emitted for an End If, so we don't emit a NOP the debugger won't let you set a breakpoint on it. Turning on optimizations forces the compiler not to emit the NOPs.

- We do a simple basic block analysis of the generated IL to remove any dead code blocks. That is, we break apart each method into blocks of IL separated by branch instructions. By doing a quick analysis of how the blocks interrelate, we can identify any blocks that have no branches into them. Thus, we can figure out code blocks that will never be executed and can be omitted, making the assembly slightly smaller. We also do some minor branch optimizations at this point as well - for example, if you GoTo another GoTo statement, we just optimize the first GoTo to jump to the second GoTo's target.
- We emit a DebuggableAttribute with IsJITOptimizerDisabled set to False. Basically, this allows the run-time JIT to optimize the code how it sees fit, including reordering and inlining code. This will produce more efficient and smaller code, but it means that trying to debug the code can be very challenging (as anyone who's tried it will tell you). The actual list of what the JIT optimizations are is something that I don't know - maybe someone like Chris Brumme will chime in at some point on this. The long and the short of it is that the optimization switch enables optimizations that might make setting breakpoints and stepping through your code harder.

answered Sep 22, 2008 at 10:03

reva
1,477 ● 2 ● 14 ● 24

2

The short answer is: use -Ox and let the compiler do its job.

The long answer: the effect of different kind of optimizations is impossible to predict accurately. Sometimes optimizing for fast code will actually yield smaller code than when optimizing for size. If you really want to get the last 0.01% of performance (speedwise or sizewise), you have to benchmark different combination of options.

Also, recent versions of Visual Studio have options for more advanced optimizations such as link-time optimization and profile-guided optimization.

Share Improve this answer Follow

answered Sep 22, 2008 at 9:14

JesperE
64.3k ● 22 ● 142 ● 199