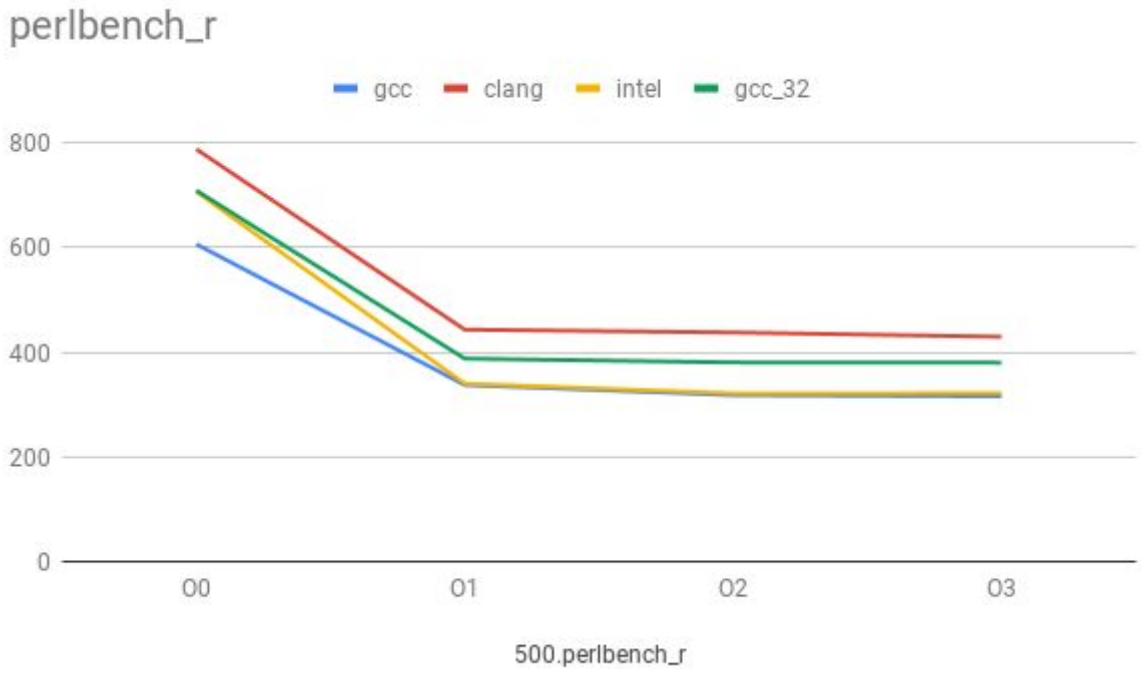
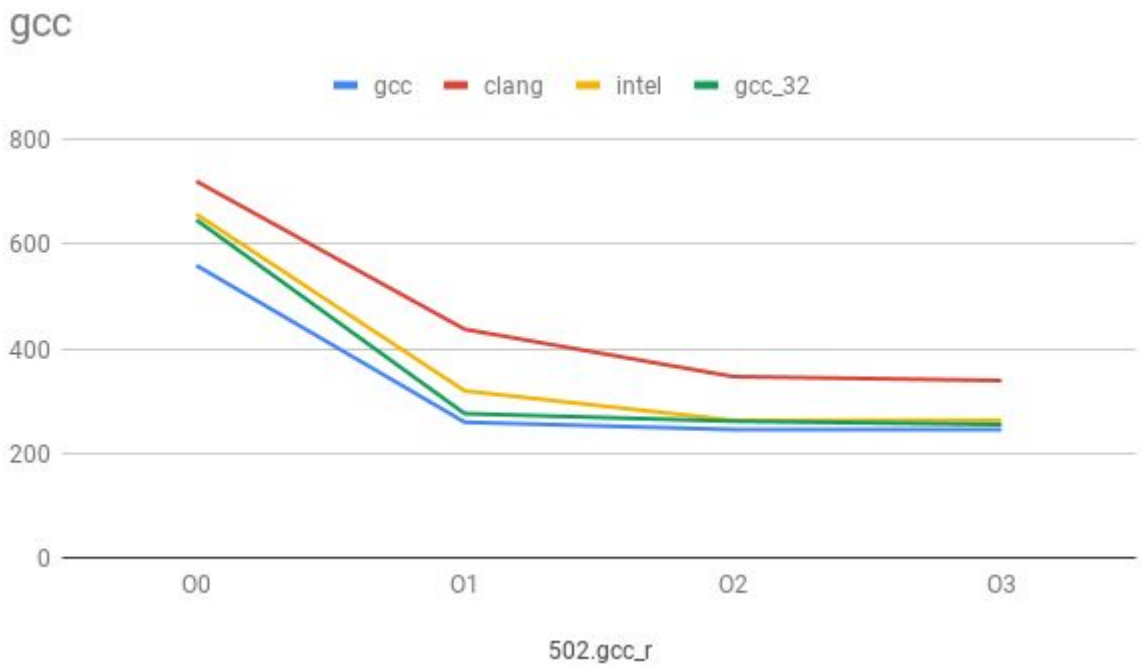
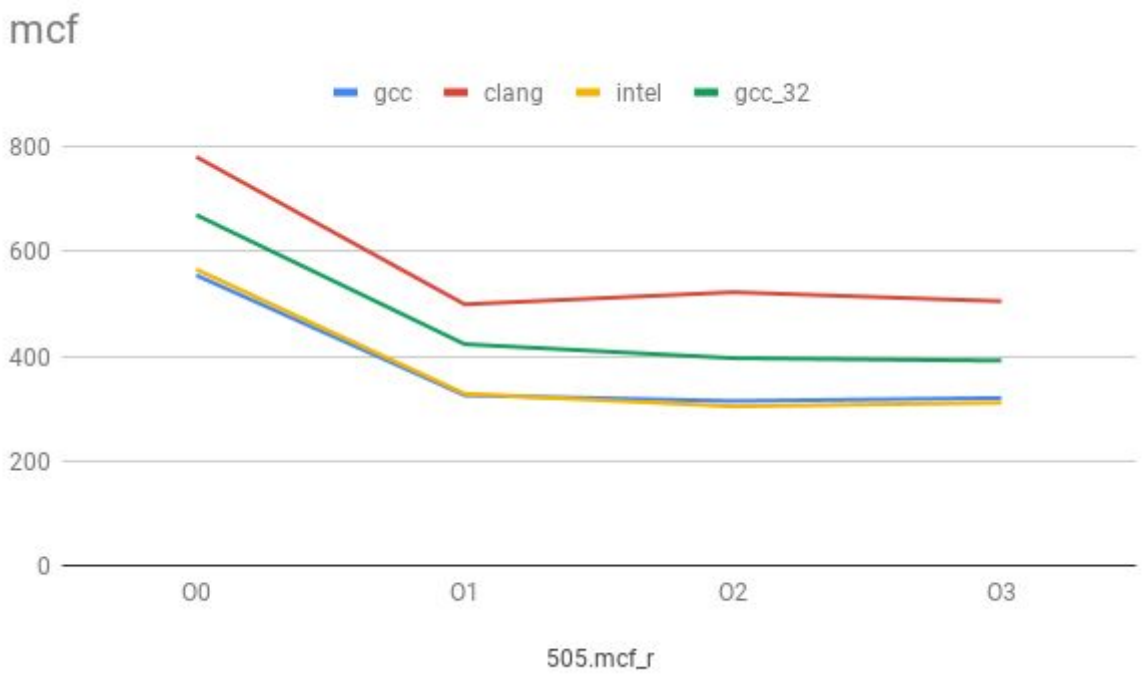
COL729 : Compiler Optimizations
Assignment 0
Praneeth Kacham (2015CS10600)

Configuration:
Benchmarks are run on a machine with Ubuntu 18.04 LTS operating system. It has a quad core Intel i7-4790S processor with 16GB of RAM. Benchmarks are run using 3 different compilers gcc-7.3.0, icc-17.0.1.132 and clang-3.9.0. Four different optimization levels are used for each compiler (-O0, -O1, -O2, -O3). Gcc-7.3.0 is also used to compile into 32-bit applications.

Plots:
Following are the plots for each of the 10 benchmarks:




gcc
gcc
clang
intel
gcc_32
800
600
400
200
0
O0
O1
O2
O3
502.gcc_r


mcf
gcc
clang
intel
gcc_32
800
600
400
200
0
O0
O1
O2
O3
505.mcf_r

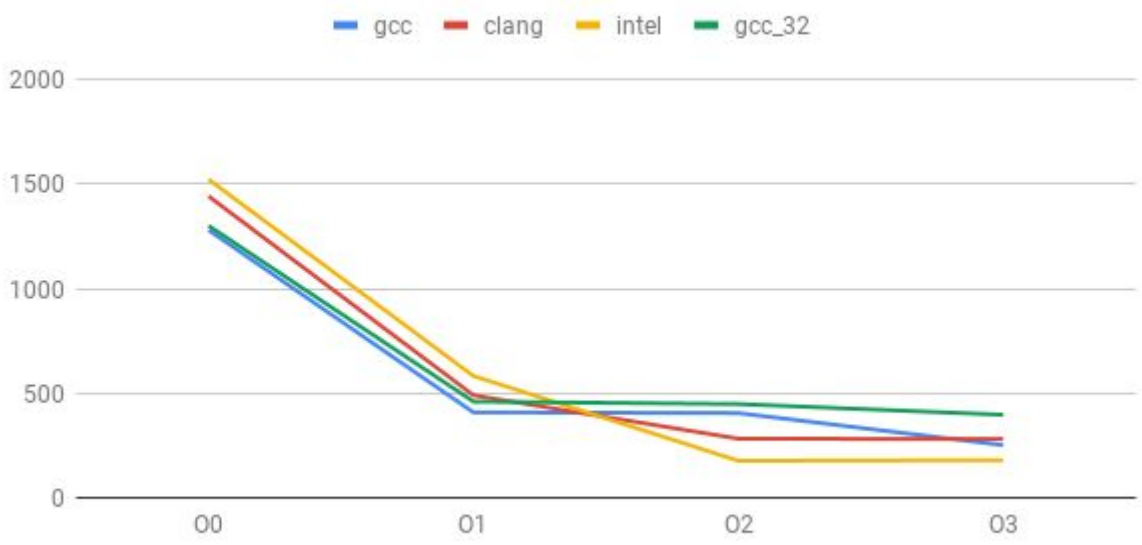
omnetpp

520.omnetpp_r

xalancbmk

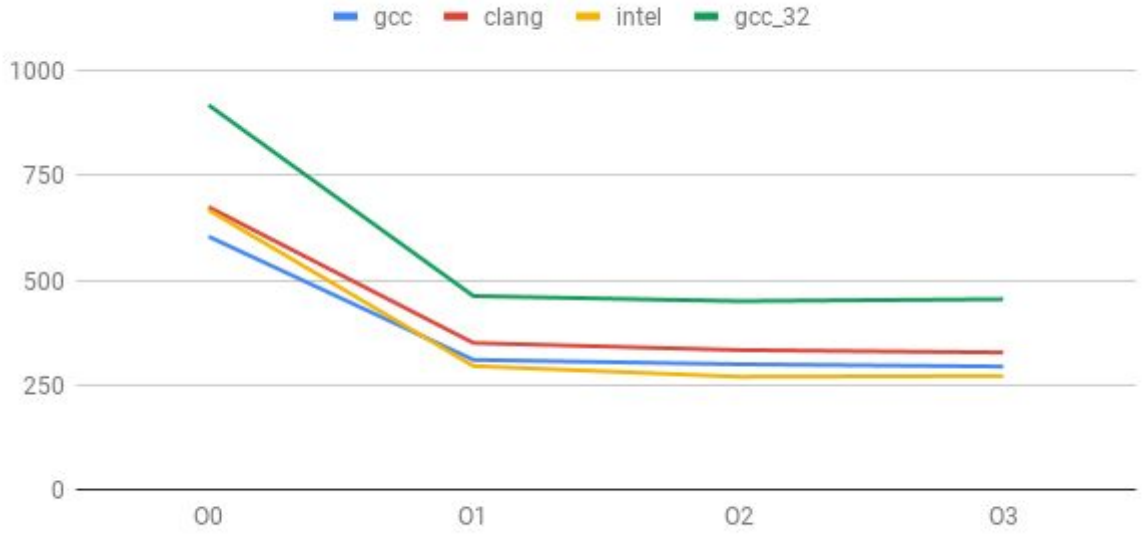
523.xalancbmk_r

x264



525.x264_r

deepsjeng

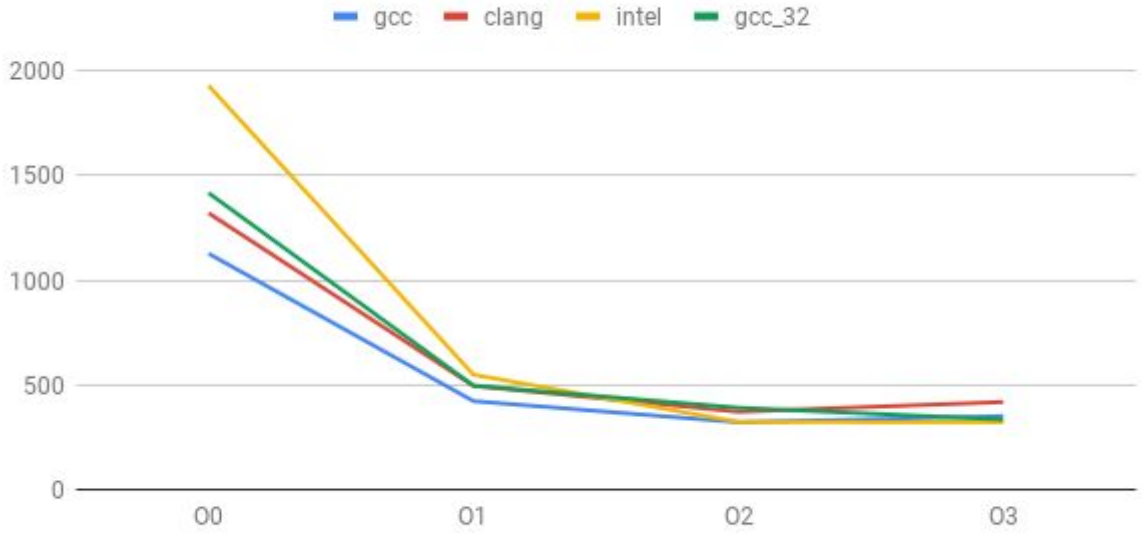


531.deepsjeng_r

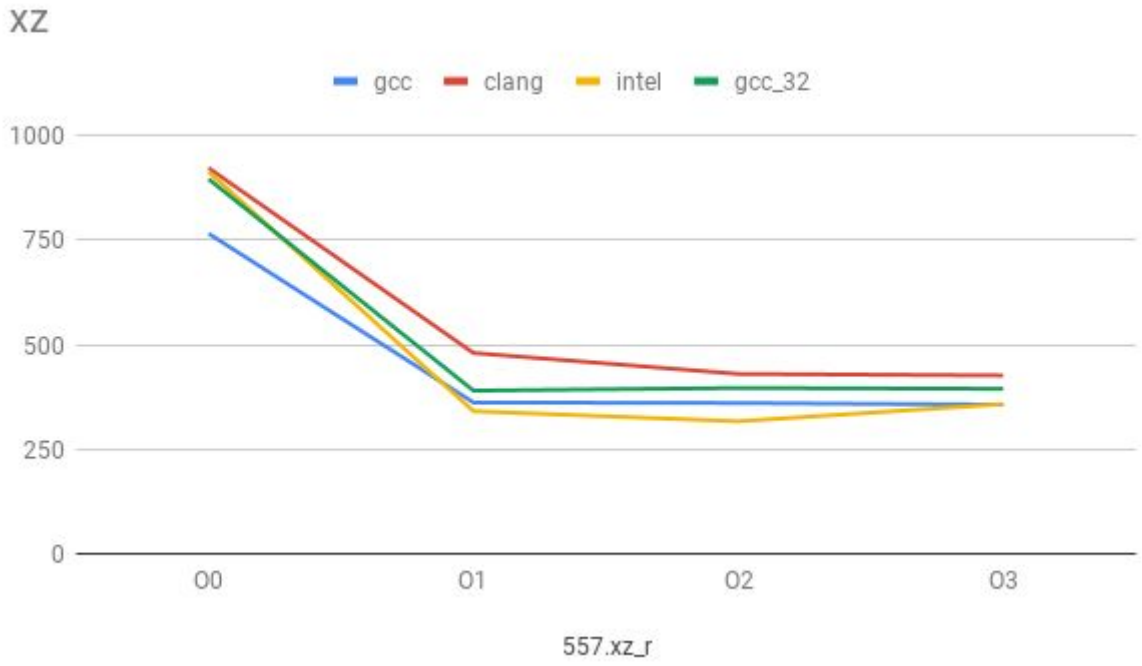
leela

541.leela_r

exchange2



548.exchange2_r



Few General Observations:

1. In almost all the benchmarks and almost all the compilers, running time decreases with increasing the optimization level.
2. In most of the benchmarks, gcc_runtime < intel_runtime < clang_runtime.
3. In all the benchmarks at all optimization levels, gcc64_runtime < gcc32_runtime.
4. In few cases, applications compiled with intel and clang compilers with -O2 flag are faster than the applications compiled with -O3 flags.

Optimization levels across compilers:

1. -O0 : Most optimizations are completely disabled at this optimization level.
2. -O1 : Activates many level 1 optimizations such as block reordering, omitting frame pointers, branch probability guessing etc.
3. -O2 : Activates level 1 optimizations along with many level 2 optimizations such as function aligning, loop aligning, expensive optimization routines, deleting null-pointer checks etc.
4. -O3 : Activates level 1&2 optimizations along with level 3 optimizations such as function inlining, loop unrolling, loop vectorizing etc.

32-bit vs 64-bit applications:

In all the benchmarks across all optimization levels, 64-bit applications are at least as fast as 32-bit applications.

1. 64-bit applications have access to more registers(16) compared to 32-bit applications(8). This allows compiler to better optimize reads and writes to the memory.
2. Applications can address more virtual memory which allows compilers to do more aggressive optimizations.

3. If the memory footprint of applications is small and only few temporary variables are used by the application, 32-bit and 64-bit applications both run in almost the same time. In the case of gcc, xalancbmk and exchange2 benchmarks, we observe that both 32-bit versions and 64-bit versions run in almost the same time at higher optimization levels.

Effects of optimization levels:

1. In almost all of the benchmarks, there is a huge drop in runtimes between applications compiled with -O0 and applications compiled with -O1.
2. In most of the benchmarks, The difference between runtimes of -O1, -O2, -O3 are very minor. But in general they follow the trend that -O0_time > -O1_time > -O2_time > -O3_time.
3. In few benchmarks, -O2_time < -O3_time. Compiling with -O3 flag in general produces binaries of larger sizes compared to -O2 flag. This may sometimes lead to higher cache miss-rates leading to performance degradation. There can also be other application dependent effects like branch prediction accuracy etc., which may lead to degraded performance. These kinds of effects are very dependent on specific applications but in general applications are faster when compiled with -O3.