**Compiler Type and Version**

Initially I stared to improve the speed of *stencil.c* by changing which compiler I was using, and the compiler versions. As seen from Table1, there was a marginal increase from the default Blue Crystal compiler (GCC 4.8.5) to 9.1.0, which is due to the latter being a more recent compiler, hence having more efficient optimisations.

Comparing the GCC compilers with the intel ICC compilers shows a 3.35X improvement on run time. From the compiler reports of both GCC and ICC it was shown that the Intel compiler was able to vectorise the loops inside *stencil.c.*

Vectorisation is able to provide such a drastic decrease in run time as it utilises vector operations. This is enabled via Single Instruction, Multiple Data (SIMD) hardware in order to replace multiple operations with a single operation, which is applied to multiple data items at once.

Table1 – showing the taken times for each compiler  
version used.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Default | 5.906s |  | 5.904s | 5.907s |
| GCC 9.1.0 | 5.875s |  | 5.875s | 5.873s |
| ICC 2017.01 | 1.796s |  | 1.796s | 1.797s |
| ICC 2018-u3 | 1.796s |  | 1.796s | 1.796s |

**Compiler flags**

Table2 – showing the taken times for each compiler  
flag that was tested.

|  |  |  |  |
| --- | --- | --- | --- |
| -O0 | 6.045s | 6.045s | 6.045s |
| -O1 | 2.003s | 2.002s | 2.002s |
| -O2 | 1.795s | 1.796s | 1.796s |
| -O3 | 1.797s | 1.796s | 1.796s |
| -Os | 2.004s | 2.003s | 2.003s |
| -Ofast | 0.251s | 0.244s | 0.245s |
| -fast | 0.196s | 0.186s | 0.186s |

**Code Optimisations**

C

**Final Results and Improvements**

D