**Running Instructions**

./highlife <pattern\_number> <grid\_size> <number\_of\_iterations> <block\_size>

The program was run on AiMOS for 7 different grid sizes and 6 block sizes. The data from each run in plotted on the following two pages. The execution time grows exponentially with grid size. Since increasing grid size increases the number of cell updates per second quadratically, this is expected. For each grid size, the slowest execution was with a block size of 8 threads. This is expected since this configuration yields the highest number of blocks. Since each SM can only run one block at a time, such a small block will drastically decrease performance. With a hardware warp size of 32 threads, any block size less than 32 threads should always run slower than a block size of 32 or above.

It was expected the speed would peak at 32 threads and remain relatively constant from 32 threads and up. In reality, most grid sizes saw speed improvements up to 128 threads, with 256 running at a similar speed. This could be because of optimizations the hardware made to run one block on multiple SM’s concurrently. This optimization effectively allowed the hardware to simulate running with a warp size of 128 threads. In other tests on a smaller system, there was a clear flatlining of performance from 32 threads and above, the expected result. The AiMOS system performed differently to that smaller system.

The second figure lists all run in order of the greatest number of cell updates per second. The greatest speed was achieved by the grid size of 65536 x 65536 and a block size of 128.

**Block Size and Grid Dimensions vs Execution Time (in seconds)**

**All 42 Simulations Ordered by Cell Updates per Second**

