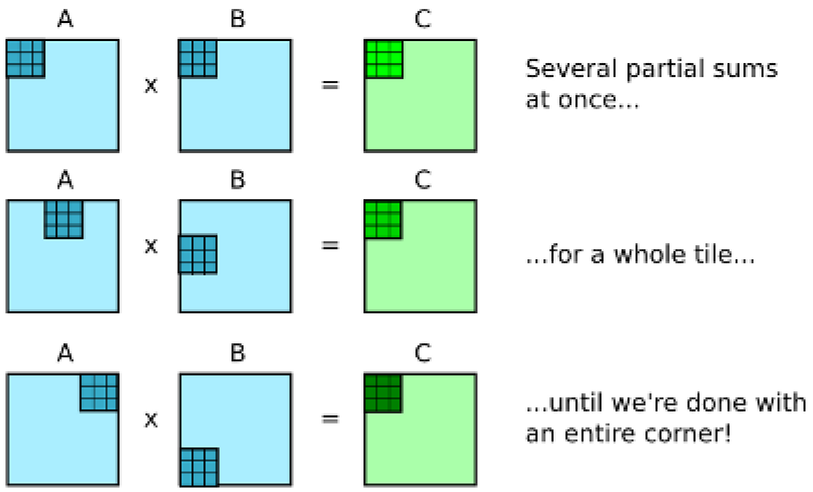
Performance improvements due to memory hierarchy

The objective of this tutorial is to observe the performance gains by using memory-hierarchy aware programming.

1) The given matrix_add.c code adds to matrices. Modify the code to improve the execution-time of the program. In addition, provide the reason for the improvement.
2) The given matrix_mul.c code multiplies two matrices. The naïve version of the matrix multiply is already written. The below image shows how a matrix can be tiled into smaller sub-matrices and then multiplied together to generate partial sums.


A
B
C
x
=
Several partial sums at once...
...for a whole tile...
...until we're done with an entire corner!

Using the logic of tiling illustrated in the above picture, write the tiled version of the matrix multiplication in the corresponding section indicated in matrix_mul.c code.
Assume that the matrix sizes (rows and columns are powers of two and the tile sizes are also powers of two. Report the performance in terms of wall clock time for different tile values (2, 4, 8, 16 so on) for matrix sizes of 1024, 2048 and 4096. Provide the reason for performance improvement with tiling. Use the defined parameters given in the code to vary tile size and matrix sizes.

Challenge question:

Use linux perf tool to measure the key performance indicators such as L1 cache accesses, hit rate report these values for the above.