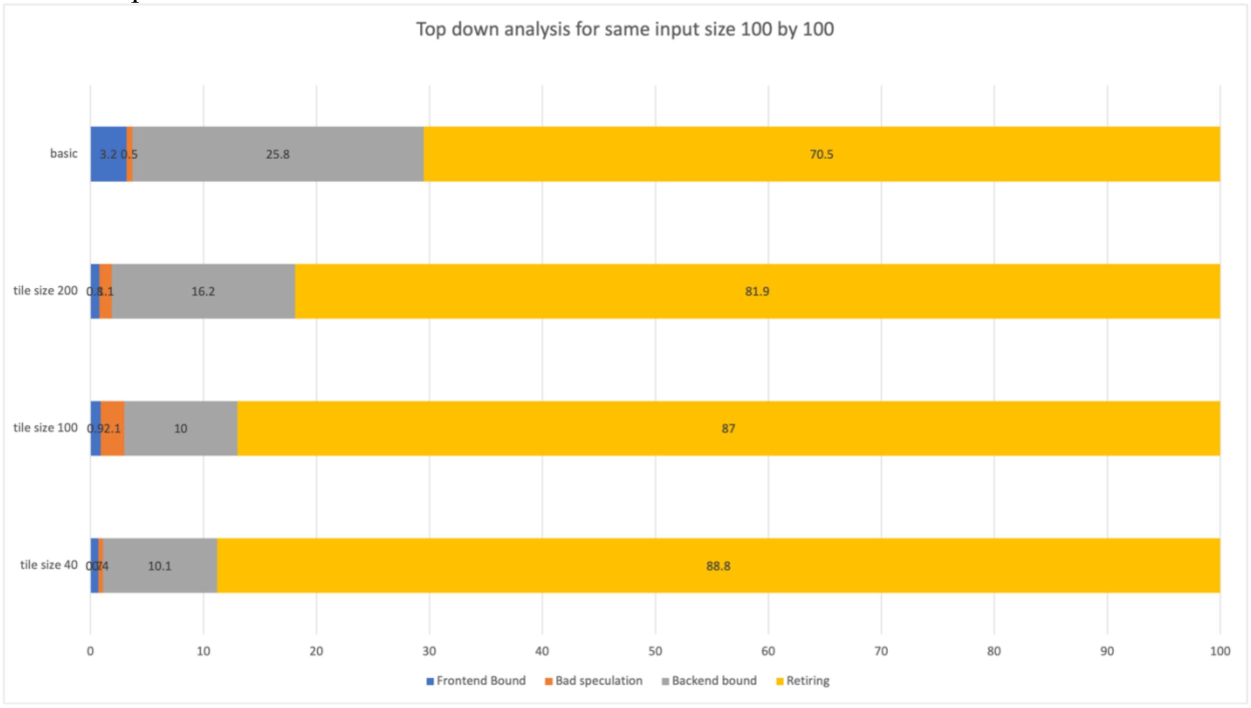
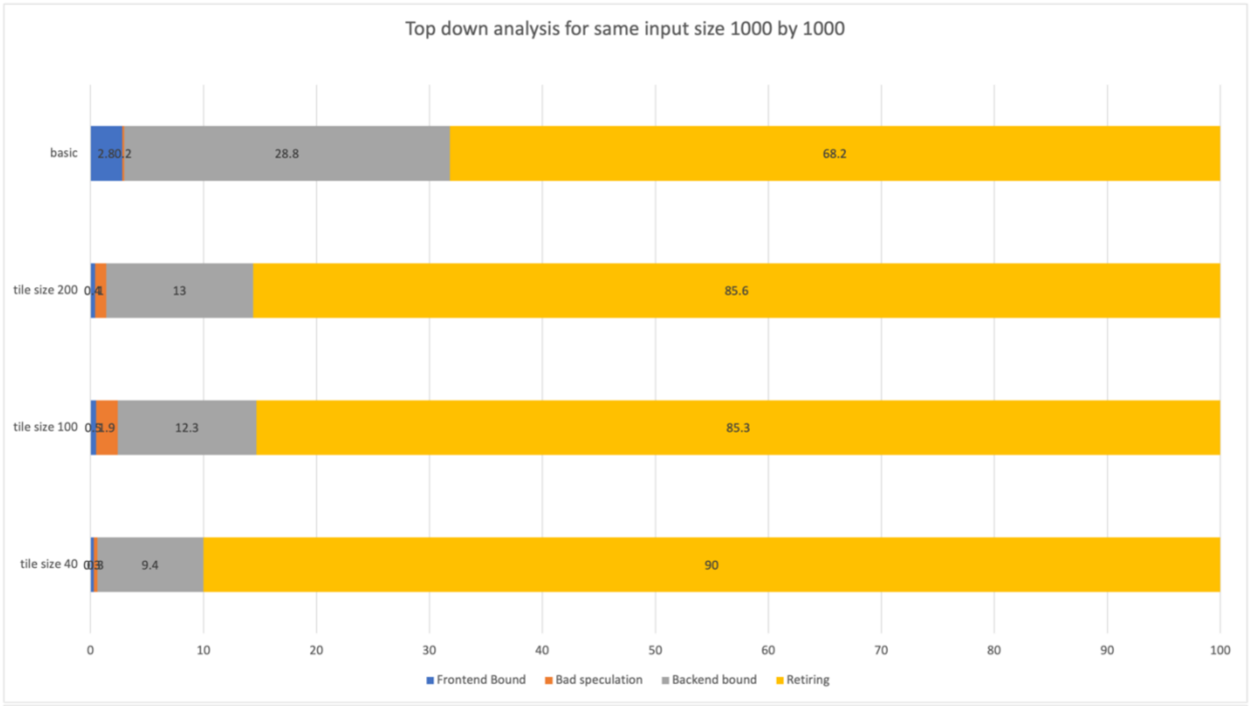
Throughout the report, when an input size is specified as 100, it means that the matrix multiplication was done with 2 matrices, each is in the shape of 100 by 100. For brevity, all references to input size will be implied as such.

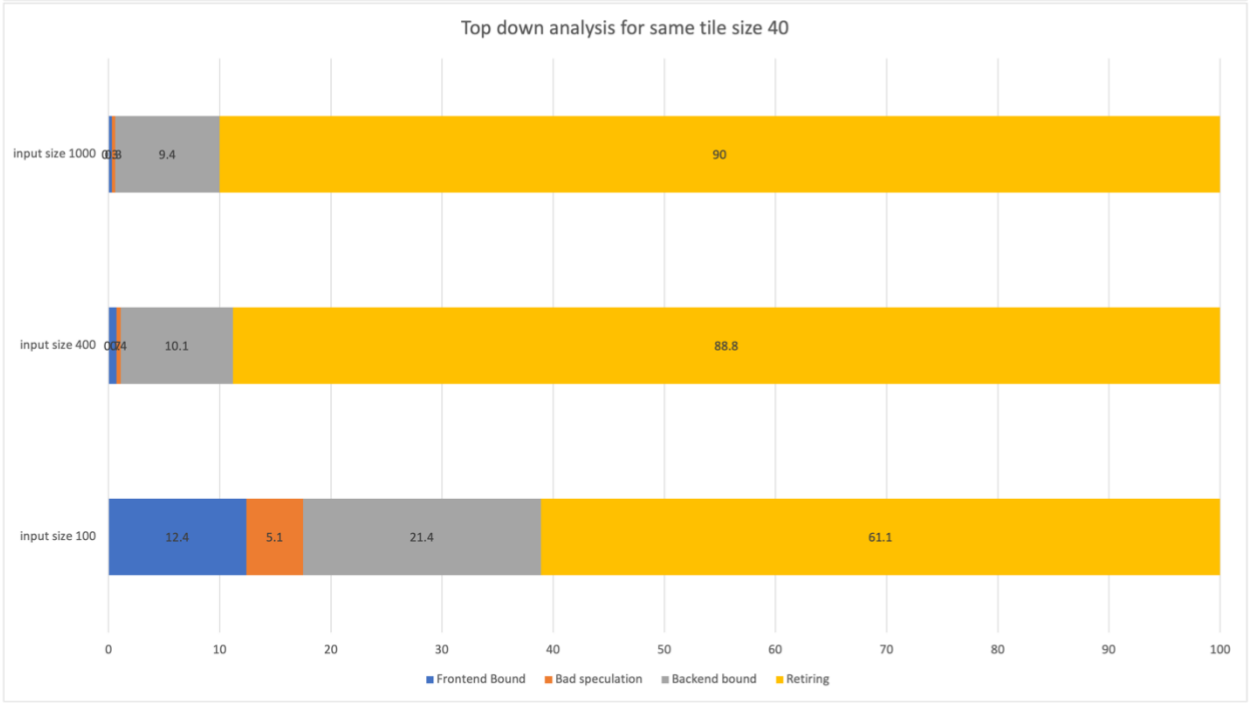
Run time analysis (in seconds)

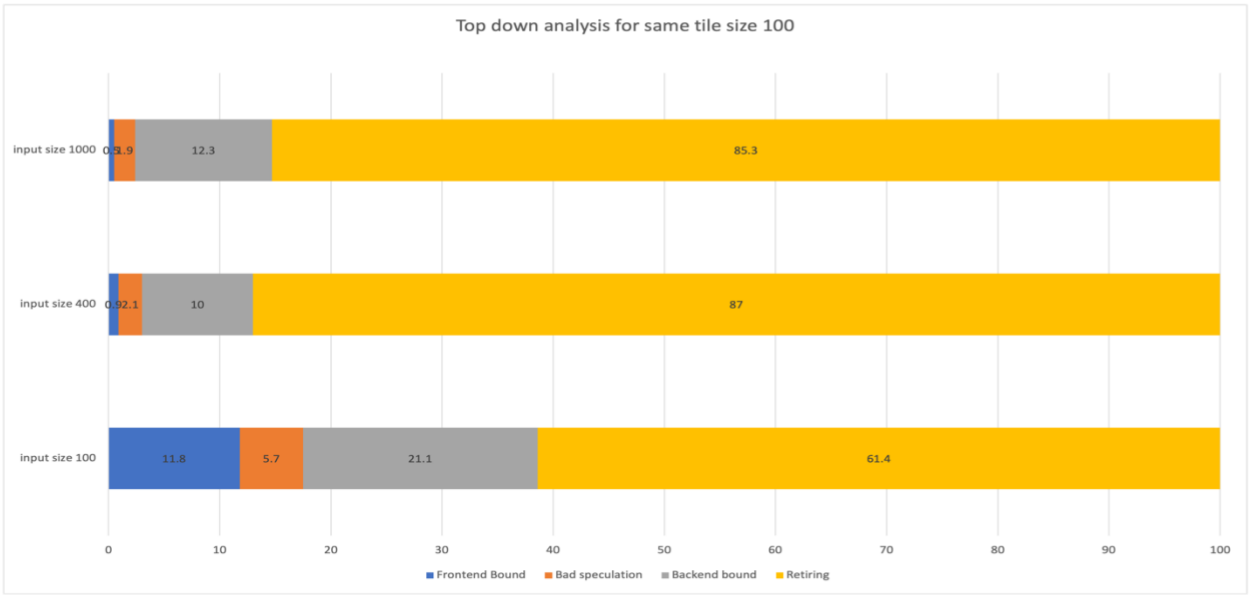
Input size / Tile size	100	400	1000
40	0.00805	0.464	7.24
100	0.00736	0.47	7.6
200	NA	0.499	7.57
basic	0.00632	0.51	8.33

I did not record the time for tile size of 200 for input size of 100 because it will just perform like a basic implementation.




Top down analysis for same input size 1000 by 1000
basic
2.8
0.2
28.8
68.2
tile size 200
0.4
1
13
85.6
tile size 100
0.5
1.9
12.3
85.3
tile size 40
0.3
0.3
9.4
90
0
10
20
30
40
50
60
70
80
90
100
Frontend Bound
Bad speculation
Backend bound
Retiring


Top down analysis for same tile size 40
input size 1000
0.3
0.3
9.4
90
input size 400
0.7
0.4
10.1
88.8
input size 100
12.4
5.1
21.4
61.1
0
10
20
30
40
50
60
70
80
90
100
Frontend Bound
Bad speculation
Backend bound
Retiring



I recorded the runtime of each multiplication 20 times and take the average, for both profiling and recording runtime. For recording runtime, a timer that starts right before the line to call the function matmul_blocking, and ends right after, is used and the time elapsed is printed onto the console.
tiling or blocking optimizations in matrix multiplication are designed to maximize the use of fast cache memory and minimize data movement between main memory and CPU. Accessing data from L1 cache is the fastest, while accessing data from the main memory or secondary storage is much slower. The idea behind tiling or blocking is to take advantage of this memory hierarchy, specifically the faster cache levels. Tiling/blocking takes advantage of spatial locality and temporal locality: when the function calls for data, it fetches an entire block of data required for the calculation at once, as opposed to the basic implementation.

Front end (FE) bound differences:
The non-tiled version might spend more time fetching instructions due to frequent cache misses, resulting in a higher FE metric. The tiled version, due to improved cache locality, might see a reduced FE metric.
Backend (BE) bound differences:
The non-tiled version showed to be more memory-bound due to frequent cache misses and increased accesses to main memory. Thus, a higher BE metric could be observed.
Retiring:
The tiled version might have a higher retiring percentage because more instructions successfully complete their operation without being stalled by other bottlenecks.

With large enough an input size, the runtime differences between the tiled and non-tile version become more pronounced. More experiments need to be done however to find out the ideal tile size for each input size. This is because the program does not have to access memories all the time and could fetch data in chunks.