```
__global__ void MatrixMulKernel(float* d_M, float* d_N, float* d_P,
    int Width) {

1.  __shared__ float Mds[TILE_WIDTH][TILE_WIDTH];
2.  __shared__ float Nds[TILE_WIDTH][TILE_WIDTH];

3.  int bx = blockIdx.x;  int by = blockIdx.y;
4.  int tx = threadIdx.x; int ty = threadIdx.y;

    // Identify the row and column of the d_P element to work on
5.  int Row = by * TILE_WIDTH + ty;
6.  int Col = bx * TILE_WIDTH + tx;

7.  float Pvalue = 0;
    // Loop over the d_M and d_N tiles required to compute d_P element
8.  for (int m = 0; m < Width/TILE_WIDTH; ++m) {

      // Coolaborative loading of d_M and d_N tiles into shared memory
9.    Mds[ty][tx] = d_M[Row*Width + m*TILE_WIDTH + tx];
10.   Nds[ty][tx] = d_N[(m*TILE_WIDTH + ty)*Width + Col];
11.   __syncthreads();

12.   for (int k = 0; k < TILE_WIDTH; ++k) {
13.     Pvalue += Mds[ty][k] * Nds[k][tx];
      }
14.   __syncthreads();
    }
15. d_P[Row*Width + Col] = Pvalue;
}
```