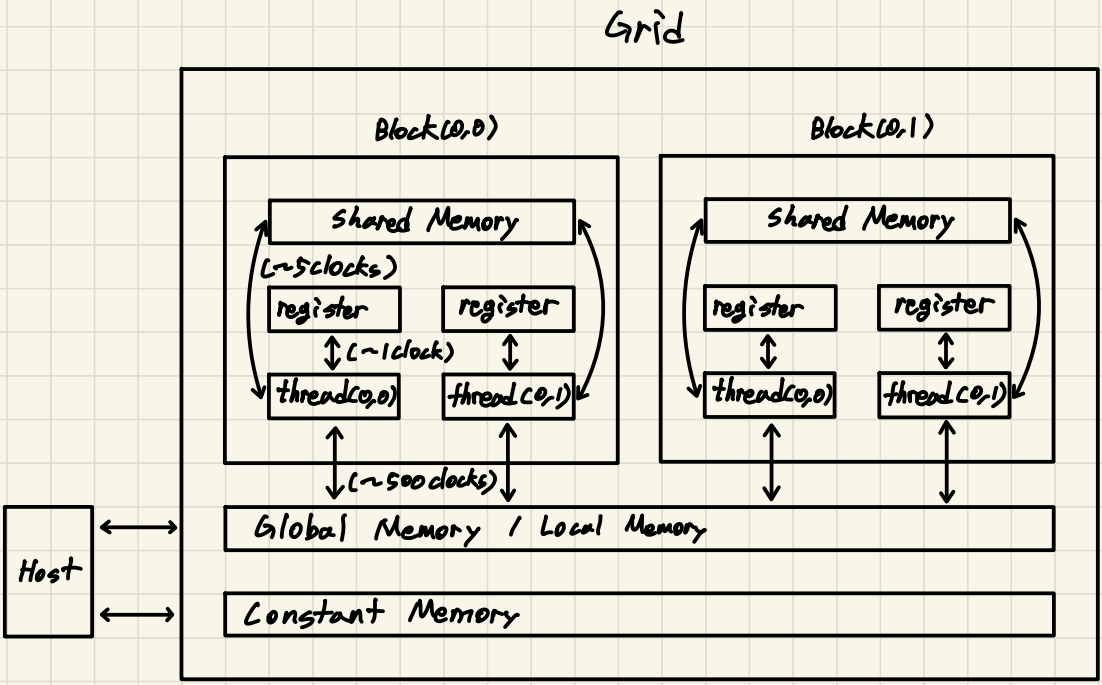
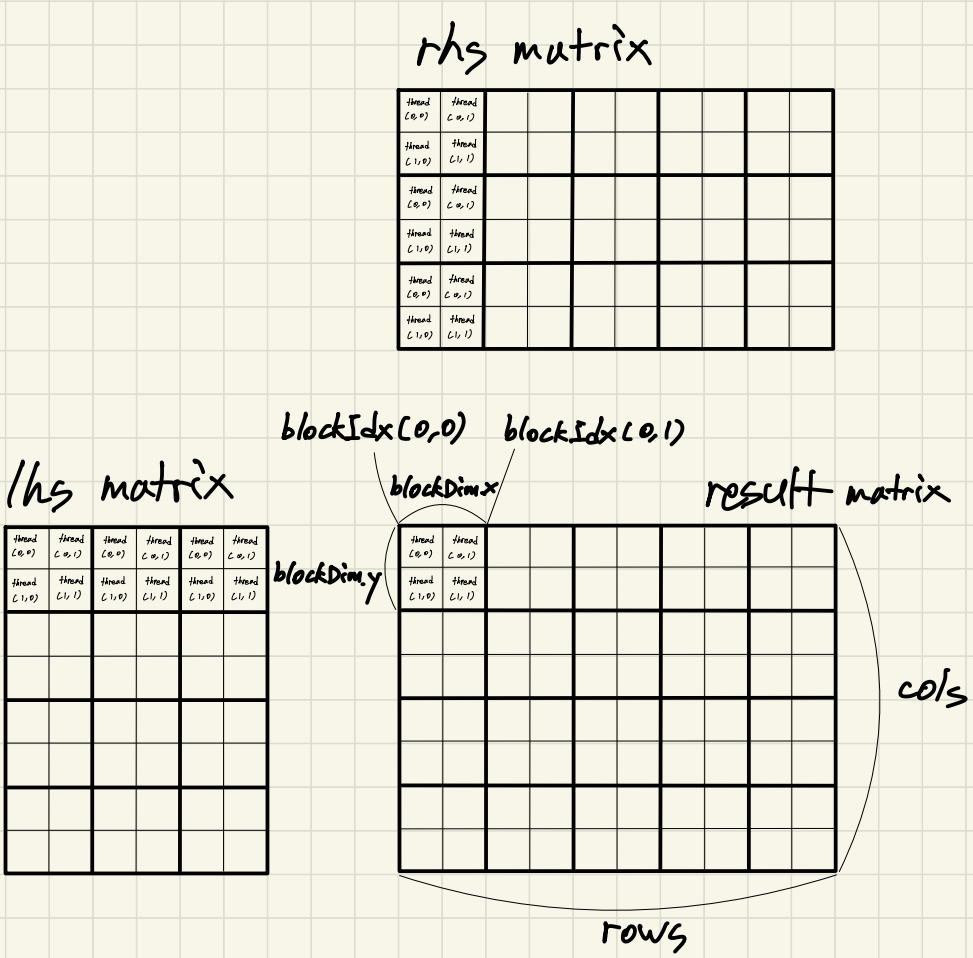
CUDA memory hierarchy



1. max 1024 threads per 1 block
2. 1 block per SM
3. (32 threads = 1 warp) per SM

```
__global__ void kernelFunc(float* dst, const float* src) {
    float p = src[threadIdx.x]; // register per thread
    float heap[10]; // local per thread
    __shared__ float partial_sum[1024]; // shared Memory per Block
}
```

TILED MATRIX MULTIPLICATION Basic

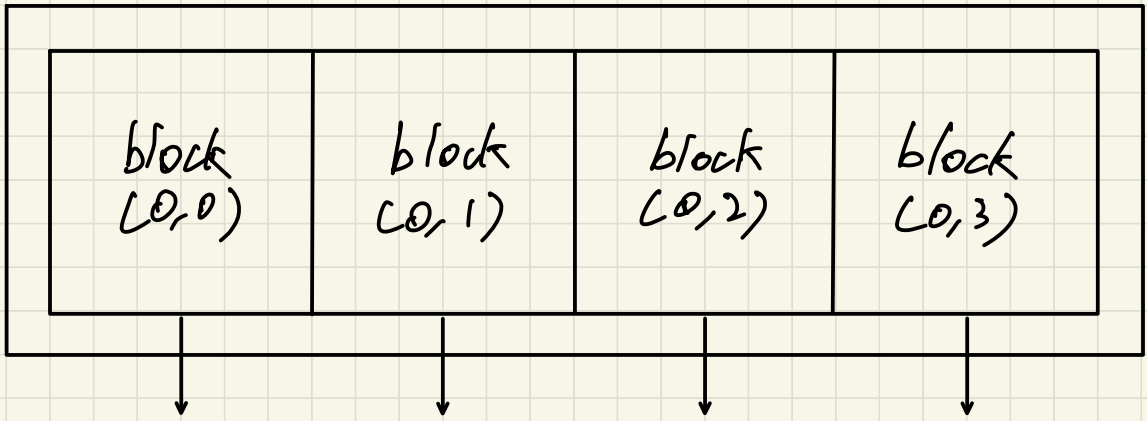


pseudo code

```
int x = blockIdx.x * blockDim.x + threadIdx.x;
int y = blockIdx.y * blockDim.y + threadIdx.y;
double sum = 0.0;
for(int i=0; i< lhs_cols; ++i)
{
    sum += lhs[y * lhs_cols + i] * rhs[i * rhs_cols + x];
}
result[y * rows + x] = sum;
```

Result matrix dimension 256 x 1
TILED MATRIX MULTIPLICATION

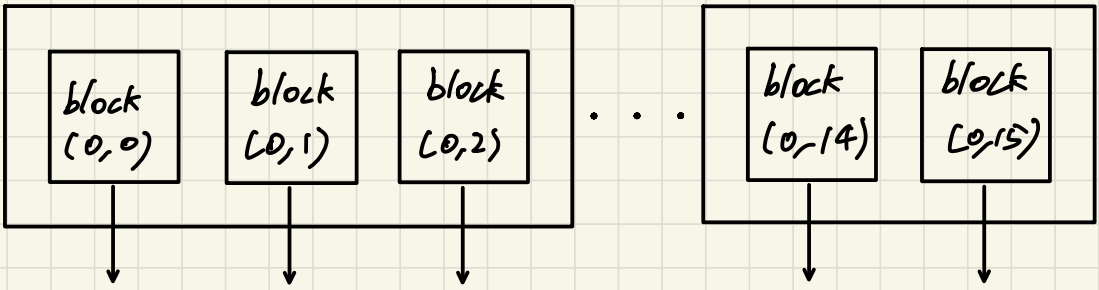
TILE WIDTH = 32



32 warps (=1024 threads) 32 warps 32 warps 32 warps

= total 128 warps

TILE WIDTH = 8



2 warps (=64 threads) 2 warps 2 warps . . . 2 warps 2 warps

= total 64 warps

TILED MATRIX MULTIPLICATION
using shared memory (TILE WIDTH = 2)