ME 471/571

Grids, blocks, threads

EXAMPLE - VECTOR ADDITION

```
void sumArraysOnHost(float *A, float *B, float *C, int N)
{
    for(int i=0; i<size i++)
        C[i] = A[i] + B[i];
}


__global__ void sumArraysOnGPU(float *A, float *B, float *C, int N)
{
    int i=threadIdx.x;
    if(i<N)
        C[i] = A[i] + B[i];
}
```

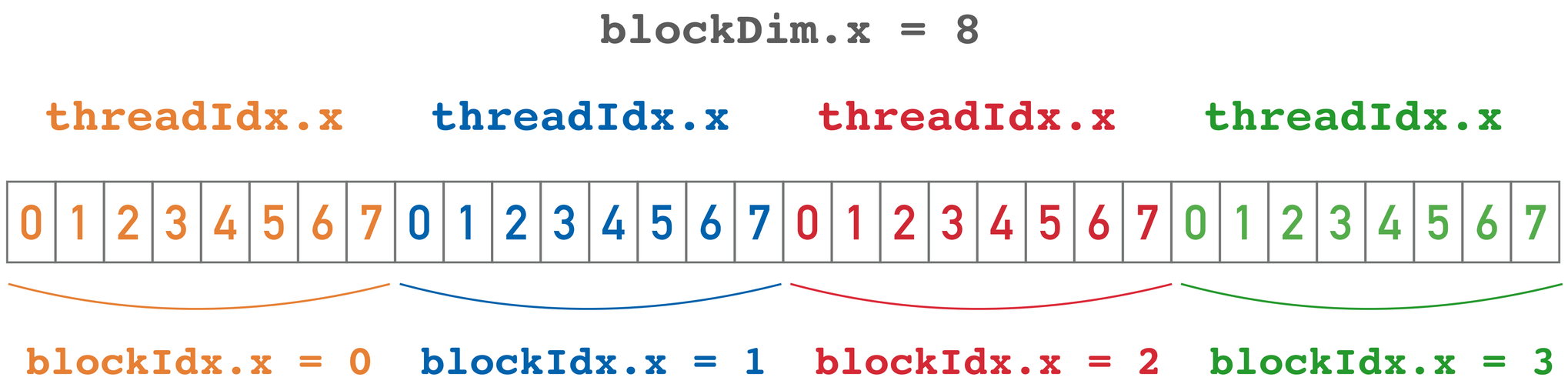
HOW TO DEBUG A KERNEL

```
__global__ void sumArraysOnGPU(float *A, float *B, float *C, int N)
    {
        int i=threadIdx.x;
        if(i<N)
            C[i] = A[i] + B[i];
    }
```

```
sumArraysOnGPU<<<1,10000>>>(d_A, d_B, d_C, N)
//DEBUG - check for kernel errors
CHECK(cudaDeviceSynchronize());
CHECK(cudaGetLastError()):
```

BLOCKS AND THREADS

`blockIdx.x`	*index of a block*
`blockDim.x`	*number of threads in a block*
`threadIdx.x`	*index of a thread in a block*



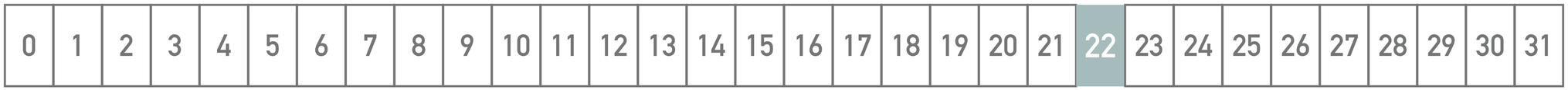
BLOCKS AND THREADS

blockDim.x = 8

threadIdx.x **threadIdx.x** **threadIdx.x** **threadIdx.x**

blockIdx.x = 0 **blockIdx.x = 1** **blockIdx.x = 2** **blockIdx.x = 3**

index = blockIdx.x * blockDim.x + threadIdx.x




```
index = blockIdx.x * blockDim.x + threadIdx.x
      =      2      *      8      +      6
      =      22
```

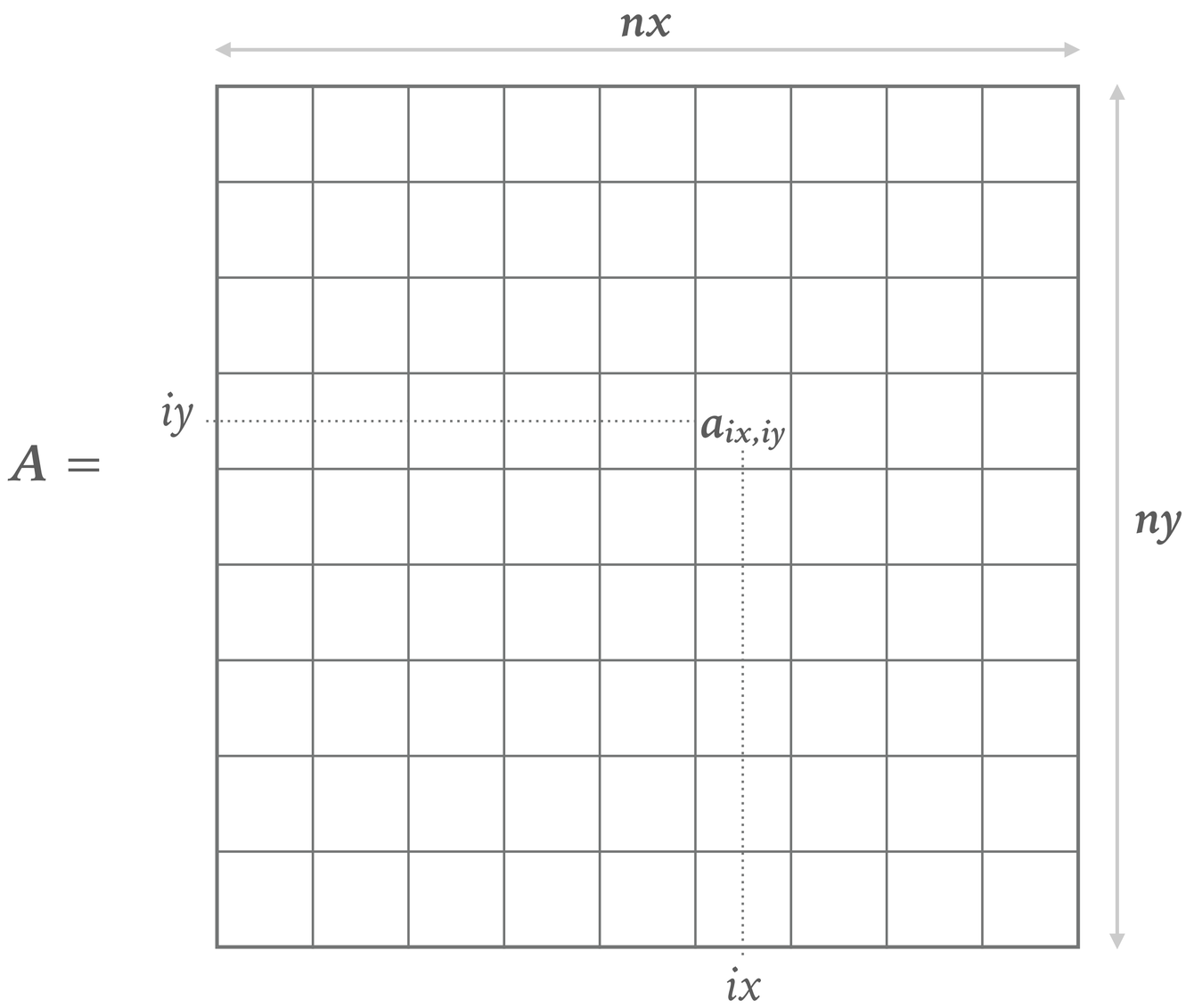
BLOCKS AND THREADS

How many blocks do we need to "cover" an array which size is not divisible by nThreads?


```
nx = 30
```

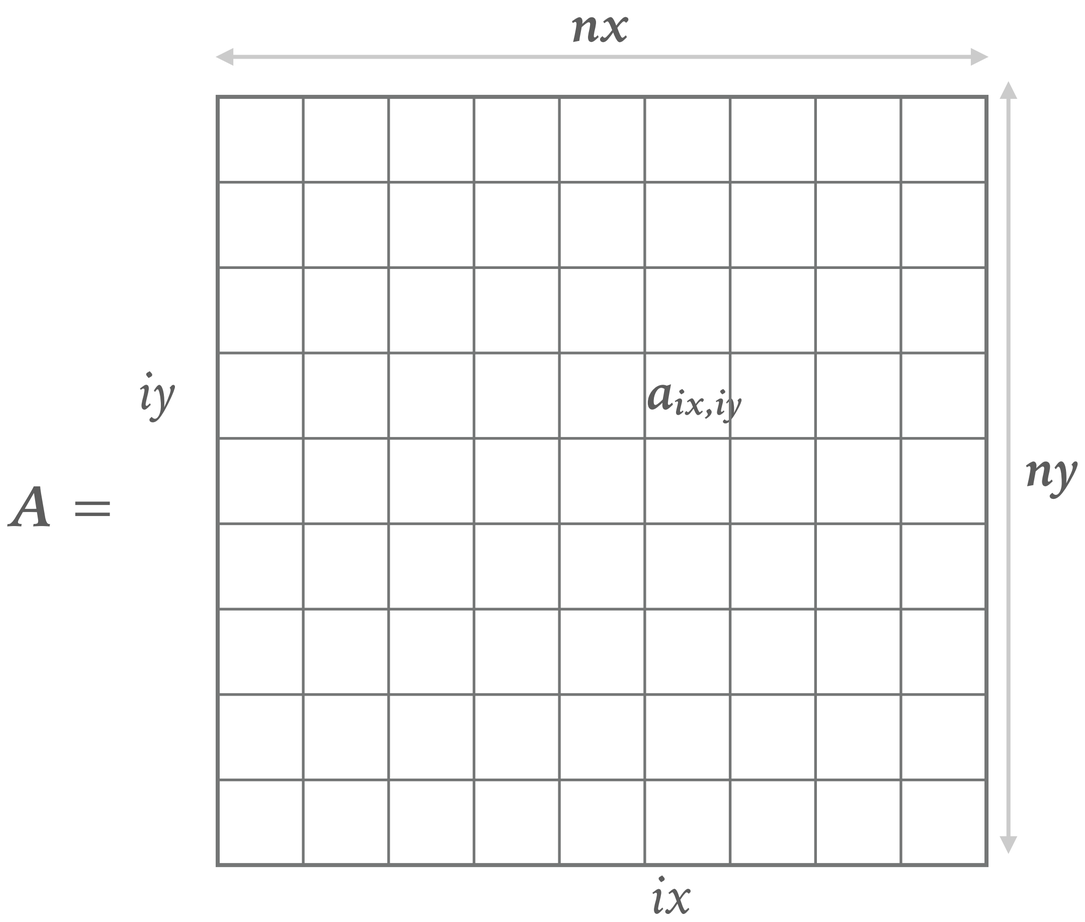

```
blockDim.x = nThreads = 8

gridDim.x = nBlocks = ?

gridDim.x = nBlocks = (nx + blockDim.x - 1) / blockDim.x
                    = (30 +      8       - 1) /      8
                    =  4
```

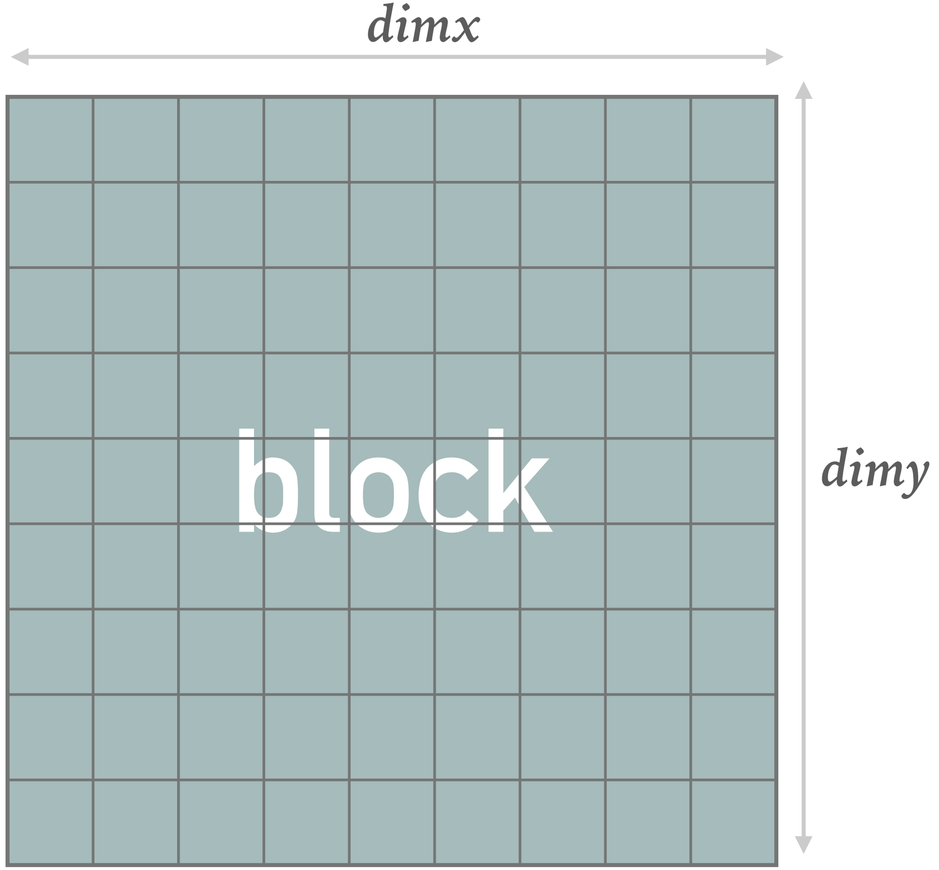
MATRIX ADDITION – MULTIDIMENSIONAL BLOCKS




```
idx = iy*nx + ix
```

MATRIX ADDITION - MULTIDIMENSIONAL BLOCKS

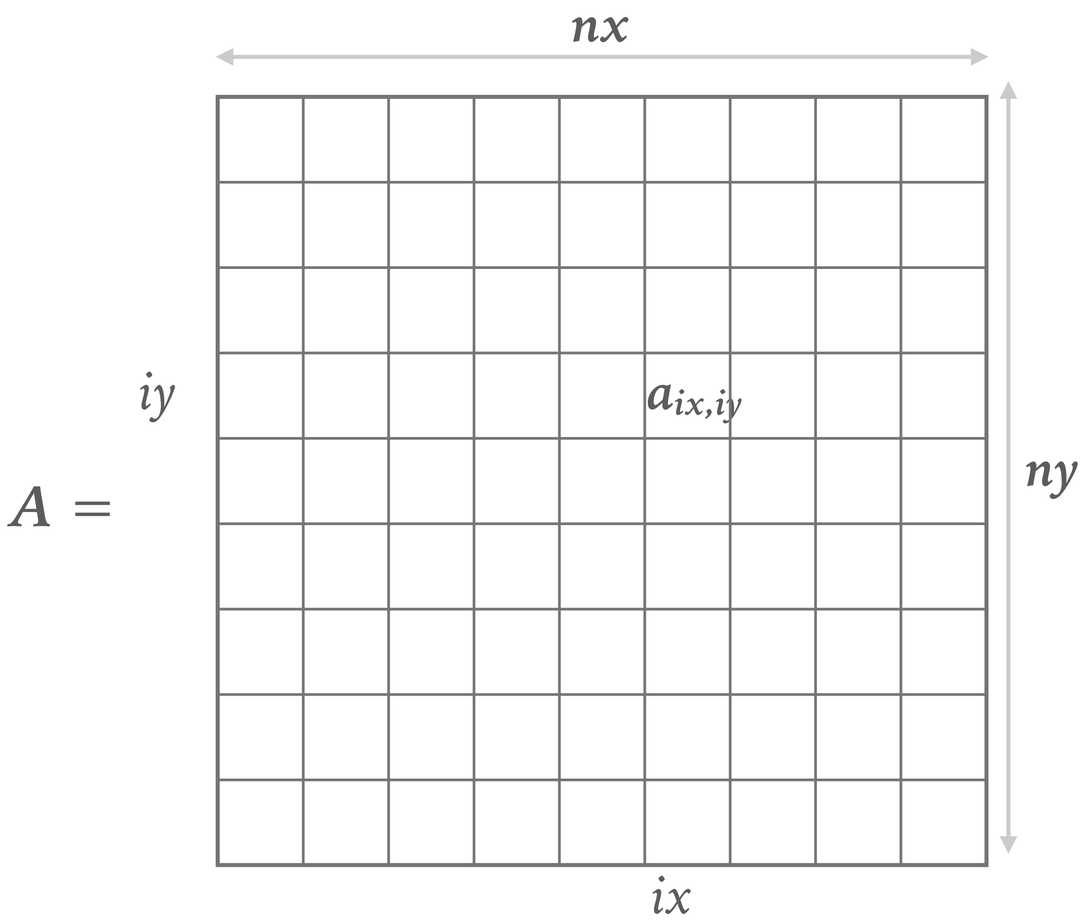


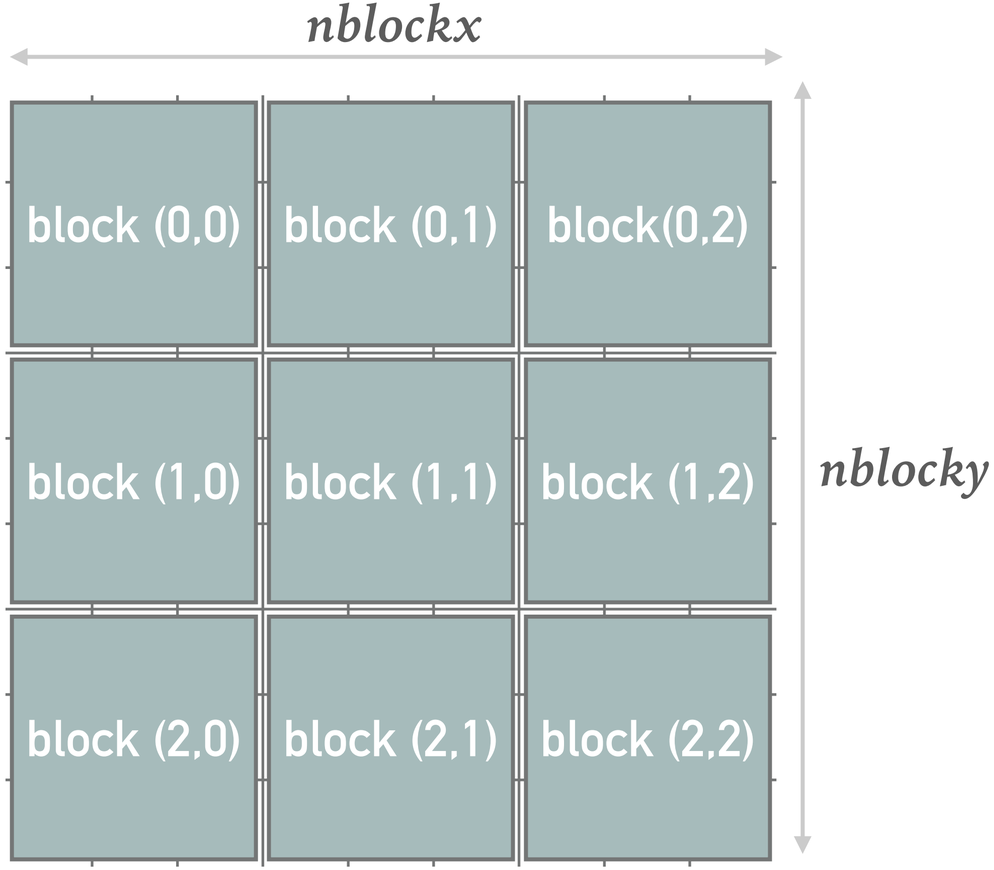



```
int idx = iy*nx + ix
```

```
dim3 block(dimx, dimy, dimz);
```

MATRIX ADDITION – MULTIDIMENSIONAL BLOCKS






```
idx = iy*nx + ix
```

```
dim3 grid(nblockx, nblocky, nblockz);
```

MATRIX ADDITION – MULTIDIMENSIONAL BLOCKS

threadIdx.x (y,z)

- what is the thread index in a block in x,y,z directions

blockIdx.x (y,z)

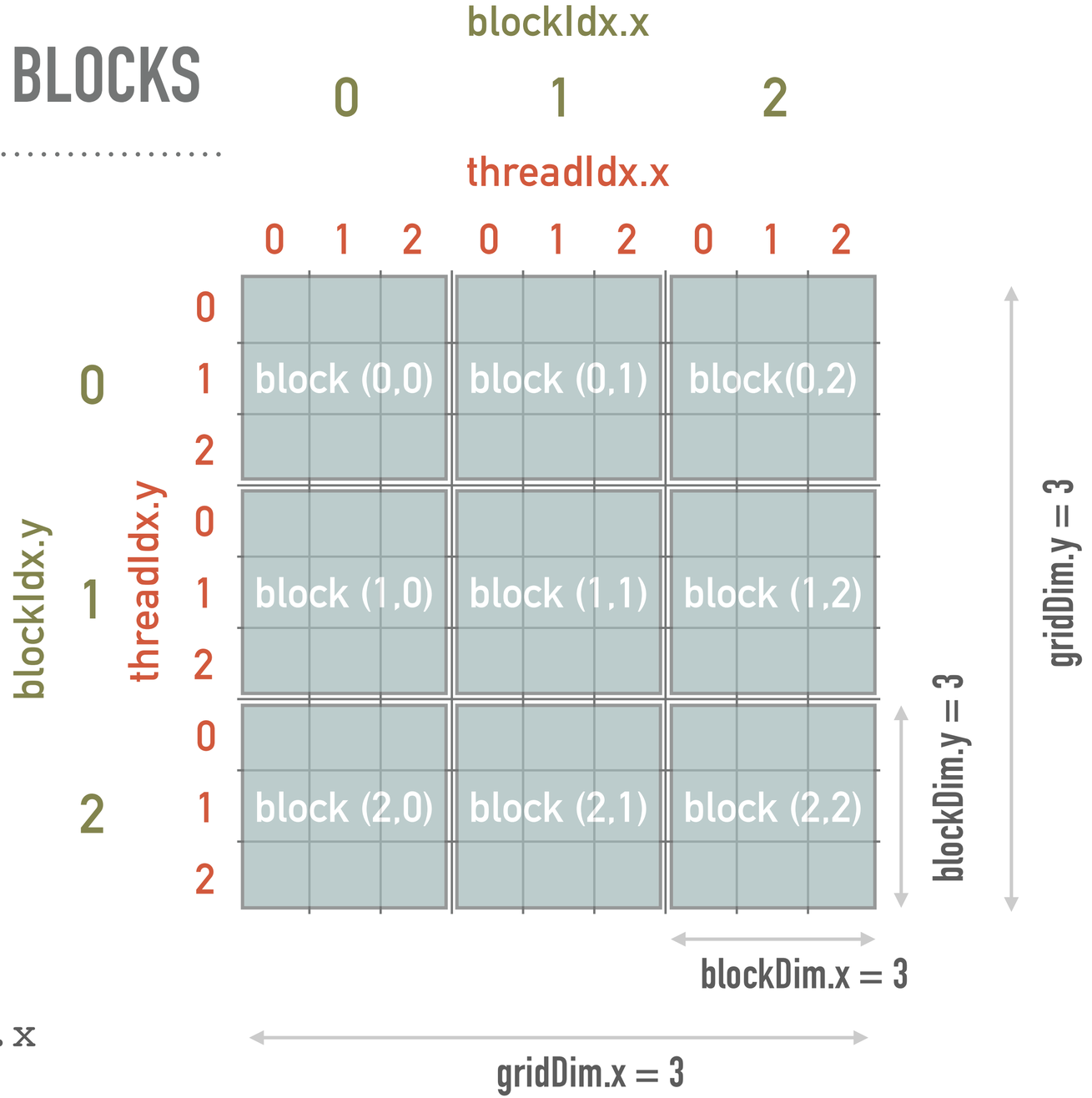
- what is the block index in a grid in x,y,z directions

blockDim.x (y,z)

- how many threads in a block in x,y,z directions

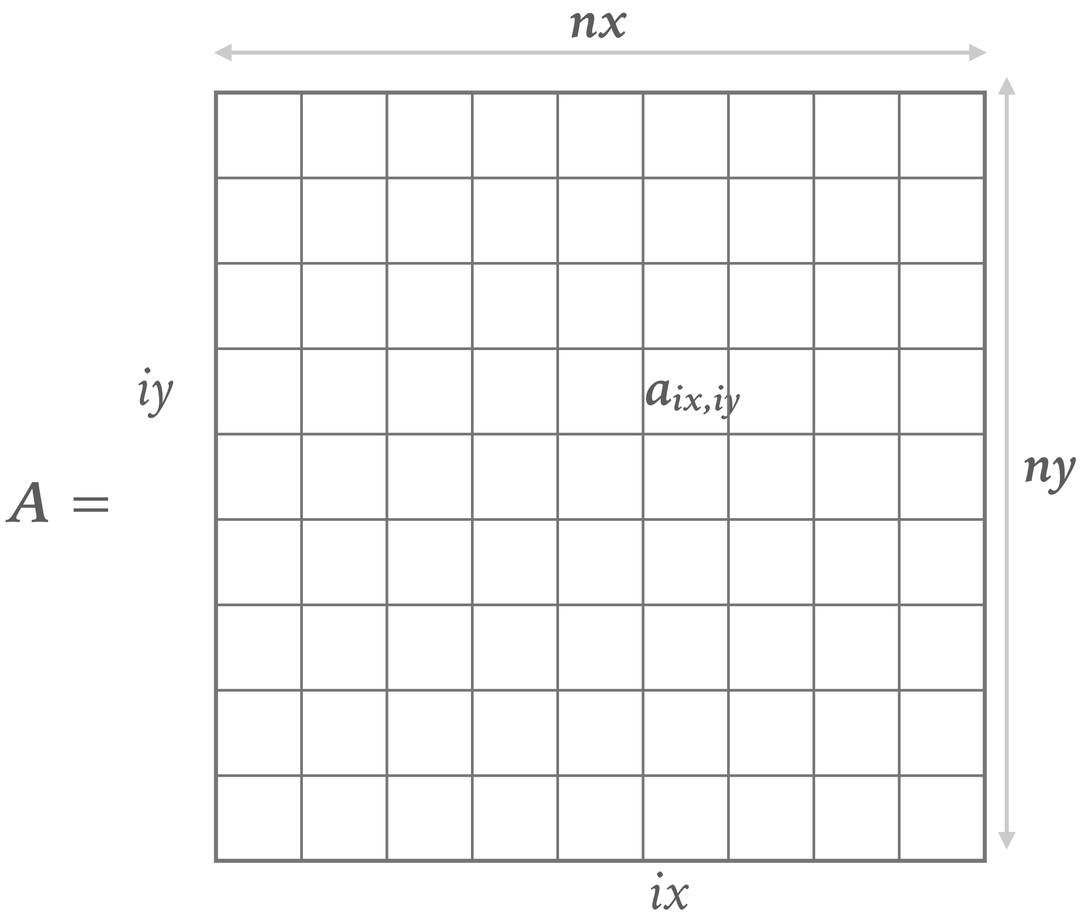
gridDim.x (y,z)

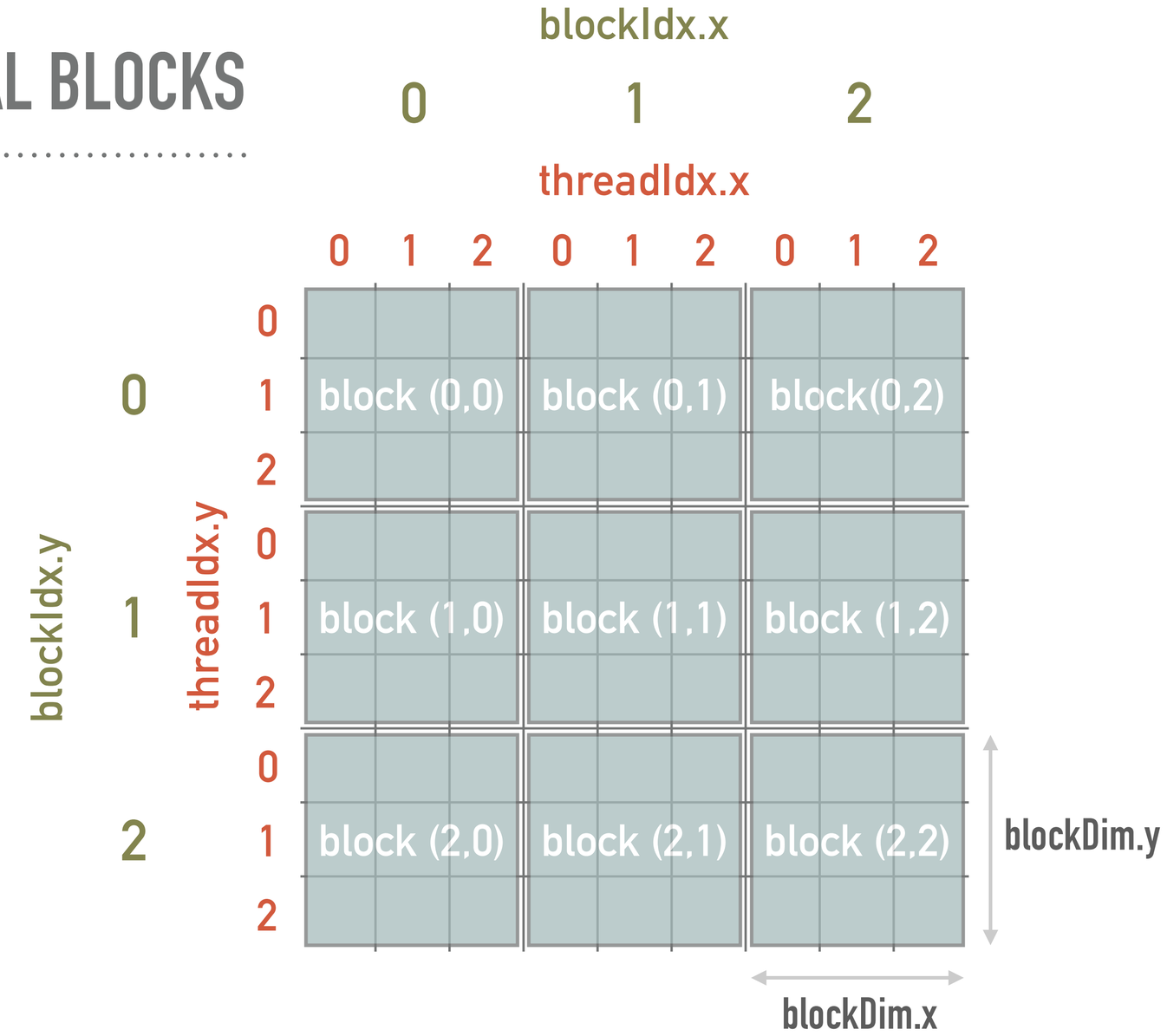
- how many blocks in a grid in x,y,z directions




```
ix = blockIdx.x*blockDim.x + threadIdx.x
iy = blockIdx.y*blockDim.y + threadIdx.y
```

MATRIX ADDITION – MULTIDIMENSIONAL BLOCKS






```
idx = iy*nx + ix
```

```
ix = blockIdx.x*blockDim.x + threadIdx.x
iy = blockIdx.y*blockDim.y + threadIdx.y
```

MATRIX ADDITION - MULTIDIMENSIONAL BLOCKS

```
__global__ void sumMatrixOnHost(float *A, float *B, float *C,
                                int nx, int ny);
{
    for (int iy = 0; iy<ny; iy++)
        for(int ix = 0; ix<nx; ix++){
            idx = iy*nx + ix;
            C[idx] = A[idx] + B[idx];
        }
    }
}
```

MATRIX ADDITION - MULTIDIMENSIONAL BLOCKS

```
__global__ void sumMatrixOnGPU2D(float *A, float *B, float *C,
                                 int nx, int ny);
{
    int ix = blockIdx.x*blockDim.x + threadIdx.x;
    int iy = blockIdx.y*blockDim.y + threadIdx.y;
    int idx = iy*nx+ix;

    if(ix<nx && iy<ny)
        C[idx] = A[idx] + B[idx];
}
```