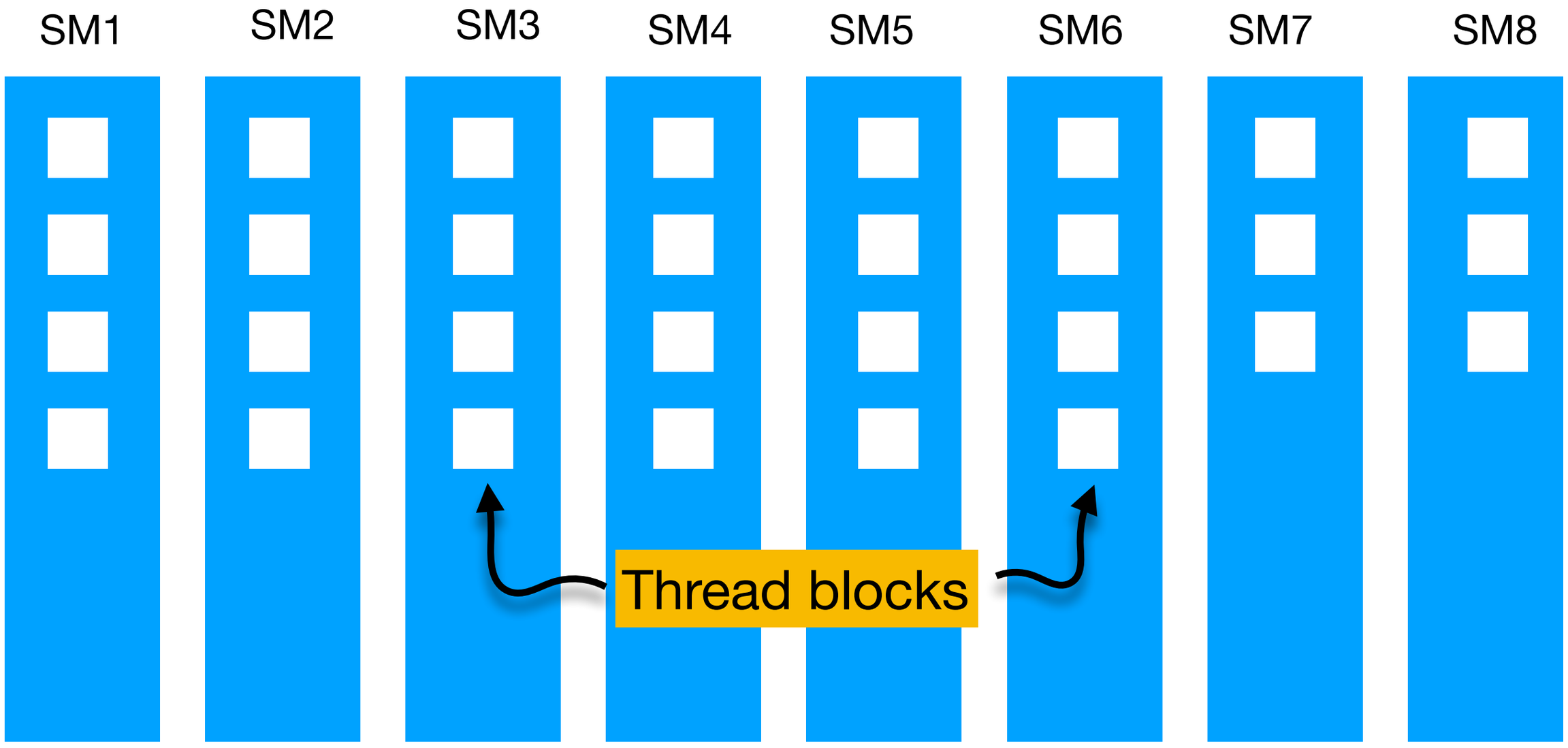
# GPU Execution Model

# GPU Execution Model

The heart of the GPU is the “Streaming Multiprocessor” or SM.



# GPU Execution Model

| GPUs | Compute Capability | Number of SMs | Blocks per SM | Warps per SM | Threads per SM |
|---|---|---|---|---|---|
| GeForce GTX 680 | 3.0 (Kepler) | 8 | 16 | 64 | 2048 |
| Tesla K20c | 3.5 (Kepler) | 13 | 16 | 64 | 2048 |
| GeForce GTX Titan X | 5.2 (Maxwell) | 24 | 32 | 64 | 2048 |

- The SM is responsible for partitioning registers and shared memory among the thread blocks
- Thread blocks are scheduled in *warps*, or groups of 32 threads

# CPU Memory Access

```
void copymat_host_x(int m, int n, int* A, int *B)
{
    int ix,iy,idx;
    for(iy = 0; iy < n; iy++)
        for(ix = 0; ix < m; ix++)
        {
            idx = iy*m + ix;
            B[idx] = A[idx];
        }
}
```

```
void copymat_host_y(int m, int n, int* A, int *B)
{
    int ix,iy,idx;
    for(ix = 0; ix < m; ix++)
        for(iy = 0; iy < n; iy++)
        {
            idx = iy*m + ix;
            B[idx] = A[idx];
        }
}
```

# GPU Execution Model

```
__global__ void copymat_x(int m, int n, int* A, int *B)
{
    int idx, ix;
    int iy = threadIdx.y + blockIdx.y*blockDim.y;
    if (iy < n)
        for(ix = 0; ix < P; ix++) {
            idx  = iy*m + ix;
            B[idx]   = A[idx];
        }
}
```

```
__global__ void copymat_y(int m, int n, int* A, int *B)
{
    int ix = threadIdx.x + blockIdx.x*blockDim.x;
    int idx, iy;
    if (ix < m)
        for(iy = 0; iy < P; iy++) {
            idx  = iy*m + ix;
            B[idx]   = A[idx];
        }
}
```