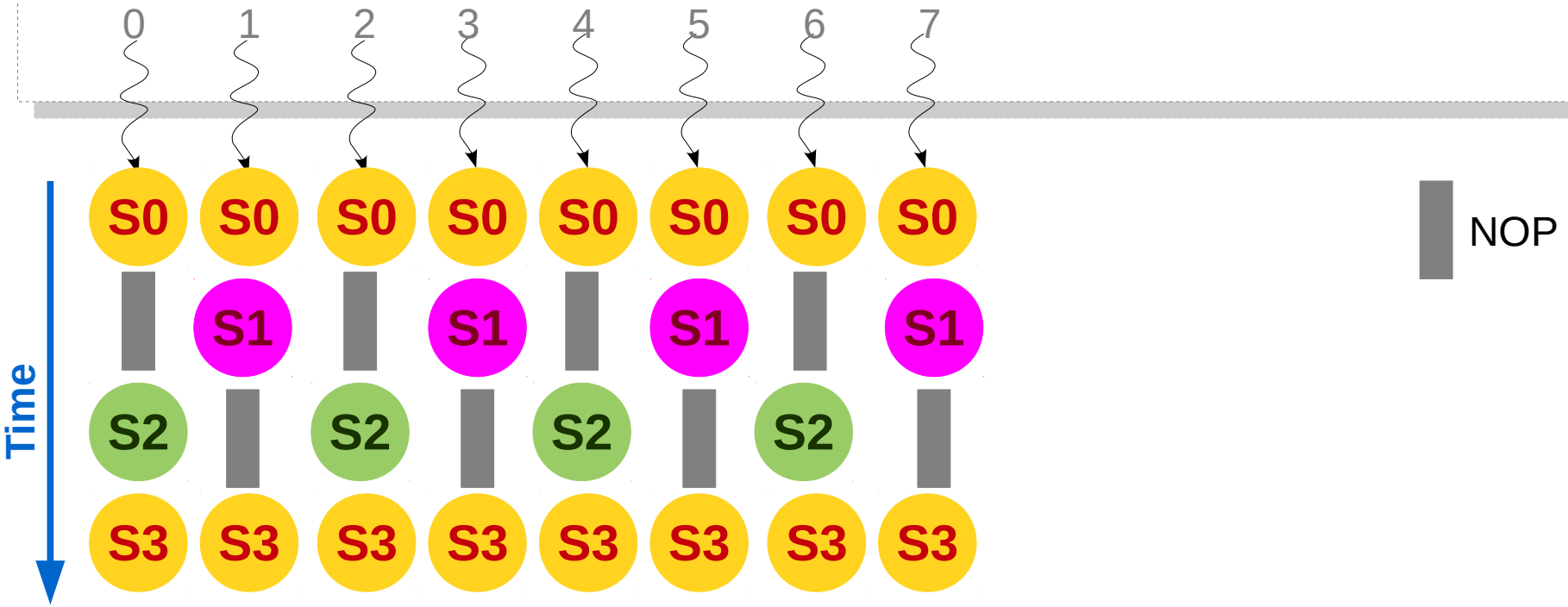
CUDA Programming

Recap

```
__global__ void dkernel(unsigned *vector, unsigned vectorsize) {
    int id = blockIdx.x * blockDim.x + threadIdx.x;
    if (id % 2) vector[id] = id;
    else vector[id] = vectorsize * vectorsize;
      vector[id]++;
}
```

S0 S1 S2 S3



Classwork

- Rewrite the following program fragment to remove thread-divergence.

```
assert(x == y || x == z || x == w );
if (x == y) x = z + w;
else if( x == z ) x = w + y;
else x = y + z;
```

```
assert(x == y || x == z || x == w );
x = y + z + w – x;
```

Classwork

- How many steps does warp threads take to execute?

```
__global__ void dkernel(unsigned *vector, unsigned vectorsize) {
    int id = blockIdx.x * blockDim.x + threadIdx.x;
    if ( id <= 0  ){
        vector[id] = 0;
        for (int i=1;i<=100;i++){
                vector[id] += i;
        }
    }
    else{
        vector[id] = 1;
    }
}
```

Classwork

- How many steps does warp threads take to execute?

```
__global__ void dkernel(unsigned *vector, unsigned vectorsize) {
    int id = blockIdx.x * blockDim.x + threadIdx.x;
    if ( id <= 0  ){
        vector[id] = (101*100) / 2;
    }
    else{
        vector[id] = 1;
    }
}
```

Classwork

- How many steps does warp threads take to execute?

```
__global__ void dkernel(unsigned *vector, unsigned vectorsize) {
    int id = blockIdx.x * blockDim.x + threadIdx.x;
    vector[id] = ( 1 + ( (-id)>>31) ) * ( ((101*100) / 2) - 1 ) + 1 ;
}
```

Thread-Divergence

```
__global__ void dkernel(unsigned *vector, unsigned vectorsize) {
    unsigned id = blockIdx.x * blockDim.x + threadIdx.x;
    switch (id) {
    case 0: vector[id] = 0;                    break;
    case 1: vector[id] = vector[id];           break;
    case 2: vector[id] = vector[id - 2];       break;
    case 3: vector[id] = vector[id + 3];       break;
    case 4: vector[id] = 4 + 4 + vector[id];   break;
    case 5: vector[id] = 5 - vector[id];       break;
    case 6: vector[id] = vector[6];            break;
    case 7: vector[id] = 7 + 7;                break;
    case 8: vector[id] = vector[id] + 8;       break;
    case 9: vector[id] = vector[id] * 9;       break;
      }
}
```

How many steps will the warp threads take?

Thread-Divergence

```
__global__ void dkernel()
 {
      if (threadidx.x <16)
      {
            printf("Inside If");
            Global_Barrier();
      }
      else if (threadidx >=16)
      {
            printf("Inside else");
            Global_Barrier();
      }
}
```

What is the Output?

Deadlock!!

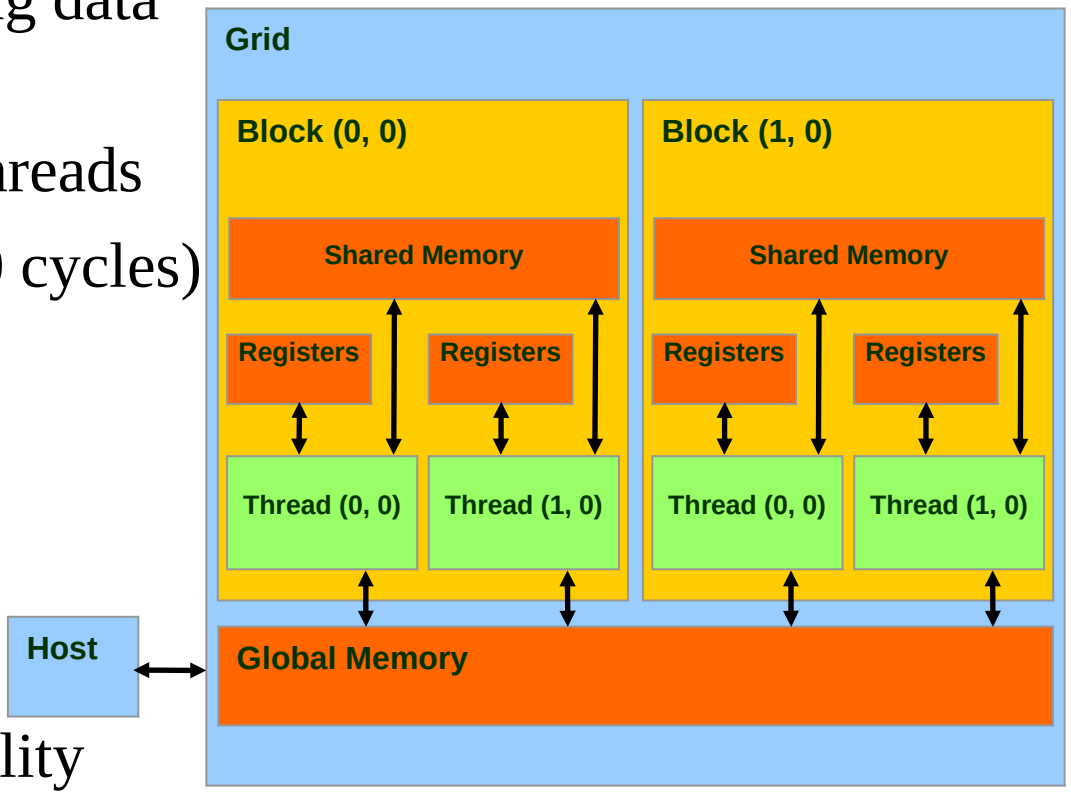
Memory

Agenda

- Computation
- **Memory**
- Synchronization
- Functions
- Support
- Topics

CUDA Memory Model Overview

- **Global / Video memory**
  - Main means of communicating data between host and device
  - Contents visible to all GPU threads
  - Long latency access (400-800 cycles)
  - Throughput ~200 GBPS
- **Texture Memory**
  - Read-only (12 KB)
  - ~800 GBPS
  - Optimized for 2D spatial locality
- **Constant Memory**
  - Read-only (64 KB)



The numbers are typical values.