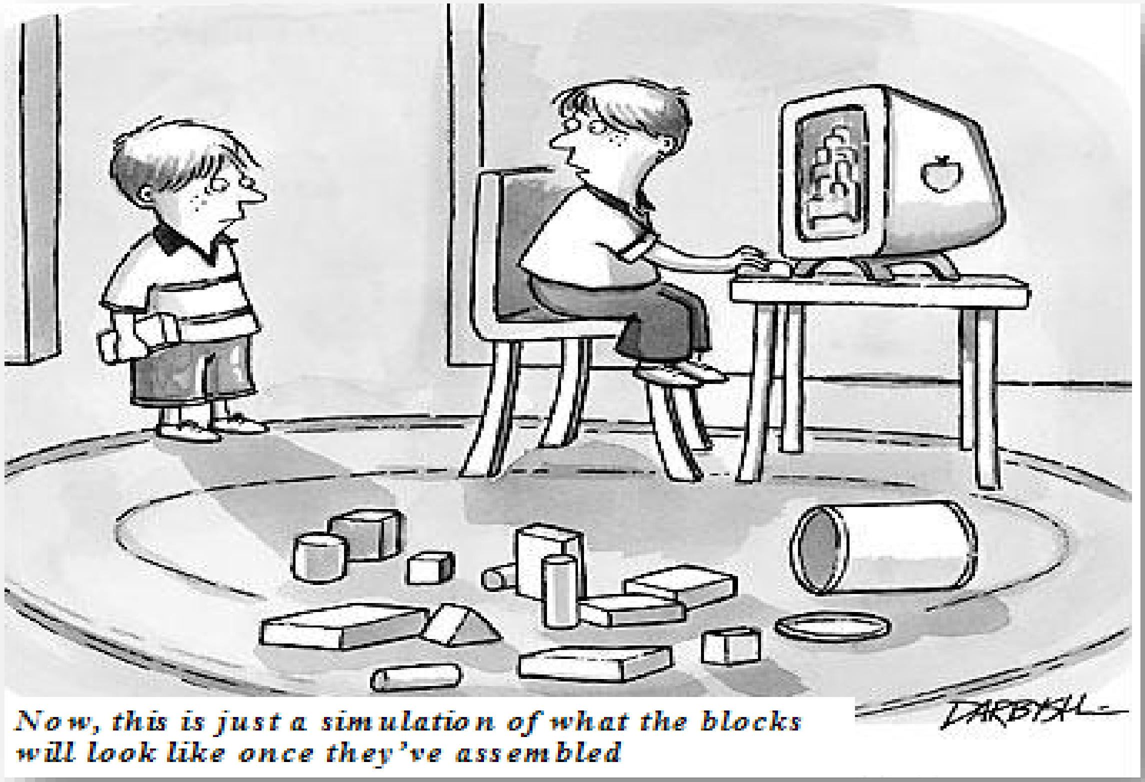
ECE569
Module 18


Now, this is just a simulation of what the blocks will look like once they've assembled
DARBYSHIRE

- Thread Synchronization

Thread Synchronization

- **Fundamental concept in parallel programming**
- **Threads can access each other's results through shared and global memory**
  - They can work together
- **What if a thread reads a result before another thread writes it**
  - Need synchronization

Barrier

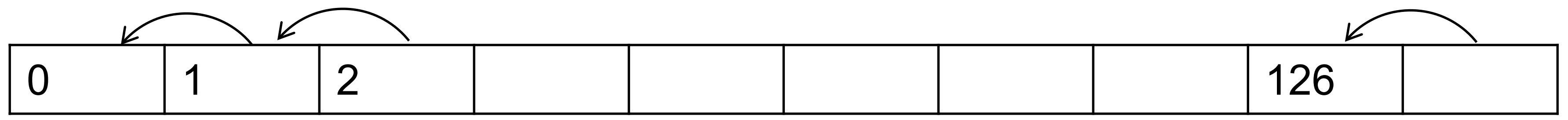
- **Basic thread coordination mechanism**
- **When a thread calls __syncthreads()**
  - Forms a barrier in the thread execution path
  - Holds each thread at calling location until every thread **in the block** reaches that location
    - All threads complete a phase before moving onto next phase
      - Make sure that all threads read neighboring pixel values before those values are updated
    - Used to avoid RAW/WAR hazards when accessing shared or global memory
    - Does not synchronize threads from two different blocks

No one is left behind!

Barrier Example: How many barriers are needed?

Shift elements of array to left by one element

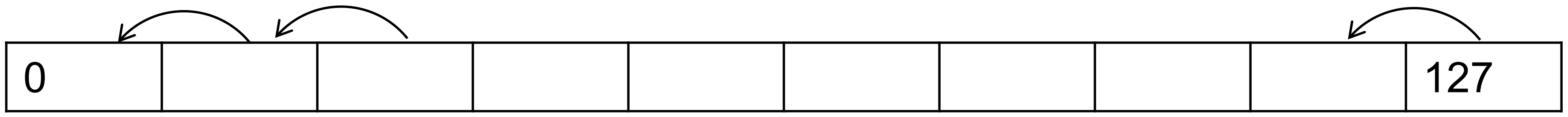
```
:
int idx = threadIdx.x
__shared__ int array[128]
array[idx] = threadIdx.x
// initialize each element to thread index
If (idx <127)
    array[idx] = array[idx+1];
:
```



Barrier Example: How many barriers are needed?

Shift elements of array to left by one element

```
:
int idx = threadIdx.x
__shared__ int array[128]
array[idx] = threadIdx.x
__synchthreads()//make sure all values written
// initialize each element to thread index
If (idx <127)
    array[idx] = array[idx+1];
:
```



Any other barrier(s)?

Barrier Example: How many barriers are needed?

Shift elements of array to left by one element

```
:
int idx = threadIdx.x
__shared__ int array[128]
array[idx] = threadIdx.x
__synchthreads()//make sure all values written
// initialize each element to thread index
If (idx <127)
    array[idx] = array[idx+1];
```

We need to complete reading [idx+1] from all entries before writing into [idx]! How do we achieve this?

Barrier Example: How many barriers are needed?

Shift elements of array to left by one element

```
:
int idx = threadIdx.x
__shared__ int array[128]
array[idx] = threadIdx.x
__synchthreads()//make sure all values written
// initialize each element to thread index
If (idx <127){
   int temp = array[idx+1]; //declare local
   __synchthreads();
    array[idx] = temp;
   __synchthreads();
//last one ensures all write operations are
completed before array is accessed again later
```

Which one will have correct functionality without synchthreads()

```
__global__ void my_function() {
__shared___ int s[1024];
int i=threadIdx.x;
__synchthreads();
s[i]=s[i-1]                                 // 1
__synchthreads();
if(i%2) s[i]=s[i-1];                        // 2
__synchthreads();
s[i] = (s[i-1]+s[i]+s[i-1])/3.0             // 3
printf(“s[%d]=%f\n”, I, s[i]);
__synchtreads;
```

Which one will have correct functionality without synchthreads()

```
__global__ void my_function() {
__shared___ int s[1024];
int i=threadIdx.x;
__synchthreads();
s[i]=s[i-1]                            // 1
__synchthreads();
if(i%2) s[i]=s[i-1];                   // 2
__synchthreads();
s[i] = (s[i-1]+s[i]+s[i-1])/3.0        // 3
printf(“s[%d]=%f\n”, i, s[i]);
__synchreads;
```

```
int temp=s[i-1];
__synchtreads();
s[i]=temp;
```

Which one will have correct functionality without synchthreads()

```
__global__ void my_function() {
__shared___ int s[1024];
int i=threadIdx.x;
__synchthreads();
s[i]=s[i-1]                                  // 1
__synchthreads();
if(i%2) s[i]=s[i-1];                         // 2
__synchthreads();
s[i] = (s[i-1]+s[i]+s[i-1])/3.0              // 3
printf("s[%d]=%f\n", i, s[i]);
__synchtreads;
```

Only evens write
Reads are all
from odds.

Which one will have correct functionality without synchthreads()

```
__global__ void my_function() {
__shared___ int s[1024];
int i=threadIdx.x;
__synchthreads();
s[i]=s[i-1]
__synchthreads();
if(i%2) s[i]=s[i-1];                              // 2
__synchthreads();
s[i] = (s[i-1]+s[i]+s[i-1])/3.0                   // 3
printf("s[%d]=%f\n", i, s[i]);
__synchreads;
```

```
float temp =
(s[i-1]+s[i]+s[i-1])/3.0
__synchtreads();
s[i]=temp;
__synchtreads;                  // 1
printf();
```

Next

- **Global, Shared Memory**
  - Static vs. Dynamic Shared Memory
  - Coalesced Memory
  - Code Review