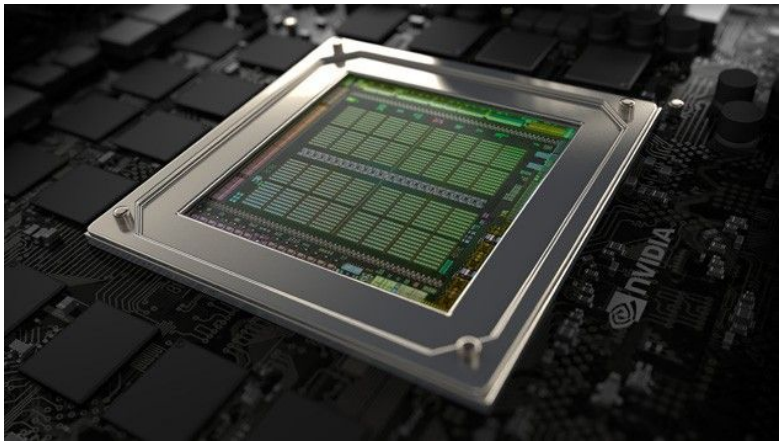
Parallel Computing with GPUs

Wayne Franz
Comp 4510
Nov. 2018

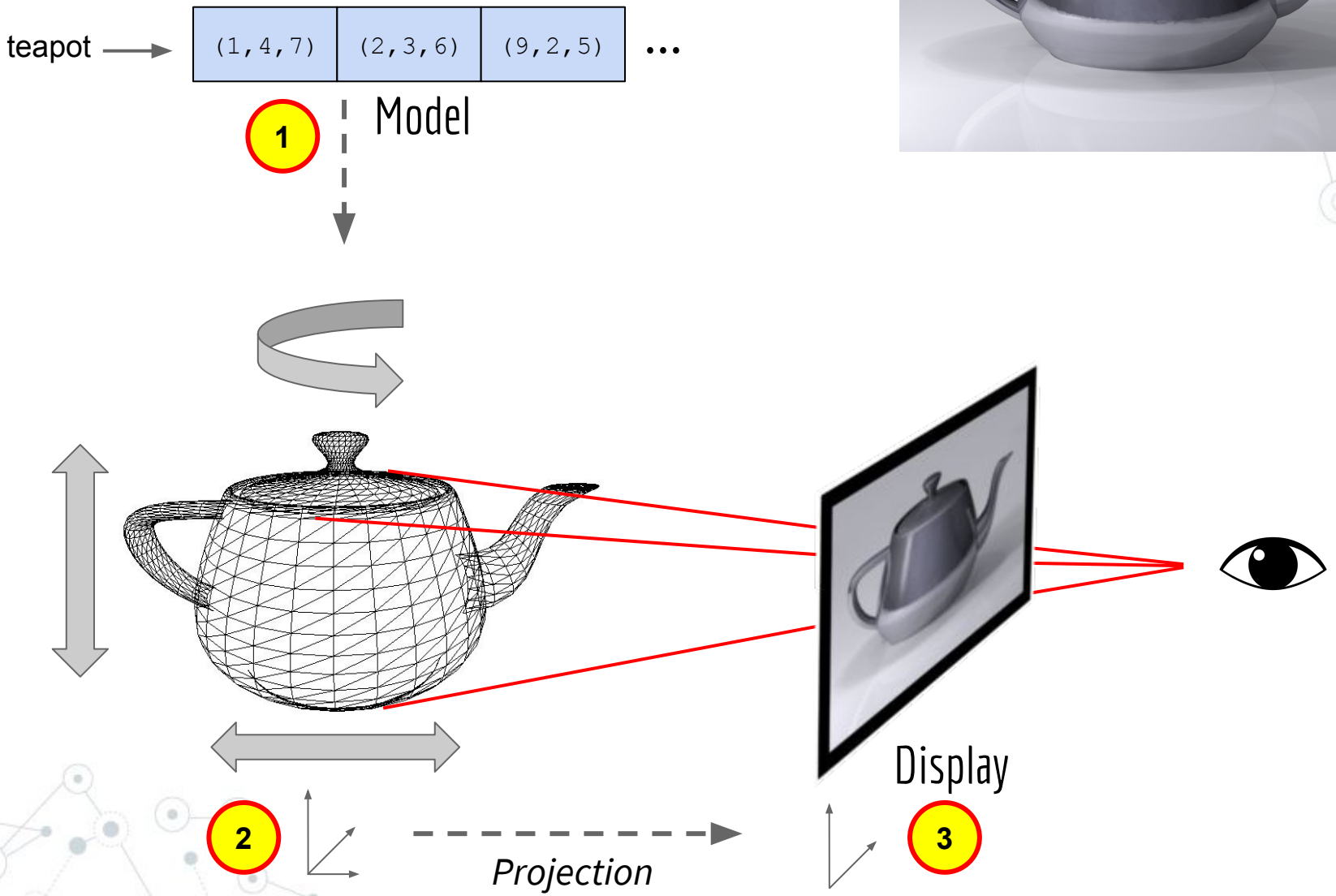
Schedule

Week 1	GPU Fundamentals
1. (Mon)	Introduction to GPUs
2. (Wed)	GPU Architecture
3. (Fri)	Quiz
Week 2	**Solving Problems on the GPU**
4. (Mon)	Programming in CUDA
5. (Wed)	Case Study: Sum Reduction
6. (Fri)	Case Study: Sum Reduction (2)

1. Introduction to GPUs



Graphics Processing

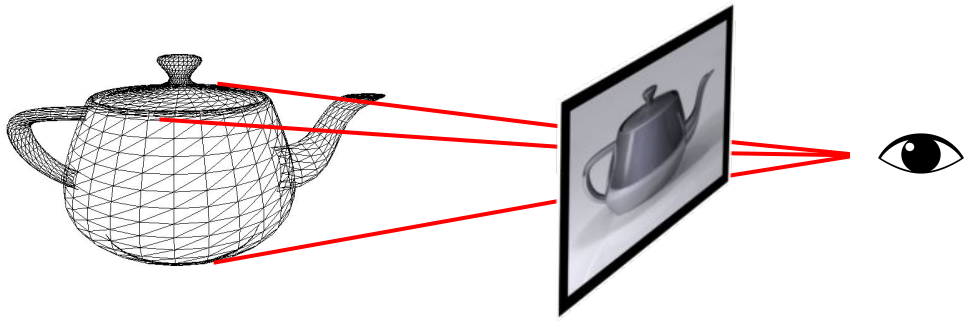


Graphics Processing

- Historically done on the CPU
  - **Problems:**
    - Computationally intensive
    - CPU also does other things
    - Requires a hard deadline: refresh rate
  - **Solution:**
    - Use an accelerator!

Graphics Processing

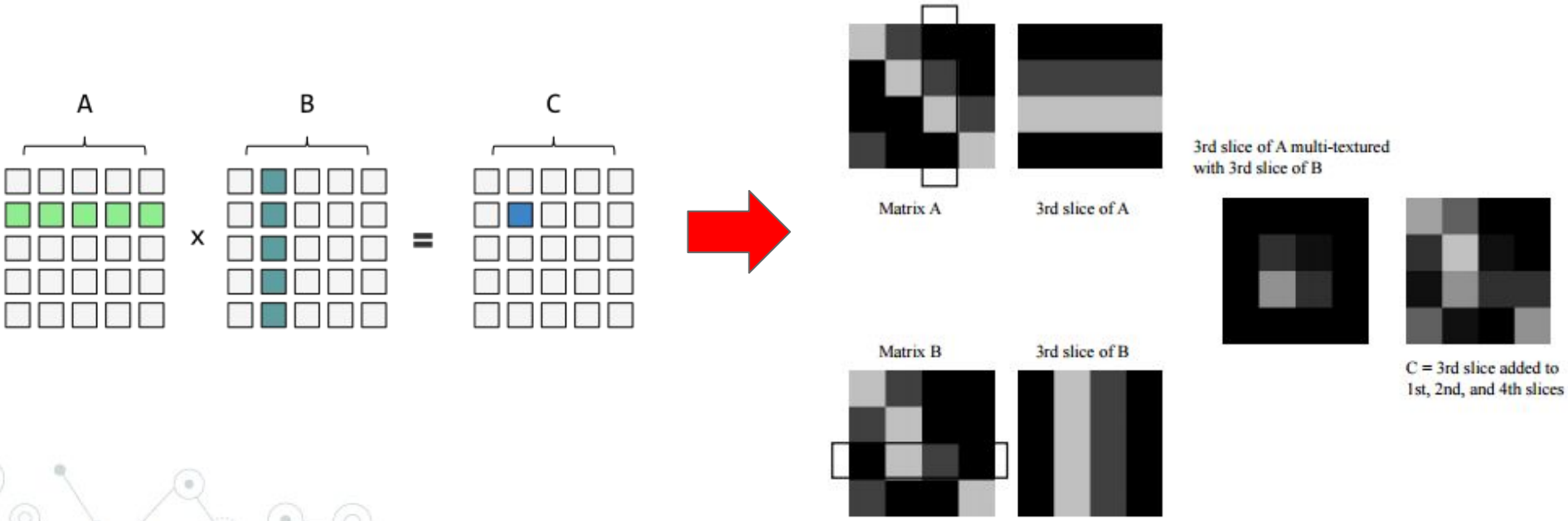
- Many graphics-related tasks are ***data parallel*** problems
  - Same calculation across an entire array/matrix
  - No dependencies between the calculations



- GPUs have evolved to accelerate *data-parallel* computation
  - This shows up in areas other than just graphics...

GPGPU Computing

- General Purpose GPU Computing (GPGPU)
  - Eg. Larson & McCallister, *Fast Matrix Multiplies using Graphics Hardware*, ACM, 2001.



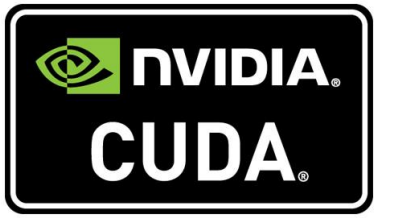
GPGPU Computing

"Compute Unified Device Architecture" (CUDA)

- Nvidia, 2006
- C/C++ with extensions for controlling the GPU
- Can operate on arrays instead of images / vertices
- Finer-grained control over GPU

CUDA®

GPGPU Programming Languages

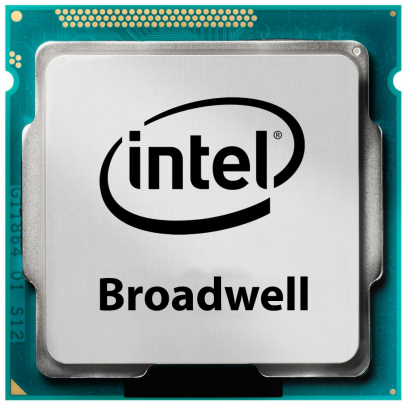


NVidia Devices Only

Cross Platform

Why should we care?

Device	Theoretical Max. Throughput (SP FP)
Intel Xeon (Broadwell) E5-2699 (v4)	~774 GFLOPS
Nvidia GeForce GTX 1080	~8228 GFLOPS

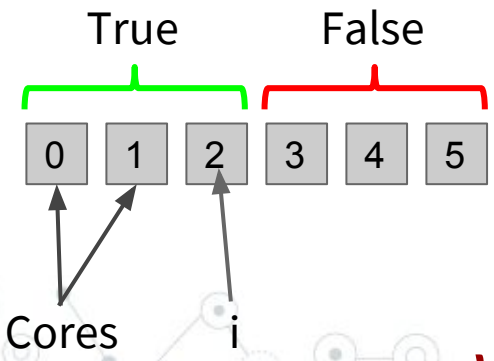


But there are trade-offs!

What we sacrifice for throughput

- GPUs are SIMD devices
- SIMD performance can suffer under:
  a. scattered (irregular) memory accesses [more on this later…]
  b. branch instructions (if statements):

SIMD: all cores execute the same instruction across an array.



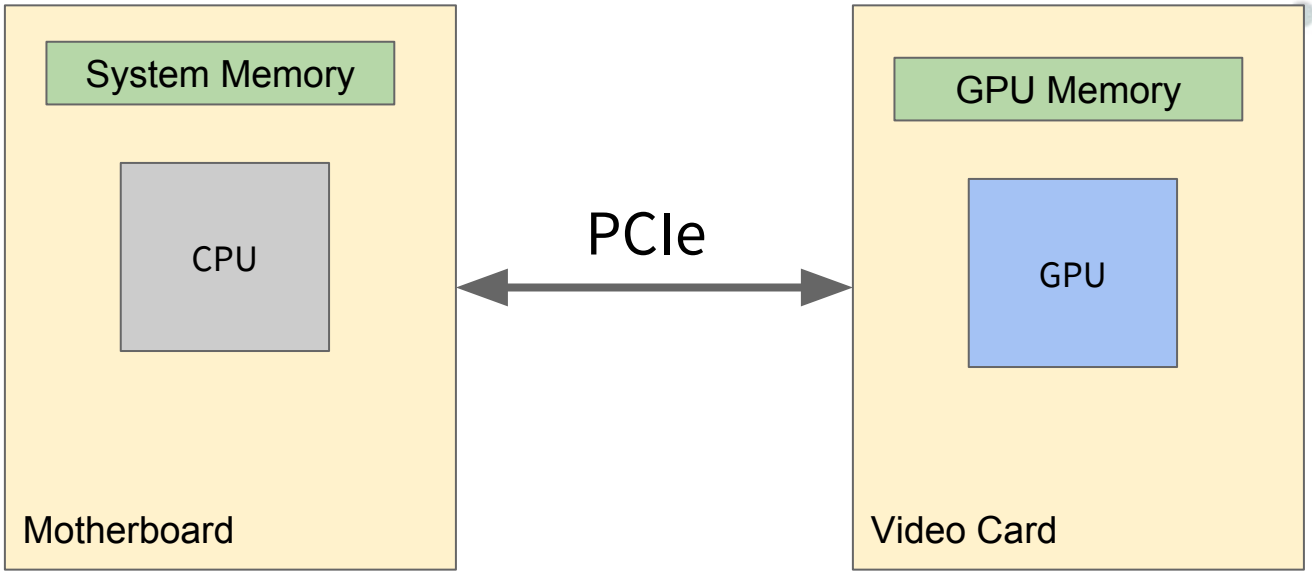

```
if (array[i] < 5) {
   ...
}
else {
   ...
}
```

1. Turn off "False" cores and do this

2. Turn off "True" cores and do this

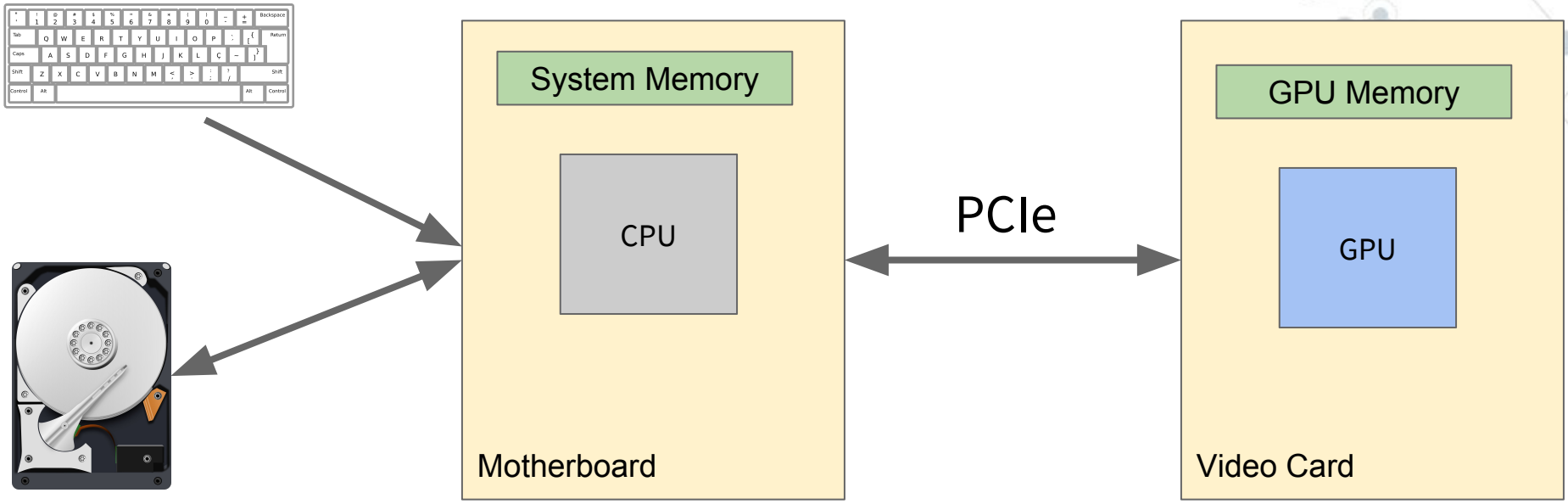
We have to execute both sides in sequence!
This reduces performance.

What we sacrifice for throughput



- GPUs have a separate memory system
  - Before computing, we must transfer our data from CPU to GPU memory…
  - …and then transfer the result back
  - This can take a long time!

Other Consequences

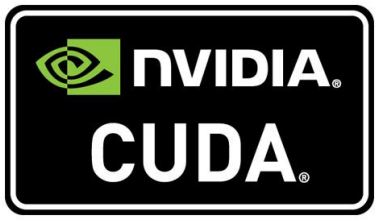


- GPUs cannot do I/O (eg. read/write files)
  - the only way to communicate with the outside world is data transfers over PCIe

Should I parallelize on the CPU or the GPU?

- Depends on the characteristics of the problem!
  - Good questions to ask:

- How many "if" statements / scattered reads does the algorithm require?
- How large will the data transfers be?
- How much of the algorithm is data parallel computation?
  - i.e. the type the GPU is good at accelerating
  - want time savings here > loss from previous two points



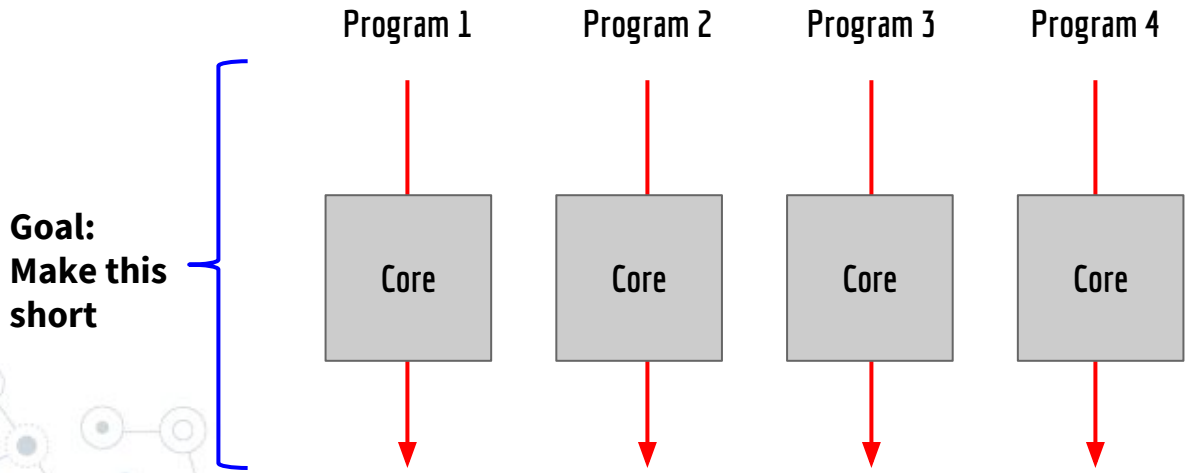
Schedule

Week 1	GPU Fundamentals
1. (Mon)	Introduction to GPUs
2. (Wed)	GPU Architecture
3. (Fri)	Quiz
Week 2	**Solving Problems on the GPU**
4. (Mon)	Programming in CUDA
5. (Wed)	Case Study: Sum Reduction
6. (Fri)	Case Study: Sum Reduction (2)

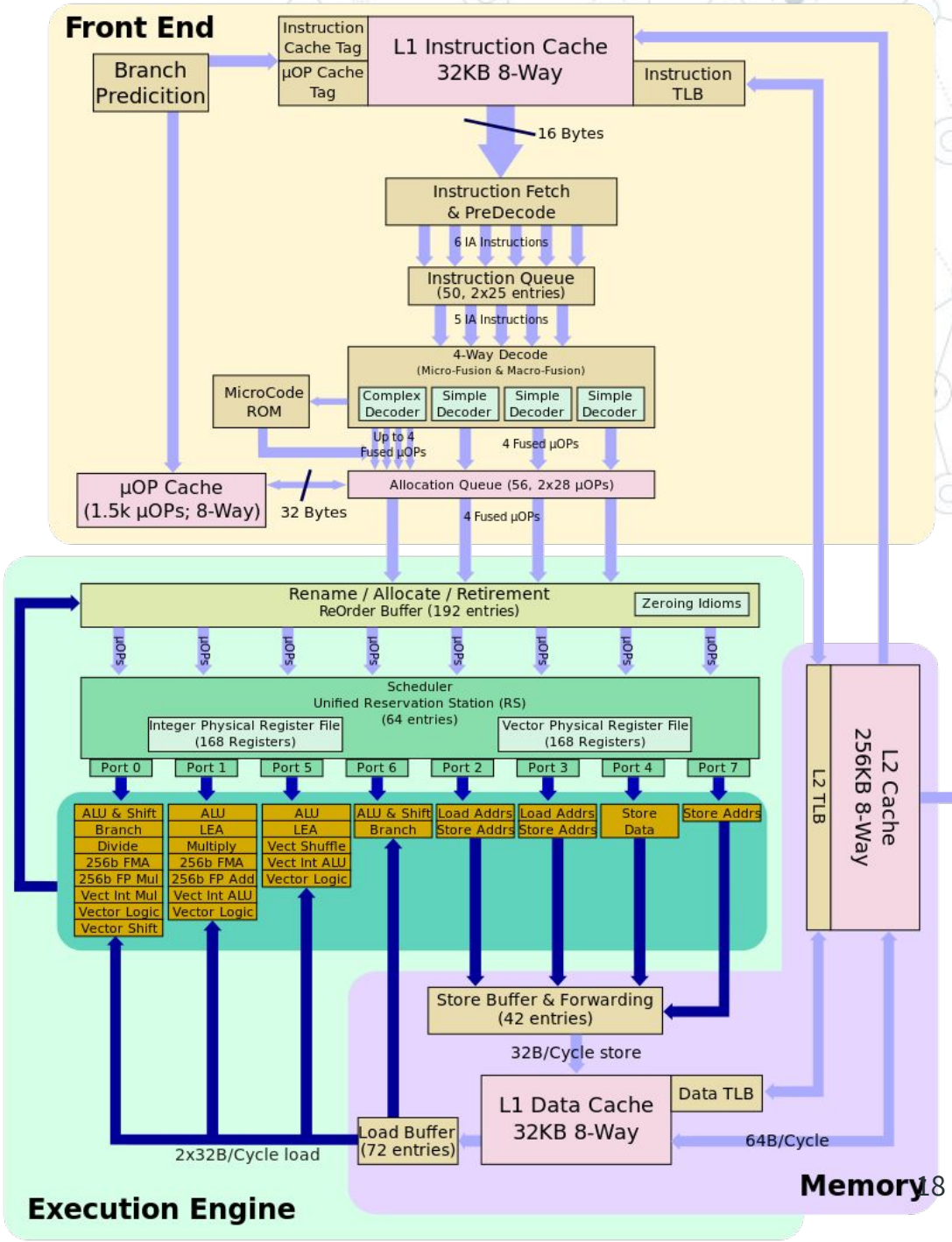
2. GPU Architecture

What's in a core?

- Modern CPU cores
  - Like multiple "independent" sub-processors
  - Tailored for *high-frequency execution of multiple, independent tasks*
- Small number of large, complex cores:

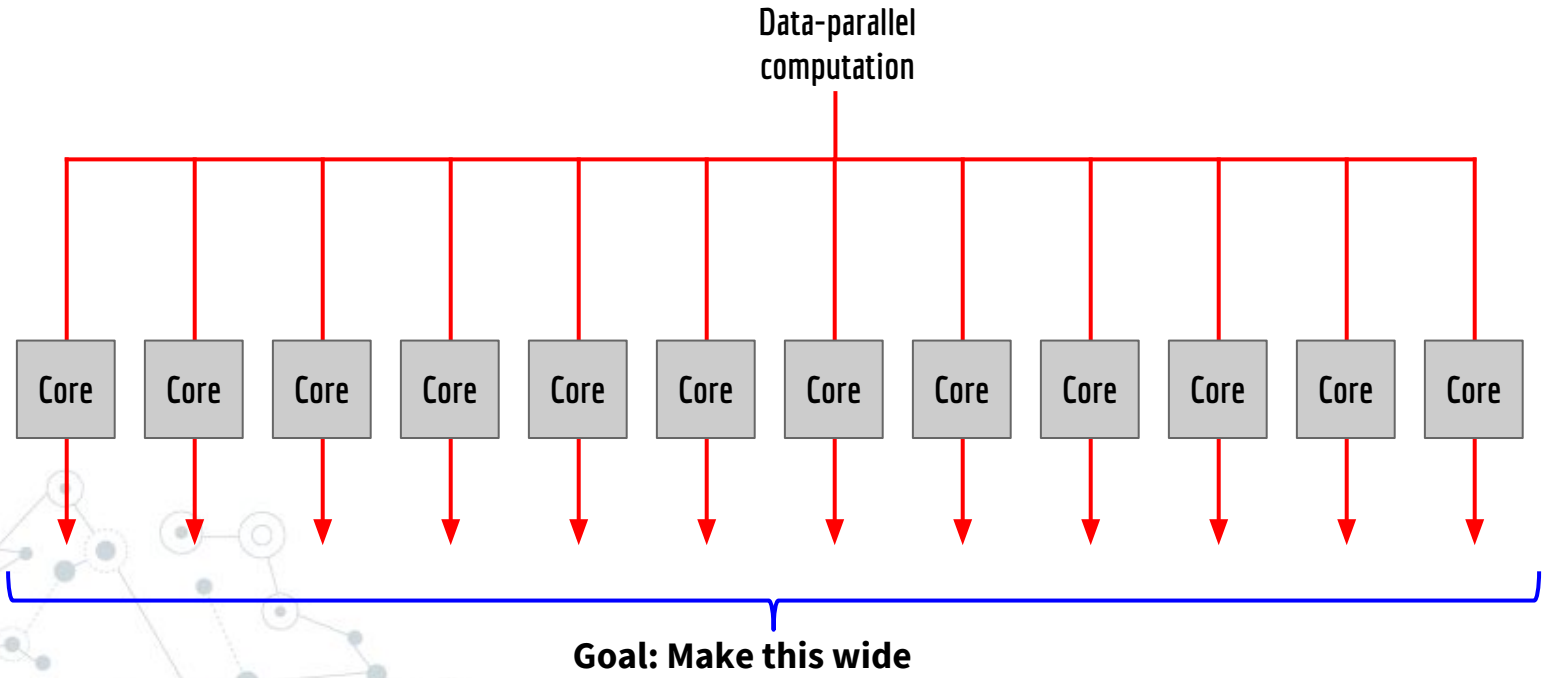


Intel Broadwell

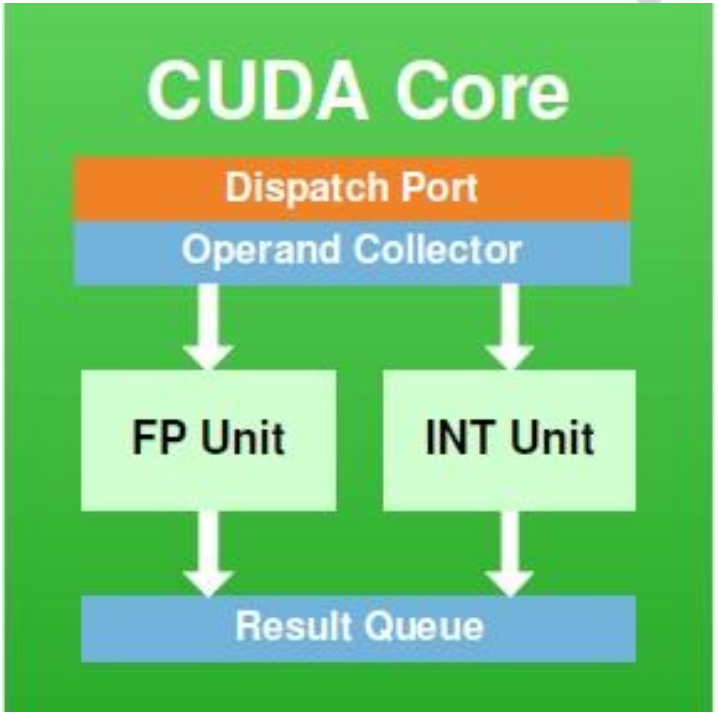


What's in a core?

- GPU cores
  - Not much inside!
  - Tailored for exploiting the *maximum amount of parallelism in a single task*
- Large number of small, simple cores:

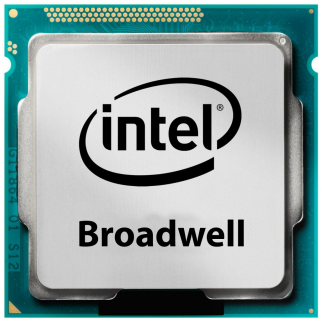


Nvidia GTX 1080



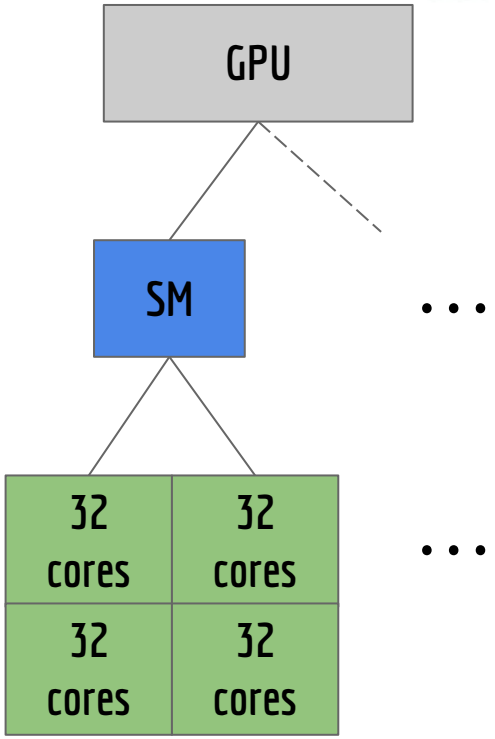
How many cores?

	Cores	Clock Rate
Intel Xeon (Broadwell) E5-2699 (v4)	22	2.2 - 3.6 GHz
Nvidia GTX 1080	2560	1.6 - 1.7 GHz



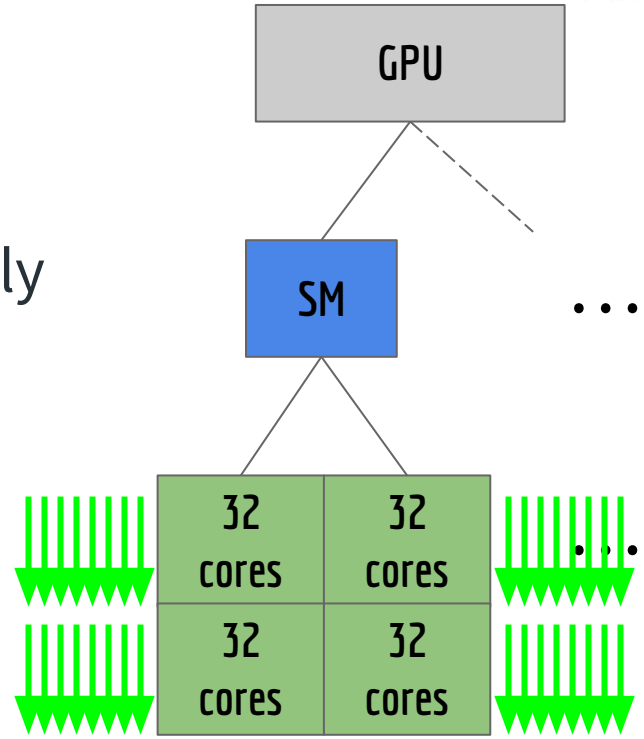
GPU Core organization

- Cores are grouped together
  - Groups of 32
  - Cores in a group perform the *same instruction* in lockstep
- Streaming Multiprocessors (SMs)
  - The “instruction control unit”
  - Controls **4 groups** of 32 cores
    - Different groups may execute *different instructions*

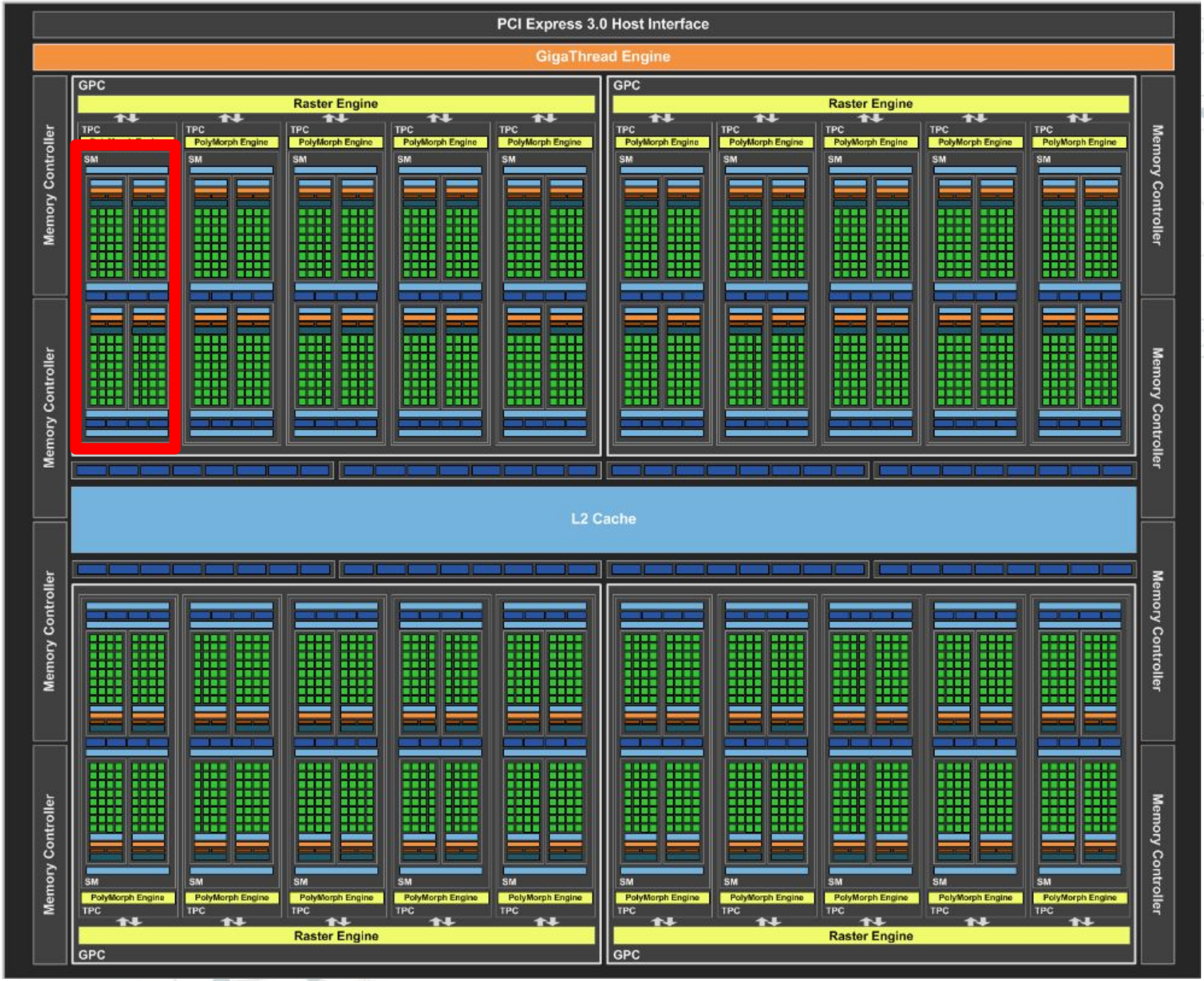


Hardware Threads

- Hardware threads are run in groups of 32
  - "***Warp***"
- Each SM has 4 x 32 cores
  - Can run 4 warps simultaneously



(This is the top half. The bottom half is identical.)


PCI Express 3.0 Host Interface
GigaThread Engine
GPC
Raster Engine
TPC
PolyMorph Engine
SM
Memory Controller
L2 Cache

Memory System

	Memory Bandwidth	Caches
Intel Xeon (Broadwell) E5-2699 (v4)	76.8 GB/s	L1: 64 KB (per core) L2: 256 KB (per core) L3: 55 MB (per CPU)
Nvidia GTX 1080	320 GB/s	L1: 48 KB (per 128 cores) L2: 2 MB (per GPU)

Note: This is "global memory" (there are several different kinds!)