EE382N (20): Computer Architecture - Parallelism and Locality
Fall 2011

Lecture 17 – GPUs (II)

Mattan Erez

The University of Texas at Austin

Recap

Streaming model

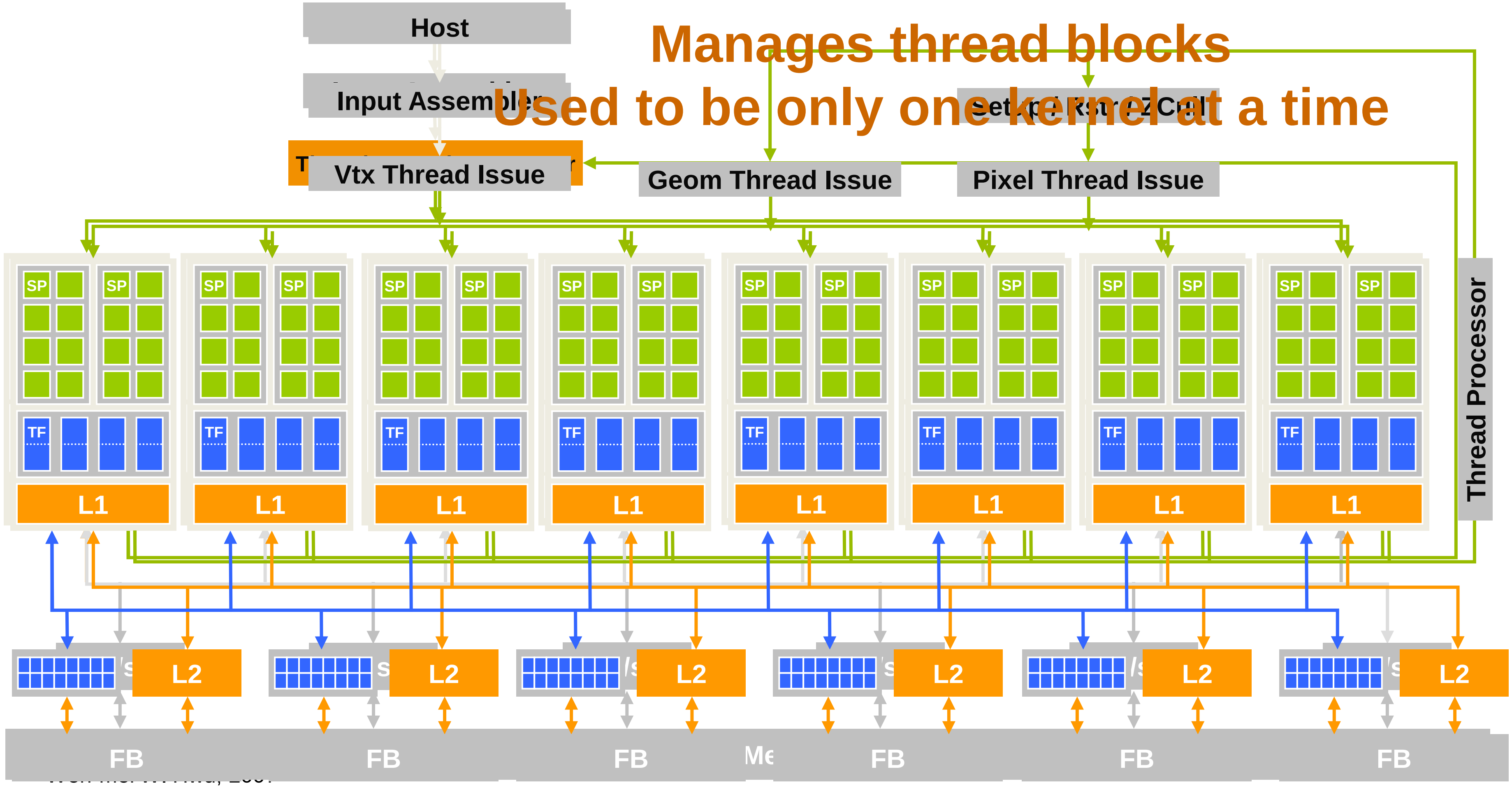
1. Use many “slimmed down cores” to run in parallel

2. Pack cores full of ALUs (by sharing instruction stream across groups of fragments)
   - Option 1: Explicit SIMD vector instructions
   - Option 2: Implicit sharing managed by hardware

3. Avoid latency stalls by interleaving execution of many groups of fragments
   - When one group stalls, work on another group

Make the Compute Core The Focus of the Architecture

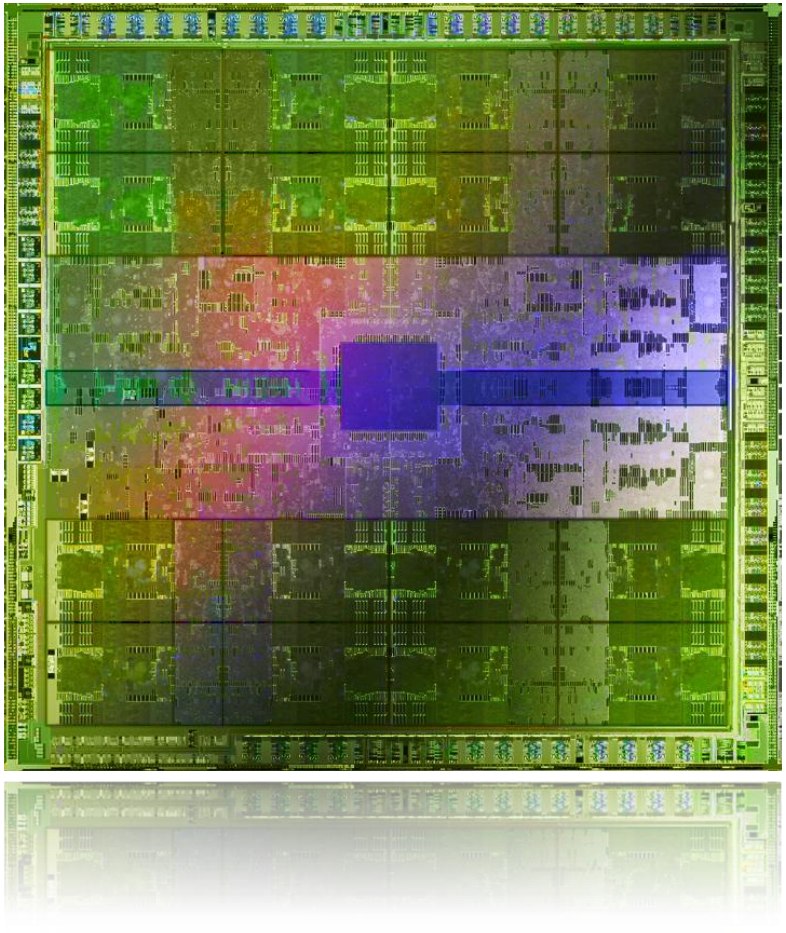
- The future of GPUs is programmable processing
- Processors execute computing threads
- So – build the architecture around the processor
- Alternative operating mode specifically for computing



Manages thread blocks

Used to be only one kernel at a time

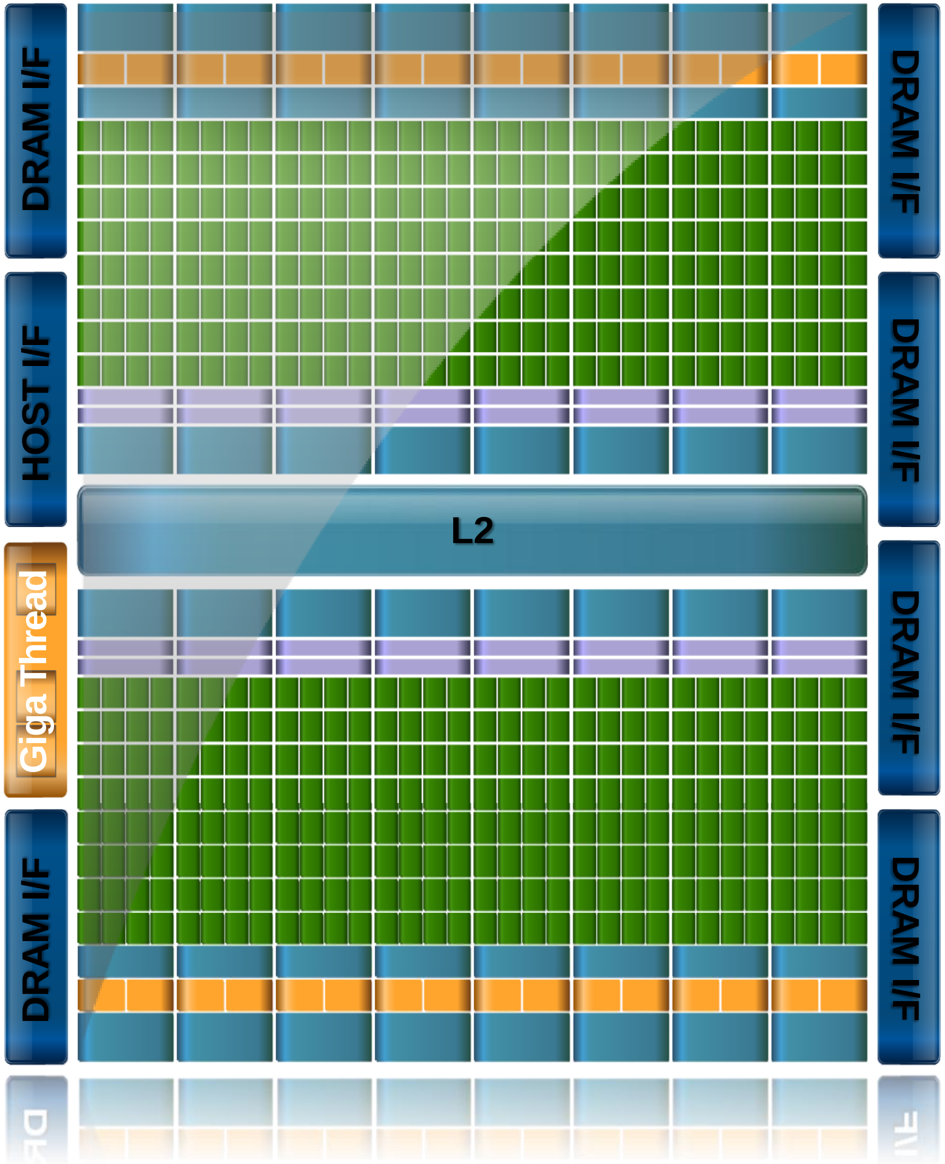
Next-Gen GPU Architecture: *Fermi*



- 3 billion transistors
- Over 2x the cores (512 total)
- ~2x the memory bandwidth
- **L1 and L2 caches**
- 8x the peak DP performance
- ECC
- C++
- **Announced Sept. 2009**

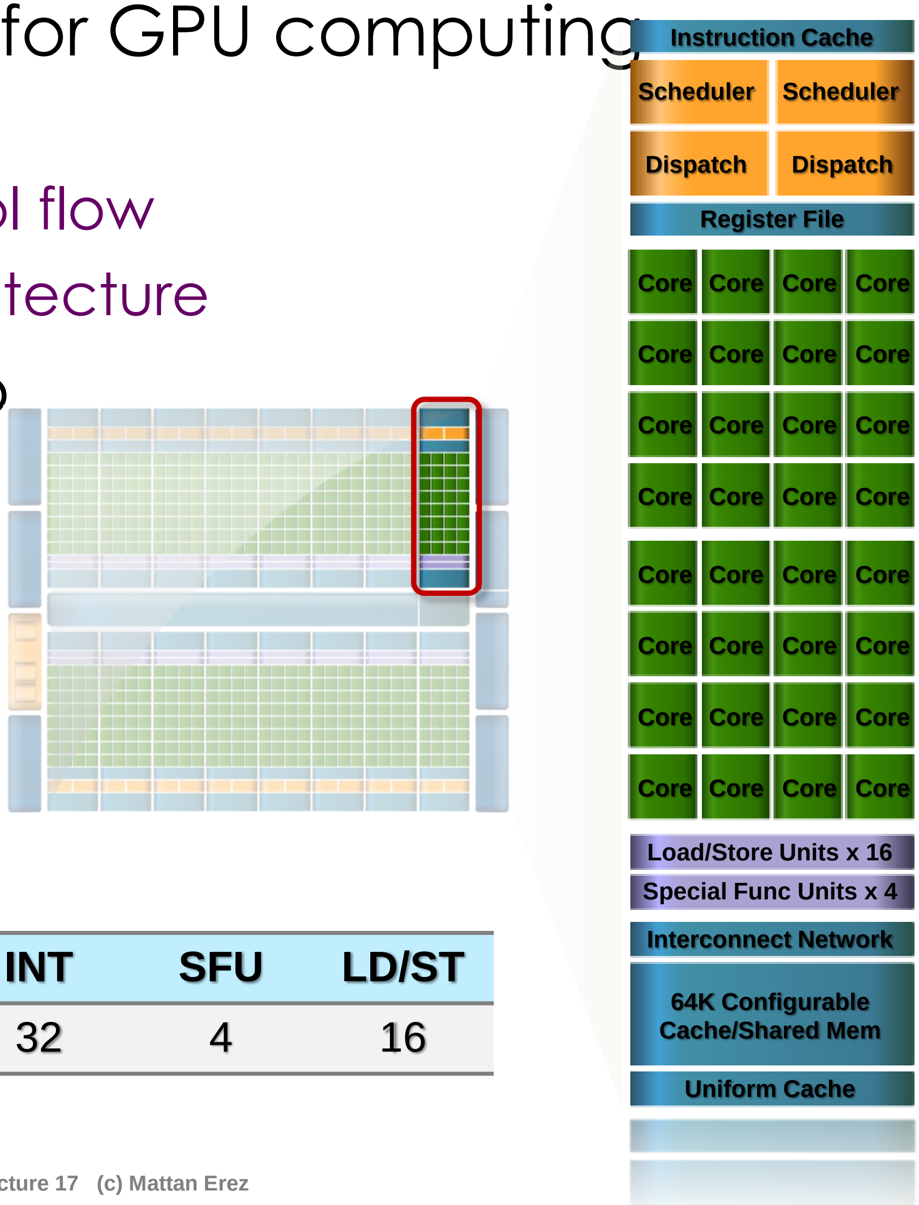
Fermi Focus Areas

- Expand performance sweet spot of the GPU
  - Caching
  - Concurrent kernels
  - FP64
  - 512 cores
  - GDDR5 memory
- Bring more users, more applications to the GPU
  - C++
  - Visual Studio Integration
  - ECC



Streaming Multiprocessor (SM)

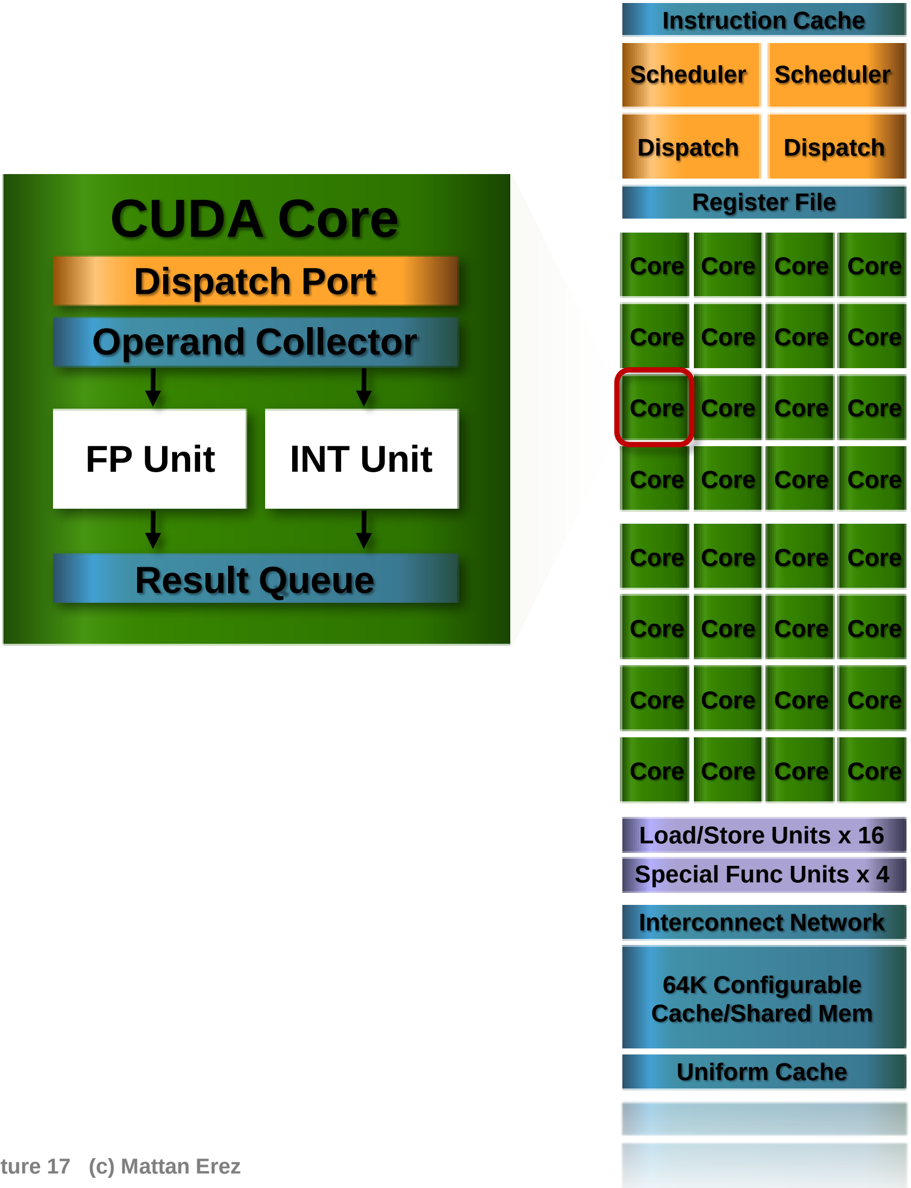
- Objective – optimize for GPU computing
  - New ISA
  - Revamp issue / control flow
  - New CUDA core architecture
- 16 SMs per Fermi chip
- 32 cores per SM (512 total)
- 64KB of configurable L1$ / shared memory



	FP32	FP64	INT	SFU	LD/ST
Ops / clk	32	16	32	4	16

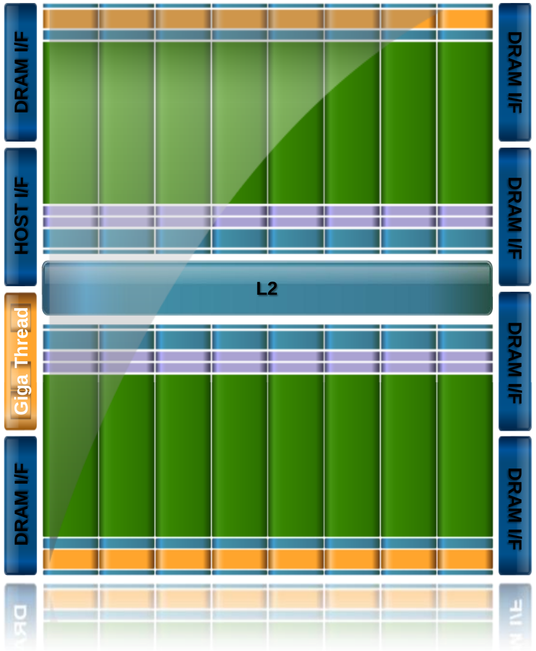
SM Microarchitecture

- New IEEE 754-2008 arithmetic standard
- Fused Multiply-Add (FMA) for SP & DP
- New integer ALU optimized for 64-bit and extended precision ops



Memory Hierarchy

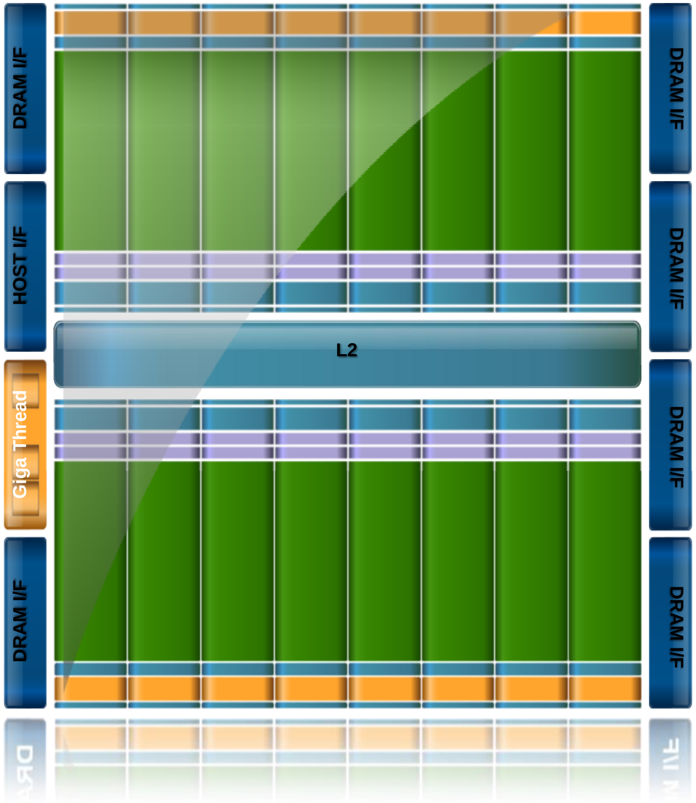
- True cache hierarchy + on-chip shared RAM
  - On-chip shared memory: good fit for regular memory access
    - dense linear algebra, image processing, …
  - Caches: good fit for irregular or unpredictable memory access
    - ray tracing, sparse matrix multiply, physics …
- Separate L1 Cache for each SM (16/48 KB)
  - Improves bandwidth and reduces latency
- Unified L2 Cache for all SMs (768 KB)
  - Fast, coherent data sharing across all cores in the GPU



Larger, Faster Memory Interface

- GDDR5 memory interface
  - 2x improvement in peak speed over GDDR3

- Up to 1 Terabyte of memory attached to GPU
  - Operate on large data sets

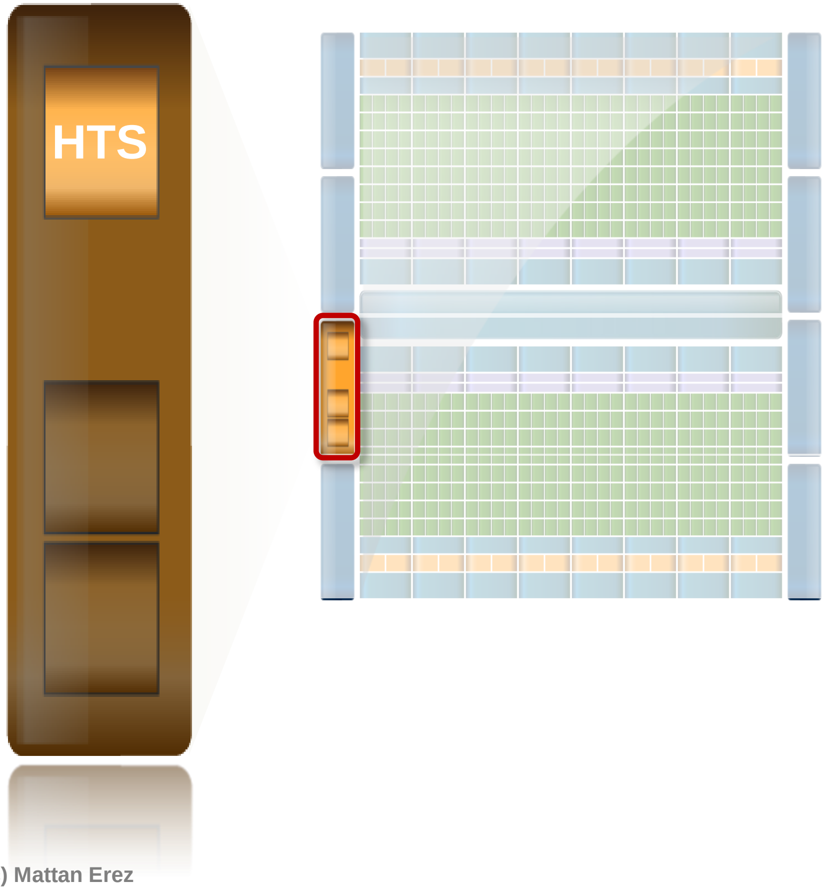


GigaThread™ Hardware Thread Scheduler

- Hierarchically manages tens of thousands of simultaneously active threads

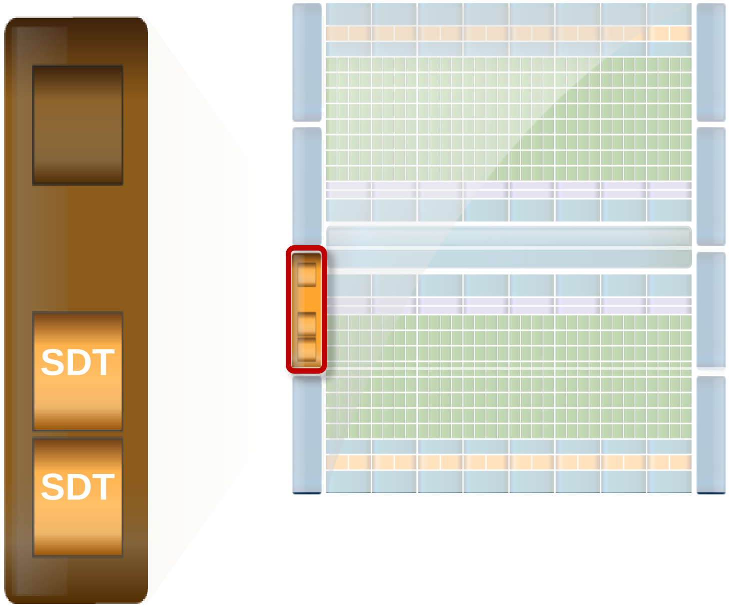
- 10x faster context switching on Fermi

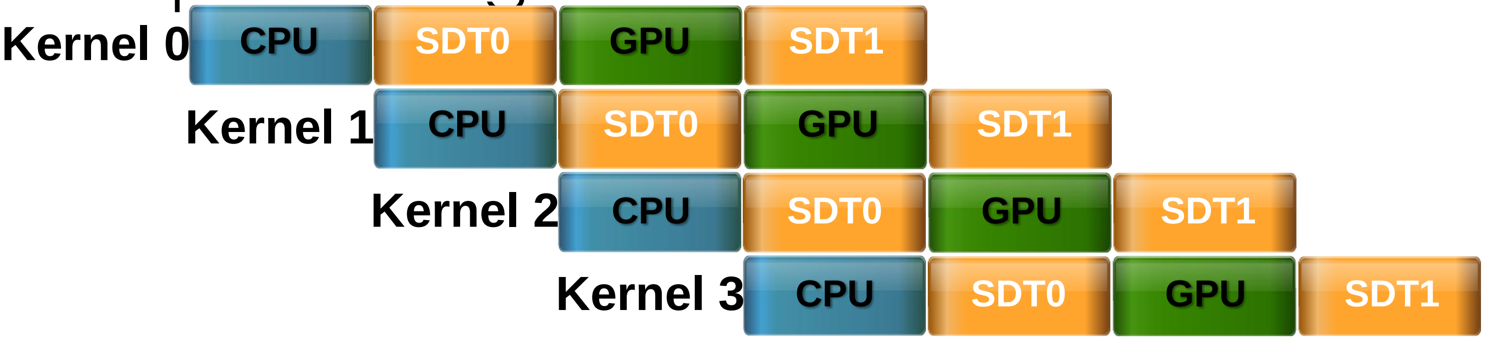
- Overlapping kernel execution



GigaThread Streaming Data Transfer Engine

- Dual DMA engines
- Simultaneous CPU→GPU and GPU→CPU data transfer
- Fully overlapped with CPU/GPU processing





Kernel 0 CPU SDT0 GPU SDT1

Kernel 1 CPU SDT0 GPU SDT1

Kernel 2 CPU SDT0 GPU SDT1

Kernel 3 CPU SDT0 GPU SDT1

Other Capabilities

- ECC protection for DRAM, L2, L1, RF
- Unified 40-bit address space for local, shared, global
- 5-20x faster atomics
- ISA extensions for C++ (e.g. virtual functions)

	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double Precision Floating Point	-	30 FMA ops/clock	256 FMA ops/clock
Single Precision Floating Point	128 MAD ops/clock	240 MAD ops/clock	512 FMA ops/clock
Special Function Units (per SM)	2	2	4
Warp schedulers (per SM)	1	1	2
Shared Memory (per SM)	16 KB	16 KB	Configurable 48/16 KB
L1 Cache (per SM)	-	-	Configurable 16/48 KB
L2 Cache	-	-	768 KB
ECC Memory Support	-	-	Yes
Concurrent Kernels	-	-	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit

NVIDIA Tesla GPUs Power 3 of Top 5 Supercomputers

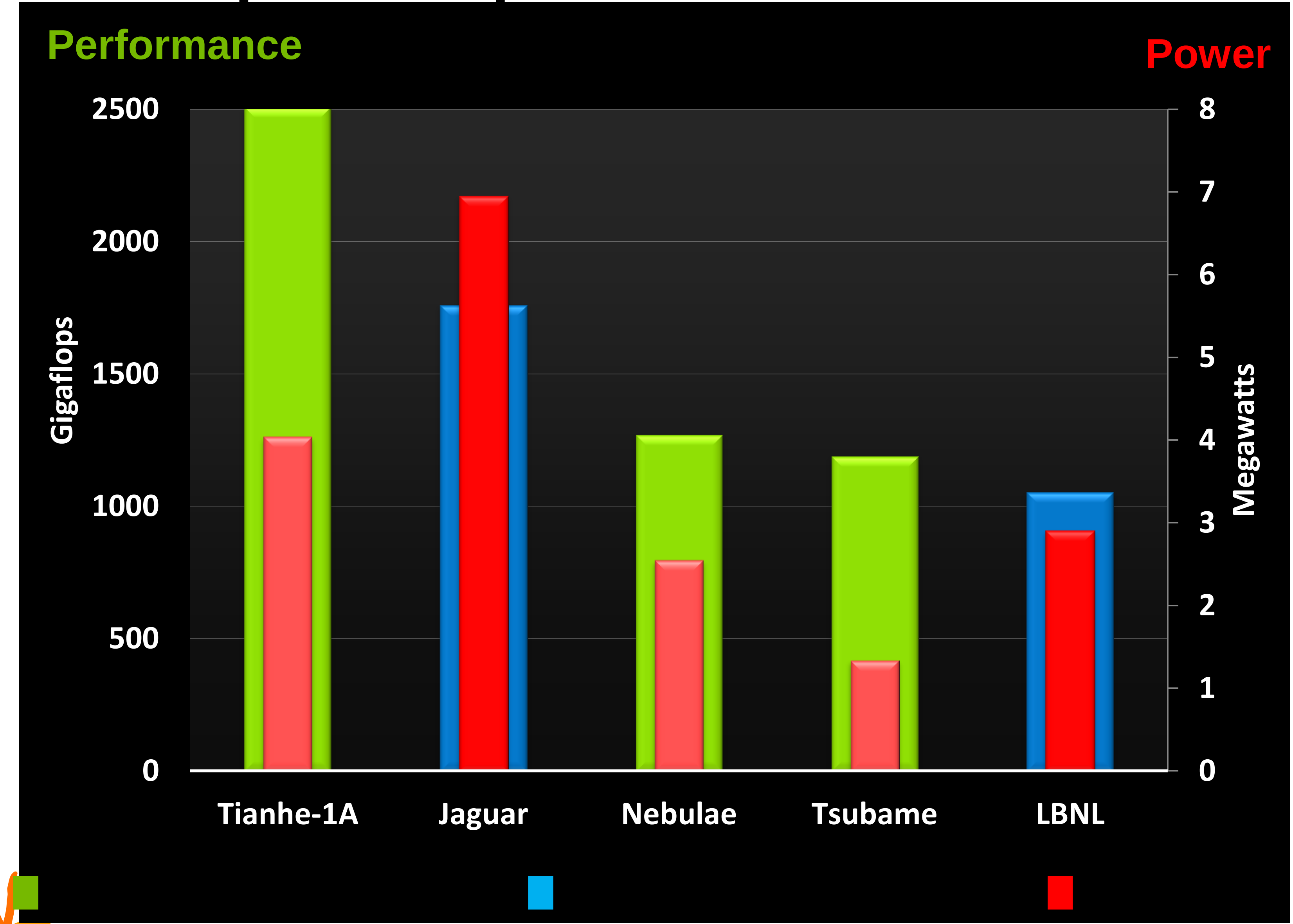
7168 Tesla GPUs 2.5 PFLOPS

4650 Tesla GPUs 1.2 PFLOPS

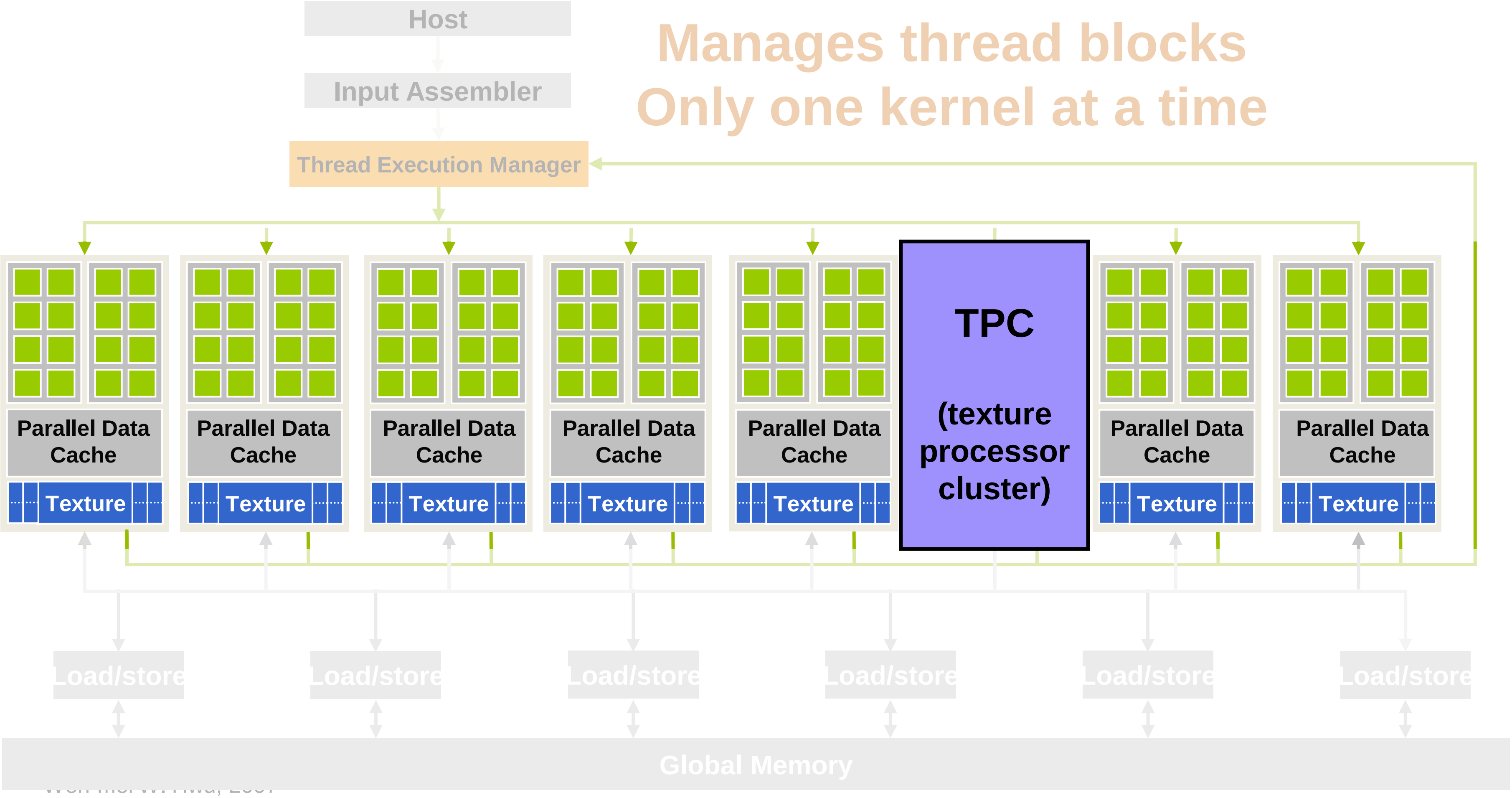
4224 Tesla GPUs 1.194 PFLOPS

8 more GPU accelerated machines in the November Top500

GPU Supercomputers: More Power Efficient



Compute Core



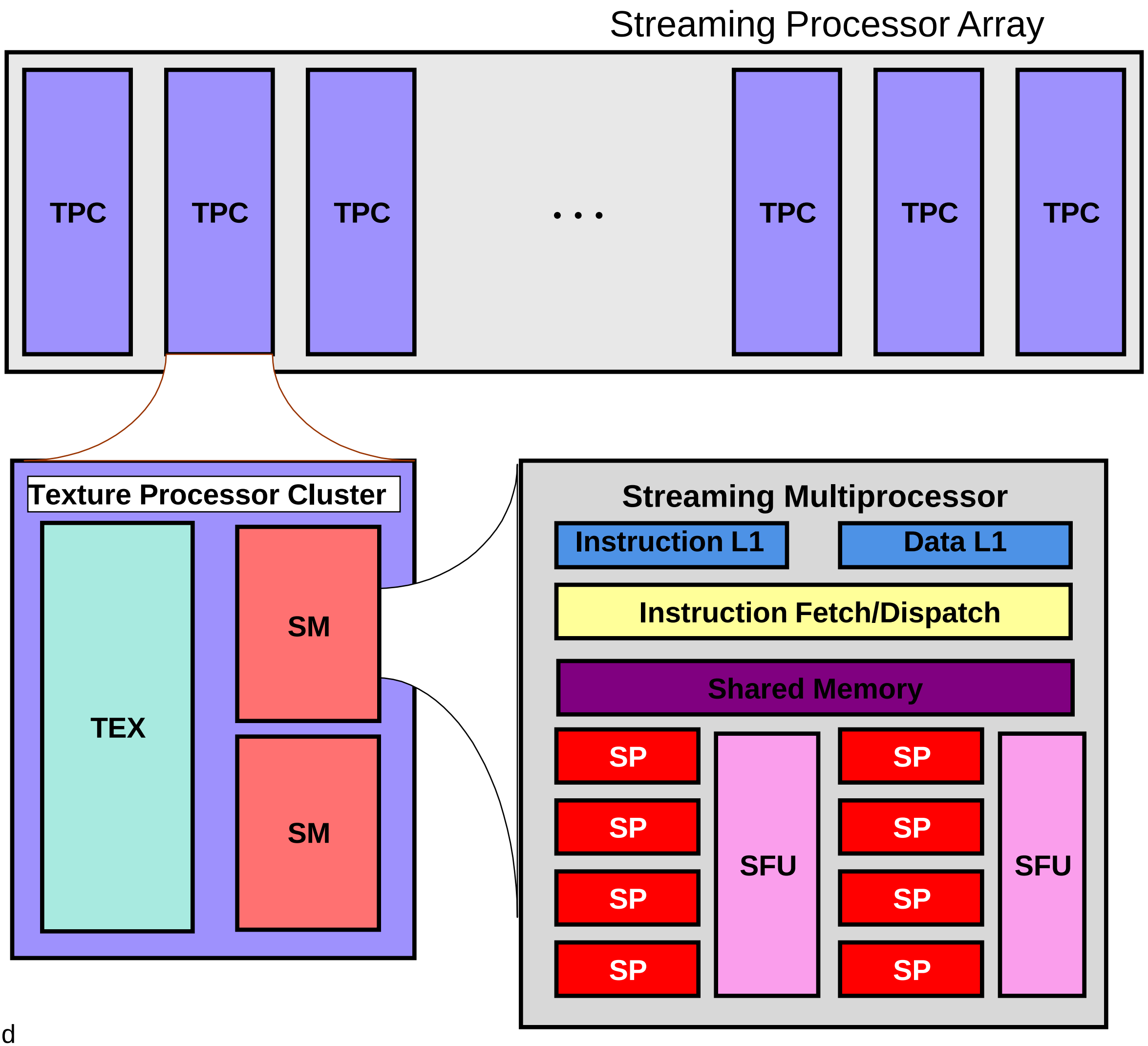
Manages thread blocks

Only one kernel at a time

Wen-mei W. Hwu, 2007

ECE 498AL, University of Illinois,
Urbana-Champaign

GeForce-8 Series HW Overview



© David Kirk/NVIDIA and
Wen-mei W. Hwu, 2007
ECE 498AL, University of Illinois,
Urbana-Champaign

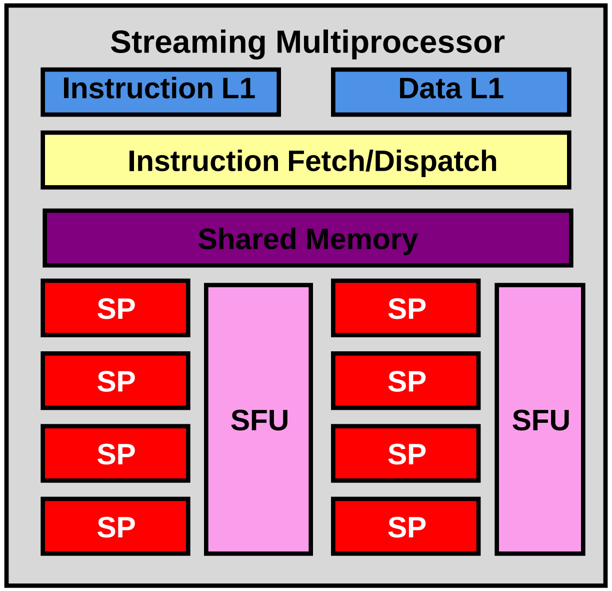
CUDA Processor Terminology

- **SPA** – Streaming Processor Array
  - Array of TPCs
    - 8 TPCs in GeForce8800
- **TPC** – Texture Processor Cluster
  - Cluster of 2 SMs + 1 TEX
    - TEX is a texture processing unit
- **SM** – Streaming Multiprocessor
  - Array of 8 SPs
  - Multi-threaded processor core
  - Fundamental processing unit for a thread block
- **SP** – Streaming Processor
  - Scalar ALU for a single thread

    With 1K of registers

Streaming Multiprocessor (SM)

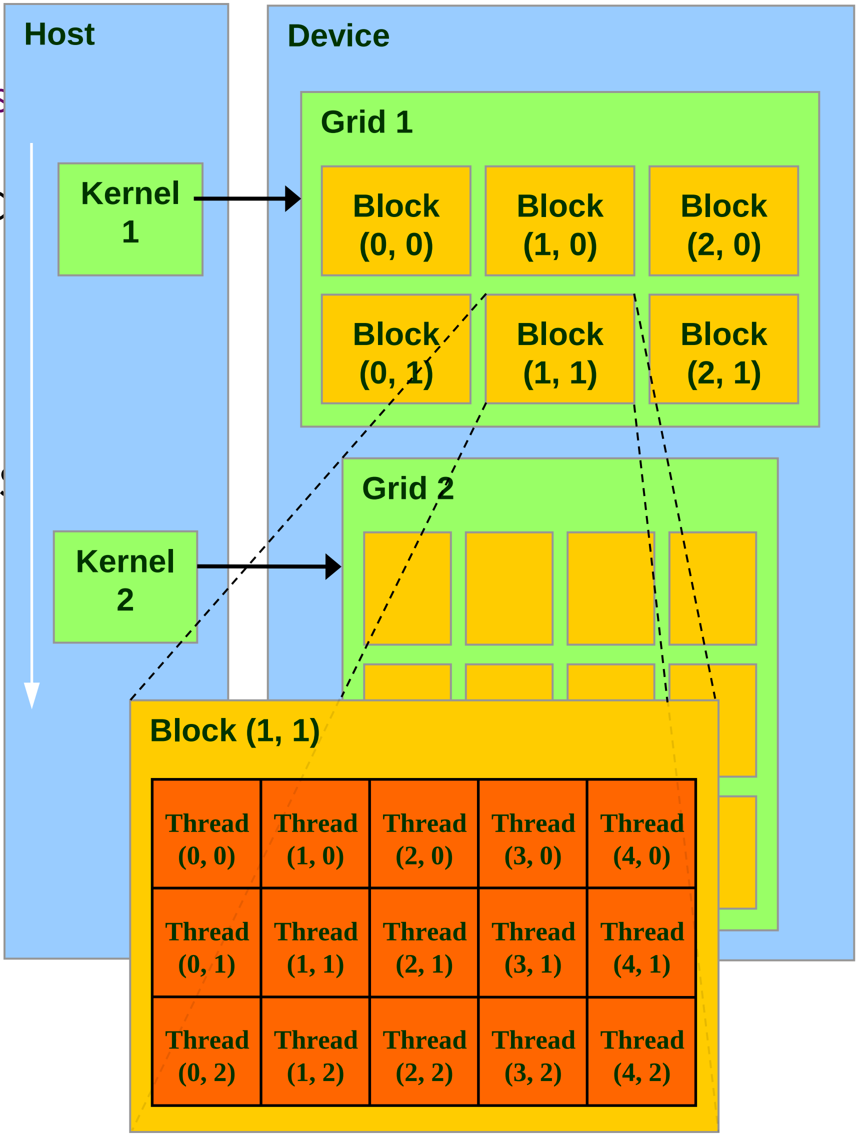
- Streaming Multiprocessor (SM)
  - 8 Streaming Processors (SP)
  - 2 Super Function Units (SFU)
- Multi-threaded instruction dispatch
  - Vectors of 32 threads (***warps***)
  - Up to 16 warps per thread block
    - HW masking of inactive threads in a warp
  - Threads cover latency of texture/memory loads
- 20+ GFLOPS
- 16 KB shared memory
- 32 KB in registers
- DRAM texture and memory access



© David Kirk/NVIDIA and Wen-mei W. Hwu, 2007
ECE 498AL, University of Illinois, Urbana-Champaign

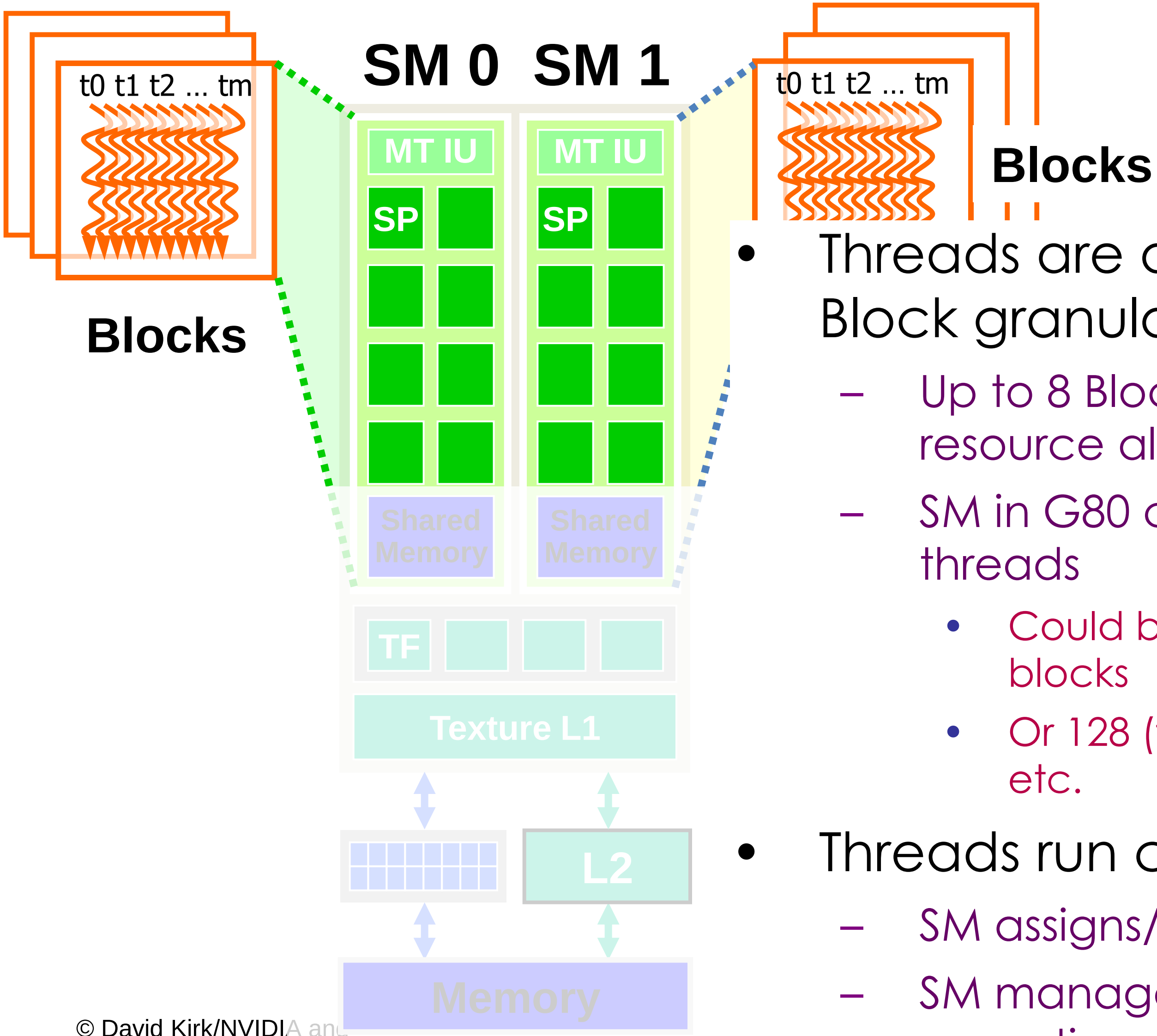
Thread Life Cycle in HW

- Kernel is launched on the SPA
  - Kernels known as ***grids*** of thread blocks
- Thread Blocks are serially distributed to all the SM's
  - Potentially >1 Thread Block per SM
  - At least 96 threads per block
- Each SM launches Warps of Threads
  - 2 levels of parallelism
- SM schedules and executes Warps that are ready to run
- As Warps and Thread Blocks complete, resources are freed
  - SPA can distribute more Thread Blocks



© David Kirk/NVIDIA and Wen-mei W. Hwu, 2007
ECE 498AL, University of Illinois, Urbana-Champaign

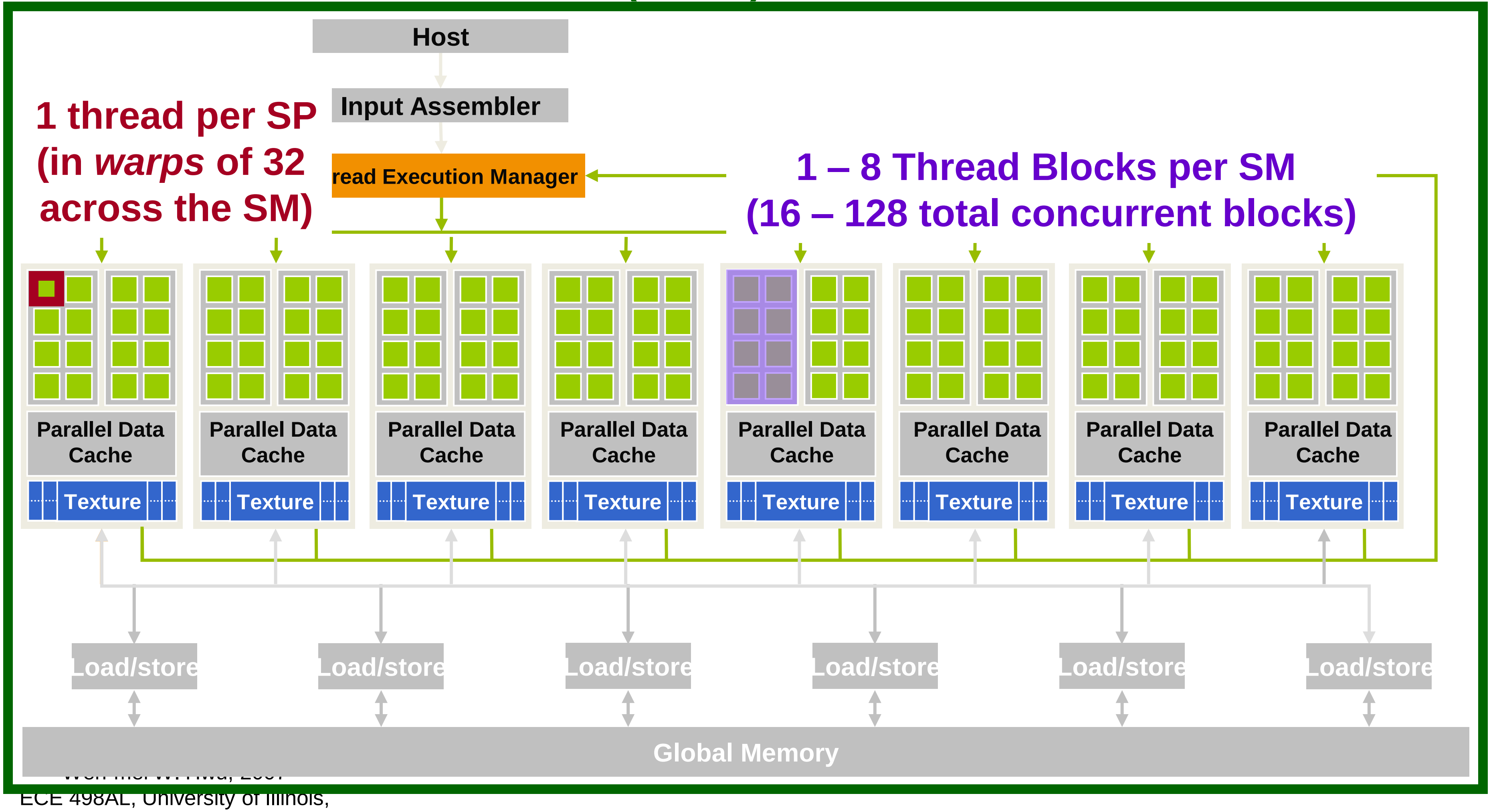
SM Executes Blocks



- Threads are assigned to SMs in Block granularity
  - Up to 8 Blocks to each SM as resource allows
  - SM in G80 can take up to 768 threads
    - Could be 256 (threads/block) * 3 blocks
    - Or 128 (threads/block) * 6 blocks, etc.
- Threads run concurrently
  - SM assigns/maintains thread IDs
  - SM manages/schedules thread execution

Make the Compute Core The Focus of the Architecture

1 Grid (kernel) at a time



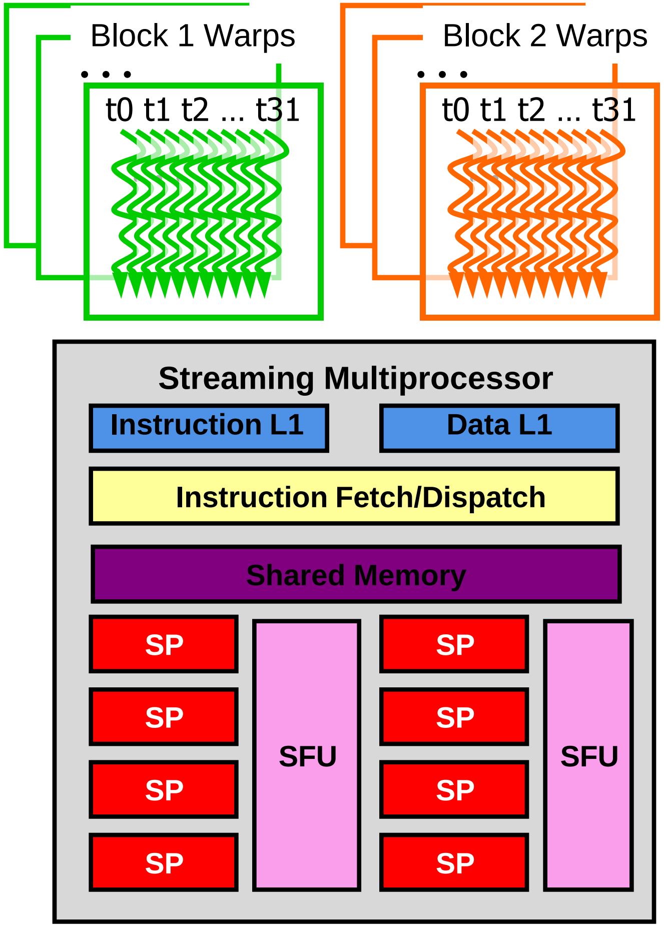
1 thread per SP (in *warps* of 32 across the SM)

1 – 8 Thread Blocks per SM (16 – 128 total concurrent blocks)

Wen-mei W. Hwu, 2007
ECE 498AL, University of Illinois,
Urbana-Champaign

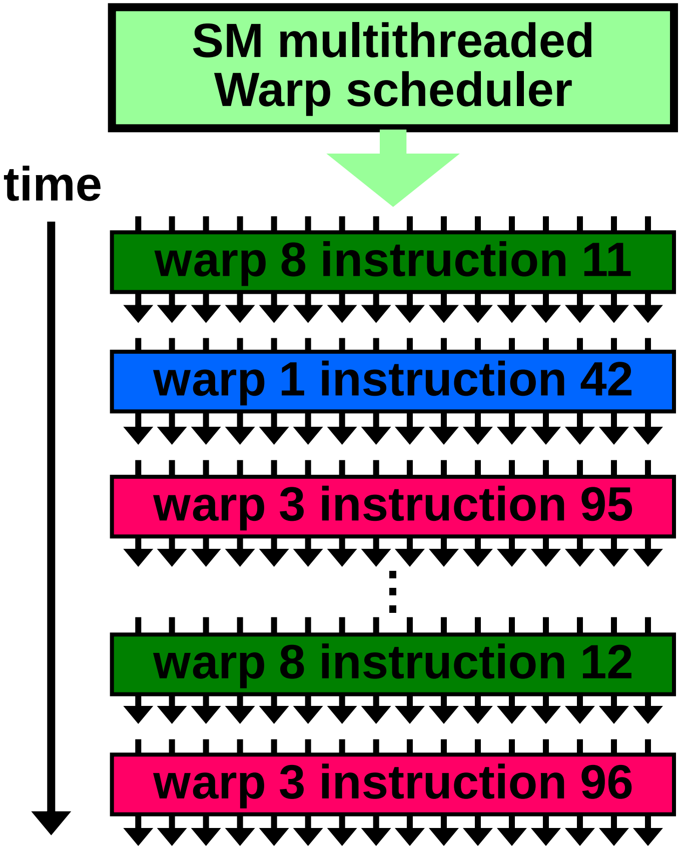
Thread Scheduling/Execution

- Each Thread Block is divided into 32-thread Warps
  - This is an implementation decision
- Warps are scheduling units in SM
- If 3 blocks are assigned to an SM and each Block has 256 threads, how many Warps are there in an SM?
  - Each Block is divided into 256/32 = 8 Warps
  - There are 8 * 3 = 24 Warps
  - At any point in time, only one of the 24 Warps will be selected for instruction fetch and execution.



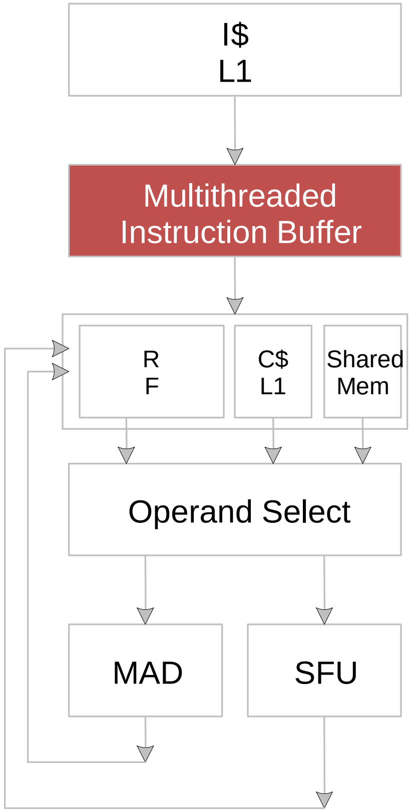
SM Warp Scheduling

- SM hardware implements zero-overhead Warp scheduling
  - Warps whose next instruction has its operands ready for consumption are eligible for execution
  - All threads in a Warp execute the same instruction when selected
  - Scoreboard scheduler
- 4 clock cycles needed to dispatch the same instruction for all threads in a Warp in G80
  - If one global memory access is needed for every 4 instructions
  - A minimal of 13 Warps are needed to fully tolerate 200-cycle memory latency



SM Instruction Buffer – Warp Scheduling

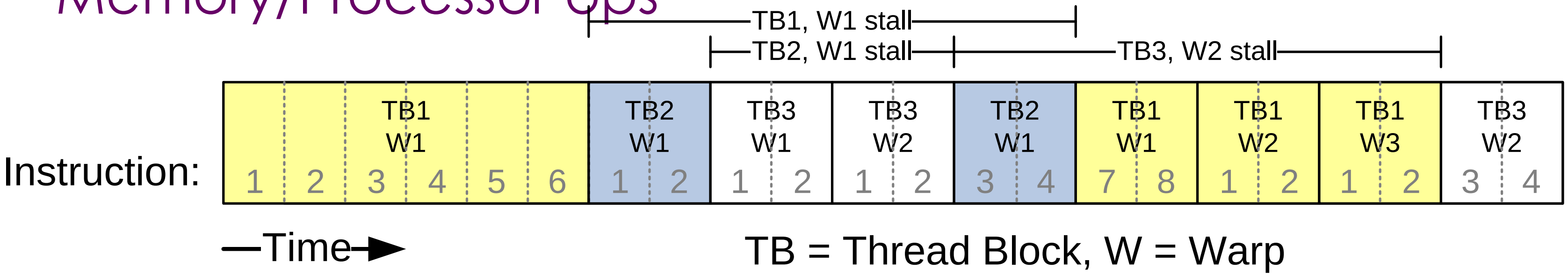
- Fetch one warp instruction/cycle
  - from instruction L1 cache
  - into any instruction buffer slot
- Issue one "ready-to-go" warp instruction/cycle
  - from any warp - instruction buffer slot
  - operand scoreboarding used to prevent hazards
- Issue selection based on round-robin/age of warp
- SM broadcasts the same instruction to 32 Threads of a Warp



© David Kirk/NVIDIA and Wen-mei W. Hwu, 2007
ECE 498AL, University of Illinois, Urbana-Champaign

Scoreboarding

- All register operands of all instructions in the Instruction Buffer are scoreboarded
  - Status becomes ready after the needed values are deposited
  - prevents hazards
  - cleared instructions are eligible for issue
- Decoupled Memory/Processor pipelines
  - any thread can continue to issue instructions until scoreboarding prevents issue
  - allows Memory/Processor ops to proceed in shadow of Memory/Processor ops



TB1, W1 stall

TB2, W1 stall

TB3, W2 stall

Instruction:

TB1 W1	TB2 W1	TB3 W1	TB3 W2	TB2 W1	TB1 W1	TB1 W2	TB1 W3	TB3 W2
1 2 3 4 5 6	1 2	1 2	1 2	3 4	7 8	1 2	1 2	3 4

Time

TB = Thread Block, W = Warp

Granularity and Resource Considerations

- For Matrix Multiplication, should I use 8X8, 16X16 or 32X32 tiles (1 thread per tile element)?
  - For 8X8, we have 64 threads per Block. Since each SM can take up to 768 threads, it can take up to 12 Blocks. However, each SM can only take up to 8 Blocks, only 512 threads will go into each SM!
  - For 16X16, we have 256 threads per Block. Since each SM can take up to 768 threads, it can take up to 3 Blocks and achieve full capacity unless other resource considerations overrule.
  - For 32X32, we have 1024 threads per Block. Not even one nto an SM!

Parallel Computing on a GPU

- NVIDIA GPU Computing Architecture
- Via a separate HW interface
- In laptops, desktops, workstations, servers

- 8-series GPUs deliver 50 to 200 GFLOPS on compiled parallel C applications

- GPU parallelism is doubling every year
- Programming model scales transparently

- Programmable in C with CUDA tools
- Multithreaded SPMD model uses application data parallelism and thread parallelism

GeForce 8800

Tesla D870

Tesla S870

Next Lectures

- NVIDIA GeForce 8800 architecture
- CUDA programming model