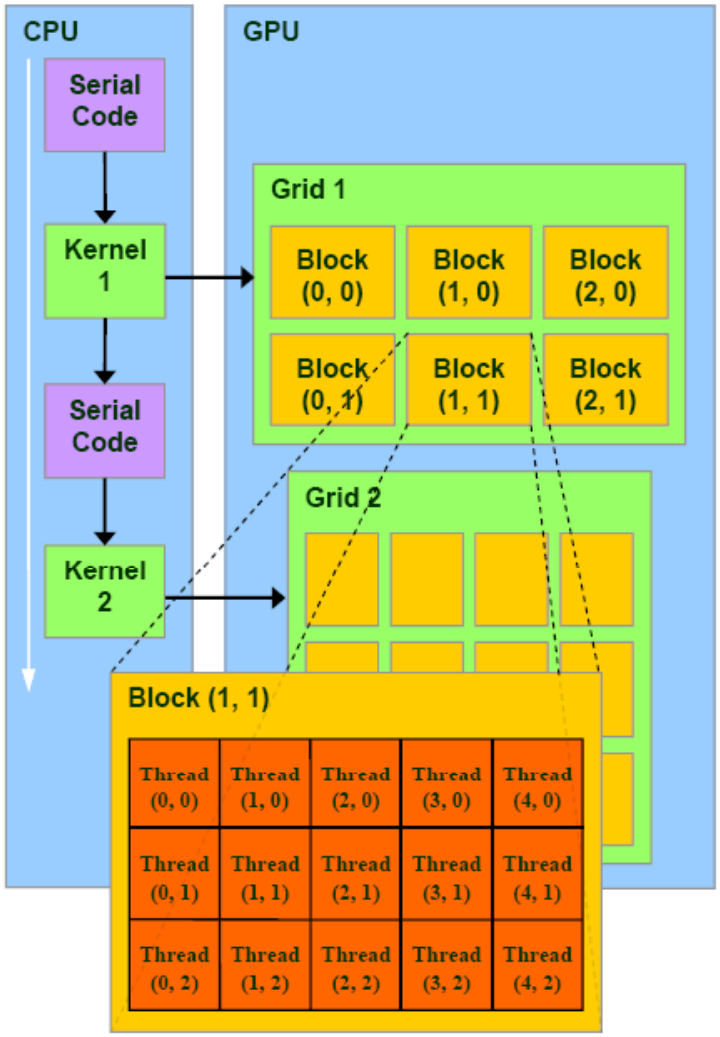
NVIDIA Tesla as Computing Architecture

Advanced Computer Architecture, 332

CUDA Execution Model

- CUDA is a C extension
  - Serial CPU code
  - Parallel GPU code (kernels)
- GPU kernel is a C function
  - Each *thread* executes kernel code
  - A group of threads forms a *thread block* (1D, 2D or 3D)
  - Thread blocks are organised into a *grid* (1D or 2D)
  - Threads within the same thread block can synchronise execution



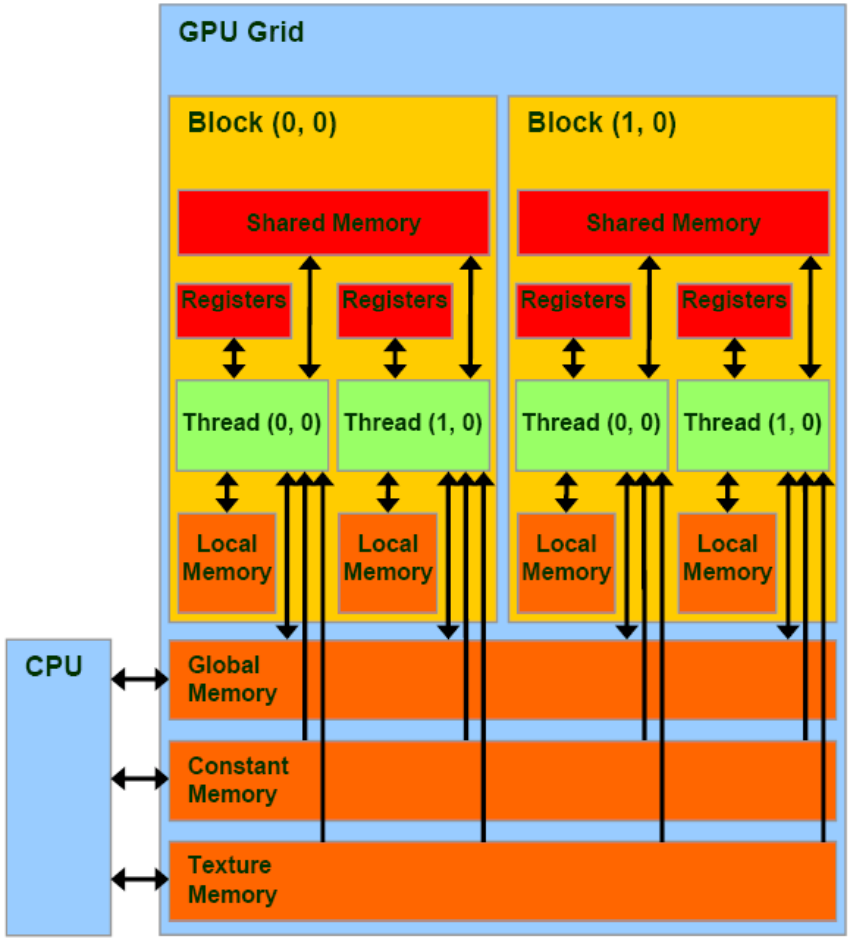
Example: Matrix addition

```
__global__ void matAdd(float A[N][N], float B[N][N], float C[N][N])
{
   int i = blockIdx.x * blockDim.x + threadIdx.x;
   int j = blockIdx.y * blockDim.y + threadIdx.y;
   if (i < N && j < N)
      C[i][j] = A[i][j] + B[i][j];
}

int main()
{
   // Kernel setup
   dim3 blockDim(16, 16);
   dim3 gridDim((N + blockDim.x – 1) / blockDim.x,
                (N + blockDim.y – 1) / blockDim.y);
   // Kernel invocation
   matAdd<<<gridDim, blockDim>>>(A, B, C);
}
```

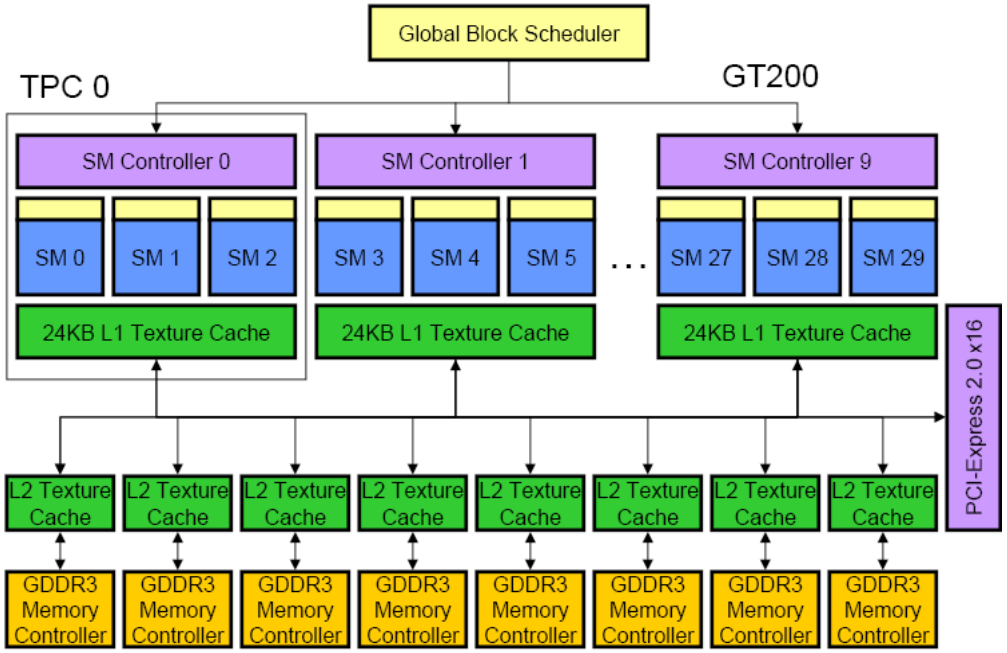
CUDA Memory Model

- Local memory – private to each thread (slow if off-chip, fast if register allocated)
- Shared memory – shared between threads in a thread block (fast on-chip)
- Global memory – shared between thread blocks in a grid (slow off-chip)
- Constant memory (read-only; off-chip but cached)
- Texture memory (read-only; off-chip but cached)



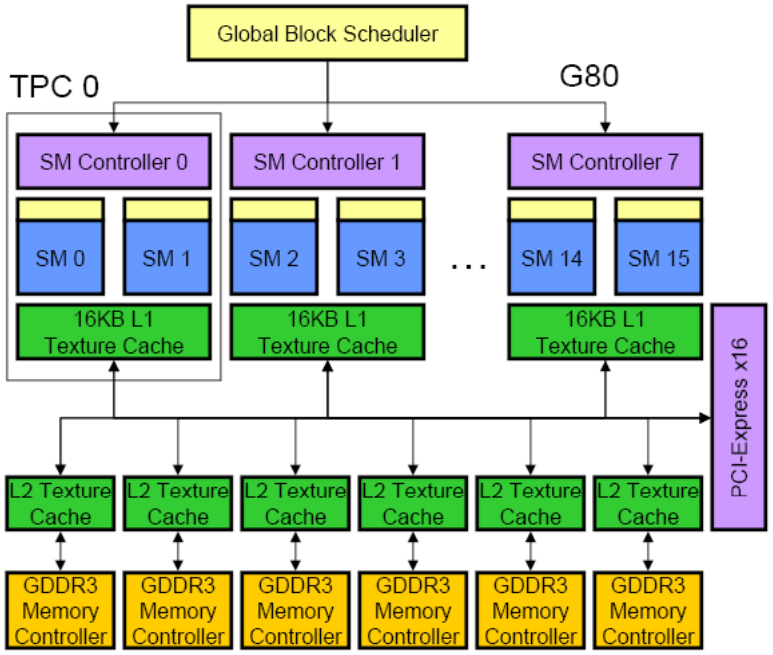
Tesla architecture

- A unified graphics and computing architecture
- Scalable array of streaming multiprocessors (SMs)
- CUDA thread blocks get mapped to SMs
- SMs have thread processors, private registers, shared memory, etc.



GT200

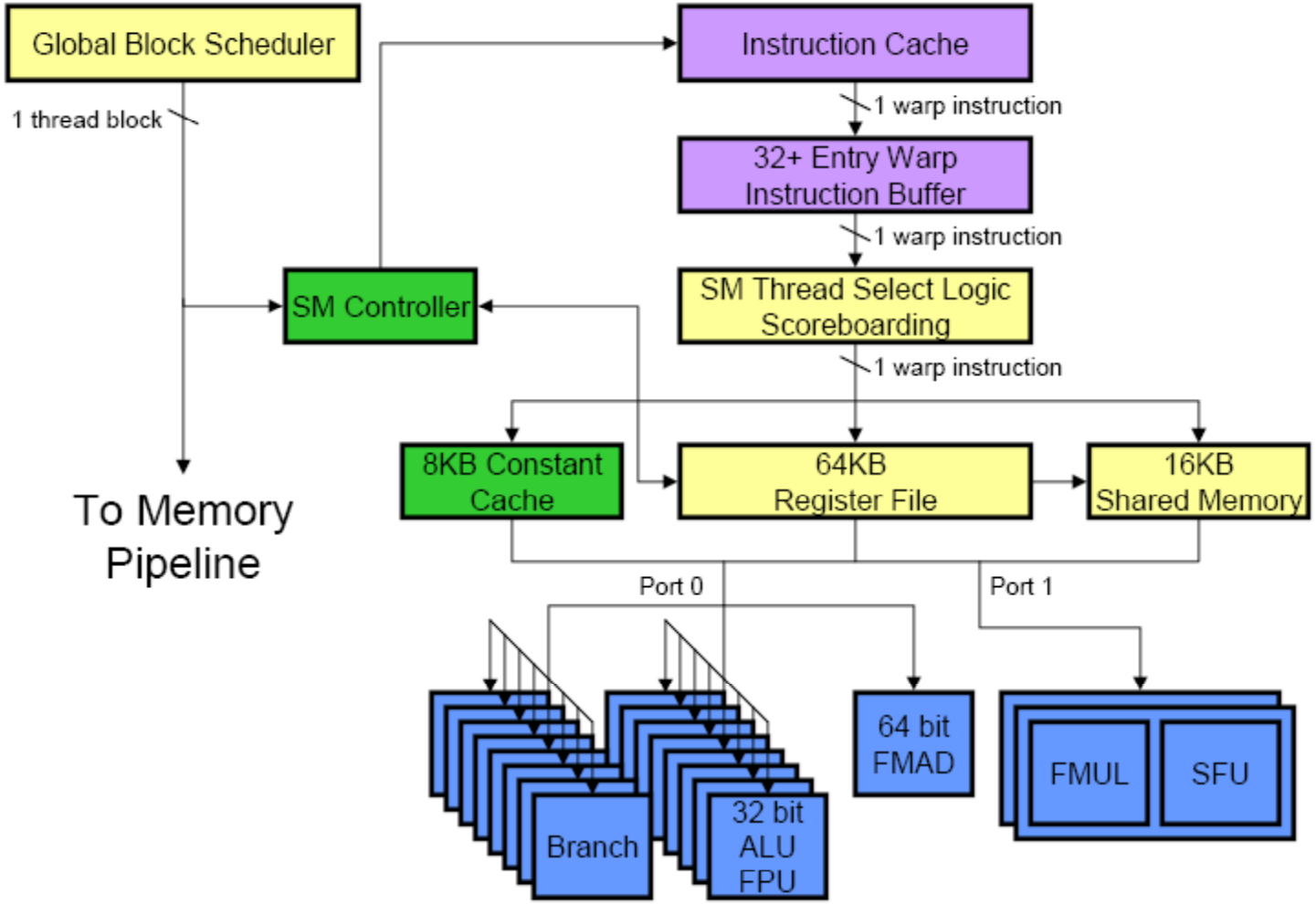
- 10 Thread Processing Clusters (TPCs)
- 3 Streaming Multiprocessors (SMs) per TPC
- 8 32-bit FPU per SM
- 1 64-bit FPU per SM
- 16K 32-bit registers per SM
- Up to 1024 threads / 32 warps per SM
- 8 64-bit memory controllers (512-bit wide memory interface)



G80

- 8 Thread Processing Clusters (TPCs)
- 2 Streaming Multiprocessors (SMs) per TPC
- 8 32-bit FPU per SM
- 8K 32-bit registers per SM
- Up to 768 threads / 24 warps per SM
- 6 64-bit memory controllers (384-bit wide memory interface)

GT200 streaming multiprocessor



GT200 streaming multiprocessor

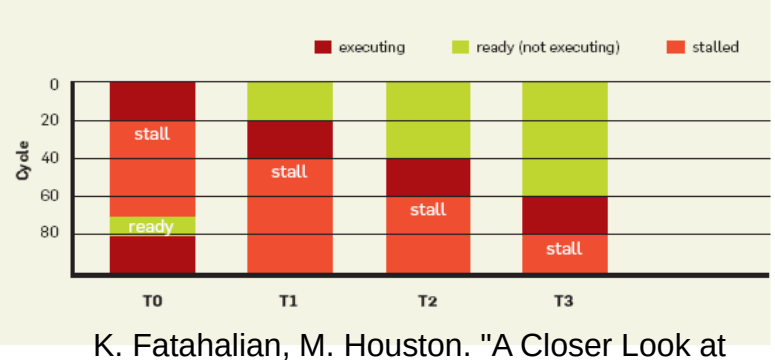
- Multithreaded instruction fetch and issue unit
- 8 streaming processors (SP)
  - 32-bit ALU and FPU (e.g. bitwise, min, max; FP add, mul, mad)
  - Branch unit (e.g. cmp)
- 2 special-function units (SFU)
  - Transcendental functions (e.g. reciprocal, sine), interpolation
  - Interpolation hardware can be used as 4 FP multipliers
- 16KB shared memory (as fast as registers, if no conflicts)
- 64-bit FP and integer unit (not in G80)
- 16K 32-bit registers (only 8K in G80)

Global block scheduler

- Reads grid information that is sent from the CPU to the GPU when a kernel is started
- Issues thread blocks to SMs that have sufficient resources for execution (up to 8 blocks per SM)

SM multithreaded instruction unit

- Instruction unit creates, manages, schedules and executes threads in groups of 32 threads called *warps*
- Warp instructions are fetched into an instruction buffer
- Scoreboard qualifies warp instructions for issue
- Scheduler prioritises ready to issue warp instructions based on their type and "fairness"
- Scheduler issues a warp instruction with highest priority

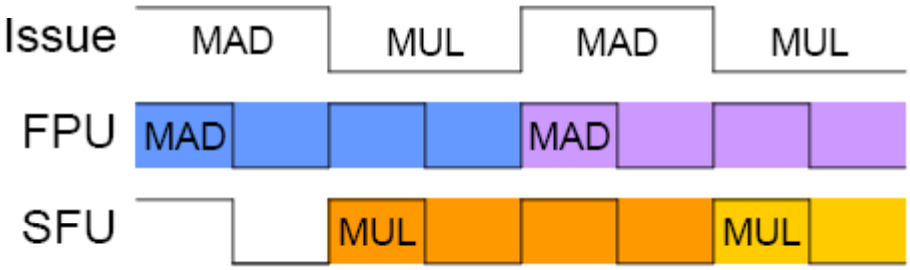


K. Fatahalian, M. Houston. "A Closer Look at GPUs," *Commun. ACM*, 51:10, pp. 50-57.

A hypothetical GPU:
- Four warp contexts T0–T3
- 20 cycle compute
- 50 cycle memory access

SM instruction issue

- To optimise for power and performance, GPU units are in multiple clock domains
  - Core clock (e.g. 600 MHz) – "slow" cycles
  - FUs clock (e.g. 1300 MHz) – "fast" cycles
  - Memory clock (e.g. 1100 MHz)
- Issue logic can issue every slow cycle (every 2 fast cycles)
- FUs can typically be issued instructions every 4 fast cycles (1 cycle per 8 threads from a warp)

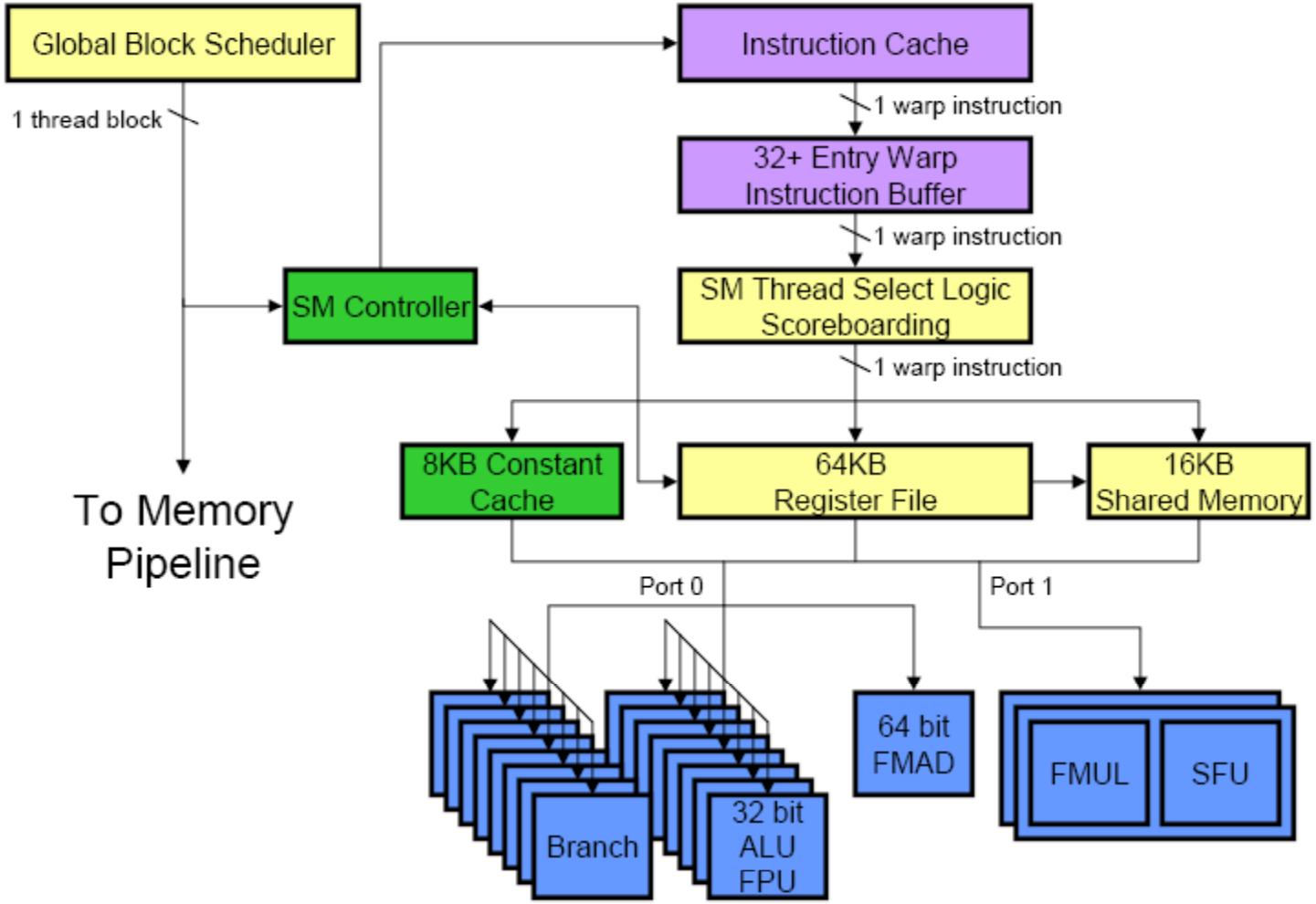


"Dual" issue to independent FUs on alternate slow cycles

SM branch management

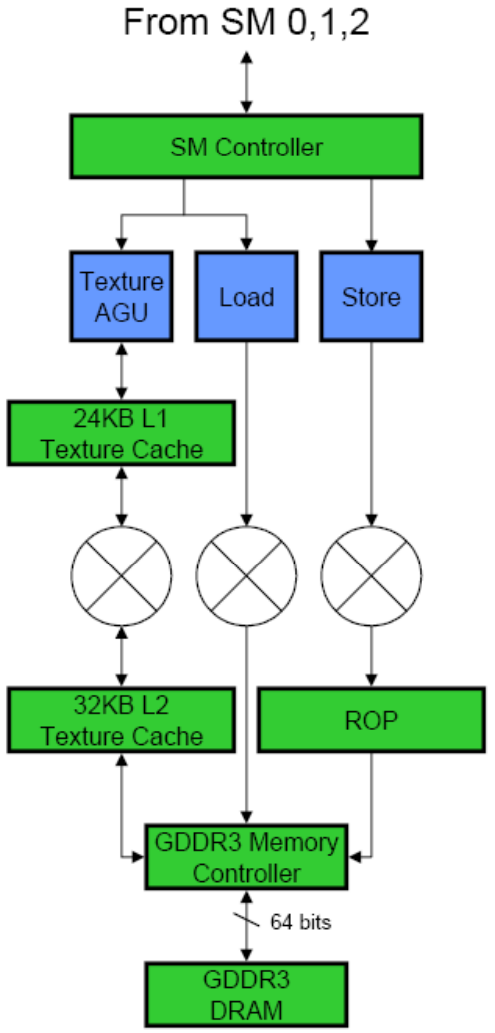
- Threads in a warp start executing from the same address
- A warp that executes a branch instruction waits until the target address for every thread in the warp is computed
- Threads in a warp can take the same branch path ☺
- Threads in in warp can diverge to different paths
  - the warp serially executes each path, disabling threads that are not on that path;
  - when all paths complete, the threads reconverge
- Divergence only occurs within a warp – different warps execute independently
- Minimising divergence is important for performance, but not correctness (cf. cache lines)

GT200 streaming multiprocessor



TPC memory pipeline

- Load and store instructions are generated in the SMs
  - Address calculation (register + offset)
  - Virtual to physical address translation
- Issued a warp at a time – executed in half-warp groups (i.e. 16 accesses at a time)
- Memory coalescing and alignment
  - Threads with adjacent indices should access adjacent memory locations (i.e. thread K should access Kth data word)
  - Accesses should be aligned for half-words



Further information

- E. Lindholm, J. Nickolls, S. Oberman, J. Montrym. "NVIDIA Tesla: A Unified Graphics and Computing Architecture". *IEEE Micro* 28, 2 (March-April 2008), 39-55. http://dx.doi.org/10.1109/MM.2008.31
- D. Kanter. "NVIDIA's GT200: Inside a Parallel Processor". http://www.realworldtech.com/page.cfm?ArticleID=RWT090808195242
- K. Fatahalian, M. Houston. "GPUs: A Closer Look". *ACM Queue* 6, 2 (March-April 2008), 18-28. http://doi.acm.org/10.1145/1365490.1365498
- K. Datta, M. Murphy, V. Volkov, S. Williams, J. Carter, L. Oliker, D. Patterson, J. Shalf, K. Yelick. "Stencil Computation Optimization and Autotuning on State-of-the-Art Multicore Architectures". *Supercomputing'08*. http://doi.acm.org/10.1145/1413370.1413375

Core Architecture	Intel Core2	AMD Barcelona	Sun Niagara2	STI Cell eDP SPE	NVIDIA GT200 SM
Type	super scalar out of order	super scalar out of order	MT dual issue†	SIMD dual issue	MT SIMD
Process	65nm	65nm	65nm	65nm	65nm
Clock (GHz)	2.66	2.30	1.16	3.20	1.3
DP GFlop/s	10.7	9.2	1.16	12.8	2.6
Local-Store	—	—	—	256KB	16KB**
L1 Data Cache	32KB	64KB	8KB	—	—
private L2 cache	—	512KB	—	—	—

System	Xeon E5355 (Clovertown)	Opteron 2356 (Barcelona)	UltraSparc T5140 T2+ (Victoria Falls)	QS22 PowerXCell 8i (Cell Blade)	GeForce GTX280
Heterogeneous	no	no	no	multicore	multichip
# Sockets	2	2	2	2	1
Cores per Socket	4	4	8	8(+1)	30 (×8)
shared L2/L3 cache	4×4MB (shared by 2)	2×2MB (shared by 4)	2×4MB (shared by 8)	—	—
DP GFlop/s	85.3	73.6	18.7	204.8	78
primary memory parallelism paradigm	HW prefetch	HW prefetch	Multithreading	DMA	Multithreading with coalescing
DRAM Bandwidth (GB/s)	21.33(read) 10.66(write)	21.33	42.66(read) 21.33(write)	51.2	141 (device) 4 (PCIe)
DP Flop:Byte Ratio	2.66	3.45	0.29	4.00	0.55
DRAM Capacity	16GB	16GB	32GB	32GB	1GB (device) 4GB (host)
System Power (Watts)§	330	350	610	270‡	450 (236)⋆
Chip Power (Watts)¶	2×120	2×95	2×84	2×90	165
Threading	Pthreads	Pthreads	Pthreads	libspe2.1	CUDA 2.0
Compiler	icc 10.0	gcc 4.1.2	gcc 4.0.4	xlc 8.2	nvcc 0.2.1221

Table 1. Architectural summary of evaluated platforms. †Each of the two thread groups may issue up to one instruction. **16 KB local-store shared by all concurrent *CUDA thread blocks* on the SM. ‡Cell Bladecenter power running Linpack averaged per blade. (www.green500.org) §All system power is measured with a digital power meter while under a full computational load. ¶Chip power is based on the maximum Thermal Design Power (TDP) from the manufacturer's datasheets. ⋆GTX280 system power shown for the entire system under load (450W) and GTX280 card itself (236W).