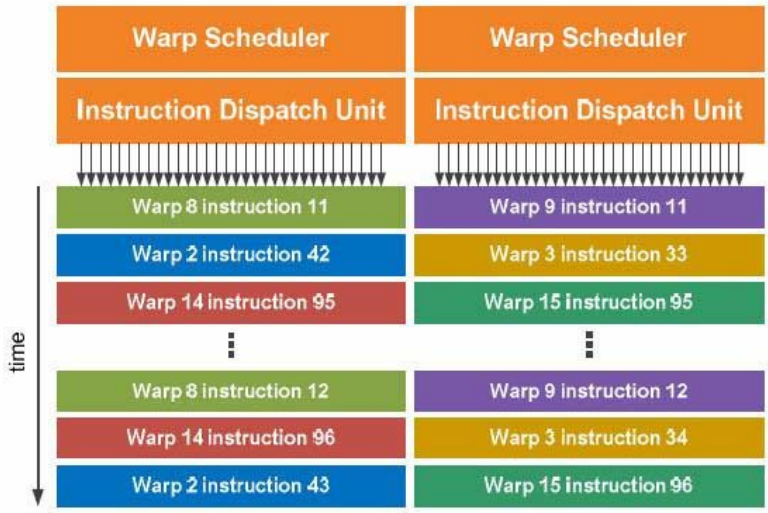
Dual Warp Scheduler

The SM schedules threads in groups of 32 parallel threads called warps. Each SM features two warp schedulers and two instruction dispatch units, allowing two warps to be issued and executed concurrently. Fermi's dual warp scheduler selects two warps, and issues one instruction from each warp to a group of sixteen cores, sixteen load/store units, or four SFUs. Because warps execute independently, Fermi's scheduler does not need to check for dependencies from within the instruction stream. Using this elegant model of dual-issue, Fermi achieves near peak hardware performance.



Most instructions can be dual issued; two integer instructions, two floating instructions, or a mix of integer, floating point, load, store, and SFU instructions can be issued concurrently. Double precision instructions do not support dual dispatch with any other operation.

64 KB Configurable Shared Memory and L1 Cache

One of the key architectural innovations that greatly improved both the programmability and performance of GPU applications is on-chip shared memory. Shared memory enables threads within the same thread block to cooperate, facilitates extensive reuse of on-chip data, and greatly reduces off-chip traffic. Shared memory is a key enabler for many high-performance CUDA applications.

G80 and GT200 have 16 KB of shared memory per SM. In the Fermi architecture, each SM has 64 KB of on-chip memory that can be configured as 48 KB of Shared memory with 16 KB of L1 cache or as 16 KB of Shared memory with 48 KB of L1 cache.

For existing applications that make extensive use of Shared memory, tripling the amount of Shared memory yields significant performance improvements, especially for problems that are

bandwidth constrained. For existing applications that use Shared memory as software managed cache, code can be streamlined to take advantage of the hardware caching system, while still having access to at least 16 KB of shared memory for explicit thread cooperation. Best of all, applications that do not use Shared memory automatically benefit from the L1 cache, allowing high performance CUDA programs to be built with minimum time and effort.

Summary Table

GPU	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double Precision Floating Point Capability	None	30 FMA ops / clock	256 FMA ops /clock
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops / clock	512 FMA ops /clock
Warp schedulers (per SM)	1	1	2
Special Function Units (SFUs) / SM	2	2	4
Shared Memory (per SM)	16 KB	16 KB	Configurable 48 KB or 16 KB
L1 Cache (per SM)	None	None	Configurable 16 KB or 48 KB
L2 Cache (per SM)	None	None	768 KB
ECC Memory Support	No	No	Yes
Concurrent Kernels	No	No	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit

Second Generation Parallel Thread Execution ISA

Fermi is the first architecture to support the new Parallel Thread eXecution (PTX) 2.0 instruction set. PTX is a low level virtual machine and ISA designed to support the operations of a parallel thread processor. At program install time, PTX instructions are translated to machine instructions by the GPU driver.

The primary goals of PTX are:

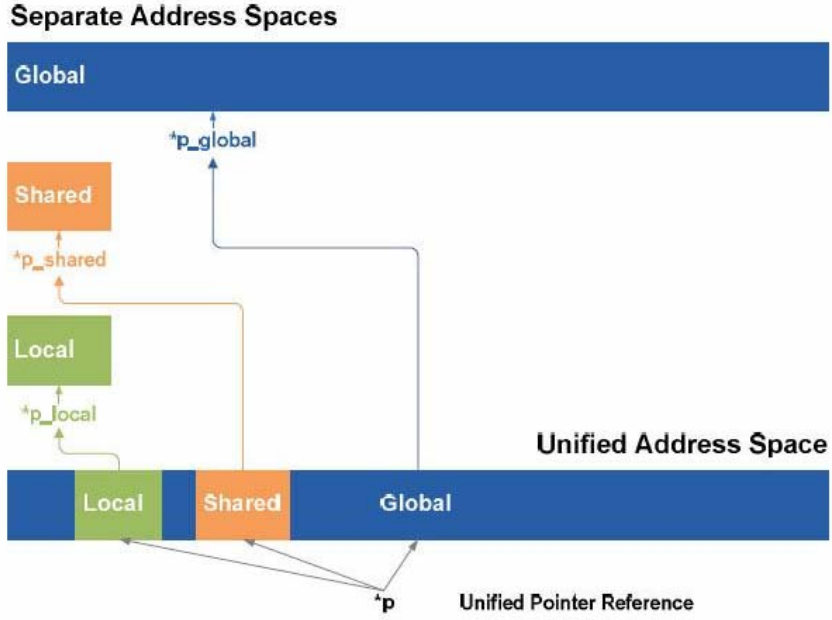
- Provide a stable ISA that spans multiple GPU generations
- Achieve full GPU performance in compiled applications
- Provide a machine-independent ISA for C, C++, Fortran, and other compiler targets.
- Provide a code distribution ISA for application and middleware developers
- Provide a common ISA for optimizing code generators and translators, which map PTX to specific target machines.
- Facilitate hand-coding of libraries and performance kernels
- Provide a scalable programming model that spans GPU sizes from a few cores to many parallel cores

PTX 2.0 introduces several new features that greatly improve GPU programmability, accuracy, and performance. These include: full IEEE 32-bit floating point precision, unified address space for all variables and pointers, 64-bit addressing, and new instructions for OpenCL and DirectCompute. Most importantly, PTX 2.0 was specifically designed to provide full support for the C++ programming language.

Unified Address Space enables Full C++ Support

Fermi and the PTX 2.0 ISA implement a unified address space that unifies the three separate address spaces (thread private local, block shared, and global) for load and store operations. In PTX 1.0, load/store instructions were specific to one of the three address spaces; programs could load or store values in a specific target address space known at compile time. It was difficult to fully implement C and C++ pointers since a pointer's target address space may not be known at compile time, and may only be determined dynamically at run time.

With PTX 2.0, a unified address space unifies all three address spaces into a single, continuous address space. A single set of unified load/store instructions operate on this address space, augmenting the three separate sets of load/store instructions for local, shared and global. The 40-bit unified address space supports a Terabyte of addressable memory, and the load/store ISA supports 64-bit addressing for future growth.



The implementation of a unified address space enables Fermi to support true C++ programs. In C++, all variables and functions reside in objects which are passed via pointers. PTX 2.0 makes

it possible to use unified pointers to pass objects in any memory space, and Fermi's hardware address translation unit automatically maps pointer references to the correct memory space.

Fermi and the PTX 2.0 ISA also add support for C++ virtual functions, function pointers, and 'new' and 'delete' operators for dynamic object allocation and de-allocation. C++ exception handling operations 'try' and 'catch' are also supported.

Optimized for OpenCL and DirectCompute

OpenCL and DirectCompute are closely related to the CUDA programming model, sharing the key abstractions of threads, thread blocks, grids of thread blocks, barrier synchronization, per-block shared memory, global memory, and atomic operations. Fermi, a third-generation CUDA architecture, is by nature well-optimized for these APIs. In addition, Fermi offers hardware support for OpenCL and DirectCompute surface instructions with format conversion, allowing graphics and compute programs to easily operate on the same data. The PTX 2.0 ISA also adds support for the DirectCompute instructions population count, append, and bit-reverse.

IEEE 32-bit Floating Point Precision

Single precision floating point instructions now support subnormal numbers by default in hardware, as well as all four IEEE 754-2008 rounding modes (nearest, zero, positive infinity, and negative infinity).

Subnormal numbers are small numbers that lie between zero and the smallest normalized number of a given floating point number system. Prior generation GPUs flushed subnormal operands and results to zero, incurring a loss of accuracy. CPUs typically perform subnormal calculations in exception-handling software, taking thousands of cycles. Fermi's floating point units handle subnormal numbers in hardware, allowing values to gradually underflow to zero with no performance penalty.

A frequently used sequence of operations in computer graphics, linear algebra, and scientific applications is to multiply two numbers, adding the product to a third number, for example, $D = A \times B + C$. Prior generation GPUs accelerated this function with the multiply-add (MAD) instruction that allowed both operations to be performed in a single clock. The MAD instruction performs a multiplication with truncation, followed by an addition with round-to-nearest even. Fermi implements the new fused multiply-add (FMA) instruction for both 32-bit single-precision and 64-bit double-precision floating point numbers (GT200 supported FMA only in double precision) that improves upon multiply-add by retaining full precision in the intermediate stage. The increase in precision benefits a number of algorithms, such as rendering fine intersecting geometry, greater precision in iterative mathematical calculations, and fast, exactly-rounded division and square root operations.