Arquitectura de Computadores

Tema 4-2bis. Arquitecturas de los Multiprocesadores para Procesamiento Gráfico

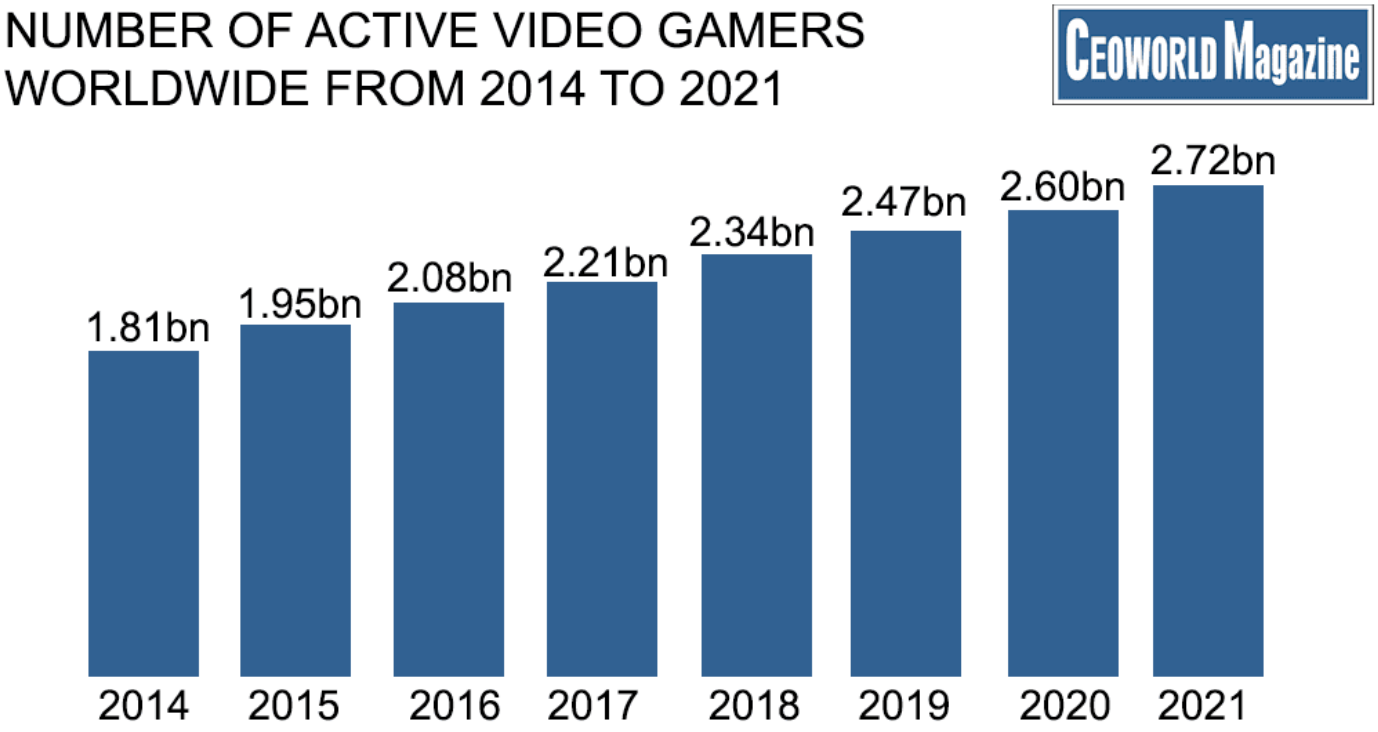
Sumario

- Introducción
- GPUs para cómputo de propósito general
- Estructuras de programación para GPUs
- Arquitectura GPU de NVIDIA

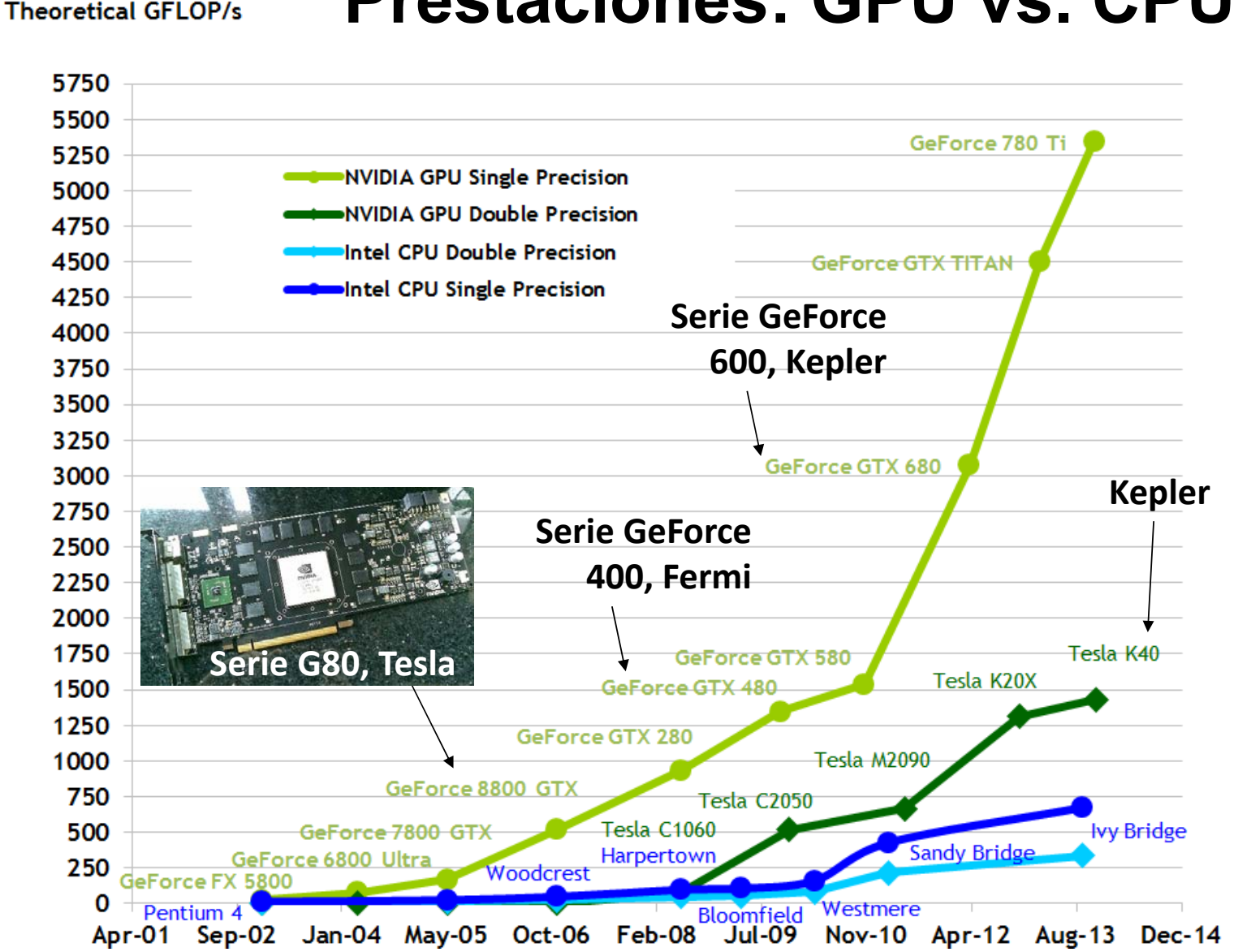
Introducción

- Muchos computadores están conectados a una pantalla gráfica
  - Una parte del procesamiento realizado por el procesador se destina a la generación de gráficos para ser mostrados en la pantalla
- Debido a la Ley de Moore (doble transistores en chip cada ~dos años)
  - Una parte del chip del procesador se destina a mejorar el procesamiento gráfico
  - La industria de los juegos por computador propuso las videoconsolas y las tarjetas gráficas
- Resultados:
  - El nivel de prestaciones en número de operacionesALU/segundo de los sistemas hardware gráficos (GPU, Graphics Processing Units) superan a los multiprocesadores (CPU)
  - Se puede conseguir una GPU que proporciona computación de altas prestaciones por unos pocos euros

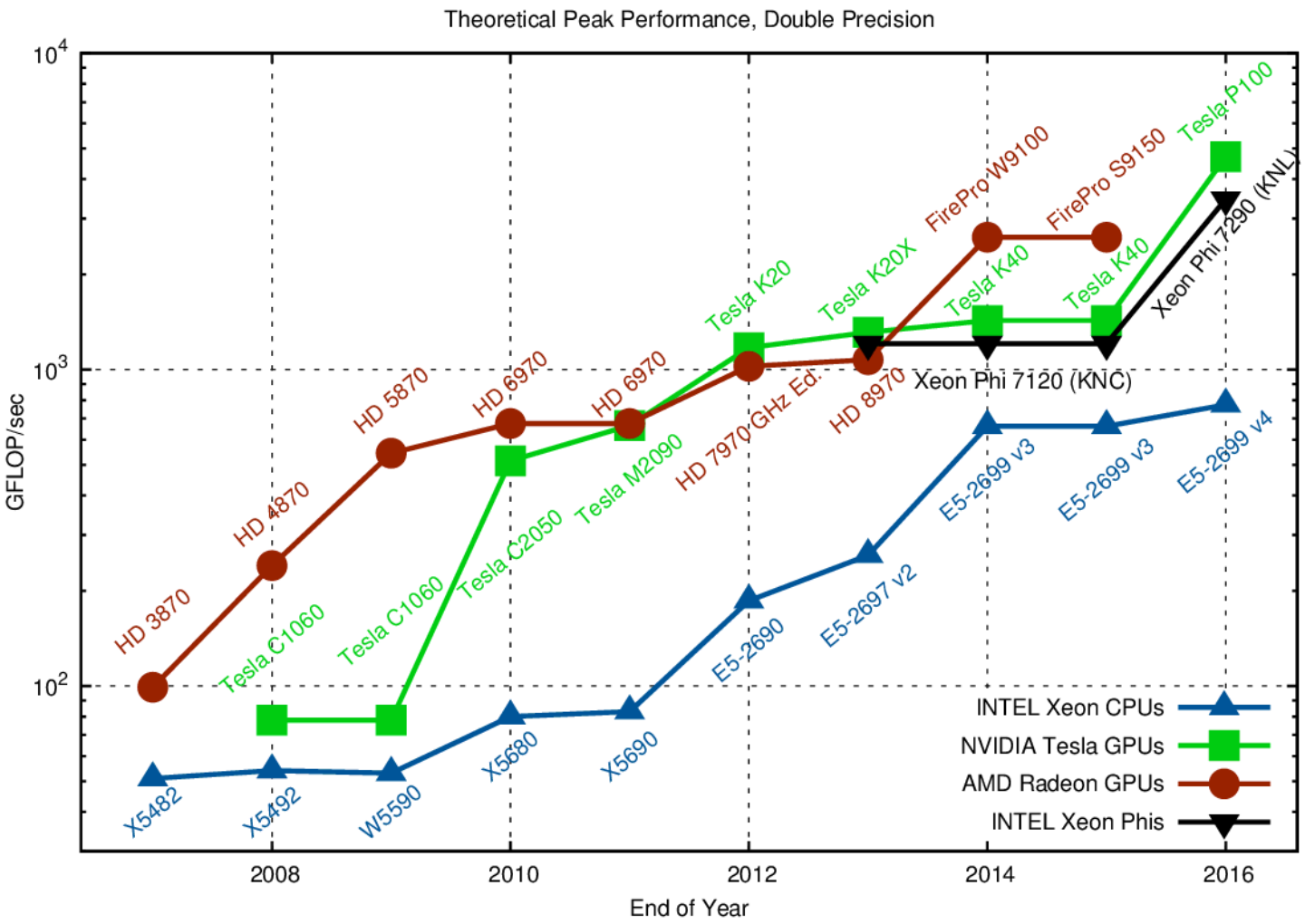
Número de jugadores de videojuegos



Prestaciones: GPU vs. CPU



Prestaciones: GPU vs. CPU

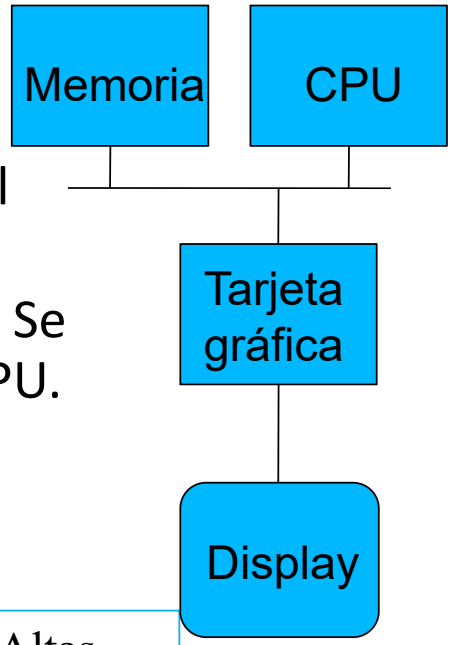


Introducción

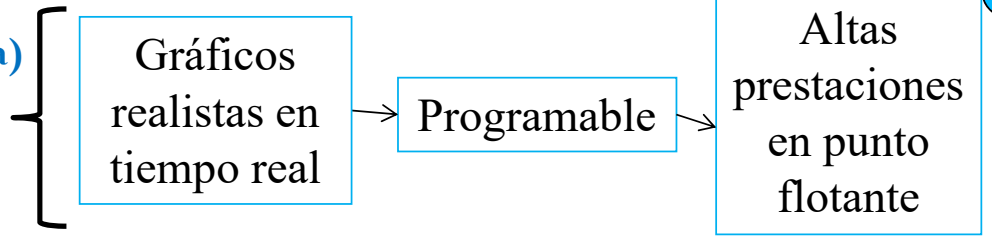
- Resultados:
  - Como el objetivo de CPU y GPU es distinto, cada hardware ha evolucionado con distinto estilo de arquitectura
  - La popularidad de GPU aumentó cuando se pudieron programar con lenguajes de alto nivel. GPUs son fáciles de programar.
- Aprovechamiento de las GPUs:
  - Periférico conectado a una pantalla para generar imágenes
  - Uso de GPU para computación de propósito general

Introducción

- ¿Qué es una GPU?
  - Multiprocesador de memoria compartida optimizado para computación gráfica y científica
- Evolución histórica
  - 1970-1980: procesadores usaban coprocesadores aritméticos. Aparecieron las tarjetas gráficas con el objetivo de mejorar la visualización.
  - 1999-200x: chips gráficos para procesamiento 3-D. Se usaban APIs gráficas. NVIDIA inventó el término GPU.
    - Evolución histórica:



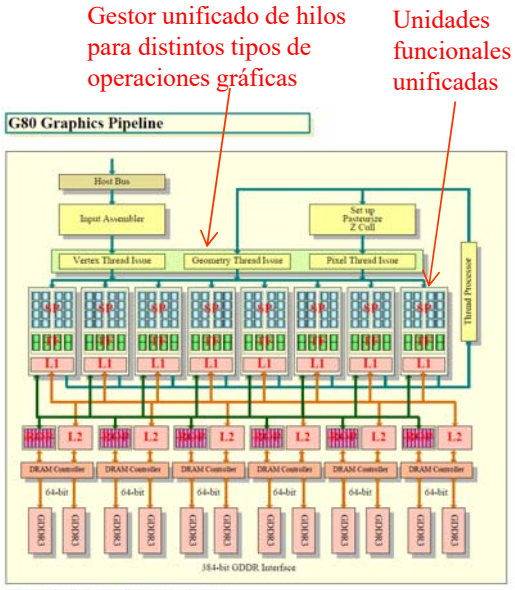
1. Vértices
2. Polígonos (geometría)
3. Generación imagen (rasterizer)
4. Pixels

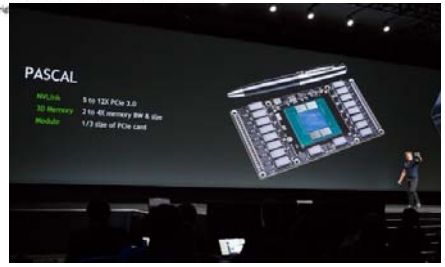


Introducción

- Evolución histórica de GPU
  - 2003: **GPGPU**, uso de tarjetas gráficas y **API** para computación de propósito general en punto flotante (muchos problemas pero posible)
  - 2005: 1ª generación arquitectura **G80 & CUDA**, 1er intento de unificar procesador gráfico y **programación en C** (no API)
    - Nuevos términos en el ámbito de GPUs: procesador multihilos, comunicación entre hilos, etc.
  - 2006-2015: arquitecturas Tesla (2006), Fermi (2009), Kepler (2012), Maxwell (2013), Pascal (2014), Volta (2017), Turing (2018), Ampere (2021)

©Tarjeta NVIDIA GeForce 8800, chip: GT80





Características de las GPU para cómputo de propósito general

- GPU tienen la forma de aceleradores de las CPUs
  - GPU no tienen que realizar todas las tareas de una CPU convencional
  - La penalización de GPU se encuentra en la transferencia de datos y resultados entre la GPU y la CPU
- Los problemas que implementan las GPU tienen una memoria:
  - 100 MiB … 1 GiB (versus CPU: … varios TiB)
- Para evitar las penalizaciones de memoria
  - GPU utiliza multi-hilos: su objetivo es el mayor ritmo de terminación de tareas (bandwidth)
  - CPU utiliza cache multinivel: su objetivo es reducir el tiempo de cada tarea (latency)

CUDA: lenguaje programación para GPUs inspirado en C (*Compute Unified Device Architecture*)

- Jerarquía de memoria ~ Jerarquía de hilos

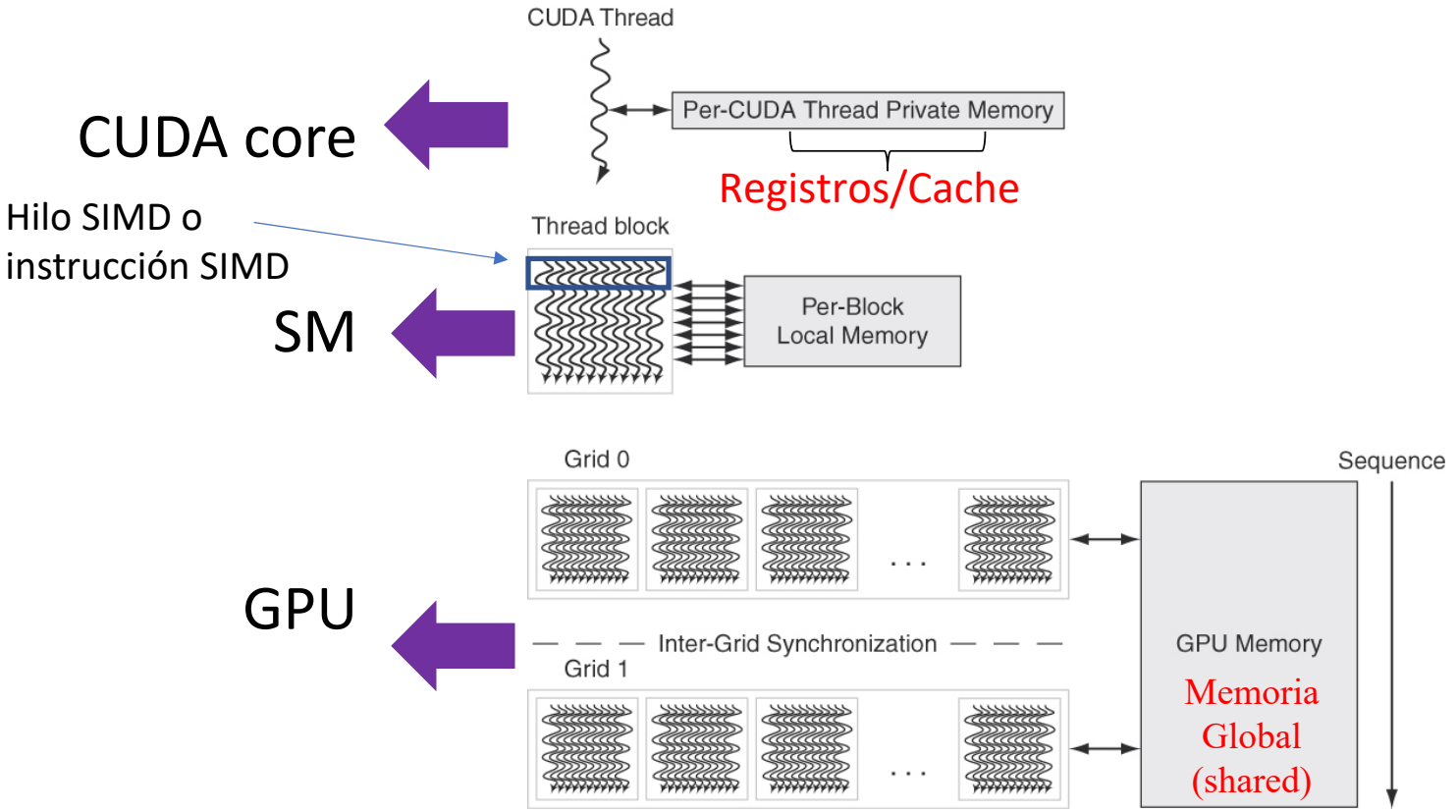


Figure 4.18 GPU Memory structures. GPU Memory is shared by all Grids (vectorized loops), Local Memory is shared by all threads of SIMD instructions within a thread block (body of a vectorized loop), and Private Memory is private to a single CUDA Thread.

Hilo: primitiva de programación, conjunto de instrucciones independientes del resto de hilos

Bloque: hilos que se ejecutan en paralelo en varios núcleos (cores) de un mismo multiprocesador

Grid: varios SM que ejecutan varios bloques de hilos en paralelo

Características de las GPU para cómputo de propósito general

- Tipos de memoria DRAM
  - GPU: DDR5 (mayor ritmo de transferencia), relativa poca capacidad almacenamiento
  - CPU: DDR4 (menor latencia), relativa alta capacidad almacenamiento
- Tipos de multiprocesador
  - GPU: cientos-miles núcleos de cómputo para dar cabida a muchos hilos
  - CPU: decenas de núcleos de cómputo
- GPU explota varios tipos de Paralelismo:
  - Multihilo: Paralelismo de hilos
  - MIMD: Paralelismo de programas
  - SIMD: Paralelismo de datos
  - Paralelismo de Instrucciones (incluso)

Arquitectura generalizada de NVIDIA para GPUs

- Optimizada para problemas donde se explota el paralelismo de datos
- Explota también el paralelismo de hilos con largas secuencias de instrucciones que solapan los accesos a memoria dentro de un multiprocesador de la GPU
- Cada multiprocesador dispone de un conjunto de unidades funcionales que operan en paralelo sobre datos distintos
- Cada GPU basada en Fermi puede disponer de múltiple multiprocesadores: 7,11,14,15
- En la GPU, existe un **planificador hardware** que asigna bloques de hilos a cada multiprocesador. Esto da compatibilidad entre versiones de Fermi

Arquitectura generalizada de NVIDIA para GPUs

- Dentro del multiprocesador existe otro planificador de hilos que indica cuándo un hilo está disponible para ser ejecutado en una unidad funcional. Cada hilo se puede ejecutar independientemente de los demás (multi-hilo de grano fino)
- Cada **instrucción SIMD** del hilo tiene cabida para realizar 32 operaciones en paralelo
- Una **línea** se corresponde con una de las unidades funcionales. El número de líneas no tiene que coincidir con el número de operaciones de cada instrucción SIMD
- Una GPU puede tener miles de líneas
- Una GPU puede proporcionar Tflops

Arquitectura generalizada de NVIDIA para GPUs

Multiprocesador multihilo (SM): procesador completo con su PC que es programado usando hilos

Unidad funcional (SP, núcleo de cómputo)

Building block

Memorias cache

Red de interconexión con memoria

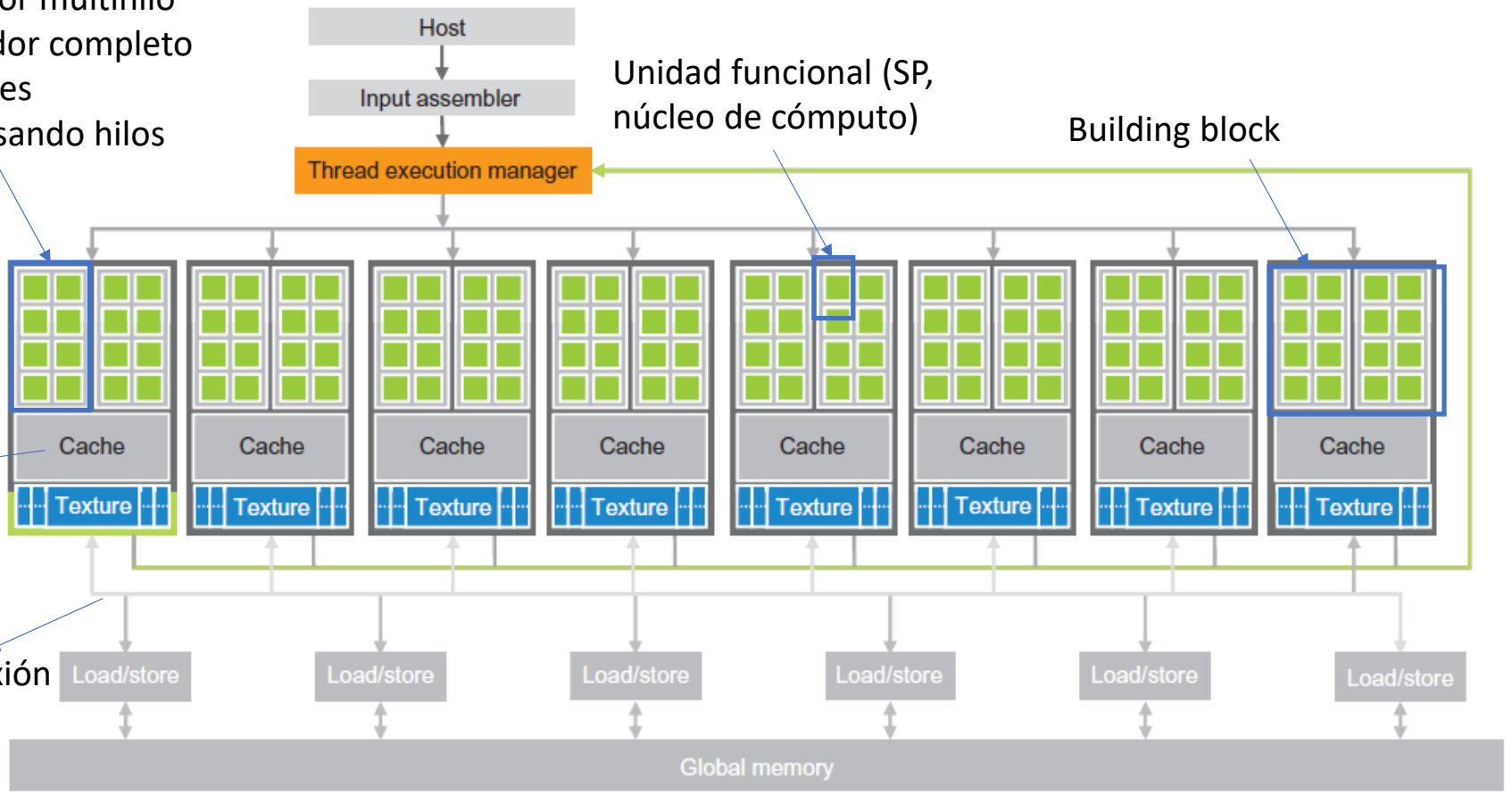


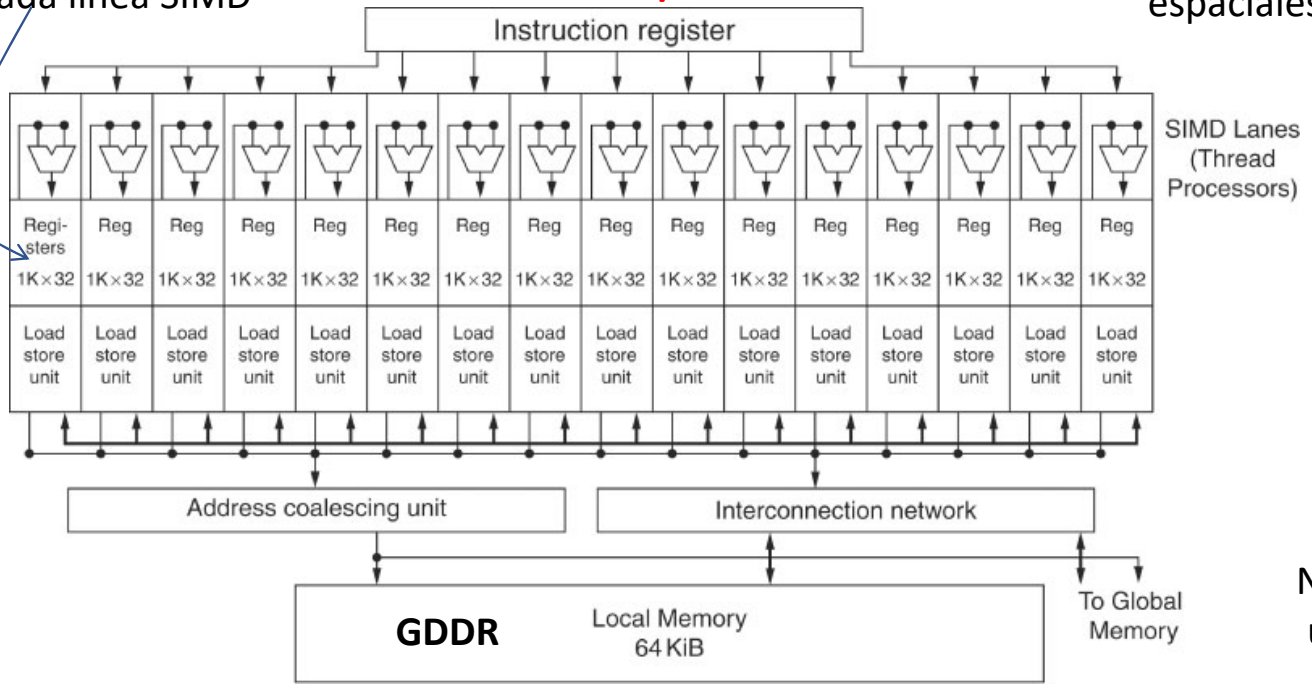
FIGURE 1.2

Architecture of a CUDA-capable GPU.

Diagramas de la microarquitectura Tesla

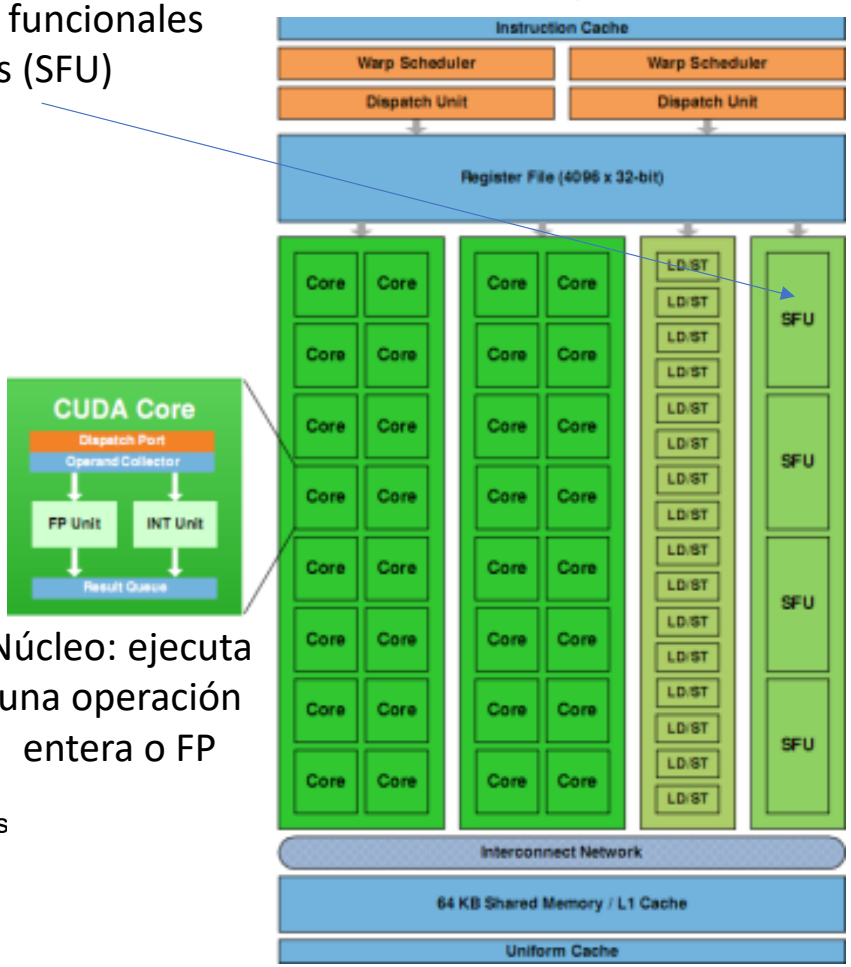
Muchos registros:
1 K registros para cada línea SIMD

16 líneas/núcleos



Unidades funcionales espaciales (SFU)

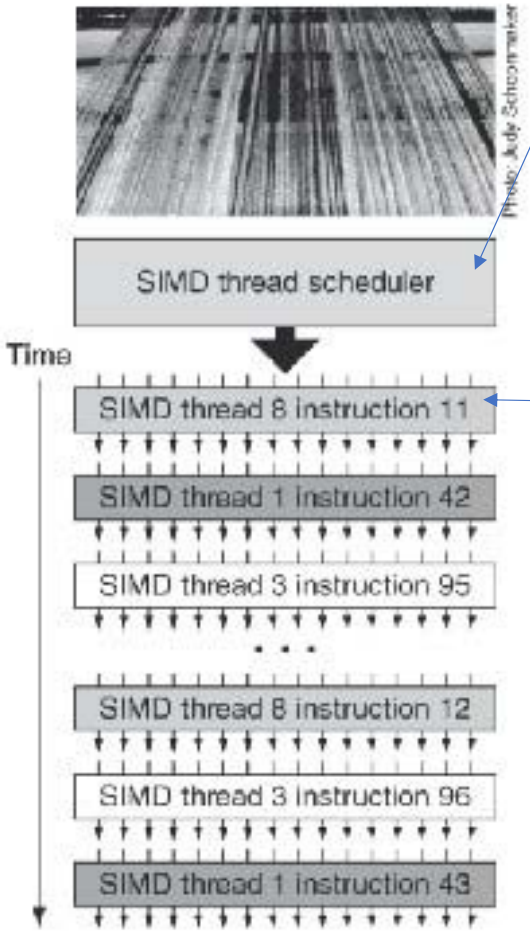
32 líneas/ núcleos



Núcleo: ejecuta una operación entera o FP

FIGURE 6.9 Simplified block diagram of the datapath of a multithreaded SIMD Processor. It has 16 SIMD lanes. The SIMD Thread Scheduler has many independent SIMD threads that it chooses from to run on this processor.

Planificador de instrucciones SIMD

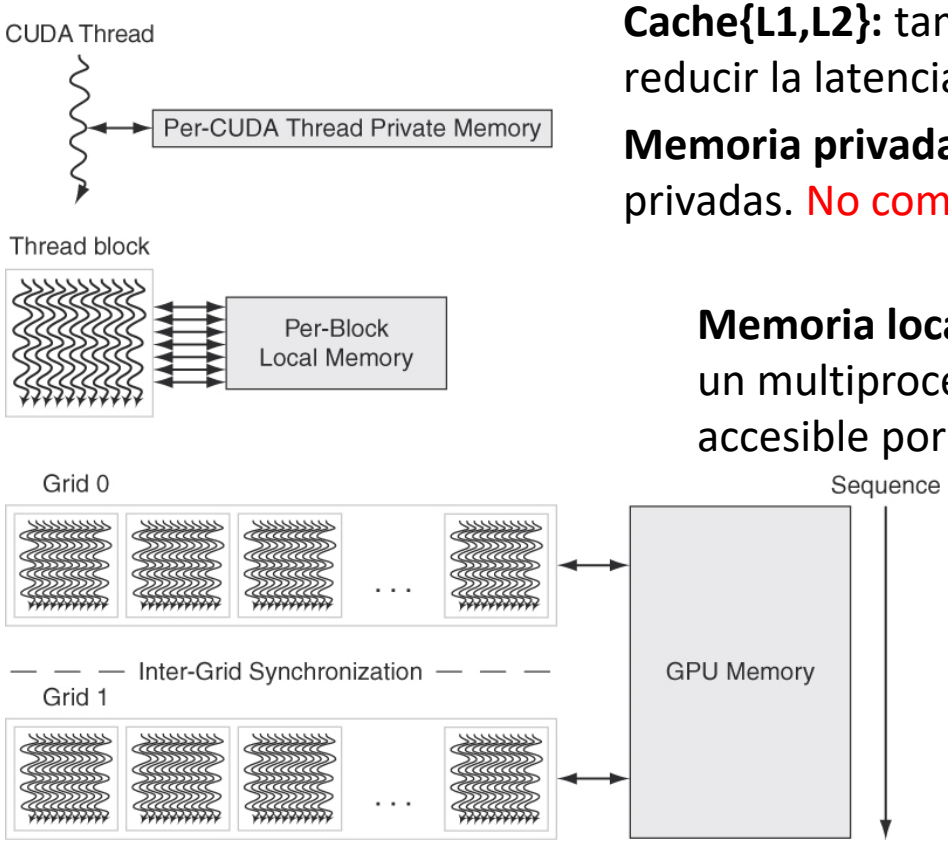


Planificador de hilos SIMD que selecciona qué hilo SIMD se envía a ejecutar a las unidades funcionales en cada ciclo de reloj. Permite solapar la latencia de los accesos a memoria DRAM

Instrucción SIMD que pertenece a un hilo de instrucciones SIMD
Se pueden alternar su ejecución debido a que los hilos son independientes entre sí. Se pueden mantener en vuelo hasta 48 hilos de instrucciones SIMD.
Cada hilo SIMD puede disponer hasta 64 registros con 32 componentes de 32 bits cada uno

Figure 4.16 Scheduling of threads of SIMD instructions. The scheduler selects a ready thread of SIMD instructions and issues an instruction synchronously to all the SIMD Lanes executing the SIMD thread. Because threads of SIMD instructions are independent, the scheduler may select a different SIMD thread each time.

Jerarquía de memoria en GPU



Registros: muy numerosos para mantener variables de un gran número de hilos y poder solapar la latencia de los accesos a memoria

Cache{L1,L2}: tamaños más pequeños que las caches del host; usadas para reducir la latencia de los accesos a la memoria DRAM y el consumo de energía

Memoria privada: para uso de pila, copia de contenido de registros, y variables privadas. No compartida entre hilos del mismo multiprocesador

Memoria local: memoria DRAM compartida por el bloque de hilos de un multiprocesador pero no compartida entre multiprocesadores. No accesible por parte del host.

Memoria global: memoria DRAM compartida por los hilos de todos los multiprocesadores; latencia grande. Puede ser accedida por el procesador del sistema (host).