

Parallel Programming

GPU Architecture

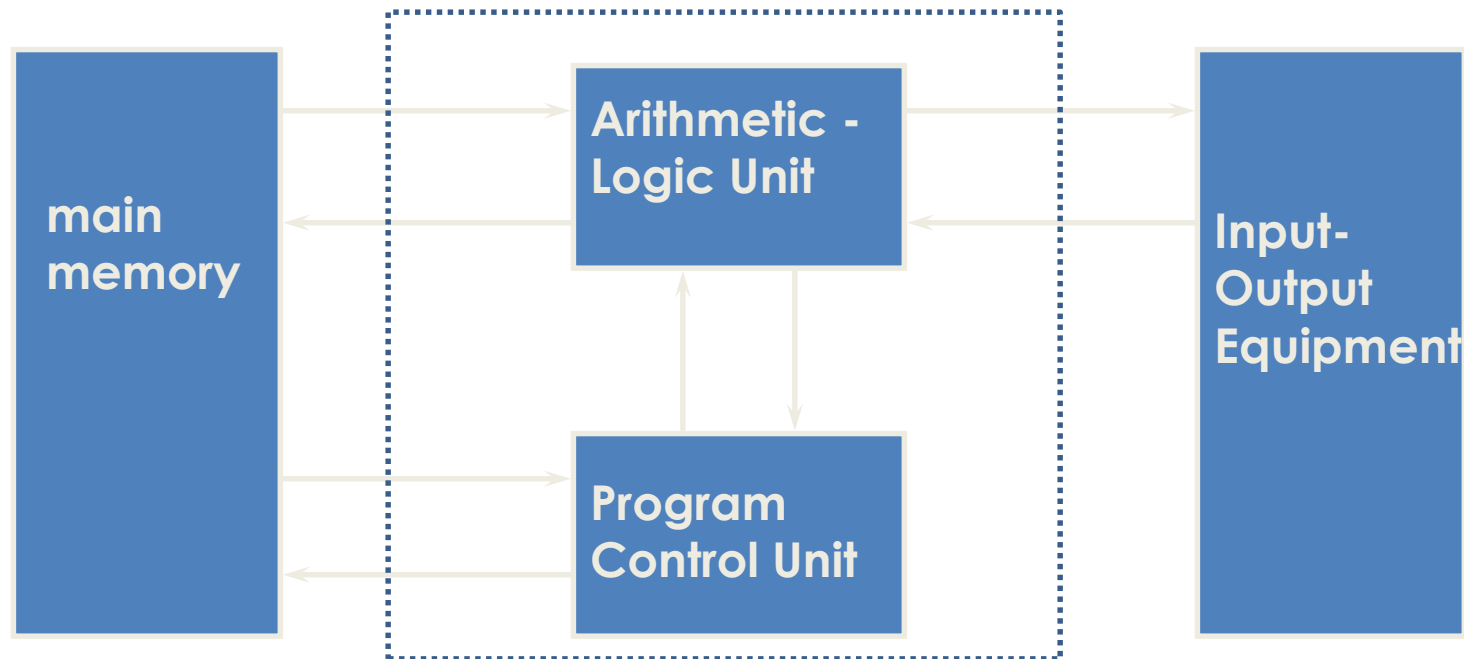
Acknowledgement: Some graphics and examples are taken from various online resources, including NVIDIA web sites and lecture slides of Prof. Wen-mei Hwu.

Overview

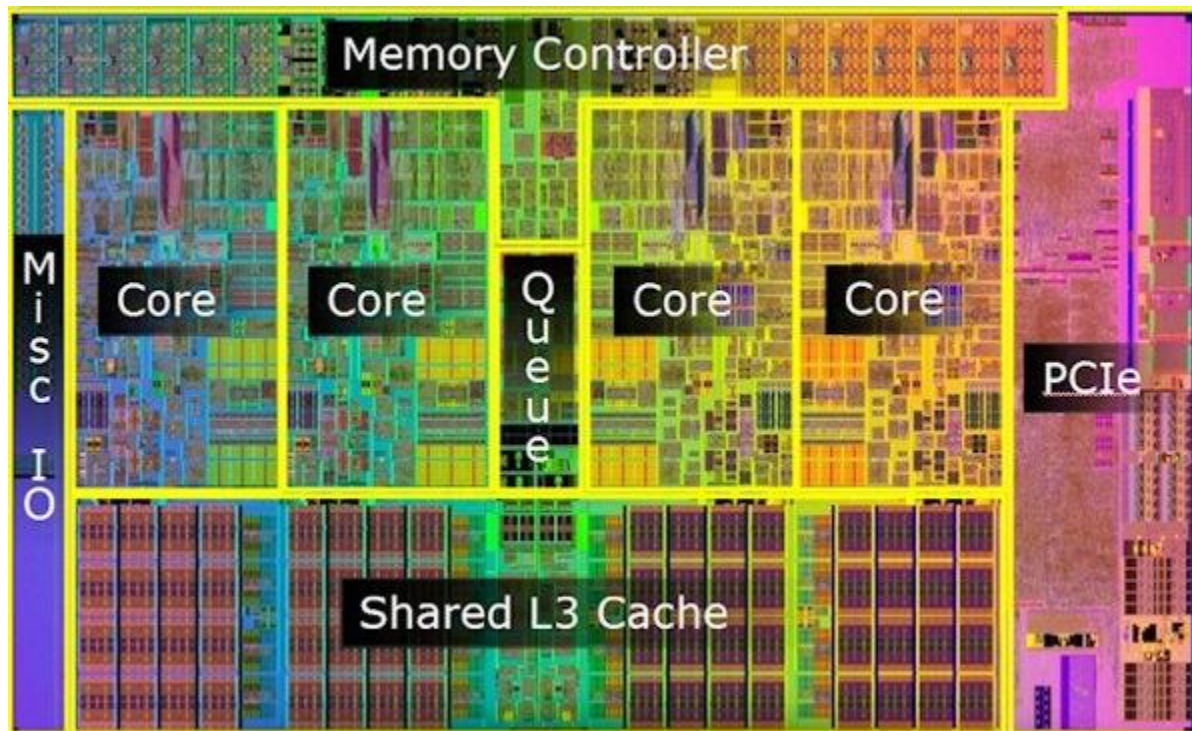
- Modern GPUs have a massively parallel architecture.
 - We use NVIDIA CUDA-enabled GPU as example.
- How are they different from CPUs?
- Where do GPUs fit in parallel architectures?

Von Neumann Machine (1947)

- Fetch-and-Execute cycle on the CPU:
 - Fetch instructions and data from memory
 - Execute instructions on ALU



Modern CPU Architecture

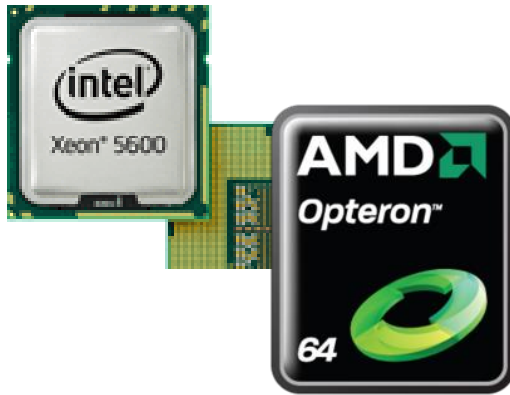


Intel i5/i7. Source: Intel

Parallelism in CPUs

- Multiple physical cores
- Hyper Threading (HT) or Simultaneous Multithreading (SMT)
 - Map each physical core to two logical processors
- Instructional level parallelism (ILP)
 - Divide each instruction into stages and pipeline multiple independent instructions by stages

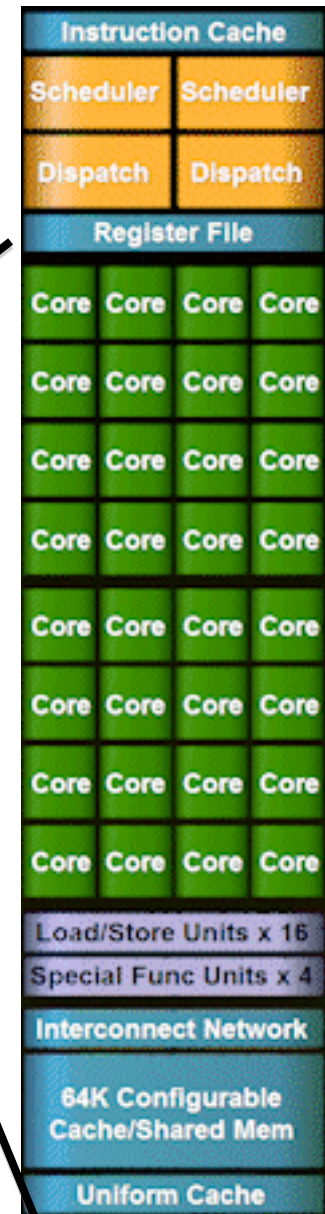
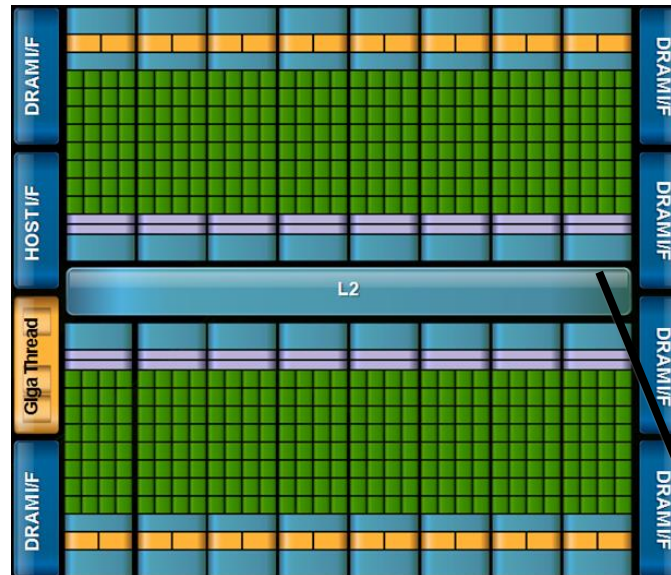
Graphics Processing Unit (GPU)



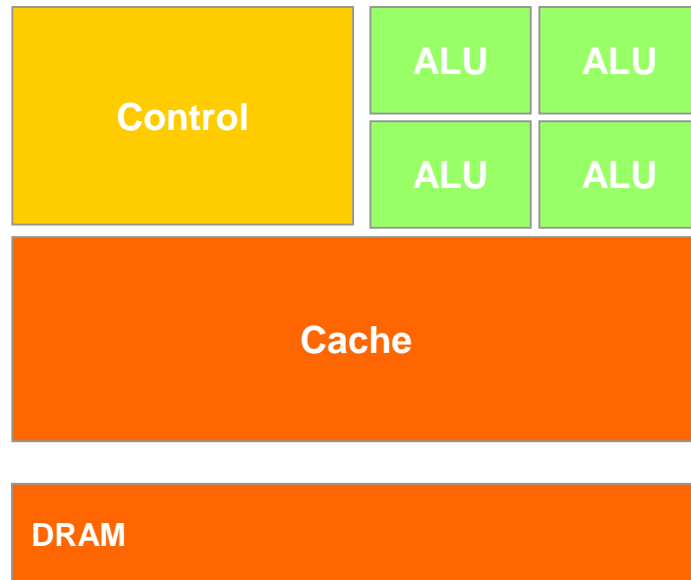
- Traditionally used for game (3D rendering) applications
- Currently major accelerators for general-purpose computing applications that exhibit data parallelism
- Work as co-processors, i.e., rely on the CPU for task control, memory allocation, data transfer, etc.

NVIDIA GPU

- General-purpose
- Fully programmable
- Massively parallel

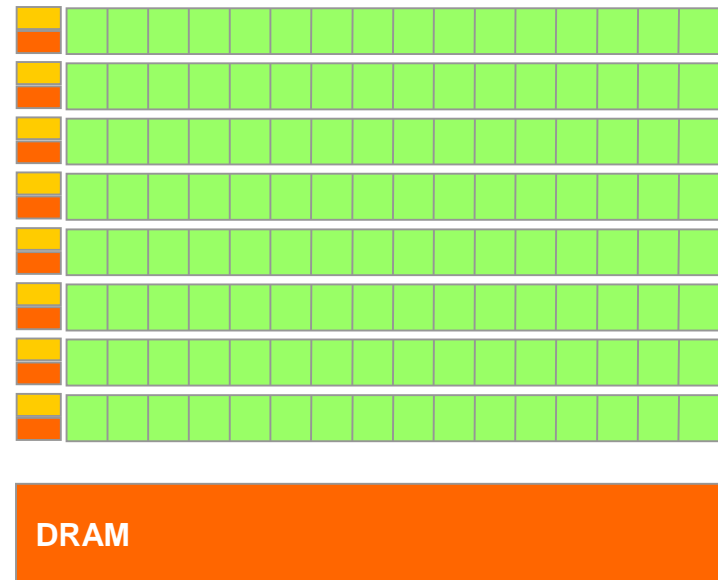


Comparison of CPU and GPU



CPU

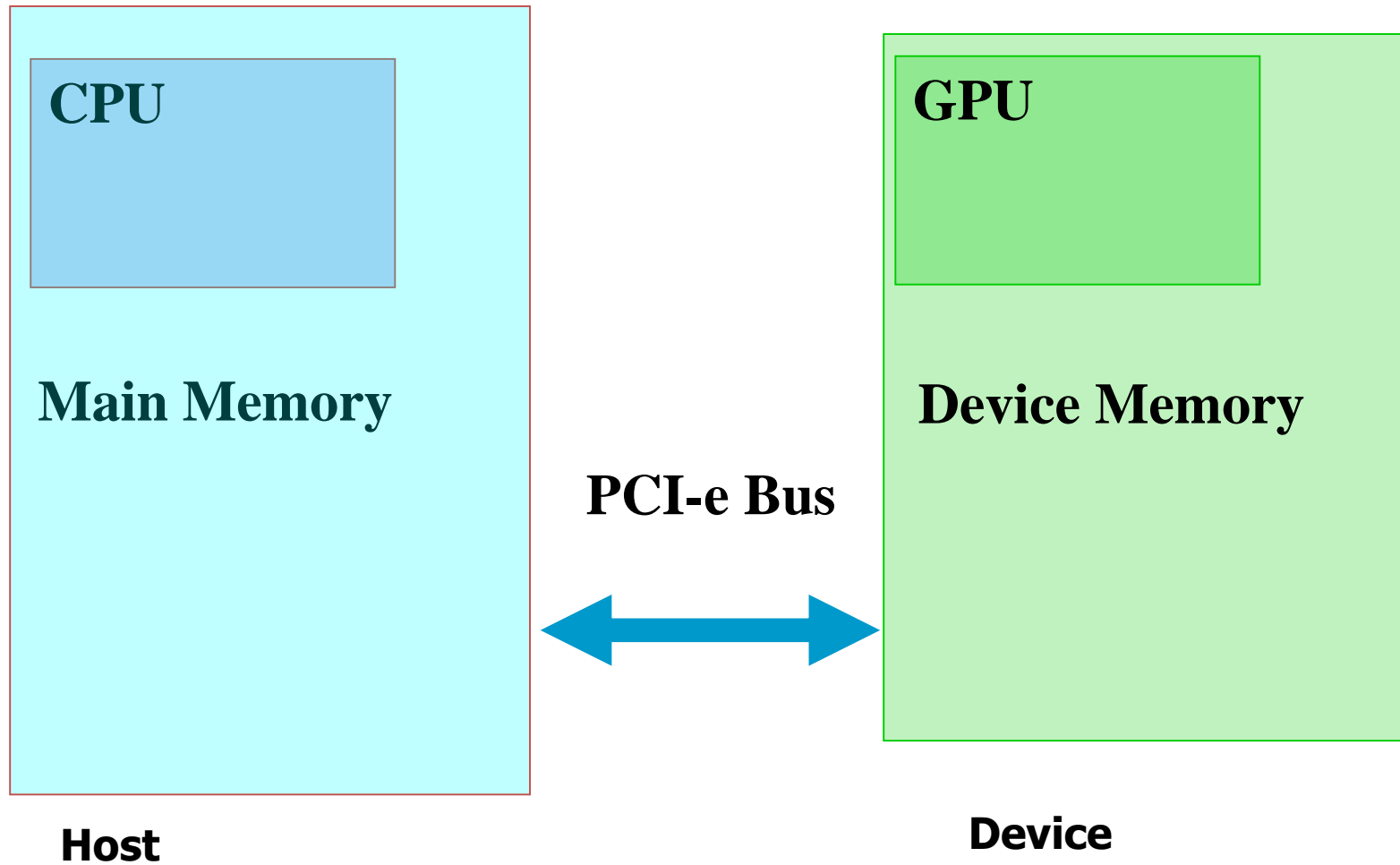
Latency oriented



GPU

Throughput oriented

GPU and CPU



Classification of Parallel Architecture

S I S D

Single Instruction, Single Data

A serial (non-parallel) computer

Oldest type of computers

S I M D

Single Instruction, Multiple Data

A type of parallel computer

Synchronous execution

Suitable for data-parallel applications

Examples: GPUs

M I S D

Multiple Instruction, Single Data

A type of parallel computer

A single data stream is fed into multiple processing units.

Few actual examples

M I M D

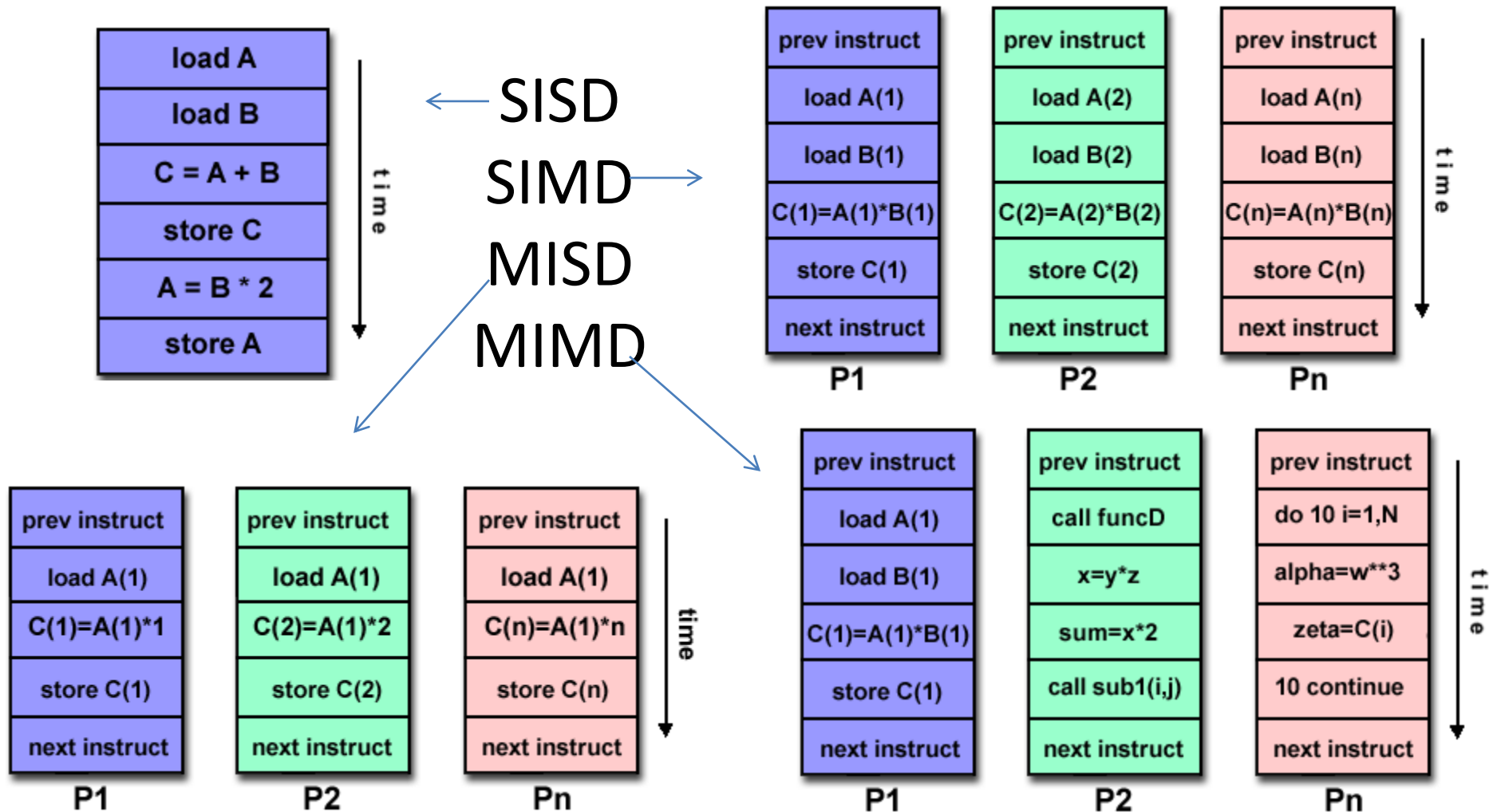
Multiple Instruction, Multiple Data

most common type of parallel computer

synchronous or asynchronous

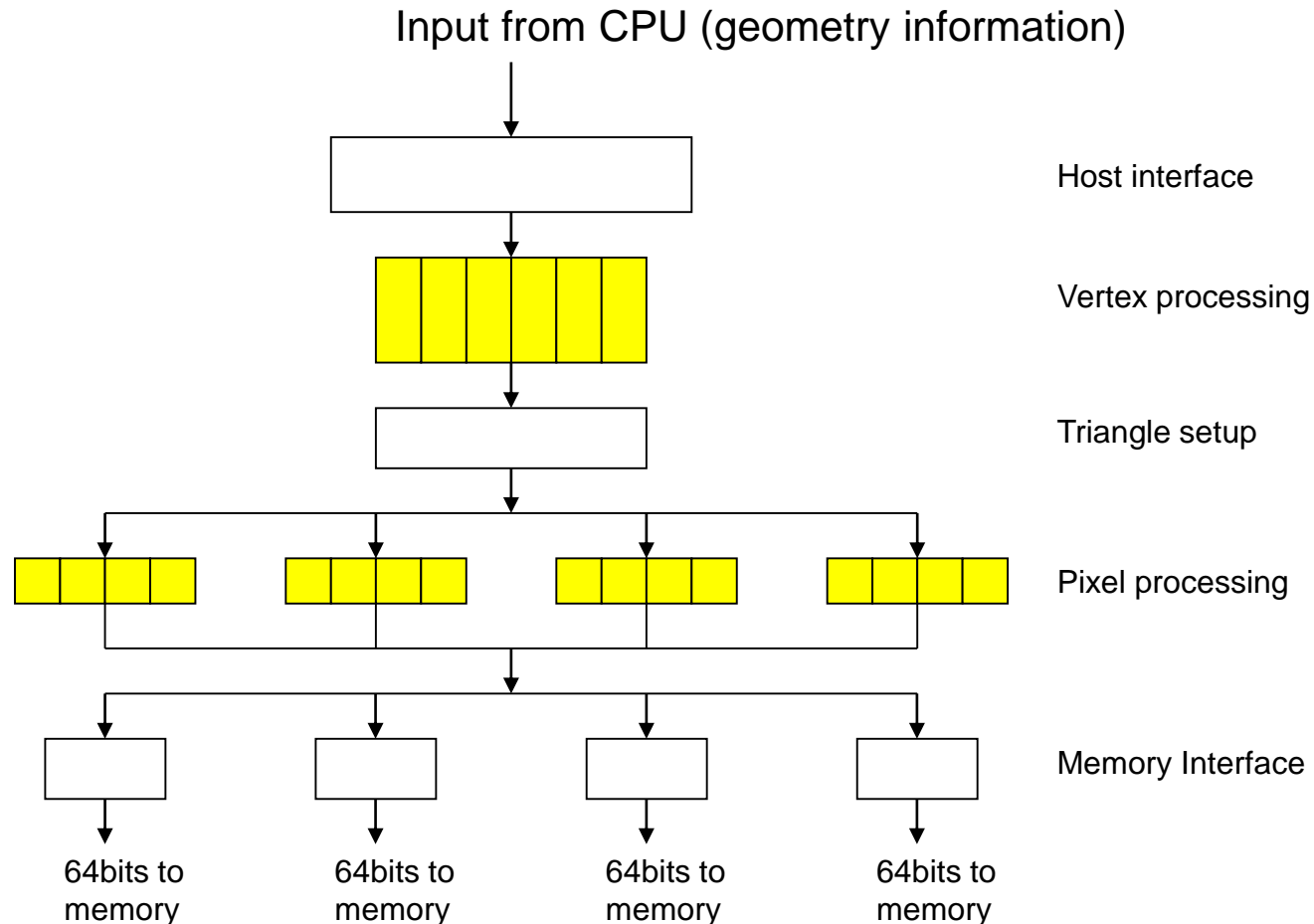
Examples: Supercomputers, clusters, multicore PCs

Illustrations of Execution Flows



Example adapted from https://computing.llnl.gov/tutorials/parallel_comp

Traditional GPU Pipeline



Traditional graphics hardware abstraction
Limited programmability (only highlighted stages programmable)

SIMT Architecture of NVIDIA GPU

- Single Instruction Multiple Threads
 - Instruction-level parallelism within a single thread
 - Thread-level parallelism through simultaneous hardware multithreading
 - Each multiprocessor creates, manages, schedules, and executes CUDA threads in groups of 32, called **warps**.
 - Branch divergence occurs only within a warp; different warps execute independently regardless of whether they are executing common or disjoint code paths.

SIMT vs SIMD

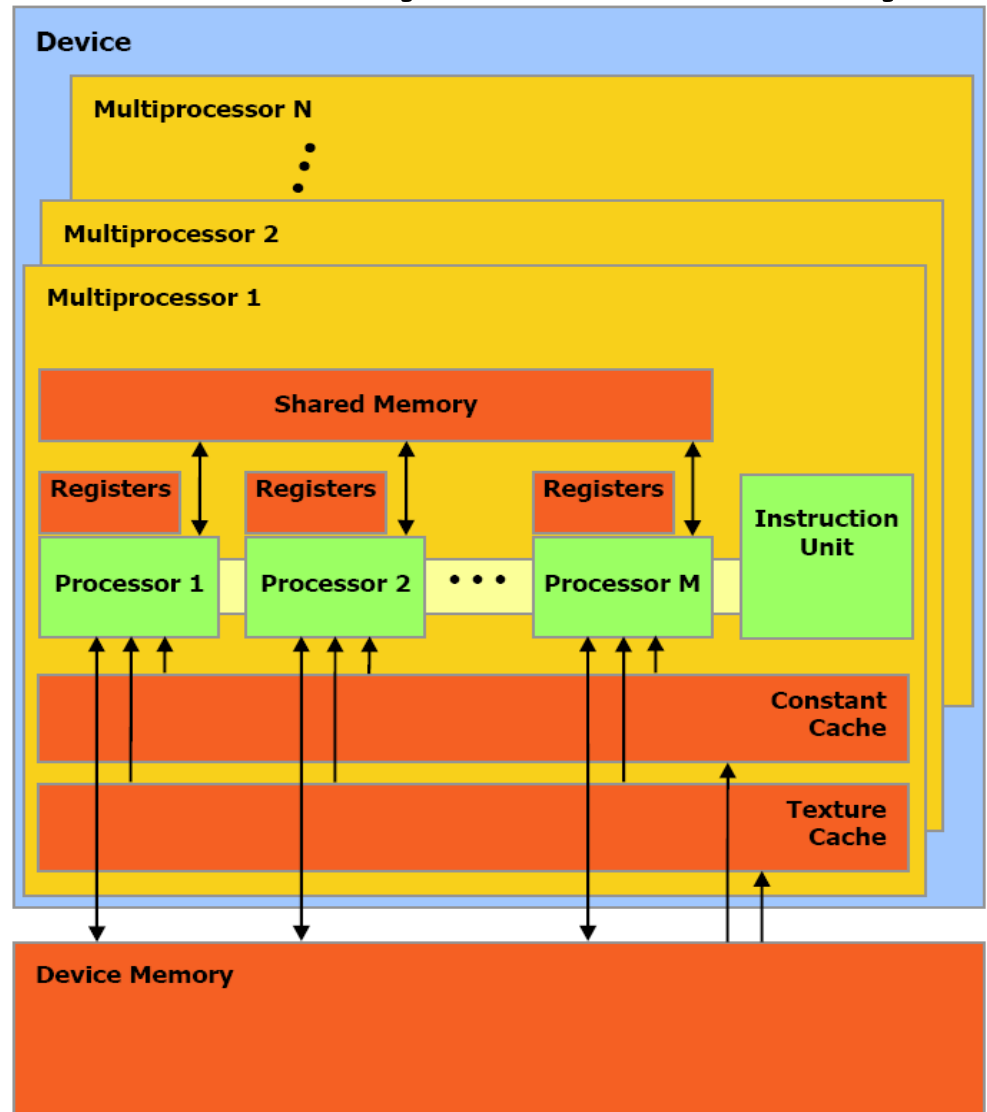
- Similar: a single instruction controls multiple processing units.
- Different:
 - SIMD vector organizations expose the SIMD width to the software
 - E.g., data items are required to aligned into vectors of a fixed size.
 - SIMT instructions specify the execution and branching behavior of a single thread
 - For simplicity, the programmer can ignore the SIMT behavior; however, substantial performance improvements can be realized by taking care of it.

CPU vs GPU Threads

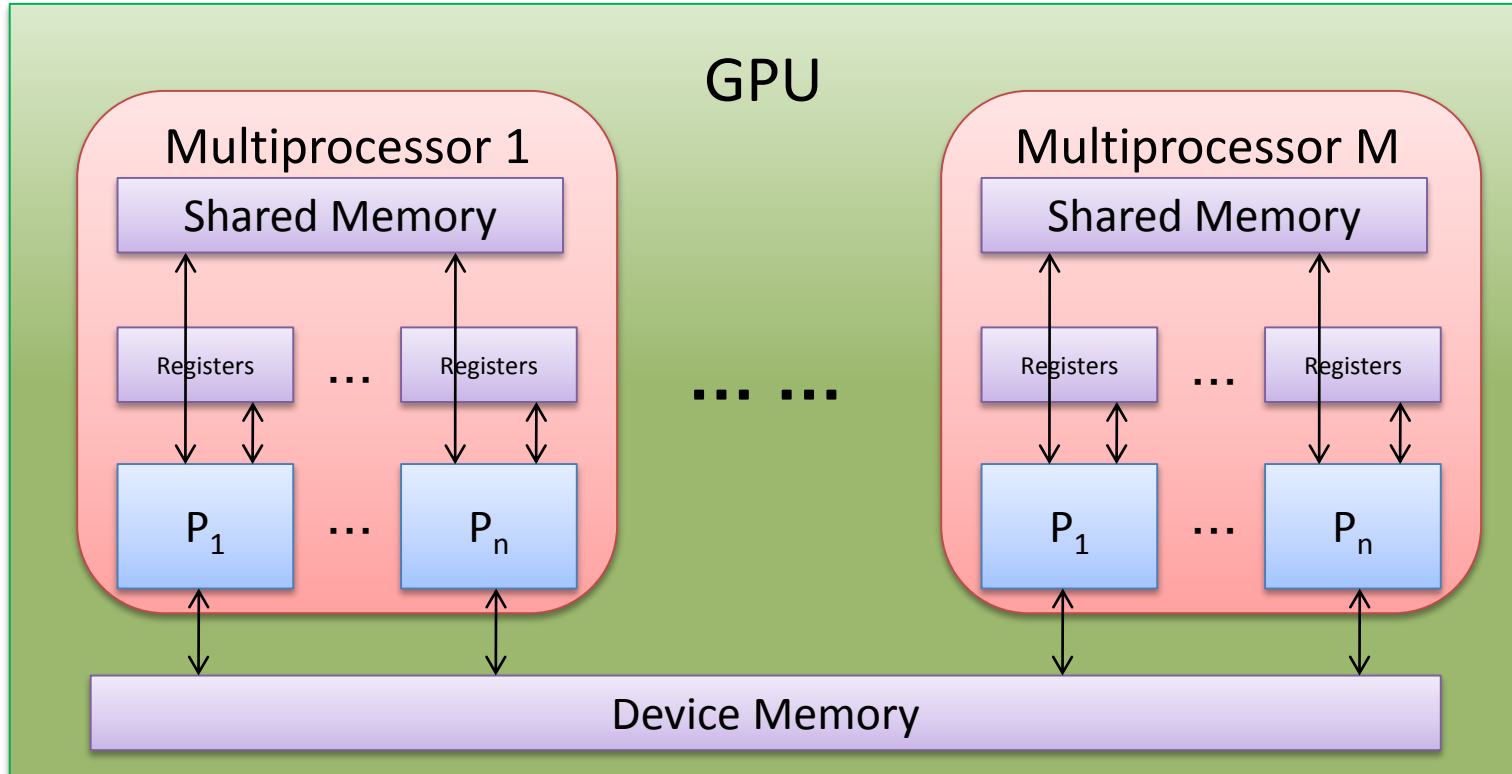
- CPU threads are much more heavyweight than GPU threads to create and maintain.
- Typically there are tens of concurrent CPU threads in a CPU program whereas there can be 1,000s to 10,000s of concurrent GPU threads in a GPU program.
- In a CPU program, threads may execute different code; in a GPU program, typically all threads execute the same piece of code (called a **kernel**).

NVIDIA GPU Memory Hierarchy

- Registers: smallest, fastest on-chip memory
- On-chip shared memory: small, fast, software-managed consistency
- Off-chip device memory: high-bandwidth, high-latency

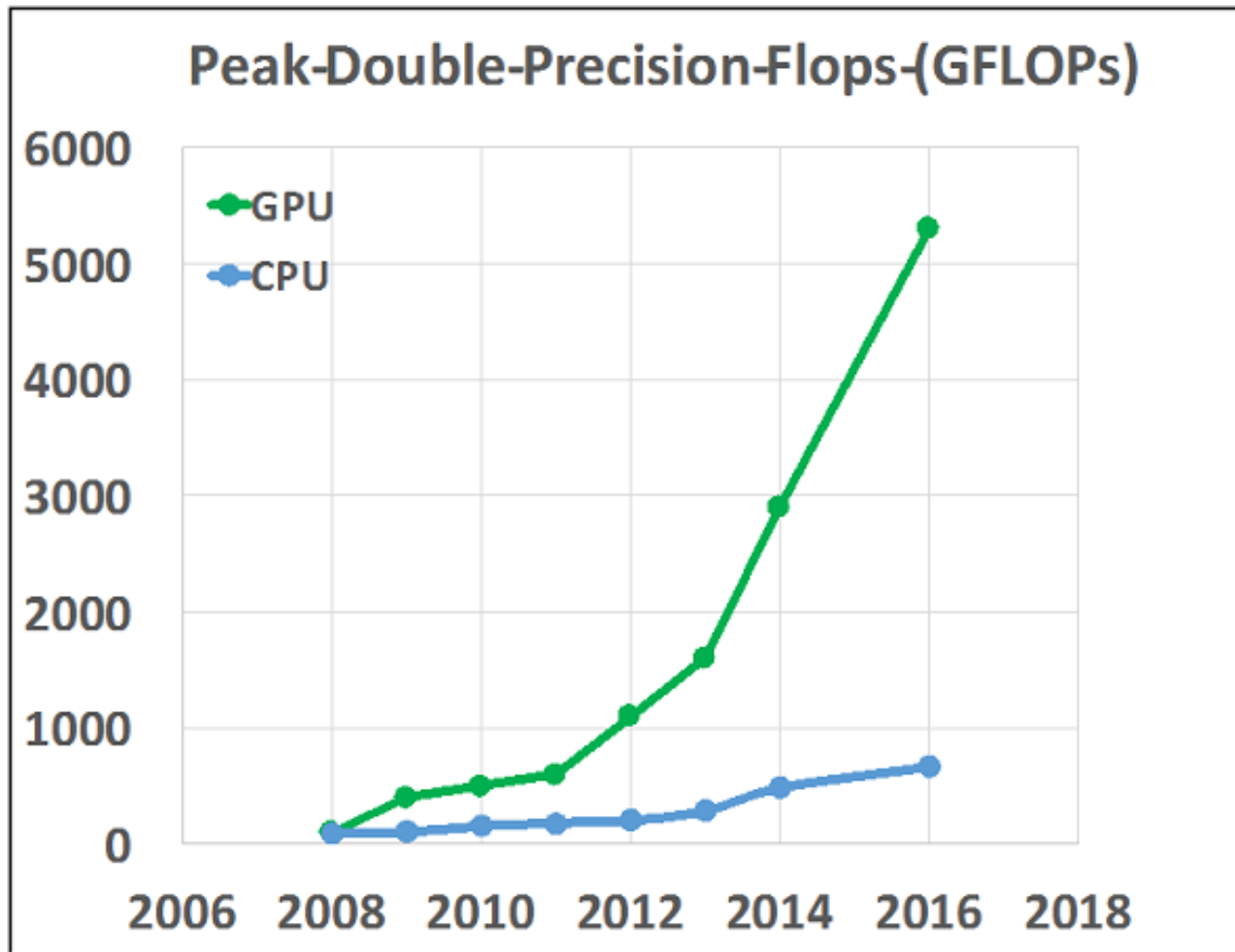


Modern GPU Architecture



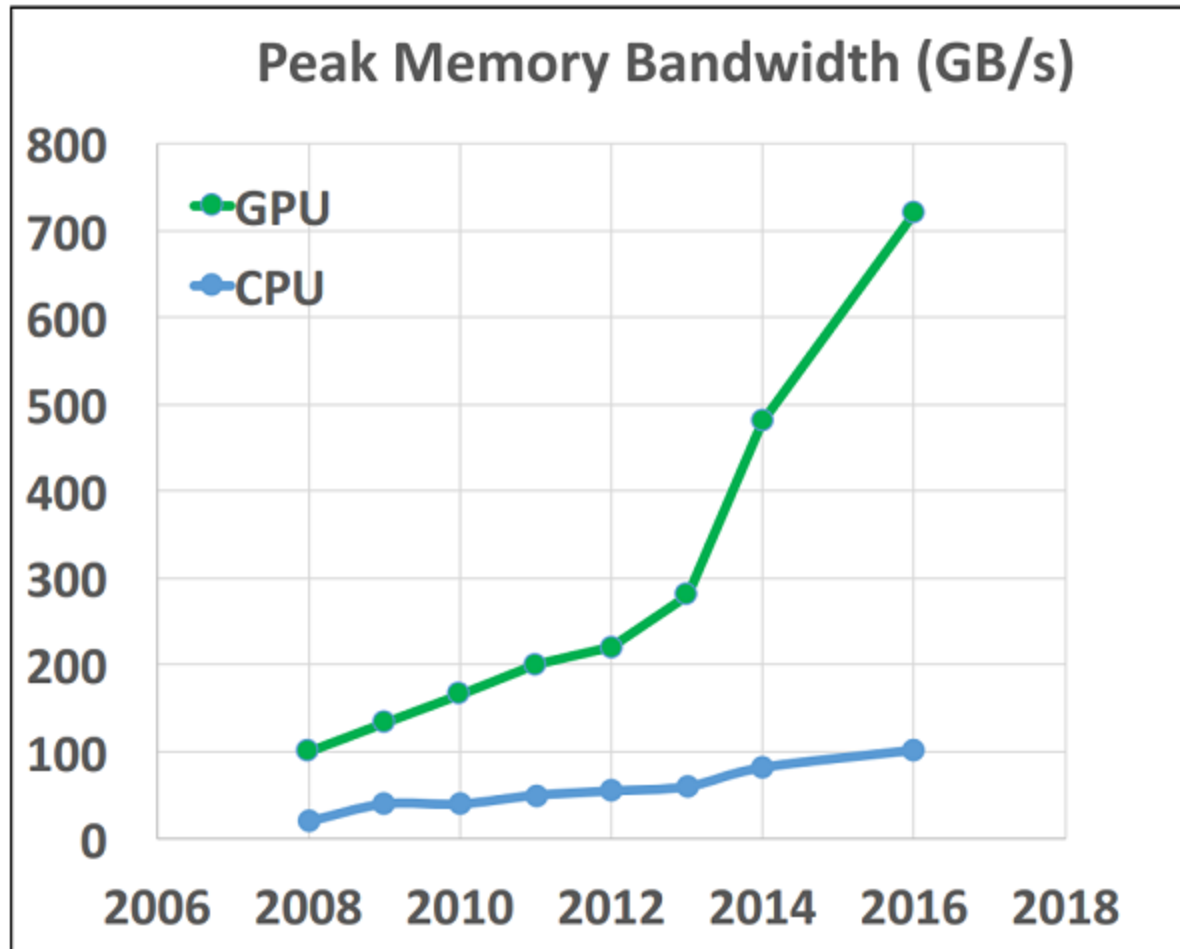
- 10s~100s of identical streaming multiprocessors (SMs)
 - 10s of identical unprocessors (cores) in a multiprocessor
- => Hundreds to thousands of cores, or thread processors

Performance: GPU versus CPU



Source: <https://www.hpcwire.com/2016/08/23/2016-important-year-hpc-two-decades/>

Memory Bandwidth: GPU vs CPU



Source: <https://www.hpcwire.com/2016/08/23/2016-important-year-hpc-two-decades/>

GPGPU Applications

- Media and entertainment
 - Adobe Photoshop, Apple Finalcut, ArcVideo Live
- Weather and climate forecast and simulation
- Molecular dynamics
- Computational finance
- Bioinformatics
- Computational physics and chemistry
- ...

Issues about GPU Architecture

- Co-processor nature
- Bus transfer bandwidth
- Suitable mainly for data-parallel applications
- Unusual memory hierarchy
- Programmer-responsible correctness
- Programmer-responsible optimizations
- High power consumption

Summary

- GPUs are highly parallel architectures.
 - Single Instruction Multiple Thread
 - Support a massive number of threads
 - Threads scheduled in unit of warps
- They are suitable for many data-parallel, computation-intensive applications.
- Programming GPU requires architectural considerations.