GPU = a Throughput-Oriented Processor

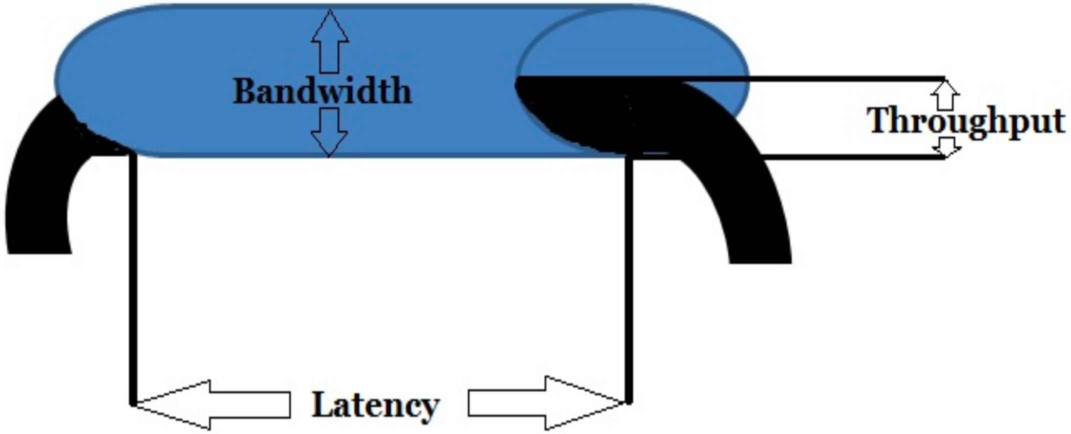
Stefano Markidis and Sergio Rivas-Gomez

Three Key-Points

1. Processors can be either *latency-oriented* or *throughput-oriented* processors.
2. GPUs are the leading exemplars of modern throughput-oriented architectures.
3. The GPU throughput-oriented architecture is based on three main design choices: many simple processing units, hardware multithreading, SIMD execution.

Two Fundamental Measures of Processor Performance

- **Task latency** = time elapsed between the initiation and completion of some task
- **Task throughput** = total amount of work completed per unit time

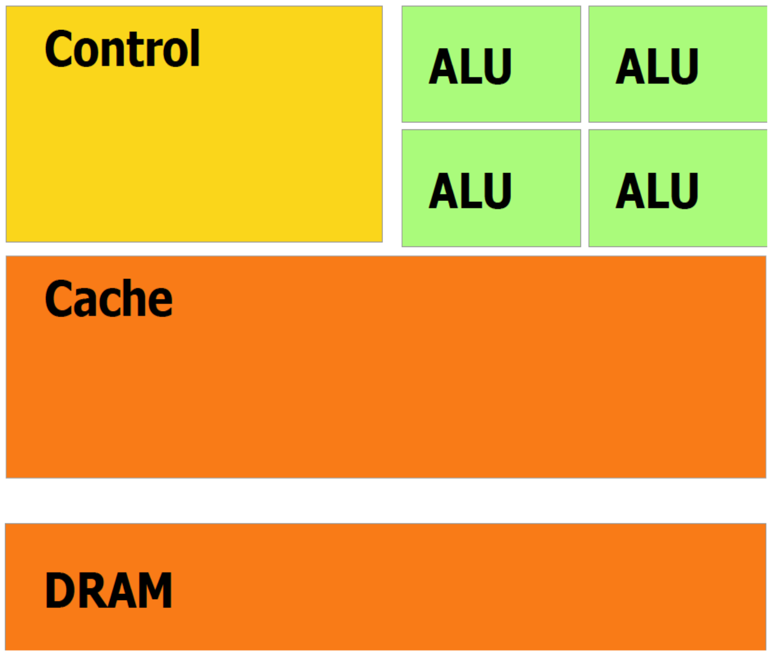


Courtesy of: http://perfmatrix.blogspot.se/2016/12/latency-bandwidth-throughput-responsetime.html

Traditional Scalar Microprocessors are Latency-oriented Architectures

- Their goal is to **minimize the running time of a single sequential program** by avoiding task-level latency whenever possible
- Major architectural design choices:
    - Memory caches
    - Out-of-order execution
    - Speculative execution
    - Pipelining

Latency-Oriented Architecture

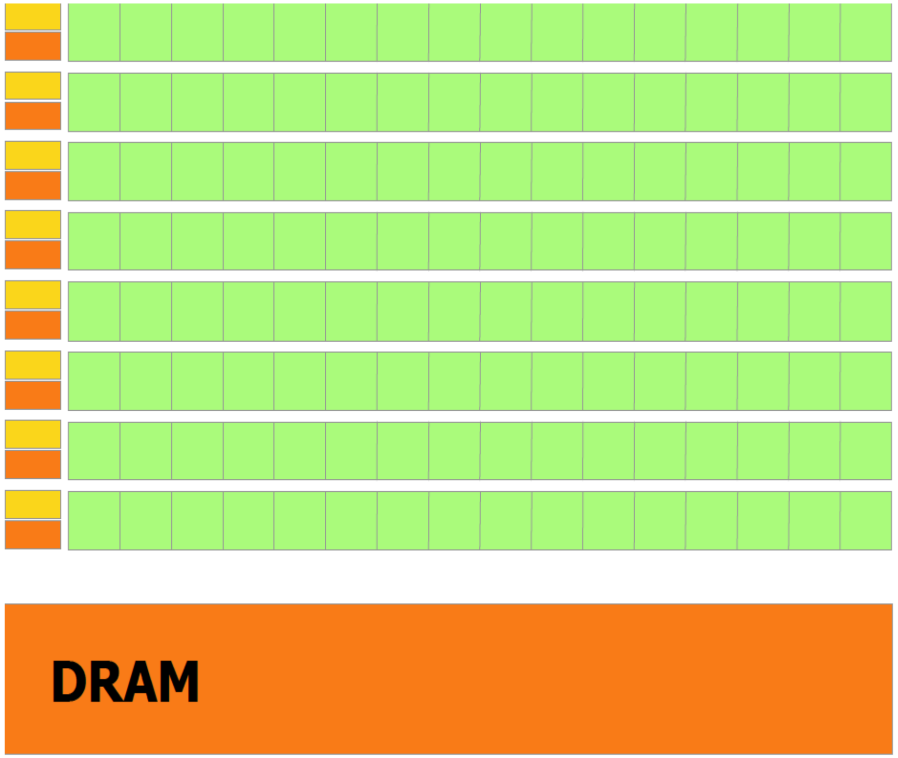


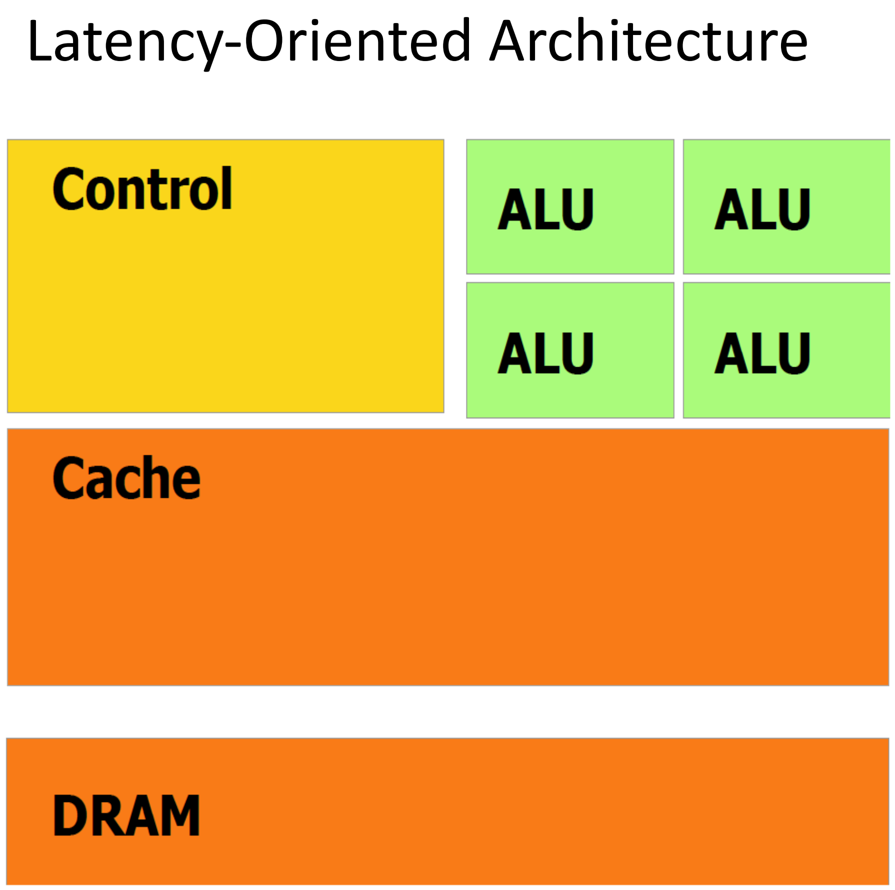
Example: Intel Pentium IV was aggressively latency-oriented architecture (2000)

Throughput-Oriented Processors …

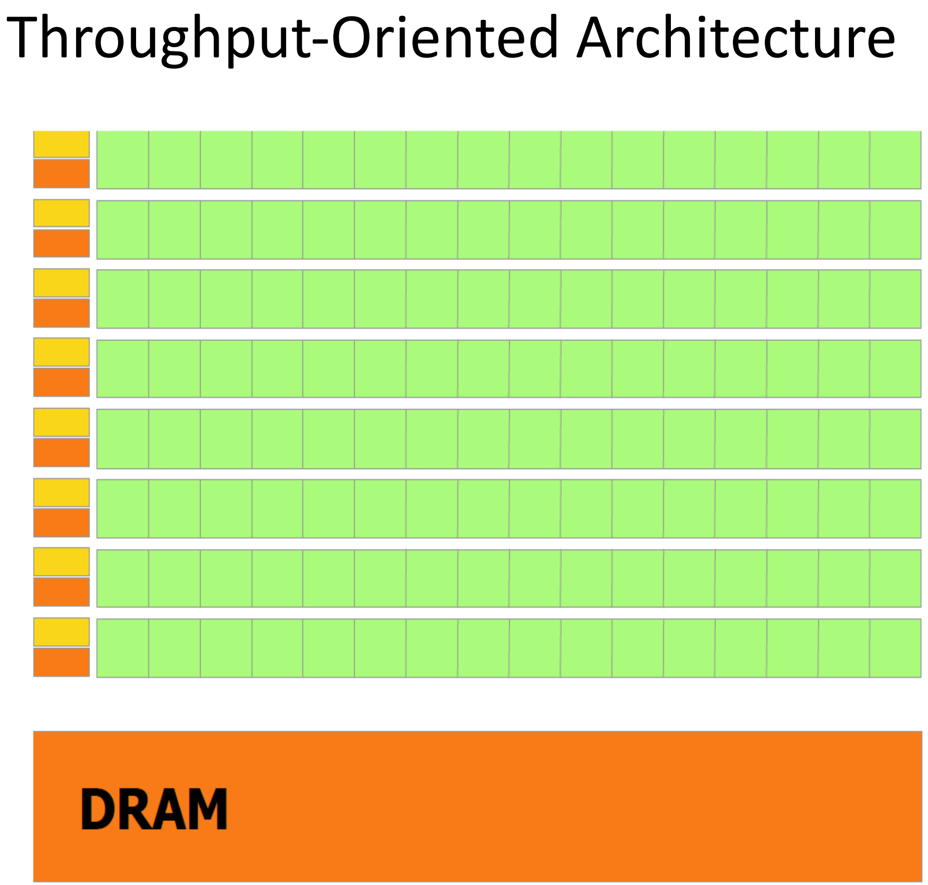
… tackle problems and workloads in **which parallelism is abundant**, yielding design decisions that are different from more traditional latency-oriented processors

Throughput-Oriented Architecture





Latency-Oriented Architecture
Control
ALU
ALU
ALU
ALU
Cache
DRAM



Throughput-Oriented Architecture
DRAM

GPUs are Exemplar of Throughput-oriented Architectures

- GPUs focus on executing on **parallel workloads** attempting to maximizing the **total throughput**, even though not minimizing the latency of the single task
- Right design decision in many domains:
  - Real-time computer graphics, video processing, medical imaging, deep learning
- Might not be the right decision when solving other problems …

Throughput-Oriented Architectures and GPUS …

… rely on three architectural features:

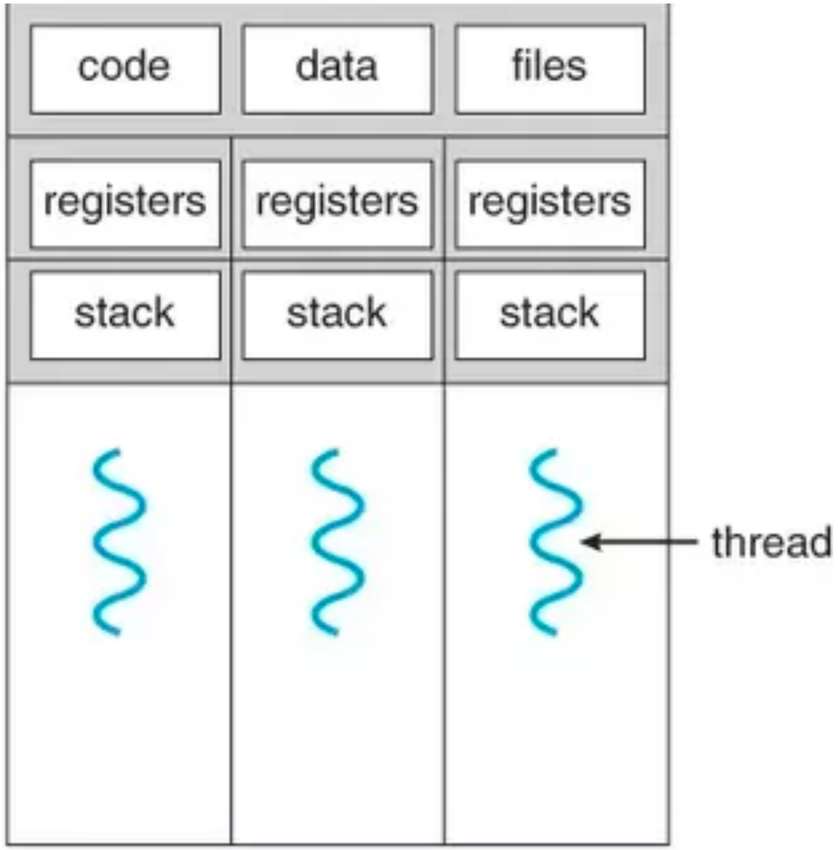
1. Many simple processing units
2. Hardware threads
3. SIMD execution

Many Simple Processing Units

- Execute instructions in the order they appear in the program (**in-order execution**)
- Throughput-oriented architectures have thousands of small cores, excelling at **regular math-intensive** work
  - **Lots of ALUs, little hardware for control.**

Hardware Multithreading

- Computation can be divided into collection of concurrent sequential tasks executed across many threads.
- Thread can be seen as **virtualized scalar processor** with a program counter, register file and associate processor state.
- Multithreading can be implemented in software (OS) or hardware
  - Throughput-oriented architectures have implemented in hardware



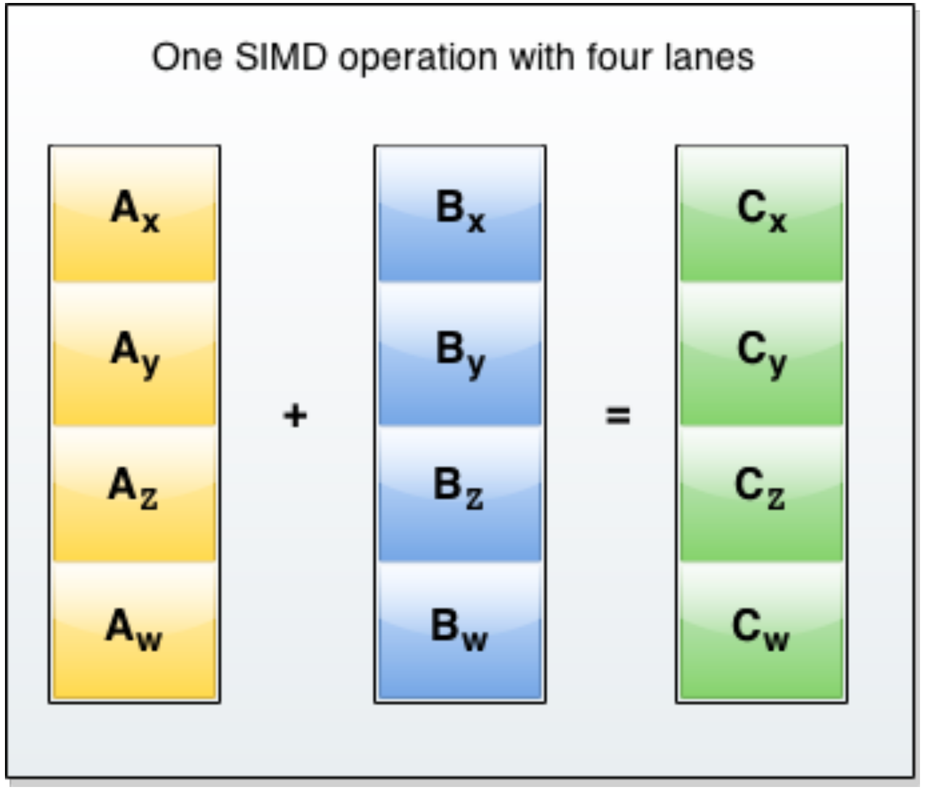
Hardware Multithreading Hides Latency

Long-latency operations of a single-thread can be hidden or covered by **ready-to-run work** from another thread, examples:

- Thread prevented to run because waiting for instruction from pipelined functional units
- Thread prevented to run because waiting data from DRAM

SIMD Execution

- Parallel processors employ some of form of Single-Instruction, Multiple Data (**SIMD**) execution to **increase the throughput**.
- Issuing a single instruction in a SIMD machine applies the given operation to potentially many data operands.



To Summarize

We focused on three major points:

1. We divided processors between either *latency-oriented* or *throughput-oriented* processors.
2. GPUs are throughput-oriented architectures targeting problems with lot of parallelism.
3. The GPU throughput-oriented architecture is based on three main design choices: many simple processing units, hardware threads to hide latency and SIMD execution.