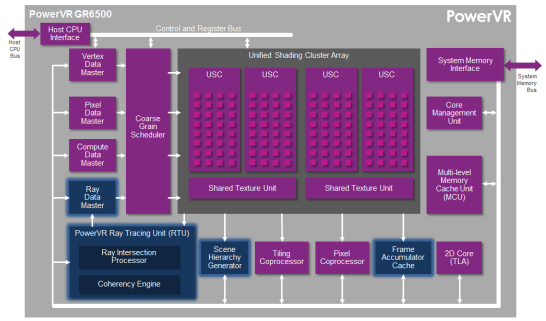
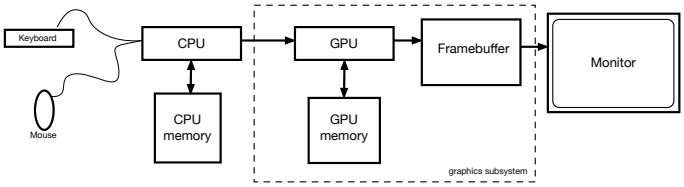
Graphics Hardware



Recall

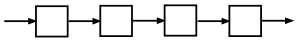


Making a CPU go fast

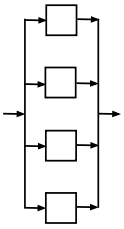
- ALU much faster than main memory (>100x)
- caches
- multiple processes
- multithreading (switch threads per clock cycle)
  needs multiple register files
- multiple cores
- SIMD

How to go fast

- pipeline

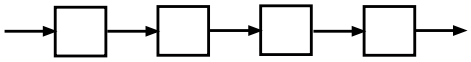


- parallel



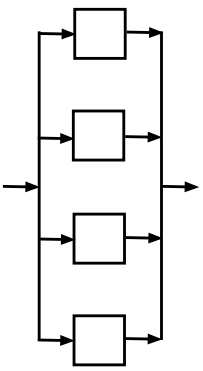
Pipeline

- good when you can divide work as a series of steps, each one *after* the other
- e.g.: ALU
- throughput vs latency

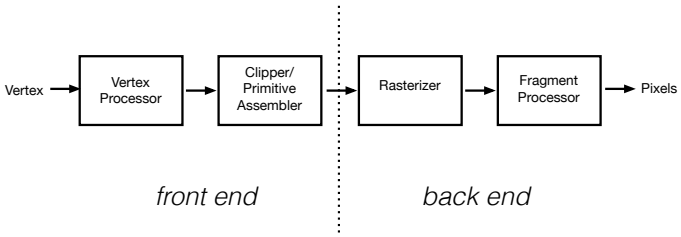


Parallel

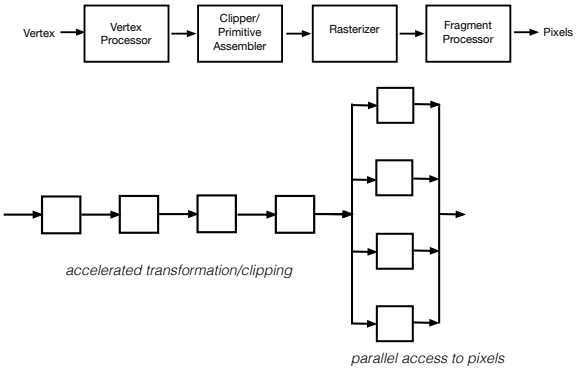
- good when you can divide work as a series of steps, each *independent*
- e.g.: multiple processes/ threads



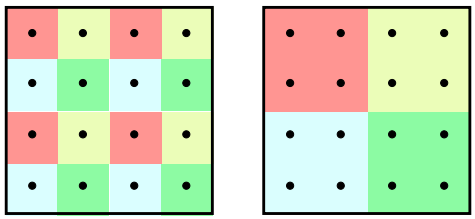
Graphics Pipeline



Graphics Pipeline

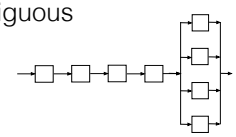


Dividing the Frame Buffer

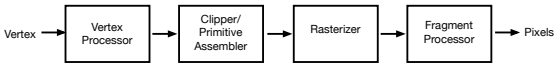


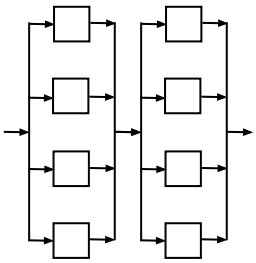
Interleaved

Contiguous



Evolving Graphics Pipeline

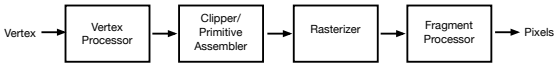




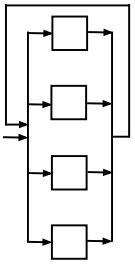
accelerated transformation/clipping

parallel access to pixels

Evolving Graphics Pipeline

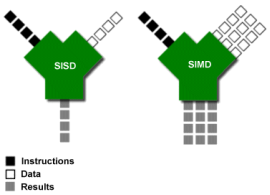


unified shaders

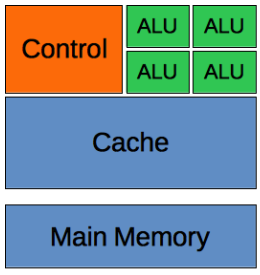


SIMD

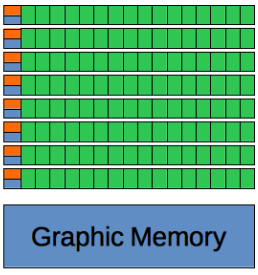
- single instruction is executed on multiple ALUs at the same time
- can work on data in parallel (vertices/fragments)



SIMD



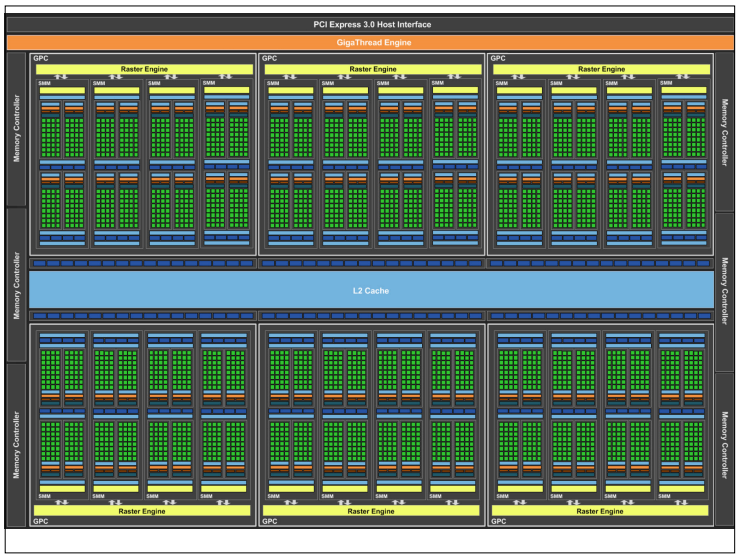
CPU



GPU

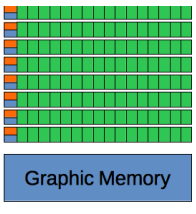
rip out parts that make CPU fast

128 ALUs in same space



Memory Contention/Latency

- shared memory slow
- shared memory contention
- ALUs very fast
- not enough cache
- memory starved



GPU

Multi-threading hides latency

- work on many groups of vertices/fragments at same time
- when one group stalls, work on other group
- modern GPUs can have 128 threads!

Three key ideas

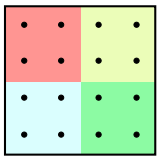
- use many “slimmed down cores” in parallel
- pack cores full of ALUs and share instruction streams
- avoid latency stalls by interleaving many groups of threads

What if you’re resource constrained?

- e.g. mobile phones
- can’t pack powerful GPU with lots of memory

Tile-based Rendering

- simpler primitives: points, lines, triangles
- frame buffer partitioned into tiles (eg 64x64) with just enough GPU rasterization hardware for 1 tile
- transform primitives and store them, noting which tiles they overlap (retained mode)
- work on one tile at a time



Tile-Based Rendering