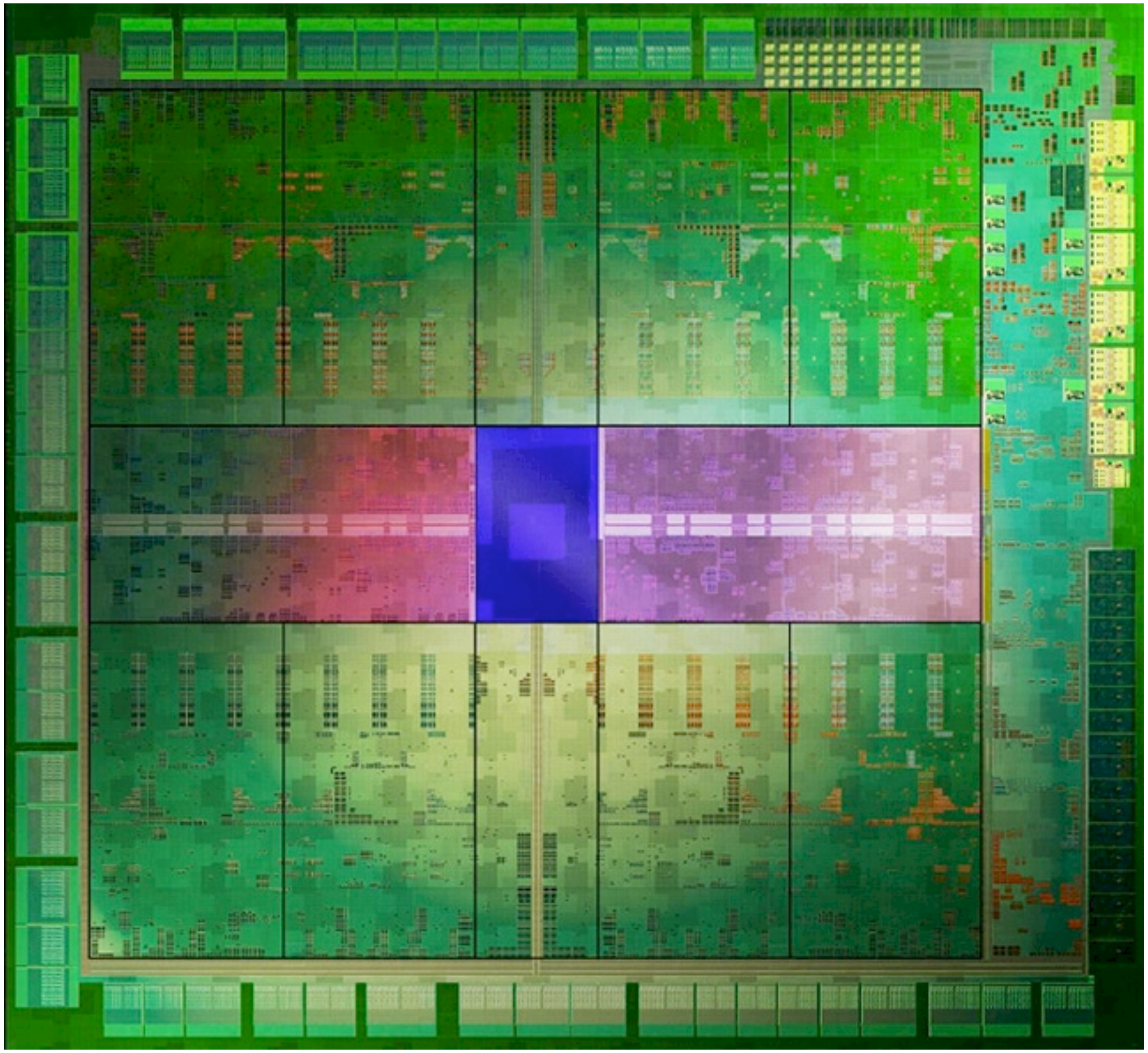
NVIDIA GeForce GTX 680 “Kepler” GPU Die



GPU Architecture

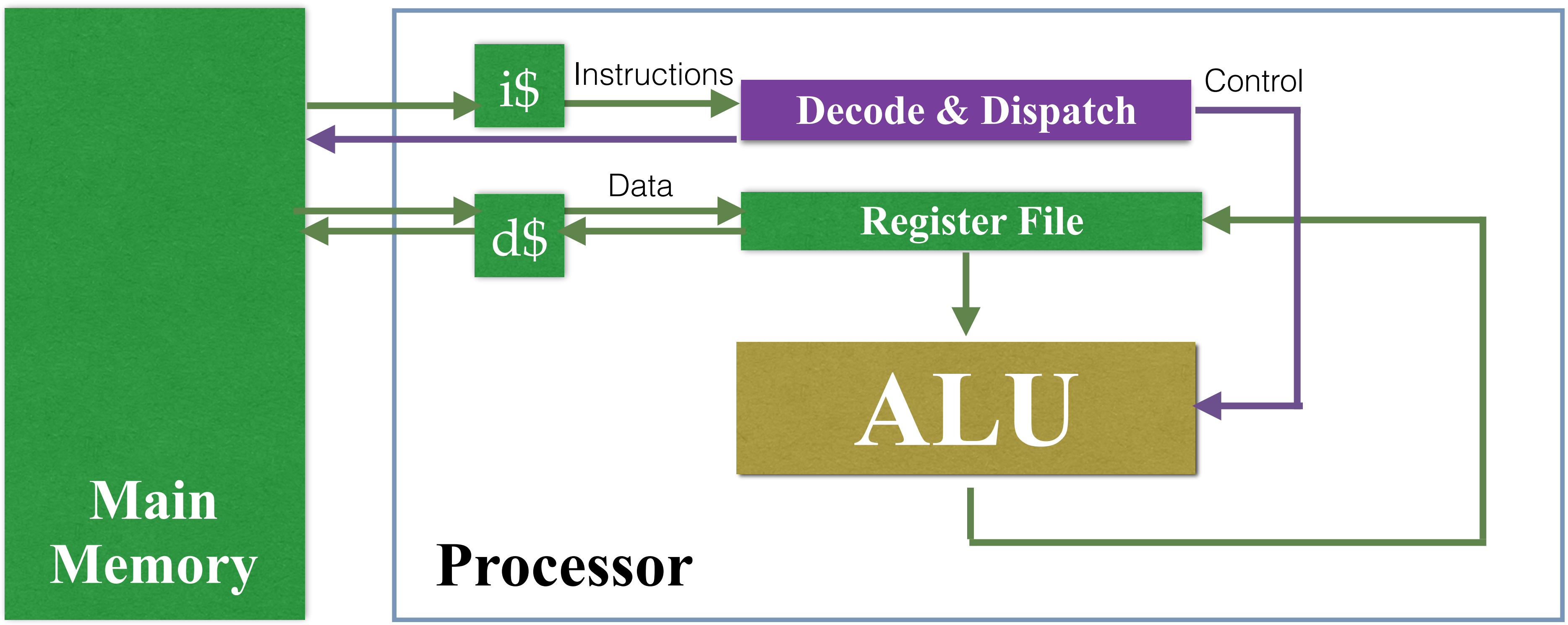
CS371: Computational Graphics - Prof. McGuire - 2014

Architecture Review

State

- **Program Counter** a.k.a. Instruction Pointer ("PC" or "IP")
- [Stack pointer ("SP"), Base pointer ("BP"), Condition codes]
- General-purpose **registers** ("reg" or "GPR")
- Fast, small **memory** (today: on-chip caches and local/shared memory)
- Slow, large **memory** (today: off-chip DRAM…and network and disk)

Generic Processor



$ = “cache”
ALU = Arithmetic Logic Unit

Goal: High Performance

(what is performance?)

Measuring Performance

Bandwidth: throughput

- 10 Mb/s network connection
- 1M pix/s rendered
- 400 Mrays/s cast

Latency: delay

Efficiency: energy consumption

Measuring Performance

Bandwidth: throughput

Latency: delay

- 30 ms delay between scan out and TV pixel changing
- 200 ms round-trip network ping (“lag”)
- 6 ms to render a shadow map

Efficiency: energy consumption

Measuring Performance

Bandwidth: throughput

Latency: delay

Efficiency: energy consumption

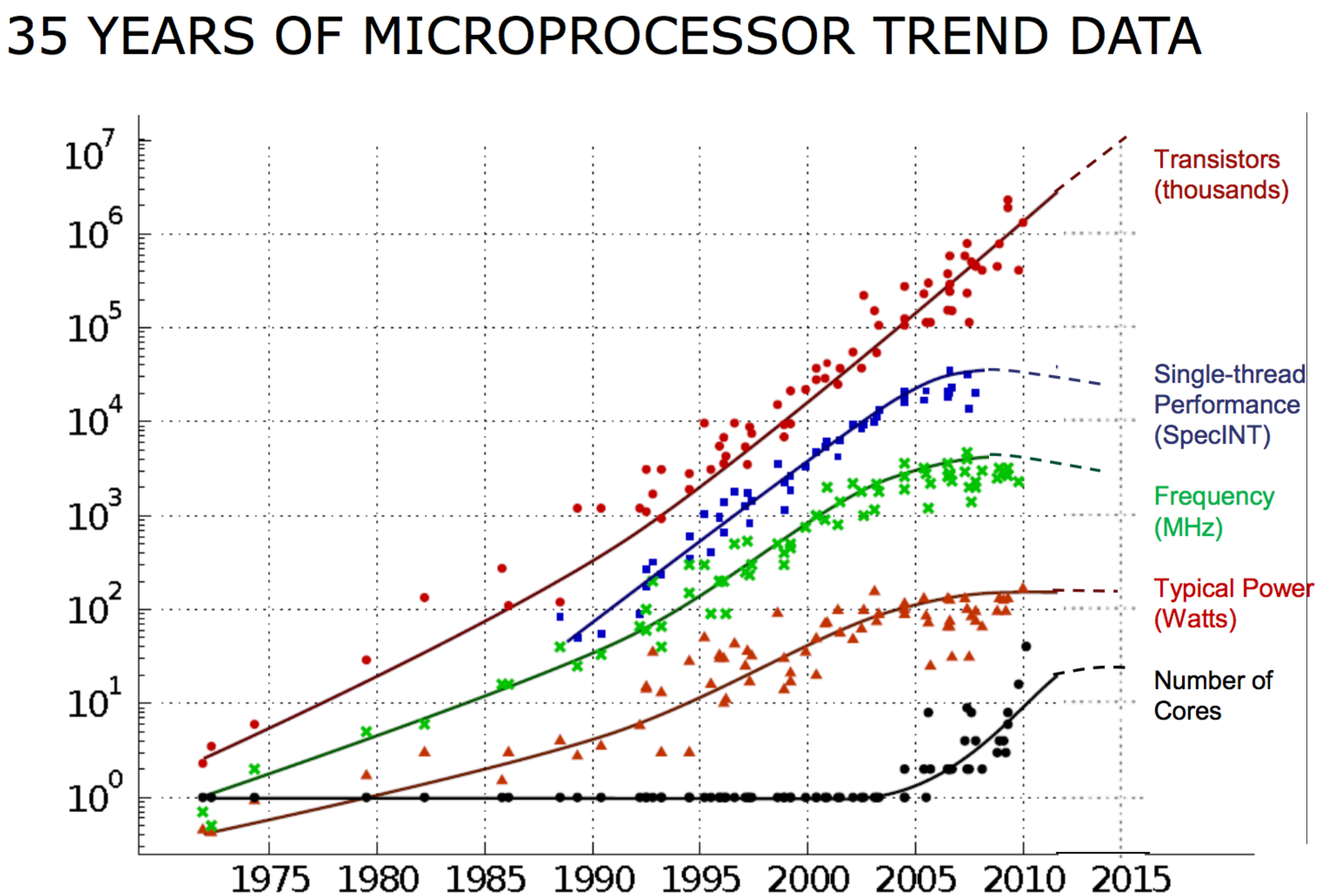
- 2 nJ to transmit a bit over WiFi
- 10^8 flops/W
- 140 nJ/pixel

See: Akenine-Möller and Johnsson, Performance per What?, Journal of Computer Graphics Techniques (JCGT), vol. 1, no. 1, 37-41, 18 Oct. 2012. Available online http://jcgt.org/published/0001/01/03/

Plan #1:

High Clock Speed

Clock Scaling Ended in 2008



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

http://www.lanl.gov/orgs/hpc/salishan/salishan2011/3moore.pdf

Plan #2: Concurrency

Vocabulary

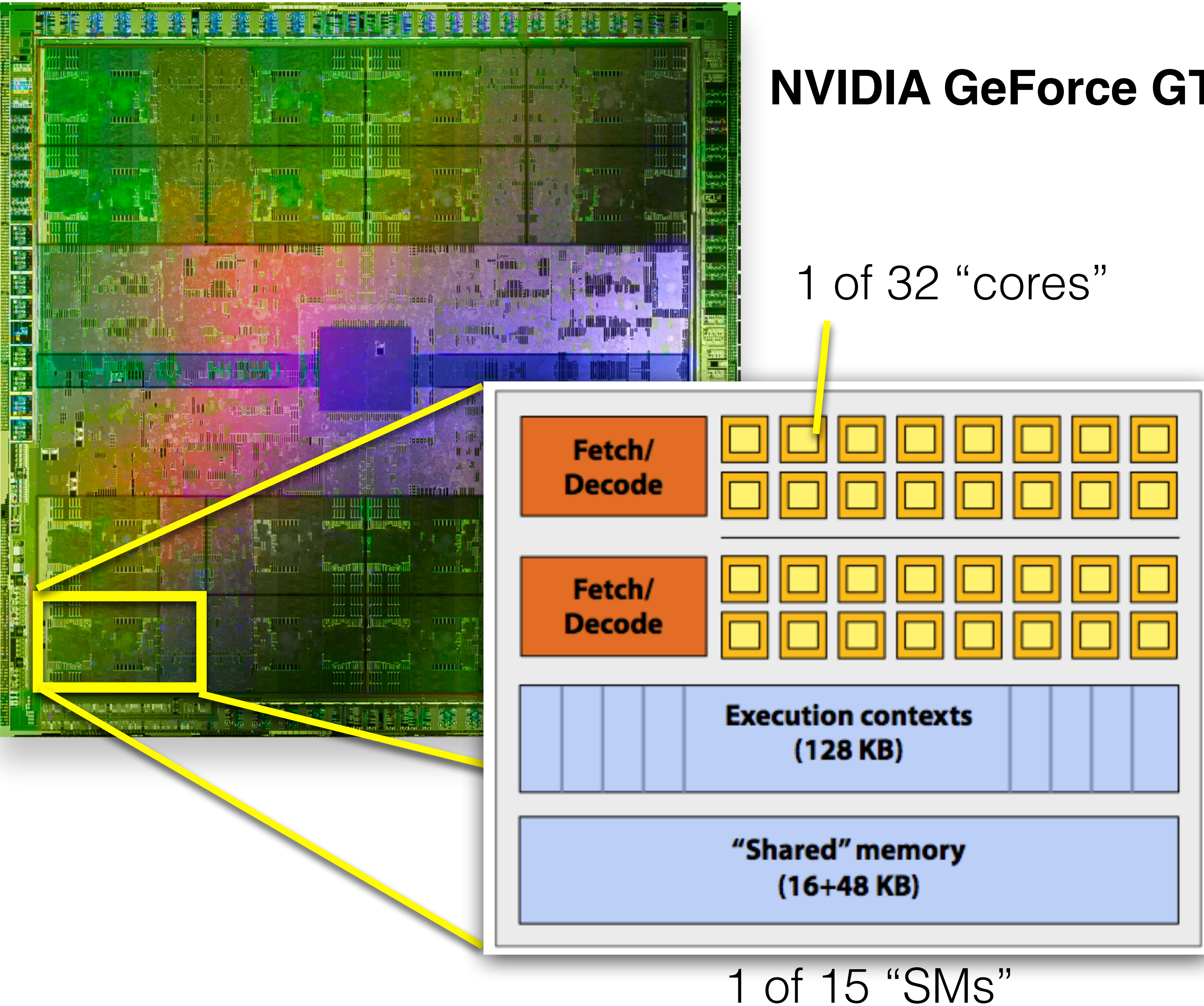
- **Concurrent** = overlapping in time
- **Parallel** = in lockstep
- **Vector**/SIMD = same instruction applied to a lot of data

…but, in practice, these words are often used interchangeably

A Carpool is Vector Parallel

a.k.a. “SIMD” or “Superscalar”

NVIDIA GeForce GTX 480



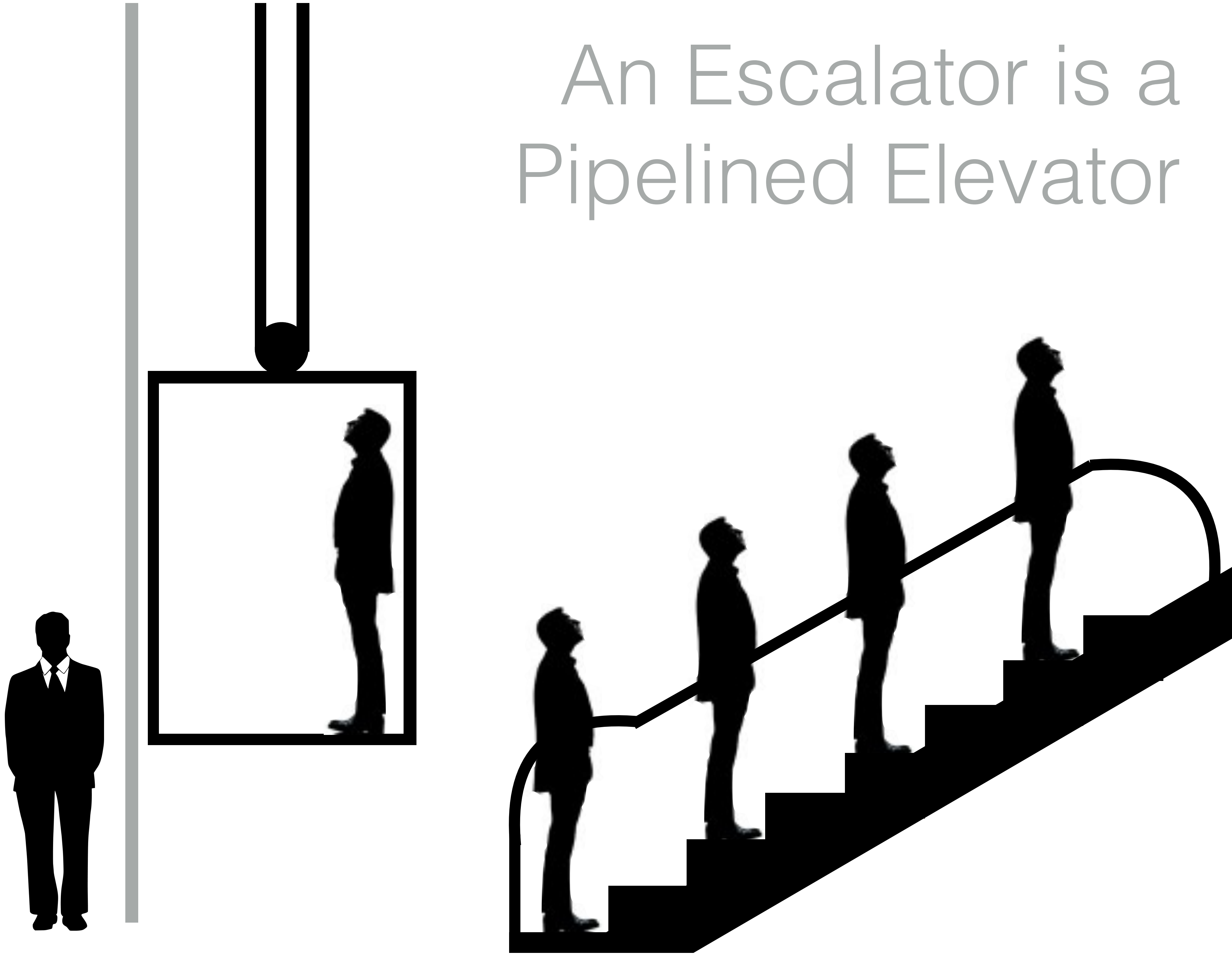
GPU Vectorization

- **Groups (“warps”) of ≈32 “threads” share a single instruction pointer**
  - Amortizes instruction fetch and decode
  - Amortizes adjacent memory fetch instructions
- **Sets of ≈32 thread groups also round-robin on a single processor**
  - Hides memory fetch and some ALU latency
  - Context swap is free

Vectorization Challenges

- If threads branch different ways, must execute both sides
- Must align memory access into simple patterns
- No speedup if there aren’t enough threads to fill a group/set
- Total threads limited by register count

An Escalator is a Pipelined Elevator



GPU Pipelining

- New data from the CPU are being transferred to the GPU while previous commands are queued for launch and others are executing.

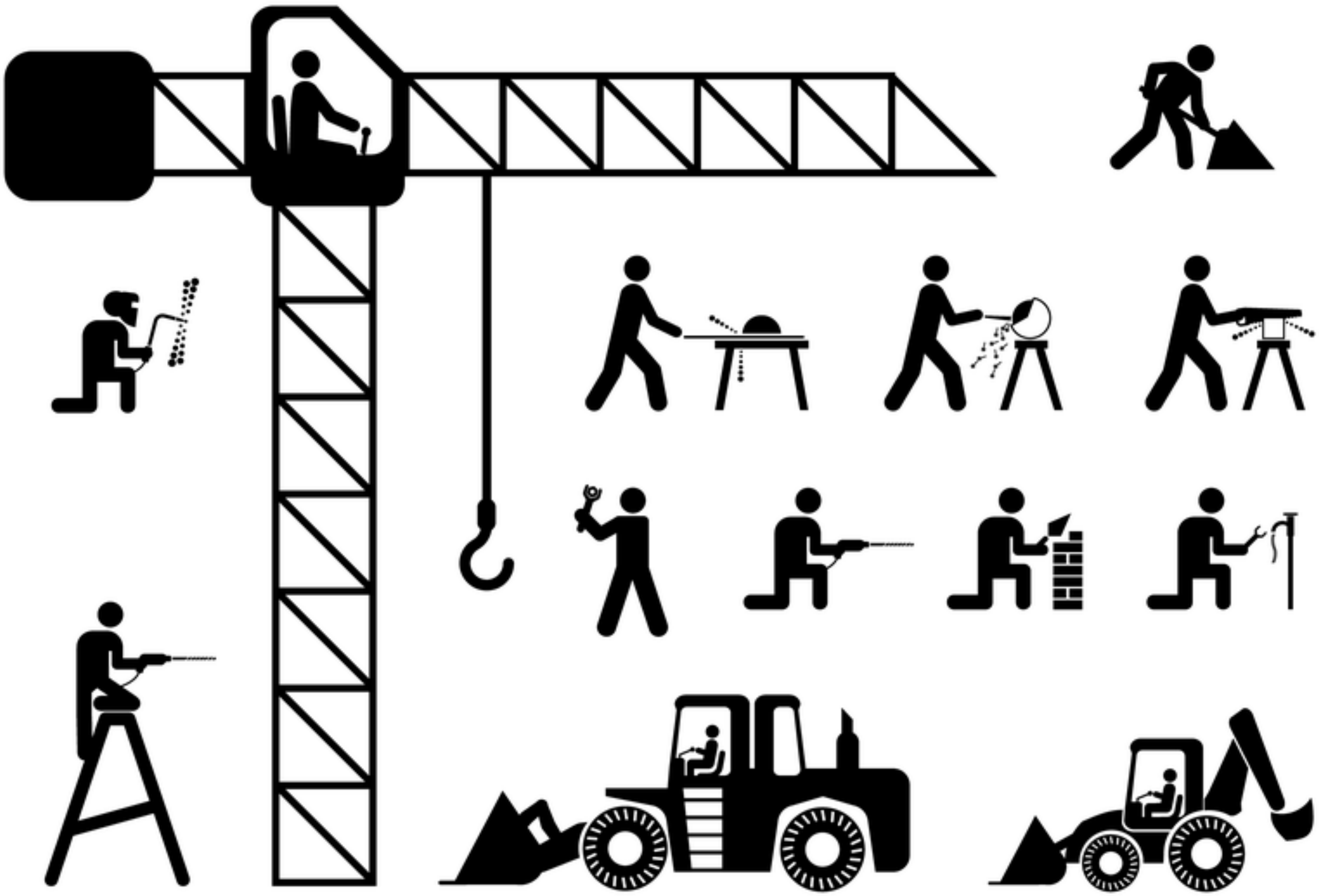
- One processor is transforming vertices of triangle 4 while another rasterizing triangle 3, another is shading pixels of triangle 2, and another is combining pixels from triangle 1 with the framebuffer

- One circuit is fetching instruction 7 while another is decoding instruction 6 and another is executing instruction 5

Pipeline Challenges

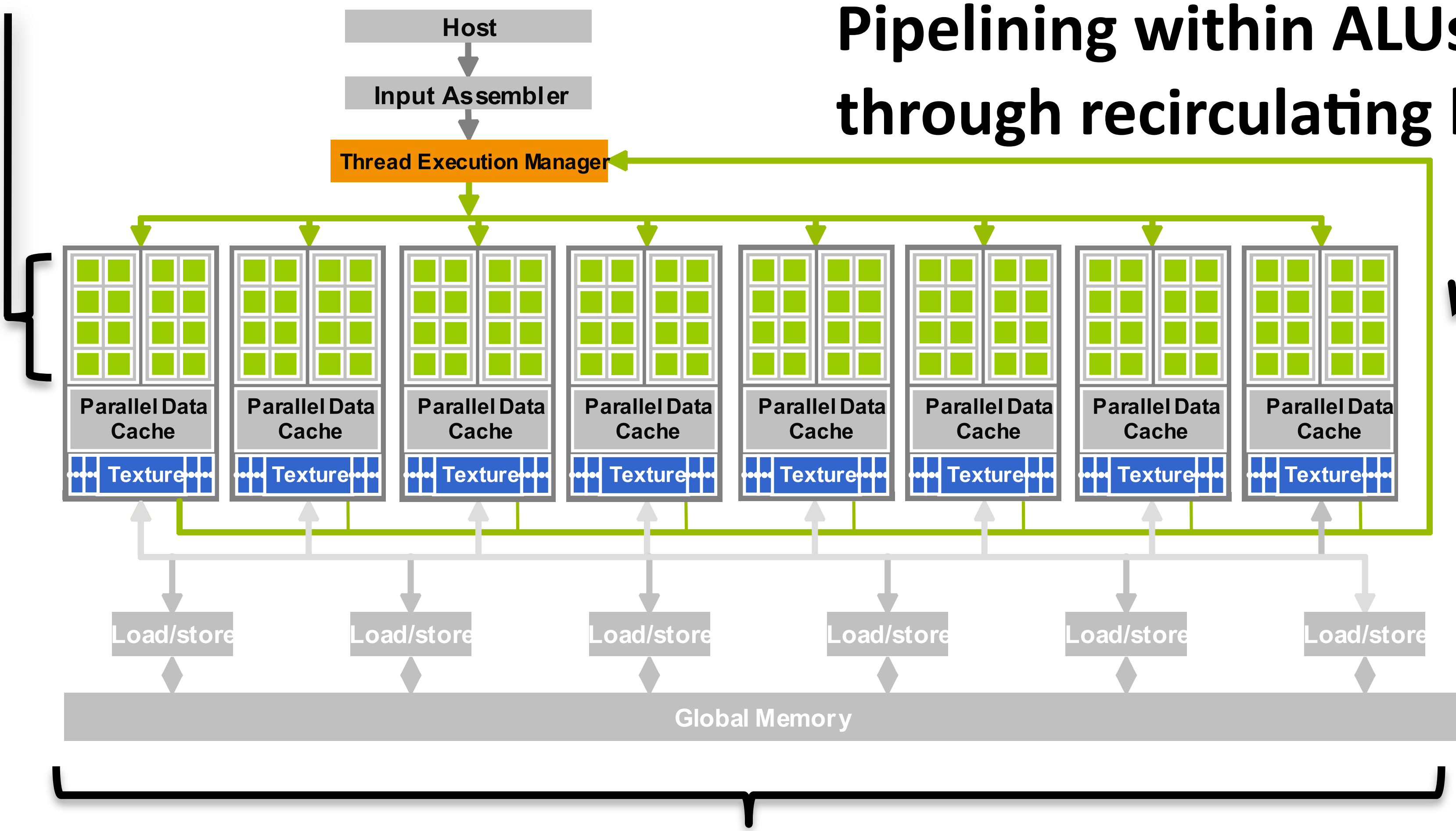
- Bubbles
- Stalls
- Reordering

A Construction Crew is Task Parallel



Data-parallel vector lanes inside each unit

Pipelining within ALUs and through recirculating loop

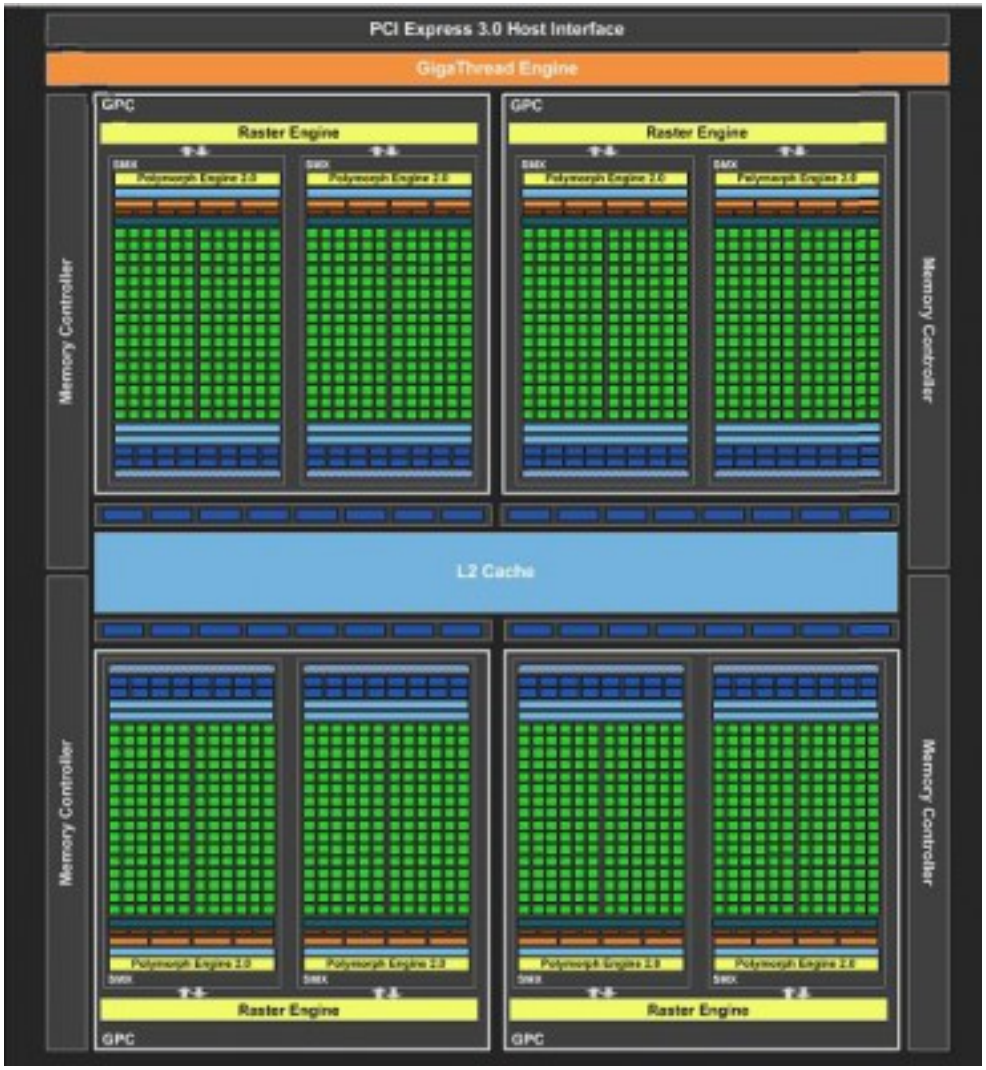


Task-parallel processing units

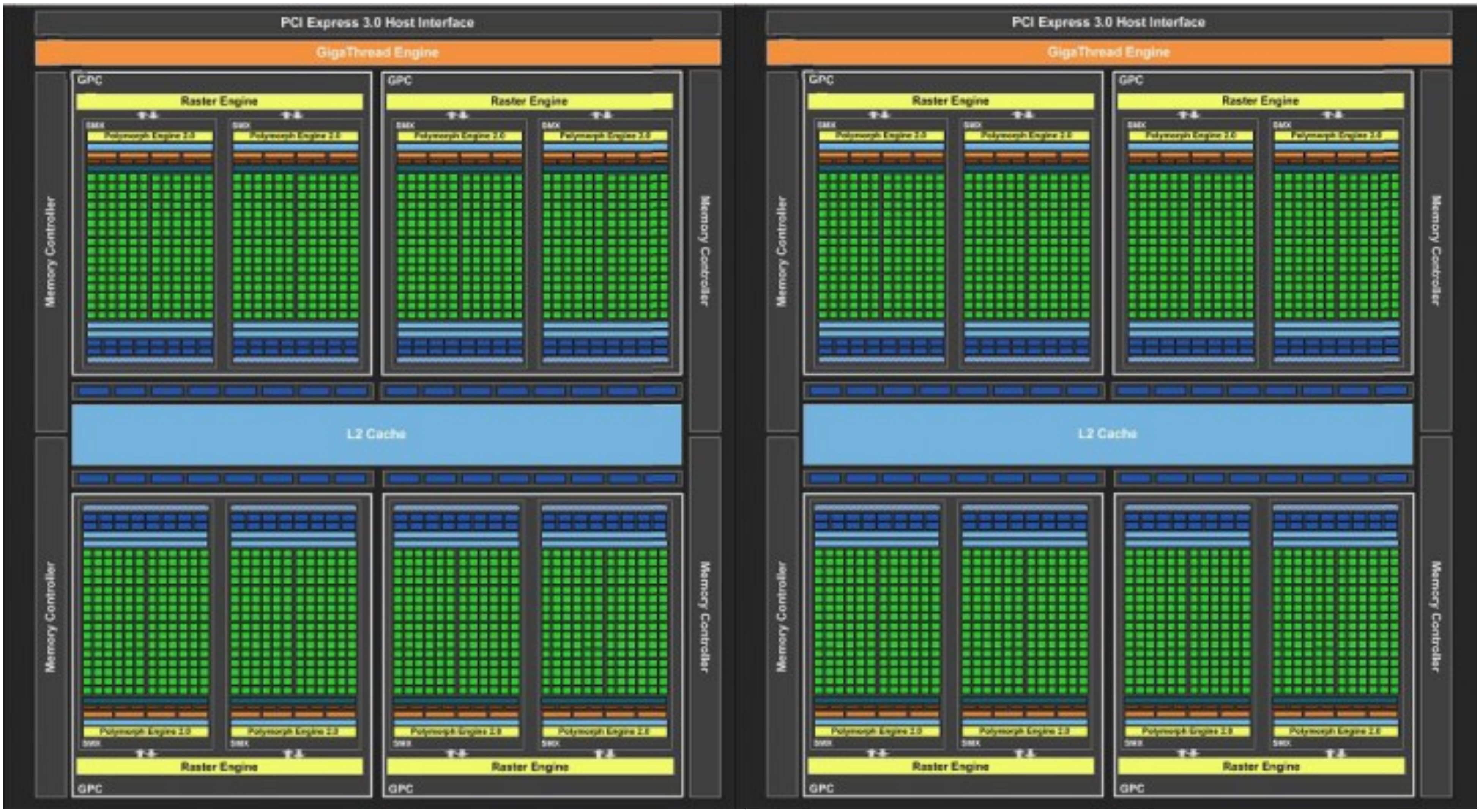
GPU Task Parallelism

- Multiple processing units (SMs/cores), each with its own kernel and local memory
- Multiple chips on a single GPU
- Multiple GPUs (SLI/Crossfire) in a machine
- (CPU + GPU)
- Multiple machines on a network cluster

NVIDIA GeForce GTX 680



NVIDIA GeForce GTX 690



Multiprocessor Challenges

- Mutex synchronization is difficult (deadlocks, race conditions)
- Memory transfers and cache management are expensive
- Less amortization than vectorization (but better divergence management)

GPU Efficiency Sources

- Amortized of instruction fetch & decode
- Large floating-point ALU
- Barrier synchronization & atomics
- Large register banks
- Fixed-function rasterization
- Relatively high memory bandwidth

Current GPU Quirks

- FMUL: floating point multiply + add is a single instruction
- Conditional operations are “free”
- Float32 is faster than integer and boolean
- Float32 is *much* faster than float64
- Caches are smaller and relatively slower than CPUs (in register is great, local memory falls off a lot, and global memory is slow)

How to write fast GPU code

Minimize CPU Sync

- Every state change or draw call forces CPU-GPU synchronization, stalling one or both
- Drivers max out around 1000 calls/frame
- This is a little better on console and will be much better in a year on PC & mobile

Minimize GPU Sync

- Avoid data hazards (adjacent passes that depend on each other)
- Avoid explicit synchronization
- Lock contexts to GPUs for peak multi-GPU performance

Maximize Occupancy

- Small iteration bounds (triangles) leave warps partially filled
- Threads that discard leave warps partially filled
- High peak register counts leave SMs partially filled

Minimize Divergence

(Execution)

Slow

```
if (x > 3.0) {
    y = 6.0 + x;
} else {
    y = 10.0;
}
```

Fast

```
y = float(x > 3.0) * (x - 4.0) + 10.0
```

Minimize Divergence

(Memory)

Slow

```
c = texelFetch(T, texelFetch(L, ivec2(gl_FragCoord.xy)).xy, 0);
```

Fast

```
c = texelFetch(T, ivec2(gl_FragCoord.xy), 0);
```

Use Fast Operations

Slow

- if, while, switch
- sqrt, division, log, exp
- int, bool, double

Fast

- min, max, clamp
- x * a + b
- b = t ? c : a
- float

Minimize Bandwidth

throughput = min(memory / bandwidth,

compute / aluRate)

GLSL & HLSL Gotchas

- GLSL & HLSL are substitution interpreters…they inline *everything*
  - Small, fixed-length loops unroll completely
  - Branches on compile-time constants are free
  - Dead code is free
  - No recursion allowed (but you can build your own stack)
  - No function pointers (and no classes or methods)
- Computed array indices are relatively slow (can’t relative index registers)
- Memory allocation is extremely slow
- Can’t store textures in arrays or structures
- These are language quirks, not GPU architecture limitations