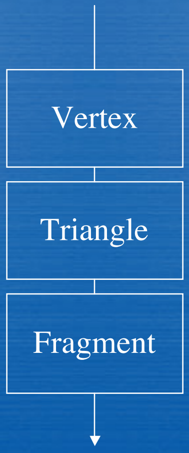
CMSC 635

Graphics Hardware

A Graphics Pipeline



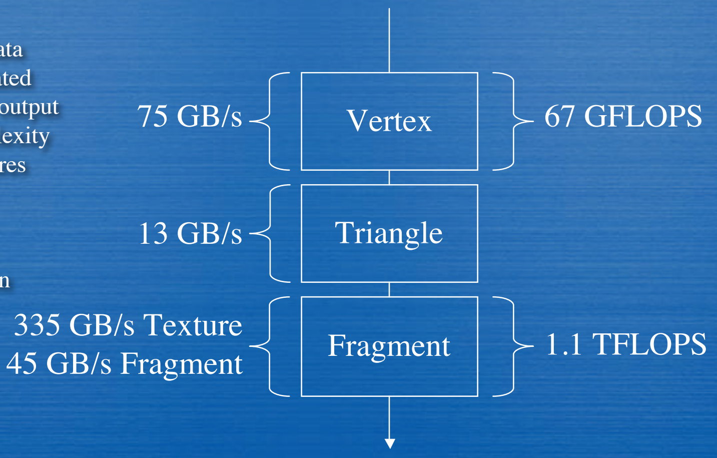
Fragment vs. Pixel

- OpenGL terminology
- Pixel = on-screen RGBA+Z
- Fragment = proto-pixel
  - RGBA + Z + Texture Coordinates + …
  - Multiple Fragments per Pixel
    - *Depth Complexity*
    - *Supersamples*

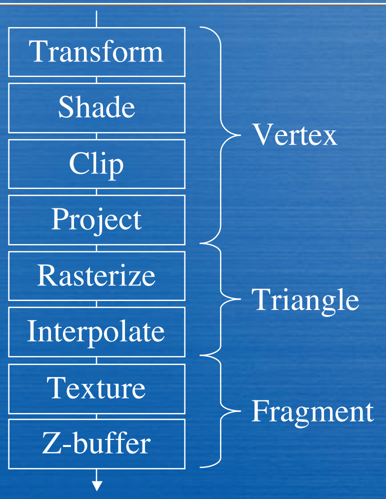
Computation & Bandwidth

Based on:

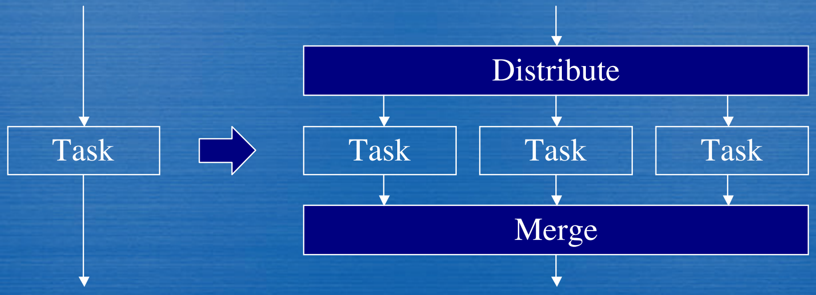
- 100 Mtri/sec
- 256 B vertex data
- 128 B interpolated
- 68 B fragment output
- 5x depth complexity
- 16 4-byte textures
- 223 ops/vert
- 1664 ops/frag
- No caching
- No compression



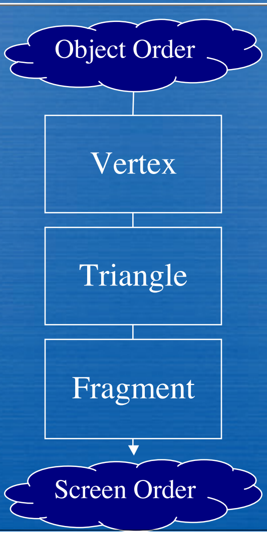
Pipeline



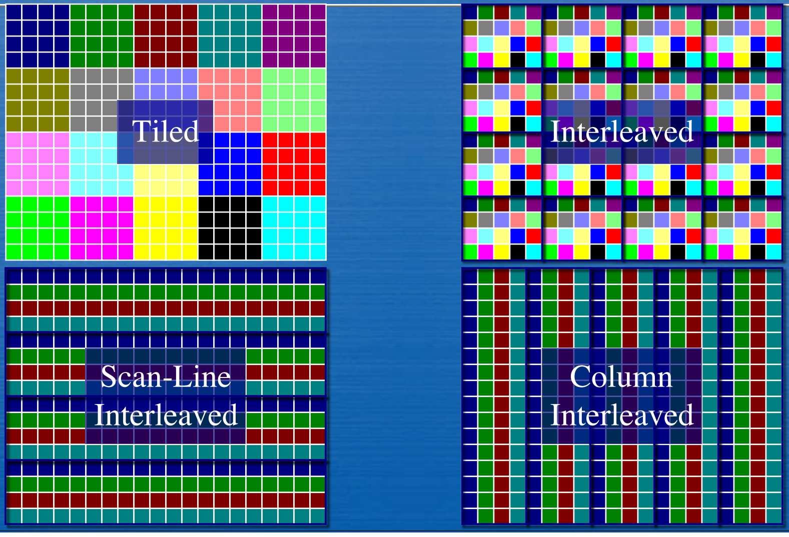
Data Parallel



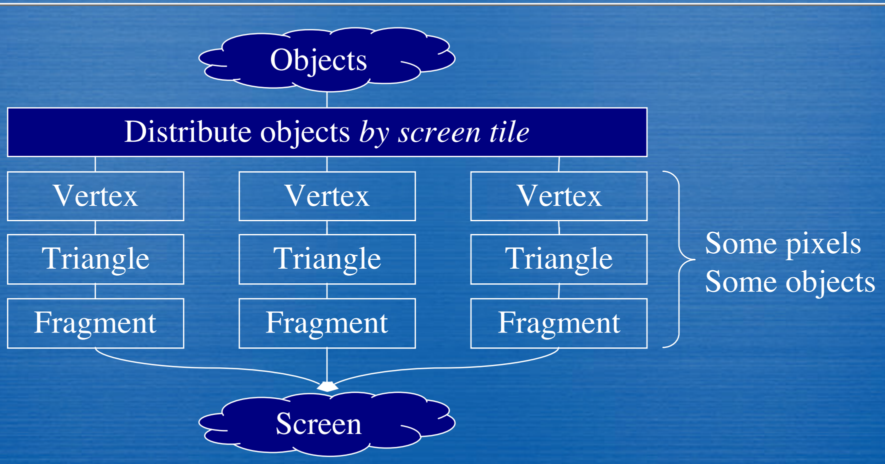
Graphics Data Organization



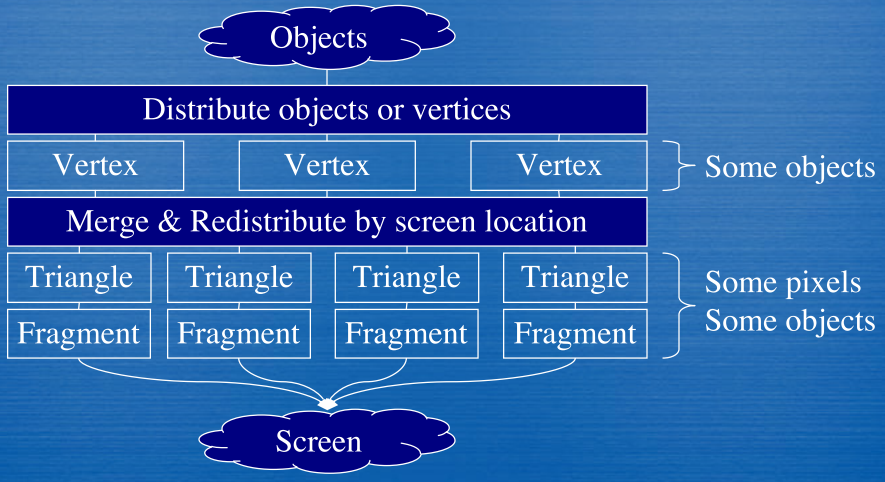
Screen Subdivision



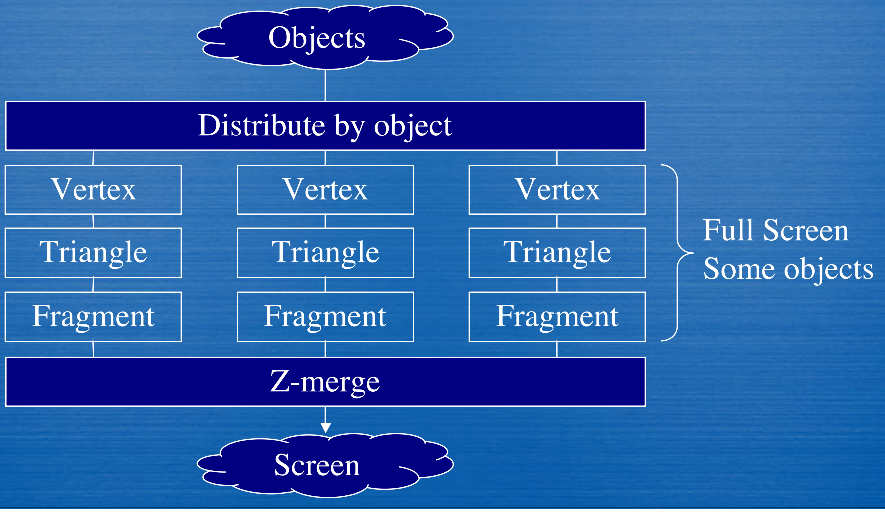
Sort First



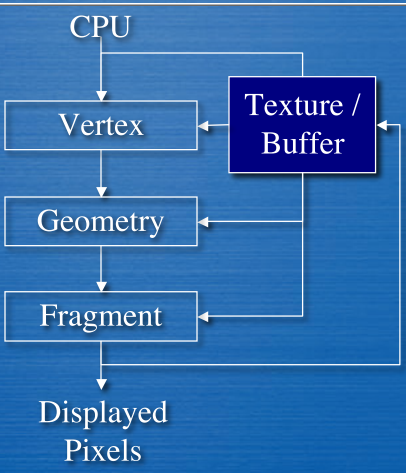
Sort Middle



Sort Last



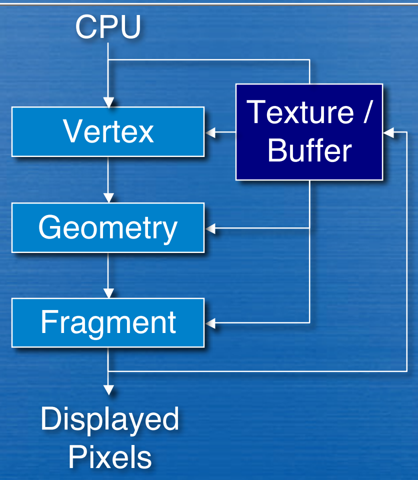
GPU computation



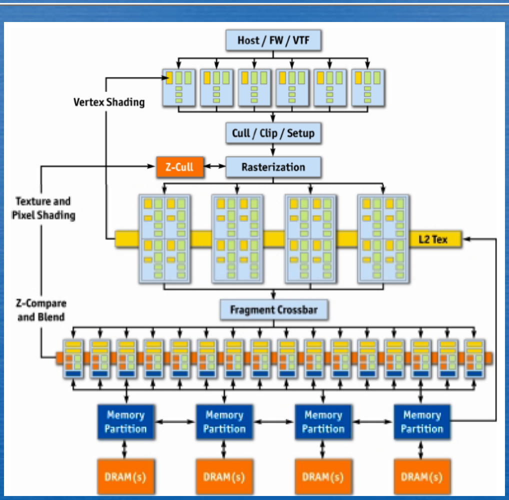
- NVIDIA GeForce 7800
  - ~860M vertices/sec
  - ~172M triangles/sec
  - ~6.9G fragments/sec
  - ~10.3G texels/sec
  - ~165 GFLOPS
  - ~2.4x increase/year
- 3GHz Dual-core Pentium 4
  - ~24.6 GFLOPS
  - ~1.5x increase/year

[Luebke, GPGPU SIGGRAPH Course, 2005; Kilgard, Real-Time Shading SIGGRAPH course, 2006]

GPU graphics processing model

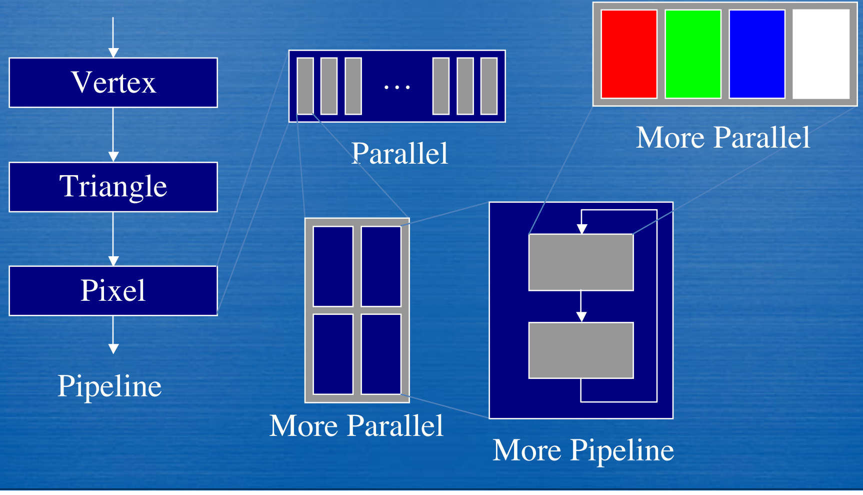


NVIDIA GeForce 6

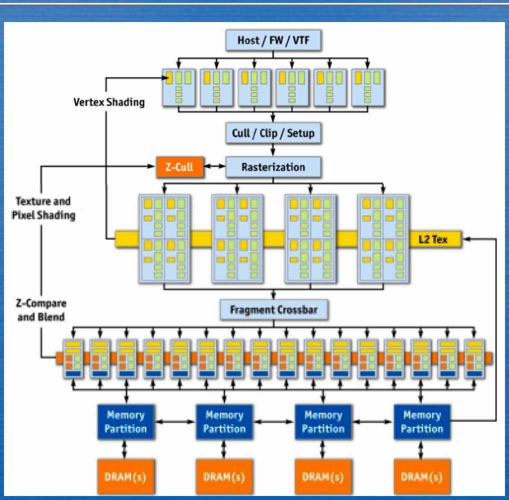


[Kilgaraff and Fernando, GPU Gems 2]

Graphics Processing Unit

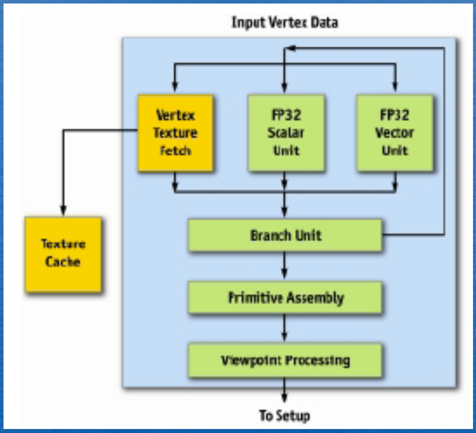


NVIDIA GeForce 6

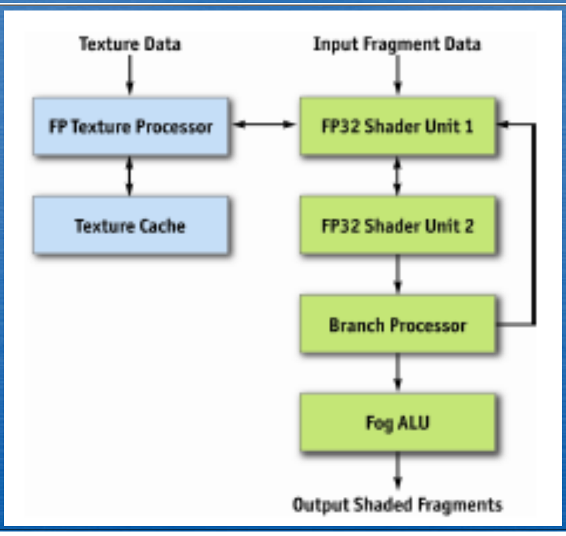


[Kilgaraff and Fernando, GPU Gems 2]

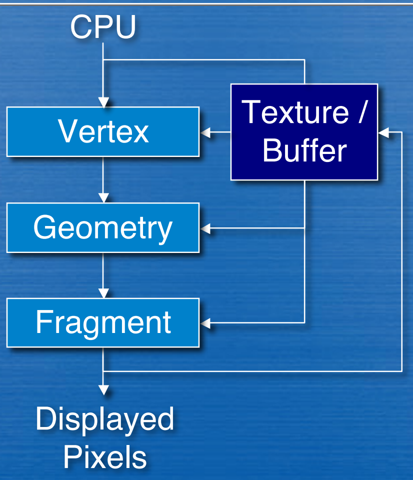
Vertex Processing



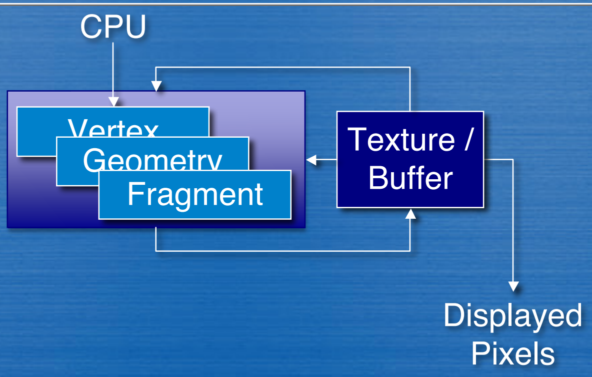
Fragment Processing



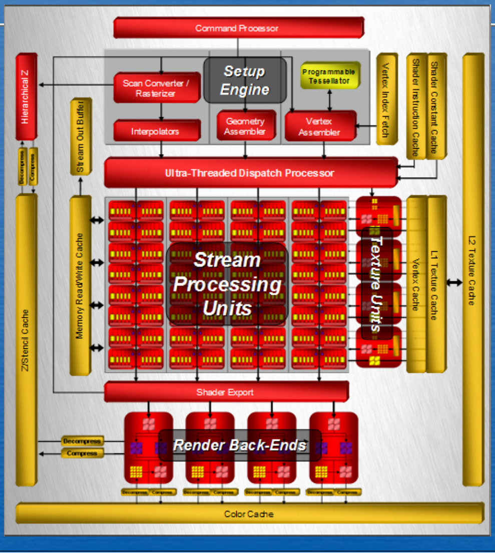
GPU graphics processing model



GPU graphics processing model

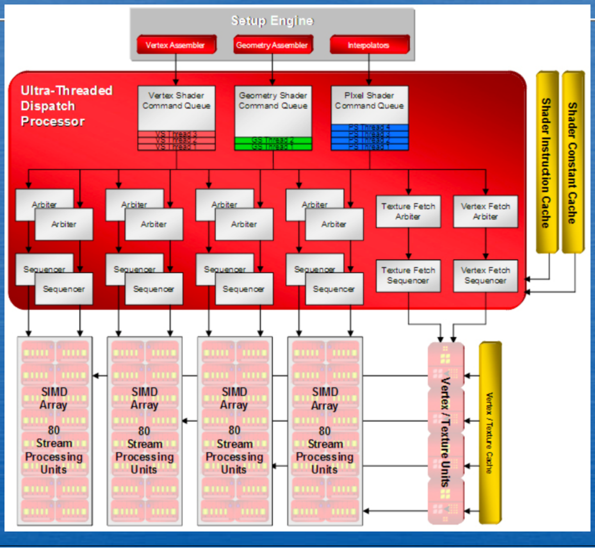


AMD/ATI R600

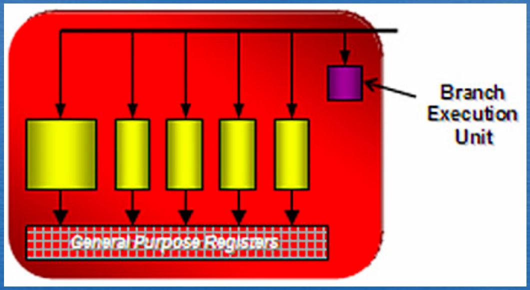


[Tom's Hardware]

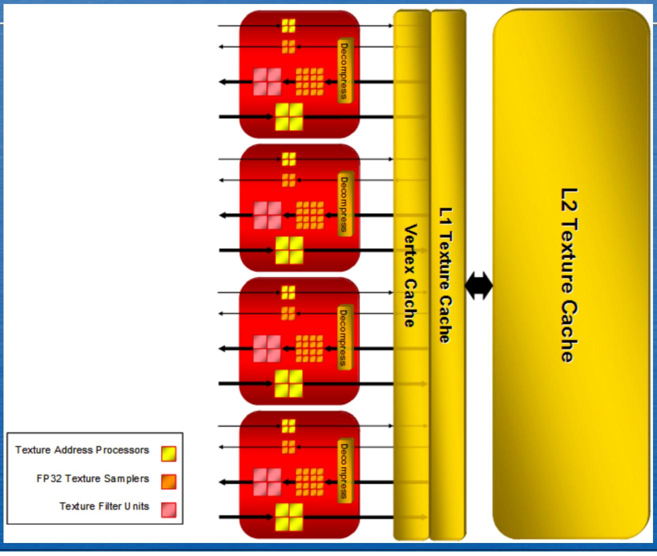
Dispatch



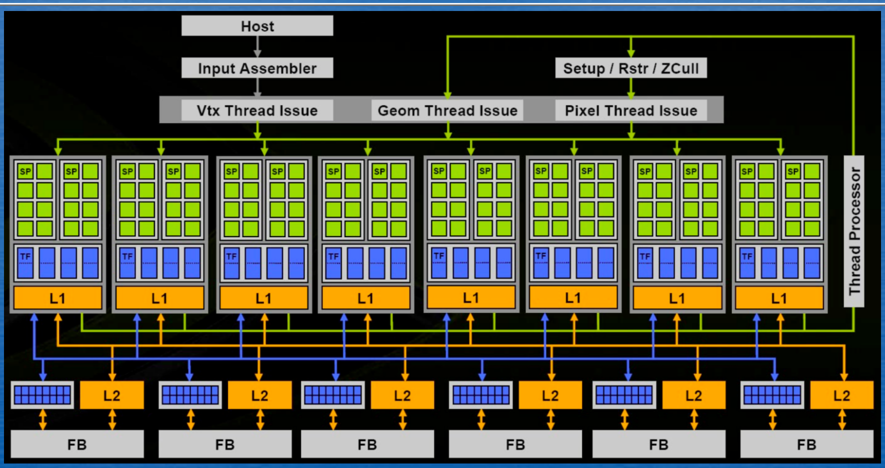
SIMD Units



Texture

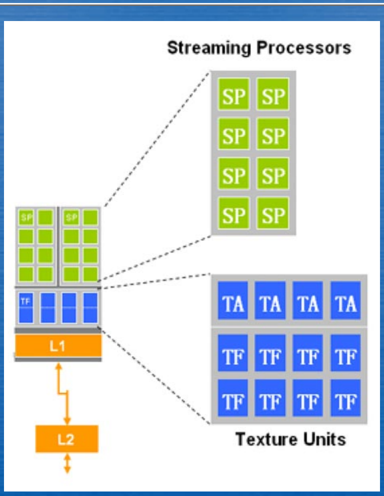


NVIDIA G80

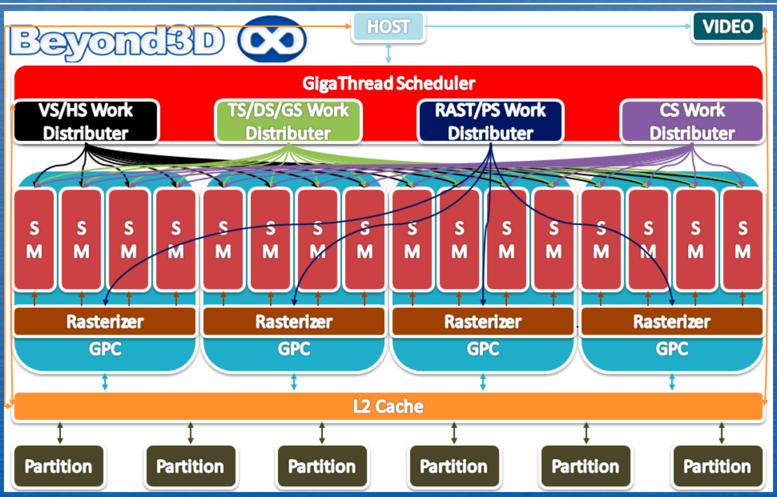


[NVIDIA 8800 Architectural Overview, NVIDIA TB-02787-001_v01, November 2006]

Streaming Processors

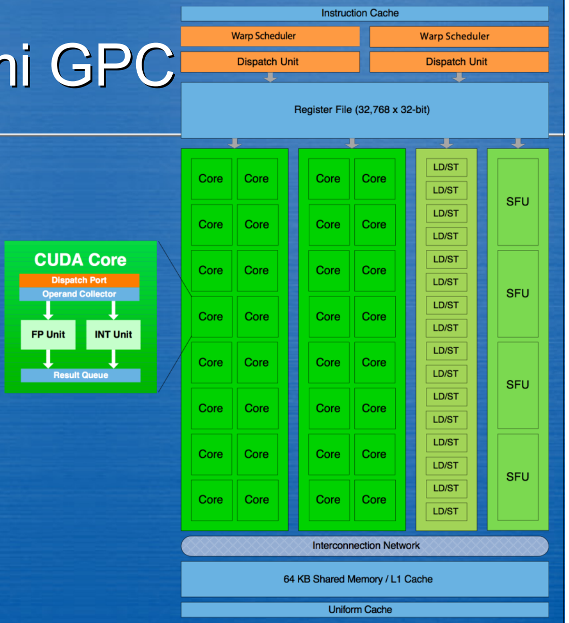


NVIDIA Fermi

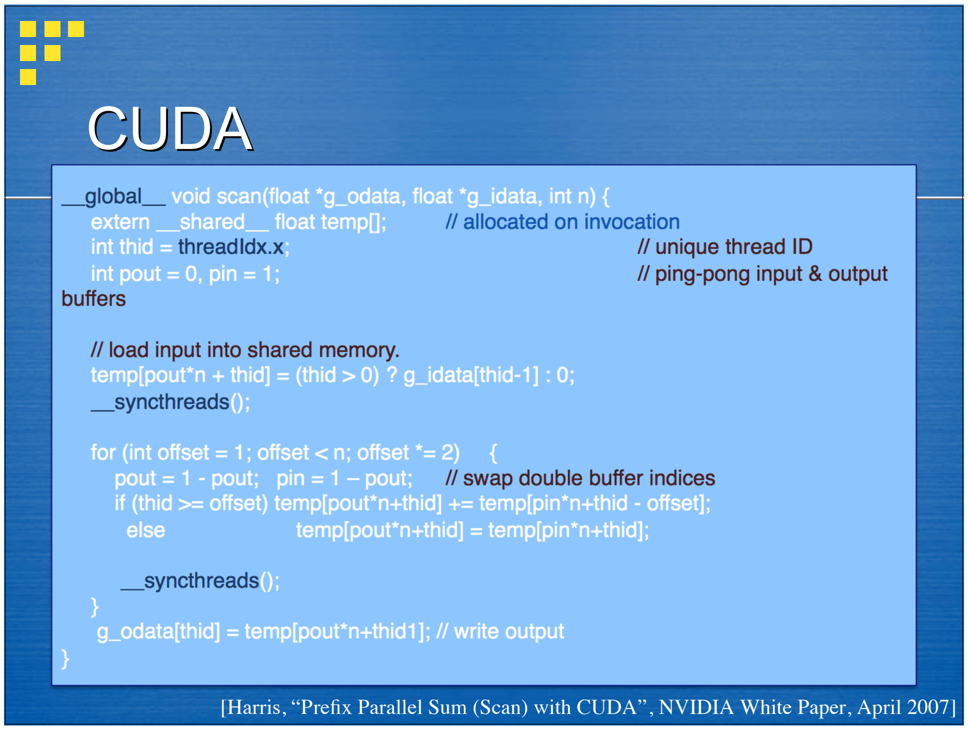


[Beyond3D NVIDIA Fermi GPU and Architecture Analysis, 2010]

NVIDA Fermi GPC



[NVIDIA, NVIDIA’s Next Generation CUDA Compute Architecture: Fermi, 2009]



CUDA

```
__global__ void scan(float *g_odata, float *g_idata, int n) {
     extern __shared__ float temp[];          // allocated on invocation
     int thid = threadIdx.x;                                       // unique thread ID
     int pout = 0, pin = 1;                                        // ping-pong input & output
buffers

     // load input into shared memory.
     temp[pout*n + thid] = (thid > 0) ? g_idata[thid-1] : 0;
     __syncthreads();

     for (int offset = 1; offset < n; offset *= 2)     {
         pout = 1 - pout;   pin = 1 – pout;      // swap double buffer indices
         if (thid >= offset) temp[pout*n+thid] += temp[pin*n+thid - offset];
           else                     temp[pout*n+thid] = temp[pin*n+thid];

          __syncthreads();
     }
      g_odata[thid] = temp[pout*n+thid1]; // write output
}
```

[Harris, "Prefix Parallel Sum (Scan) with CUDA", NVIDIA White Paper, April 2007]