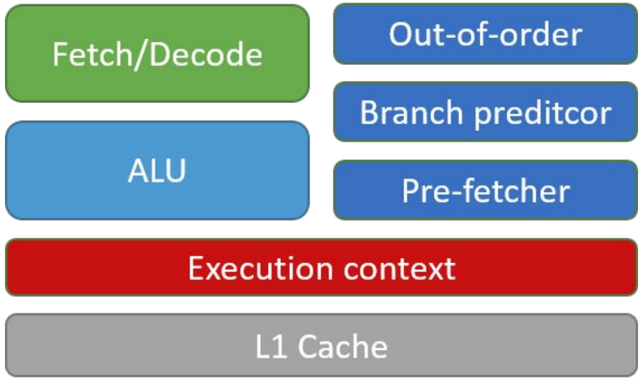
OpenCL Workshop
GPU architectures

Ljubljana, May 2019

Outline

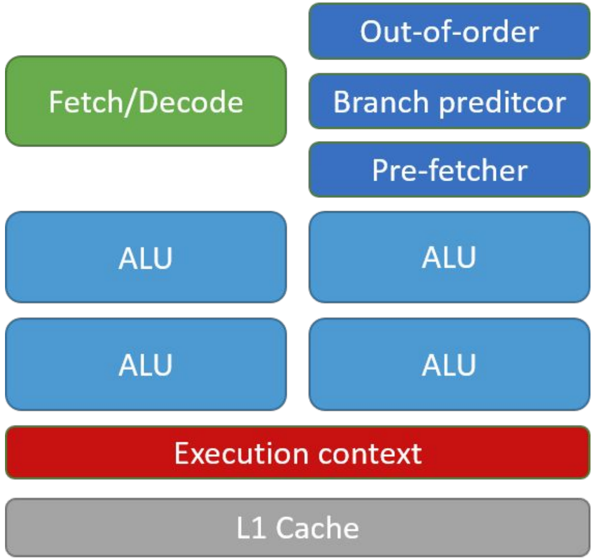
- How we got from x86 to GPUs
  - multi-core vs many-core
  - GPU programming model
- OpenCL language
  - vector addition
  - image processing

Single-core CPU



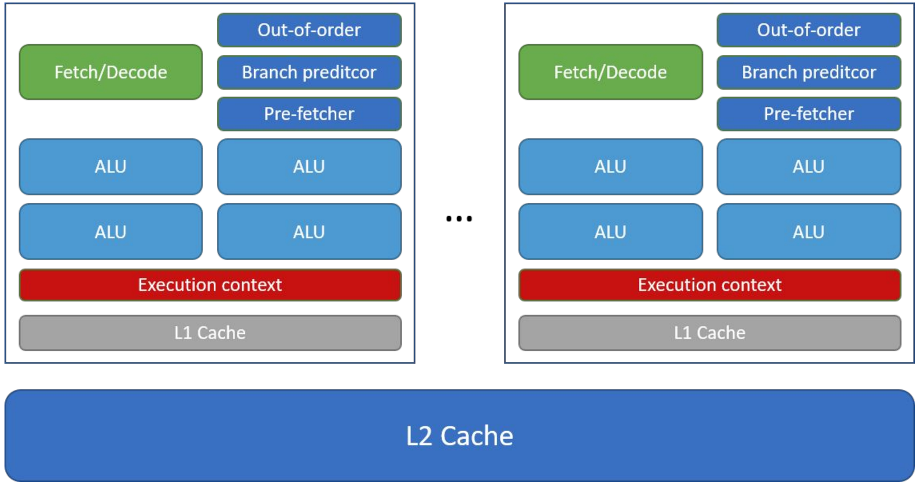
```
int i = 0;
while ( i < N ){
      C[i] = A[i] + B[i];
      i++;
}
```

SIMD Extensions (1999)



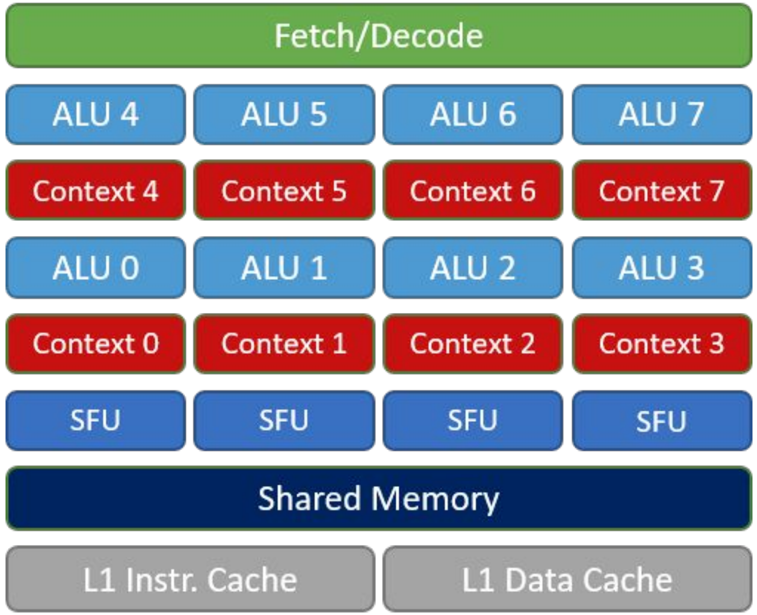
```
int i = 0;
while ( i < N ){
    C[i] = add4(A[i], B[i]);
    i = i + 4;
}
```

Multi-core (~2001)

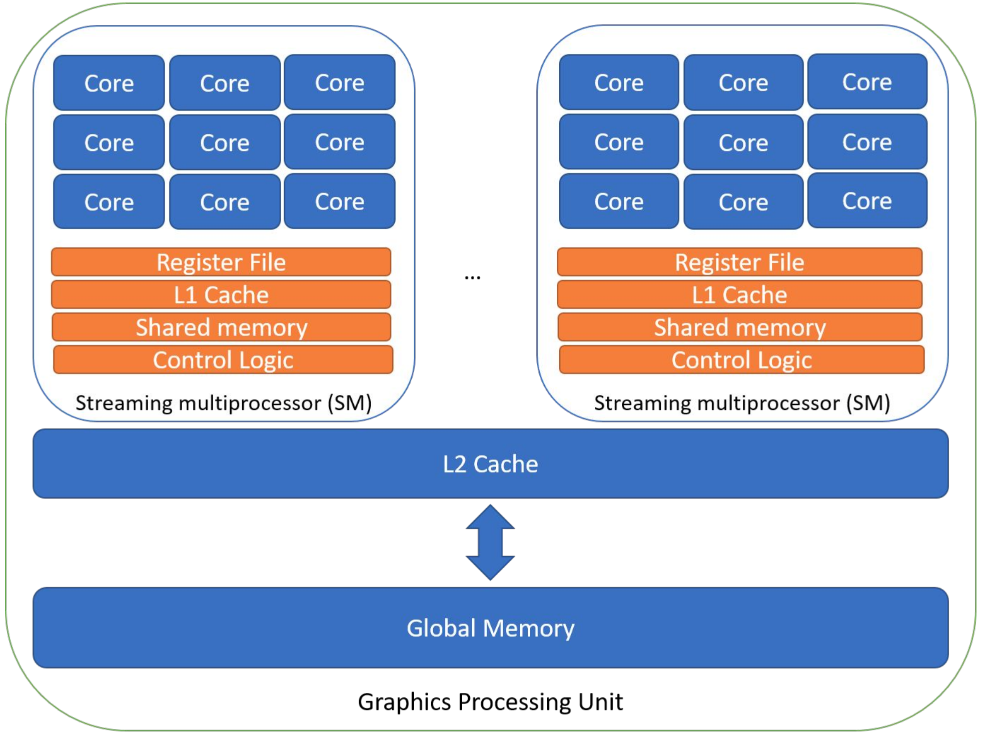


```
i = tID * (N / num_of_cores);
end = (tID +1) * (N /
num_of_cores);
while( i < end ){
       C[i] = A[i] + B[i];
       i++;
}
```

GPU Streaming multiprocessor (SM)



GPU

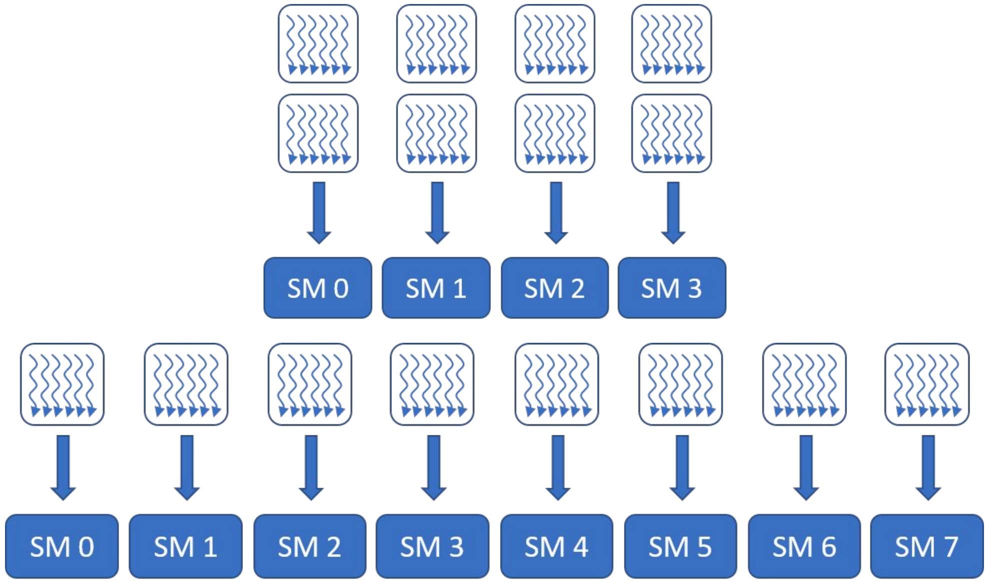


Core
Core
Core
Core
Core
Core
Core
Core
Core
Register File
L1 Cache
Shared memory
Control Logic
Streaming multiprocessor (SM)
...
Core
Core
Core
Core
Core
Core
Core
Core
Core
Register File
L1 Cache
Shared memory
Control Logic
Streaming multiprocessor (SM)
L2 Cache
Global Memory
Graphics Processing Unit

GPU

```
i = tID * (N / num_of_cores); // just a larger number of cores
end = (tID + 1) * (N / num_of_cores);
while( i < end ){
     C[i] = A[i] + B[i];
     i++;
}
```

Thread blocks & scheduling

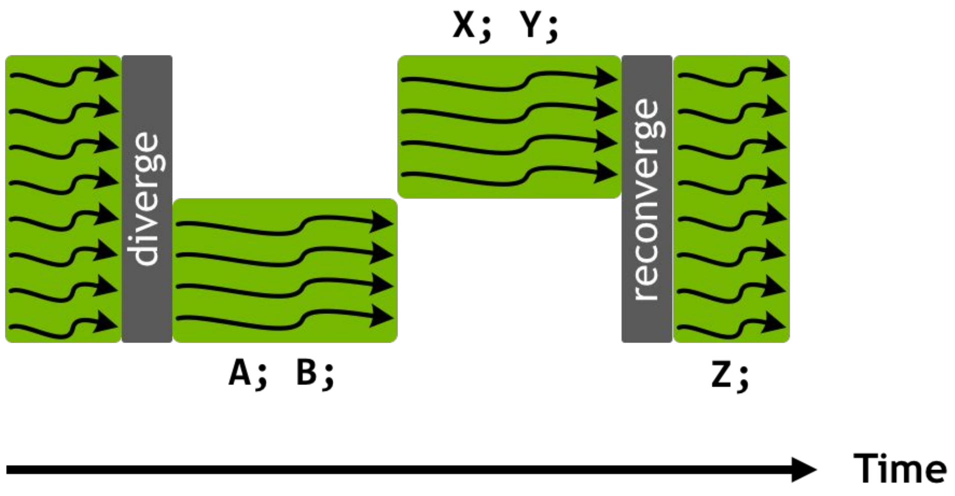


Scheduling threads - warps

- SM is issued a single thread-block
  - all threads will be executed in the assigned SM, from start to finish
- Threads are issued to the execution cores in **warps** (typically 32 threads)
  - each instruction issue step 1 warp is selected, usually trying to avoid stalls
- Large number of thread blocks -> parallelism on the SM level
  - all SMs are occupied
- Large thread blocks -> more efficient warp scheduling
  - reduces stalls in the issue steps

Thread divergence

```
if (threadID < 4) {
     A;
     B;
} else {
     X;
     Y;
}
Z;
```

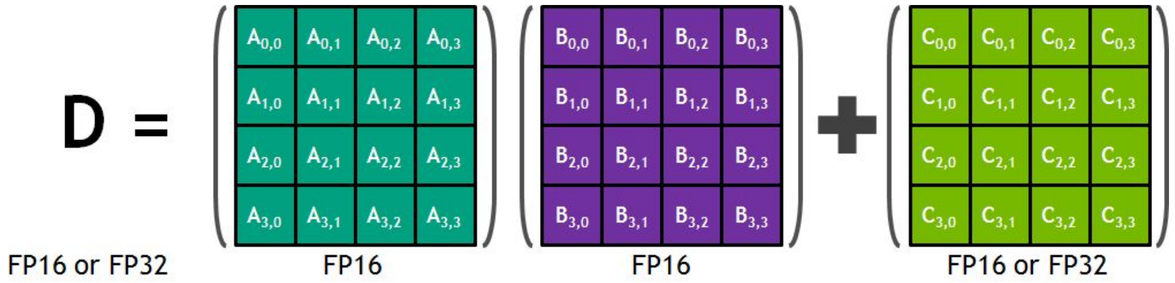


NVIDIA GPU microarchitecture evolution

Name	Year	# FP32 cores/SM	#FP64 cores/SM	#SM
Tesla	2007	8	0	up to 30
Fermi	2010	32		up to 32
Kepler	2012	192	64	up to 8
Maxwell	2014	128	4	up to 24
Pascal	2016	64 or 128	32 or 4	up to 30
Volta	2017	64	32	80
Turing	2018	64	2	72

GPU architecture trends

Tensor cores



The return of scheduling

Improved caches

Mixed precision