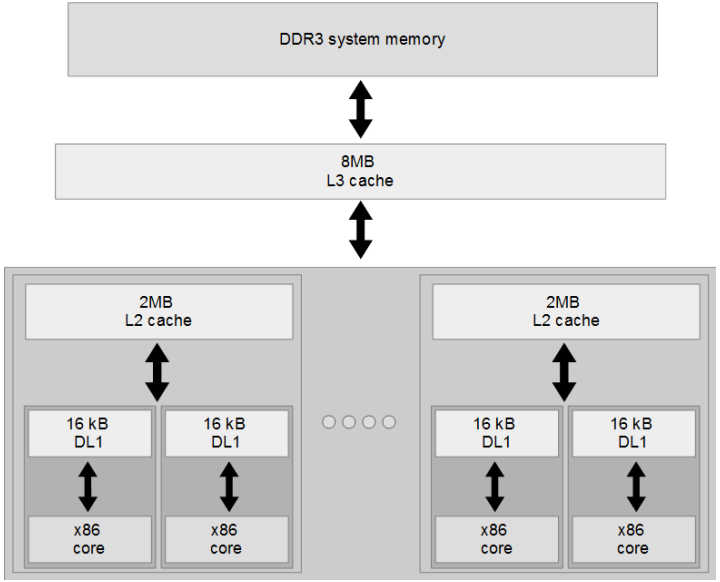
GPU ARCHITECTURE

OPENCL 2.0 UNIVERSITY TOOLKIT

Zhongliang Chen and Yash Ukidave,
Northeastern University Computer Architecture Research Lab
with
Perhaad Mistry and Dana Schaa, AMD

Conventional CPU Architecture

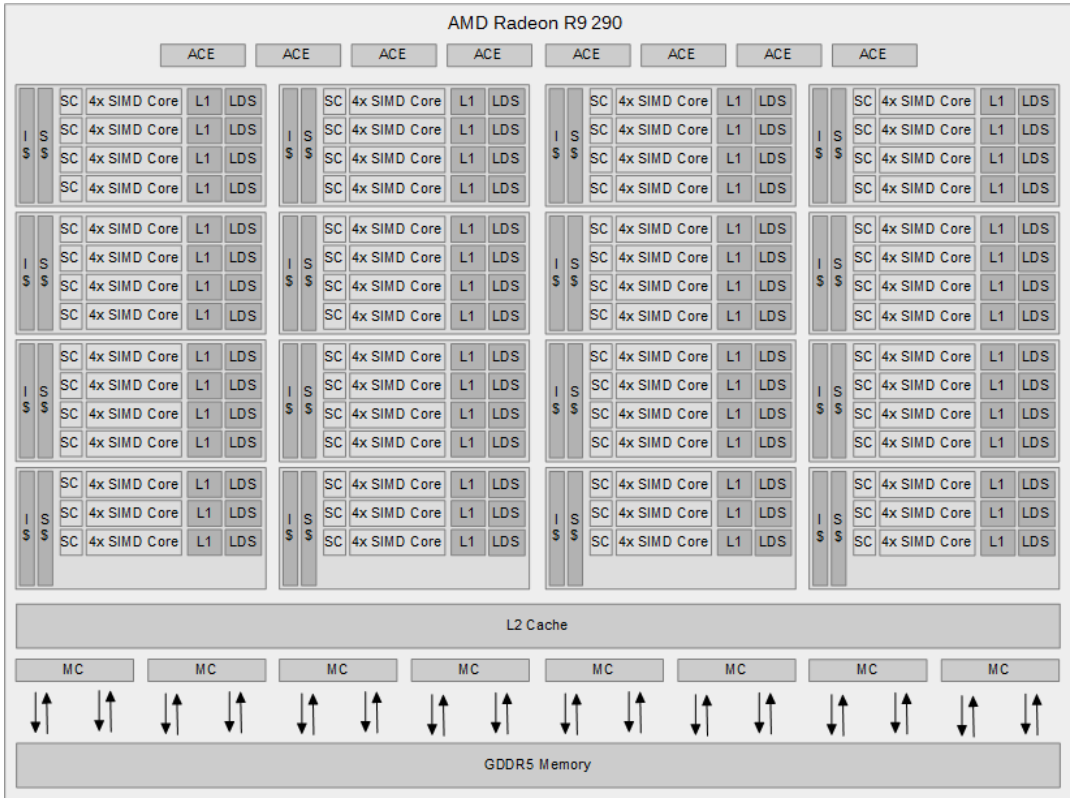
- CPUs are optimized to minimize the latency of a single thread
  - Can efficiently handle control flow intensive workloads
- Lots of space devoted to caching and control logic
  - Multi-level caches used to avoid latency
- Limited number of registers due to smaller number of active threads
- Control logic to reorder execution, provide ILP and minimize pipeline stalls



Example Piledriver-based CPU architecture

Modern GPGPU Architecture

- Array of independent “cores” called Compute Units (AMD) or Streaming Multiprocessors (NVIDIA)
- High bandwidth, banked L2 caches and main memory
  - Banks allow multiple accesses to occur in parallel
  - 100s of GB/s
- Memory and caches are generally non-coherent



Modern GPGPU Architecture

AMD

- Compute units are based on SIMD hardware
  - Both AMD and NVIDIA have 16- wide SIMDs, but map different sized hardware threads onto these units
- Large register files are used for fast context switching
  - No saving/restoring state
  - Data is persistent for entire thread execution
- Both vendors have a combination of automatic L1 cache and a user-managed scratchpad
  - Scratchpad is called local data share (LDS) by AMD and shared memory by NVIDIA
- Scratchpad is heavily banked and very high bandwidth (~ TB/s)

Modern GPGPU Architecture

AMD

- Work-items are automatically grouped into hardware threads called “wavefronts” (AMD) or “warps” (NVIDIA)
  - Single instruction stream executed on SIMD hardware
  - 64 work-items in a wavefront, 32 in a warp
    - Instruction is issued multiple times on 16-lane SIMD unit
- Control flow is handled by masking SIMD lanes

SIMT and SIMD

AMD

- NVIDIA coined “Single Instruction Multiple Threads” (SIMT) to denote multiple (software) threads sharing an instruction stream
- Although each work-item has its own Program Counter (PC), they execute in lock-step on SIMD hardware
  - Multiple software threads are executed on a single hardware thread
- Divergence between threads handled using masking or predication
  - Divergence between work-items is transparent to the OpenCL model and it appears as if work-item has its own PC
- Performance is highly dependent on understanding work-items to SIMD mapping

AMD R9 290X

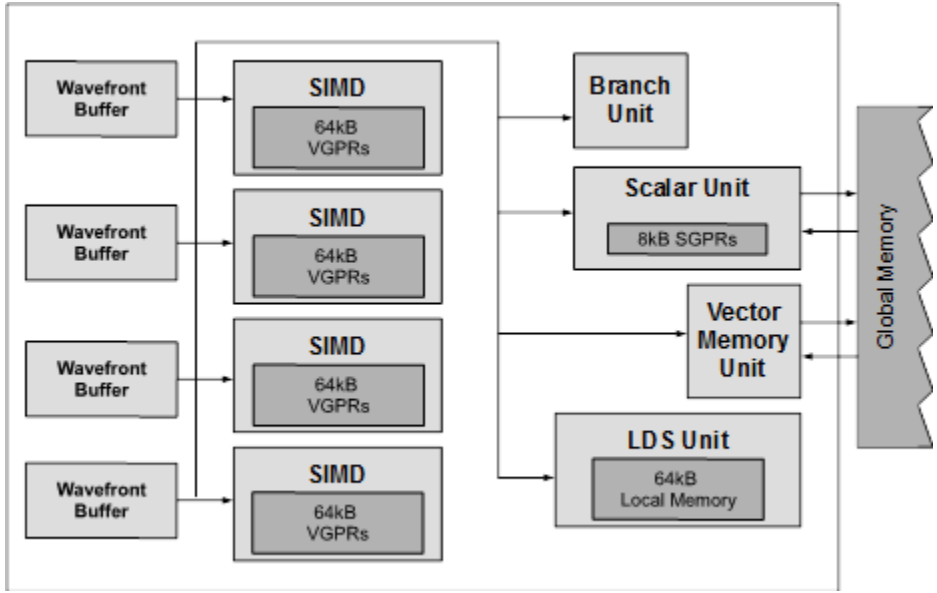
AMD

- R9 200 series GPUs based on Graphics Core Next (GCN) architecture
- 4 SIMDs per compute unit
- 1 Scalar Unit to handle instructions common to wavefront
  - Loop iterators, constant variable accesses, branches
  - Has a single, integer-only ALU unit
  - Separate branch unit used for some conditional instructions
- Radeon R9 290X
  - 44 compute units
  - 5.6 TFLOPS peak performance for single precision

AMD R9 290X

AMD

- Compute unit microarchitecture



AMD R9 290X

- Wavefronts are associated with a SIMD unit and a subset of the vector registers
  - Up to 10 wavefronts can be associated with each SIMD
  - 4 SIMDs
  - 40 wavefronts can be active per compute unit
- All hardware units except for the SIMDs are shared by all wavefronts on a compute unit

AMD R9 290X

- Each cycle, wavefronts targeting one of the SIMDs are allowed to issue instructions
  - Every 4th cycle a wavefront will be active
- An instruction takes 4 cycles to enter the SIMD pipeline (4 subwavefronts per wavefront)
- Scalar unit and branch unit can take 1 instruction per cycle
- All hardware units can remain fully utilized with a simplified front-end using this round-robin technique

AMD R9 290X

AMD

- Up to 5 instructions can be issued per cycle
  - Only 1 per wavefront
  - Only 1 per instruction type (i.e., per hardware unit)
  - Need multiple instructions types present to fully utilize hardware units
- Instruction types
  - Vector Arithmetic Logic Unit (ALU)
  - Scalar ALU or Scalar Memory Read
  - Vector memory access (Read/Write/Atomic)
  - Branch/Message
  - Local Data Share (LDS)
  - Export or Global Data Share (GDS)
  - Internal (s_nop, s_sleep, s_waitcnt, s_barrier, s_setprio)

AMD R9 290X

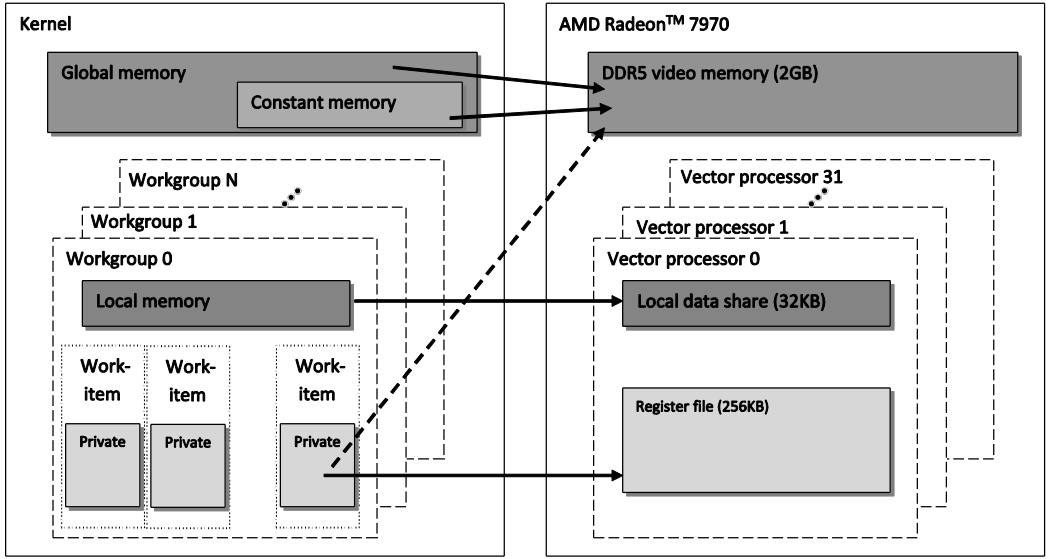
- R/W L1 caches
  - 16 KB per compute unit
  - Write through (dirty-byte mask)
  - 64B lines
- R/W L2 caches
  - 16 partitions with 64 KB/partition
  - Write back (dirty-byte mask)
  - 64B lines
- LDS
  - 64 KB per compute unit
  - 32 banks
  - Contains integer atomic units

AMD R9 290X

- Cache coherence supported at L2-level
  - GLC-bit allows L1 caches to be bypassed
  - Data is strided across L2s (all CUs access all caches)
  - Bypassing L1 allows coherent view at L2-level

AMD Memory Model in OpenCL

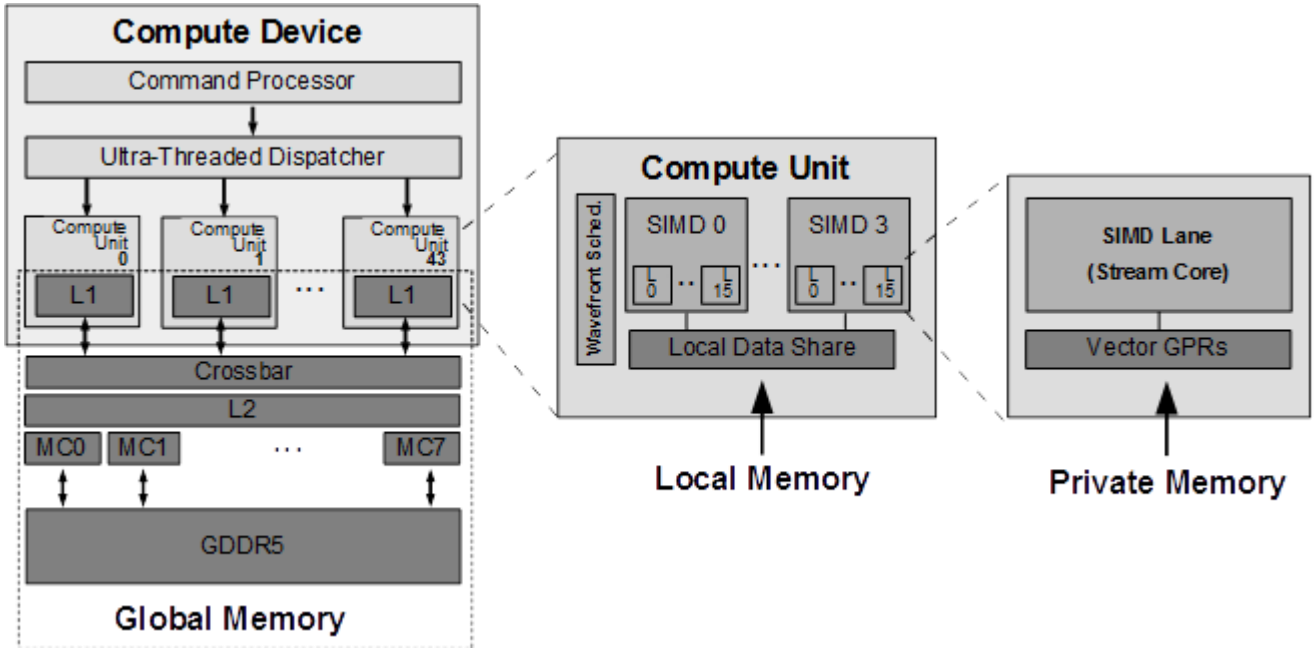
- Global Memory
  - Maps to cache hierarchy
  - GDDR5 video main memory
- Constant Memory
  - Maps to scalar unit reads
- Local Memory
  - Maps to the LDS
  - Shared data between work-items of a work group
  - High Bandwidth access from SIMDs
- Private memory
  - Maps to vector registers



AMD GPU Architecture in OpenCL

AMD

- R9 290X mapped to OpenCL



An Optimal GPGPU Kernel

AMD

- An ideal kernel for a GPU
  - Has thousands of independent pieces of work
    - Uses all available compute units
    - Allows context switching to hide latency
  - Is amenable to instruction stream sharing
    - Maps to SIMD execution by preventing divergence between work items
  - Has high arithmetic intensity
    - Ratio of math operations to memory access is high
    - Not limited by memory bandwidth

Summary

AMD

- We have examined a GPU architecture and how the OpenCL specification maps onto it
  - An important take-away is that even though vendors have implemented the spec differently the underlying ideas for obtaining performance by a programmer remain similar
- We have looked at the runtime compilation model for OpenCL to understand how programs and kernels for compute devices are created at runtime