TU Berlin, Scientific Computing
Winter Semester 2021/2022

Slide lecture 2

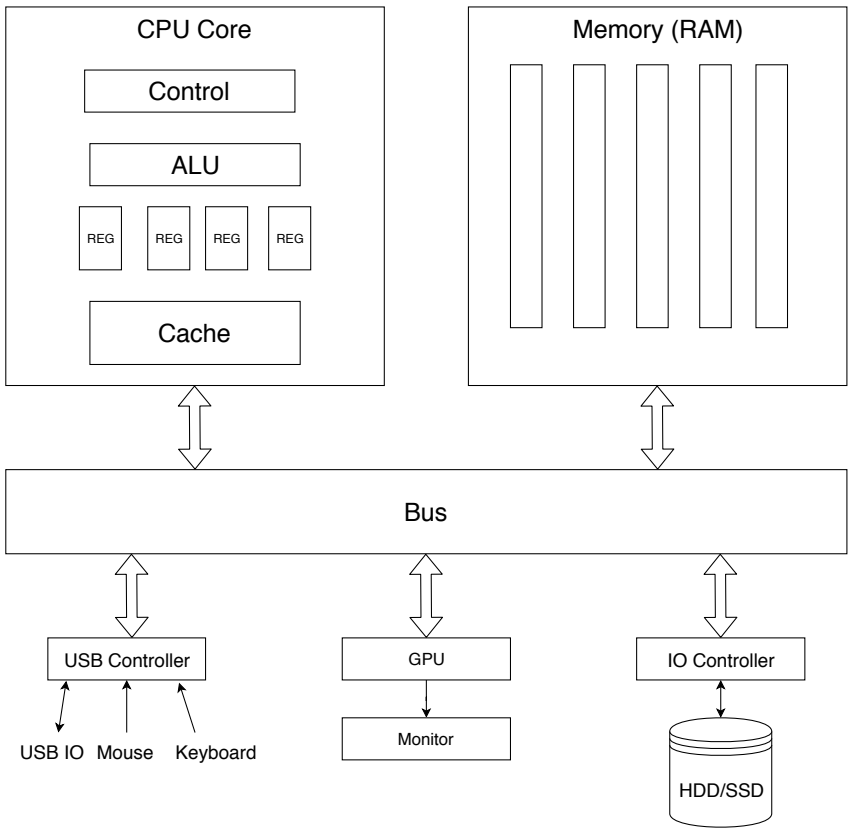
Jürgen Fuhrmann

juergen.fuhrmann@wias-berlin.de

Hardware aspects

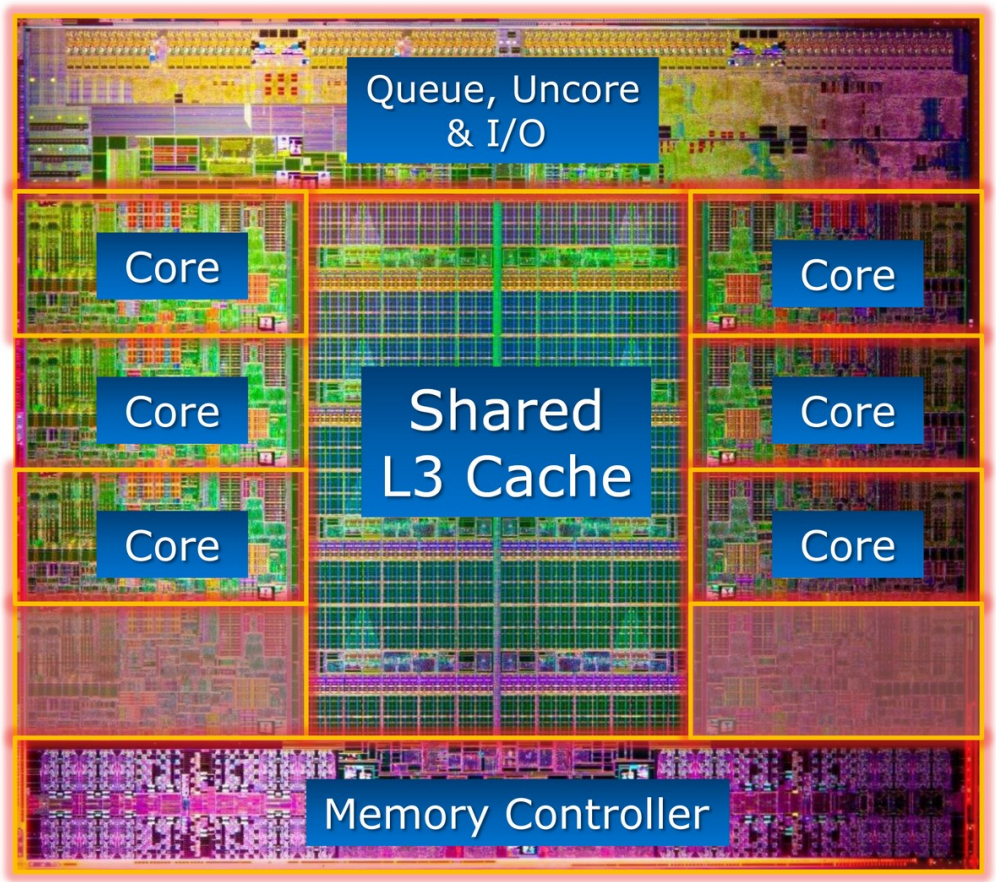
Inspired by "Introduction to High-Performance Scientific Computing" by Victor Eijkhout (http://pages.tacc.utexas.edu/~eijkhout/istc/istc.html)

von Neumann Architecture



- Data and code stored in the same memory $\Rightarrow$ encoded in the same way, stored as binary numbers
- Instruction cycle:
  - Instruction decode: determine operation and operands
  - Get operands from memory
  - Perform operation
  - Write results back
  - Continue with next instruction
- Controlled by clock: "heartbeat" of CPU
- Traditionally: one instruction per clock cycle

Multicore CPU

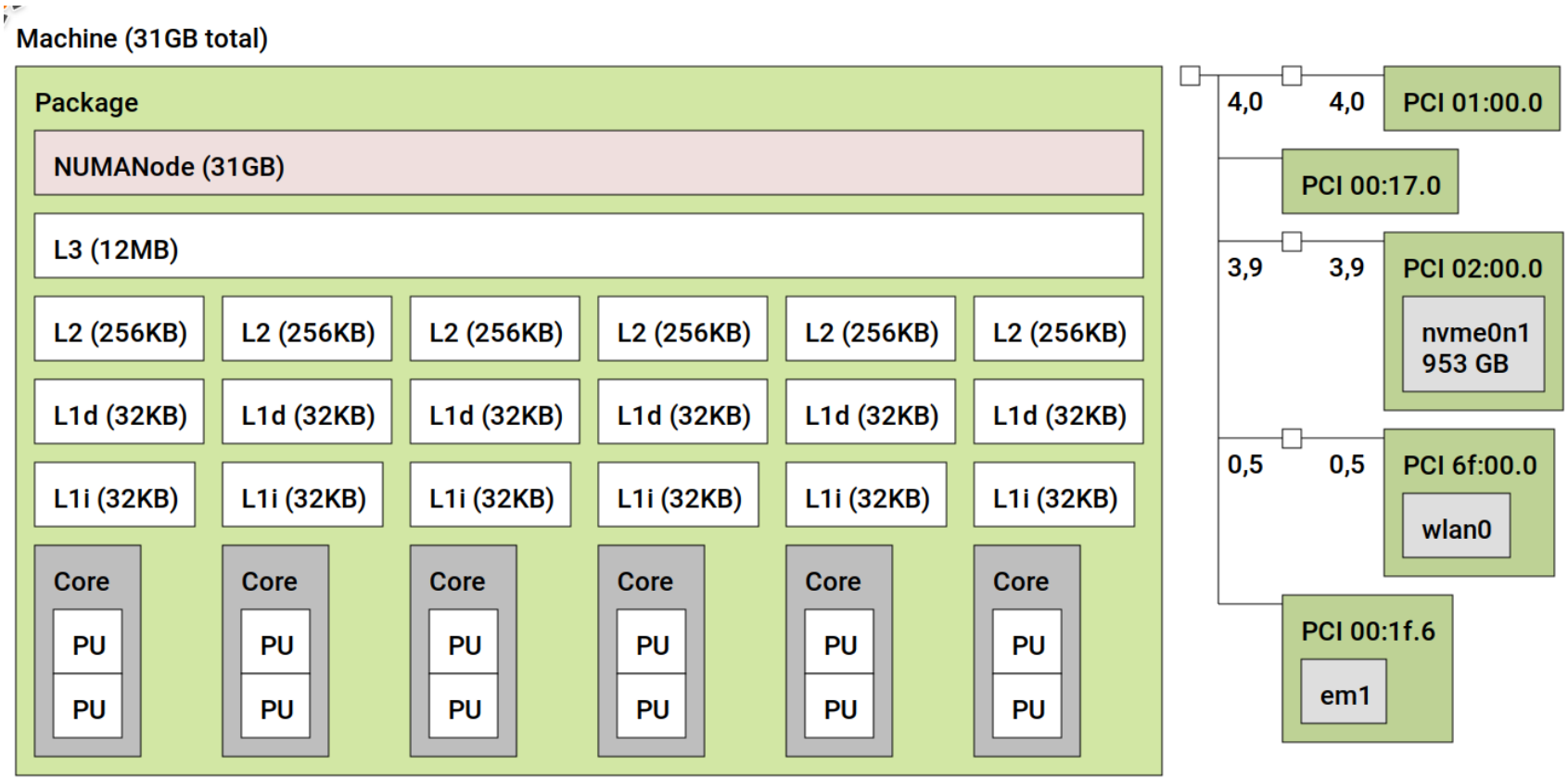


Modern CPU. From: https://www.hartware.de/review_1411_2.html

- Several computational cores on one CPU
- Cache: fast intermediate memory for often used operands

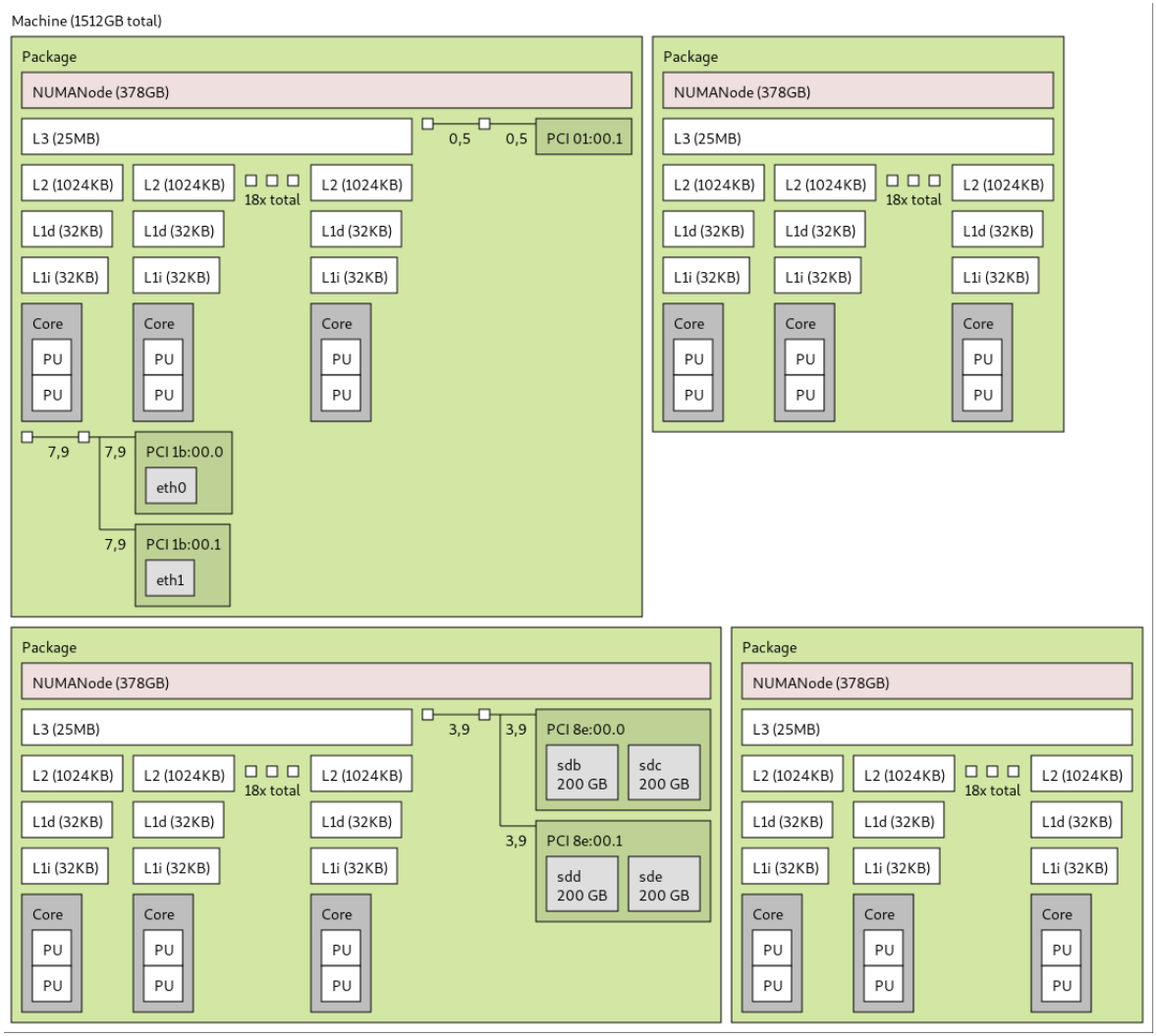
Multicore CPU: this Laptop

Obtained with `lstopo` from the Portable Hardware Locality (hwloc) project
https://www.open-mpi.org/projects/hwloc/



- Three cache levels
- 6 Cores with similar pathways to memory

NUMA Architecture: compute server



- NUMA: Non Uniform Memory Access
- Several packages with 1 NUMA node each
- Each NUMA node has part of the system RAM attached to it.

Modern CPU workings

Levels of parallelism

- Multicore parallelism: independent parallel computations on different cores
- On-core parallelism: Multiple floating point units in one core
- Pipelining
  - A single floating point instruction takes several clock cycles to complete:
  - Steps to execute a floating point instruction:
    - Instruction decode
    - Operand exponent align
    - Actual operation
    - Normalize
  - Pipeline: separate piece of hardware for each step
  - Like assembly line
- Complex instructions, e.g. one multiplication + one addition

Consequences for performance:

- Peak performance is several operations/clock cycle for well optimized code
- Operands can be in memory, cache, register $\Rightarrow$ influence on perfomance
- Performance depends on availability of data from memory

Memory Hierachy

- Main memory access is slow compared to the processor
  - 100–1000 cycles latency before data arrive
  - Data stream maybe 1/4 floating point number/cycle;
  - processor wants 2 or 3 for full performance
- Faster memory is expensive
- *Cache* is a small piece of fast memory for intermediate storage of data
- Operands are moved to CPU *registers* immediately before operation
- Memory hierarchy:

  Registers in different cores
  Fast on-CPU cache memory (L1, L2, L3)
  Main memory
- Registers are filled with data from main memory via cache:
  - L1 Cache: Data cache closest to registers
  - L2 Cache: Secondary data cache, stores both data and instructions
  - Data from L2 has to go through L1 to registers
  - L2 is 10 to 100 times larger than L1
  - Multiple cores on one NUMA node share L3 cache , $\approx$10x larger than L2

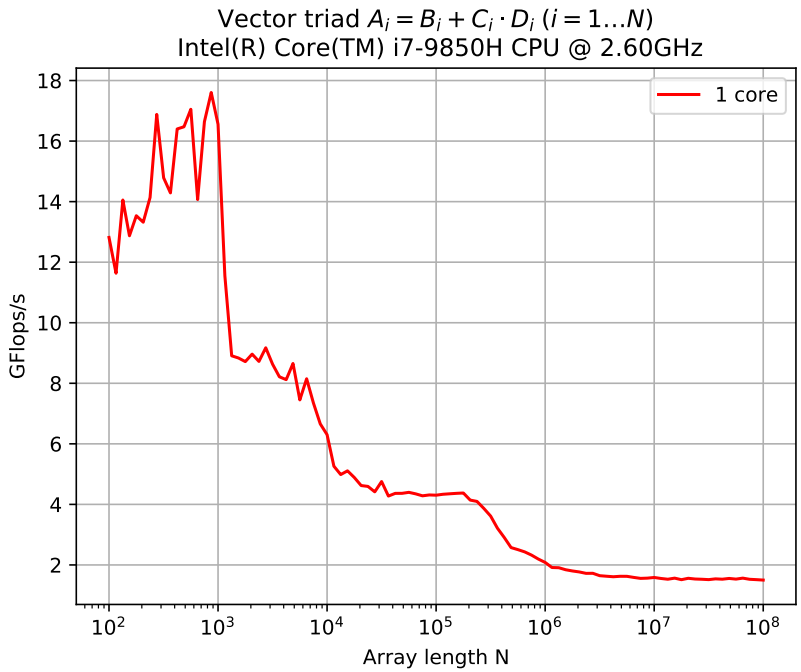
Cache line

- Smallest unit of data transferred between main memory and the caches (or between levels of cache)
- Fixed number of sequentially stored bytes. A floating point number typically uses 8 bytes, and cache lines can be e.g. 128 bytes long (16 numbers)
- If you request one number you get several numbers at once - the whole cache line
  - For performance, make sure to use all data arrived, you've paid for them in bandwidth
  - Sequential access good, "strided" access ok, random access bad
- Cache hit: location referenced is found in the cache
- Cache miss: location referenced is not found in cache
  - Triggers access to the next higher cache or memory
- Cache thrashing
  - Two data elements can be mapped to the same cache line: loading the second "evicts" the first
  - Now what if this code is in a loop? "thrashing": really bad for performance
- Performance is limited by data transfer rate
- High performance if data items are used multiple times

Schönauer Vector Triad

After an idea of G. Hager, see https://blogs.fau.de/hager/archives/7825

Let A, B, C, D vectors of length N. Compute $A_i = B_i + C_i D_i$ for i fron 1 to N.



- $N < 10^3$: L1-cache
- $10^3 < N < 10^4$: L2-cache
- $10^4 < N < 2 \cdot 10^5$: L3-cache

- Four performance zones defined by memory access: L1-cache, L2 cache, L3 cache, memory bound

Conclusion

- Achieving peak performance on modern CPUs is nontrivial
- Caring about performance means caring about data layout and memory access
- This is for your information - there is no time to go deeper here, if you are interested, Victor Eijhout's page `http://pages.tacc.utexas.edu/~eijkhout/istc/istc.html` is a good starting point
- We will revisit this topic when we talk about parallelization