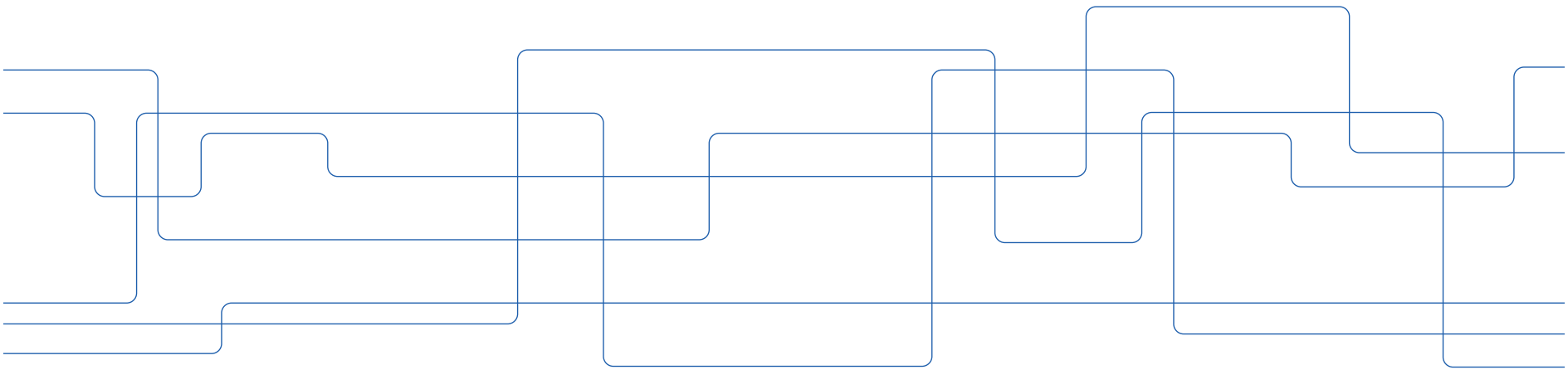
HPC Computer Architectures: Part III

AQTIVATE Training Workshop I

Dirk Pleiter

CST | EECS | KTH

December 2023



Overview

Processor Core Architecture (2)

Memory Architecture (2)

Processor Solutions

Content

Recap Part 1

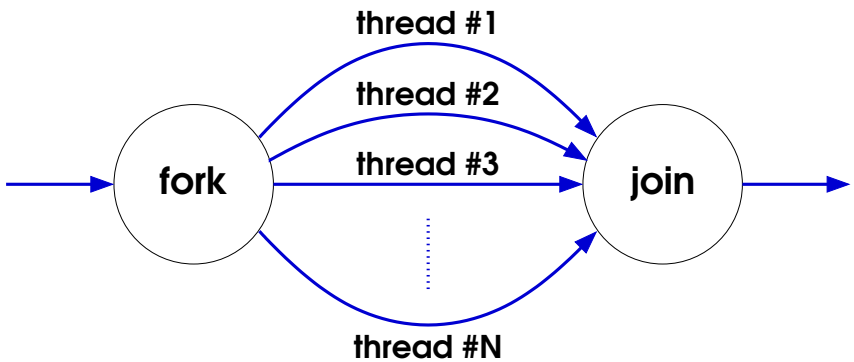
- Processor architecture overview
- Introduction to ISA
- Processor core performance

$$\Delta t = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Instruction-level parallelism (ILP)
- SIMD instructions

Thread Level Parallelism

- **Thread** = Smallest unit of processing that can be scheduled by operating system
- Single core use case: Hide pipeline stalls
  - Time-division multiplexing: If stall occurs then schedule other thread
- Typical execution model:



- Popular Application Programming Interfaces (API):
  - POSIX Threads
  - OpenMP

Simultaneous Multithreading (SMT)

- **SMT** architecture = Architecture which allows > 1 thread per core to be executed simultaneously
  ☞ TLP and ILP are exploited simultaneously
- Architectural requirements
  - Dynamic management of resources, e.g.
    - Dynamic scheduling
    - Register renaming
  - Duplication of resources for each thread
    - Enlarged register file
    - Seperate program counters (PC)
  - Capability for instructions from multiple threads to commit
- Operating systems view: Multiple logical processors
- Also known as: Hyperthreading, hardware threading

Control Flow

- **Control flow** refers to the ability to change the order in which instructions are executed
  - Control flow instructions = **jump** or **branch** instructions
- Example loop

```
for (i = 0; i < 10; i++) {
  ... /* loop body */
}
```

  - At end of loop jump to beginning of loop
  - If condition `i < 10` is false, jump out of loop
- Types of control flow change
  - Conditional branches
  - Jumps
  - Procedure calls
  - Procedure returns

Control Flow (2)

- Control flow instructions must have a destination address
  - Run-time address (e.g. return address)
  - Compile time address ☞ **label**
- Jump instruction in x86 ISA: `jmp`
  Example for an (infinite) loop:

```
        instrA
.L2:
        instrB
        ...
        jmp .L2
```

- Branching typically causes performance degradation
  - Pipelines must be drained and possibly re-filled
  - New instructions have to loaded

Branch Prediction

- Techniques to minimize costs of branching
  - Avoid branching
    - Code re-organisation
    - Use conditioned instructions (e.g., x86 ISA: `cmov`)
  - Loop unrolling
  - Branch prediction
- **Branch prediction** = Anticipate one branch being taken with higher probability then the other branch
  - Practical observation: An individual branch is often highly biased towards taken or untaken
  - Static branch prediction
    - Compiler generates code to assume particular branch to be taken (e.g. branch hints instructions)
    - Decision based on "educated guess" or profiling information
  - Dynamic branch prediction

Content

Recap Part 1

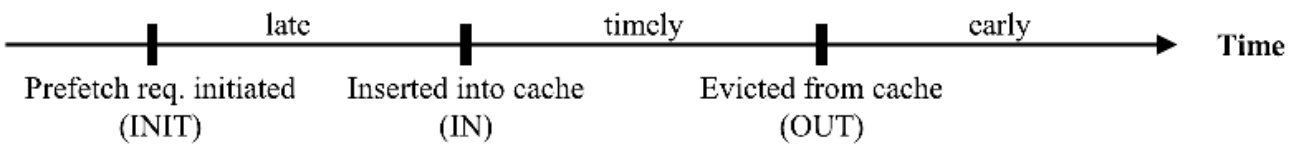
- Memory in modern node architectures
- Memory technologies (DRAM, SRAM)
- Memory locality
- Memory hierarchy and cache architectures

Memory Prefetching

- **Prefetching** = Technique to hide cache miss latencies
  - Idea: Load data before program issues load instruction
- Hardware data prefetching
  - Generic implementation
    - Monitor memory access pattern
    - Predict memory addresses to be accessed
    - Speculatively issue a prefetch request
  - Prefetching schemes
    - Stream prefetching
    - Global History Buffer prefetching
    - List prefetching
  - Prefetching at different cache levels
- Software data prefetching: Compiler or programmer inserts prefetching instructions

Memory Prefetching (2)

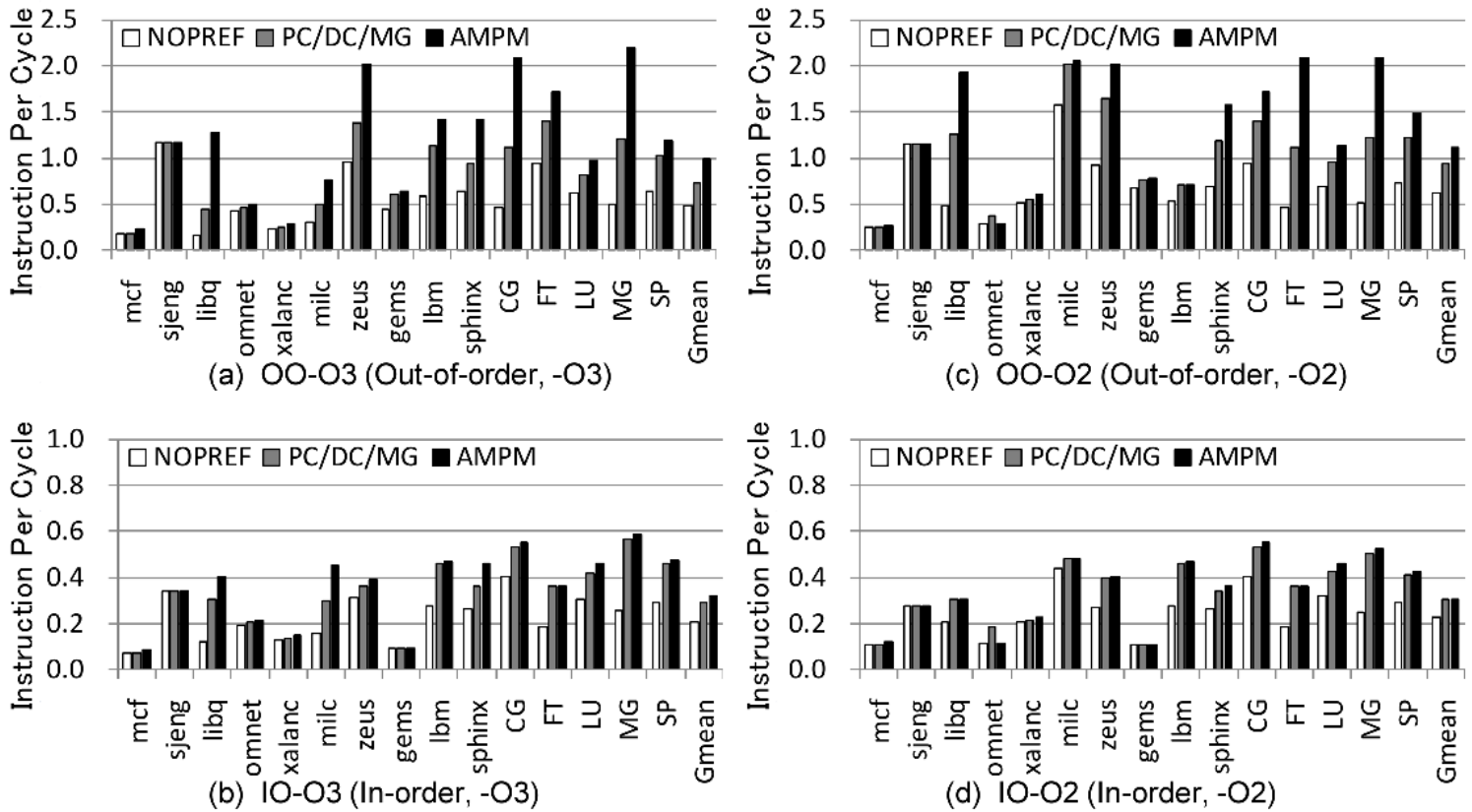
- Risk of negative performance impact
  - Memory bus contention
  - Cache pollution
- Metrics to characterize prefetching algorithms
  - **Coverage**: Number of misses that can be prefetched
  - **Timeliness**: Distance between prefetch and original miss
  - **Accuracy**: Probability that a prefetched line is used before being replaced



[Lee et al., 2012]

Memory Prefetcher Research Example

Access Map Pattern Matching (AMPM) prefetcher:



[Yasuo Ishii et al., 2011]

Cache Coherence

- **Cache coherence** problem
  - Processors/cores view memory through individual caches
  - Different processors could have different values for the same memory location
- Example with 2 processors P_1 and P_2:

Time	Event	Cache P_1	Cache P_2	Memory location A
0				123
1	P_1 reads from A	123		123
2	P_2 reads from A	123	123	123
3	P_1 writes to A	567	123	567

- Value in processor P_2 will only be updated after cache line was evicted ☞ **Need for cache-coherence mechanism**

Cache Coherence (2)

Requirements to achieve coherency

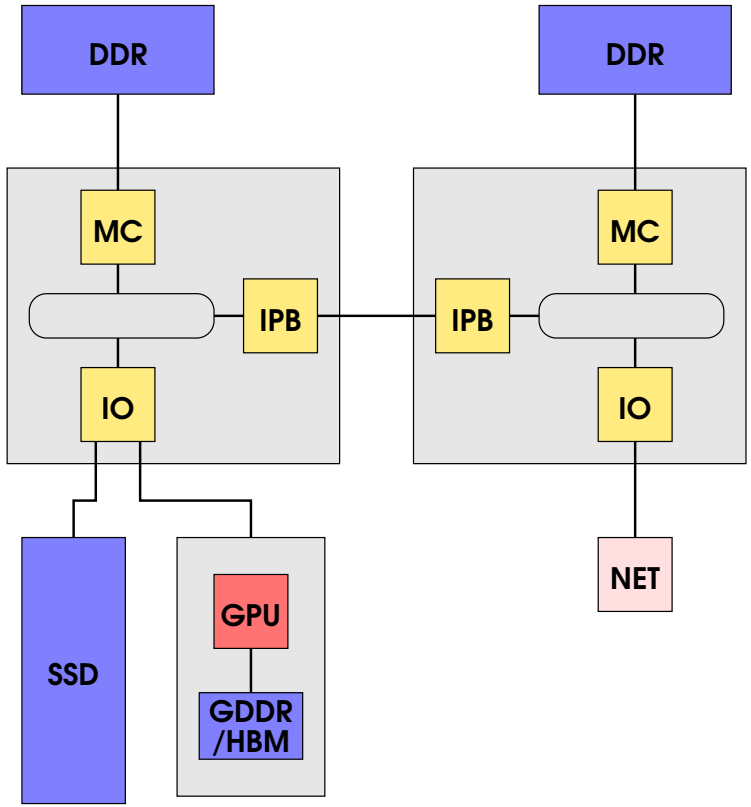
- **Program order preservation**: If a read by processor P_1 from memory location A follows a write to the same location then a consecutive read should return the new value unless another processor performed a write operation
- **Coherent memory view**: If a read by processor P_1 from A follows a write by processor P_2 to A then the new value is returned if both events are sufficiently separated in time
- **Serialized writes**: 2 writes to location A by one processor are seen in the same order by all processors

Cache Coherence (3)

- Classes of cache coherence protocols
  - **Directory based**: The sharing status of a block of memory is kept in one central location (directory)
  - **Snooping**:
    - Protocols rely on existence of a shared broadcast medium
    - Each cache keeps track of cache line status
    - All caches listen to broadcasted snoop messages and act when needed
    - Usually on write other copies are invalidated (**write invalidate protocol**)
- Pros/cons
  - Snoop protocols are relatively easy to implement
  - Directory based protocols have the potential for scaling to larger processor counts

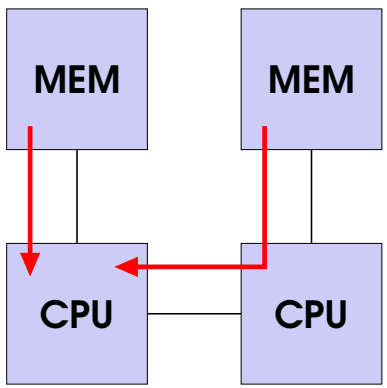
Typical Node Architecture

- Dual-socket node
  - Nodes interconnected via inter-processor bus (IPB)
- Different I/O devices
  - Storage (SSD)
  - Compute accelerator (GPU)
  - Network (NET)



NUMA Architectures

- **NUMA** = Non-Uniform Memory Access
- Memory design in multi-processor system where memory is shared
  - A processor can access another processor's memory



  - Memory access latency depends on which memory is accessed

NUMA: Hardware Locality Control

- Scheduling of processes to cores and placement of memory is controlled by operating system
- NUMA memory allocation policies in Linux

Name	Description
default	Allocate on the local node
bind	Allocate on a specific set of nodes
interleave	Interleave allocations on a set of nodes
preferred	Try to allocate on a node first

- Linux tools to control NUMA policy
  - taskset: Control of process' CPU affinity
  - numactl: Control of process scheduling and memory placement
  - Portable Hardware Locality (hwloc) library/tools

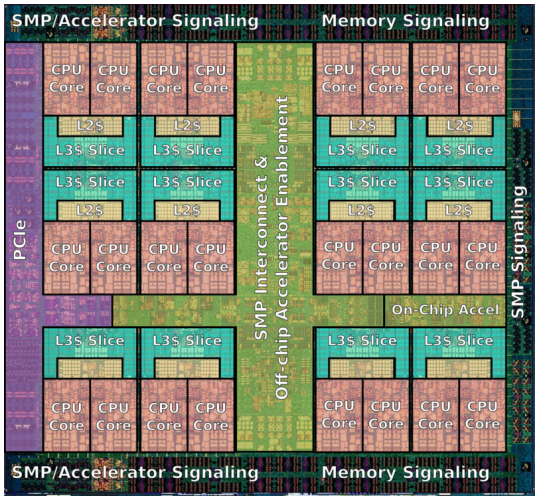
Content

Power Architecture

- Load-store RISC architecture
- Implementations currently used for HPC
  - POWER8, POWER9
    - Multi-core and multi-chip processors
    - Very high memory bandwidth
    - Targets server market and HPC
  - BlueGene L/P/Q processors
    - System-on-a-Chip (SoC) design with network links on chip
    - Only used for BlueGene parallel computers
  - Cell Broadband Engine Architecture (CBEA)
    - Heterogenious multi-core processor
    - Supercomputers based on CBEA: RoadRunner, QPACE
- Also used for embedded technologies and network processors

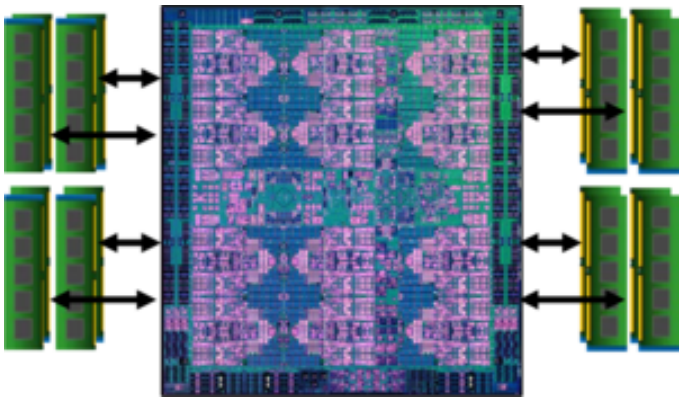
IBM POWER9

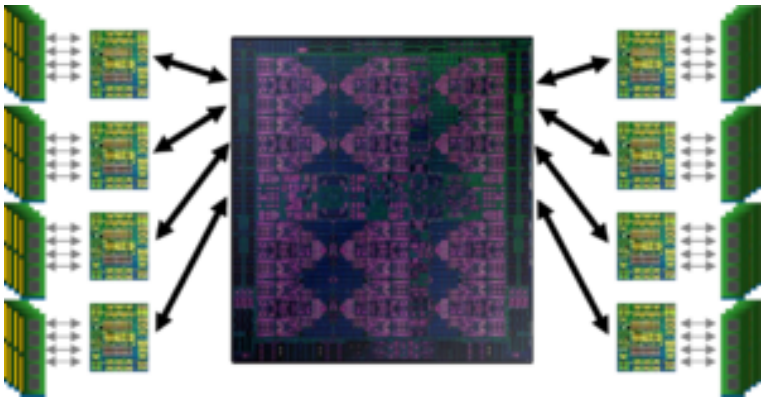
- Introduced in 2017 using 14nm technology
- POWER v3.0 ISA
- Up to 24 SMT4 cores, frequency up to 4 GHz
- Cache parameters
  - L1-D/L1-I: 32/32 kiByte/core, private, 8-way associative
  - L2: 512 kiByte/2 SMT4 cores, partially shared, 8-way associative
  - L3: 10 MiByte/2 SMT4 cores, shared
  - Cache line size 128 Byte
- 8× memory controllers
- Performance figures
  - 8 DP Flop/cycle/SMT4 core



IBM POWER9 (cont.)

[IBM, 2017]





Memory attachment options

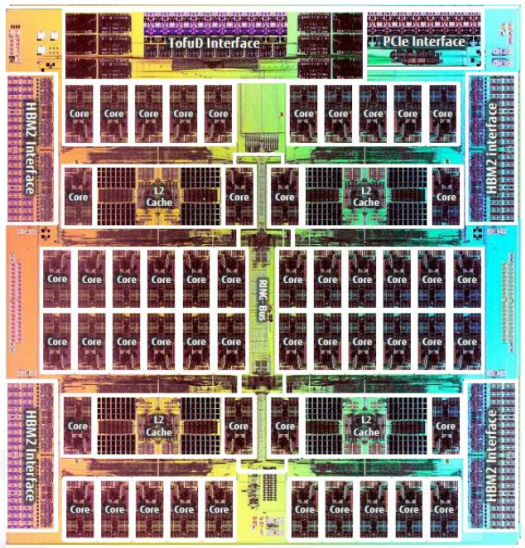
- 8 direct DDR4 channels with up to 120 GByte/s sustained bandwidth
- 8 buffered channels with up to 230 GByte/s sustained bandwidth

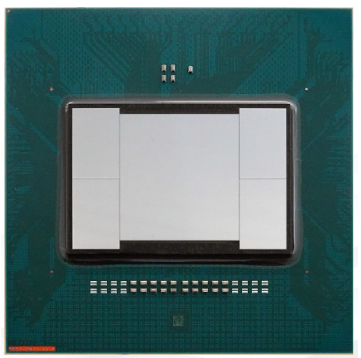
Arm Architecture

- ARM = Advanced RISC Machine
  - Current main market: mobile and embedded electronics
  - Emerging server processor market
- Load/store RISC architecture
- Server-class Arm-based processors
  - ThunderX2 (Marvell)
  - Kunpeng 920 (HiSilicon)
  - A64FX (Fujitsu)
  - Graviton 2, Graviton 3 (AWS)
  - Ampere Altra (Ampere)
  - GRACE (NVIDIA, upcoming)
  - Rhea (SiPearl, upcoming)

Fujitsu's A64FX Processor

- ARMv8 + Scalable Vector Extension (SVE) ISA
- Number of cores: $4 \times (12 + 1)$
- Normal/boost frequency: 2.0 GHz/2.2 GHz
- Cache parameters
  - L1-D/L1-I: 64/64 kiByte per core, 4-way
  - L2: 8 MiByte per (12+1) cores, 16-way
  - Cache line size: 256 Byte
- Performance figures
  - Floating-point performance: 2×16 FP64/cycle/core
  - HBM memory bandwidth: 1024 GByte/s





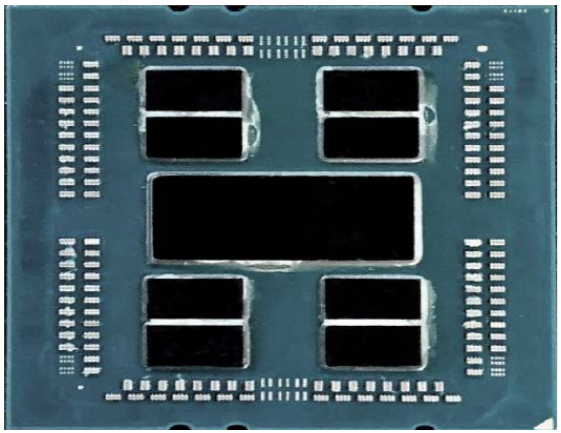
x86 Architecture

- ISA based on Intel 8086 CPU (launched 1978)
- Architecture dominates PC mass market and is used in most supercomputers [Top500, 11/2023]
  - Top500: $\sim$ 95.4% of the top500 systems use x86 CPUs
- Manufacturers: Intel, AMD, VIA
- Main features
  - CISC-type ISA
  - One operand can be a memory location
- Extensions (selection):
  - x86-64
  - SSE2, SSE3, SSE4.x
  - AVX, AVX2, AVX-512

x86 Architecture: AMD EPYC

Hardware parameters of AMD EPYC 7742 (Rome)

Number of cores	64
Base core clock	2.25 GHz
Max. core clock	3.4 GHz
L1 cache (data+instr)	32+32 kBytes/core
L2 cache	512 kBytes/core
L3 cache	256 MBytes
Memory technology	DDR4
Number of channels	8
Max. memory bandwidth	205 GBytes/s
Peak DP performance	16 Flop/cycle/core
Max. TDP	225 Watt
Lithography	7/14 nm
Release date	2019

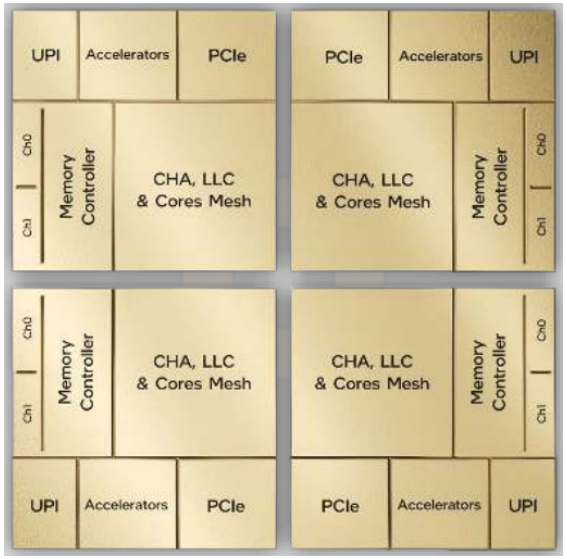


[AMD]

x86 Architecture: Intel Xeon Platinum

Hardware parameters of Intel Xeon Max 9480 (Sapphire Rapids)

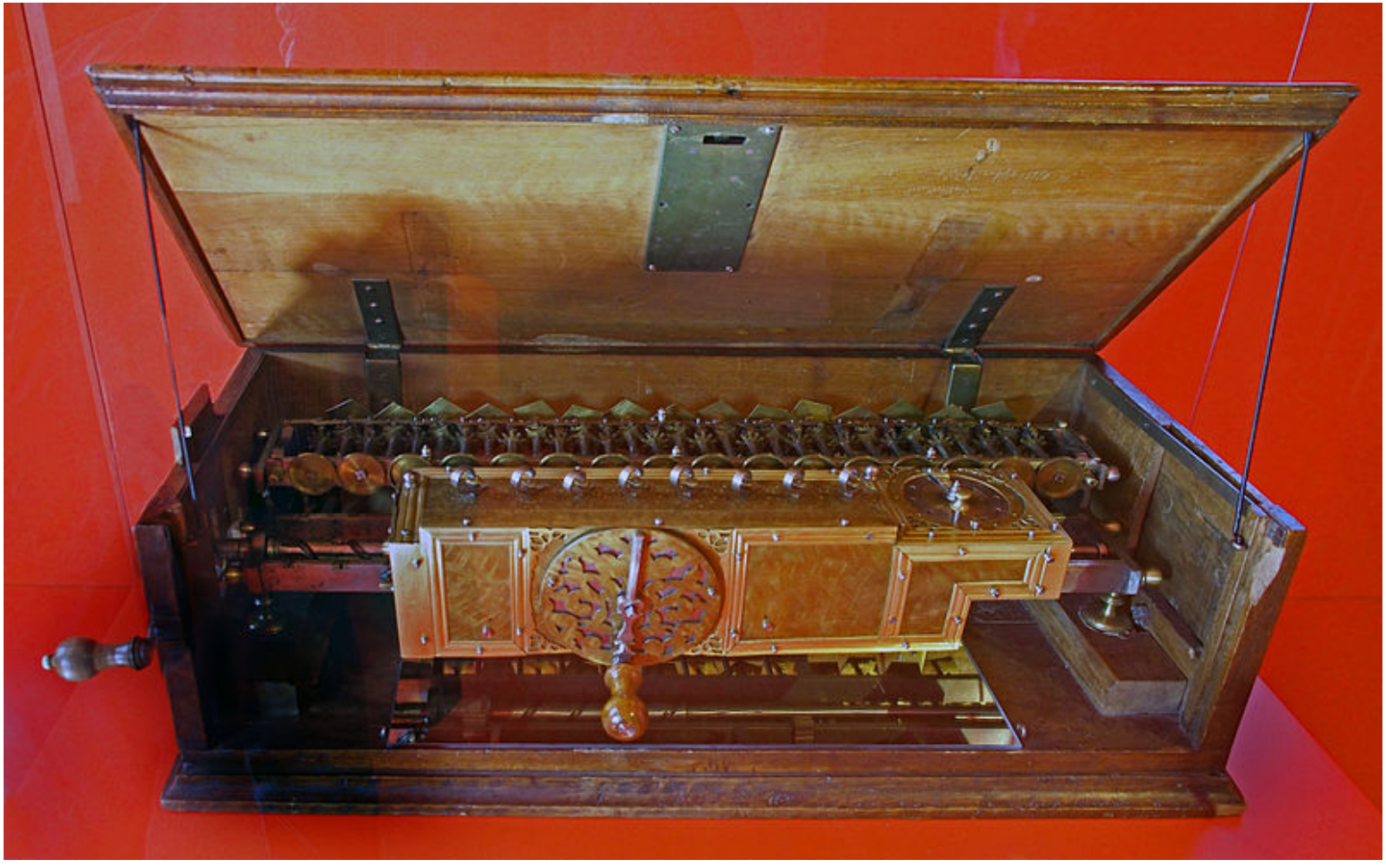
Number of cores	56
Base core clock	1.9 GHz
Max. core clock	3.5 GHz
L1 cache (data+instr)	64+48 kBytes/core
L2 cache	2 MBytes/core
L3 cache	112.5 MBytes
Memory technology	DDR4 + HBM
No. of channels/stacks	8 + 4
Max. memory bandwidth	307+1638 GBytes/s
Peak DP performance	32 Flop/cycle/core
Max. TDP	350 Watt
Lithography	Intel 7
Release date	Q1/2023



[Intel, Hotchips, 2023]

Nominal performance numbers as specified by manufacturer

Finish with a Simple Architecture: Leibniz' Reckoner



[Museum Schloss Herrenhausen]