FACULTY
OF INFORMATICS
Masaryk University

PA039: Supercomputer Architecture and Intensive Computing

Luděk Matyska

Spring 2023

Rules of engagement – proposal

- Regular lectures (like this one) versus pre-recorded lectures combined with an interactive seminar at the regular scheduled time
- Interactive seminar form:
  - Your questions
  - More detailed discussion around selected topics
  - Questions through Sli.do
  - Examination questions/subjects
- Examination will be in an **Open Book** format
  - IS MU
  - The first (mandatory) examination on 18th May at 4 PM (the time of the last possible lesson in the semester)
  - Other terms as needed (for those who can't make it for serious reason)

However, due to the AI ChatGPT (and derivatives), this form can't by confirmed yet

High Performance Computing

- Formula One in the IT area
  - Extremely expensive machines, but with exceptional features (performance, memory, ...)
- Specific users' groups
  - Extensive simulations
  - Modelling (automobile, airplanes, physical phenomena, ...)
  - Recently also Artificial Intelligence (DNN)
- Appetite comes with eating
  - Requirements rise faster than performance
  - Also the complexity of processors rises

Quality of programming (code generation) defines the usability and real performance

High Performance Computing II

- Processors
  - CISC
  - RISC
  - Vector processors
  - Streaming processors (e.g. GPU, TPU)
  - Special processors
    - Programmable – FPGA
    - Static – ASICs
- Memories – performance (speed) lagging behind processors

HPC– requirements

- The ratio Theoretical vs. Actual performance decreases
- Reaction: need to better understand
  - architecture of the used processor and computer
  - reasons, why a specific code is much faster than seemingly similar equivalent code
  - tools and methods how to measure real performance (of a program or a computer)

High Throughput Computing

- Highest actual performance vs Highest utilization
    - long-term efficient use of computer systems
    - large number of smaller tasks
        - the processing time of a single task is not critical
        - the time of processing **all** tasks is critical
    - Efficiency
        - maximizing “investment”
        - total throughput of the system

Clouds and HPC

- Clouds – virtualized infrastructure
  - higher flexibility (use as much as you need (and can pay for))
  - robustness (high availability)
  - hidden and exposed heterogeneity
  - massive capacity
    - resembles High Throughput Computing goals
- Basic scenario – overcommitment
  - not directly usable in the HPC environment
- The flexibility and fast availability of resources may be the primary force for using clouds in the HPC environment
  - But it may broke some of the efficiency expectations
  - New features are needed
- Area to follow (e.g. EOSC or GAIA X)

Fundamental aspects – what influences performance

- Latency (delay)
  - processing and transmission of signals within processors and memories
  - transmission between processor and memory
  - intra-memory latency
- Speed of (signal) recovery (cycle times)
  - speed of circuits switching
  - circuits frequency (internal “clock”)
  - memory refresh (dynamic memories)
- Throughput (speed of the data unit transfer)
  - speed of data transport on a chip
  - number of instructions per a cycle
  - transport speed between components
- Granularity
  - density on a chip
  - memory density
  - task size

Processors – CISC

Complex **I**nstruction **S**et **C**omputer

- Examples:
  - PDP 11, VAX, IBM 370, Intel 80x86, Motorola 680x0, ...
- Principle:
  - Don't use program where you can use hardware
- The term "CISC" was in fact created as an opposite to RISC (i.e. not used since the introduction of CISC processors)

FACULTY
OF INFORMATICS
Masaryk University

Raison d’etre

- Size and speed of memory
    - when compared with the speed of processors
- CISC directly supports compilers (principles of co-design)
- Rich addressing modes (how to address data in a memory)

Microprogramming

CISC – complex instructions

- Control part of the processor too complex/extensive
  - Microinstructions: decomposition to simpler instructions
- Complex instruction == microprogram

Simpler hardware design

- Instructions are in fact *emulated* (internal “computer”)

It is “easy” to change instruction set of a particular computer
$\Longrightarrow$ *computer families* (IBM 360, 370, VAX, ...)

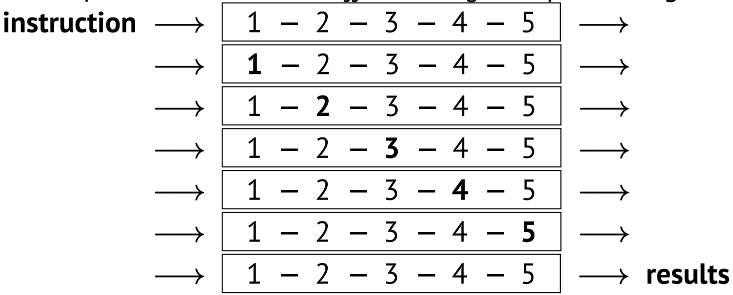
Disadvantages: too complex instructions, increasingly complex instruction analysis, cross instruction relationships, backward compatibility cost (within a family)

Performance increase

- Clock cycles define processor's performance
  - Limited by contemporary technology
  - Impossible to continuously increase
    - dependencies between components
    - signal transport speed
- Solution: **parallelization**

Pipelining

Overlap of instructions in *different stages* of processing



Three basic areas:

1. Instruction processing
2. Memory access
3. Floating point instructions

Pipelining II

- “Standard” (classical) instruction decomposition (five-stage pipelining):

  Instruction Fetch instruction is loaded from a memory

  Instruction Decode instruction is decoded (recognized)

  Operand Fetch operands are ready (fetched from registers and/or memory)

  Execute instruction is executed

  Writeback results are written back (to registers and/or memory)

  Individual stages are processed in parallel, shifted by one stage

Pipelines and memory

- “Invisible” pipelines
  - Reading (writing) from (to) memory is moved ahead of the actual instruction that works with the data
- “Visible” pipelines
  - Explicit instructions, with know number of cycles to complete
  - E.g. Intel 80860

Processors – RISC

Reduced **I**nstruction **S**et **C**omputer

- First RISC: CDC 6600 (Seymour Cray)
  - First half of sixties (1964)

  Explicit RISC concept during eighties
- (Favourable) conditions for RISC processors
  - Introduction of caches
  - Dramatic decrease in the memory cost paralleling increase of memory size
  - Better pipelining
  - Improved compilers

RISC conditions II

- Architecture removed the speed of memory access bottleneck
  - use of caches
  - use of internal registers (decreased number of direct memory accesses)
- Size of a program became irrelevant (even extensive code can fit into a memory)
- Problem: *stall* when waiting for a next instruction execution finalization (too complex relationship between instructions, microcode, ...)
- Solution: complex instructions are not needed, microprograms can be replaced by explicit code
  - also, readability of code (assembler) no more critical

RISC characteristics

- All instructions of the same size/length (e.g. 4 bytes)
- Careful selection of really needed instructions
- Simple addressing
- Load/Store architecture
- Sufficient number of internal registers
- “Delayed’ branches
- Examples:
  - Initially some foreruners: MIPS (Stanford) a SUN SPARC (UoC, Berkeley) architectures
  - IBM and their Power Architecture (PowerPC, family of POWER processors)
  - HP with PA-RISC
  - DEC Alpha
  - Intel I860 and i960 or Motorola 88000
  - ARC, ARM, ...

RISC – advanced design

- First generation RISC ideal:
  - One instruction finished per each clock tick
- Reality nowadays:
  - Several instructions graduated in a single clock tick

New features

- Superscalar
- Superpipeline
- (Very) Long Instruction Word, (V)LIW

Superscalar processors

- Multiple processing units
  - Arithmetic (ALU), Floating point (FPU) and other
- Examples:
  - RS/6000, SuperSPARC and newer, Motorola 88110, HP PA 7100 and newer, DEC Alpha, MIPS R8000 and newer, Intel processors, IBM POWER processor, ARM

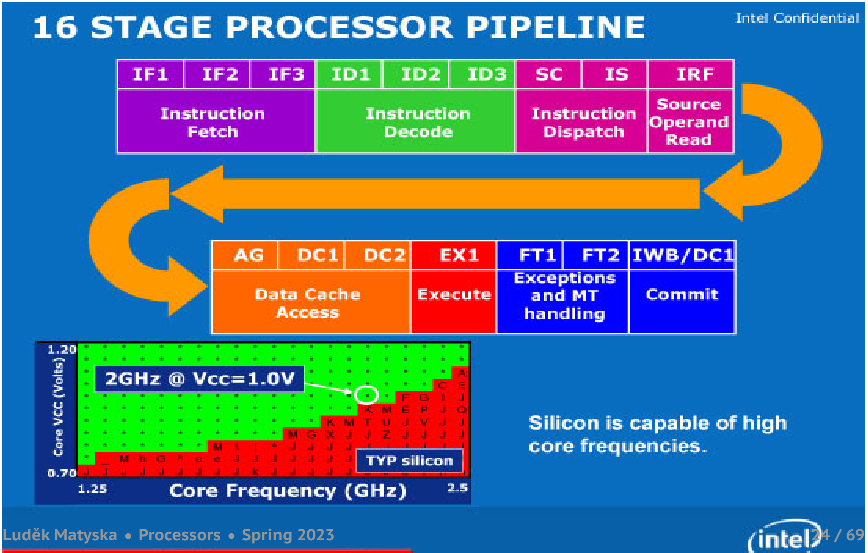
Superscalar processors – features

- Parallelism in a hardware
  - Sequential programs
  - “Automatic” parallelization (intra-processor parallelization)
    - Several instructions fetched to pipeline
  - MADD (Multiply Add)
    - Operation X*Y+Z

Superpipeline

- Another circuits simplification
  - More extensive pipeline decomposition
  - Faster execution of individual stages
- resulting in faster processing
  - A different form of parallelism
- These pipelines also called *deep* pipelines
  - 16 and more stages
  - instructions use only some of the whole set of stages

16 stage pipeline



VLIW

- Analogy of superscalar processors (many units)
- Parallelization under compiler control
  - Increased complexity of compilers
  - Simplified hardware leads to higher performance
  - Decision which instructions can be run in parallel taken by the compiler
- Advantages:
  - Simpler instructions
  - No complex control hardware needed
  - Lower energy consumption (at least a potential for it)
- Examples:
  - Intel i860
  - triMedia media processors
  - C6000 DSP family (Texas Instruments)
  - Itanium IA-64 EPIC (partially)
  - Crusoe processors from Transmeta
  - Russian supercomputers Elbrus

RISC – additional features

- Register’s bypass
- Register’s renaming
- Branches
  - null operation
  - conditional assignment (`a = b<c ? d : e;`)
  - multiple “pre-fetch” from memory
  - buffer of potential branch targets
  - branch prediction
    - static (complied)
    - dynamic

ANDES

Architecture with **N**on-sequential **D**ynamic **E**xecution **S**cheduling

- Foundations
    - Waiting for data causes a slowdown
    - Dynamic parallelism can help
- Examples
    - HP PA 8000, MIPS R10000, ...
- Also called *out-of-order execution (OoOE)* or *dynamic execution*

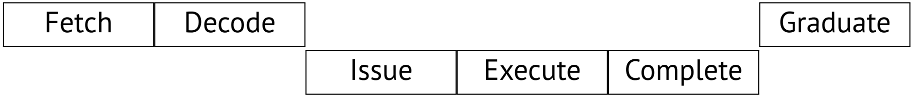
ANDES – Architecture

- Multiple instructions queues
  - a queue for fixed point (arithmetic and logic) instructions
  - an address queue for **Load/Store** instructions
  - a queue for floating point instructions

  Independent pipeline for each queue
- Features
  - *readiness* decides which instructions are executed
  - the original order of instructions in the program is not kept
  - instruction *graduation* guarantees restoration of the original order

FACULTY
OF INFORMATICS
Masaryk University

ANDES – speculative execution



ANDES – Additional properties

- Speculative branches:
  - execution continues through *predicted* branch
  - does not wait for the result of the actual branch instruction
- Non-blocking Load/Store instructions
- Register renaming
  - Just part revealed (to a complier)
  - Multiple versions of the “same” register
  - Allows new data to be taken to a register “blocked” by a not yet finished execution of another instruction

ANDES – Summary

- Instead of waiting for memory access, other instructions are executed
- Internally changes the *program order* of instructions to the *data order*
  - instructions are executed when their operands are ready (in registers)
  - the original program order is not relevant
- Avoids (or at least reduces) **stalling** of execution
- *Speculates* on branches or even takes both branches in parallel
  - Only one branch graduates
- Complex circuitry, higher power consumptions

ANDES – Problems

- **Meltdown** and **Spectre** vulnerabilities
  - Disclosed in January 2018
  - Many variants followed
- Consequences of speculative execution
  - The not yet graduated instruction should not cause an exception
    - it may never graduate
  - yet it can have side effects that are observable
- Speed of access to the cache
  - Depends on whether the item is already in cache or not
  - You can **force** upload of a memory that is not accessible through your program
  - and measure the time it takes
- More at `https://meltdownattack.com/` and also `https://arxiv.org/pdf/1811.05441.pdf`

Memory

- Memory organization:
  - rows and columns (a 2D matrix)
  - address has two parts
  - *page mode* – a block or continuous bytes (the “row”) is read in one shot

Memory Features

- Memory access time
  - access row **plus** access column **plus** access (read or write) data
- Memory cycle time
  - defines how often we can read data from a memory
- Both depends on type of the memory (static or dynamic)

Virtual Memory

- Physical vs. logical address
  - More address spaces
- *Translation Lookaside Buffer (TLB)*
  - translates logical addresses to their physical equivalent
  - a part of processor hardware
  - TLB can have misses like any other cache
- Virtual memory and supercomputers
  - usually not used in specialized architectures
  - now common due to the synergic architecture (clusters)

Cache

- Size: several kB to tens of MB
- Organization: fixed length rows (16–128 bytes)
- Types:
  - direct mapped
  - set-associative
  - fully-associative
- Hit ratio

Memory Architectures

- *Harvard Memory Architecture*
  - separated memory for data and instructions
- Programmable cache
  - cache directly controlled at some superscalar processors (e.g. DEC Alpha)

Direct mapped cache

- Static mapping
  - each cache row can keep only pre-defined memory rows
- Fast
- Simple circuits
- Potentially inefficient
  - used data may map to a single cache row

Fully associative cache

- Dynamic mapping
  - associative memory
  - each row in a cache knows where data lie in the main memory
  - access to cache goes to all rows
  - need to select a row for *invalidation*
- Very efficient
- Very complex circuits – expensive

Set associative cache

- Sets of direct addressable caches
- Combination of positive properties of the previous cases
  - usually 2 and 4 way
- Not full associativity (cheaper)
- Still options where a memory row can be stored
  - invalidated row selection
  - must keep the memory address

Width of data flow

- **Bandwidth** = maximal throughput of a memory system
  - measured in bytes per second

  Throughput is not the same among all components
  - Processor – registers – cache – main memory – external memory
- Latency
  - Time between the request and the actual data delivery
  - Extremely important esp. when moving small data chunks

Interleaved Memory

- The whole memory split to smaller blocks
  - Consecutive addresses mapped to different blocks
  - Allows immediate access
- 2 − 8 way interleaved memories common
  - supercomputers can have higher level of interleaving
    - Example: Convex C3 with 256-way interleaving
    - Clock 16 ns
    - Repeat access to the bank: 300 ns (almost 20 time slowdown)
- Higher latency
  - Mitigated by use of pipeline

Re-ordering of memory accesses

- ANDES predecessor
- Minimization of consecutive accesses to the same memory bank
- Run-time check on Load and Store interdependencies
- Example: Motorola 88110

Processor MIPS R8000

- Introduced in 1993
- Clear example of the basic concepts
- 4 way superscalar, max 6 ops/cycle
  - Dual ALU, dual FPU, and two Load/Store units
  - FPU with IEEE-754 standard arithmetic but imprecise interrupt
  - 32 registers (64 bits) for integer and 32 registers (64 bit) for floating point operands
  - Conditional move instructions (IF command support)
- Fully 64 bit architecture
  - 128 bit data bus
  - 40 bit address bus (up to 1 TB addressable memory)
  - 2-way TLB, 384 entries

MIPS R8000 (II)

- Caches
  - 16 KB I-cache (instructions)
  - 16 KB D-cache (2-way, for arithmetic/integer data)
  - 2 KB branch prediction cache
  - 4 MB streaming cache (floating point instructions)

MIPS R8000 – I-Cache

- Instruction cache
  - direct mapped
  - 1024 entries 128 bits each
  - accessed and tagged by virtual address
    - By passes TLB
  - tag RAM – 512 entries (for each row)
    - tag
    - ASID (Address space identifier)
      ASID distinguishes same virtual but different physical addresses
    - validity bit
    - two area bits

MIPS R8000 – D-Cache

- Data cache
  - direct mapped
  - two parallel accesses
    - 2 `load` or one `load` and one `store` instructions concurrently
    - accessed by virtual, tagged by physical address
    - Write-through protocol

MIPS R8000 (IV)

Comparison of implemented caches

Parameter	I-cache	Branch	D-Cache	TLB
Sice	16 KB	2 KB	16 KB	
Entry	128 bit	16 bit	64 bit	
No of entries	1024	1024	2048	384
Port No	one	one	two	two
Mapped	direct	direct	direct	3-way
Index	Virtual	Virtual	Virtual	Virtual
Tag	Virtual	N/A	Physical	N/A
Access	one cycle	one	one	one
Width	128 bit	16 bit	64 bit	
Throughput	1,2 GB/s	159 MB/s	1,2 GB/s	
Row	32 bytes	N/A	32 bytes	
Miss penalty	11 cycles	3 cycles		

MIPS R8000 (V) – Instruction execution speed

Fixed point	Latency	
Add, shift, logical	1	
Load, store	1	
Multiply	4 (6)	
Divide	21	(denominator $\leq$ 15 bitů)
	39	(denominator 16–31 bitů)
	73	(denominator 32–64 bitů)

Floating point	Latency	Stall
Move, negate, abs value	1	1
Add, Multiply, MADD	4	1
Load, Store	1	1
Compare, cond. move	1	1
Divide	14 (20)	11 (17)
Square root	14 (23)	11 (20)
Reciprocal	8 (14)	5 (11)
Reciprocal sq. root	8 (17)	5 (14)

Processor MIPS R10000

- Introduced 1996
- ANDES architecture, three queues
- Superscalar, 4 concurrent instructions
  - 2 ALU and 2 FPU (non-equivalent)
  - FPU with standard IEEE-754 arithmetic and precise interrupts
  - 32 (64 physical) registers (64 bit) for integer operands
  - 32 (64 physical) registers for floating point operands
  - register renaming
- Fully 64 bit architecture
  - 128 bit data bus, 40 bit address bus
  - TLB fully associative, 64 entries (dual)
  - Page size 4 KB−16 MB

MIPS R10000 (II)

- Caches
  - 32 KB I-cache (2-set associative)
  - 32 KB D-cache (2-way, 2-set associative)
  - branch prediction (4 levels)
  - 1 MB L2 cache
- Non-blocking `Load` and `Store` instructions

MIPS R10000 (III)

- Computational units
- 2 ALU
    - In both
        - Add, Sub, and logical instructions
    - Specific
        - ALU1: branches and shift instructions
        - ALU2: multiplication and division (iteratively)
    - 2 FPU (plus two other units outside the pipeline for division and square root (iteratively))
        - FPU1: adder
        - FPU2: multiplier

MIPS R10000 – Queues

- **Integer**
  - 16 entries
  - up to 4 instructions written concurrently
- **Float**
  - 16 entries
  - up to 4 instructions written concurrently
  - impossible to start concurrently the `Divide` and `Square root` instructions
  - MADD instruction uses both FPUs

MIPS R10000 -Queues (II)

- **Address**
  - 16 entries (FIFO)
  - instructions could be executing an arbitrary order
  - write and fetch must be sequential (guaranteed by the FIFO buffer)
  - re-execution of an instruction in case of failure (cache miss, conflict, dependency)

MIPS R10000 (V) – Execution speed

Integer	**Latency**	**Stall**
Add, shift, logical, branch	1	1
Load, store	2	1
Multiply (32 bit)	5–6	6
Multiply (64 bit)	9–10	10
Divide (32 bit)	34–35	35
Divide (64 bit)	66–67	67
Int to Float (32 bit)	4	1

Float	**Latency**	**Stall**
Move, negate, abs value	1	1
Add, Conversion, Mult	2	1
Load, Store	3	1
MADD	4	1
Divide	12 (19)	14 (21)
Square root	18 (33)	20 (35)
Reciprocal sq. root	30 (52)	20 (35)

Processor UltraSPARC-I

- Introduced 1987 (Sparc V9)
- 4 way superscalar architecture
  - 2 ALU, FPU (plus 2 instructions), **GRU (Graphics)**
  - 32 FPU (64 bit) registers
- 64 bit architecture; could select between little and big endian
  - 128 bit data bus, 41 bit physical address, 44 bit virtual address
  - 64 entries in TLB, pages 8 kB, 64 kB, 512 kB or 4 MB
- Visual Instruction Set – GRU

UltraSPARC-I (II)

- Caches
  - 16 KB non-blocking D-cache
  - 16 KB I-cache (with branch prediction)
  - 0,5 – 4 MB L2 cache (throughput 3,2 GB/s)
- Blocking load/store instructions

UltraSPARC-I – Execution units

- FPU
  - Division and square root independently (outside FPU pipeline)
  - 12 (22) cycles for single (double) precision
  - non-blocking pipelined FPU instructions
  - precise interrupts
- GRU
  - 16 and 32 bit combined add and logical instructions
  - 8 and 16 bit multiplication
  - scatter and gather
  - direct access to (graphical) memory bypassing D-cache
  - direct support of a “motions compensation”

Intel and AMD

- 32 bit CISC architecture (IA32)
  - Legacy from 16 bit 8086 + 8087 a 80286
  - Representatives: 80386 (i386), i486, Pentium (i586), i686
- 2001: Itanium (IA64)
  - new design, not backward compatible with IA32
  - collaboration with HP, “clear” RISC architecture
- 2003–2004: AMD Opteron and Intel Xeon Nocona
  - conservative (backward compatible) IA32 extension
  - AMD64, EM64T/Intel64, x64, a joint title **x86-64**

Intel Itanium

- First generation (till 2001)
  - speculative execution, branch prediction, register renaming
  - coarse grained multithreading
  - 128 64 bit int a 128 82 bit float registers
  - up to 6 instructions per a cycle
  - 6 ALUs, 4 MADD units
  - special instruction for multimedia etc.
  - hardware support for virtualization
  - slow IA32 *emulation*, missing compilers, average performance
- Second generation (2002–2010)
  - joint development with HP
  - targeting enterprise systems and not HPC
  - Tukwile (65nm) the last version
  - Intel QuickPath for interconnect (not a bus)
  - Enhanced memory subsystem, 4 cores
- Itanium 9500 (2012)
  - 32nm, 8 cores, up to 54 MB cache
  - Convergence towards Intel Xeon processors

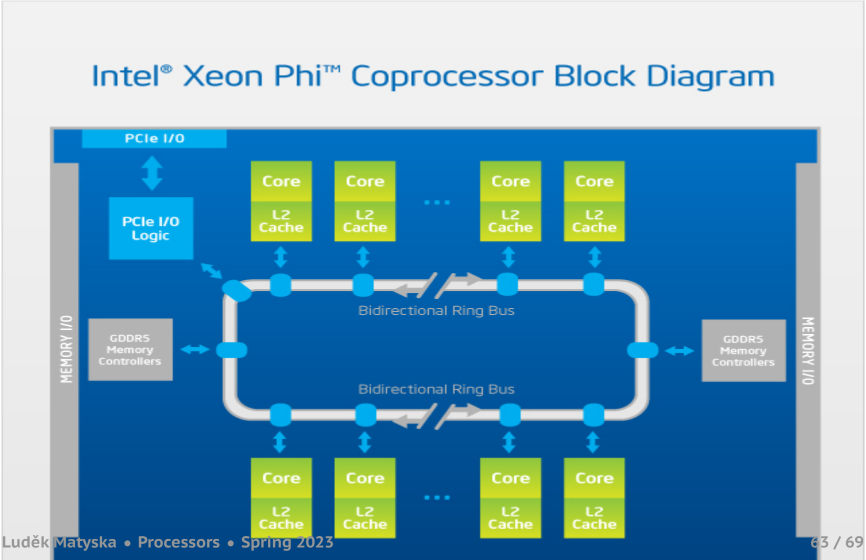
Contemporary x86-64 processors

- Originally introduced by AMD (K8 microarchitecture)
- Intel Core architecture
  - Nehalem, Sandy Bridge (32nm), Ivy Bridge (22nm), Haswell, Broadwell, Skylake and other “lake” families up to Comet and Ice lake
  - forthcoming Alder lake (12th generation, 10nm)
- AMD
  - Opteron, Athlon 64, ..., Ryzen/Epyc
- VIA Technologies
  - VIA Nano (2008)

Intel Core 10th gen

- Memory
  - L1 instruction/data cache: 32/48 KiB
  - L2 cache: 512 KiB
  - L3/smart cache: up to 20 MiB
  - 52 bit address space (4 PB), 57 bit virtual address space (up to 128 PB)
  - 352 TLB entries
- CPU
  - up to 10 cores
  - clock (turbo) rate up to 4,2 (5,3) GHz
  - hardware acceleration for SHA
  - AVX-512 instructions
  - Intel Deep Learning Boost
- GPU
  - 64 execution units
  - more than 1 TFLOPS

Xeon Phi


Intel® Xeon Phi™ Coprocessor Block Diagram
PCIe I/O
PCIe I/O Logic
Core
L2 Cache
...
Bidirectional Ring Bus
GDDR5 Memory Controllers
MEMORY I/O

IBM Power10 processor

- Emphasis for HPC (and energy efficiency)
- Memory
  - L1 instruction/data cache: 48/32 KiB
  - L2 cache: 2 MB
    L1 & L2 are per core
  - L3 cache: 120 MB per chip
  - 4096 TLB entries
  - memory RAM encryption with no latency penalty
- CPU
  - 15 cores, 8 threads per core (SMT8)
  - 512 entry instruction table (ANDES)
  - matrix math assist (SIMD code)
  - clock rate up to 4 GHz
- Up to 16 socket cluster with 240 cores (1920 threads) and 2 PB memory

POWER 10

POWER10 Processor Chip

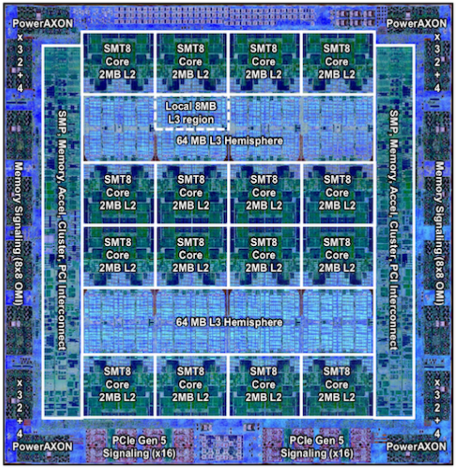
Technology and Packaging:
- 602mm^2 7nm Samsung (18B devices)
- 18 layer metal stack, enhanced device
- Single-chip or Dual-chip sockets

Computational Capabilities:
- Up to 15 SMT8 Cores (2 MB L2 Cache / core)
  (Up to 120 simultaneous hardware threads)
- Up to 120 MB L3 cache (low latency NUCA mgmt)
- 3x energy efficiency relative to POWER9
- Enterprise thread strength optimizations
- AI and security focused ISA additions
- 2x general, 4x matrix SIMD relative to POWER9
- EA-tagged L1 cache, 4x MMU relative to POWER9

Open Memory Interface:
- 16 x8 at up to 32 GT/s (1 TB/s)
- Technology agnostic support: near/main/storage tiers
- Minimal (< 10ns latency) add vs DDR direct attach

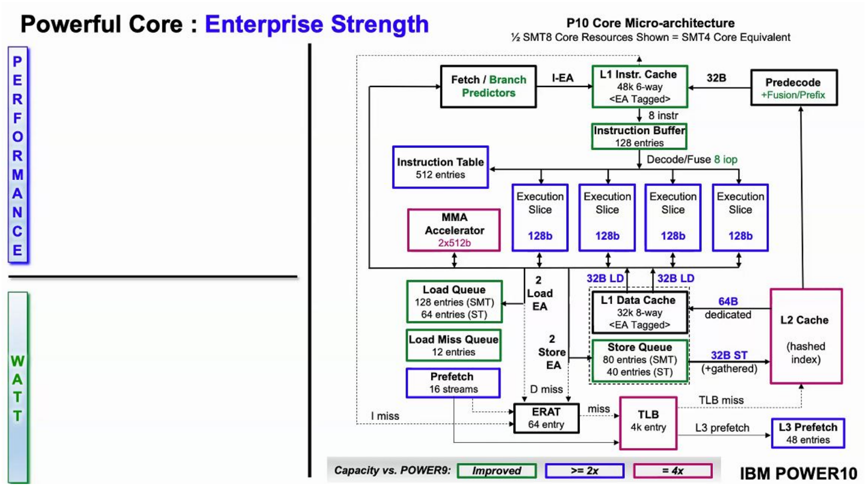
PowerAXON Interface:
- 16 x8 at up to 32 GT/s (1 TB/s)
- SMP interconnect for up to 16 sockets
- OpenCAPI attach for memory, accelerators, I/O
- Integrated clustering (memory semantics)

PCIe Gen 5 Interface:
- x64 / DCM at up to 32 GT/s



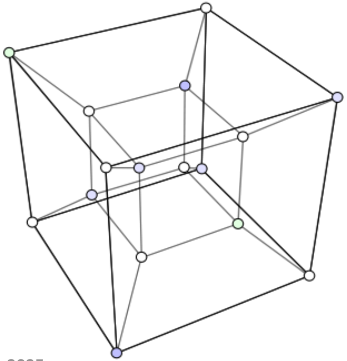
Die Photo courtesy of Samsung Foundry

POWER 10


Powerful Core : Enterprise Strength
PERFORMANCE
WATT
P10 Core Micro-architecture
½ SMT8 Core Resources Shown = SMT4 Core Equivalent
Fetch / Branch Predictors
I-EA
L1 Instr. Cache 48k 6-way <EA Tagged>
32B
Predecode +Fusion/Prefix
8 instr
Instruction Buffer 128 entries
Instruction Table 512 entries
Decode/Fuse 8 iop
Execution Slice 128b
Execution Slice 128b
Execution Slice 128b
Execution Slice 128b
MMA Accelerator 2x512b
32B LD
32B LD
2 Load EA
Load Queue 128 entries (SMT) 64 entries (ST)
Load Miss Queue 12 entries
Prefetch 16 streams
2 Store EA
L1 Data Cache 32k 8-way <EA Tagged>
64B dedicated
Store Queue 80 entries (SMT) 40 entries (ST)
32B ST (+gathered)
L2 Cache (hashed index)
D miss
I miss
ERAT 64 entry
miss
TLB 4k entry
TLB miss
L3 prefetch
L3 Prefetch 48 entries
Capacity vs. POWER9:
Improved
>= 2x
= 4x
IBM POWER10

Multiprocessor systems

- Problems with further frequency increase
  - Heat dissipation
- Parallelization
  - Performance increase via multiple cores
  - Performance increase via multiple CPUs (sockets)



Multiprocessor systems

- Scaling ratio (number of sockets) for symmetric memory
    - 2-8 for AMD and Intel, 16-32 for IBM POWER family
    - Special solutions up to hundreds (SGI, now HPE)
- Distributed memory
    - Centralized (symmetric) memory a bottleneck
    - NUMA (Non-Uniform Memory Architecture)
- Clusters with huge number of multiprocessor nodes
    - Symmetric memory within a node
    - Distributed memory across nodes

Multiprocessor systems

- Cache coherency
  - I see what I wrote
  - I see what anyone write before me
  - Order of writes is globally identical
- Cache row state
  - uncached, shared, modified, …
- Cache coherency protocols