CPU Hardware Architecture and Performance Optimization

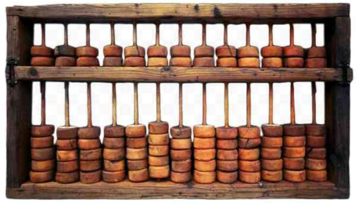
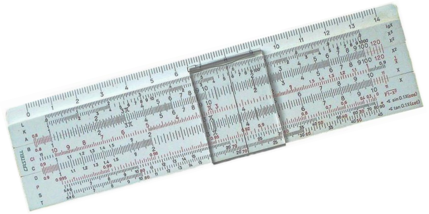
G. Amadio (CERN)

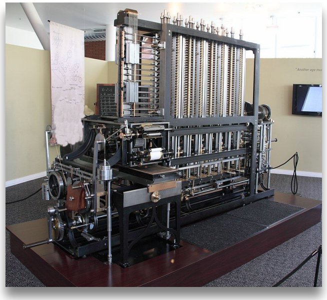
Our Main Goals

- Understand the architecture of modern CPU hardware
    - Hardware evolution
    - Main features of modern hardware
- Understand how to analyze the performance of our code
    - How to identify performance bottlenecks
    - What to measure and how to measure it
- Combine architectural knowledge and performance analysis
    - How to interpret performance measurements
    - What changes to make to the software

CPU Hardware Architecture and Evolution

Early Computing Devices



2700 – 2300 BC	1620– 1630	1642	1820s
Abacus	**Slide Rule**	**Pascaline**	**Difference Engine**
Used since ancient times, until Arabic numerals became the norm. Still in use as an educational tool.	Uses logarithm scales to help with multiplications and also computing other functions. Extensively used by engineers in the last century, before computers became powerful.	Mechanical calculator invented by Blaise Pascal to help his father with tax calculations. Could add and subtract.	Automatic mechanical calculator designed to tabulate polynomial functions. Designed and first created by Charles Babbage.

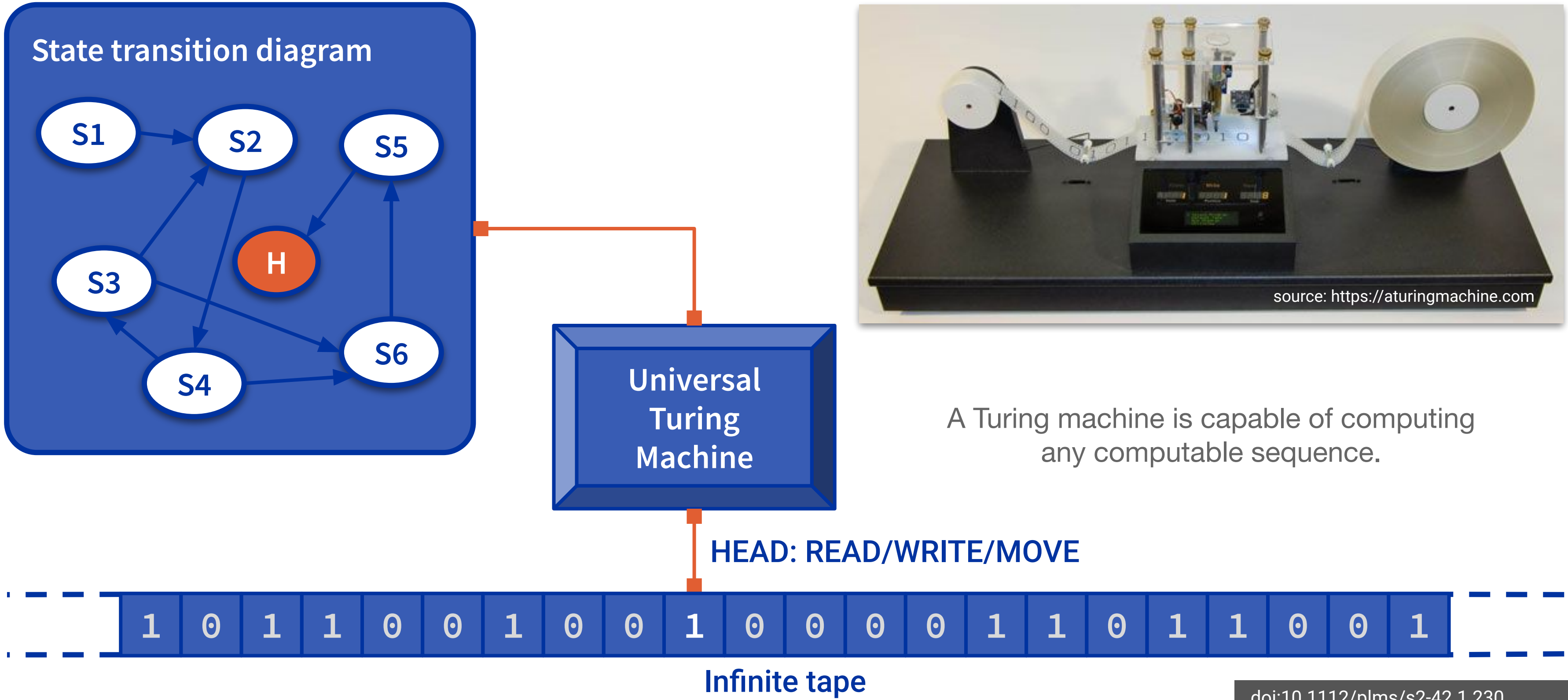
Images: Wikipedia

Ada Lovelace, the first computer programmer

Augusta Ada King, Countess of Lovelace (10 December 1815 – 27 November 1852) was an English mathematician and writer, known for her work on Charles Babbage's proposed mechanical general-purpose computer, the Analytical Engine. She was the first to recognize that the new machine had applications beyond simple calculations. She arguably wrote the first “computer program”. In her article entitled “note G” on the Analytical Engine, she described in detail an algorithm to compute a sequence of Bernoulli numbers using it.

Source: Wikipedia

The Turing Machine: concept of first generic computer



A Turing machine is capable of computing any computable sequence.

doi:10.1112/plms/s2-42.1.230

From Turing Machine to Stored-Program Computer

1936

Turing Machine

Conceptually the first general computing machine.

1937

Harvard IBM Mark I

Inspired on the Analytical Engine. One of the earliest general-purpose electromechanical computers. First computer bug discovered on it by Grace Hopper.

1942-1945

ABC, Colossus, ENIAC

First truly digital computers, based on boolean logic and vacuum tubes.

1947

Solid-State Transistor

The first solid-state transistor was a based on a point-contact connection to a crystal by closely spaced thin gold foils.

1949

Assembly Language

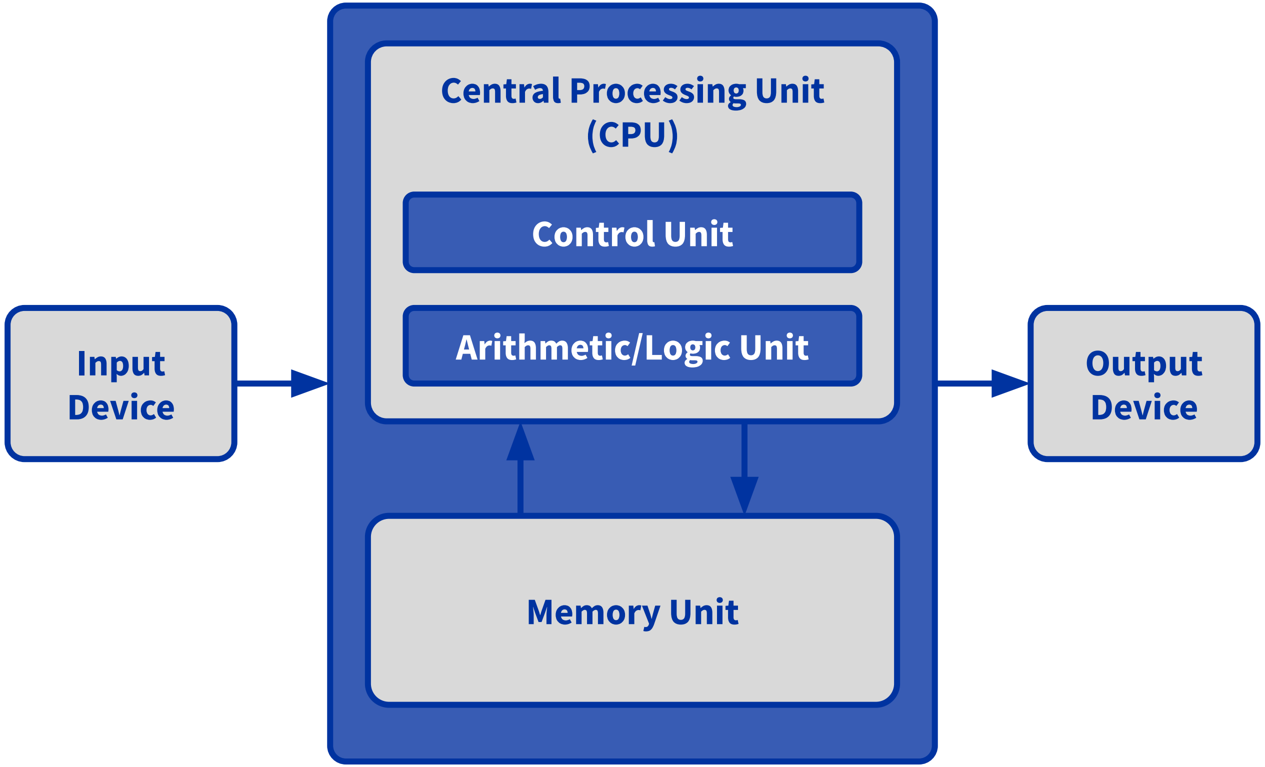
Beginning of standardization of how to program computer with abstract instruction sets.

1950

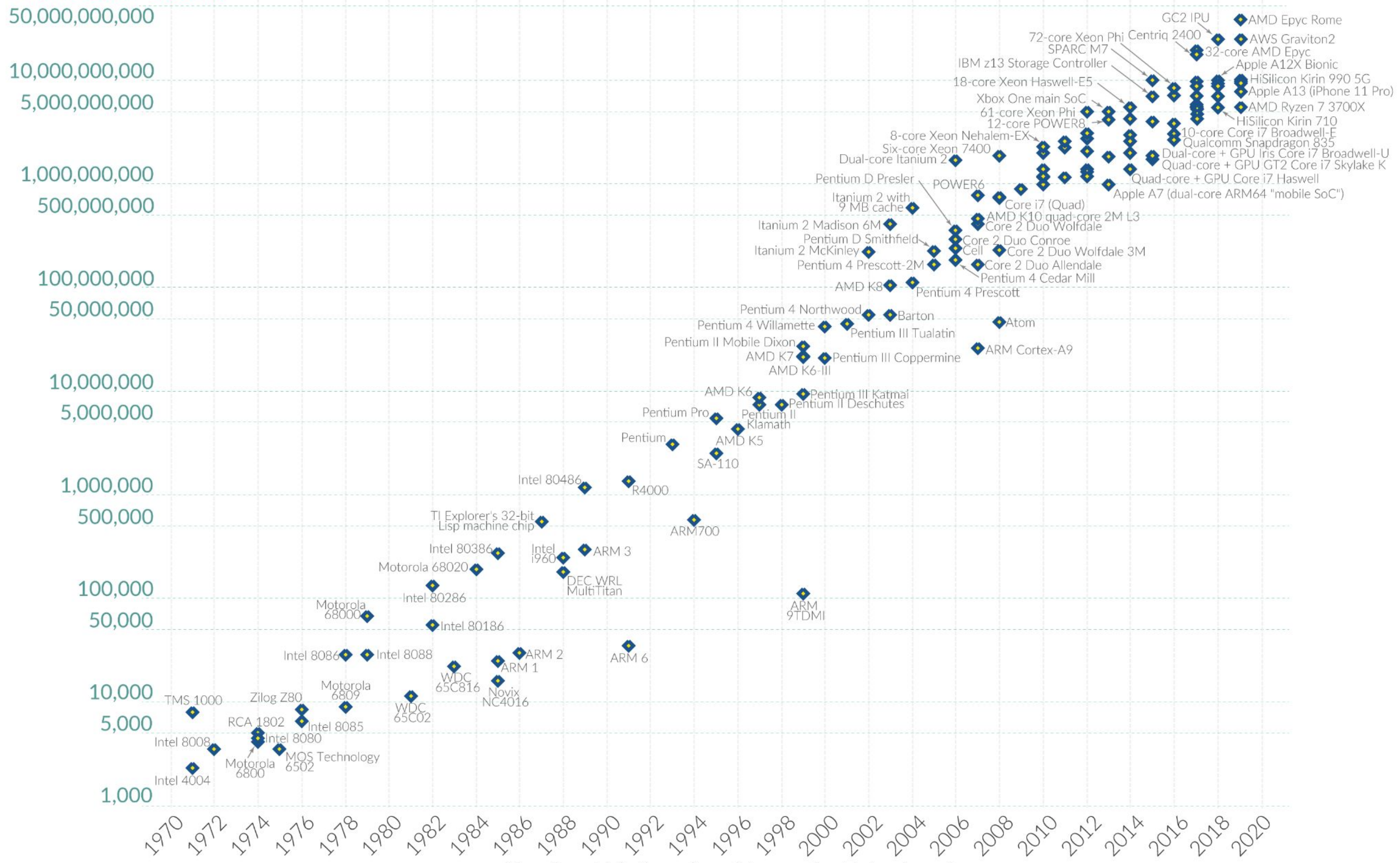
EDVAC

First stored-program computer, based on John von Neumann’s architecture concept from 1945.

John von Neumann Architecture



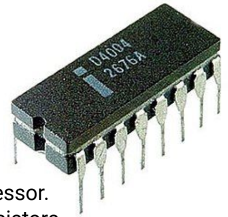
Transistor count



Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

OurWorldinData.org – Research and data to make progress against the world's largest problems.

Integrated Circuit-Based Microprocessors



Intel 4004

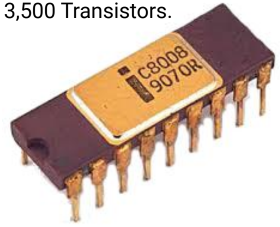
First Intel microprocessor. 2,300 Transistors.

1970

1972

Intel 8008

First 8 bit microprocessor. 3,500 Transistors.



MOS 6502

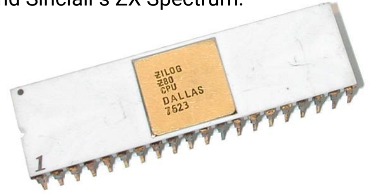
Powered many popular devices, such as the Apple II, Atari 2600, Commodore 64, and the NES. 3,510 Transistors.

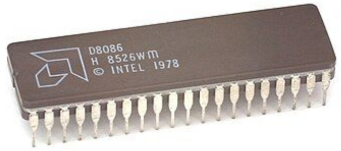
1975

1976

Zilog Z80

8-bit microprocessor.
Powered devices such as
Sega Master System and Mega Drive,
and Sinclair's ZX Spectrum.





Intel 8086

First 16-bit microprocessor. 29,000 Transistors. Its successor, the Intel 8088, a slightly modified version, powered first IBM PC.

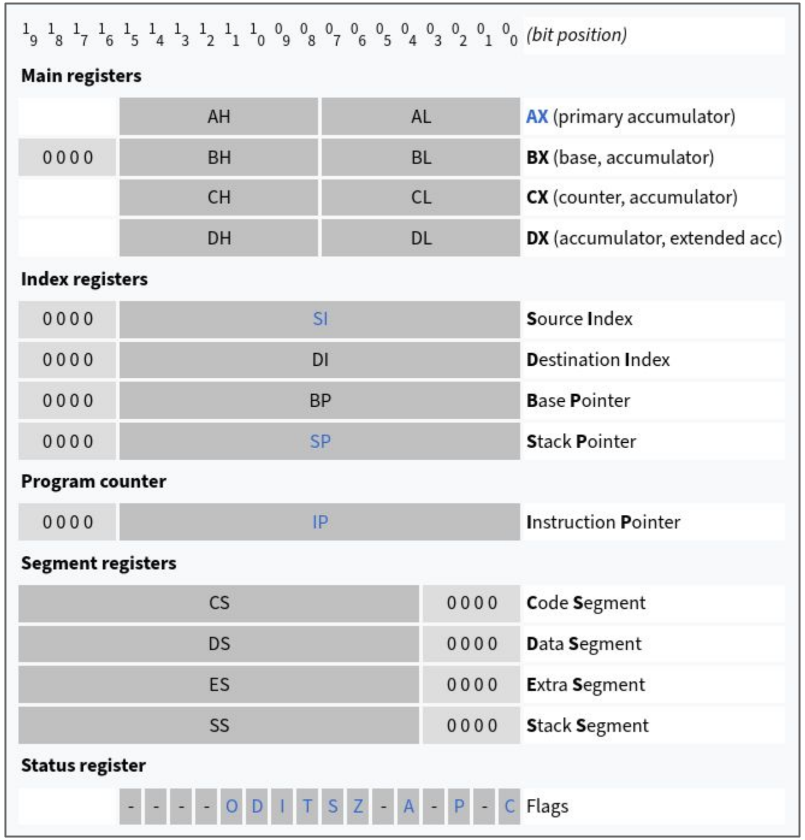
1978

1985

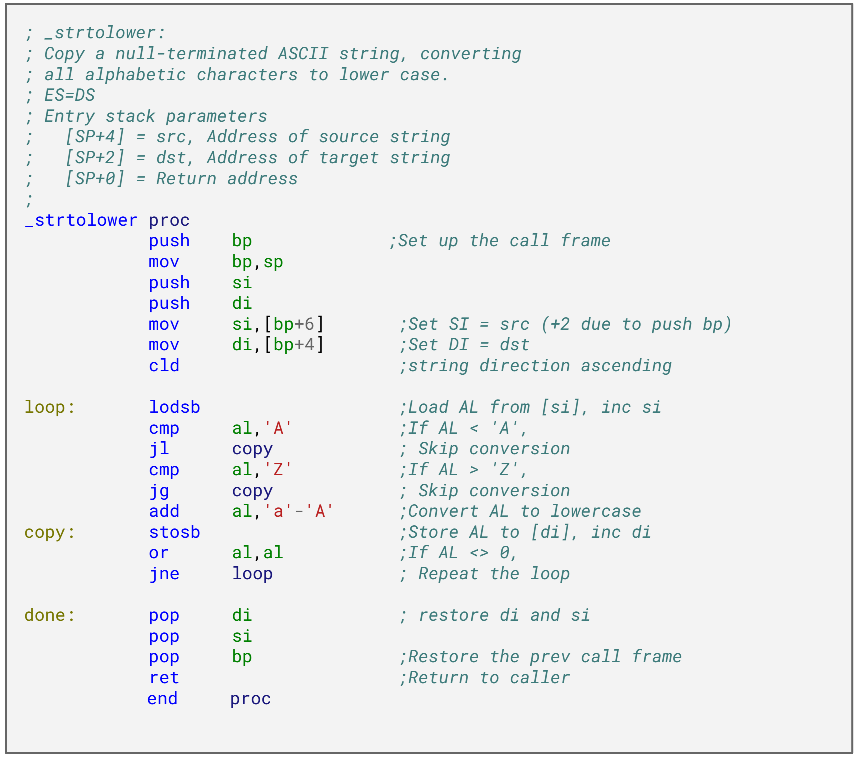
Intel 80386

32-bit microprocessor.
275,000 Transistors, 33MHz.
Cemented Intel's PC market dominance.

Intel’s 8086 Registers and Assembly



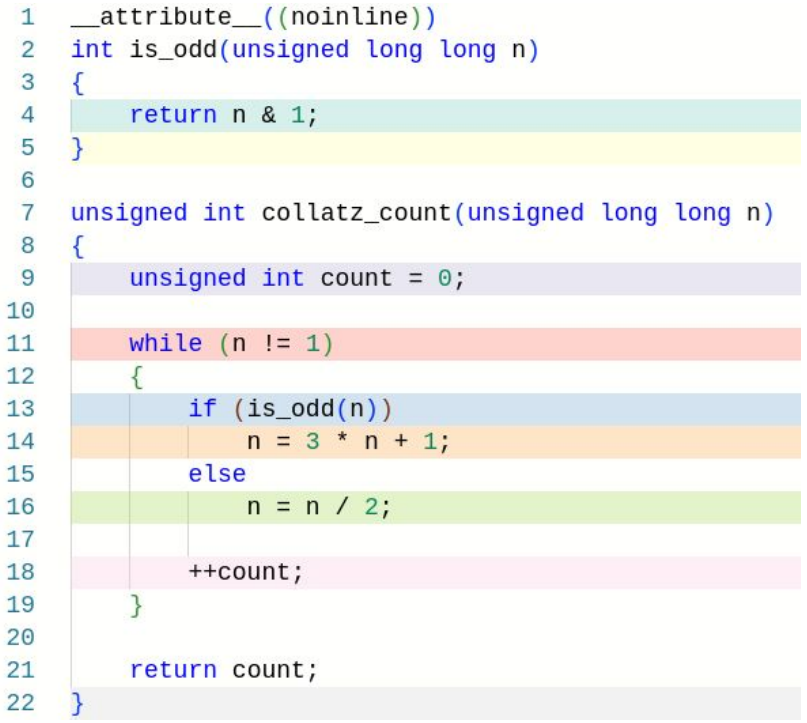
19 18 17 16	15 14 13 12 11 10 09 08	07 06 05 04 03 02 01 00	*(bit position)*
Main registers			
	AH	AL	**AX** (primary accumulator)
0000	BH	BL	**BX** (base, accumulator)
	CH	CL	**CX** (counter, accumulator)
	DH	DL	**DX** (accumulator, extended acc)
Index registers			
0000	SI	SI	**S**ource **I**ndex
0000	DI	DI	**D**estination **I**ndex
0000	BP	BP	**B**ase **P**ointer
0000	SP	SP	**S**tack **P**ointer
Program counter			
0000	IP	IP	**I**nstruction **P**ointer
Segment registers			
CS	CS	CS 0000	**C**ode **S**egment
DS	DS	DS 0000	**D**ata **S**egment
ES	ES	ES 0000	**E**xtra **S**egment
SS	SS	SS 0000	**S**tack **S**egment
Status register			
	- - - - O D I T	S Z - A - P - C	Flags



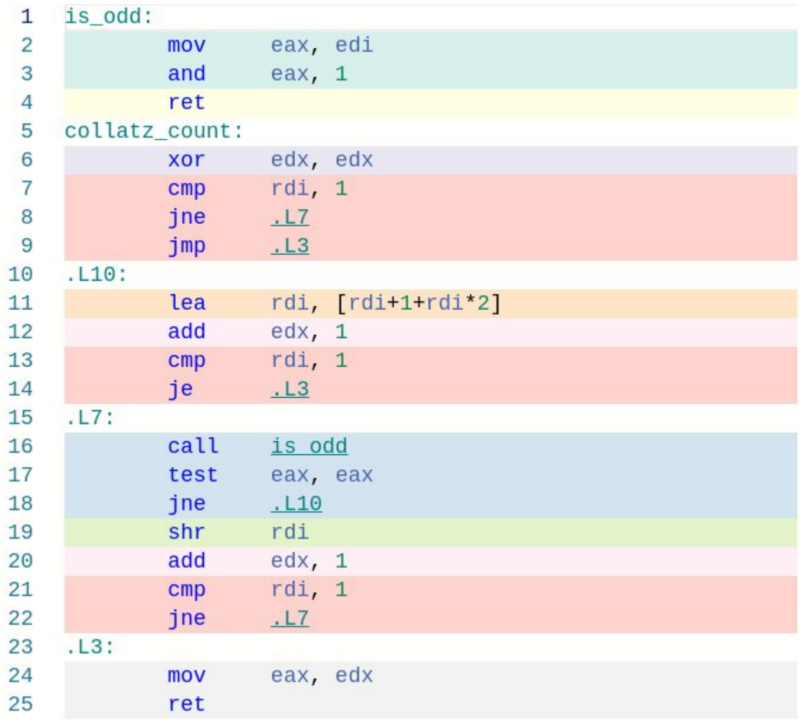
```
; _strtolower:
; Copy a null-terminated ASCII string, converting
; all alphabetic characters to lower case.
; ES=DS
; Entry stack parameters
;   [SP+4] = src, Address of source string
;   [SP+2] = dst, Address of target string
;   [SP+0] = Return address
;
_strtolower proc
            push    bp              ;Set up the call frame
            mov     bp,sp
            push    si
            push    di
            mov     si,[bp+6]        ;Set SI = src (+2 due to push bp)
            mov     di,[bp+4]        ;Set DI = dst
            cld                      ;string direction ascending

loop:       lodsb                    ;Load AL from [si], inc si
            cmp     al,'A'           ;If AL < 'A',
            jl      copy             ; Skip conversion
            cmp     al,'Z'           ;If AL > 'Z',
            jg      copy             ; Skip conversion
            add     al,'a'-'A'       ;Convert AL to lowercase
copy:       stosb                    ;Store AL to [di], inc di
            or      al,al            ;If AL <> 0,
            jne     loop             ; Repeat the loop

done:       pop     di               ; restore di and si
            pop     si
            pop     bp               ;Restore the prev call frame
            ret                      ;Return to caller
            end     proc
```

Source: Wikipedia

Intel x86 Assembly

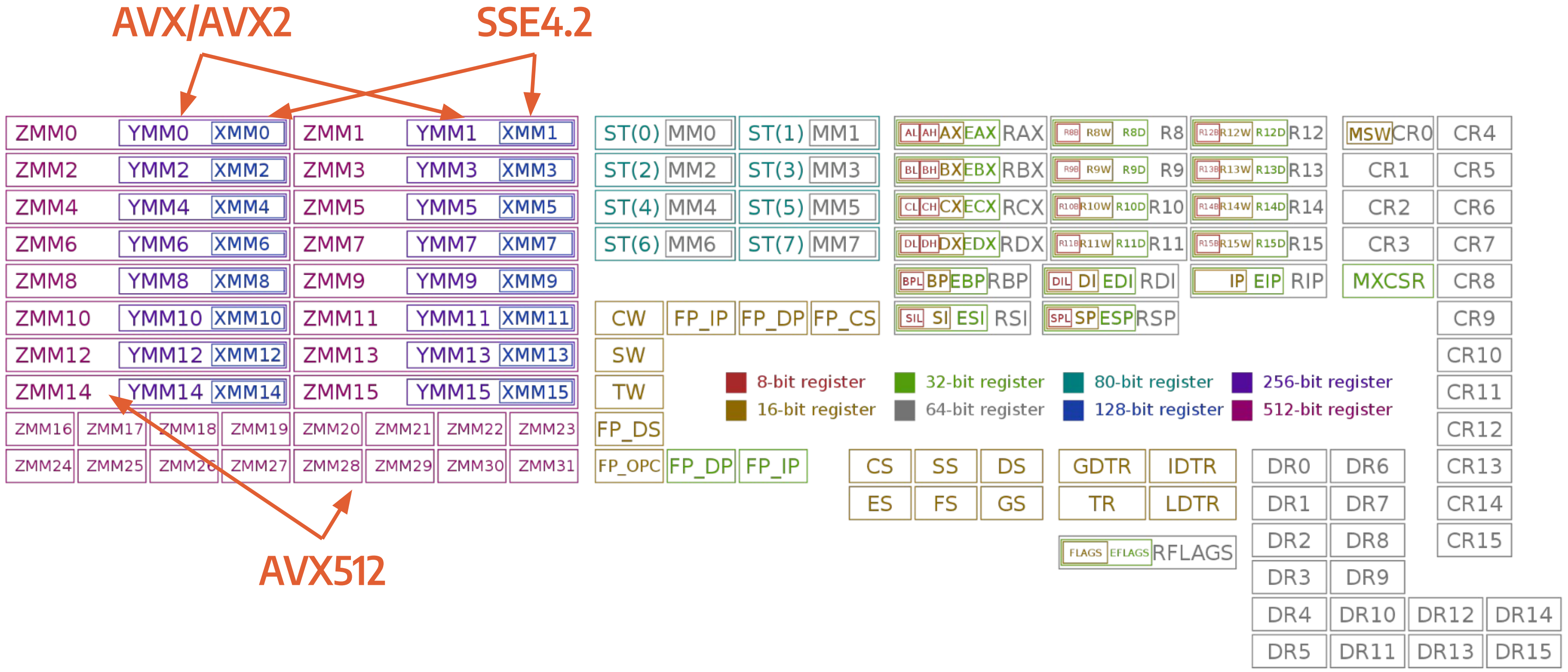


```c
__attribute__((noinline))
int is_odd(unsigned long long n)
{
    return n & 1;
}

unsigned int collatz_count(unsigned long long n)
{
    unsigned int count = 0;

    while (n != 1)
    {
        if (is_odd(n))
            n = 3 * n + 1;
        else
            n = n / 2;

        ++count;
    }

    return count;
}
```



```asm
is_odd:
        mov     eax, edi
        and     eax, 1
        ret
collatz_count:
        xor     edx, edx
        cmp     rdi, 1
        jne     .L7
        jmp     .L3
.L10:
        lea     rdi, [rdi+1+rdi*2]
        add     edx, 1
        cmp     rdi, 1
        je      .L3
.L7:
        call    is_odd
        test    eax, eax
        jne     .L10
        shr     rdi
        add     edx, 1
        cmp     rdi, 1
        jne     .L7
.L3:
        mov     eax, edx
        ret
```

Registers available in the x86-64 instruction set



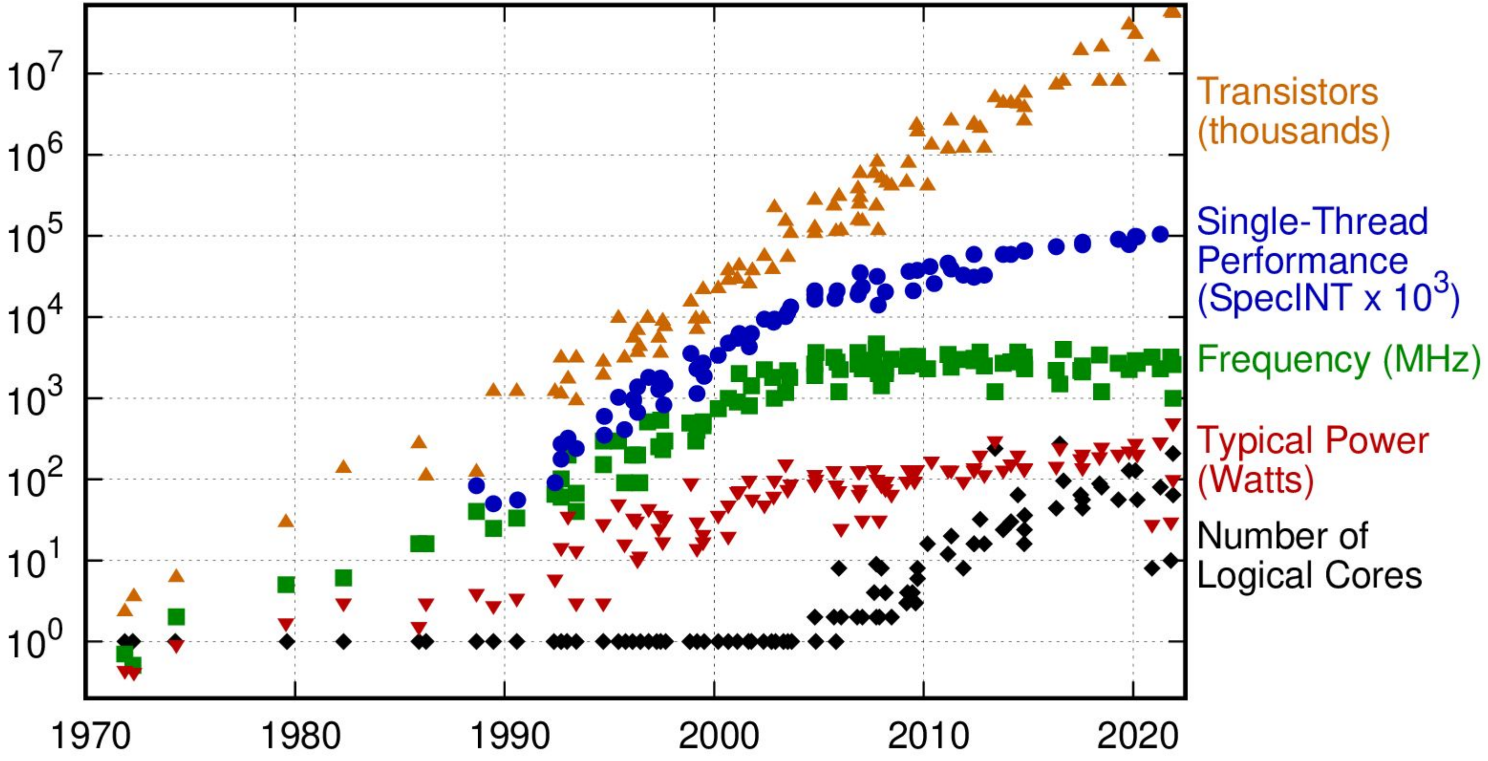
Instruction Sets

- **CISC** (Complex Instruction Set Computer)
  - Intel x86 and AMD64
    - Most laptop and desktop PCs, Playstation 5, Xbox One
  - IBM System z (mainframe computers)
- **RISC** (Reduced Instruction Set Computer)
  - ARM
    - Amazon Graviton (AWS VMs)
    - Apple M1–M4 (iPhone, iPad, iMacs)
    - Ampere Altra, Fujitsu A64FX, etc
    - Qualcomm (mobile phones, tablets)
    - Nintendo Game Boy Advance, DS, 3DS and Switch, Raspberry Pi, etc
  - IBM's PowerPC
    - Apple Macintosh (1994–2005), Nintendo GameCube and Wii, Playstation 3, Xbox 360
  - DEC Alpha, MIPS, Motorola 68000, RISC-V, SPARC, SuperH
    - Apple II (M68k), Nintendo 64, PlayStation 1 and 2 (MIPS), Sega Saturn and Dreamcast (SuperH)

Programming Language Evolution

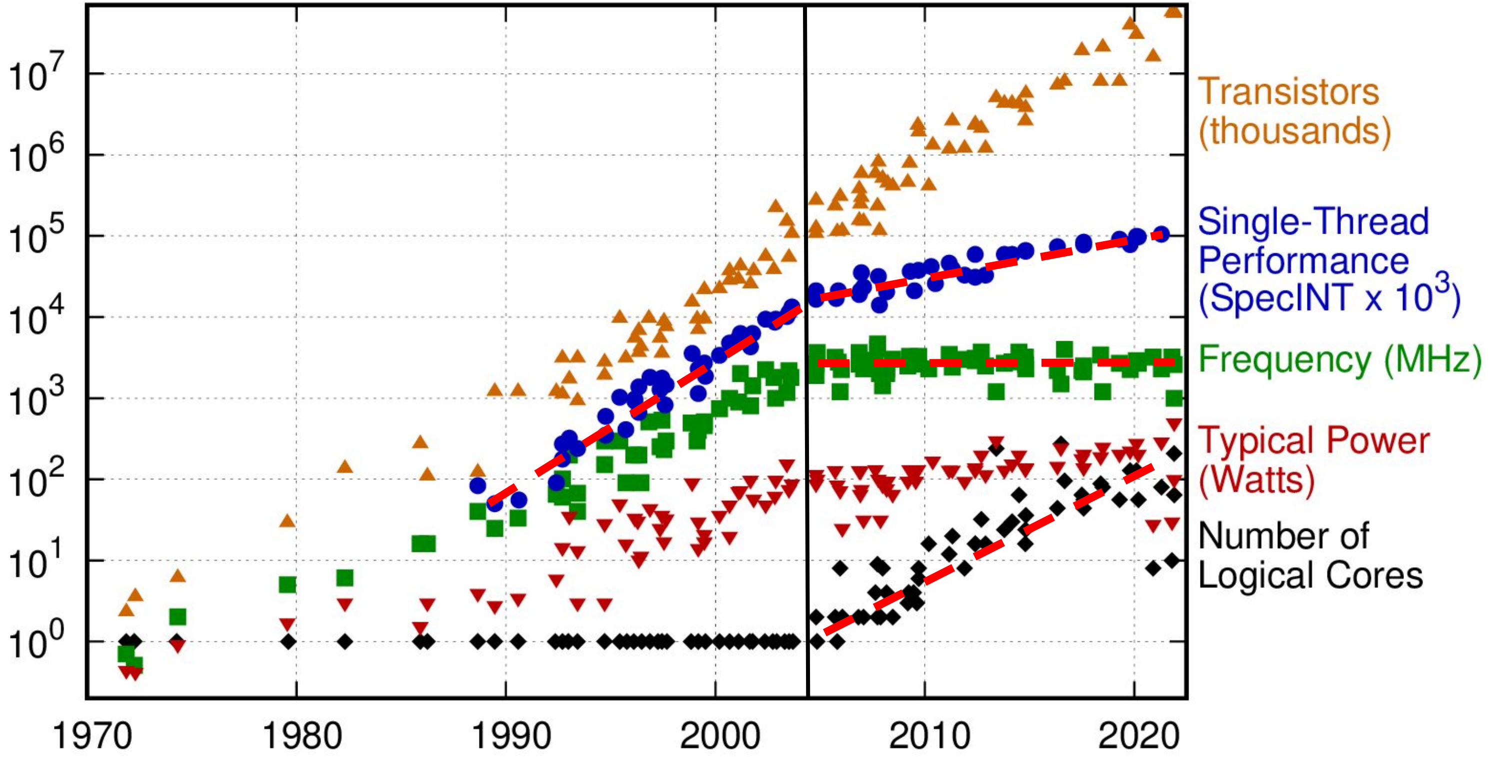
1947 Assembly	1954 Fortran	1963 BASIC	1972 C	1979 C++
Low-level language. High correspondence between language and hardware instructions.	One of the earliest high-level imperative programming languages.	**B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode is a family of high-level languages.	Originally developed to implement many of the utilities for UNIX OSs. Still in wide use today.	C++ was designed with systems programming, embedded software, and efficiency in mind.
Code written in assembly is converted to machine code using an *assembler*, which was a big upgrade over previous forms of programming.	Introduced procedural programming, double precision, and complex numbers. Still popular in HPC, including in GPU application programming.	BASIC became popular during the 8-bit era, but declined in popularity in the 90s, when more advanced languages like C were the norm.	C is a portable, imperative language with a static type system and which supports structured programming.	Although many think of C++ as a superset of C or C with classes, their latest versions are not fully compatible. Used extensively in HEP and HPC nowadays.

50 Years of Microprocessor Trend Data



source: https://github.com/karlrupp/microprocessor-trend-data

50 Years of Microprocessor Trend Data

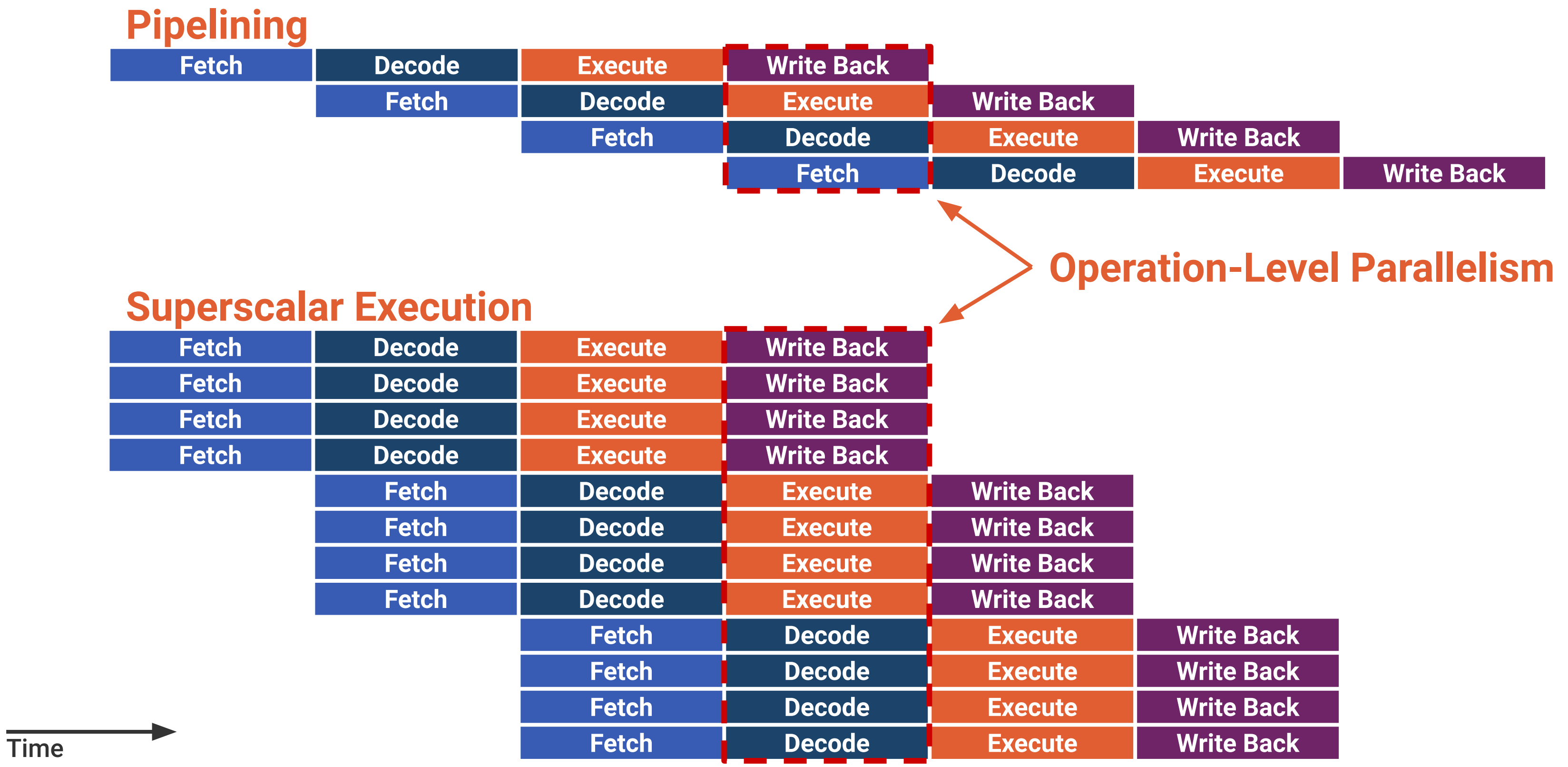


source: https://github.com/karlrupp/microprocessor-trend-data

Breakdown of Dennard’s Scaling

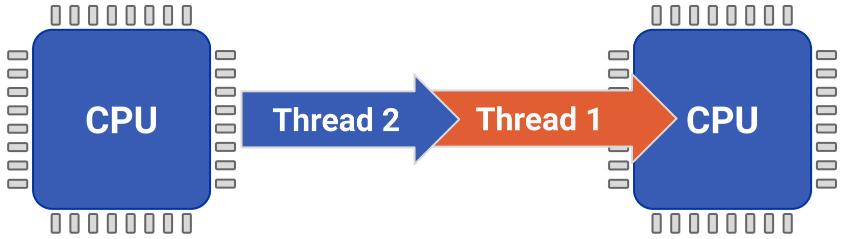
- Power density per unit area stopped decreasing
- Frequency could no longer keep increasing after each die shrink
  - But the transistor numbers kept growing
- Single-thread performance gains continued, albeit at a slower pace
  - More complexity: pipelining, superscalar, out-of-order execution, SIMD
- AMD and Intel bring 64-bit CPUs to the mainstream market
  - Intel with IA-64, and AMD with amd64 (x86_64), announced in 1999
- From symmetric multiprocessing (SMP) to multithreading (SMT)
  - In the ‘90s, dual socket high-end servers became popular
  - First SMT capable CPU was the Intel Pentium 4, released in 2002
- First dual core processors began to appear in mid 2000s
- The era of parallelism is born

Instruction-Level Parallelism

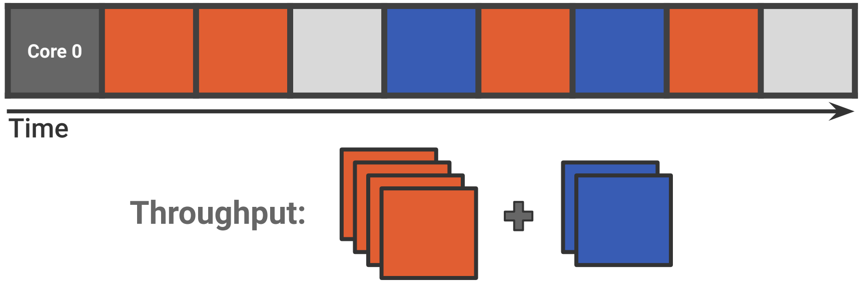


Symmetric multithreading (SMT)

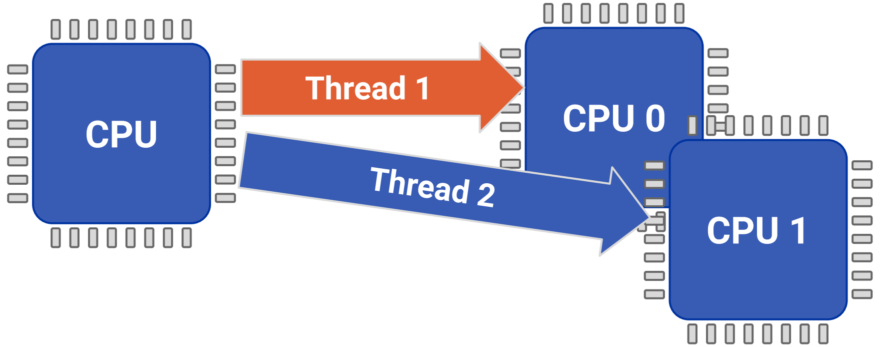
Without SMT



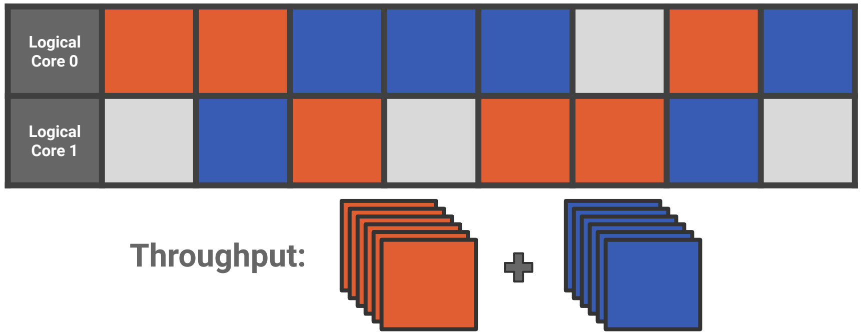
Threads scheduled one at a time on each physical core



With SMT



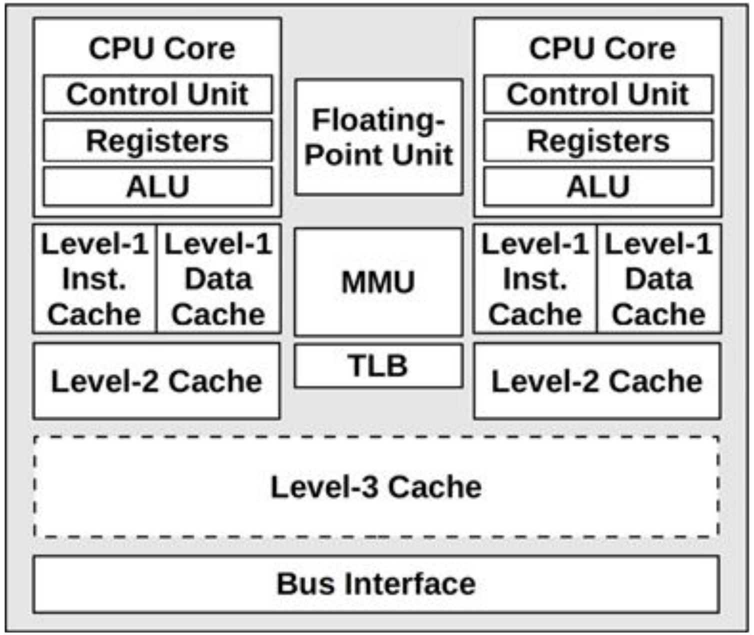
Threads run simultaneously on two logical cores



CPU Architecture

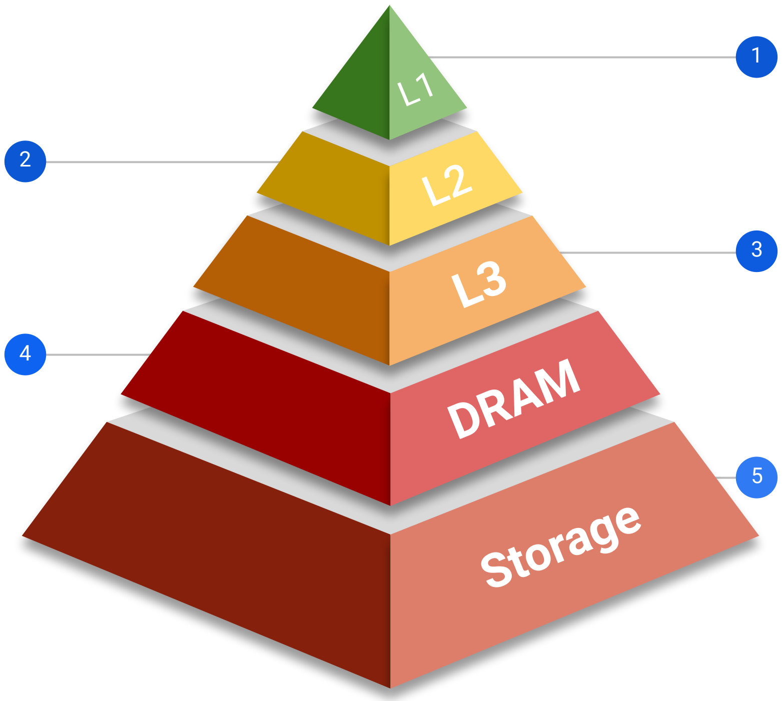
- Logical Components
  - Control Unit
  - Arithmetic Logic Units (ALUs)
  - Floating Point Unit (FPU)
  - Branch Predictor Unit (BPU)
  - Memory Management Unit (MMU)
  - Translation Lookaside Buffer (TLB)
- Memory Subsystem
  - L1(~32–512KB per core)
    - L1 Instruction Cache
    - L1 Data Cache
  - L2 (~1–8MB per core)
    - Instruction/Data Shared Cache
  - L3 (up to ~8MB–1.1GB per socket)
    - Last level cache (LLC)

Generic Dual Core CPU



Source: Systems Performance 2nd Edition, Brendan Gregg

Memory Hierarchy



1 **CPU Registers & L1 Cache**

L1 Latency: 1–5 cycles
L1 Size: 32–512K / core

2 **L2 Cache**

Latency: ~12 cycles
Size: ~1M / core

3 **L3 Cache**

Latency: ~40 cycles
Size: 8MB – 1.1GB

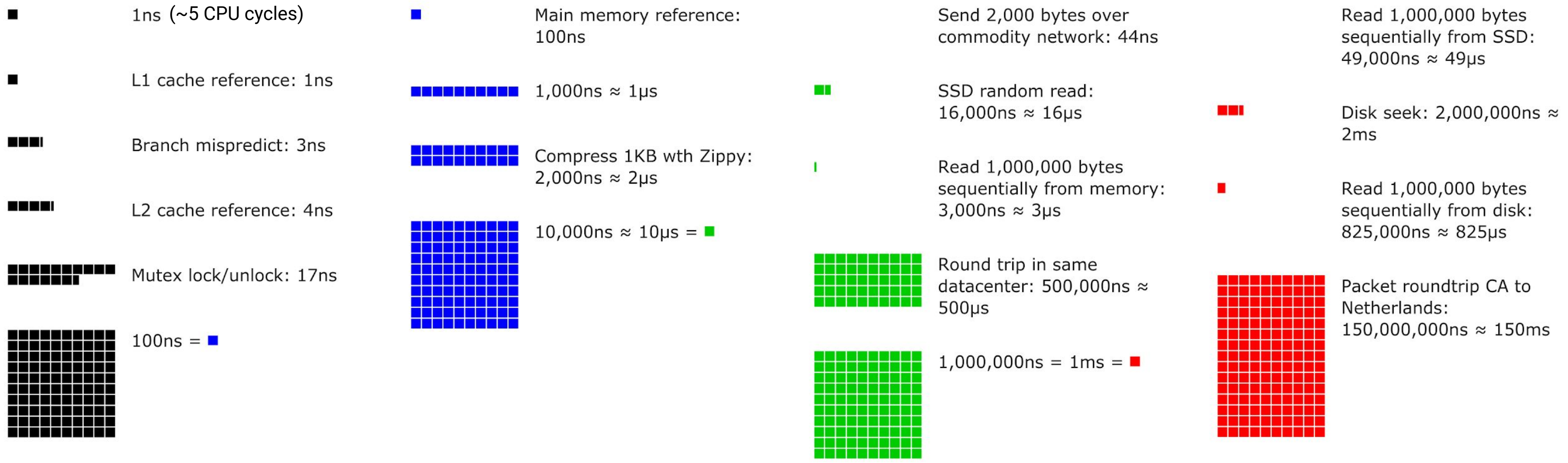
4 **Main Memory**

Latency: 100–400 cycles
Size: 10s of GB to few TB

5 **SSD, HDD, Tape Storage**

SSD Latency: 10–70 μs
HDD Latency: 10–15 ms
Tape Latency: varies, minutes to hours, depends on if tape is already mounted

Latency Numbers Every Programmer Should Know



1ns (~5 CPU cycles)

L1 cache reference: 1ns

Branch mispredict: 3ns

L2 cache reference: 4ns

Mutex lock/unlock: 17ns

100ns = 🟦

Main memory reference: 100ns

1,000ns ≈ 1μs

Compress 1KB wth Zippy: 2,000ns ≈ 2μs

10,000ns ≈ 10μs = 🟩

Send 2,000 bytes over commodity network: 44ns

SSD random read: 16,000ns ≈ 16μs

Read 1,000,000 bytes sequentially from memory: 3,000ns ≈ 3μs

Round trip in same datacenter: 500,000ns ≈ 500μs

1,000,000ns = 1ms = 🟥

Read 1,000,000 bytes sequentially from SSD: 49,000ns ≈ 49μs

Disk seek: 2,000,000ns ≈ 2ms

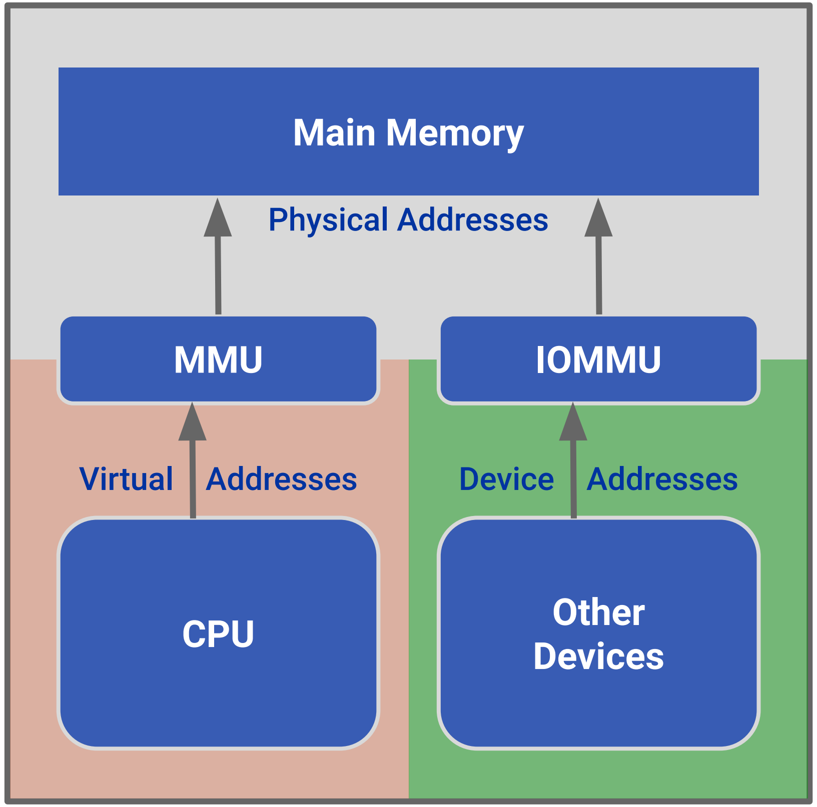
Read 1,000,000 bytes sequentially from disk: 825,000ns ≈ 825μs

Packet roundtrip CA to Netherlands: 150,000,000ns ≈ 150ms

Source: https://colin-scott.github.io/personal_website/research/interactive_latency.html

Virtual Memory

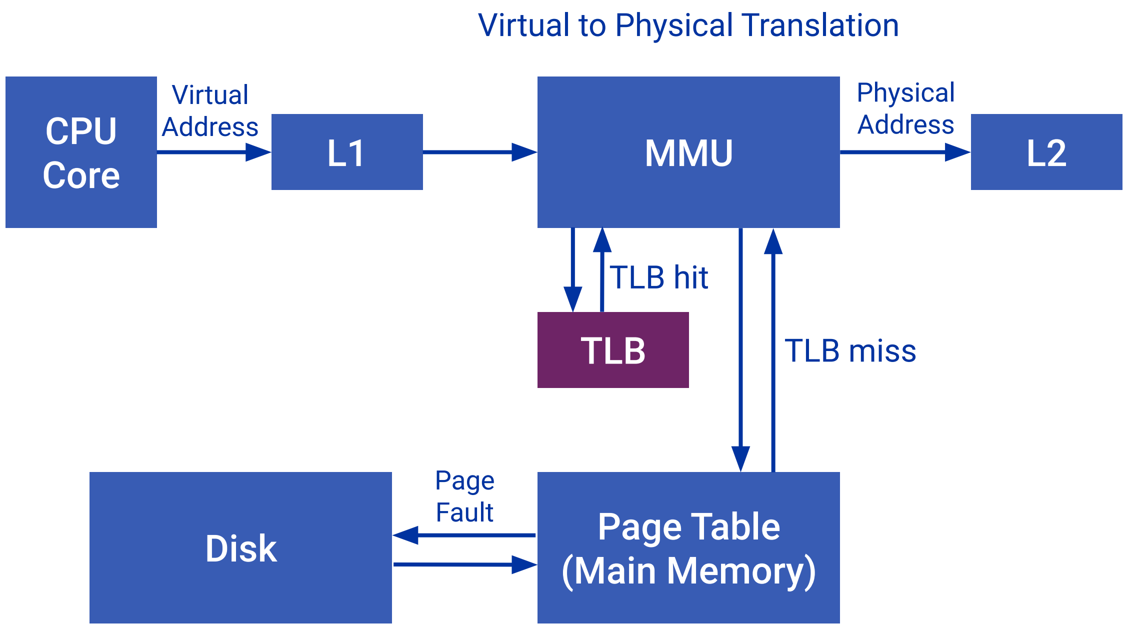
- First appeared in the Atlas computer in 1962
- Memory Management Unit (MMU)
- Memory managed in pages
  - Page sizes are usually 4K, 16K, 64K
  - May also support “huge pages” of 2MB, 1GB
- Hides fragmentation of physical memory
- Memory hierarchy managed by the kernel
- Makes application programming easier
  - Memory looks contiguous
  - No need to worry about fragmentation
  - Seems to own whole address space
  - Enabled timesharing features



The Translation Lookaside Buffer

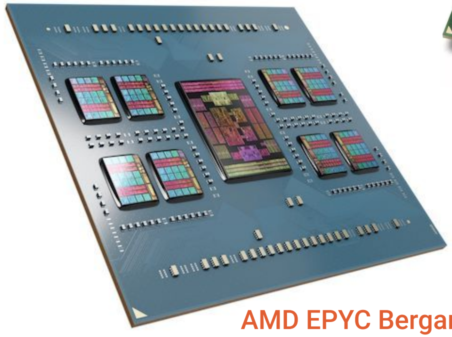
“A translation lookaside buffer (TLB) is a memory cache that is used to reduce the time taken to access a user memory location.
It is a part of the chip's memory management unit (MMU). The TLB stores the recent translations of virtual memory to physical memory and can be called an address-translation cache.”

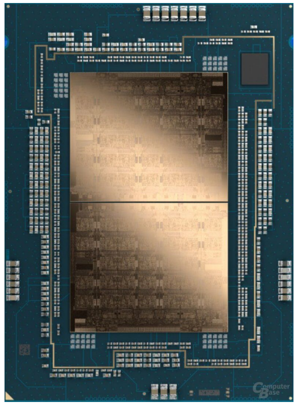
https://en.wikipedia.org/wiki/Translation_lookaside_buffer

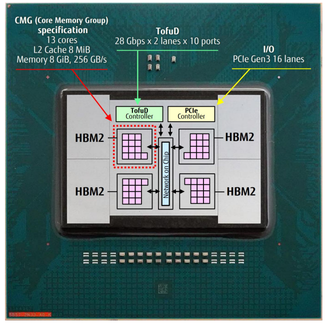


The Parallel Era

- First dual core CPUs debut in 2004
  - Pentium D, based on Pentium 4
  - AMD Athlon X2
- Quickly evolved from 2 to 4 cores
  - Stagnated at 4 cores for several years
- Ryzen brought AMD back in the game
  - Offered more cores, forced Intel to do the same
- ARM finally begins move from phones to servers
  - Amazon Graviton, Fujitsu A64FX, Ampere Altra
- Innovations in packaging led to multi-chip CPUs
  - AMD EPYC, Intel Sapphire and Emerald Rapids


AMD EPYC Bergamo


Intel Emerald Rapids



Fujitsu A64FX

Ampere Roadmap 2020 – 2026

2020	2021	2022	2023	2024	2025	2026

Ampere® Altra® Family

Up to 80 Cores
7nm

Up to 128 Cores
7nm

AmpereOne® Family

Up to 192 Cores
5nm

Up to 256 Cores
3nm

8-Channel DDR4

8-Channel DDR5

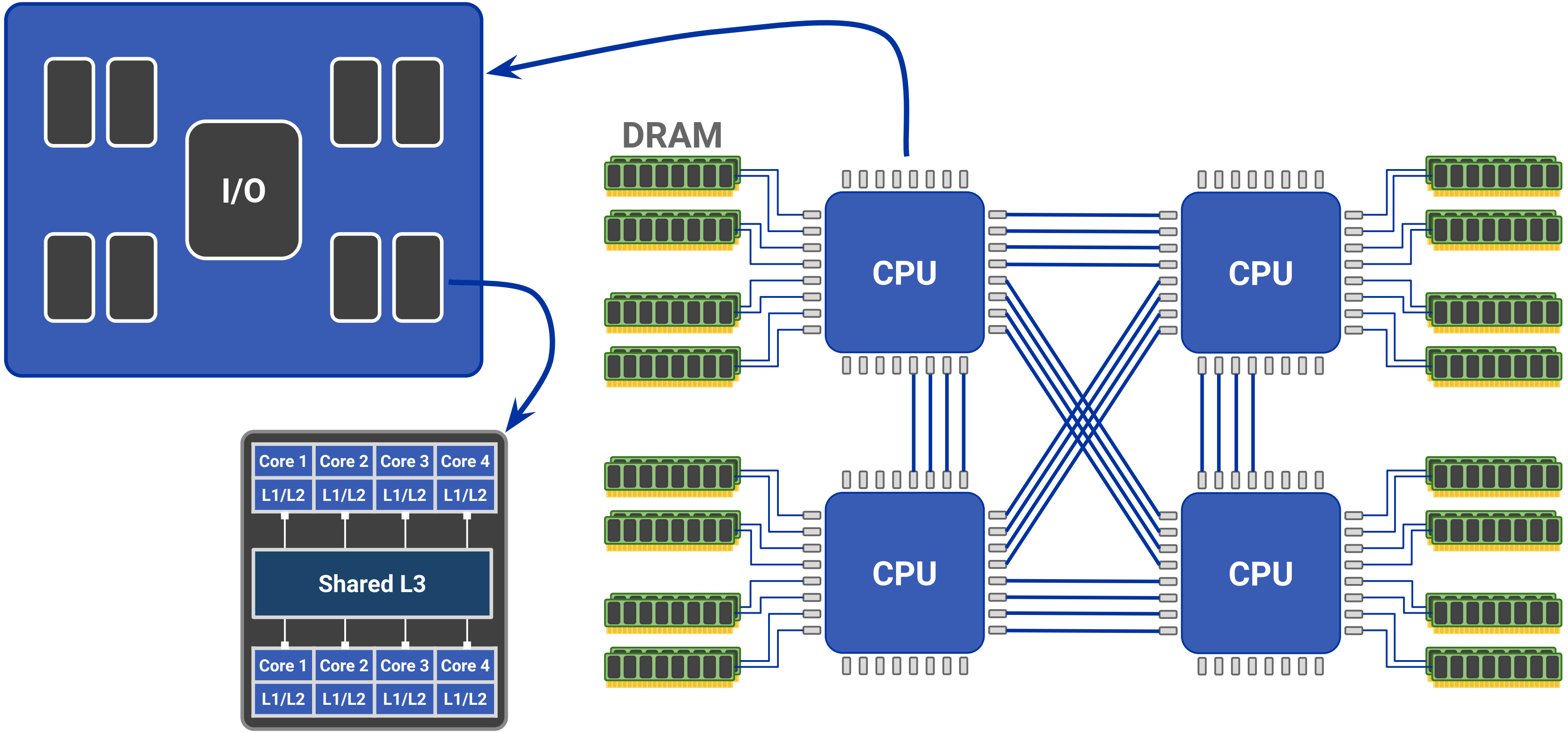
12-Channel DDR5

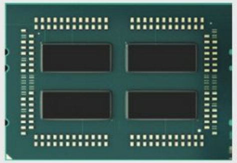
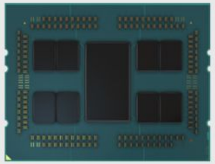
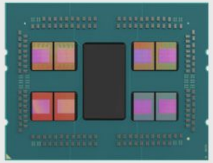
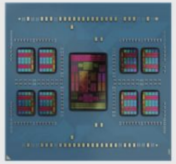
Generally Available

Ready at Fab

Source: Ampere

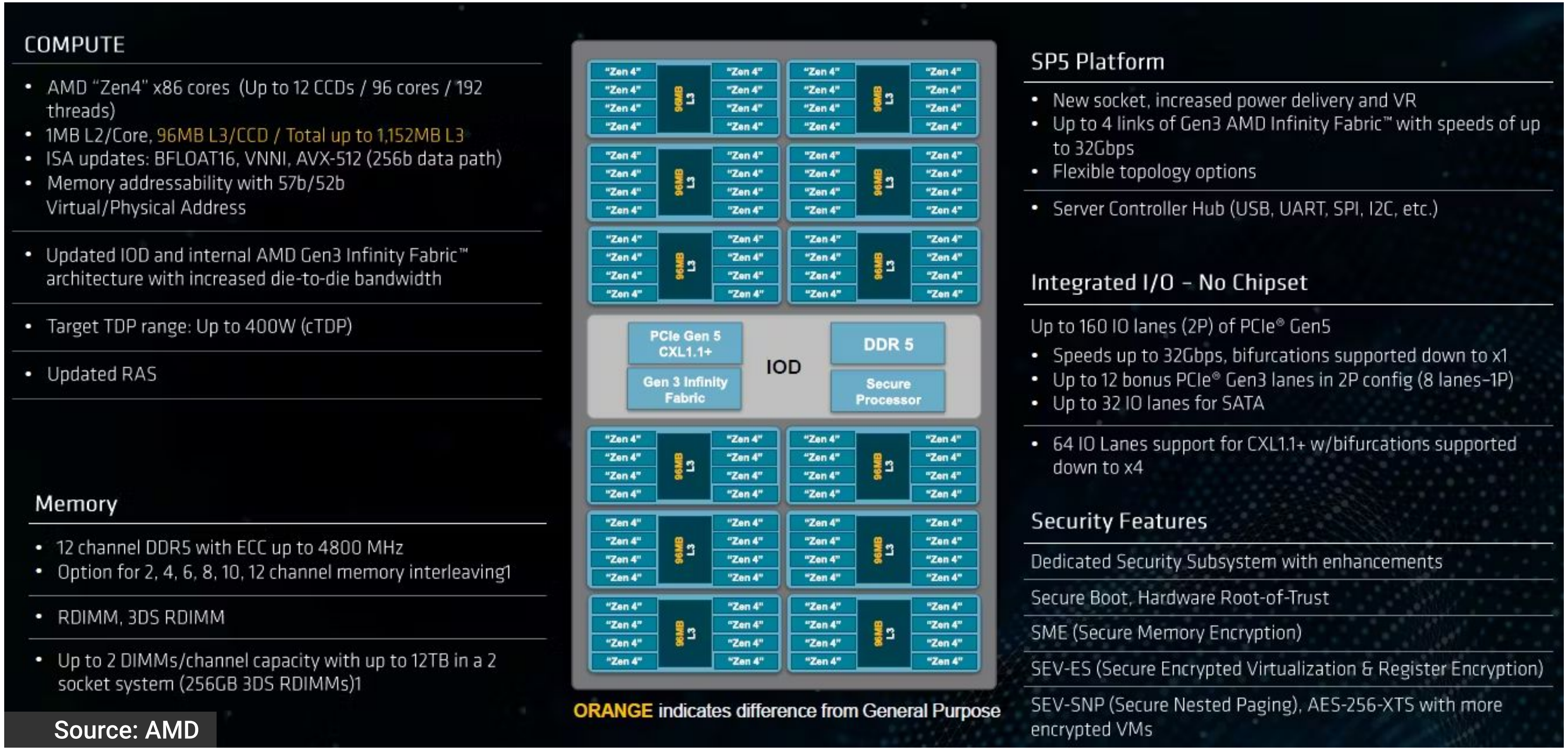
Non-Uniform Memory Architecture (NUMA)



	AMD EPYC 7001 'NAPLES'	AMD EPYC 7002 'ROME'	AMD EPYC 7003 'MILAN'	AMD EPYC 9004, 8004 'GENOA', 'SIENA'
				
Core Architecture	'Zen'	'Zen 2'	'Zen 3'	'Zen 4' and 'Zen 4c'
Cores	8 to 32	8 to 64	8 to 64	8 to 128
IPC Improvement Over Prior Generation	N/A	~24%[ROM-236]	~19%[MLN-003]	~14%[EPYC-038]
Max L3 Cache	Up to 64 MB	Up to 256 MB	Up to 256 MB	Up to 384 MB (EPYC 9004) Up to 128 MB (EPYC 8004)
Max L3 Cache with 3D V-Cache™ technology			768 MB	Up to 1152 MB
PCIe® Lanes	Up to 128 Gen 3	Up to 128 Gen 3	Up to 128 Gen 4	Up to 128 Gen 5 8 bonus lanes Gen 3
CPU Process Technology	14nm	7nm	7nm	5nm
I/O Die Process Technology	N/A	14nm	14nm	6nm
Power (Configurable TDP [cTDP])	120-200W	120-280W	155-280W	70-400W
Max Memory Capacity	2 TB DDR3-2400/2666	4 TB DDR4-3200	4 TB DDR4-3200	6 TB DDR5-4800

Source: AMD

AMD Zen4 Architecture in detail



COMPUTE

- AMD "Zen4" x86 cores (Up to 12 CCDs / 96 cores / 192 threads)
- 1MB L2/Core, 96MB L3/CCD / Total up to 1,152MB L3
- ISA updates: BFLOAT16, VNNI, AVX-512 (256b data path)
- Memory addressability with 57b/52b Virtual/Physical Address

- Updated IOD and internal AMD Gen3 Infinity Fabric™ architecture with increased die-to-die bandwidth

- Target TDP range: Up to 400W (cTDP)

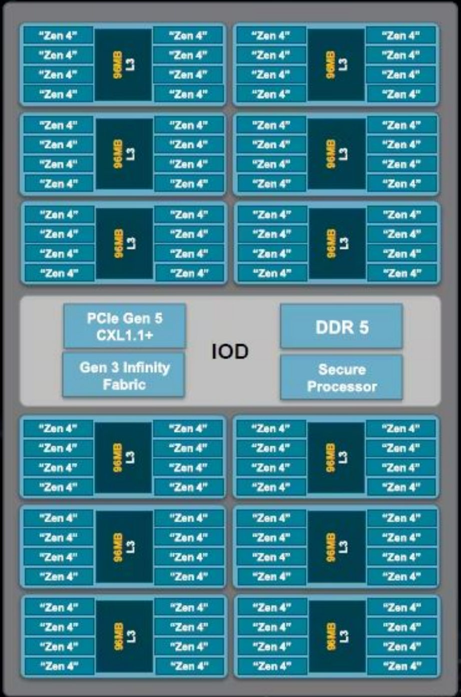
- Updated RAS

Memory

- 12 channel DDR5 with ECC up to 4800 MHz
- Option for 2, 4, 6, 8, 10, 12 channel memory interleaving1

- RDIMM, 3DS RDIMM

- Up to 2 DIMMs/channel capacity with up to 12TB in a 2 socket system (256GB 3DS RDIMMs)1



ORANGE indicates difference from General Purpose

SP5 Platform

- New socket, increased power delivery and VR
- Up to 4 links of Gen3 AMD Infinity Fabric™ with speeds of up to 32Gbps
- Flexible topology options

- Server Controller Hub (USB, UART, SPI, I2C, etc.)

Integrated I/O – No Chipset

Up to 160 IO lanes (2P) of PCIe® Gen5

- Speeds up to 32Gbps, bifurcations supported down to x1
- Up to 12 bonus PCIe® Gen3 lanes in 2P config (8 lanes-1P)
- Up to 32 IO lanes for SATA

- 64 IO Lanes support for CXL1.1+ w/bifurcations supported down to x4

Security Features

Dedicated Security Subsystem with enhancements

Secure Boot, Hardware Root-of-Trust

SME (Secure Memory Encryption)

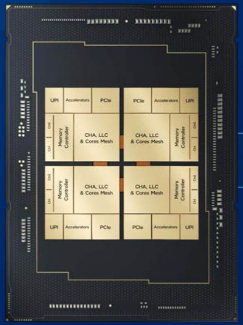
SEV-ES (Secure Encrypted Virtualization & Register Encryption)

SEV-SNP (Secure Nested Paging), AES-256-XTS with more encrypted VMs

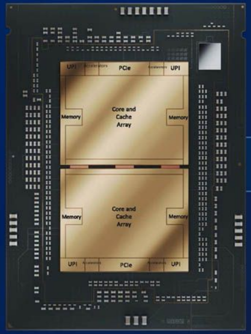
Source: AMD

Intel® Xeon® Die Package Enhancements

Scalable, Balanced Architecture



4th Gen Intel Xeon
(Four-Tile Architecture)



5th Gen Intel Xeon
(Two-Tile Architecture)

intel.

5th Gen Intel® Xeon® Processors Turbo Frequencies

Introducing Improved 5 Turbo Ratio Levels

- Improves Turbo Frequencies for Intel® AVX heavy and Intel® AMX light workloads including HPC and AI
- ~2 bins Turbo Frequency Upside on Intel® AVX-512 Heavy usage
- ~9% performance improvement on low load (4T or 8T) LINPACK AVX512
- ~5% performance improvement on low instance (4 or 32) Resnet50 amx_int8, amx_bfloat16 and avx_fp32
- Lowers the Turbo frequency penalty for using AVX512 or AMX, broadening usability of these instruction sets

4th Gen Intel® Xeon® CPU

Instruction Class	Cdyn Class			
	0	1	2	3
SSE	128 Light	128 Heavy		
AVX2	256 Light	256 Moderate	256 Heavy	
AVX512	512 Ultra-Light	512 Light	512 Moderate	512 Heavy
AMX		AMX Light	AMX Moderate	AMX Heavy
Turbo Frequency	SSE	AVX2	AVX512	AMX

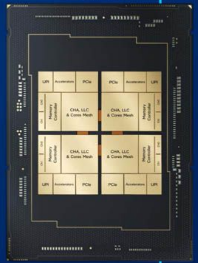
5th Gen Intel® Xeon® CPU

Instruction Class	Cdyn Class				
	0	1	2	3	4
SSE	128 Light	128 Heavy			
AVX2	256 Light	256 Moderate	256 Heavy		
AVX512	512 Ultra-Light	512 Light	512 Moderate	512 Heavy	
AMX ENHANCED		AMX Ultra-Light	AMX Light	AMX Moderate	AMX Heavy
Turbo Frequency	SSE	AVX2	AVX512	AVX512H NEW	AMX

5th Gen Intel® Xeon® Processors

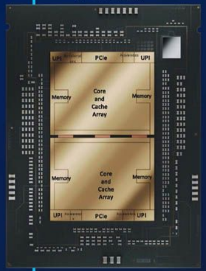
Platform enhancements delivering significant gains at the same power envelope[1]

4th Gen Xeon



- **Drop-in compatible** with 4th Gen Xeon processors
- Up to **64** cores per CPU
- Up to **5600** MT/s memory speed
- Up to **3x** shared Last Level Cache (up to **320 MB** total)
- Up to **20** GT/s UPI 2.0 Speed
- **Type 3** memory support with Compute Express Link® 1.1*
- Intel® Trust Domain Extension **broad support**

5th Gen Xeon



intel.

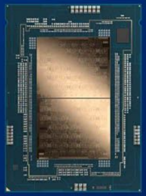
* Type 3 CXL targeted deployment
See backup for configuration details. Results may vary.

Compute Express Link® 1.1 Enhancements

Type 3 memory support with 5th Gen Intel® Xeon® processors

CXL Memory Buffer

DDR5 DDR5 DDR5 DDR5



DDR5 DDR5 DDR5 DDR5

CXL Memory Buffer

2 Tier Memory Support Example

2–Tier Memory Support

Type 3 Memory Expansion Devices:

Capacity Expansion

- Tier 1 memory = native DDR, Tier 2 memory = CXL® attached memory
- Supports up to 4 channels of CXL memory across two CXL type 3 devices
- Supports CXL memory latency QoS distress signaling

Increased transactions per second for In-Memory databases (e.g. Redis)

Single Tier Memory Support

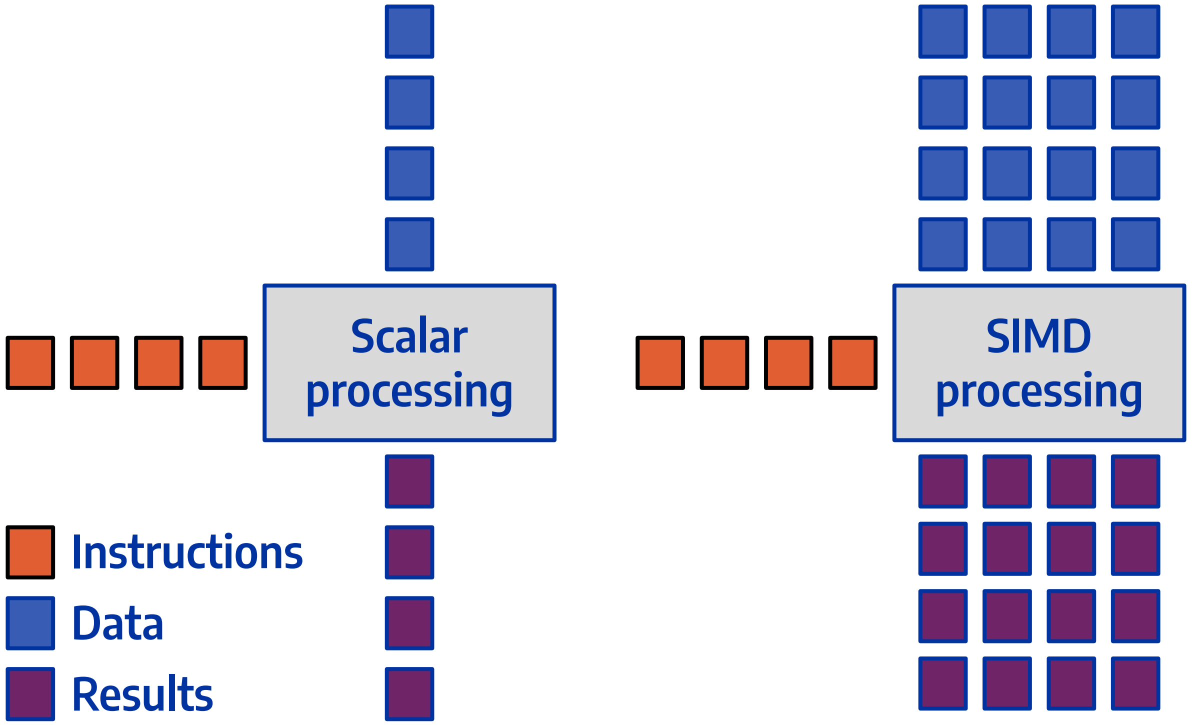
12 channel DDR+CXL interleaved memory

- Either for capacity or bandwidth expansion

intel.

Source: Intel

SIMD Vectorization



History of Intel® SIMD ISA Extensions

- Intel® Pentium Processor (1993)

  ☐ *32bit*

- Multimedia Extensions (MMX in 1997)

  ☐☐ *64bit integer support only*

- Streaming SIMD Extensions (SSE in 1999 to SSE4.2 in 2008)

  ☐☐☐☐ *32bit/64bit integer and floating point, no masking*

- Advanced Vector Extensions (AVX in 2011 and AVX2 in 2013)

  ☐☐☐☐☐☐☐☐ *Fused multiply-add (FMA), HW gather support (AVX2)*

- Many Integrated Core Architecture (Xeon Phi™ Knights Corner in 2013)

  ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ *HW gather/scatter, exponential*

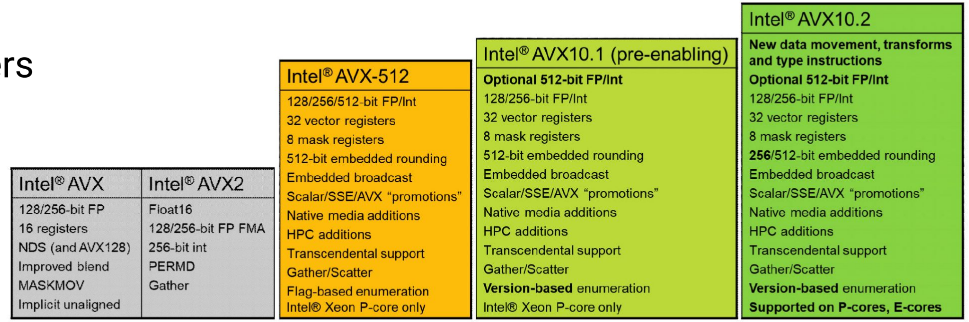
- AVX512 on Knights Landing, Skylake Xeon, and Core X-series (2016/2017)

  ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ *Conflict detection instructions*

☐ = 32 bit word

Evolution of Intel® SIMD ISA Extensions

- AVX 10
  - Supported on both P-cores and E-cores
  - Brings benefits of AVX512 to smaller registers
- Advanced Matrix Extensions (AMX)
  - Targeted at AI applications
  - SIMD for small matrix operations
  - Available on 4th and 5th generation Xeon
- Advanced Performance Extensions (APX)
  - Adds new features that improve general-purpose performance
  - Expands x86 instruction set with more general-purpose registers (from 16 to 32)
  - New REX2 prefix provides uniform access to the new registers
  - Adds conditional forms of load, store, and compare/test instructions
  - New prefix increase average instruction length, but there are less instructions overall

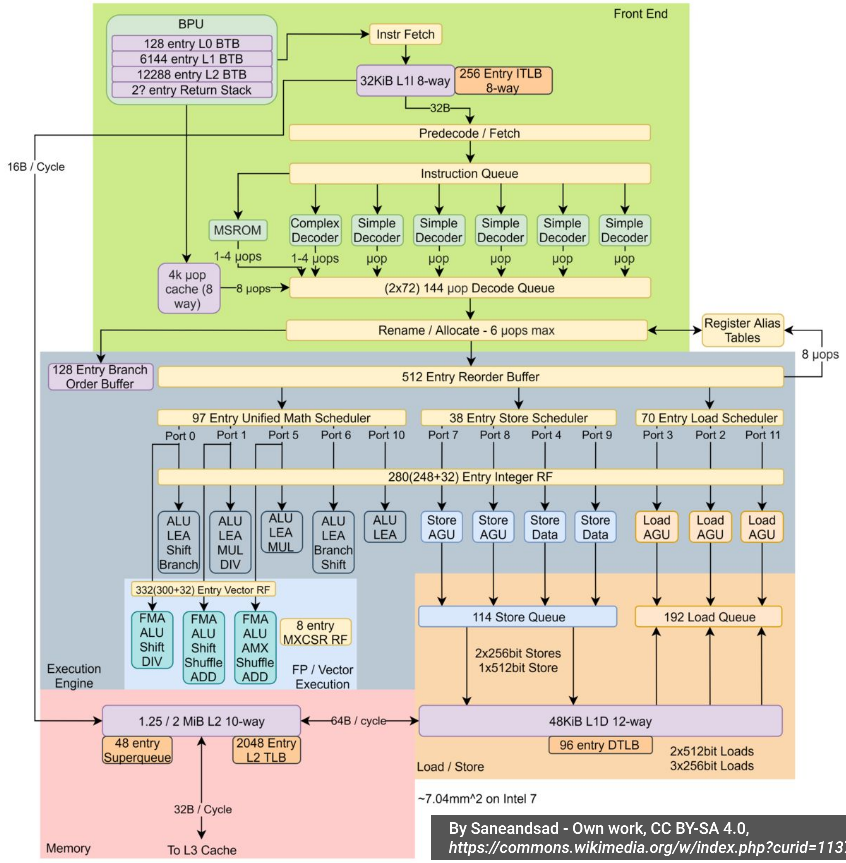


Intel® AVX	Intel® AVX2	Intel® AVX-512	Intel® AVX10.1 (pre-enabling)	Intel® AVX10.2
128/256-bit FP	Float16	128/256/512-bit FP/Int	**Optional 512-bit FP/Int**	**New data movement, transforms and type instructions**
16 registers	128/256-bit FP FMA	32 vector registers	128/256-bit FP/Int	**Optional 512-bit FP/Int**
NDS (and AVX128)	256-bit int	8 mask registers	32 vector registers	128/256-bit FP/Int
Improved blend	PERMD	512-bit embedded rounding	8 mask registers	32 vector registers
MASKMOV	Gather	Embedded broadcast	512-bit embedded rounding	8 mask registers
Implicit unaligned		Scalar/SSE/AVX "promotions"	Embedded broadcast	**256**/512-bit embedded rounding
		Native media additions	Scalar/SSE/AVX "promotions"	Embedded broadcast
		HPC additions	Native media additions	Scalar/SSE/AVX "promotions"
		Transcendental support	HPC additions	Native media additions
		Gather/Scatter	Transcendental support	HPC additions
		Flag-based enumeration	Gather/Scatter	Transcendental support
		Intel® Xeon P-core only	**Version-based** enumeration	Gather/Scatter
			Intel® Xeon P-core only	**Version-based** enumeration
				Supported on P-cores, E-cores

Microarchitecture of a Modern Intel Core

- Front End
  - Instruction Fetch and Decode
  - Branch Predictor Unit (BPU)
  - L1 Instruction Cache
  - Instruction TLB
- Back End
  - Execution Engine
    - Scheduler
    - Register File
    - Execution Units (EUs)
  - Memory Subsystem
    - Load/Store Units (LSU)
    - L1 / L2 Data Cache
    - Data TLB

Intel Golden Cove Core



By Saneandsad - Own work, CC BY-SA 4.0, *https://commons.wikimedia.org/w/index.php?curid=113727886*

Meteor Lake Hybrid Architecture

CPU Tile
GFX Tile
SOC Tile
IO Tile

Meteor Lake
built on Intel 4 process node
powering on in Q2'22, shipping in 2023

New Flexible Tiled Architecture

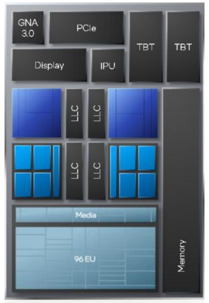
Hybrid Cores

Lower Power

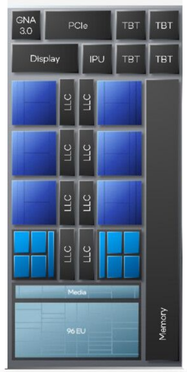
Next Gen Graphics Engine: tGPU

Integrated AI Acceleration

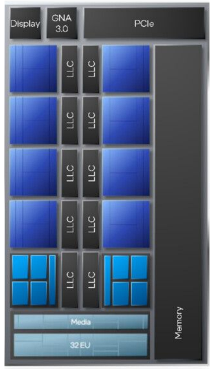
Ultra Mobile



Mobile

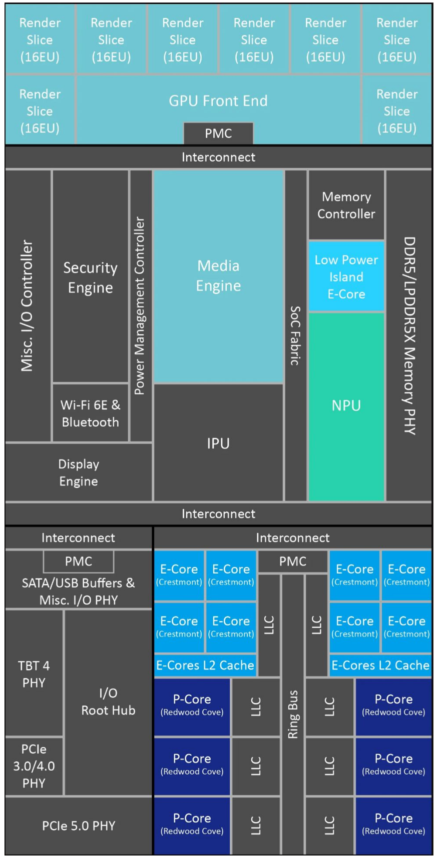


Desktop



Source: Intel

Meteor Lake Block Diagram



REDWOOD COVE

New P-core

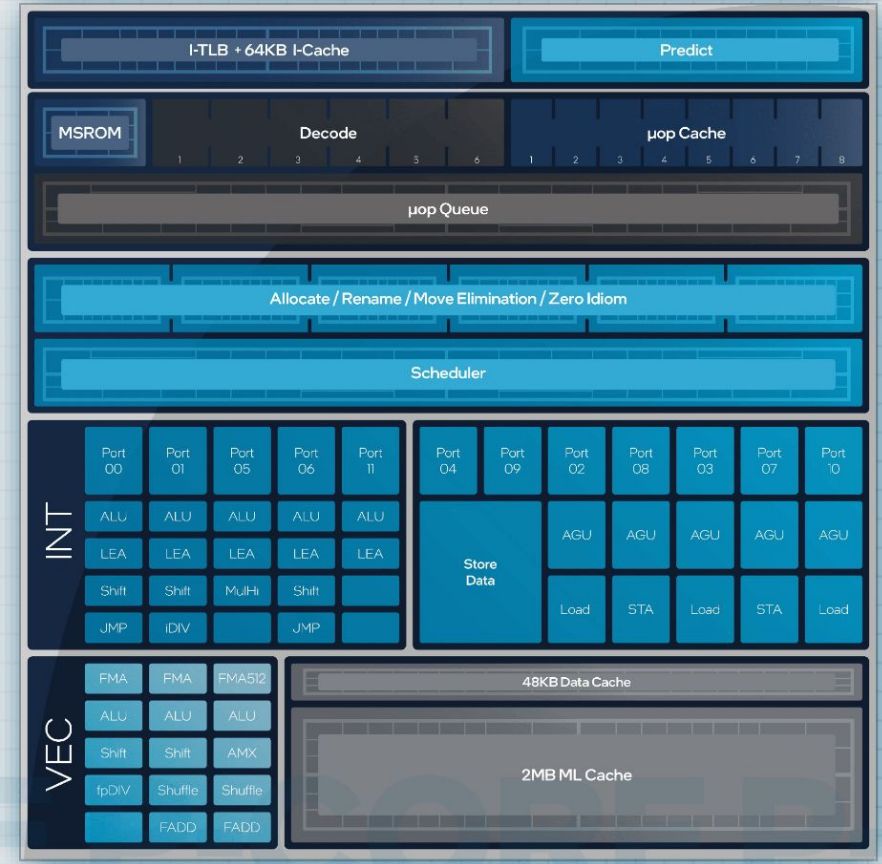
Targeted for efficient performance

Improved performance efficiency*

Increased BW per core package*

Improved Performance Monitoring Unit

Improved feedback Intel Thread Director



Source: Intel

*Architectural simulation vs. Golden Cove architecture. Results may vary across workloads.

CRESTMONT

New E-core

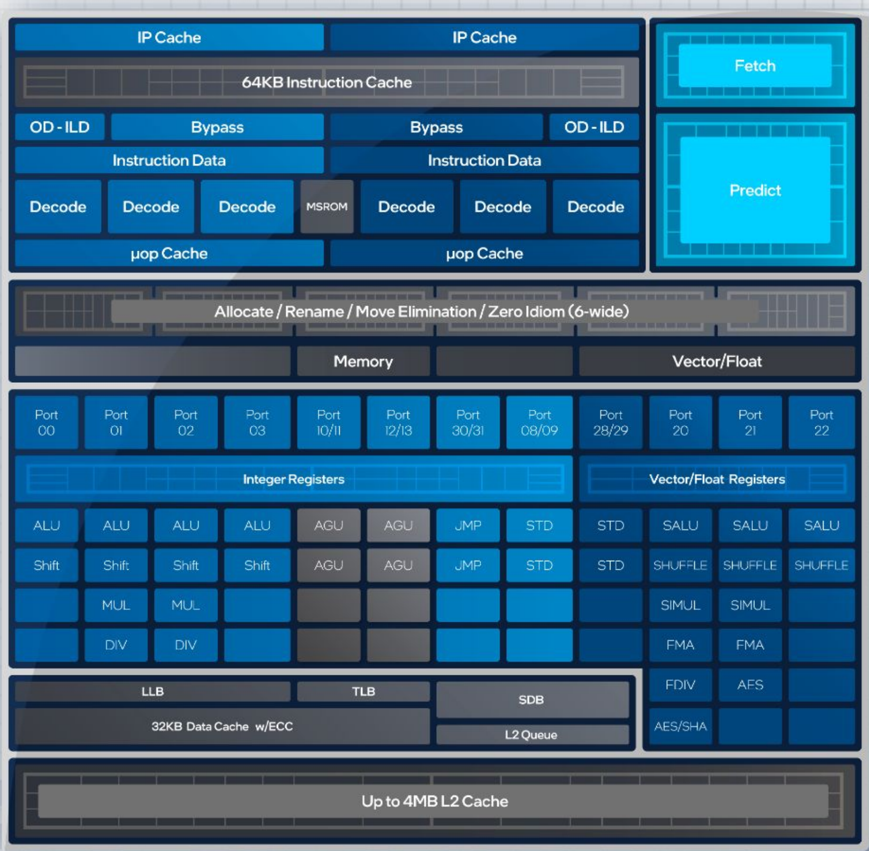
Significant improvements over prior E-core

IPC gains over prior E-cores*

AI acceleration VNNI, ISA improvements*

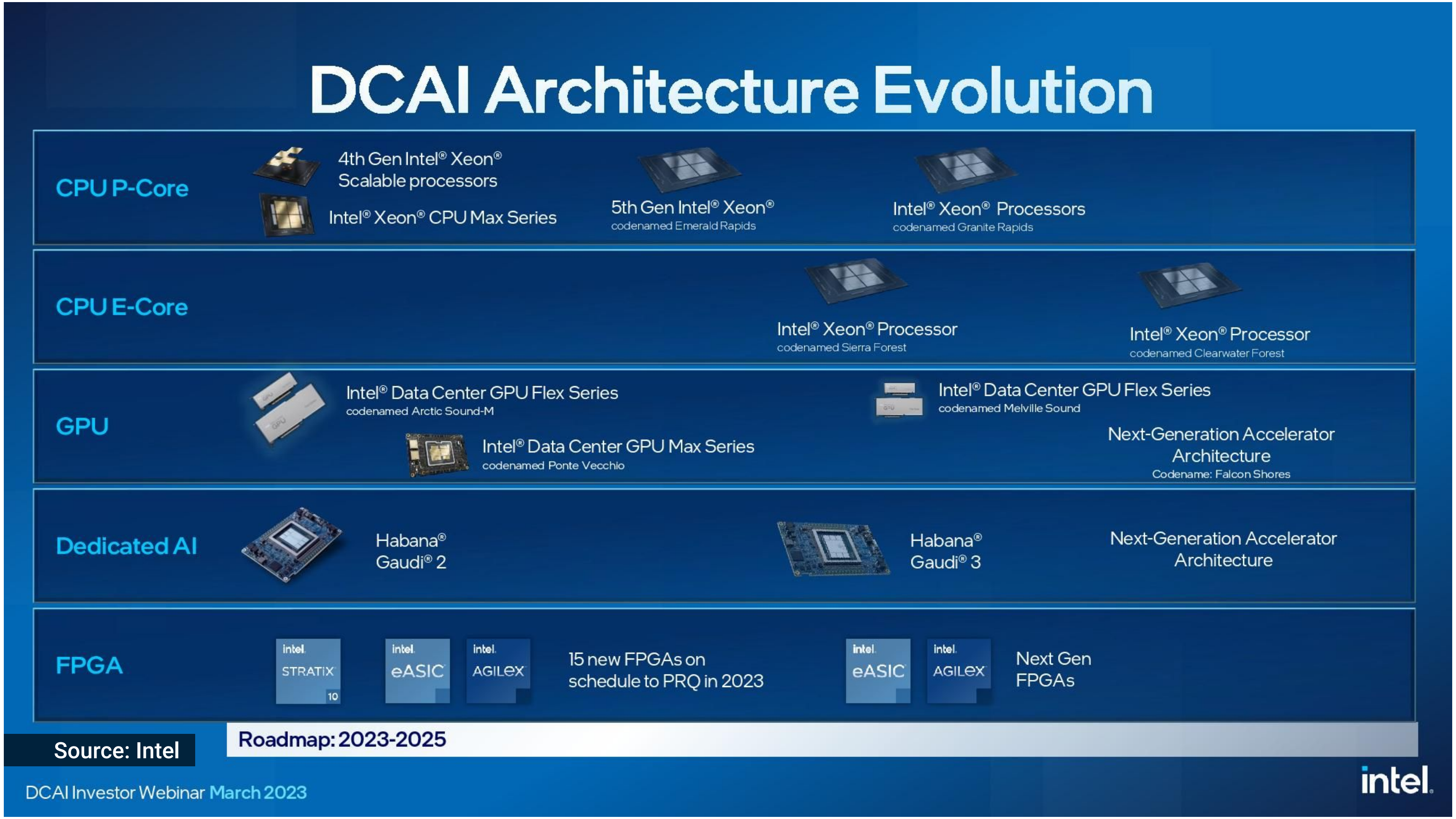
Enhanced branch prediction

Enhanced Feedback Intel Thread Director



*Architectural simulation vs. Gracemont architecture across a broad set of workloads. VNNI improvements based on doubling the number of VNNI ports. Results may vary.

Source: Intel



DCAI Architecture Evolution

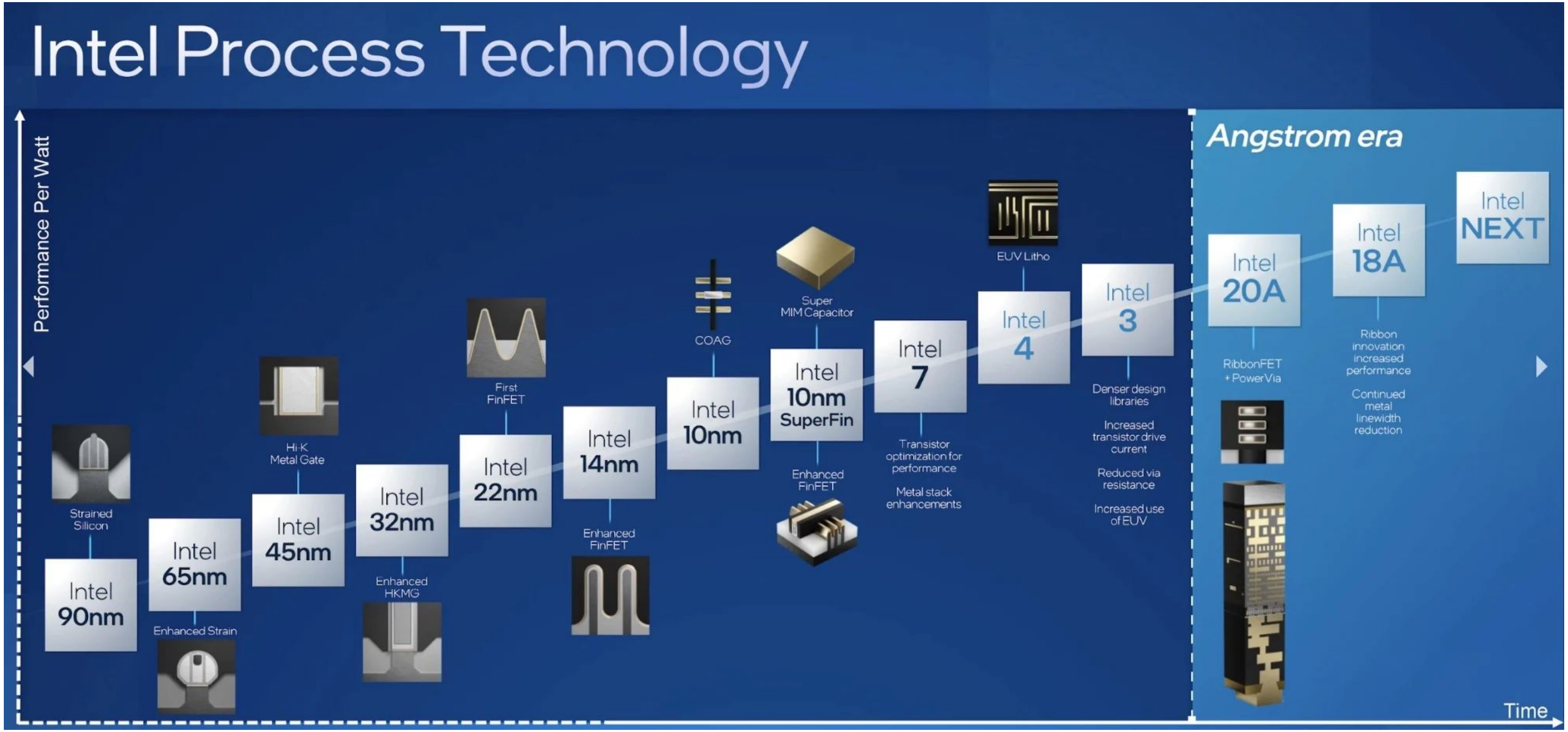
CPU P-Core	4th Gen Intel® Xeon® Scalable processors; Intel® Xeon® CPU Max Series	5th Gen Intel® Xeon® codenamed Emerald Rapids	Intel® Xeon® Processors codenamed Granite Rapids
CPU E-Core		Intel® Xeon® Processor codenamed Sierra Forest	Intel® Xeon® Processor codenamed Clearwater Forest
GPU	Intel® Data Center GPU Flex Series codenamed Arctic Sound-M; Intel® Data Center GPU Max Series codenamed Ponte Vecchio	Intel® Data Center GPU Flex Series codenamed Melville Sound	Next-Generation Accelerator Architecture Codename: Falcon Shores
Dedicated AI	Habana® Gaudi® 2	Habana® Gaudi® 3	Next-Generation Accelerator Architecture
FPGA	intel STRATIX 10; intel eASIC; intel AGILEX; 15 new FPGAs on schedule to PRQ in 2023	intel eASIC; intel AGILEX; Next Gen FPGAs	

Roadmap: 2023-2025

Source: Intel

DCAI Investor Webinar March 2023

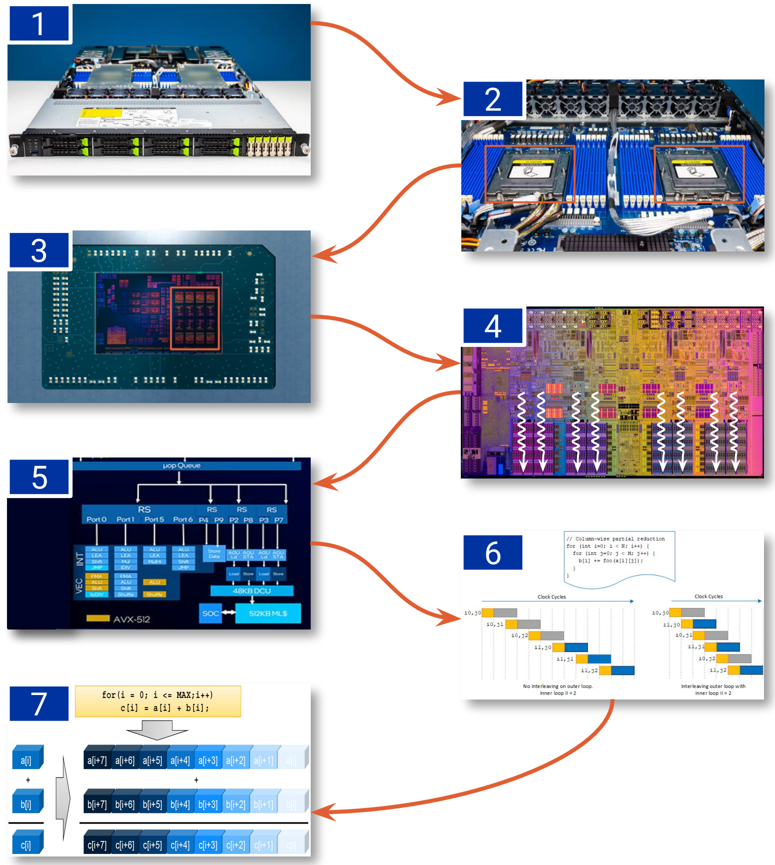
intel



Source: Intel

Modern Hardware

1	Systems	• Has always been there • Run copies of your code on each node
2	Sockets	• Modern machines have up to 8 sockets • Trend towards reduction back to 2 only
3	Cores	• Higher frequency no longer possible • 128 cores in each socket now common
4	Threads	• Hardware has N physical cores • OS sees 2N logical cores, shared exec.
5	Ports	• Superscalar execution • Multiple instructions executed at once
6	Pipelining	• Parallel instruction execution steps • Fetch / Decode / Execute / Write Back
7	SIMD Vectors	• SSE4.2, AVX2, AVX512, AVX10 • PowerPC Altivec, ARM SVE, etc



```
// Column-wise partial reduction
for (int i=0; i < N; i++) {
  for (int j=0; j < M; j++) {
    b[i] += foo(a[i][j]);
  }
}
```

```
for(i = 0; i <= MAX;i++)
    c[i] = a[i] + b[i];
```

Summary

- We’ve come a long way, modern hardware is quite complex
  - NUMA Architecture (multi socket)
  - High parallelism (multicore, superscalar)
  - Advanced Packaging (chiplets)
  - Hybrid Architectures (performance/efficiency)
  - Variable CPU frequency scaling (turbo boost, thermal throttling)
  - Accelerators and Heterogeneity (GPUs, NPUs, FPGAs, ASICs)
- Performance does not come for free, we needed to adapt our software
  - Concurrency and Parallelism (processes, threads, SIMD)
  - Memory alignment, access patterns, fragmentation
  - Code layout, compiler optimizations, data structures, software design
  - Need the right tools to guide us: profilers, static analysis, etc
  - Need the right methodology: identify causes of bottlenecks, address the right issue