LECTURE 6

- Random and Biased Random Walk in 1D (Gambler's Ruin)
- The rate of exponential growth of technical civilization (1974-2024): Moore's law
- Dennard's law, which enabled Moore's law. Derivation from the physics of electrical circuits.
- The great slow-down of exponential growth, its causes and effects for supercomputers and programmers today
- The need for parallelism
- Concurrent execution and Amdahl's law
- Programming and parallelism: C, Fortran, OpenMP

◆ Simple benchmark loop (simpler-nb-3da and -3aa) C, F95 heap vs. stack variables, comp-bound and comm-bound programs; timing and benchmarking programs

Literature: see links on our course home page, the references page, and on the coding page.

PHYD57. Advanced Comp. Methods in Physics.

Gambler’s ruin problem

- Simplest version: variable X is integer & evolves by sudden jumps by $\pm$ 1, with probabilities q and 1-q, correspondingly.
- For example, q =1/2 produces an unbiased random walk on a line of integers.
- Starting from value x, quantity X wanders around until either condition occurs:

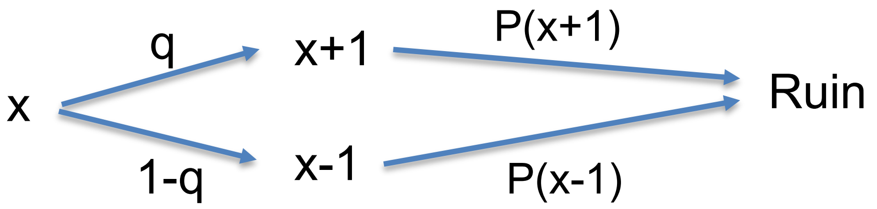
  X=0 (ruin), or X=N (happy ending). N >> 1 is some large integer number.
- What is the probability of gambler’s ruin, P(X=0 | x; q, N) ?

Numerical solution involving a 1-loop random walk simulations yields a statistical, MtC solution. But there is a more elegant way.

Analytical solution can be obtained by using an approach formerly used by the co-creator of MtC method, Stanisław Ulam, in Manhattan project at Los Alamos, and surely others even earlier.

Random walk has no memory of the past steps. It is a Markov process. Probability of ruin, starting at x, which we shorten to P(x), does not depend on prior history. Consider where one step of the walk starting at x leads, with what probabilities, then express P(x) using P(x+1) and P(x-1):

$$P(x) \;=\; \underset{\text{upper branch}}{q\,P(x+1)} \;+\; \underset{\text{lower branch}}{(1-q)\,P(x-1)}$$



Walk to ruin: 1st step The rest of steps (independent events)

- Gambler’s ruin problem (The case of unbiased random walk)

Rearrange the equation to read

$$q\,P(x+1) \;-\; P(x) \;+\; (1-q)\,P(x-1) \;=\; 0 \qquad (1)$$

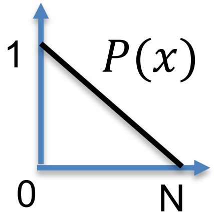
If $q = 1/2$ (unbiased random walk), then the equation is the estimator of the second derivative of $P(x)$, with step h (the smallest amount of change of x) equal unity in the denominator:

$$P''(x) \;=\; P(x+1) \;-\; 2\,P(x) \;+\; P(x-1) \;=\; 0$$

$P'' \;=\; 0$ implies $P' = const.$, so $P(x)$ is a linear function of x of the form $P \;=\; c\,x \;+\; d$, where $c, d = \; const$. Since the probability of ruin when starting from ruin condition is 1, and probability of ruin when starting from success condition is 0, those two cases provide two boundary conditions for the 2nd order ODE: P(0) = 1, P(N) = 0, making the solution unique. Boundary conditions are: $P(0) \;=\; d \;=\; 1,$ and $P(1) \;=\; c + d \;=\; 0, \quad so \quad d = 1, c \;=\; -1.$

The ruin after an unbiased random walk starting at x happens with probability

$$P(x) \;=\; 1 - x/N. \qquad (2)$$



- Gambler's ruin problem (The case of biased random walk)

$$q\,P(x+1) \;-\; P(x) \;+\; (1-q)\,P(x-1) \;=\; 0 \tag{1}$$

More generally, introduce a bias parameter b as

$b := q - ½$ or $q =: ½ + b$

Positive bias b=0...½ means a greater tendency to increase x than to decrease it, in any biased random walk step. Conversely, negative bias will bias the walk to end up at smaller x than the starting value

(think of roulette game at the casino, which has $b < 0$).

Plugging $q = ½ + b$ into eq. (1), and using the symmetric (2nd order) numerical approximation to 1st derivative

$P'(x) = [P(x+1) - P(x-1)]/(2\cdot 1)$, we obtain

$$P''(x) \;+\; 4b\,P'(x) \;=\; 0 \tag{3}$$

with the same two boundary conditions as in unbiased walk. $P(0) = 1$, and $P(1) = 0$. We can solve this ODE analytically.

- Gambler’s ruin problem (continued)

$$P''(x) \;+\; 4b\,P'(x) \;=\; 0; \qquad P(0) = 1,\;\; P(N) = \;0.$$

One integration reduces this 2nd order ODE to 1st order:

$$P'(x) \;+\; 4b\,P(x) \;=\; a, \qquad a \;=\; const. \tag{4}$$

The general solution consists of the homogeneous part $P_h(x)$, satisfying

$$P_h'(x) + \;4bP_h(x) \;=\; 0, \tag{5}$$

[multiplied by a constant chosen for the full solution to satisfy boundary conditions], plus the particular solution of the inhomogeneous eq. 4,

$$P_{in} = \;P_h\; \int^x P_h^{-1} \;\; a\,dx. \tag{6}$$

Homogeneous solution (cf. eq. 5) is obtained by separation of variables

$$P_h(x) = \;\exp(-4bx) \tag{7}$$

and the full solution, satisfying the boundary conditions (verify!), reads

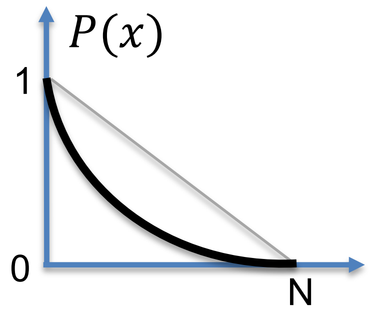
$$P(x) = \frac{e^{4b(N-x)}\,-1}{e^{4bN}\,-1} \tag{8}$$

Analyze the behavior of this solution for $b\; \to 0$, and for very large bias $(b \to \frac{1}{2}\; or -\frac{1}{2})$. Show that the ruin is less likely with $b > 0$ than with $b \to 0$, which is the same $P(x)$ as when $b = 0$. That’s good.

But what if $|b| = \frac{1}{2}\;\; (non - random\; walk)\;\;\; \&\; 0 < \; x < N\;?$

Why are the probabilities of ruin not exactly 0 or 1?

Evaluate $P(1)$, $P(N-1)$ according to eq. 8, for N=10.

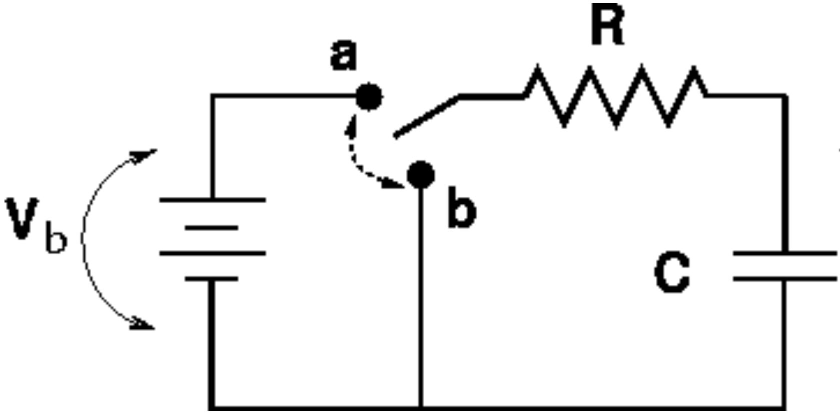


60 years of technical civilization, 1960-2020, in graphs and numbers

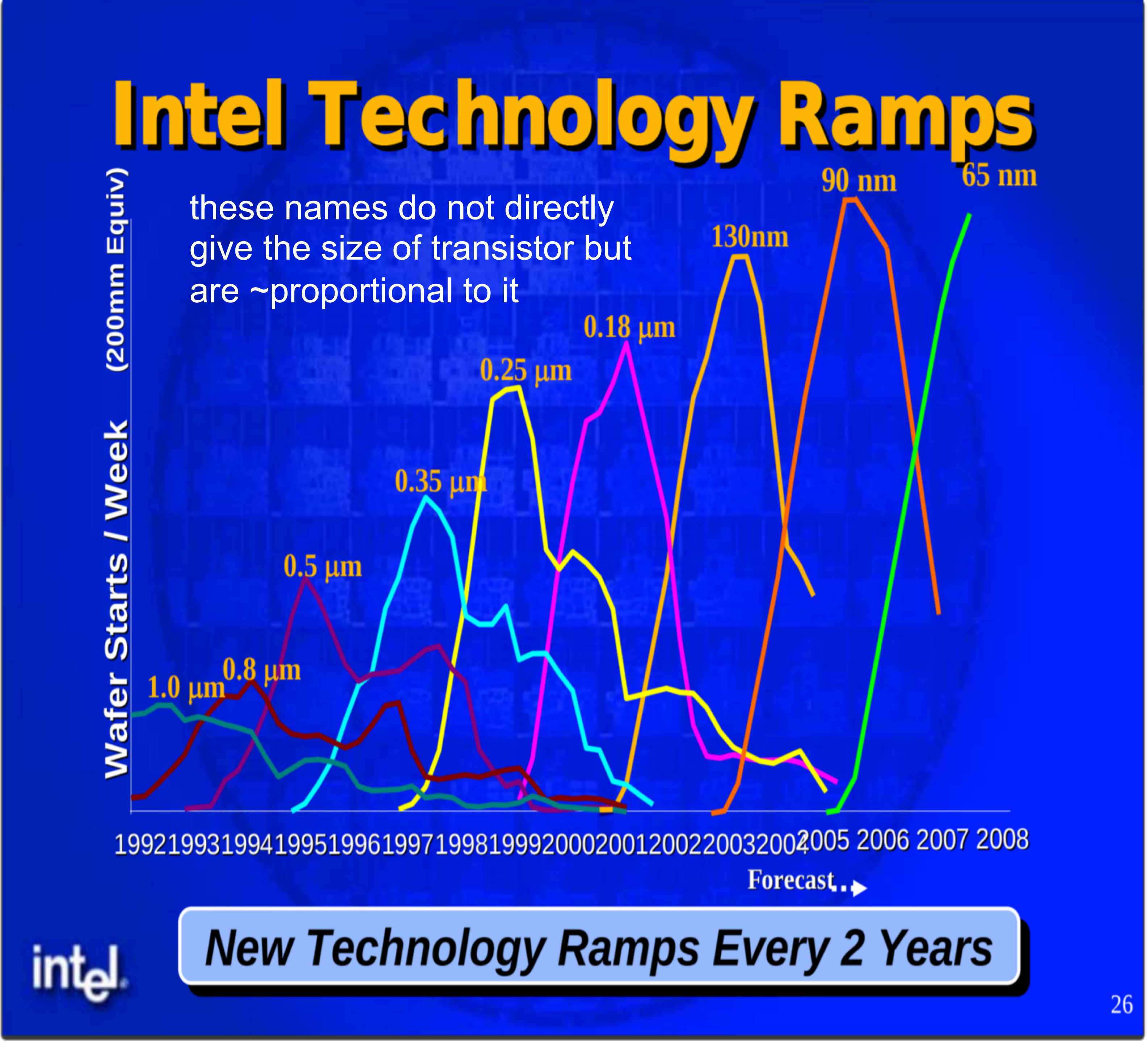
Literature:

- https://hasler.ece.gatech.edu/Published_papers/Technology_overview/gordon_moore_1965_article.pdf
- https://www.tha.de/Binaries/Binary20965/Post-Dennard-HSA-Forschungstag2014.pdf
- https://top500.org/ the ‘Top 500’ site about top supercomputers, lists published 2x a year.

The rise and the present slow-down of the exponential growth of our TECHNICAL CIVILIZATION



~10 G transistors, acting like RC circuits, are etched in *every* CPU and GPU chip of a modern Workstation, and even the phone.



Intel Technology Ramps

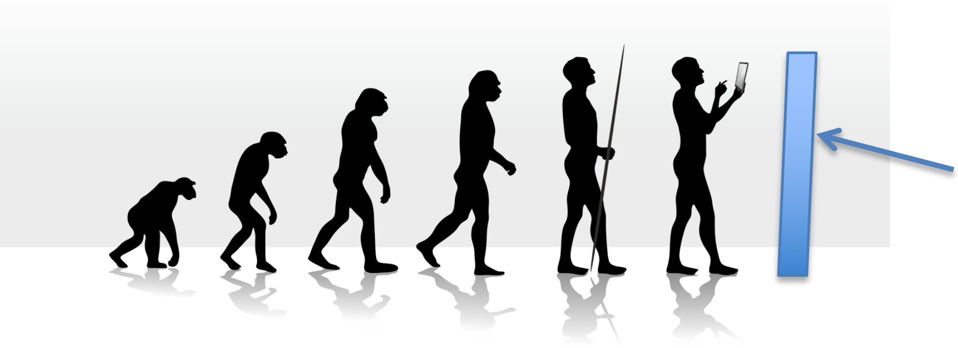
these names do not directly give the size of transistor but are ~proportional to it

New Technology Ramps Every 2 Years

Name of technology cycle:

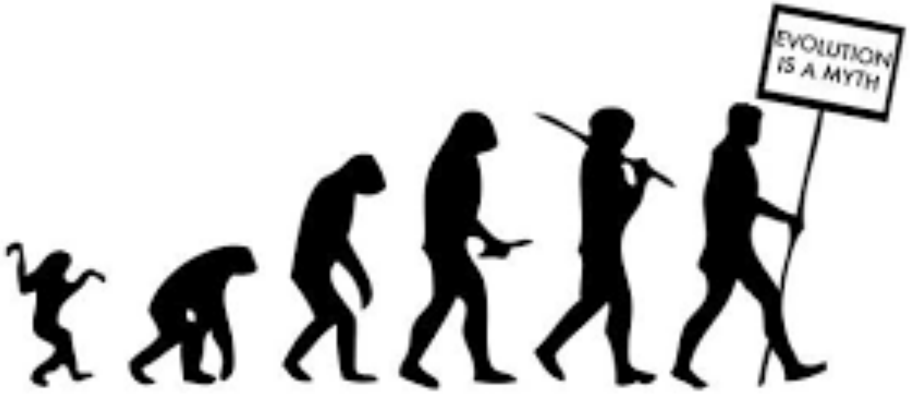
10 µm	1971
6 µm	1974
3 µm	1977
1.5 µm	1981
1 µm	1984
800 nm	1987
600 nm	1990
350 nm	1993
250 nm	1996
180 nm	1999
130 nm	2001
90 nm	2003
65 nm	2005
45 nm	2007
32 nm	2009
22 nm	2012
14 nm	2014
10 nm	2016
7 nm	2019
5 nm	???

The surface density of transitors up to ~100 MT/mm^2



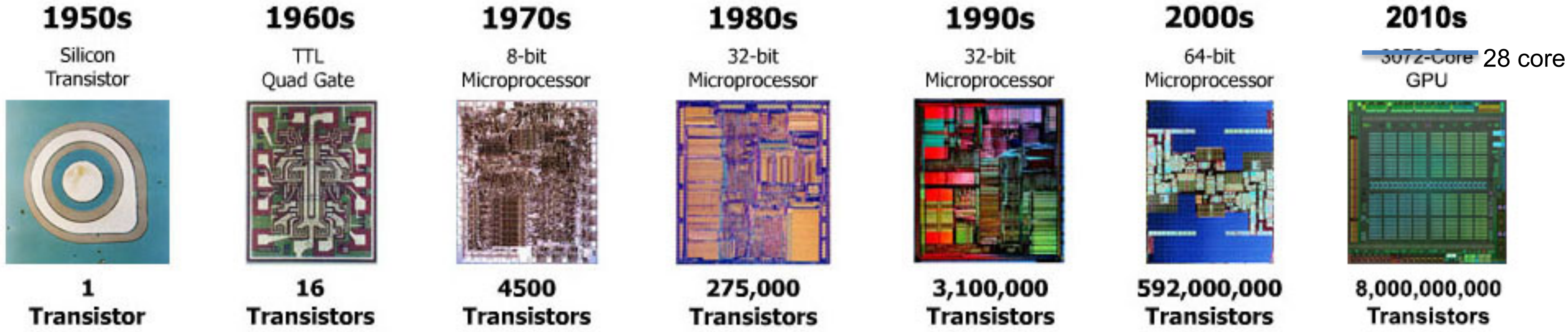
the power wall in front of
homo smartphoniensis

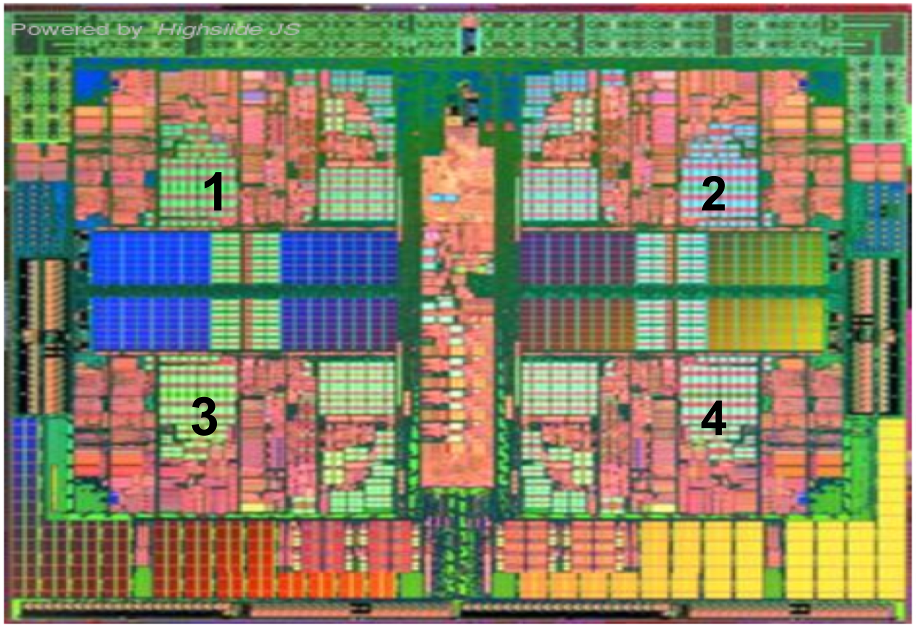
THE RISE & CURRENT SLOWDOWN
of the exponential growth of our technical civilization



1970s—2010s, the integrated circuit revolution

as a basis of all our current technical civilization: 50 years of **Moore's law**





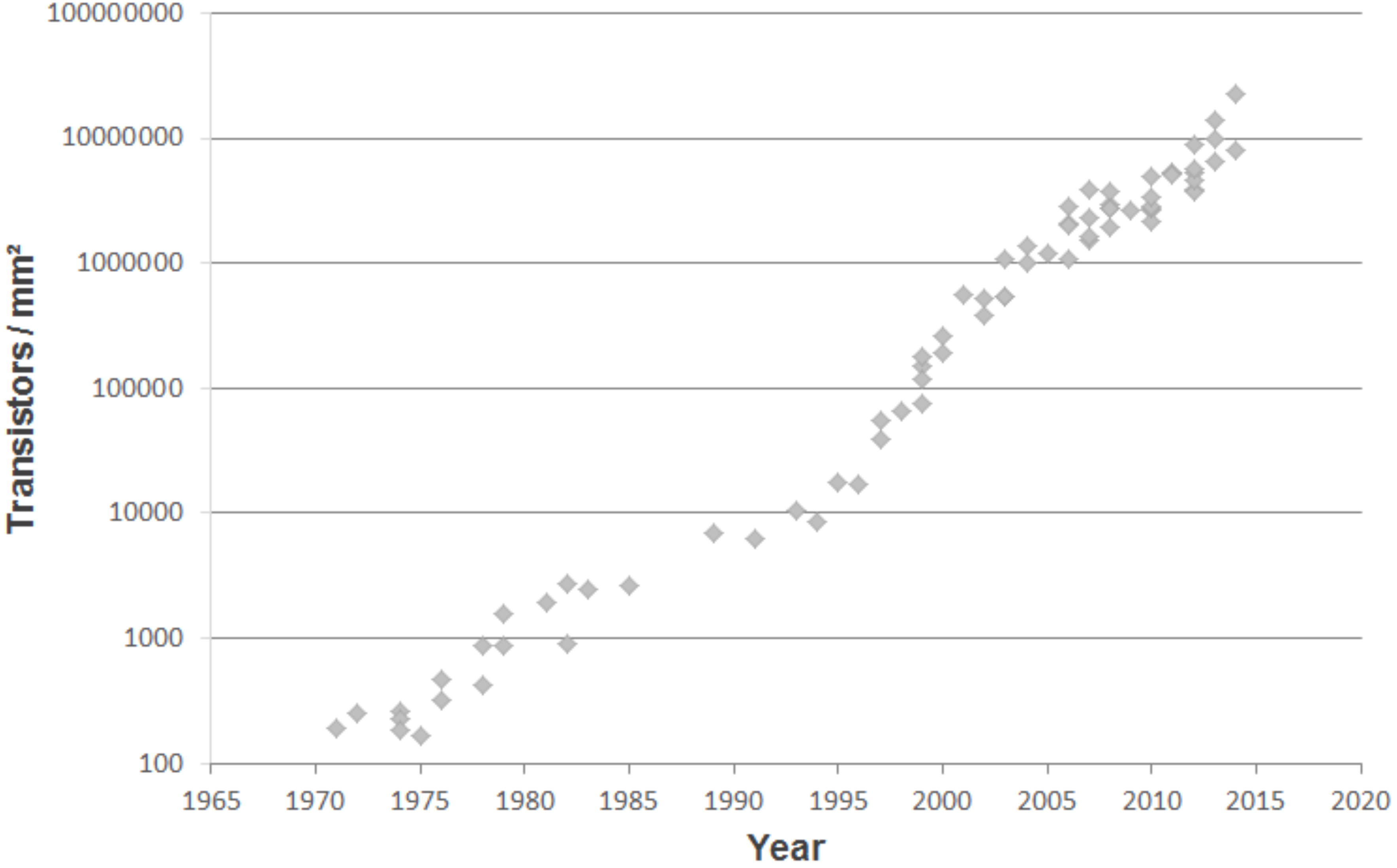
A highly integrated circuit (IC) has both arithmetic + logic units (ALU), fast cache memory hierarchy, and communication circuits, all in a small package of a Central Processing Unit (CPU).

N = up to $\mathbf{10^{11}}$ transistors (2024)

This processor looks like a 4-core CPU or old GPU (Central or Graphics Processing Unit).
It is not easy to execute one calculation simultaneously on many **cores (~ sub-CPUs)** of a CPU.
We call such processors **multicore**, and programs running on all cores **multithreaded**.

Moore's law: Transistor density grew exponentially.

It used to double every 2 years, partly a self-fulfilling prophecy of the chip industry.



TECHNICAL CIVILIZATION in the last quarter of 20th century

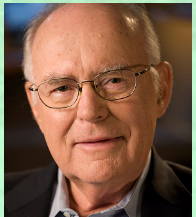
The physics of transistors over the last half-century allowed a complete and unprecedented transformation of the way we store, process, transfer and use information in modern society.

Hardware was historically ahead of software.
Sometimes, vary rarely, our bad programming is actually dangerous (we lost to it space probes; two B737 MAX airplanes crashed recently) but these are exceptions.

Our bicycles, cars, ships & airplanes do not travel twice as fast every year...
(estimate how fast we'd travel if they did!)

But soon only bicycles may do well without the help of computers, although even that is changing. There are computer-equipped gear changers on modern bikes :-|

An empirical observation made in 1964 by Fairchild Semiconductor & Intel Corp. co-founder Gordon Moore spelled out the past rate of development of computing hardware from 1959. It deals with # of transistors per unit surface of a microchip. Similar laws described similar, exponential, growth of bandwidth and other aspects of computing.

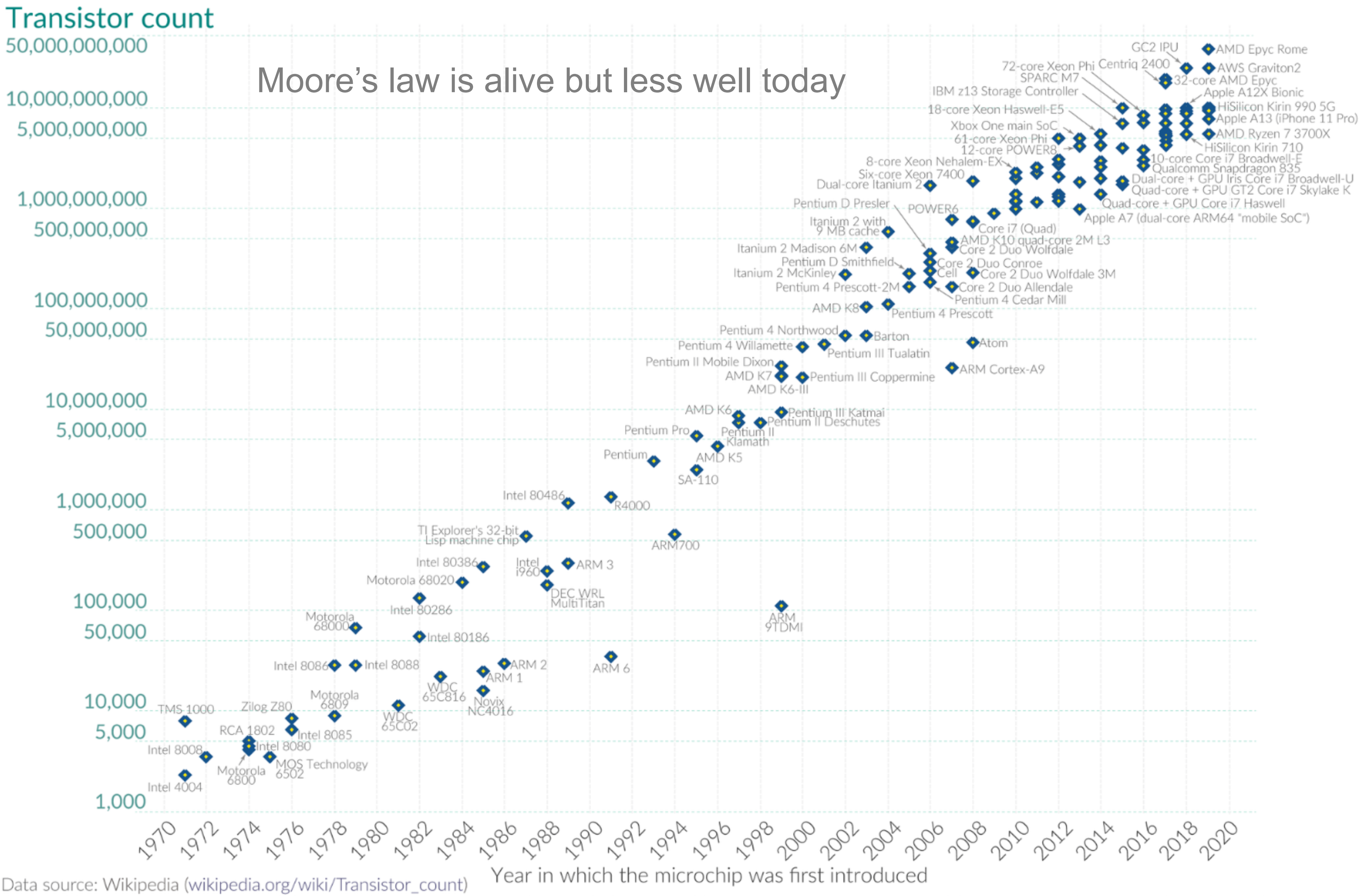


https://www.cs.utexas.edu/~fussell/courses/cs352h/papers/moore.pdf

Moore's Law: The number of transistors on microchips doubles every two years

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

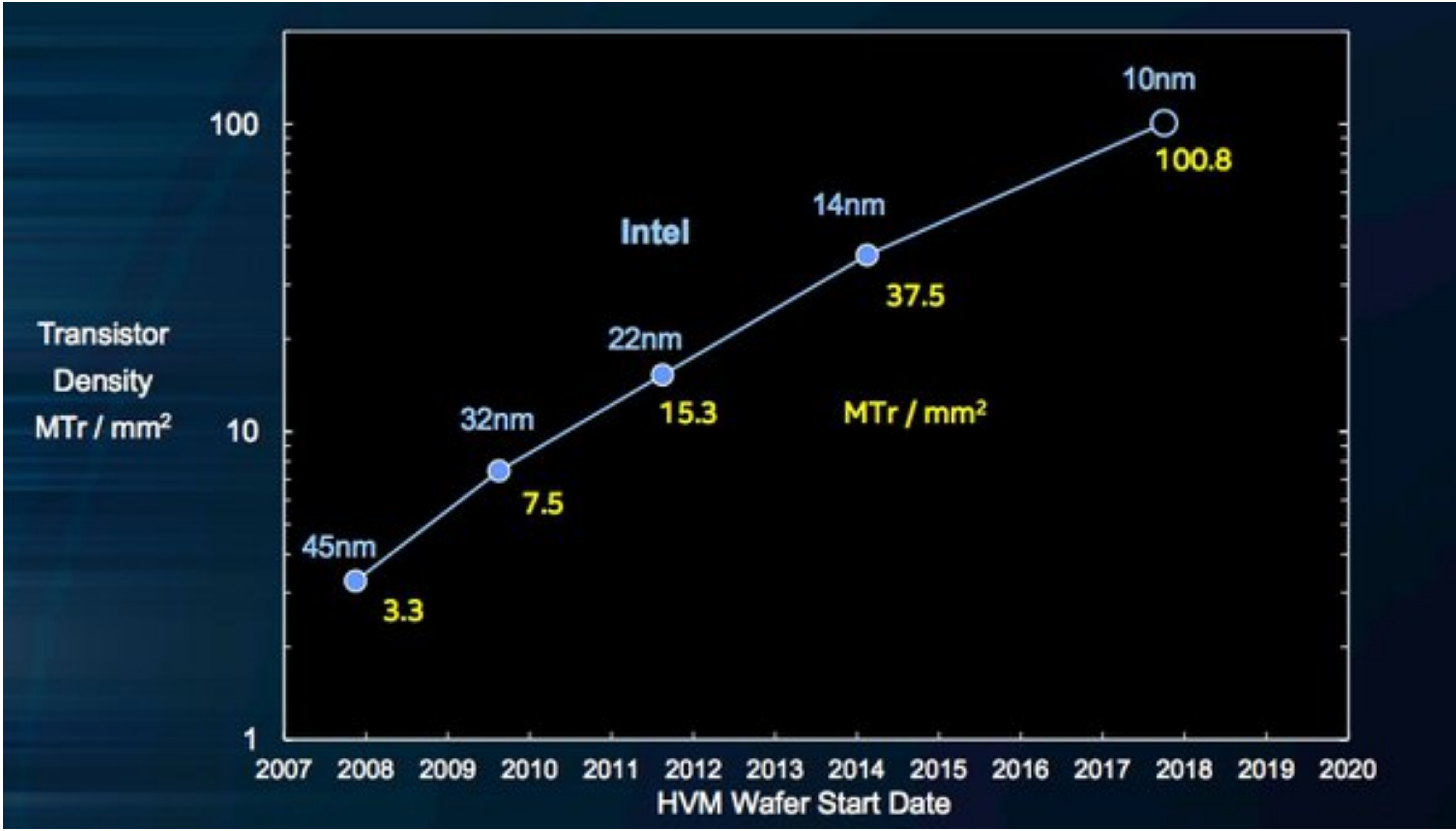


Moore's law is alive but less well today

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

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

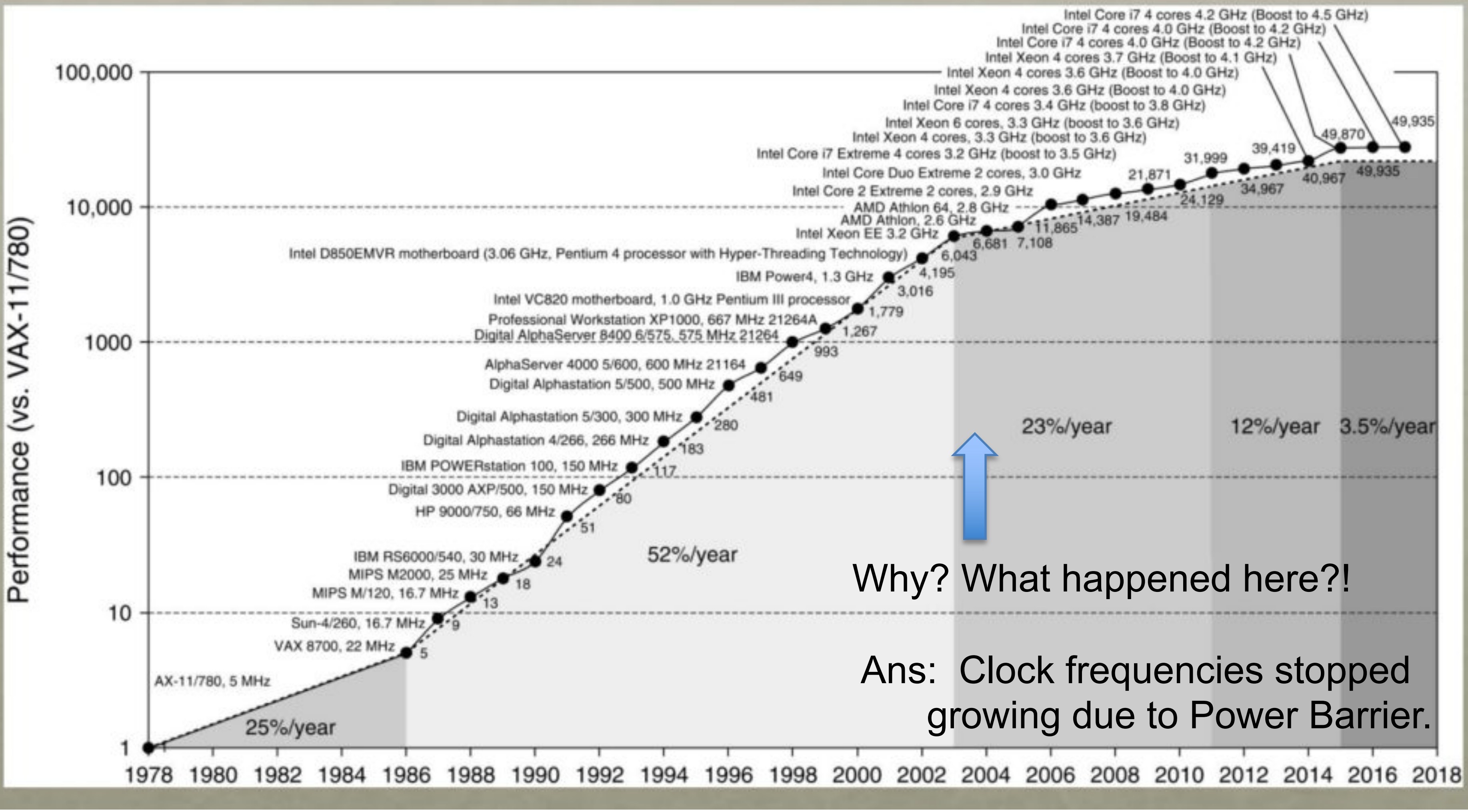
Moore’s law: Transistor density grew exponentially because of increasing integration factor. Density used to double every 2 years, satisfying a partly self-fulfilling prophecy of the chip industry.



CPU

UNIPROCESSOR PERFORMANCE (SINGLE CORE)

40 years of evolution

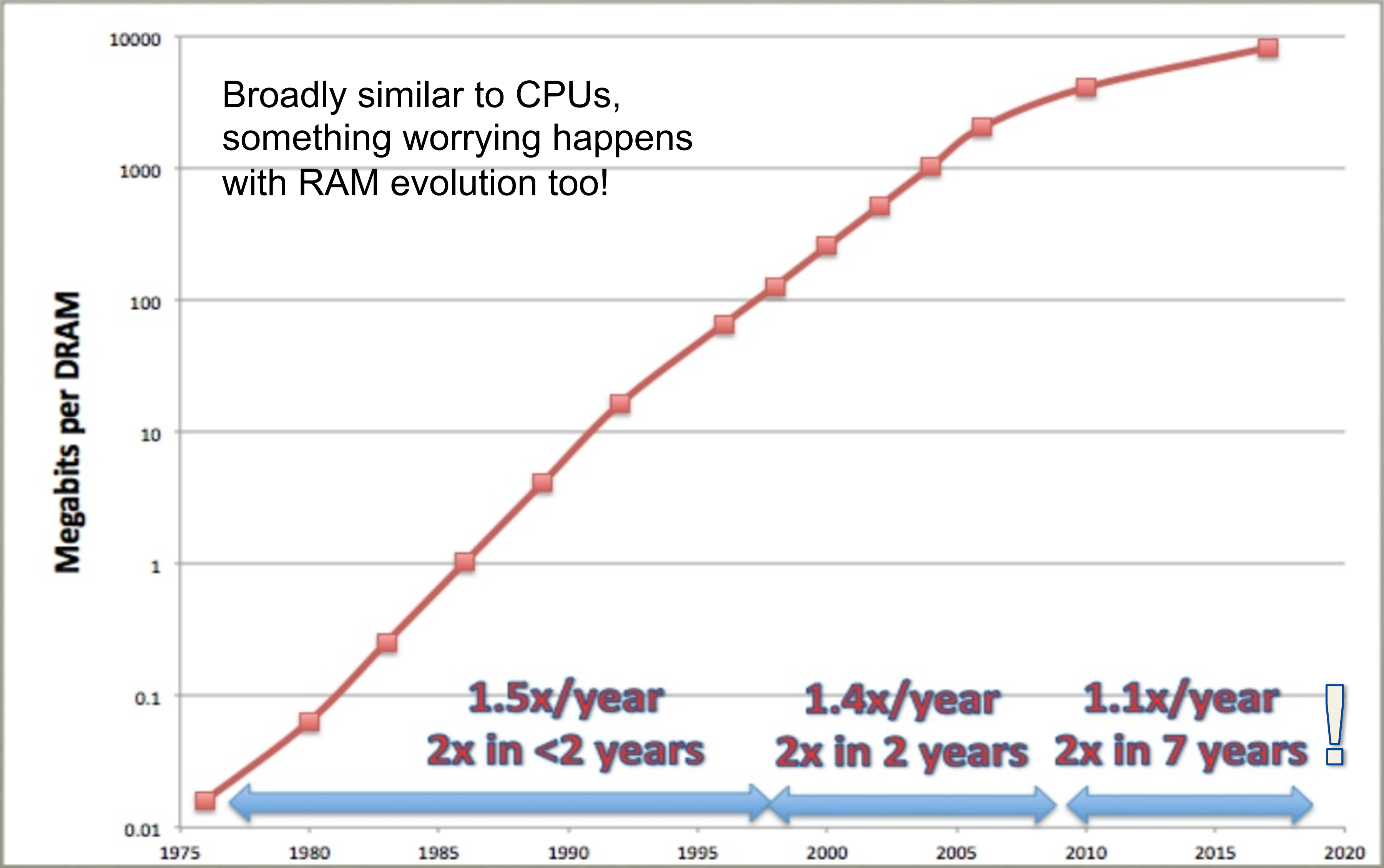


Why? What happened here?!

Ans: Clock frequencies stopped growing due to Power Barrier.

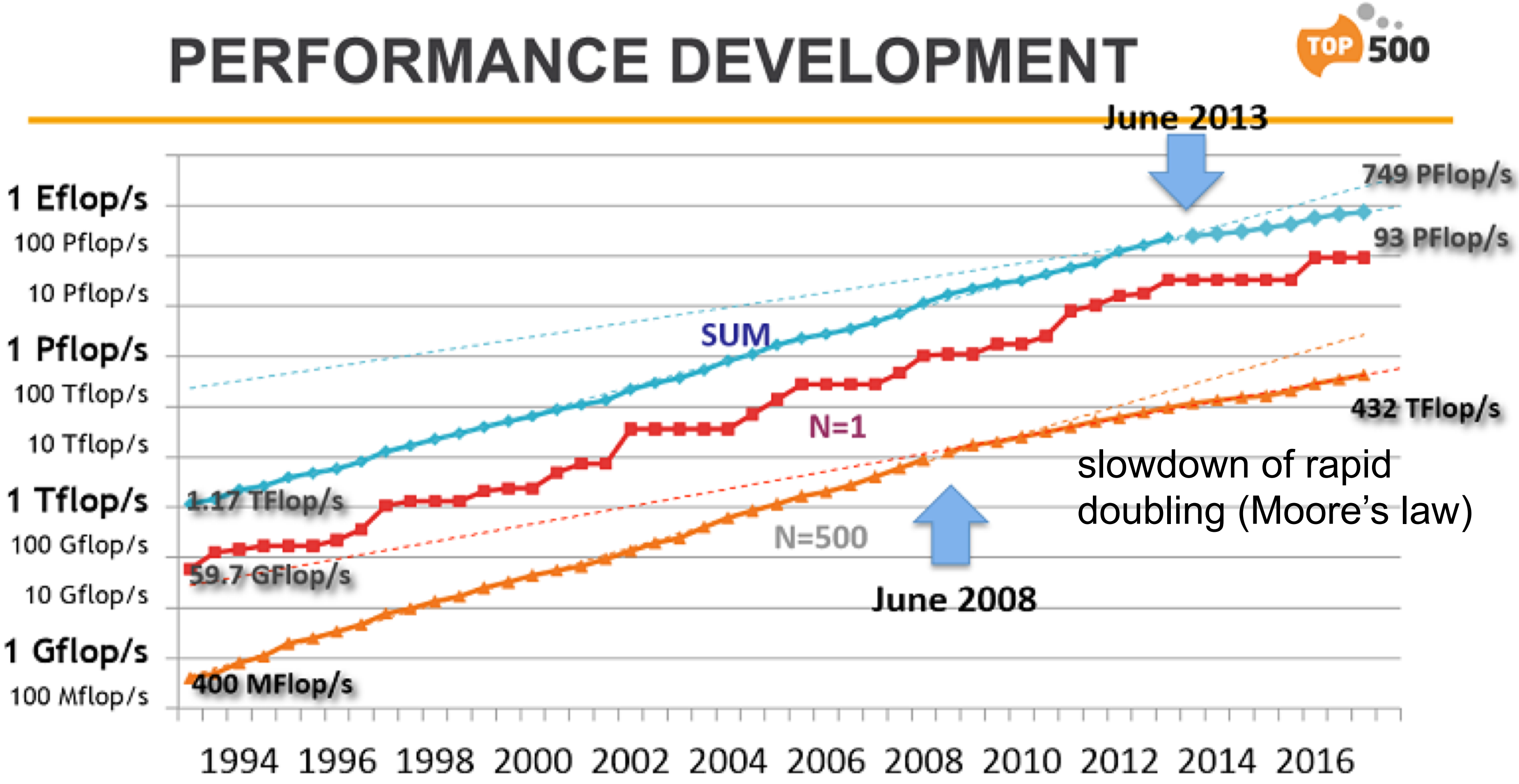
MOORE'S LAW IN DRAMS (memory)

https://www.nextbigfuture.com/2019/02/the-end-of-moores-law-in-detail-and-starting-a-new-golden-age.html = a DARPA talk

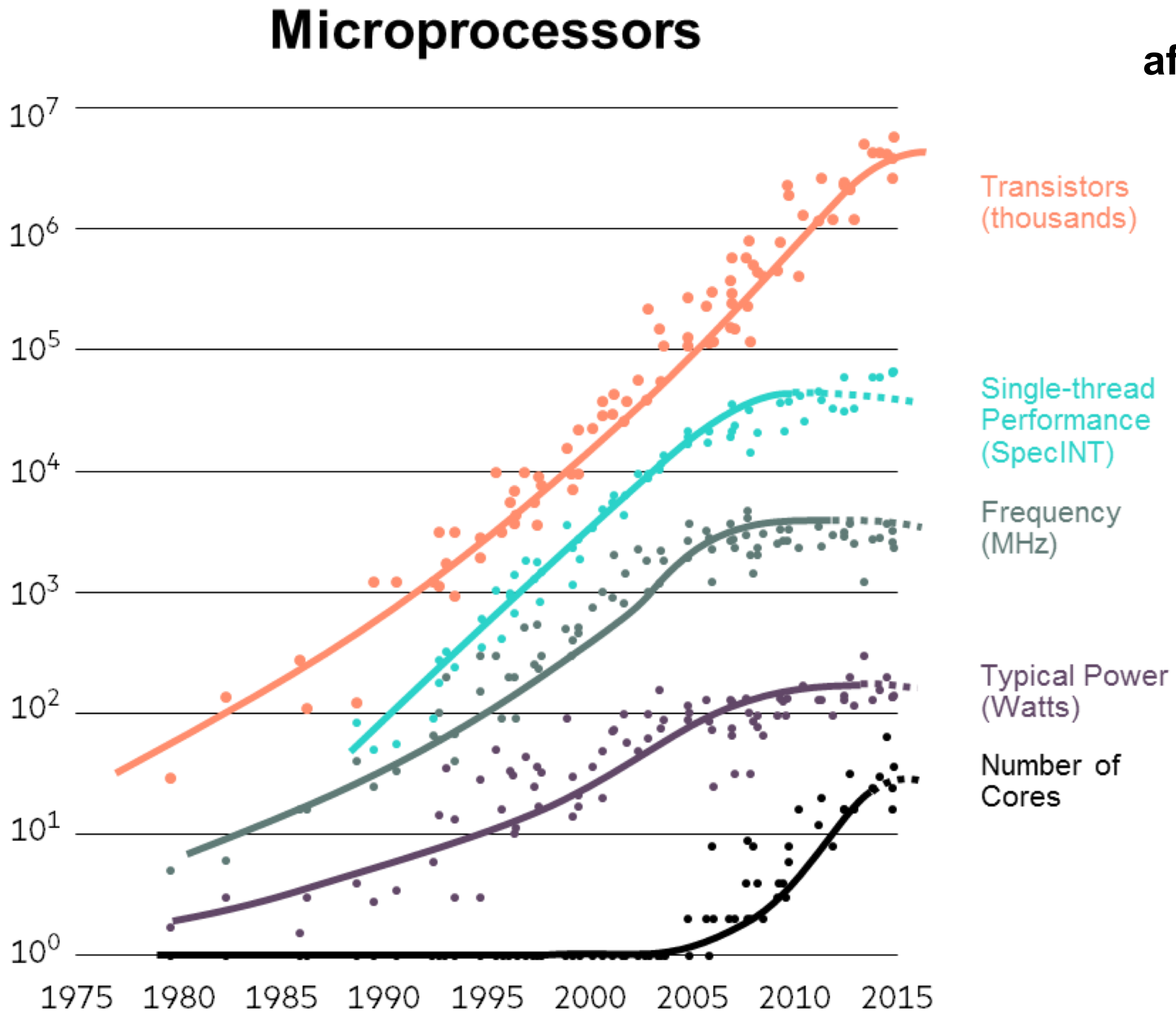


Broadly similar to CPUs, something worrying happens with RAM evolution too!

Something happened with the performance (speed) of top 500 supercomputers in the world. Since 2008-2013, it still grows exponentially but slower than before.



slowdown of rapid doubling (Moore's law)



after ~2004:

still goes up, a bit slower

stagnated

stagnated

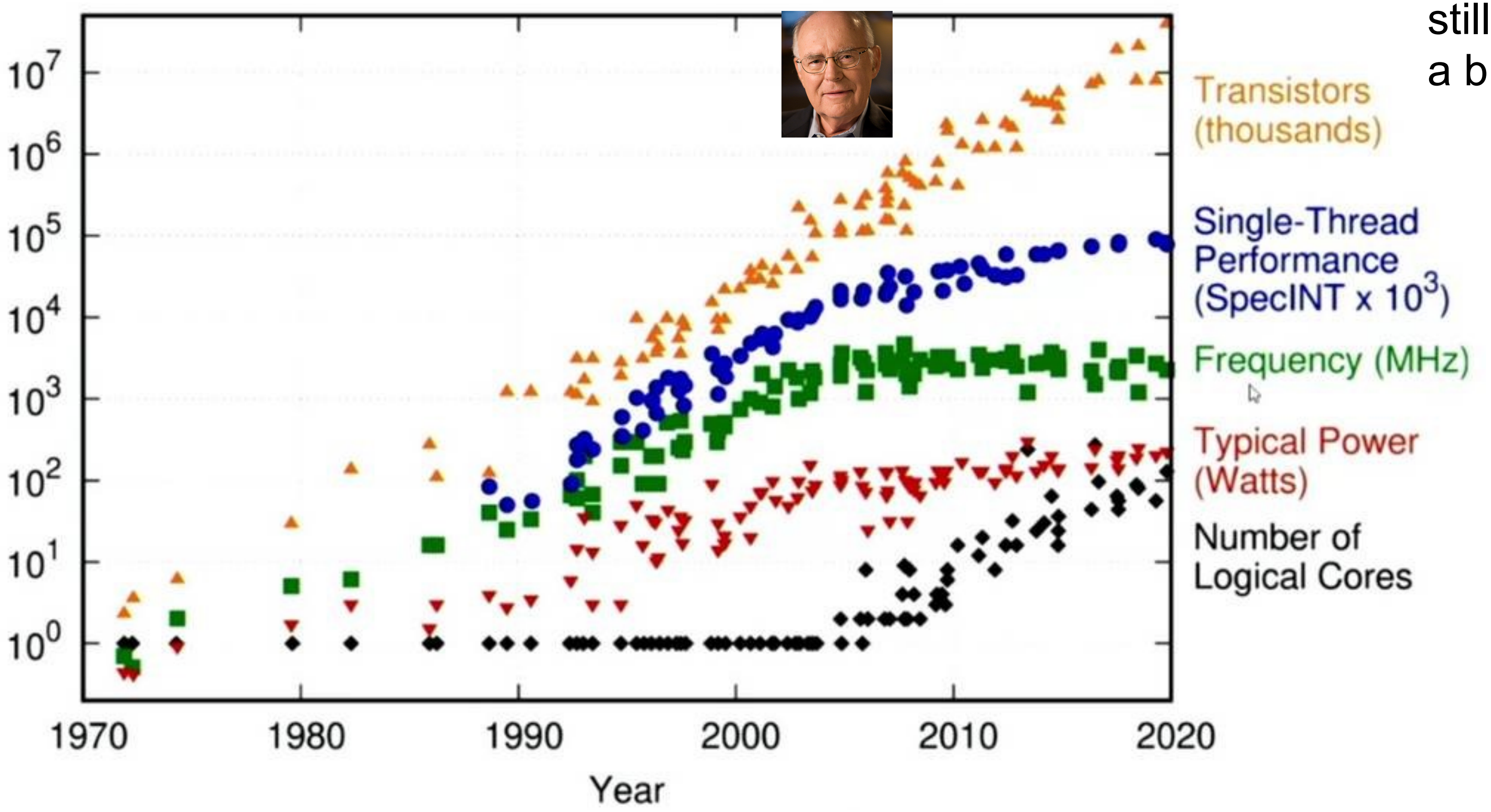
hit a wall

started rapidly climbing

after ~2004:

50 Years of Technology Scaling

48 Years of Microprocessor Trend Data



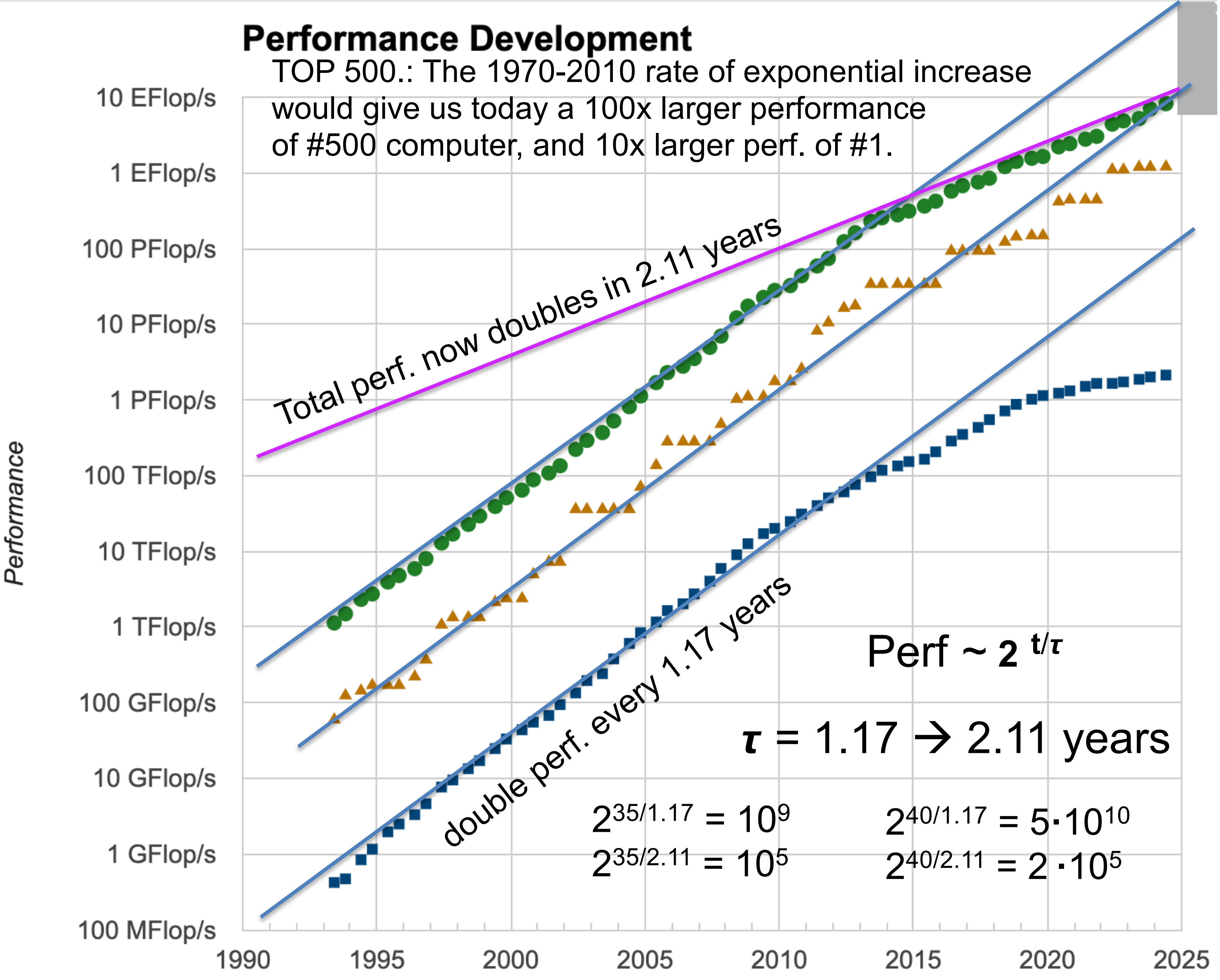
still goes up, a bit slower

stagnated

stagnated

hit a wall

started rapidly climbing



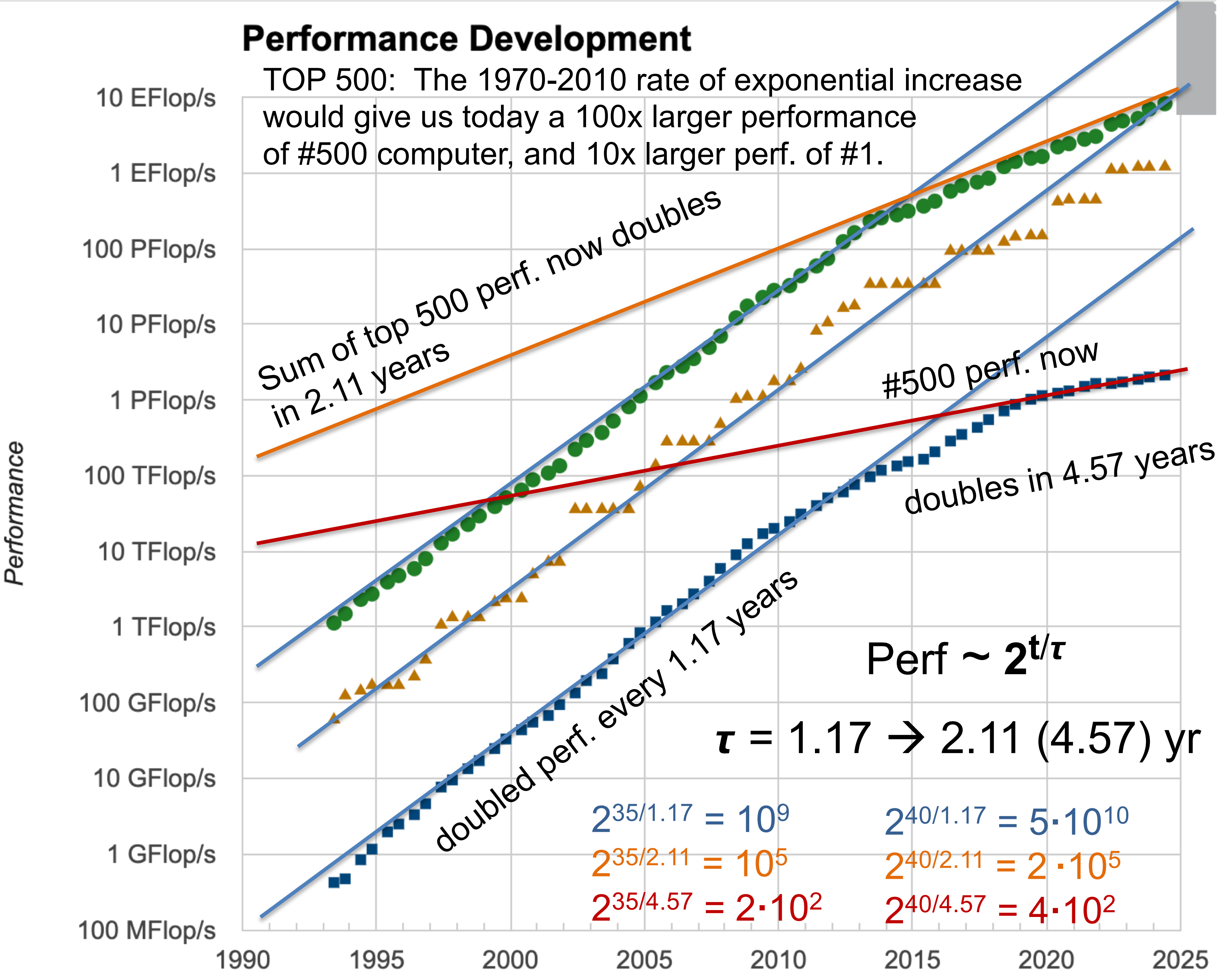
TOP 500.: The 1970-2010 rate of exponential increase would give us today a 100x larger performance of #500 computer, and 10x larger perf. of #1.

$$\text{Perf} \sim \mathbf{2}^{\,t/\tau}$$

$$\tau = 1.17 \rightarrow 2.11 \text{ years}$$

$$2^{35/1.17} = 10^9 \qquad 2^{40/1.17} = 5{\cdot}10^{10}$$

$$2^{35/2.11} = 10^5 \qquad 2^{40/2.11} = 2\cdot10^5$$



TOP 500: The 1970-2010 rate of exponential increase would give us today a 100x larger performance of #500 computer, and 10x larger perf. of #1.

Sum of top 500 perf. now doubles in 2.11 years

#500 perf. now doubles in 4.57 years

doubled perf. every 1.17 years

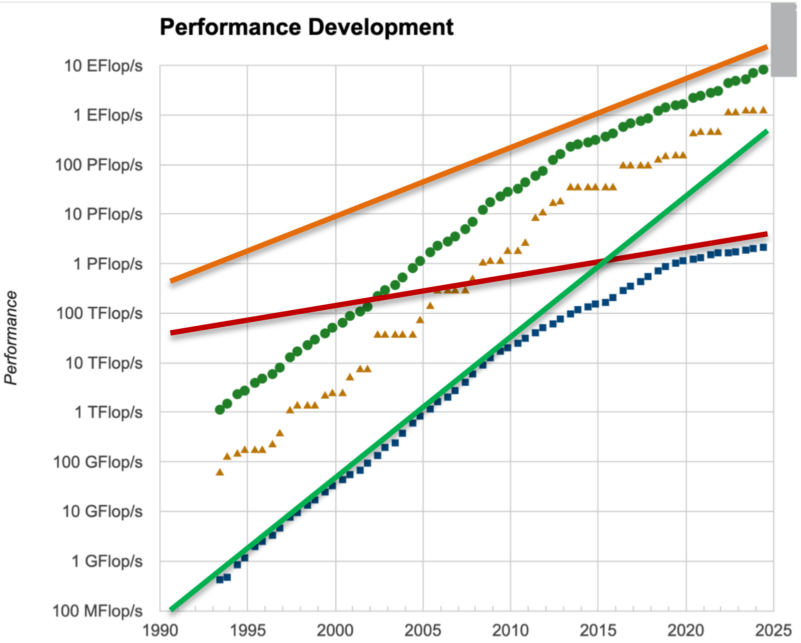
$$\text{Perf} \sim \mathbf{2^{t/\tau}}$$

$$\tau = 1.17 \rightarrow 2.11\ (4.57)\ \text{yr}$$

$2^{35/1.17} = 10^9$ $2^{40/1.17} = 5 \cdot 10^{10}$

$2^{35/2.11} = 10^5$ $2^{40/2.11} = 2 \cdot 10^5$

$2^{35/4.57} = 2 \cdot 10^2$ $2^{40/4.57} = 4 \cdot 10^2$



We had been very lucky!

$$\mathrm{Perf} \sim 2^{t/\tau}$$

$\tau = 1.17$ yr → 2.11 (4.57) yr

pre-2010 post-2010

Had the current slowdown occurred at the beginning of the era of integrated circuits in 1975, there would have been 40 years of slow instead of fast evolution.

That would produce a much much smaller growth factor:

$2^{40/1.17} = 5\cdot10^{10}$

$2^{40/2.11} = 2\cdot10^{5}$

$2^{40/4.57} = 4\cdot10^{2}$

Our computers & phones might now be 200000x or 100000000x slower, like they were around 1997 or 1988

1984

Nokia 3210
1989

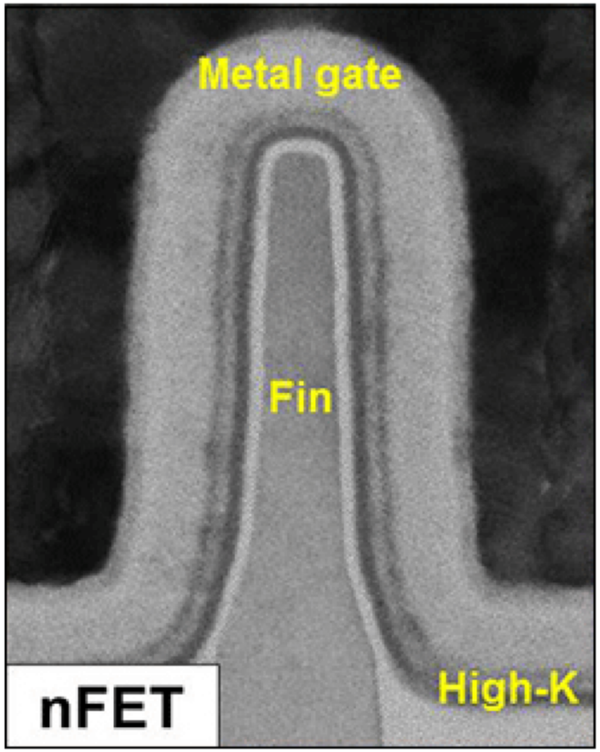
Early 1990s

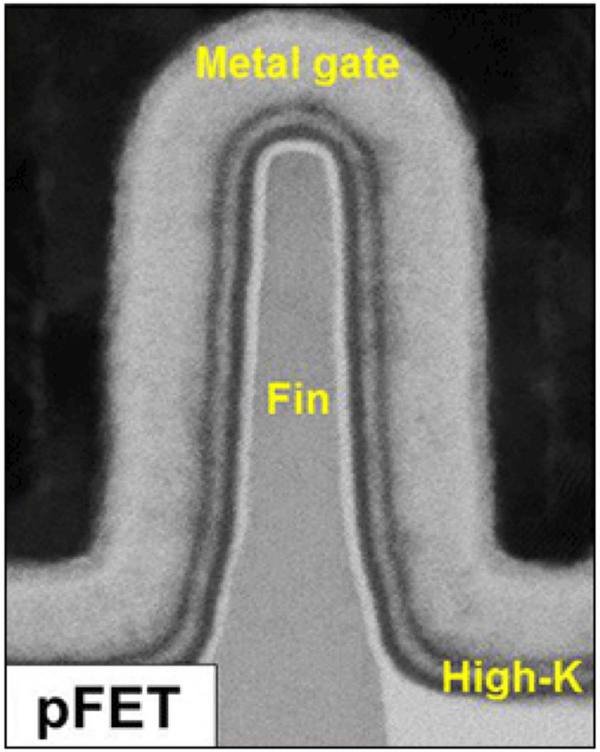
Motorola MicroTAC
1999

Understanding the changes in circuit production

This is how individual electronic elements look inside an Integr. Circuit (IC)
Their scale was deceasing by a factor of
$S=\sqrt{2}$
in each new generation if ICs, introduced roughly every 2 years.

The name like “28 nm” or “14 nm” or “5 nm technology” is related to the width of conducting lanes. The spacing between transistors is ~10 times bigger than the indicated number.





MOSFET =
Metal-Oxide
Semiconductor
Field-Effect
Transistor

technology also called
CMOS =
Complementary
Metal-Oxide Semicond.

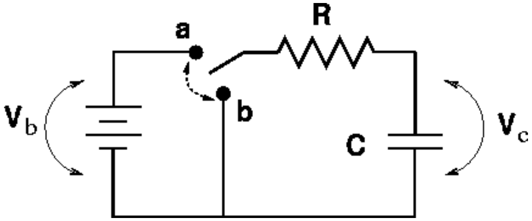
Moore's law says that the number of elements (logical gates, transistors) on an integrated circuit (on 1 cm^2 area), grows exponentially with time, such that it doubles every ~2 years. It was an empirical observation.

This was not a coincidence, but a combination of:

(i) repeated shrinkage cycles of circuitry etched onto silicon wafers, &

(ii) physics of materials, which after the shrinkage allowed to keep the *same amount* of electric power to the increasing number of logical gates

Let's prove it using physics of RC circuits.



A transistor on an Integrated Circuit has capacitance **C** and some stored charge Q. It also has resistance R, on which the current's energy can be dissipated. Transistors discharge and re-charge, under voltage **V**.

The process of charging/discharging is not for free; it dissipates energy equal to $\Delta E = CV^2$, half of it during charging, which is like pushing electrons up the growing potential hill. Energy drawn from voltage source by charge Q is $\sim QV = (CV)V = CV^2$. Usually all of that energy must be supplied but is wasted as heat during discharge through resistance R, however ΔE does not depend on R.

If the chip works with clock frequency **f**, then the rate of energy dissipation (power) is

$\mathbf{P \sim N\, C\, V^2\, f}$ + **N***(leakage current***V**) *Dennard's law*

where **N** is the number of transistors.

Now let's **shrink** the circuitry on an IC. Each side of the chip is constant but the pattern of circuits is shrunk by a factor S = √**2 =1.4128..** on a side, as it was in fact done every 2 years or so for 4 or 5 decades.

Q: What is happening to quantities C, V, and f of a transistor, and their number N?

N grows by a factor S^2 [1.41x transistors on each side of chip]

C grows by a factor $1/S$ 0.707x

[this follows from electrostatics: C ~ area of capacitor/spacing between electrodes ~$(1/S^2)/(1/S) = 1/S$]

V was increased by a factor $1/S$ 0.707x (because 'why not?')

f was increased by a factor S

[because (i) a smaller transistor can switch its state proportionally faster, and (ii) why not?]

We have estimated that power needed to do calculations scales like $P \sim N C V^2 f$.

P grows by a factor of: $S^2 (1/S)\ (1/S)^2\ S$ **== 1 (!)**

We have shown why ~17 cycles of shrinkage of transistors & circuits on a microchip over 4 decades required neither more efficient cooling nor more power supply to the chip. (Electric power requirement did in fact grow, as the IC's area slowly grew.)

The exponentially growing number N was utilized to complicate the logical structure of CPUs, e.g., by introducing more and more levels of *cache memory* inside the processor. This memory is much fast but much smaller than RAM. It can be used as a buffer between CPU and the outside operational memory (RAM).

This allowed to mask the fact that CPU-RAM communication and the speed of the RAM (memory) was growing slower than the processing needs of CPU. As a result, frequency f grew ~2 times, instead of just 1.41 in each new generation of processors.

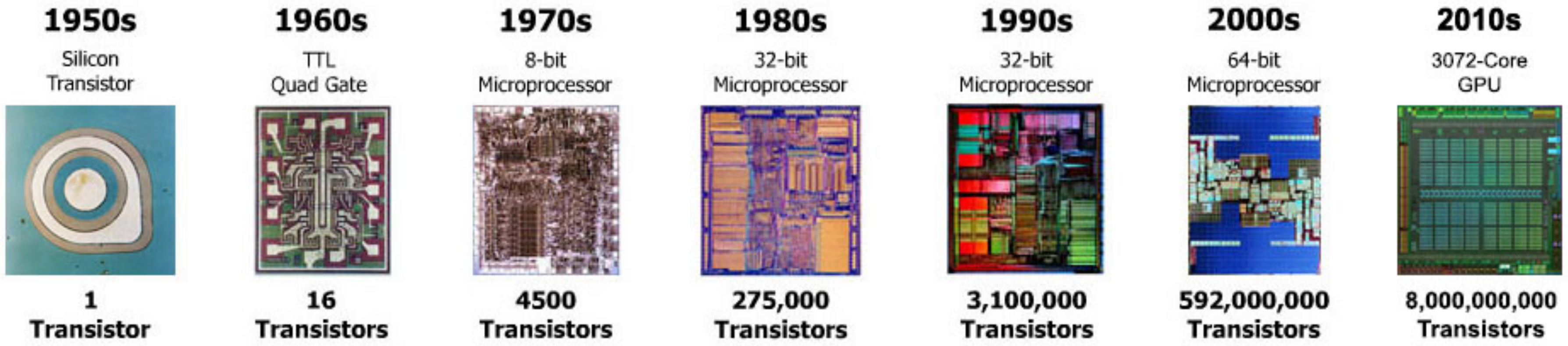
The progress *was* phenomenal. Among others, it allowed a similar exponential Internet bandwidth growth with doubling time ~18 months.

If our cars were increasing in speed at the rate equal to the increases in computing power, then we would now drive at about the speed of light, once around the Earth per second.

No other area of human activity saw a million-, let alone a billion-fold quantitative improvement in one human generation, but computing did.

It was Physics that allowed us to keep providing a similar power to faster, smaller processors. We wer fortunate that Dennard’s law was working.

So why did the size reduction and exponential growth of clock speed have to end around 2004?



Transistors inside the microchips, now counted in billions, became so small that, due to microscopic material imperfections, electric power gets dissipated in new, unavoidable way, via the *leakage current* in volatile memory (RAM, CPU).

Transistor should either block or transmit current, but now the blocked state became not 100% blocked. Electric energy is slowly going to waste, without carrying out any computation. That's how the doped semiconductors work.
Leaks occurred before, but were masked by much larger heat dissipation due to (useful) toggling from state "0" to state "1", and from "1" to "0", during binary data processing.

Now: $\mathbf{P} \sim \mathbf{N\, C\, V^2\, f} \; + \mathbf{N}\, (\mathbf{I_{leak}\, V})$ *Dennard's law*

(i) size of circuitry still diminishes (with manufacturing problems! ASML lithography ok)
(ii) number of transistors in new generations of processors still grows (a bit slower)
(iii) f ~ const. ~2-5 GHz (speed of 1 core of CPU inches forward *imperceptibly*)
in order to try to keep energy supply and dissipation at a reasonable level!
(iv) Cost of manufacturing next chip technology increases very rapidly, and the number of microchip foundries decreased to 2 or 3. Will the 3nm technology cycle be the last?

Post-Dennard era, leakage current contributes about the same as the re-charging power losses:

Shrink the circuitry as before by a factor **$\sqrt{2}$ =1.4128..**

Q: What is happening to quantities C, V, and f of a transistor, and their number N?

N still grows by a factor ~2

C grows by a factor $1/\sqrt{2}$

[electrostatics; C ~ area of capacitor/spacing, as before]

V has to be kept constant (~1 Volt) ~1

f has to practically stay constant ~1

We have estimated that power needed to do calculations scales like

$$P \sim N\,C\,V^2 f + N\,V\,I_{leak}$$

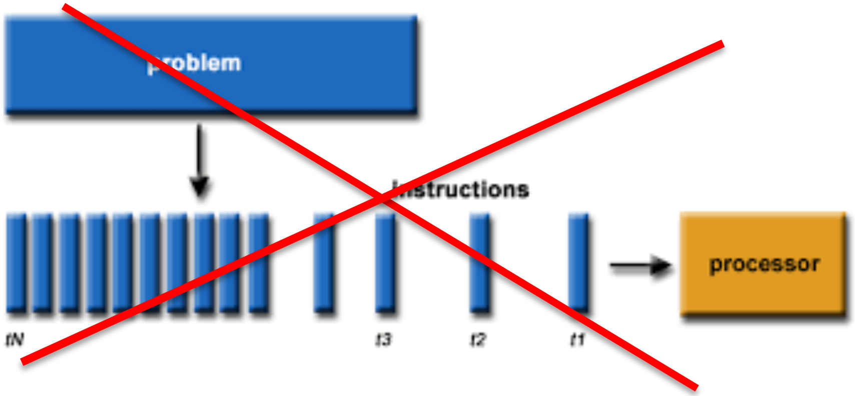
The 1st term grows by a factor $2\ [1/\sqrt{2}]\ 1^2\ (\sim 1)$ **~ 1.4x, the second likely too**

Dark silicon: switching off temporarily unused transitors. In your phone and laptop there are so-called fast and slow CPU cores → overall ~1 (?)

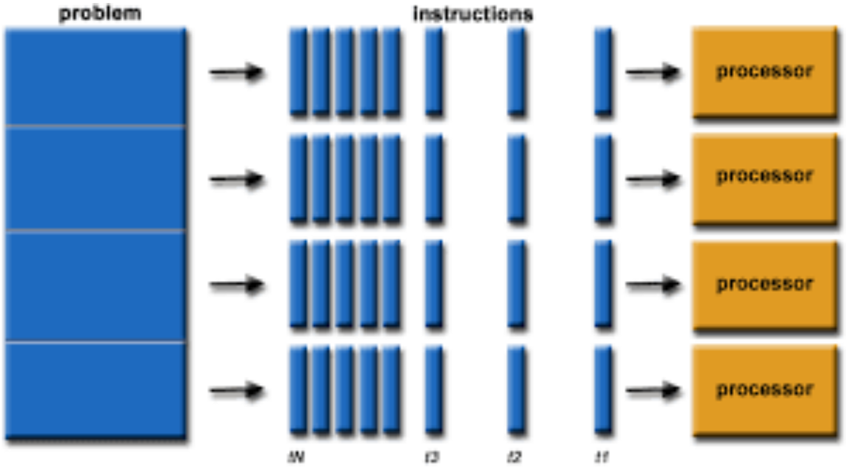
https://www.hs-augsburg.de/Binaries/Binary20963/PostDennard.pdf

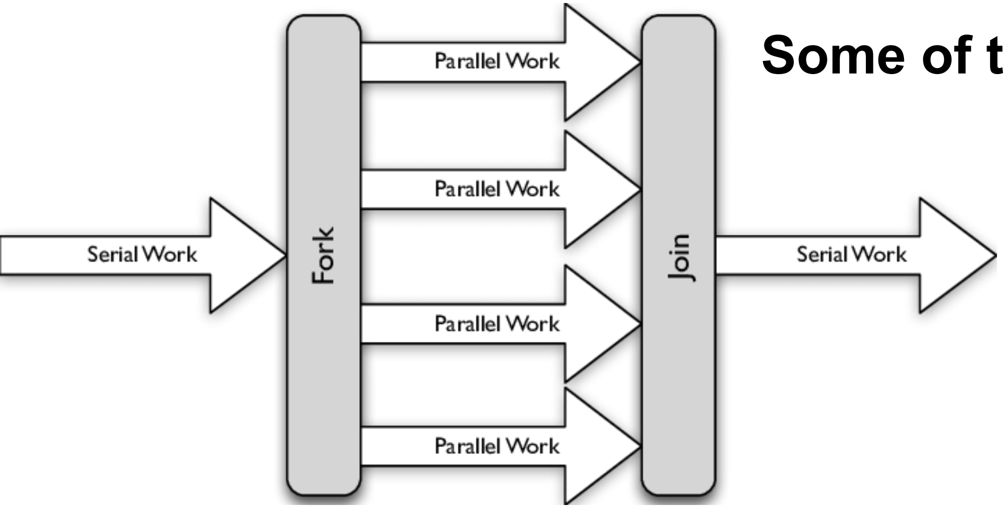
- The problems boil down to the power barrier, although there is also a rising manufacturing difficulty and cost. So the wait for 2x performance lengthens...
- It is no longer possible to play the game of “increase clock speed and keep one core”, as before. On the contrary, increasing the number of cores in a processor comes cheap (power-wise), and therefore is now practiced by manufacturers.
- We are (and will be, in the future) getting processors with more and more cores. Each core will NOT be much more capable to crunch numbers than in the previous generation. Hardware engineers can’t keep up doing miracles. They hope that **you** as a programmer will do them, adjusting to the changing world.
- Power barrier forces us to use more and more cores **in parallel**, if we hope to practice HPC and computational branch of all science
- This task is significantly more difficult in Python than e.g. in C, Fortran, or Julia. Therefore we are going to learn parallelization and other, less effective speedup methods, such as vectorization, in the rest of this course.

Serial program & execution 😐
(try to avoid!)



Parallel program: concurrent execution ☺





Some of the modern programming methods:

If you find a fork in a road,
take it.
'Yogi' Berra

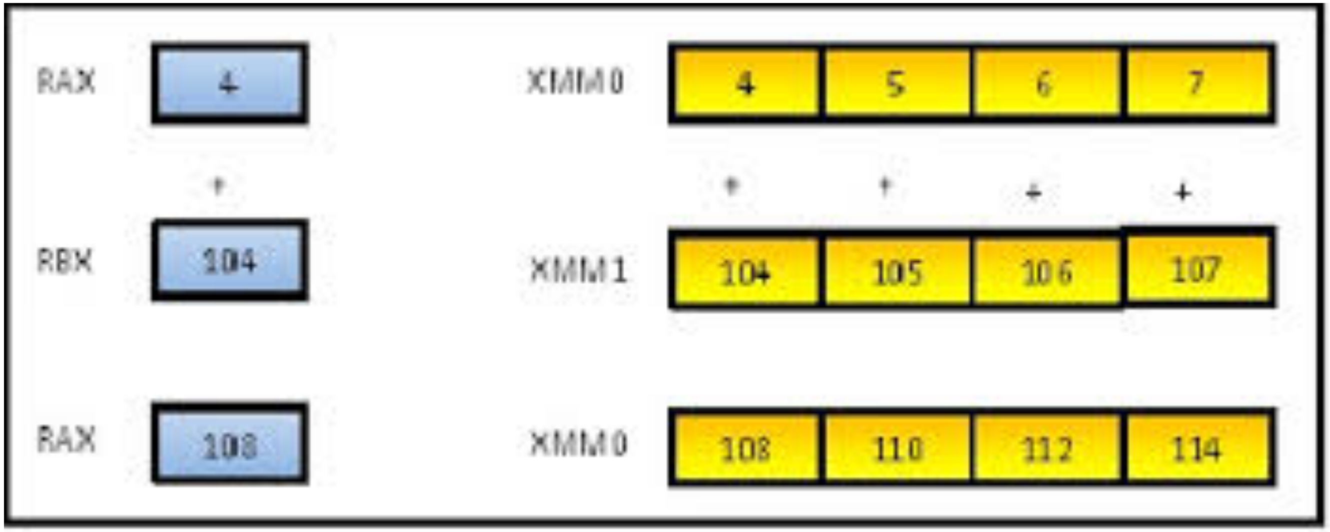
1. One of the most successful **parallelization** paradigms is the fork-join scheme (on all platforms; Message Passing Interface = MPI, essential on supercomputers, OpenMP mostly in shared memory sys.: **multithreading** on many cores).

2. Every thread may, in addition, be **vectorized,** using a growing set of long AVX registers (on CPU, MIC, currently mostly 256 bit = 8 single prec. or 4 dp floats)

Vectorization:

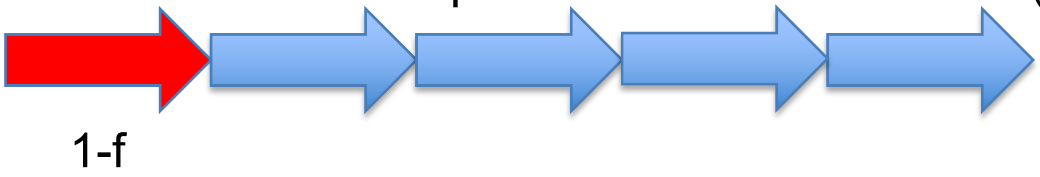
- manual (intrinsic C/C++ functions)

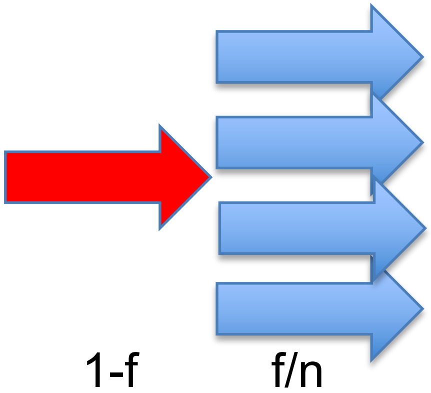
- automatic (compiler + SIMD directives)



Derivation of **Amdahl's law** of parallel program's speedup

f = fraction parallelizable = 4/5 = 80% (for example!)





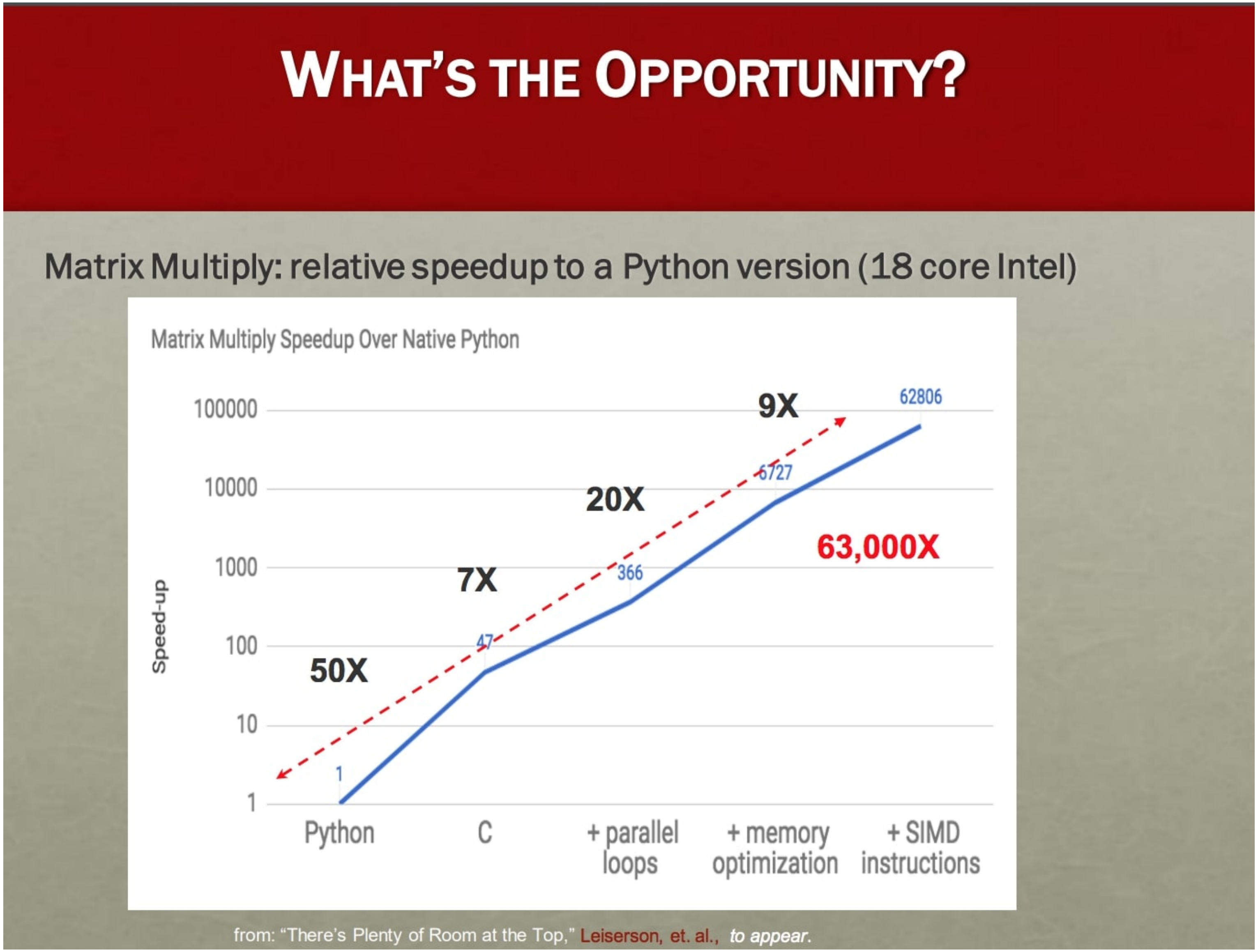
$n = 4$ (# of threads)

S = Speedup factor = 5/2 = 2.5x

In general: $S = 1 : (1-f + f/n) < 1/(1-f)$

e.g., if $1-f = 1/20$ then $S < 20$ even if $n = 512$

We have already seen an example of 40x speedup over Python possible with Numpy, and 40x over Numpy with Fortran (OpenMP & CUDA), Here's someone's success story with the matrix multiplication algorithm, which has f~1 (is ~fully parallelizable)



WHAT'S THE OPPORTUNITY?

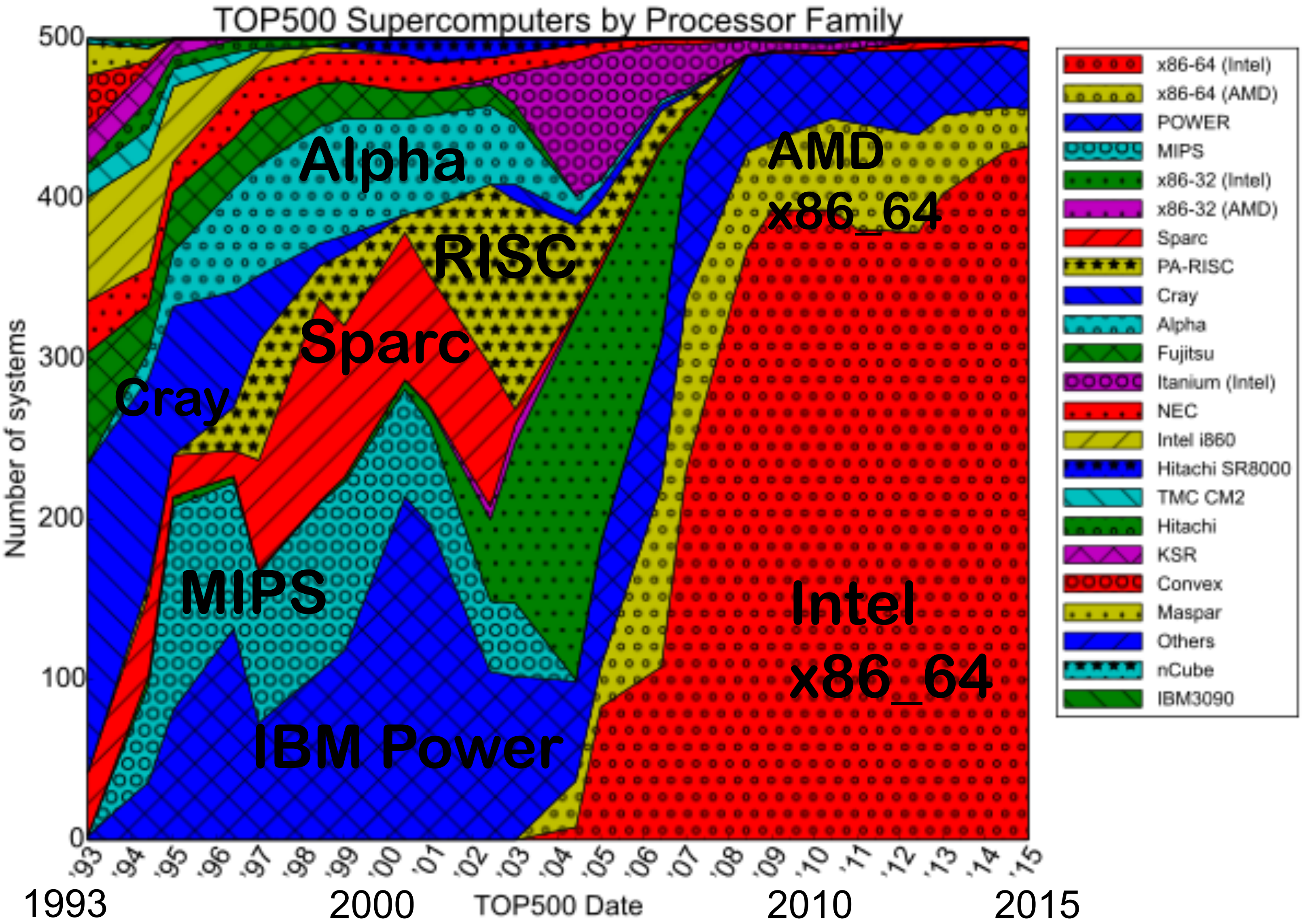
Matrix Multiply: relative speedup to a Python version (18 core Intel)

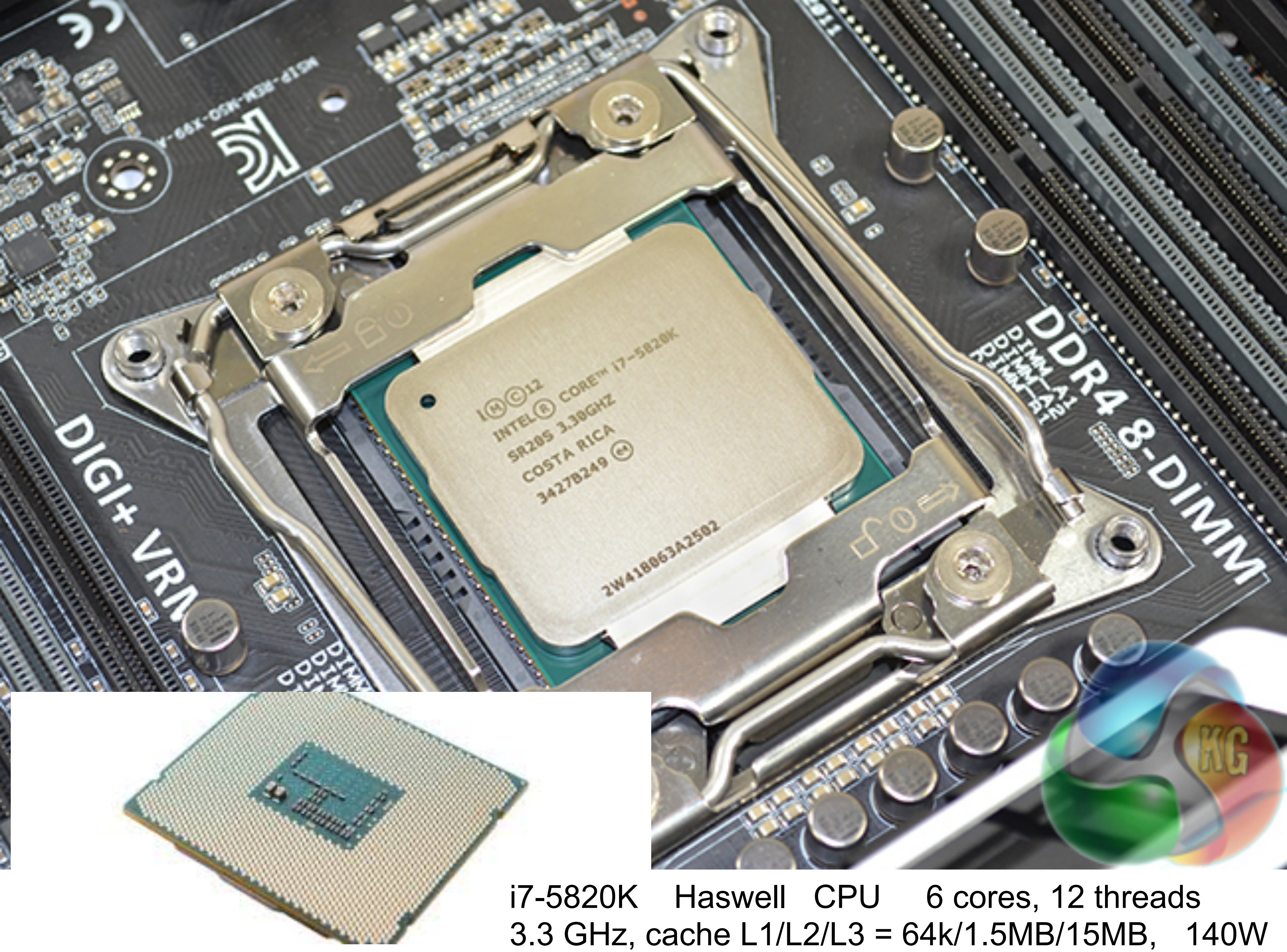
from: "There's Plenty of Room at the Top," Leiserson, et. al., *to appear.*

CLUSTERS, SUPERCOMPUTERS:
additional, final boost by their hardware's massive scale.
But these beasts should run efficient, smart codes.

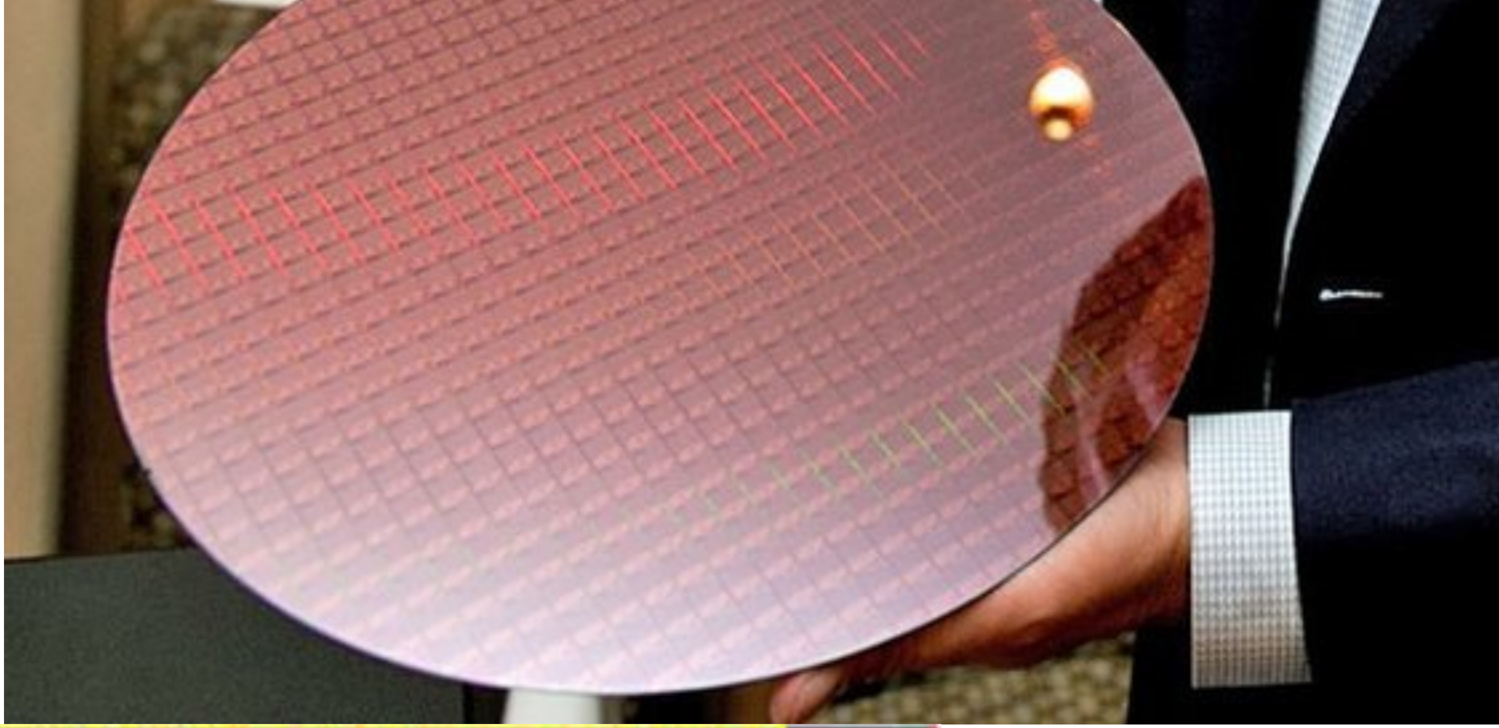
# stars ~ # cores
in supercomputer

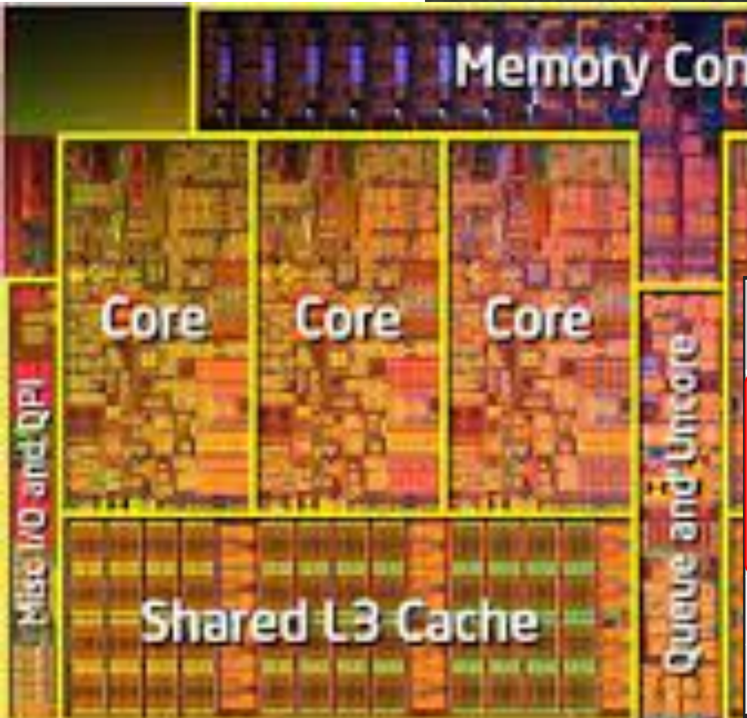
Processor architecture & instruction set: Out of creative chaos of different CPUs before ~2004, Intel's x86_64 processors emerged as a big winner. Your CPU and CPUs in supercomputers, until 2019 were likely all x86_64 microprocessors. They understand the same instruction set. Here is the full story, in 1993-2015:

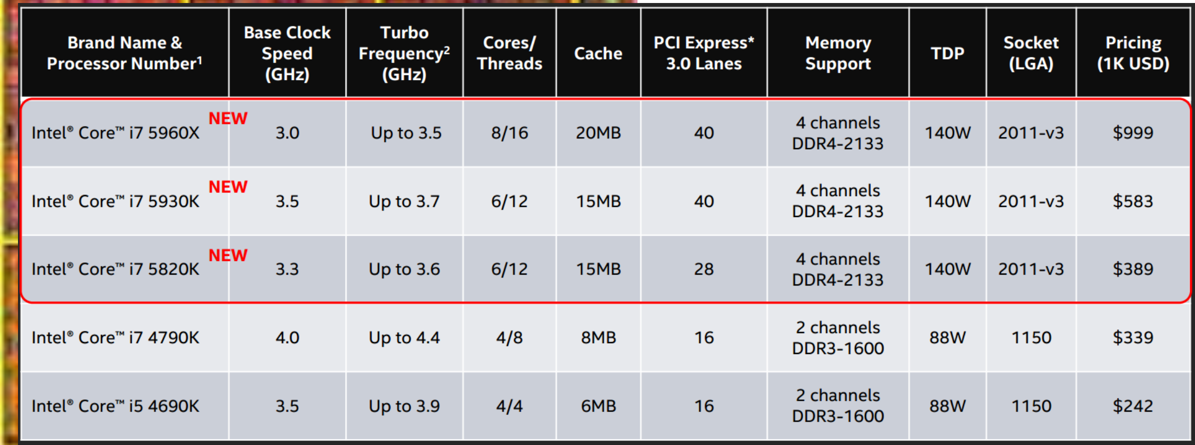




i7-5820K Haswell CPU 6 cores, 12 threads
3.3 GHz, cache L1/L2/L3 = 64k/1.5MB/15MB, 140W

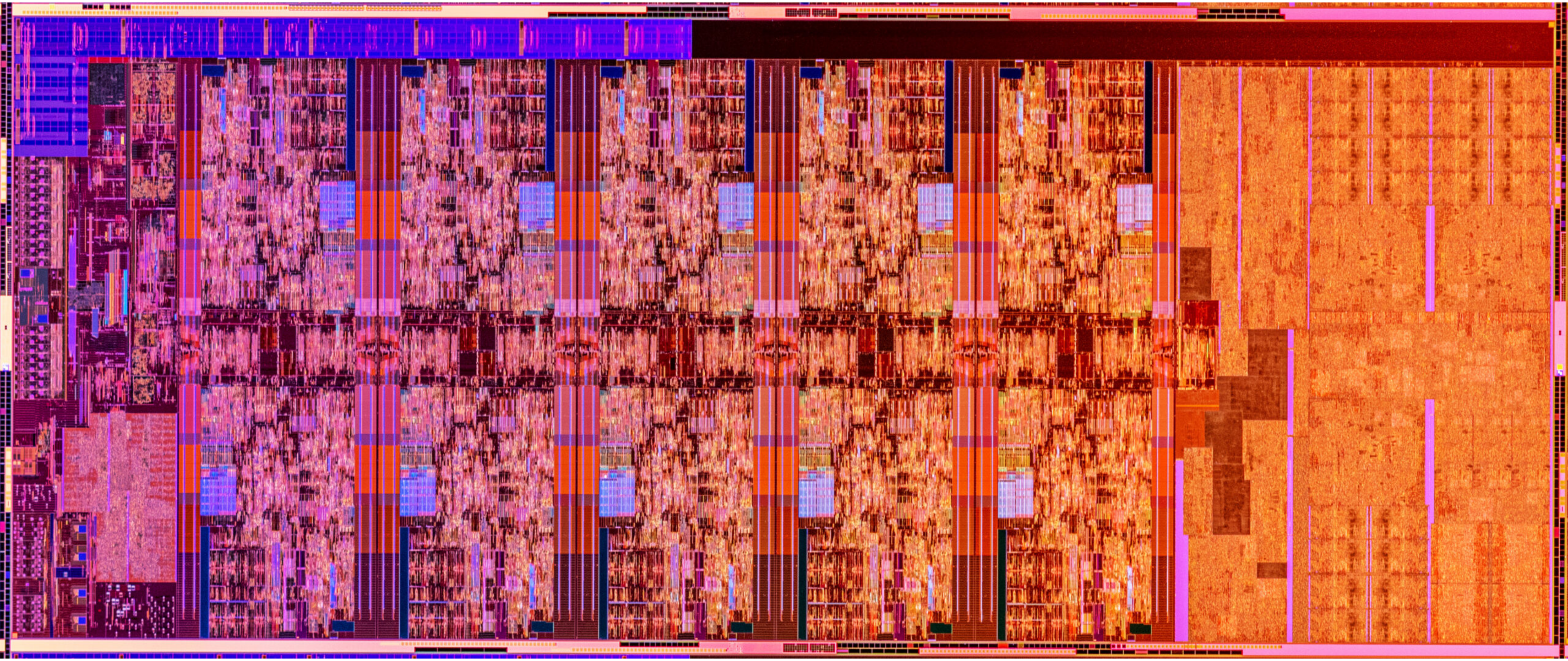




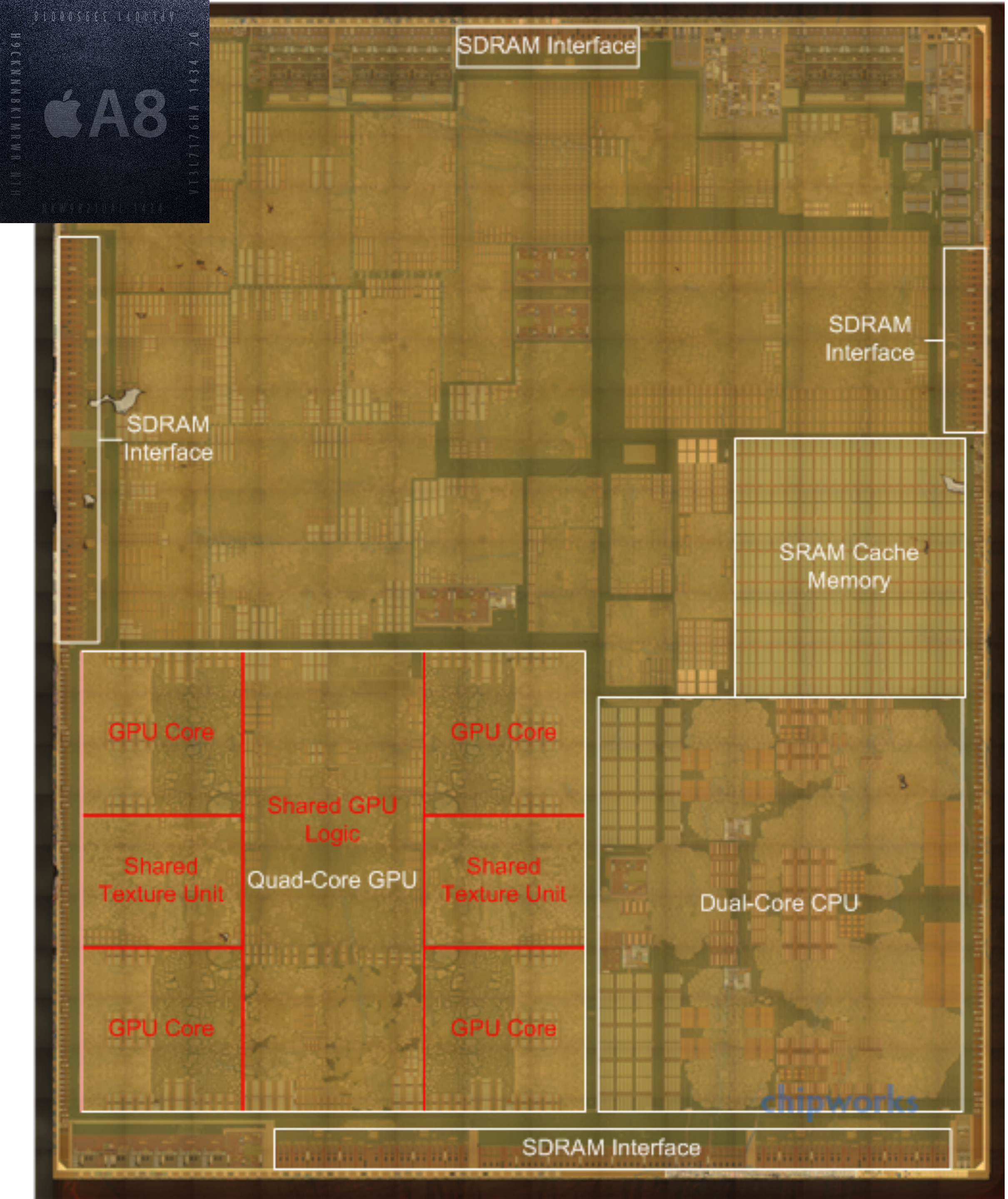


Brand Name & Processor Number[1]	Base Clock Speed (GHz)	Turbo Frequency[2] (GHz)	Cores/ Threads	Cache	PCI Express* 3.0 Lanes	Memory Support	TDP	Socket (LGA)	Pricing (1K USD)
Intel® Core™ i7 5960X NEW	3.0	Up to 3.5	8/16	20MB	40	4 channels DDR4-2133	140W	2011-v3	$999
Intel® Core™ i7 5930K NEW	3.5	Up to 3.7	6/12	15MB	40	4 channels DDR4-2133	140W	2011-v3	$583
Intel® Core™ i7 5820K NEW	3.3	Up to 3.6	6/12	15MB	28	4 channels DDR4-2133	140W	2011-v3	$389
Intel® Core™ i7 4790K	4.0	Up to 4.4	4/8	8MB	16	2 channels DDR3-1600	88W	1150	$339
Intel® Core™ i5 4690K	3.5	Up to 3.9	4/4	6MB	16	2 channels DDR3-1600	88W	1150	$242

Intel Alder Lake 10-core desktop CPU



socket : LGA 1200
area : 200 mm^2



A8 SoC (system on a chip)
designed by Apple Inc. for:
iPad mini 4, iPhone 6(+)

produced by TSMC
(Taiwan Semiconductor)

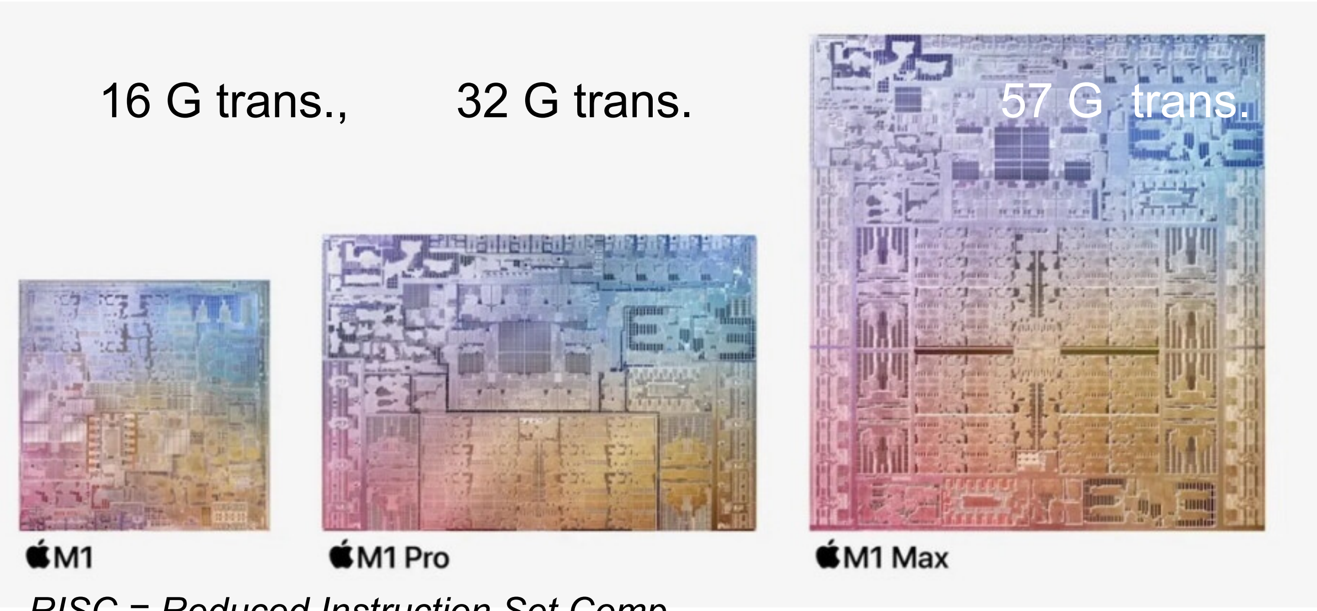
process: 20 nm

area : 89 mm^2
# of transistors: 3 billion

instruction set: ARMv8-A

dual CPU: 64 bit, 1.5 GHz
caches: L1 / L2 / L3
64+64kB/2MB/4MB
1.1 – 1.4 GHz clock freq.

GPU: 8 cores,
148/296 GFlops (FP32/FP16)



M1 Max processor
5 nm process (N5)
10-core CPU,
32-core GPU

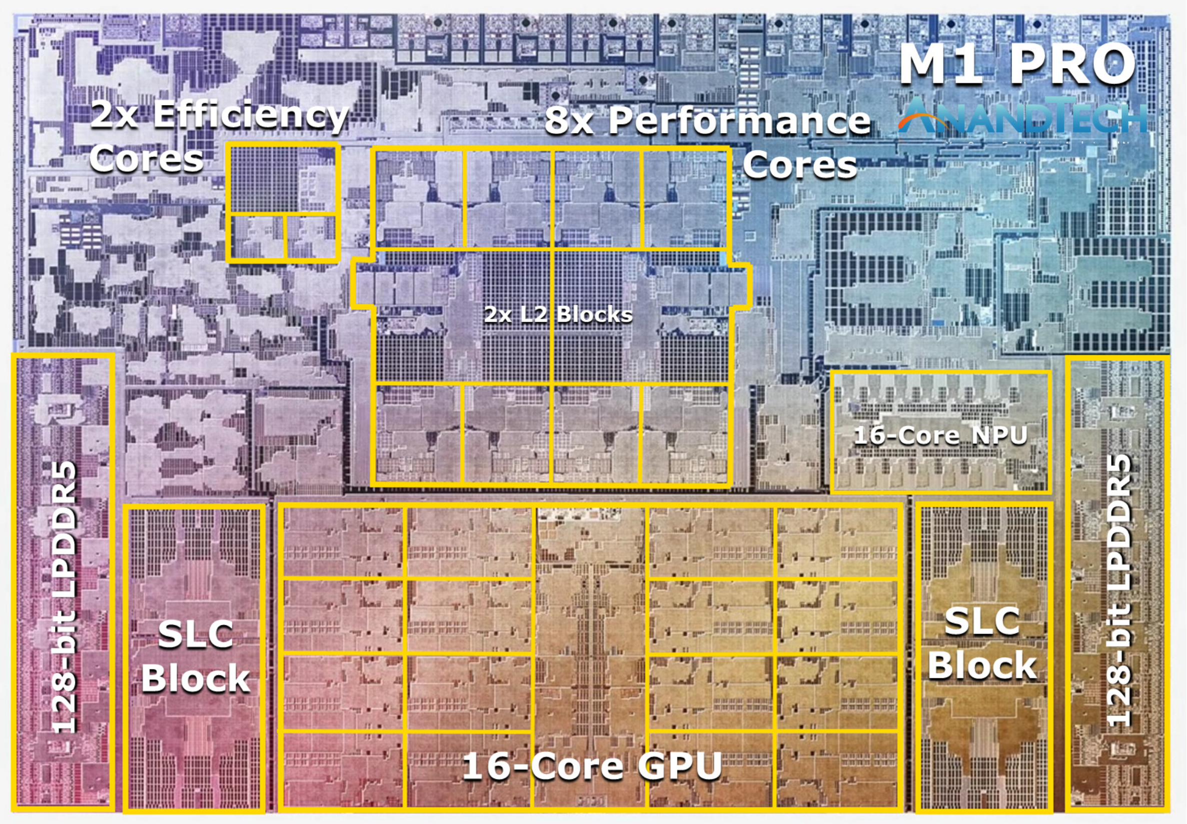
64 GB unified memory
400 GB/s bandwidth

RISC = Reduced Instruction Set Comp.
RISC proliferated after 1990 (when it was introduced in Sun Microsystems workstations). Used a reduced instruction set for faster code execution. Sun also had the top compilers.

Apple used CPU with:
Motorola (1980s), PowerPc (1990s), Intel (2000s), and ARM (2020s).

ARM = (orig.) Advanced RISC Machines
An architecture different from x_86 Intel, It's a 40-year old architecture(!). In 2000s used in 95-98% of smartphones.

In 2024, the top version of M2 Apple Processor (ARM) has a record # of transistors: **134 G trans.**



OpenMP = Open Multiprocessing (cf. wiki)

"Hands-on intro to OpenMP", a slide show by Intel programmers from SC08, Austin, TX

https://www.openmp.org/wp-content/uploads/omp-hands-on-SC08.pdf

Full description of OpenMP (fairly tedious reading but good to have for reference)

https://www.openmp.org/wp-content/uploads/openmp-4.5.pdf

Examples of good and erroneous application of OMP. Lots of interesting code snippets in C/C++ and Fortran:

https://www.openmp.org/wp-content/uploads/openmp-examples-4.5.0.pdf

Intel’s writeup about performance limitations of OpenMP-instrumented codes.

https://software.intel.com/en-us/articles/performance-obstacles-for-threading-how-do-they-affect-openmp-code

Our code page http://planets.utsc.utoronto.ca/~pawel/PHYD57/
has links to program tetra*.f90/.f95 that solves a computationally demanding task

OpenMP

(Tutorial T4 material)

Our code page http://planets.utsc.utoronto.ca/~pawel/PHYD57/
&
Your account phyd57@art-1.utsc.utoronto.ca:/home/phyd57/progD57
as well as phyd57@art-1.utsc.utoronto.ca:/home/phyd57/math
have links to many programs using omp.

- tetra*.f90/.f95 which solves a comp. demanding task of finding the chance that a random tetrahedron inscribed in a sphere contains the sphere's center
- buckets-1000.f90 tries to find the optimum way of distributing a given amount of water in each move into 8 buckets, 2 fullest of which get emptied, so that maximum amount is reached in one of the buckets.
- There is always something to play with, e.g.
- vary num_threads, or schedule, or collapse, options in OMP directives
- Also, come back to the iterated 2-D Laplace operator in cudafor-laplace3-dp.f95 and cudafor-laplace3-sp.f95. These codes contain sequential and parallel parts: host code, OMP, and a comparison with CUDA Fortran.

- **Tutorial** programs for Fortran and C are on our web page.
  Use the last link in the Languages section to review the usage of C.

- Programs are placed in /progD57 directory and in art-1 program directory

- Look at C-version of arithmetic/bandwidth demonstration program simpler-nb-3ra.c, which does not work (produces Segmentation fault, problem with memory management, as we say this code 'segfaulted'). The reason is that the program tries to deposit results in a non-existing array element tab[6][k] . This needs to be changed to tab[5][k] because 6 indices of the 1st dimension are: 0,1,..,5. (C, like Numpy, has 0-base arrays!)

- simpler-nb-3ra.f90 is the corrected F90 program, and the version with swapped indices is called simpler-nb-3da.c

- simpler-nb-3da.f90 and simpler-nb-3ra.f90 are the Fortran versions, which we will analyze during day 7.