NYU New York University

Multicore Processors: Architecture & Programming

Multicore/Manycore Revolution

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com

Who Am I?

- Mohamed Zahran (aka Z)
- http://www.mzahran.com
- Research interest:
  - computer architecture
  - hardware/software interaction
- Office hours:
  - online: zoom link on Brighspace and course website
  - Wed 2-3pm
  - or we can set an appointment to meet online some other time if you cannot make it to office hours

Goals of This Course

- What are multicore/manycore processors?
- Why do we have them?
- What are the challenges in dealing with them?
- How to make the best use of them in our software?
- What will the future likely be for both hardware and software? and how to prepare for it?

My wishes for this course

- Don’t be afraid of hardware.
- Understand the hardware/software interaction.
- Enjoy the challenge of making the best use of hardware to the benefit of software.
- Enhance your way of thinking about parallelism and parallel programming models.
- Build a vision about technology and its future.

Grading

Homework assignments	25%
Programming assignments	25%
Project	50%

Policies: Assignments/Labs

- You must work alone on all assignments
  - Post all questions on discussion section of Brightspace
  - You are encouraged to answer others' questions but refrain from explicitly giving away solutions.
- Hand-ins
  - All assignments submitted before the deadline through Brightspace.
- To discuss an assignment grade:
  - You have one week from the grade release.
  - Discuss it with the grader first.
  - If the issue is not resolved, bring it to me.

Policies: Project

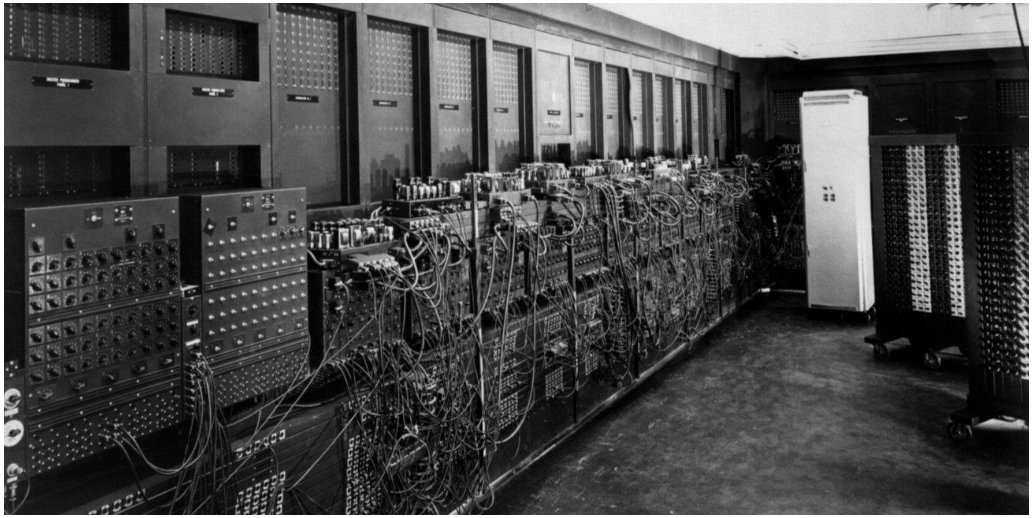
- Will post suggested projects on the course website.
- You can:
  - Pick one of the suggested projects.
  - Pick one of the projects but suggest a modification.
  - Suggest a project yourself to work on.
- You can work alone or in a group of up to four students.

Web Presence

- Course website:
  - lecture slides
  - reading material
  - project information
  - important links
- Brightspace:
  - Discussion forums
  - Announcements: from instructor or grader
  - submitting assignments and projects
  - getting assignment/labs grades

First … A bit of history

Computer History



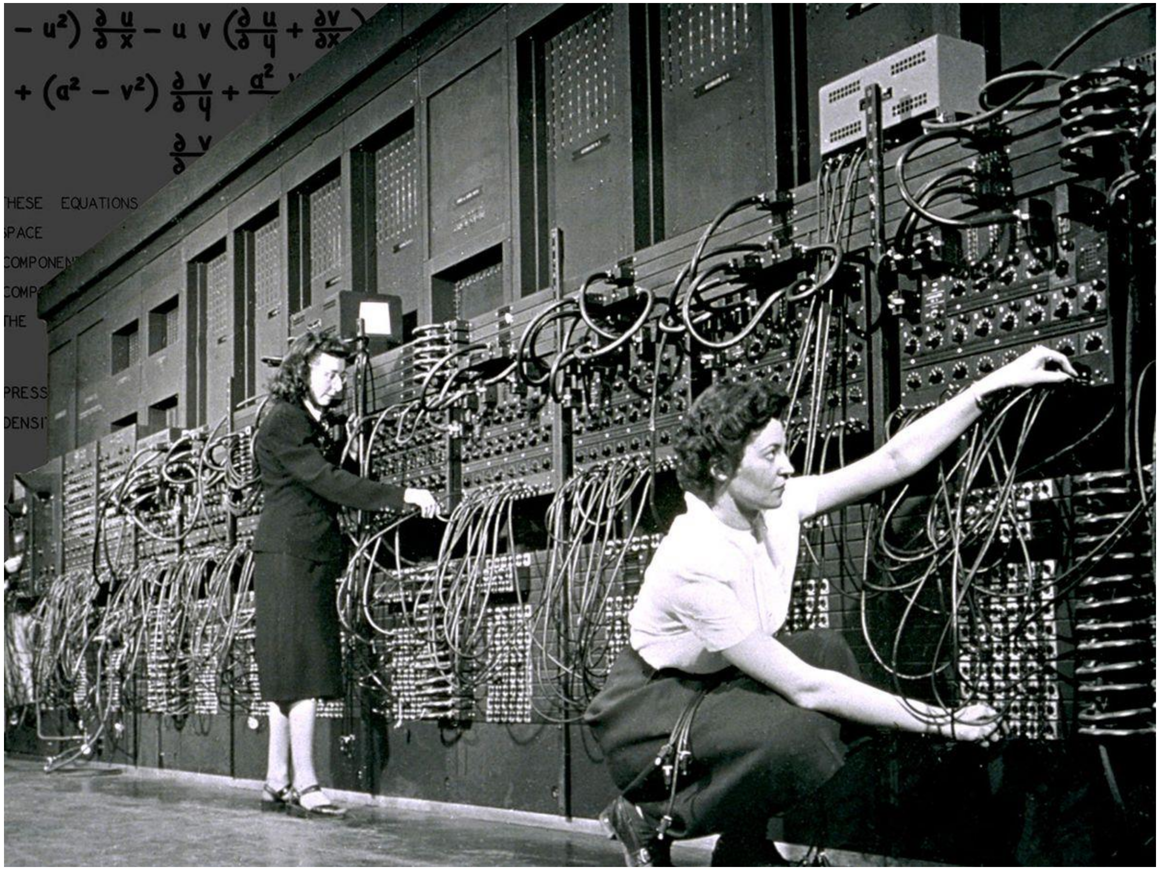
ENIAC

Electronic Numerical Integrator and Computer

J. Presper Eckert and John Mauchly

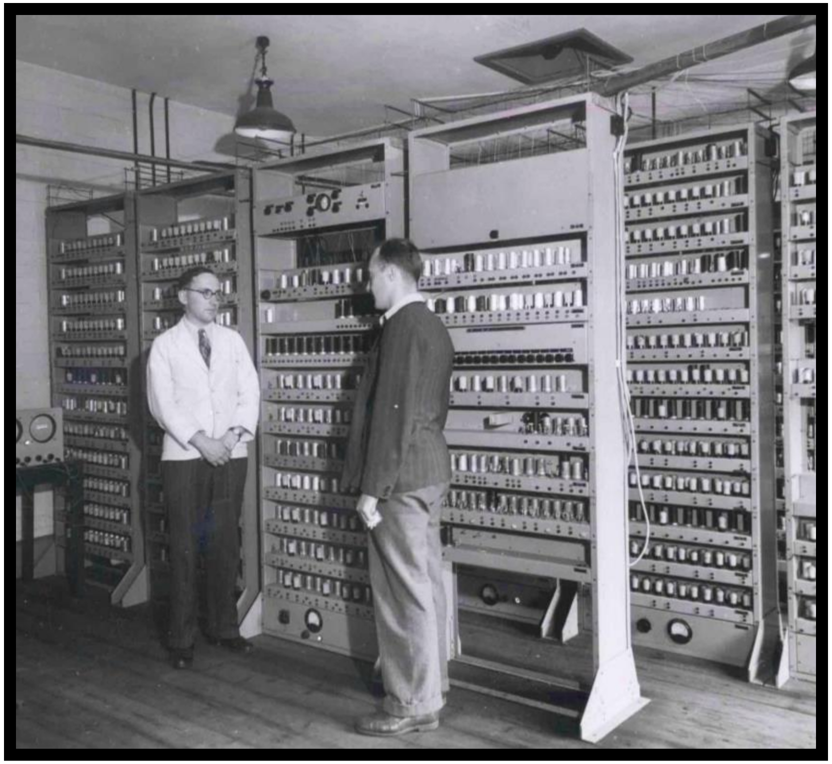
- 1^{st} working electronic computer (1946)
- To reprogram it you need to re-arrange the cords
- 18,000 Vacuum tubes
- 1,800 instructions/sec
- 3,000 ft^3

Computer History



Programming the ENIAC!

Computer History



EDSAC 1 (1949)

http://www.cl.cam.ac.uk/UoCCL/misc/EDSAC99/

- Von Neumann presented his idea of stored program concept.
- Maurice Wilkes built it.

1st stored program computer
650 instructions/sec
1,400 ft^3

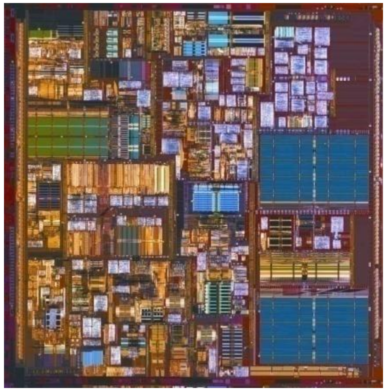
Computer History

- After the vacuum tubes, transistors were invented (1947) → Starting 2nd generations of computers

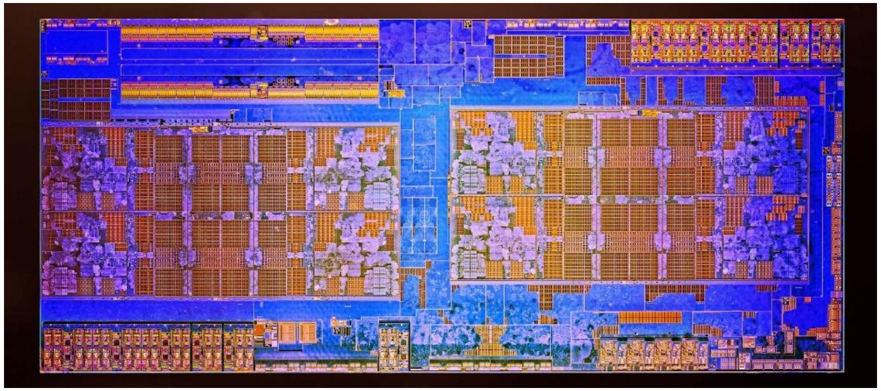
- UNIVAC (UNIversal Automatic Computer)
- Introduced in the 50s

Computer History

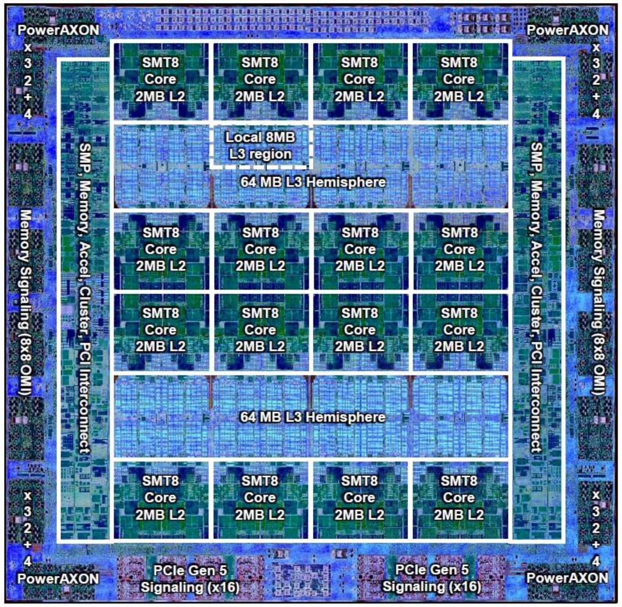
- From transistors to integrated circuits(IC) → Starting 3rd generation of computers
- One IC can host hundreds (then thousands, then millions then billions) of transistors → computers are getting smaller in size yet more powerful.



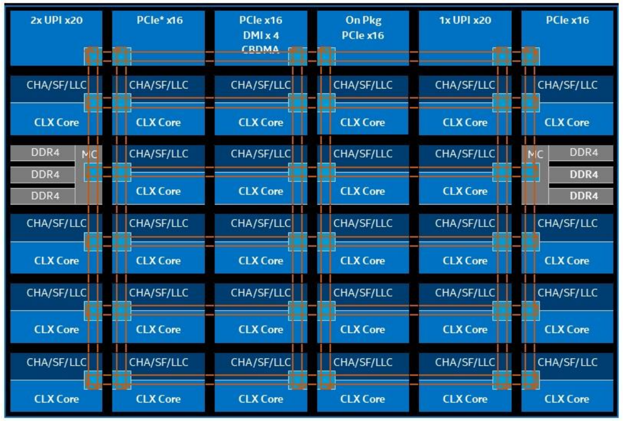
Pentium 4 (last single core)



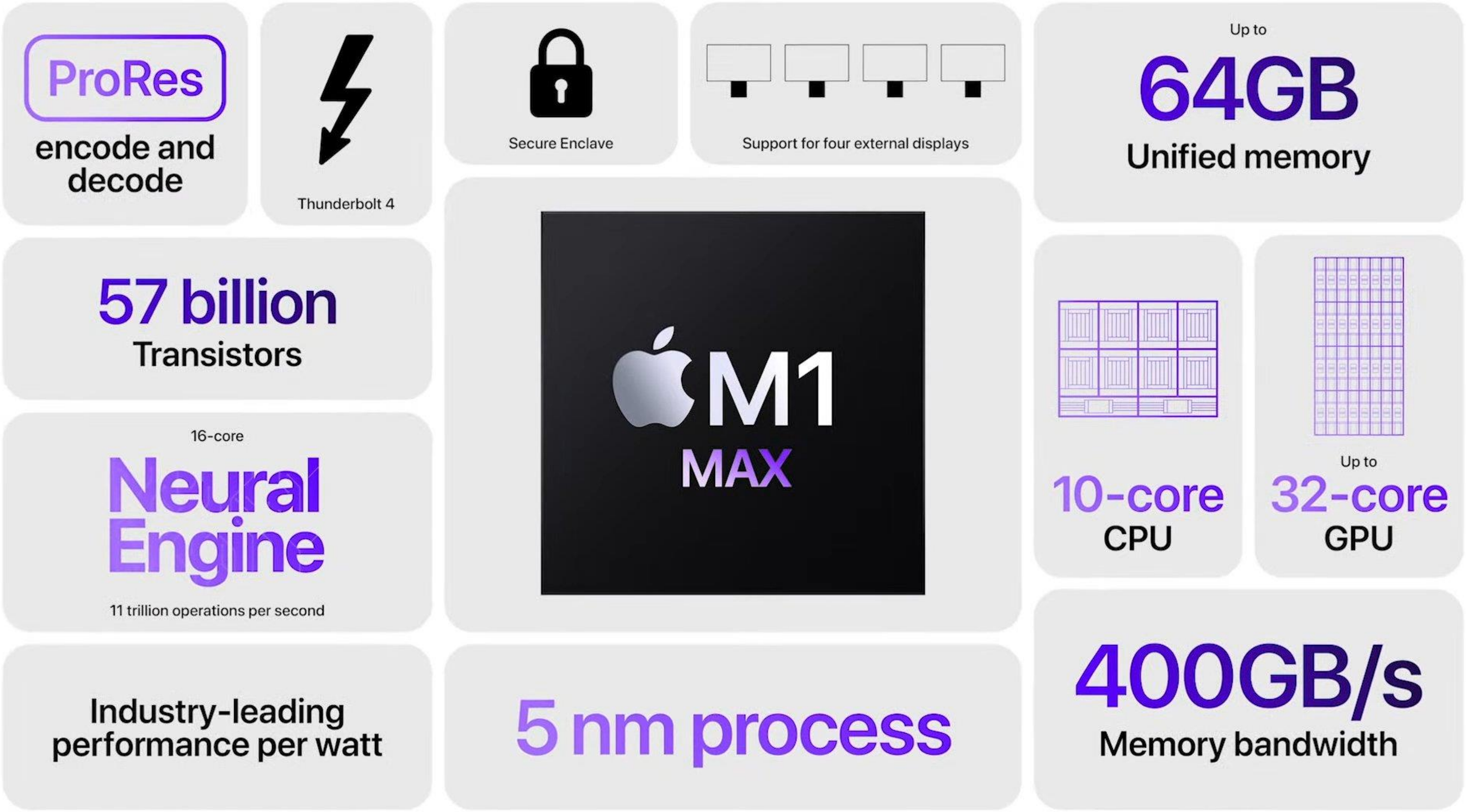
AMD Threaddripper (32 Cores)


PowerAXON x 3 2 + 4
PowerAXON x 3 2 + 4
SMT8 Core 2MB L2
Local 8MB L3 region
64 MB L3 Hemisphere
SMP, Memory, Accel, Cluster, PCI Interconnect
Memory Signaling (8x8 OMI)
SMT8 Core 2MB L2
64 MB L3 Hemisphere
SMP, Memory, Accel, Cluster, PCI Interconnect
Memory Signaling (8x8 OMI)
x 3 2 + 4 PowerAXON
PCIe Gen 5 Signaling (x16)
PCIe Gen 5 Signaling (x16)
x 3 2 + 4 PowerAXON

IBM Power 10 (15-30 cores)



Intel Xeon (28 cores)

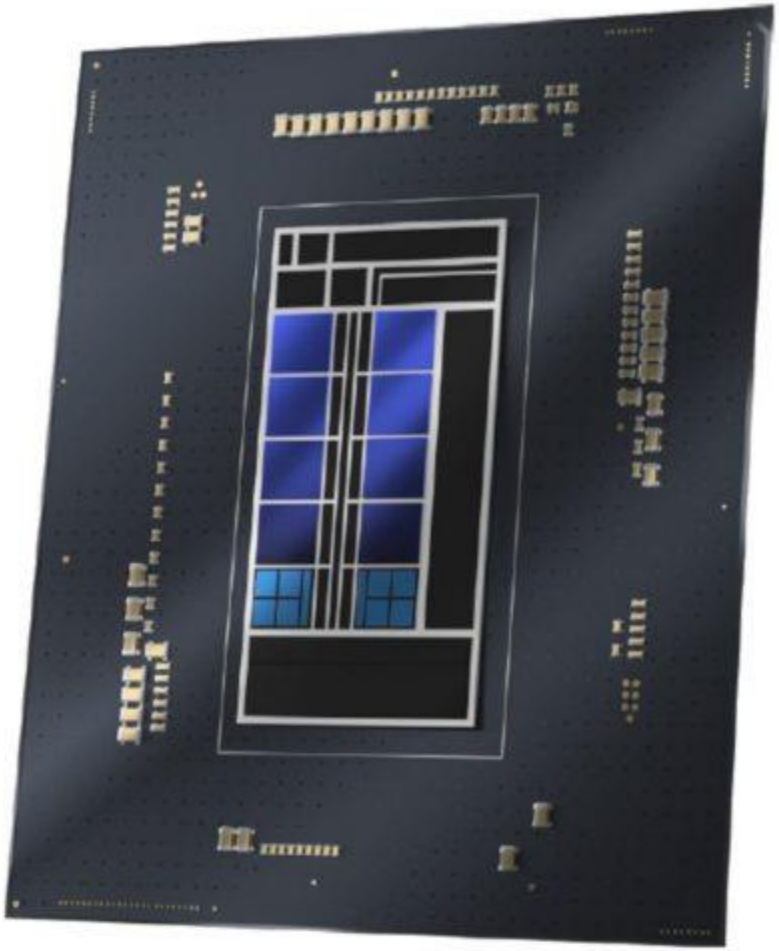


Source: Apple

Intel Alder Lake Architecture

Hybrid Architecture:

- P-cores
- E-cores

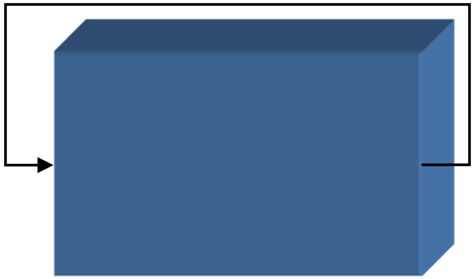


Source: Intel

How did the hardware evolve like that?

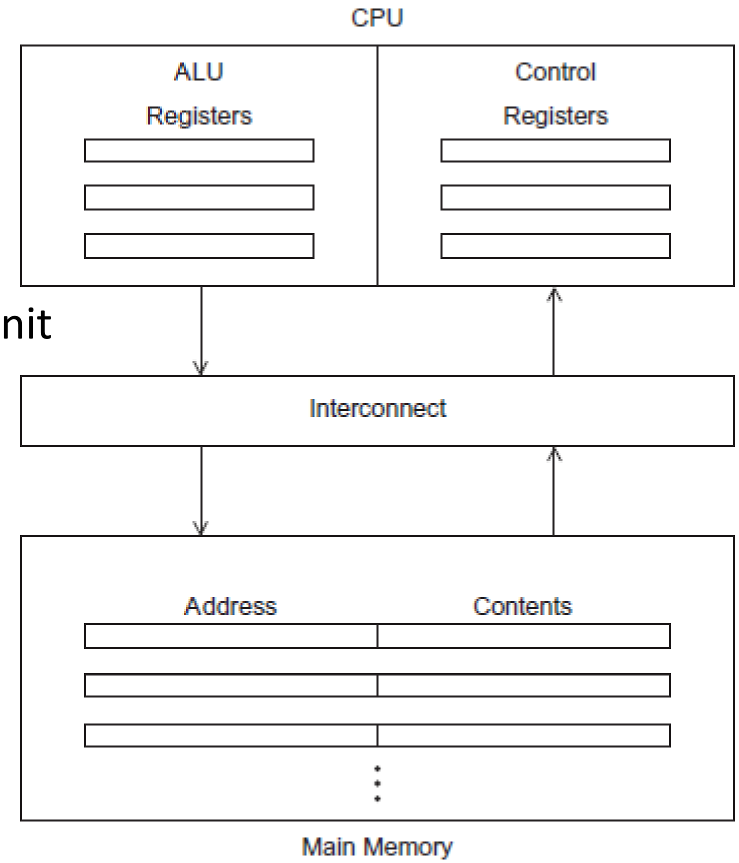
Let’s look at different waves (generations of architectures)

First Generation (1970s)



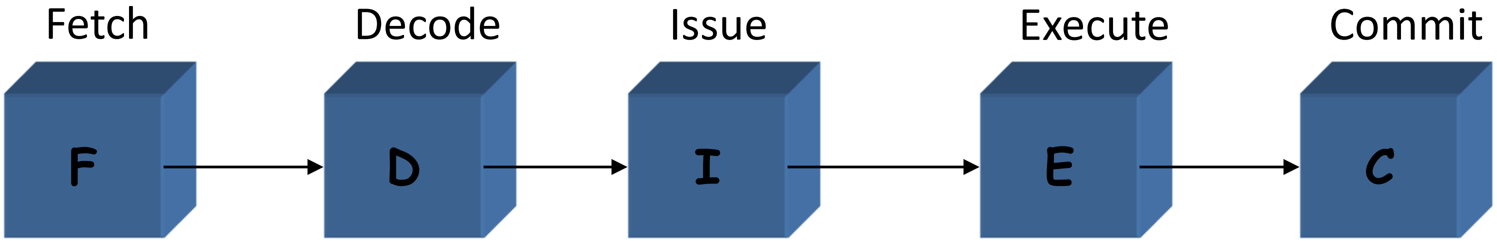
Single Cycle Implementation

The Von Neumann Architecture



ALU: Arithmetic and Logic Unit

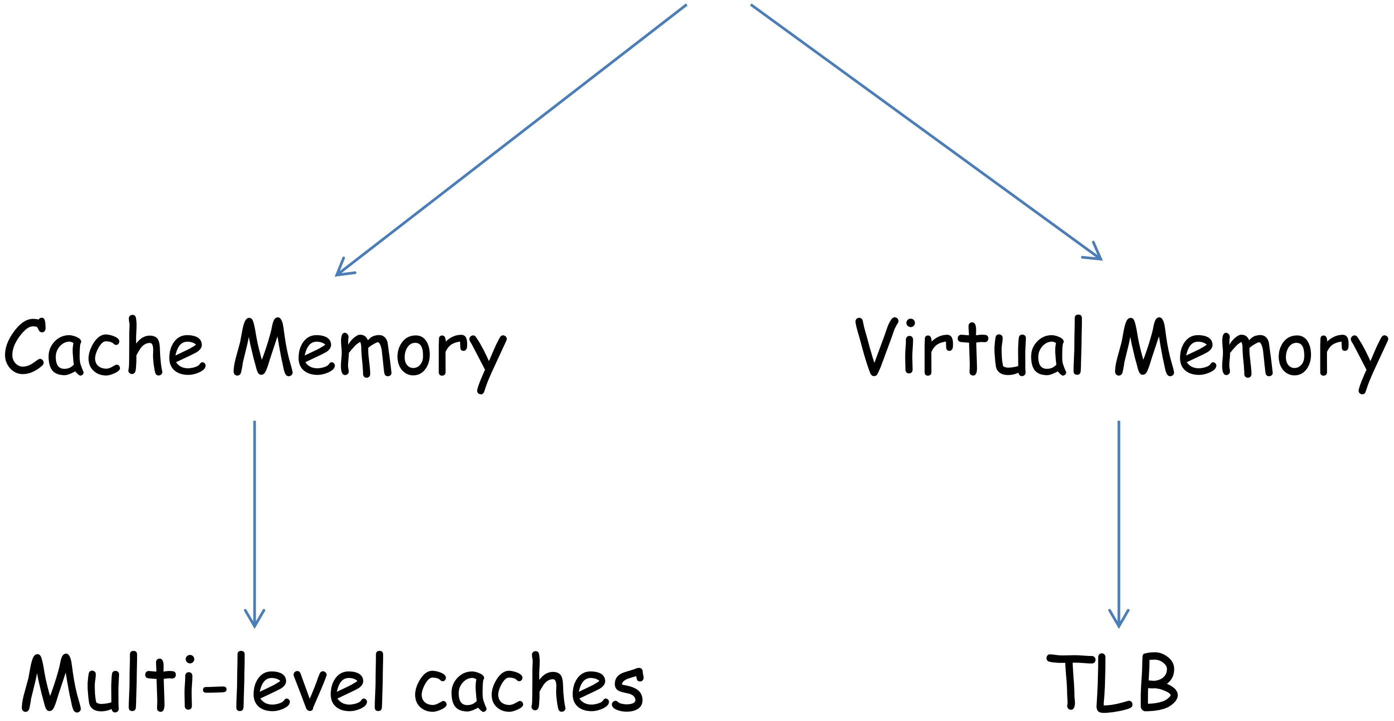
Second Generation (1980s)



- Pipelinining:
  - the hardware divided into stages
  - temporal parallelism
  - Number of stages increases with each generation

- CPI (Cycles Per Instruction) can be more than 1
  - Due to dependencies
    (i.e. an instruction must wait
    for the result of another instruction)

Some Enhancements



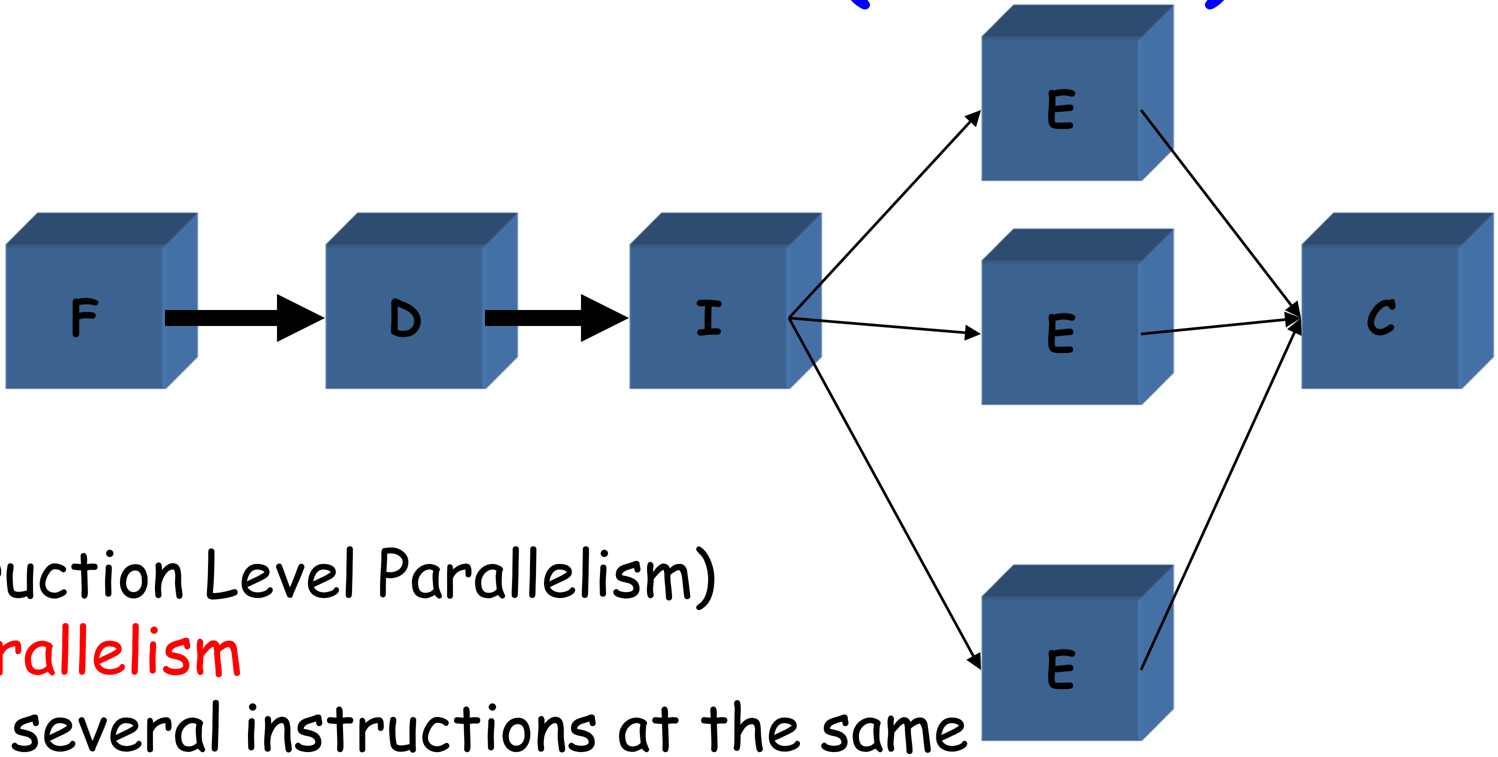
Cache Memory

Virtual Memory

Multi-level caches

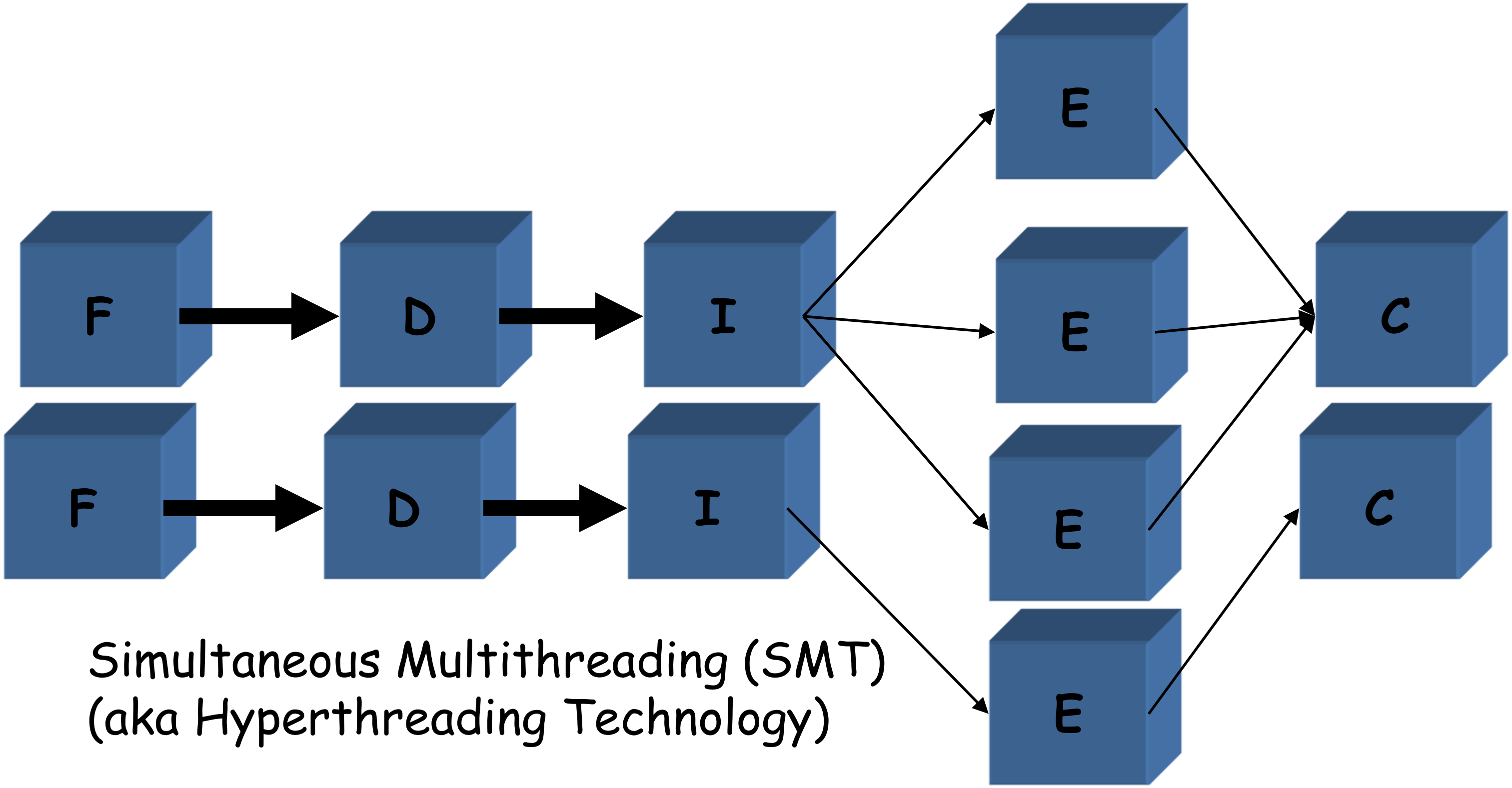
TLB

Third Generation (1990s)



- ILP (Instruction Level Parallelism)
- Spatial parallelism
- Executing several instructions at the same time is called superscalar capability.
- performance = instructions per cycle (IPC)
- Speculative Execution (prediction of branch direction) is introduced to make the best use of superscalar capability → This can make some instructions execute out-of-order!!

Fourth Generation (2000s)



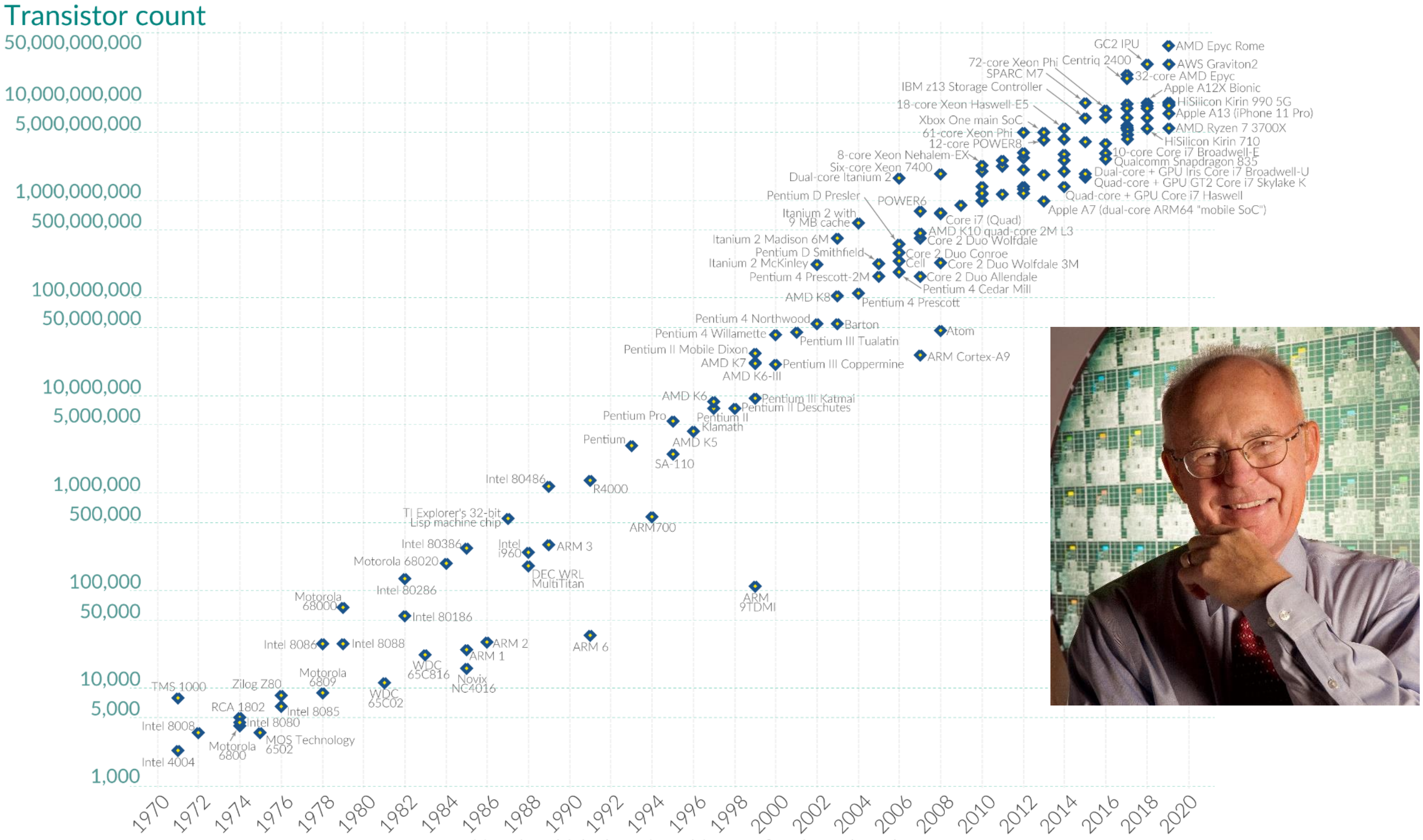
Simultaneous Multithreading (SMT)
(aka Hyperthreading Technology)

Double (or triple or …) some resources in the pipeline to host several programs at *the same time*.
This allows better use of the execution resources.

The Status-Quo

- We moved from single core to multicore to manycore:
  - for technological reasons … As we will see shortly.
- Free lunch is over for software folks
  - The software will not become faster with every new generation of processors
- Not enough experience in parallel programming
  - Parallel programs of old days were restricted to some elite applications -> very few programmers
  - Now we need parallel programs for many different applications

The Famous Moore’s Law



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

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.

Our World in Data

Year in which the microchip was first introduced

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 works because of …

Dennard scaling

MOSFETs continue to function as voltage-controlled switches while all key figures of merit such as layout density, operating speed, and energy efficiency improve provided geometric dimensions, voltages, and doping concentrations are consistently scaled to maintain the same electric field.

Robert Dennard

Simply speaking scaling down transistor dimensions leads to:

- Reduce circuit delay
- Increase operating frequency
- Operating voltage is reduced → reducing power

If number of transistors doubles
for the same chip area, the circuit
Becomes 40% faster and power consumption stays the same.

Effect of Moore's law

- ~1986 - 2002 → 50% performance increase
- Since 2002 → ~20% performance increase

Hmmm …

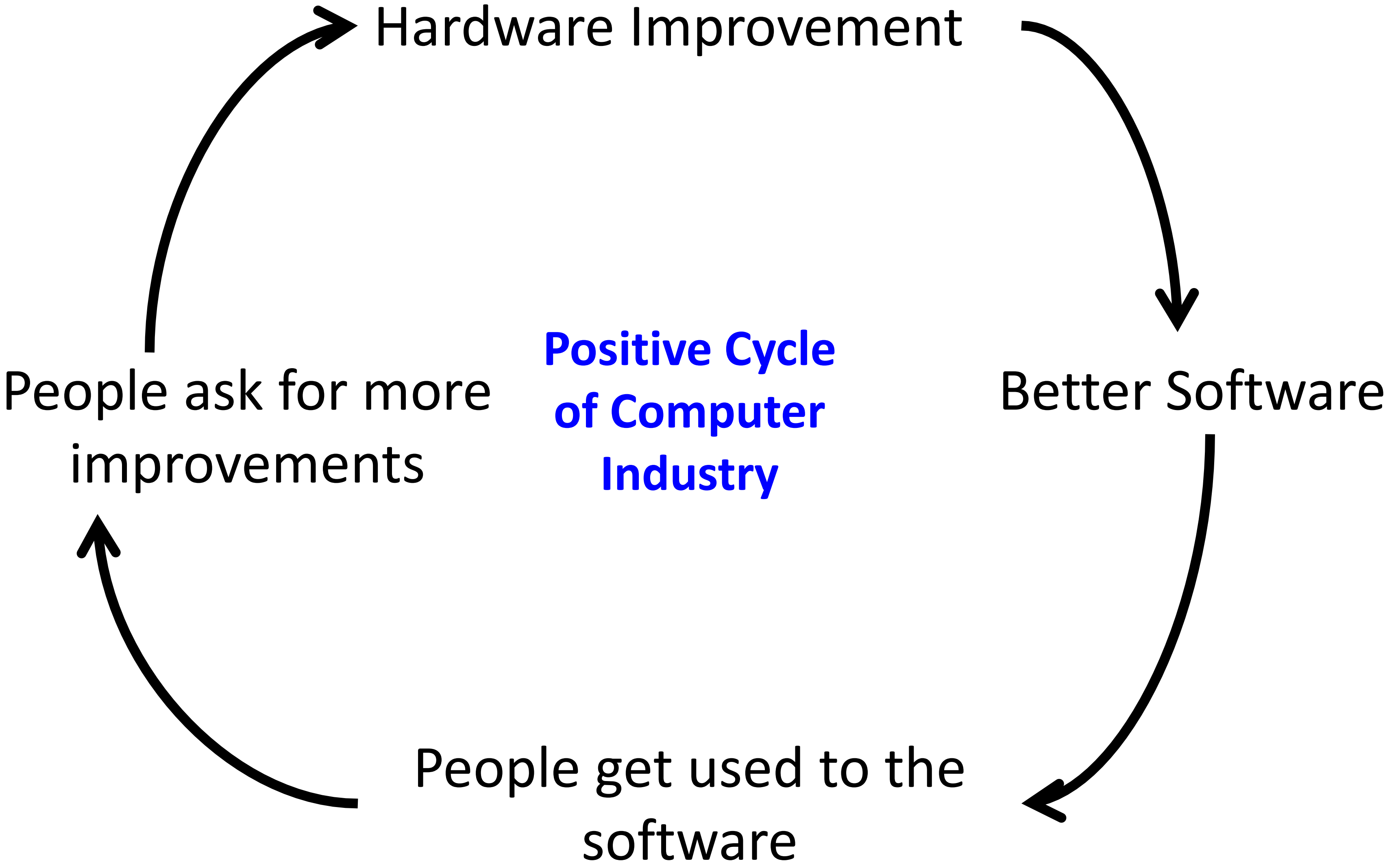
- Why do we care? 20%/year is still nice.
- What happened at around 2002?
- Can't we have auto-parallelizing programs?

Why do we care?

- More realistic games
- Decoding the human genome
- More accurate medical applications

The list goes on and on ….

As our computational power increases → the number of problems we can seriously consider also increases.



Hardware Improvement

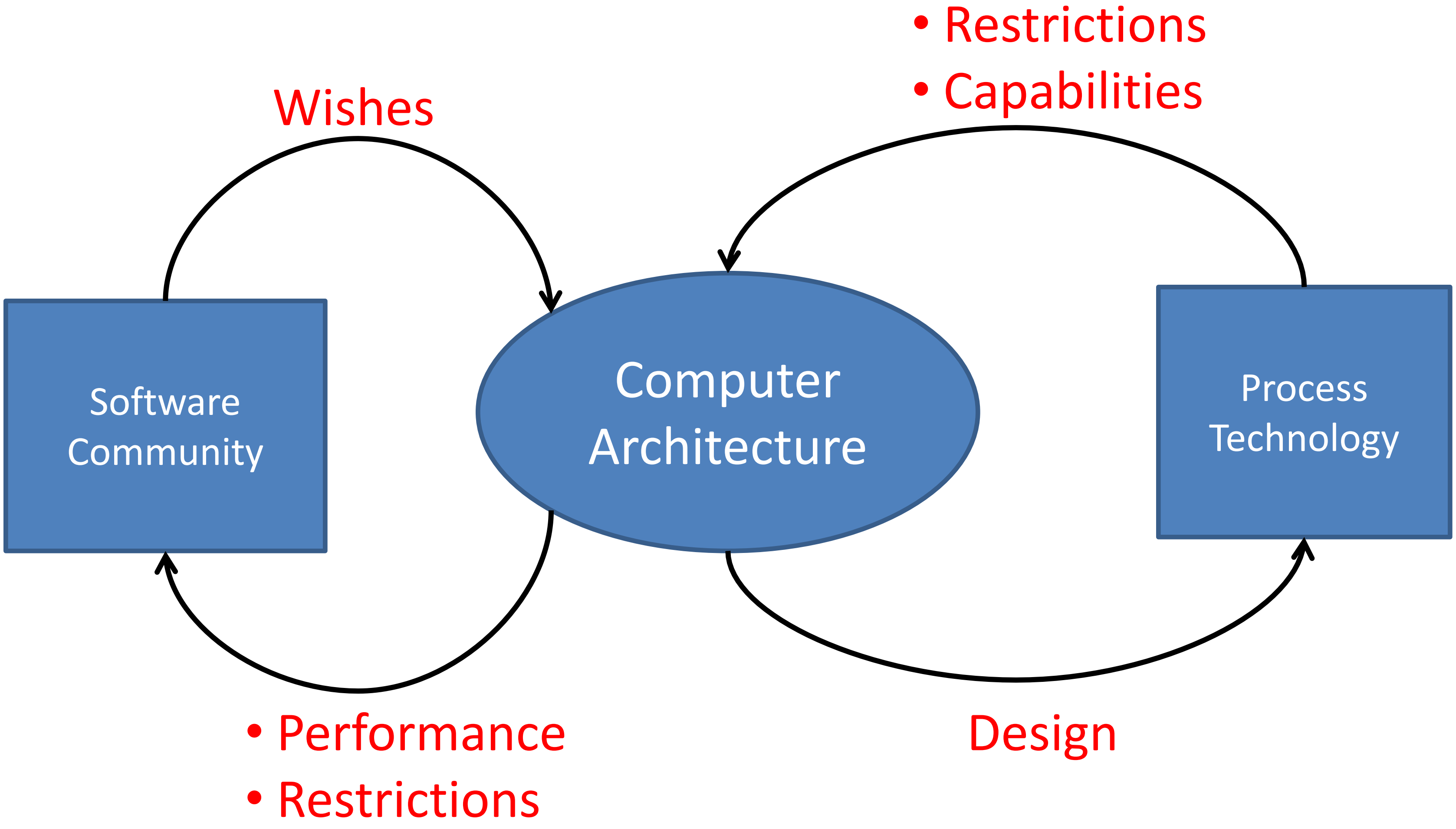
Better Software

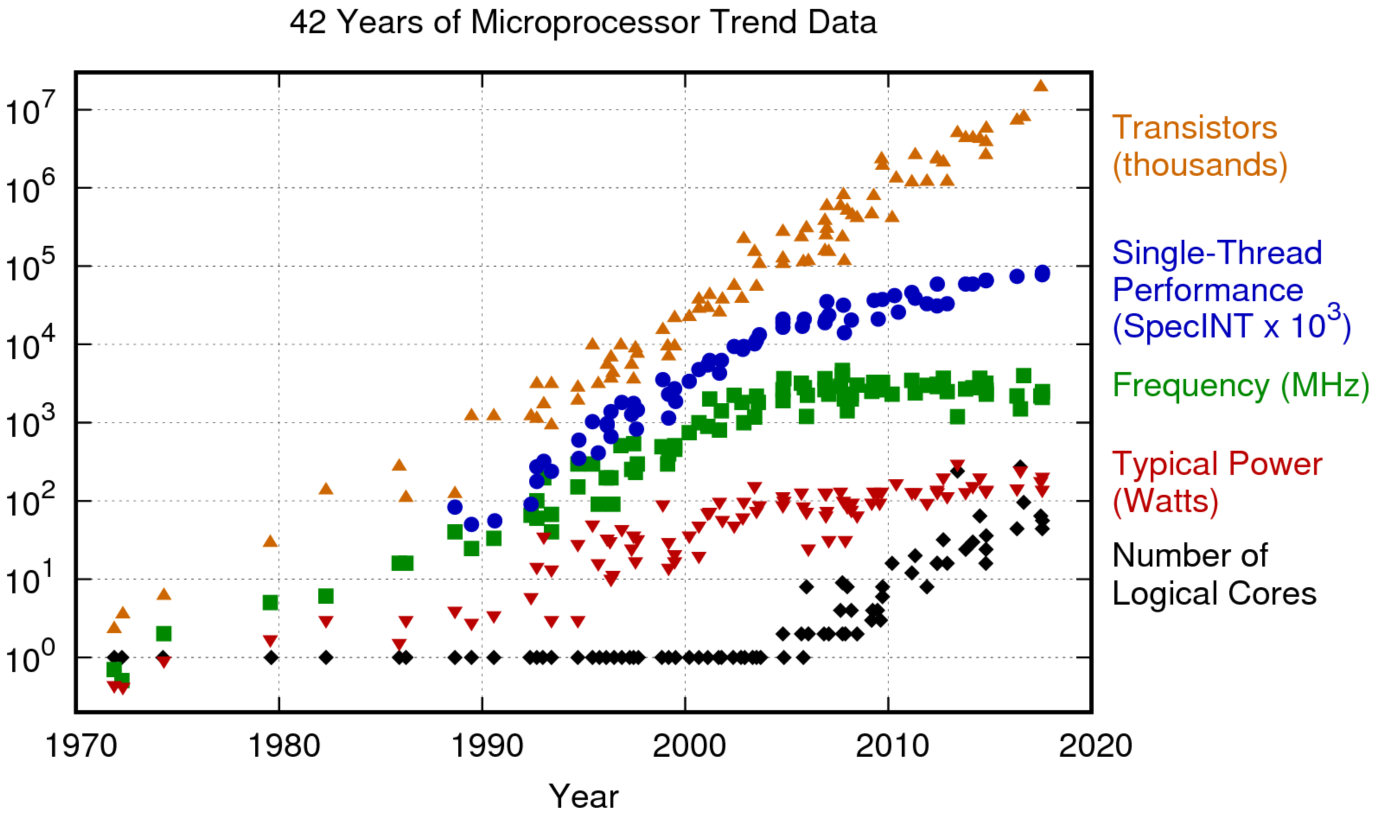
People get used to the software

People ask for more improvements

Positive Cycle of Computer Industry

How Did These Advances Happen?





Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Data compiled by Karl Rupp.
https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

Performance in the past achieved by:

- clock speed
- execution optimization
- cache

Performance now achieved by:

- hyperthreading
- multicore
- cache

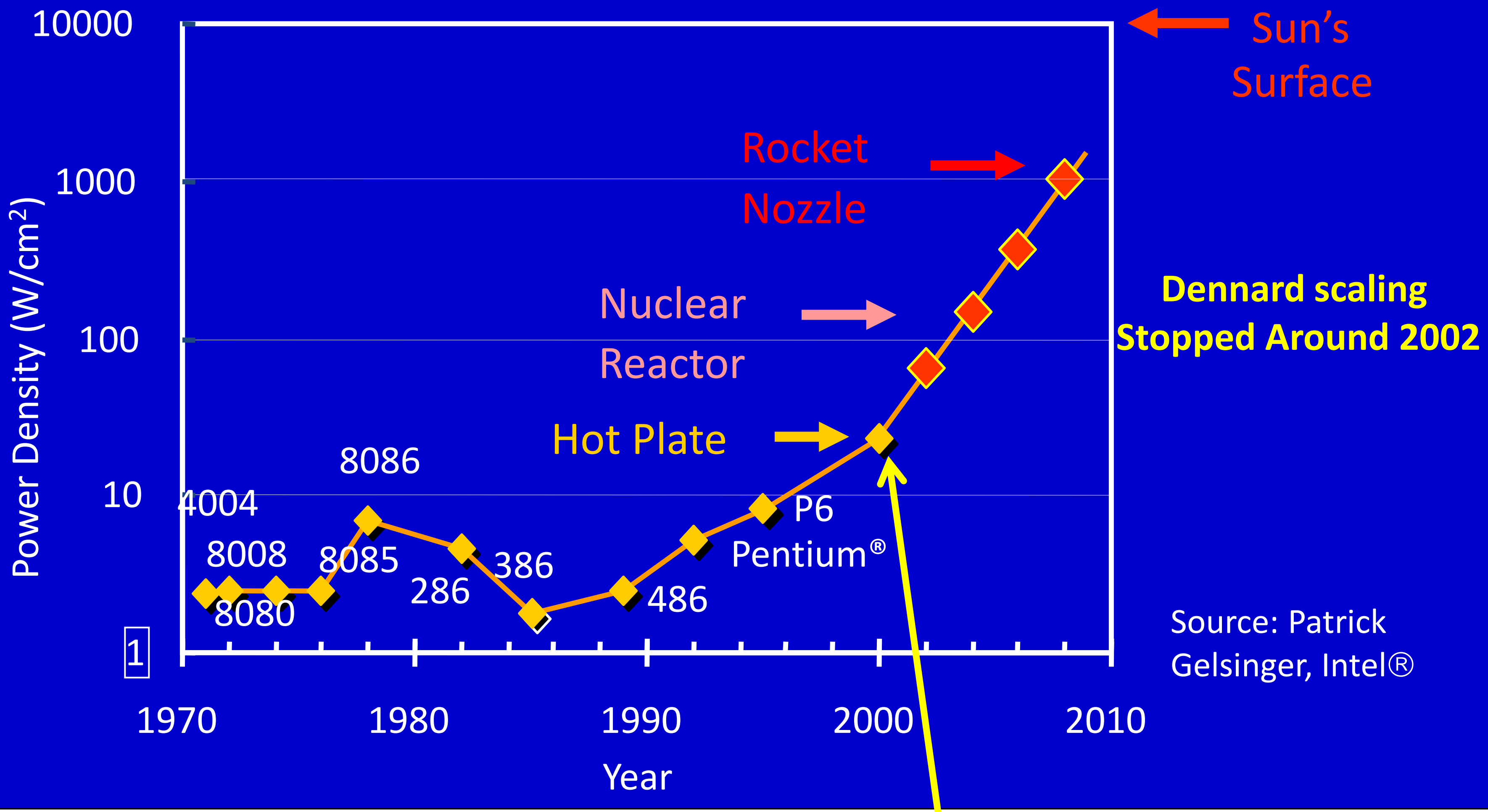
Why did we build parallel machines (and continue to do so)?

(multicore, multiprocessors, multi-anything!)

Power Density

Moore's law is giving us more transistors than we can afford!

Scaling clock speed (business as usual) will not work



Dennard scaling Stopped Around 2002

Source: Patrick Gelsinger, Intel®

This is what happened at around 2002!

Why Did Power Become a Problem?

- Power needed per chip is:
  - **Power consumed**: to do the calculations the processor does to execute your programs.
  - **Power dissipated**: in the form of temperature.
- More power means:
  - Exponential increase in the cost of packaging
  - Electricity bill (or shorter battery life)
- Power per chip increases due to two factors:
  - Increasing the number of transistors per chip
  - Increasing the frequency (i.e. clock scaling)

Multicore Processors Save Power

C: Capacitance V: Voltage F: Frequency

Power = $C * V^2 * F$

Performance = Cores * F

Let's double the number of cores

Power = $2*C * V^2 * F$

Performance = 2*Cores * F

But decrease frequency by 50% **(V is proportional to F)**

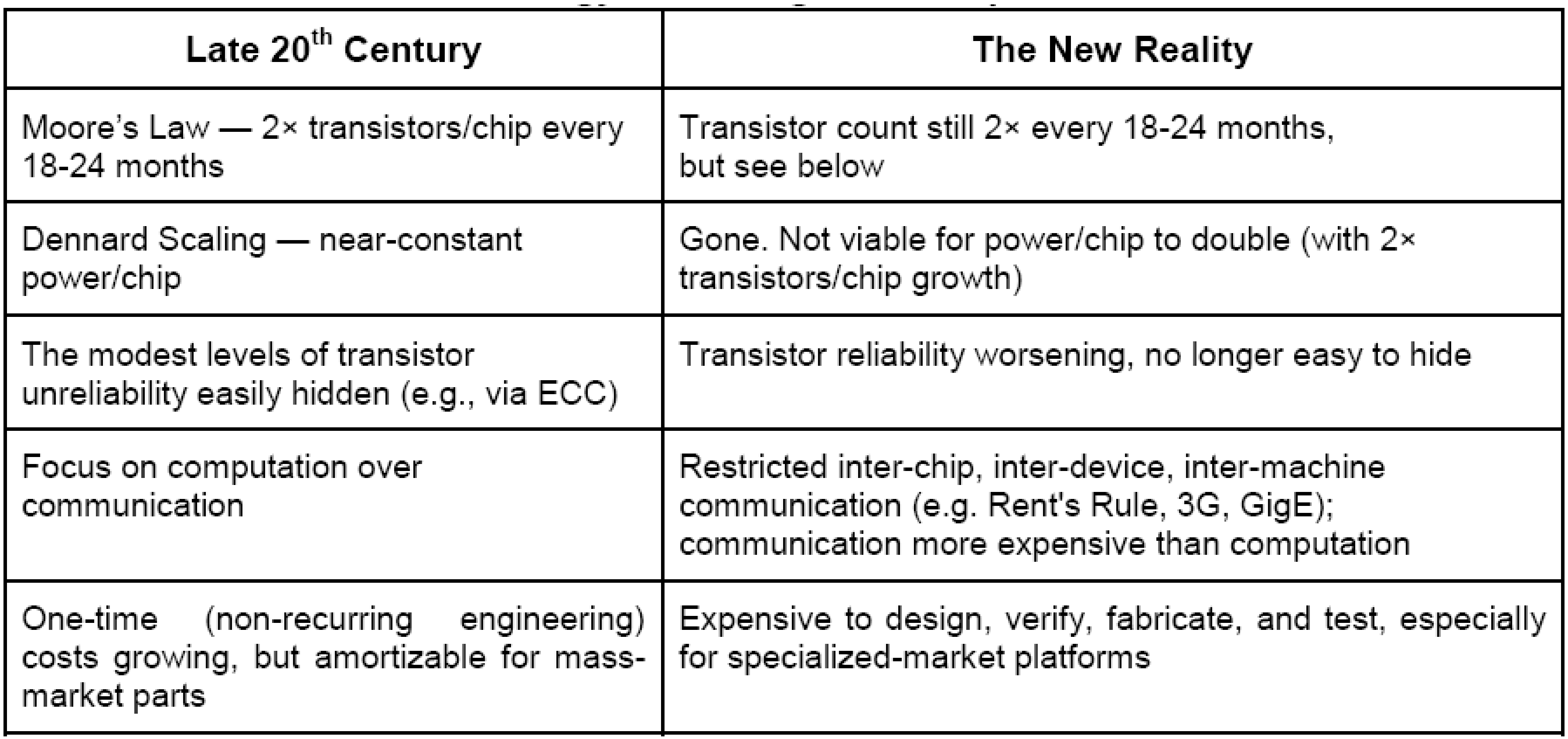
Power = $2*C * V^2/4 * F/2$

Performance = 2*Cores * F/2

Power = $C * V^2/4 * F$

Performance = Cores * F

Inflection Point in Computing



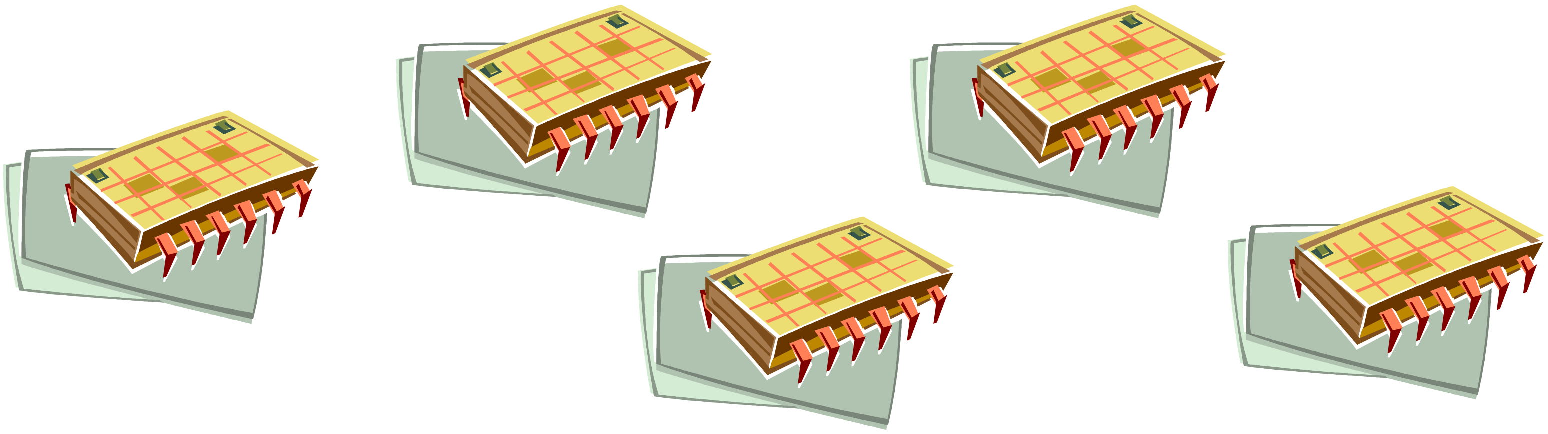
Late 20th Century	The New Reality
Moore's Law — 2× transistors/chip every 18-24 months	Transistor count still 2× every 18-24 months, but see below
Dennard Scaling — near-constant power/chip	Gone. Not viable for power/chip to double (with 2× transistors/chip growth)
The modest levels of transistor unreliability easily hidden (e.g., via ECC)	Transistor reliability worsening, no longer easy to hide
Focus on computation over communication	Restricted inter-chip, inter-device, inter-machine communication (e.g. Rent's Rule, 3G, GigE); communication more expensive than computation
One-time (non-recurring engineering) costs growing, but amortizable for mass-market parts	Expensive to design, verify, fabricate, and test, especially for specialized-market platforms

A Case for Multicore Processors

- Can exploit different types of parallelism
- Reduces power
- An effective way to hide memory latency
- Simpler cores → easier to design and test → higher yield → lower cost

An intelligent solution

- Instead of designing and building faster microprocessors, put multiple processors on a single integrated circuit.



Now it's up to the programmers

- Adding more processors doesn't help much if programmers aren't aware of them…
- … or don't know how to use them.

- Serial programs don't benefit from this approach (in most cases).

The Need for Parallel Programming

Parallel computing: using multiple processors (or cores) in parallel to solve problems more quickly than with a single processor

Examples of parallel machines:

***A* cluster computer** that contains multiple PCs combined together with a high speed network

***A* shared memory multiprocessor** (SMP) by connecting multiple processors to a single memory system

***A* Chip Multi-Processor** (CMP) contains multiple cores on a single chip

Cost and Challenges of Parallel Execution

- Communication cost
- Memory hierarchy access overhead
- Synchronization cost
- Not all problems are amenable to parallelization
- Hard to think in parallel
- Hard to debug

Attempts to Make Multicore Programming Easy

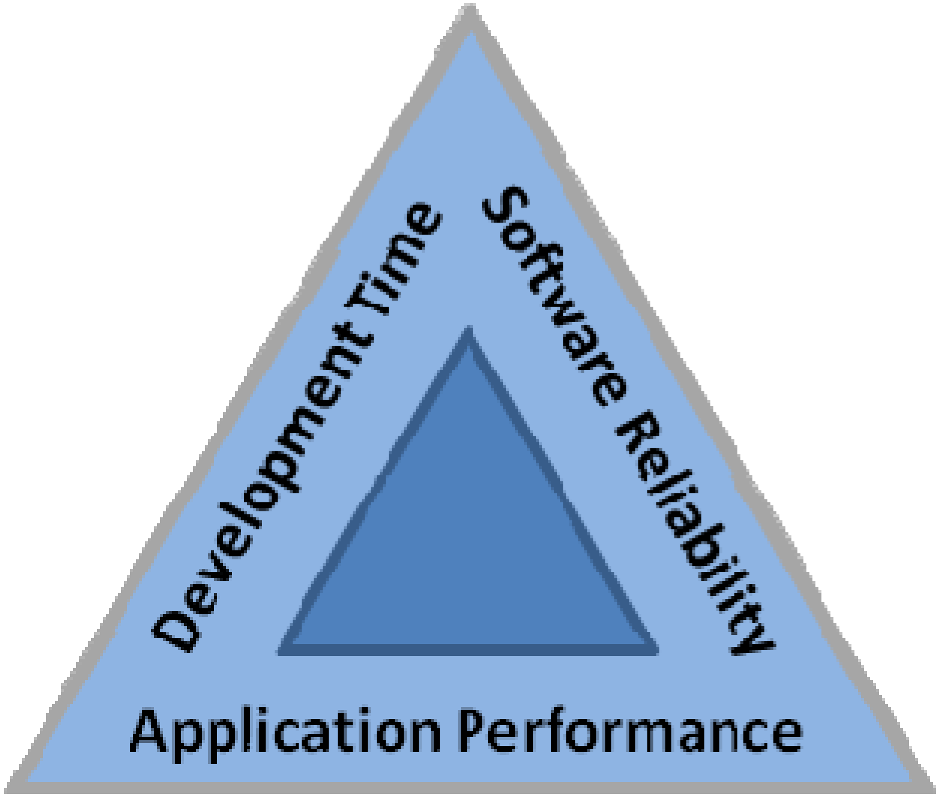
- 1st idea: The right computer language would make parallel programming straightforward
  - **Result so far**: Some languages made parallel programming easier, but none has made it as fast, efficient, and flexible as traditional sequential programming.

Attempts to Make Multicore Programming Easy

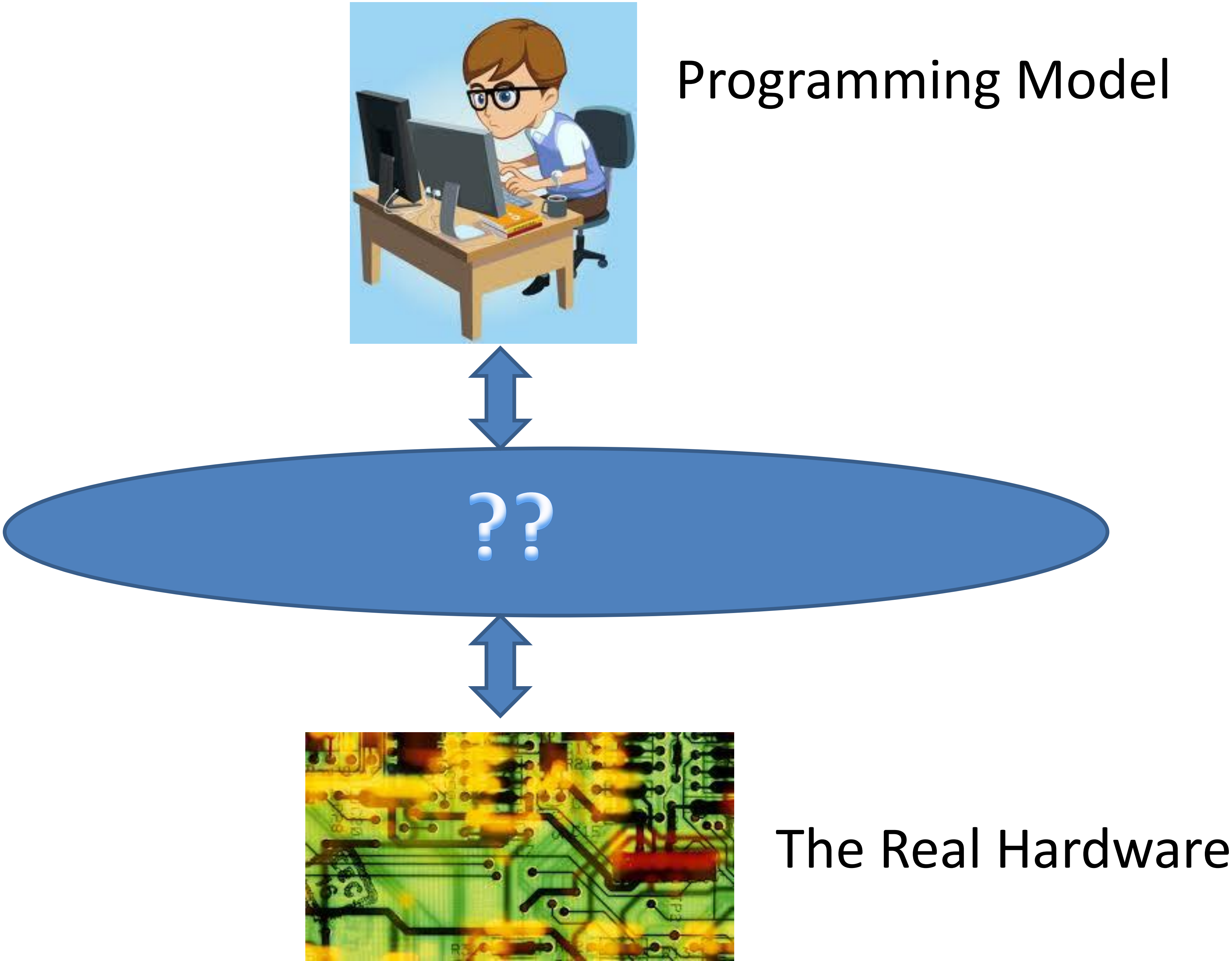
- 2nd idea: If you just design the hardware properly, parallel programming would become easy.
  - **Result so far**: no one has yet succeeded!

Attempts to Make Multicore Programming Easy

- 3rd idea: Write software that will automatically parallelize existing sequential programs.
  - **Result so far**: Success here is inversely proportional to the number of cores!



The Multicore Software Triad



Programming Model

??

The Real Hardware

Conclusions

- We are always in need for more performance.
- The free lunch is over for software folks!
- Mulicore/Manycore processors are here to stay, so we have to deal with them.
  - Power density pushed us to move to multicore before we are ready for it!
- Knowing about the hardware will make you way more efficient in software!