Overview of 15-740

15-740 SPRING'18

NATHAN BECKMANN

Topics

What is computer architecture?

Underlying technology

Information about the class

History

The most important thing

I WELCOME YOUR INTERRUPTIONS!

An interactive class is better for everyone…

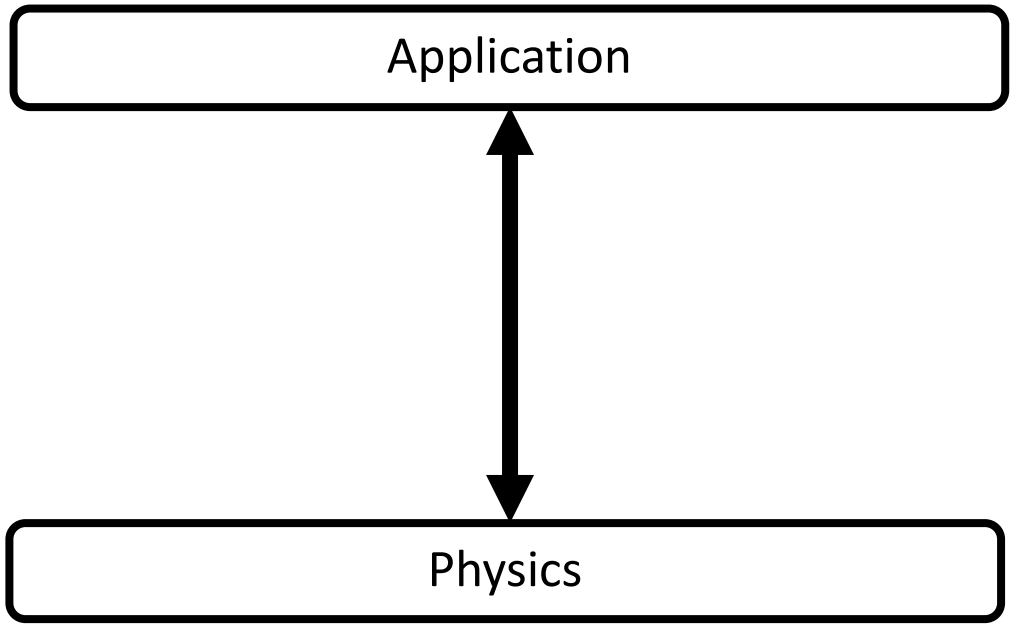
We will all learn more and have more fun!

(also it helps your grade…)

What is computer architecture?

The science and art of selecting and interconnecting hardware components to create computers that meet functional, performance and cost goals. [wikipedia]

Abstractions to bridge gap



Responsive to technology

- Underlying components:
  - Relays → Tubes → Transistors → VLSI → ?
  - Magnetic core → SRAM → DRAM → FLASH → ?

- What to optimize for:
  - Transistors
  - Memory
  - Instructions
  - Performance
  - Power
  - Parallelism

- **Technology constantly changing!**

Responsive to applications

So what is computer architecture?

- **The answer is constantly changing.**

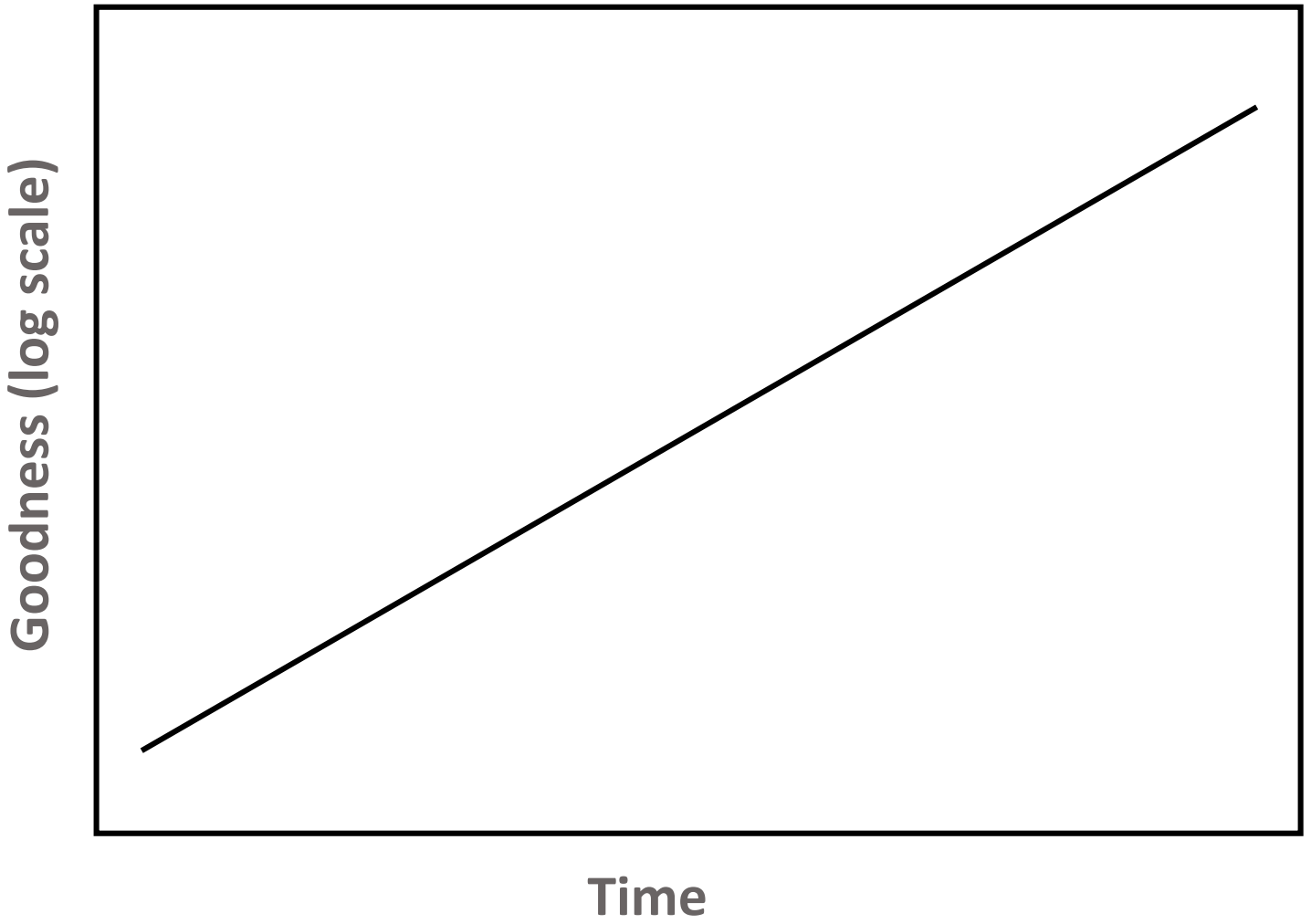
- As technology and application space change, so too the focus of computer architecture:

- 1950s-60s: Computer arithmetic
- 1970s-80s: Instruction set architecture
- 1980s-90s: CPU design
- 1990s-2000s: Memory system, I/O, networks
- 2000s-today: Power, multicore
- 2010s: Specialized accelerators

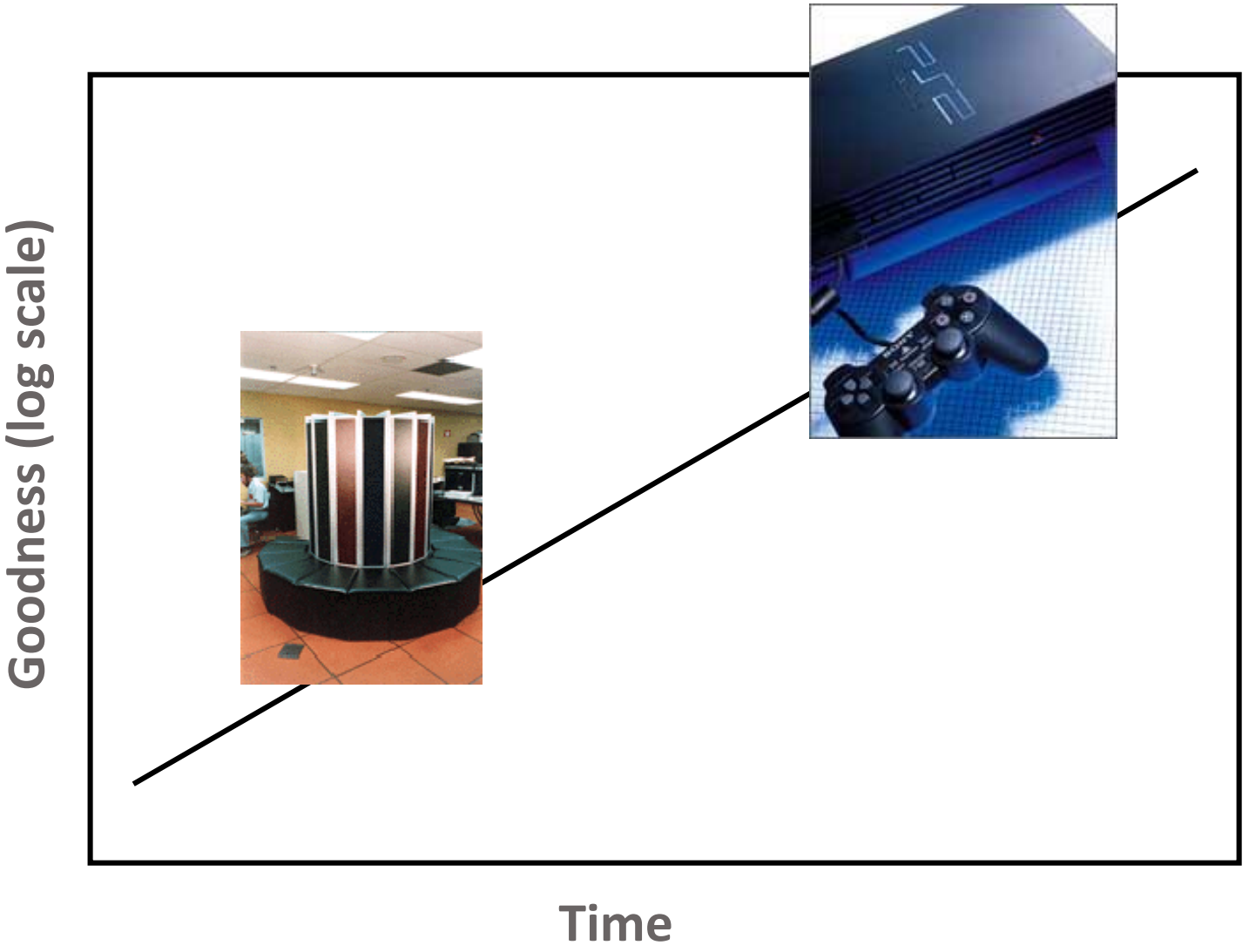
Ever-Changing Technology

50 YEARS IN 15 SLIDES

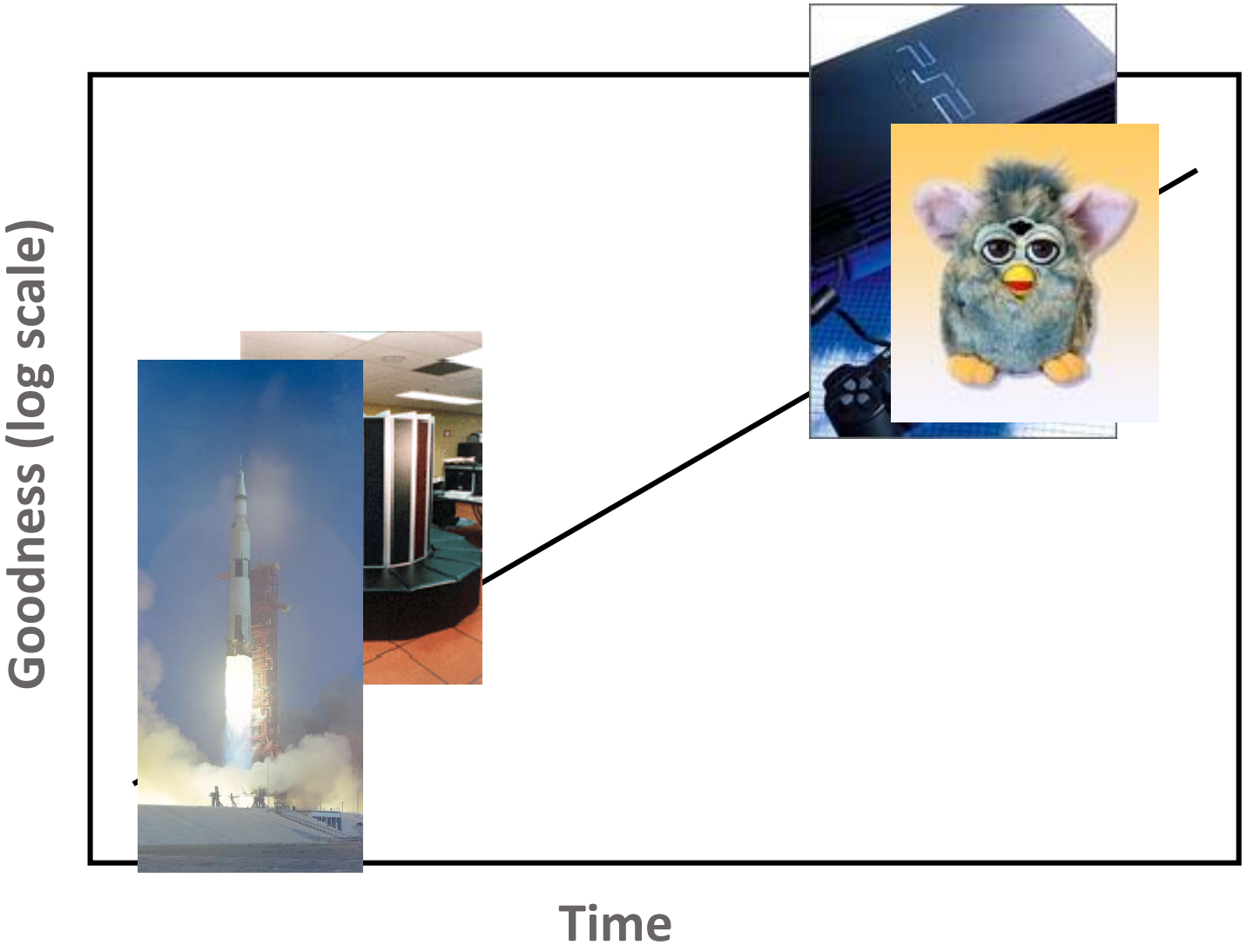
Moore’s Law



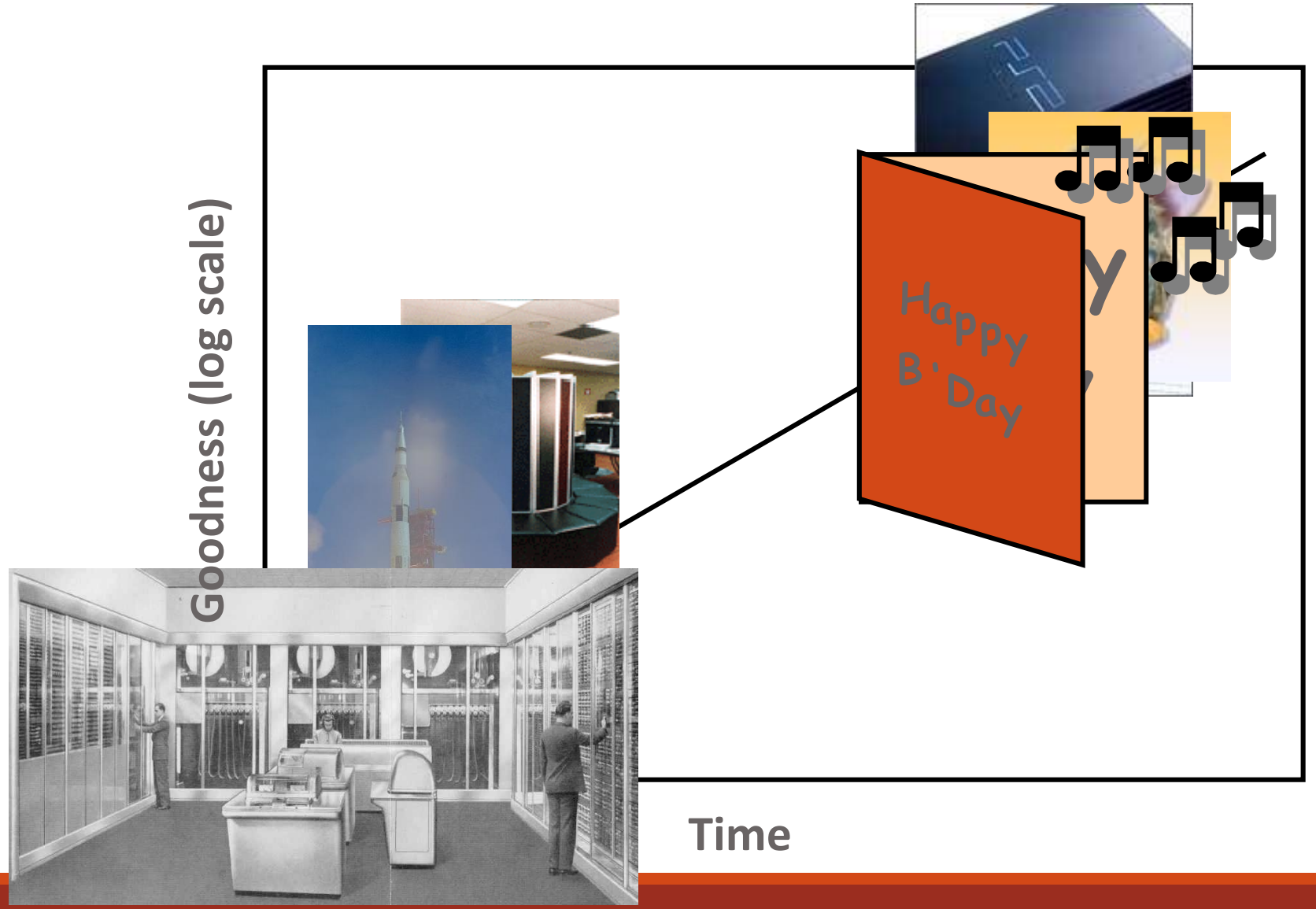
Moore’s Law



Moore’s Law

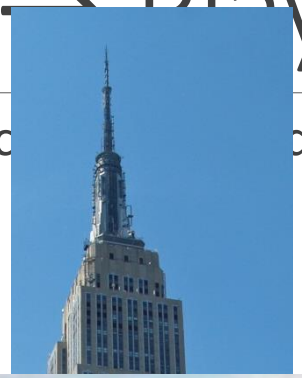


Moore’s Law



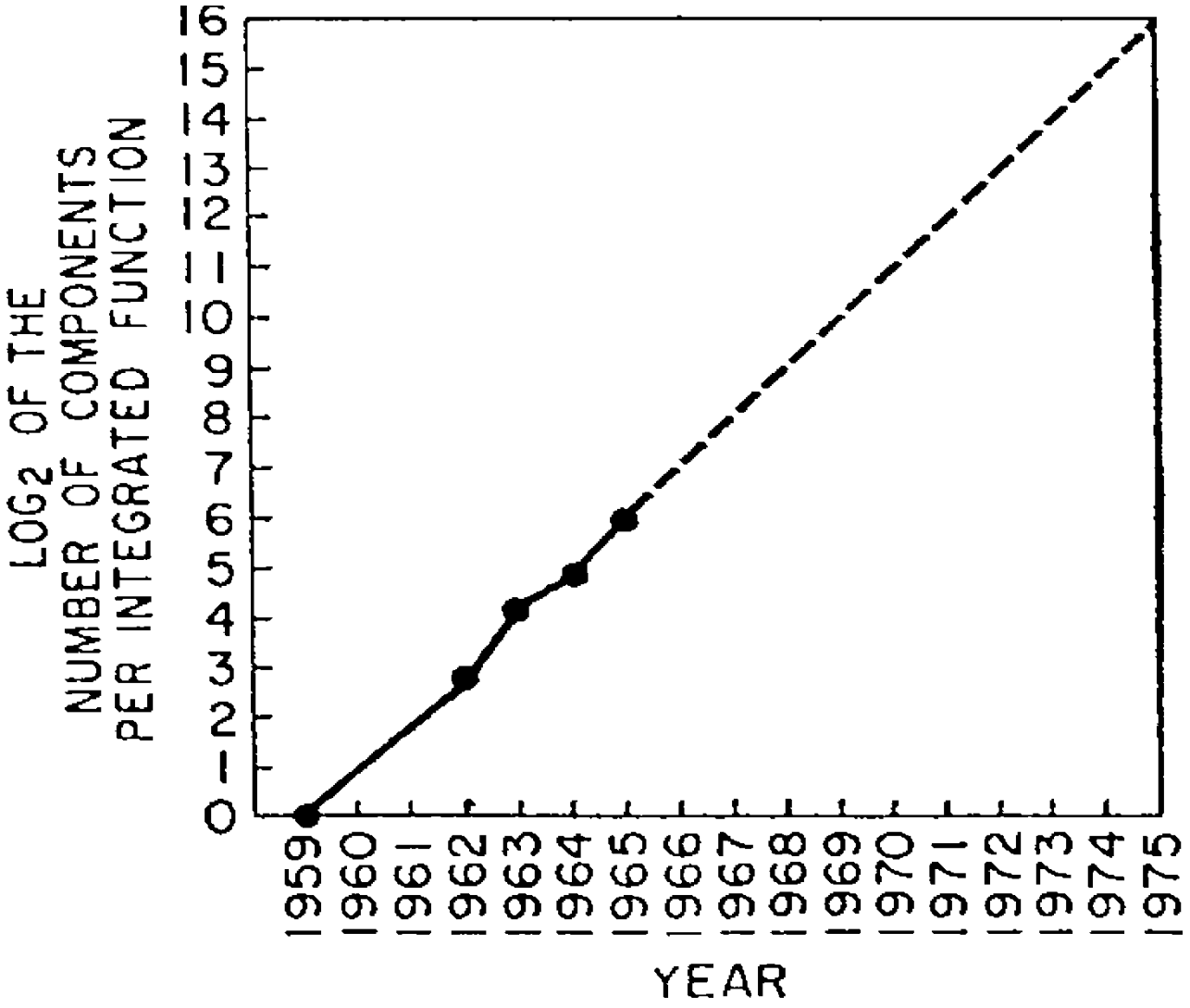
Case study: Eniac → Playstation 4

How much would enough Eniac [illegible] qual 2.8Kg of PS/4 computing?

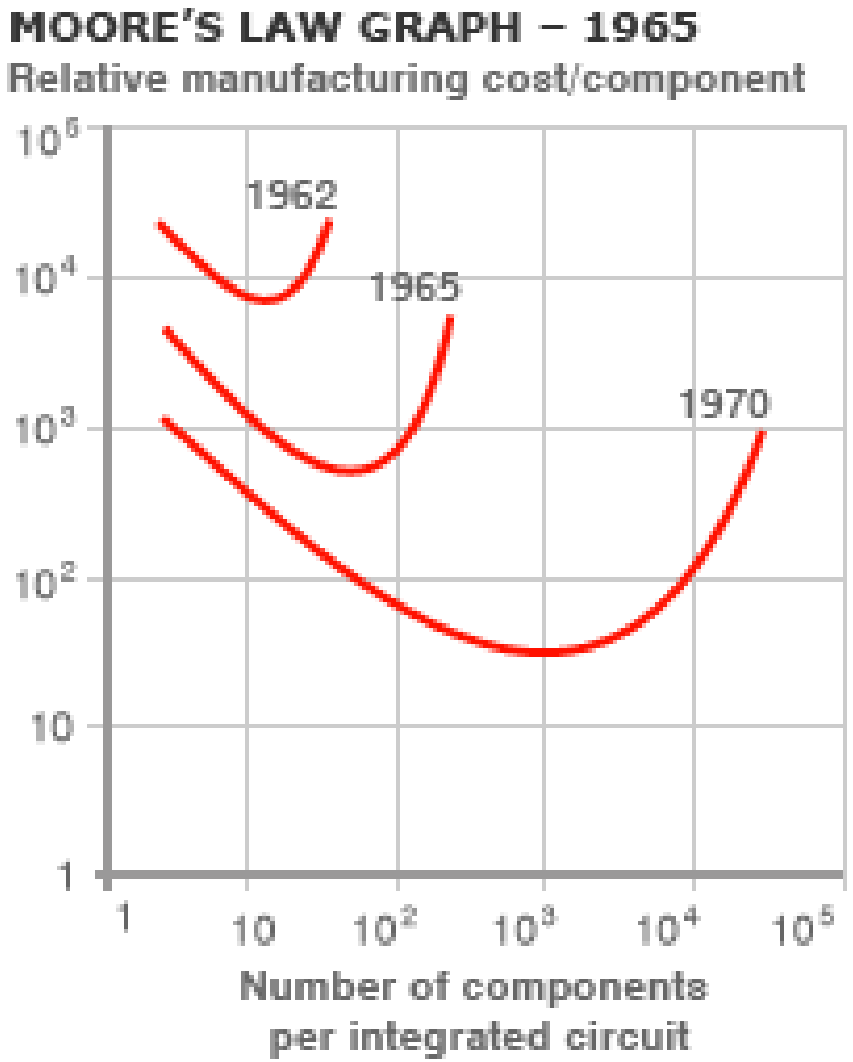


Alternatively, more than all the buildings in Pittsburgh!

Moore’s original prediction from 1965



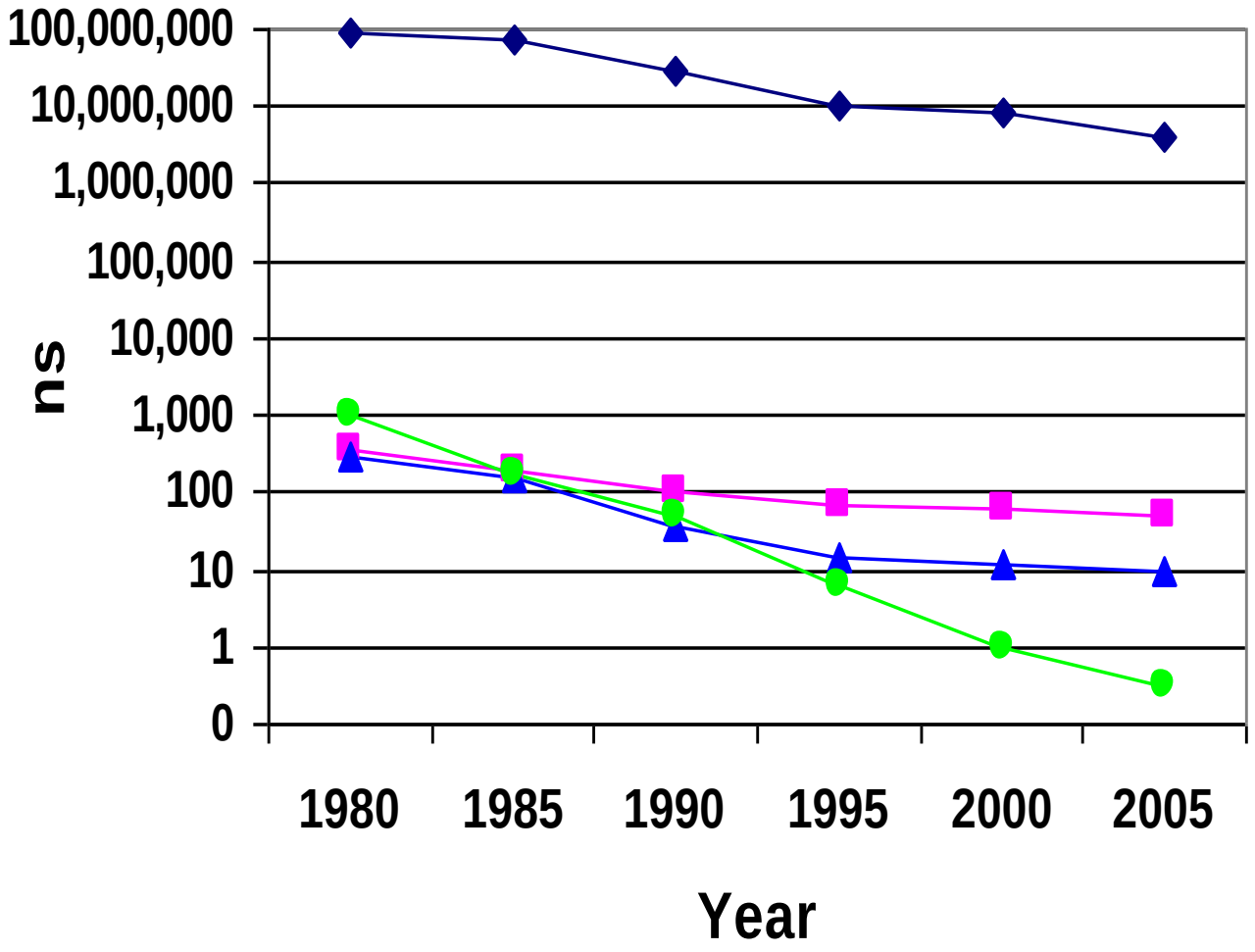
Moore’s law really about economics?



Technology changes architecture

- It isn't just transistor density
  - Transistor size, density, speed, power, cost
  - Memory size, density, latency, throughput
  - Disks
  - Networks
  - Communication
- These trends lead to exponential increase in ops/sec-$-$m^3$-watt
- Which in turn leads to changes in applications
  Mainframes → Desktops → Mobile
- Which leads to new design goals

Case study: The CPU-memory gap



Q: Why isn't memory getting faster?

A: Exponential growth in memory **size.**
➔ *It's not all about performance!*

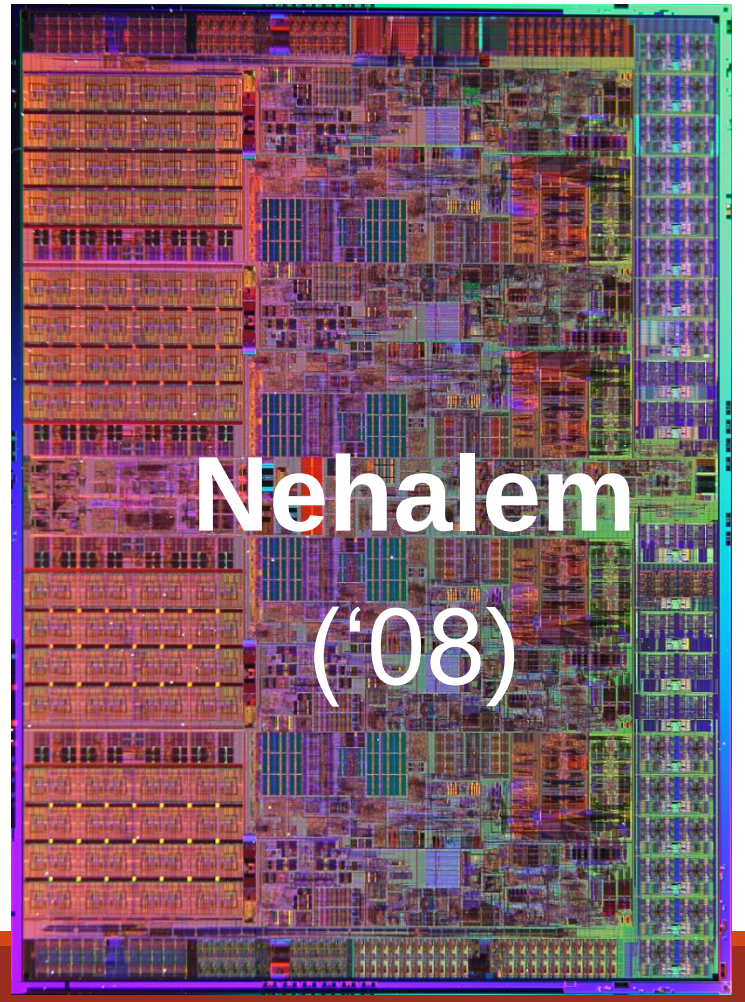
When did architects optimize multiplies?

When did architects optimize loads?

➔ **Technology has a dramatic impact on what's important in architecture.**

Technology constantly on the move!

- Not optimizing for # transistors anymore
  - Currently > 1 billion transistors/chip
- Issues:
  - Complexity
  - Power
  - Heat
  - Latency
  - Parallelism
- Huge change in thinking
  - Improve sequential vs. parallel performance?
  - Improve throughput vs. decrease power?
  - Specialized vs. general purpose?



Case study: Deep learning

- "Deep learning" (a.k.a. neural networks) are taking over the world
- …An old technique that had fallen out of favor for decades

- *What happened?*
  - 1) Big data – massive training datasets
  - 2) GPUs – **massive compute** available for **little $$**

- Now, "neural accelerators" are the hottest topic in computer architecture
  - E.g., 11 out of ~50 papers at ISCA in 2016
  - Google, Apple, Microsoft building & deploying custom hardware at scale

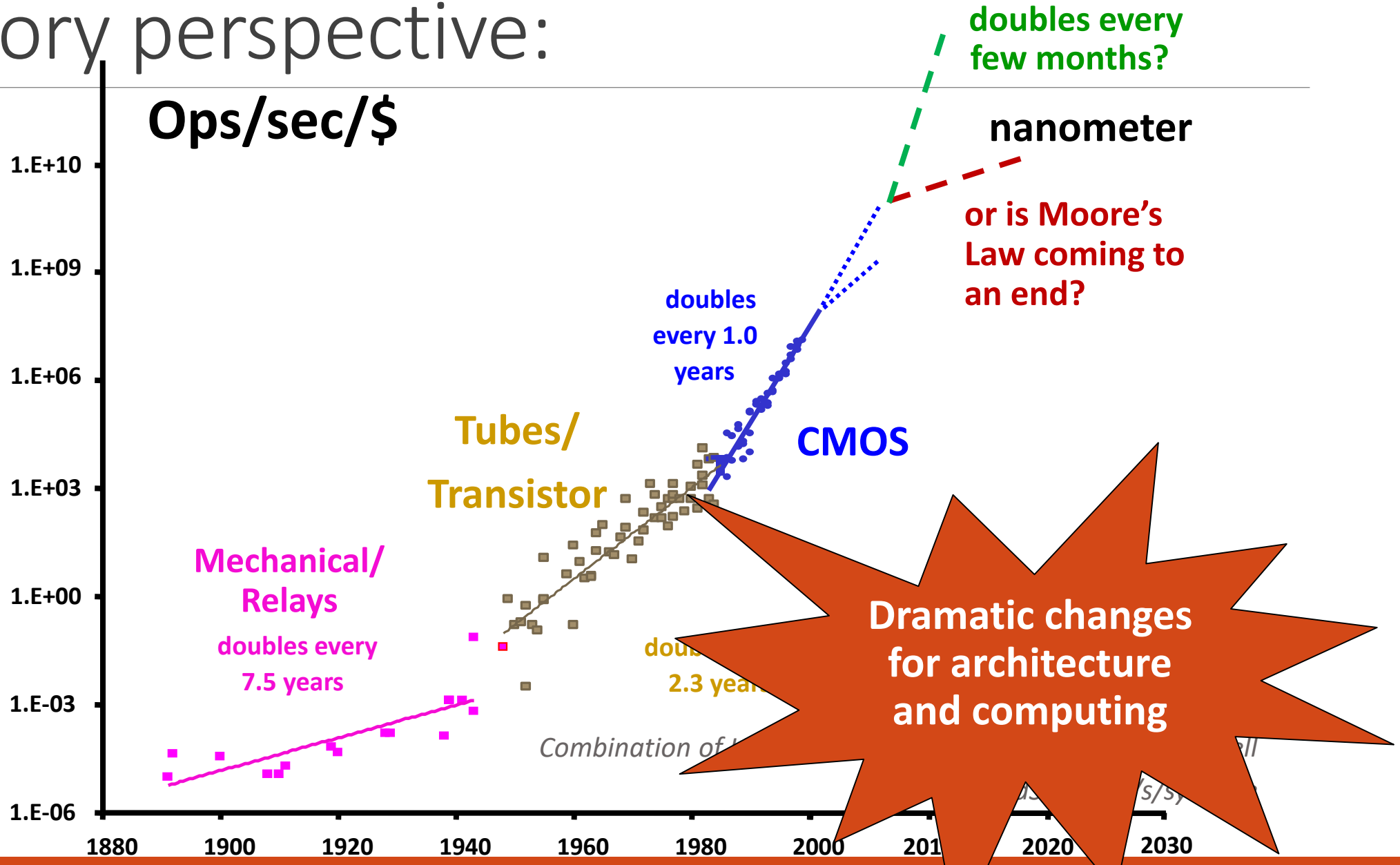
Why **you** should study computer architecture

- Understand how computers work

- Its not just how to build them:
  - Why does my program run slowly?
  - How do I increase performance?
  - How do I improve reliability?
  - Is my system secure?
  - What can I expect tomorrow?

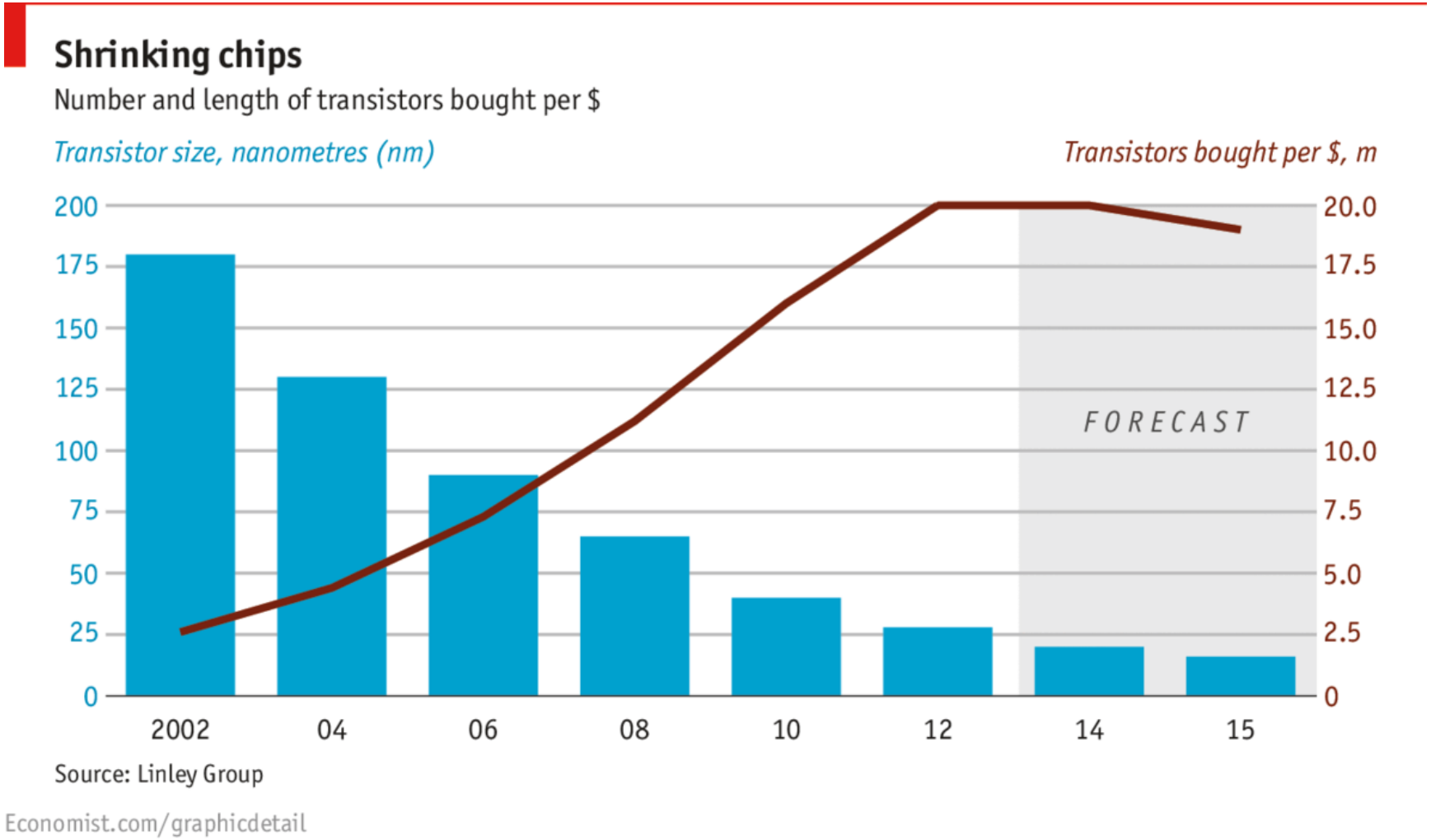
- We are at a crossroads…

History perspective:



[Gray Turing Award Lecture]

Moore's Law already finished?



- Graph: *The Economist*, April 2015

A Brief History of Computer Architecture

The microprocessor

- **Microprocessor revolution**
  - Technology threshold crossed in 1970s:
    Enough transistors (~25K) to fit a 16-bit processor on one chip
  - Huge performance advantages: fewer slow chip-crossings
  - Even bigger cost advantages: one “stamped-out” component

- Created new applications
  - Desktops, CD/DVD players, laptops, game consoles, set-top boxes, mobile phones, digital camera, mp3 players, GPS, automotive, …
- And replaced incumbents in existing segments
  - Supercomputers, “mainframes”, “minicomputers”, etc.

First microprocessor

- Intel 4004 (1971)
  - The first single-chip CPU!
  - Application: calculators
  - Technology: 10000 nm
  - 2300 transistors in 13 mm^2
  - 740 KHz, 8 or 16 cycles/instr.
    - Multiple cycles to xfer data
  - 12 Volts
  - 640-byte address space
  - 4-bit data

Reminder: This looks silly today, but it is a miraculous machine by any historical standard.

Tracing the microprocessor revolution

- How were growing transistor counts used?

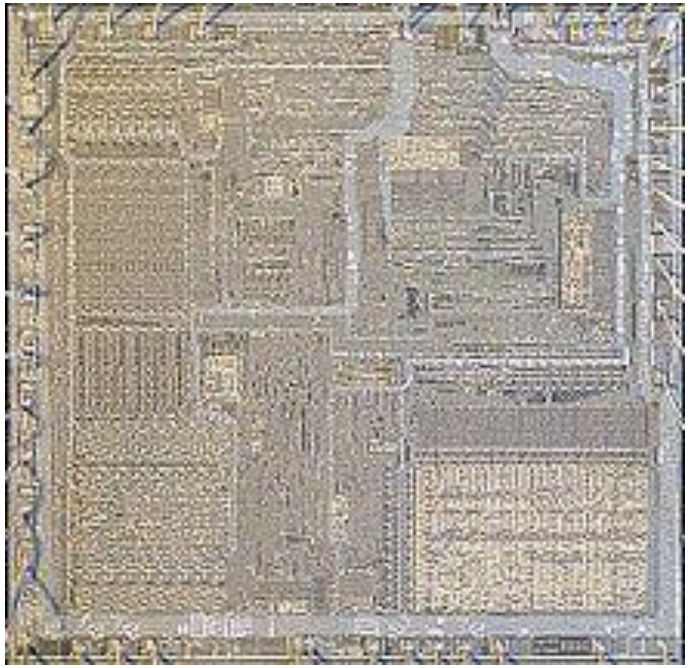
- Initially to widen the datapath
  - 4004: 4 bits → Pentium4: 64 bits

- … and also to add more powerful instructions
  - To amortize overhead of fetch and decode
  - To simplify programming (which was done by hand then)
  - To reduce memory requirements for program

  - Could get absurd: e.g., VAX “POLY” instruction

The first x86

- Intel 8086 (1978)
  - Application: microcomputers
  - Technology: 3000nm
  - 29,000 transistors in 33 mm^2
  - 5-10 MHz, 2-190 cycles/instr.
    - What took 190 cycles? (Not memory!)
  - 5 Volts
  - 1MB address space
  - 16-bit datapath
  - *Microcoded* design: each instruction invokes a *microprogram* with architecture-specific micro-instructions
    - *Idea from MIT Whirlwind in 1950s!*



Implicit parallelism

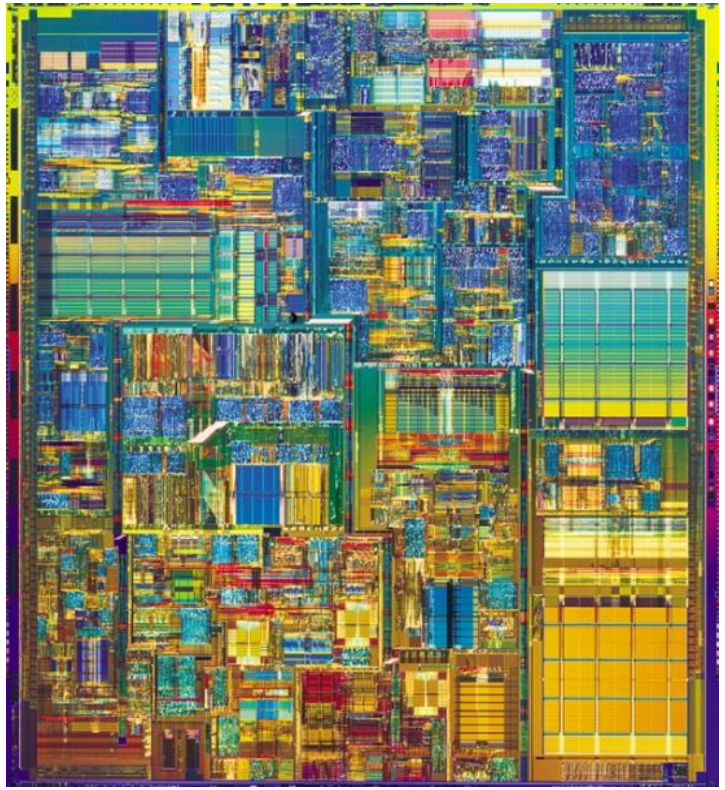
- Then to **extract implicit instruction-level parallelism (ILP)**
  - Hardware provides parallel resources, figures out how to use them
  - Software is oblivious – *for the most part!*

- Initially using pipelining …
  - Which also enabled increased clock frequency
- … caches …
  - Which became necessary as processor clock frequency increased
- … deeper pipelines and branch speculation
- … multiple instructions per cycle (superscalar)
- … dynamic scheduling (out-of-order execution)

- Meanwhile, also continued to add features, e.g., integrated floating point

Nearing the end of uniprocessors

- Intel Pentium4 (2003)
  - Application: desktop/server
  - Technology: 90nm *(1% of 4004)*
  - 55M transistors *(20,000x)*
  - 101 mm^2 *(10x)*
  - 3.4 GHz *(10,000x)*
  - **3 instrs / cycle** *(superscalar)*
  - 1.2 Volts *(1/10x)*
  - 32/64-bit data *(16x)*
  - 22-stage pipelined datapath
  - Two levels of on-chip cache
  - Data-parallel “vector” (SIMD) instructions, hyperthreading



Explicit parallelism

- Then to support **explicit data & thread-level parallelism**
  - Hardware provides parallel resources, software specifies usage
  - Why? diminishing returns on instruction-level-parallelism

- First using (sub-word) vector instructions …
  - E.g., in Intel's SSE, one instruction does four parallel multiplies

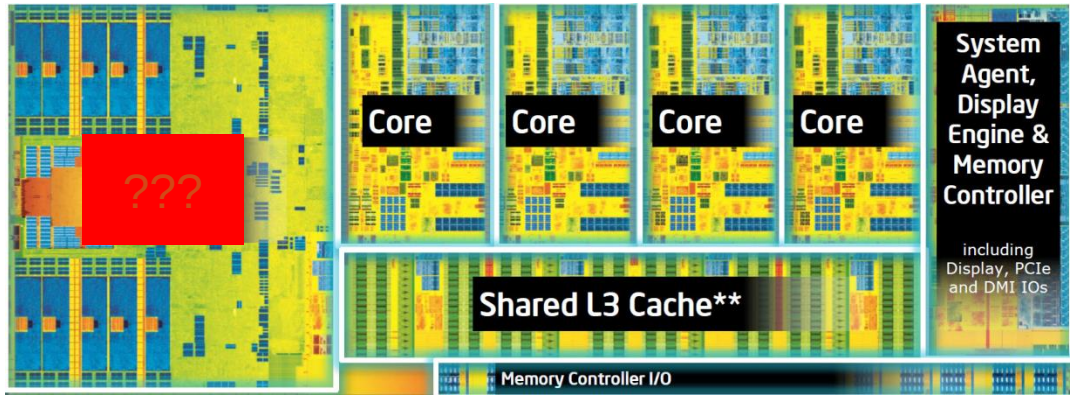
- … adding support for multi-threaded programs …
  - Coherent caches, hardware synchronization primitives

- … multiple concurrent threads …
  - First single-core multi-threading, now multi-core

- New architectures, e.g., GPUs
  - GPUs becoming more programmable
  - ➔ convergence between CPUs and GPUs (e.g., Intel's Xeon Phi)

Multicore

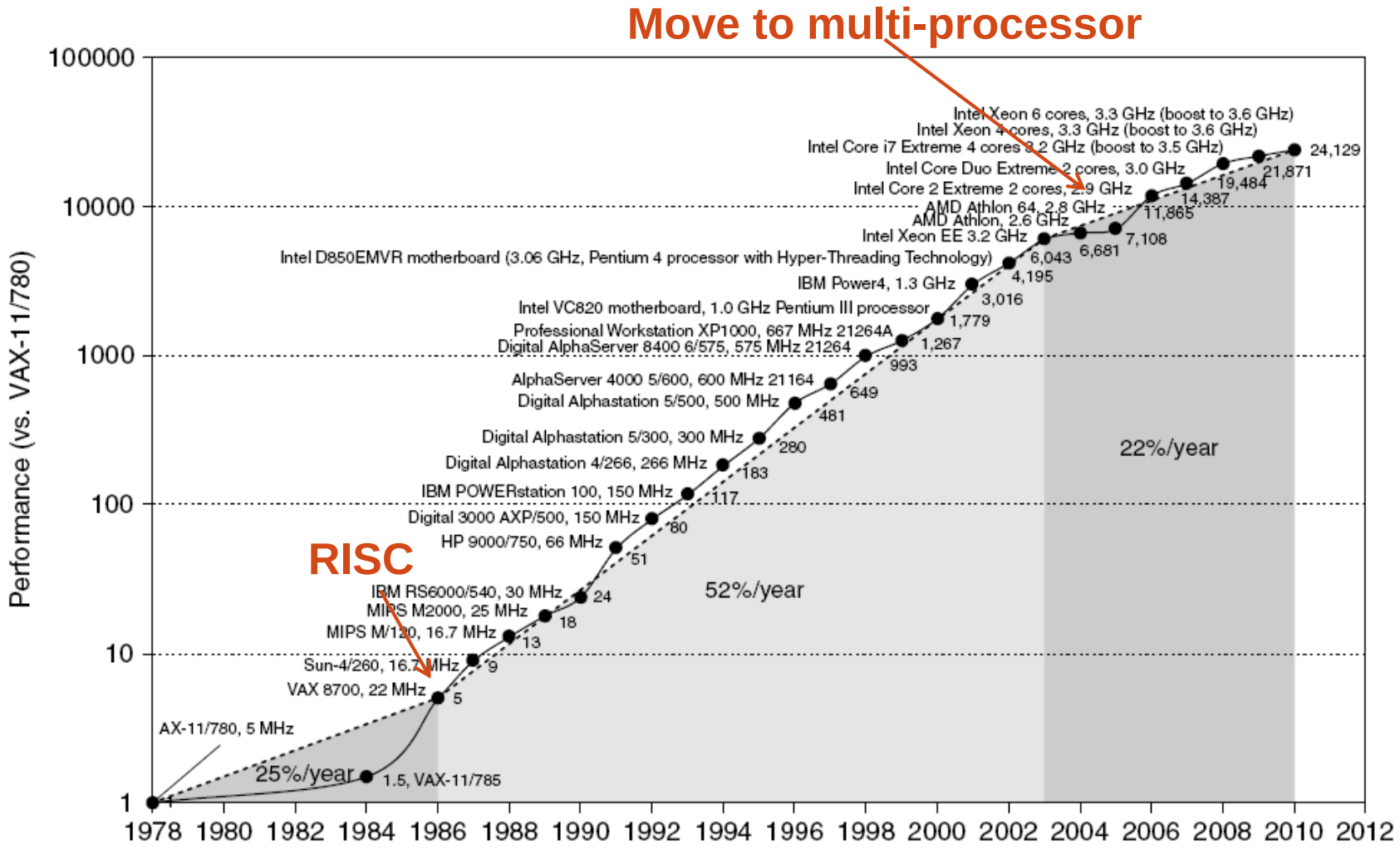
- Intel Core i7 (2013)
  - Application: desktop/server
  - Technology: 22nm *(25% of P4)*



  - 1.4B transistors *(30x)*
  - 177 mm^2 *(2x)*
  - 3.5 GHz to 3.9 Ghz *(~1x)*
  - 1.8 Volts *(~1x)*

  - 256-bit data *(2x)*
  - 14-stage pipelined datapath *(0.5x)*
  - 4 instructions per cycle *(~1x)*
  - Three levels of on-chip cache *(1.5x)*
  - Data-parallel “vector” (SIMD) instructions, hyperthreading
  - **Four-core multicore** (4x)

Performance over the years



Specialization

Hard to get parallel speedup for many applications

- Writing parallel software is hard!
- Extra cores give little benefit
- ➔ Can we put those transistors to better use?

Specialized processors are much more efficient

- Customized datapath for common operations (many instructions ➔ 1 cycle)
- Customized memories keep data near where its used
- Eliminate features that aren't needed (less power)

…But only worthwhile for the most important workloads

- Design & verification is expensive
- Software now must support custom hardware
- Wastes chip real estate when idle

System-on-chip

Qualcomm Snapdragon 835

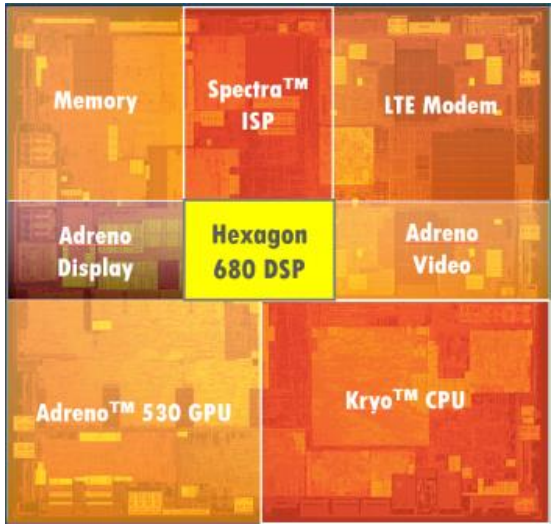
- Application: Mobile
- Technology: 10nm

- ARM CPUs – heterogeneous “big.LITTLE” design
  - 4 “performance” cores – 2.45 GHz, 2MB L2 cache
  - 4 “efficiency” cores – 1.9 GHz, 1MB L2 cache
  - “Performance” cores are 20% faster; “efficiency” cores used 80% of the time

- Graphics processing unit (GPU)
  - ~650 MHz
  - Specialized floating-point datapath, e.g., for interpolation of textures
  - Data-parallel: 16 pixels / clock
  - Processor dynamically finds & schedules work (“warp scheduling”)

- Digital signal processor (DSP)
  - Data-parallel SIMD architecture with 4 instructions / cycle
  - No floating-point
  - Compiler statically schedules parallelism (“VLIW”)

- Other custom accelerators (camera, modem, etc)



*Snapdragon 820
(only die shot I could find)

Architectures today

Multicore CPUs (e.g., Intel Xeon)

- Traditional hard-to-parallelize code – web serving
- Renewed focus on CPU microarchitecture – sequential performance still matters!

GPUs (e.g., Nvidia)

- “Embarrassingly parallel” code – science, graphics, DNNs
- Increasing programmability, converging towards traditional vector design

System-on-chip & domain-specialized accelerators

- Energy-efficiency – embedded, mobile, (datacenter – Google’s TPU???)
- Lots of open questions ...
  - How many accelerators do we need?
  - Which ones?
  - How specialized should they be?

What computer architects do

Given constraints of

- Technology
- Application

Use essential themes

- Locality (e.g., caching)
- Prediction / speculation
- Pipelining
- Parallelism
- Virtualization / indirection
- Specialization

And, always, using abstraction…

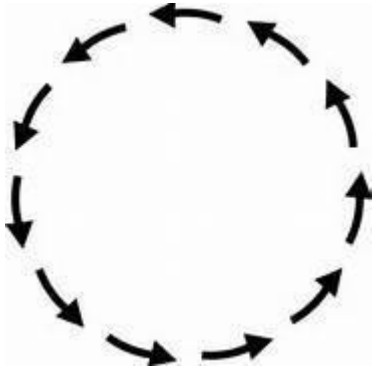
What computer architects do

Given constraints of

- Technology
- Application

Use essential themes

- Locality (e.g., caching)
- Prediction / speculation
- Pipelining
- Parallelism
- Virtualization / indirection
- Specialization



Can often seem like going in circles…

What computer architects do

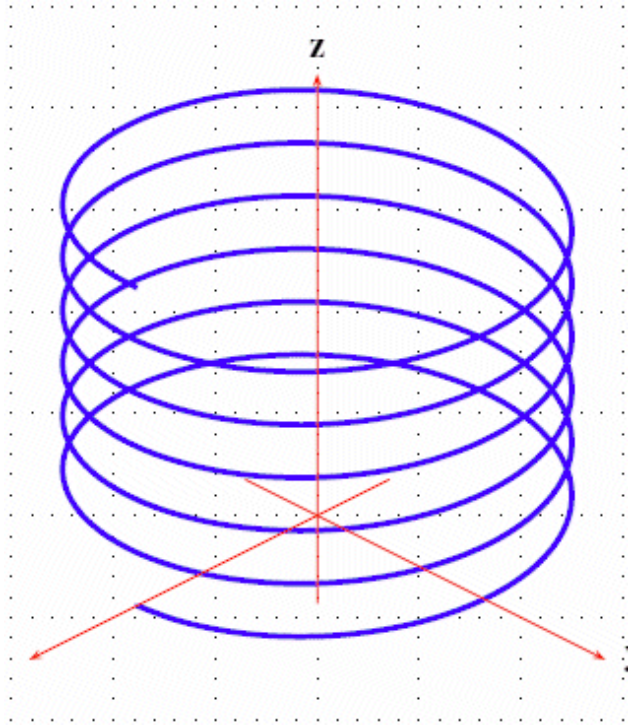
Given constraints of

- Technology
- Application

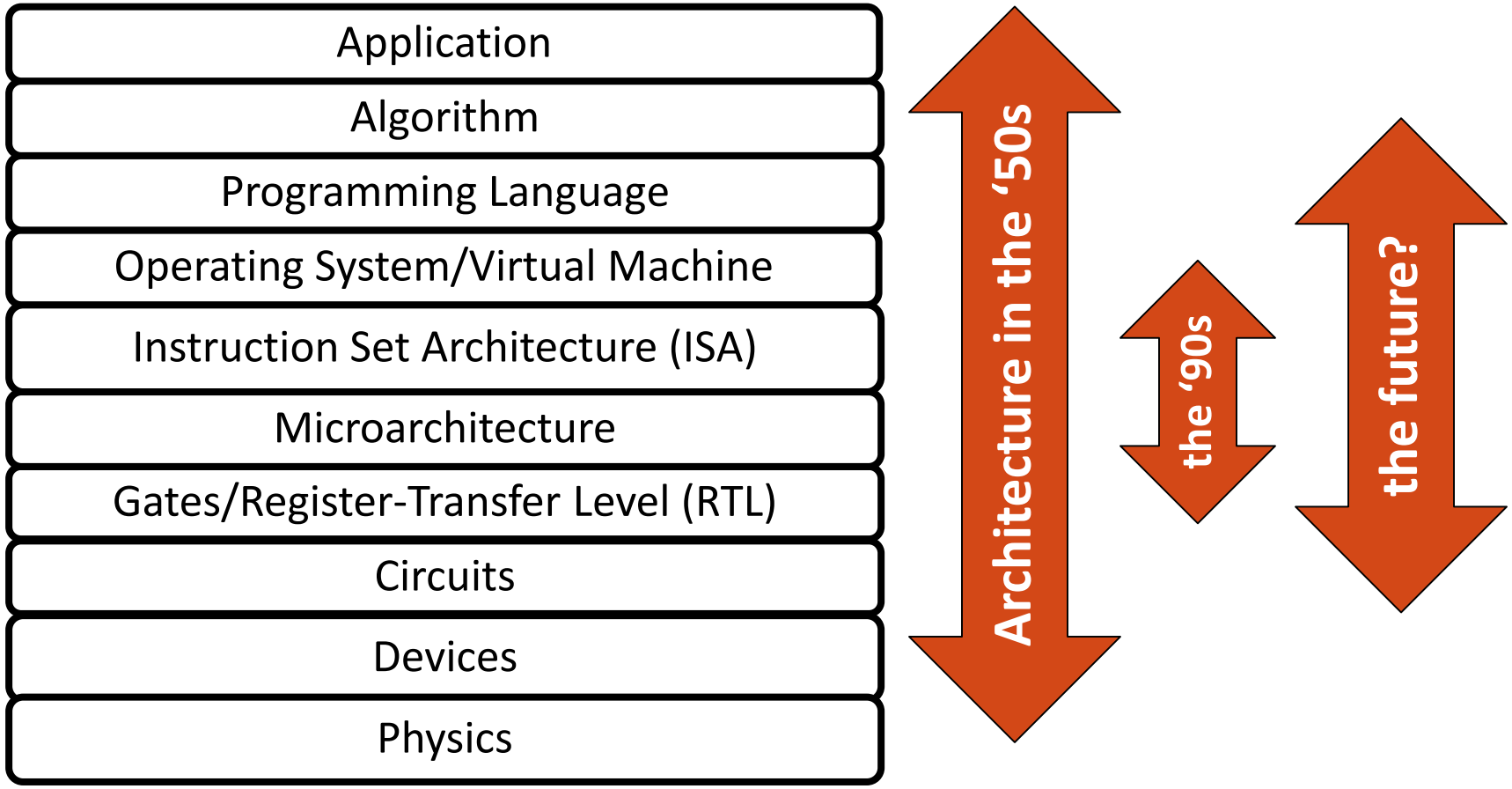
Use essential themes

- Locality (e.g., caching)
- Prediction / speculation
- Pipelining
- Parallelism
- Virtualization / indirection
- Specialization

Can often seem like going in circles… but there is progress!



Abstraction layers in modern systems



What would you do with one trillion transistors?

Course logistics

Logistics

- Lectures
- Paper readings & reviews
- Paper presentations
- Labs
- Project
- Exams

Lectures

- Please come
- Please come prepared
- Participation (10% of grade)

- Lecture schedule and slides are online
  - Until Spring break: Memory hierarchy & parallelism
  - After Spring break: Microarchitecture & specialization

Paper readings & reviews

Paper readings instead of textbook

Reviews/summaries

- 10% of grade – more importantly, an essential skill
- Submit before class on Gradescope

Review contents:

- Identify essential (good) idea
- What is the goal of the paper?
- How does it relate to other papers/ideas?
- What questions does it raise?
- How are its ideas evaluated?
- What are the results? Broader conclusions?

- **At most half a page**
- **Include 3 questions you would ask authors**

Paper presentations

Logistics

- Pick a topic by 2/1 – hopefully related to project
- Groups of 2-3
- All group members read all papers on chosen topic
- 20 minute presentation in class

Presentation

- Background question/problem for topic
- What are the good ideas?
- How are they evaluated?
- Follow-up ideas
- More advice on website

Labs

2 labs early in semester

- 5% grade each

Work in groups of 2-3

Goal:

- Become familiar with some tools
- Understand performance measurement
- Understand optimization aka
  How architecture affects use

First lab out this week! (more on Thursday)

Project

Main focus, takes ~half of semester!

- 40% of grade

Do some real research

Work in groups of 2-3

Timeline:

Proposal	3/10
Milestone presentation	4/10, 4/12, 4/17
Poster	5/1
Final report	5/8

Exams

2 in-class exams

- 90 mins each
- 15% grade each

Closed book

Not cumulative

Exam 1:	3/8	(before Spring break)
Exam 2:	5/3	(last class)

Waitlist

Keeping class at $\leq$25 people

- Only 1 TA

Will accept students off waitlist in following order:

- Urgent circumstance (e.g., must take this class right now to graduate)
- Students who have kept up with readings & assignments
  - In waitlist order

You should know within a few weeks

Overview

- Architecture: physics → applications

- Constantly changing field:
  - New problems
  - New solutions
  - …But many common patterns and useful insights

- One must understand architecture to understand computer systems