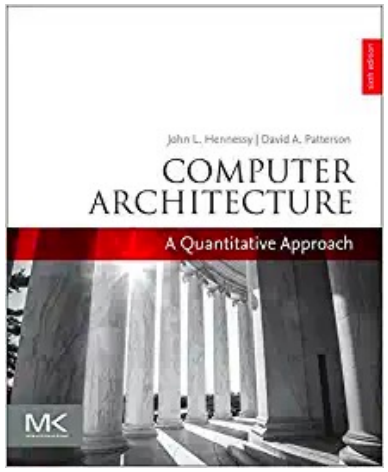


Chapter 1

Fundamentals of Quantitative Design and Analysis

Part 1: Overview

"I think it's fair to say that personal computers have become the most empowering tool we've ever created. They're tools of communication they're tools of creativity, and they can be shaped by their user." – Bill Gates, February 2004

Acknowledgements

- Thanks to many sources for slide material

© 1990 Morgan Kaufmann Publishers, © 2001-present Elsevier
Computer Architecture: A Quantitative Approach by J. Hennessy & D. Patterson

© 1994 Morgan Kaufmann Publishers, © 2001-present Elsevier
Computer Organization and Design by D. Patterson & J. Hennessy

© 2002 K. Asinovic & Arvind, MIT

© 2002 J. Kubiatowicz, University of California at Berkeley

© 2006, © 2010 No Starch Press for Inside the Machine by J. Stokes

© 2007 W.-M. Hwu & D. Kirk, University of Illinois & NVIDIA

© 2007-2010 J. Owens, University of California at Davis

© 2010 CRC Press for Introduction to Concurrency in Programming Languages by M. Sottile, T. Mattson, and C. Rasmussen

© 2017, IBM POWER9 Processor Architecture by Sadasivam et al., IBM

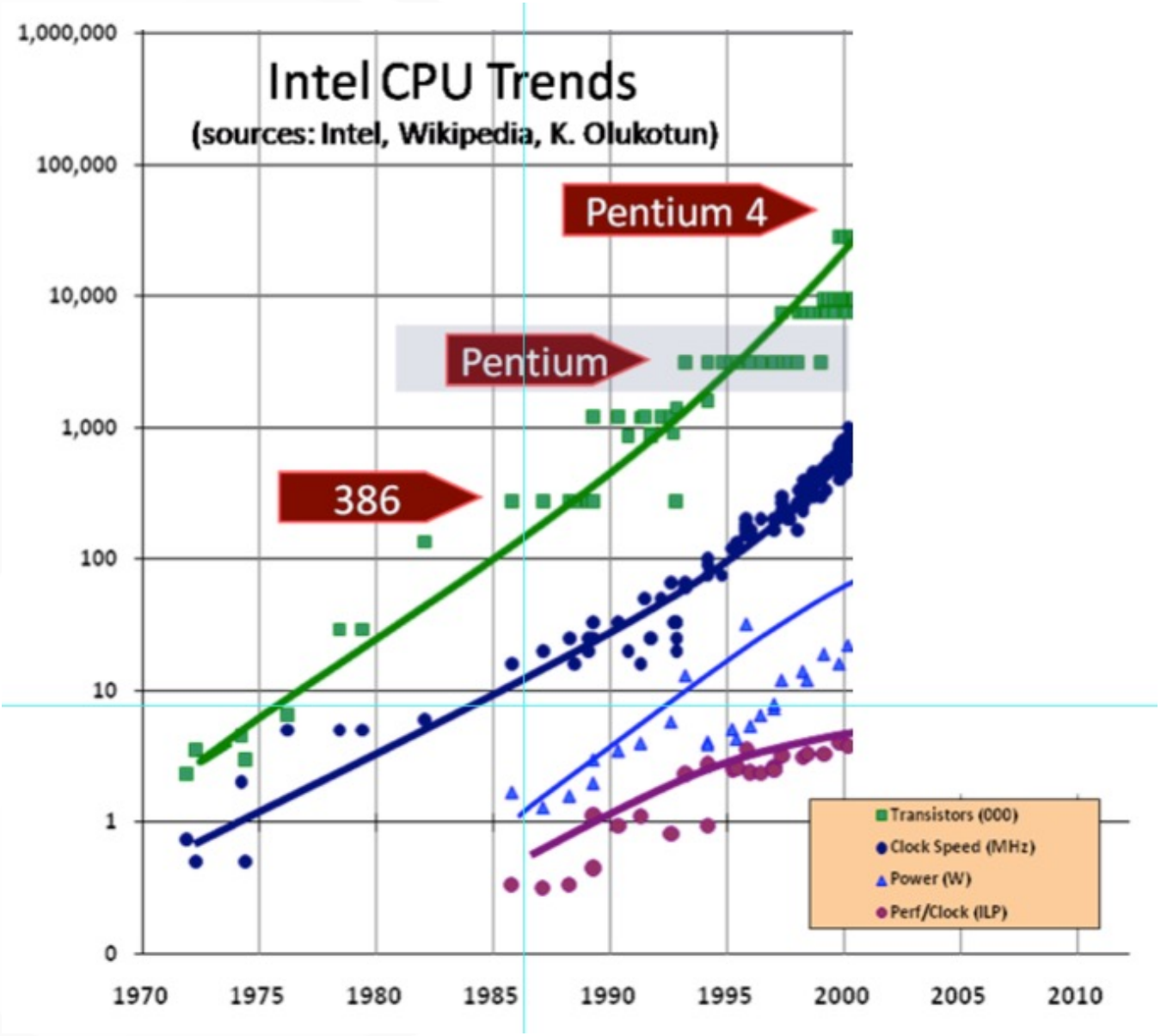
© 2016, © 2019 POWER9 Processor User's Manual, IBM

© The OpenPOWER Foundation

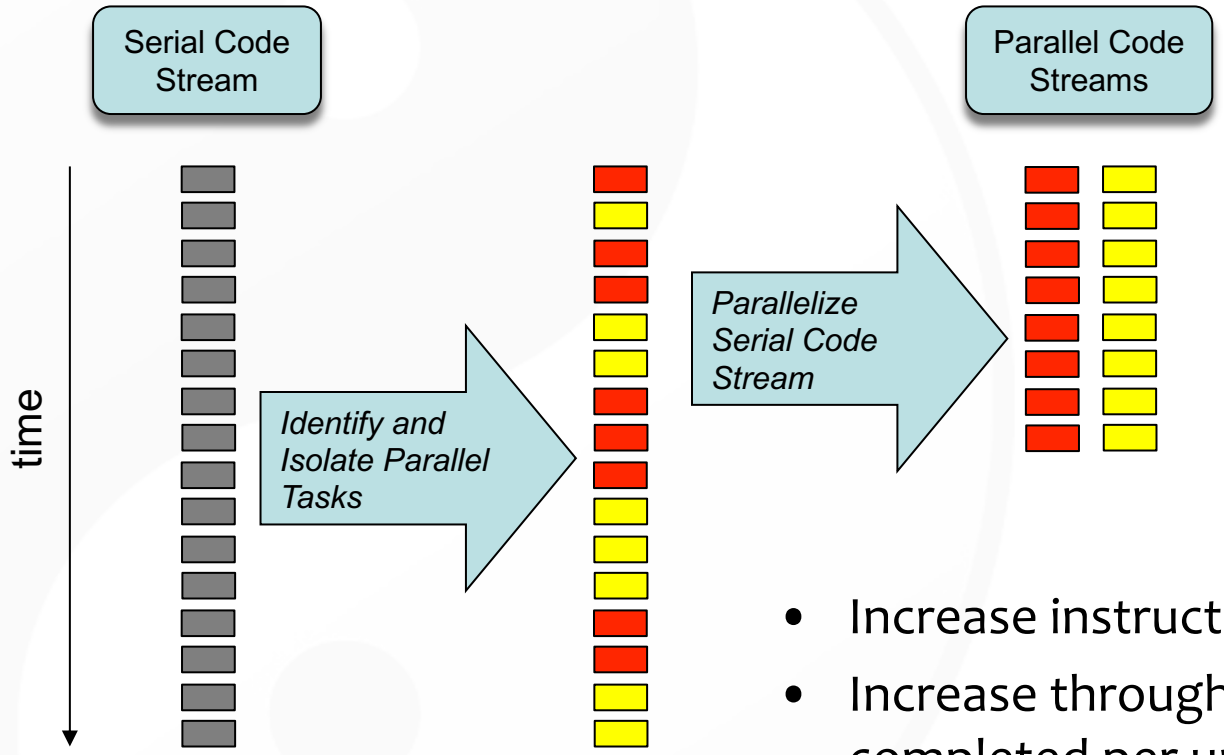
Computer Technology

- Performance Improvements
  - Improvements in semiconductor technology
    - Feature size *and* clock speed
  - Improvements in computer architectures
    - Enabled by HLL compilers *and* UNIX
    - RISC (Reduced Instruction Set Architecture)
      - Invented by John Cocke @ IBM (early 1970s)
        - » Resulted in general-purpose IBM 801
        - » Led to the RISC architectural revolution in 1980s (Patterson & Hennessy)
- End Result
  - Lightweight computers
  - Productivity-based managed/interpreted programming languages

10 µm – 1971
6 µm – 1974
3 µm – 1977
1.5 µm – 1982
1 µm – 1985
800 nm – 1989
600 nm – 1994
350 nm – 1995
250 nm – 1997
180 nm – 1999
130 nm – 2001
90 nm – 2004
65 nm – 2006
45 nm – 2008
32 nm – 2010
22 nm – 2012
14 nm – 2014
10 nm – 2016–2017
7 nm – 2018–2019
5 nm – 2020–2021


Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)
Pentium 4
Pentium
386
Transistors (000)
Clock Speed (MHz)
Power (W)
Perf/Clock (ILP)
1,000,000
100,000
10,000
1,000
100
10
1
0
1970
1975
1980
1985
1990
1995
2000
2005
2010

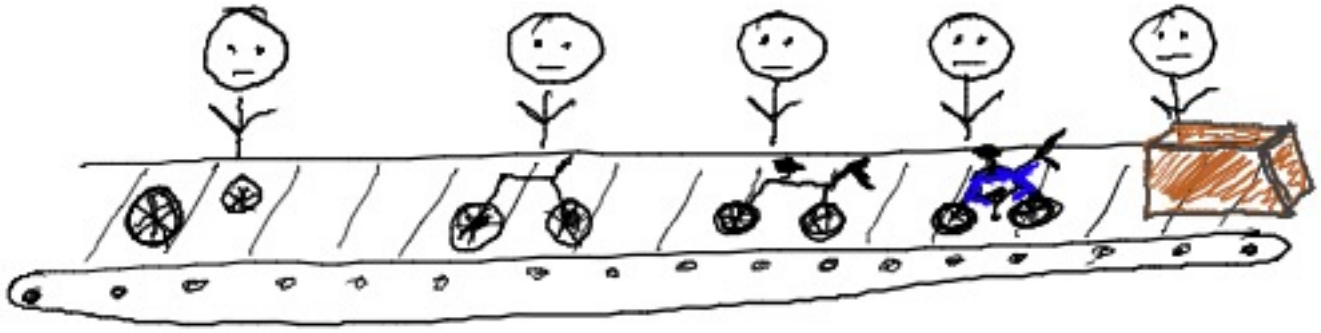
How to Improve Performance



- Increase instructions per clock cycle.
- Increase throughput or work completed per unit time.
- Lower latencies intrinsic in the system that limit the above metrics.

How To Increase Instructions Per Clock Cycle?

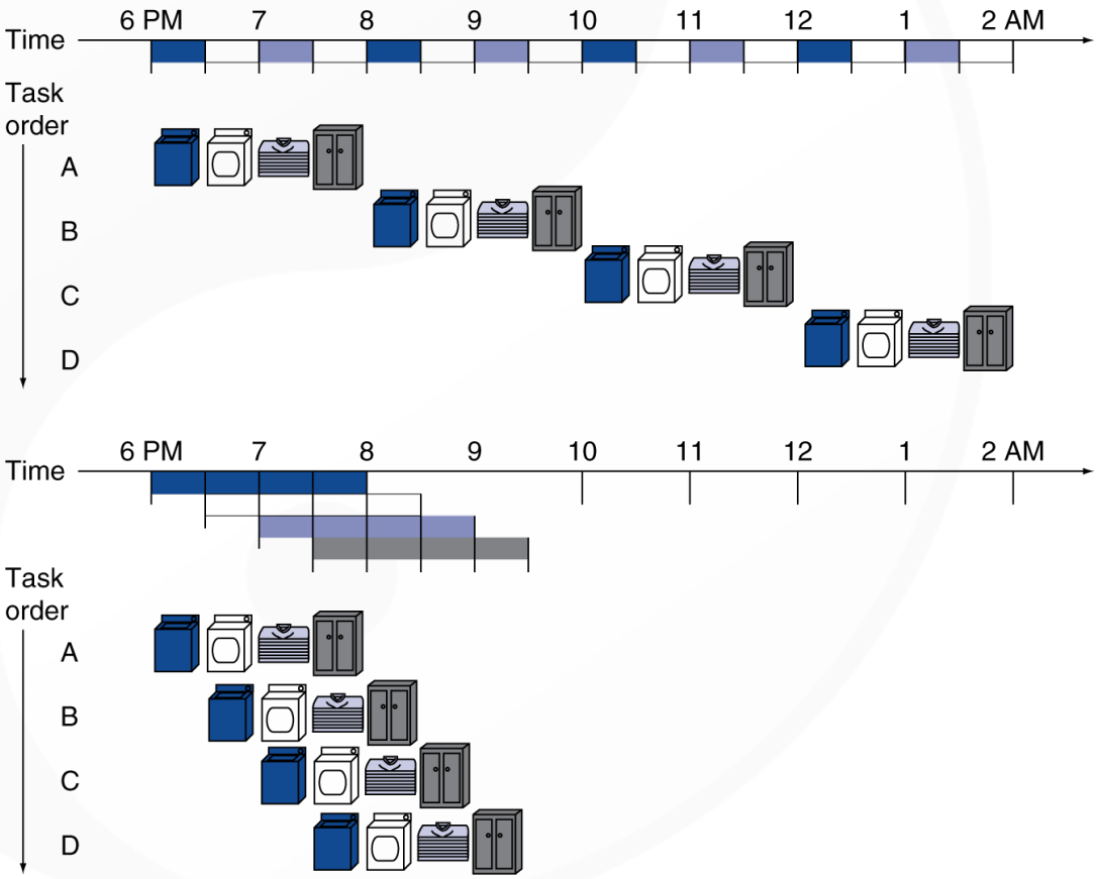
- Pipelining
  - Breakdown complex instructions into a set of smaller steps that are executed in order like a factory assembly line.



Source: Matthew Sottile

Pipelining: Analogy

- Pipelining laundry overlaps execution
  - Parallelism improves performance



One load = 2 hours

Four loads:

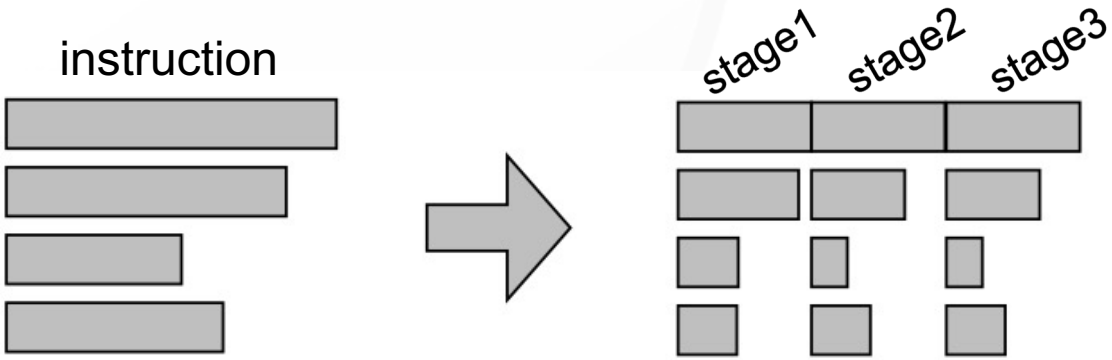
- serial throughput:
  0.5 load/hour
- pipelined throughput:
  1.14 load/hour
- speedup:
  8 / 3.5 ≈ 2.3

Non-stop speedup:

$2n / (0.5n + 1.5) \approx 4$

Pipelining: Evolution

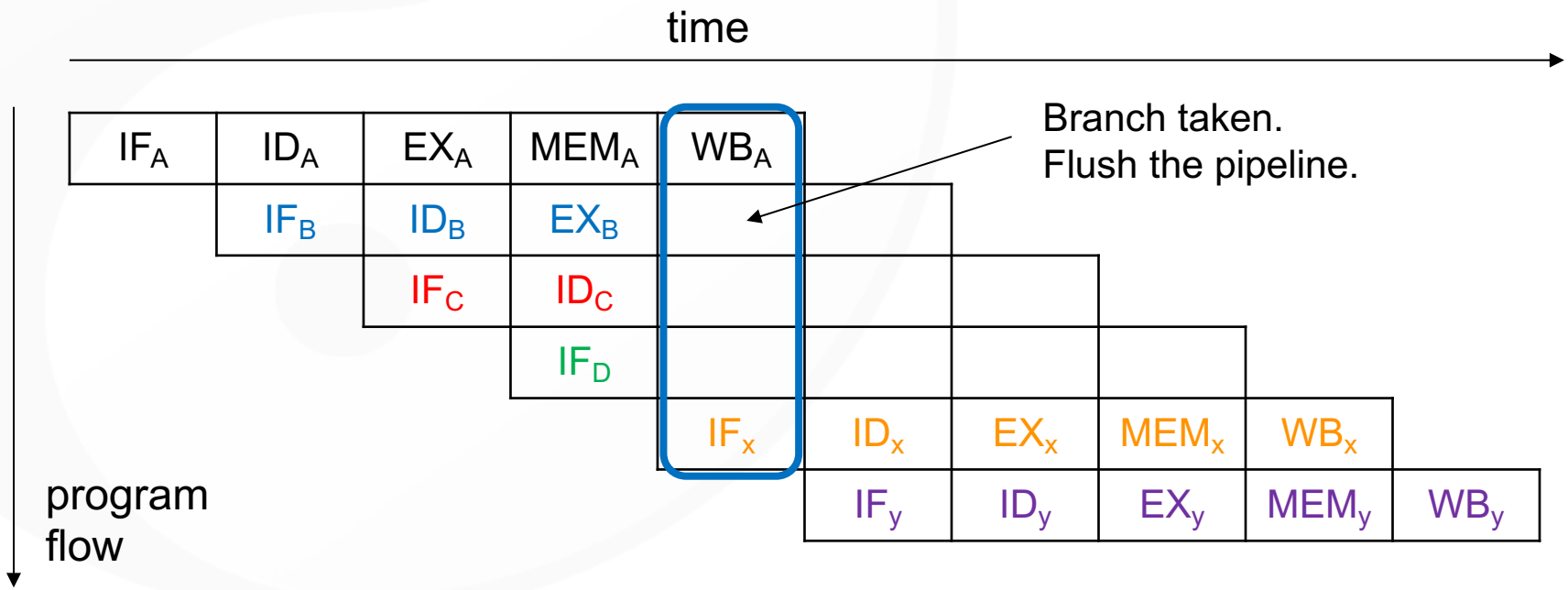
- Increase # of steps, thus decreasing complexity of each step and allowing each step to complete faster.
- Fine-grained steps reduce the difference between times in each stage for instructions of differing complexity.
- If all instructions use the same pipeline, each instruction takes effectively the same amount of time to complete.



Pipelining in Practice

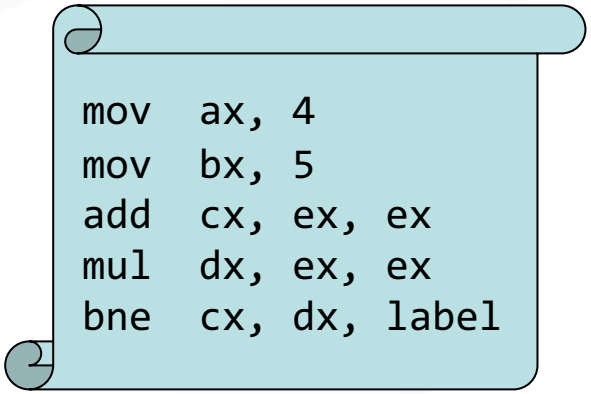
PROGRAM
instr A
branch instruction → instr B
instr C
instr D
instr E
instr F
:
instr W
instr X
instr Y

- A fully saturated pipeline can ideally yield one completed instruction per cycle.
- As pipelines get deeper and deeper, more important than ever to avoid bubbles or pipeline flushes that result in an increase in average cycles per instruction.



Pipelining Requires ILP

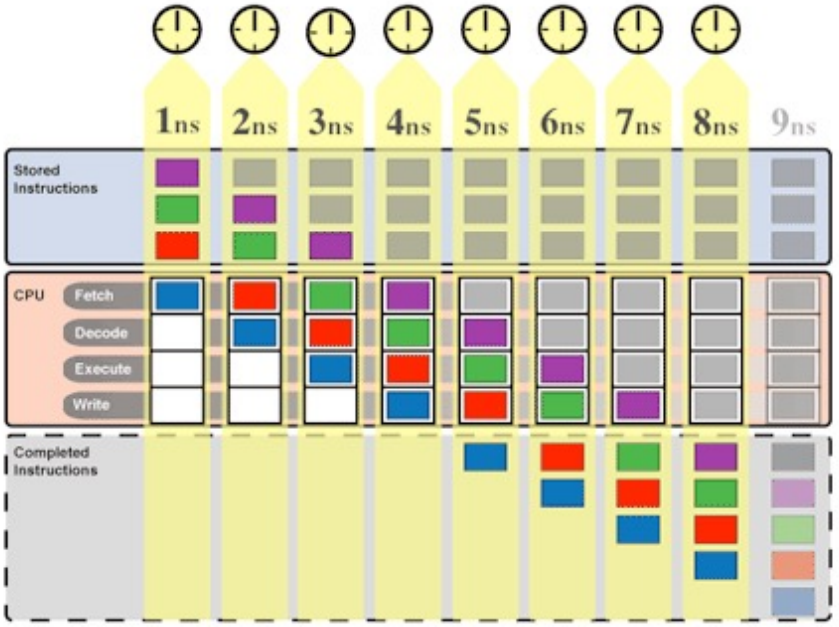
- Why does pipelining work?
- Instruction-level parallelism (ILP)
  - Non-trivial portions of program code are composed of sequences of instructions that can be reordered and executed in parallel without impacting the output of the program.



- The pipeline allows multiple instructions to be “in-flight” at any given time.

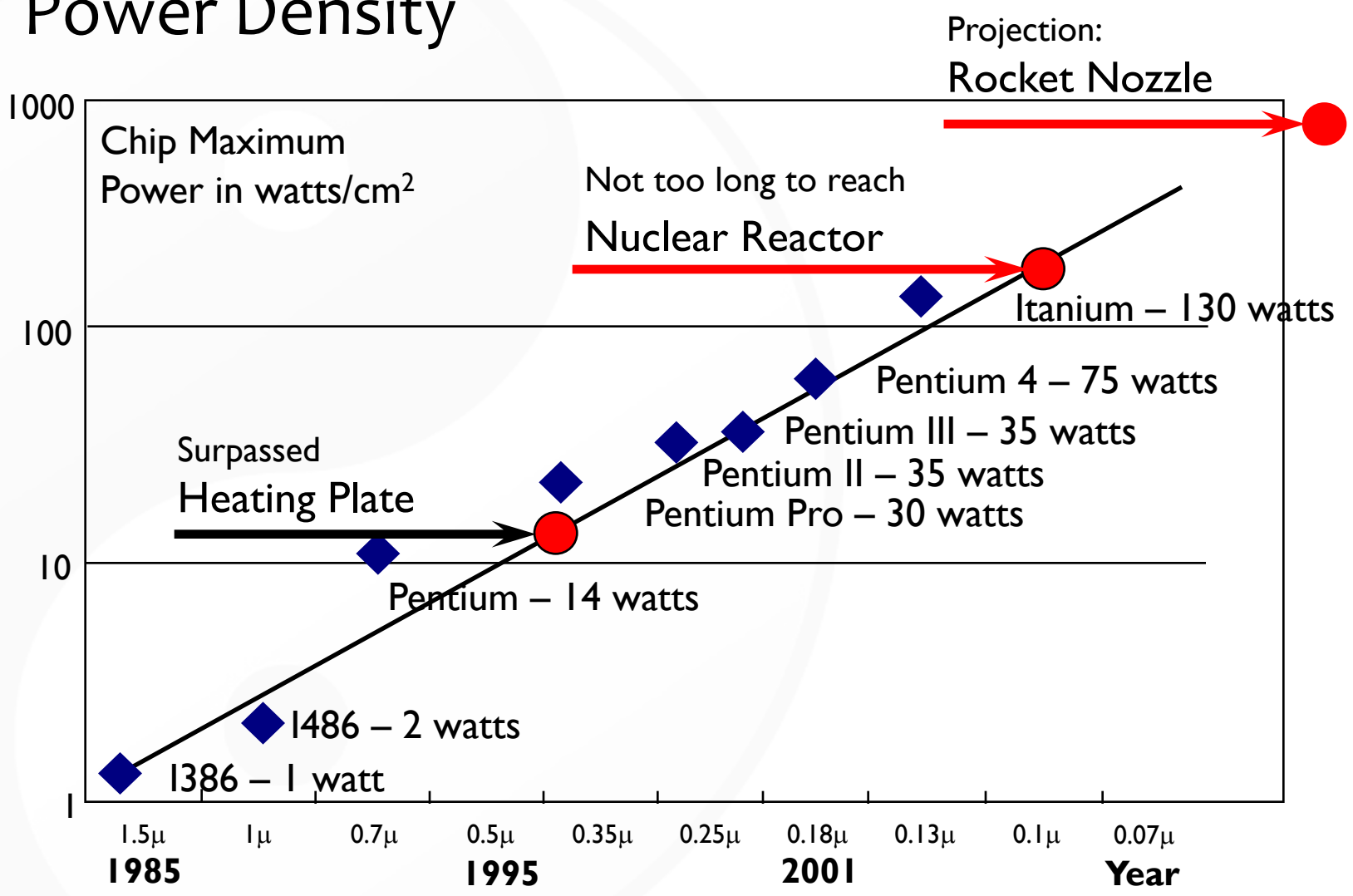
The MHz Race …

- Industry got away with increases in pipeline depth and related hardware complexity (see *Intel CPU Trends* slide) to ramp-up processor clock speeds.



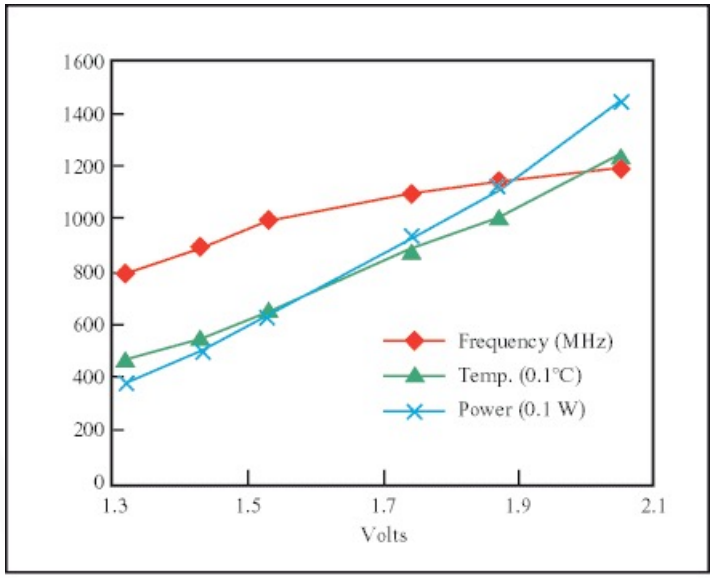
- "Houston … we have a problem …"
  - *Power and cooling are design constraints of equal importance to performance now.*
    - Cooling: amount of cooling necessary, type of cooling, packaging.
    - Power: battery requirements, electric bills.

Power Density



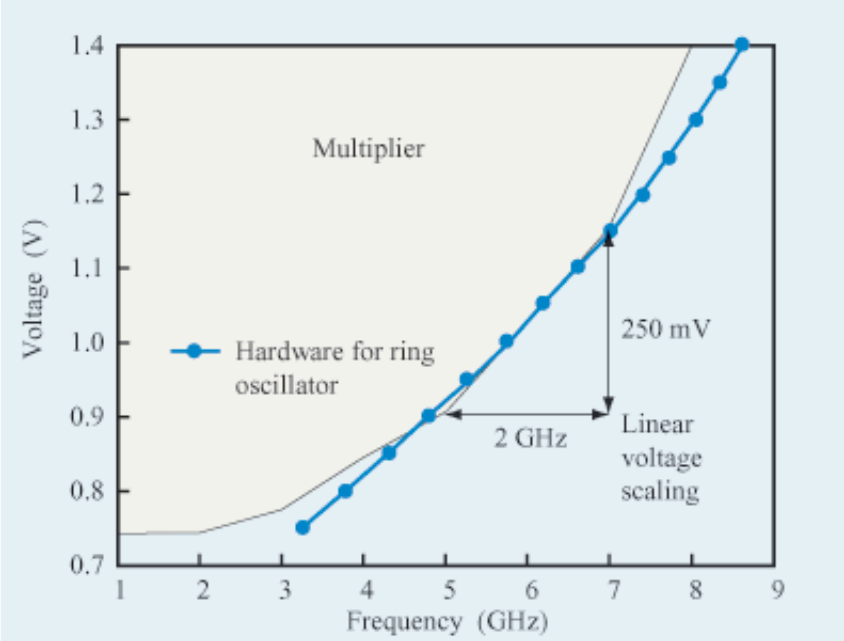
Source: Fred Pollack, Intel. New Microprocessor Challenges in the Coming Generations of CMOS Technologies, MICRO32 and Transmeta

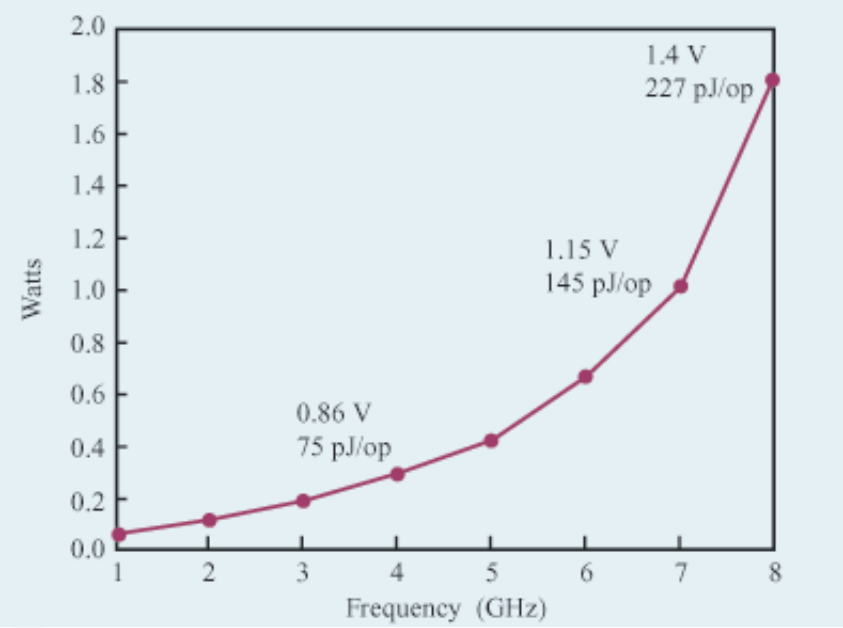
Power



- Power is really a function of …
  - Die size
    - More transistors and wires to feed
  - Frequency *f*
    - How often do you need to feed them
  - Voltage *V*
    - At what level do you need to feed them
- Pipelining: A double-edged sword
  - Better Performance (Maybe) and Higher Power Consumption
    - Increase # of small steps to realize an instruction.
      More small steps = deeper pipeline = **higher *f* and *V*,** *thus enabling better performance* ***but*** *…*
    - Better performance *only* if pipeline can be kept full. Difficult to do as pipelines get deeper.

Frequency, Voltage, and Power



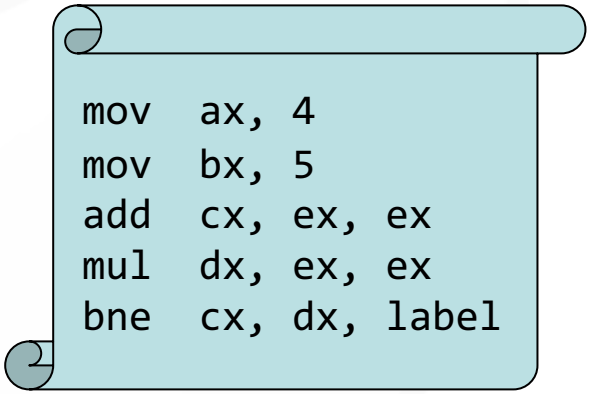


Solutions to Power & Cooling

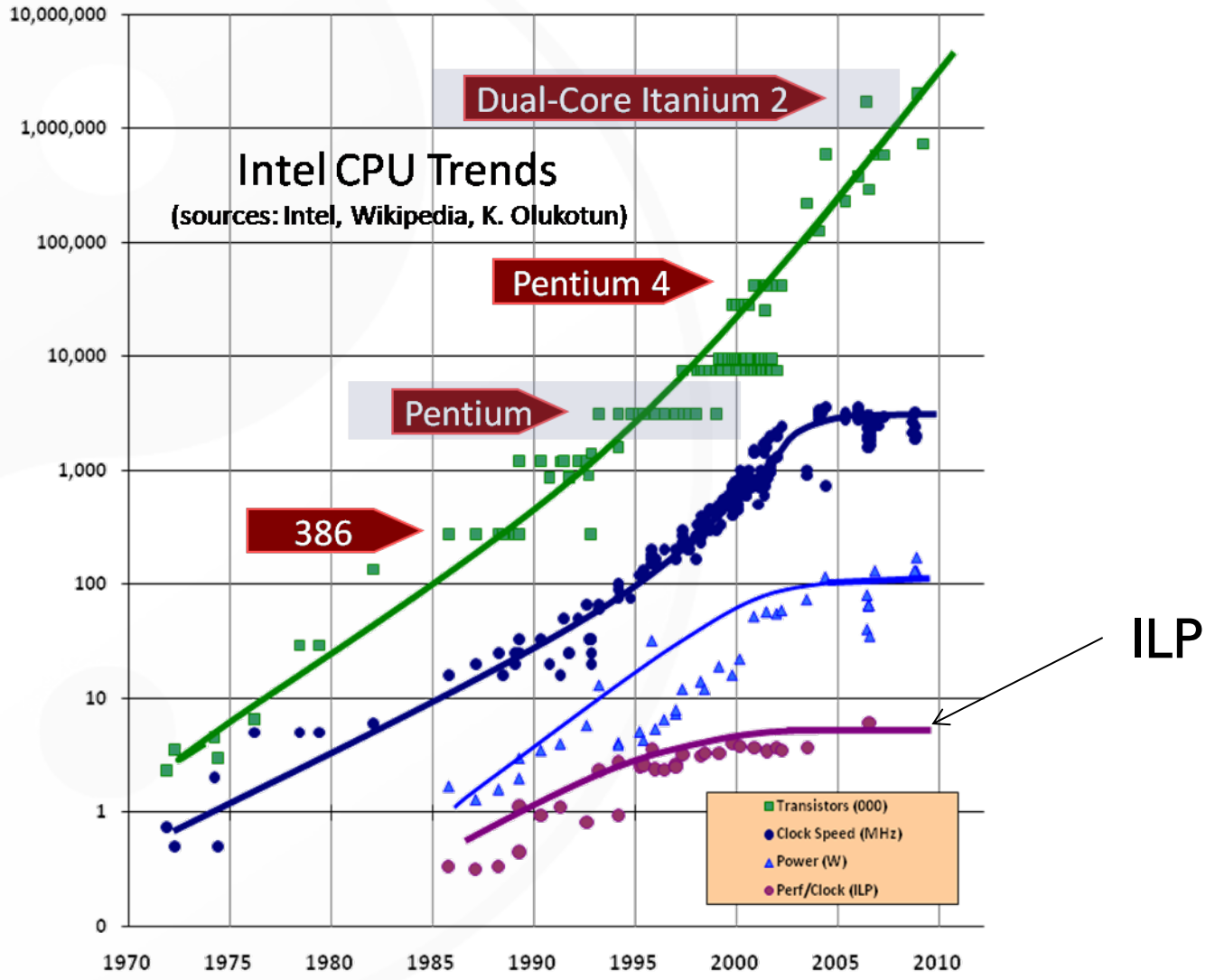
- Reduce component count (transistors and wires)
  - Reduce *complexity*, e.g., ARM and Transmeta CPUs
- Shrink components
  - Moore's Law: An observation that the number of transistors on a chip doubles every year while costs are halved.
  - Caveat: The End of Moore's Law?
    - 2019: Intel @ 10nm. AMD & Apple @ 7nm.
      - Further shrinking has gotten more complicated.
      - No transistor shrink from Intel between 2014-2019.
- Reduce leakage of transistors
- Manage power intelligently via software
  - Example: C. Hsu & W. Feng, "A Power-Aware Run-Time System for High-Performance Computing," *ACM/IEEE SC 2005*, November 2005.

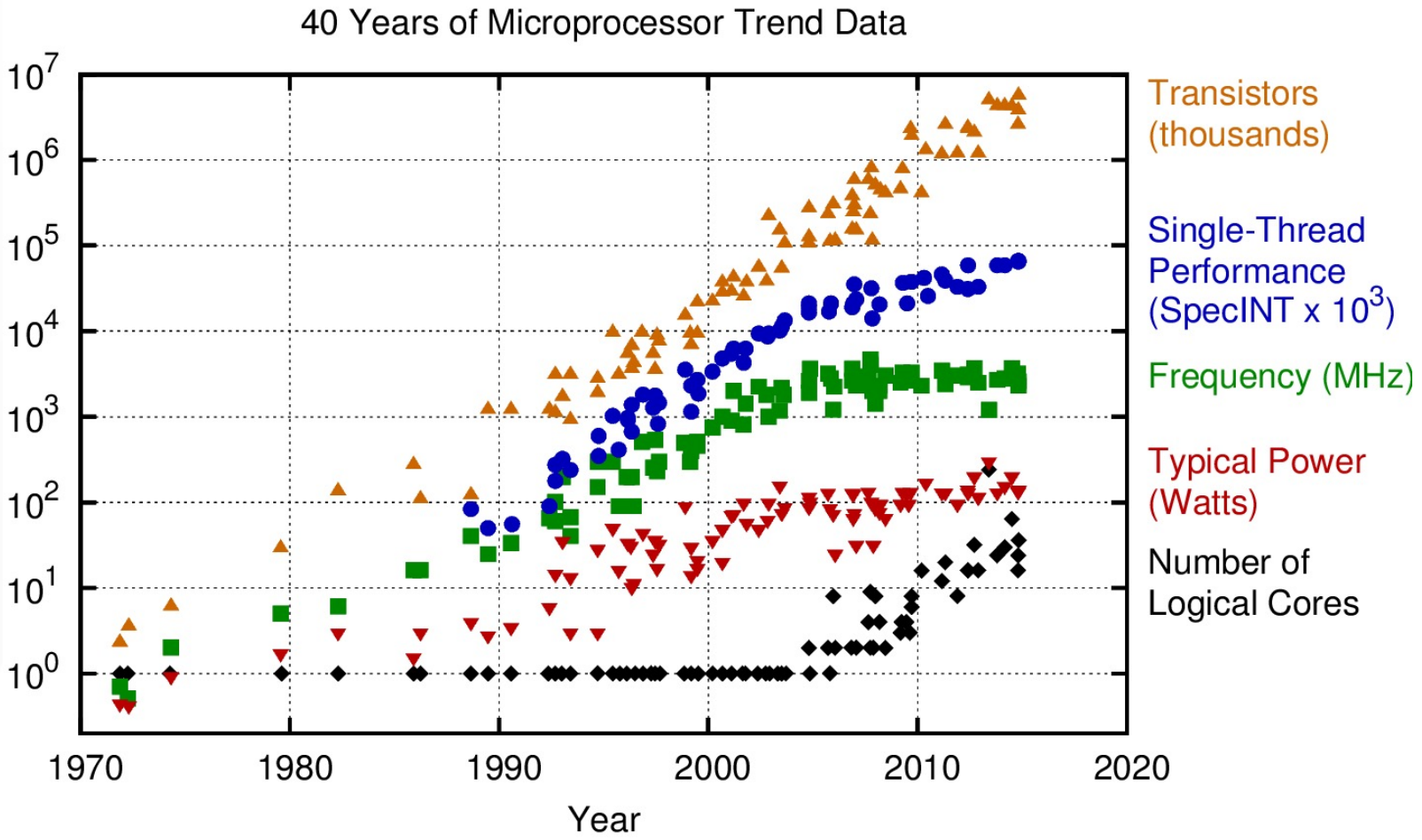
Recall: Pipelining Requires ILP

- Why does pipelining work?
- Instruction-level parallelism (ILP)
  - Non-trivial portions of program code are composed of sequences of instructions that can be reordered and executed in parallel without impacting the output of the program.



- The pipeline allows multiple instructions to be "in-flight" at any given time.


Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)
Dual-Core Itanium 2
Pentium 4
Pentium
386
ILP
Transistors (000)
Clock Speed (MHz)
Power (W)
Perf/Clock (ILP)
10,000,000
1,000,000
100,000
10,000
1,000
100
10
1
0
1970
1975
1980
1985
1990
1995
2000
2005
2010



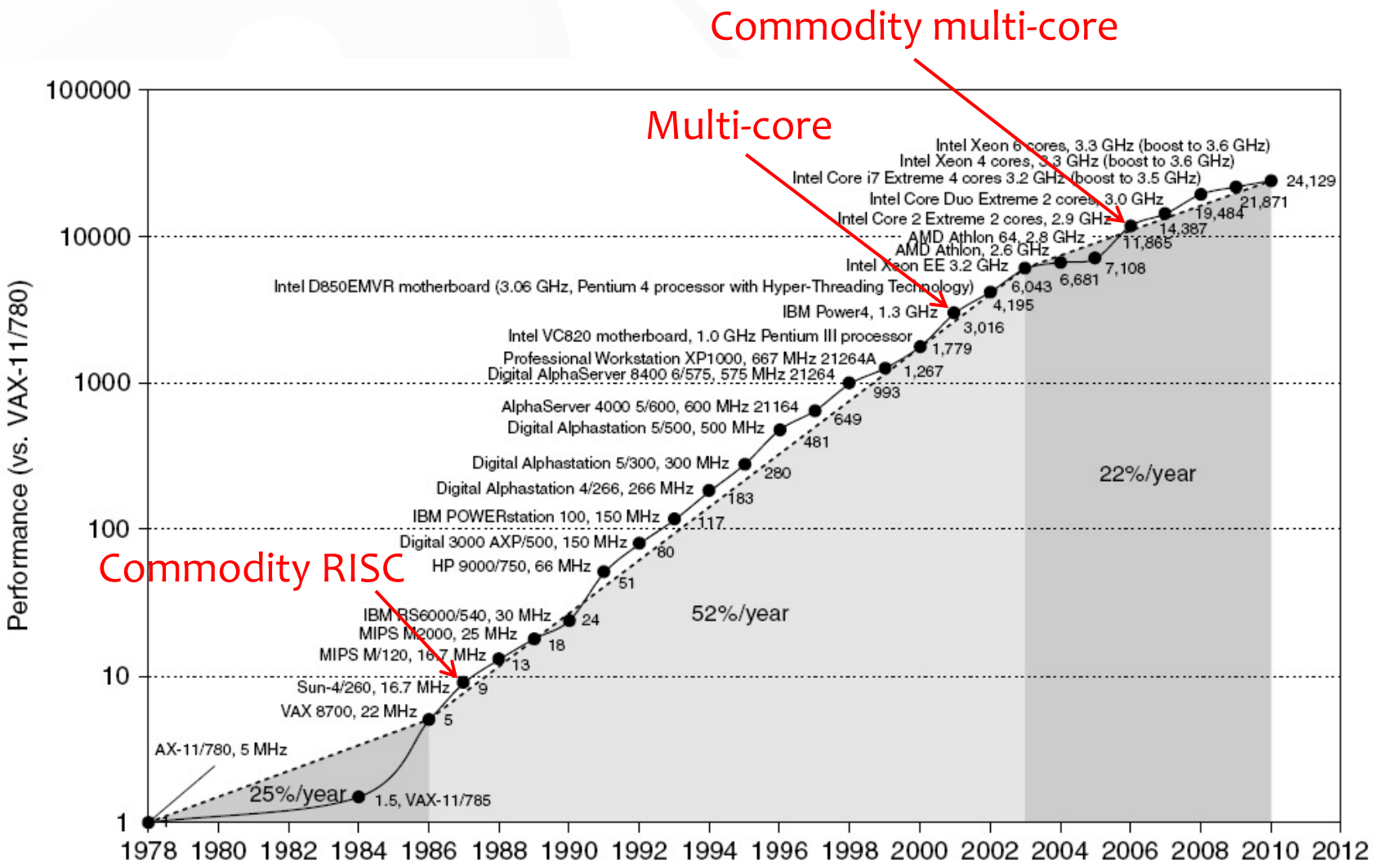
https://www.karlrupp.net/wp-content/uploads/2015/06/40-years-processor-trend.png

Key to Performance: ILP?

- Smaller steps = deeper pipeline = more ILP = higher *f*
  but …
  - Hardware much more complex
  - Hardware runs hotter
  - Deeper pipelines and better ILP identification has gotten harder and more costly → diminishing returns
- Solution? Other forms of parallelism
  - *Data-level parallelism (DLP)*
    - Vector architectures and graphics processing units (GPUs)
  - *Thread-level parallelism (TLP)*
    - Multicore
  - *Request-level parallelism (RLP)*

A change in technology direction that forces a fundamental rethinking of how to continue boosting performance.

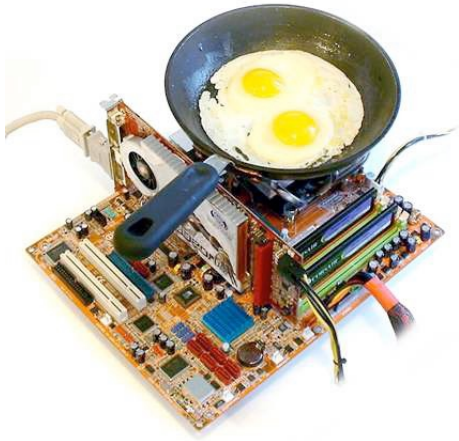
Single-Processor Performance



Current Trends in Architecture

- Cannot continue to rely only on *instruction-level parallelism (ILP)*
  - Single-processor performance improvement ended in 2003
  - The "power wall" is rearing its ugly head
- New models for performance
  - *Data-level parallelism (DLP)*
    - Vector architectures and graphics processing units (GPUs)
  - *Thread-level parallelism (TLP)*
    - Multicore
  - *Request-level parallelism (RLP)*
- Consequence
  - Explicit restructuring of an application

Why Multicore?

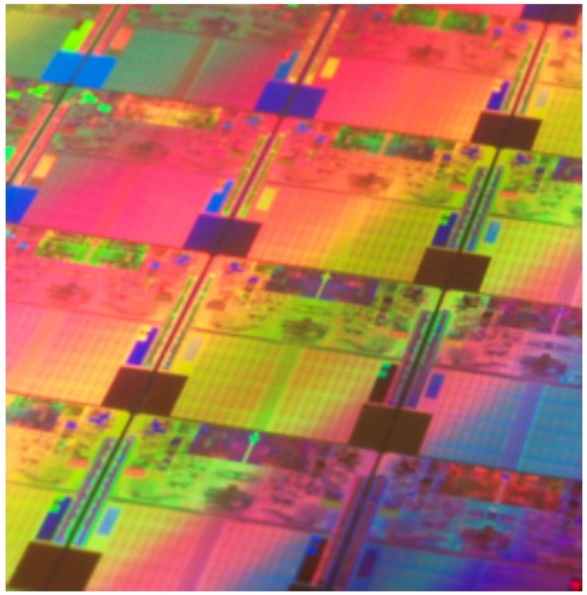


- The MHz race ran out of steam
  - The "free ride" is over
  - Trade-off between raw performance and practical physical implementation and utilization.
    - Power consumption an increasingly "hot" issue … enough to nearly fry an egg.

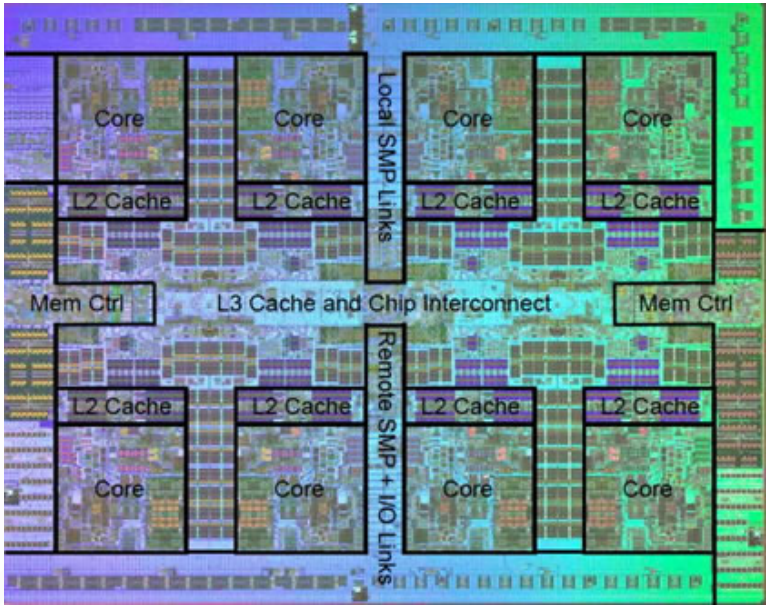
- Result
  - Virtually impossible to buy a non-multicore processor in any computer now.

What is Multicore?

- A multicore chip is a single silicon die containing multiple fully functional sequential processor cores tied together to form a small parallel computer.

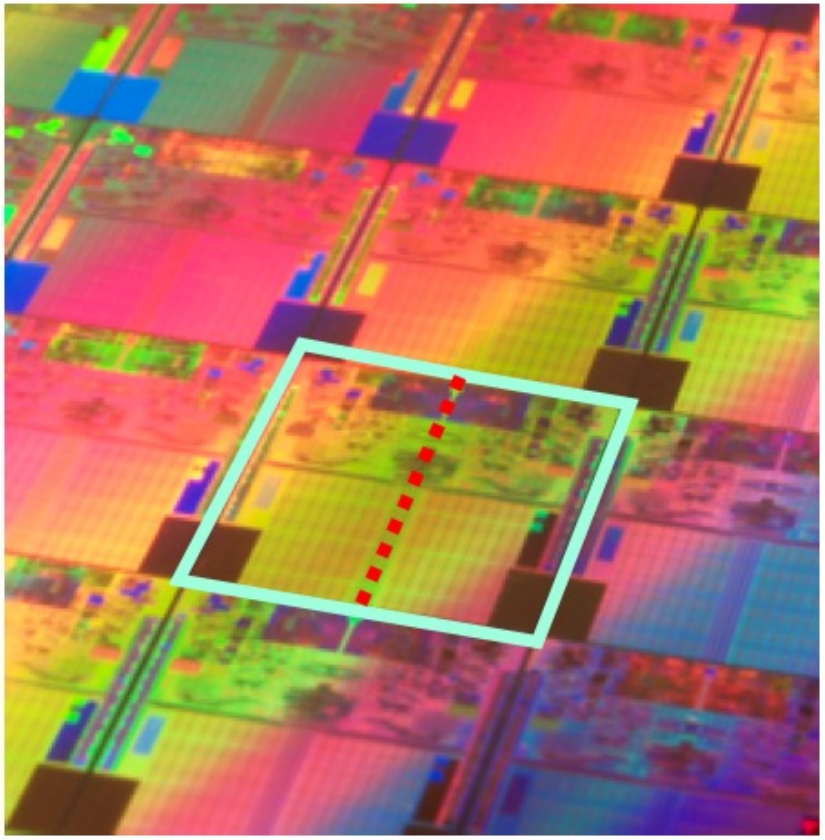


Source: Intel Corporation



Source: IBM

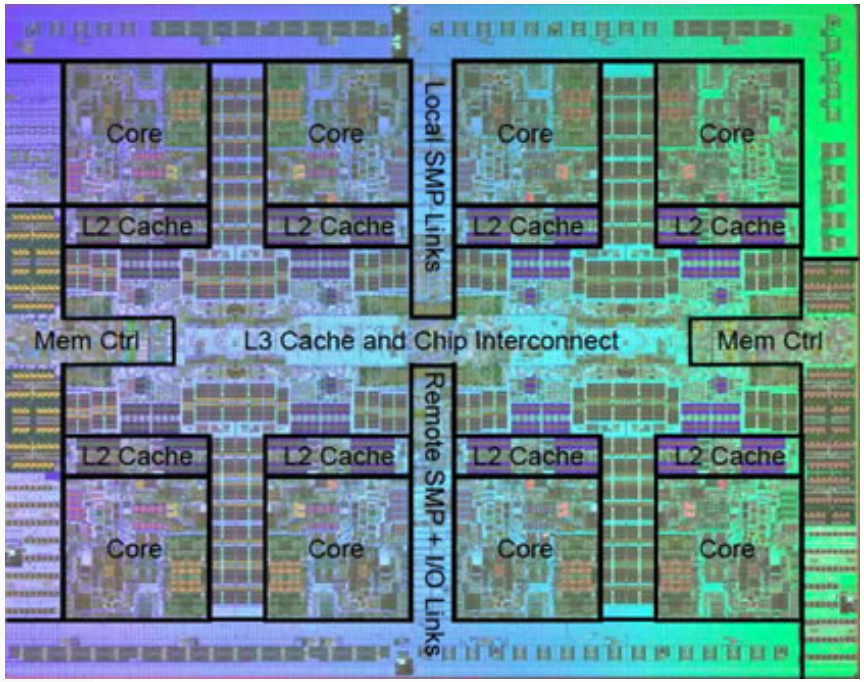
A Closer Look at Multicore



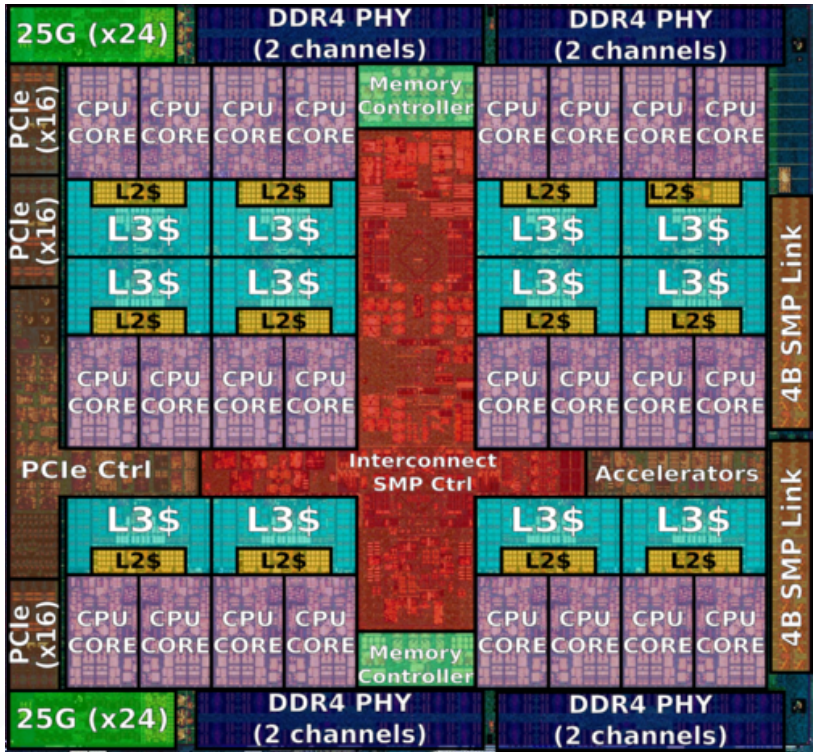
- Multicore Highlighted
  - Intel Core 2 processor on a larger silicon wafer.
- Anatomy
  - Two cores of Intel Core 2 processor separated by a red line.
  - Cache is the very regular portion. Light colored.
  - Functional units are up above. Dark colored.

A Closer Look at Multicore

IBM Power7



IBM Power9



Multicore: Solution to Power & Performance

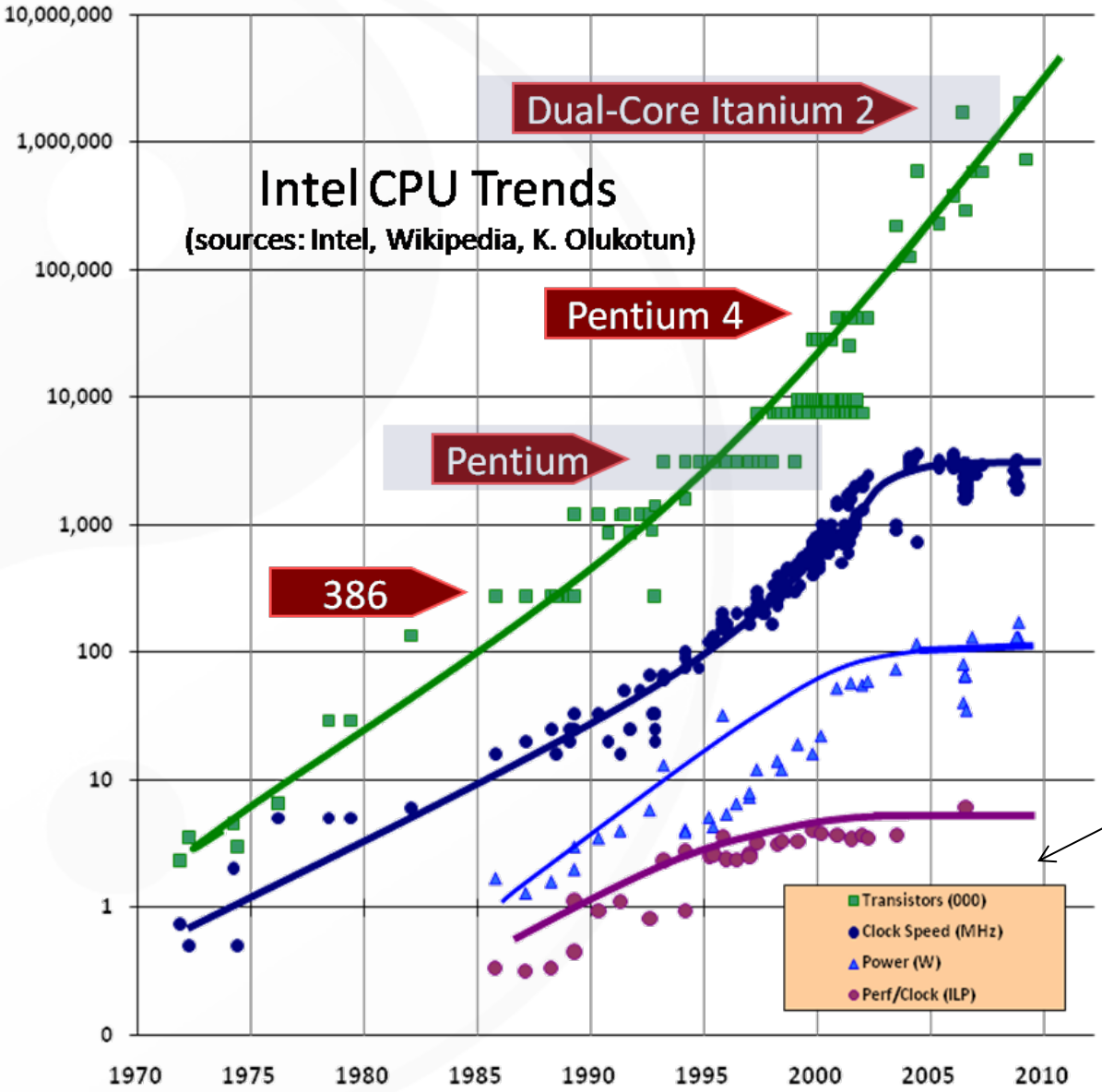
- Multicore Recipe
  - Stop increasing the complexity of single cores.
  - Replicate simpler cores on a die and exploit thread-level parallelism (TLP).
  - Reduce clock rates as pipelines do *not* have to be so deep, thus reducing power requirements.
- Hyperthreading NOT?!
  - Many units in the CPU are shared between the "virtual cores" that hyperthreading provides, limiting performance seen in practice.
  - Hyperthreaded processor (i.e., two-way simultaneous multithreading) still suffers from the complexity problem, leading to higher power and cooling requirements.

Don't Be Misled!

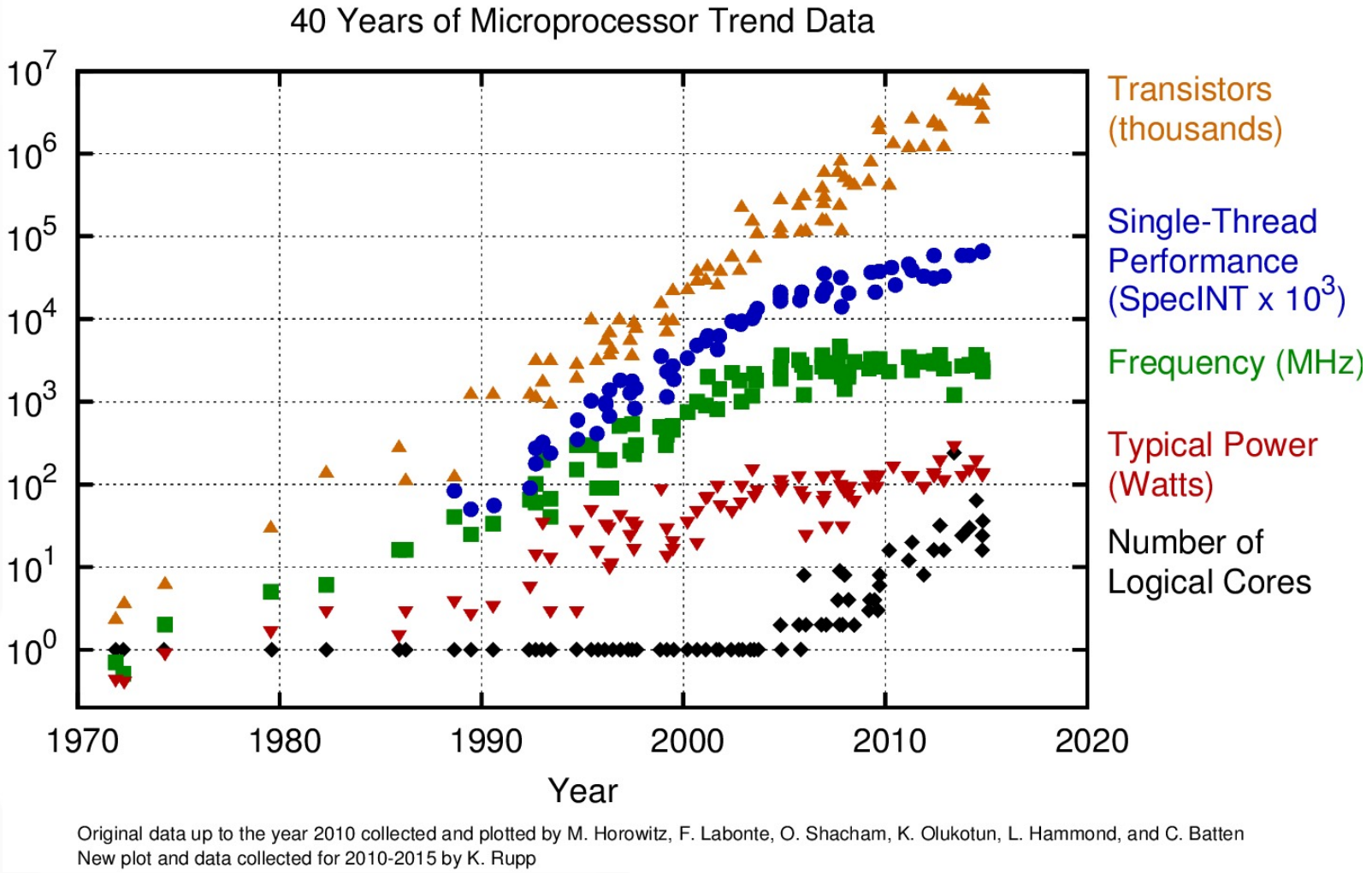
- Pipelining is here to stay.
  - At a certain granularity, blocks of instructions in typical programs exhibit a good degree of ILP.

- New CPUs still improve pipelining and related single-threaded performance improvements.

  … *but it's not the focus anymore.*


Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)
Dual-Core Itanium 2
Pentium 4
Pentium
386
10,000,000
1,000,000
100,000
10,000
1,000
100
10
1
0
1970
1975
1980
1985
1990
1995
2000
2005
2010
Transistors (000)
Clock Speed (MHz)
Power (W)
Perf/Clock (ILP)

ILP



https://www.karlrupp.net/wp-content/uploads/2015/06/40-years-processor-trend.png

Defining Computer Architecture

- "Old" View of Computer Architecture:
  - Instruction set architecture (ISA) design
  - Decisions regarding
    - registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, instruction encoding

- "Real" Computer Architecture
  - Specific requirements of the target machine
  - Design to maximize performance within constraints: cost, power, and availability
    - Includes ISA, microarchitecture, hardware

Power and Energy

- Problem: Get power in, get power out
- Thermal Design Power (TDP)
  - Characterizes *sustained* power consumption
  - Used as target for power supply and cooling system
  - Lower than peak power, higher than average power consumption
- Clock rate can be reduced dynamically to limit power consumption
  - $P \alpha CV^2f$, where C is capacitance being switched per clock cycle, V is supply voltage of processor, and f is switching frequency
- Energy per task is often a better measurement

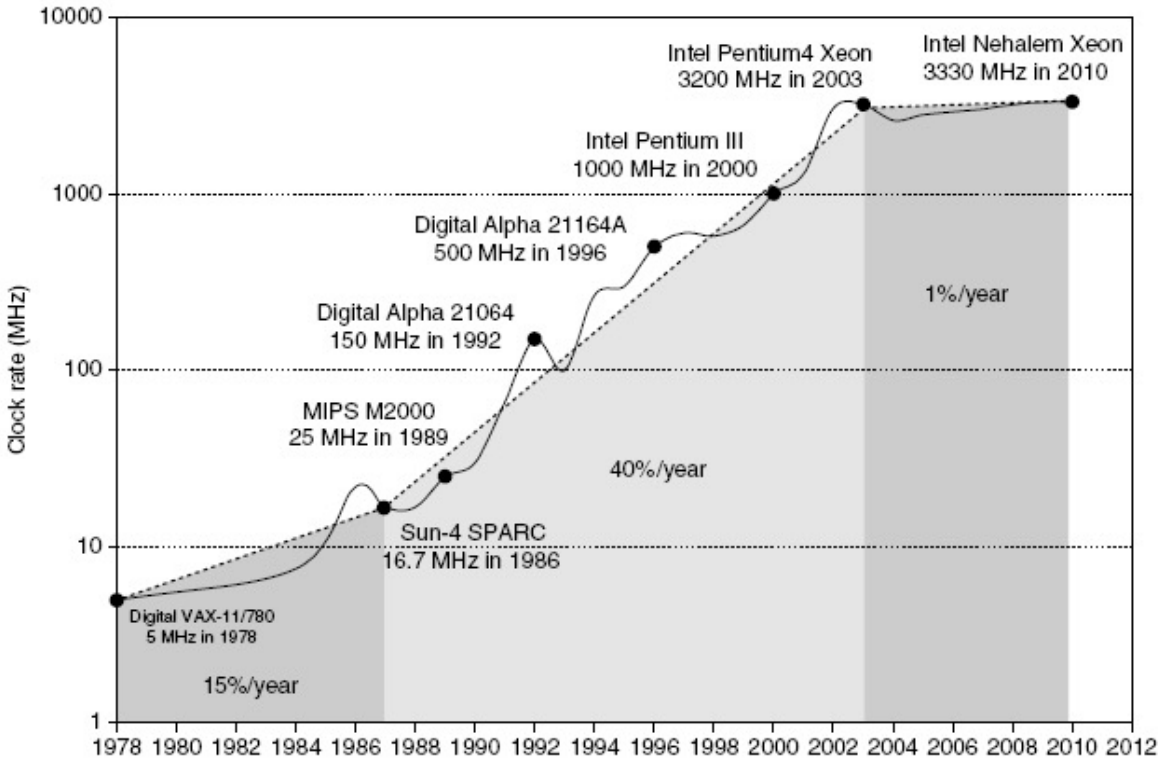
Dynamic Energy and Power

- Dynamic energy
  - Transistor switch from 0 → 1 or 1 → 0
  - ½ x *capacitive load* x *voltage*^{2}
- Dynamic power
  - ½ x *capacitive load* x *voltage*^{2} x *frequency switched*
- Reducing clock rate reduces power, ***not*** energy

Power

- Intel 80386 ~ 2 W
- 3.3-GHz Intel Core i7 ~ 130 W

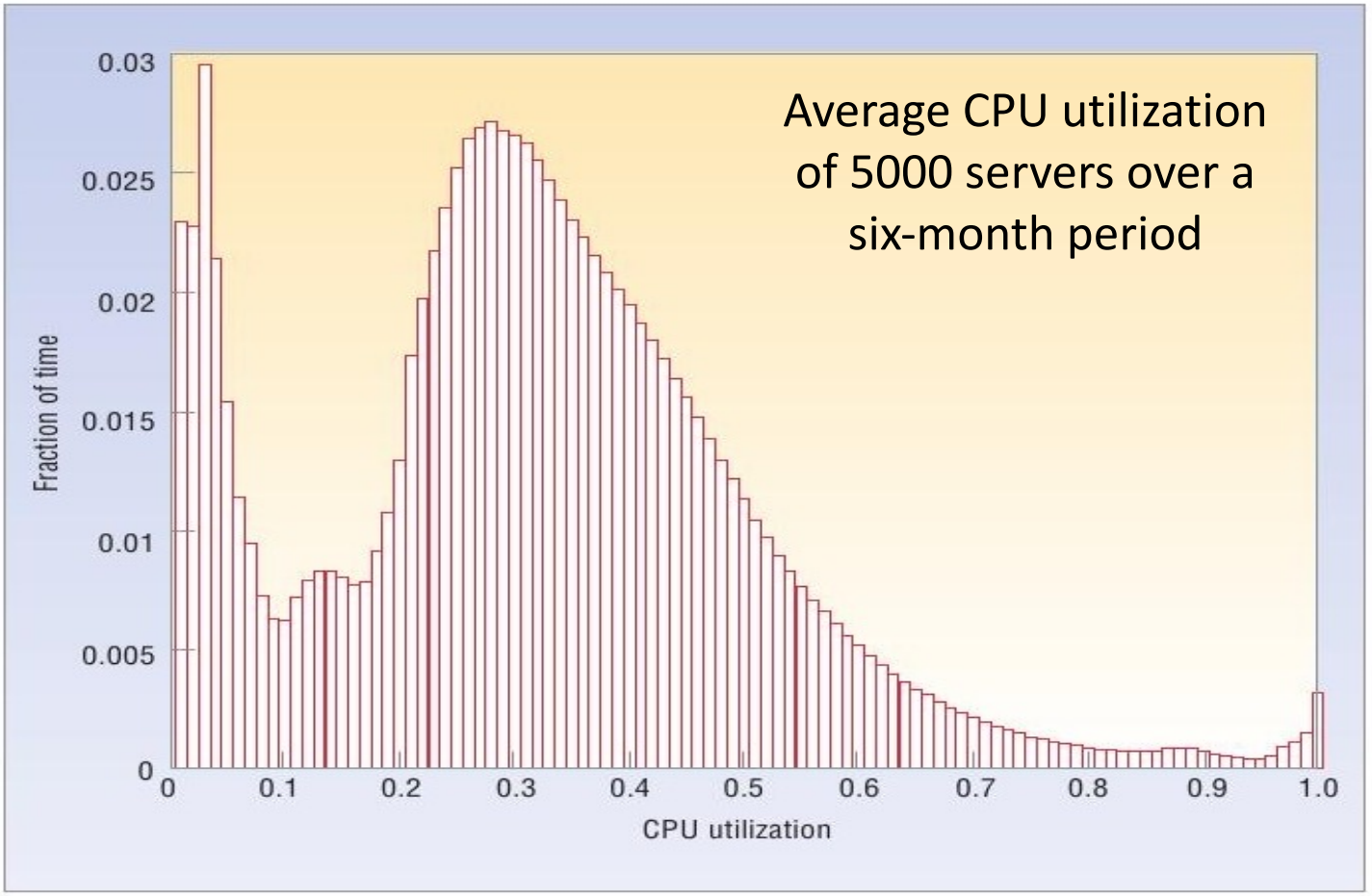
- Heat must be dissipated from 1.5 x 1.5 cm chip
- This is the limit of what can be cooled by air



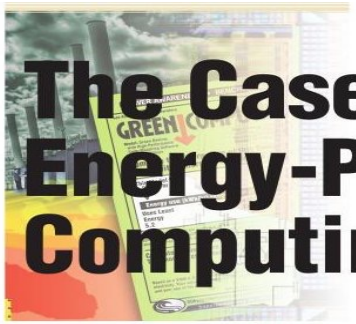
Reducing Power

- Techniques for Reducing Dynamic Power
  - Do nothing well
  - Dynamic voltage-frequency scaling (2005)
    - C. Hsu and W. Feng, "A Power-Aware Run-Time System for High-Performance Computing," *ACM/IEEE SC*, Nov. 2005.
  - Low-power state for DRAM and disks
  - Overclocking and turning off cores
    - Intel *Turbo* mode (2008)
  - Power capping via RAPL (2012-2013)
    - A technique available on Intel processors
    - B. Subramaniam and W. Feng, "Towards Energy-Proportional Computing for Enterprise Server Workloads," In *3rd ACM/SPEC International Conference on Performance Engineering (ICPE)*, **Best Paper Award**, April 2013.

Severe Underutilization in Data Centers



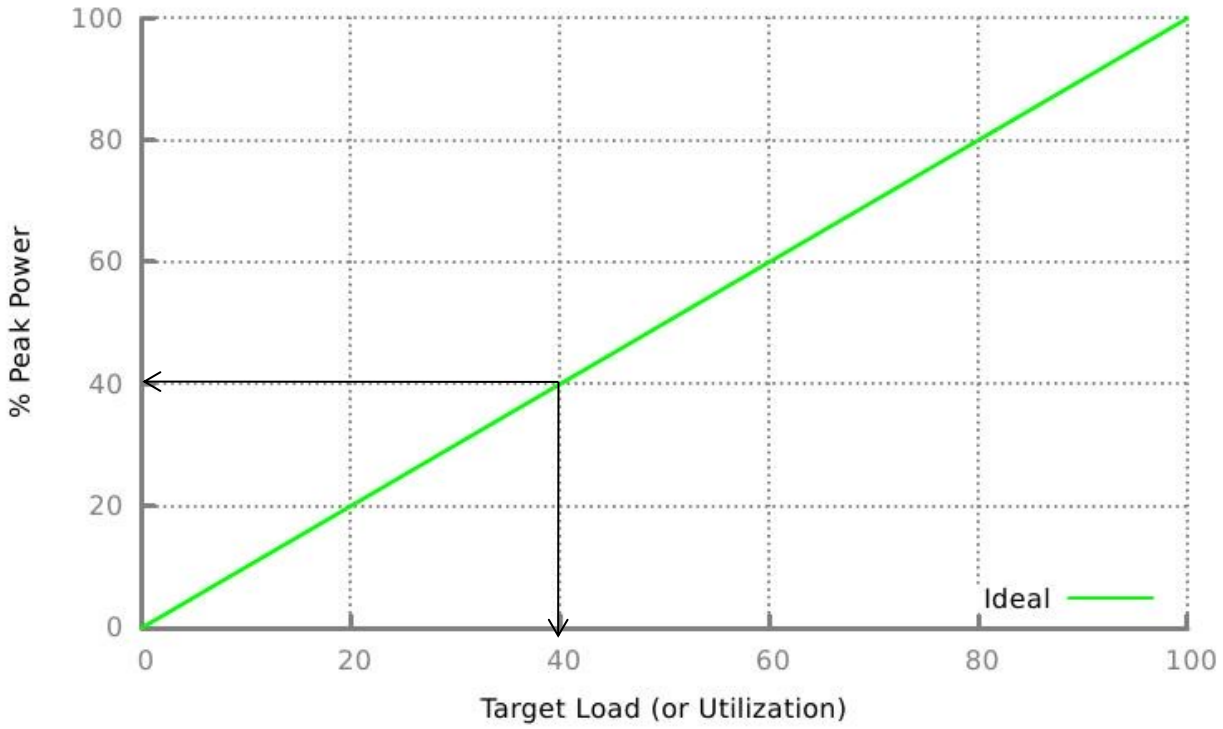
Source: L. A. Barroso et al., The Case for Energy-Proportional Computing, IEEE Computer, 2007



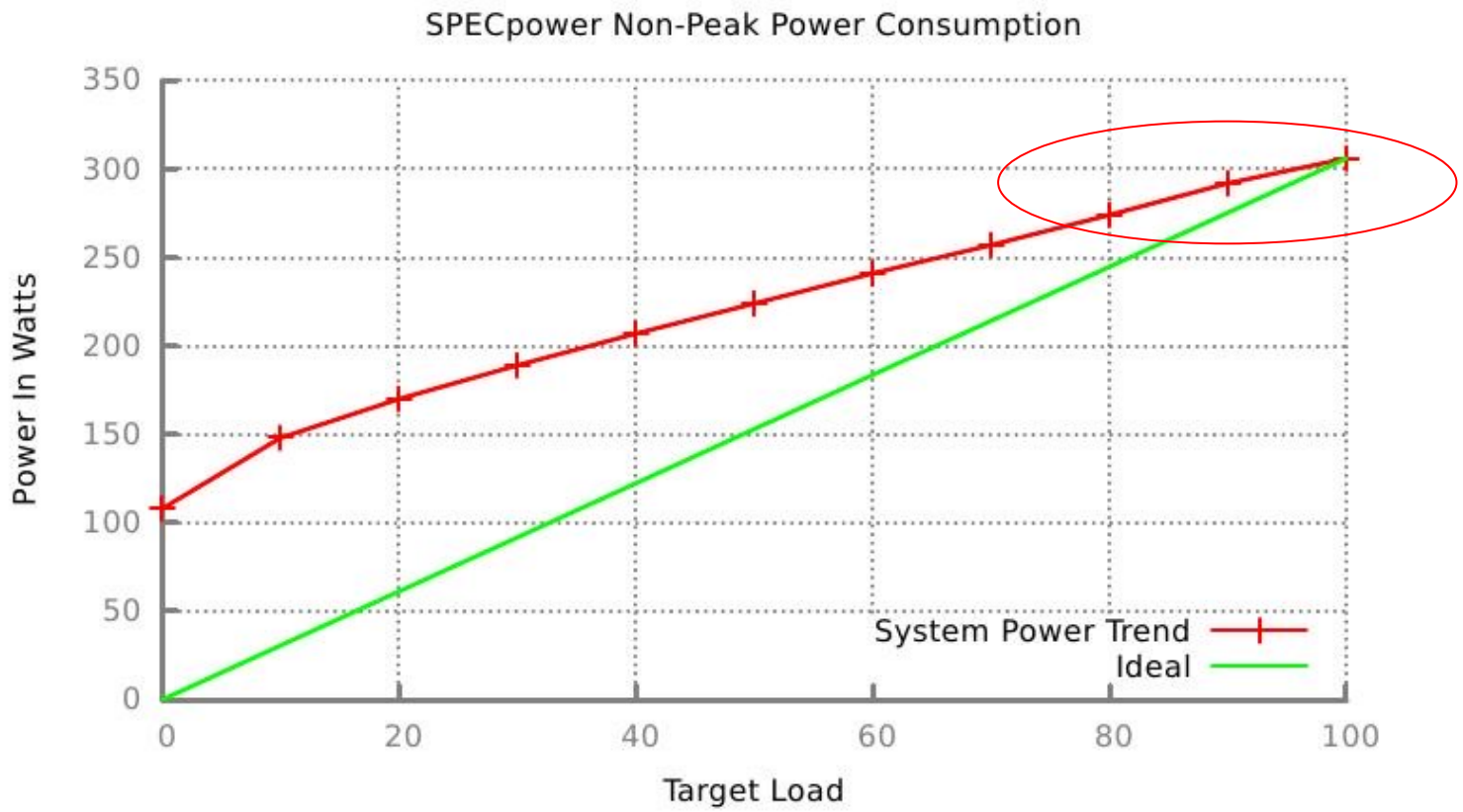
The Case for Energy-Proportional Computing

Luiz André Barroso and Urs Hölzle

- Consume power proportional to utilization (or load level)



Power Efficiency at Different Levels of Utilization



- Near energy-proportional power consumption at high levels of utilization

B. Subramaniam and W. Feng, "Towards Energy-Proportional Computing for Enterprise Server Workloads," In *3rd ACM/SPEC International Conference on Performance Engineering (ICPE)*, April 2013.

Static Power

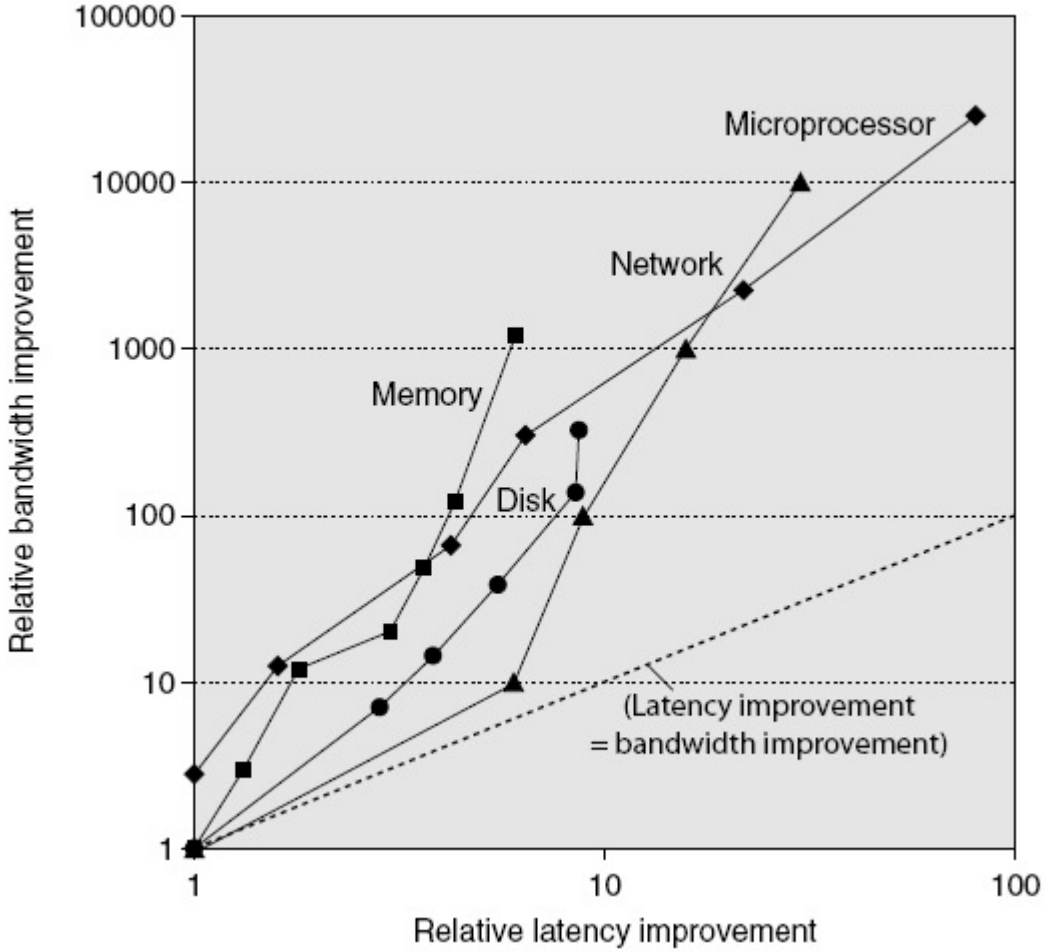
- Static power consumption
  - $Current_{static}$ x Voltage
  - Scales with number of transistors
  - How to reduce? Power gating

Bandwidth and Latency

- Bandwidth or throughput
  - Total work done in a given time
  - 10,000-25,000x improvement for processors
  - 300-1200x improvement for memory and disks

- Latency or response time
  - Time between start and completion of an event
  - 30-80x improvement for processors
  - 6-8x improvement for memory and disks

Bandwidth and Latency



Log-log plot of bandwidth and latency milestones

Transistors and Wires

- Feature Size
  - Minimum size of transistor or wire in *x* or *y* dimension
  - 40 years of shrinking
    - 10 microns in 1971 to 0.032 microns (32 nm) in 2011
  - Further shrinking slowing & getting more complicated
    - No transistor shrink from Intel 2014-2019! @ 14nm.
    - 2019: Intel @ 10nm while AMD & Apple @ 7nm.
    - 2020: IBM @ 7nm for POWER10 → DOE supercomputers: Sierra (LLNL) and Summit (ORNL)
  - Transistor performance scales linearly
    - Wire delay does *not* improve with feature size
  - Integration density scales quadratically

10 µm – 1971
6 µm – 1974
3 µm – 1977
1.5 µm – 1982
1 µm – 1985
800 nm – 1989
600 nm – 1994
350 nm – 1995
250 nm – 1997
180 nm – 1999
130 nm – 2001
90 nm – 2004
65 nm – 2006
45 nm – 2008
32 nm – 2010
22 nm – 2012
14 nm – 2014
10 nm – 2016–2017
7 nm – 2018–2019
5 nm – 2020–2021

Case Study for Transistors & Wires: IBM POWER

Feature	POWER7+	POWER8	POWER9	POWER10
Year	2012	2014	2017	2020
Terminology	32 nm	22 nm	14 nm	7 nm
Die Size	567 mm^2	649 mm^2	693 mm^2	602 mm^2
# Transistors	2.1B	4.2B	8.0B	18.0B
Max. Cores	8	12	12 / 24	15 / 30
Max. SMT Threads/Core	4	8	8 / 4	8 / 4
Max. Threads	32	96	96	120
Max. Frequency	4.0 GHz	4.15 GHz	4.4 GHz	4.0 GHz
L2 Cache	256 kB / core	512 kB / core	512 kB shared	512 kB / core
L3 Cache	80 MB	96 MB	120 MB	120 MB
Memory Support	DDR3	DDR3, DDR4	DDR4	DDR3-DDR5, GDDR6, HBM
I/O Bus	GC++	PCIe Gen3	PCIe Gen4	PCIe Gen5

Trends in Cost

- Cost driven down by learning curve
  - Yield
- DRAM: price closely tracks cost
- Microprocessors: price depends on volume
  - 10% less for each doubling of volume

Integrated Circuit Cost

- Integrated Circuit

$$\text{Cost of integrated circuit} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging and final test}}{\text{Final test yield}}$$

$$\text{Cost of die} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}}$$

$$\text{Dies per wafer} = \frac{\pi \times (\text{Wafer diameter}/2)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2 \times \text{Die area}}}$$

- Bose-Einstein formula:

$$\text{Die yield} = \text{Wafer yield} \times 1/(1 + \text{Defects per unit area} \times \text{Die area})^N$$

- Defects per unit area = 0.016-0.057 defects per cm^2 (2010)
- N = process-complexity factor = 11.5-15.5 (40 nm, 2010)

Dependability

- Module Reliability
  - Mean time to failure (MTTF)
  - Mean time to repair (MTTR)
  - Mean time between failures (MTBF) = MTTF + MTTR
  - Availability = MTTF / MTBF

Recall: Defining Computer Architecture

- "Old" View of Computer Architecture:
  - Instruction set architecture (ISA) design
  - Decisions regarding
    - registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, instruction encoding

- "Real" Computer Architecture
  - Specific requirements of the target machine
  - *Design to maximize performance within constraints: cost, power, and availability*
  - Includes ISA, microarchitecture, hardware

Principles of Computer Design

- Take Advantage of Parallelism
  - e.g., multiple processors, disks, memory banks, pipelining, multiple functional units
- Principle of Locality
  - Reuse of data and instructions
- Focus on the Common Case
  - Amdahl's Law

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times \left((1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Principles of Computer Design

- Equation for Processor Performance

$$\text{CPU time} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

$$\text{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}}$$

$$\text{CPU time} = \text{Instruction count} \times \text{Cycles per instruction} \times \text{Clock cycle time}$$

$$\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}} = \frac{\text{Seconds}}{\text{Program}} = \text{CPU time}$$

Principles of Computer Design

- Different instruction types having different CPIs

$$\text{CPU clock cycles} = \sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i$$

$$\text{CPU time} = \left(\sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock cycle time}$$

Recall: Defining Computer Architecture

- "Old" View of Computer Architecture:
  - Instruction set architecture (ISA) design
  - Decisions regarding
    - registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, instruction encoding

- "Real" Computer Architecture
  - Specific requirements of the target machine
  - *Design to maximize performance within constraints: cost, power, and availability*
  - Includes ISA, microarchitecture, hardware

What about models for cost, power, and availability?

Classes of Computers

- Personal Mobile Device (PMD)
  - Examples: Smartphones & tablet computers
  - Emphasis: Energy efficiency and real-time performance
- Desktop Computing
  - Emphasis: Price-performance ratio
- Servers
  - Emphasis: Availability, scalability, throughput
- Clusters / Warehouse-Scale Computers
  - Used for "Software as a Service (SaaS)"
  - Emphasis: Availability and price-performance
  - Sub-class: Supercomputers
    - Emphasis: Floating-point performance and fast internal networks

Flynn's Taxonomy (1966)

Classification of Computer Architectures

- Single instruction stream, single data stream (SISD)
- Single instruction stream, multiple data streams (SIMD)
  - Vector architectures
  - Multimedia extensions
  - Graphics processor units
- Multiple instruction streams, single data stream (MISD)
  - No commercial implementation
- Multiple instruction streams, multiple data streams (MIMD)
  - Tightly-coupled MIMD
  - Loosely-coupled MIMD

Additional Reading Assignments

- Brian Hayes, "Computing in a Parallel Universe," *American Scientist*, November-December 2007. https://www.americanscientist.org/article/computing-in-a-parallel-universe
- Herb Sutter, "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software," *Dr. Dobb's Journal*, 30(3), March 2005. http://gotw.ca/publications/concurrency-ddj.htm