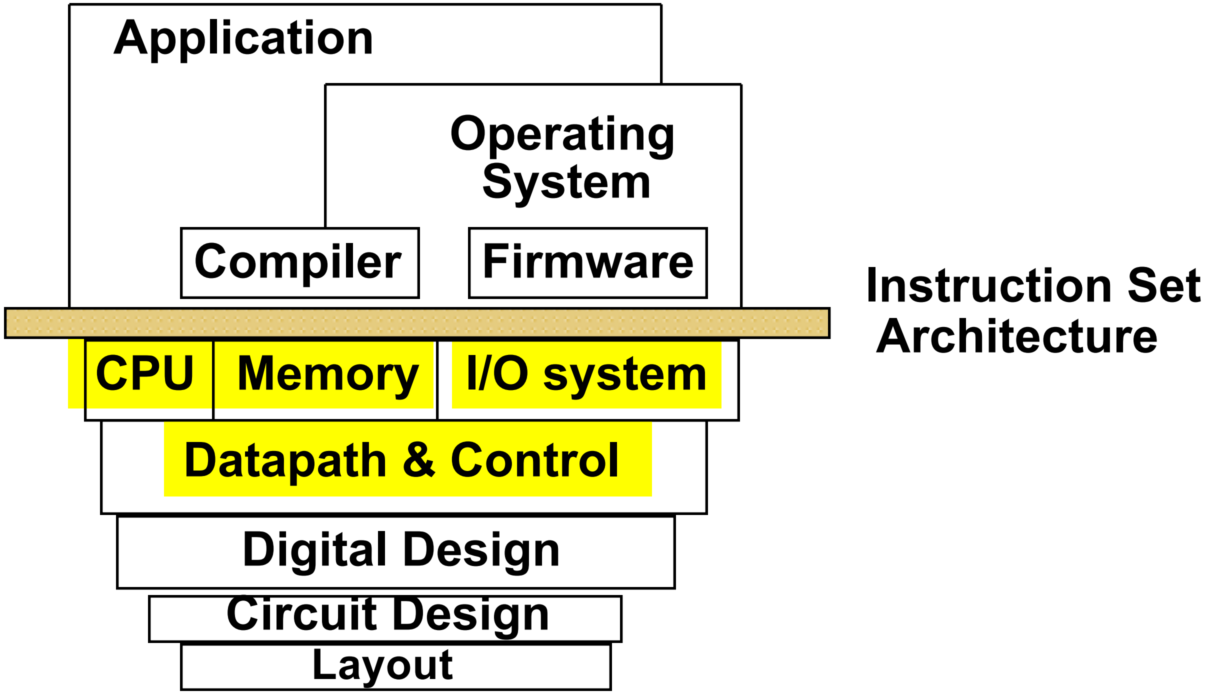
Lecture 2

- **Performance/Power/Cost**

What is "Computer Architecture" ?

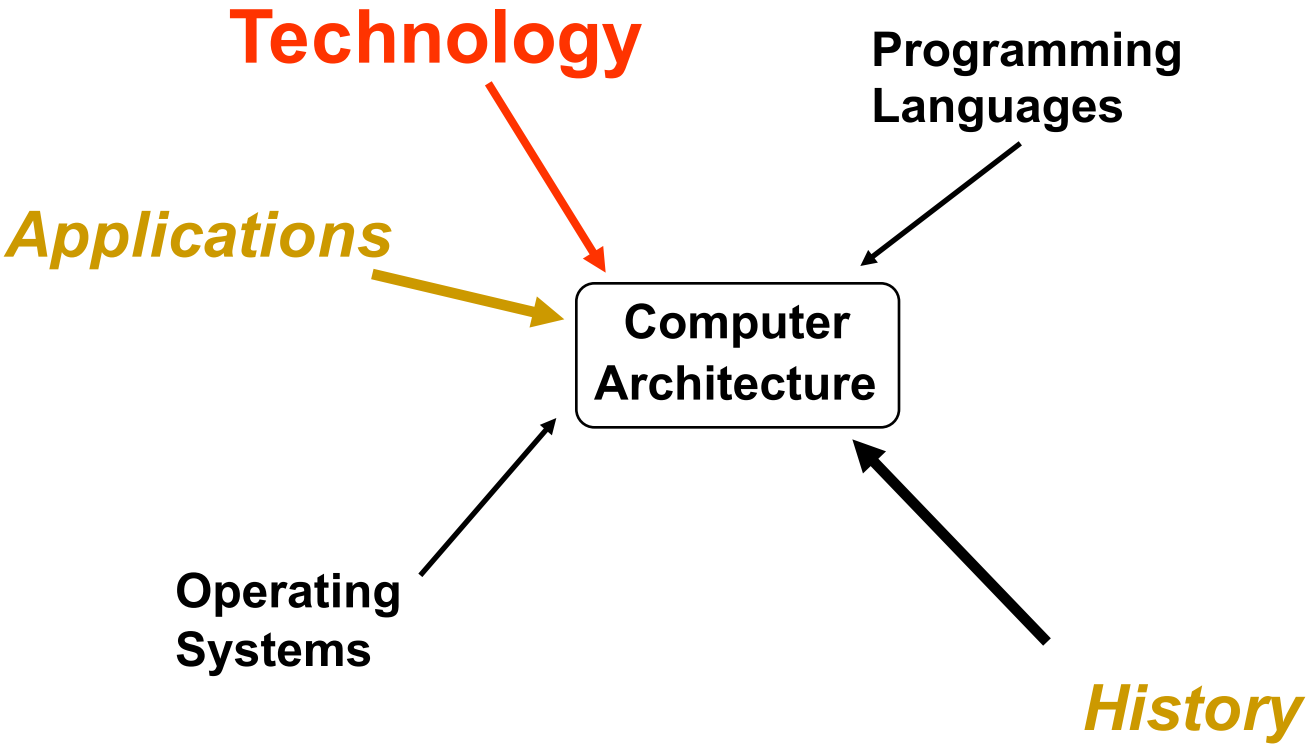


"What really matters is the functioning of the complete system, hardware, runtime system, compiler, operating system, and application"

"In networking, this is called the "End to End argument"

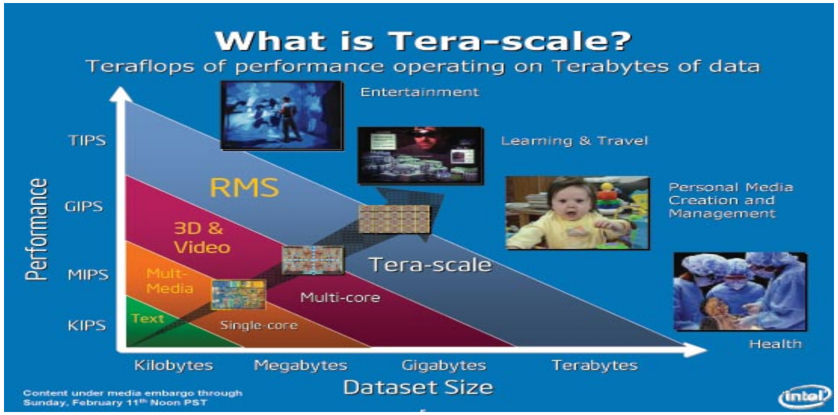
--- H&P

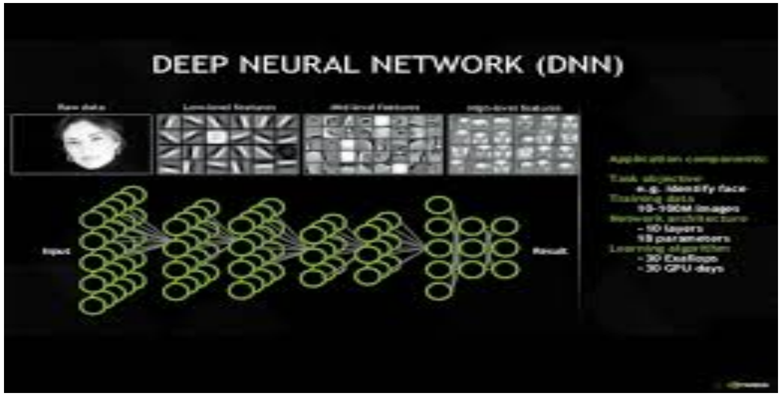
Forces on Computer Architecture



Application

1990'~ Multimedia applications, 3D &Video ==== MMS, SSE, GPU
2000'~ RMS (Recognition, Mining and Synthesis) == Many-core/ GPGPU/ 3D memory
2010' ~ Machine Learning == DNN accelerator architecture

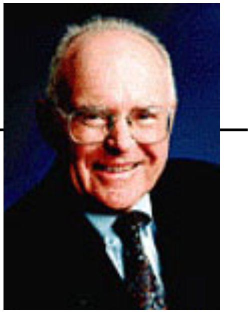




Technologies Used in Computer

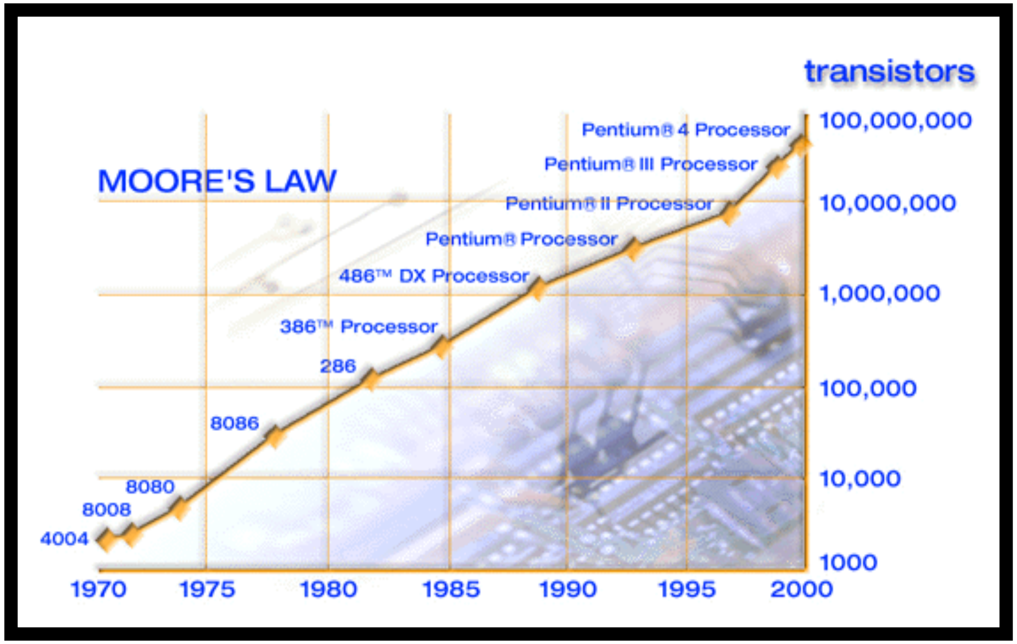
Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2013	Ultra large scale IC	250,000,000,000

Moore's Law



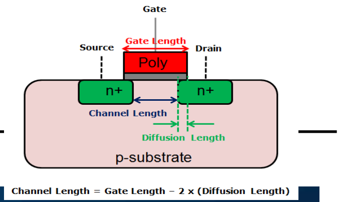
- Moore's Law (1965)
  - Gordon Moore, Intel founder
  - "The density of transistors in an integrated circuit will double every year."

- Reality
  - "The density of silicon chips doubles every 18 months."

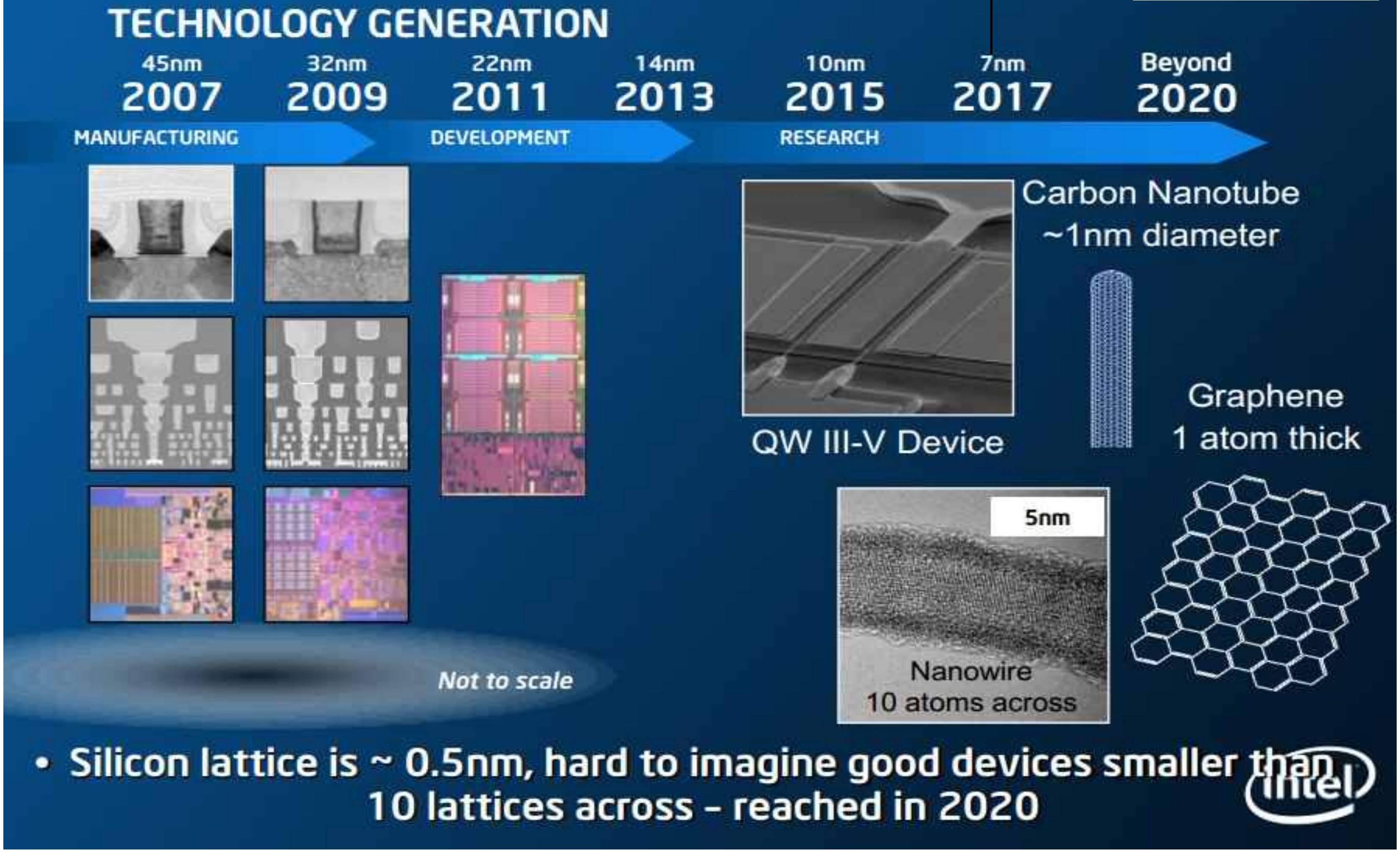


Technology Outlook

Technology Node : transistor gate length



Channel Length = Gate Length – 2 x (Diffusion Length)



TECHNOLOGY GENERATION

45nm	32nm	22nm	14nm	10nm	7nm	Beyond
2007	2009	2011	2013	2015	2017	2020

MANUFACTURING DEVELOPMENT RESEARCH

Carbon Nanotube ~1nm diameter

QW III-V Device

Graphene 1 atom thick

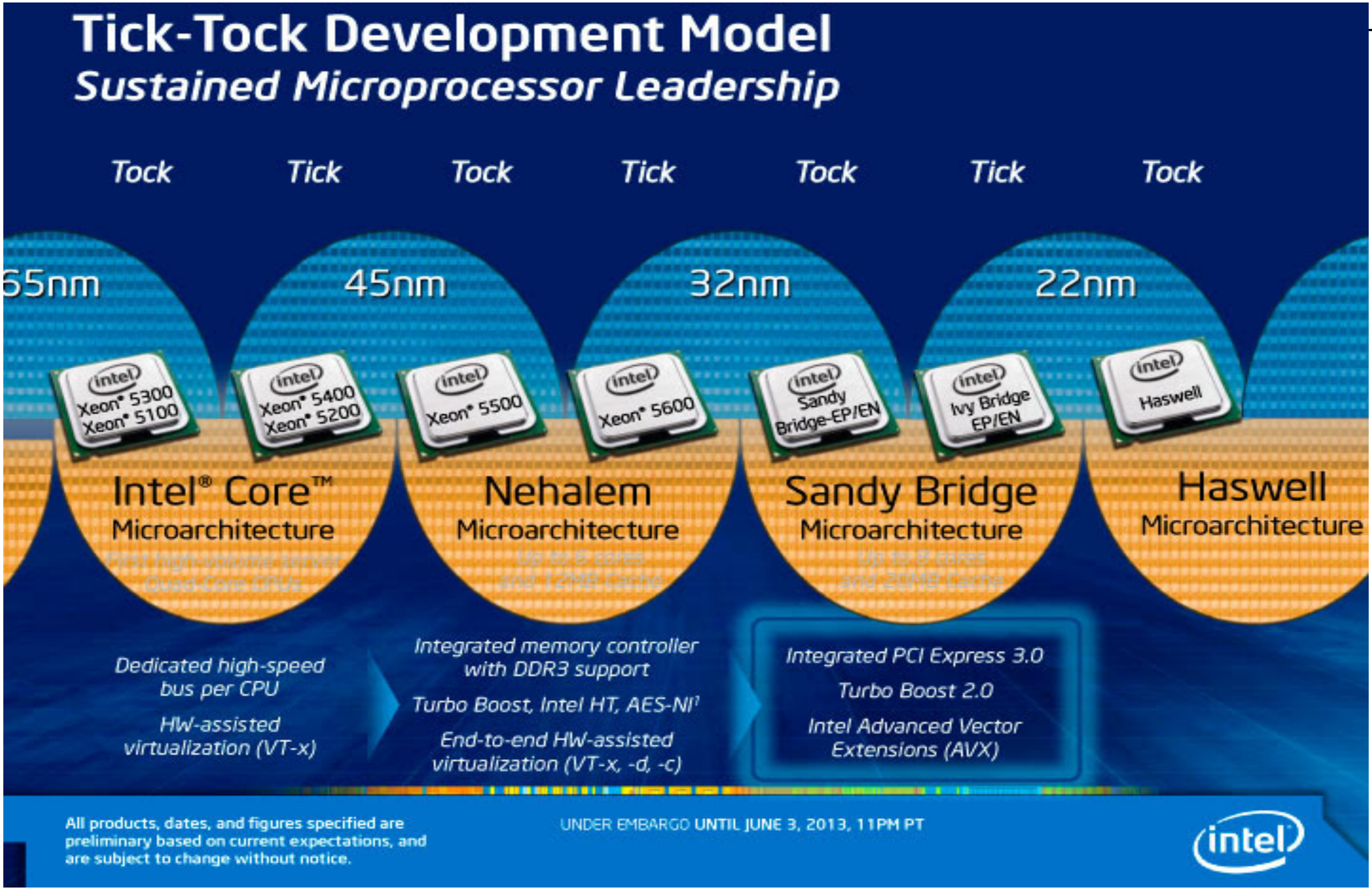
5nm

Nanowire 10 atoms across

Not to scale

- Silicon lattice is ~ 0.5nm, hard to imagine good devices smaller than 10 lattices across – reached in 2020

Monolithic 3D is now on the roadmap for 2019, 08/01/2013
http://www.electroiq.com/articles/sst/2013/08/monolithic-3d-is-now-on-the-roadmap-for-2019.html



Tick-Tock Development Model

Sustained Microprocessor Leadership

Intel® Core™ Microarchitecture
First high-volume server Quad-Core CPUs

Nehalem Microarchitecture
Up to 6 cores and 12MB Cache

Sandy Bridge Microarchitecture
Up to 8 cores and 20MB Cache

Haswell Microarchitecture

Dedicated high-speed bus per CPU

HW-assisted virtualization (VT-x)

Integrated memory controller with DDR3 support

Turbo Boost, Intel HT, AES-NI[1]

End-to-end HW-assisted virtualization (VT-x, -d, -c)

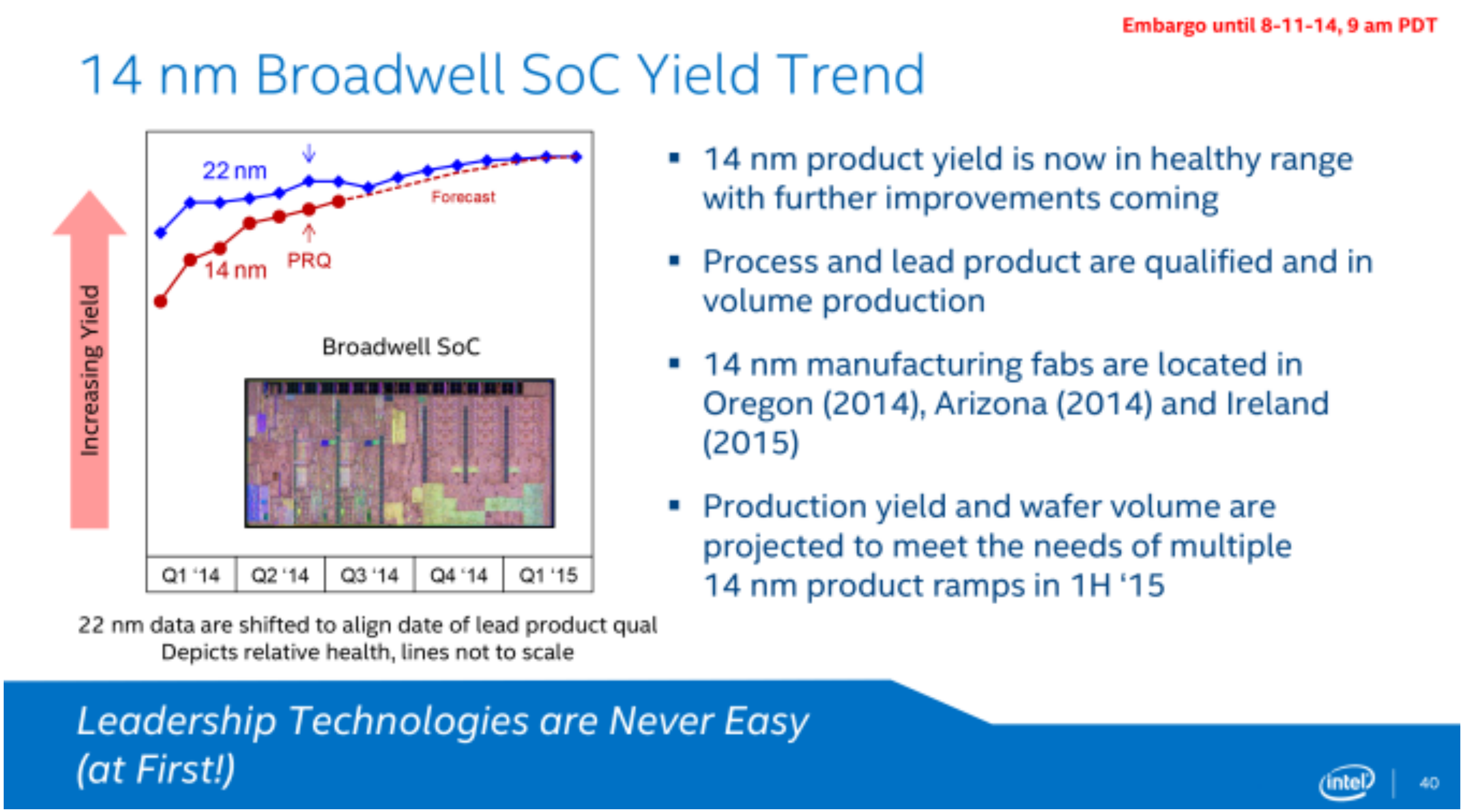
Integrated PCI Express 3.0

Turbo Boost 2.0

Intel Advanced Vector Extensions (AVX)

All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

UNDER EMBARGO UNTIL JUNE 3, 2013, 11PM PT



Embargo until 8-11-14, 9 am PDT

14 nm Broadwell SoC Yield Trend

22 nm data are shifted to align date of lead product qual
Depicts relative health, lines not to scale

- 14 nm product yield is now in healthy range with further improvements coming
- Process and lead product are qualified and in volume production
- 14 nm manufacturing fabs are located in Oregon (2014), Arizona (2014) and Ireland (2015)
- Production yield and wafer volume are projected to meet the needs of multiple 14 nm product ramps in 1H '15

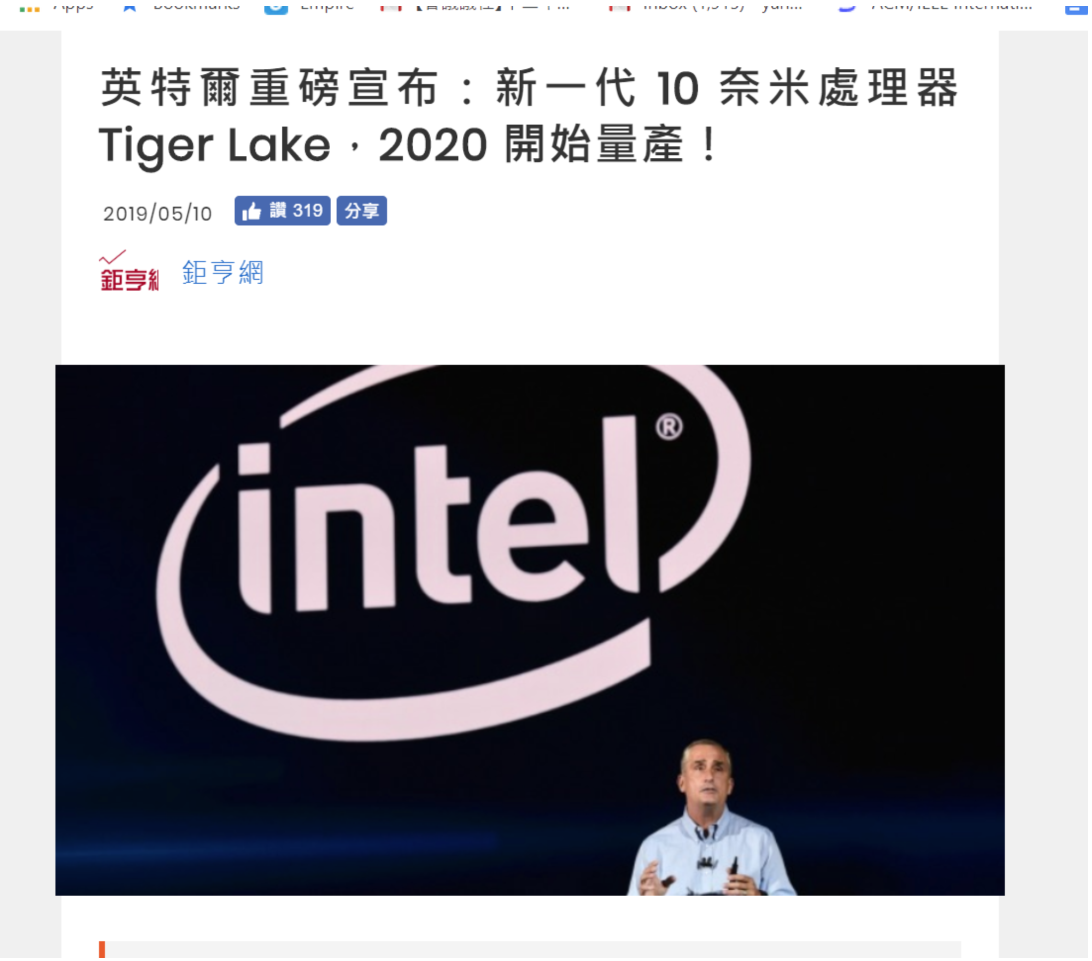
Leadership Technologies are Never Easy (at First!)

Process-Architecture-Optimization (PAO)

- Process: With each process, Intel advances their manufacturing process technology in line with Moore's Law.
- Architecture: With each architecture, Intel uses the their latest manufacturing process technology from their "process" to manufacture a newly designed microarchitecture.
- Optimization: With each optimization, Intel improves upon their previous microarchitecture by introducing incremental improvements and enhancements without introducing any large charges.

Intel PAO Schedule

Cycle	Process	Introduction	Microarchitecture
Process	14 nm	2014	Broadwell
Architecture	14 nm	2015	Skylake
Optimization	14 nm	2016	Kaby Lake
Optimization	14 nm	2017	Coffee Lake
Process	10 nm	2017	Cannonlake
Architecture	10 nm	2018	Icelake
Optimization	10 nm	2019	Tigerlake

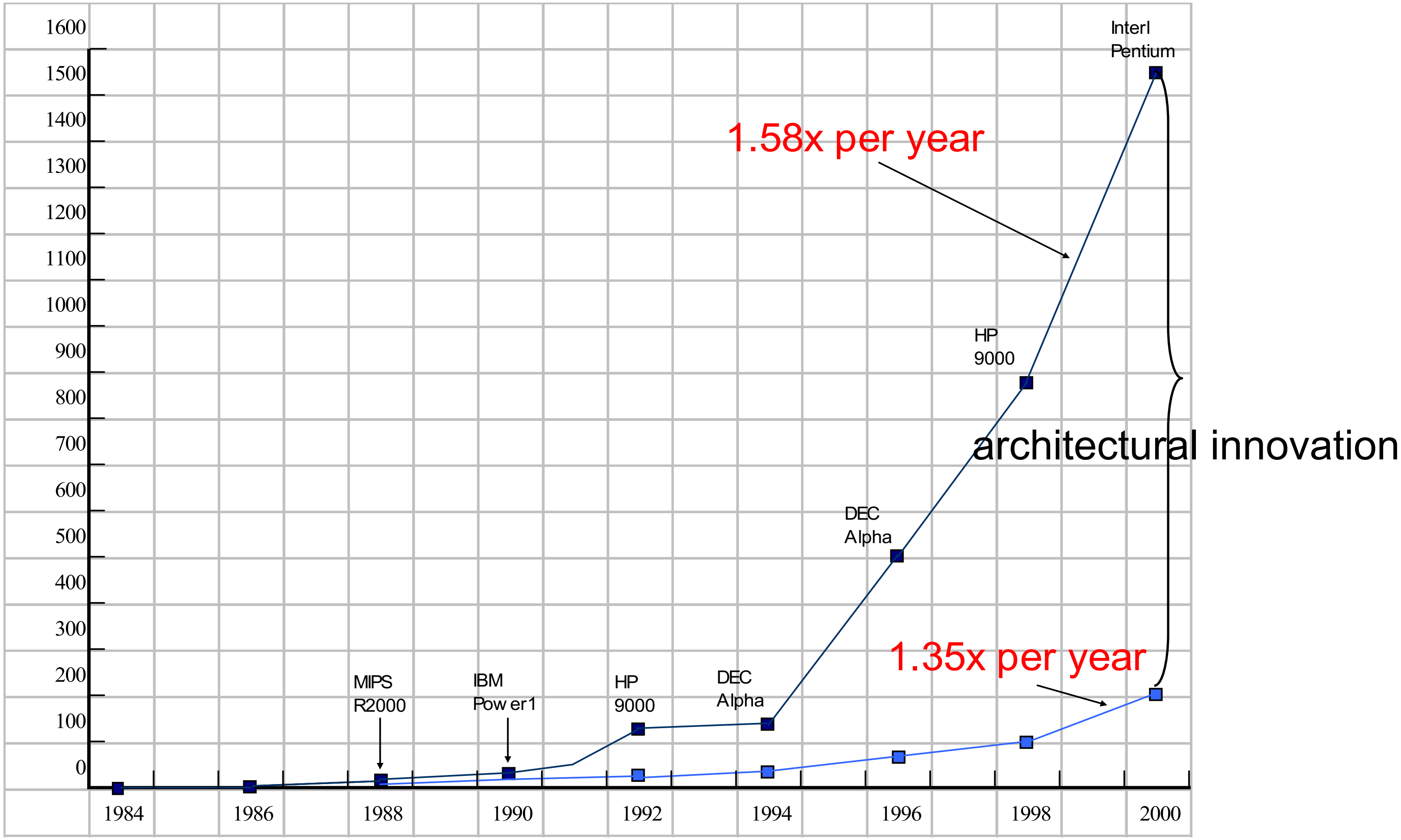


英特爾重磅宣布：新一代 10 奈米處理器 Tiger Lake，2020 開始量產！

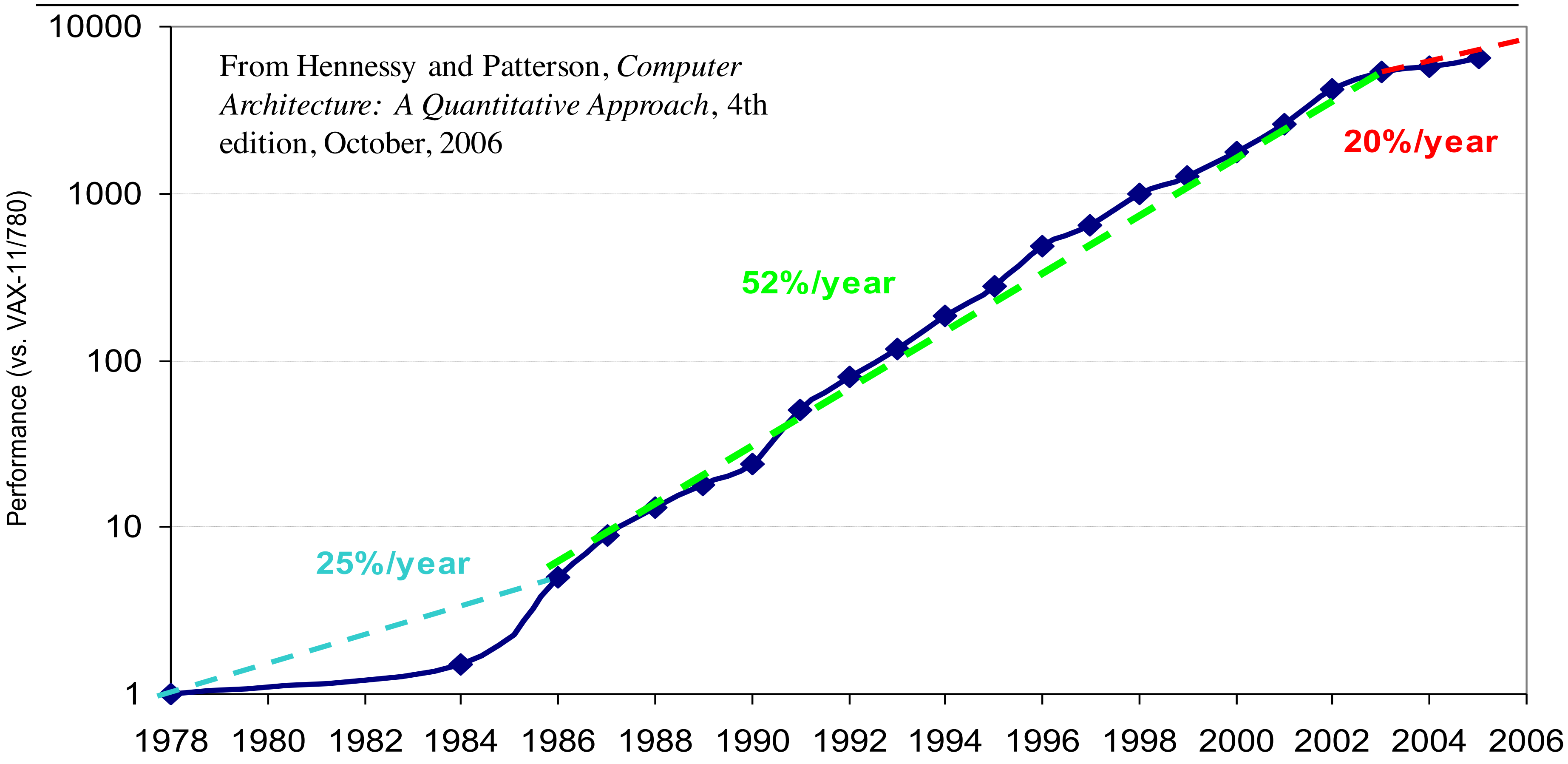
2019/05/10

鉅亨網

Processor Performance

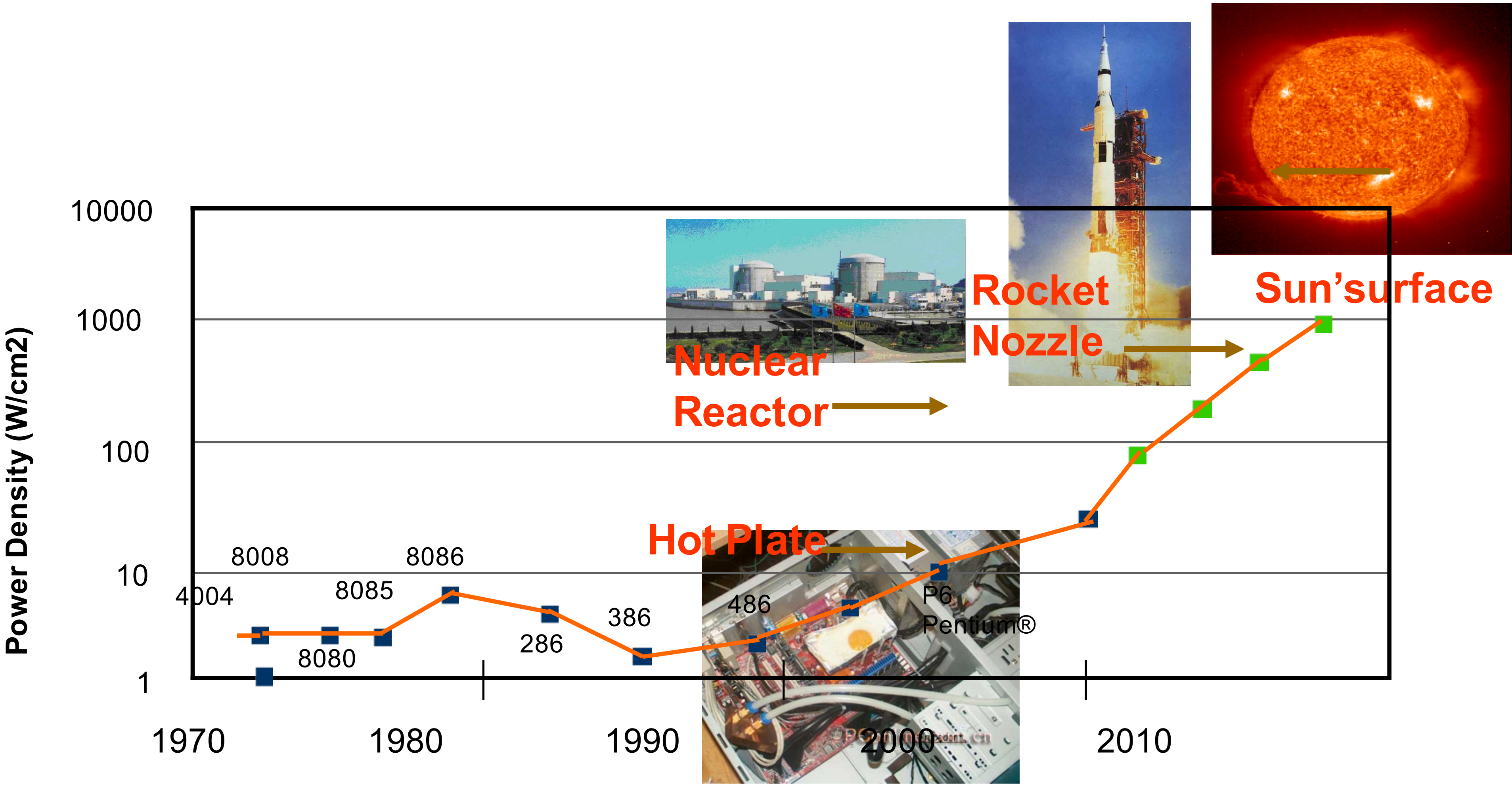


Crossroads: Uniprocessor Performance

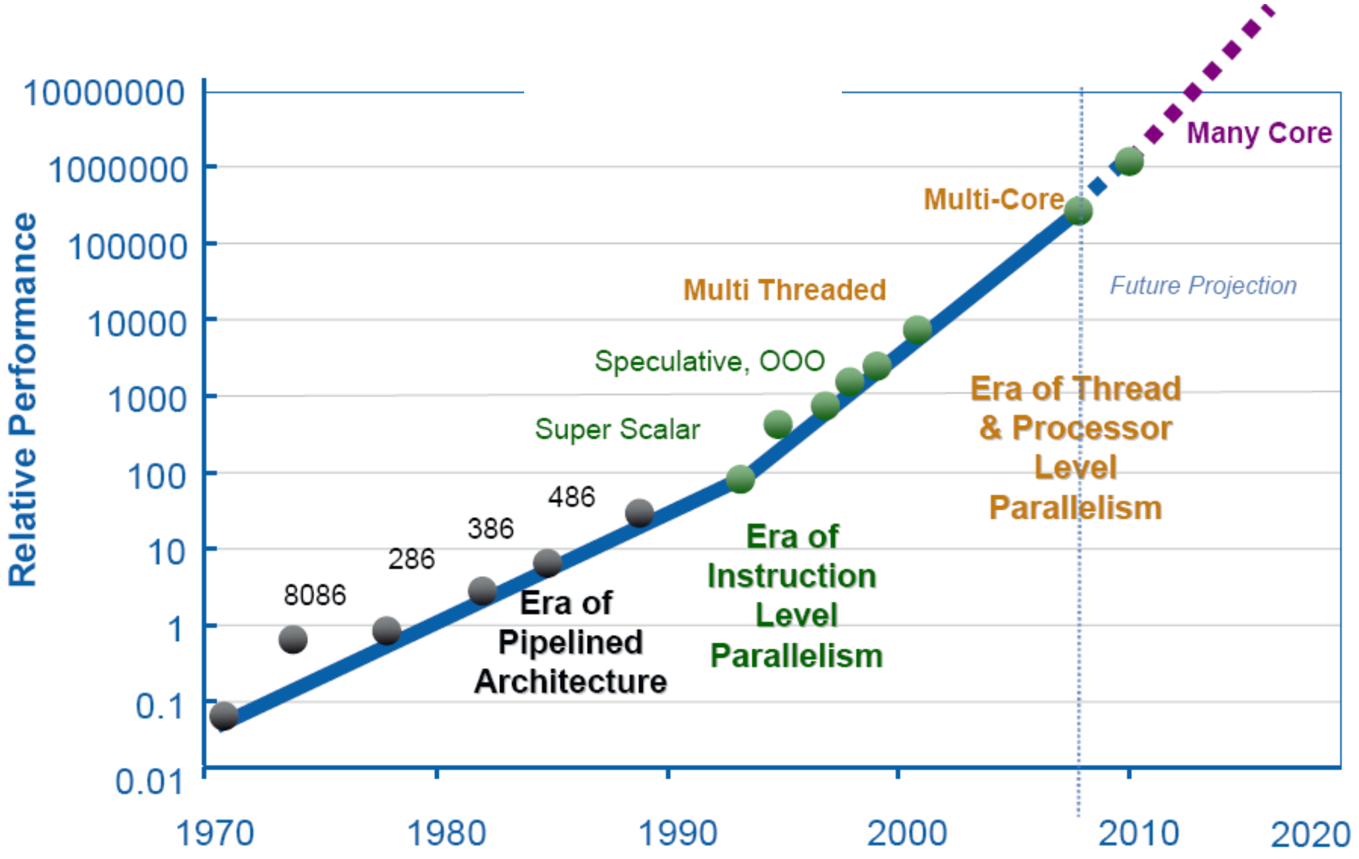


- **VAX : 25%/year 1978 to 1986**
- **RISC + x86: 52%/year 1986 to 2002**
- **RISC + x86: 20%/year 2002 to present**

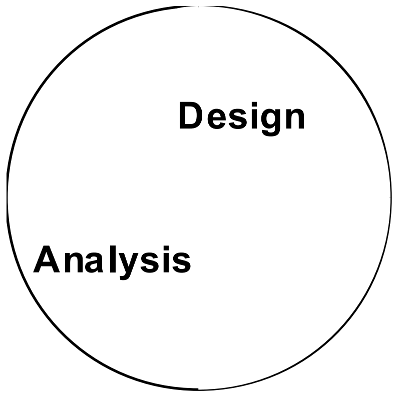
Power Dissipation Increases ….



Parallelism for Energy Efficiency Present and Future



Computer Engineering Methodology

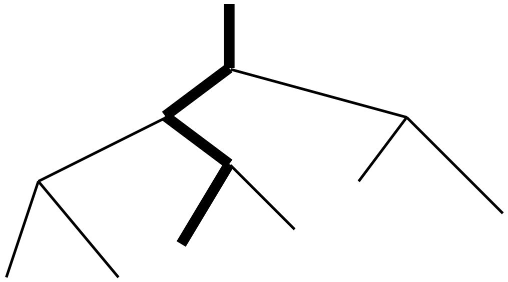


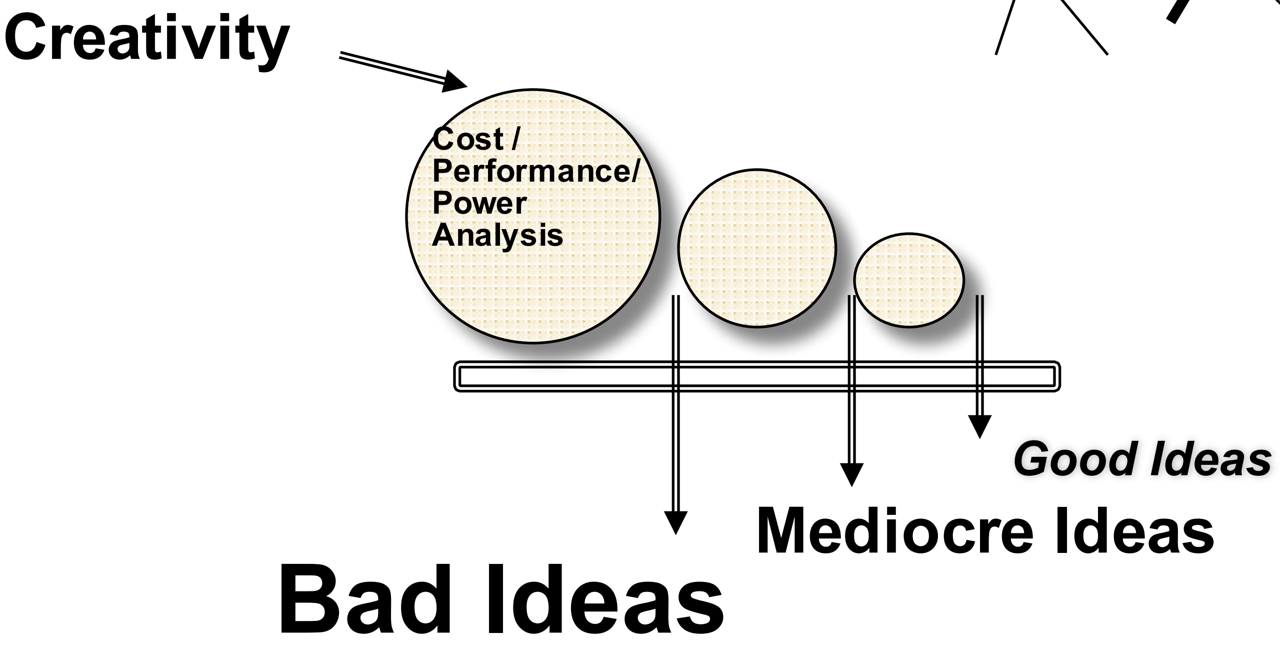
Design

Analysis

Architecture is an iterative process:

- **Searching the space of possible designs**
- **At all levels of computer systems**





Creativity

Cost / Performance/ Power Analysis

Good Ideas

Mediocre Ideas

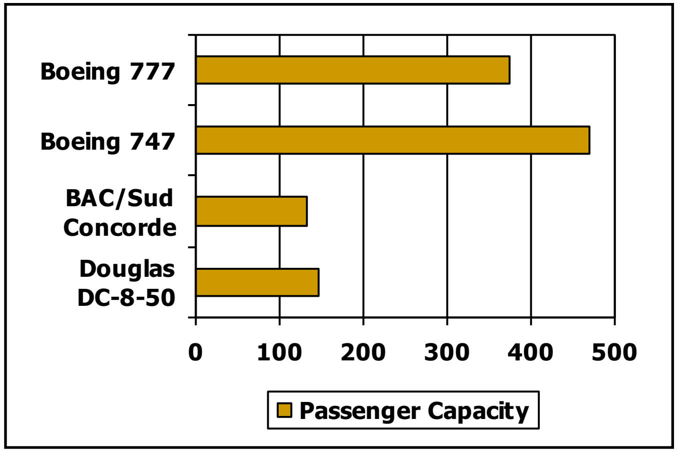
Bad Ideas

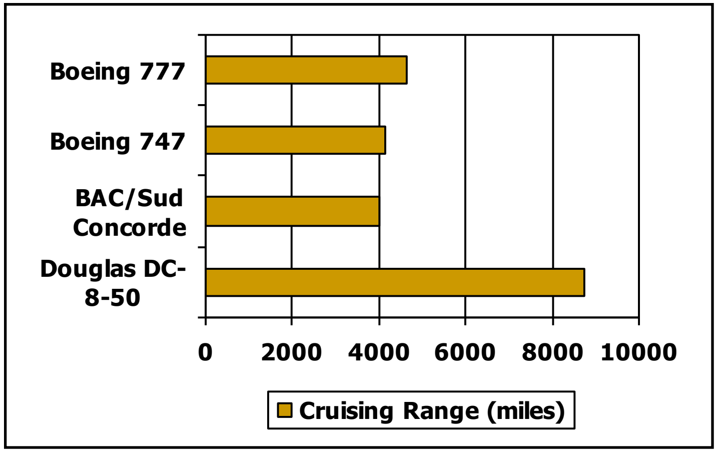
Importance of Evaluation/Analysis

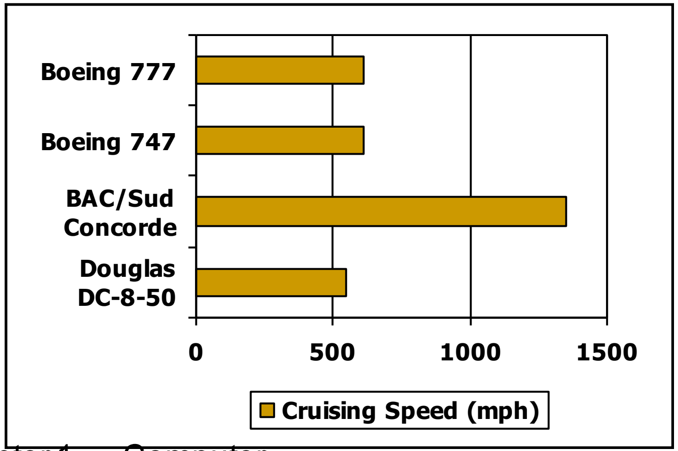
- Why do we care about performance/power/cost evaluation?
  - Purchasing perspective
    - given a collection of machines, which has the
      - best performance ? lowest power
      - least cost ?
      - best performance / cost ?
  - Design perspective
    - faced with design options, which has the
      - best performance improvement ? best energy-efficiency?
      - least cost ?
      - best performance / cost ?
- How to measure, report, and summarize performance/power/cost?
  - Metric
  - Benchmark

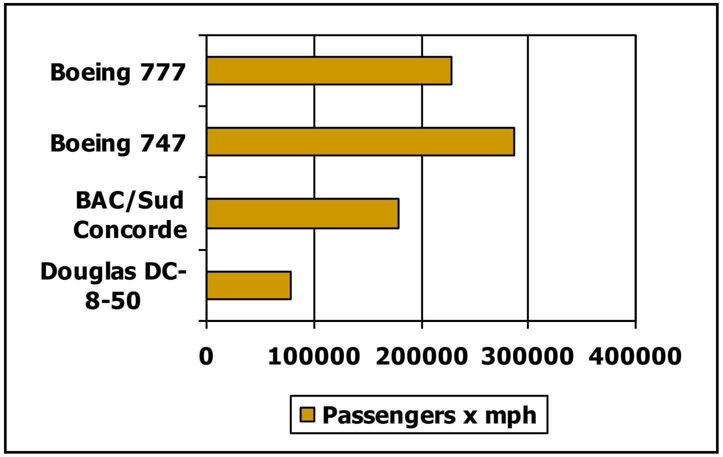
Defining Performance

- Which airplane has the best performance?









Response Time and Throughput

- Response time
  - How long it takes to do a task
- Throughput
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour
- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?
- We'll focus on response time for now…

Relative Performance

- Define Performance = 1/Execution Time
- "X is n time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y$$
$$= \text{Execution time}_Y / \text{Execution time}_X = n$$

- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time$_B$ / Execution Time$_A$ = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

Measuring Execution Time

- Elapsed time
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determine system performance
- CPU time
  - Time spent processing a given job
    - Discounts I/O time, other jobs' shares
  - Comprises user CPU time and system CPU time
  - Different programs are affected differently by CPU and system performance

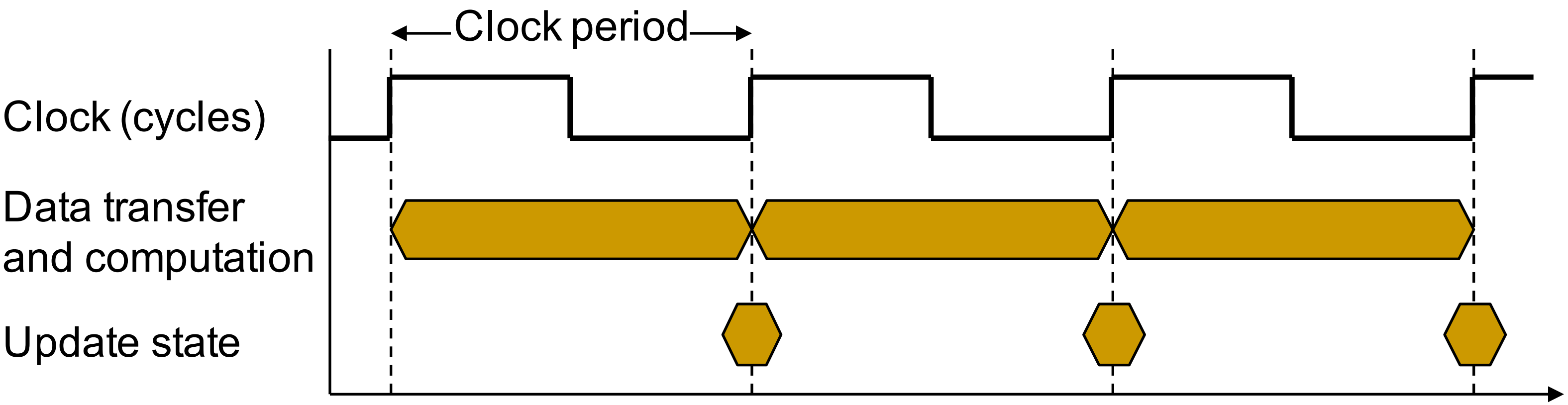
CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \cdot \text{Clock Cycle Time}$$
$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock



- Clock period: duration of a clock cycle
  - e.g., 250ps = 0.25ns = 250×10^{-12}s
- Clock frequency (rate): cycles per second
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \cdot \text{Clock Cycles}_A}{6\text{s}}$$

$$\text{Clock Cycles}_A = \text{CPU Time}_A \cdot \text{Clock Rate}_A$$

$$= 10\text{s} \cdot 2\text{GHz} = 20 \cdot 10^9$$

$$\text{Clock Rate}_B = \frac{1.2 \cdot 20 \cdot 10^9}{6\text{s}} = \frac{24 \cdot 10^9}{6\text{s}} = 4\text{GHz}$$

Instruction Count and CPI

$$\text{CPU Time} = \text{CPU Clock Cycles} \cdot \text{Clock Cycle Time}$$

$$\text{Clock Cycles} = \text{Instruction Count} \cdot \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \cdot \text{CPI} \cdot \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \cdot \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction
  - Affected by both
    - Hardware - CPI per instruction type)
    - software (instruction mix)

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \cdot \text{CPI}_A \cdot \text{Cycle Time}_A$$

$$= I \cdot 2.0 \cdot 250\text{ps} = I \cdot 500\text{ps}$$

A is faster…

$$\text{CPU Time}_B = \text{Instruction Count} \cdot \text{CPI}_B \cdot \text{Cycle Time}_B$$

$$= I \cdot 1.2 \cdot 500\text{ps} = I \cdot 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \cdot 600\text{ps}}{I \cdot 500\text{ps}} = 1.2$$

…by this much

CPI in More Detail

- If different instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \cdot \text{Instruction Count}_i)$$

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \text{CPI}_i \cdot \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\text{Relative frequency}}$$

CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5
  - Clock Cycles
    = 2×1 + 1×2 + 2×3
    = 10
  - Avg. CPI = 10/5 = 2.0

- Sequence 2: IC = 6
  - Clock Cycles
    = 4×1 + 1×2 + 1×3
    = 9
  - Avg. CPI = 9/6 = 1.5

Aspects of CPU Performance

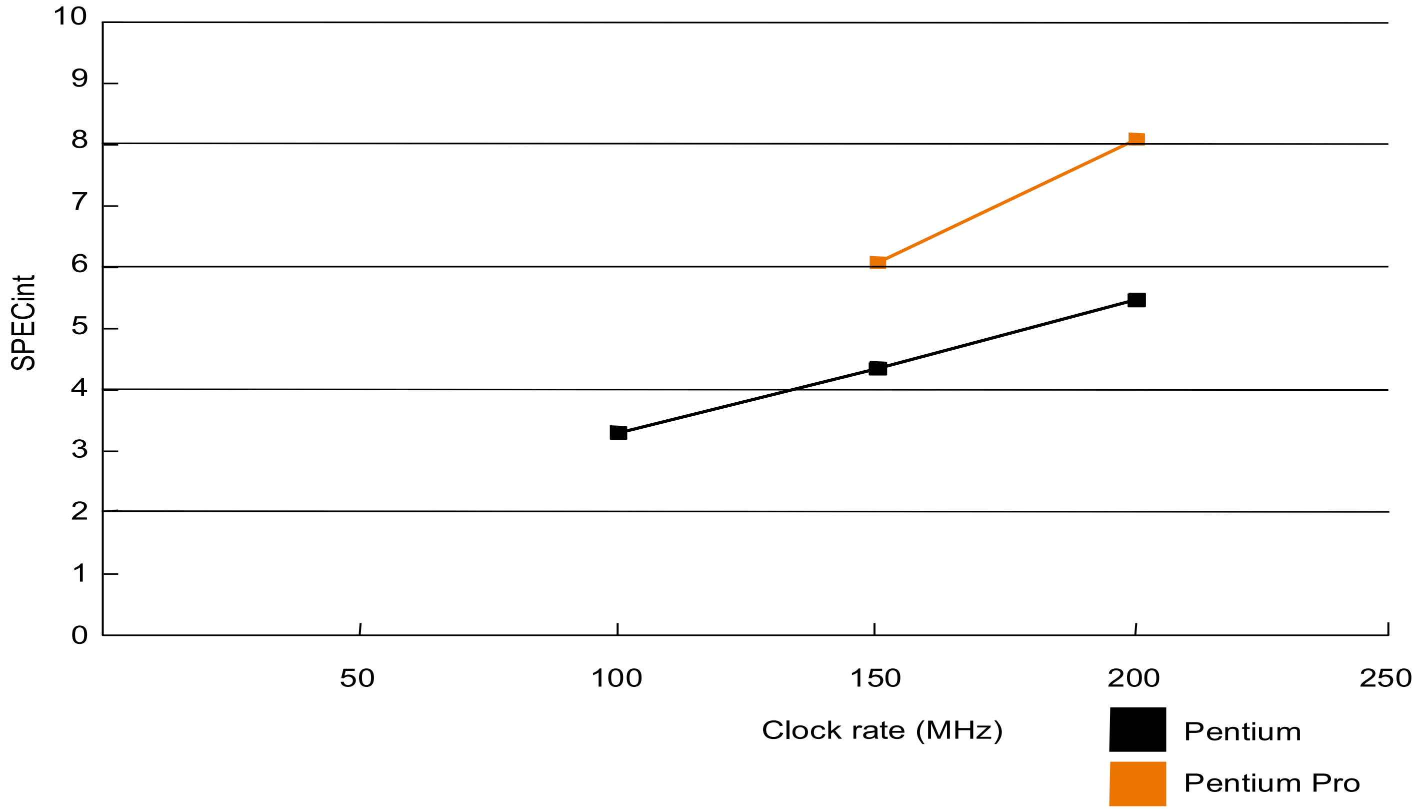
$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \text{ x } \frac{\text{Cycles}}{\text{Instruction}} \text{ x } \frac{\text{Seconds}}{\text{Cycle}}$$

	Inst Count	CPI	Clock Rate
Algorithm	X	X	
Programming Language	X	X	
Compiler	X	X	
ISA (instruction set architecture)	X	X	X

When comparing 2 machines, these "3 components" must be considered!

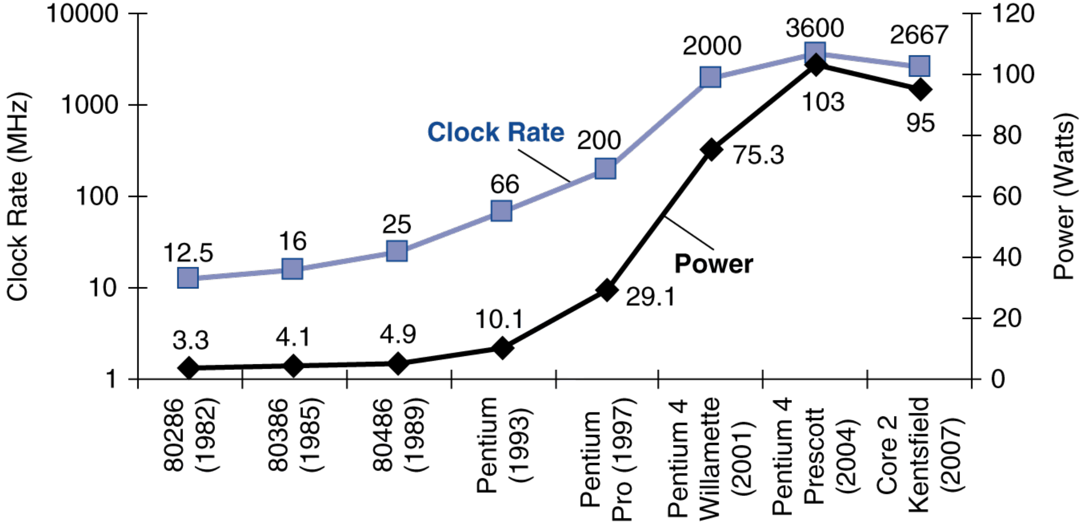
Now, you can answer this question..

- Q2: CPU frequency ? Performance



Power Trends

§1.5 The Power Wall



- In CMOS IC technology

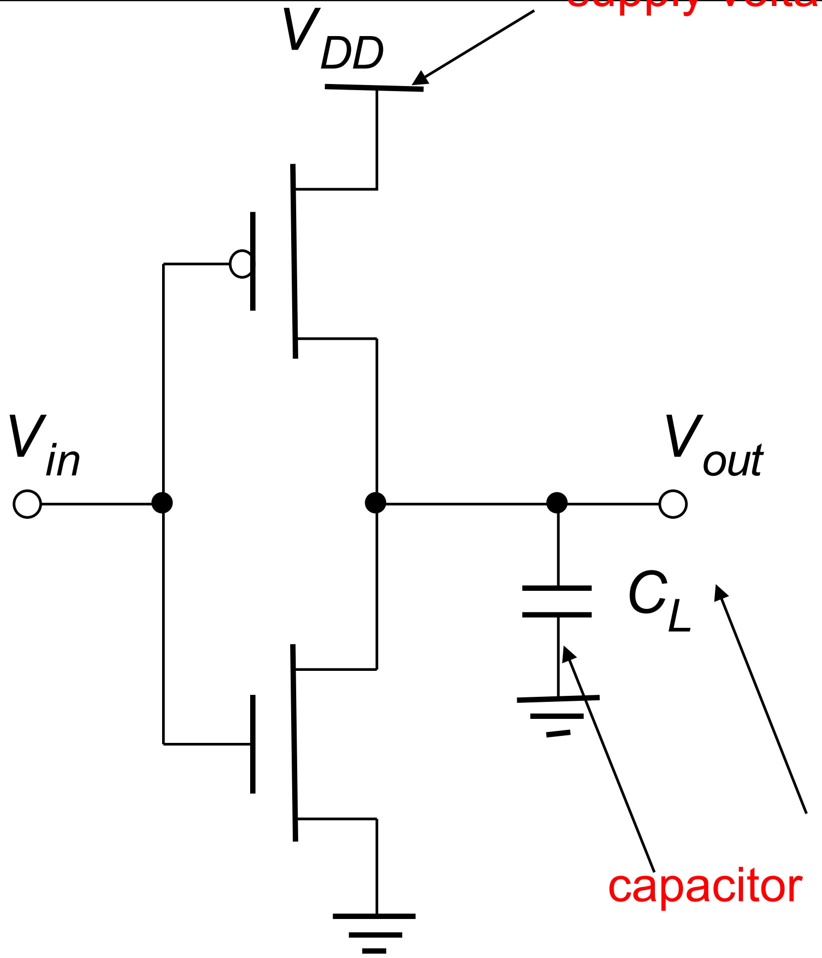
$$\text{Power} \quad 1/2 \cdot \text{Capacitive load} \cdot \text{Voltage}^2 \cdot \text{Frequency}$$

×30

5V → 1V

×1000

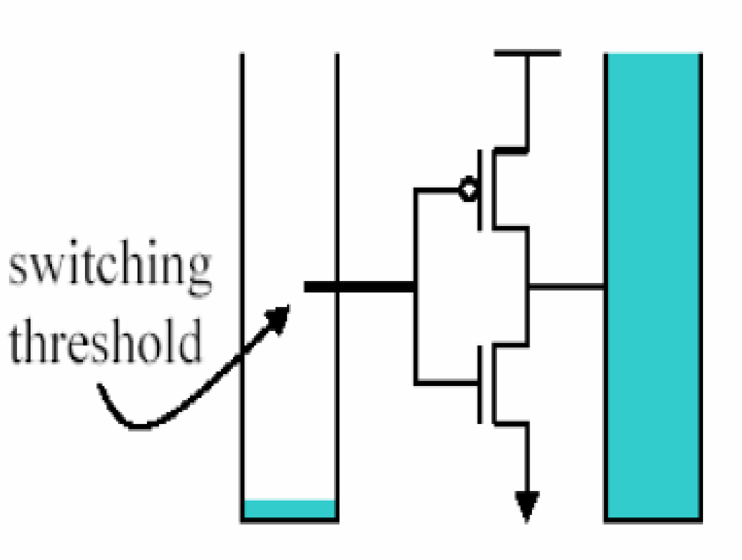
The CMOS Inverter



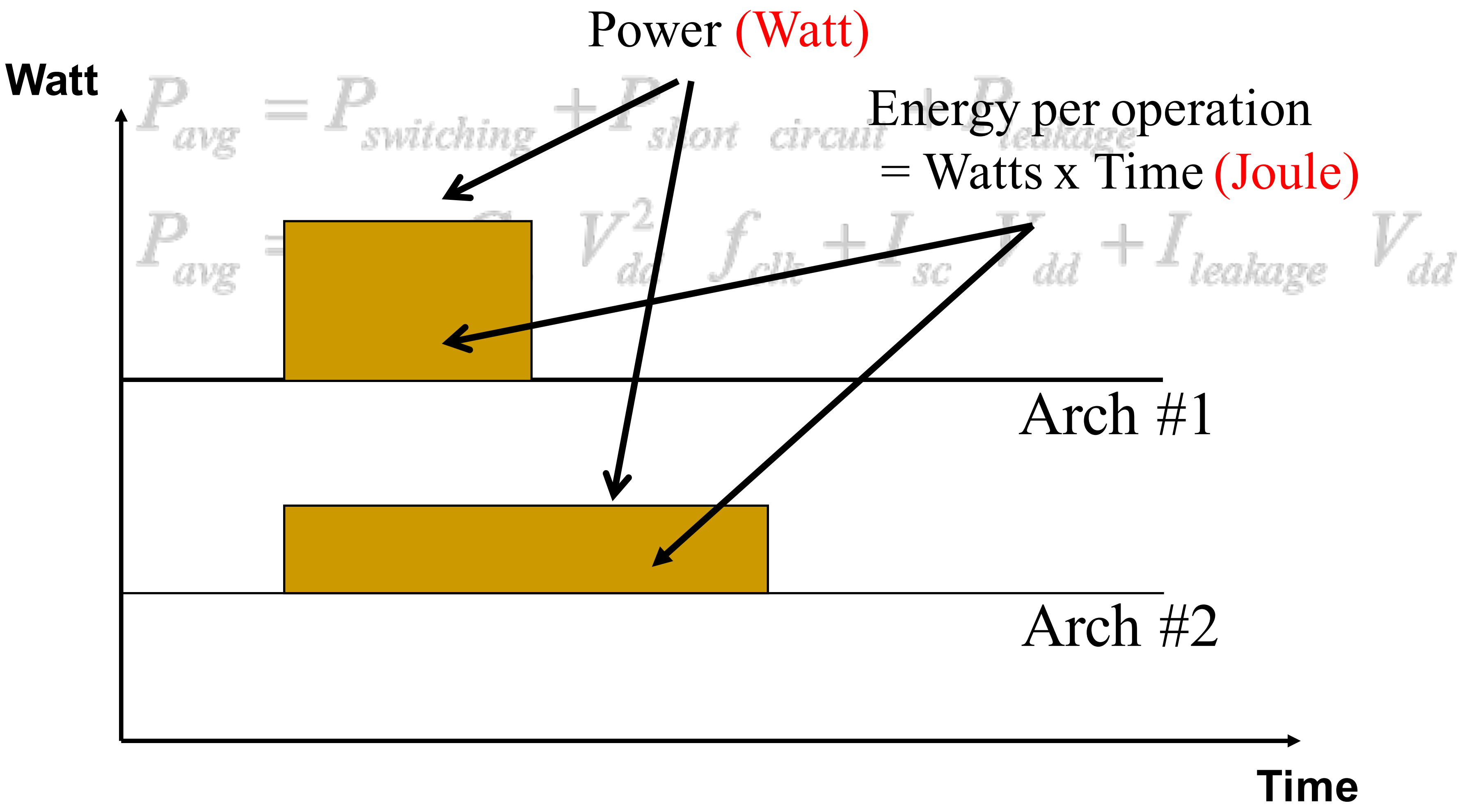
supply voltage

Capacitance

capacitor



Energy vs. Power



Power (Watt)

Energy per operation
= Watts x Time (Joule)

$$P_{avg} = P_{switching} + P_{short\ circuit} + P_{leakage}$$

$$P_{avg} = \alpha C V_{dd}^2 f_{clk} + I_{sc} V_{dd} + I_{leakage} V_{dd}$$

Reducing Power

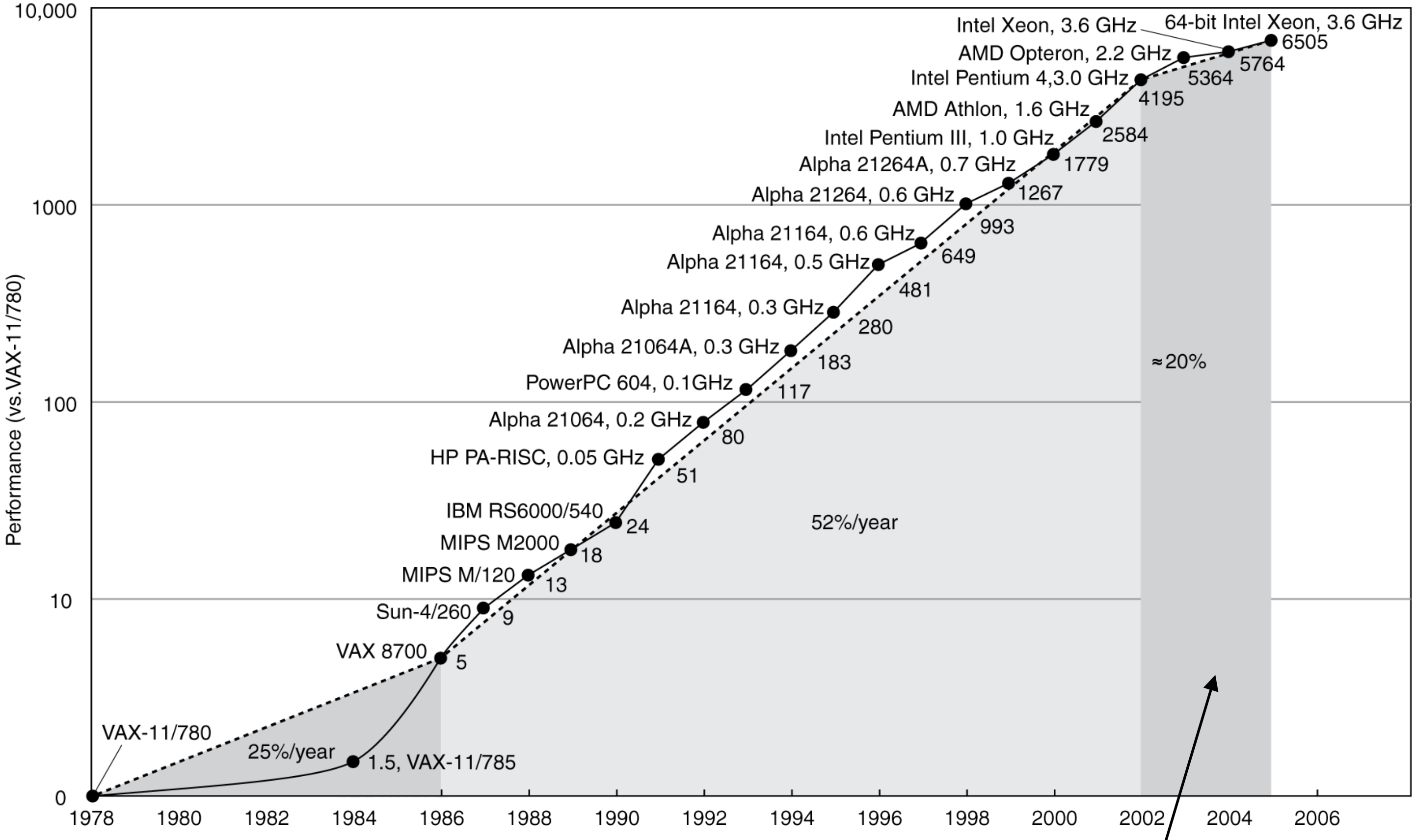
- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \cdot 0.85 \cdot (V_{old} \cdot 0.85)^2 \cdot F_{old} \cdot 0.85}{C_{old} \cdot V_{old}{}^2 \cdot F_{old}} = 0.85^4 = 0.52$$

- The power wall
  - We can’t reduce voltage further
  - We can’t remove more heat
- How else can we improve performance?

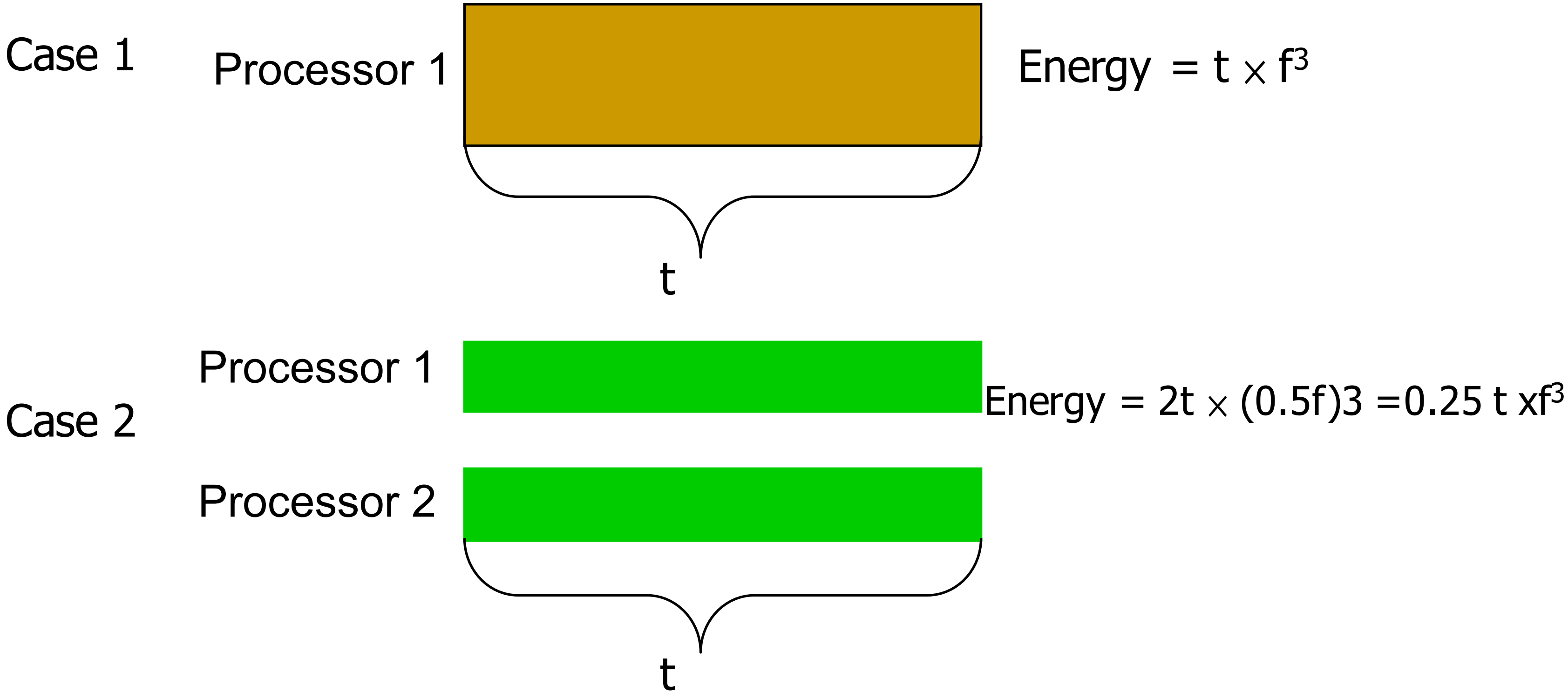
Uniprocessor Performance

§1.6 The Sea Change: The Switch to Multiprocessors



Constrained by power, instruction-level parallelism, memory latency

Why is Multi-Core Good for Energy-Efficiency?



Case 1

Processor 1

Energy = t × f^3

t

Case 2

Processor 1

Processor 2

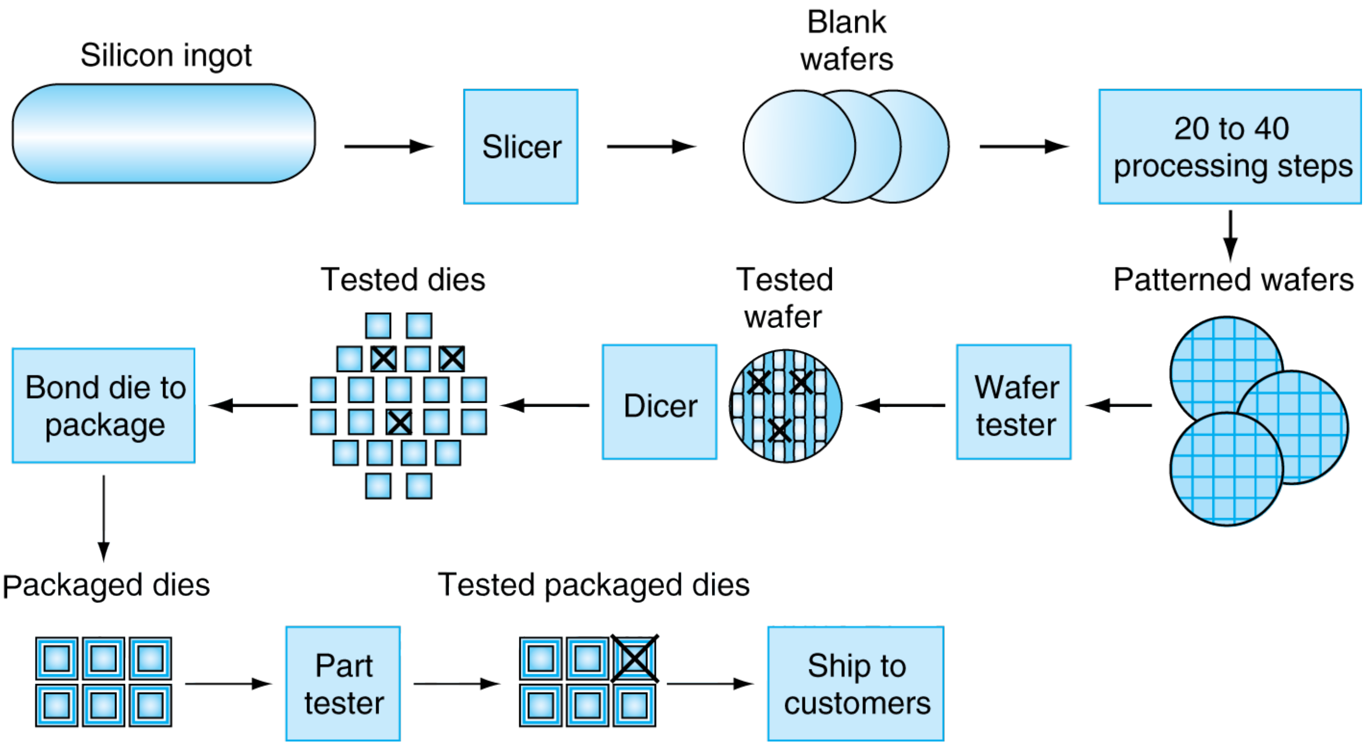
Energy = 2t × (0.5f)3 =0.25 t xf^3

t

Multiprocessors

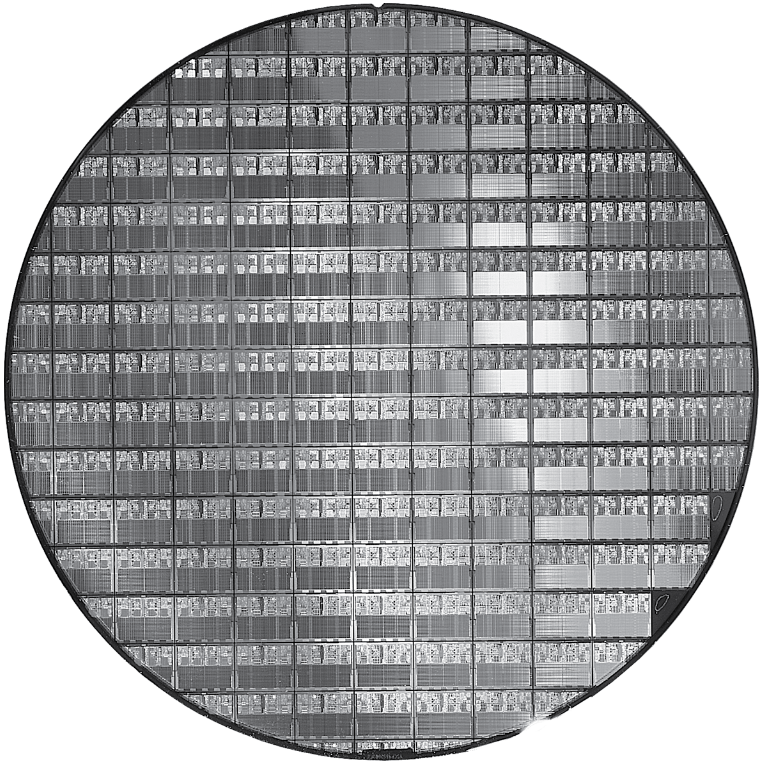
- Multicore microprocessors
  - More than one processor per chip
- Requires explicitly parallel programming
  - Compare with instruction level parallelism
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Hard to do
    - Programming for performance
    - Load balancing
    - Optimizing communication and synchronization

Manufacturing ICs



- Yield: proportion of working dies per wafer

AMD Opteron X2 Wafer



- X2: 300mm wafer, 117 chips, 90nm technology
- X4: 45nm technology

Integrated Circuit Cost

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \cdot \text{Yield}}$$

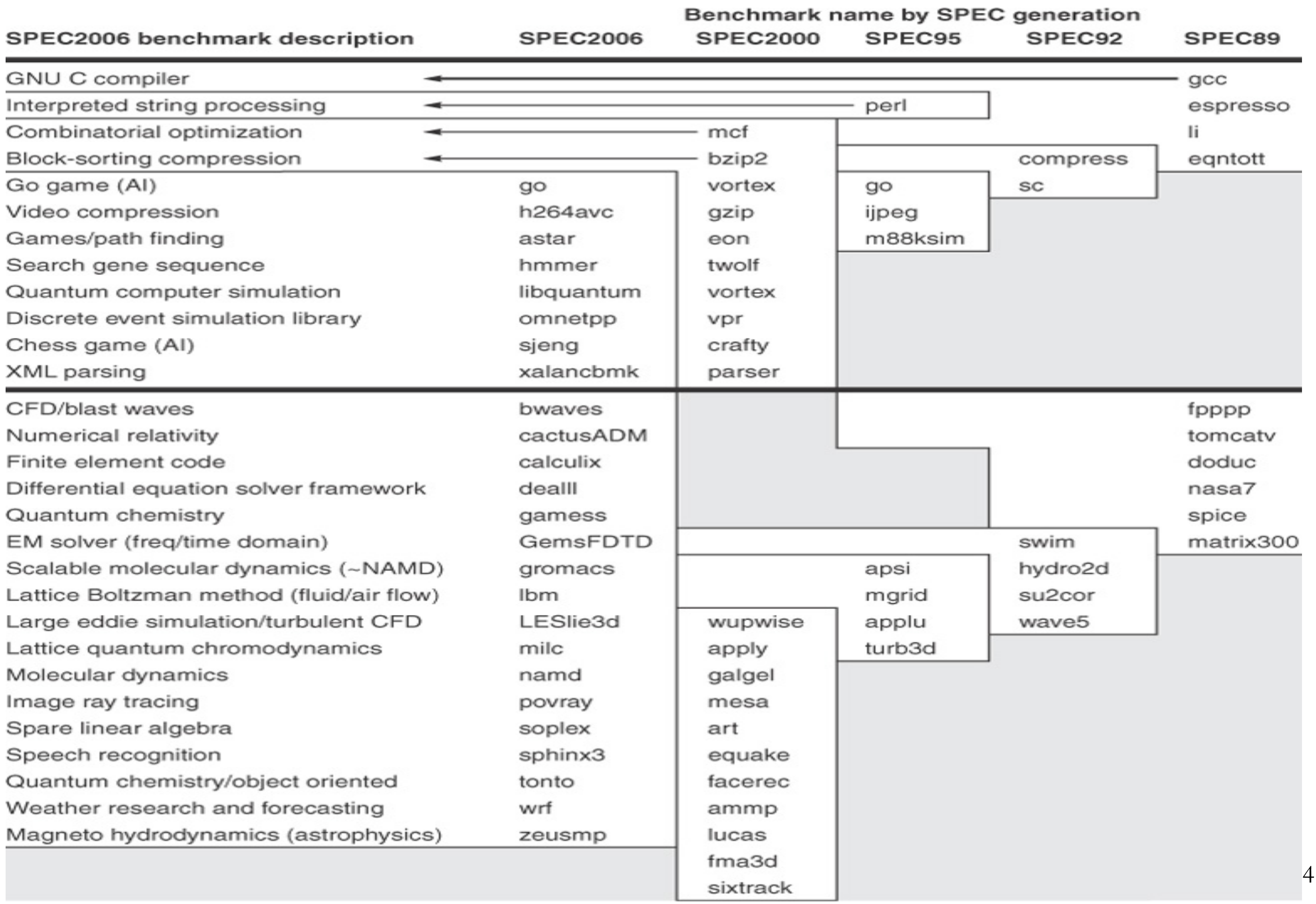
$$\text{Dies per wafer} \quad \text{Wafer area}/\text{Die area}$$

$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \cdot \text{Die area}/2))^2}$$

- Yield - proportion of working dies per wafer
- IC cost is nonlinear relation to area and defect rate
  - Wafer cost and area are fixed
  - Defect rate determined by manufacturing process
  - Die area determined by architecture and circuit design

SPEC CPU Benchmark

- Programs used to measure performance
  - Supposedly typical of actual workload
- Standard Performance Evaluation Corp (SPEC)
  - Develops benchmarks for CPU, I/O, Web, …
- Elapsed time to execute a selection of programs
  - Negligible I/O, so focuses on CPU performance
- Contain both integer and floating point applications
  - CINT (integer) and CFP (floating-point)



		Benchmark name by SPEC generation			
SPEC2006 benchmark description	**SPEC2006**	**SPEC2000**	**SPEC95**	**SPEC92**	**SPEC89**
GNU C compiler	←				gcc
Interpreted string processing	←		perl		espresso
Combinatorial optimization	←	mcf			li
Block-sorting compression	←	bzip2		compress	eqntott
Go game (AI)	go	vortex	go	sc	
Video compression	h264avc	gzip	ijpeg		
Games/path finding	astar	eon	m88ksim		
Search gene sequence	hmmer	twolf			
Quantum computer simulation	libquantum	vortex			
Discrete event simulation library	omnetpp	vpr			
Chess game (AI)	sjeng	crafty			
XML parsing	xalancbmk	parser			
CFD/blast waves	bwaves				fpppp
Numerical relativity	cactusADM				tomcatv
Finite element code	calculix				doduc
Differential equation solver framework	dealII				nasa7
Quantum chemistry	gamess				spice
EM solver (freq/time domain)	GemsFDTD			swim	matrix300
Scalable molecular dynamics (~NAMD)	gromacs		apsi	hydro2d	
Lattice Boltzman method (fluid/air flow)	lbm		mgrid	su2cor	
Large eddie simulation/turbulent CFD	LESlie3d	wupwise	applu	wave5	
Lattice quantum chromodynamics	milc	apply	turb3d		
Molecular dynamics	namd	galgel			
Image ray tracing	povray	mesa			
Spare linear algebra	soplex	art			
Speech recognition	sphinx3	equake			
Quantum chemistry/object oriented	tonto	facerec			
Weather research and forecasting	wrf	ammp			
Magneto hydrodynamics (astrophysics)	zeusmp	lucas			
		fma3d			
		sixtrack			

How to Summarize Suite Performance

- Arithmetic average of execution time of all pgms?
  - But they vary by 4X in speed, so some would be more important than others in arithmetic average
- SPECRatio: Normalize execution times to reference computer, yielding a ratio proportional to performance =

$$\frac{\text{time on reference computer}}{\text{time on computer being rated}}$$

SPECRatio

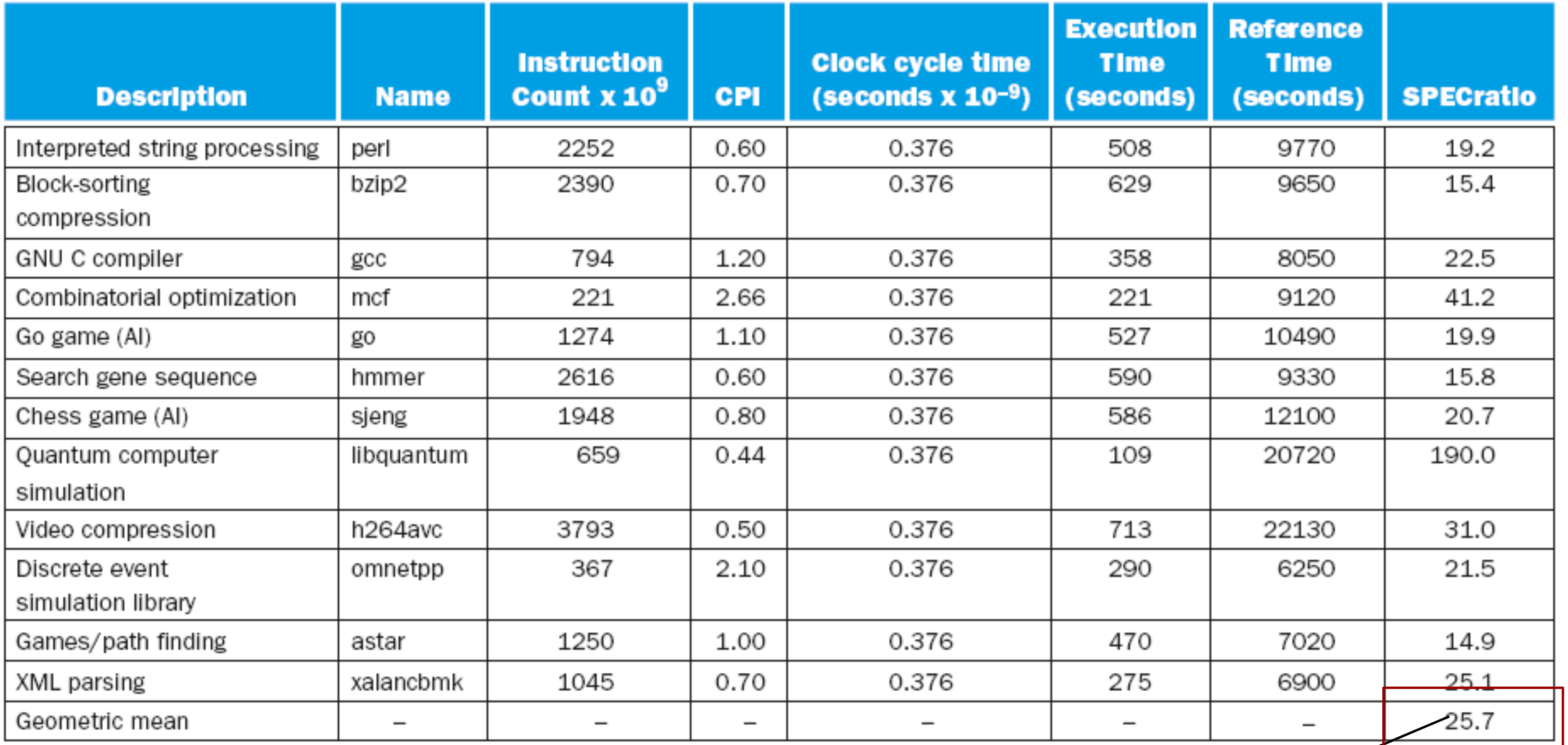
- If program SPECRatio on Computer A is 1.25 times bigger than Computer B, then

$$1.25 = \frac{SPECRatio_A}{SPECRatio_B} = \frac{\dfrac{ExecutionTime_{reference}}{ExecutionTime_A}}{\dfrac{ExecutionTime_{reference}}{ExecutionTime_B}}$$

$$= \frac{ExecutionTime_B}{ExecutionTime_A} = \frac{Performance_A}{Performance_B}$$

  - Note that when comparing 2 computers as a ratio, execution times on the reference computer drop out, so choice of reference computer is irrelevant

CINT2006 for Intel Core i7 920



Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

$$GeometricMean = \sqrt[n]{\prod_{i=1}^{n} SPECRatio_i}$$

SPEC Power Benchmark

- Specpower: Power consumption of server at different workload levels
  - Run SPECJBB2005 (Java Business Application)
  - Report power consumption of servers at different workload levels, divided into 10% increments
  - Performance: ssj_ops/sec
  - Power: Watts (Joules/sec)

$$\text{Overall ssj\_ops per Watt} = \sum_{i=0}^{10} \text{ssj\_ops}_i \bigg/ \sum_{i=0}^{10} \text{power}_i$$

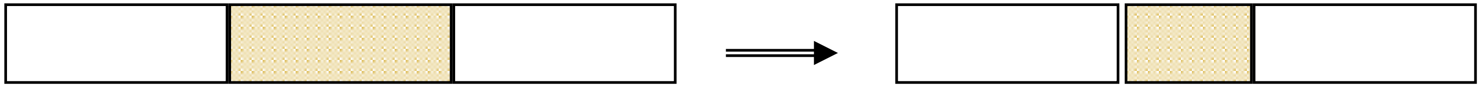
SPECpower_ssj2008 for X4

Target Load %	Performance (ssj_ops/sec)	Average Power (Watts)
100%	231,867	295
90%	211,282	286
80%	185,803	275
70%	163,427	265
60%	140,160	256
50%	118,324	246
40%	920,35	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall sum	1,283,590	2,605
∑ssj_ops/ ∑power		493

Pitfall: Amdahl's Law

§1.8 Fallacies and Pitfalls

- Improving an aspect of a computer and expecting a proportional improvement in overall performance



$$T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}}$$

- Example: multiply accounts for 80s/100s
  - How much improvement in multiply performance to get 5× overall?

$$20 = \frac{80}{n} + 20$$

  - Can't be done!
- Corollary: make the common case fast

Fallacy: Low Power at Idle

- Look back at X4 power benchmark
  - At 100% load: 295W
  - At 50% load: 246W (83%)
  - At 10% load: 180W (61%)
- Google data center
  - Mostly operates at 10% – 50% load
  - At 100% load less than 1% of the time
- Consider designing processors to make power proportional to load

Pitfall: MIPS as a Performance Metric

- MIPS: Millions of Instructions Per Second
  - Doesn’t account for
    - Differences in ISAs between computers
    - Differences in complexity between instructions
      - CPI varies between programs on a given CPU (Can’t have single MIPS index for a processor)

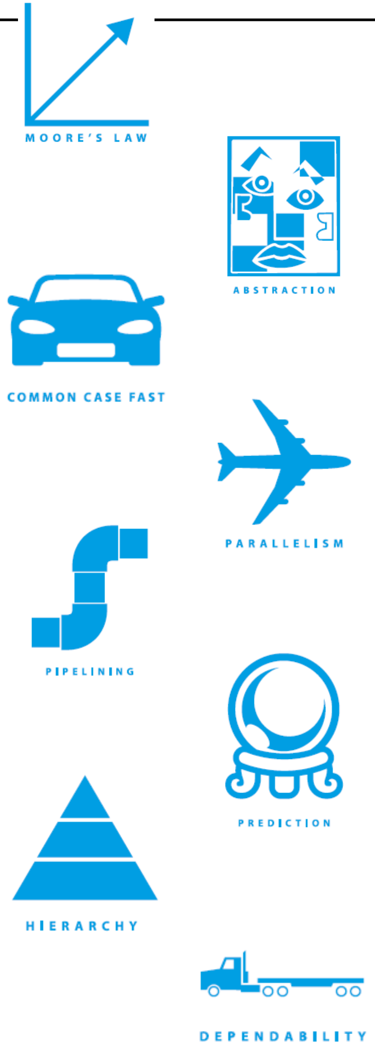
$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \cdot 10^6} = \frac{\text{Instruction count}}{\dfrac{\text{Instruction count} \cdot \text{CPI}}{\text{Clock rate}} \cdot 10^6} = \frac{\text{Clock rate}}{\text{CPI} \cdot 10^6}$$

- CPU’s performance can’t be represented by a single MIPS value
- Different CPUs can’t be compared with MIPS

§1.2 Eight Great Ideas in Computer Architecture

Eight Design Principle for Computer Architecture/System

- Design for ***Moore's Law***
- Use ***abstraction*** to simplify design
- Make the ***common case fast***
- Performance *via* ***parallelism***
- Performance *via* ***pipelining***
- Performance *via* ***prediction***
- ***Hierarchy*** of memories
- ***Dependability*** *via* redundancy



Homework #1

- – Due next week before classes