Introduction to Computer Design

E. Sanchez

Politecnico di Torino
Dipartimento di Automatica e Informatica

Computer evolution

The first general-purpose computer was created in the late 40s.

A Personal Computer, that can now be bought for about $500, is practically equivalent (in terms of performance and memory) to what could be bought for about $1M in 1985.

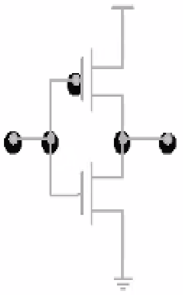
This evolution has been made possible by

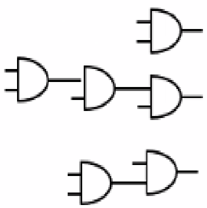
- Advances in semiconductor technology
- Innovations in computer design.

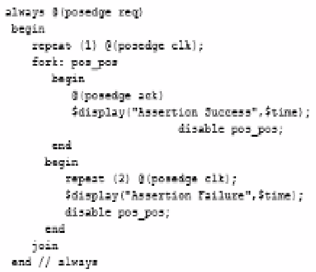
Chip-design history

	1970	1980	1990	2000	2015	Today
# transistors	2,000	30,000	1,000,000	40,000,000	>10,000,000,000	HUGE
Design complexity	Design Rule Checking	P&R, Timing closure	Logic Synthesis	IP-Based Design	Design platform and Collaboration	AI-Based Design

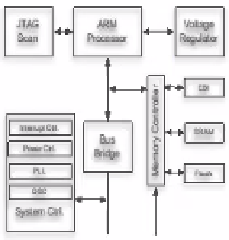
Design entry

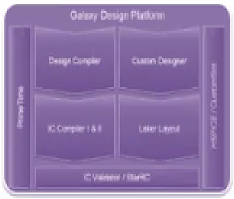


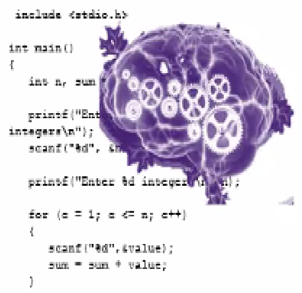




```
always @(posedge req)
 begin
    repeat (1) @(posedge clk);
    fork: pos_pos
       begin
          @(posedge ack)
          $display("Assertion Success",$time);
                          disable pos_pos;
       end
      begin
         repeat (2) @(posedge clk);
         $display("Assertion Failure",$time);
         disable pos_pos;
      end
    join
 end // always
```



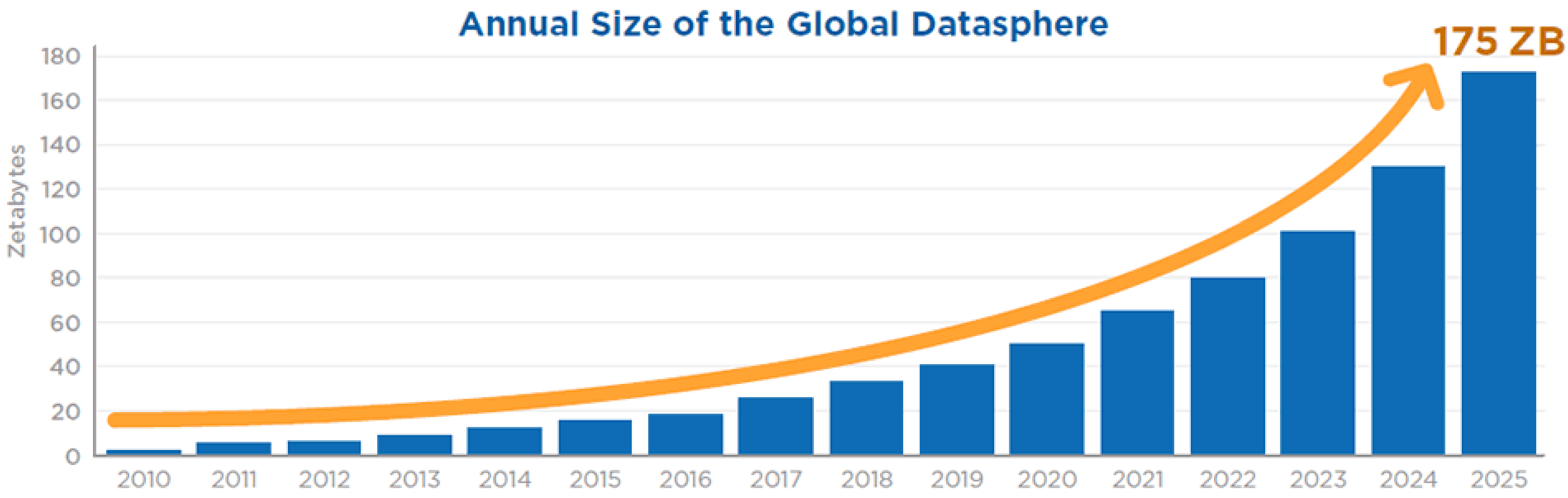




```
 include <stdio.h>

int main()
{
   int n, sum

   printf("Ent
integers\n");
   scanf("%d", &n

   printf("Enter %d integer

   for (c = 1; c <= n; c++)
   {
      scanf("%d",&value);
      sum = sum + value;
   }
```

Annual size of the global Datasphere

Figure 1 – Annual Size of the Global Datasphere

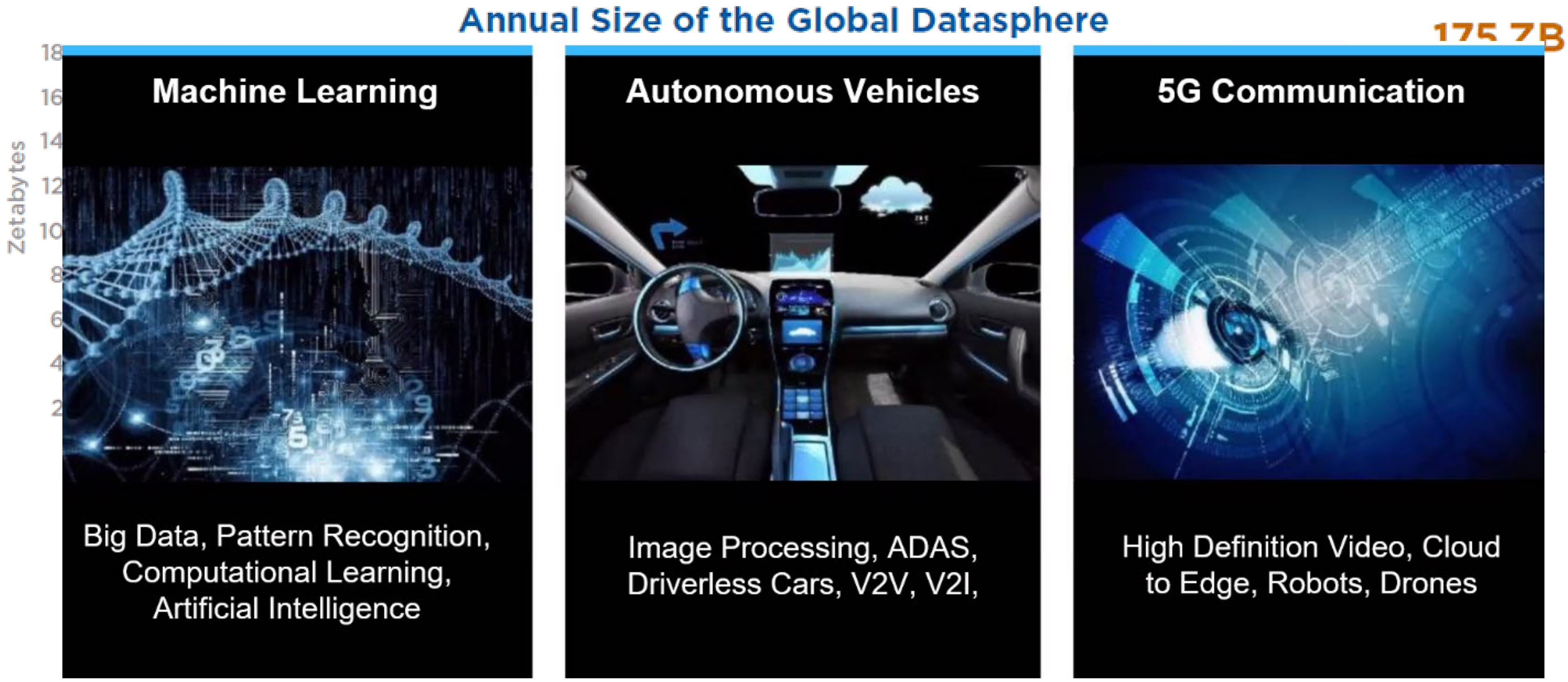


source: IDC Datasphere whitepaper

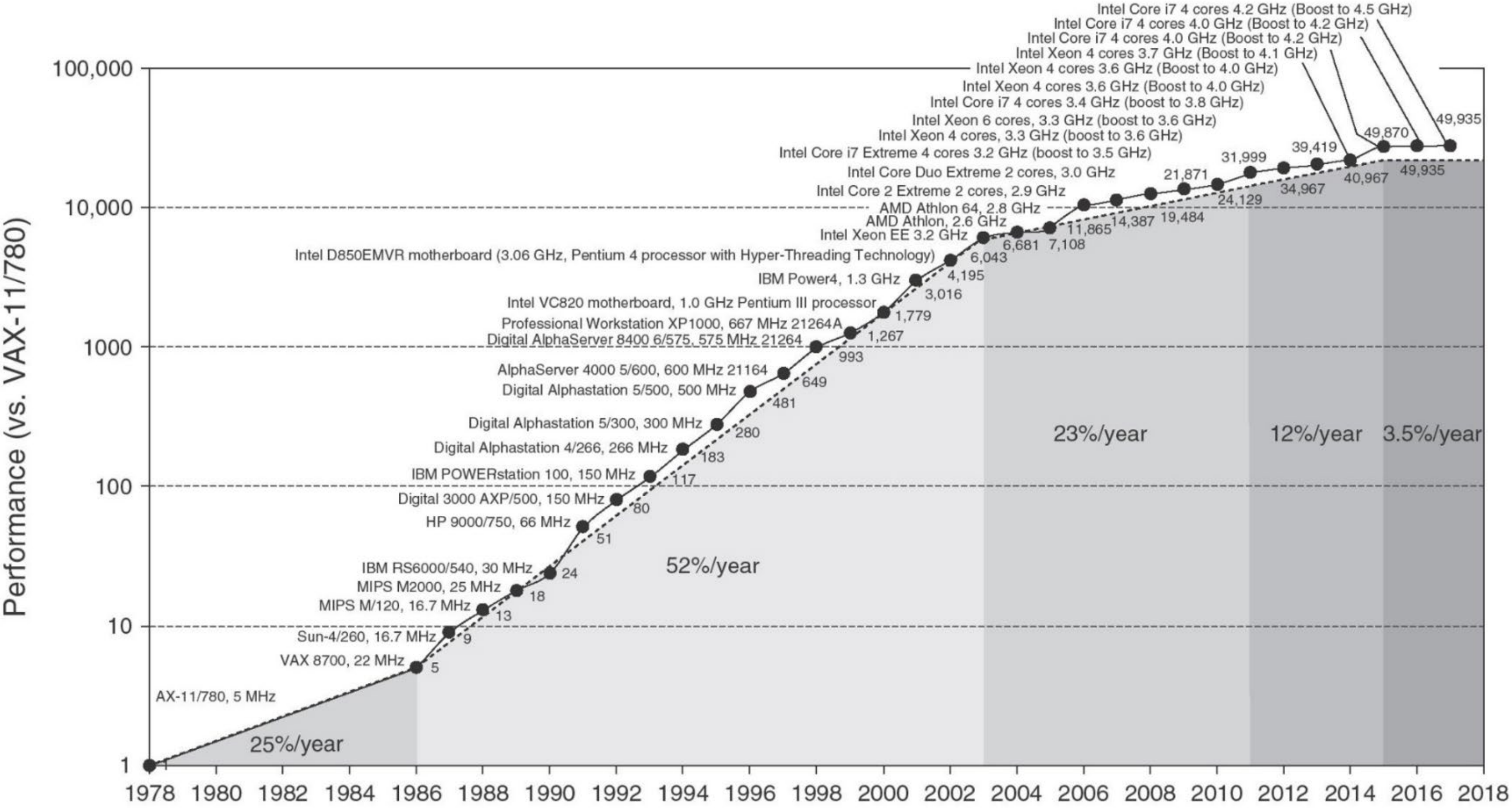
zettaByte ZB -> 10^{21} -> 2^{70}

Annual size of the global Datasphere

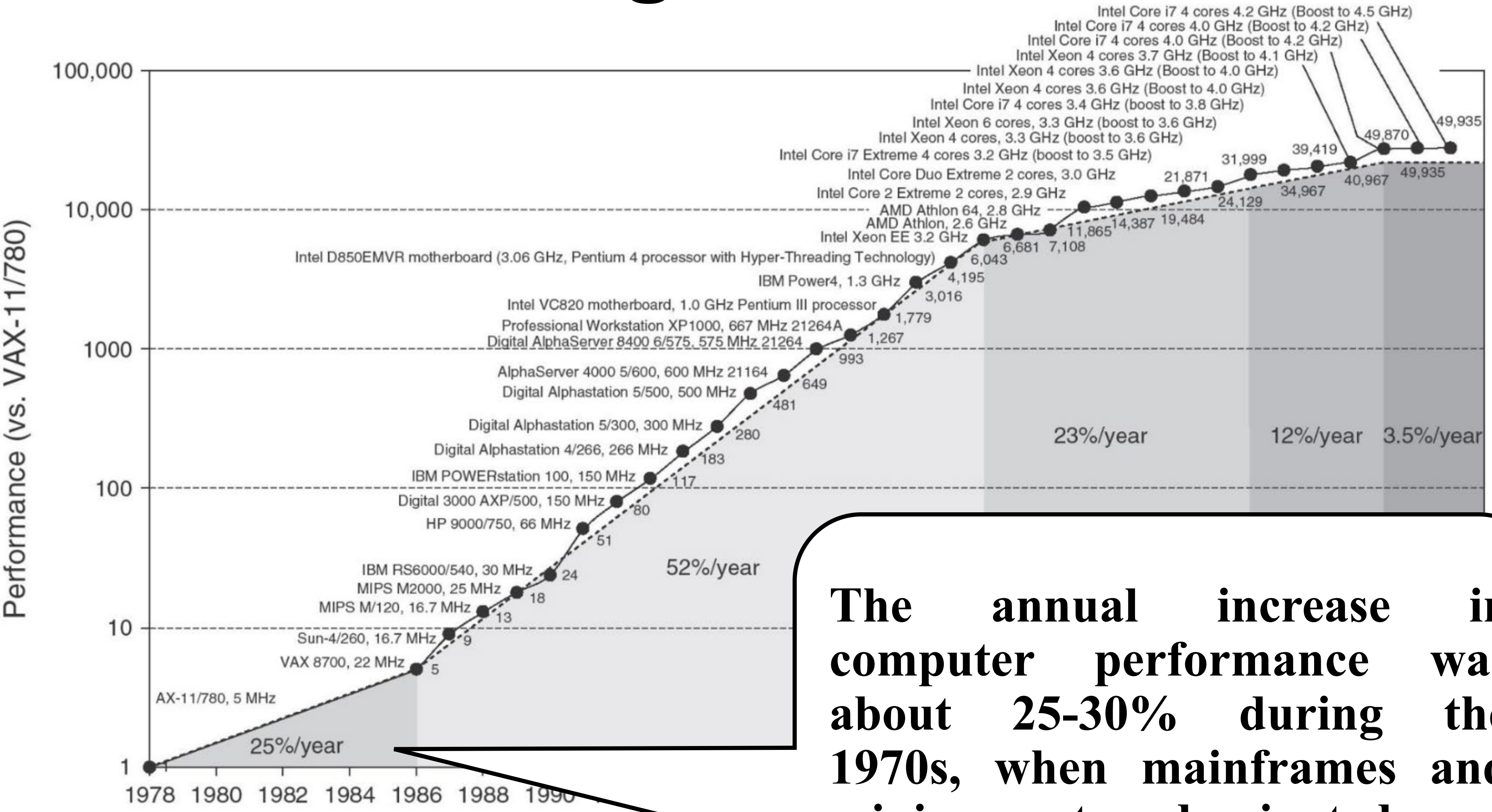
Figure 1 – Annual Size of the Global Datasphere


Annual Size of the Global Datasphere
175 ZB
18
16
14
12
10
8
6
4
2
Zetabytes
Machine Learning
Big Data, Pattern Recognition, Computational Learning, Artificial Intelligence
Autonomous Vehicles
Image Processing, ADAS, Driverless Cars, V2V, V2I,
5G Communication
High Definition Video, Cloud to Edge, Robots, Drones

Microprocessor performance growth



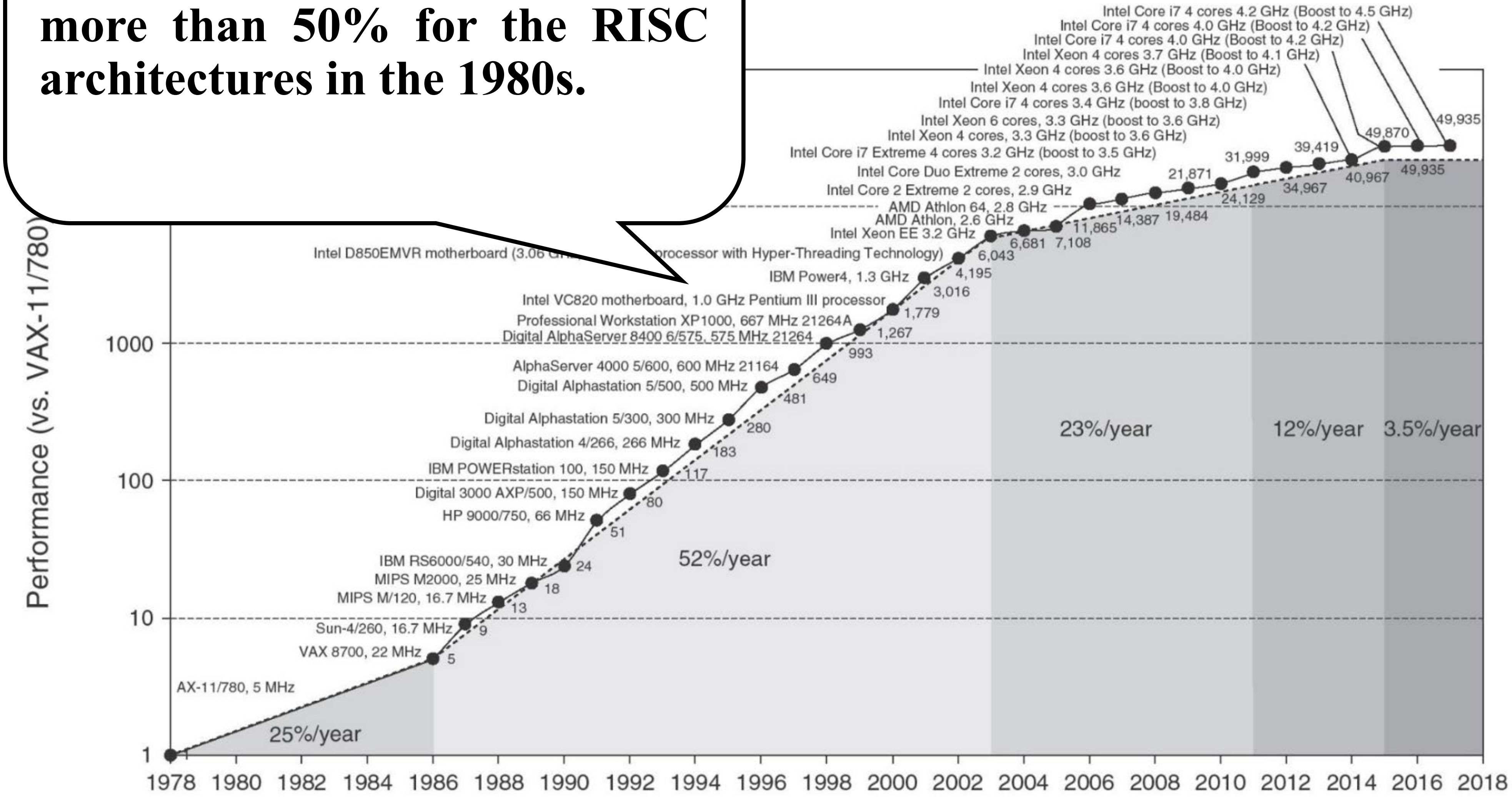
Microprocessor performance growth



The annual increase in computer performance was about 25-30% during the 1970s, when mainframes and minicomputers dominated.

Mi sor performance
owth

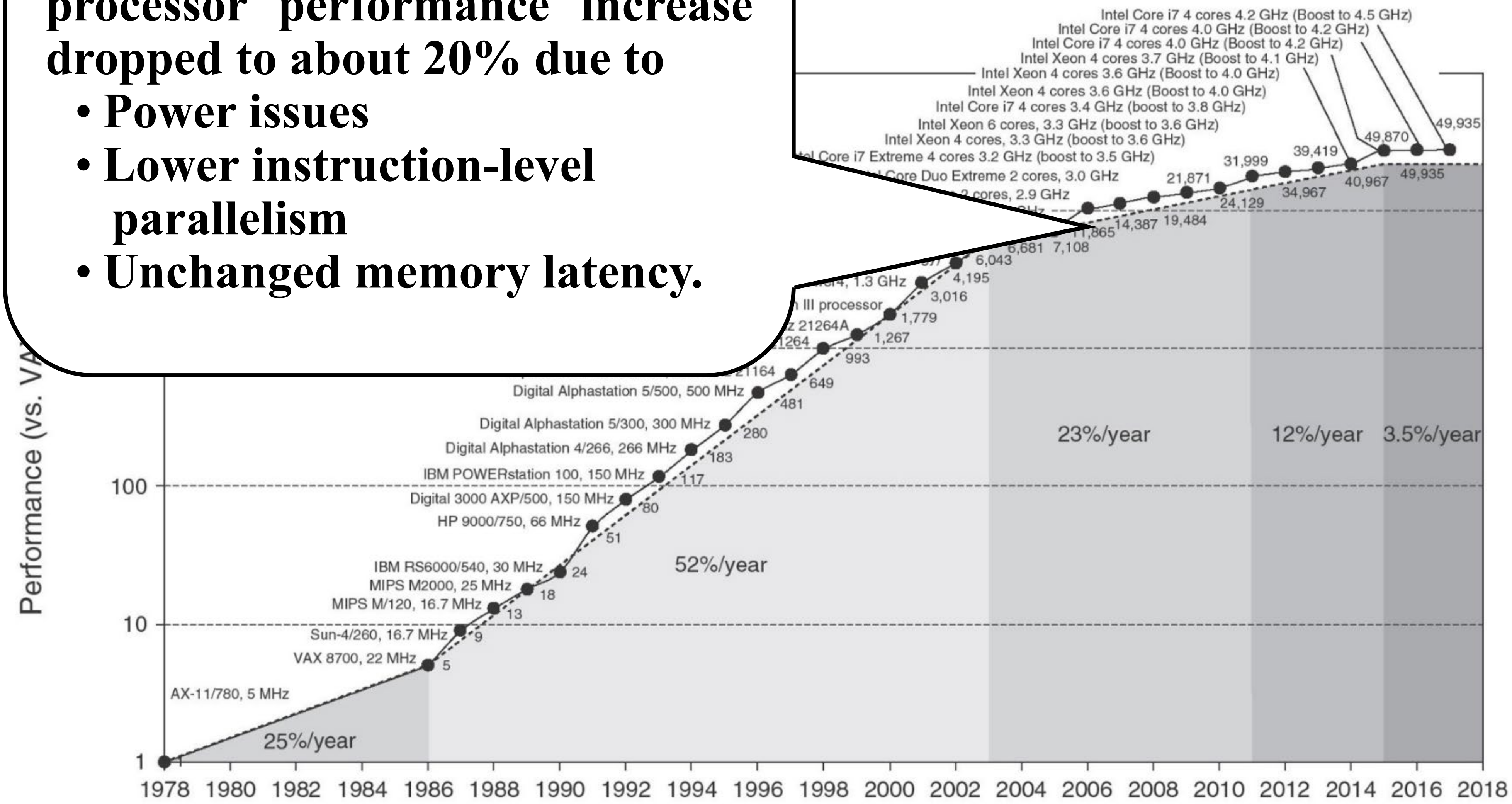
The annual increase raised to more than 50% for the RISC architectures in the 1980s.



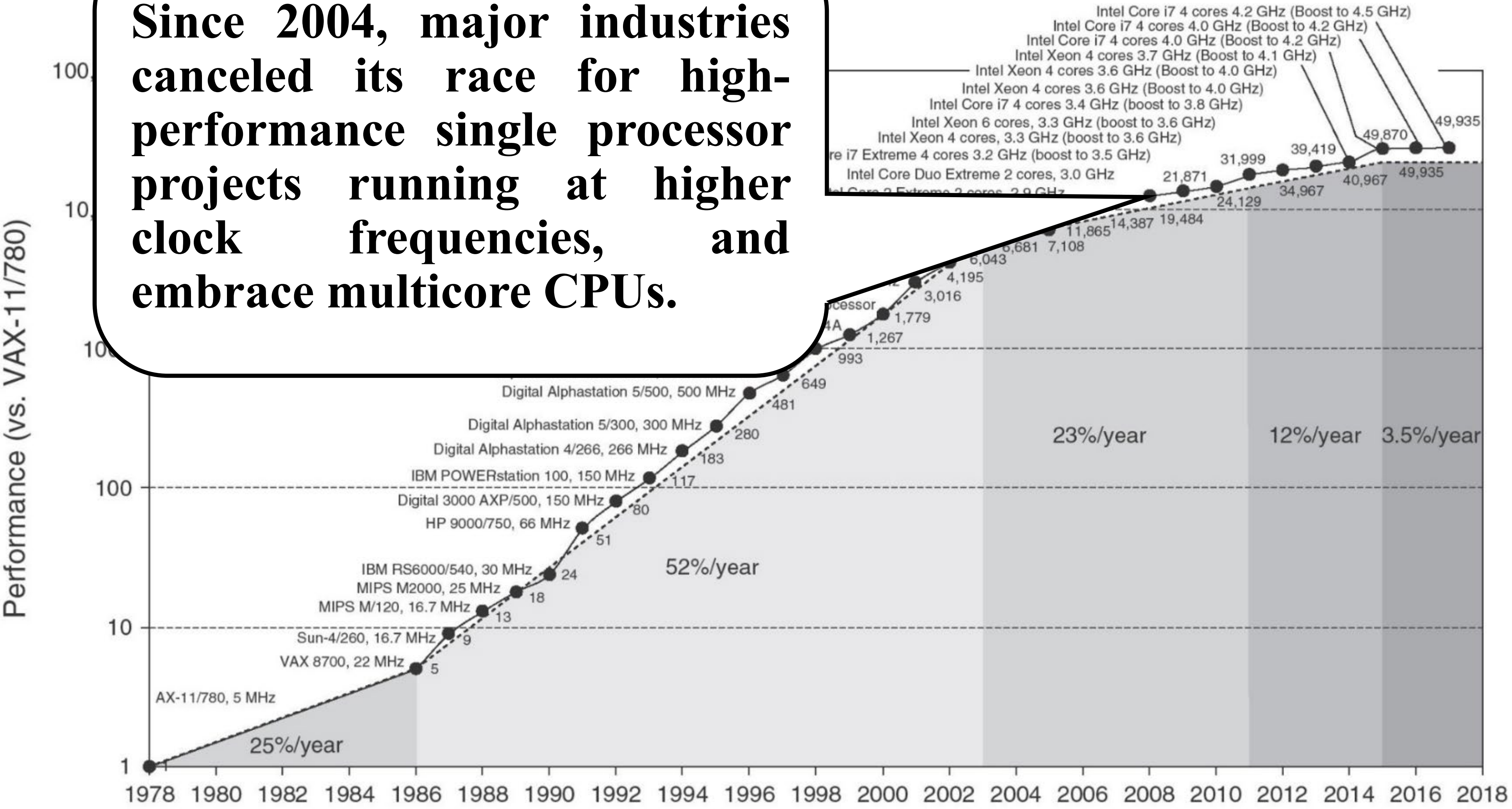
Microprocessor performance
wth

Starting from 2002, the yearly processor performance increase dropped to about 20% due to

- Power issues
- Lower instruction-level parallelism
- Unchanged memory latency.



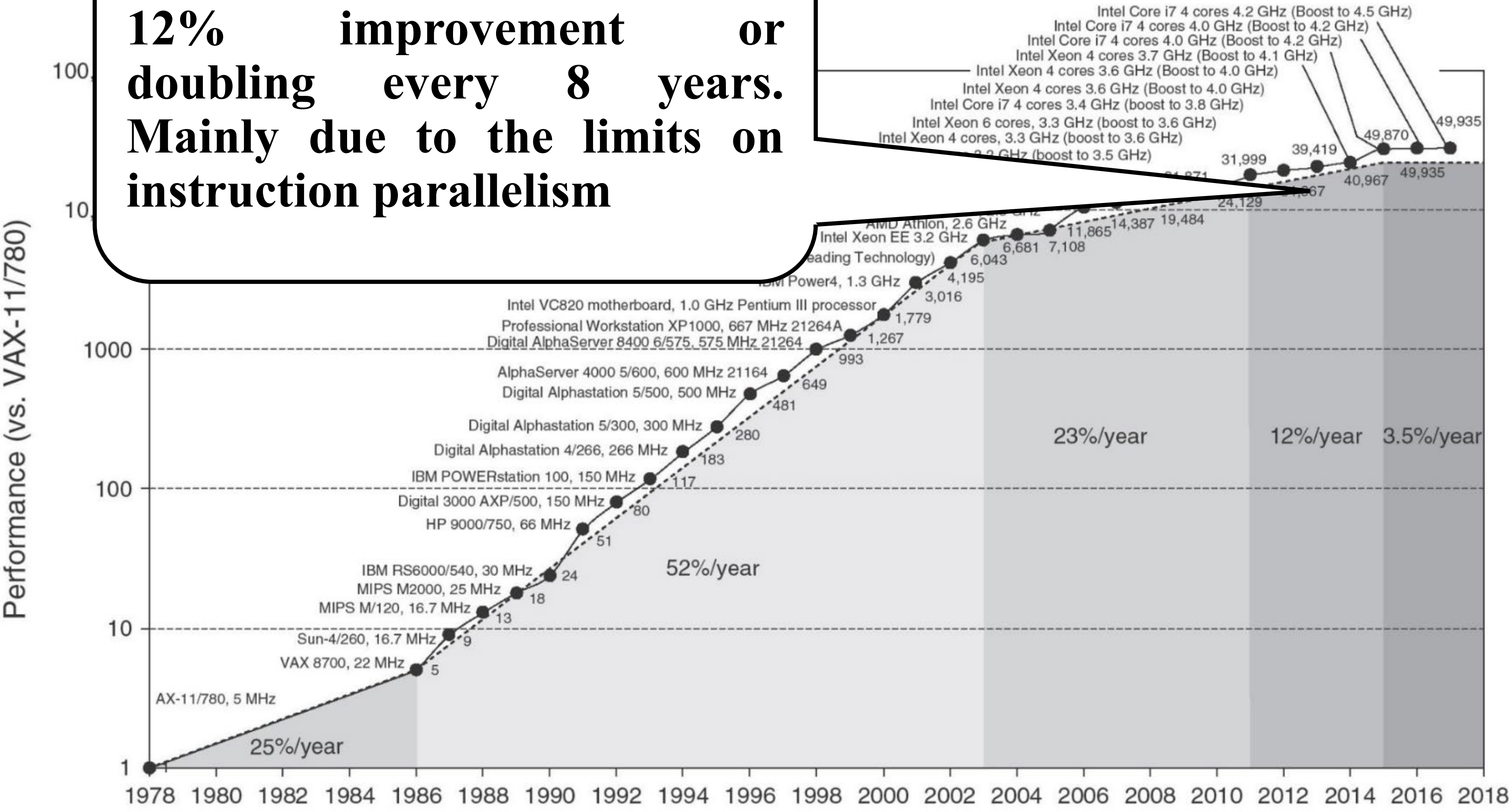
Microprocessor performance growth



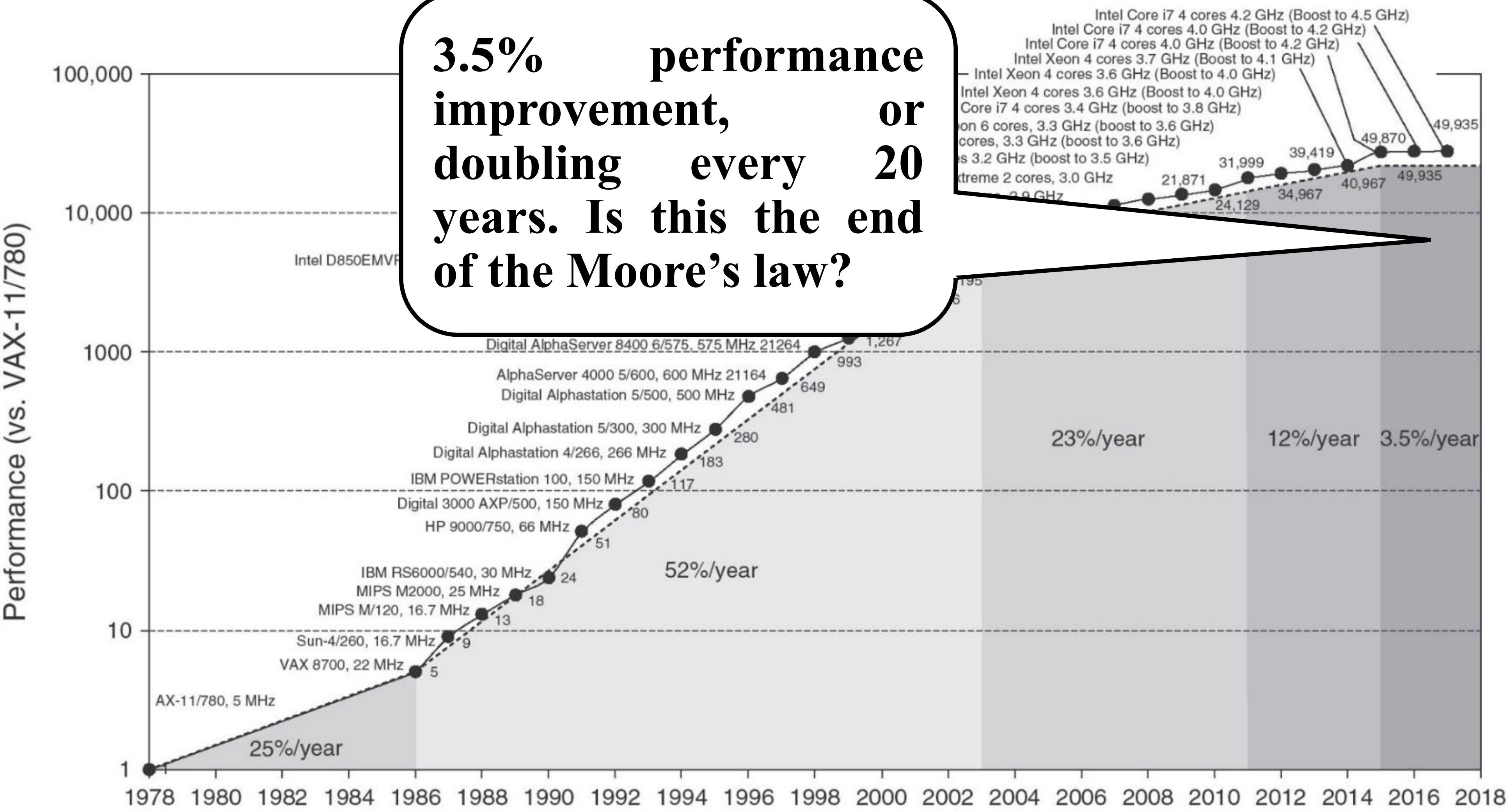
Since 2004, major industries canceled its race for high-performance single processor projects running at higher clock frequencies, and embrace multicore CPUs.

Microprocessor performance growth

12% improvement or doubling every 8 years. Mainly due to the limits on instruction parallelism

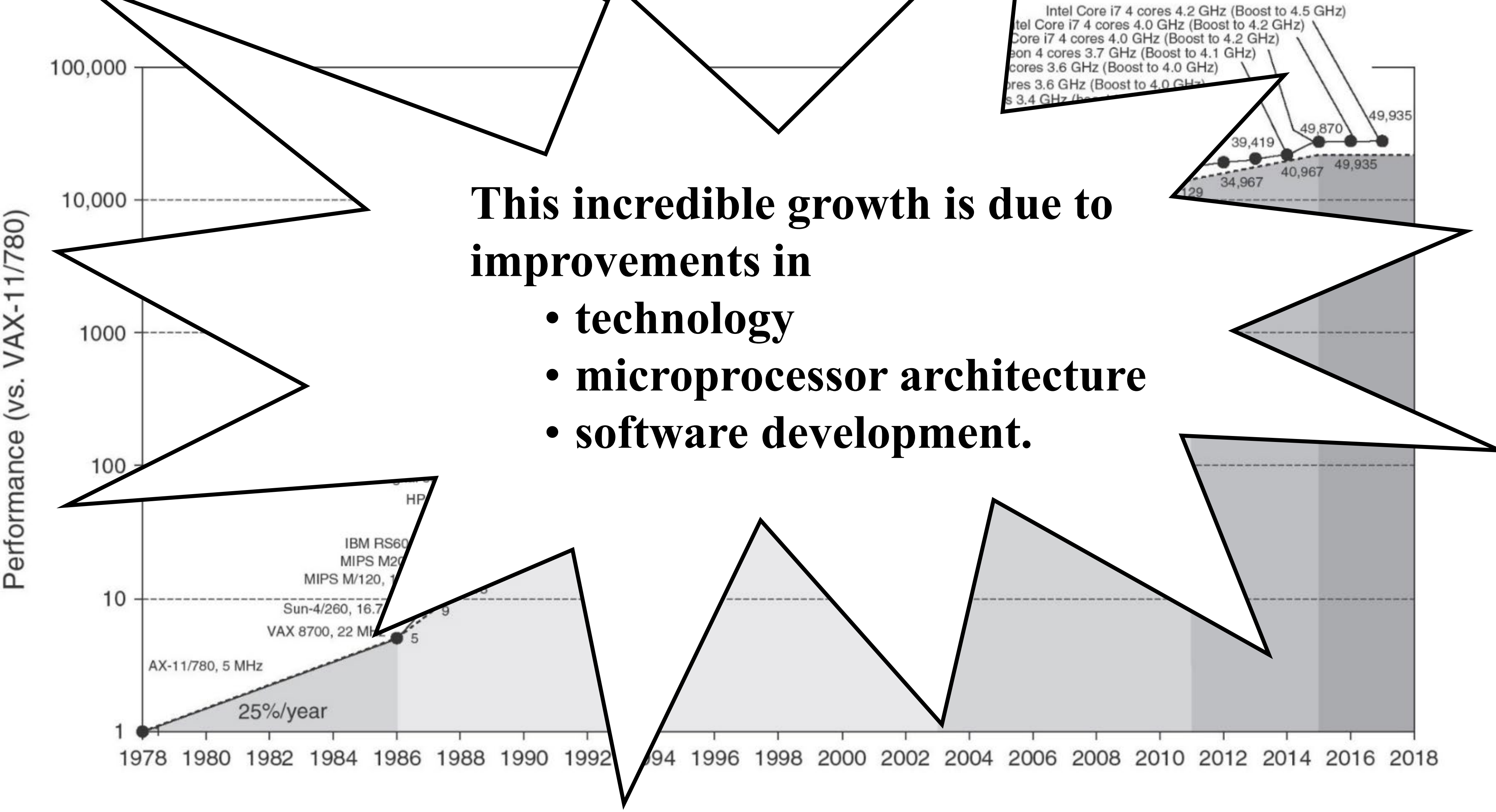


Microprocessor performance growth



3.5% performance improvement, or doubling every 20 years. Is this the end of the Moore's law?

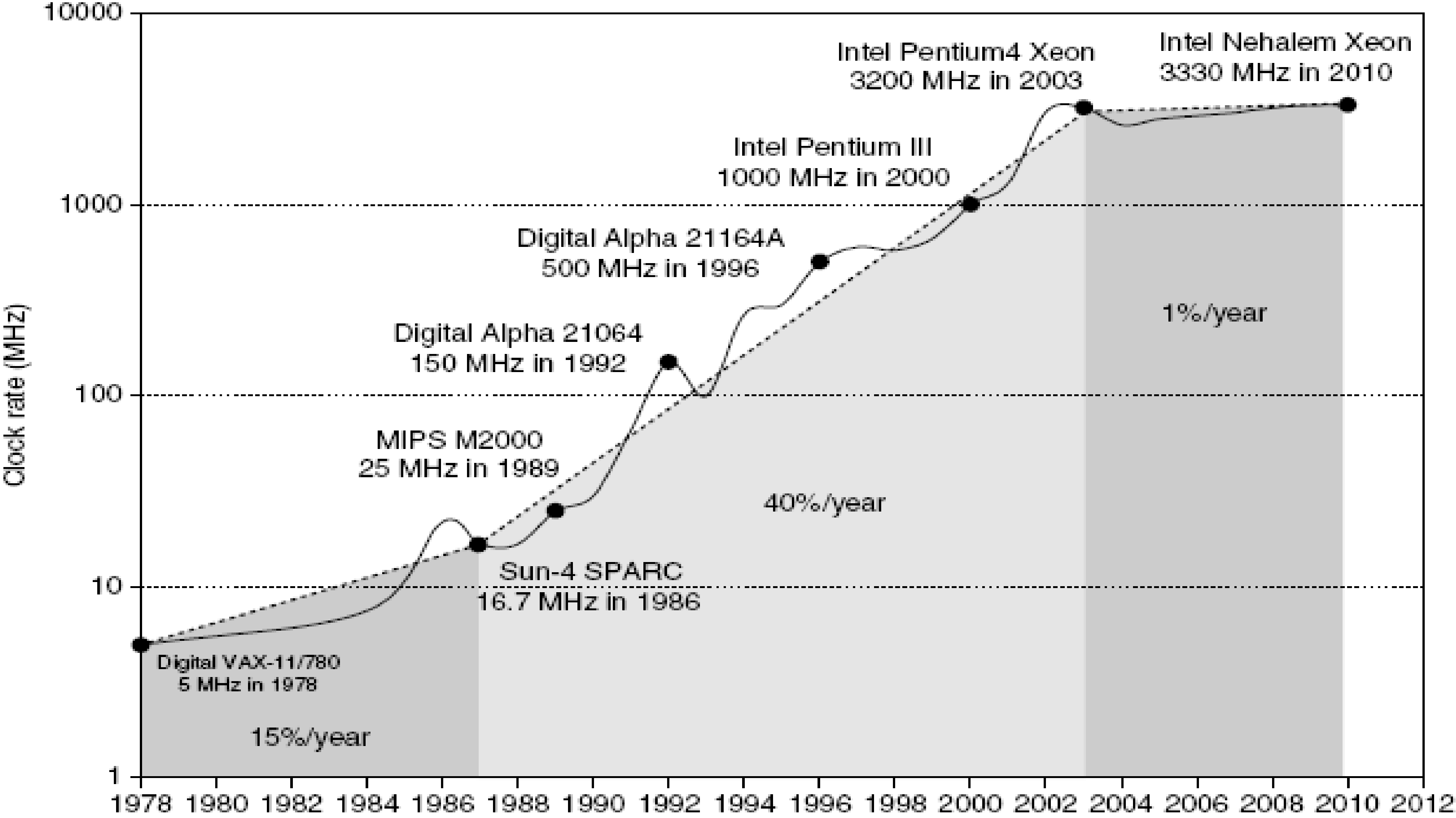
Microprocessor performance growth



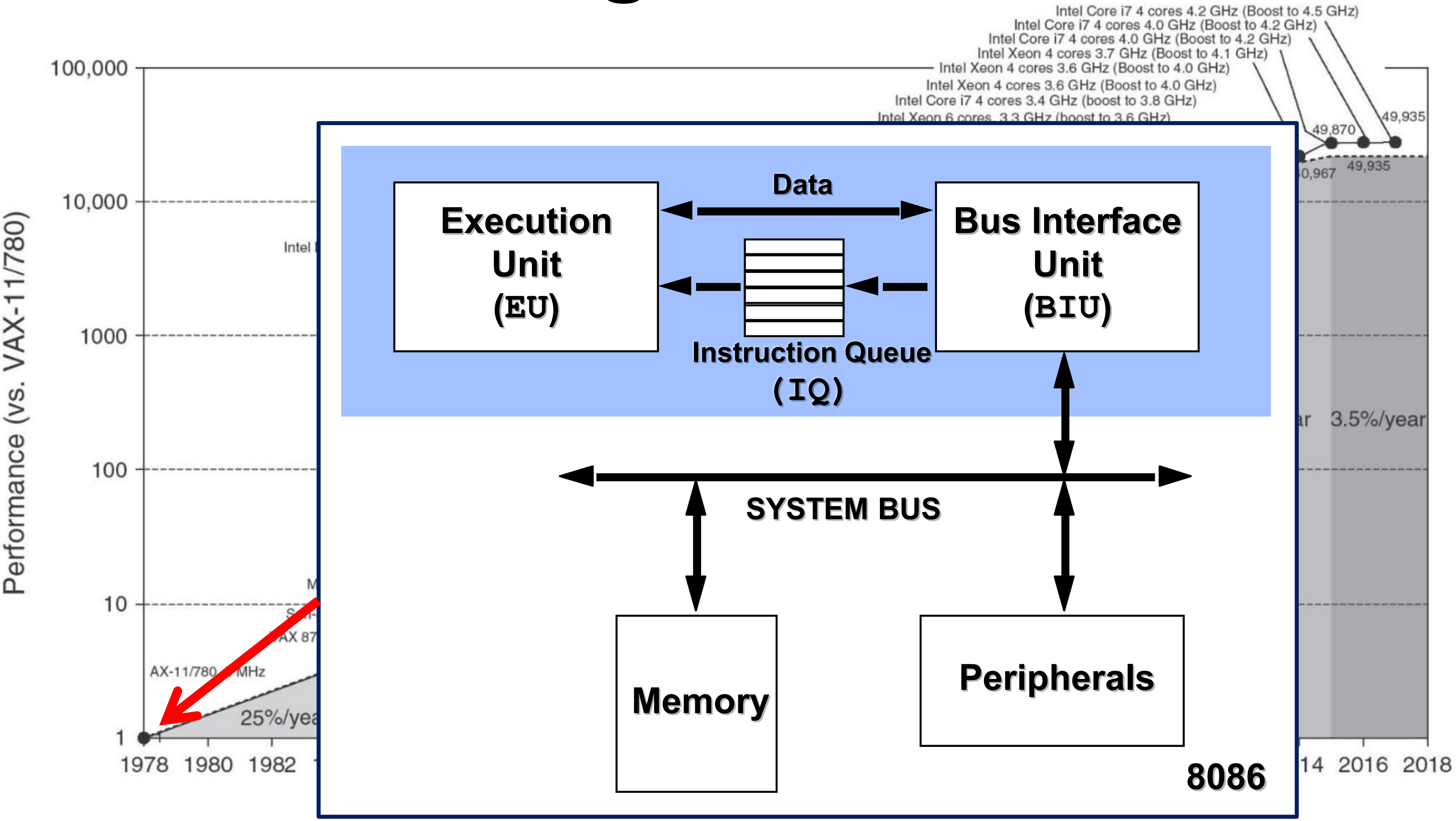
This incredible growth is due to improvements in

- **technology**
- **microprocessor architecture**
- **software development.**

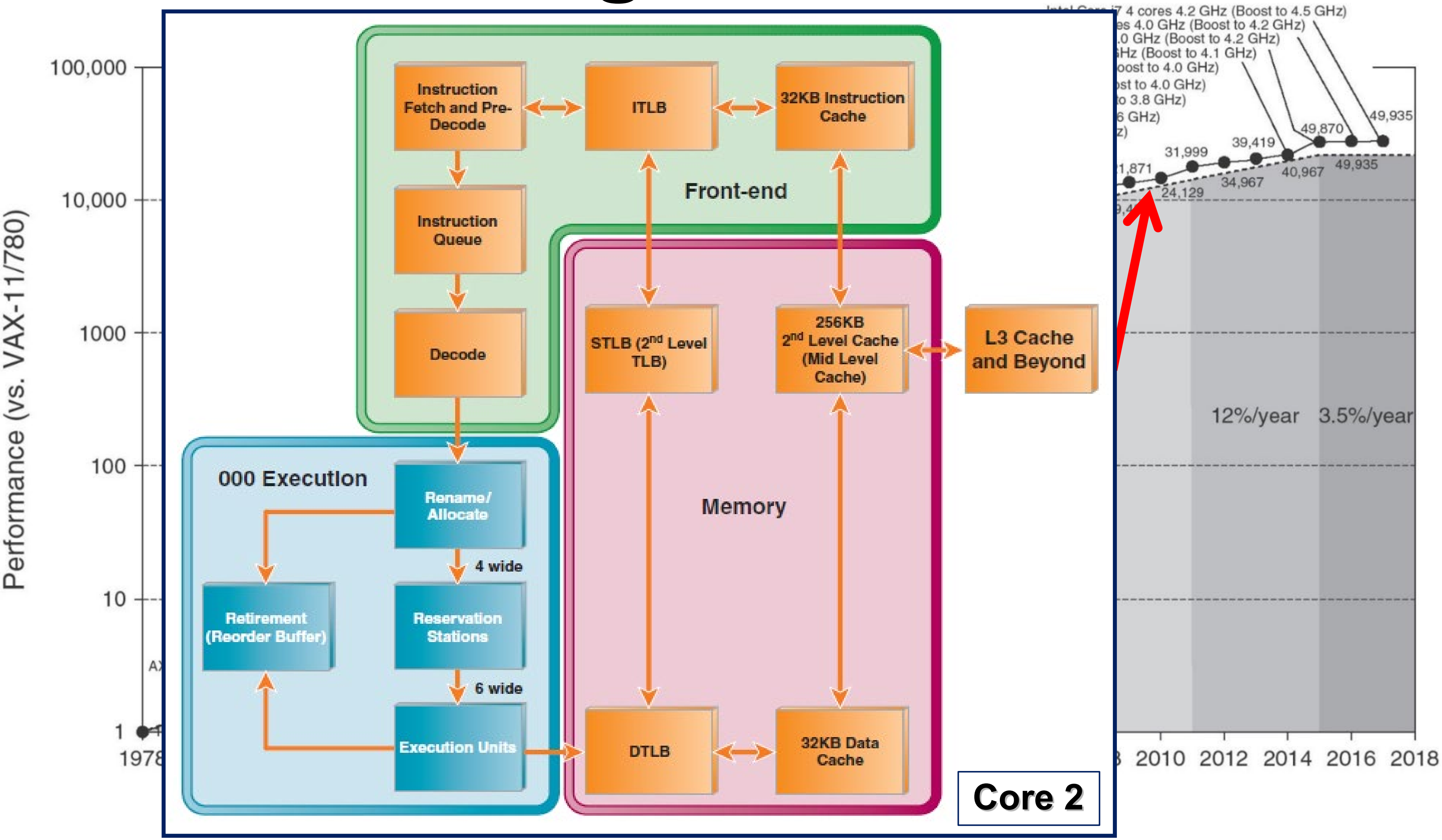
Microprocessor performance growth



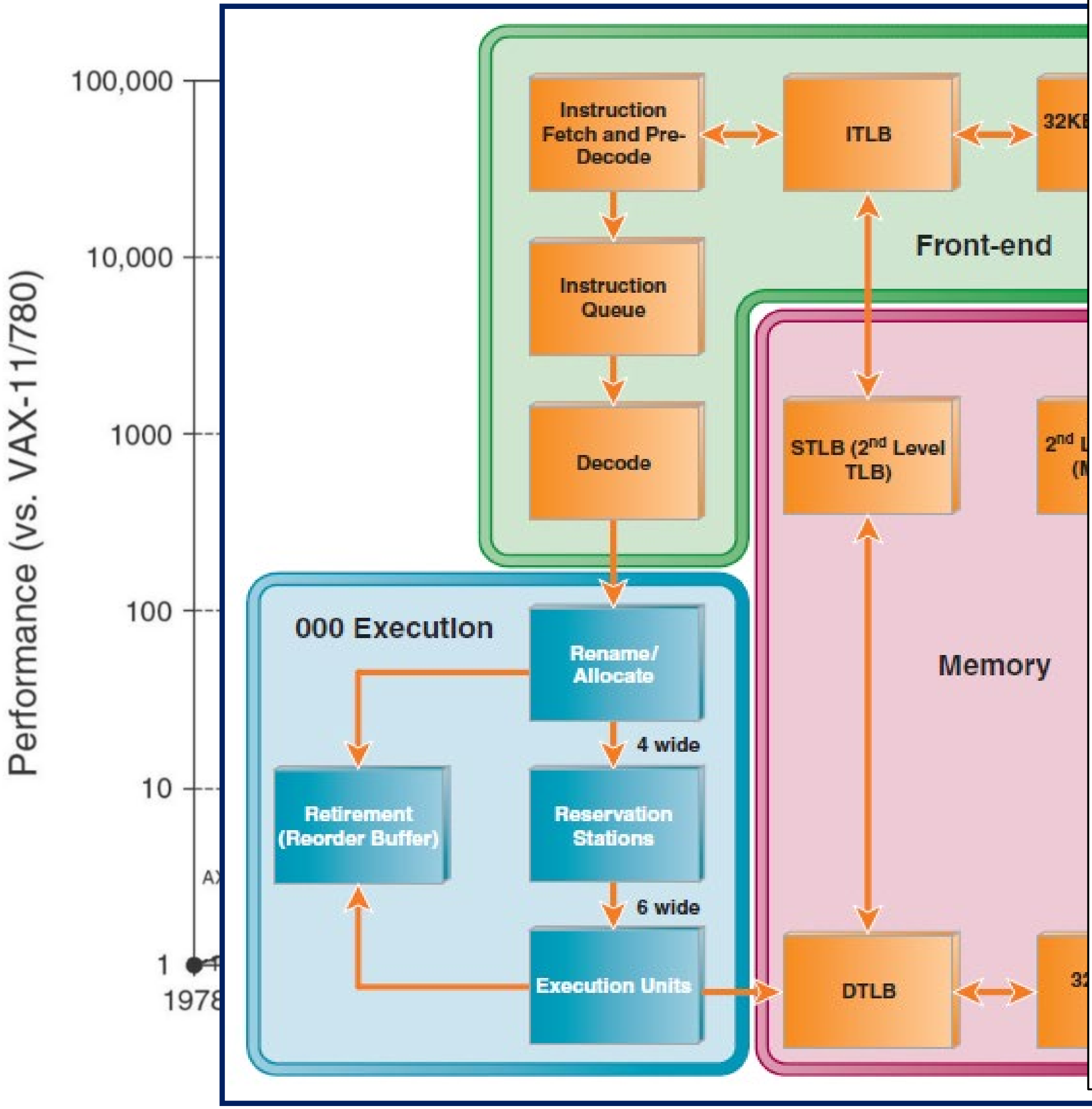
Microprocessor performance growth

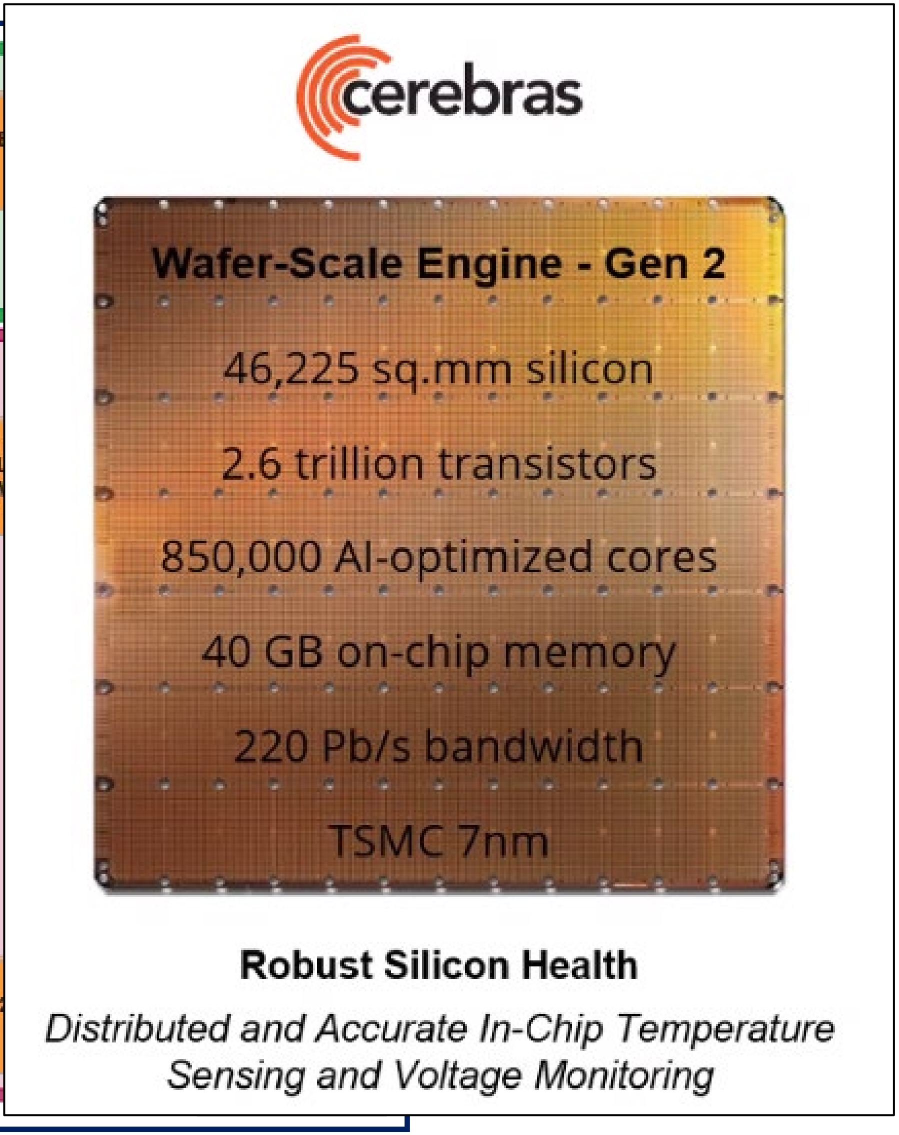

Intel Core i7 4 cores 4.2 GHz (Boost to 4.5 GHz)
Intel Core i7 4 cores 4.0 GHz (Boost to 4.2 GHz)
Intel Core i7 4 cores 4.0 GHz (Boost to 4.2 GHz)
Intel Xeon 4 cores 3.7 GHz (Boost to 4.1 GHz)
Intel Xeon 4 cores 3.6 GHz (Boost to 4.0 GHz)
Intel Xeon 4 cores 3.6 GHz (Boost to 4.0 GHz)
Intel Core i7 4 cores 3.4 GHz (boost to 3.8 GHz)
Intel Xeon 6 cores, 3.3 GHz (boost to 3.6 GHz)
49,935
49,870
0,967
49,935
100,000
10,000
1000
100
10
1
Performance (vs. VAX-11/780)
Intel
AX-11/780
MHz
VAX 87
25%/yea
3.5%/year
1978
1980
1982
14
2016
2018
Data
Execution Unit (EU)
Bus Interface Unit (BIU)
Instruction Queue (IQ)
SYSTEM BUS
Memory
Peripherals
8086

Microprocessor performance growth



Microprocessor performance growth





Wafer-Scale Engine - Gen 2

46,225 sq.mm silicon

2.6 trillion transistors

850,000 AI-optimized cores

40 GB on-chip memory

220 Pb/s bandwidth

TSMC 7nm

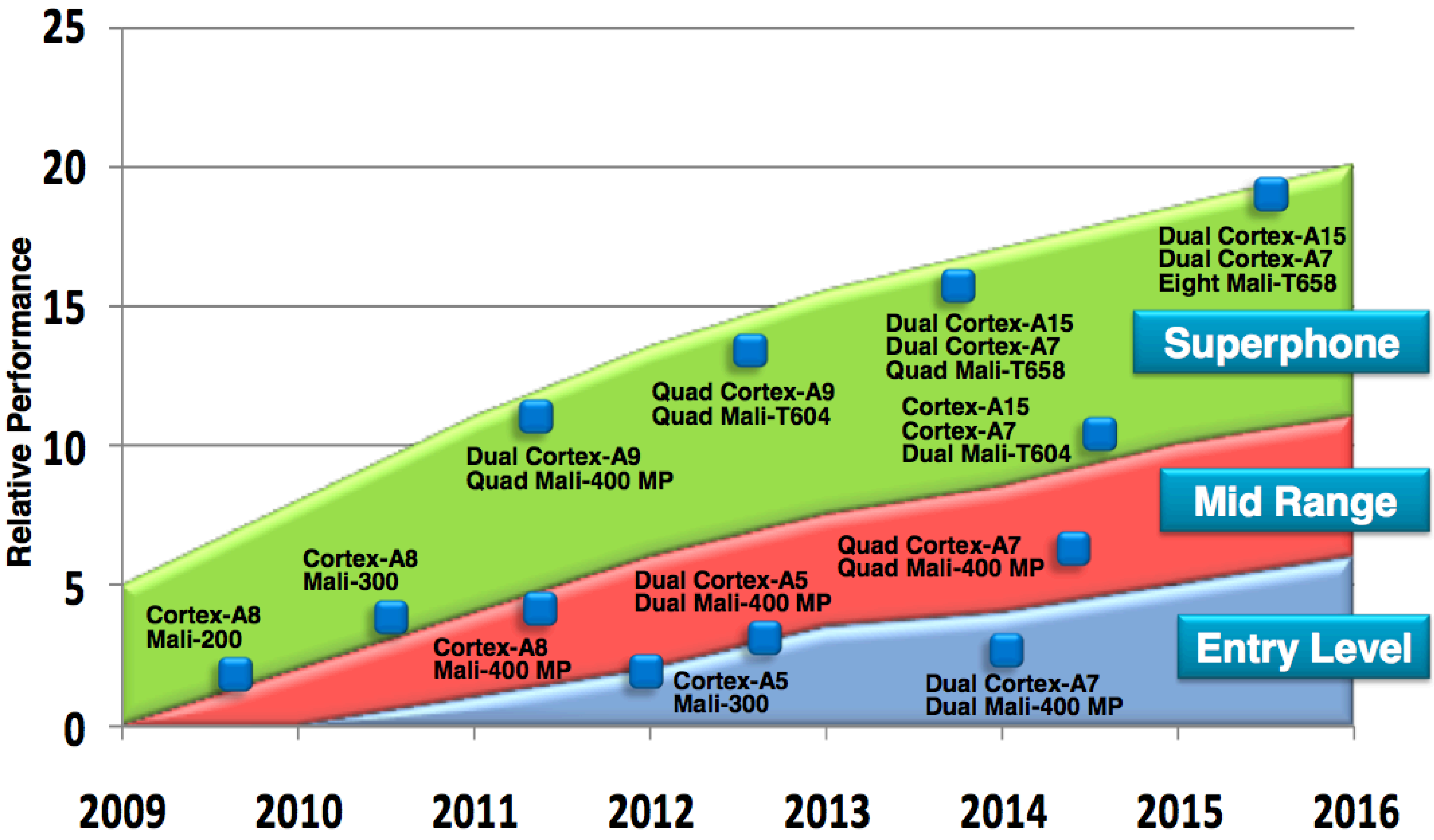
Robust Silicon Health

Distributed and Accurate In-Chip Temperature Sensing and Voltage Monitoring

Trend

The major players in the microprocessor market (Intel, AMD, IBM, ARM) are not investing anymore in the development of faster processors, but rather on multiprocessor single chip systems (i.e., multicore devices).

Scalable Mobile Processor Evolution



mali Bringing Visual Computing to Life

ARM

The computer market

It is currently split in 5 different areas:

- Personal Mobile Device (PMD)
- Desktop computing
- Servers
- Clusters / Warehouse Scale Computers (WSC)
- Embedded computers.

Personal Mobile Device (PMD)

This area includes smart phones, and tablet computers

Emphasis on energy efficiency and real-time applications.

System price: $100 - $1,000

Microprocessor price: $10 - $100

Desktop computing

This area covers from PCs to workstations.

The main target of this area is to optimize the *price-performance* ratio.

System price: $300 - $2,500

Microprocessor price: $50 - $500

Servers

These systems provide larger-scale and more reliable computing services.

The main parameters of products in this area are:

- Availability
- Scalability
- Throughput

System price: $5,000 - $10,000,000

Microprocessor price: $200 - $2,000

Clusters / Warehouse-Scale Computers (WAS)

“Software as a Service (SaaS)”

“Platform as a Service (PaaS)”

Emphasis on availability, price-performance, and power consumption

Supercomputers, emphasis: floating-point performance and fast internal networks

System price: $100,000 - $200,000,000

Microprocessor price: $50 - $250

Embedded computers

This area is the fastest growing portion of the computer market.

It covers all special-purpose computer-based applications (from microwaves to coffee machines, from automotive to videogames)

Adopted microprocessors vary from cheap low-end 8-bit processors to very efficient (and expensive) high-end processors, but are not able to run third-party software.

System price: $10 - $100,000

Microprocessor price: $0.01 - $100

Embedded computers

Special requirements often existing in this area are

- Real-time performance requirements
- Memory minimization
- Power consumption minimization
- Reliability constraints.

Embedded computers

Embedded problems are often solved resorting to one of the following solutions:

- Standard processor + custom logic + custom SW
- Standard processor + custom SW
- Standard DSP + custom SW.

Programmable devices (FPGAs) are playing a growing role.

Classes of Parallelism

- Data-level Parallelism (DLP):

Many data items that can be operated on at the same time

- Task-level Parallelism (TaskLP):

Different tasks of a work can operate independently.

Parallel Architectures

- Instruction-level Parallelism (ILP):

  Modestly exploits Data-level Parallelism
- Vector Architectures and Graphic processor unit (GPUs):

  Exploits Data-level Parallelism
- Thread-level Parallelism (TLP):

  Exploits Data-level Parallelism and Task-level Parallelism
- Request-level Parallelism (RLP):

  Exploits parallelism among decoupled tasks.

Designing a computer

It means

- determining which attributes are important for the new machine
- designing a machine which
  - maximizes performance and
  - matches cost and power constraints.

Computer architecture

In the last decades, computer design took advantage of both

- Architectural innovation
- Technology improvements.

It was estimated that the difference between the highest-performance microprocessors available in 2001 and what would have been obtained by relying solely on technology is more than a factor of 15.

Computer architecture

It includes three aspects of computer design:

- Instruction set architecture
- Organization
- Hardware.

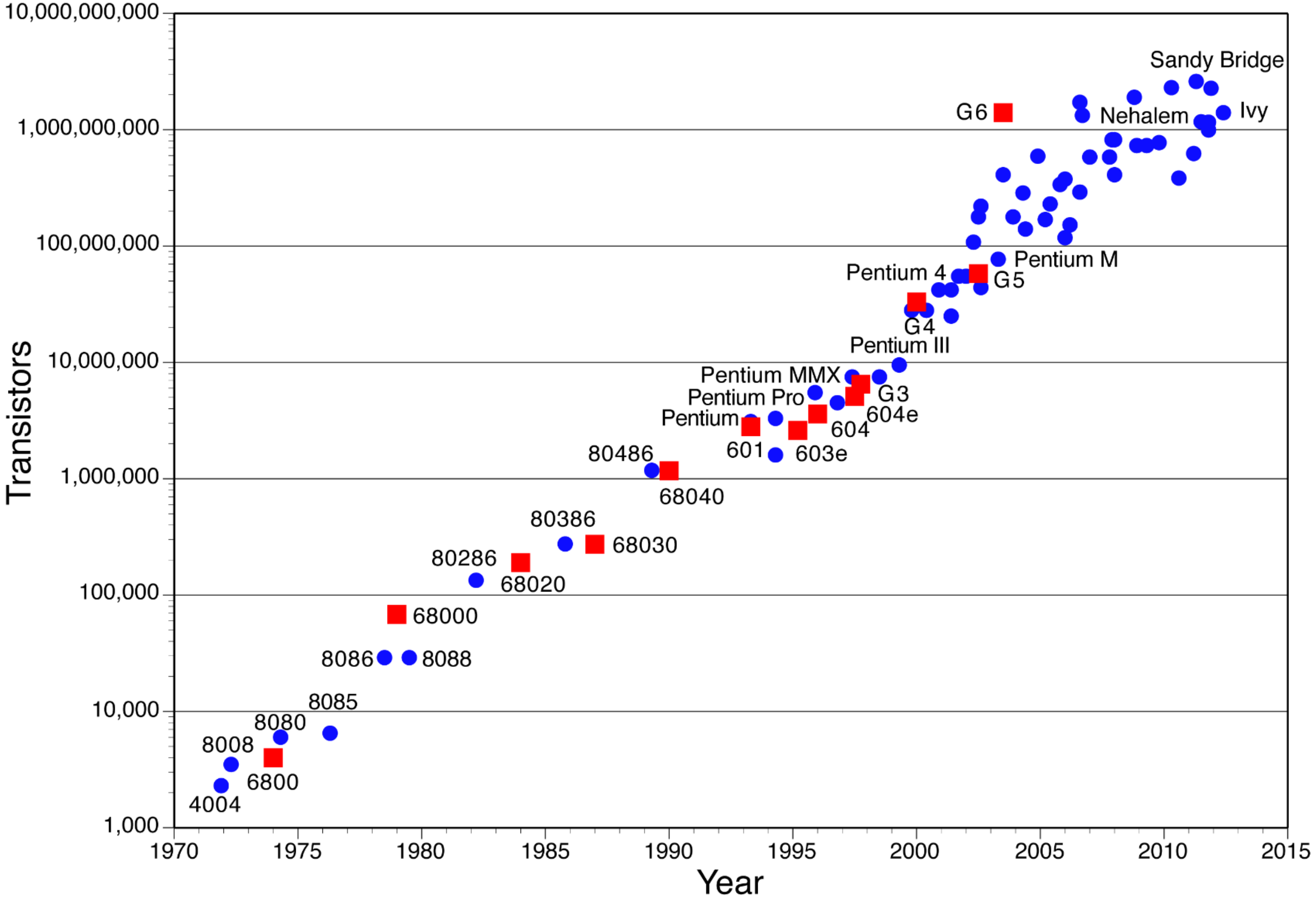
Computer architect must design a computer meeting:

- Functional requirements
- Price
- Power
- Performance
- Dependability.

Moore's Law

The number of devices (i.e., transistors) that can be integrated into a single chip doubles every 18/24 months.

Intel processors complexity growth



IC manufacturing cost

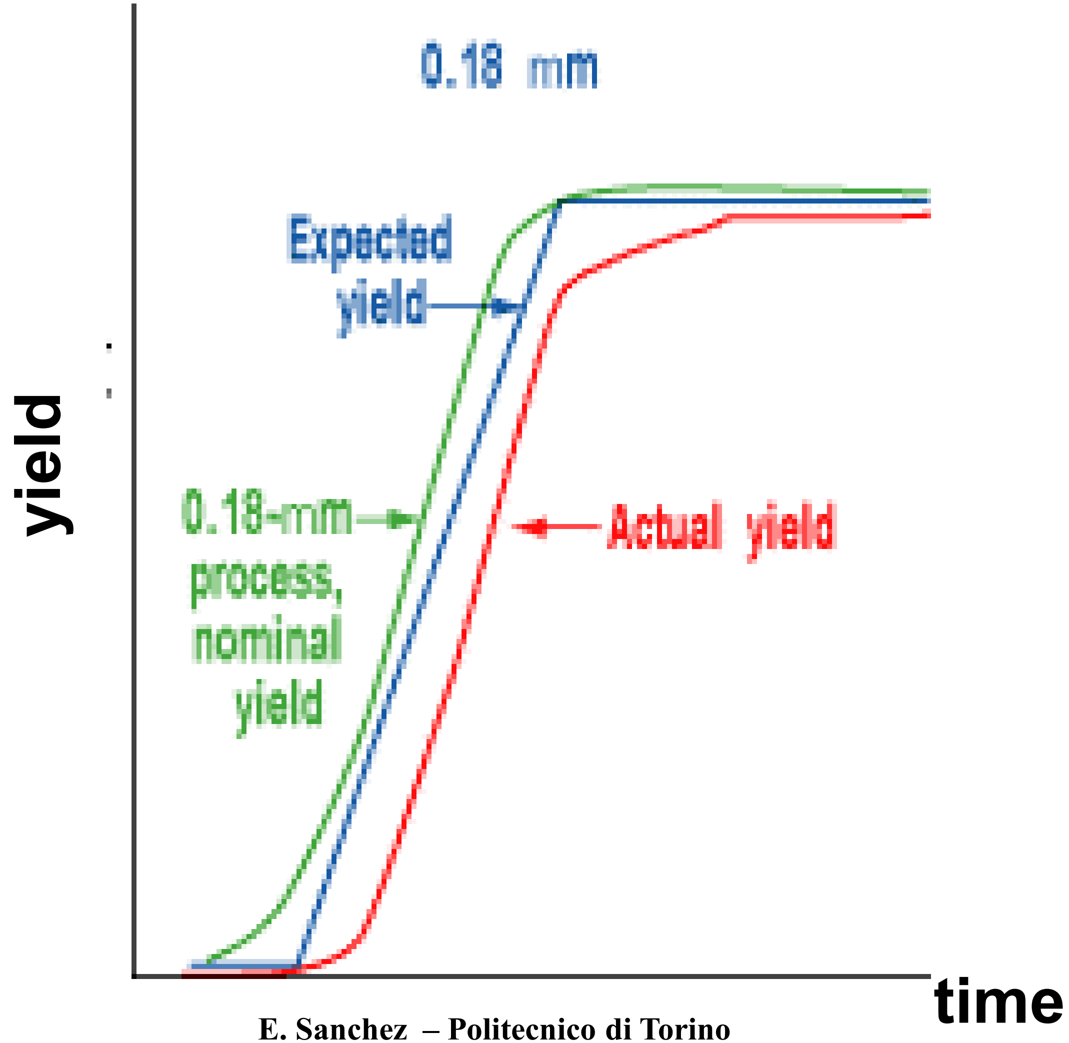
When evaluating it, it is important not to forget the impact of *yield*, i.e., the percentage of products that pass the test phase.

The production process for every product undergoes an evolution which normally leads to an improvement in yield (also known as *learning curve*).

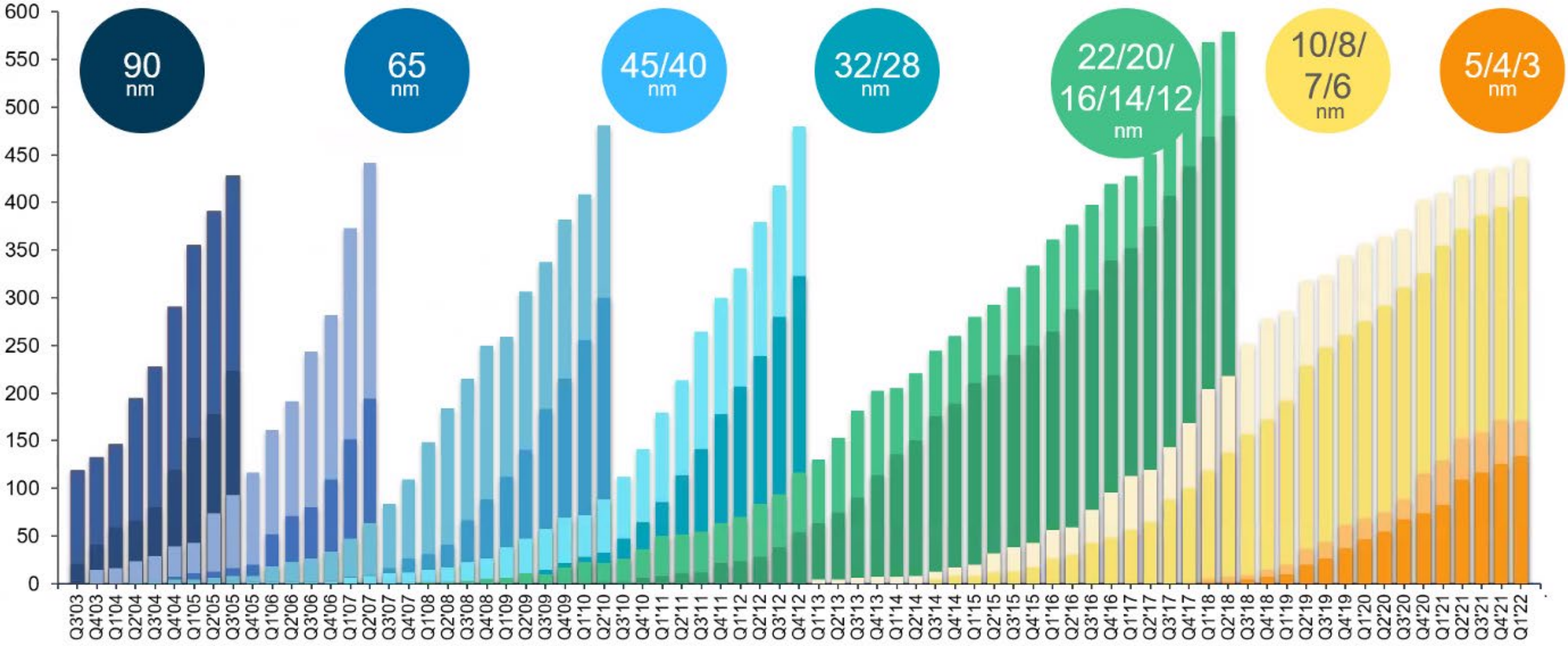
When yield increases, the cost decreases.

More than 50% of manufacturing cost is due to validation and testing procedures!!

Yield behavior



Industry adoption of new transistor technologies



Power consumption

Continuous increase in system complexity and device integration often lead to problems with power consumption.

Power consumption may be critical under two aspects:

- Power (static and dynamic)
- Energy (mainly for portable devices).

Power

Until now it has been dominated by dynamic power, i.e., that consumed by each transistor when switching between different states.

The dynamic power for each transistor can be evaluated by the formula

$\text{Power}_{\text{dynamic}} = \frac{1}{2} \times \text{capacitive load} \times \text{voltage}^2 \times \text{frequency}$

$\text{Power}_{\text{static}} = V \times I$ (25% of total power consumption)

For this reason, voltage continuously dropped in the last years.

Energy

It is given by

$$\text{Energy}_{\text{dynamic}} = \text{capacitive load} \times \text{voltage}^2$$

It is mainly of interest for mobile devices.

Dependability

Dependability is the quality of the system to deliver a correct service.

Dependability of computer systems is traditionally very high, but it can be lowered by

- Bugs in the design of the hardware
- Bugs in the software
- Defects in the hardware (introduced by the manufacturing process)
- Faults happening during the product operation.

Safety-critical applications

In the past, possible misbehaviors of computer systems were critical in some areas, such as

- **Space**
- **Avionics**
- **Nuclear plants control.**

In more recent years, computer-based systems expanded to other safety-critical areas, such as

- **Rail-road traffic control**
- **Automotive**
- **Biomedical**
- **Telecommunications.**

Dependability importance

It increased very much, because in several areas it is crucial to guarantee that the system matches the dependability constraints, e.g., in terms of probability of behaving as expected for long periods.

Dependability evaluation

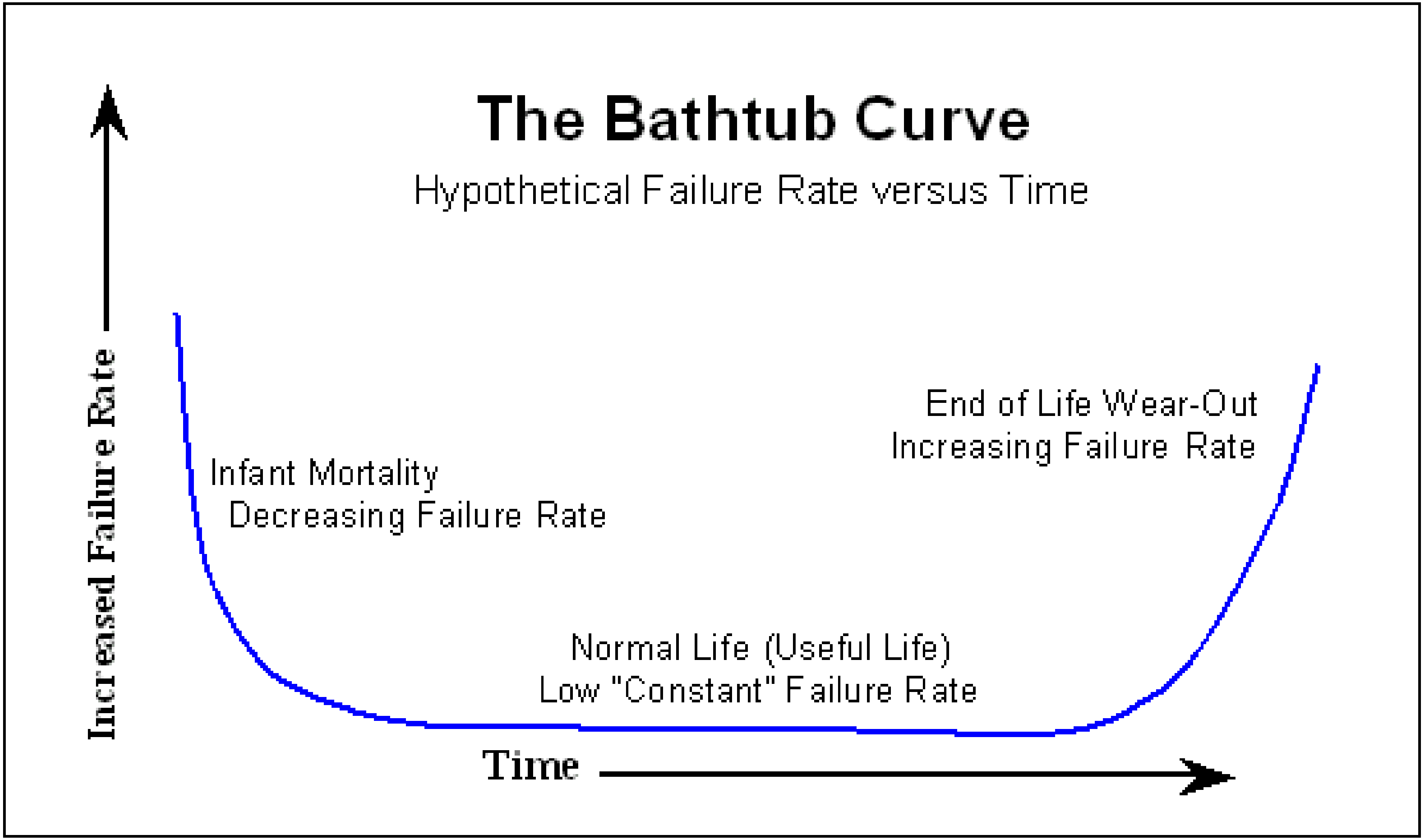
Dependability is often measured using

- Mean Time To Failure (MTTF) or Failures In Time (FIT), which is its reciprocal. 1 FIT = 1 failure in one billion hours
- Mean Time Between Failures (MTBF)
- Mean Time To Repair (MTTR)

The three measures are related by the formula

$$\text{MTBF} = \text{MTTF} + \text{MTTR}$$

Finally, availability is the probability that a system works correctly in a generic time instant.


The Bathtub Curve
Hypothetical Failure Rate versus Time
Increased Failure Rate
Infant Mortality
Decreasing Failure Rate
Normal Life (Useful Life)
Low "Constant" Failure Rate
End of Life Wear-Out
Increasing Failure Rate
Time

Computer performance

What is performance?

<u>User</u> point of view:

performance = *response time* (time between start and completion of an operation)

<u>System manager</u> point of view:

performance = *throughput* (total amount of work done in the time unit).

Time

Which time has to be considered for performance computation?

- **Elapsed time**
- **CPU time**
  - **user CPU time**
  - **system CPU time.**

All these measures can be of interest. UNIX provides all of them through the `time` command

```
90.7s  12.9s  2:39  65%
```

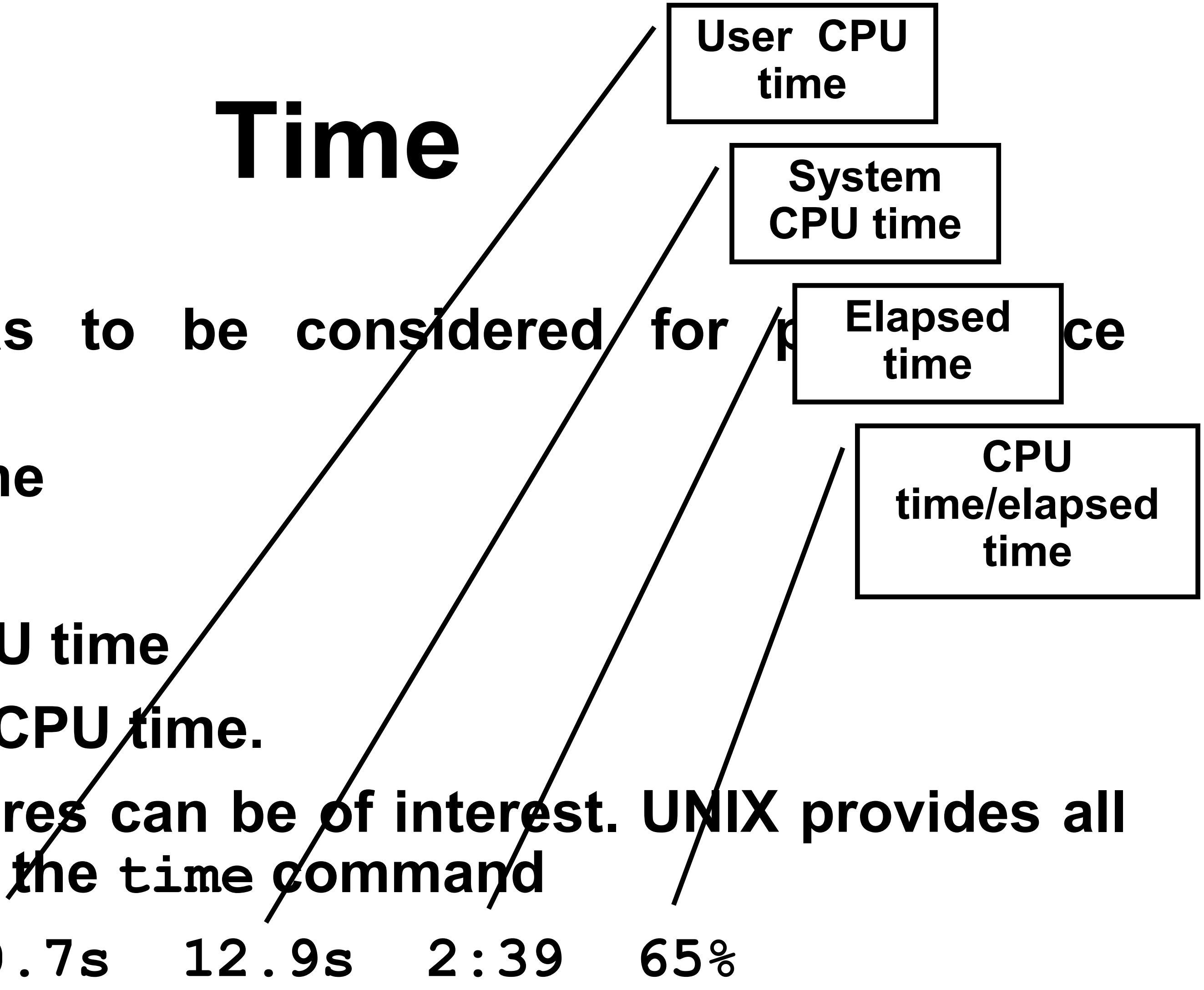
Time

Which time has to be considered for p ce computation?

- **Elapsed time**
- **CPU time**
  - **user CPU time**
  - **system CPU time.**

All these measures can be of interest. UNIX provides all of them through the `time` command

```
90.7s  12.9s  2:39  65%
```



Performance evaluation

It is often performed by letting the computer to execute applications and observing its behavior.

Unfortunately, the choice of the applications severely affects the performance.

In the ideal case, one should use as *workload* the mix of applications the user will run.

However, they are normally unknown, and largely variable from one user to another.

Therefore, some *benchmarks* are selected to mimic real cases.

Program benchmarks

Possible benchmarks:

- *Real programs* (e.g., C compilers, text processors, special-purpose tools), possibly modified
- *Kernels* (e.g., Livermore Loops, Linpack)
- *Toy benchmarks* (e.g., Quicksort, Sieve of Eratostenes)
- *Synthetic benchmarks* (e.g., Whetstone, Dhrystone).

Benchmark suites

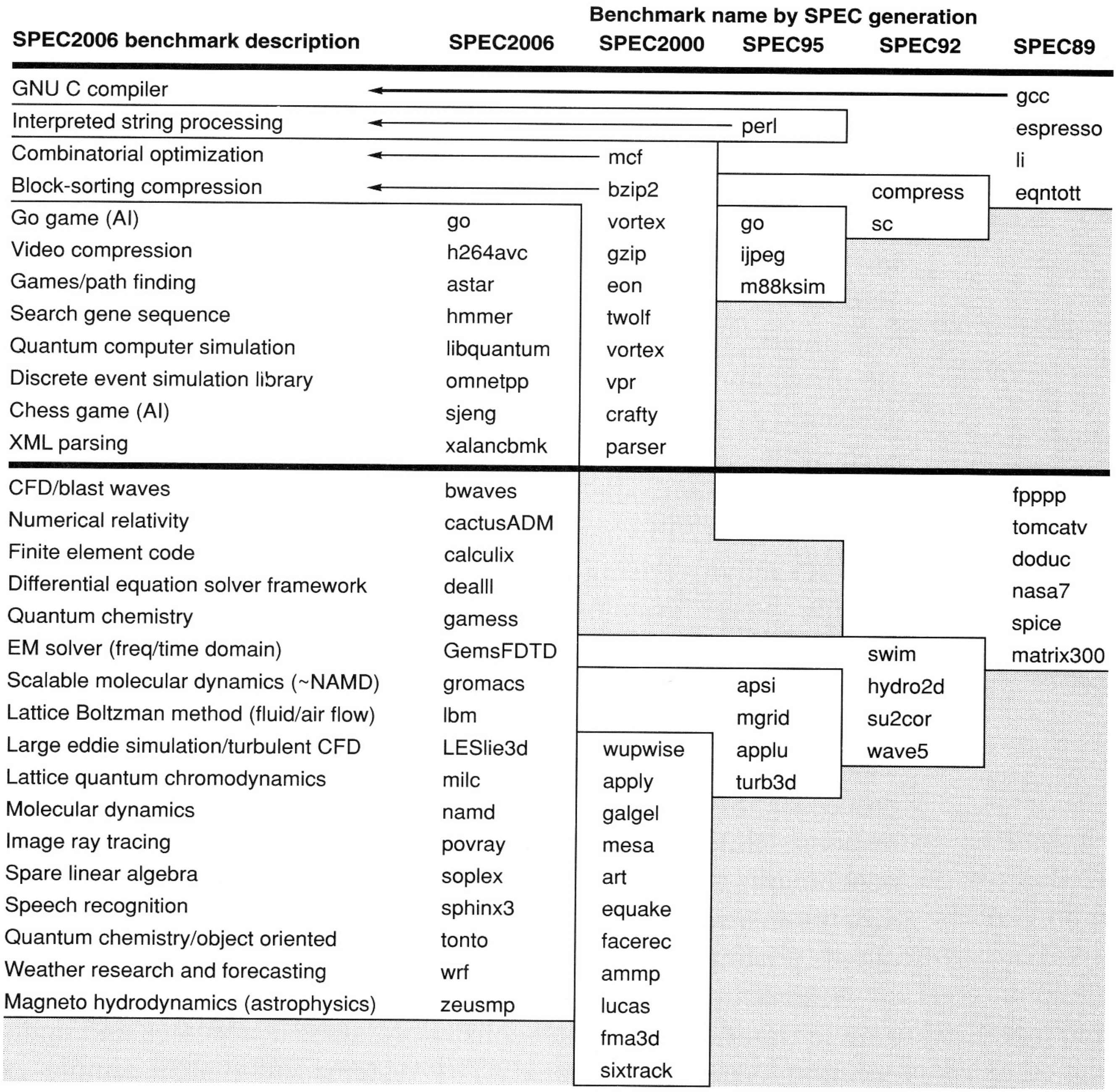
They contain a number of different programs, so that the weakness of any component is lessened by the presence of the others.

Benchmark sets are normally composed of:

- kernels
- program fragments
- applications.

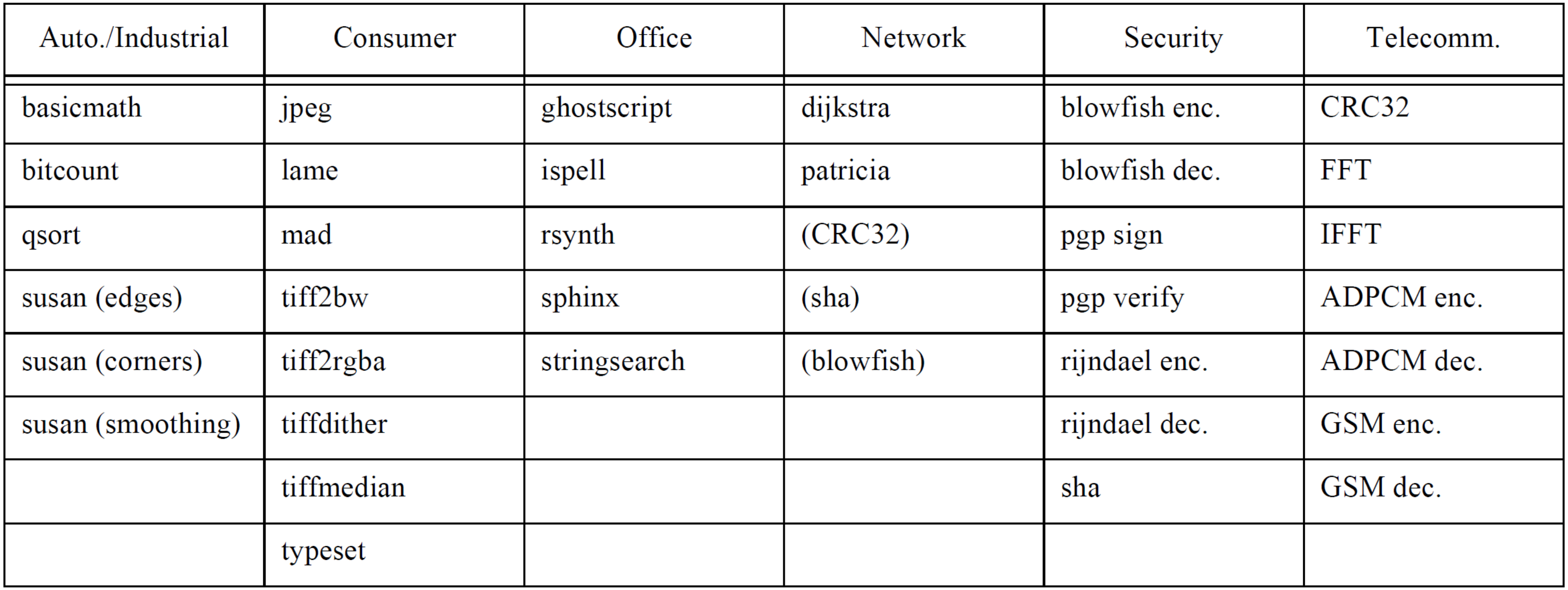
SPEC evolution

Standard
Performance
Evaluation
Corporation



	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			

MiBench Benchmarks



Auto./Industrial	Consumer	Office	Network	Security	Telecomm.
basicmath	jpeg	ghostscript	dijkstra	blowfish enc.	CRC32
bitcount	lame	ispell	patricia	blowfish dec.	FFT
qsort	mad	rsynth	(CRC32)	pgp sign	IFFT
susan (edges)	tiff2bw	sphinx	(sha)	pgp verify	ADPCM enc.
susan (corners)	tiff2rgba	stringsearch	(blowfish)	rijndael enc.	ADPCM dec.
susan (smoothing)	tiffdither			rijndael dec.	GSM enc.
	tiffmedian			sha	GSM dec.
	typeset				

Reproducibility

Information about execution times on benchmarks should allow reproducibility.

This means reporting detailed information about

- **hardware (system configuration)**
- **software (OS, compiler, program)**
- **program input.**

Comparing and summarizing performance

Problem 1

I know the performance of one machine on a set of programs: which is its global performance?

Problem 2

I know the performance of two machines on the same set of programs: which is their relative performance?

A number of metrics have been proposed.

Index i runs over all benchmarks in the set

Total execution time

$$\sum_{i=1}^{n} \text{Time}_i$$

Normalized execution time

A reference machine is adopted (e.g., VAX-11/780) and execution times are normalized with respect to it.

Arithmetic Mean

Arithmetic mean:

$$\frac{1}{\mathrm{n}}\sum_{\mathrm{i}=1}^{\mathrm{n}}\mathrm{Time}_{\mathrm{i}}$$

Weighted mean

Weighted arithmetic mean:

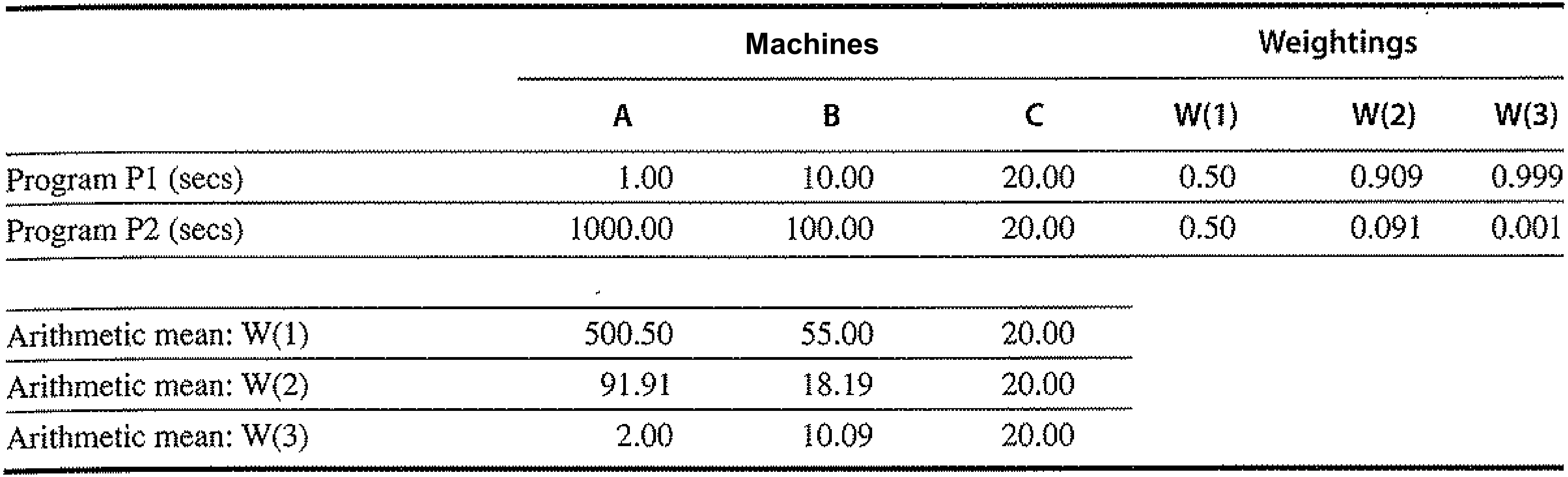
$$\sum_{i=1}^{n} \mathrm{Weight}_i \times \mathrm{Time}_i$$

Suggested solution

To measure a real workload and weight the programs according to their frequency of execution.

Program inputs should be carefully specified.

Example of weighted arithmetic mean execution time



	Machines			Weightings		
	A	B	C	W(1)	W(2)	W(3)
Program P1 (secs)	1.00	10.00	20.00	0.50	0.909	0.999
Program P2 (secs)	1000.00	100.00	20.00	0.50	0.091	0.001
Arithmetic mean: W(1)	500.50	55.00	20.00			
Arithmetic mean: W(2)	91.91	18.19	20.00			
Arithmetic mean: W(3)	2.00	10.09	20.00			

Guidelines and Principles for Computer Design

- Amdahl’s law
- CPU performance equation.

Amdahl’s Law: preliminaries

$$\text{speedup} = \frac{\text{performance with enhancement}}{\text{performance without enhancement}}$$

The speedup resulting from an enhancement depends on two factors:

- $\text{fraction}_{\text{enhanced}}$: the fraction of the computation time that takes advantage of the enhancement
- $\text{speedup}_{\text{enhanced}}$: the size of the enhancement on the parts it affects.

Amdahl's Law

$$\text{execution time}_{\text{new}} =$$

$$\text{execution time}_{\text{old}} \times ((1 - \text{fraction}_{\text{enhanced}}) + \frac{\text{fraction}_{\text{enhanced}}}{\text{speedup}_{\text{enhanced}}})$$

$$\text{speedup}_{\text{overall}} = \frac{\text{execution time}_{\text{old}}}{\text{execution time}_{\text{new}}} =$$

$$\frac{1}{(1 - \text{fraction}_{\text{enhanced}}) + \frac{\text{fraction}_{\text{enhanced}}}{\text{speedup}_{\text{enhanced}}}}$$

Example

An enhancement makes one machine 10 times faster for 40% of the programs the machine runs. Which is the overall speedup?

$\textbf{fraction}_{\textbf{enhanced}} = \textbf{0.4}$

$\textbf{speedup}_{\textbf{enhanced}} = \textbf{10}$

$$\text{speedup}_{\text{overall}} = \frac{1}{(1-0.4)+\dfrac{0.4}{10}} = 1.56$$

Choosing between two solutions: an example

Two solutions are available for increasing the floating-point performance of one machine.

Solution 1

Increasing by 10 the performance of square root operations (responsible for 20% of the execution time) by adding specialized hardware.

Solution 2

Increasing by 2 the performance of all the floating-point operations (responsible for 50% of the execution time).

Which solution makes the machine faster?

Amdahl's Law application

Solution 1

$$\mathrm{speedup}_1 = \frac{1}{(1-0.2)+\frac{0.2}{10}} = 1.22$$

Solution 2

$$\mathrm{speedup}_2 = \frac{1}{(1-0.5)+\frac{0.5}{2}} = 1.33$$

Measuring the time required to execute a program

Possible approaches:

- by observing the real system
- by simulation
- by applying the CPU performance equation.

The CPU Performance Equation

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

where

- ***CPI_i*** **is the number of clock cycles required by instruction i**
- ***IC_i*** **is the number of times instruction i is executed in the program**
- ***clock cycle time*** **is the inverse of clock frequency.**

The CPU Performance Equation

Depends on the hardware organization and instruction set architecture

ock cycle time

where

- ***CPI$_i$*** **is the number of clock cycles required by instruction i**
- ***IC$_i$*** **is the number of times instruction i is executed in the program**
- ***clock cycle time*** **is the inverse of clock frequency.**

The CPU Performance Equation

n

CPU ti ime

Depends on the instruction set architecture and compiler technology

where

- ***CPI$_i$*** **is ed by instructio**
- ***IC$_i$*** **is the number of times instruction i is executed in the program**
- ***clock cycle time*** **is the inverse of clock frequency.**

The CPU Performance Equation

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

where

- ***CPI_i* is the by instruction**
- ***IC_i* is the nu ted in the program**
- ***clock cycle time* is the inverse of clock frequency.**

Depends on the hardware technology and organization

The CPU Performance Equation limitations

In pipelined processors, CPI_i may vary for a given instruction, depending on different parameters

- Instructions executed before and after
- Memory system behavior (e.g., cache miss or hit)

Therefore, evaluating the execution time analytically becomes much harder.