董剑

容错与移动计算研究中心

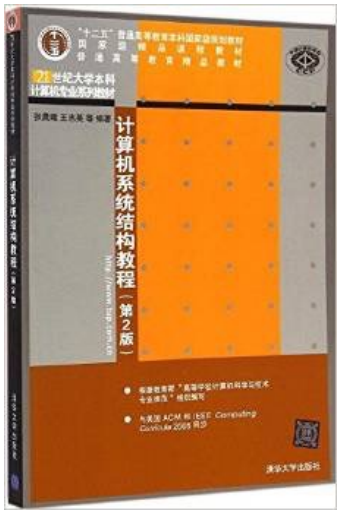
2021年秋

计算机系统结构

联系方式

1. 电话：13936244393
2. Mail：dan@hit.edu.cn
3. 办公地点：综合楼508（一校区，哈尔滨）

参考教材



《计算机系统结构教程》第2版

张晨曦

清华大学出版社

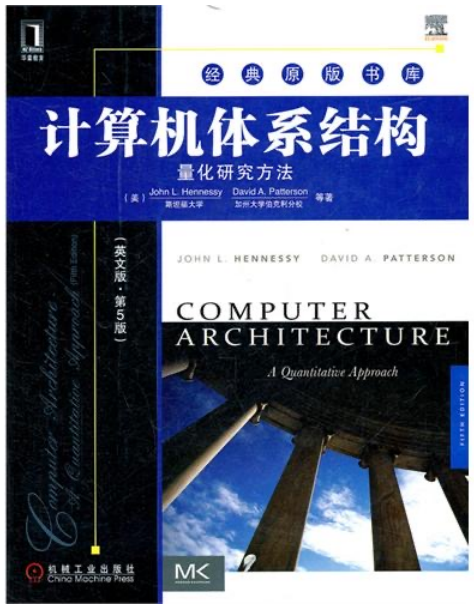
参考教材 — 推荐阅读

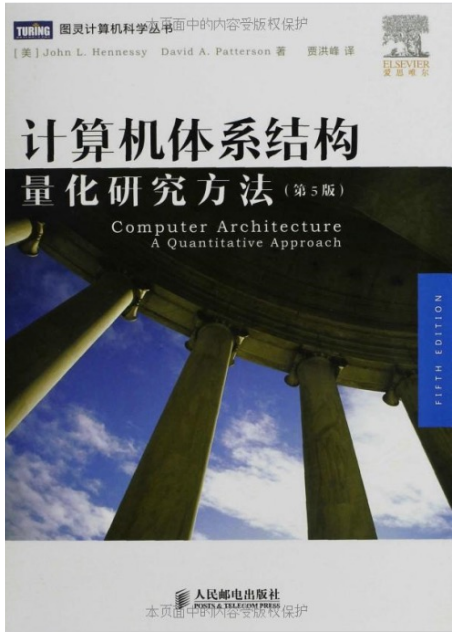
《计算机系统结构-量化的研究方法》

John L. Hennessy

David A. Patterson

第五版





“他们为计算机体系结构的设计和评估开创了一个系统的、量化的方法，并对微处理器行业产生了持久影响”

David A.Patterson 加州大学伯克利分校计算机科学系教授，Patterson 因为对RISC技术的贡献荣获1995年的IEEE技术成就奖，他领导并设计了第一台RISC计算机RISC I而在RAID技术方面的成就就为他赢得了1999年的IEEE Reynold B.Johnson信息存储奖。2000年他和John L.Hennessy 被IEEE授予John von Neumann奖，以表彰他们“通过对系统结构方面的创新进行不懈的探索，推广和商业化，掀起了一波计算机系统结构的革新浪潮”。Patterson 也是美国国家工程研究院院士，IEEE和ACM会士。

John L.Hennessy，斯坦福大学校长，1977年开始在斯坦福大学电子工程系和计算机系任教。 他是IEEE和ACM会士，美国国家工程院院士及美国科学与艺术院院士。由于其在RISC技术领域的杰出贡献，于2001年被授予Eckert-Mauchly奖；他获得的其他奖项还包括2001年度Seymour cray计算机工程奖以及2000年度同David Patterson共同获得的John von Neumann奖。同时他还获得了荣誉博士学位。

1981年，Hennessy带领几个研究生开始其在斯坦福大学的MIPS项目。在1984年完成该项目之后，他从学校离开了1年时间去开发一个MIPS计算机系统，在开发该系统的过程中研制出了世界上第一个商用的RISC微处理器。MIPS Technologies公司从1991年被SGI公司收购，后来在1998年脱离出来成为一个独立的公司，并将其战略重点转向嵌入式微处理器。截至2006年，已经有超过5亿个MIPS微处理器被用于视频游戏、掌上电脑、激光打印机和网络交换机等设备中。

计算机科学

- Computer Architecture
- Computer Software& Theory
- Computer Application

SYSTEM

Computer Architecture is NOT a hardware course,

BUT SYSTEM!

1. Software
2. IO System
3. Memory system
4. CPU

All about how to improve computer system's performance and reduce its cost!

How to do that?

We will study that since TODAY

1.为什么要研究系统结构？

2.计算机系统结构研究什么？

为什么要研究系统结构？

半个多世纪以来，计算机技术取得了惊人的发展。1945年时还没有能存储程序的计算机。现在，花不到一千美元买到的个人计算机比1980年花一百万美元买的计算机具有更高的性能、更大的主存和磁盘空间。这一高速发展既得益于计算机制造技术的进步，又离不开计算机设计的创新。

——《计算机系统结构-量化的研究方法》

经典比喻

1. 从20世纪40年代末开始，如果交通运输业保持计算机产业的发展速度，今天我们就可以花50美分，在5秒内穿越大洋。（D. A. Patterson & J. L. Hennessy）

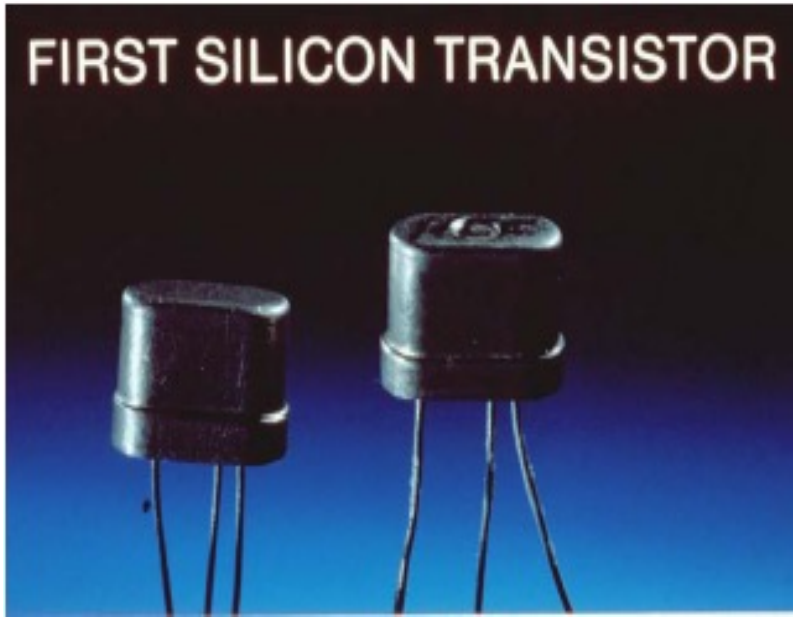
2. 如果汽车的性能价格比以同样的速度提升，那么一辆“劳斯莱斯”现在只值1美元，每消耗4.5升汽油，就能行驶16亿公里。（A. S. Tanenbaum, M. van Steen）

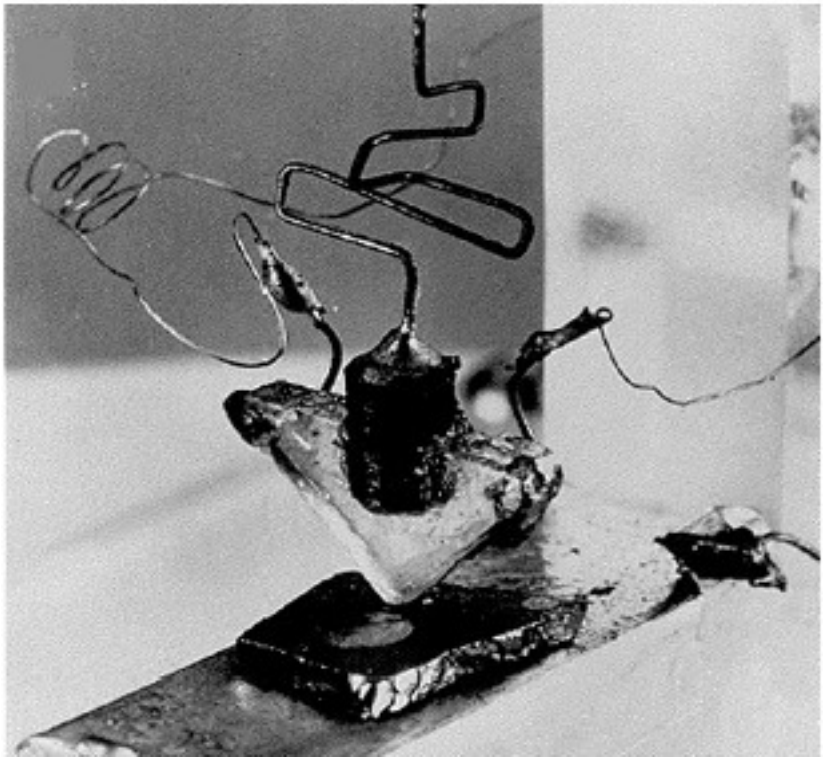
计算机技术快速进步的原因(1)

- 电子技术进步－－集成电路技术的进步，还有存储器（包括内外存）和各类外设的进步。

电子原件的一大突破—晶体管

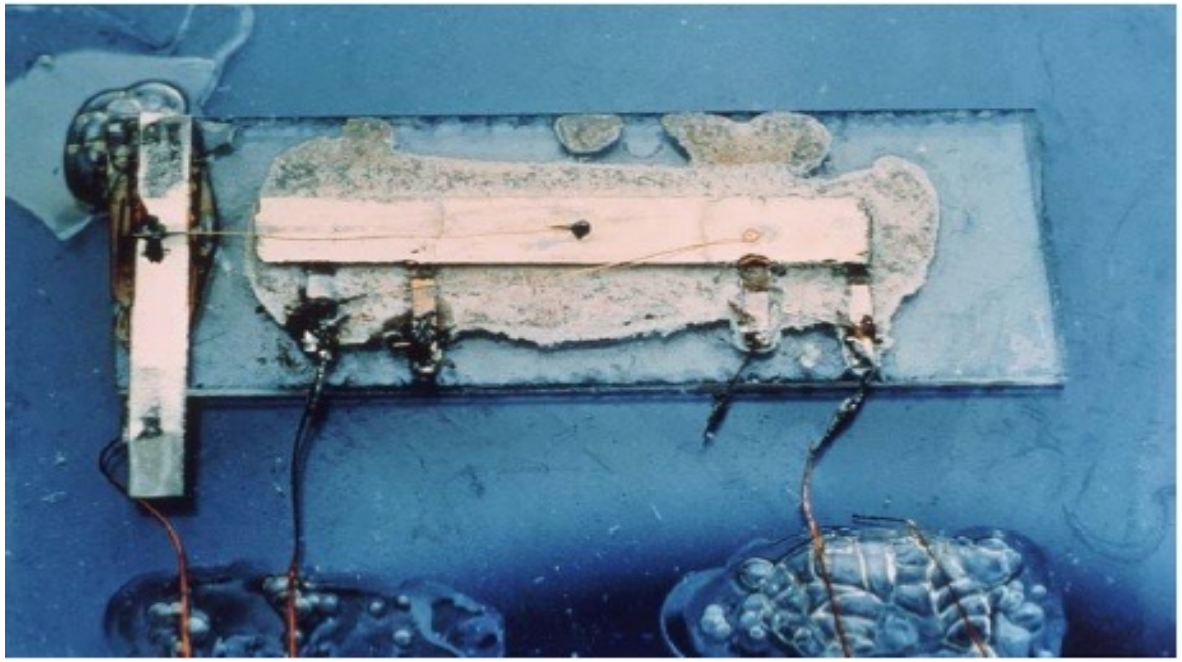
- **By W. Shockley, J. Bardeen, W. Brattain of Bell Lab. In 1947**
  - **Much more reliable than vacuum tubes**
  - **Much smaller than vacuum tubes**





计算机元件的另一大突破—IC

- **1958年德州儀器公司的Jack Kilby: integrated a transistor with resistors and capacitors on a single semiconductor chip, which is a monolithic IC.**



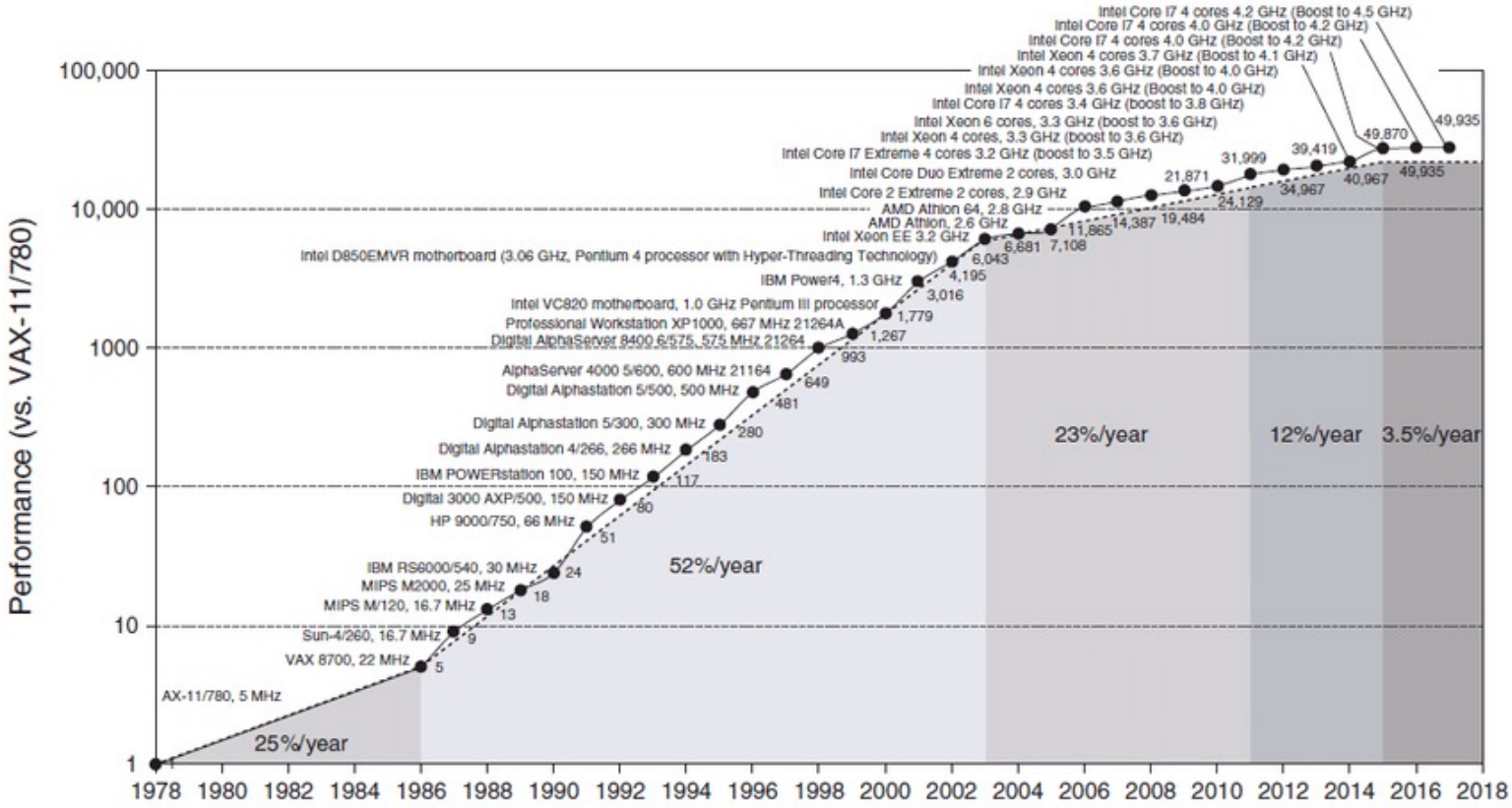
分代	器件特征	结构特征	软件特征	典型实例
第一代 （1945—1954年）	电子管和继电器	存储程序计算机 程序控制I/O	机器语言 汇编语言	普林斯顿ISA， ENIAC，IBM 701
第二代 （1955—1964年）	晶体管、磁芯 印刷电路	浮点数据表示 寻址技术 中断、I/O处理机	高级语言和编译 批处理监控系统	Univac LAPC， CDC 1604， IBM 7030
第三代 （1965—1974年）	SSI和MSI 多层印刷电路 微程序	流水线、Cache 先行处理 系列机	多道程序 分时操作系统	IBM 360/370， CDC 6600/7600， DEC PDP-8
第四代 （1975—1990年）	LSI和VLSI 半导体存储器	向量处理 分布式存储器	并行与分布处理	Cray-1，IBM 3090，DEC VAX 9000，Convax-1
第五代 （1991年—）	高性能微处理器 高密度电路	超标量、超流水 SMP、MP、MPP 机群	大规模、可扩展 并行与分布处理	SGI Cray T3E， IBM SP2，DEC AlphaServer 8400

计算机技术快速进步的原因(2)

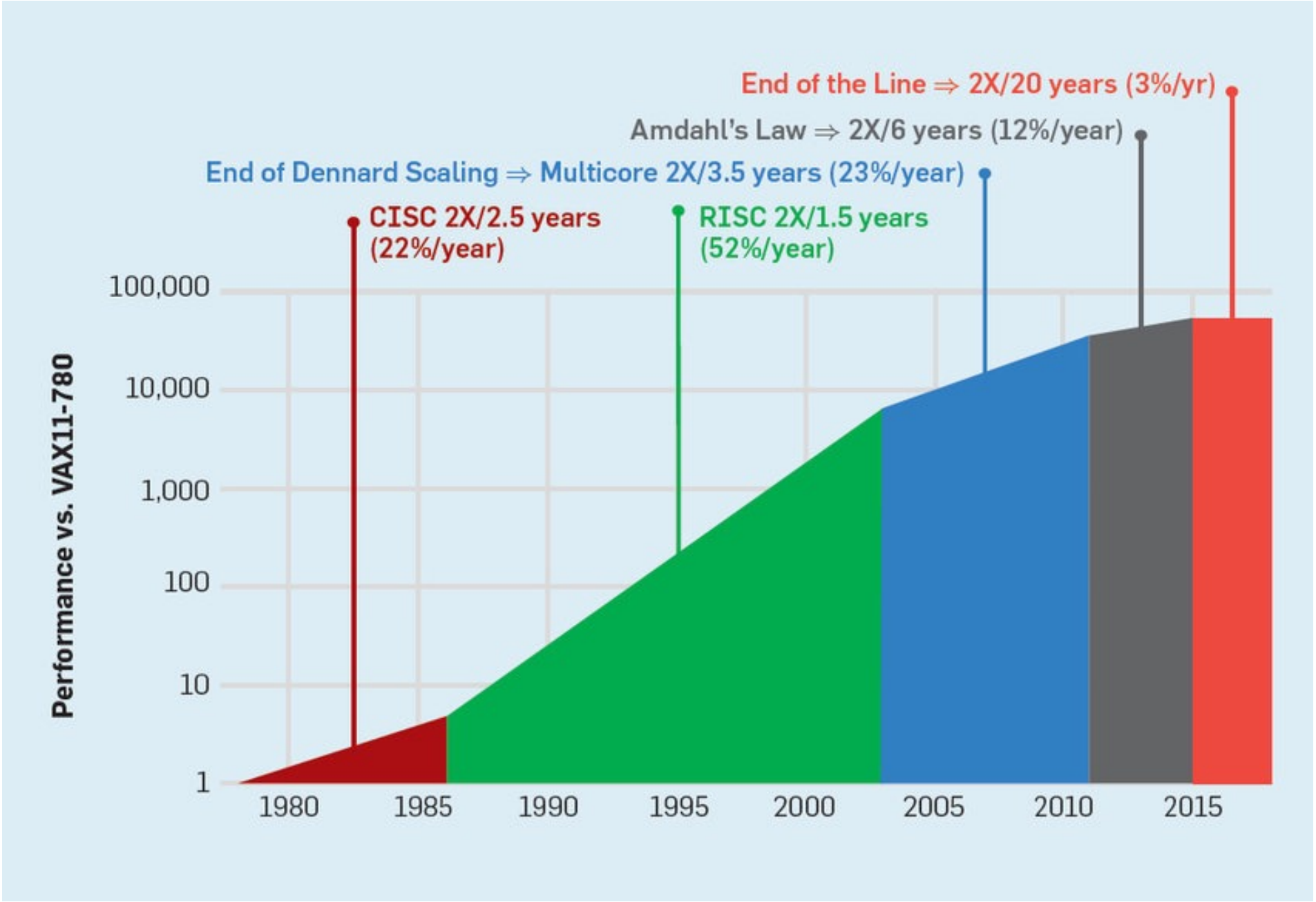
- **计算机技术快速进步的原因二**
  - **计算机系统结构的不断创新**
  - 相对制造技术的稳步发展，系统结构的发展总是相对滞后
  - 在电子计算机发展的最初25年中，这两股力量的贡献都很大

- 大约从七十年代开始，计算机设计者开始更多地依赖于集成电路技术
  - 当时计算机工业占统治地位的大型机和小型机的性能以每年**25—30%**的速度提高
- 七十年代末出现了微处理器，它比大型机和小型机集成度更高，促进了集成电路技术的发展，计算机性能以大约每年35%的速度提高
- 35%的发展速度，再加上微处理器批量生产的成本优势，使得计算机产业中以微处理器为基础的部分迅速膨胀

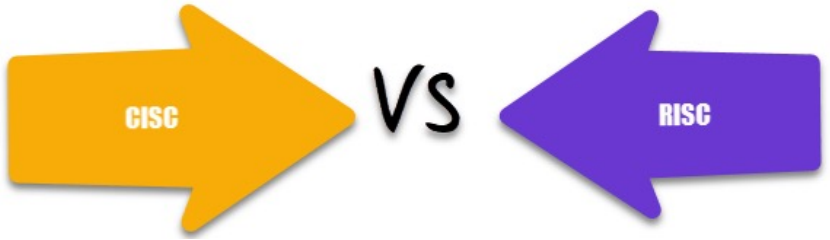
- 两个重大的市场变化极大推动了系统结构的发展
  - 人们实际已经极少使用汇编语言编程，这就降低了对目标代码兼容性的要求
  - 标准的、与厂商无关的操作系统（如UNIX）的出现，减小了推出新系统结构的成本和风险
- 计算机市场的两个重大变化使新的计算机系统结构比以往更容易取得商业运作的成功



- 长达16年的飞速发展已经告一段落
  - 2002年起，速度已经降为22%左右


End of the Line ⇒ 2X/20 years (3%/yr)
Amdahl's Law ⇒ 2X/6 years (12%/year)
End of Dennard Scaling ⇒ Multicore 2X/3.5 years (23%/year)
CISC 2X/2.5 years (22%/year)
RISC 2X/1.5 years (52%/year)
Performance vs. VAX11-780
100,000
10,000
1,000
100
10
1
1980
1985
1990
1995
2000
2005
2010
2015

- 80年代初，RISC技术出现，80年代中期投放市场，使用RISC处理器的计算机的性能以每年52%的高速度增长
  - 增长速度持续了近16年，截止2002年，微处理器的最高性能与单纯依赖技术进步能够达到的性能相比，前者几乎是后者的7倍



CISC指令系统结构存在的问题

（1979年开始，Patterson等人的研究）

- 各种指令的使用频度相差悬殊，许多指令很少用。
  - 据统计：只有20%的指令使用频度比较高，占运行时间的80%。而其余80%的指令只在20%的运行时间内才会用到。
  - 使用频度高的指令也是最简单的指令。

Rank	80x86 instruction	Integer average (% total executed)
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
Total		**96%**

10条简单指令包括了在流行的Intel 80x86计算机上运行的一系列整数程序的所有指令的96%（SPECint92的5个程序运行结果的平均值）

- 指令系统庞大，指令条数很多，许多指令的功能又很复杂，使得控制器硬件非常复杂。

  导致的问题：

  - 占用了大量的芯片面积（如占用CPU芯片总面积的一半以上），给VLSI设计造成很大的困难；
  - 增加了研制时间和成本，容易造成设计错误。

- 许多指令由于操作繁杂，执行速度慢。采用这些复杂指令有可能使整个程序的 执行时间反而增加。
- 由于指令功能复杂，规整性不好，不利于采用流水技术来提高性能。

设计RISC机器遵循的原则

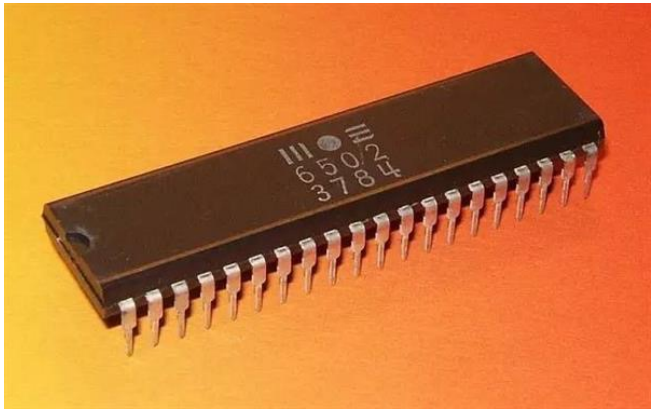
- 指令条数少、指令功能简单。只选取使用频度很高的指令，在此基础上补充一些最有用的指令；
- 采用简单而又统一的指令格式，并减少寻址方式；指令字长都为32位或64位；
- 指令的执行在单个机器周期内完成；

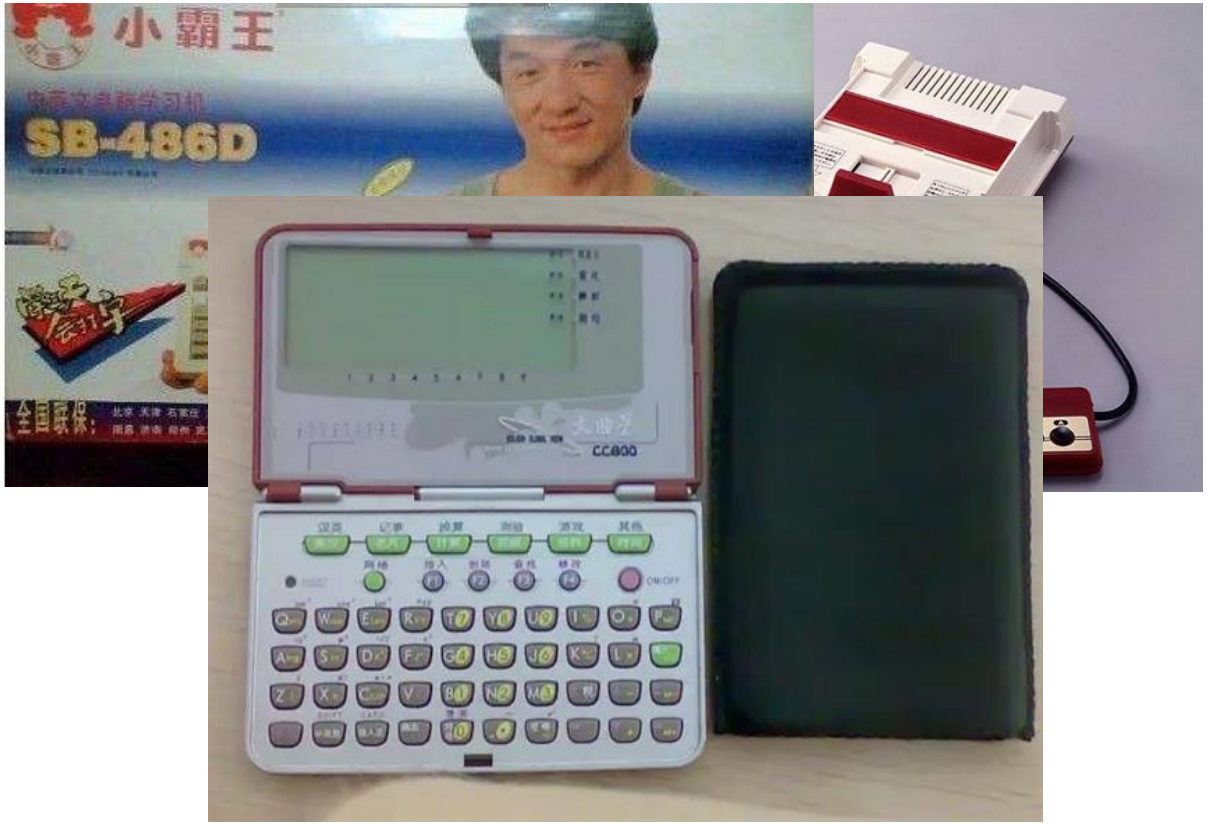
  （采用流水线机制）
- 只有load和store指令才能访问存储器，其它指令的操作都是在寄存器之间进行；

  （即采用load-store结构）
- 大多数指令都采用组合逻辑来实现；
- 强调优化编译器的作用，为高级语言程序生成优化的代码；
- 充分利用流水技术来提高性能。

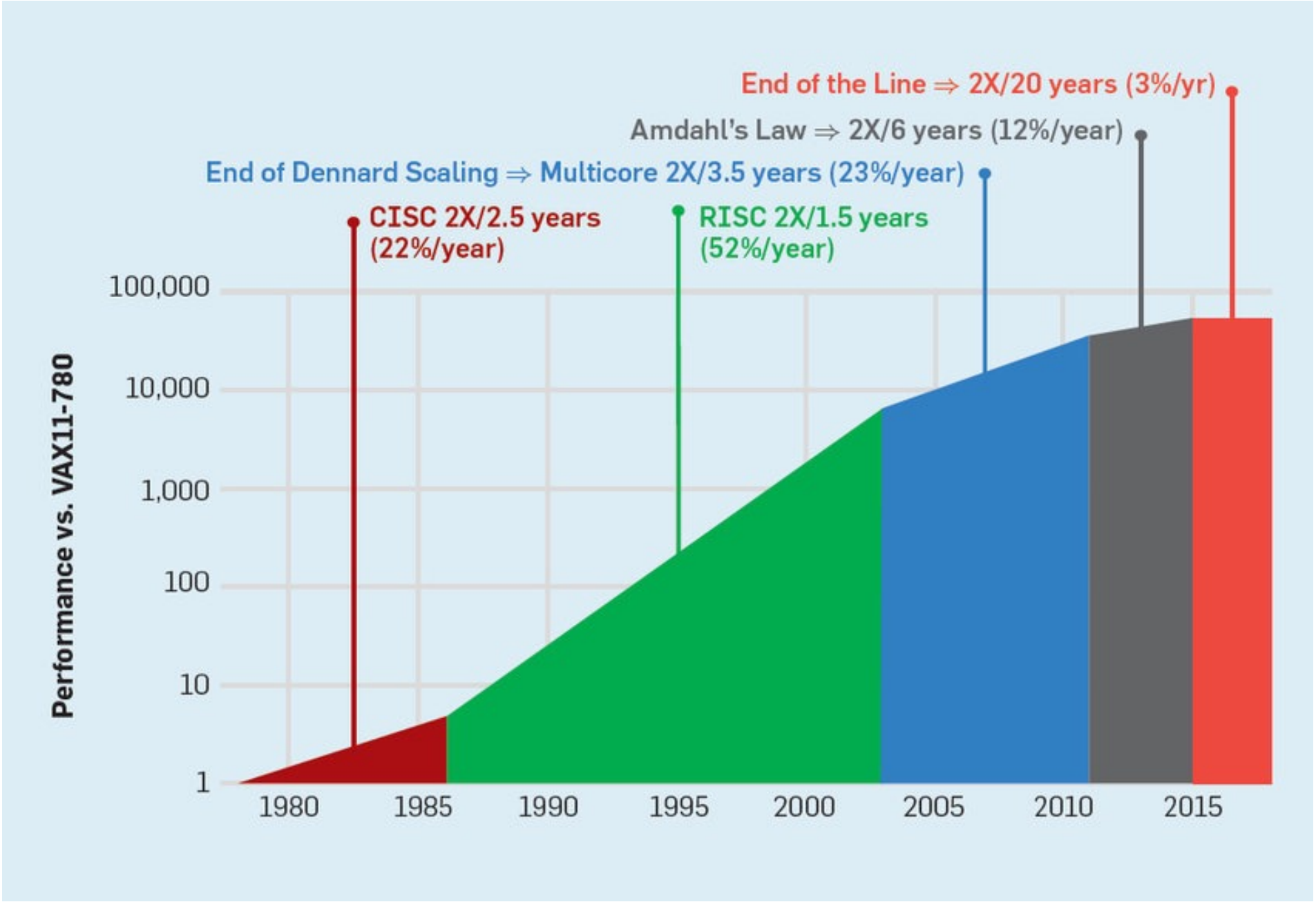
指令级并行

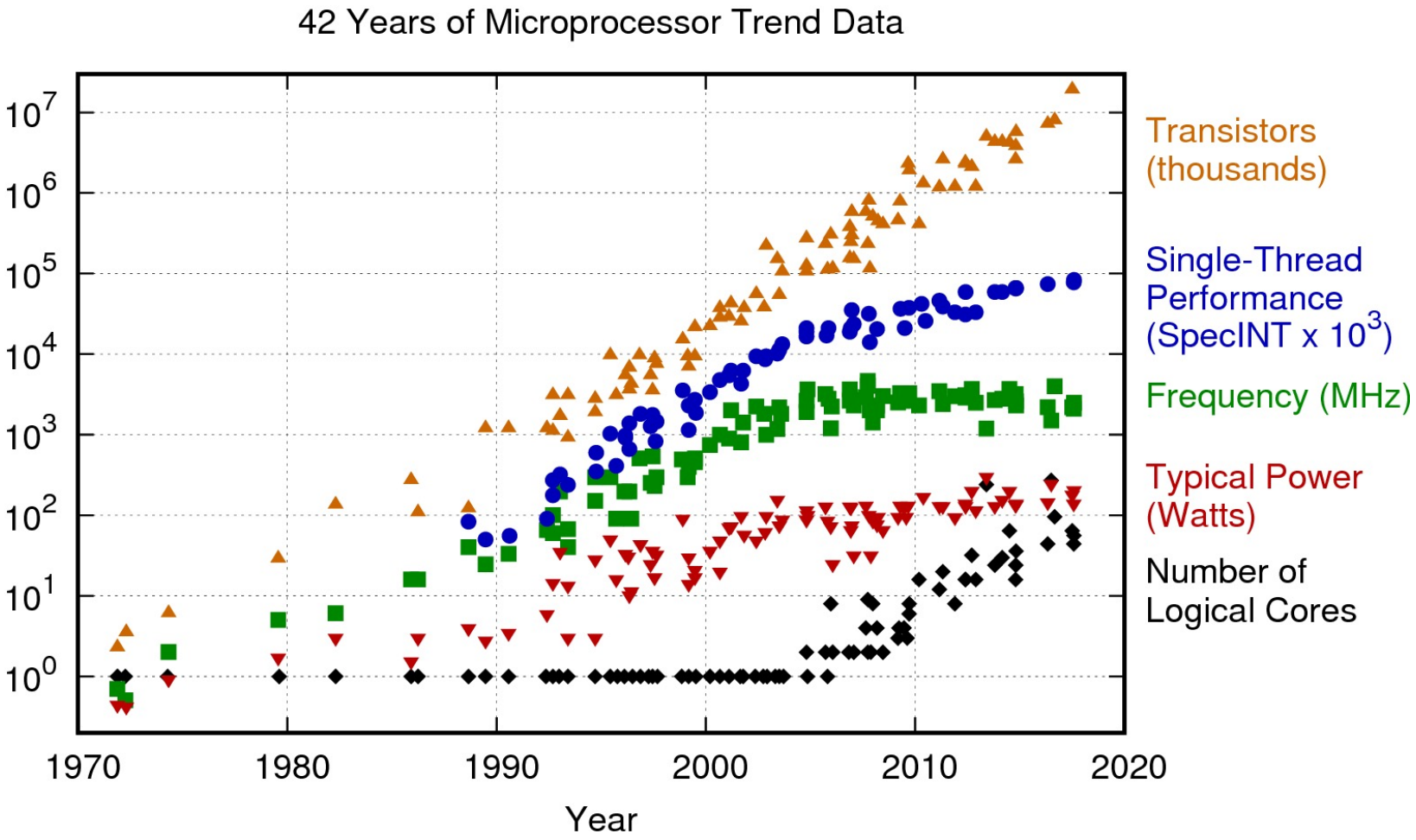
- 流水
  - 让指令并行起来
    - **6502** 取值+执行





- 动态调度
- 分支预测
- 超标量
- 。。。


End of the Line ⇒ 2X/20 years (3%/yr)
Amdahl's Law ⇒ 2X/6 years (12%/year)
End of Dennard Scaling ⇒ Multicore 2X/3.5 years (23%/year)
CISC 2X/2.5 years (22%/year)
RISC 2X/1.5 years (52%/year)
Performance vs. VAX11-780
100,000
10,000
1,000
100
10
1
1980
1985
1990
1995
2000
2005
2010
2015



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

➢ 主要因素：

- 大功耗问题（风冷已到达极限）；

$$\text{Power}_{\text{dynamic}} \propto 1/2 \times \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency switched}$$

降低电压是个好方法 缩放定律失效

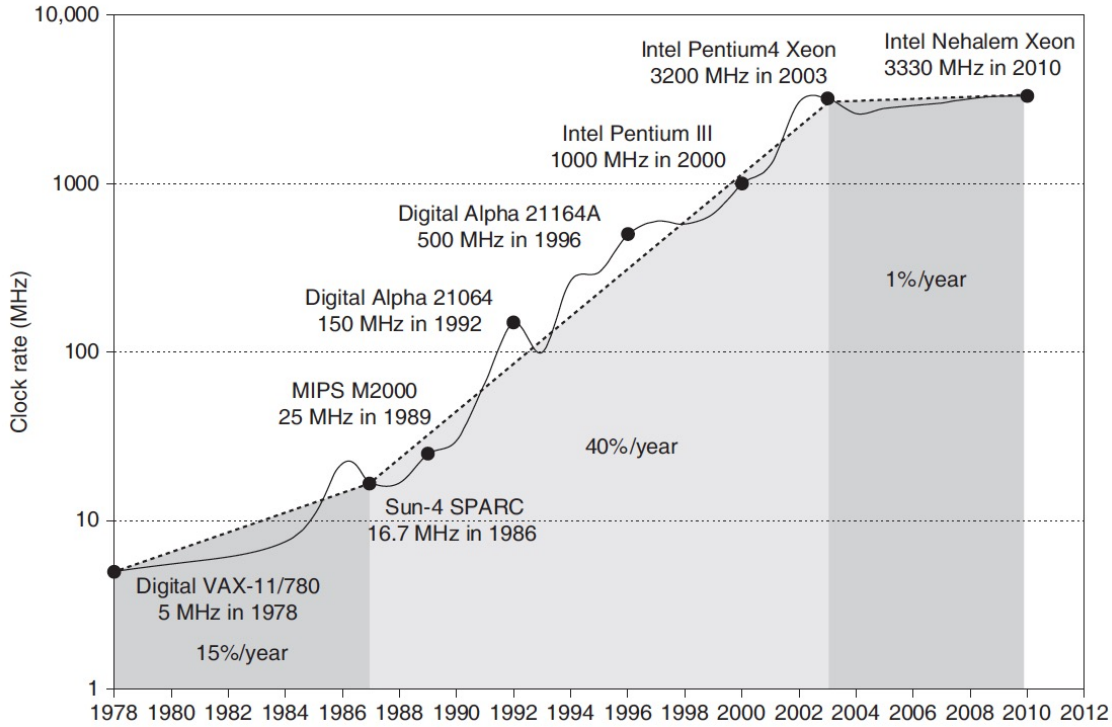
功率约束终究会限制频率的提高

第一代微处理器功率低于1

Core i7 (3.3G) 130W

将这些热量从边长1.5cm的芯片表面消散出去，

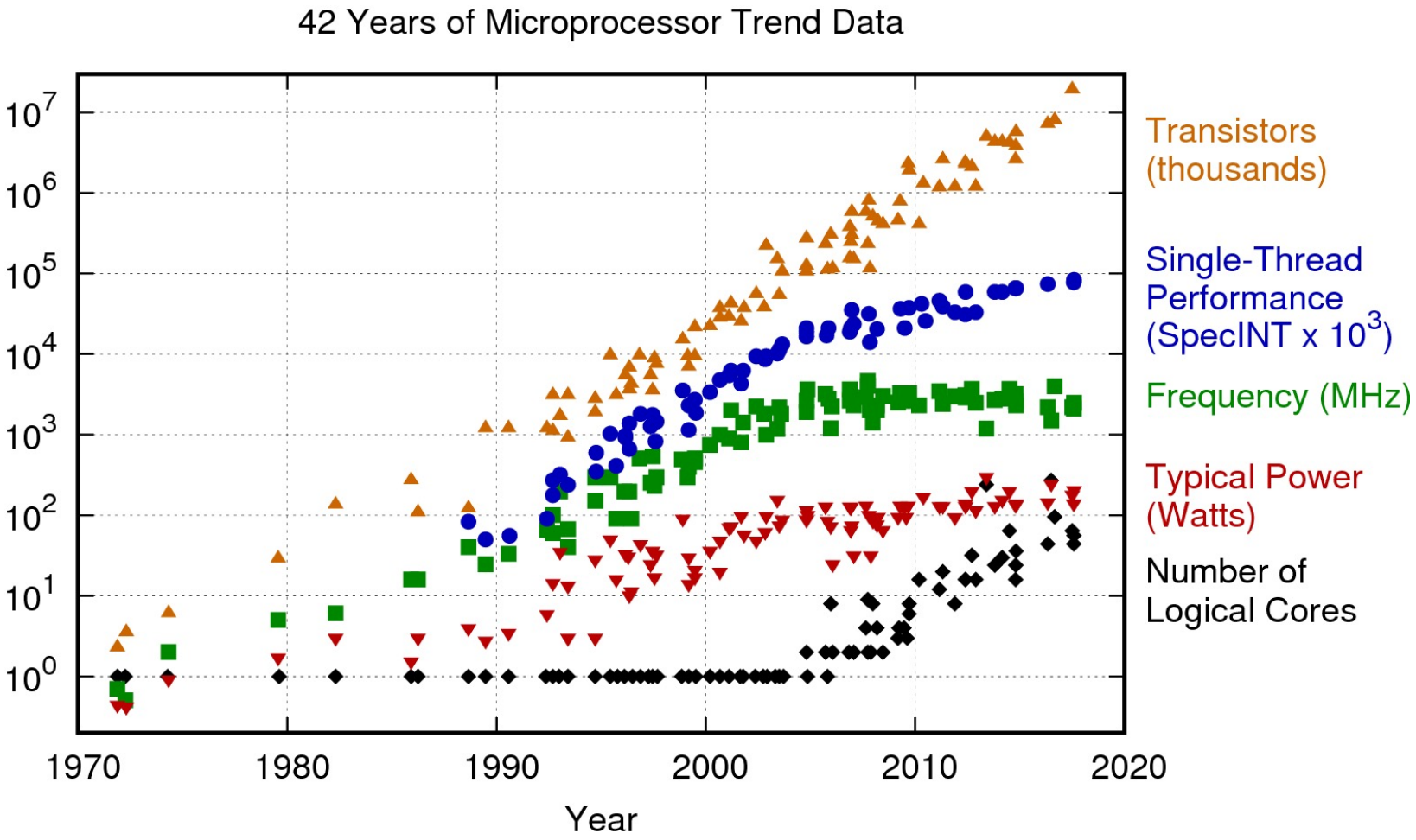
风冷的极限。



动态电压-频率调整（DVFS）

近似计算

- 主要因素：
  - 大功耗问题（风冷已到达极限）；
  - 可以进一步有效地开发的指令级并行性已经很少；
  - 难以降低的存储器访问延时（存储器访问速度的提高缓慢）

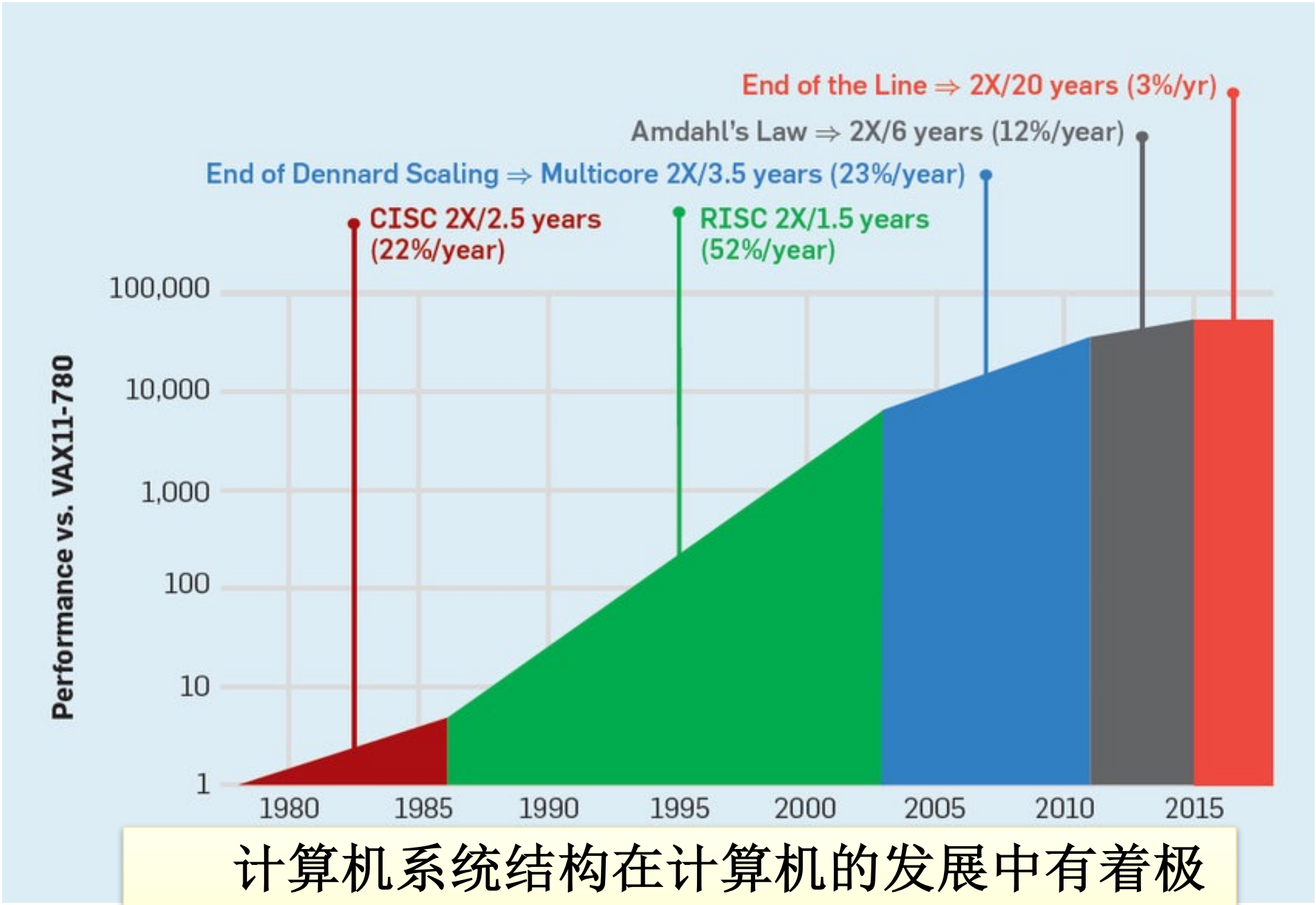


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

系统结构的重大转折：

从单纯依靠指令级并行转向开发线程级并行和数据级并行。

Intel从04年开始也取消了单一高性能处理器的研究，通过多核技术进一步提高处理器性能。



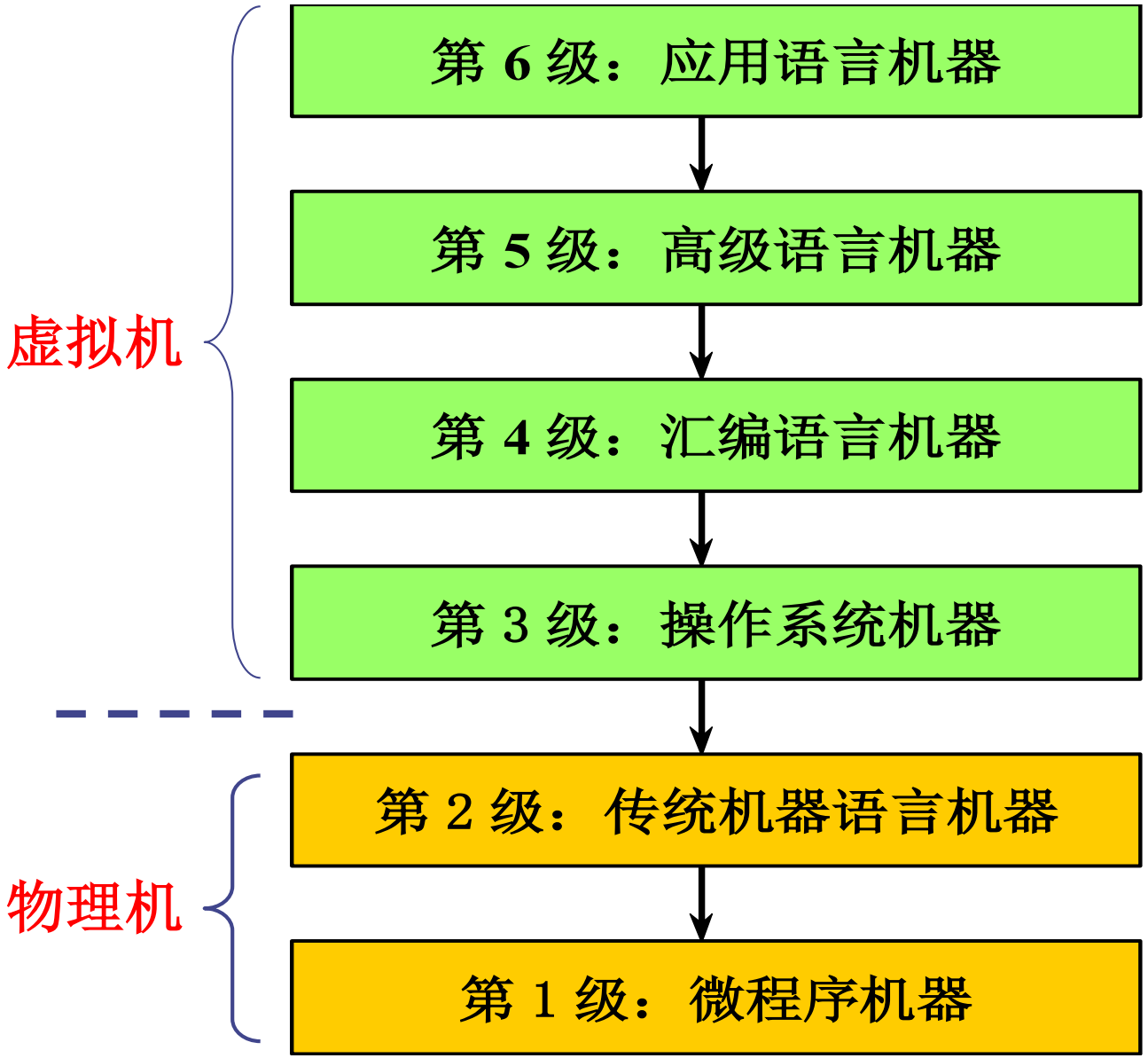
计算机系统结构在计算机的发展中有着极其重要的作用。

计算机系统结构研究什么？

1 计算机系统结构的定义

1.1 计算机系统的层次结构

A computing system is like an onion, made up of many layers. Each layer plays a specific role in the overall design of the system. Nell Dale & John Lewis, Computer Science Illuminated: Second Edition ——摘自：计算机科学概论（英文版·第2版），机械工业出版社，2005.1，第4页


第 6 级：应用语言机器
第 5 级：高级语言机器
第 4 级：汇编语言机器
第 3 级：操作系统机器
第 2 级：传统机器语言机器
第 1 级：微程序机器
虚拟机
物理机

- **物理机**：用硬件/固件实现的机器

  （最下面的两级机器）

- **虚拟机**：由软件实现的机器
  - 虚拟机中有些操作可以由硬件或固件实现。
- 各机器级的实现主要靠翻译或解释，或两者的结合。
  - **翻译**：先用转换程序把高一级机器上的程序转换为低一级机器上等效的程序，然后再在这低一级机器上运行，实现程序的功能。
  - **解释**：对于高一级机器上的程序中的每一条语句或指令，都是转去执行低一级机器上的一段等效程序。执行完后，再去高一级机器取下一条语句或指令，再进行解释执行，如此反复，直到解释执行完整个程序。

解释执行比编译后再执行所花的时间多，但占用的存储空间较少

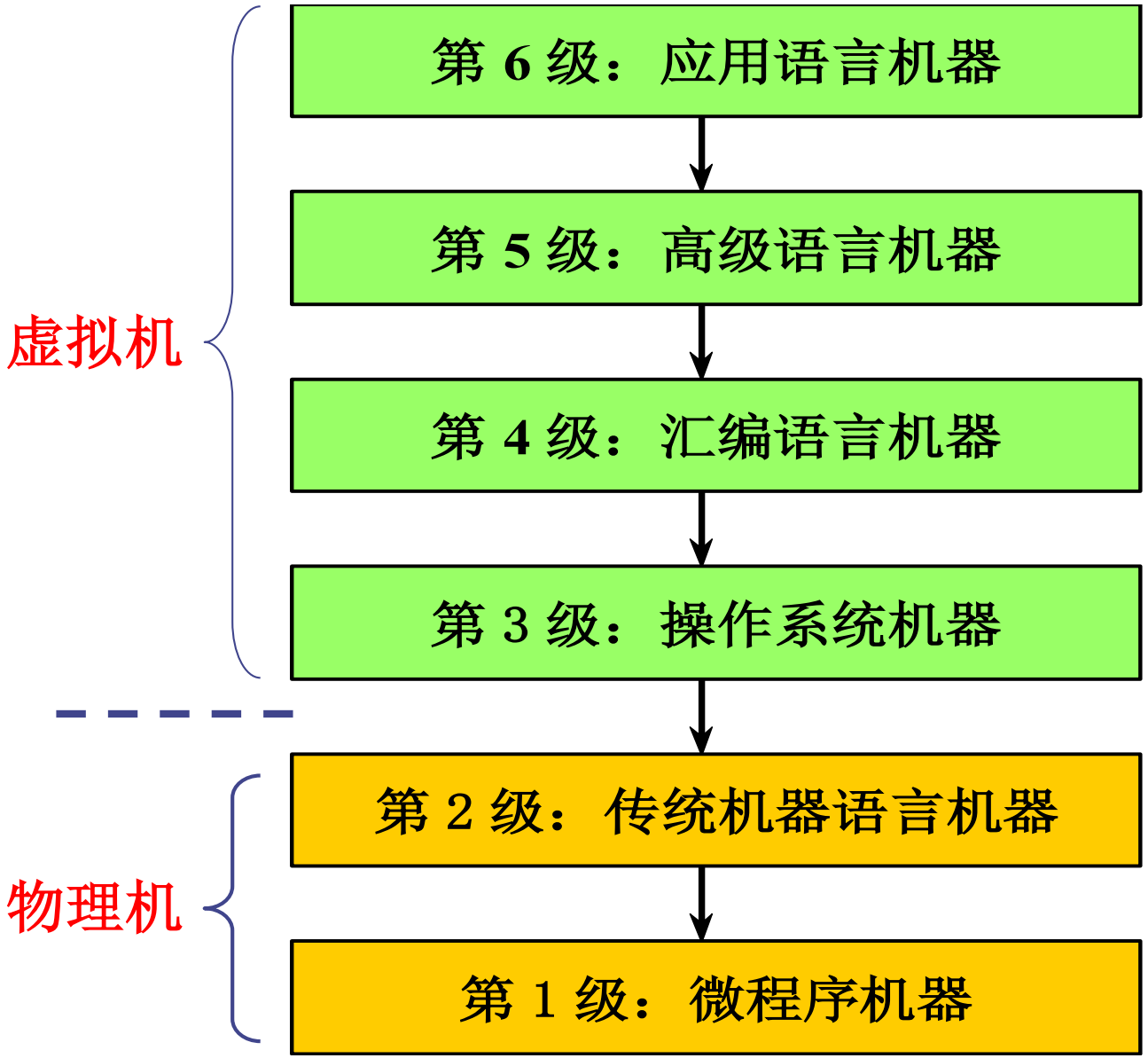
按照计算机系统的多级层次结构，不同级程序员所看到的计算机具有不同的属性。

1.2 计算机系统结构的经典定义

传统机器程序员所看到的计算机属性，即概念性结构与功能特性。

（1964年 Amdahl在介绍IBM360系统时提出的）

到底能看到些什么？


第 6 级：应用语言机器
第 5 级：高级语言机器
第 4 级：汇编语言机器
第 3 级：操作系统机器
第 2 级：传统机器语言机器
第 1 级：微程序机器
虚拟机
物理机

- 计算机系统结构通常仅指指令集设计（仅包括**功能与概念性的结构**），计算机设计的其他方面则被称为**实现**
  - 这种说法不仅不正确，甚至应对新指令集设计中出现的错误负责—— Hennessy and Patterson
- 一台计算机的**实现**包括两部分内容：组成（结构）和硬件
  - **组成**涵盖了计算机设计的高层次方面，如存储系统、总线结构以及内部CPU（中央处理单元——实现算术运算、逻辑运算、分支和数据传输的部件）的设计
    - **AMD的Opteron 与Intel Core i7拥有同样的指令集系统，但却有着完全不同的组成。两者都是X86指令集，却使用完全不同的流水线和Cache结构**

- **硬件**是指一台计算机实现的具体细节。包括具体的逻辑设计和封装技术。
  - 同一系列的计算机通常具有相同的指令集系统结构和几乎完全相同的组成，但它们的具体硬件实现却不同。例如，**Intel Core i7**和**Xeon 7560**，二者具有几乎完全相同的组成，但时钟频率和存储系统的实现细节的有所不同使得**7560**更加适合服务器。
  - 乘法举例
- 计算机系统结构（指令系统结构）、计算机组成和计算机实现是三个不同的概念，但随着技术、器件和应用的发展，三者之间的界限越来越模糊

广义的系统结构定义：指令系统结构、组成、硬件
（计算机设计的3个方面）

1.3 计算机系统设计者的主要任务

计算机系统结构的设计者必须设计能够满足包括价格、功耗、性能和可用性指标的计算机。

大致需要3个方面的工作

① 确定用户对计算机系统的功能、价格和性能的要求

- 由市场需求激发的特定功能
- 应用软件通常能帮助我们筛选出特定的功能要求

设计者需要考虑的一些主要因素

Functional requirements	Typical features required or supported
Application area	*Target of computer*
Personal mobile device	Real-time performance for a range of tasks, including interactive performance for graphics, video, and audio; energy efficiency (Ch. 2, 3, 4, 5; App. A)
General-purpose desktop	Balanced performance for a range of tasks, including interactive performance for graphics, video, and audio (Ch. 2, 3, 4, 5; App. A)
Servers	Support for databases and transaction processing; enhancements for reliability and availability; support for scalability (Ch. 2, 5; App. A, D, F)
Clusters/warehouse-scale computers	Throughput performance for many independent tasks; error correction for memory; energy proportionality (Ch 2, 6; App. F)
Embedded computing	Often requires special support for graphics or video (or other application-specific extension); power limitations and power control may be required; real-time constraints (Ch. 2, 3, 5; App. A, E)
Level of software compatibility	*Determines amount of existing software for computer*
At programming language	Most flexible for designer; need new compiler (Ch. 3, 5; App. A)
Object code or binary compatible	Instruction set architecture is completely defined—little flexibility—but no investment needed in software or porting programs (App. A)
Operating system requirements	*Necessary features to support chosen OS (Ch. 2; App. B)*
Size of address space	Very important feature (Ch. 2); may limit applications
Memory management	Required for modern OS; may be paged or segmented (Ch. 2)
Protection	Different OS and application needs: page vs. segment; virtual machines (Ch. 2)
Standards	*Certain standards may be required by marketplace*
Floating point	Format and arithmetic: IEEE 754 standard (App. J), special arithmetic for graphics or signal processing
I/O interfaces	For I/O devices: Serial ATA, Serial Attached SCSI, PCI Express (App. D, F)
Operating systems	UNIX, Windows, Linux, CISCO IOS
Networks	Support required for different networks: Ethernet, Infiniband (App. F)
Programming languages	Languages (ANSI C, C++, Java, Fortran) affect instruction set (App. A)

- **应用领域**

  专用还是通用？面向科学计算还是面向商用处理？

- ✓ 个人移动设备
- ✓ 桌面机Desktop
- ✓ 服务器Server
- ✓ 集群/超大规模集群（数据中心）
- ✓ 嵌入式

Feature	Personal mobile device (PMD)	Desktop	Server	Clusters/warehouse-scale computer	Embedded
Price of system	$100–$1000	$300–$2500	$5000–$10,000,000	$100,000–$200,000,000	$10–$100,000
Price of micro-processor	$10–$100	$50–$500	$200–$2000	$50–$250	$0.01–$100
Critical system design issues	Cost, energy, media performance, responsiveness	Price-performance, energy, graphics performance	Throughput, availability, scalability, energy	Price-performance, throughput, energy proportionality	Price, energy, application-specific performance

设计者需要考虑的一些主要因素

Functional requirements	Typical features required or supported
Application area	*Target of computer*
Personal mobile device	Real-time performance for a range of tasks, including interactive performance for graphics, video, and audio; energy efficiency (Ch. 2, 3, 4, 5; App. A)
General-purpose desktop	Balanced performance for a range of tasks, including interactive performance for graphics, video, and audio (Ch. 2, 3, 4, 5; App. A)
Servers	Support for databases and transaction processing; enhancements for reliability and availability; support for scalability (Ch. 2, 5; App. A, D, F)
Clusters/warehouse-scale computers	Throughput performance for many independent tasks; error correction for memory; energy proportionality (Ch 2, 6; App. F)
Embedded computing	Often requires special support for graphics or video (or other application-specific extension); power limitations and power control may be required; real-time constraints (Ch. 2, 3, 5; App. A, E)
Level of software compatibility	*Determines amount of existing software for computer*
At programming language	Most flexible for designer; need new compiler (Ch. 3, 5; App. A)
Object code or binary compatible	Instruction set architecture is completely defined—little flexibility—but no investment needed in software or porting programs (App. A)
Operating system requirements	*Necessary features to support chosen OS (Ch. 2; App. B)*
Size of address space	Very important feature (Ch. 2); may limit applications
Memory management	Required for modern OS; may be paged or segmented (Ch. 2)
Protection	Different OS and application needs: page vs. segment; virtual machines (Ch. 2)
Standards	*Certain standards may be required by marketplace*
Floating point	Format and arithmetic: IEEE 754 standard (App. J), special arithmetic for graphics or signal processing
I/O interfaces	For I/O devices: Serial ATA, Serial Attached SCSI, PCI Express (App. D, F)
Operating systems	UNIX, Windows, Linux, CISCO IOS
Networks	Support required for different networks: Ethernet, Infiniband (App. F)
Programming languages	Languages (ANSI C, C++, Java, Fortran) affect instruction set (App. A)

- **软件兼容性的选择**

  软件兼容是指一台计算机上的程序不加修改就可以移植另一台计算机上正常运行。

- ✓ 高级语言层面的兼容：设计更加灵活，需要设计新编译器
- ✓ 目标代码或二进制层面的兼容：限制比较严格
  - ✓ 指令集兼容
  - ✓ 软件可移植性

- **软件可移植性的实现**

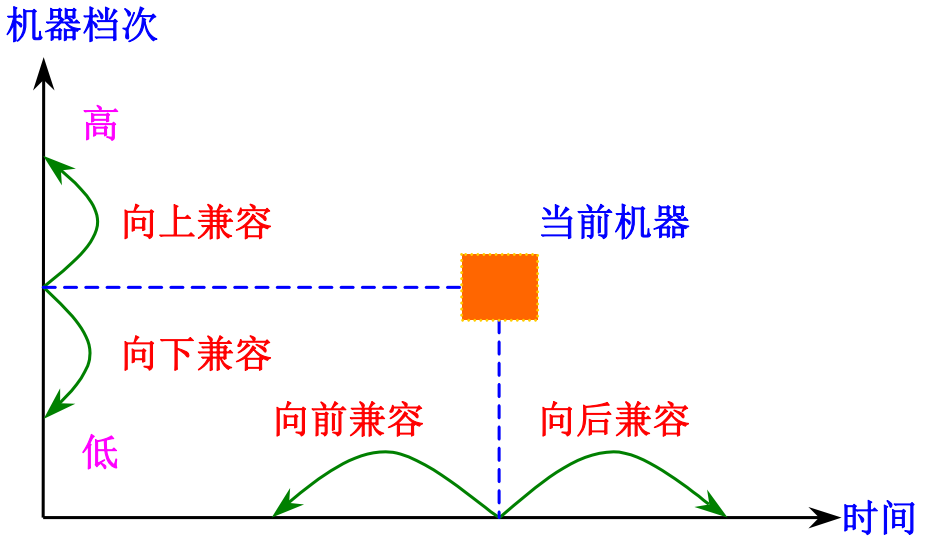
1. 统一高级语言
2. 系列机
3. 模拟和仿真

1. 统一高级语言
   - 实现软件移植的一种理想的方法
   - 较难实现
2. 系列机

   由同一厂家生产的具有相同的系统结构(ISA)，但具有不同组成和实现的一系列不同型号的机器

   - 较好地解决软件开发要求系统结构相对稳定与器件、硬件技术迅速发展的矛盾。

- 软件兼容
  - 向上（下）兼容：按某档机器编制的程序，不加修改就能运行于比它高（低）档的机器。
  - 向前（后）兼容：按某个时期投入市场的某种型号机器编制的程序，不加修改地就能运行于在它之前（后）投入市场的机器。
  - 向后兼容是系列机的根本特征。
- 兼容机：由不同公司厂家生产的具有相同系统结构的计算机 。



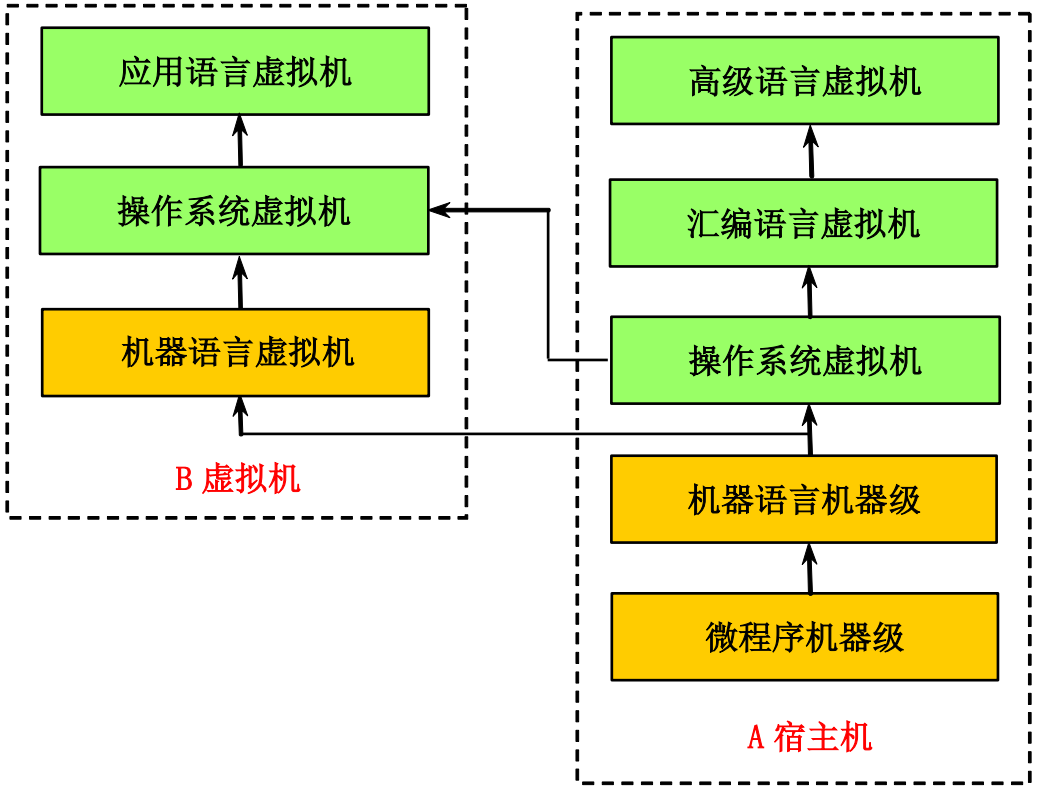
具有代表性的计算机（IBM360）

- 1961年开始研制，1964年获得成功，电脑系统用360为名， 代表着360电脑从工商业到科学界的全方位应用
- 系列产品，大中小型360计算机系统，都能处理相同的指令，使用相同的软件，配置相同的外部设备，而且能够相互连接在一起工作。IBM360共有6个型号的大、中、小型系统和44种新式的配套设备。

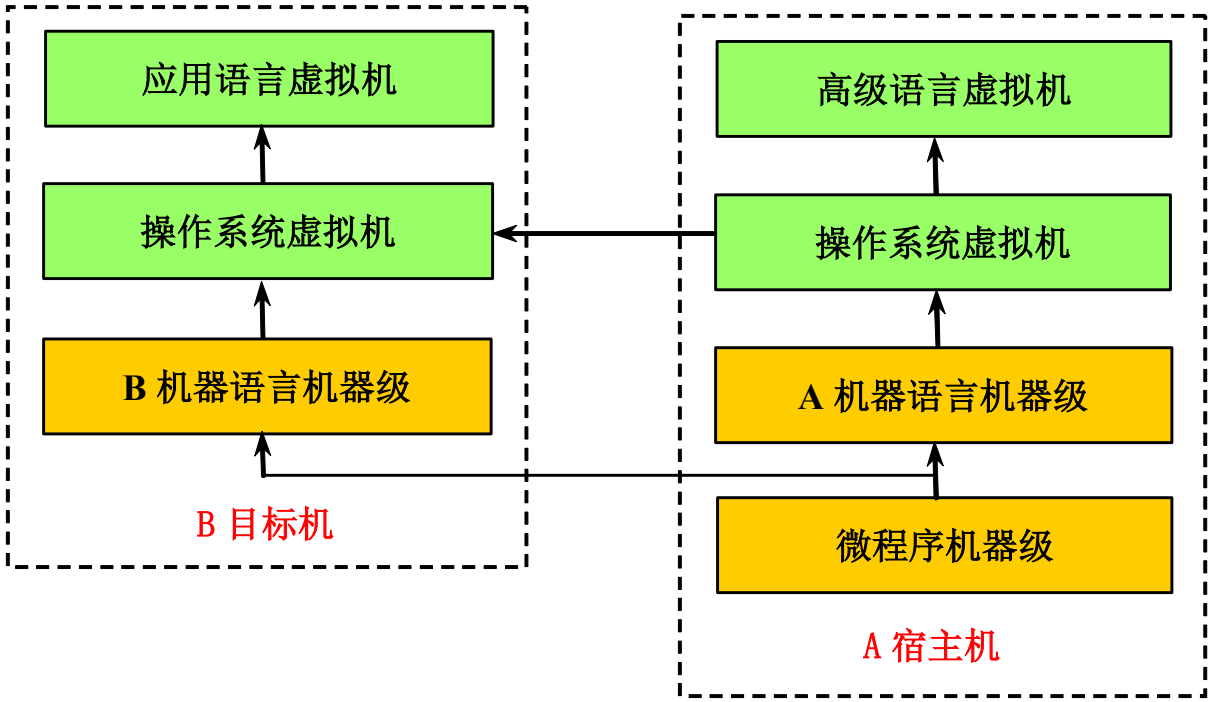
3. 模拟和仿真

- 使软件能在具有不同系统结构的机器之间相互移植。
  - 在一种系统结构上实现另一种系统结构。
  - 从指令集的角度来看，就是要在一种机器上实现另一种机器的指令集。

- **模拟**：用软件的方法在一台现有的机器（称为宿主机）上实现另一台机器（称为虚拟机）的指令集。
  - 通常用解释的方法来实现。
  - 运行速度较慢，性能较差。



- **仿真**：用一台现有机器（宿主机）上的微程序去解释实现另一台机器（目标机）的指令集。
  - 运行速度比模拟方法的快
  - 仿真只能在系统结构差距不大的机器之间使用



设计者需要考虑的一些主要因素

Functional requirements	Typical features required or supported
Application area	*Target of computer*
Personal mobile device	Real-time performance for a range of tasks, including interactive performance for graphics, video, and audio; energy efficiency (Ch. 2, 3, 4, 5; App. A)
General-purpose desktop	Balanced performance for a range of tasks, including interactive performance for graphics, video, and audio (Ch. 2, 3, 4, 5; App. A)
Servers	Support for databases and transaction processing; enhancements for reliability and availability; support for scalability (Ch. 2, 5; App. A, D, F)
Clusters/warehouse-scale computers	Throughput performance for many independent tasks; error correction for memory; energy proportionality (Ch 2, 6; App. F)
Embedded computing	Often requires special support for graphics or video (or other application-specific extension); power limitations and power control may be required; real-time constraints (Ch. 2, 3, 5; App. A, E)
Level of software compatibility	*Determines amount of existing software for computer*
At programming language	Most flexible for designer; need new compiler (Ch. 3, 5; App. A)
Object code or binary compatible	Instruction set architecture is completely defined—little flexibility—but no investment needed in software or porting programs (App. A)
Operating system requirements	*Necessary features to support chosen OS (Ch. 2; App. B)*
Size of address space	Very important feature (Ch. 2); may limit applications
Memory management	Required for modern OS; may be paged or segmented (Ch. 2)
Protection	Different OS and application needs: page vs. segment; virtual machines (Ch. 2)
Standards	*Certain standards may be required by marketplace*
Floating point	Format and arithmetic: IEEE 754 standard (App. J), special arithmetic for graphics or signal processing
I/O interfaces	For I/O devices: Serial ATA, Serial Attached SCSI, PCI Express (App. D, F)
Operating systems	UNIX, Windows, Linux, CISCO IOS
Networks	Support required for different networks: Ethernet, Infiniband (App. F)
Programming languages	Languages (ANSI C, C++, Java, Fortran) affect instruction set (App. A)

- 操作系统需求

  包括地址空间大小、存储管理、保护等。从系统结构上对操作系统的需求提供支持，是很重要的一点。

- 标准

  确定系统中哪些方面要采用标准以及采用什么标准。如：浮点数标准、I/O总线标准、网络标准、程序设计语言标准等。

1.3 计算机系统设计者的主要任务

② 软硬件功能分配

- 考虑如何优化设计？

  必须考虑软硬件功能的合理分配。

- 软件和硬件在实现功能上是等价的
  - 用软件实现的优点：设计容易、修改简单，而且可以减少硬件成本。但是所实现的功能的速度较慢。
  - 用硬件实现的优点：速度快、性能高，但它修改困难，灵活性差。
- 在软硬件之间进行折中和取舍。

③ 设计出生命周期长的系统结构

- 特别注意计算机应用和计算机技术的发展趋势
- 设计出具有一定前瞻性的系统结构，以使得它具有较长的生命周期。