高级计算机系统结构

沈海华
shenhh@ucas.ac.cn

课程基本信息

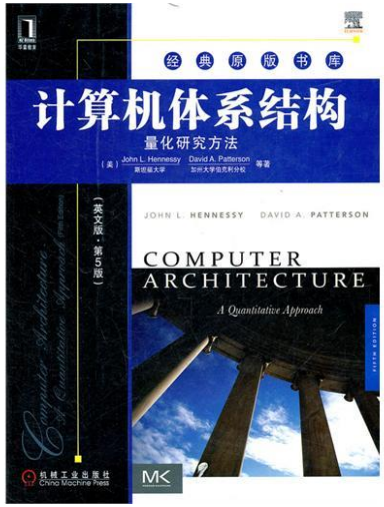
- 上课时间：每周一、三上午1~2节（8:30-10:10）

  地点：教一208
- 课程要求：熟练掌握计算机系统结构的基础知识，了解系统结构的前沿发展，初步掌握现代处理器设计的基础流程和方法。
- 答疑和考勤：本课程配备2~3名助教，时间允许时教师会在课间答疑，大多数情况下教师或助教课后会在微信群里答疑。本课程会抽查考勤，结果计入平时成绩。
- 课程设计：采用开源流程全程设计开源处理器（从RTL到GDS）
- 考核方式：闭卷考试
- 成绩分布（暂定）=平时成绩10%+课程设计40%+闭卷考试50%

课程参考书

- John L. Hennessy, David A. Patterson “Computer Architecture: A Quantitative Approach” ，机械工业出版社

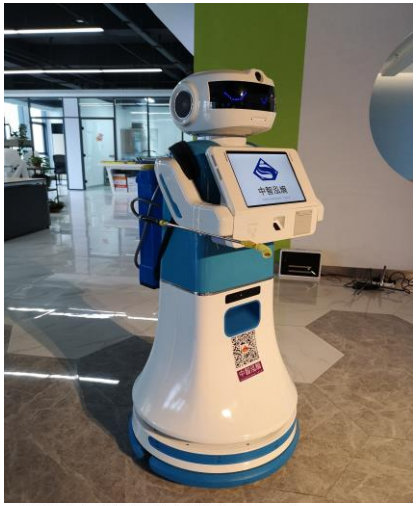
- 胡伟武等，《计算机体系结构》 清华大学出版社



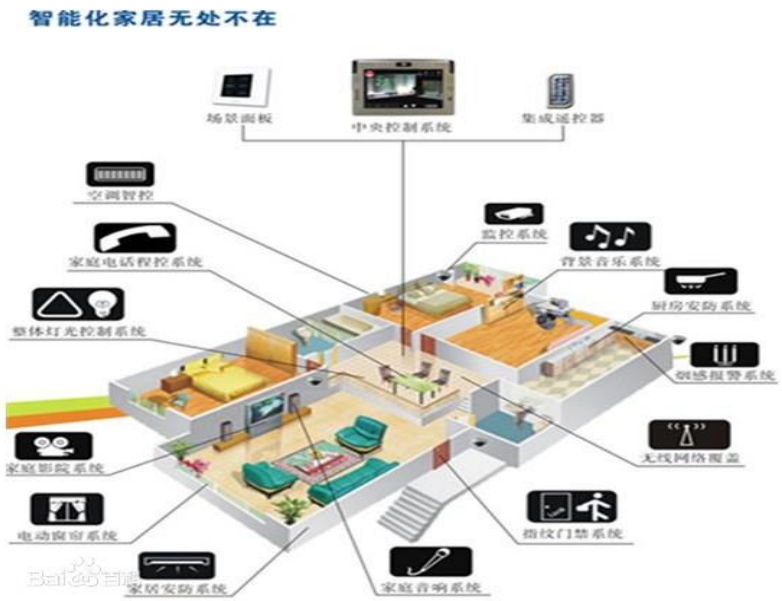
本课程没有直接对应的参考书，推荐阅读：

- HPCA, ISCA, MICRO等顶级期刊会议的论文

生活中各种各样的“计算机”



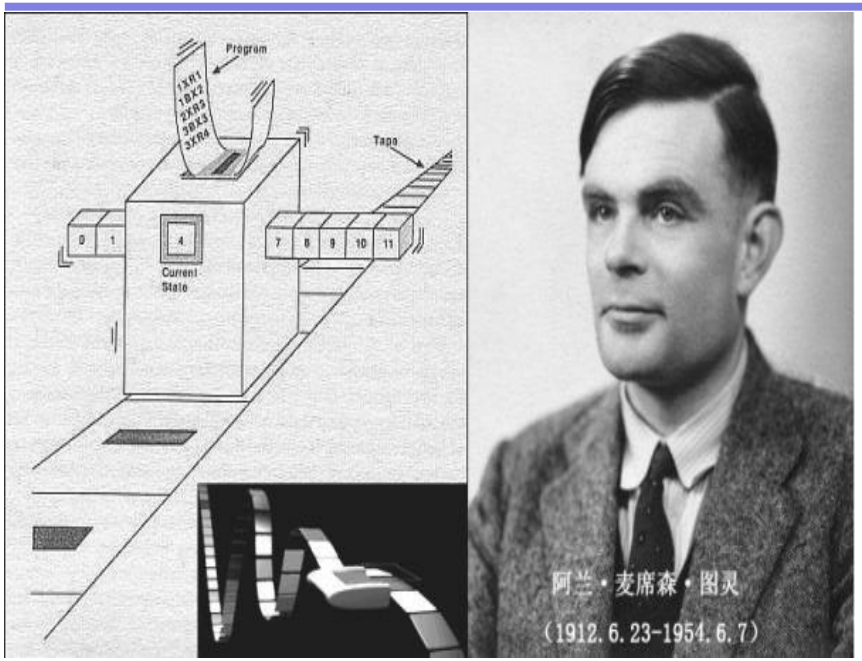
HUAWEI P50 Pocket



第一讲 引言

- 计算机发展简史
- 计算机系统结构的演变及发展趋势
- 高级计算机系统结构课程覆盖范围

图灵机



Alan Mathison Turing（ 1912－1954 ）

英国数学家、逻辑学家，计算机科学之父。

- 1936年模仿人类的计算过程提出一种抽象的计算模型，构造了现代计算机的雏形。
- 包括状态部分（输入、输出）和运算控制逻辑（程序）

- **计算复杂性理论**

研究计算问题时所需的资源，比如时间和空间，以及如何尽可能的节省这些资源。

计算复杂性理论

研究计算问题时所需的资源，比如时间和空间，以及如何尽可能的节省这些资源。

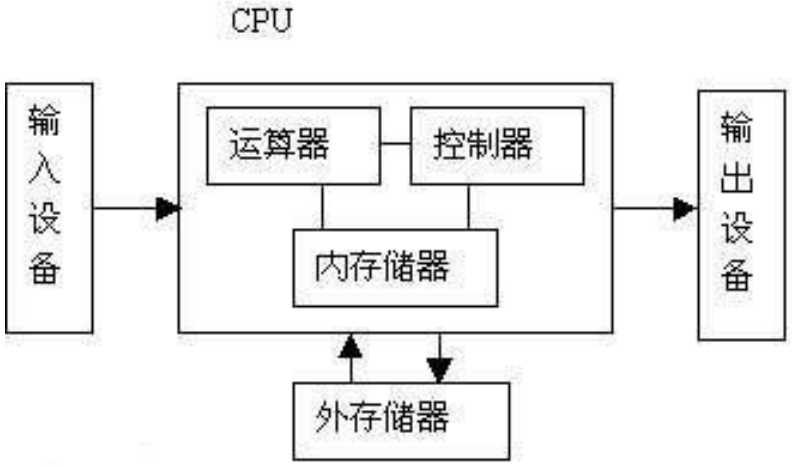
- **Church-Turing理论**

任何能直观计算的问题都能被图灵机计算。 如果证明了某个问题使用图灵机是不可解决的，那么这个问题就是不可解决的。

P和NP问题

- P类问题：算法能在多项式时间内完成的问题。
- NP问题：对于一类问题，我们可能没有一个已知的快速的方法得到问题的答案，但是如果给我们一个candidate answer，我们能够用多项式时间来验证这个解是否正确。这类问题叫做NP problem。所以很显然 P Problem是NP problem的一个子集。
- NP-Hard问题：概括特征就是“at least as hard as the hardest problems in NP Problem”， 就是NP-hard问题至少和NP问题一样难。
- NP-Complete问题：这类问题满足两个性质：1）在polynomial时间内可以验证一个candidate answer是不是真正的解；2）是我们可以把任何一个NP问题在polynomial时间内转化成为NP- complete问题。
- P＝**NP**？，或者P是**NP**的一个真子集？

冯诺依曼结构



John Von Neumann

（1903－1957）

美籍匈牙利人，上世纪最伟大的科学家之一，数字计算机之父。

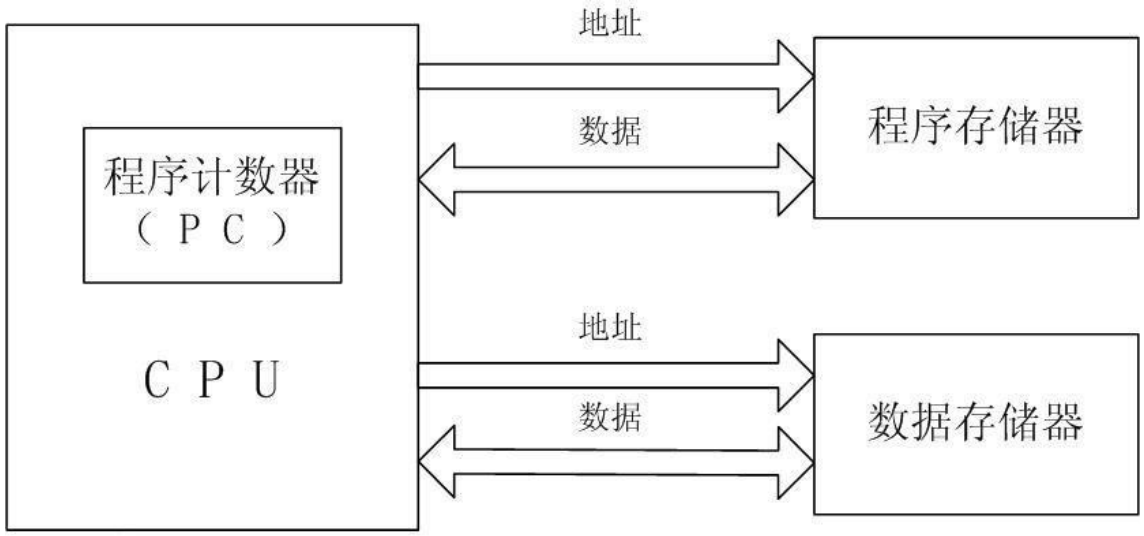
- 冯·诺依曼结构（也称普林斯顿结构），是一种将程序指令存储器和数据存储器合并在一起的存储器结构。程序指令存储地址和数据存储地址指向同一个存储器的不同物理位置，因此程序指令和数据的宽度相同或呈倍数。

冯·诺伊曼瓶颈（von Neumann bottleneck）

- 冯诺依曼结构的特点是CPU与内存分开，随着计算机系统结构的发展，人们发现这样的设计并非十全十美，会导致所谓的memory wall。

- 冯·诺伊曼瓶颈一词在1977年ACM图灵奖得主约翰·巴科斯获奖致词时第一次出现，目前，这一问题已越来越严重。

哈佛结构



哈佛结构体系结构

- 哈佛结构的主要特点是将程序和数据独立存储在不同的存储空间中，各自独立编址、独立访问，以减轻程序运行时的访存瓶颈。哈佛结构本质上是一种并行体系结构。

- 采用哈佛结构的芯片有：Microchip公司的PIC系列芯片，摩托罗拉公司的MC68系列、Zilog公司的Z8系列、ATMEL公司的AVR系列和ARM公司的ARM9、ARM10和ARM11。

冯诺依曼结构 vs.哈佛结构

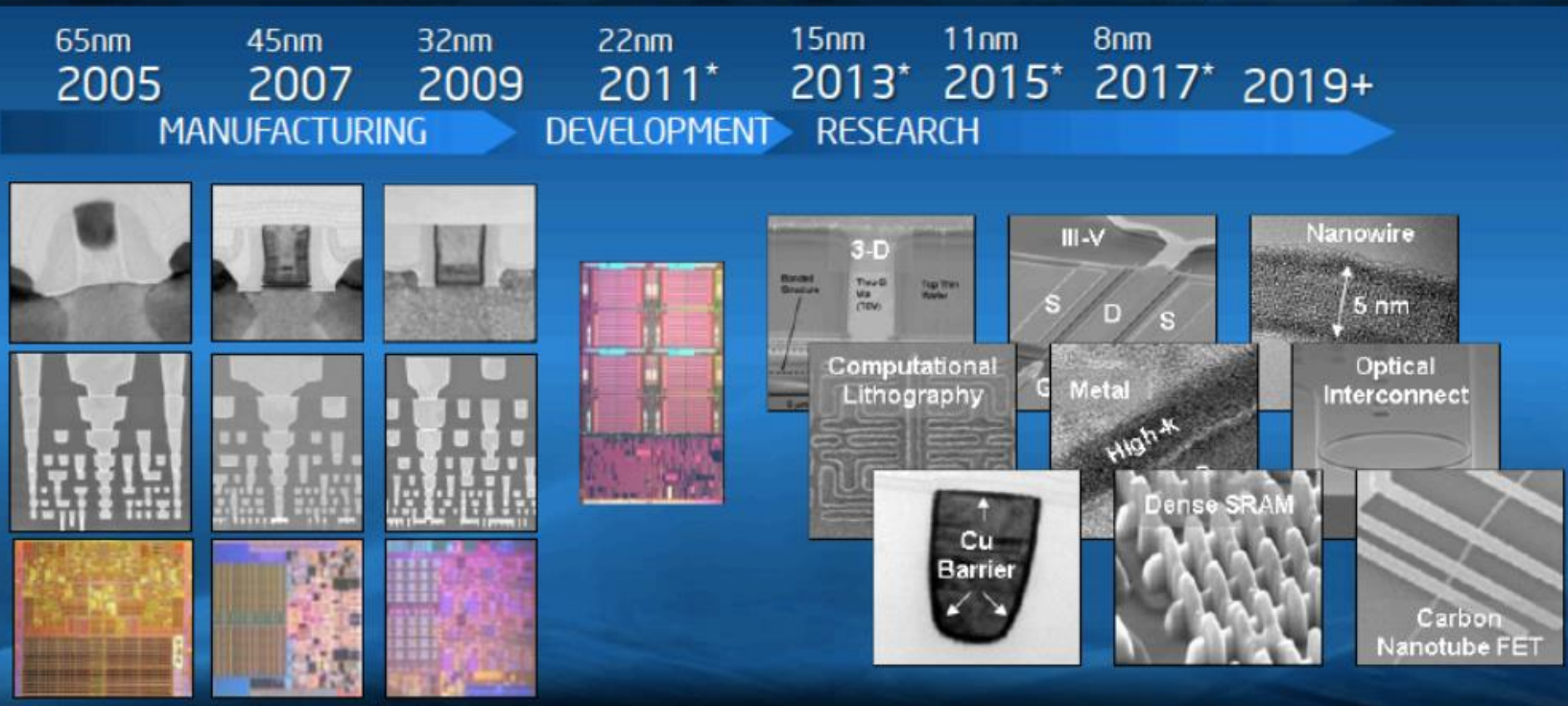
- 冯诺依曼结构设计简单、成本低，是现代处理器的基本结构。缺点是存在访存瓶颈问题。
- 哈佛结构具有较高的执行效率，可以在访存时方便地进行指令预取，可以更好地支持并行处理。缺点是复杂，对外围设备的连接与处理要求高，不适合外围存储器的扩展。
- cache的应用使得二者有机会结合起来：在CPU内部部分使用哈佛结构，通过使用不同的数据和指令cache，可以有效提高指令执行的效率；在CPU外部仍使用冯·诺依曼结构，避免过高的设计复杂度和成本。

计算机系统基本分类

- CISC（Complex Instruction Set Computer ）
  - 早期的CPU寄存器少且价格昂贵，主频低，运算速度慢。为了提高运算性能，人们不得不将越来越多的复杂功能的指令加入到指令系统中，以提高处理器的处理效率。
  - INTEL,AMD

- RISC（Reduced Instruction Set Computer ）
  - 2-8原则：80%的程序只使用20%的指令。
  - 简化计算机指令功能，尽量保留那些功能简单、能在一个节拍内执行完成的指令，以简化CPU硬件设计，提高处理器的工作主频，同时大量使用通用寄存器来提高程序执行的速度。

Moore's Law: Alive and Well at Intel



Intel Innovation-Enabled Technology Pipeline is Full

intel

问题

- RISC诞生的原因？

✓集成电路工艺进步，通用寄存器(latch, Flip-Flop) 使用限制大幅减小；

✓处理器主频超过了存储器；

✓随着指令集日趋庞大，编译器设计复杂度成为必须考虑的因素。

通用寄存器对指令集架构的影响

0, 1, 2, 3 address machines

- 0-address: stack machine (push A, pop A, op)
- 1-address: accumulator machine (ld A, st A, op A)
- 2-address: 2-operand machine (one is both source and dest)
- 3-address: 3-operand machine (source and dest are separate)

问题

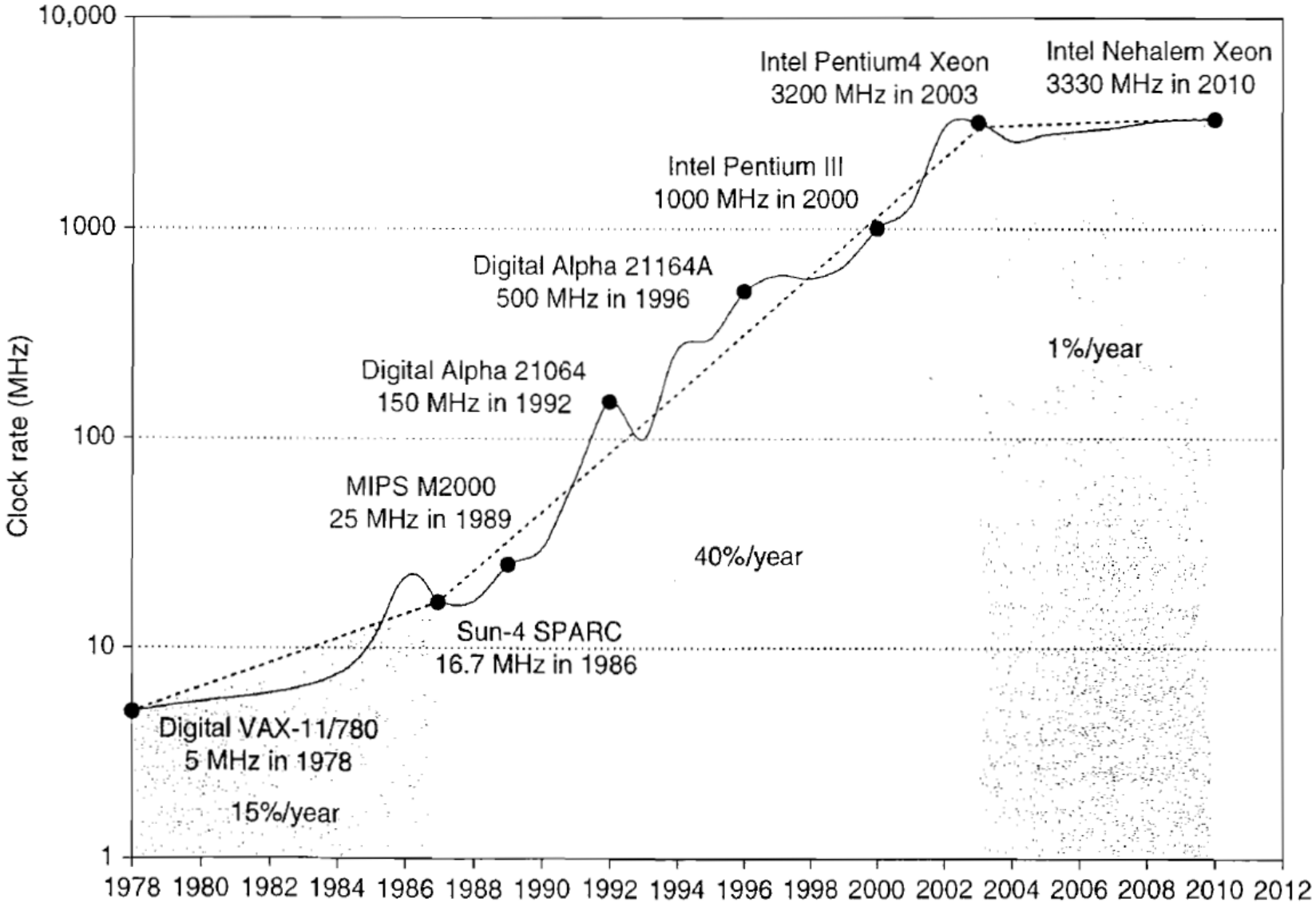
- RISC诞生的原因？

✓集成电路工艺进步，通用寄存器(latch, Flip-Flop) 使用限制大幅减小；

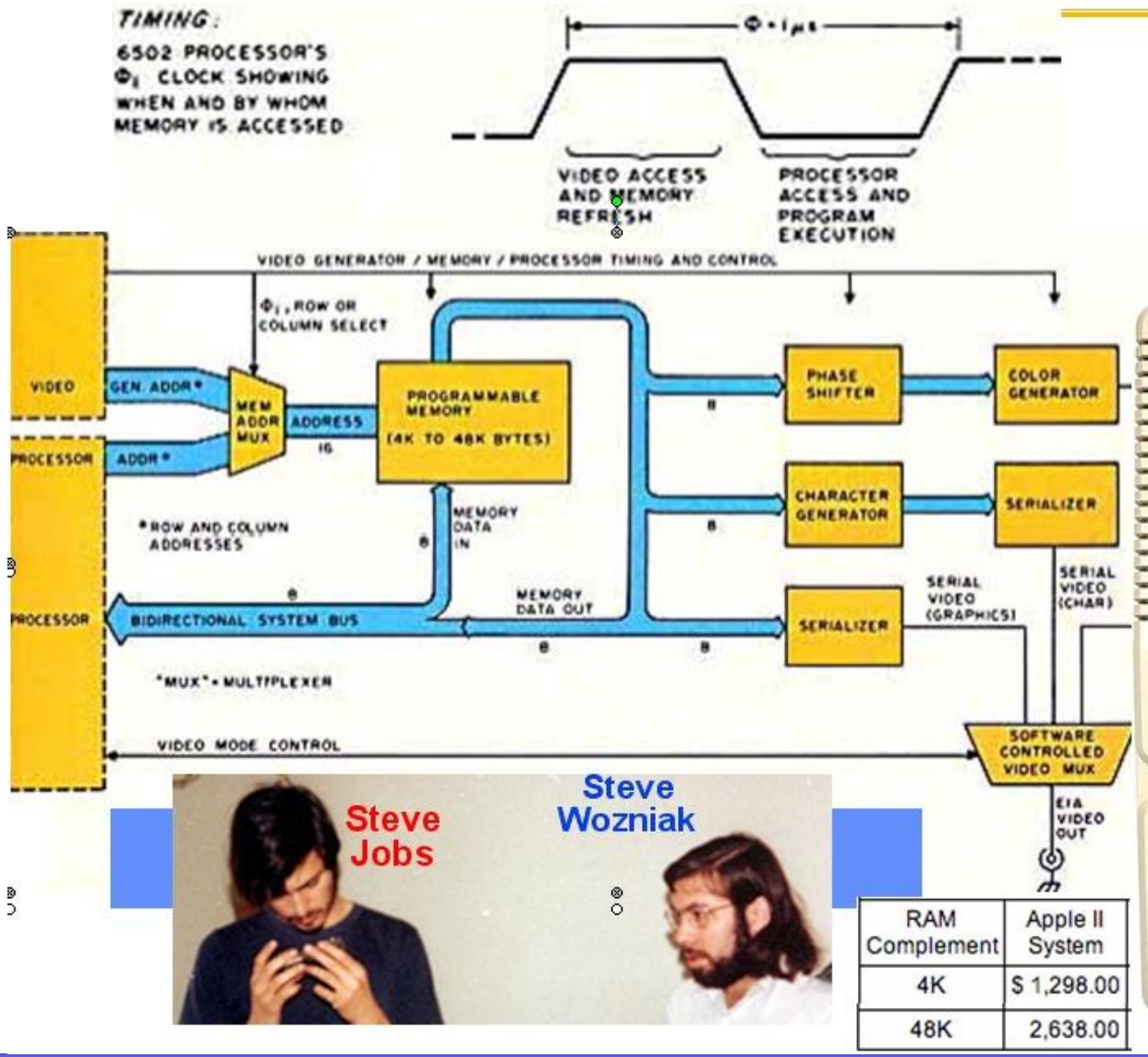
✓处理器主频超过了存储器；

✓随着指令集日趋庞大，编译器设计复杂度成为必须考虑的因素；

微处理器主频变化


10,000
1000
100
10
1
Clock rate (MHz)
Intel Pentium4 Xeon
3200 MHz in 2003
Intel Nehalem Xeon
3330 MHz in 2010
Intel Pentium III
1000 MHz in 2000
Digital Alpha 21164A
500 MHz in 1996
Digital Alpha 21064
150 MHz in 1992
MIPS M2000
25 MHz in 1989
Sun-4 SPARC
16.7 MHz in 1986
Digital VAX-11/780
5 MHz in 1978
15%/year
40%/year
1%/year
1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012

1977年：DRAM比microprocessors更快



RAM Complement	Apple II System
4K	$ 1,298.00
48K	2,638.00

Apple][(1977)

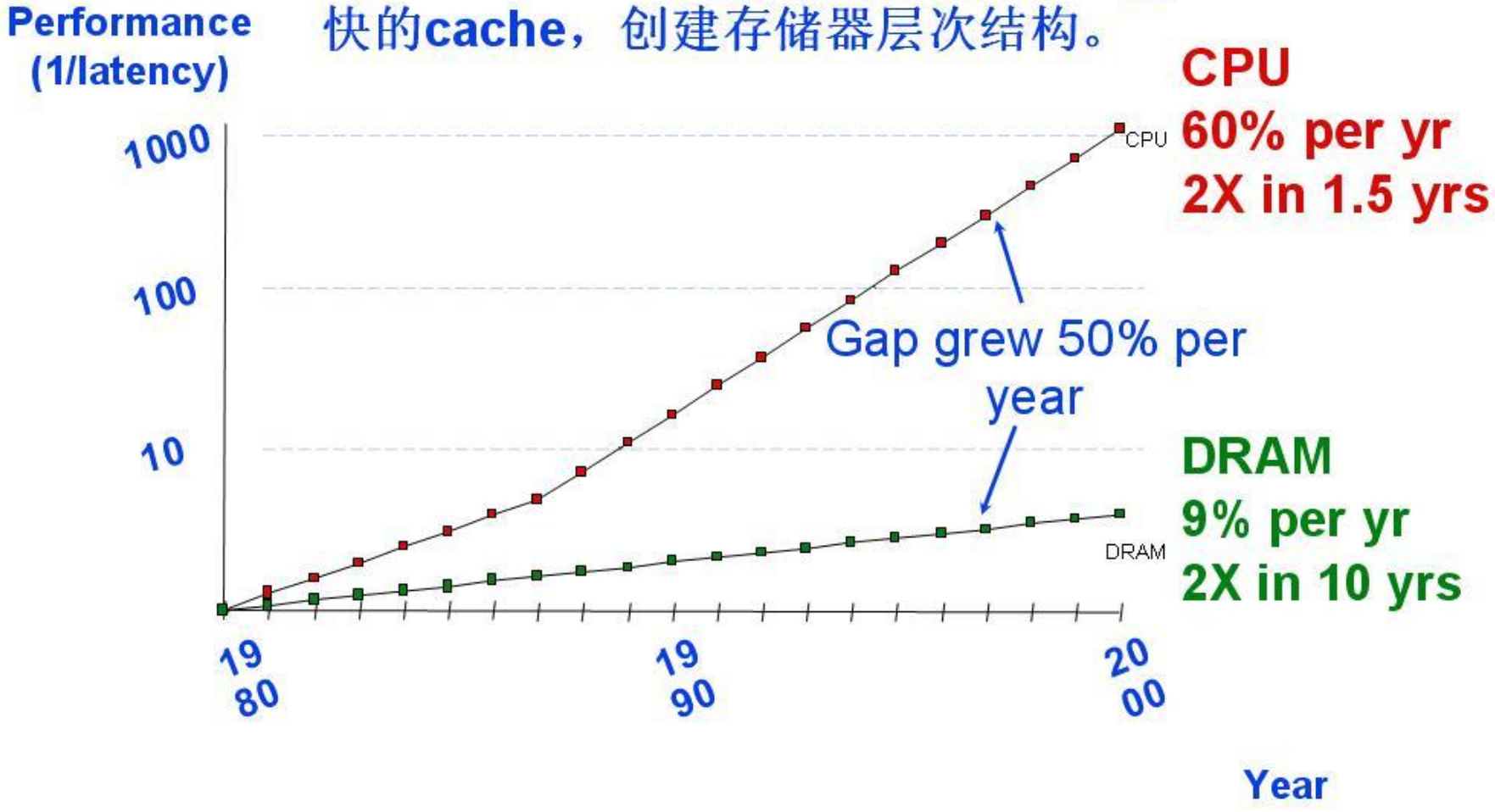
CPU: 1000 ns

DRAM: 400 ns

1980年后，CPU超过了DRAM的速度 ...

Q. 架构师如何解决这一差距？

A. 在CPU和DRAM之间，放上更小、更快的cache，创建存储器层次结构。



问题

- RISC诞生的原因？

✓集成电路工艺进步，通用寄存器(latch, Flip-Flop) 使用限制大幅减小；

✓处理器主频超过了存储器；

✓随着指令集日趋庞大，编译器设计复杂度成为必须考虑的因素；

获得一个操作数可以有多么困难？

- Addressing modes specify how to obtain the operands
  - Absolute LW rt, 10000

    use immediate value as address
  - Register Indirect: LW rt, (r_{base})

    use GPR[r_{base}] as address
  - Displaced or based: LW rt, offset(r_{base})

    use offset+GPR[r_{base}] as address
  - Indexed: LW rt, (r_{base}, r_{index})

    use GPR[r_{base}]+GPR[r_{index}] as address
  - Memory Indirect LW rt ((r_{base}))

    use value at M[GPR[r_{base}]] as address
  - Auto inc/decrement LW Rt, (r_{base})

    use GRP[r_{base}] as address, but inc. or dec. GPR[r_{base}] each time

CISC vs RISC

二者都试图在体系结构、操作运行、软件硬件、编译时间和运行时间等诸多因素中做出某种平衡，但方法不同：

- 指令系统：
  - RISC设计者把主要精力放在那些经常使用的指令上（2-8原则），尽量使它们具有简单高效的特色，其它采用指令组合完成。
  - CISC计算机的指令系统比较丰富，有专用指令来完成特定的功能。
- 存储器操作：
  - RISC力求控制简单化，限制存储器操作；CISC存储器操作指令繁复多样。

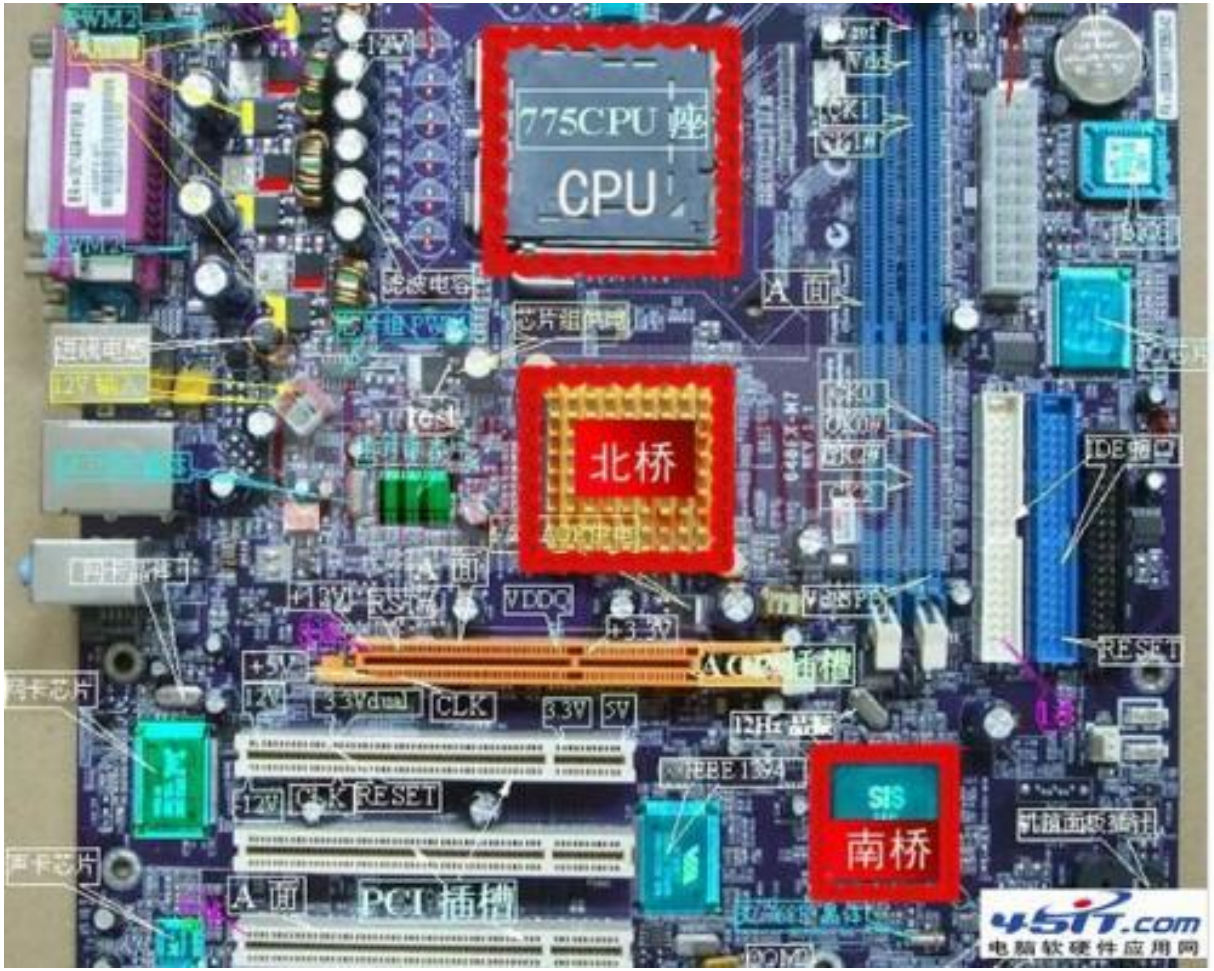
CISC vs RISC（续）

- 程序设计：
  - RISC汇编语言程序比较复杂，实现某些科学计算等任务需要指令比较多；CISC汇编语言程序相对简单。

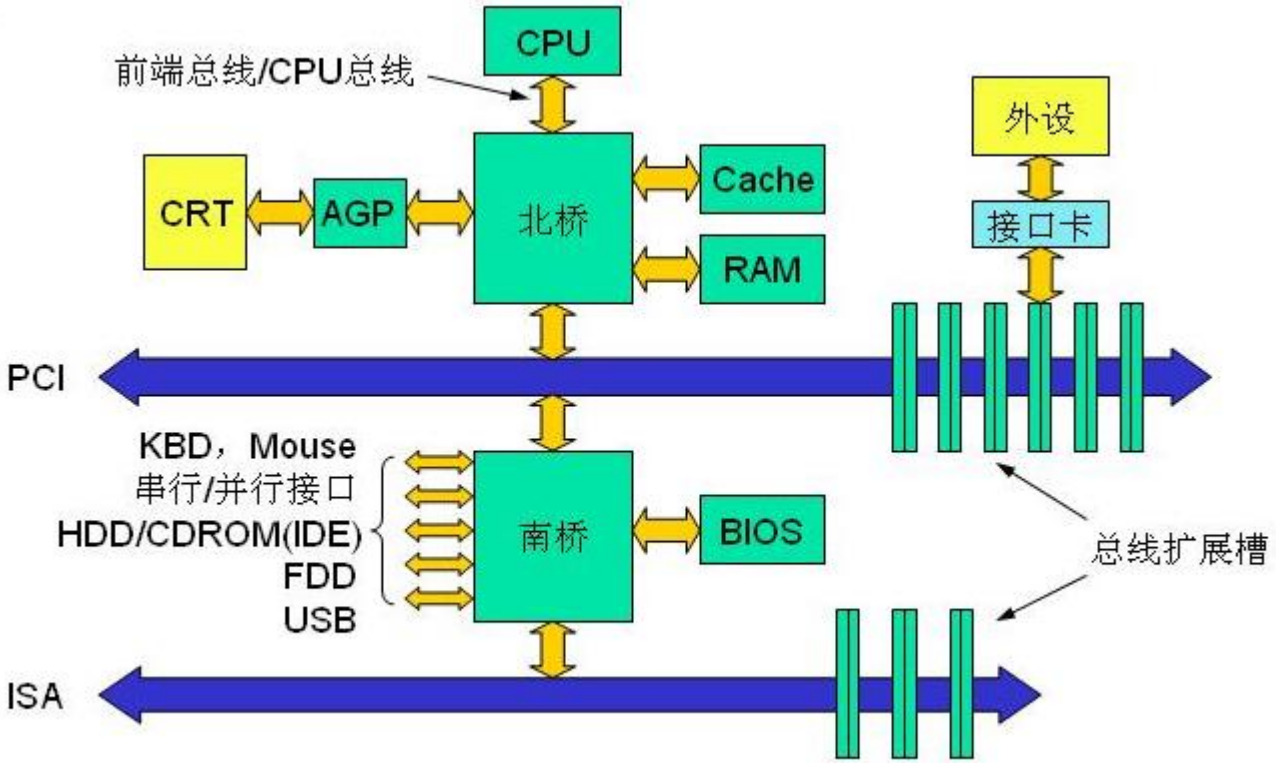
- CPU设计：
  - RISC微处理器结构简单，设计周期短，在面积、功耗、设计成本等方面具有优势；
  - CISC微处理器结构和数据通路复杂，一般来说设计周期较长。

传统计算机系统结构

CPU+北桥+南桥+内存+外设

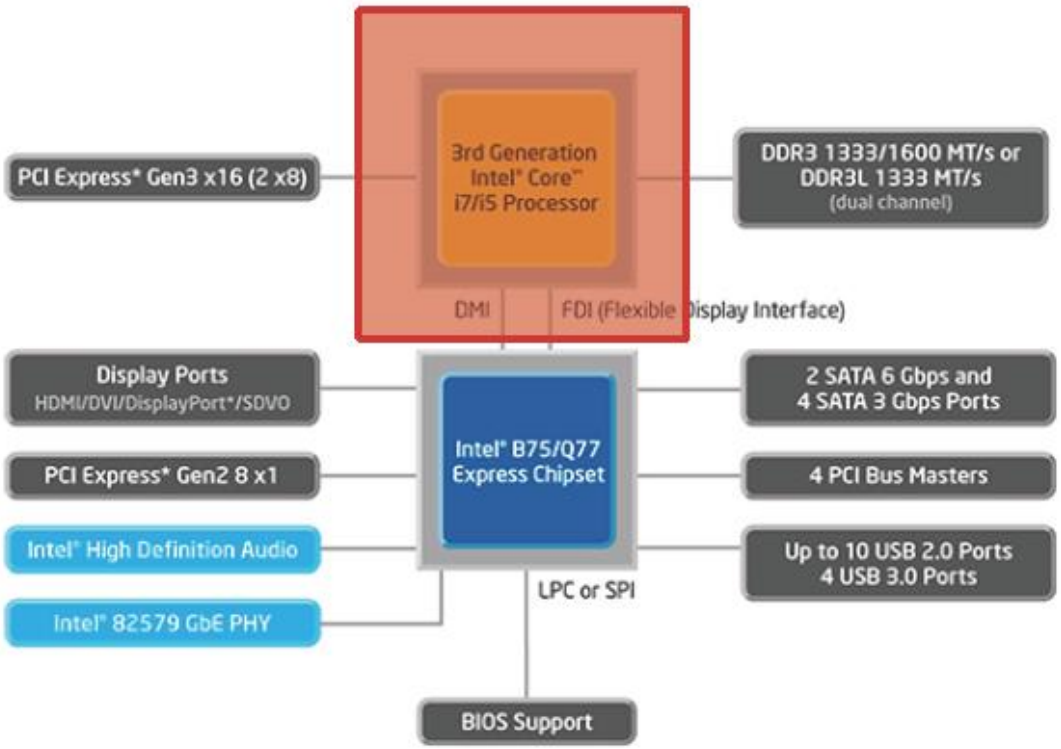


传统计算机系统结构：CPU+北桥+南桥



现代计算机系统结构

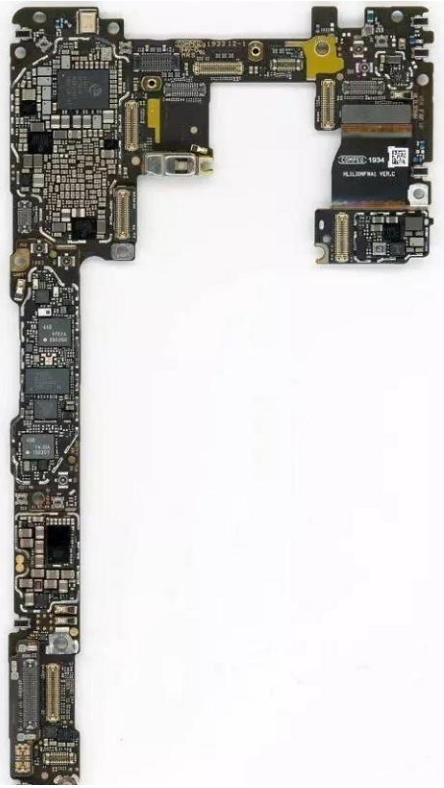
CPU+南桥+内存+外设（台式机、笔记本电脑）



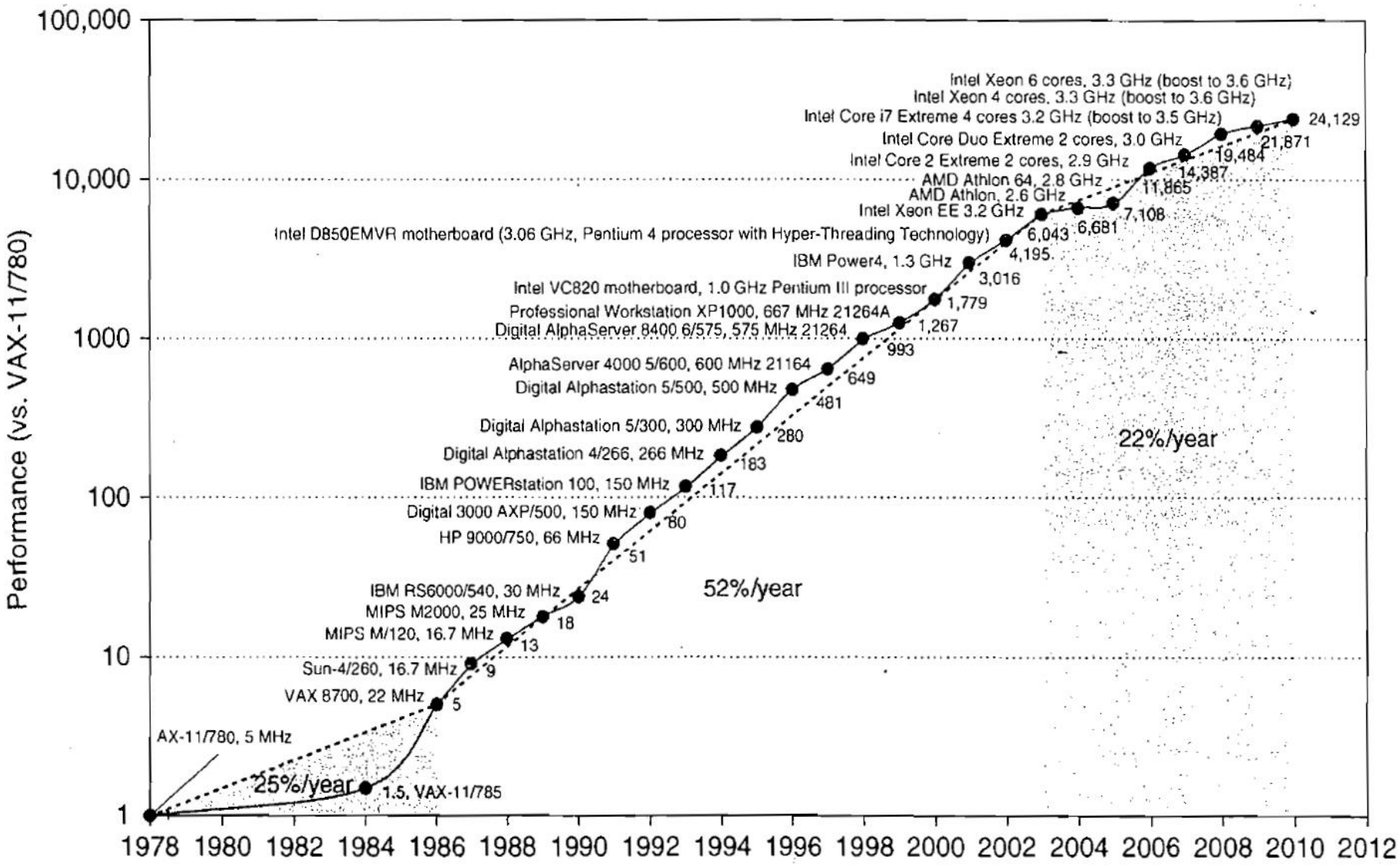
现代计算机系统结构

CPU+外设（手机架构）

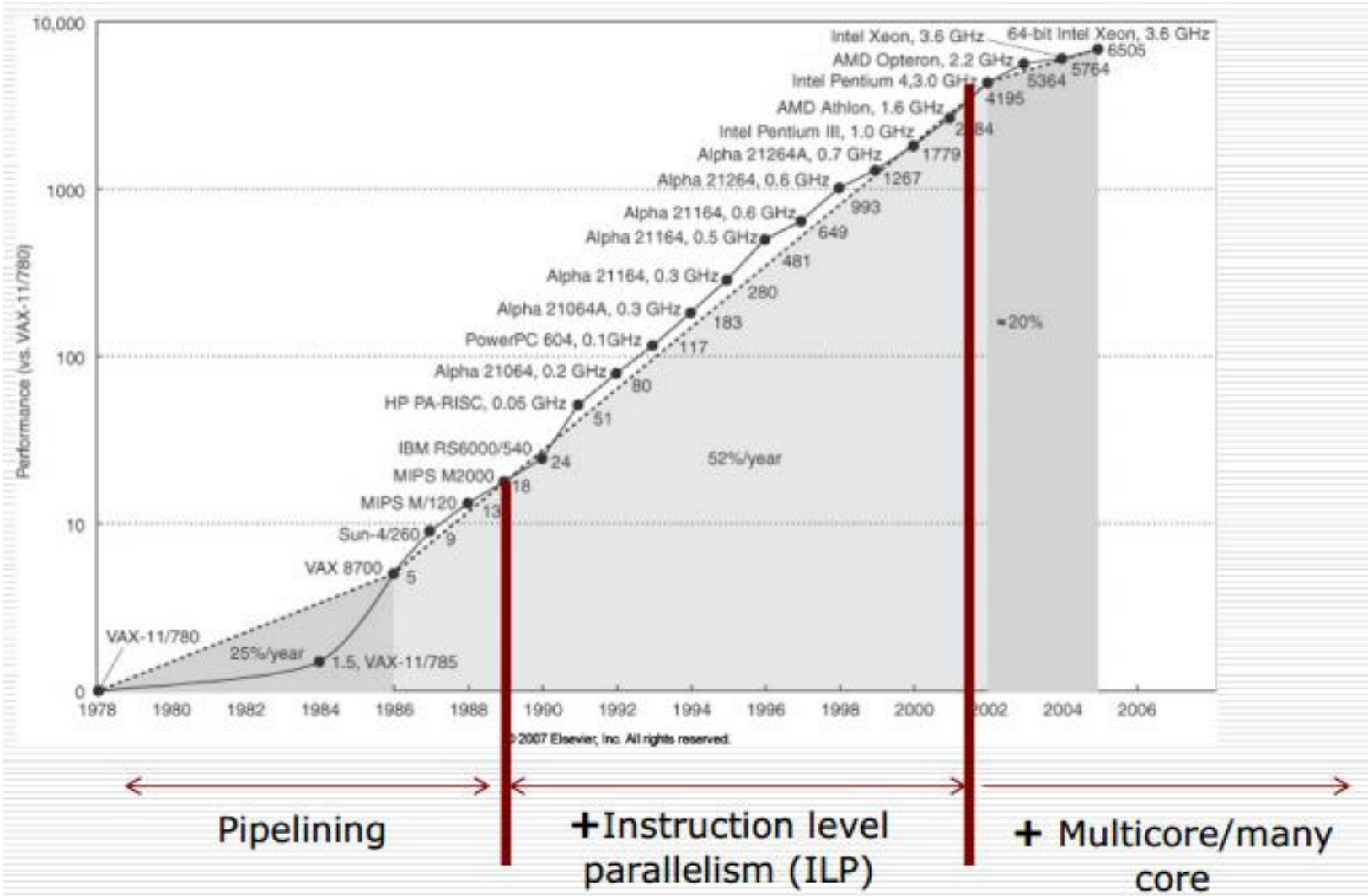
华为mate30 pro



1970年以来计算机性能的变化



CPU主要体系结构改进

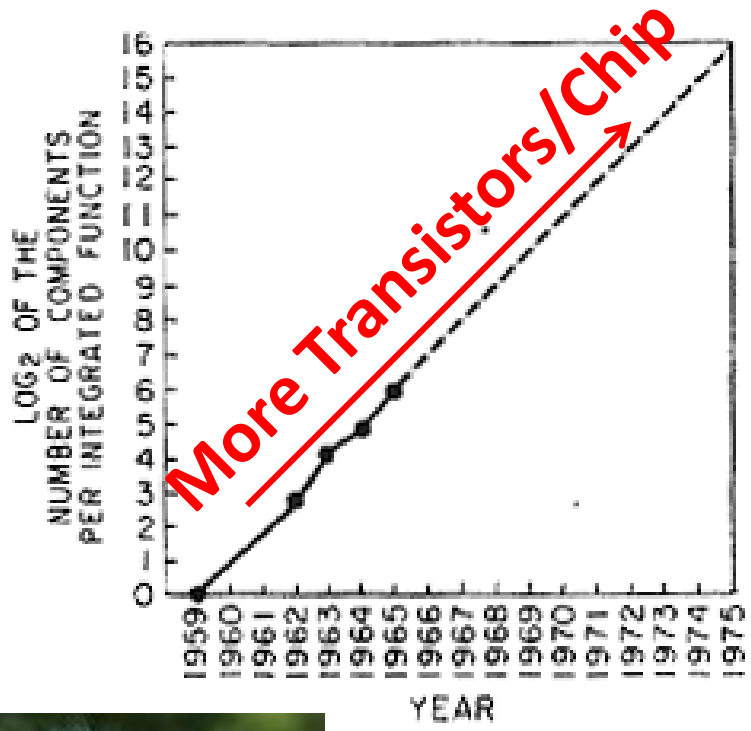


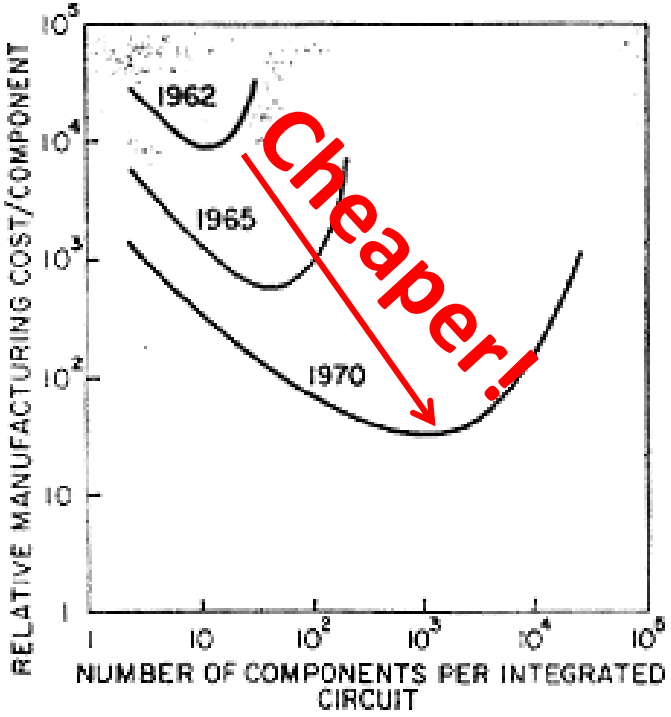
2021主流CPU性能比较图

- 桌面版
- 移动版

来源：“台式机电脑CPU性能最新天梯图”， https://new.qq.com/rain/a/20210603a0853300

Moore's Law





"Cramming more components onto integrated circuits", Gordon E. Moore, Electronics, 1965

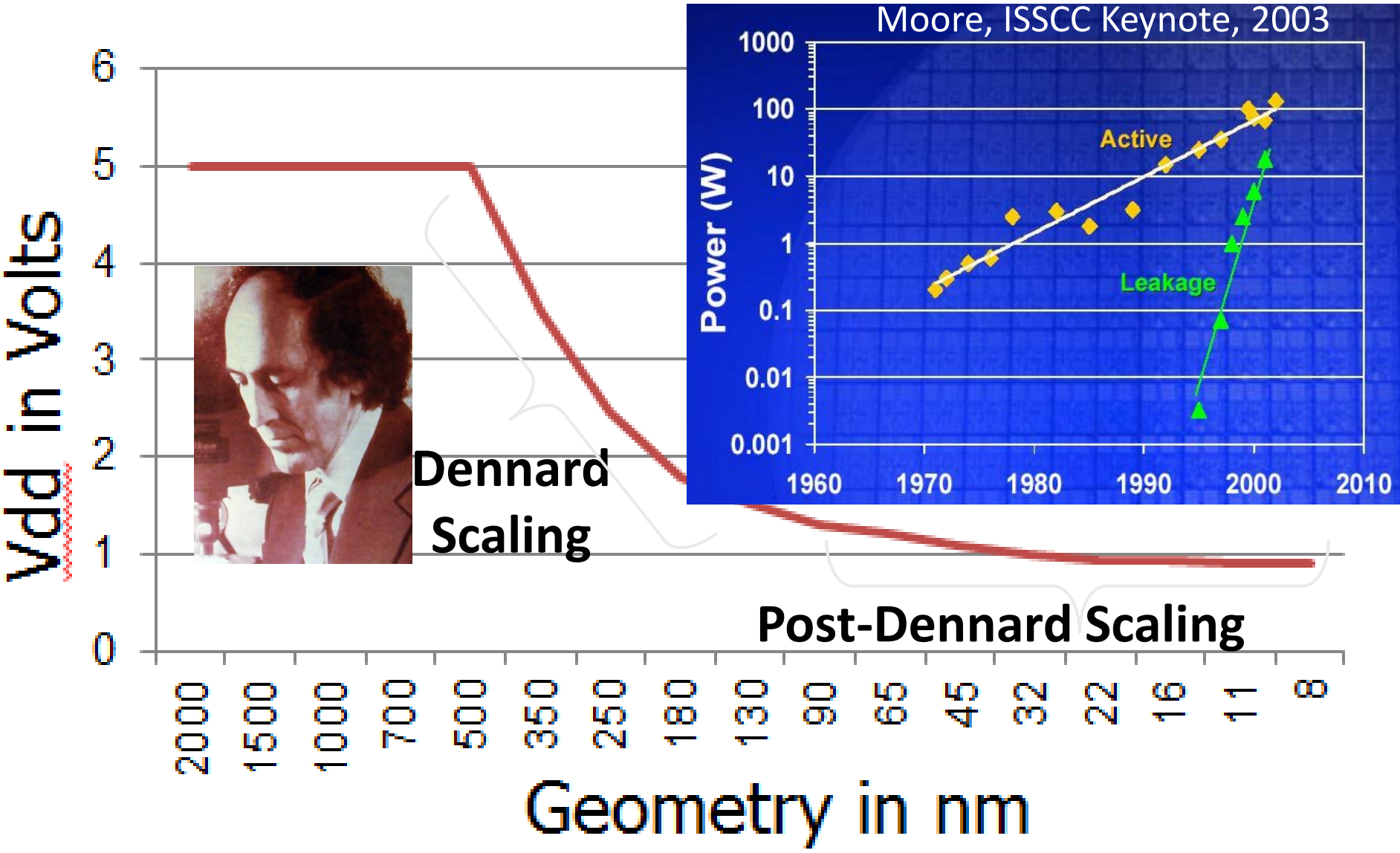
Moore’s Law 增速放缓

- 摩尔定律增速放缓

以DRAM为例：

  - 1977-1997: 1.46x/年
  - 1997-2017: 1.34x/年
  - 在过去的五年中，只有1.1x/年了
- 量子霍尔效应，硅工艺即将走向尽头
- 石墨烯具有特殊的量子霍尔效应

End of Dennard (Voltage) Scaling



Data courtesy S. Borkar/Intel 2011

Dennard Scaling的终结带来功耗问题

- Dennard Scaling的终结对于传统处理器设计方法来说是一个危机
- 功耗问题变得越来越重要（无论是移动设备还是云服务器）
  - 处理器散热已接近封装极限
  - 架构设计必须防止过热问题，尽可能改善能效比
- 然而，我们必须面对的问题是：从ILP到多核处理器，传统的通用处理器架构设计方法已达能效比的极限……

Dennard Scaling终结带来多核处理器架构设计挑战

- 在能效比约束下，单纯依靠处理器核的数量增加已经很难增加性能。
- 过多增加核数会导致Dark Silicon问题---芯片无法同时点亮所有的核，因为所有核同时开处理器功耗过大，会导致散热问题。
  - 22nm工艺的Intel E7-8890，24-core@2.2GHz，功耗极限165W；
  - 11nm工艺时，96-core@4.9GHz，如果功耗极限仍是165W，只可能同时用54/96核；如果极限是180W，则可支持59/96；如果极限是200W，则可支持65/96。

Domain Specific Architecuture (DSAs)

- 为了提高能效比，体系结构发展趋势开始回归到针对应用领域做优化的道路----即专用处理器架构（DSA）。
- 相对于通用（general purpose）处理器，DSA的优点是：针对性强，可以实现更好的能效比；缺点是：设计时要考虑专用领域的需求，需要设计者对该领域有比较深入的理解。
  - 例如：面向机器学习和AI应用的神经网络处理器、面向图像和虚拟现实的GPU、面向安全应用的加密处理器等。

摩尔定律的终结对于计算机领域还意味着什么？

“The easy ride of software is over.”

此前，很多软件从业者不关心硬件，但未来，软件行业将不得不更多了解硬件---软件从业者未来需要掌握必要的专有硬件接口，学习更多的domain-specific编程语言，才能在专有硬件上编写出能高效执行的程序。

计算机体系结构2030 -Mark D. Hill

- 摩尔定律放缓，Dennard scaling的衰落，面向领域的专用架构（DSA）兴起（机器学习作为核心负载，AI加速器）。
- 硬件设计大众化，使用高级语言（例如Chisel）加速开发周期，打造开源芯片设计的生态系统，包括开源芯片（例如RISC-V）和开源EDA设计流程，让硬件设计变得像软件设计那样敏捷、便宜和开放。
- 实现专用硬件单元，支持云计算等虚拟化技术。
- 三维(3D)集成技术
- 体系结构将“更接近物理层”
  - 模拟计算再次兴起，采用更有效信息编码，实现更密集地信息存储、更高效的计算、以及更低的功耗。易受噪声影响，需要新的容错方法来解决。
  - “新”材料（新型存储器件、碳纳米管、量子计算、超导逻辑、生物芯片工程等）

影响计算机系统结构的技术因素

- 半导体工艺发展
- 晶体管数目增加
- 存储技术发展

- 多级存储结构
- GPU
- 专用体系结构（AI加速器）

- 材料及存储技术
- 低功耗及封装技术
- 硬件安全

第一讲 引言

- 计算机发展简史
- 计算机系统结构的演变及发展趋势
- 高级计算机系统结构课程覆盖范围

课程内容　计算机系统结构>>ISAs + CA

- 计算机系统结构的演进
- 从单核处理器到多核处理器（NoC、缓存一致性）
- 从向量计算机到GPU
- 面向领域的计算机架构（ Domain-Specific Architectures）

  深度学习处理器（谷歌TPU、寒武纪……）
- 存储器的演变及存储系统设计（内存、硬盘---DRAM、SRAM、flash、RRAM、其它NVM等）

本课程内容（续）

- Moore’s Law、 Dennard (Voltage) Scaling
- 并行与可扩展性：Amdahl、Gustafson定律
- 指令集架构设计的规则和原理
- 计算机系统的性能设计与评估
- 功耗、可靠性、可用性
- 数据中心设计（与云计算技术）
- 开源芯片与开源设计流程
- 芯片安全与计算机体系结构