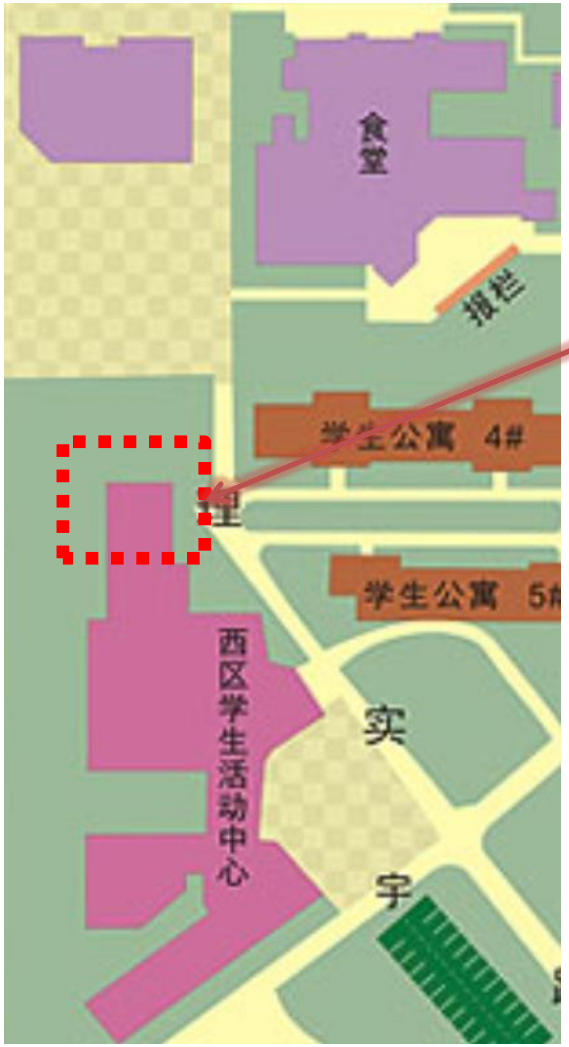
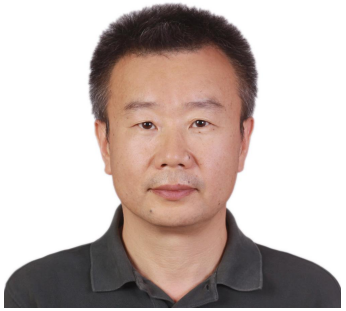
计算机体系结构

周学海
xhzhou@ustc.edu.cn
0551-63492149
中国科学技术大学

Welcome to ……



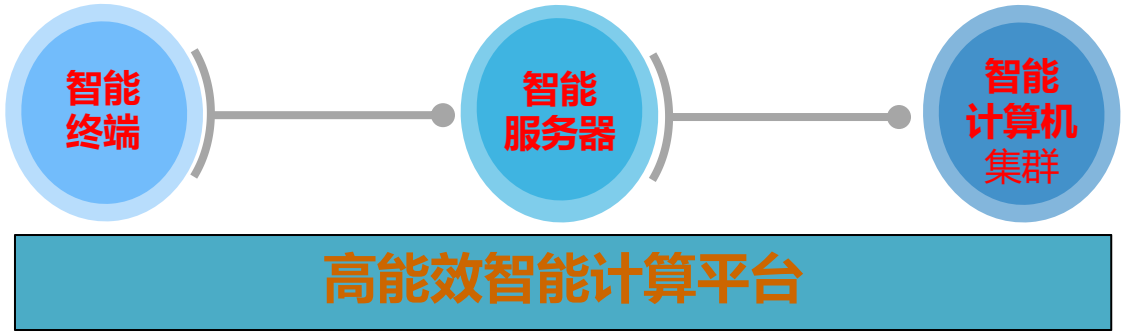
- **主讲：周学海(xhzhou@ustc.edu.cn)**
- **办公地点: 高能效智能计算实验室**
- **办公电话：0551-63492149**



面向嵌入式应用

服务于智慧城市与企业级智能应用的智能服务器

可提供公有云服务的智能计算机集群



- **嵌入式智能系统**
- **功耗、性能、小型化**
- **时间可预测性问题**
- **软硬件协同设计**

- **异构并行体系结构**
- **单节点系统的资源调度**
- **智能处理板卡**

- **任务管理**
- **资源调度**

Welcome to ……

- **助教：**

姓名	电子邮件
付薇	fw1219@mail.ustc.edu.cn

- **课程主页:**
  - bb.ustc.edu.cn
  - https://ca2023spring.github.io/
- **QQ群号：653702928**

Chapter1 量化设计与分析基础

- **1.1 课程简介**
  - 计算机体系结构的概念
- **1.2 体系结构发展历史、现状及趋势**
  - 现代计算机系统发展趋势
- **1.3 定量分析基础**

1.1 课程简介

什么是计算机体系结构

体系结构设计者的任务

本课程的基本要求

Computer

Computer system

Software + Hardware

Computer Hardware System

系统、组件、接口和环境

- **WebSter词典定义的系统: 众多不同的部分为了共同目标组合而成的复杂整体**
  - 用“组件”描述“不同部分”；用组件间的“互连”描述“整体”；用通过“接口”展示给外部“环境”的行为描述“共同的目标”

- **系统是互连在一起的组件的集合，在环境中通过其接口表现出预期的行为**
  - 现实世界可二分为：系统（讨论范围内的事物）和环境（范围外）
  - 人们通过研究系统组件的行为及组件间的互连来预测系统整体行为
  - 系统的组件可以是独立的系统，因此系统分解是一个递归的过程
    - 为了避免命名上的递归，系统设计师使用的表达同样含义的术语有：系统、子系统、组件、元素、对象、模块、子模块、部件、子部件等等
- **计算机系统：由软件控制的数字系统**

Definition of “System”

ISO/IEC/IEEE 15288:

是人为创造的、用于在所定义的环境中提供产品或服务，以造福于用户和其他利益相关者。

由相互作用的要素组合而成，以达到一个或多个目标。

The International Council on Systems Engineering (INCOSE) ：

完成所定义目标的要素、子系统或组件的集合，这些要素包括产品、过程、人员、信息、技术、设施、服务和其他支持要素。

NASA：两种定义

- ✓ “若干功能要素组合在一起，以产生满足需求的能力。要素包括为达到目标所需的所有硬件，软件，设备，设施，人员，流程和程序。
- ✓ 完成操作功能的**最终产品**，以及为操作最终产品提供生命周期支持服务的**使能产品**。

In Summary: 一个系统是完成特定目标的整体，它由相互作用的（要素、子系统或组件）部分组成。

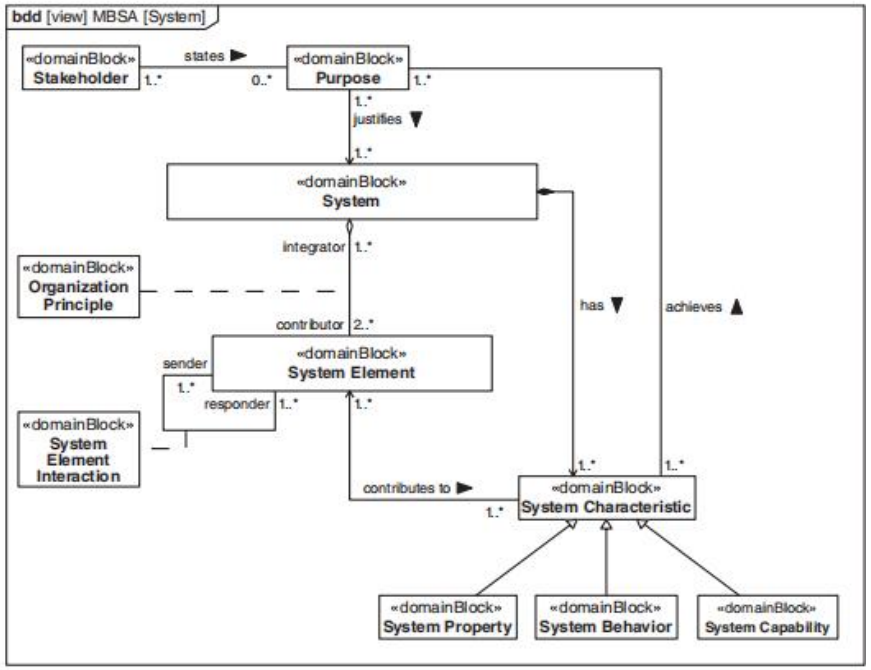


Figure 4.1. Definition of “System”.

计算机系统与其他系统

- **计算机系统与其他系统是否相同?**
- **复杂系统存在的相同特征:**
  - 存在的共性问题：突生属性、传播效应、扩展不相称、权衡。
    - 复杂系统：空间站设计、摩天大楼建设、计算机系统设计等
  - 应对**复杂性**的通用方法：**模块化、抽象化、层次化、分级化**
    - 模块化：分而治之
    - 抽象化：接口与内部细节分离；规范和实现分离。支持抽象化的模块化也称为“功能模块化”
    - 层次化：分层组织系统，新的完整的功能集（上层）基于另一完整的功能集（下层），以减少模块间的互连
    - 分级化：类似树形结构的组织方式，以减少模块间的互连。层次化组织可以看成一维的分级化
- **显著区别:**
  - 计算机系统的复杂性不受物理定律限制
  - 计算机系统的发展速度是前所未有的快
- **计算机系统处理复杂性的额外方法和原则：迭代法、保持简单**
  - 面向迭代的设计：设计不可能一次成功，务必使其易于修改
  - 全面实施简单性：才能对所做之事了然于胸

System Architecture

- **系统架构是系统组件的组织结构、组件之间的关系、系统与环境的关系、以及指导其设计和进化的原则。**
- **系统与系统架构之间的关系**
  - 每个系统都有一个系统架构，每个系统架构属于一个系统。
  - **系统架构包括关于系统组织、设计和系统演化的一些原则。**

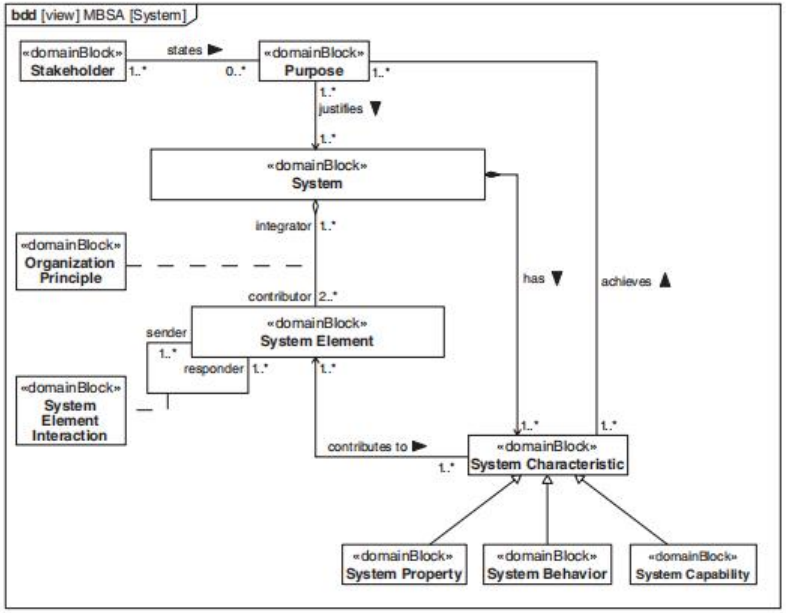


Figure 4.1. Definition of "System".

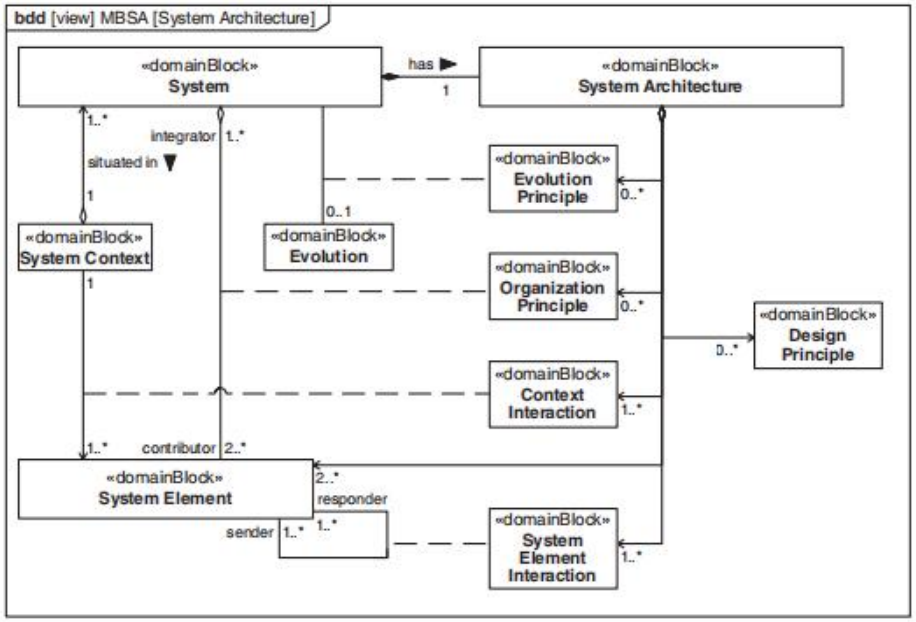


Figure 4.2. Definition of "System Architecture."

Weilkiens T , Lamm J G , Roth S , et al. Model-Based System Architecture[M]. John Wiley & Sons, Inc, 2015.

What is Computer Architecture?

计算机系统：
由软件控制的数字系统

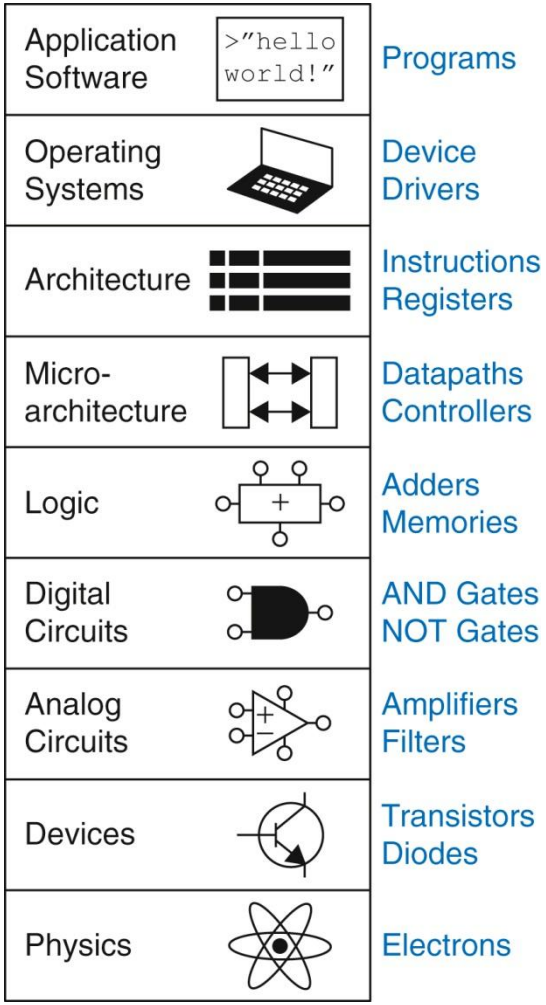
What is CA ?

CA 关注哪些问题?

计算机体系结构：指计算机（硬件子系统）系统的**抽象表示**，基于这些抽象表示使得我们可以**更好地**使用可用的制造（工艺）技术**实现（信息处理）硬件系统**，**高效地设计与实现（信息处理）软件系统**

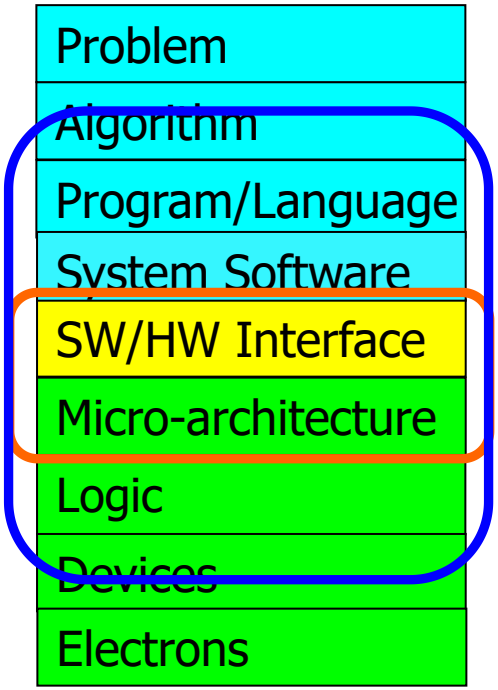
现代计算机系统的抽象层次

- Application
- Algorithm
- Programming Language
- Operating System/Virtual Machine
- Instruction Set Architecture (ISA)
- Microarchitecture
- Gates/Register-Transfer Level (RTL)
- Circuits
- Devices
- Physics



The Transformation Hierarchy

Computer Architecture (expanded view)



Computer Architecture (narrow view)

Computer

↓

Computer system

↙ ↘

Software + Hardware

↓

Computer Hardware System

计算机体系结构研究范畴

早期的定义：Instruction-Set Architecture

程序员可见的计算系统的属性。包括：概念性的（抽象）结构和功能行为。不包括：数据流和控制流的组织、逻辑设计以及物理实现。

– Amdahl, Blaauw, and Brooks, 1964

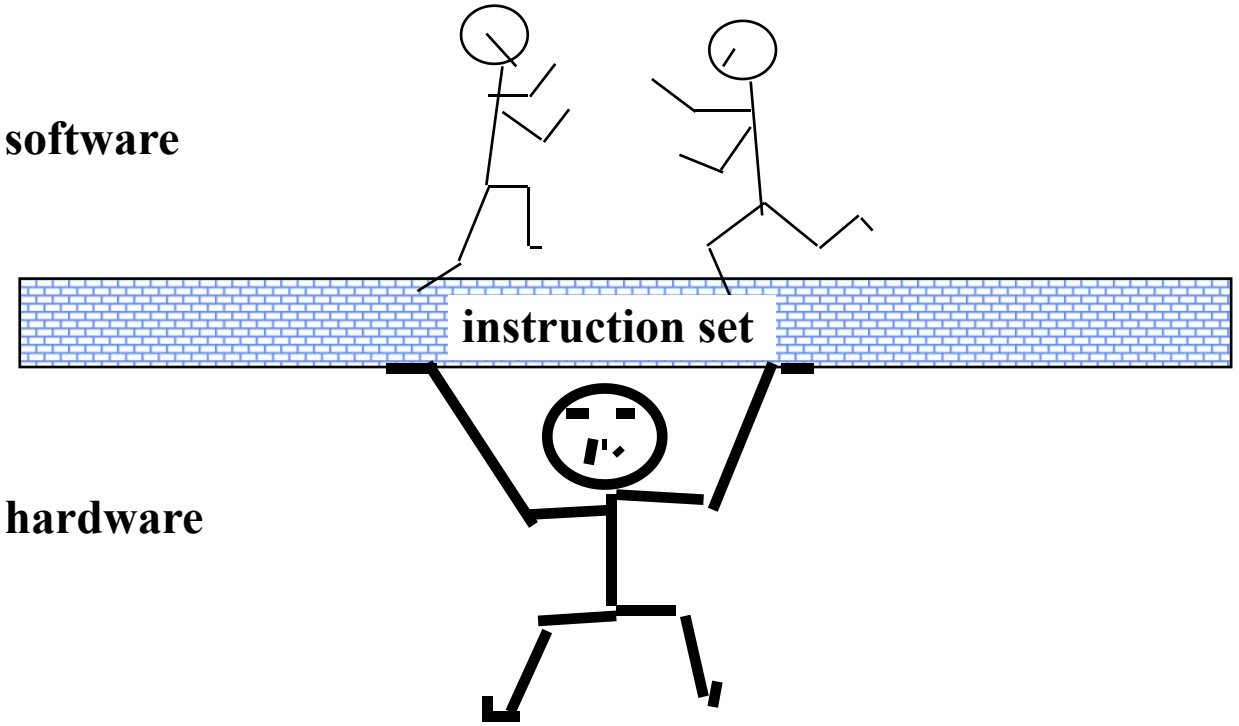
狭隘的观点（narrow view）：

包括：软件设计者与硬件设备设计者（VLSI）之间的中间层（ISA），以及微体系结构

扩展的观点（expanded view）：

包括：算法层、程序/语言层、系统软件层、ISA层、微体系结构、逻辑电路层、以及器件层

ISA: a Critical Interface



Problem
Algorithm
Program/Language
System Software
SW/HW Interface
Micro-architecture
Logic
Devices
Electrons

- **ISA：是对计算机系统的底层硬件子系统的结构和功能行为的抽象**
  - 存储器：记录计算中所用数据和指令的系统组件
  - 处理器（解释器）执行构成计算的具体动作
  - 输入输出（组件间的通信链路）实现了计算机与外界的通信链路

ISA需说明的主要内容

- Memory addressing
- Addressing modes
- Types and sizes of operands
- Operations
- Control flow instructions
- Encoding an ISA
- ……

- **优秀的ISA所具有的特征**
  - 可持续用于很多代机器上(portability)
  - 可以适用于多个领域 (generality)
  - 对上层提供方便的功能（convenient functionality)
  - 可以由下层有效地实现（efficient implementation)
  - ……

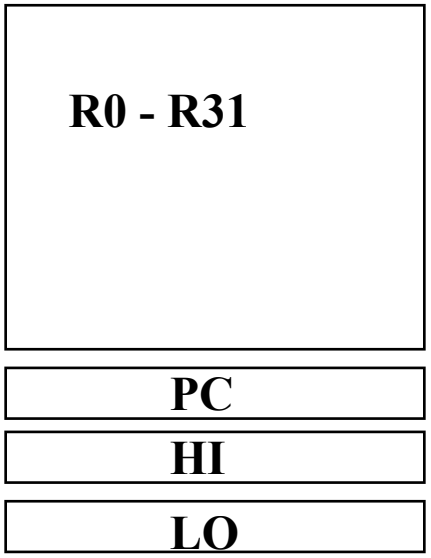
指令集结构举例

- **通用指令集架构**
  - Digital Alpha(v1, v3) 1992-97
  - HP PA-RISC (v1.1, v2.0) 1986-96
  - Sun Sparc(v8, v9) 1987-95
  - SGI MIPS (MIPS I, II, III, IV, V) 1986-96
  - Intel(8086,80286,80386, 80486,Pentium, MMX, ...) 1978-96
  - RISC-V now
  - LoongArch now
- **领域专用的指令集架构 （DSA)**
  - TI DSP Instruction Set
  - Cambricon: ISA for Neural Networks
  - ……

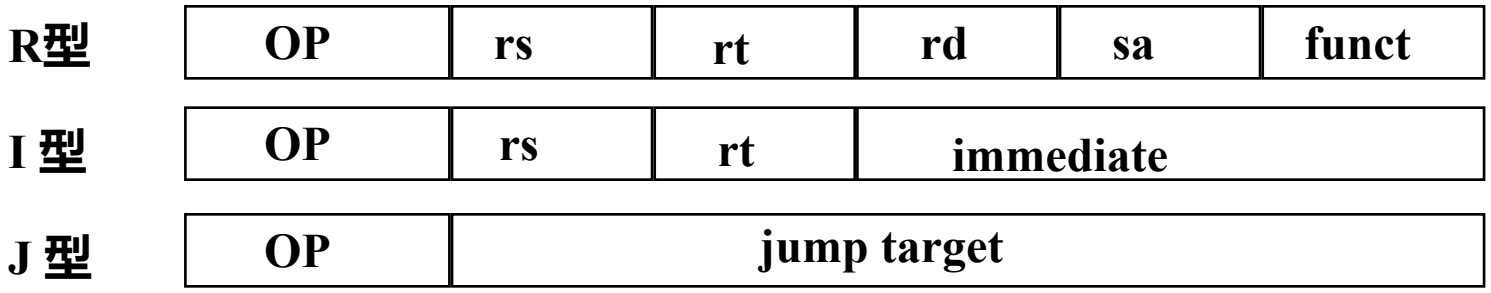
MIPS R3000 Instruction Set Architecture (Summary)

- **指令类型**
  - Load/Store
  - Computational
  - Jump and Branch
  - Floating Point
    - coprocessor
  - Memory Management
  - Special

Registers



3 种指令格式: all 32 bits wide

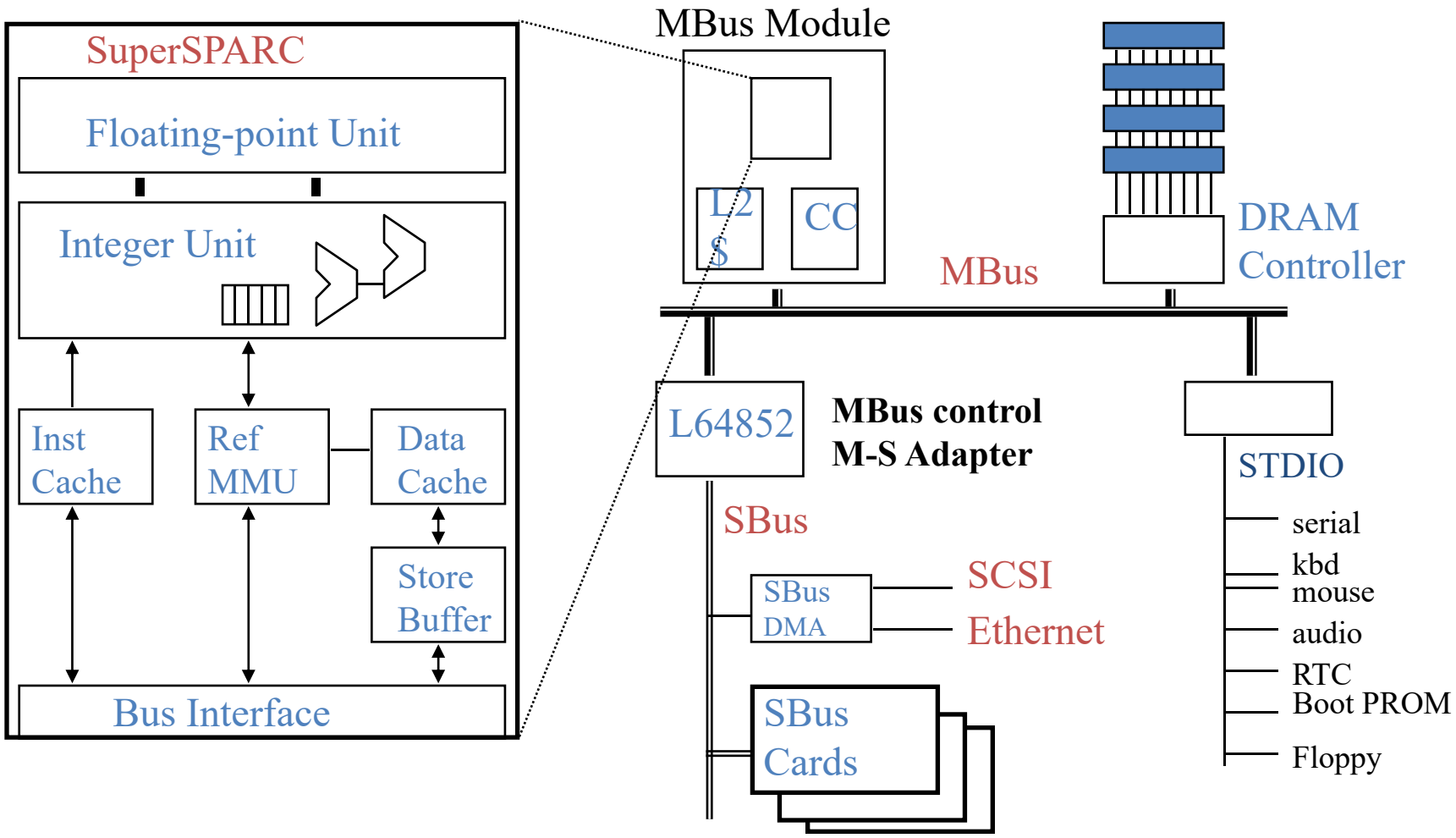


计算机组成与实现

- **计算机组成（Computer Organization or Microarchitecture): ISA的逻辑实现**
  - 物理机器级中的数据流和控制流的组成以及逻辑设计等
- **计算机实现（Computer Implementation):计算机组成的物理实现**
  - CPU，MEMORY等的物理结构，器件的集成度、速度，模块、插件、底板的划分与连接、信号传输、电源、冷却及整机装配技术等
- **例如**
  - 确定指令系统中是否有乘法指令 (Architecture)
  - 确定用加法器实现乘法 还是用专门的乘法器实现（Organization)
  - 器件的选定及所用的微组装技术 （Implementation)

Example Organization

- TI SuperSPARC™ TMS390Z50 in Sun SPARCstation20



指令集架构 vs.微体系结构

- **Architecture / Instruction Set Architecture (ISA)**
  - Class of ISA: register-memory or register-register architectures
  - Programmer visible state (Register and Memory)
  - Addressing Modes: how memory addresses are computed
  - Data types and sizes for integer and floating-point operands
  - Instructions, encoding, and operation
  - Exception and Interrupt semantics

- **Microarchitecture / Organization**
  - Tradeoffs on how to implement the ISA for speed, energy, cost
  - Pipeline width and depth, cache size, peak power, bus width, execution order, etc

计算机体系结构研究范畴

早期的定义：Instruction-Set Architecture

程序员可见的计算系统的属性。包括：概念性的结构和功能行为。不包括：数据流和控制流的组织、逻辑设计以及物理实现。

– Amdahl, Blaauw, and Brooks, 1964

狭隘的观点（narrow view）：

包括：软件设计者与硬件设备设计者（VLSI）之间的中间层（ISA），以及微体系结构

扩展的观点（expanded view）：

包括：算法层、程序/语言层、系统软件层、ISA层、微体系结构、逻辑电路层、以及器件层

1.1 引言

什么是计算机体系结构

体系结构设计者的任务

本课程的基本要求

Computer Architecture

- **计算机体系结构（narrow）：关注硬件系统的组织和设计**
  - 设计、选择和连接硬件组件，以及设计硬件/软件接口
  - 以构建满足功能、性能、能耗、成本和其他具体目标的计算系统的科学和艺术。
- **体系结构设计目标：与应用场景密切相关**
  - E.g., 在工作负载X, Y, Z上的最高性能
  - E.g., 最长的电池寿命，可装进口袋，成本< X
  - E.g., 在所有已知工作负载中以最佳性能/成本比获得最佳平均性能…
  - 设计一台超级计算机与设计一部智能手机设计目标是不同的，但是，**许多基本原则是相似的**

Different Platforms, Different Goals

Source: http://www.sia-online.org (semiconductor industry association)

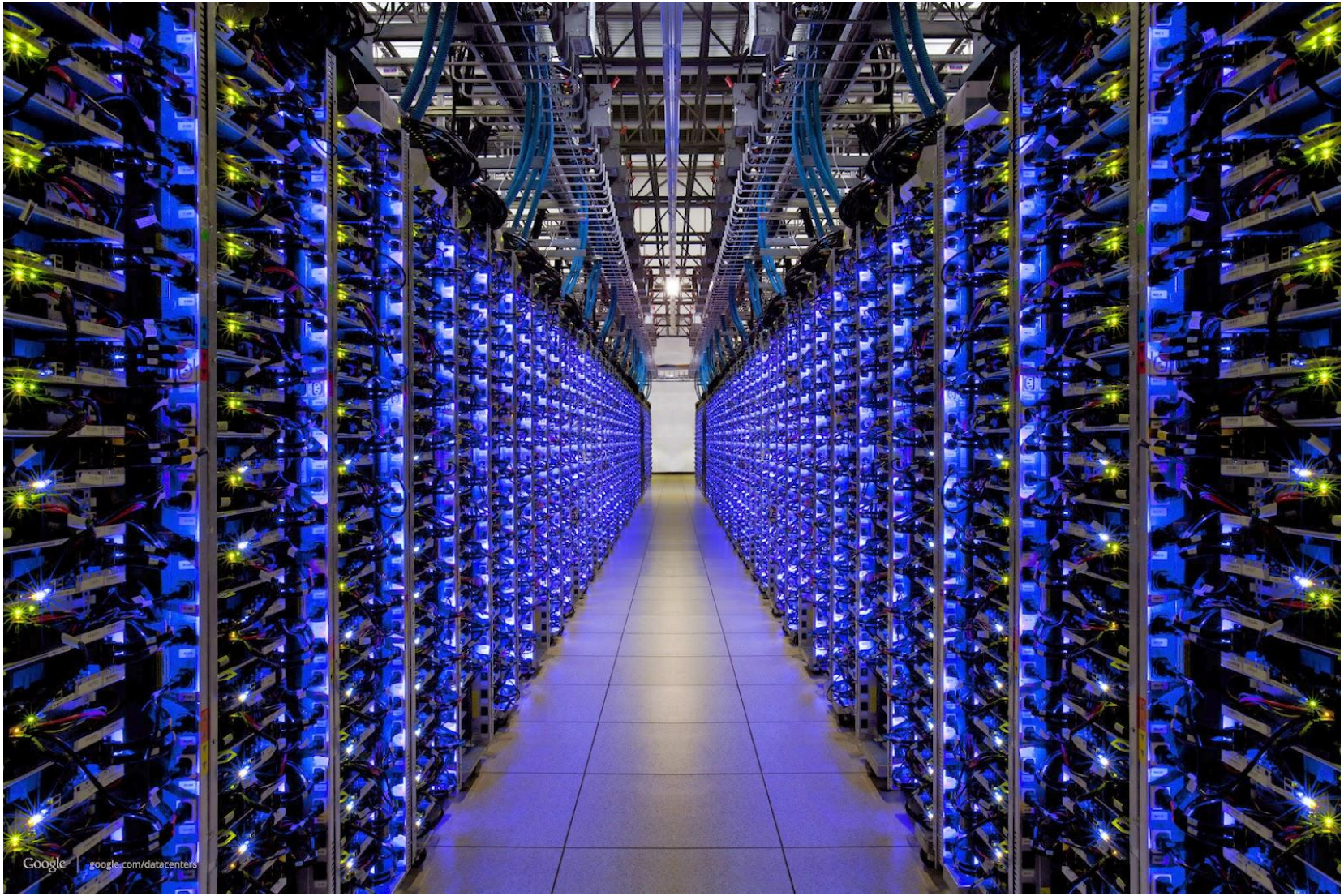
Different Platforms, Different Goals

Source: https://iq.intel.com/5-awesome-uses-for-drone-technology/

Different Platforms, Different Goals

Source: http://sm.pcmag.com/pcmag_uk/photo/g/google-self-driving-car-the-guts/google-self-driving-car-the-guts_dwx8.jpg

Different Platforms, Different Goals



Source: http://datacentervoice.com/wp-content/uploads/2015/10/data-center.jpg

Different Platforms, Different Goals

Source: https://fossbytes.com/wp-content/uploads/2015/06/Supercomputer-TIANHE2-china.jpg

Different Platforms, Different Goals

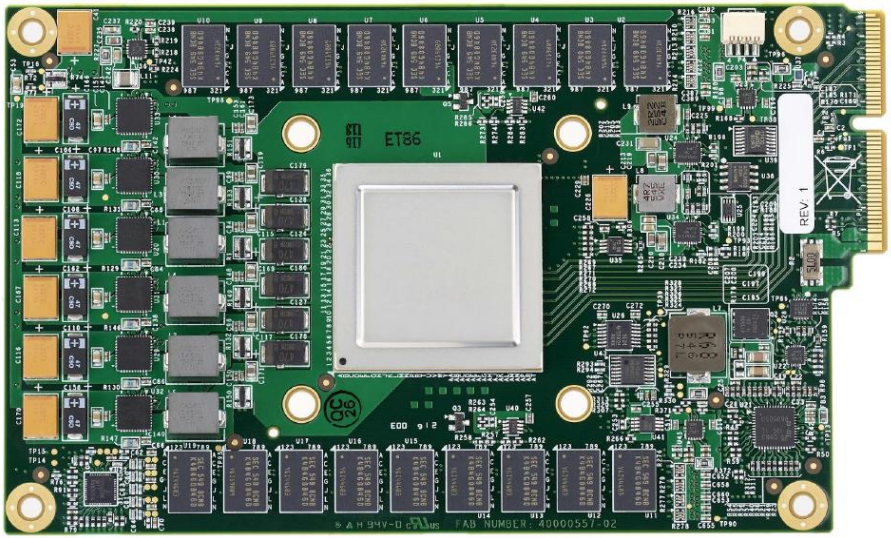


Figure 3. TPU Printed Circuit Board. It can be inserted in the slot for an SATA disk in a server, but the card uses PCIe Gen3 x16.

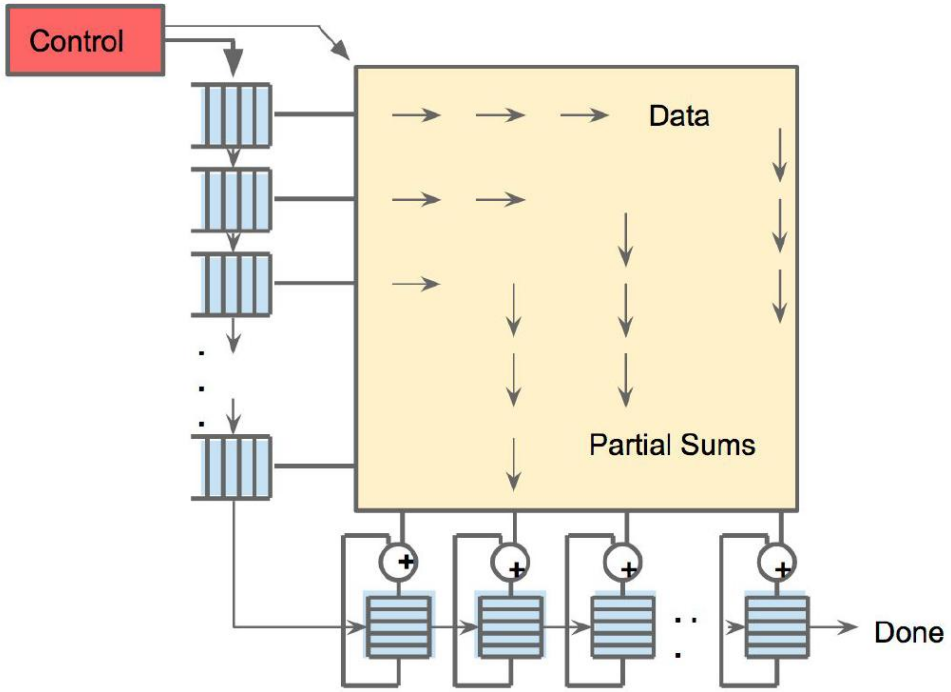
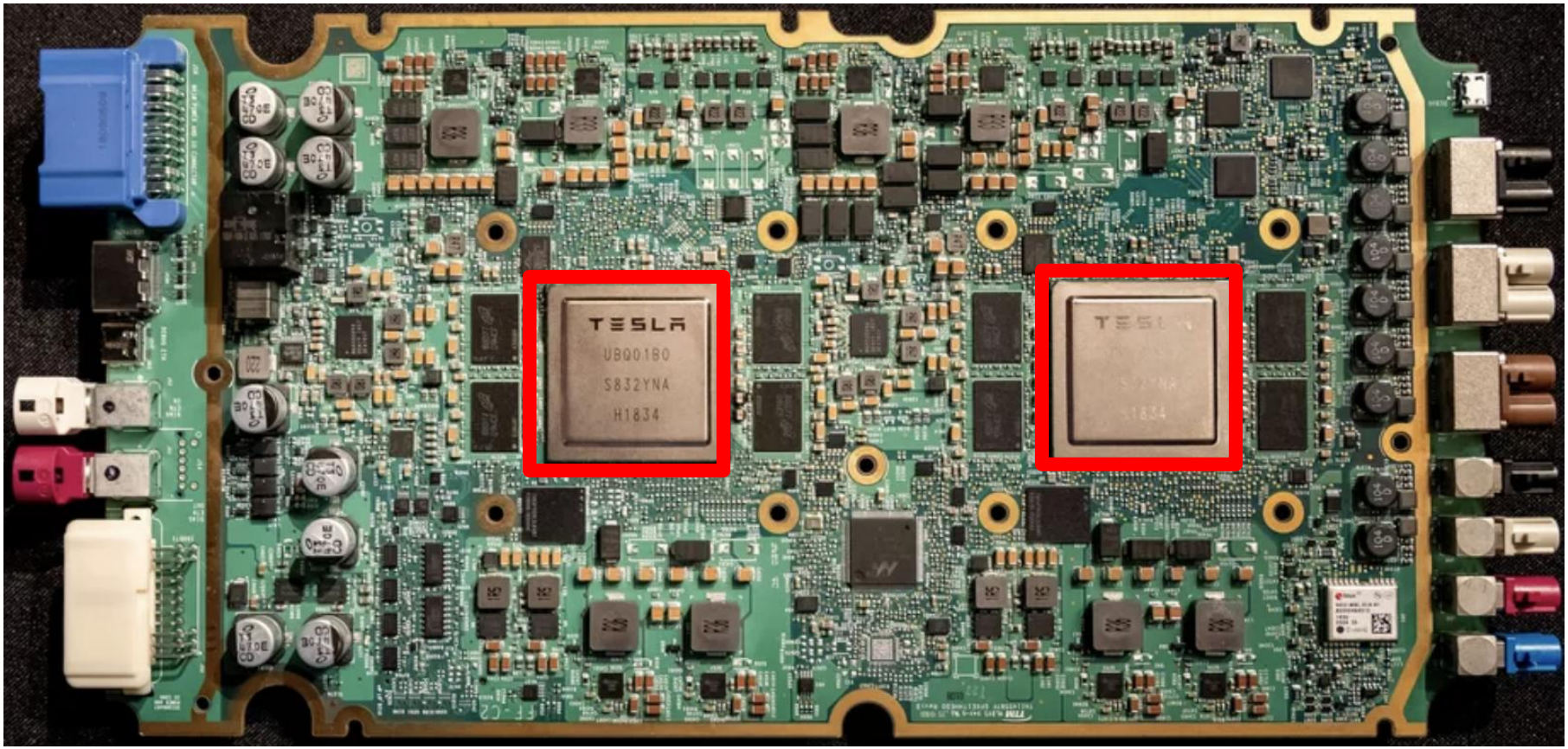


Figure 4. Systolic data flow of the Matrix Multiply Unit. Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.

Jouppi et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit", ISCA 2017.

Different Platforms, Different Goals

- ML accelerator: 260 mm², 6 billion transistors, 600 GFLOPS GPU, 12 ARM 2.2 GHz CPUs.
- Two redundant chips for better safety.

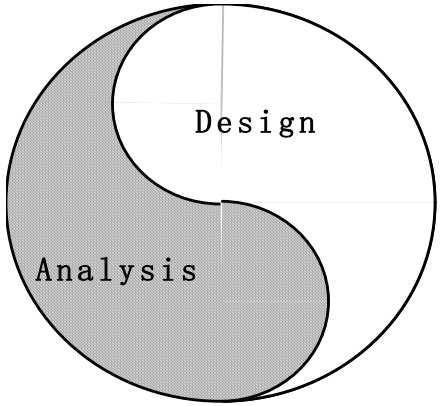


https://youtu.be/Ucp0TTmvqOE?t=4236

计算机体系结构设计者的任务

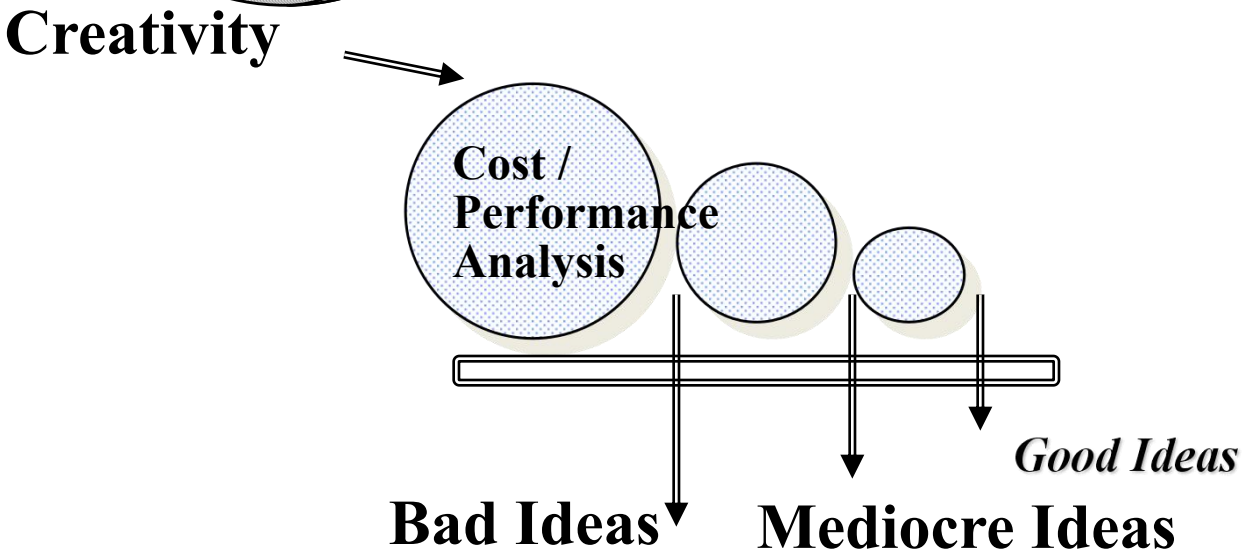
- **设计和实现不同档次的计算机系统**
  - Understand software demands
  - Understand technology trends
  - Understand architecture trends
  - Understand economics of computer systems
- **最大化性能、能效、可编程性等指标**
  - 在一定的技术和成本的限制下
- **体系结构现状:**
  - 现代微处理器大多为多核处理器
  - 单芯片中通常集成多个处理器核心
  - 每个处理器核心支持多线程执行

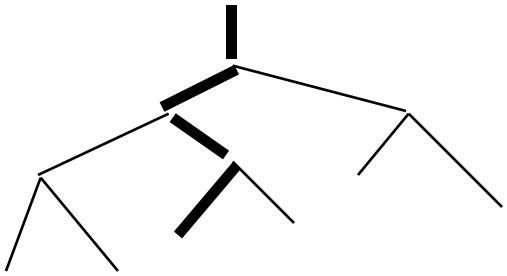
计算机体系结构设计过程



体系结构设计是循环渐进的过程:

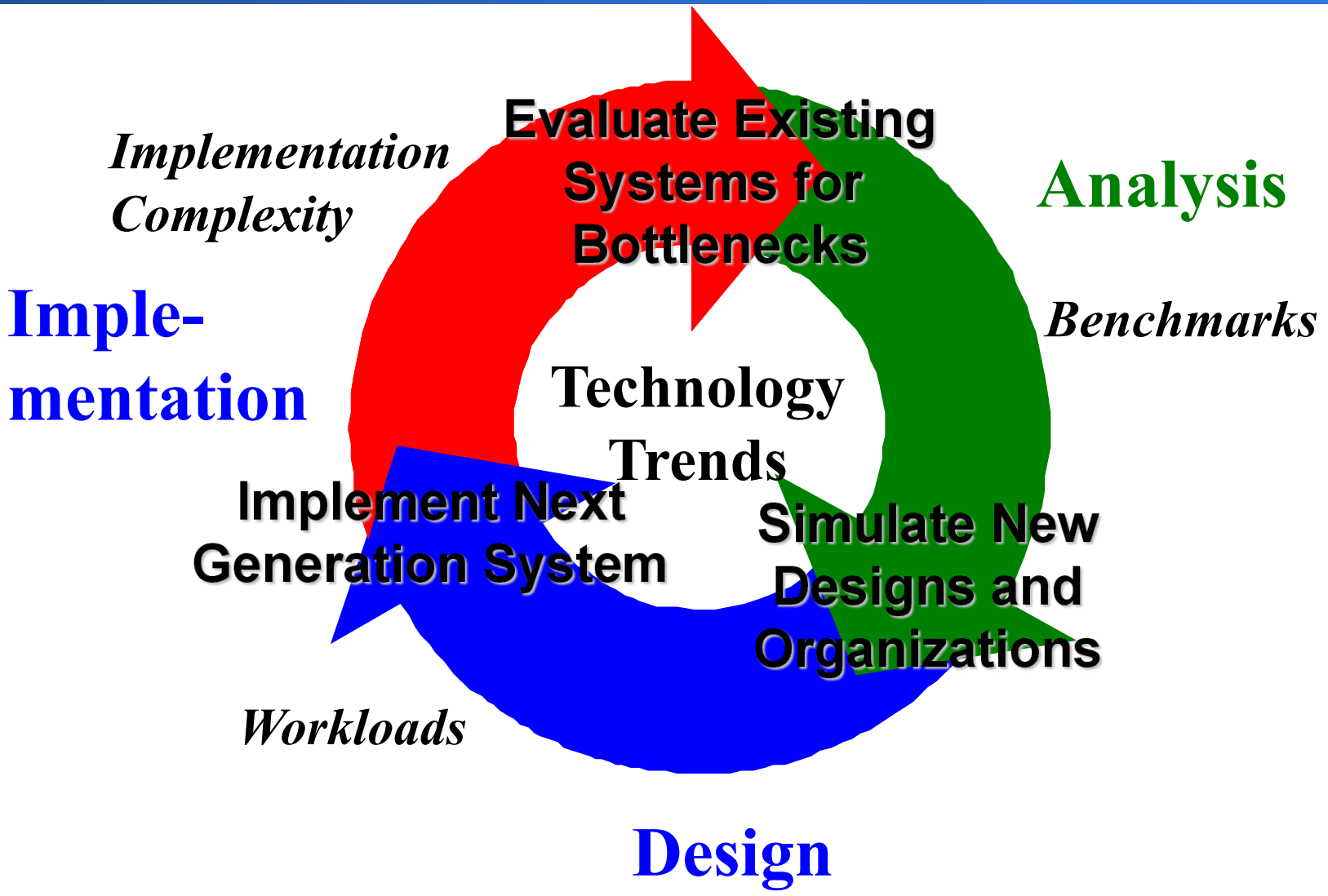
- Search the possible design space
- Make selections
- Evaluate the selections made



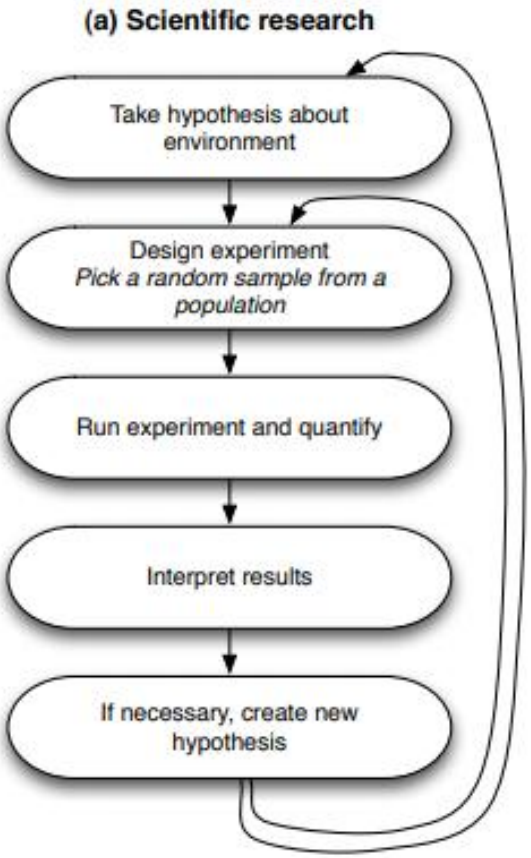


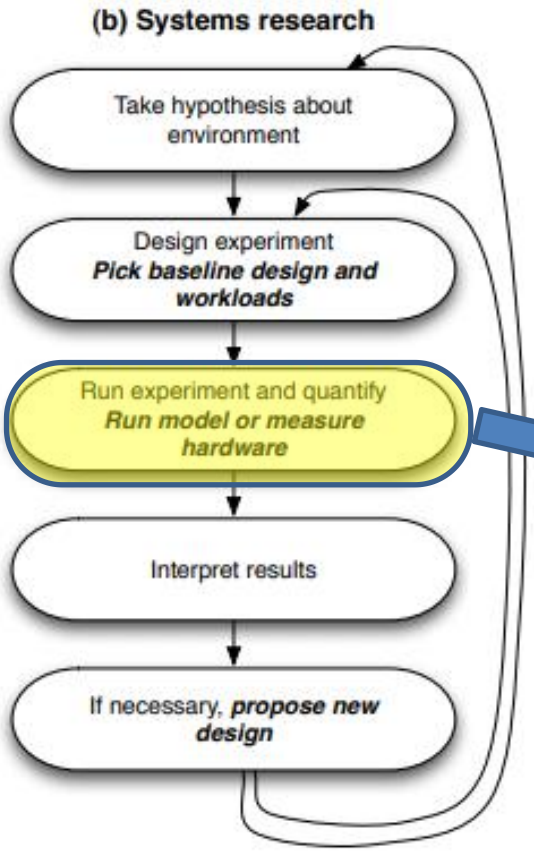
Good measurement tools are required to accurately evaluate the selection.

计算机工程方法学


Evaluate Existing Systems for Bottlenecks
Analysis
Benchmarks
Technology Trends
Simulate New Designs and Organizations
Design
Workloads
Implement Next Generation System
Implementation Complexity
Imple-mentation

Computer systems research/engineering





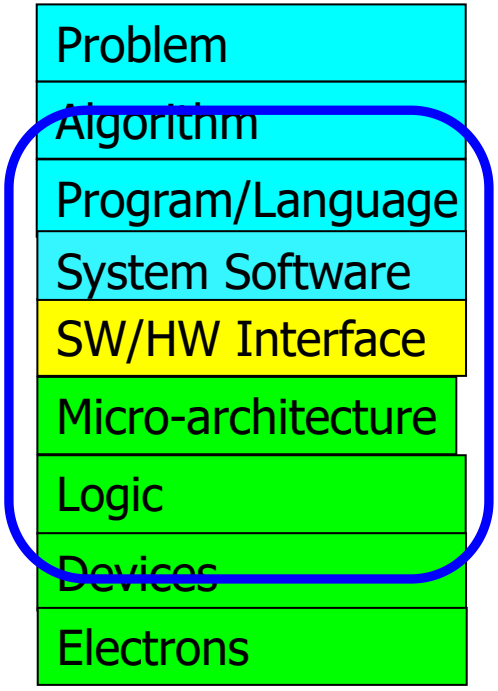
From Computer Architecture Performance Evaluation Methods by Lieven Eeckhout

Computer architecture simulation!

Axiom

为达到最高的能效和性能:

我们必须以扩展的眼光看待计算机架构



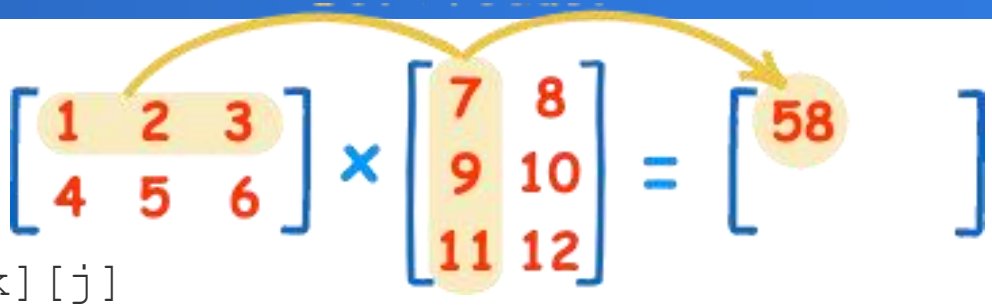
算法层到器件层的跨层次协同设计

在设计目标范围内尽可能地定制化

Software & Hardware Optimizations

Multiplying Two 4096-by-4096 Matrices

```
for i in xrange(4096):
  for j in xrange(4096):
    for k in xrange(4096):
      C[i][j] += A[i][k] * B[k][j]
```



Implementation	Running time (s)	Absolute speedup
Python	25,552.48	1x
Java	2,372.68	11x
C	542.67	47x
Parallel loops	69.80	366x
Parallel divide and conquer	3.80	6,727x
plus vectorization	1.10	23,224x
plus AVX intrinsics	0.41	62,806x

Leiserson+, "There's plenty of room at the Top: What will drive computer performance after Moore's law?", Science, 2020

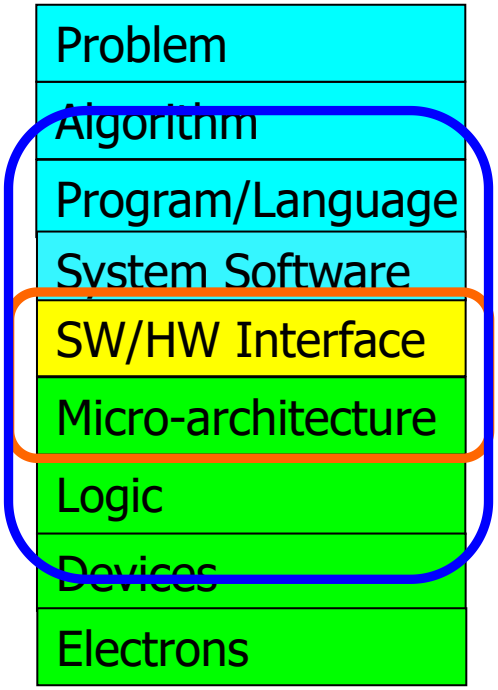
体系结构进步带来的益处

- **实现更好的系统：使计算机更快、更便宜、更小、更可靠…**
  - By exploiting advances and changes in underlying technology/circuits
- **算力的提高使新的应用成为可能**
  - Life-like 3D visualization 20 years ago? Virtual reality?
  - Self-driving cars?
  - Personalized genomics? Personalized medicine?
- **能够更好地解决实际问题**
  - Software innovation is built on trends and changes in computer architecture
    - > 50% performance improvement per year has enabled this innovation

为什么要学计算机体系结构

深入理解计算机体系结构:

- **更好地理解计算机系统的工作原理和设计原则**
- **开展体系结构研究与设计的基础**
  - 体系结构领域仍然存在许多挑战性问题
    - 机器学习成为计算机系统的核心负载
    - 应用的专用化与硬件系统的鸿沟 （特征适配）
    - 体系结构 “更接近物理层” 新的器件技术和电路设计技术
    - 云计算模式：利用规模化和虚拟化技术
    - 存储墙、功耗墙仍然存在
    - .....
- **更好地设计与实现操作系统、编译器**
  - 需要重新评估当前的假设和权衡
    - 例如: 当面临网络性能持续提升、并行系统、异构系统日益普遍
  - 现代计算机需要更好的编译器和更好的编程语言
- **更好地设计与实现应用程序**
  - 可更好地理解算法、数据结构和编程语言选择对性能的影响

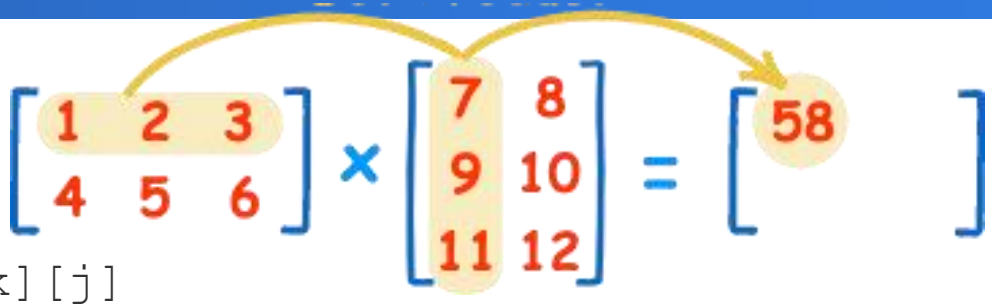


Computer Architecture (expanded view)

回顾：Software & Hardware Optimizations

Multiplying Two 4096-by-4096 Matrices

```
for i in xrange(4096):
  for j in xrange(4096):
    for k in xrange(4096):
      C[i][j] += A[i][k] * B[k][j]
```



Implementation	Running time (s)	Absolute speedup
Python	25,552.48	1x
Java	2,372.68	11x
C	542.67	47x
Parallel loops	69.80	366x
Parallel divide and conquer	3.80	6,727x
plus vectorization	1.10	23,224x
plus AVX intrinsics	0.41	62,806x

Leiserson+, "There's plenty of room at the Top: What will drive computer performance after Moore's law?", Science, 2020

1.1 引言

什么是计算机体系结构

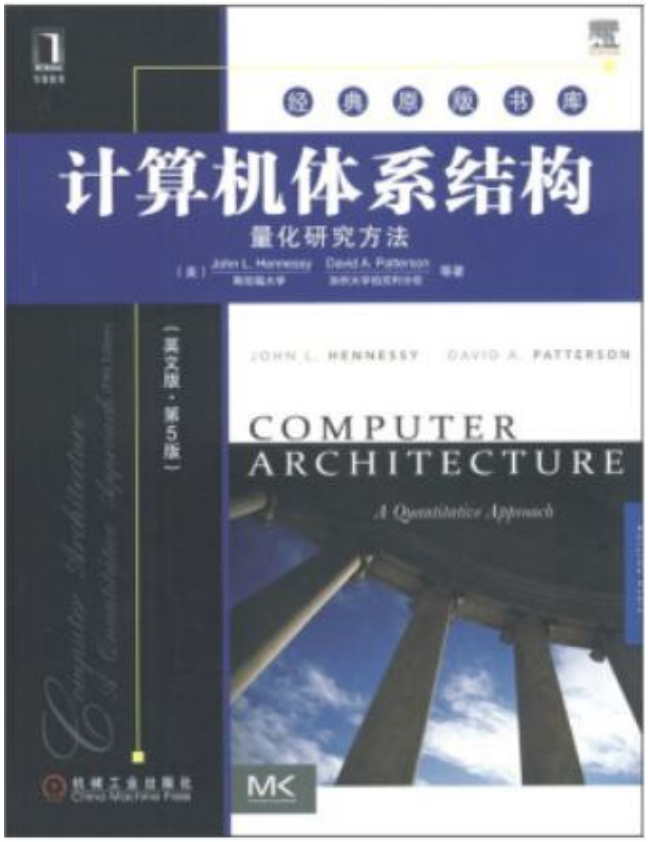
体系结构设计者的任务

本课程的基本要求

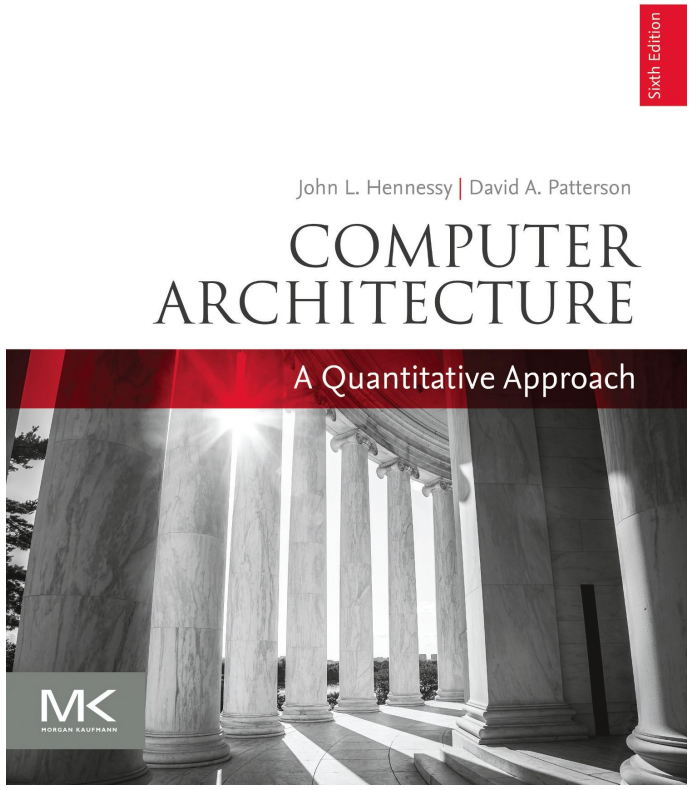
课程目标

- **掌握**系统定量分析的基本方法和技术
- **深入理解**提高CPU性能的基本方法
- **深入理解**存储系统的基本原理和优化方法
- **理解**数据级并行、指令级并行、线程级并行的基本原理和方法
- **初步理解**面向特定领域的处理器设计

教材与主要参考书

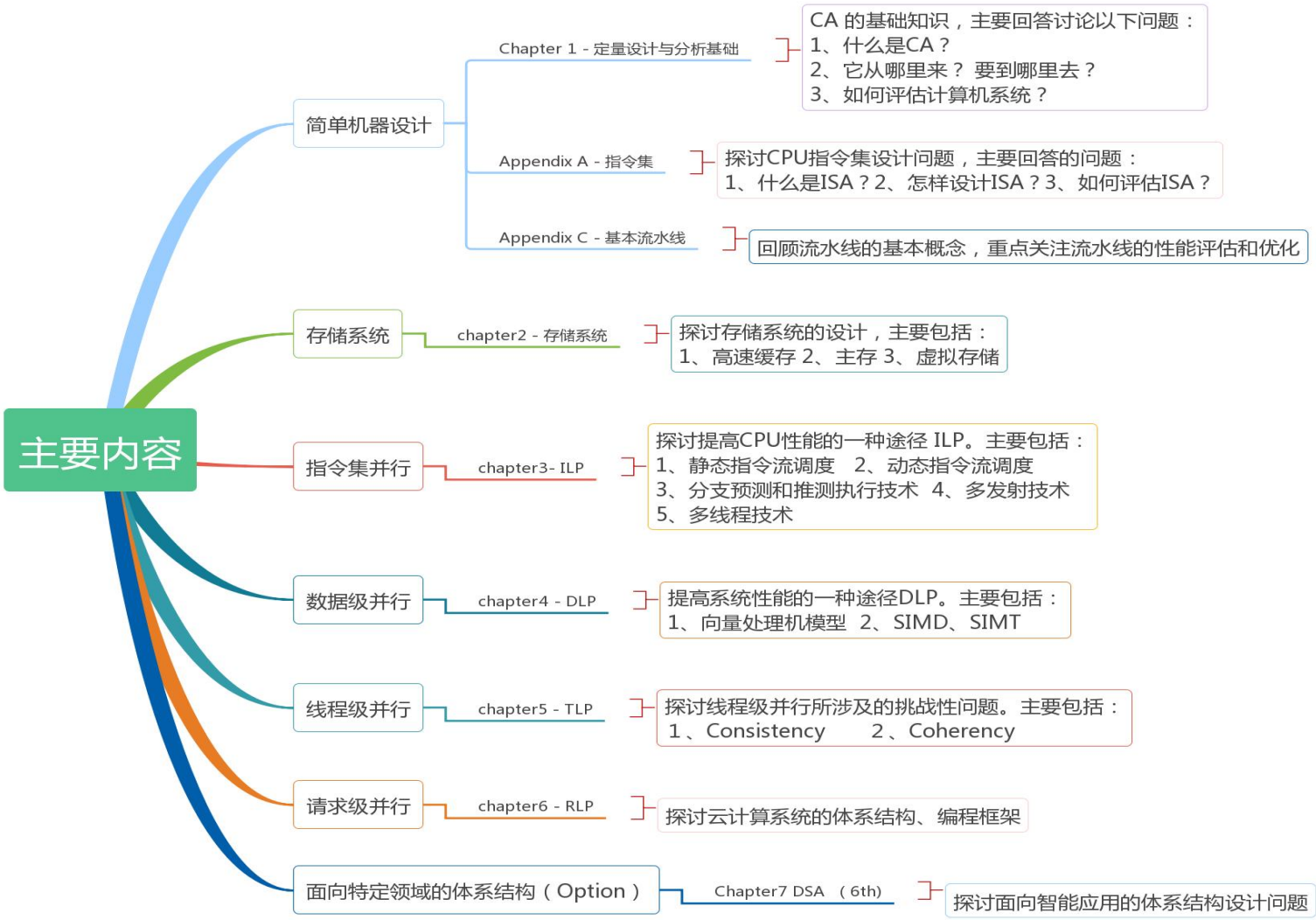


John L. Hennessy, David A. Patternson; Computer Architecture: A Quantitative Approach. Fifth Edition. 机械工业出版社, 2012



John L. Hennessy, David A. Patternson; Computer Architecture: A Quantitative Approach; sixth Edition.

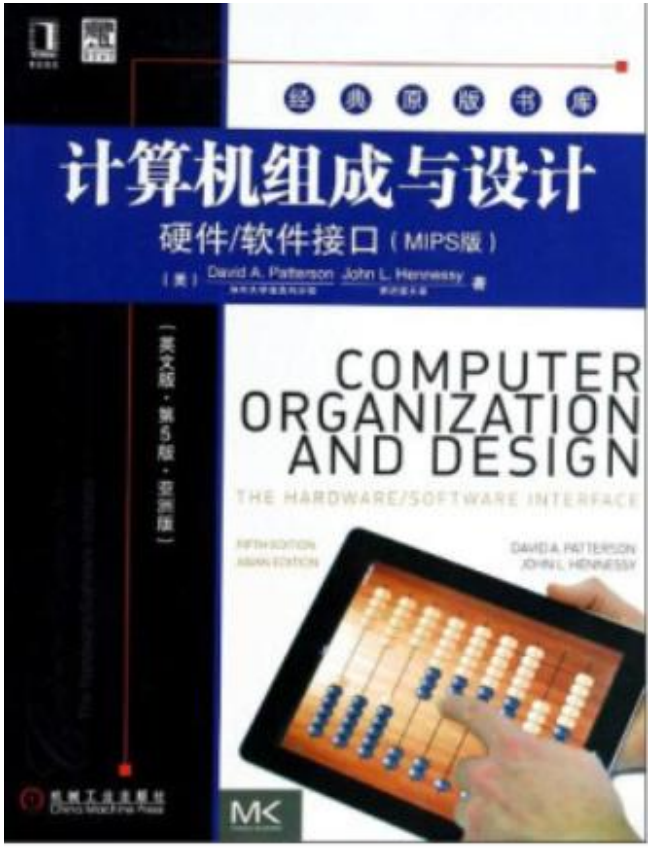
教材包含的主要内容


主要内容
简单机器设计
Chapter 1 - 定量设计与分析基础
CA 的基础知识，主要回答讨论以下问题：
1、什么是CA？
2、它从哪里来？ 要到哪里去？
3、如何评估计算机系统？
Appendix A - 指令集
探讨CPU指令集设计问题，主要回答的问题：
1、什么是ISA？2、怎样设计ISA？3、如何评估ISA？
Appendix C - 基本流水线
回顾流水线的基本概念，重点关注流水线的性能评估和优化
存储系统
chapter2 - 存储系统
探讨存储系统的设计，主要包括：
1、高速缓存 2、主存 3、虚拟存储
指令集并行
chapter3- ILP
探讨提高CPU性能的一种途径 ILP。主要包括：
1、静态指令流调度 2、动态指令流调度
3、分支预测和推测执行技术 4、多发射技术
5、多线程技术
数据级并行
chapter4 - DLP
提高系统性能的一种途径DLP。主要包括：
1、向量处理机模型 2、SIMD、SIMT
线程级并行
chapter5 - TLP
探讨线程级并行所涉及的挑战性问题。主要包括：
1 、Consistency 2 、Coherency
请求级并行
chapter6 - RLP
探讨云计算系统的体系结构、编程框架
面向特定领域的体系结构（Option）
Chapter7 DSA （6th）
探讨面向智能应用的体系结构设计问题

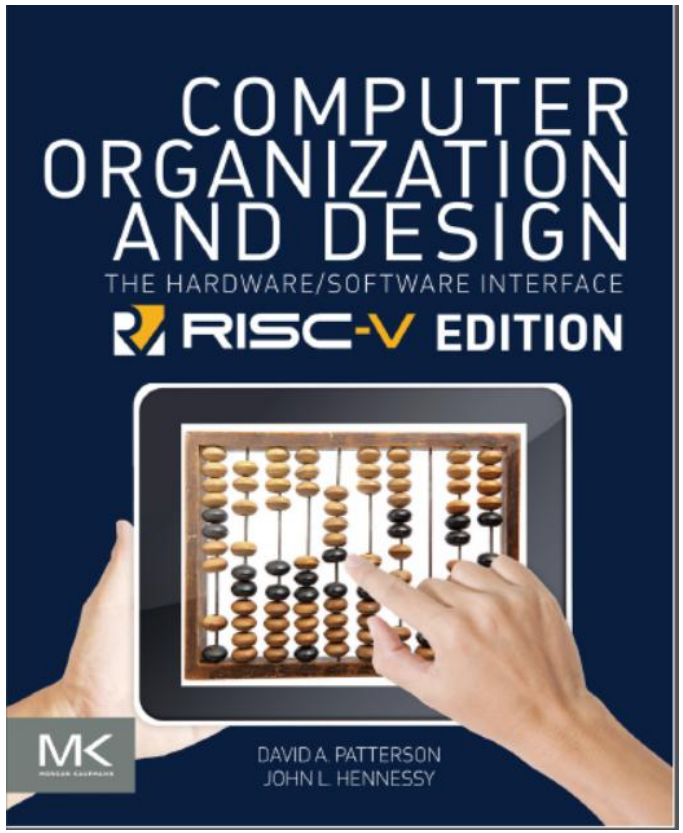
本课程的主要内容

- **简单机器设计（Chapter 1, Appendix A, Appendix C)**
  - ISAs, Iron Law, simple pipelines
- **存储系统(Chapter 2，Appendix B)**
  - DRAM, caches, virtual memory systems
- **指令级并行(Chapter 3)**
  - score-boarding, out-of-order issue
  - SuperScalar, VLIW machines, multithreaded machines
- **数据级并行(Chapter 4)**
  - vector machines,SIMD, SIMT (GPU)
- **线程级并行(Chapter 5)**
  - memory models, cache coherence, synchronization
- **面向特定领域的处理器体系结构（DSA)**
  - IPU、DSP、GPU

其他主要参考书

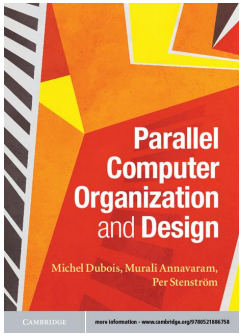


David A. Patternson, John L. Hennessy, Computer Organization & Design : The Hardware/Software Interface, Third Edition. San Francisco: Morgan Kaufmann Publishers, Inc. 2005



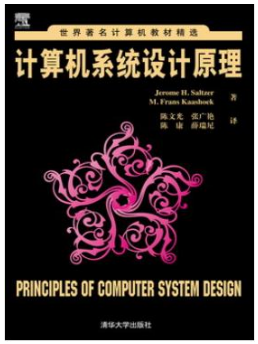
David A. Patternson, John L. Hennessy ; Computer Organization and Design- The Hardware/Software Interface; RISC-V Edition.

其他主要参考书

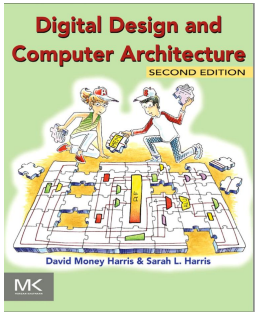


M Dubois, M Annavaram, P Stenström. Parallel Computer Organization and Design. Cambridge University Press 2012.

M Dubois, M Annavaram, P Stenström. 并行计算机组成与设计. 机械工业出版社 2017.



Jerome H. Saltzer，M. Frans Kaashoek 著 陈文光 张广艳 陈 康 薛瑞尼 等译，计算机系统设计原理，清华大学出版社，2012.12



David Money Harris & Sarah L. Harris; Digital Design and Computer Architecture; Second Edition; Morgan Kaufmann Publishers, Inc. 2013

Sarah L. Harris & David Money Harris ; Digital Design and Computer Architecture; ARM Edition; Morgan Kaufmann Publishers, Inc. 2016

评分规则

- **授课**
  - 授课总学时60学时，实验30学时
  - 3C201: 2(3,4)，5(3,4)
- **评分**
  - 平时作业　10%
  - 实验　40%
  - 随堂测验　15%
  - 期终考试　35%

  - 最终总评成绩控制：优秀率40%以内

关于实验

- **重点基于Gem5 模拟器：偏重于系统评估**
  - lab1: 学习使用gem5模拟器
  - lab2: 使用gem5对benchmark做性能分析
  - lab3: 使用gem5探索指令级并行
  - lab4: 使用gem5探索缓存的设计与优化
  - lab5: Cache一致性与Tomasulo模拟器
  - lab6: 数据级并行实验

- **Gem5 Simulator 参考资料：**
  - https://www.gem5.org/documentation/

- **其他选择：参加龙芯杯CPU设计竞赛（需指导教师与授课老师确认）**

关于作弊

- **作业**
- **实验**
- **考试（测验）**

小结：体系结构的定义

- **计算机体系结构**
  - 设计、选择和连接硬件组件以及设计硬件/软件接口，以创建一个满足功能、性能、能耗、成本和其他具体目标的计算系统的科学和艺术。
  - 体现为描述计算机系统的功能、结构组织和实现的一组规则和方法。

- **目标约束**
  - 实现一组设计目标。性能、能效等
  - 不同的平台有不同的设计目标

小结：计算机体系结构研究范畴

早期的定义：Instruction-Set Architecture

程序员可见的计算系统的属性。包括：概念性的结构和功能行为。不包括：数据流和控制流的组织、逻辑设计以及物理实现。

– Amdahl, Blaauw, and Brooks，1964

狭隘的观点（narrow view）：

包括：软件设计者与硬件设备设计者（VLSI）之间的中间层（ISA），以及微体系结构

扩展的观点（expanded view）：

包括：算法层、程序/语言层、系统软件层、ISA层、微体系结构、逻辑电路层、以及器件层

主观题 10分

设置

1、简要描述你对计算机体系结构这一概念的理解。

2、请根据你的理解阐述计算机体系结构和计算机组成原理这两门课程主要的差异在哪里？

正常使用主观题需2.0以上版本雨课堂

作答

Acknowledgements

- **These slides contain material developed and copyright by:**
  - John Kubiatowicz (UCB)
  - Krste Asanovic (UCB)
  - John Hennessy (Standford)and David Patterson (UCB)
  - Chenxi Zhang (Tongji)
  - Muhamed Mudawar (KFUPM)
  - Onur Mutlu（ETH Zürich）
- **UCB material derived from course CS152、CS252、CS61C**
- **KFUPM material derived from course COE501、COE502**