第1章 计算机设计基础

1.1 引言
1.2 计算机的分类
1.3 计算机系统结构定义和计算机的设计任务
1.4 实现技术的趋势
1.5 集成电路功耗的趋势
1.6 成本的趋势
1.7 可靠性
1.8 测量、报告和总结计算机性能
1.9 计算机设计的量化原则
1.10 综合：性能和性价比

1.3 计算机系统结构定义与计算机的设计任务

- 计算机系统结构的初始定义？现在的定义？
- 现代计算机的大概设计过程？

计算机系统结构的原始概念

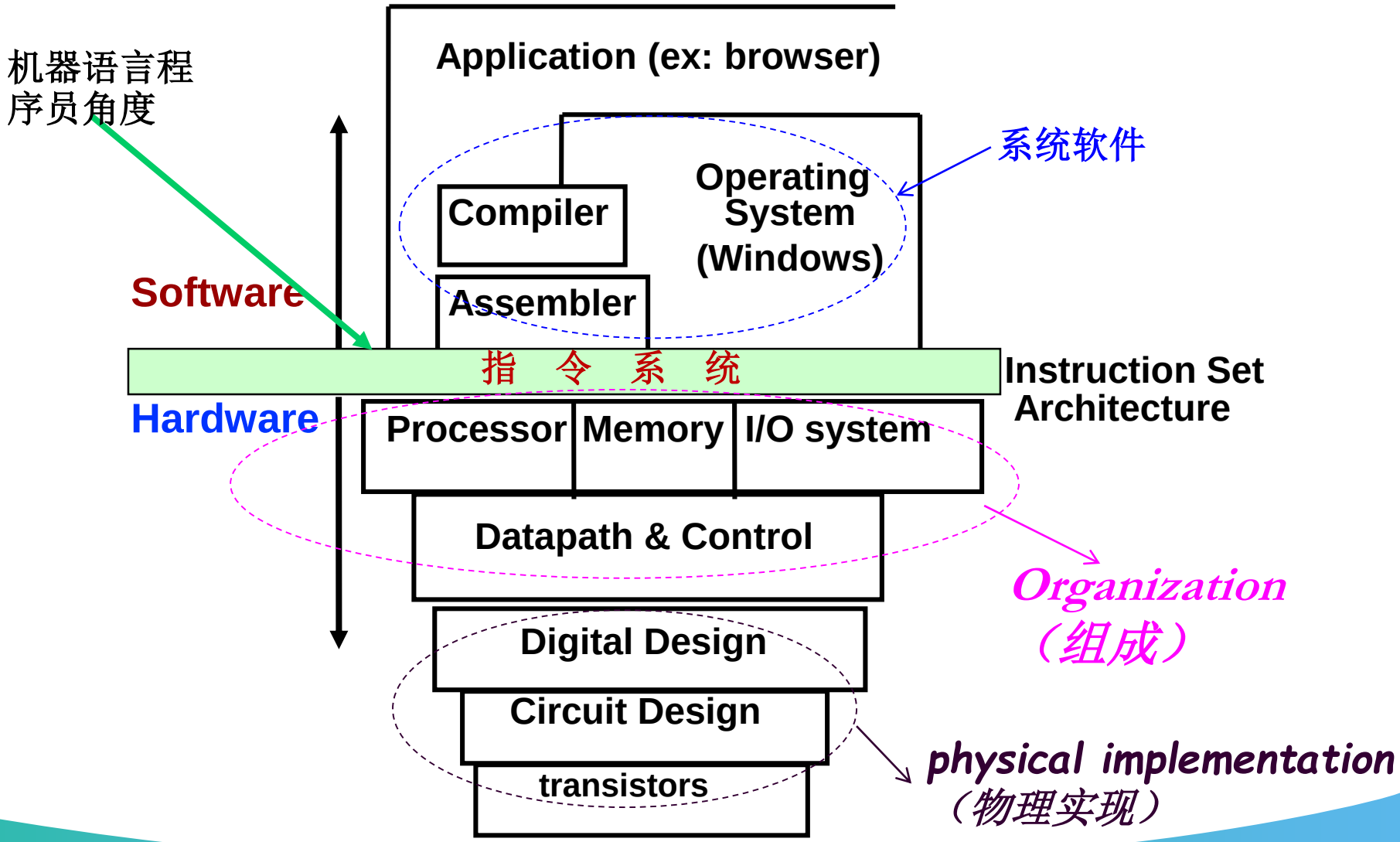
由程序员（机器语言）看见的（计算机）系统属性，即概念性结构和功能行为，以区分数据流动和控制逻辑设计的组成及物理实现。

The attributes of a [computing] system as seen by the programmer, i.e.,

the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls the logic design, and the physical implementation.

Amdahl, Blaaw, and Brooks, 1964

指令集（系统）结构
Instruction Set Architecture (ISA)



经典的计算机系统结构是机器语言程序员所看到的传统机器级所具有的主要属性-----ISA（计算机系统的软、硬件界面）。

计算机组成指的是计算机系统结构的逻辑实现，包括功能部件组成以及逻辑设计等。它着眼于机器级内各事件的排序方式与控制方式，各部件的功能以及各部件的联系。

计算机实现指的是计算机组成的物理实现，包括处理机、主存等部件的物理结构，器件的集成度和速度功耗，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。它着眼于器件技术和微组装技术，其中器件技术在实现技术中占主导作用。

【例 】 ① 主存容量与编址方式(按位、按字节、按字访问等)的确定属于 计算机系统结构 。

② 为达到所定性能价格比，主存速度应多快，在逻辑结构上需采用什么措施(如多体交叉存储等)属于 计算机组成 。

③ 主存系统的物理实现，如存储器器件的选定、逻辑电路的设计、微组装技术的选定属于 计算机实现 。

计算机系统结构、组成和实现三者的相互影响

具有相同计算机系统结构(指令系统相同)的计算机，因为速度要求不同等因素可以采用不同的计算机组成。如，取指令、译码、取操作数、运算、存结果可以在时间上按顺序方式进行，也可以让它们在时间上按流水线重叠方式进行以提高执行速度。

例如，**AMD Opteron 64**与 **Intel Pentium 4**的指令系统相同，即两者的系统结构相同；但内部组成不同，流水线和**Cache**结构是完全不同的，相同的程序在两个机器上的运行时间可能不同。

同样，一种计算机组成可以采用多种不同的计算机实现。例如，主存器件可以采用SRAM芯片，也可以采用DRAM芯片。

例如，同一系列机中的Pentium 4与移动Pentium 4具有相同的指令系统和基本相同的组成，但由于是不同档次的机器，其硬件实现是不同的，两者的时钟频率和存储系统是不同的。

系列机（family machine）：是指由一个制造商生产的具有相同的系统结构，但具有不同组成和实现的一系列不同型号的计算机。

- **例如：IBM 370系列有370/115、125、135、145、158、168等一系列从低速到高速的各种型号。**

  它们的系统结构相同，具有同样的指令系统，从机器程序设计者所看到的机器属性是相同的。

  但它们采用不同的组成和实现技术，在低档机上可以采用指令串行执行的方式，而在高档机上则采用重叠、流水和其他并行处理方式等，因此它们各有不同的性能和价格。

软件兼容性：同一个软件可以**不加修改**地运行于**系统结构相同**的各档机器上，而且运行结果一样，差别只是运行时间不同。

向后兼容：在某一时间生产的机器上运行的**目标软件**能够直接运行于**更晚生产**的机器上。

向上兼容：在**低档机器**上运行的**目标软件**能够直接运行于**高档机器**上。

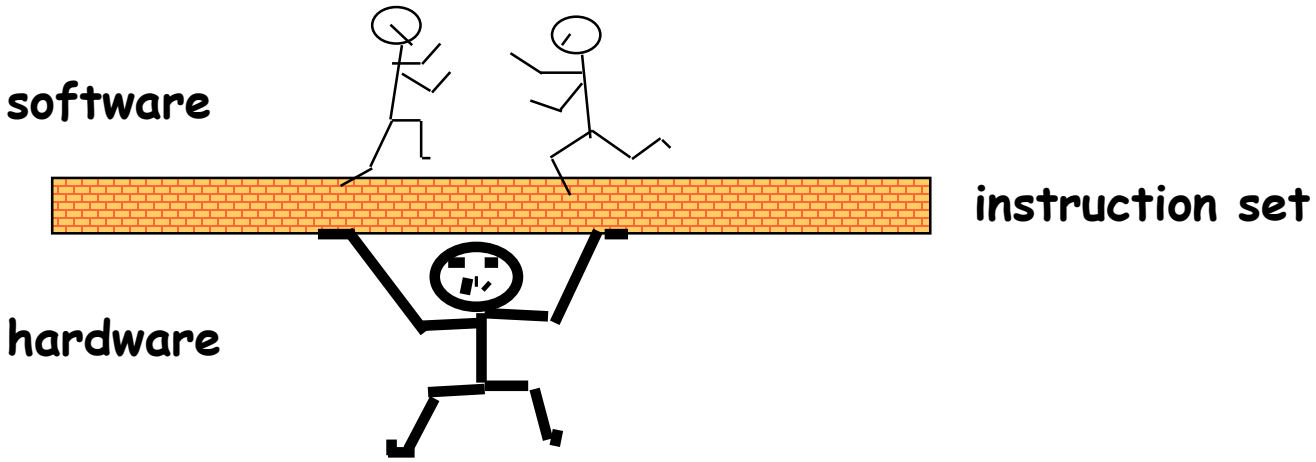
系列机后续各档机器的系统结构可以在原有基础扩充，但要保持向后兼容。如，Intel公司的80x86系列微处理器，从1979年的8086到1999年的Pentium III，增加了保护方式指令集、MMX多媒体指令集和SEE单指令流多数据流指令集，但它保持了极好的二进制代码级的向后兼容性。

系列机的主要缺点：

系列机为了保证软件的向后兼容，要求体系结构基本不改变，这无疑又妨碍了计算机体系结构的发展。

ISA（指令集结构）：硬件与软件之间的接口

- 用途：
  - 软件与硬件之间的接口



接口设计

- 一个好的接口：
  - 允许有多种实现(portability，compatability)
  - 用在很多不同的方面(generality)
  - 为更高层提供方便的功能
  - 允许更低层能有效实现

指令集系统结构ISA的7个重要特征

- ISA的类型：现代通用寄存器结构，早期累加器结构
- 存储器访问：如，按字节访问
- 寻址方式
- 操作数类型和大小：8位字符，32位整型数
- 操作类型：数据传输，算术/逻辑
- 控制流指令：转移，子程序调用/返回
- ISA编码：固定长度，可变长度

Figure 1.4 MIPS registers

Name	Number	Use	Preserved across a call?
$zero	0	The constant value 0	N.A.
$at	1	Assembler temporary	No
$v0 – $v1	2–3	Values for function results and expression evaluation	No
$a0 – $a3	4–7	Arguments	No
$t0 – $t7	8–15	Temporaries	No
$s0 – $s7	16–23	Saved temporaries	Yes
$t8 – $t9	24–25	Temporaries	No
$k0 – $k1	26–27	Reserved for OS kernel	No
$gp	28	Global pointer	Yes
$sp	29	Stack pointer	Yes
$fp	30	Frame pointer	Yes
$ra	31	Return address	Yes

Figure 1.5 Subset of the instructions in MIPS64

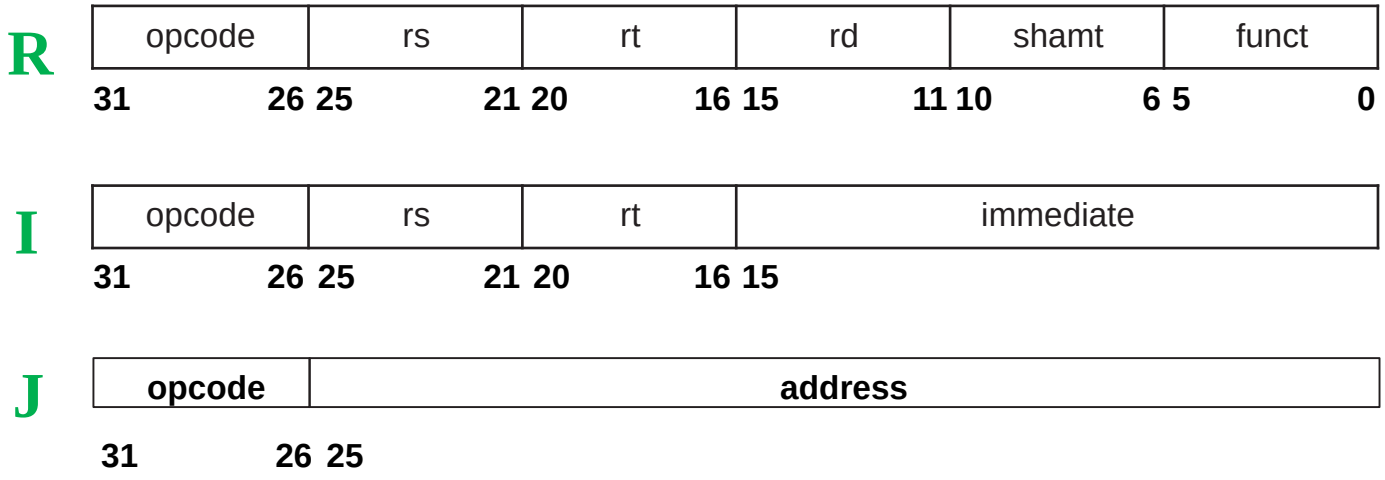
Instruction type/opcode	Instruction meaning
Data transfers	*Move data between registers and memory, or between the integer and FP or special registers; only memory address mode is 16-bit displacement + contents of a GPR*
LB, LBU, SB	Load byte, load byte unsigned, store byte (to/from integer registers)
LH, LHU, SH	Load half word, load half word unsigned, store half word (to/from integer registers)
LW, LWU, SW	Load word, load word unsigned, store word (to/from integer registers)
LD, SD	Load double word, store double word (to/from integer registers)
L. S, L. D, S. S, S. D	Load SP float, load DP float, store SP float, store DP float
MFC0, MTC0	Copy from/to GPR to/from a special register
MOV. S, MOV. D	Copy one SP or DP FP register to another FP register
MFC1, MTC1	Copy 32 bits to/from FP registers from/to integer registers
Arithmetic/logical	*Operations on integer or logical data in GPRs; signed arithmetic trap on overflow*
DADD, DADDI, DADDU, DADDIU	Add, add immediate (all immediates are 16 bits); signed and unsigned
DSUB, DSUBU	Subtract, signed and unsigned
DMUL, DMULU, DDIV, DDIVU, MADD	Multiply and divide, signed and unsigned; multiply-add; all operations take and yield 64-bit values
AND, ANDI	And, and immediate
OR, ORI, XOR, XORI	Or, or immediate, exclusive or, exclusive or immediate
LUI	Load upper immediate; loads bits 32 to 47 of register with immediate, then sign-extends
DSLL, DSRL, DSRA, DSLLV, DSRLV, DSRAV	Shifts: both immediate (DS_) and variable form (DS_V); shifts are shift left logical, right logical, right arithmetic
SLT, SLTI, SLTU, SLTIU	Set less than, set less than immediate, signed and unsigned

Figure 1.5 Subset of the instructions in MIPS64

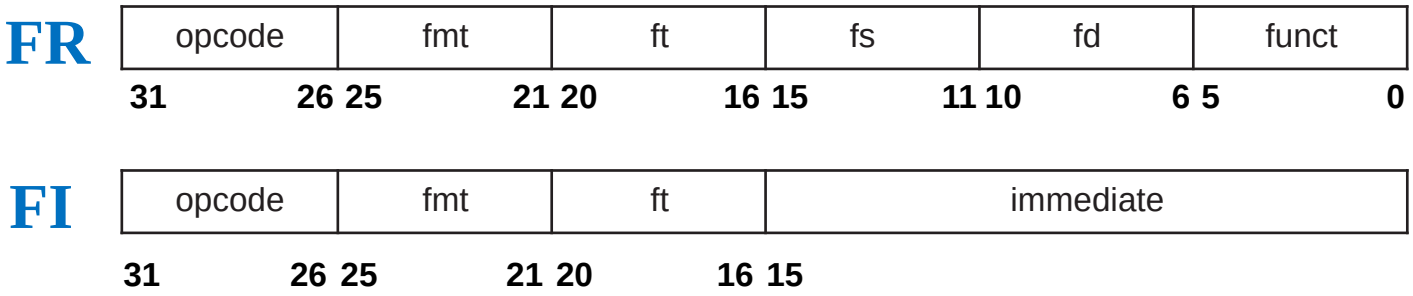
Instruction type/opcode	Instruction meaning
Control	*Conditional branches and jumps; PC-relative or through register*
BEQZ, BNEZ	Branch GPRs equal/not equal to zero; 16-bit offset from PC + 4
BEQ, BNE	Branch GPR equal/not equal; 16-bit offset from PC + 4
BC1T, BC1F	Test comparison bit in the FP status register and branch; 16-bit offset from PC + 4
MOVN, MOVZ	Copy GPR to another GPR if third GPR is negative, zero
J, JR	Jumps: PC+4 high 4 bits and 28-bit address (J) or target in register (JR)
JAL, JALR	Jump and link: save PC + 8 in r31, target is PC-relative (JAL) or a register (JALR)
TRAP	Transfer to operating system at a vectored address
ERET	Return to user code from an exception; restore user mode
Floating point	*FP operations on DP and SP formats*
ADD.D, ADD.S, ADD.PS	Add DP, SP numbers, and pairs of SP numbers
SUB.D, SUB.S, SUB.PS	Subtract DP, SP numbers, and pairs of SP numbers
MUL.D, MUL.S, MUL.PS	Multiply DP, SP floating point, and pairs of SP numbers
MADD.D, MADD.S, MADD.PS	Multiply-add DP, SP numbers, and pairs of SP numbers
DIV.D, DIV.S, DIV.PS	Divide DP, SP floating point, and pairs of SP numbers
CVT._._	Convert instructions: CVT.x.y converts from type x to type y, where x and y are L (64-bit integer), W (32-bit integer), D (DP), or S (SP). Both operands are FPRs.
C._.D, C.__.S	DP and SP compares: "__" = LT,GT,LE,GE,EQ,NE; sets bit in FP status register

Figure 1.6 MIPS 64 instruction set architecture formats

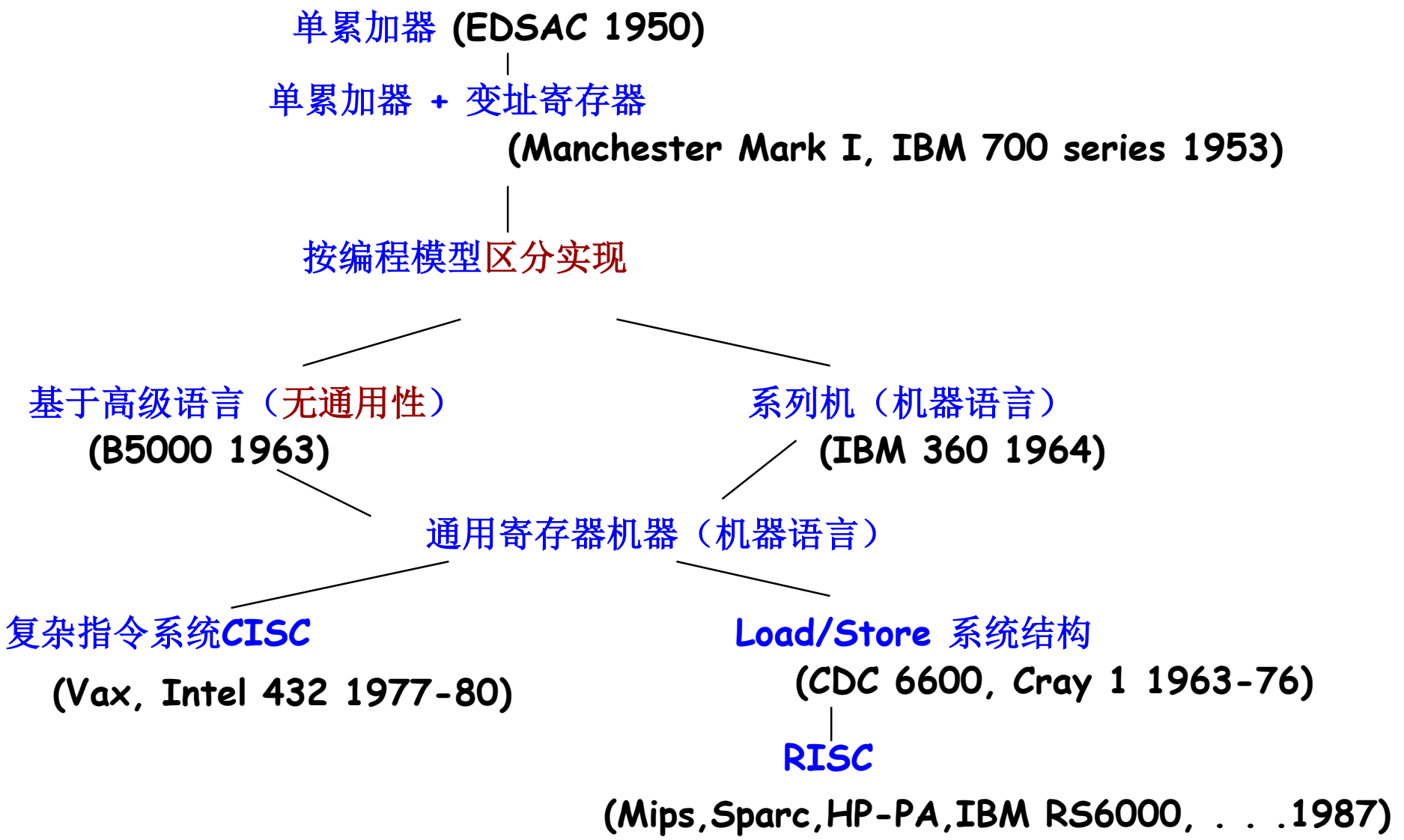
Basic instruction formats



Floating-point instruction formats



指令集（系统）的演化


单累加器 (EDSAC 1950)
单累加器 + 变址寄存器
(Manchester Mark I, IBM 700 series 1953)
按编程模型区分实现
基于高级语言（无通用性）
(B5000 1963)
系列机（机器语言）
(IBM 360 1964)
通用寄存器机器（机器语言）
复杂指令系统CISC
(Vax, Intel 432 1977-80)
Load/Store 系统结构
(CDC 6600, Cray 1 1963-76)
RISC
(Mips,Sparc,HP-PA,IBM RS6000, . . .1987)

课程演化

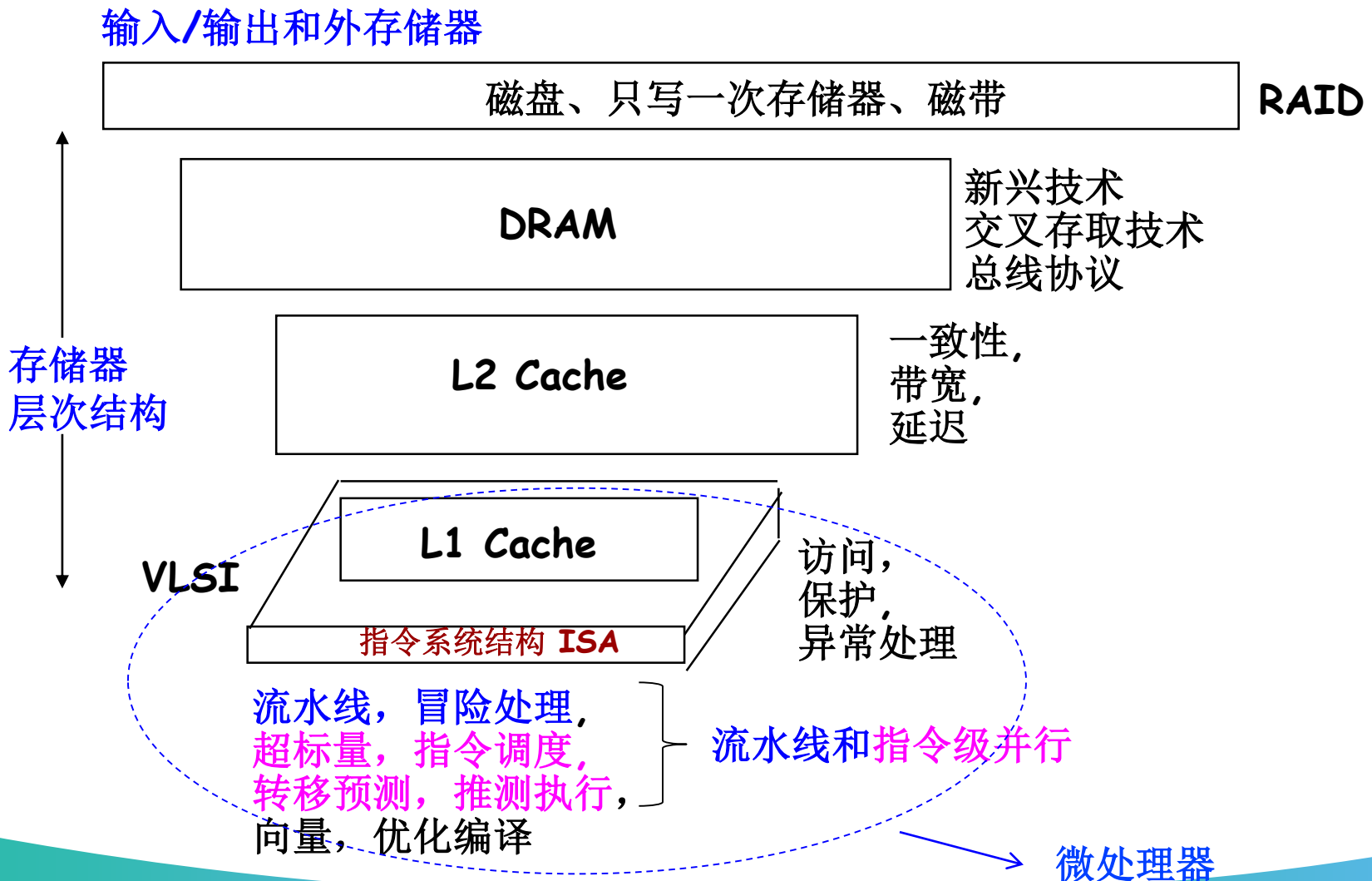
- 1950s to 1960s: Computer Architecture Course:
  - 计算机算法
- 1970s to mid 1980s: Computer Architecture Course:
  - 指令系统设计，尤其是适应编译器的ISA
- 1990s: Computer Architecture Course:
  - CPU、存储系统、I/O系统、多处理器、网络
- 2010s: Computer Architecture Course:
  - 多核
  - 功率监控设计（Power-aware design），可重构的（reconfigurable ）

计算机系统结构的现代定义

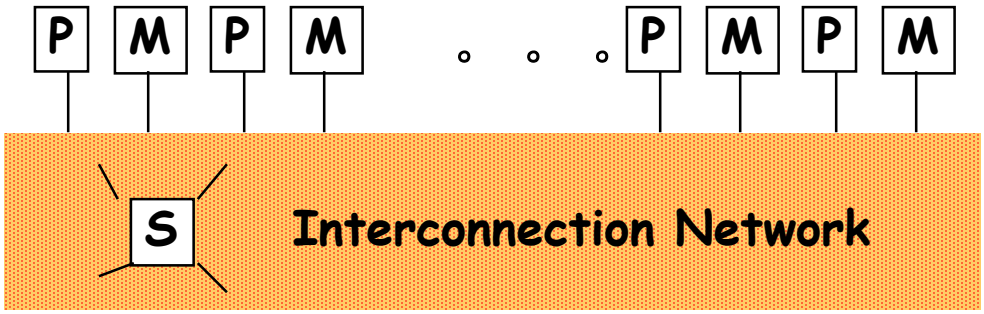
- ***计算机系统结构（现代定义）：***是在满足功能、性能和价格目标的条件下，设计、选择和互连硬件部件构成计算机。
- **系统结构覆盖：**
  - 指令系统设计（*系统结构的原始定义*）
  - 组成（Organization）:计算机设计方面的高层次
    - CPU内部结构、存储器、I/O系统、多处理器、网络
  - 硬件: 计算机的具体实现技术
    - 详细逻辑设计、封装、冷却系统、板级设计，功耗等

当前计算机系统设计涉及多方面技术，从编译器、操作系统到指令系统、组成逻辑、实现技术及功耗、成本、可靠性等设计。

计算机系统结构的课题（单处理器）



计算机系统结构的课题（多处理器）



Processor-Memory-Switch

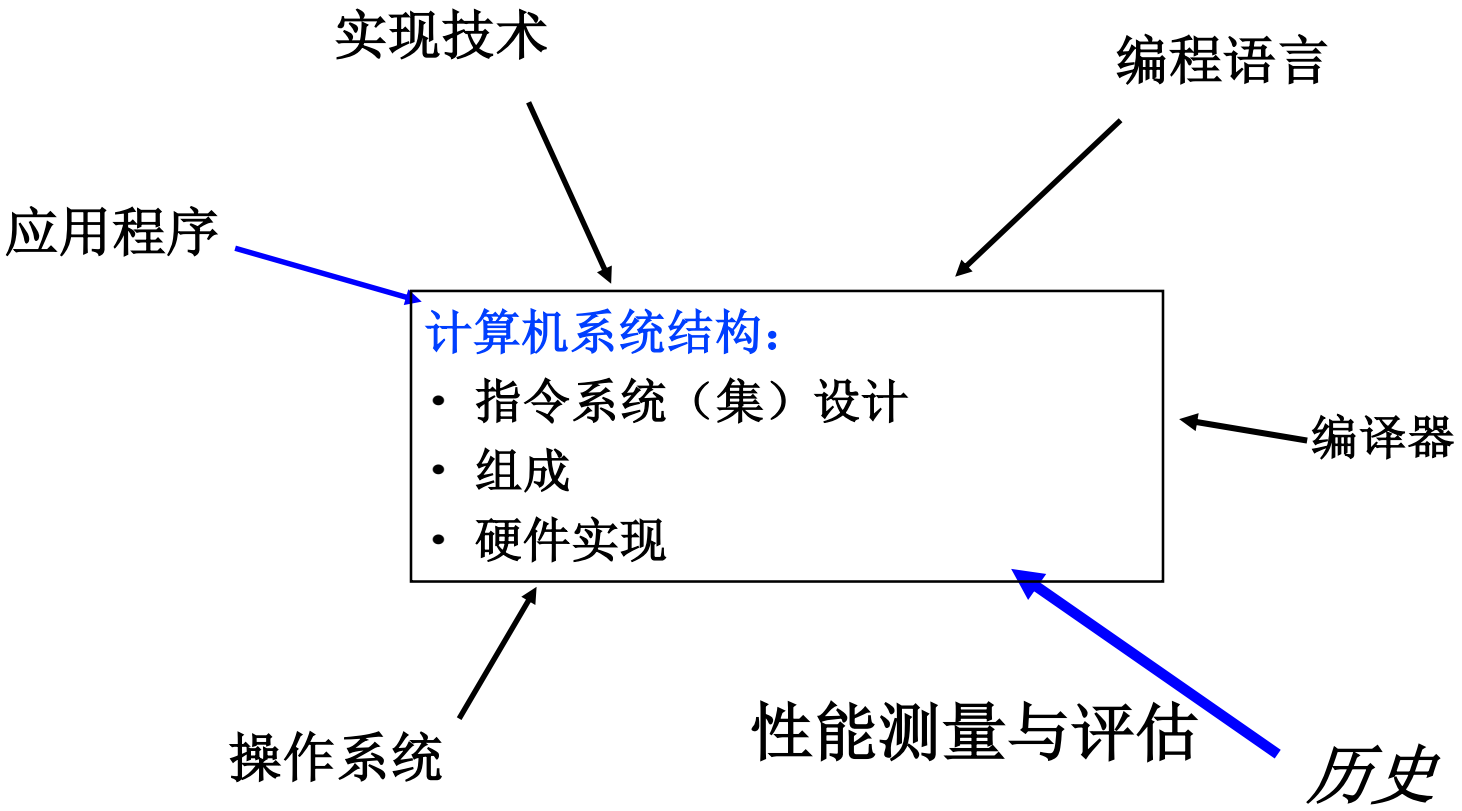
多处理器网络和互连

共享存储器，
消息专递，
数据并行

网络接口

拓扑结构，
路由，
带宽，
延迟，
可靠性

影响计算机系统结构的因素



计算机系统结构已经是这些技术发展的核心，而且将来也仍然是。

计算机设计的任务--1

- 定义用户需求：
  - 功能需求：　　表1-7(5e)
    - 应用领域：5种计算机
    - 软件兼容级别
    - OS　需求
    - 标准：浮点、I/O接口、OS、网络等
  - 非功能需求：
    - 性价比
    - 可用性，可扩展性，吞吐量，...
    - 功耗，大小，温度，　...

Figure 1-7(5e)

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) UNIX,
Operating systems	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)

Figure 1.7 Summary of some of the most important functional requirements an architect faces. The left-hand column describes the class of requirement, while the right-hand column gives specific examples. The right-hand column also contains references to chapters and appendices that deal with the specific issues.

考虑计算机应用

- 设计者需要知道应用的行为
  - 声称设计通用处理器，但是实际上是针对部分应用
  - 结构能够调整以适应应用
- 如今的应用类型
  - 科学方面
    - 天气预报，核爆炸分析，地震分析，医学成像，地球成像
  - 商业
    - 数据库，数据挖掘，视频
  - 办公
    - Microsoft Word，Excel
  - 实时
    - 自动控制系统
  - 其他： 游戏，移动

调整系统结构适应应用

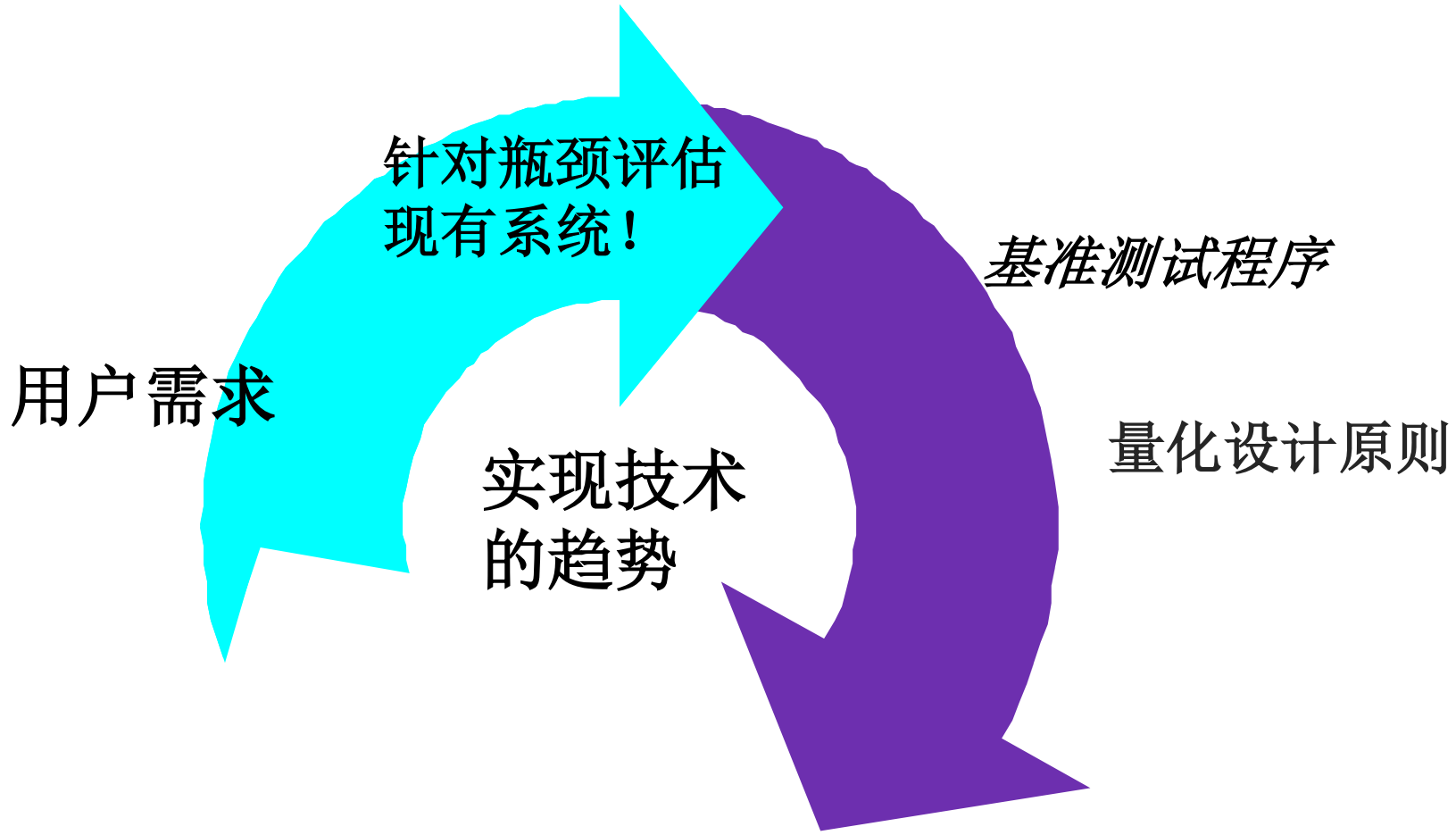
- HP's 处理器　　1.5 MB cache 为事务处理
- Alpha　　高速FP为科学计算
- StrongARM　　为嵌入式
- Intel MMX　　为成像和视频
- Sony EE　　为绘制图形

应用驱动处理器设计

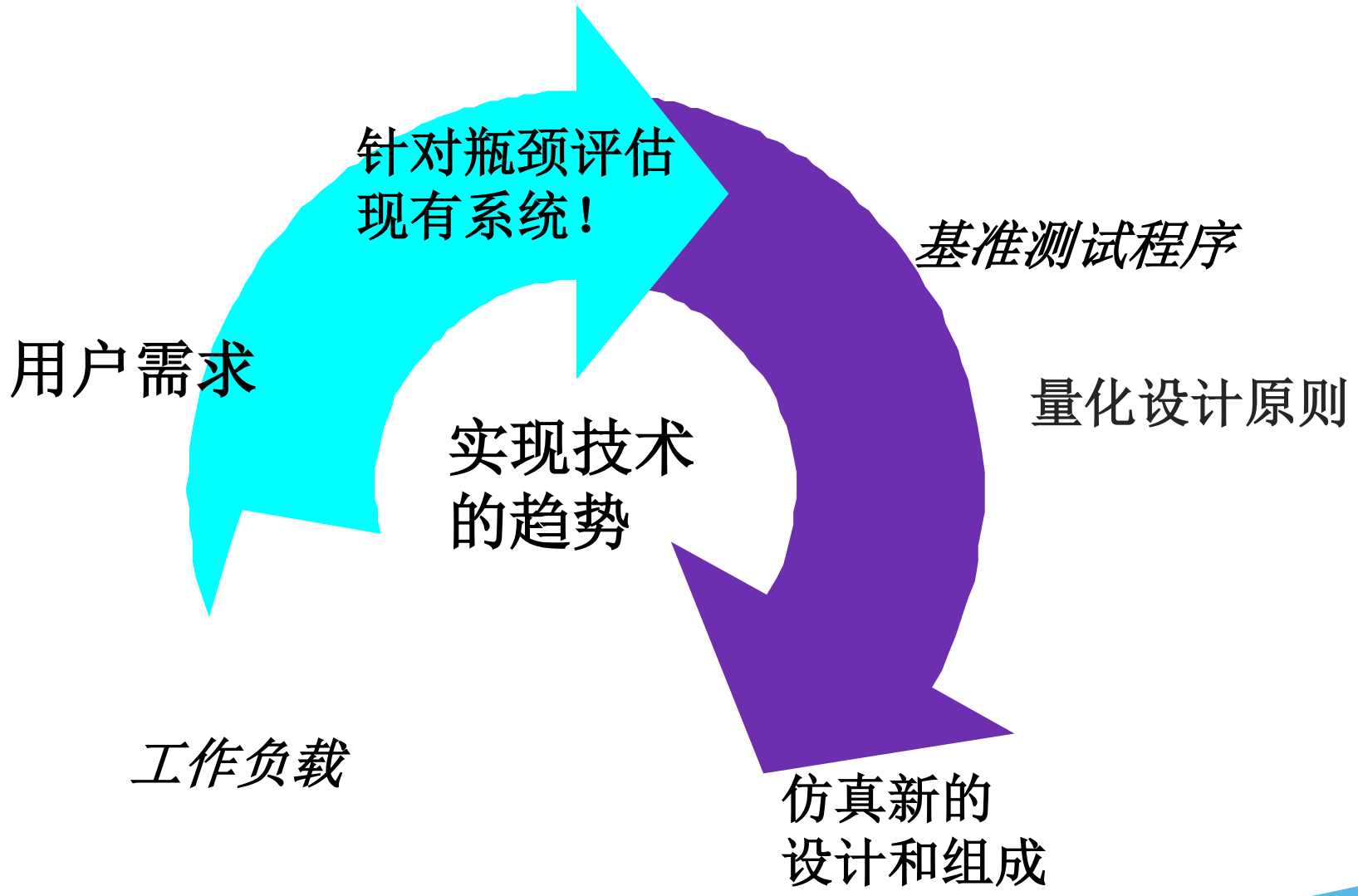
计算机的设计任务--2

- 确定一台新计算机的*重要属性*，必须是在考虑成本 、功耗、可用性等限制下使其*性能最优*
  - 指令集（系统）结构ISA设计
  - 功能组成
    - 计算机设计的高级方面，如：CPU内部结构设计，存储系统和总线结构。
  - 逻辑设计(hardware)
  - 实现 (hardware)

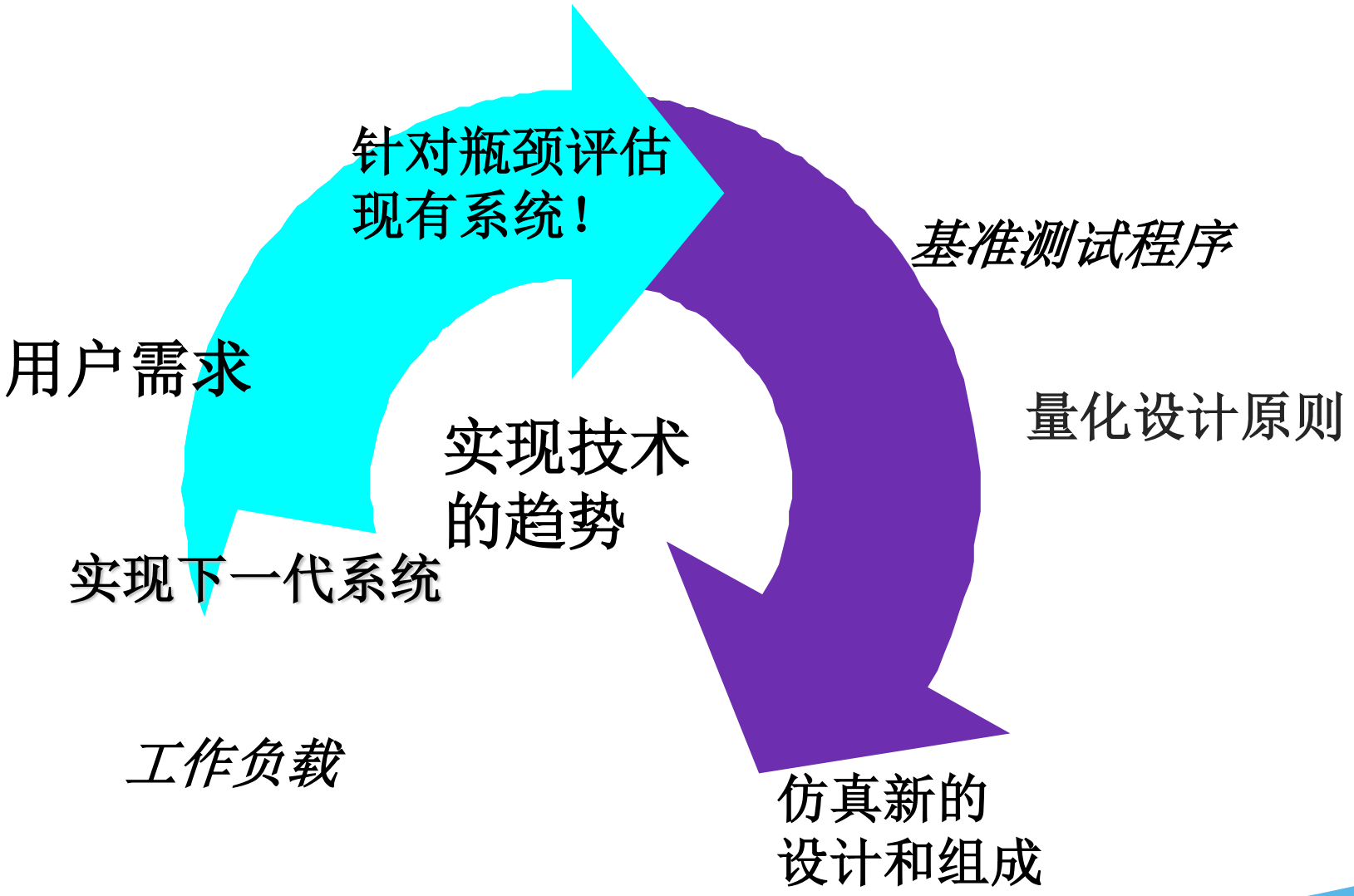
计算机设计的工程方法



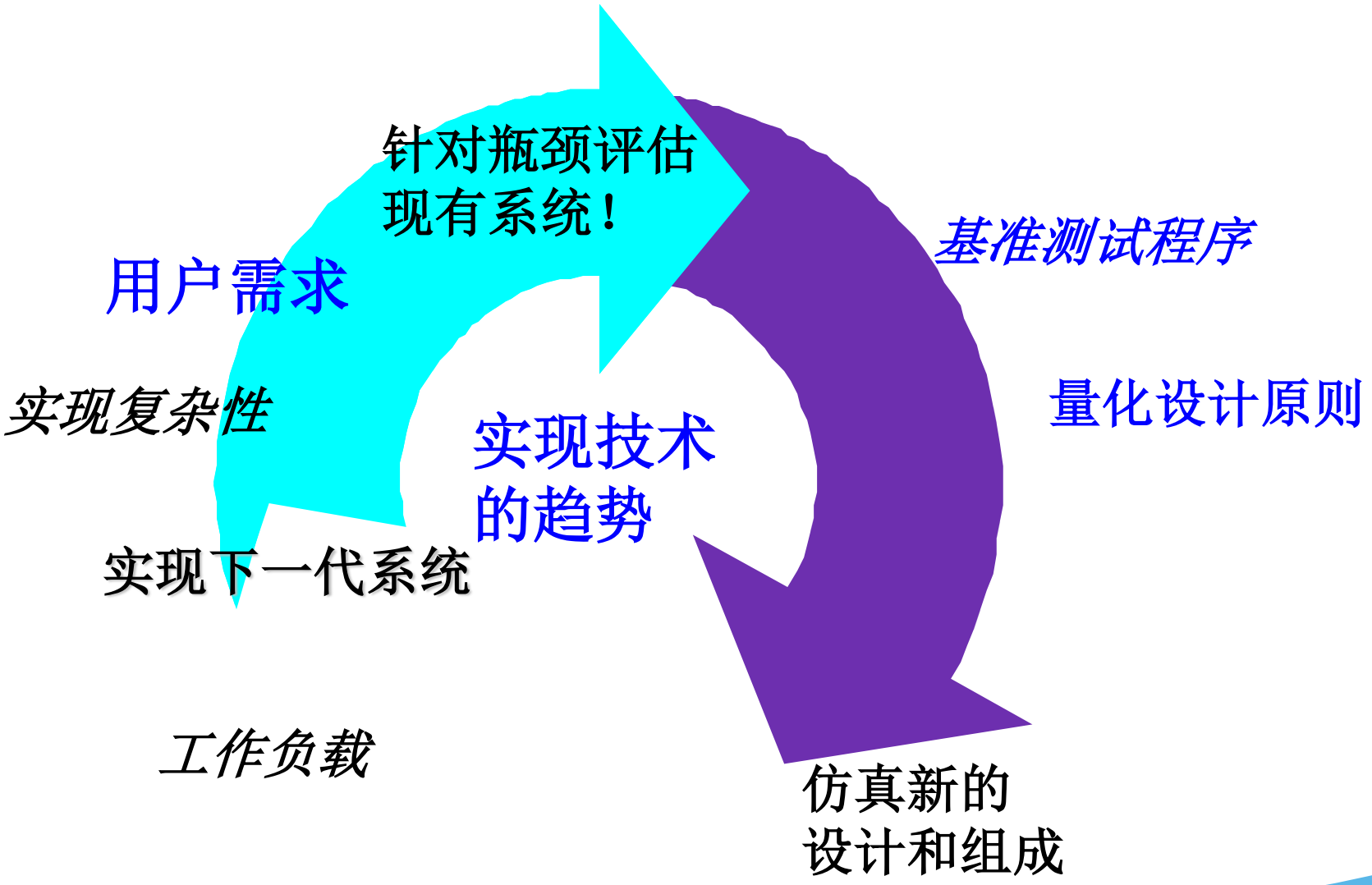
计算机设计的工程方法



计算机设计的工程方法


针对瓶颈评估
现有系统！
基准测试程序
用户需求
量化设计原则
实现技术
的趋势
实现下一代系统
工作负载
仿真新的
设计和组成

计算机设计的工程方法


针对瓶颈评估
现有系统！
基准测试程序
用户需求
量化设计原则
实现复杂性
实现技术
的趋势
实现下一代系统
工作负载
仿真新的
设计和组成

总结：计算机设计的任务

- 考虑（Considerations）：
  - 功能和非功能性需求
  - 实现复杂性
    - 复杂设计花费更长时间完成
    - 复杂设计必须提供更高性能才能具有竞争性
  - 实现技术的趋势
    - 不仅满足当前可用，而且保证以后一段时间可用
  - 集成电路功耗的趋势
  - 成本的趋势
- 依据（Arguments）
  - 针对瓶颈问题评估现有系统
- 量化原则（Quantitative Principles）