

计算机系统结构 *Computer Architecture*

Ch.01 – Basics of CA 计算机系统结构基础知识

Dr. Borui Li
李博睿 博士

2026年秋季·东南大学





- **Amdahl经典定义**

- 传统级机器(机器语言)程序员所看到的计算机属性
- 包括概念性结构和功能特性

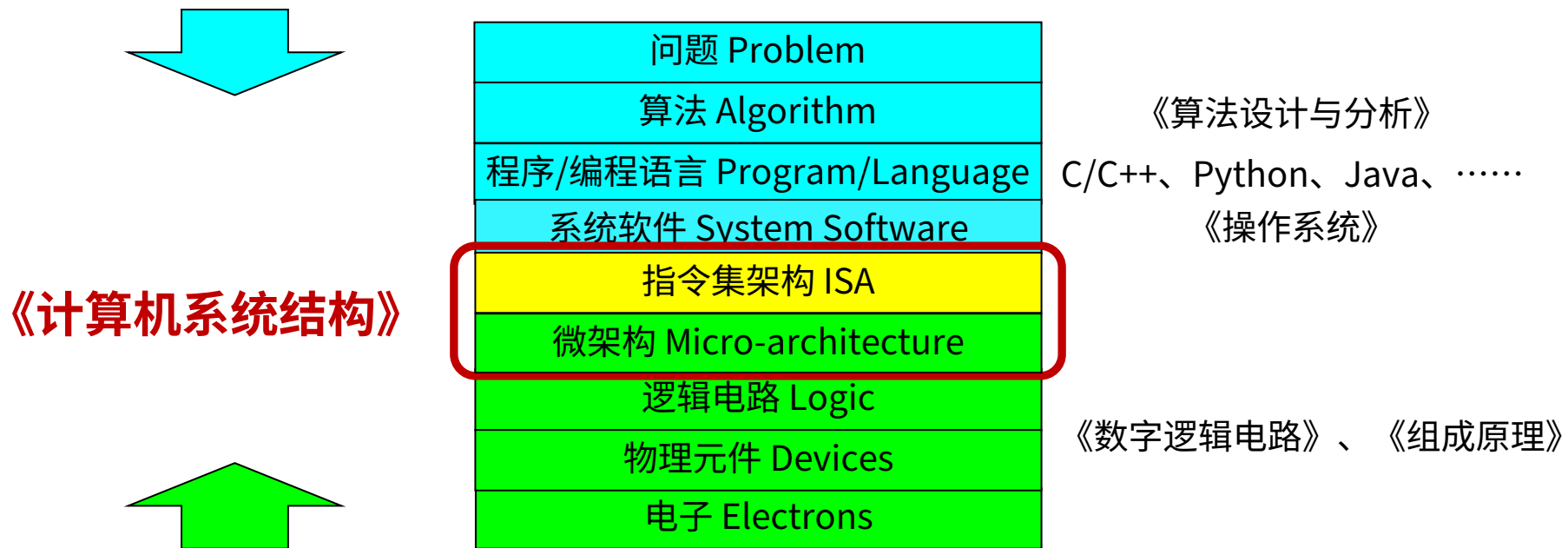
- **Hennessy and Patterson广义定义**

- 系统结构包含指令集结构、组成及硬件

- **Onur Mutlu定义**

- 是在考虑各种独特目标（性能、功耗、成本等）的情况下
- 硬件组件设计、选择、连接和软硬接口设计的科学和艺术

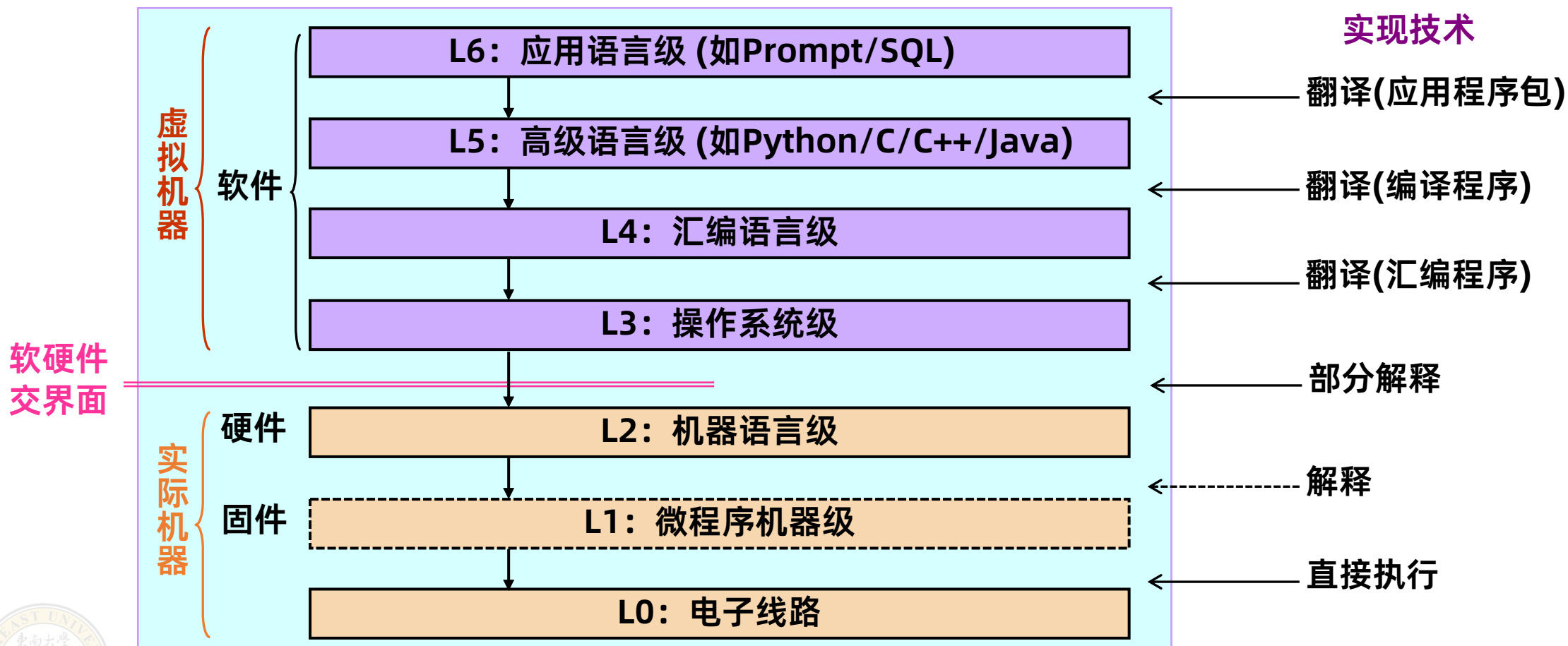




计算机系统的层次结构



• 从计算机语言角度划分：



ISA包含的内容



***概念性结构：** 主要为指令系统，涉及硬件子系统、软件管理等

└→程序(指令序列)←┐←结构&原理┐←面向OS

OS级：进程管理/存储管理/文件管理/设备管理/人机接口



格式： $OP_i | F_1 | A_1 | F_2 | A_2$

功能： $A_D \leftarrow (A_1') OP(A_2')$

***功能特性：**

CO级：CPU/存储器/外设，部件互连

思考：硬件如何触发软件运行？

指令系统	数据表示—	硬件能直接识别和处理的数据类型
	寻址方式—	数据的最小寻址单元、寻址种类及地址计算规则等
	指令功能—	指令的数据操作类型、下条指令地址计算方法等
	指令格式—	表示指令功能的指令字格式及各信息的编码方式
硬件子系统	存储系统—	主存的编址单位、地址空间，层次结构等
	I/O系统—	外设的数据传送控制，I/O操作及状态表示等
	中断系统—	中断&异常的类型、响应机制等
软件管理	机器工作状态—	系统态/用户态的定义与切换
	信息保护—	保护方式、保护机构



结构(ISA)与组成及实现的关系



- *计算机组成(CO):** 是ISA的逻辑实现
包含内容— CPU、存储系统、总线结构等的组织(设计)
如：部件功能、控制机构、排队与缓冲、预估与预判技术等
- *计算机实现(CI):** 是计算机组成的物理实现
包含内容— 部件物理结构、器件使用、组装技术等
- *三者关系：关系—** 硬件的总体框架→逻辑实现→物理实现
要求— 结构：组成：实现的方案数=1：m：m*n

示例：	ISA	CO	CI
乘法功能	是否有乘法指令	1位/阵列乘法器	芯片及电路
主存系统	容量&编址方式	速度&措施	器件&电路
总线系统	带宽&时延	线数&传输控制	介质&线距





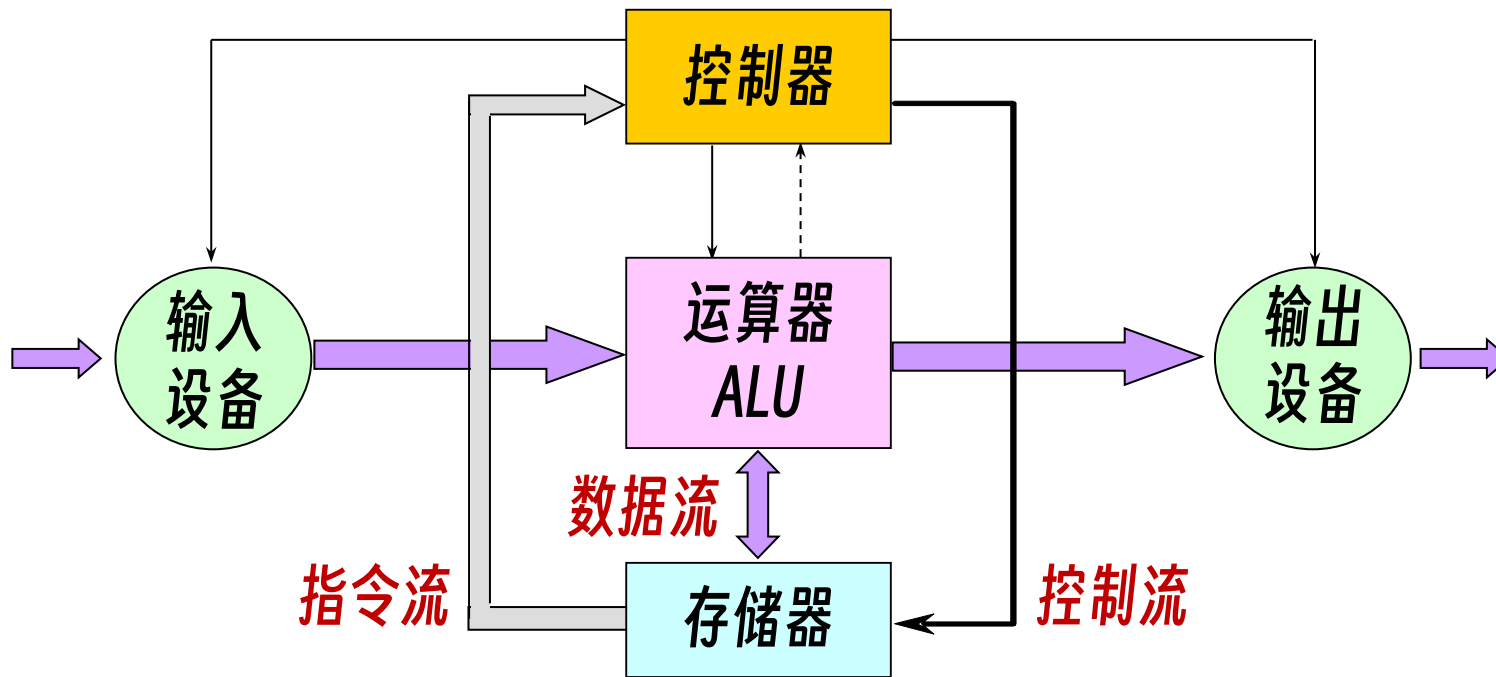
Part I. 计算机系统结构的发展

冯·诺依曼架构



• 工作方式

- 程序及数据预先存放在存储器中
- 机器工作时，自动、逐条地取出指令并执行



冯·诺依曼架构的不断演进（工艺上）

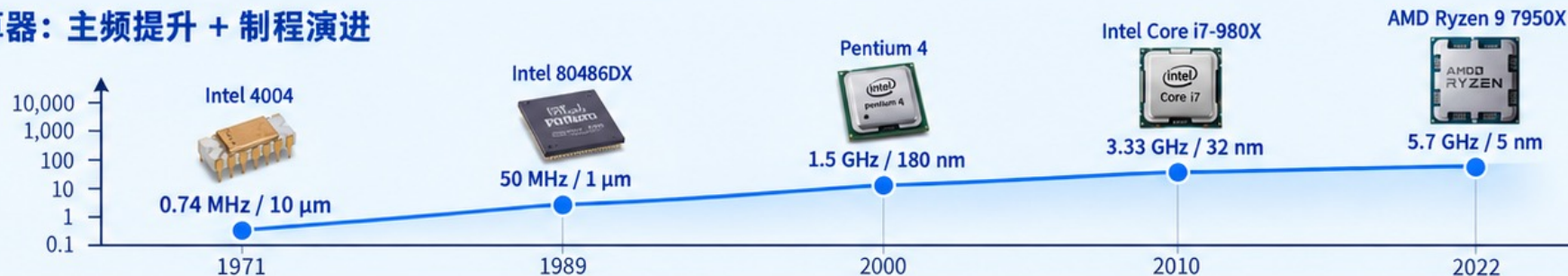


• 最直接的提升：让各个硬件本身变得更快



运算器：主频提升 + 制程演进

主频
(MHz / GHz,
对数尺度)

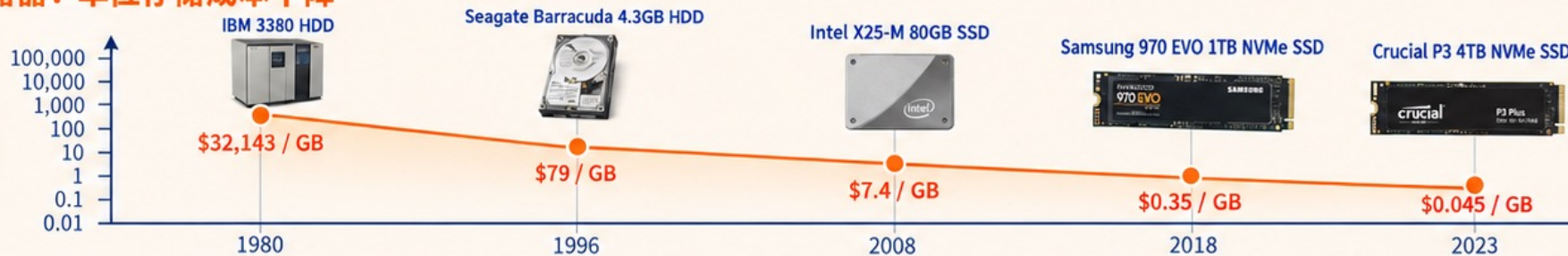


主频提升 + 制程缩小
→ 单核算力持续增强



存储器：单位存储成本下降

单位成本
(USD / GB,
对数尺度,
越低越好)

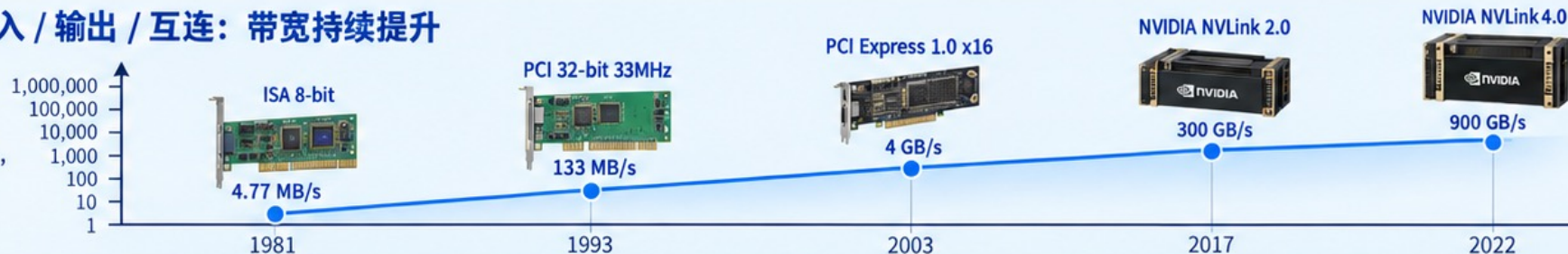


每 GB 成本大幅下降
→ 容量更大、普及更广



输入 / 输出 / 互连：带宽持续提升

峰值带宽
(MB/s 或 GB/s,
对数尺度)



更高带宽、更低通信瓶颈
→ 支撑更大规模计算



不断演进（系统架构方面）：提高并行性 (Parallelism)

- 计算机系统在同一时刻或者同一时间间隔内进行多种运算或操作
 - 只要在时间上相互重叠，就存在并行性
- 同时性 (Simultaneity)
 - 两个或两个以上的事件在**同一时刻**发生
- 并发性 (Concurrency)
 - 两个或两个以上的事件在**同一时间间隔内**发生

操作级、指令级、线程级、任务/过程级、作业或程序级



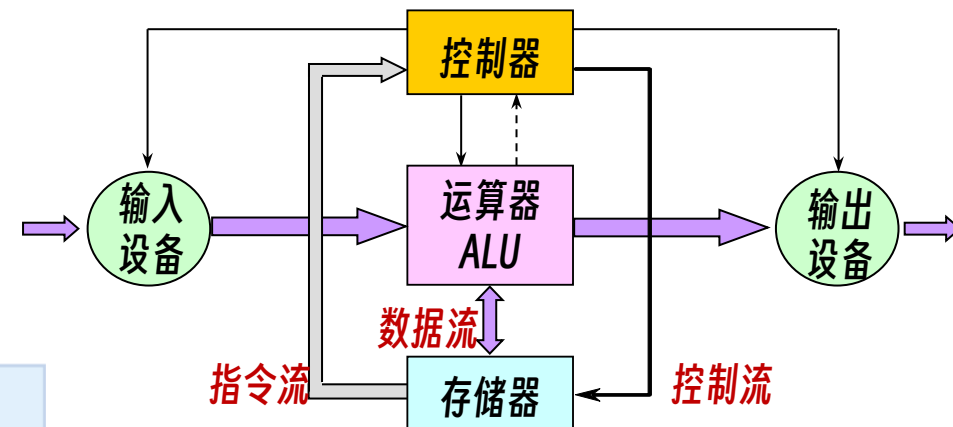
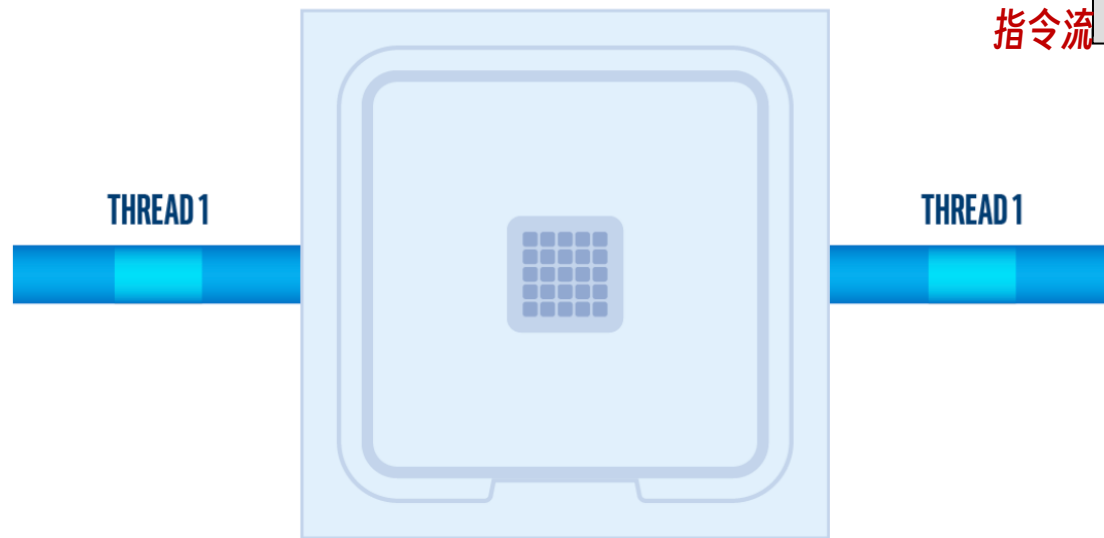
并行颗粒度由细变粗



不断演进（系统架构方面）：提高并行性 (Parallelism)

- 提高并行性 (Parallelism)

- 流水线技术
- 双核、多核、众核……
- Intel超线程技术
 - “4核8线程”



提高并行性是计算机系统结构设计最有效的优化方法！



并行性的开发途径



- 三种主要途径：时间重叠、资源重复、资源共享
- 时间重叠
 - 从时间上开发并行性(并发性)
 - 多个处理过程同时使用同一硬件设备的不同部分
- 资源重复
 - 从空间上开发并行性(同时性)
 - 多个处理过程同时使用重复设置的多个硬件设备
- 资源共享
 - 从软件上开发并行性(并发性)
 - 多个任务轮流使用同一硬件设备

例：流水线技术(操作流水线、指令流水线)

例：多CPU、多核CPU

例：网络打印机
共享单车





Part I. 计算机系统结构的发展

***Part II.* 计算机系统结构的分类**

系统结构的分类



- 按指令流和数据流的多倍性分类
 - 弗林分类法
- 按最大并行度分类
 - 冯氏分类法
- 按并行及流水处理程度分类
 - 汉德勒分类法



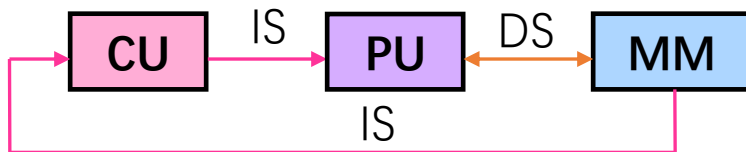
系统结构的分类



- 弗林Flynn分类法

- 分类方法：按指令流和数据流的多倍性分类
- 分类结果：SISD、SIMD、MISD、MIMD

Multiple Instruction Multiple Data



SISD结构(顺序处理机)

通常是早期的机器，例如IBM的大型机等



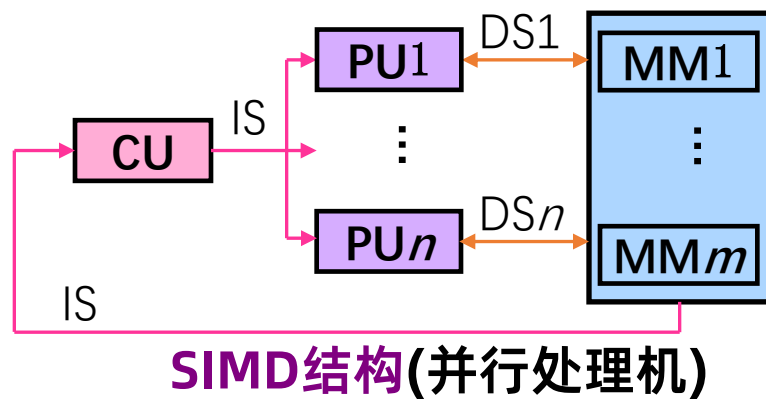
系统结构的分类



- 弗林Flynn分类法

- 分类方法：按指令流和数据流的多倍性分类
- 分类结果：SISD、SIMD、MISD、MIMD

Multiple Instruction Multiple Data



我们现在用的单核计算机基本上都属于SIMD机器



SIMD Instruction Sets



Instruction Sets			
Each intrinsic is only available on machines which support the corresponding instruction set. This list depicts the instruction sets and the first Intel and AMD CPUs that supported them. The green numbers behind the CPUs depict the year of release.			
I.Set	Intel	AMD	Description
x86	all	all	x86 base instructions
SSE	Pentium III 99	K7 Palomino 01	Streaming SIMD Extensions
SSE2	Pentium 4 01	K8 03	
SSE3	Prescott 04	K9 05	
SSSE3	Woodcrest 06	Bulldozer 11	
SSE4.1	Penryn 07	Bulldozer 11	
SSE4.2	Nehalem 08	Bulldozer 11	
AVX	Sandy Bridge 11	Bulldozer 11	Advanced Vector Extensions
AVX2	Haswell 13	-	
FMA	Haswell 13	Bulldozer 11	Fused Multiply and Add
AES	Westmere 10	Bulldozer 11	Advanced Encryption Standard
CLMUL	Westmere 10	Bulldozer 11	Carryless Multiplication
CVT16	Ivy Bridge 12	Bulldozer 11	16-bit Floats
TSX	Haswell 13	-	Transactional Sync. Extensions
POPCNT	Nehalem 08	K10 07	Bit Manipulation Instructions
LZCNT	Haswell 13	K10 07	
BMI1	Sandy Bridge 11	Bulldozer 11	
BMI2	Haswell?? 14	-	

知乎 @不知道先生

CPU-Z

CPU

Caches

Mainboard

Memory

SPD

Graphics

Bench

About

Processor

Name

Intel Core i5 10600K

Code Name

Comet Lake

Max TDP

125.0 W

Package

Socket 1200 LGA

Technology

14 nm

Core Voltage

0.828 V

Specification

Intel®Core™i5-10600K CPU @ 4.10GHz

Family

6

Model

5

Stepping

5

Ext. Family

6

Ext. Model

A5

Revision

Q0/G1

Instructions

MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, AVX, AVX2, FMA3

Clocks (Core #0)

Core Speed

4700.00 MHz

Multiplier

x 47.0 (8 - 48)

Bus Speed

100.00 MHz

Rated FSB

Cache

L1 Data

6 x 32 KBytes

8-way

L1 Inst.

6 x 32 KBytes

8-way

Level 2

6 x 256 KBytes

4-way

Level 3

12 MBytes

16-way

Selection

Socket #1

Cores

6

Threads

12

CPU-Z

Ver. 1.92.0.x64

Tools

Validate

Close

Credit: 知乎 <https://zhuanlan.zhihu.com/p/55327037>

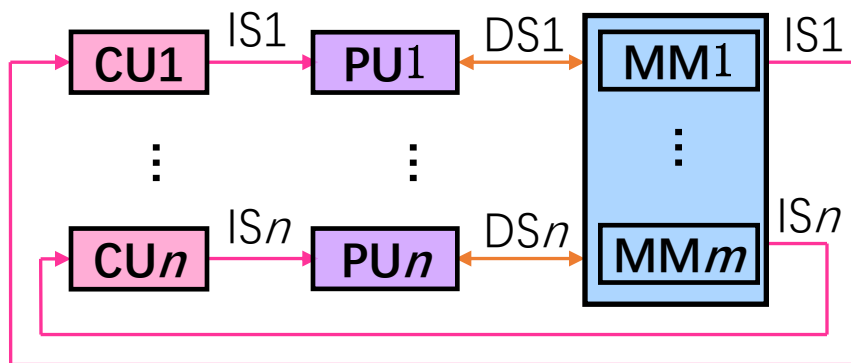
系统结构的分类



- 弗林Flynn分类法

- 分类方法：按指令流和数据流的多倍性分类
- 分类结果：SISD、SIMD、MISD、MIMD

Multiple Instruction Multiple Data



MIMD结构(多处理机)

多核计算平台就属于MIMD的范畴，例如Intel和AMD的双核处理器等都属于MIMD



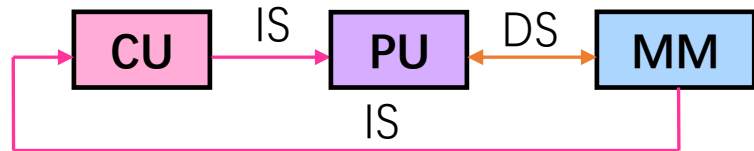
系统结构的分类



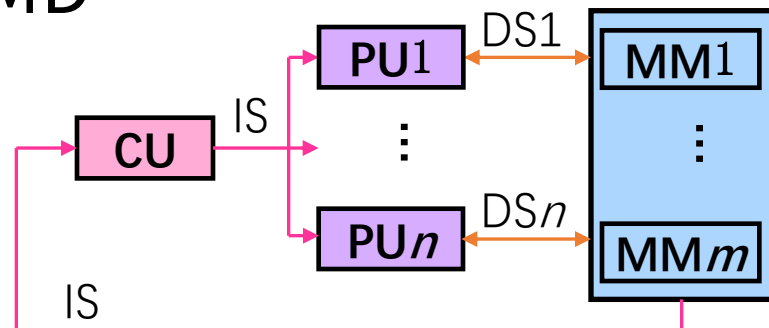
• 弗林Flynn分类法

- 分类方法：按指令流和数据流的多倍性分类
- 分类结果：SISD、SIMD、MISD、MIMD

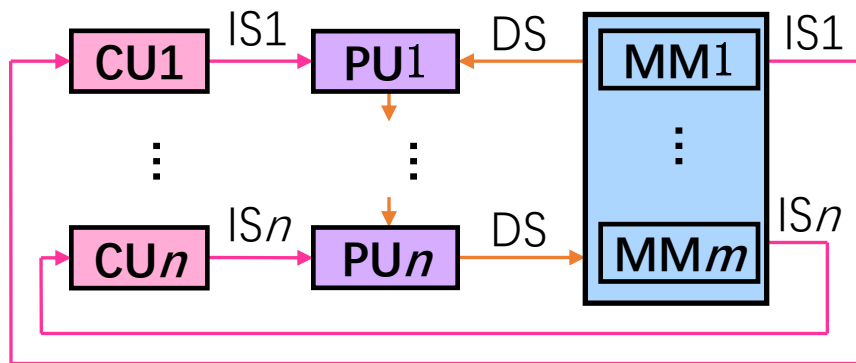
Multiple Instruction Multiple Data



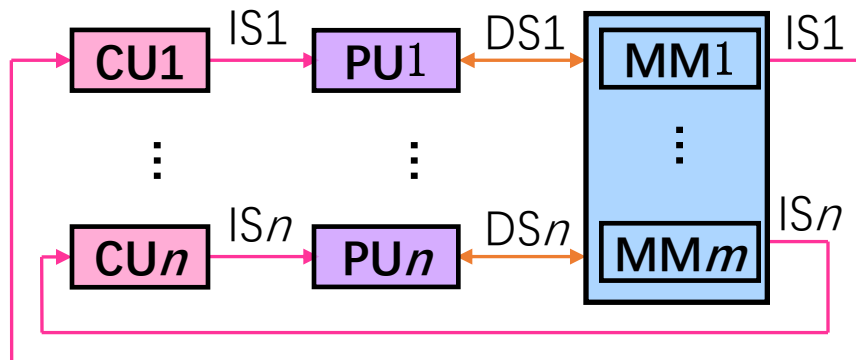
SISD结构(顺序处理机)



SIMD结构(并行处理机)



MISD结构(脉动阵列机)



MIMD结构(多处理机)





Part I. 计算机系统结构的发展

Part II. 计算机系统结构的分类

Part III. 计算机系统的性能评测

测什么？用什么测？怎么测？测完结果怎么分析比较？

如何公平、科学地评价一个计算机？



综合

视频 99+

番剧 0

影视 0

直播 6

专栏 99+

用户 50

综合排序

最多播放

最新发布

最多弹幕

最多收藏

更多筛选



谁最猛？2024热门中端机横评：一加Ace3、红米K70、iQOO Neo...
UP 大米评测 · 2-1



「小白」200元小改千元手机！2K原神！征服星铁？
UP 小白测评 · 2-18



【闭眼可入】2024年最新12款骁龙8gen3手机点评，看看都有哪些优...
UP 三门数码 · 1-11



两千价位手机哪个好？iQOO Neo9 /一加ace3/荣耀90 GT/红米k70...
UP 吾知评测 · 2-1



【建议三连】2024年1月份手机购买推荐，过年买手机，必买这些高...
UP 三门数码 · 2023-12-27



「小白」营销骗局还是感知强烈？内存对手机影响有多大？
UP 小白测评 · 2023-11-11



中低端手机芯片大横评：鱼龙混杂！
UP 极客湾Geekerwan · 2023-9-5



32款骁龙8 Gen2大横评，次旗舰谁敢称王？
UP WHYLAB · 1-31



299元能买新手机？中兴畅行40SE评测
UP 穷评 · 1-28



「小白」小米最成功/失败的手机大盘点 成王败寇系列3
UP 小白测评 · 2022-1-27



「小白」历经30天！十大品牌手机换电池服务大调查！
UP 小白测评 · 2023-4-1



为什么2023会是手机钉子户爆发元年？为什么不推荐买首发 | 大米...
UP 大米评测 · 2023-5-10



「小白」手机处理器 极限到底可以多热？
UP 小白测评 · 2022-6-9



「小白测评」2000元手机对比横评 谁是最香中端机？X10/z1/10XPro
UP 小白测评 · 2020-6-2



40款旗舰大比拼：谁是续航王中王？为什么有些手机续航尿崩 | 大米...
UP 大米评测 · 2023-8-23



「小白」2023年8大旗舰手机屏幕大比拼：国产崛起？
UP 小白测评 · 2023-8-21



这次会翻车吗？骁龙7Gen3性能评测，荣耀100体验 | 大米评测
UP 大米评测 · 1-13



VIVO X100 | IQOO 12 等蓝厂的手机暂时不推荐购买了！
UP 我是柏森 · 2023-11-28



「小白」你的手机为啥越用越卡
UP 小白测评 · 2023-4-22



「小白」五年前的手机测评 长期系统维护可以让手机不卡么？
UP 小白测评 · 2021-12-30



1500元手机谁最「垃」？Redmi/iQOO/真我，17款中低端机大横评
UP WHYLAB · 2023-12-4



如何公平、科学地评价一个计算机？



桌面级CPU排行榜

排行榜		评论	Q 请输入关键字		确认	数据更新时间：2024-1-19		+ 关注
排行	品牌	型号	发布时间	综合成绩				
1	AMD	AMD线程撕裂者7980X	2023	综合成绩				
2	AMD	AMD线程撕裂者7970X	2023	3DMark Time Spy物理分数				
3	AMD	AMD线程撕裂者3990X	2020	7-Zip				
4	Intel	Intel酷睿i9-14900K	2023	CineBench R15				
5	Intel	Intel酷睿i9-13900KS	2022	CineBench R20				
6	AMD	AMD线程撕裂者3970X	2019	CineBench R23				
7	AMD	AMD锐龙9 7950X	2022	CPU-Z				
8	Intel	Intel酷睿i9-13900K	2022	FritzChess BenchMark				
9	Intel	Intel酷睿i7-14700K	2023	POV-Ray				
10	AMD	AMD锐龙9 7950X3D	2023	wPrime-1024M				



Part I. 计算机系统结构的发展
Part II. 计算机系统结构的分类
Part III. 计算机系统的性能评测

测什么？ 用什么测？ 怎么测？ 测完结果怎么分析比较？

测什么？——性能评测量化标准



- 如何评测一台计算机的性能，与测试者看问题的角度有关
 - 普通用户关注的是：**单个程序的执行时间**
 - 执行单个程序所花的时间很少
 - 数据处理中心的管理人员关心的是：**吞吐率**
 - 单位时间内能够完成任务很多
 - 商业服务运维人员关心的是：**可靠性**





- **执行时间 Execution time**
 - 任务从提交→完成的总时间
- **特点**
 - 可以反映总体性能，但不易测量（可能每次不同）
- **Linux系统的time命令**

```
File Edit View Search Terminal Help
naman@root:~$ time sleep 3

real    0m3.003s
user    0m0.003s
sys     0m0.000s
naman@root:~$
```

- Real
 - 从进程 开始到完成所耗费的总时间
 - 包括CPU时间与阻塞IO时间
- User
 - 执行用户态代码所需要的时间
- Sys
 - 执行内核态代码所需要的时间



量化性能指标 (Cont'd)



- **吞吐率 (Throughput)**
 - 单位时间内能够处理的工作量
- **CPU的吞吐率计量单位**
 - MIPS (Million Instructions Per Second, 每秒百万次指令)
 - CISC/RISC指令?
 - MFLOPS (Million Floating-Point Operations Per Second)
- **特点**
 - 与软硬件关系均密切, 能够反映系统的多任务响应能力



量化性能指标 (Cont'd)



- 其他量化指标

- 可用性

- 可靠性

- 可维护性

- 性价比

-

可用性级别	年度宕机时间	周宕机时间	日宕机时间
90%	36.5天	16.8小时	2.4小时
99%	87.6小时	1.68小时	14分钟
99.9%	8.76小时	10.1分钟	86秒
99.99%	52.6分钟	1.01分钟	8.6秒
99.999%	5.26分钟	6.05秒	0.86秒

可用性与可靠性的区别

- 如果系统在每小时崩溃1ms，那么它的可用性就超过99.9999%，但是它还是高度不可靠，因为它只能无故障运行1小时
- 如果一个系统从来不崩溃，但是每年要停机两星期，那么它是高度可靠的，但是可用性只有96%

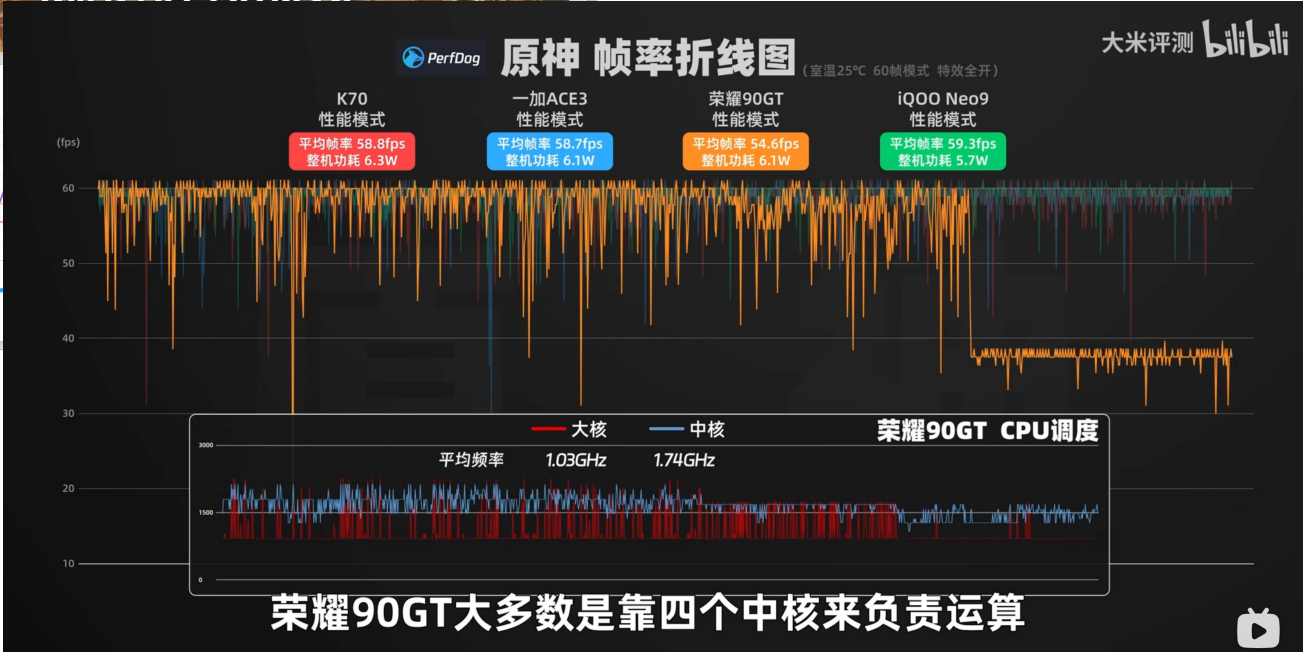
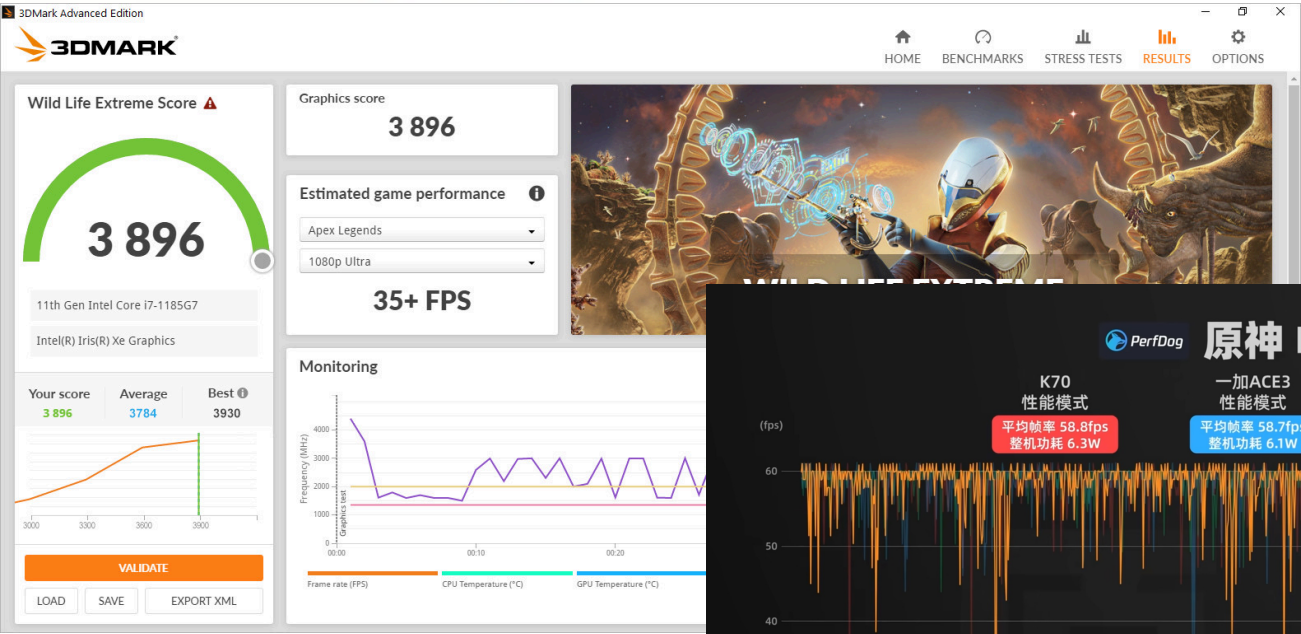




Part I. 计算机系统结构的发展
Part II. 计算机系统结构的分类
Part III. 计算机系统的性能评测

测什么？ **用什么测？** 怎么测？ 测完结果怎么分析比较？

常用基准测试程序



基准测试程序 Benchmark



程序类型	程序内容	测试目标	可信度
核心测试程序	真实程序中的关键代码段	部件	低
小型测试程序	几十行的代码段		
合成测试程序	模拟实际应用特征的程序	系统	高
真实测试程序	测试各方面性能的程序	系统	

- 用于测试和比较性能的基准测试程序的最佳选择是**真实应用程序**
- 从测试性能的角度，其他测试方法是**不完全可信的**
 - 这些程序比较小，具有片面性
 - 系统结构设计者和编译器的设计者可以“合谋”把他们的机器面向这些测试程序进行优化设计，使得该机器显得性能更高



基准测试程序 Benchmark



程序类型	程序内容	测试目标	可信度
核心测试程序	真实程序中的关键代码段	部件	低
小型测试程序	几十行的代码段		
合成测试程序	模拟实际应用特征的程序	系统	高
真实测试程序	测试各方面性能的程序	系统	

- **基准测试程序套件：由各种不同的真实应用程序构成**
 - 能比较全面地反映计算机在各个方面的处理性能
- **SPEC系列：最成功和最常见的测试程序套件**
 - 美国的标准性能测试公司创建



常用基准测试程序



- **SPEC系列（商用）**

- SPEC CPU 1992/1995/2006
 - 整数计算12个+浮点计算17个
- SPEC SFS/Web

- **Polybench/C（开源）**

- 社区贡献

- **MLPerf**

- 来自学术界、研究实验室和工业界



基准测试程序 *Benchmark*

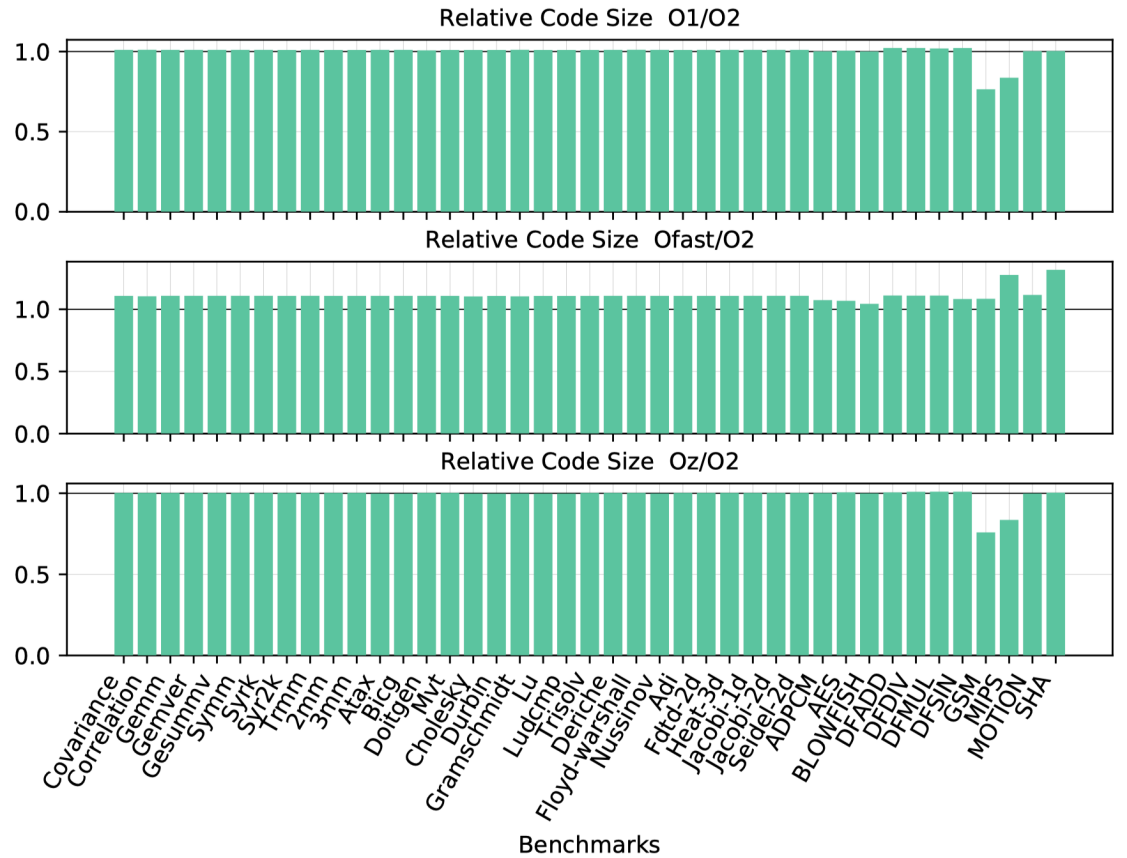
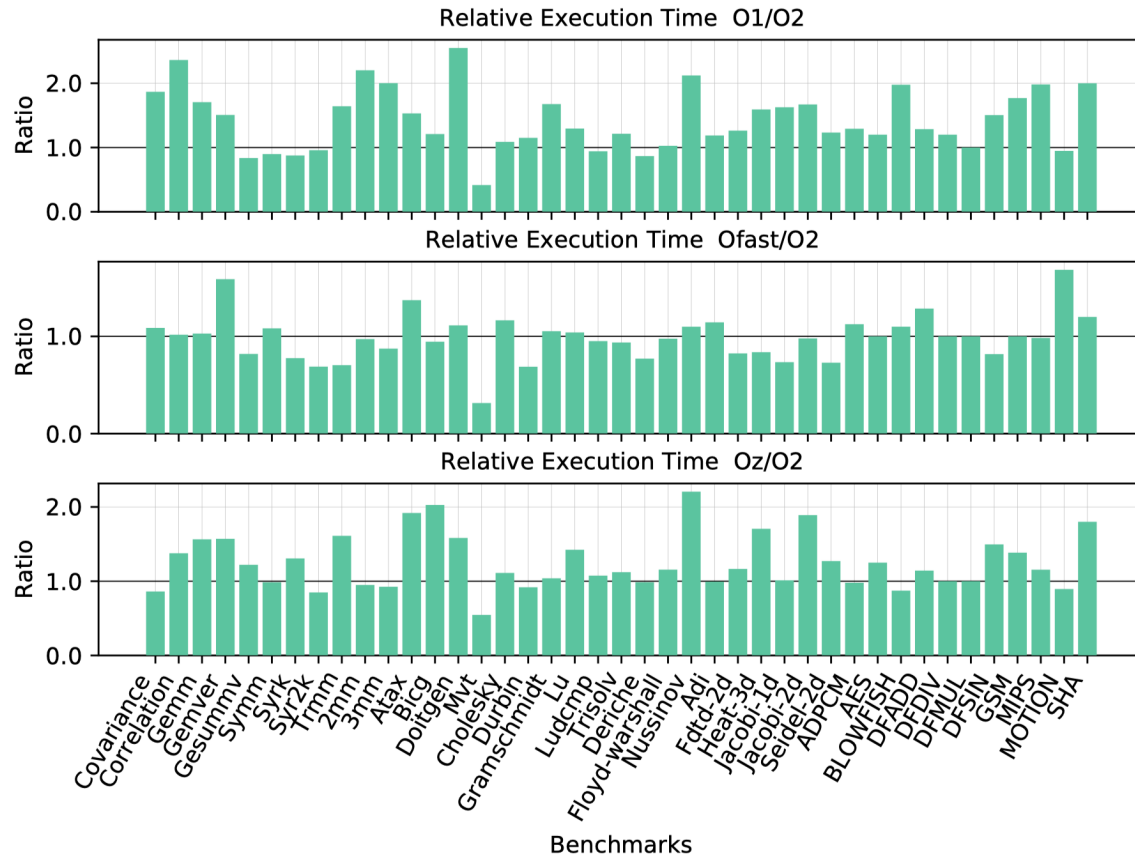


程序类型	程序内容	测试目标	可信度
核心测试程序	真实程序中的关键代码段	部件	低
小型测试程序	几十行的代码段		
合成测试程序	模拟实际应用特征的程序	系统	
真实测试程序	测试各方面性能的程序	系统	高

- **测试结果除了和测试程序有关，还和测试条件有关**
 - 采用同一种编译器
 - 对同一种语言的程序都采用相同的一组编译标志
 - gcc -O1/O2/Oz/Ofast



IMC'21 – WebAssembly Perf. Measurement





Part I. 计算机系统结构的发展

Part II. 计算机系统结构的分类

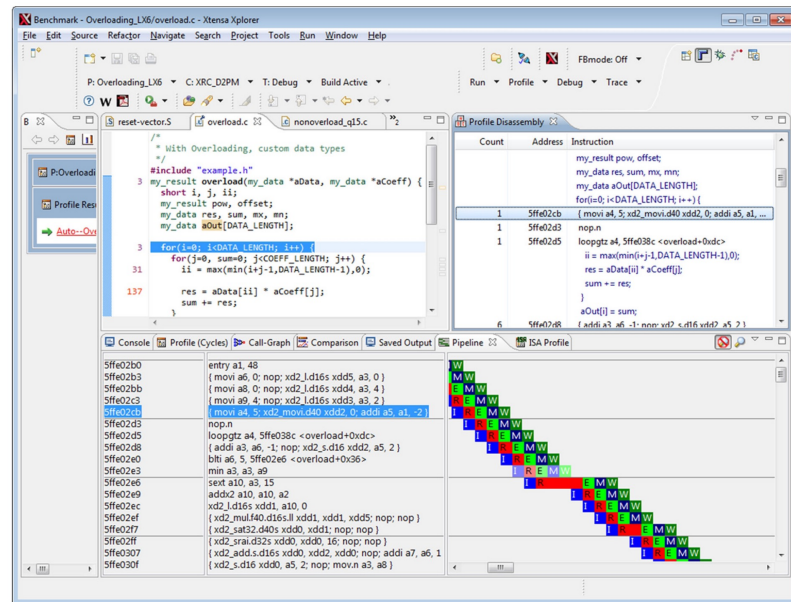
Part III. 计算机系统的性能评测

测什么？用什么测？**怎么测？** 测完结果怎么分析比较？

怎么测？系统结构的测试方法



- 设计与已实现的系统上均可以/需要做测试
- 模拟技术 ——可面向设计中系统
 - ✓ 建立模拟器，模拟系统模型、工作负载模型
 - ✓ 运行模拟程序，获得模拟的性能
 - × 需要对性能指标的组成有较好的分析/建模
- 测量技术 ——面向已实现系统
 - ✓ 运行基准测试程序(多个)，测量实际的性能
 - × 大型系统的测量时间较长，且消耗较多资源





Part I. 计算机系统结构的发展

Part II. 计算机系统结构的分类

Part III. 计算机系统的性能评测

测什么？用什么测？怎么测？ **测完结果怎么分析比较？**

性能比较



示例	计算机X	计算机Y	计算机Z
执行程序A(s)	20	10	40
执行程序B(s)	40	80	20

- **总执行时间**

- 较为综合的度量标准

$$S_n = \sum T_i$$

- **存在问题**

- 真实工作负载中各程序所占比例可能不同





示例	计算机X	计算机Y	计算机Z
执行程序A(s)	20	10	40
执行程序B(s)	40	80	20

- **加权执行时间**

- 较为精准的度量标准

$$A_n = \sum W_i T_i$$

- **存在问题**

- 工作负载中各程序的执行频度无法给出 (无法统一)



性能比较



示例	以X归一化			以Y归一化			以Z归一化		
	X	Y	Z	X	Y	Z	X	Y	Z
程序A	1.00	0.50	2.00	2.00	1.00	4.00	0.50	0.25	1.00
程序B	1.00	2.00	0.50	0.50	1.00	0.25	2.00	4.00	1.00
算术平均值A _m	1.00	1.25	1.25	1.25	1.00	2.13	1.25	2.13	1.00

- 归一化执行时间（相对度量标准）

- 算术平均值（结论随参考的机器而变）

$$A_m = \frac{1}{n} \sum_{i=1}^n R_i = \frac{1}{n} \sum_{i=1}^n \frac{1}{T_i}$$

- 几何平均值（与参考机无关）

$$G_m = \sqrt[n]{\left(\prod_{i=1}^n R_i\right)} = \sqrt[n]{\left(\prod_{i=1}^n \frac{1}{T_i}\right)}$$

- 调和平均值（较为精确）

$$H_m = n / \left(\sum_{i=1}^n (1 / R_i) \right) = n / \left(\sum_{i=1}^n T_i \right)$$





示例	以X归一化			以Y归一化			以Z归一化		
	X	Y	Z	X	Y	Z	X	Y	Z
程序A	1.00	0.50	2.00	2.00	1.00	4.00	0.50	0.25	1.00
程序B	1.00	2.00	0.50	0.50	1.00	0.25	2.00	4.00	1.00
算术平均值A _m	1.00	1.25	1.25	1.25	1.00	2.13	1.25	2.13	1.00
几何平均值G _m	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
调和平均值H _m	1.00	0.80	0.80	0.80	1.00	0.47	0.80	0.47	1.00

• 归一化执行时间（相对度量标准）

- 算术平均值（结论随参考的机器而变）

$$A_m = \frac{1}{n} \sum_{i=1}^n R_i = \frac{1}{n} \sum_{i=1}^n \frac{1}{T_i}$$

- 几何

※性能评测的局限性

- 调和

- 不同应用的性能无法归一化(程序不同)
- 同类应用的性能仅作为参考(程序不一定符合真实)

$$\left(\prod_{i=1}^n \frac{1}{T_i} \right)^{\frac{1}{n}} \bigg/ \left(\sum_{i=1}^n T_i \right)^{\frac{1}{n}}$$





Part I. 计算机系统结构的发展
Part II. 计算机系统结构的分类
Part III. 计算机系统的性能评测
Part IV. 计算机系统设计步骤、原理、思路

步骤→原理→思路

系统结构设计的步骤（类似软件工程）



- 需求分析

进行需求分析—应用环境、语言、OS、外设等方面，技术经济指标、市场分析等

形成需求说明—设计准则、功能说明、器件性能说明等

└→ 造价/可靠性/可扩展性/兼容性/速度等方面

- 具体设计

- 进行软/硬件功能划分，确定机器级界面；

- 组织机器级界面各个方面的功能，可考虑几种方案（回顾： $1 \times m \times n$ ）

- 评价与优化





Part I. 计算机结构的发展
Part II. 计算机结构的分类
Part III. 计算机系统的性能评测
Part IV. 计算机系统设计步骤、原理、思路

步骤→**原理**→思路

计算机系统设计定量原理



- 重点关注经常性事件
- Amdahl定律
- CPU性能公式
- 程序的局部性原理



重点关注经常性事件

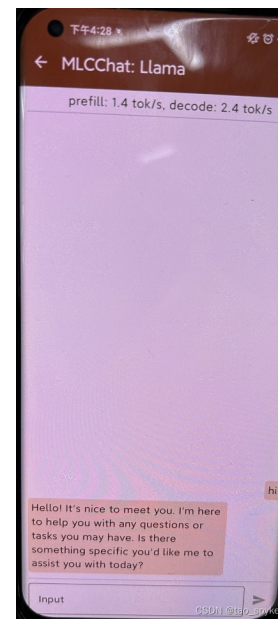


• 主要思想

- 对经常发生的情况采用优化方法的原则进行选择，以得到更多的总体上的改进
- 优化是指分配更多的资源、达到更高的性能或者分配更多的电能等

• 案例：大模型端侧推理

- LLAMA-7B：7GB内存，经常出现OOM
- Android系统：4GB；其他应用~GB



https://blog.csdn.net/tao_spyker/article/details/134847800





- 基本思想

- 优化某部件执行速度所能获得的系统性能加速比，受限于该部件的执行时间占系统中总执行时间的百分比
- “好钢用在刀刃上”

- 系统性能加速比

$$\text{加速比} = \frac{\text{系统性能}_{\text{改进后}}}{\text{系统性能}_{\text{改进前}}} = \frac{\text{总执行时间}_{\text{改进前}}}{\text{总执行时间}_{\text{改进后}}}$$



Amdahl定律 (Cont'd)



- 系统性能加速比

$$\text{加速比} = \frac{\text{系统性能}_{\text{改进后}}}{\text{系统性能}_{\text{改进前}}} = \frac{\text{总执行时间}_{\text{改进前}}}{\text{总执行时间}_{\text{改进后}}} = \frac{1}{(1 - Fe) + \frac{Fe}{Se}}$$

- 可改进比例 (Fe)

- 在改进前的系统中，可改进部分的执行时间占总执行时间的比例

- $Fe \leq 1$

- 部件加速比 (Se)

- 可改进部分改进以后性能提高的倍数，是改进前所需的执行时间与改进后执行时间的比

- 一般情况下， $Se > 1$



Amdahl定律的应用 (1)



• 计算案例

- 将计算机系统中某一功能的处理速度加快**15**倍，但该功能的处理时间仅占整个系统运行时间的**40%**，则采用此增强功能方法后，能使整个系统的性能提高多少？

由题可知： $F_e = 40\% = 0.4$

$S_e = 15$

根据Amdahl定律可知：

$$S_n = \frac{1}{(1 - F_e) + \frac{F_e}{S_e}} = \frac{1}{(1 - 0.4) + \frac{0.4}{15}} \approx 1.6$$

关注经常性事件！

- 采用此增强功能方法后，能使整个系统的性能提高到原来的**1.6**倍



Amdahl定律的应用 (2)



• 计算案例

- 某计算机系统采用浮点运算部件后，使浮点运算速度提高到原来的**25**倍，而系统运行某一程序的整体性能提高到原来的**4**倍，试计算该程序中浮点操作所占的比例？

$$4 = \frac{1}{(1 - Fe) + \frac{Fe}{25}}$$

- 由此可得： $Fe = 78.1\%$
- 即程序中浮点操作所占的比例为78.1%



Amdahl定律的推论



$$\text{加速比} = \frac{\text{系统性能}_{\text{改进后}}}{\text{系统性能}_{\text{改进前}}} = \frac{\text{总执行时间}_{\text{改进前}}}{\text{总执行时间}_{\text{改进后}}} = \frac{1}{(1 - Fe) + \frac{Fe}{Se}} \quad \begin{array}{l} \text{(可改进比例)} \\ \text{(部件加速比)} \end{array}$$

• Amdahl定律：边际效用递减

- 如果仅仅对计算任务中的一部分做性能改进，则改进得越多，所得到的总体性能的提升就越有限

• 推论

- 如果只针对整个任务的一部分进行改进和优化，那么所获得的加速比不超过： $1 / (1 - \text{可改进比例})$



CPU性能公式



- 执行一个程序所需的CPU时间

- 程序CPU时间= 时钟周期时间 × 执行程序所需的时钟周期数

- 时钟周期时间是CPU时钟频率的倒数 ← 取决于硬件实现技术和计算机组成

- 执行程序所需的时钟周期数

- 总时钟周期= $IC \times CPI$

↓ 取决于指令集和编译技术

- IC (Instruction Count) : 所执行的指令条数

- CPI (Cycles Per Instruction) : 每条指令的平均时钟周期数

- 总时钟周期= $\sum_{i=1}^n CPI_i \times IC_i$

↑ 取决于计算机组成和指令集架构

- 程序执行的CPU时间可以写成

- 程序CPU时间 = $IC \times CPI \times \text{时钟周期时间}$



程序的局部性原理



- **主要思想**

- 程序执行时所访问的存储器地址分布不是随机的，而是相对地簇聚

两个程序比快慢？

- **常用的一个经验规则**

- 程序执行时间的90%都是在执行程序中10%的代码

- **程序的时间局部性**

- 程序即将用到的信息很可能就是目前正在使用的信息

- **程序的空间局部性**

- 程序即将用到的信息很可能与目前正在使用的在空间上相邻或者临近





Part I. 计算机结构的发展
Part II. 计算机结构的分类
Part III. 计算机系统的性能评测
Part IV. 计算机系统设计步骤、原理、思路

步骤→原理→**思路**

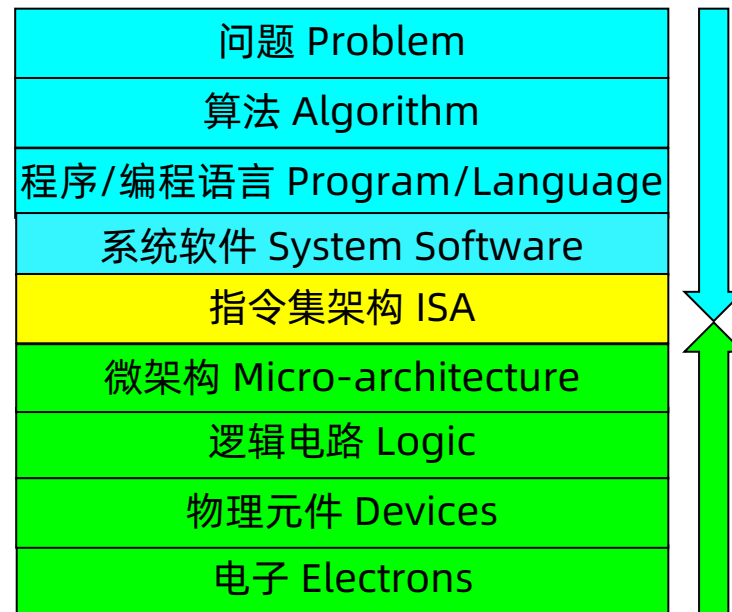
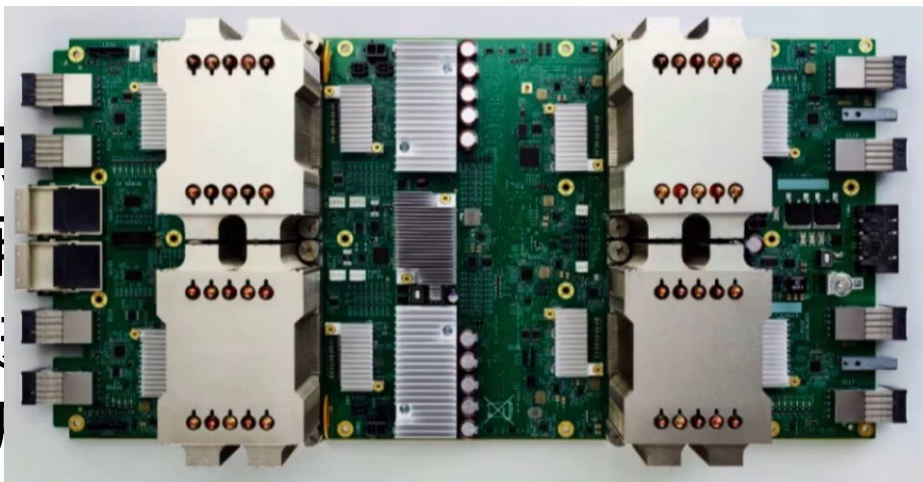
系统设计的思路



- **自下而上 (Bottom-up)** 最早的计算机设计思路
 - 从层次结构的最下面一级开始，逐层往上设计各层的机器
 - 有了硬件之后再设计软件

- **自上而下 (Top-down)** 适合专用机的设计
 - 从软件开始设计，首先确定算法，再逐级设计

- **自中间向两头** 最常用的思路
 - 解决系统性能瓶颈
 - 首先设计系统交互界面
 - 软件开始





Part I. 计算机系统结构的发展

Part II. 计算机系统结构的分类

Part III. 计算机系统的性能评测

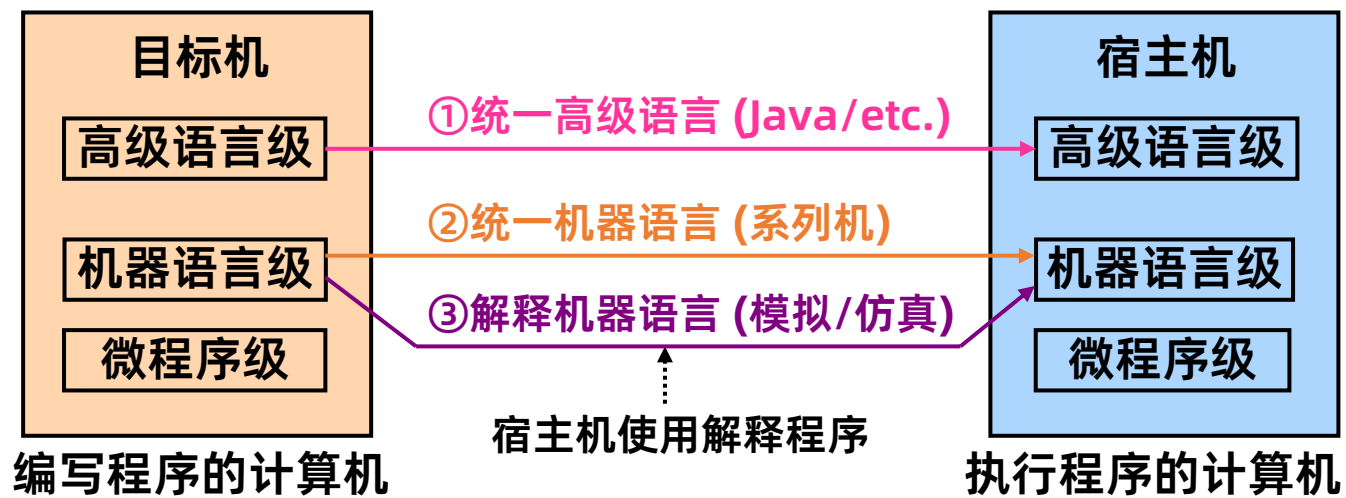
Part IV. 计算机系统设计步骤、原理、思路

Part V. 软件、硬件、应用的发展对系统结构的影响

软件对系统结构的影响



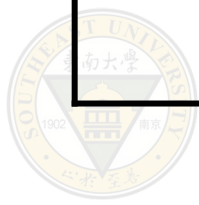
- 影响因素：主要为软件**可移植性**
 - 以前的程序，可在新的计算机上运行
- 解决方法：主要有三种



硬件对系统结构的影响



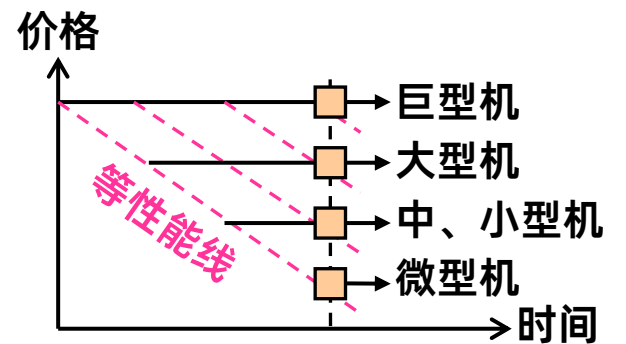
分代	器件特征	结构特征	软件特征	典型实例
第一代 (1945—1954年)	电子管和继电器	存储程序计算机 程序控制I/O	机器语言 汇编语言	普林斯顿ISA， ENIAC，IBM 701
第二代 (1955—1964年)	晶体管、磁芯 印刷电路	浮点数据表示、寻址技 术、中断、I/O处理机	高级语言和编译 批处理监控系统	Univac LAPC， IBM 7030
第三代 (1965—1974年)	SSI和MSI 多层印刷电路、微程序	流水线、Cache 先行处理、系列机	多道程序 分时操作系统	IBM 360/370， CDC 6600/7600
第四代 (1975—1990年)	LSI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
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



应用对系统结构的影响



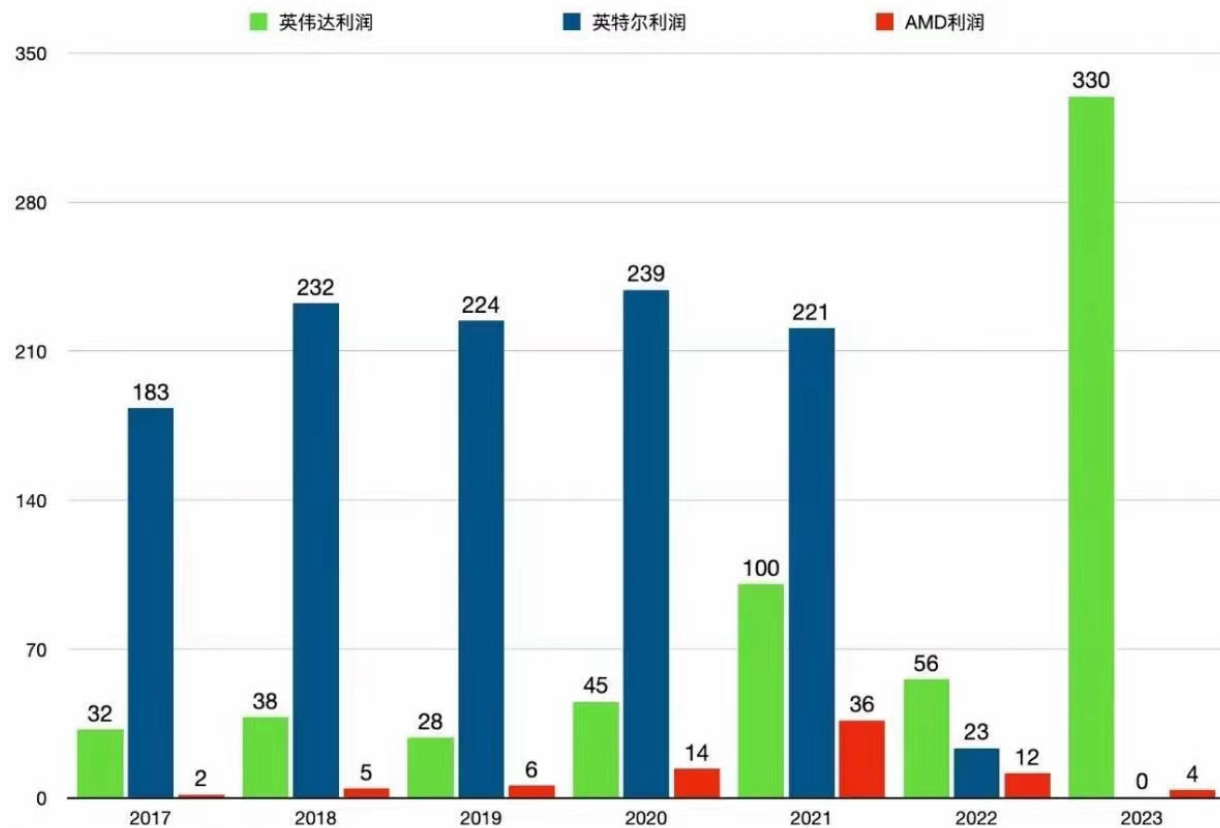
- 应用需求是促使系统结构发展的**最根本动力**
- 随时间推移、工艺提升，部件价格会逐渐降低
- 不同的应用对系统结构设计提出**不同的要求**
 - 高结构化的数值计算：气象模型、流体动力学、有限元分析
 - 非结构化的数值计算：蒙特卡洛模拟、稀疏矩阵
 - 大存储容量和输入输出密集的问题：数据库系统、事务处理系统
 - 人工智能：人脸识别→DeepSeek、Sora



应用对系统结构的影响 (Cont'd)



英伟达、英特尔、AMD营业利润变化 (2017-2023)



资料来源：公司财报，《财经十一人》

制表：吴俊宇

备注：1、英伟达财年为每年1月至次年1月，如2024财年为2023年1月至2024年1月

2、为便于统计，英伟达选取财年数据口径





- **系统结构的发展**
 - 冯诺伊曼架构；并行性开发的不同方式
- **系统结构的分类**
 - Flynn分类、冯氏分类、汉德勒分类
- **系统结构的性能评测**
 - 量化指标、基准测试、比较方法
- **系统结构的设计步骤、原理、思路**
 - 4大量化原理
- **软件、硬件、应用对系统结构的影响**





- **书面作业**

- 课本P28—5、7、9、11
- *后续会在课程群中同步发送题目内容

- **课后阅读**

- 课本关于影响计算机系统结构发展的因素、并行性发展的段落

