

第1章 计算机体系结构的基本概念

1.1 计算机体系结构的概念

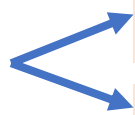
1.2 定量分析技术

1.3 计算机体系结构的发展

1.4 计算机体系结构中并行性的发展



第1章 计算机体系结构的基本概念

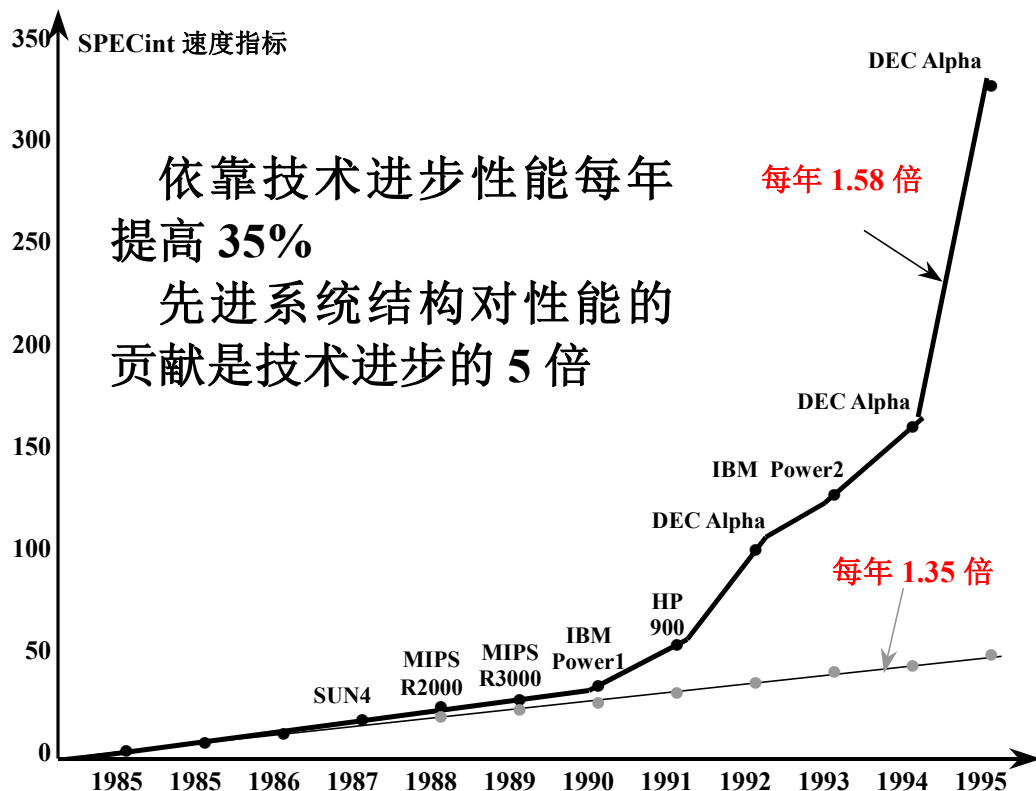
- 计算机技术的飞速发展得益于 

计算机制造技术的发展

计算机体系结构的创新
- 计算机技术经历了4个发展过程：

时 间	原 因	每年性能增长
1946年起的25年	两种因素都起着主要的作用	25%
20世纪70年代末 — 80年代初	大规模集成电路和微处理器出现,以集成电路为代表的制造技术的发展	约35%
20世纪80年代 中开始	RISC结构 的出现，体系结构不断更新和变革，制造技术不断发展	50%以上 维持了约16年
2002年以来	功耗问题（已经很大），可开发的指令级并行性已经很少，存储器访问速度提高缓慢	约20%

计算机体系结构在计算机的发展中有着极其重要的作用



体系结构的重大转折：
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

ILP（指令级并行）是隐式并行，程序员不可见

DLP（数据级并行）、TLP（线程级并行）、RLP（请求级并行）是显式并行，需要调整应用程序的结构才能开发显式并行

1.1 计算机体系结构的概念

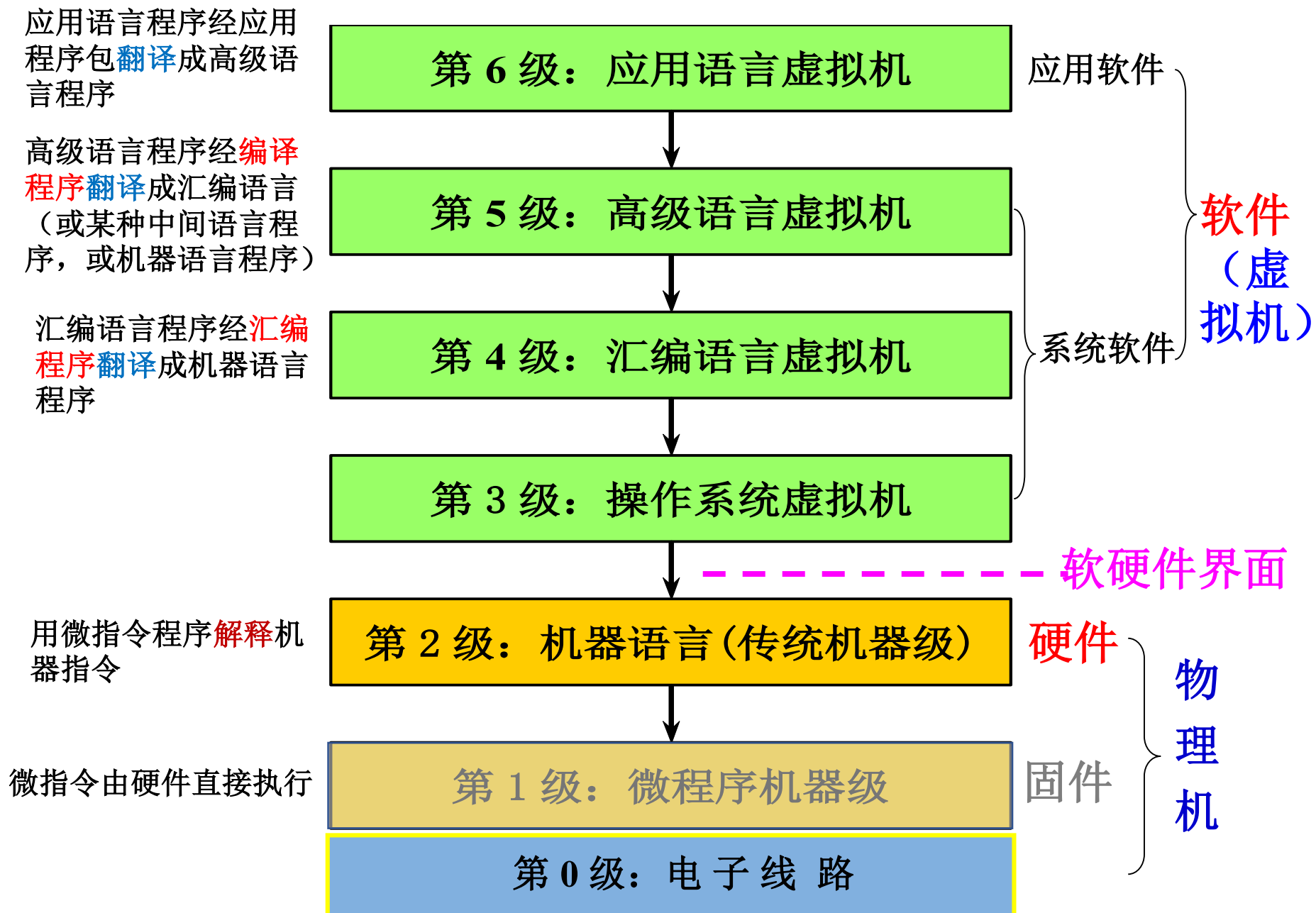
- 计算机系统 = 硬件/固件 + 软件
- 计算机语言从低级向高级发展

机器语言 → 汇编语言 → 高级语言 → 应用语言

高一级语言的语句相对于低一级语言来说功能更强，更便于应用，但又都以低级语言为基础。

- 从计算机语言的角度，把计算机系统按功能划分成**多级层次结构**
 - 每一层以一种语言为特征

计算机系统的层次结构



1.1 计算机体系结构的概念

- **虚拟机**：由软件实现的机器
- 语言实现的两种基本技术
 - **翻译**：先把 $N+1$ 级程序全部转换成 N 级程序后，再去执行新产生的 N 级程序，在执行过程中 $N+1$ 级程序不再被访问。
 - **解释**：每当一条 $N+1$ 级指令被译码后，就直接去执行一串等效的 N 级指令，然后再去取下一条 $N+1$ 级的指令，依此重复进行。
- **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**

1.1 计算机体系结构的概念

- 计算机体系结构的经典定义

程序员看到的计算机属性（即概念性结构与功能特性）

- 程序员：系统程序员（包括：汇编语言、机器语言、编译程序、操作系统）
- 看到的：编写出能在机器上正确运行的程序所必须了解的
- 计算机属性：指令系统、数据表示、寻址方式

- 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

1.1 计算机体系结构的概念

● **透明性：** 本来存在的事物或**属性**，从某种角度看似乎不存在

➤ 例如：CPU类型、型号、主存储器容量等

对应用程序员

透明

对系统程序员、硬件设计人员等

不透明

➤ 例如：浮点数表示、乘法指令

对高级语言程序员、应用程序员

透明

对汇编语言程序员、机器语言程序员

不透明

➤ 例如：数据总线宽度、微程序

对汇编语言程序员、机器语言程序员

透明

对硬件设计人员、计算机维修人员

不透明

1.1 计算机体系结构的概念

- Amdahl提出的**体系结构**：

传统**机器语言级程序员**所看到的计算机属性。

- **广义的体系结构定义**：**指令集结构、组成、硬件**
(计算机设计的3个方面)

- 这些属性是由硬件或固件完成的功能，程序员在了解这些属性后才能编写出在传统机器上正确运行的程序——**实质上是确定计算机系统中软硬件的界面**

1.1 计算机体系结构的概念

- 对于通用寄存器型机器来说，这些属性主要是指：
 - **指令系统**：包括机器指令的操作类型和格式、指令间的排序和控制机构等。
 - **数据表示**：硬件能直接识别和处理的**数据类型**。
 - **寻址规则**：包括最小寻址单元、寻址方式及其表示。
 - **寄存器定义**：包括各种寄存器的定义、数量和使用方式
 - **中断系统**：中断的类型和中断响应硬件的功能等
 - **机器工作状态的定义和切换**：如管态和目态等
 - **存储系统**：主存容量、程序员可用的最大存储容量等
 - **信息保护**：包括信息保护方式和硬件对信息保护的支持
 - **I/O结构**：包括I/O连结方式、处理机/存储器与I/O设备之间数据传送的方式和格式以及I/O操作的状态等

1.1 计算机体系结构的概念

- 计算机体系结构概念的实质：
 - 确定计算机系统中软、硬件的界面，界面之上是软件实现的功能，界面之下是硬件和固件实现的功能。
- 计算机系统由软件、硬件和固件组成，**它们在功能上是等价的。**
 - 同一种功能可以用硬件实现，也可以用软件或固件实现
 - 不同的实现只是性能和价格不同，他们的体系结构是相同的

1.1 计算机体系结构的概念

● 计算机组成和计算机实现

1. 计算机体系结构：计算机系统的软、硬件的界面

即机器语言程序员所看到的传统机器级所具有的属性。

2. 计算机组成：计算机体系结构的逻辑实现

- 从内部研究计算机系统：包含物理机器级中的数据流和控制流的组成以及逻辑设计等
- 计算机组成是指计算机主要部件的类型、数量、组成方式、控制方式和信息流动方式及其相互连接构成的系统
- 着眼于：物理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之间的联系

1.1 计算机体系结构的概念

➤ 计算机组成主要研究数据和指令的组织，基本运算的算法，数据的存取、传送和加工处理，数据流和指令流的控制方式等。包括：

- 确定数据通路的宽度
- 确定各种操作对功能部件的共享程度
- 确定专用的功能部件
- 确定功能部件的并行度
- 设计缓冲和排队策略
- 设计控制机构
- 确定采用何种可靠性技术

1.1 计算机体系结构的概念

3. 计算机实现：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，专用器件的设计，模块、插件、底板的划分与连接，信号传输技术，电源、冷却及整机装配技术，制造工艺及技术等
- **着眼于：** 器件技术（起主导作用）、微组装技术。

一种体系结构可以有多种组成；
一种组成可以有多种物理实现。

随着技术、器件和应用的发展，计算机体系结构、计算机组成和计算机实现三者之间的界限越来越模糊。

1.1 计算机体系结构的概念

4. 系列机

由同一厂家生产的具有相同体系结构、但具有不同组成和实现的一系列不同型号的计算机。

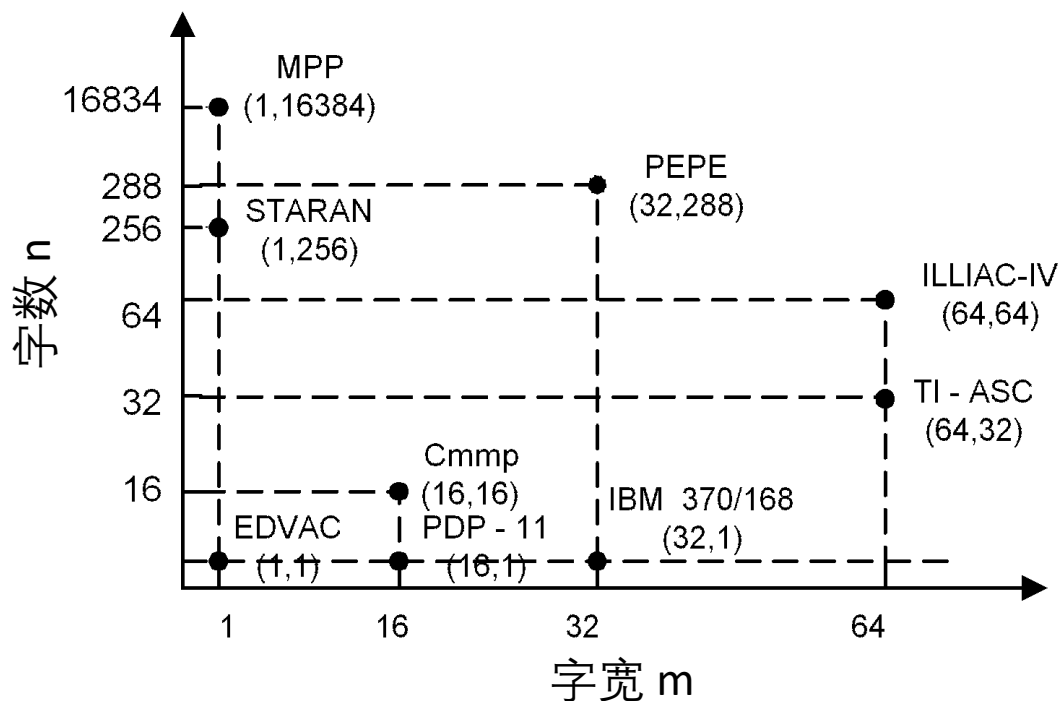
例如，IBM公司的IBM 370系列，Intel公司的x86系列
SUN公司的SPARC系列等

1.1 计算机体系结构的概念

- 常见的计算机体系结构分类法有两种：

- **冯氏分类法**：用系统的**最大并行度**对计算机进行分类

- **Flynn分类法**：按照指令流和数据流的多倍性进行分类



最大并行度：计算机系统在单位时间内能够处理的最大的二进制位数。

用平面直角坐标系中的一个点代表一个计算机系统，其横坐标表示字宽 (m 位)，纵坐标表示一次能同时处理的字数 (n 字)， $m \times n$ 就表示了其**最大并行度**。

1.1 计算机体系结构的概念

m 字宽
n 字数

(1)字串位串WSBS(Word Serial and Bit Serial)

串行计算机； $m = 1, n = 1$ ； 如：EDVAC(1, 1)

(2)字并位串WPBS(Word Parallel and Bit Serial): 位片处理

并行计算机、MPP、相联计算机； $m = 1, n > 1$ ；

如：MPP(1, 16384), STARAN(1, 256), DAP

(3)字串位并WSBP(Word Serial and Bit Parallel): 字片处理

传统单处理机； $m > 1, n = 1$ ； 如：Pentium(32, 1)

(4)字并位并WPBP(Word Parallel and Bit Parallel)

全并行计算机； $m > 1, n > 1$ ； 如：ASC(64, 32),

ILLIAC IV(64, 64), PEPE(32, 288), Cmmp(16, 16)

● 冯氏分类法主要缺点：

仅考虑数据并行，没有考虑指令，任务，作业的并行

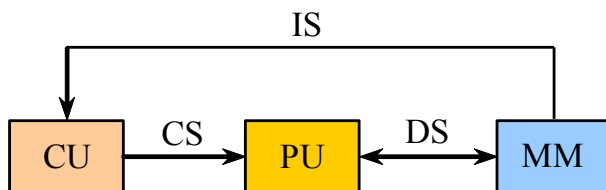
1.1 计算机体系结构的概念

➤ **Flynn分类法：**按照**指令流**和**数据流**的**多倍性**进行分类。

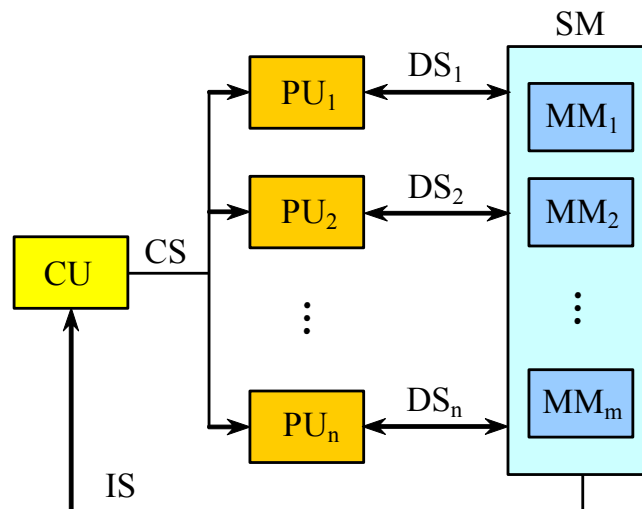
- ✓ **指令流：**计算机执行的指令序列。
- ✓ **数据流：**由指令流调用的数据序列。
- ✓ **多倍性：**在系统受限的部件上，同时处于同一执行阶段的指令或数据的最大数目。
- ✓ Flynn分类法把计算机系统的结构分为4类：
 - 单指令流单数据流(SISD)
 - 单指令流多数据流(SIMD)
 - 多指令流单数据流(MISD)
 - 多指令流多数据流(MIMD)

4类计算机的基本结构

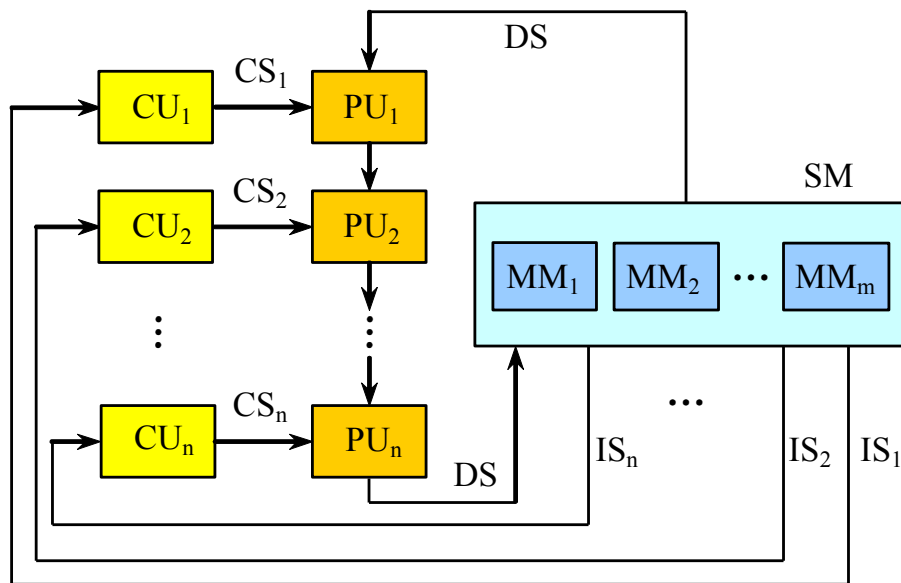
IS: 指令流, DS: 数据流, CS: 控制流,
CU: 控制部件, PU: 处理部件,
MM和SM: 存储器。



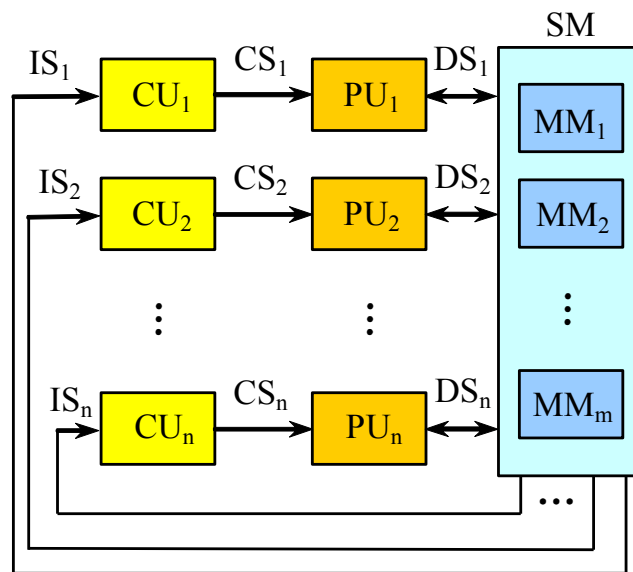
(a) SISD 计算机



(b) SIMD 计算机



(c) MISD 计算机



(d) MIMD 计算机

第1章 计算机体系结构的基本概念

1.1 计算机体系结构的概念

1.2 定量分析技术



1.3 计算机体系结构的发展

1.4 计算机体系结构中并行性的发展

- 系统设计的定量原理
 - 以经常性事件为重点
 - Amdahl定律
 - CPU性能公式
 - 程序的局部性原理
- 计算机系统的性能评测
- 基准测试程序
- 性能比较

1.2 定量分析技术

1.2.1 计算机系统设计的定量原理 (4个)

(1) 以经常性事件为重点

- 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。

(2) Amdahl定律:

$$\text{加速比} = \frac{1}{(1 - \text{可改进比例}) + \frac{\text{可改进比例}}{\text{部件加速比}}}$$

(3) CPU性能公式

$$\text{CPU时间} = \text{指令数} \times \text{CPI} \times \text{时钟周期}$$

(4) 程序的局部性原理

时间局部性、空间局部性

1.2 定量分析技术

(2) Amdahl定律

- 加快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。
- **系统性能加速比:**

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

1.2 定量分析技术

(2) Amdahl定律

- **加速比**依赖于两个因素：

- **可改进比例**：在改进前的系统中，可改进部分的执行时间在总的执行时间中所占的比例（总是 ≤ 1 ）
 - 例如：一个需运行60秒的程序中有20秒的运算可以加速，那么这个比例就是 $20/60$ 。
- **部件加速比**：可改进部分改进以后性能提高的倍数。它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 > 1 ）
 - 例如：若系统改进后，可改进部分的执行时间是2秒，而改进前其执行时间为5秒，则部件加速比为 $5/2$ 。

1.2 定量分析技术

- 改进后程序的总执行时间

总执行时间_{改进后} = 不可改进部分的执行时间 +
可改进部分改进后的执行时间

总执行时间_{改进后} = (1 - 可改进比例) × 总执行时间_{改进前}
+ $\frac{\text{可改进比例} \times \text{总执行时间}_{\text{改进前}}}{\text{部件加速比}}$

= $\left[(1 - \text{可改进比例}) + \frac{\text{可改进比例}}{\text{部件加速比}} \right] \times \text{总执行时间}_{\text{改进前}}$

1.2 定量分析技术

系统加速比为改进前与改进后总执行时间之比

$$\begin{aligned}\text{加速比} &= \frac{\text{总执行时间}_{\text{改进前}}}{\text{总执行时间}_{\text{改进后}}} \\ &= \frac{1}{(1 - \text{可改进比例}) + \frac{\text{可改进比例}}{\text{部件加速比}}}\end{aligned}$$

1.2 定量分析技术

例1.1 将计算机系统中某一功能的处理速度提高到原来的**20**倍，但该功能的处理时间仅占整个系统运行时间的**40%**，则采用此提高性能的方法后，能使整个系统的性能提高多少？

解：由题可知，可改进比例 = 40% = 0.4，

部件加速比 = 20

根据Amdahl定律可知：

$$\text{总加速比} = \frac{1}{0.6 + \frac{0.4}{20}} = 1.613$$

能使整个系统的性能提高到原来的**1.613**倍。

1.2 定量分析技术

例1.2 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**20**倍，而系统运行某一程序的整体性能提高到原来的**5**倍，试计算该程序中浮点操作所占的比例。

解： 由题可知，部件加速比 = 20，系统加速比 = 5

根据Amdahl定律可知

$$5 = \frac{1}{(1 - \text{可改进比例}) + \frac{\text{可改进比例}}{20}}$$

由此可得：可改进比例 = 84.2%

即程序中浮点操作所占的比例为**84.2%**

1.2 定量分析技术

(2) Amdahl定律：一种性能改进的递减规则

- 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

➤ 例如：可改进比例 $F=50\%$
$$S_p = \frac{1}{0.5 + \frac{0.5}{S}}$$

部件加速比S	1	2	4	8
总加速比 S_p	1	1.333...	1.6	1.777...

- **重要推论：**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 $1/(1-F)$

第1章 计算机体系结构的基本概念

1.1 计算机体系结构的概念

1.2 定量分析技术

1.3 计算机体系结构的发展

1.4 计算机体系结构中并行性的发展



1.2 定量分析技术

1.2.1 计算机系统设计的定量原理 (4个)

(1) 以经常性事件为重点： 优化原则

(2) Amdahl定律： 加速比 =
$$\frac{1}{(1 - \text{可改进比例}) + \frac{\text{可改进比例}}{\text{部件加速比}}}$$

(3) CPU性能公式： CPU时间 = 指令数 × CPI × 时钟周期

(4) 程序的局部性原理： 时间局部性、空间局部性

1.2 定量分析技术

(3) CPU性能公式

- 执行一个程序所需的CPU时间

CPU时间 = 执行程序所需的**时钟周期数** × **时钟周期时间**

其中，时钟周期时间是系统时钟频率的倒数。

- 每条指令执行的平均时钟周期数**CPI** (Cycles Per Instruction)

CPI = 执行程序所需的时钟周期数 / IC

IC: 所执行的指令条数

- 程序执行的CPU时间

CPU时间 = **IC** × **CPI** × **时钟周期时间**

1.2 定量分析技术

计算机性能评价：影响计算机性能的因素

$$\text{CPU执行时间} = \text{指令数} \times \text{CPI} \times \text{时钟周期}$$

影响因素	影响
算法	指令数、CPI
程序设计语言	指令数、CPI
编译器	指令数、CPI
ISA 指令集体系结构	指令数、CPI、时钟周期
硬件实现	CPI、时钟周期

1.2 定量分析技术

性能有关的参数:

I. 指令数

II. CPI

III. 时钟频率

测试时的选择:

A. 测试方法

B. 编译器技术

C. 硬件技术

对于每一列性能，选择哪一行测试时的影响最小？原因？

	I.指令数	II. CPI	III. 时钟频率
1.	A	B	C
2.	A	C	B
3.	B	A	C
4.	B	C	A
5.	C	A	B
6.	C	B	A

1.2 定量分析技术

- CPU的性能取决于3个参数

- **时钟周期时间**：取决于硬件实现技术和计算机组成
- **CPI**：取决于计算机组成和指令集结构
- **IC**：取决于指令集结构和编译技术

- 对CPU性能公式进行进一步细化

假设：计算机系统有 n 种指令；

CPI_i ：第 i 种指令的处理时间；

IC_i ：在程序中第 i 种指令出现的次数；

则：

$$\text{CPU时钟周期数} = \sum_{i=1}^n (CPI_i \times IC_i)$$

1.2 定量分析技术

CPU时间 = 执行程序所需的时钟周期数 × 时钟周期时间

$$= \sum_{i=1}^n (\text{CPI}_i \times \text{IC}_i) \times \text{时钟周期时间}$$

CPI可以表示为

$$\text{CPI} = \frac{\text{时钟周期数}}{\text{IC}} = \frac{\sum_{i=1}^n (\text{CPI}_i \times \text{IC}_i)}{\text{IC}} = \sum_{i=1}^n (\text{CPI}_i \times \frac{\text{IC}_i}{\text{IC}})$$

- 其中(IC_i / IC)反映了第*i*种指令在程序中所占的比例

1.2 定量分析技术

例1.3 某台计算机只有Load/Store指令能对存储器进行读/写操作，其它指令只对寄存器进行操作。根据程序跟踪实验结果，已知每种指令所占的比例及CPU时钟周期数如下：

指令类型	指令所占比例	CPI
算逻指令	43%	1
LOAD指令	21%	2
SRORE指令	12%	2
转移指令	24%	2

求平均CPI。

$$\begin{aligned} \text{CPI} &= 1 \times 0.43 + 2 \times 0.21 + 2 \times 0.12 + 2 \times 0.24 \\ &= 0.43 + 0.42 + 0.24 + 0.48 \\ &= 1.57 \end{aligned}$$

1.2 定量分析技术

例1.4 考虑条件分支指令的两种不同设计方法：

- (1) **CPU_A**: 通过比较指令设置条件码，然后测试条件码进行分支
- (2) **CPU_B**: 在分支指令中包括比较过程

在这两种CPU中，条件分支指令都占用**2**个时钟周期，而所有其他指令占用**1**个时钟周期。对于**CPU_A**，执行的指令中分支指令占20%；由于每条分支指令之前都需要有比较指令，因此比较指令也占20%。由于**CPU_A**在分支时不需要比较，**CPU_B**的时钟周期时间是**CPU_A**的**1.25**倍。

问：哪一个CPU更快？如果**CPU_B**的时钟周期时间只是**CPU_A**的**1.1**倍，哪一个CPU更快呢？ **CPU_A**

1.2 定量分析技术

解: 不考虑所有系统问题, **CPU时间 = 指令数 × CPI × 时钟周期**

占用2个时钟周期的分支指令占总指令的20%,
剩下的指令占用1个时钟周期。所以:

$$CPI_A = 0.2 \times 2 + 0.80 \times 1 = 1.2$$

则CPU_A性能为:

$$\text{总CPU时间}_A = IC_A \times 1.2 \times \text{时钟周期}_A$$

根据假设, 有

$$\text{时钟周期}_B = 1.25 \times \text{时钟周期}_A$$

在CPU_B中没有独立的比较指令, 所以CPU_B的程序量为CPU_A的80%, 分支指令的比例为: **20%/80%=25%**

1.2 定量分析技术

这些分支指令占用2个时钟周期，而剩下的75%的指令占用1个时钟周期，因此：

$$CPI_B = 0.25 \times 2 + 0.75 \times 1 = 1.25$$

因为CPU_B不执行比较，故： $IC_B = 0.8 \times IC_A$

因此CPU_B性能为

$$\begin{aligned} \text{总CPU时间}_B &= IC_B \times CPI_B \times \text{时钟周期}_B \\ &= 0.8 \times IC_A \times 1.25 \times (1.25 \times \text{时钟周期}_A) \\ &= 1.25 \times IC_A \times \text{时钟周期}_A \end{aligned}$$

- 在这些假设之下，尽管CPU_B执行指令条数较少，CPU_A因为有着更短的时钟周期，所以比CPU_B快

1.2 定量分析技术

如果CPU_B的时钟周期时间仅仅是CPU_A的1.1倍，则

$$\text{时钟周期}_B = 1.10 \times \text{时钟周期}_A$$

CPU_B的性能为:

$$\begin{aligned}\text{总CPU时间}_B &= IC_B \times CPI_B \times \text{时钟周期}_B \\ &= 0.8 \times IC_A \times 1.25 \times (1.10 \times \text{时钟周期}_A) \\ &= 1.10 \times IC_A \times \text{时钟周期}_A\end{aligned}$$

因此CPU_B由于执行更少指令条数，比CPU_A运行更快。

1.2 定量分析技术

例1.5： 如果FP操作比例为25%，FP的平均CPI_{FP原}=4，其他指令的平均CPI_{其它}为1.33。FPSQR操作比例为2%，而FPSQR的CPI_{SQR原}=20。

现有两种改进方案：

(1) 提高FP操作的速度，使其增加一倍，即CPI_{FP改}=2；

(2) 提高FPSQR的速度10倍，即CPI_{SQR改}=2

试比较这两种改进的方案。

解： 求出改进前后的CPI值进行比较

$$\begin{aligned}\text{CPI (原)} &= \sum_{i=1}^n \left(\text{CPI}_i \times \frac{I_i}{I_N} \right) \\ &= (4 \times 0.25) + (1.33 \times 0.75) = 2\end{aligned}$$

两种改进方案后的CPI

1.2 定量分析技术

方案1： 改进所有浮点操作指令使FP的平均CPI值提高一倍，即：

$$CPI_{FP改}=2$$

整个程序在改进后的值为：

$$CPI(改)=(2 \times 0.25) + (1.33 \times 0.75) = 1.5$$

或

$$\begin{aligned} &= CPI(原) - 0.25 \times (CPI_{FP原} - CPI_{FP改}) \\ &= 2.0 - 0.25 \times (4 - 2) = 1.5 \end{aligned}$$

方案2： 只改进FPSQR指令，使其CPI值提高10倍，即 $CPI_{SQR改}=2$

整个程序改进后的值应为：

$$\begin{aligned} CPI(改) &= CPI(原) - 0.02 \times (CPI_{SQR原} - CPI_{SQR改}) \\ &= 2 - 0.02 \times (20 - 2) = 1.64 \end{aligned}$$

● 两者比较还是方案1更好

1.2 定量分析技术

- 提高计算机性能的途径

$$\text{CPU执行时间} = \text{指令数} \times \text{CPI} \times \text{时钟周期}$$

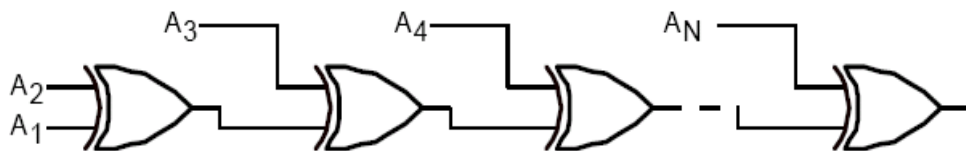
- 可以通过减少公式中任意一项来提高处理器的性能
- 但是，公式中的三项并不是相互独立的，它们之间有着复杂的联系，减少三项中的任意一项都有可能增加其他两项

1.2 定量分析技术

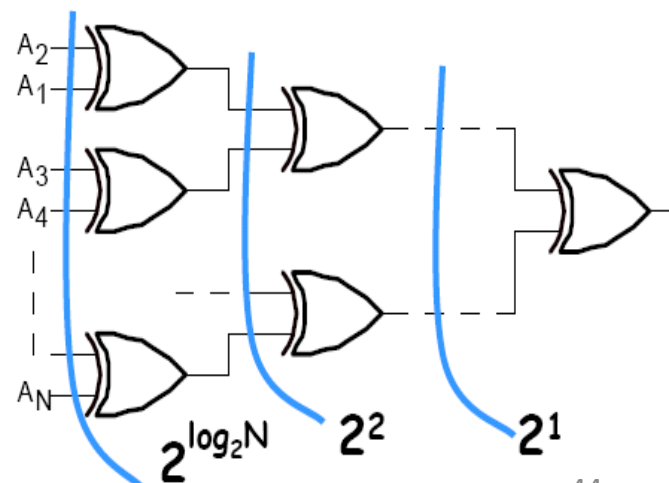
● 提高计算机性能的途径

- 一些技术可在保持其他两项不变的前提下减少其中一项
 - ✓ 采用优化编译技术，在目标代码中消除冗余代码，可以减少指令数，但是并不提高CPI和时钟周期
 - ✓ 采用快速电路技术或更为先进的结构减少信号传输延迟，可以减少时钟周期，但是并不提高CPI和指令数

Let 's just design an N input Xor gate.....



新的设计速度更快而面积不变



1.2 定量分析技术

● 提高计算机性能的途径

- 一些技术可在保持其他两项不变的前提下减少其中一项
 - ✓ 例如，指令集可以包含更多复杂的指令，使每条指令执行更多的动作，可以减少指令数 → CISC
 - ✓ 虽然指令数减少，但是执行部件的复杂性增加，从而导致CPI和时钟周期的增加
- 只有在减少项的作用大于增加项的情况下，才能获得性能的提高

CISC: Complex Instruction Set Computer

RISC: Reduced Instruction Set Computer

1.2 定量分析技术

● 提高计算机性能的途径

➤ 减少CPI的愿望激发了许多体系结构和微体系结构(即体系结构的逻辑实现)技术

✓ 体系结构方面：采用精简的指令集，减少每条指令的复杂性，从而减少CPI → RISC

✓ 微体系结构方面：同时重叠执行多条指令

流水线技术、超标量技术

CISC: Complex Instruction Set Computer

RISC: Reduced Instruction Set Computer

1.2 定量分析技术

(4) 程序的局部性原理

- 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。
 - 常用的一个经验规则
程序执行时间的90%都是在执行程序中的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信息在空间上相邻或者临近。

1.2 定量分析技术

1.2.2 计算机系统的性能评测

➤ 执行时间和吞吐率

如何评测一台计算机的性能，与测试者看问题的角度有关

- 用户关心的是：单个程序的**执行时间**

执行单个程序所花的时间--少

- 数据处理中心的管理人员关心的是：**吞吐率**

单位时间里能够完成的任务--多

1.2 定量分析技术

- 假设两台计算机为X和Y，**X比Y快**的意思是：
对于给定任务，X的执行时间比Y的执行时间少

X的性能是Y的 **n 倍**，即：

$$\frac{\text{执行时间Y}}{\text{执行时间X}} = n$$

- 而**执行时间**与**性能**成反比，即：

$$n = \frac{\text{执行时间Y}}{\text{执行时间X}} = \frac{\frac{1}{\text{性能Y}}}{\frac{1}{\text{性能X}}} = \frac{\text{性能X}}{\text{性能Y}}$$

1.2 定量分析技术

- 执行时间可以有多种定义：
 - 计算机完成某一任务所花费的**全部时间**，包括磁盘访问、存储器访问、输入/输出、操作系统开销等。
 - **CPU时间**：CPU执行所给定的程序所花费的时间，不包含I/O等待时间以及运行其他程序的时间。
 - ✓ **用户CPU时间**：用户程序所耗费的CPU时间。
 - ✓ **系统CPU时间**：用户程序运行期间操作系统耗费的CPU时间。

1.2 定量分析技术

● MIPS和MFLOPS

- **IPS**: 每秒执行的指令条数。
- **MIPS**: 以百万来计量。
- **MFLOPS**: 每秒百万次浮点操作次数。

$$\text{MIPS} = \frac{\text{指令总数}}{\text{执行指令所需的时间} \times 10^6} = \frac{\text{时钟频率}}{\text{CPI} \times 10^6}$$

$$\text{MFLOPS} = \frac{\text{程序中的浮点操作次数}}{\text{执行时间} \times 10^6}$$

MIPS: Million Instructions Per Second

vs: Microprocessor without Interlocked Piped Stages architecture

1.2 定量分析技术

1.2.3 基准测试程序

- 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**（如编译器）。
- 以前常采用简化了的程序，例如：
 - **核心测试程序**：从真实程序中选出的关键代码段构成的小程序。
 - **小测试程序**：简单的只有几十行的小程序。
 - **合成的测试程序**：人工合成出来的程序。
Whetstone与**Dhrystone**是最流行的合成测试程序。
- ✓ **Whetstone**：用FORTRAN语言编写的综合性测试程序，主要由执行浮点运算、整数算术运算、功能调用、数组变址、条件转移和超越函数的程序组成
- ✓ **Dhrystone**：测试处理器整型运算性能的最常见基准程序之一

1.2 定量分析技术

- 从测试性能的角度来看，上述测试程序就不可信了
 - 这些程序比较小，具有片面性
 - 体系结构设计者和编译器的设计者可以“合谋”把他们的计算机面向这些测试程序进行优化设计，使得该计算机显得性能更高。**峰值性能**
- 性能测试的结果除了和采用什么测试程序有关以外，还和**在什么条件下进行测试**有关。
- 基准测试程序设计者对制造商的要求
 - 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

1.2 定量分析技术

- **一个问题：**是否允许修改测试程序的源程序
三种不同的处理方法：
 - 不允许修改。
 - 允许修改，但因测试程序很复杂或者很大，几乎是无法修改。
 - 允许修改，只要保证最后输出的结果相同。
- **基准测试程序套件(Benchmark)：**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
(美国的标准性能评估公司开发)

1.2 定量分析技术

- 台式计算机的基准测试程序套件可以分为两大类：
 - ✓ 处理器性能测试程序
 - ✓ 图形性能测试程序
- **SPEC89**：用于测试处理器性能的10个程序（4个整数程序，6个浮点程序）
- 演化出了4个版本
 - ✓ SPEC92：20个程序
 - ✓ SPEC95：18个程序
 - ✓ SPEC2000：26个程序
 - ✓ SPEC CPU2006：29个程序

1.2 定量分析技术

- **SPEC CPU2006：29个程序**
 - 整数程序12个（CINT2006）：9个C，3个C++
 - 浮点程序17个（CFP2006）：6个FORTRAN，4个C++，3个C，4个C和FORTRAN混合编程
- **SPEC测试程序套件中的其他一系列测试程序组件**
 - **SPECapc**：用于测试图形密集型应用的性能
 - **SPECSFS**：用于NFS（网络文件系统）文件服务器的测试程序。它不仅测试处理器的性能，而且测试I/O系统的性能。它重点测试吞吐率
 - **SPECWeb**：Web服务器测试程序
 - **SPECviewperf**：用于测试图形系统支持OpenGL库的性能

1.2 定量分析技术

- **事务处理（TP）性能测试基准程序：**用于测试计算机在事务处理方面的能力，包括数据库访问和更新等。
 - 20世纪80年代中期，一些工程师成立了称为**TPC**的独立组织。目的是开发用于TP性能测试的真实而又公平的基准程序。
 - 先后发布了多个版本：
TPC-A、TPC-C、TPC-H、TPC-W、TPC-App等
(主要是用于测试服务器的性能)

1.2 定量分析技术

- **SPEC CPU 2017: 4大种类共43个测试**

SPECrate®2017 Int	SPECspeed®2017 Int	编译语言	用例含义
500.perlbench_r	600.perlbench_s	C	Perl解释程序
502.gcc_r	602.gcc_s	C	GNU C 编译器
505.mcf_r	605.mcf_s	C	路线规划
520.omnetpp_r	620.omnetpp_s	C++	离散事件模拟 - 计算机网络
523.xalancbmk_r	623.xalancbmk_s	C++	通过 XSLT 进行 XML 到 HTML 转换
525.x264_r	625.x264_s	C	视频压缩
531.deepsjeng_r	631.deepsjeng_s	C++	人工智能: α -beta 树搜索 (国际象棋)
541.leela_r	641.leela_s	C++	人工智能: Monte Carlo树搜索 (GO)
548.exchange2_r	648.exchange2_s	Fortran	人工智能: 递归式解决方案发生器 (数独)
557.xz_r	657.xz_s	C	一般数据压缩

SPECrate®2017 浮点	SPECspeed®2017 浮点	编译语言	用例含义
503.bwaves_r	603.bwaves_s	Fortran	爆炸建模
507.cactuBSSN_r	607.cactuBSSN_r	C++, C, Fortran	物理: 相对论
508.namd_r		C++	分子动力学
510.parest_r		C++	生物医学成像: 有限元素的光学断层扫描
511.povray_r		C++, C	光线跟踪
519.lbm_r	619.lbm_s	C	流体力学
521.wrf_r	621.wrf_s	Fortran, C	天气预报
526.blender_r		C++, C	3D 渲染和动画
527.cam4_r	627.cam4_s	Fortran, C	大气建模
	628.pop2_s	Fortran, C	大规模海洋建模 (气候水平)
538.imagick_r	638.imagick_s	C	图像操作
544.nab_r	644.nab_s	C	分子动力学
549.fotonik3d_r	649.fotonik3d_s	Fortran	计算电磁学
554.roms_r	654.roms_s	Fortran	区域海洋建模

1.2 定量分析技术

- 用于测试基于Microsoft公司的**Windows**系列操作系统平台的测试套件
 - **PCMark04**：中央处理器测试组、内存测试组、图形芯片测试组、硬盘测试组等。
 - **Business Winstone 2004**：主要用于测试计算机系统商业应用的综合性能。
 - **Multimedia Content Creation Winstone 2004**：主要用于测试计算机系统多媒体应用的综合性能。
 - **SiSoft Sandra Pro 2004**：一套功能强大的系统分析评比工具，拥有超过30种以上的分析与测试模块。主要包括：CPU、存储器、I/O接口、I/O设备、主板等

1.2 定量分析技术

1.2.4 性能比较

两个程序在A、B、C三台计算机上的执行时间：

	A机	B机	C机
程序1	1	10	20
程序2	1000	10	20

如何比较这3台计算机的性能？

1.2 定量分析技术

1.2.4 性能比较

- 从表中可以得出：

- 执行程序1:

- A机的速度是B机的**10**倍
 - A机的速度是C机的**20**倍
 - B机的速度是C机的**2**倍

- 执行程序2:

- B机的速度是A机的**100**倍
 - C机的速度是A机的**50**倍
 - B机的速度是C机的**2**倍

	A机	B机	C机
程序1	1	10	20
程序2	1000	10	20

1.2 定量分析技术

	A机	B机	C机
程序1	1	10	20
程序2	1000	10	20

- **总执行时间：** 计算机执行所有测试程序的总时间
 - B机执行程序1和程序2的速度是A机的50.05倍
 - C机执行程序1和程序2的速度是A机的25.025倍
 - B机执行程序1和程序2的速度是C机的2倍
- **平均执行时间：** 各测试程序执行时间的算术平均值

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

其中， T_i ： 第*i*个测试程序的执行时间

n ： 测试程序组中程序的个数

1.2 定量分析技术

- **加权执行时间：** 各测试程序执行时间的加权平均值

$$A_m = \sum_{i=1}^n W_i \cdot T_i \quad \sum_{i=1}^n W_i = 1$$

其中， W_i ：第 i 个测试程序在测试程序组中所占的比重
 T_i ：该程序的执行时间

	A机	B机	C机	$W(1)$	$W(2)$	$W(3)$
程序1	1.00	10.00	20.00	0.50	0.909	0.999
程序2	1000.00	10.00	20.00	0.50	0.091	0.001
算术平均值 $A_m(1)$	500.50	10.00	20.00			
加权平均值 $A_m(2)$	91.91	10.00	20.00			
加权平均值 $A_m(3)$	2.00	10.00	20.00			

不同
权值

1.2 定量分析技术

1.2.1 计算机系统设计的定量原理 (4个)

(1) 以经常性事件为重点： 优化原则

(2) Amdahl定律： 加速比 =
$$\frac{1}{(1 - \text{可改进比例}) + \frac{\text{可改进比例}}{\text{部件加速比}}}$$

(3) CPU性能公式： CPU时间 = 指令数 × CPI × 时钟周期

(4) 程序的局部性原理： 时间局部性、空间局部性

第1章 计算机体系结构的基本概念

1.1 计算机体系结构的概念

1.2 定量分析技术

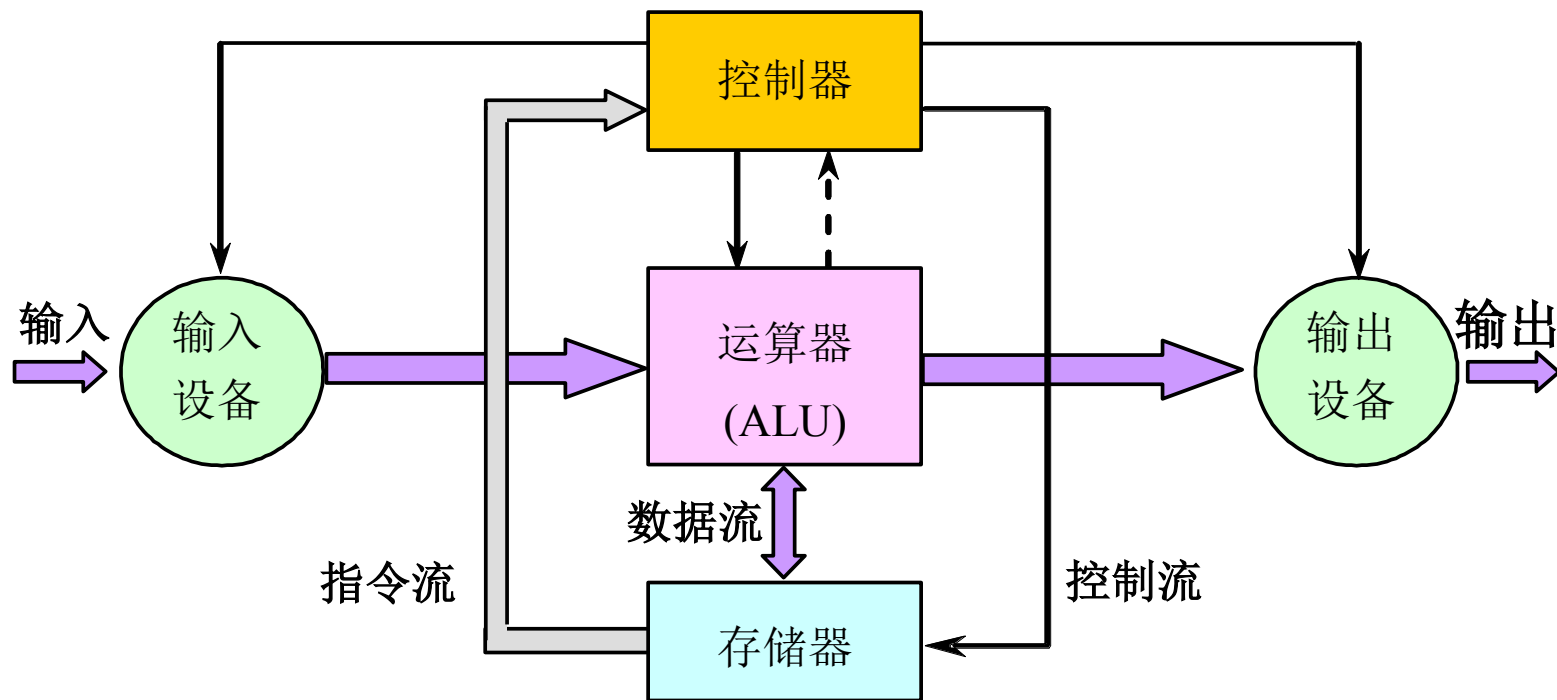
1.3 计算机体系结构的发展

1.4 计算机体系结构中并行性的发展

- 冯·诺依曼结构
- 软件对体系结构的影响
- 器件对体系结构的影响
- 应用对体系结构的影响
- 体系结构的生命周期

1.3 计算机体系结构的发展

1.3.1 冯·诺依曼结构



存储程序计算机的结构

1.3 计算机体系结构的发展

- **存储程序原理的基本点：指令驱动**

程序和数据预先存放在计算机存储器中，计算机一旦启动，就能按照程序指定的逻辑顺序执行这些程序，**自动**完成由程序所描述的处理工作。

- **冯·诺依曼结构的主要特点**

- 以运算器为中心
- 在存储器中，指令和数据同等对待
由指令组成的程序是可以修改的。
- 存储器是按地址访问、按顺序线性编址的一维结构，每个单元的位数是固定的。

1.3 计算机体系结构的发展

● 指令顺序执行

- 一般是按照指令在存储器中存放的顺序执行。
- 程序的分支由转移指令实现。
- 由指令计数器PC指明当前正在执行的指令在存储器中的地址。

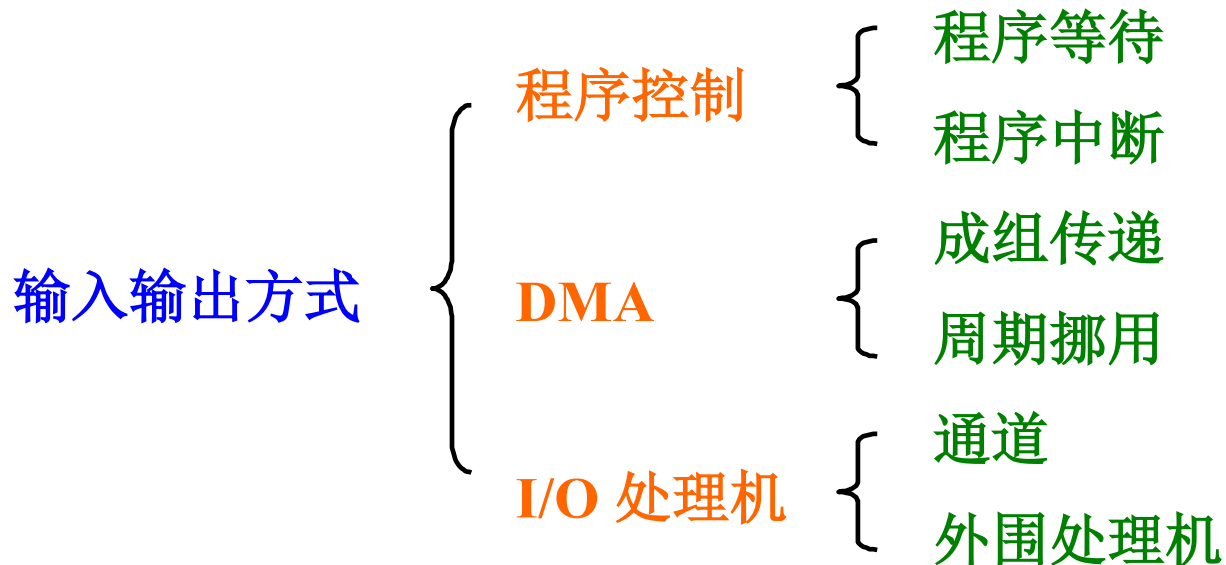
指令由操作码和地址码/操作数组成

指令和数据均以二进制编码表示，采用二进制运算

1.3 计算机体系结构的发展

- 对体系结构进行的改进

- (1) 输入/输出方式的改进



1.3 计算机体系结构的发展

(2) 采用并行处理技术

- 如何挖掘传统机器中的并行性？
- 在不同的级别采用并行技术（微操作级、指令级、线程级、进程级、任务级等）

(3) 存储器组织结构的发展

- **相联存储器（TLB）** 与相联处理机
- 通用寄存器组
- 高速缓冲存储器Cache

(4) 指令集的发展（两个方向）

- 复杂指令集计算机（CISC）
- **精简指令集计算机（RISC）**

1.3 计算机体系结构的发展

1.3.2 软件对体系结构的影响

- **软件的可移植性**：一个软件可以不经修改或者只需少量修改就可以由一台计算机移植到另一台计算机上正确地运行。差别只是执行时间的不同。
 - 我们称这两台计算机是**软件兼容**的。
- 实现**软件移植性**的常用方法：
 - (1) 采用系列机
 - (2) 模拟与仿真
 - (3) 统一高级语言

1.3 计算机体系结构的发展

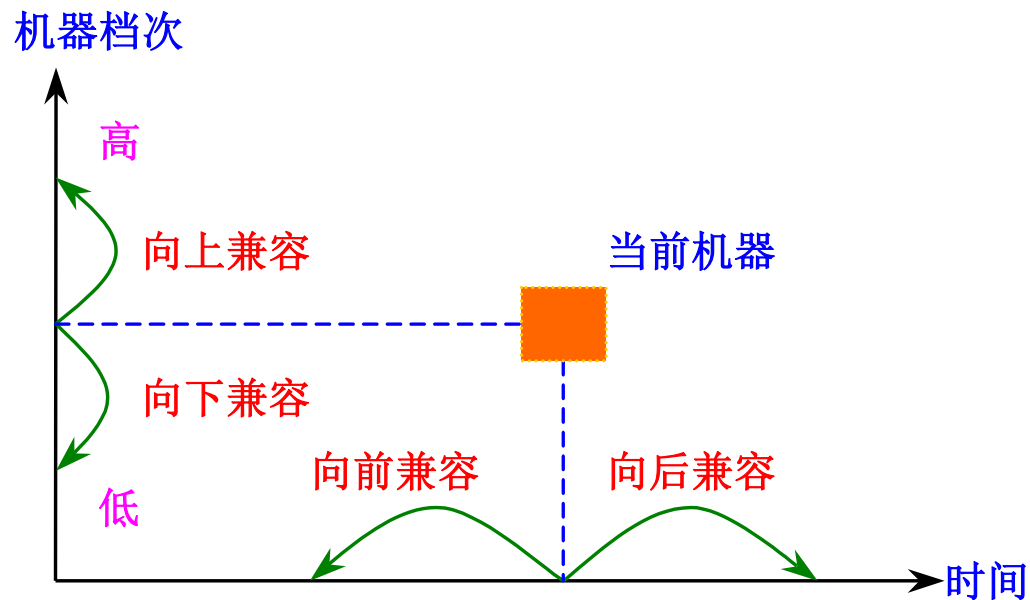
(1) 系列机

由同一厂家生产的具有相同的体系结构，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体系结构相对稳定与器件、硬件技术迅速发展的矛盾

- 软件兼容

- 兼容机：由不同公司厂家生产的具有相同体系结构的计算机。



1.3 计算机体系结构的发展

- 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，**必须做到**
- 向上兼容**尽量做到**
- 向前兼容和向下兼容，可以**不考虑**

1.3 计算机体系结构的发展

(2) 模拟和仿真

- 使软件能在具有**不同体系结构**的机器之间相互移植。
 - 在一种体系结构上实现另一种体系结构。
 - 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。

1.3 计算机体系结构的发展

- **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
 - 仿真只能在体系结构差距不大的机器之间使用

(3) 统一高级语言

- 实现软件移植的一种**理想的方法**
- 较难实现

1.3 计算机体系结构的发展

1.3.3 器件发展对体系结构的影响

- 摩尔定律

集成电路芯片集成的晶体管数目每隔18个月翻一番

- 计算机的分代主要以器件作为划分标准

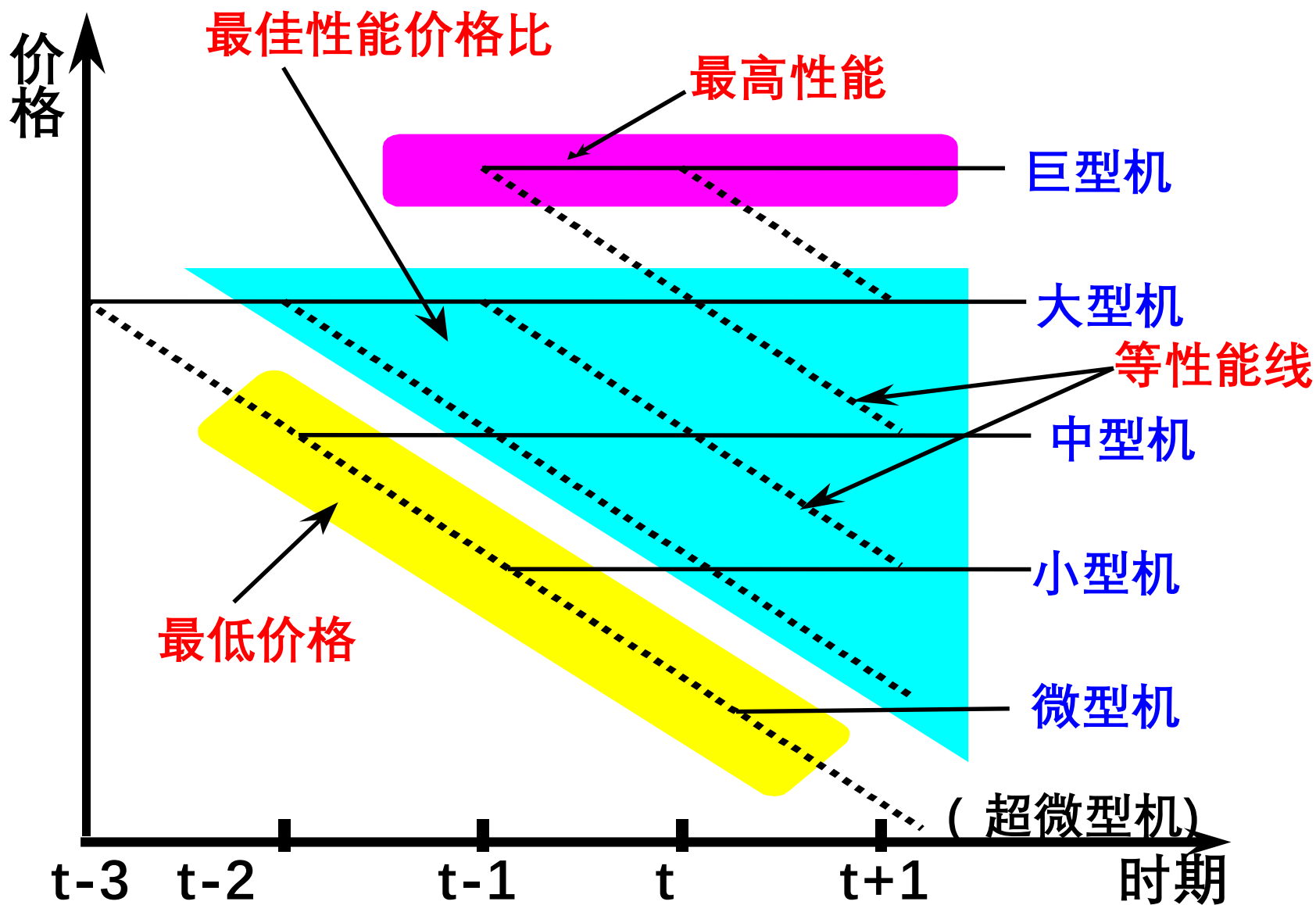
在器件、体系结构和软件技术等方面都有各自的特征

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)	小/中规模S/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 智能和逻辑推理	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
第六代	生物计算机、类脑计算机、量子计算机			

1.3 计算机体系结构的发展

1.3.4 应用对体系结构的影响

- 不同的应用对计算机体系结构的设计提出了不同的要求
- 应用需求是促使计算机体系结构发展的**最根本的动力**
- 一些特殊领域：**需要高性能的体系结构**
 - 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
 - 实时多因素问题：语音识别、图像处理、计算机视觉
 - 大存储容量和输入输出密集的问题：数据库、事务处理系统
 - 图形学和设计问题：计算机辅助设计
 - 人工智能：面向知识的系统、推理系统、深度学习等 (TPU)



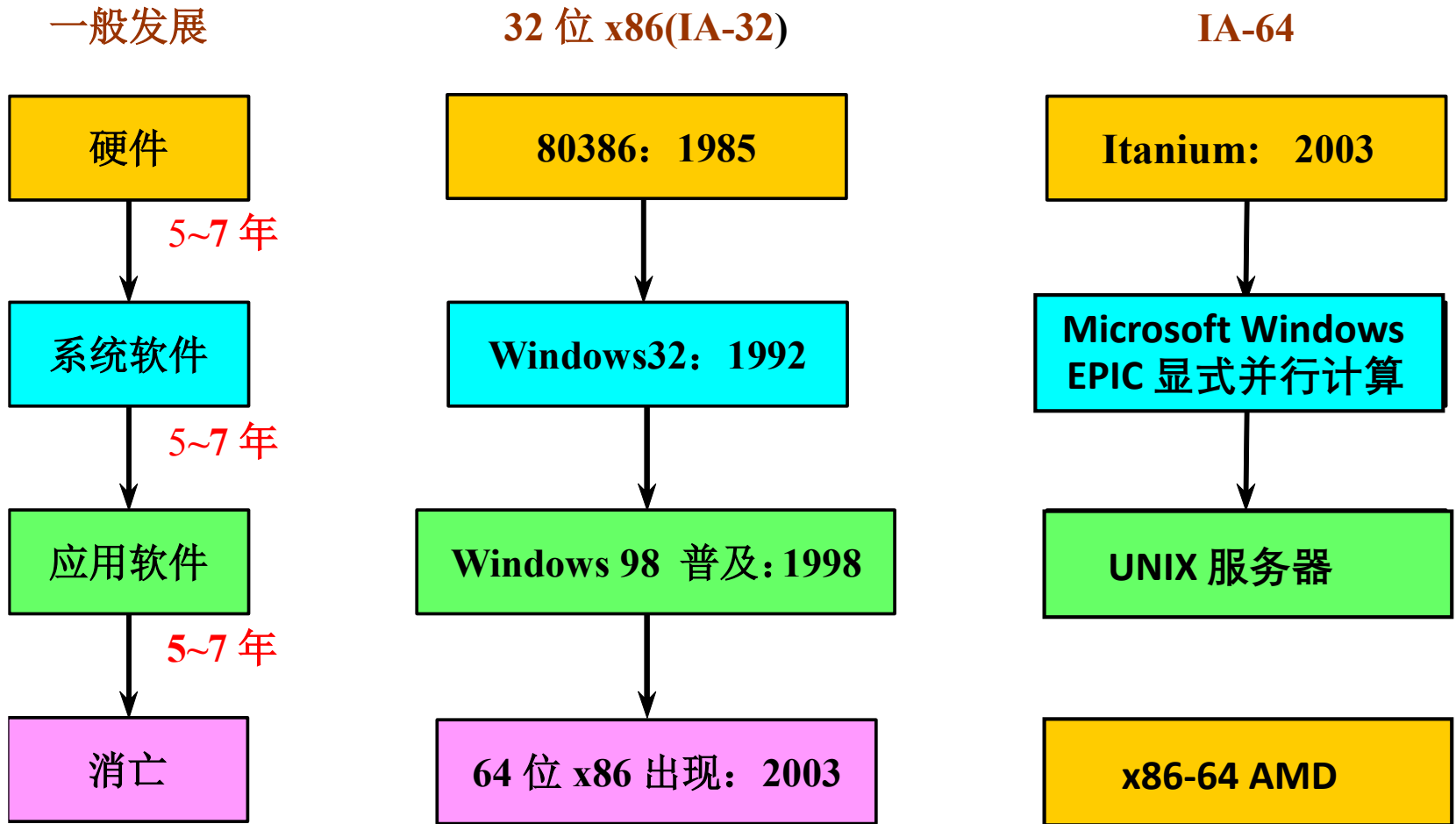
计算机性能随时间下移

1.3 计算机体系结构的发展

1.3.5 体系结构的生命周期

- 体系结构的**生命周期**：从诞生、发展、成熟到消亡
 - 从硬件成熟到系统软件成熟大约需要**5~7年**的时间
 - 从系统软件成熟到应用软件成熟，大约也需要**5~7年**时间。
 - 再过**5~7年**的时间，这种体系结构就不会作为主流体系结构存在了。
- Intel的x86系列微处理器中32位体系结构的发展

1.3 计算机体系结构的发展



Intel的x86系列微处理器系结构的发展

第1章 计算机体系结构的基本概念

1.1 计算机体系结构的概念

1.2 定量分析技术

1.3 计算机体系结构的发展

1.4 计算机体系结构中并行性的发展



- 并行性的概念
- 提高并行性的技术途径
- 单机系统中并行性的发展
- 多机系统中并行性的发展

1.4 计算机体系结构中并行性的发展

1.4.1 并行性的概念

- **并行性**：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**：两个或两个以上的事件在**同一时刻**发生
 - **并发性**：两个或两个以上的事件在**同一时间间隔**内发生
- **并行处理**：挖掘计算过程中的并行事件，并使**并行性达到较高的级别**

1.4 计算机体系结构中并行性的发展

1.4.1 并行性的概念

- 从处理数据的角度来看，并行性等级从低到高可分为：

- 字串位串：每次只对一个字的一位进行处理。最基本的串行处理方式，**不存在并行性**
- 字串位并：同时对一个字的全部位进行处理，不同字之间是串行的。**开始出现并行性**
- 字并位串：同时对许多字的同一位（称为位片）进行处理。**具有较高的并行性**
- 全并行：同时对许多字的全部位或部分位进行处理。**最高一级的并行**

1.4 计算机体系结构中并行性的发展

1.4.1 并行性的概念

●从**执行程序的角度**来看，并行性等级从低到高可分为：

- **指令内部并行**：单条指令中各微操作之间的并行。
- **指令级并行**：并行执行两条或两条以上的指令。
- **线程级并行**：并行执行两个或两个以上的线程。通常是以一个进程内派生的多个线程为调度单位。
- **任务级或过程级并行**：并行执行两个或两个以上的过程或任务（程序段）。以子程序或进程为调度单元。
- **作业或程序级并行**：并行执行两个或两个以上的作业或程序。

1.4 计算机体系结构中并行性的发展

1.4.2 提高并行性的技术途径（三种）

(1) 时间重叠：引入时间因素

让多个处理过程在时间上相互错开，轮流重叠地使用同一套硬件设备的各个部分，以加快硬件周转而加速。

(2) 资源重复：引入空间因素，以数量取胜

通过重复设置硬件资源，大幅提高计算机系统的性能

(3) 资源共享：一种软件方法

使多个任务按一定时间顺序轮流使用同一套硬件设备

1.4 计算机体系结构中并行性的发展

1.4.3 单机系统中并行性的发展

(1) 时间重叠原理：在高性能单处理机中起主导作用

➤ **实现时间重叠的基础：部件功能专用化**

- 把一件工作按功能分割为若干相互联系的部分；
- 把每一部分指定给专门的部件完成；
- 按**时间重叠原理**把各部分的执行过程在时间上重叠起来，使所有部件依次分工完成一组同样的工作。



1.4 计算机体系结构中并行性的发展

1.4.3 单机系统中并行性的发展

例如：对于解释指令的5个过程，就分别需要5个专用的部件：

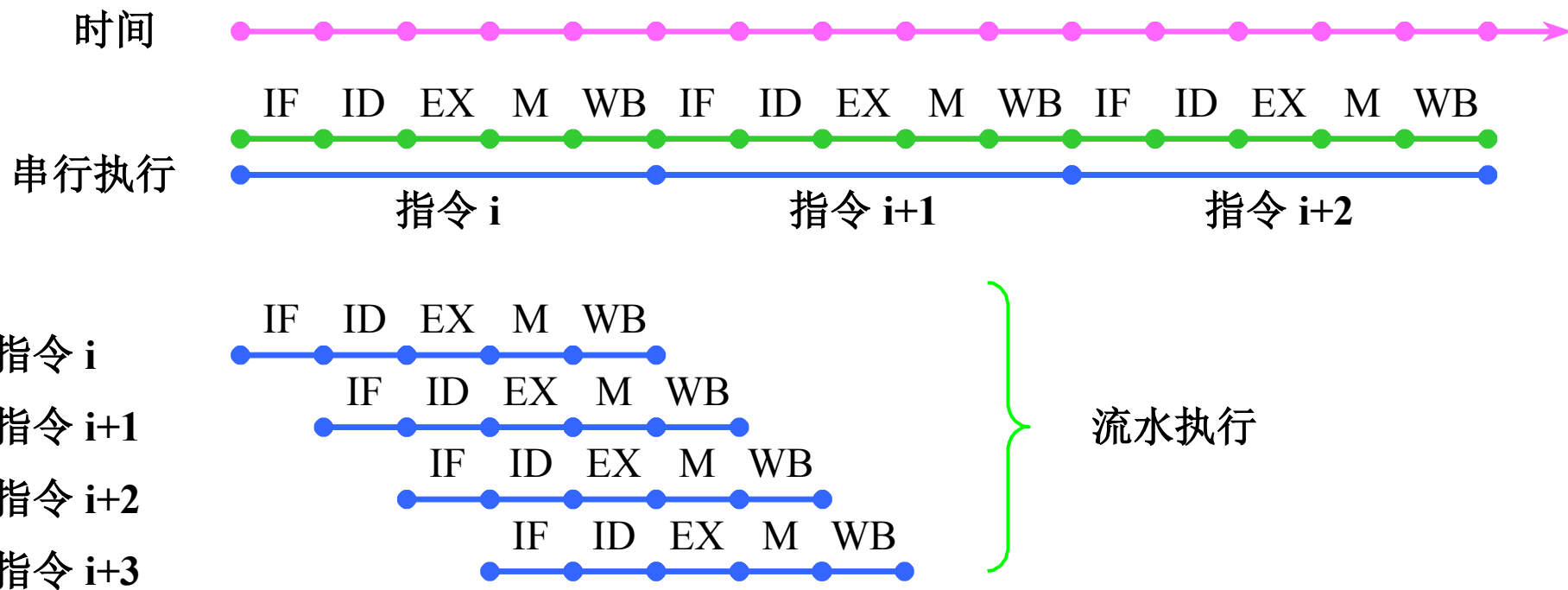
取指令部件(IF)

指令译码部件(ID)

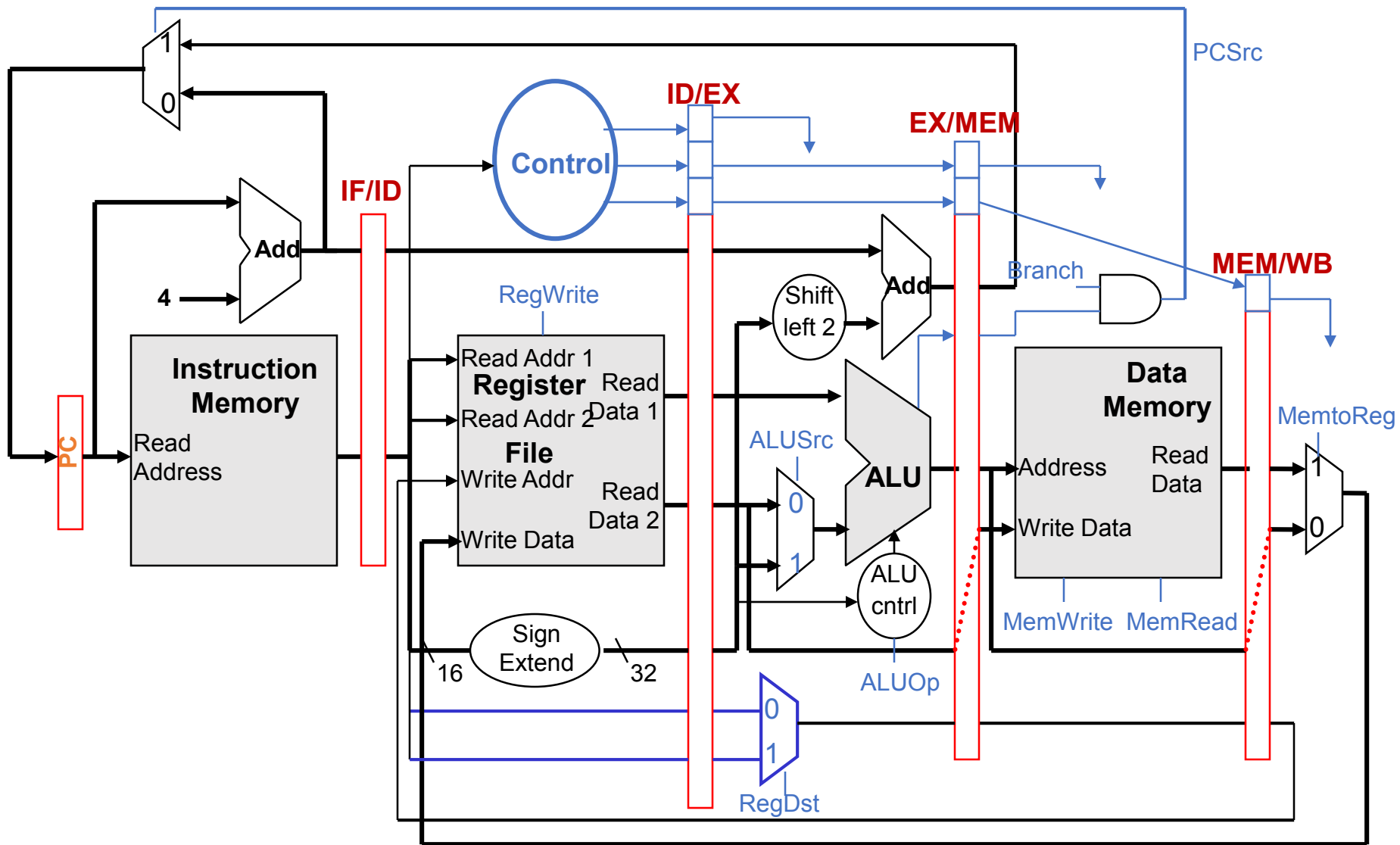
指令执行部件(EX)

访问存储器部件(M)

写结果部件(WB)



MIPS 流水数据和控制通路



1.4 计算机体系结构中并行性的发展

1.4.3 单机系统中并行性的发展

(2) 资源重复原理：在单处理机中运用也已经十分普遍。

- 多体存储器

- 多操作部件

- 通用部件被分解成若干个专用部件，如加法部件、乘法部件、除法部件、逻辑运算部件等，而且同一种部件也可以重复设置多个
- 只要指令所需的操作部件空闲，就可以开始执行这条指令（如果操作数已准备好）
- 实现了**指令级并行**

1.4 计算机体系结构中并行性的发展

1.4.3 单机系统中并行性的发展

(3) **资源共享**的概念实质上是用单处理机模拟多处理机的功能，形成所谓**虚拟机**的概念。

- 分时系统

阵列处理机（并行处理机）：设置许多相同的处理单元，在同一个控制器的指挥下，按照同一条指令的要求，对向量或数组的各元素同时进行同一操作

1.4 计算机体系结构中并行性的发展

1.4.4 多机系统中并行性的发展

- 多机系统遵循**时间重叠、资源重复、资源共享原理**，发展为3种不同的多处理机：
异构型多处理机、同构型多处理机、分布式系统
- **耦合度**：反映多机系统中各机器之间物理连接的紧密程度和交互作用能力的强弱
 - **紧密耦合系统（直接耦合系统）**：在这种系统中，计算机之间的物理连接的频带较高，一般通过**总线或高速开关**互连，可以共享主存。

1.4 计算机体系结构中并行性的发展

1.4.4 多机系统中并行性的发展

- **松散耦合系统（间接耦合系统）**：一般通过**通道或通信线路**实现计算机之间的互连，可以共享外存设备（磁盘、磁带等）。机器之间的相互作用是在文件或数据集一级上进行的。**表现为两种形式：**
- 多台计算机和共享的外存设备连接，不同机器之间实现功能上的分工（功能专用化），机器处理的结果以文件或数据集的形式送到共享外存设备，供其他机器继续处理。
 - 计算机网络，通过通信线路连接，实现更大范围的资源共享。

1.4 计算机体系结构中并行性的发展

1.4.4 多机系统中并行性的发展

- 功能专用化（实现时间重叠）

- 专用外围处理机：如输入/输出功能的分离。
- 专用处理机：如数组运算、高级语言翻译、数据库管理等，分离出来。
- 异构型多处理机系统：由多个不同类型、至少担负不同功能的处理机组成，它们按照作业要求的顺序，利用时间重叠原理，依次对它们的多个任务进行加工，各自完成规定的功能动作。

1.4 计算机体系结构中并行性的发展

1.4.4 多机系统中并行性的发展

- 机间互连

- 容错系统：**保证系统的可靠性**
- 可重构系统：对计算机之间互连网络的性能提出了更高的要求。高带宽、低延迟、低开销的机间互连网络是高效实现程序或任务一级并行处理的前提条件。
- 同构型多处理机系统：**各处理机具有相同的功能**
由多个同类型或至少担负同等功能的处理机组成，它们同时处理同一作业中能并行执行的多个任务。

第1章 计算机体系结构的基本概念

本章重点：

- 1.计算机系统的层次结构
 - 2.计算机体系结构的定义及研究内容
 - 3.计算机系统的评价方法
 - 4.冯·诺依曼结构及其发展
 - 5.透明性、系列机、兼容性等概念
 - 6.了解计算机系统的分类方法
- **课本作业：** 1.6 （CPU性能CPI）
1.8 （Amdahl定律）