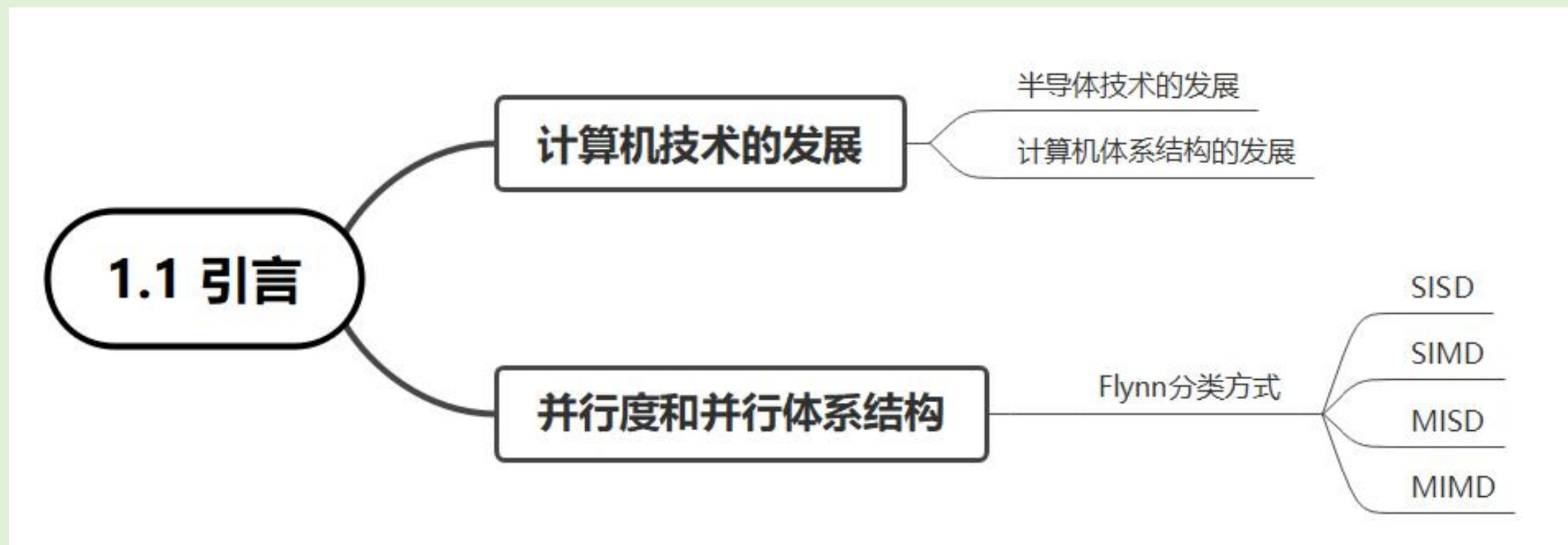


# 计算机系统结构

## 第一章 概述

# 1.1 引言



社会的需求对科技进步的作用要超过**10**所大学。

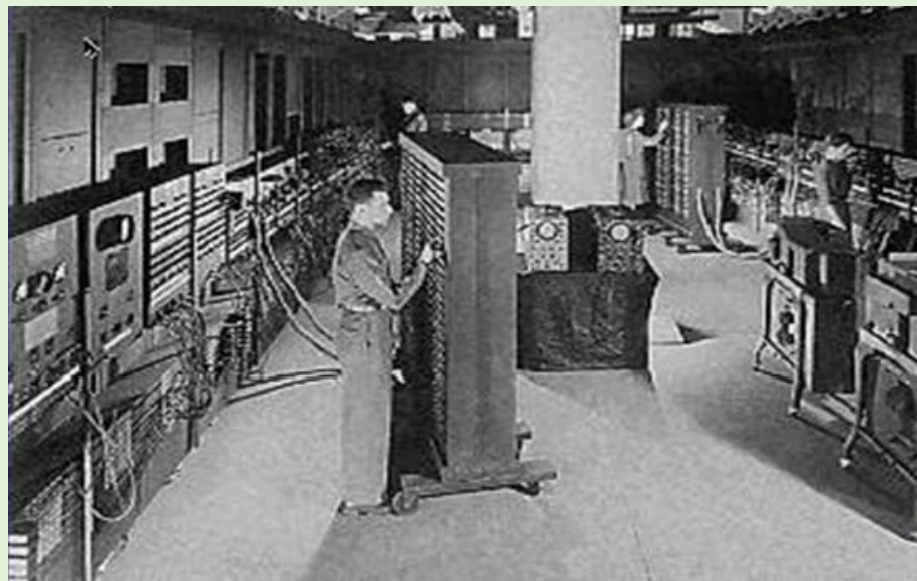
---- 恩格斯

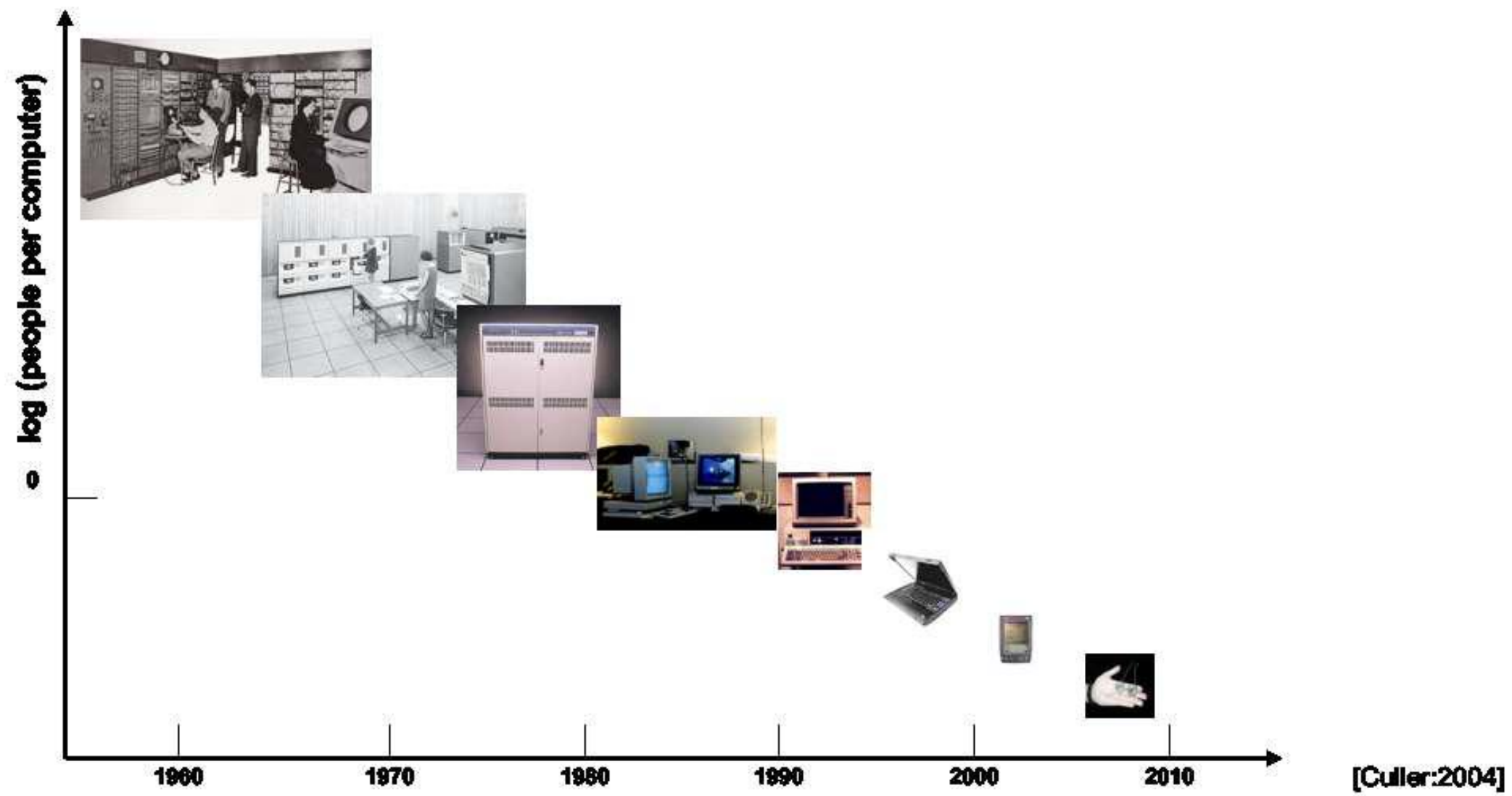
## 第一台通用电子计算机诞生于1946年

世界上第一台现代电子计算机“埃尼阿克”（LJC32），诞生于1946年2月14日的美国宾夕法尼亚大学，并于次日正式对外公布。ENIAC长30.48米，宽1米，占地面积约170平方米，30个操作台，约相当于10间普通房间的大小，重达30吨，耗电量150千瓦，造价48万美元。它包含了17,468真空管7,200水晶二极管，1,500 中转，70,000电阻器，10,000电容器，1500继电器，6000多个开关，每秒执行5000次加法或400次乘法，是继电器计算机的1000倍、手工计算的20万倍。

ENIAC于1955年退役。

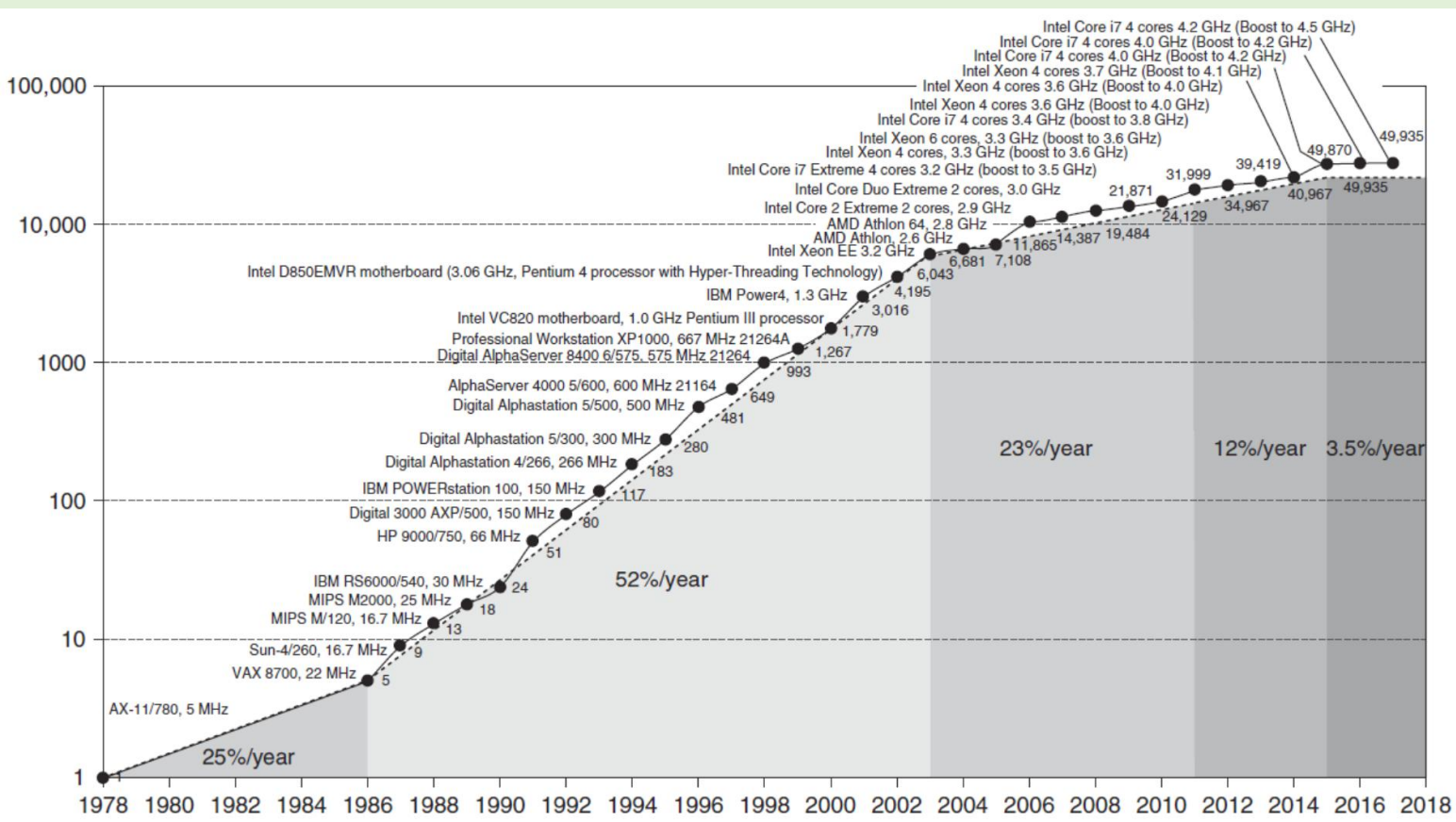
*Pentium 4 2.0 CPU，每秒约80亿次二进制加法*





## 1.1.1 计算机技术的发展

- 计算机性能的发展源于如下两个方面
  - 半导体技术的发展
  - 计算机体系结构的发展



I. 计算机体系结构和组织方式的发展一起促成了计算及性能以超过50%的年增长率持续增长17年（1986-2003）。

这一飞速发展带来四重效果：

- （1）增强可供计算机用户使用的功能；
- （2）导致新型计算机的出现；
- （3）半导体制造业的持续发展使得基于微处理器的计算机在整个计算机设计领域中占主导地位；
- （4）对软件开发的影响。

处理器性能的增长主要是半导体技术驱动。

虽然半导体技术发展迅速，但是这一阶段的飞速发展更归功于体系结构和组织思想的发展。

功耗和指令级的并行限制了单处理机的性能。2007年开始，单芯片上核数每年都在增长。

性能增长在降低，摩尔定律受限，并行化受限（Amdahl定律）

II. 硬件复兴结束（2004-）。

原因：功耗，ILP开发受限。

发展方向：数据级并行（DLP），线程级并行（TLP）。

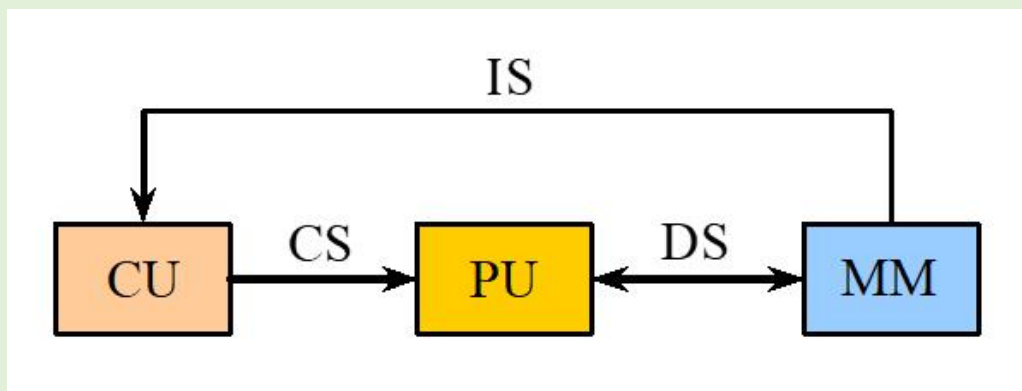
## 1.1.2 并行度和并行体系结构

20世纪60年代，Michael Flynn在研究并行计算工作量时，提出一种简单的分类方式，至今我们仍在使用：

- 单指令流、单数据流，Single Instruction Stream, Single Data Stream, SISD。
- 单指令流、多数据流，Single Instruction Stream, Multiple Data Stream, SIMD。
- 多指令流、单数据流，Multiple Instruction Stream, Single Data Stream, MISD。
- 多指令流、多数据流，Multiple Instruction Stream, Multiple Data Stream, MIMD。

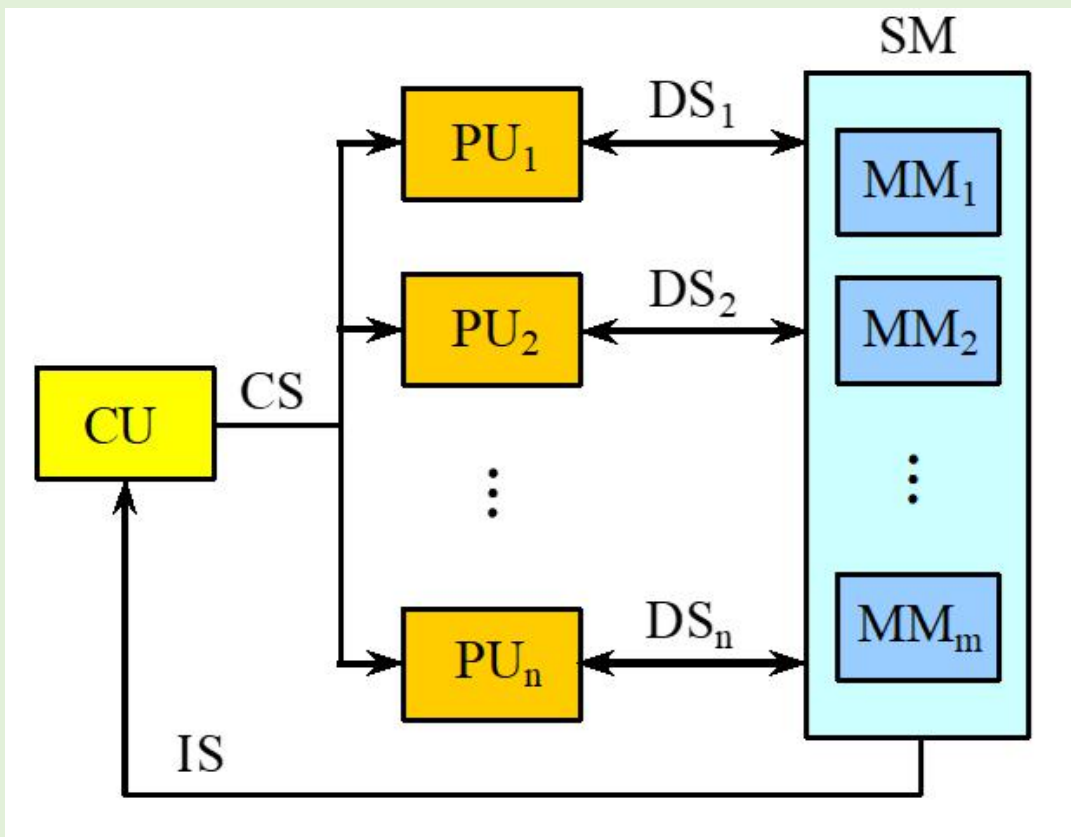


# 单指令流单数据流，SISD



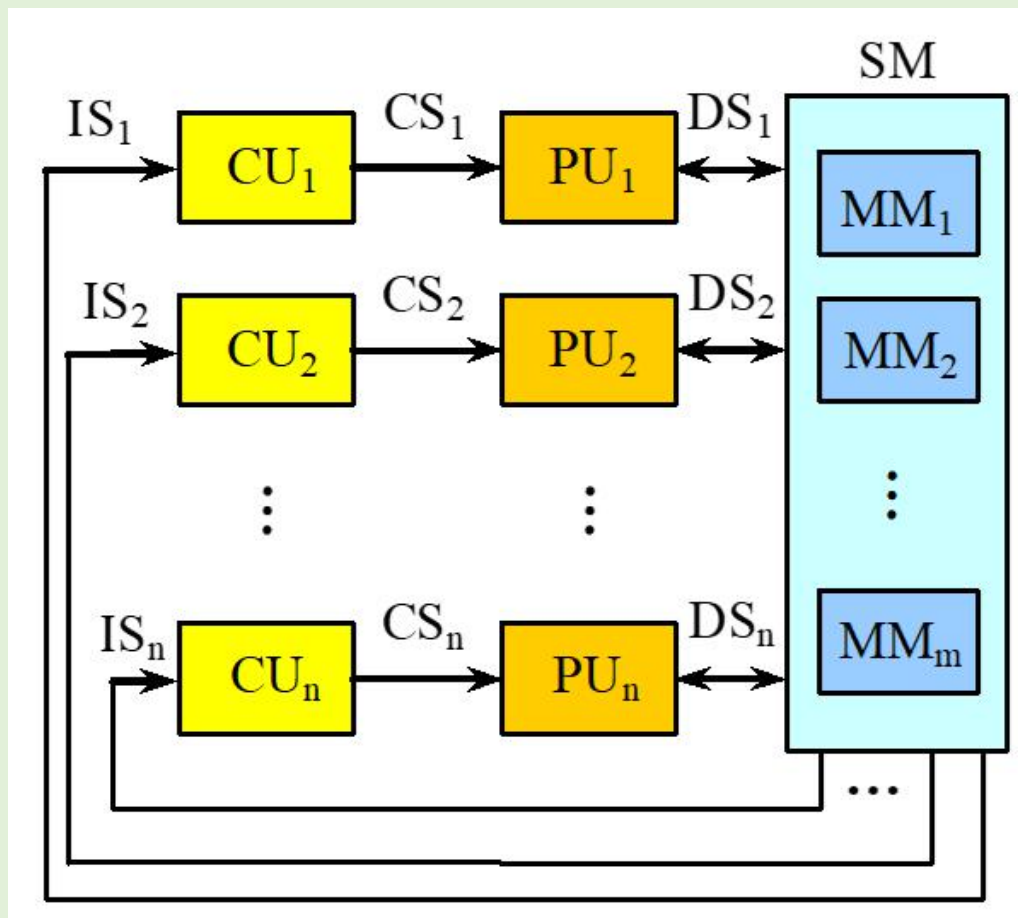
- 这个类别是单处理器；
- 可以使用指令级并行，如流水线、动态调度、超标量。

# 单指令流多数据流，SIMD



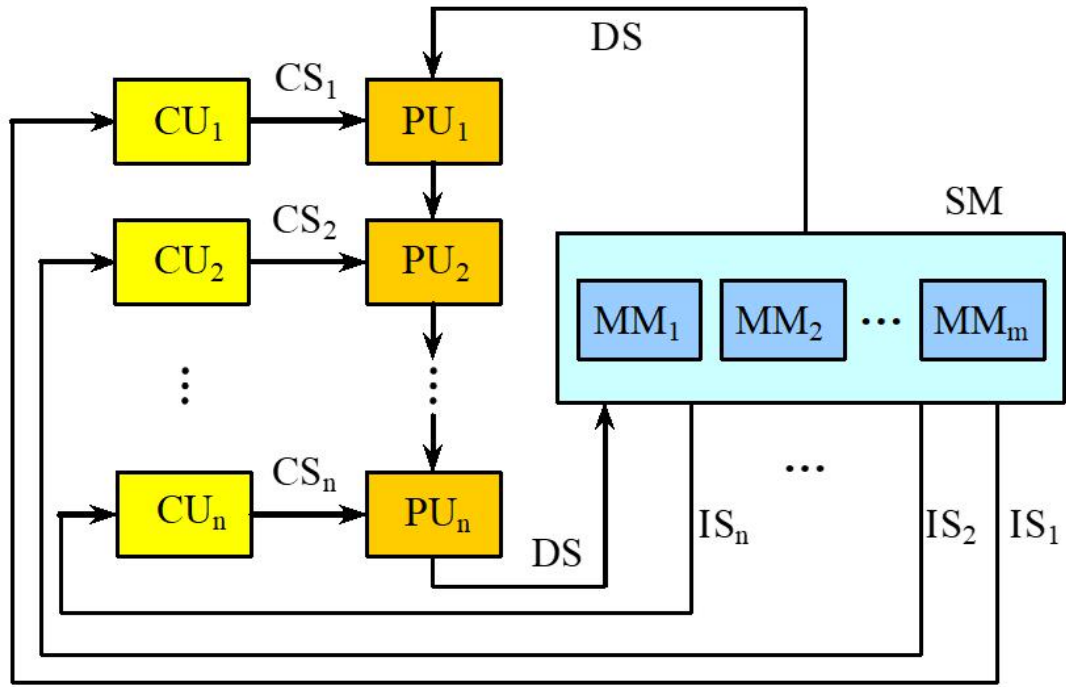
- 同一指令操纵多条数据流，由不同处理器，对多个数据项进行并行数据处理；
- 每个数据处理器可有自己的数据存储器，但是只有一个指令存储器和控制器；
- 向量体系结构，GPU

# 多指令流多数据流，MIMD



- 每个处理器都自己取指令，并对自己的数据进行处理；
- 任务级并行；
- 紧密耦合的MIMD结构—线程级并行；
- 松散耦合MIMD结构—集群。

# 多指令流单数据流, MISD



- 目前为止，没有此种类型的商用机；
- 脉动阵列。

# 1.1. 3. 关于本课程

## 计算机组成原理

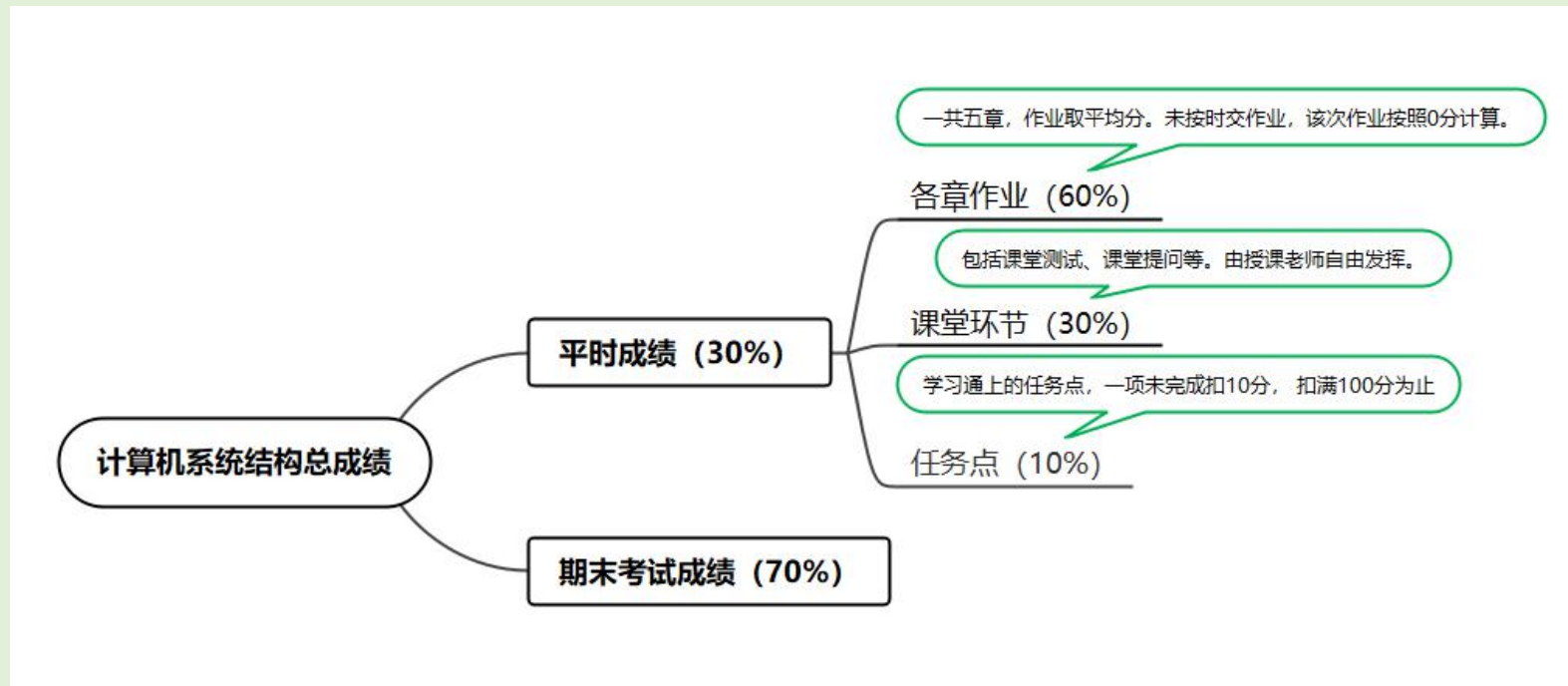
## 计算机系统结构 (32课时)

课时安排根据具体情况  
可能略有调整

该小节暂时不作为课堂  
讲授内容



# 成绩构成



本班课堂环节得分方式：

- (1) 课堂活动积分累计；
- (2) 随机点名提问，答了，无论对错+2积分；人到了，未答上，加0积分；未见人，扣1积分，扣完为止；
- (3) 最后课堂成绩按照积分分数段给出。例如 满积分为20分，那么15-20分折合成100分，等。

# 1.2 计算机体系结构

- 狭义的定义（Amdahl的定义）
  - 指令集的设计；
  - 需要考虑：寄存器、内存地址、访存方式、指令集的设计；
  - 经典定义：机器语言程序员所见到的计算机的属性，即概念性结构和功能特性。
- 广义的定义（《计算机体系结构-量化研究方法》中的定义）
  - 包括计算机设计所有三个方面
    - 指令集结构
    - 组成
    - 硬件

- 指令集结构（ISA）

- 分类

- 现在几乎所有的ISA都划分为通用寄存器体系结构中；
    - 两种主流版本：寄存器-存储器，如80X86； 载入-存储，如ARM和MIPS。

- 存储器寻址

- 字节寻址
  - 访问对齐

- 寻址模式：立即寻址、寄存器寻址、偏移寻址等

- 操作数类型和大小

- 指令类型：数据传输、算逻运算、控制指令、浮点指令 等；

- 指令格式：固定、可变



- 组成和硬件

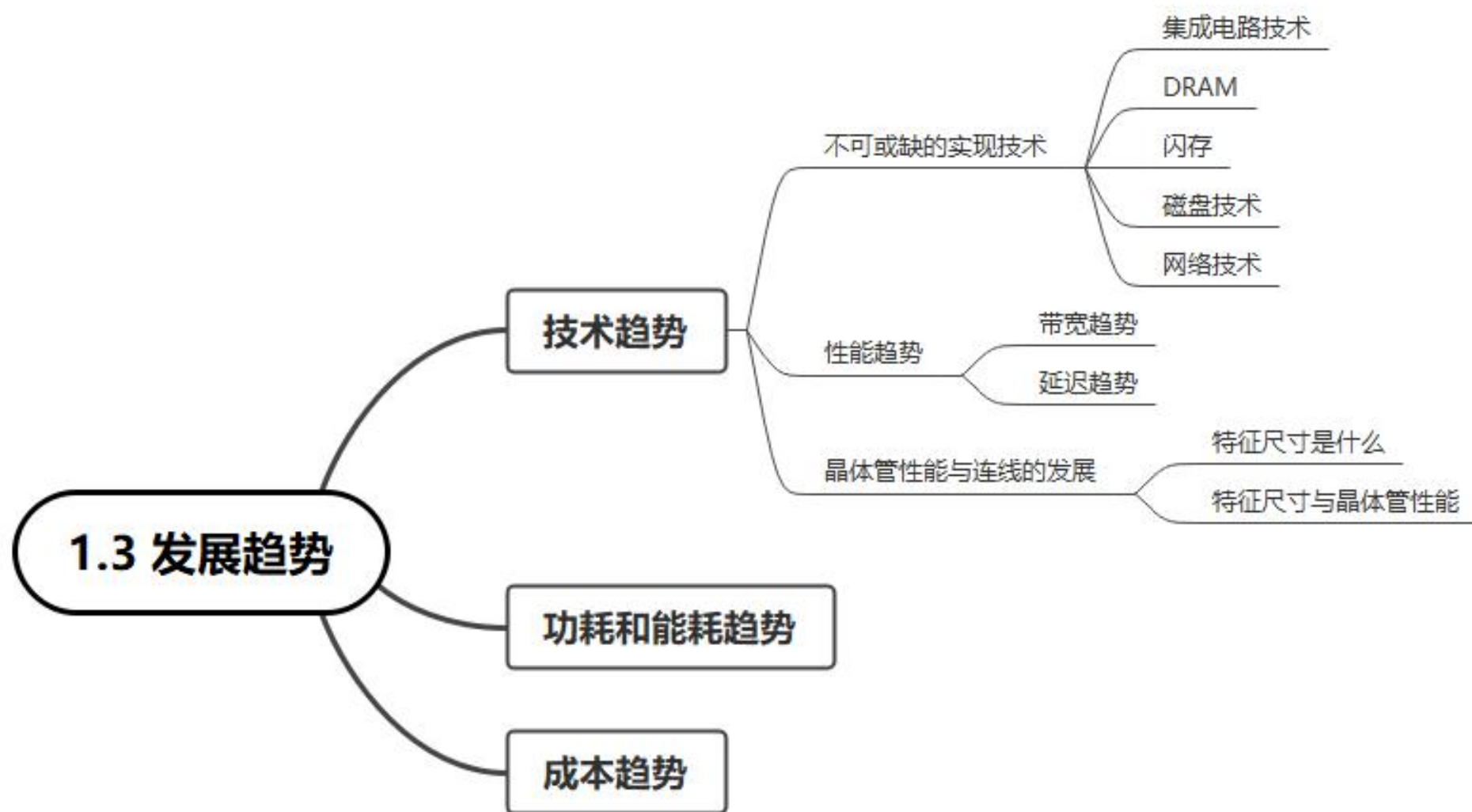
- **组成**是计算机的逻辑实现，包括存储系统、存储器连接、CPU内部设计。
- **硬件**是计算机的具体实现，包括详尽的逻辑设计和封装技术。

真正的计算机体系结构：设计满足目标和功能需求的组织和硬件。

--- 《计算机体系结构-量化研究方法》

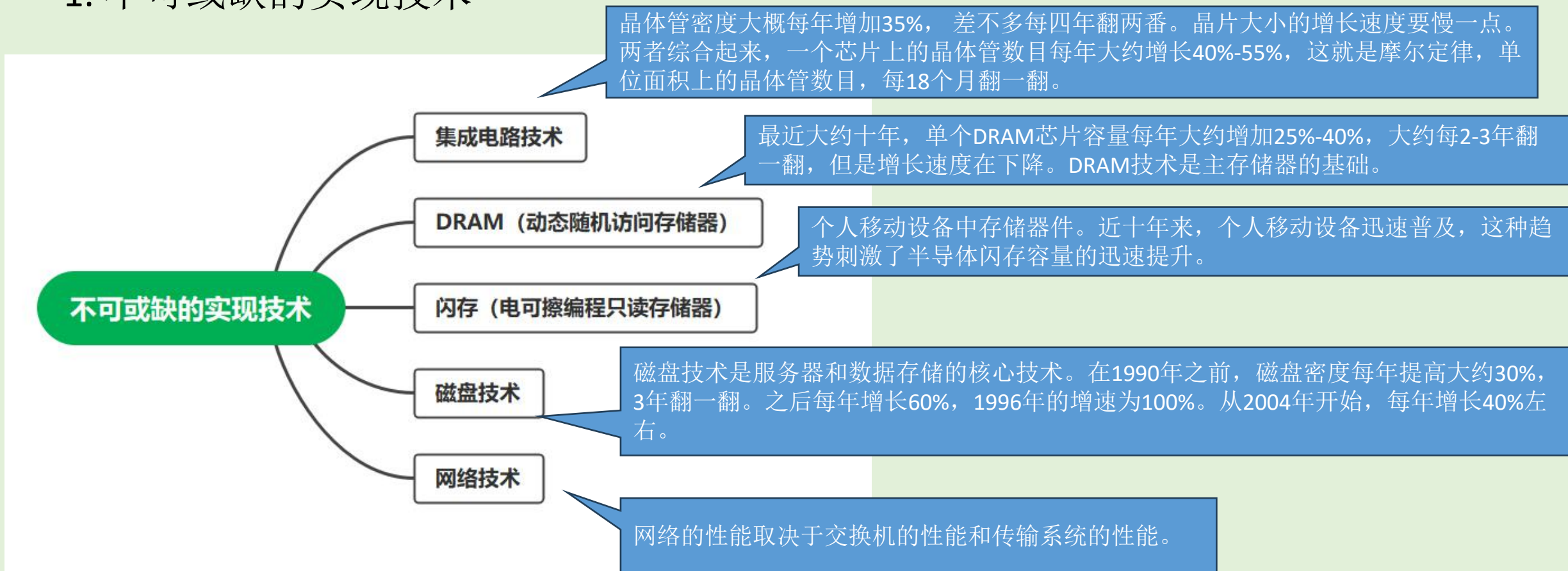
关于计算机体系结构的相关内容，请阅读学习通上1.2中的拓展阅读资料。

## 1.3 发展趋势



# 1.3.1 技术趋势

## 1. 不可或缺的实现技术



## 2. 性能趋势：带宽和延迟

- 带宽 或 吞吐量
  - 单位时间的工作量
  - 处理器的带宽增长 32,000-40,000 倍
  - 内存和磁盘的带宽增长 300-1200 倍
- 延迟 或 响应时间
  - 一个事件从开始到完成所经历的时间
  - 处理器响应时间改进 50-90 倍
  - 内存和磁盘的响应时间改进 6-8 倍

显然，在这些技术的发展过程中，**带宽的改进要明显优于延迟**。一个简单的经验法则是：带宽的增长速度至少是延迟改进速度的平方。

表 1-7 微处理器、存储器、网络 and 磁盘在过去 20~40 年里的性能里程碑

微处理器	16位地址/总线、微编码	32 位 地址/总线、微编码	5级流水线、片上I&D缓存、FPU	2路超标量、64位总线	乱序3路超标量	乱序超流水线、片上L2缓存	多核 OOO 4路片上L3缓存、Turbo
产品	Intel 80286	Intel 80386	Intel 80486	Intel Pentium	Intel Pentium Pro	Intel Pentium 4	Intel Core i7
年份	1982	1985	1989	1993	1997	2001	2010
晶片大小(平方毫米)	47	43	81	90	308	217	240
晶体管数	134 000	275 000	1 200 000	3 100 000	5 500 000	42 000 000	1 170 000 000
处理器数/芯片	1	1	1	1	1	1	4
管脚	68	132	168	273	387	423	1366
延迟(时钟周期)	6	5	5	5	10	22	14
总线宽度(位)	16	32	32	64	64	64	196
时钟频率(MHz)	12.5	16	25	66	200	1500	3333
带宽(MIPS)	2	6	25	132	600	4500	50 000
延迟(ns)	320	313	200	76	50	15	4

存储器、磁盘和网络数据1.3必读资料中，请自己阅读。

### 3. 晶体管（Transistors）性能与连线（wire）的发展

- 集成电路的制造工艺是以**特征尺寸**来衡量的

- 特征尺寸 就是一个晶体管或者一条连线在x方向或者y方向的最小尺寸。
- 特征尺寸：1971为 10 微米， 2011年0.032微米（32纳米技术）， 2017年 0.016微米（16纳米技术）。
- 特征尺寸下降时，晶体管性能以线性提升，晶体管的密度将以平方曲线上升。

借助这种增长，微处理器从4位迅速发展到了64位，并从单核发展到多核。
- 随着特征尺寸缩小，连线会变短，但是单位长度的电阻和电容都会变差。和晶体管性能相比，连线延迟方面的改进少的可怜。



## 1.3.2 功率和能耗趋势

对于几乎所有类型的计算机来说，功率都是计算机设计人员面临的挑战。  
第一，必须将功率引入芯片，并进行分配；  
第二，功率以热的形式耗散，必须消除。

### 1. 功率和能耗：系统观点

- 处理器需要的最大功率是多少？

如果处理器需要的最大功率超过电源系统能提供的功率，就会导致电压下降，而电压下降会导致期间无法正常工作；

- 持续功耗 -- 热设计功耗（Thermal Design Power, TDP）

决定了冷却需求，当温度接近节点温度上限时，电路降低时钟频率，从而减小功率；如果不成功，启动热过载启动装置，以降低芯片的功率。

- 能耗和能耗效率。

所谓的功率，就是单位时间的能耗。用能耗更适合作为对比处理器的参数。因为能耗这个参数同时包含了执行任务的时间。也就是，如果对比两个处理器对于某一给定任务，那个更高效，那么应该对比的是执行该项任务的能耗，而非功率。

## 2. 微处理器内部的能耗和功率

- 主要能耗源：开关晶体管（动态能耗）
- 时钟频率和电源电压保持不变的情况下，提高能耗效率。
  - 关闭非活动模块，以节省能耗和动态功率；
  - 动态电压-频率调整。微处理器可以提供不同工作频率。
  - 针对典型情景的设计，PMD、笔记本等的省电模式；
  - 超频。Intel在2008年开始提供Turbo模式，在这种模式下，芯片可以判定在少数几个核心上以较高的时钟频率短时间运行。

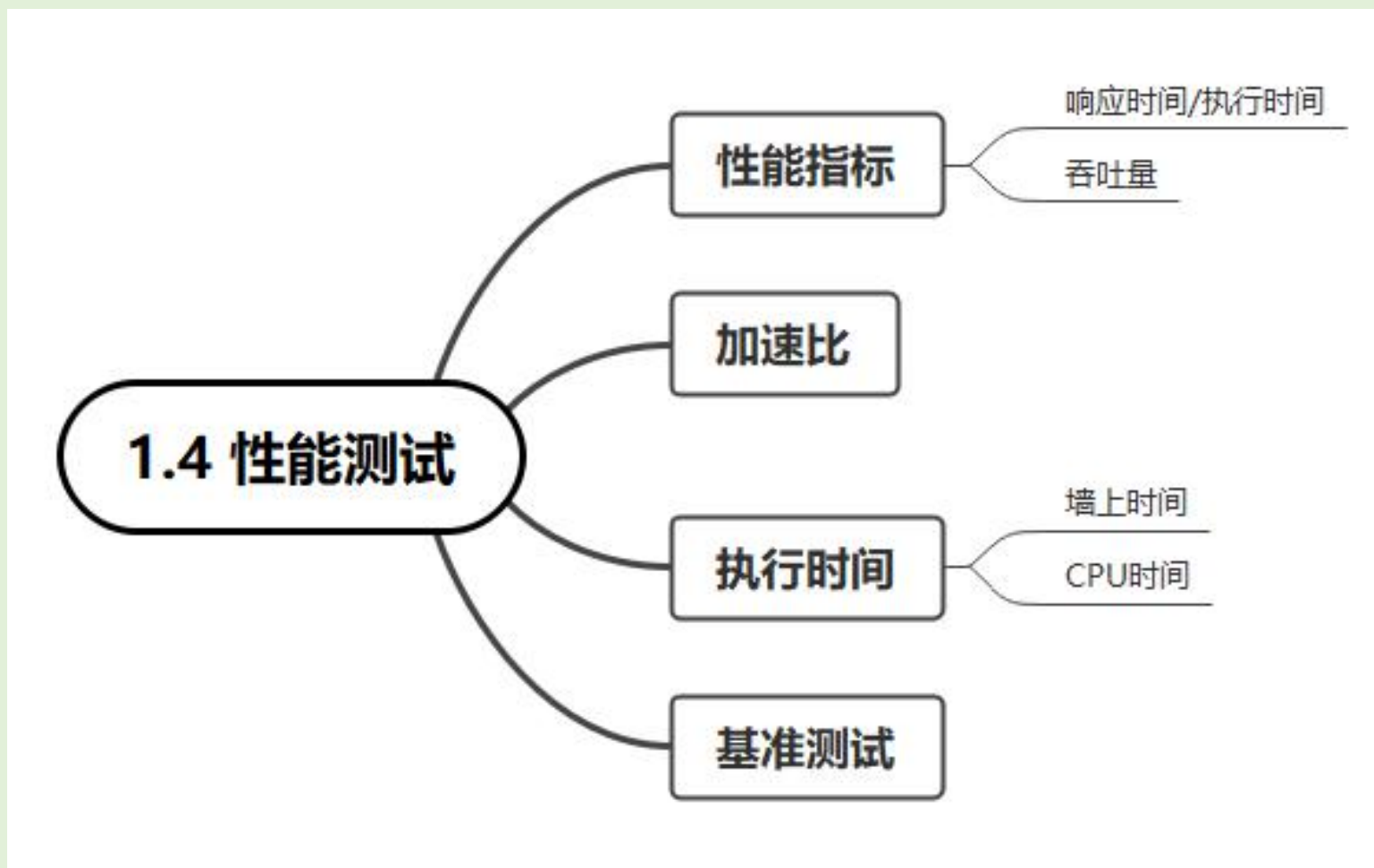


## 1.3.3 成本趋势

学习通1.3的必读资料，请自行阅读

- 时间、产量和大众化影响
- 集成电路成本
- 成本与价格
- 制造成本与运行成本

## 1.4 性能测试



## 1. 如何评价性能？ -- 性能指标

- 响应时间/执行时间 -- 用户关心的
- 吞吐量 -- 仓库级计算机的操作人员关心的

## 2. 加速比

- 机器X比机器Y快， 是什么意思？ -- 执行时间是稳定、可靠的性能度量

## 3. 执行时间

- **单个用户** - 访问内存所需要的时间，访问磁盘所需要的时间，输入输出操作所需要的时间，操作系统开销，CPU执行时间。。。
- **多道程序** - 操作系统调度。。。太难了。  
----- 墙上时间： 包括所有的系统开销
- CPU时间： only computation time

## 4. 基准测试

- 程序内核： 实际应用程序中的短小、关键部分；
- 玩具程序： 为了完成编程入门作业而编写的小程序，通常不超过100行，如快排程序；
- 合成基准测试程序： 为了匹配实际应用程序的特征和行为编写的虚拟程序；
- 基准测试套件（Benchmark suites）： 衡量处理器处理各种应用程序的性能。优势在于任何一个基准测试的弱点都会因为其他基准测试的存在而变小。最常用的是SPEC，所有SPEC基准测试套件及其报告都可以在[www.spec.org](http://www.spec.org)中找到。

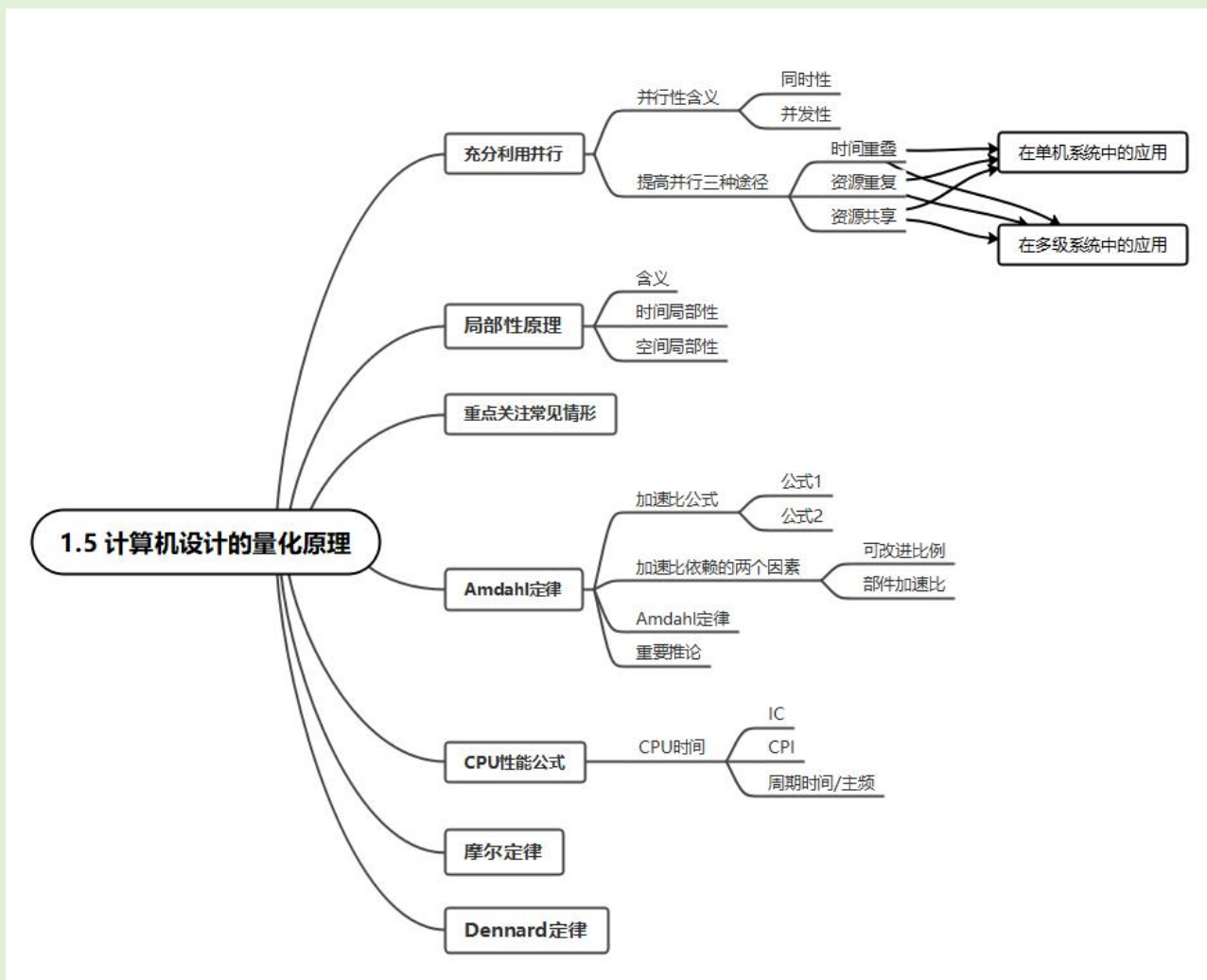
# 关于基准程序测试

1. 计算机性能的一种常用的评估方式是使用基准评测程序，也就是**benchmark**。基准评测程序是专门设计用来评价计算机性能的一组程序，从而反映机器在实际工作负载时的时间。在不同的机器上运行同一套基准程序，通过比较运行时间来评价机器性能。

2. **SPEC (System Performance Evaluation Cooperative)** 是由许多计算机销售商联合出资赞助并支持的组织，目的是为现代计算机系统建立基准评测程序集，该组织成立于1988年。1989年，SPEC建立了重点面向处理器性能的基准程序集，即SPEC89。历经五代发展，应用时间最长和最广泛的是SPEC CPU 2006。它是由SPECint和SPECfp组成，前者包括了12个整数基准评测程序，后者包括19个基准评测程序。随着SPEC CPU 2017的发布，CPU 2006已经于2018年1月正式退休。

描述	IC (10 <sup>9</sup> )	CPI	周期时间 (10 <sup>-9</sup> )	执行 时间 (秒)	参考 时间 (秒)	SPEC 分值
字符串处理程序	2252	0.60	0.376	508	9770	19.2
块排序压缩	2390	0.70	0.376	629	9650	15.4
GNU C编译器	794	1.20	0.376	358	8050	22.5
组合优化	221	2.66	0.376	221	9120	41.2
围棋游戏	1274	1.10	0.376	527	10490	19.9
基因序列搜索	2616	0.60	0.376	590	9330	15.8
国际象棋游戏	1948	0.80	0.376	586	12100	20.7
量子计算机模拟	659	0.44	0.376	109	20720	190.0
视频压缩	3793	0.50	0.376	713	22130	31.0
离散事件模拟库	367	2.10	0.376	290	6250	21.5
游戏/寻径	1250	1.00	0.376	470	7020	14.9
XML解析	1045	0.70	0.376	275	6900	25.1

# 1.5 计算机设计的量化原理



## 1.5.1 充分利用并行

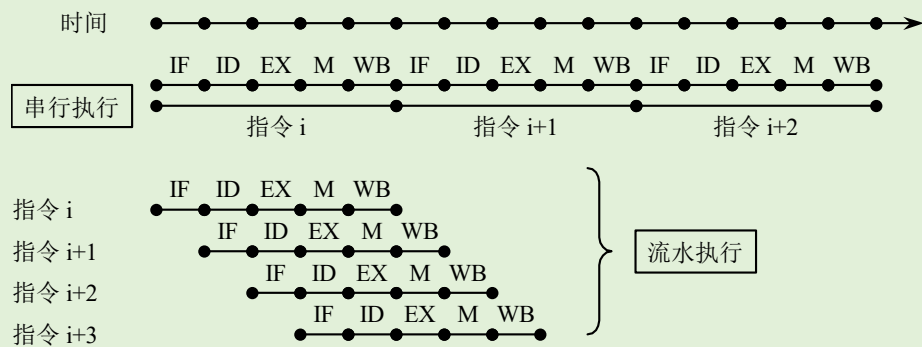
- 并行性的含义：

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。即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。
- 提高并行性的三种途径：
  - **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，以加快硬件周转而赢得速度。
  - **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，大幅度地提高计算机系统的性能。
  - **资源共享：**这是一种软件方法，它使多个任务按一定时间顺序轮流使用同一套硬件设备。如，多道程序。

# 时间重叠

- 单处理机中 -- **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。



- 多处理机中 – **处理机专用化**
  - 专用外围处理机  
例如，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。



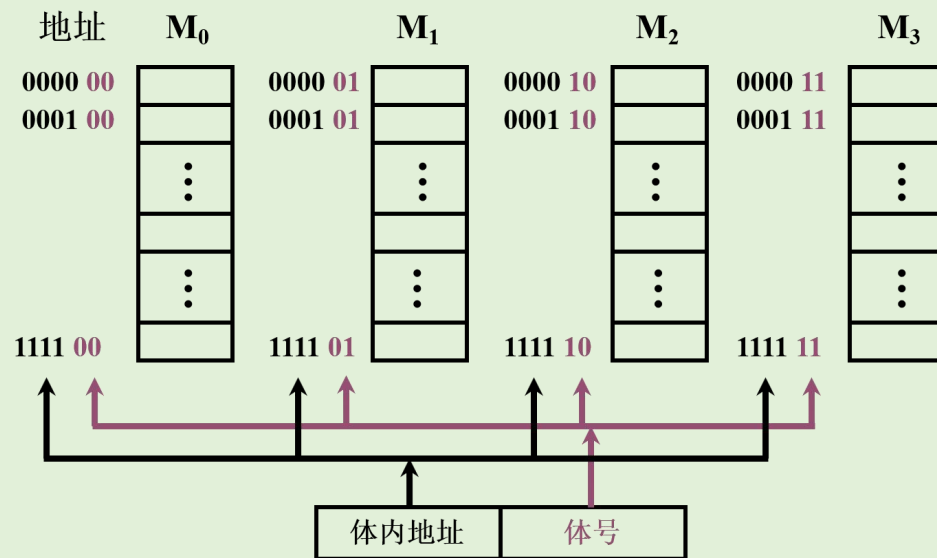
# 资源重复

- 单处理机中 -- 重复设置功能部件
  - 多存储体并行
  - 数据/指令独立存储体
  - 重复设置运算部件

- 多处理机中 – 重复设置处理机

- 容错系统
- 同构型多处理机系统

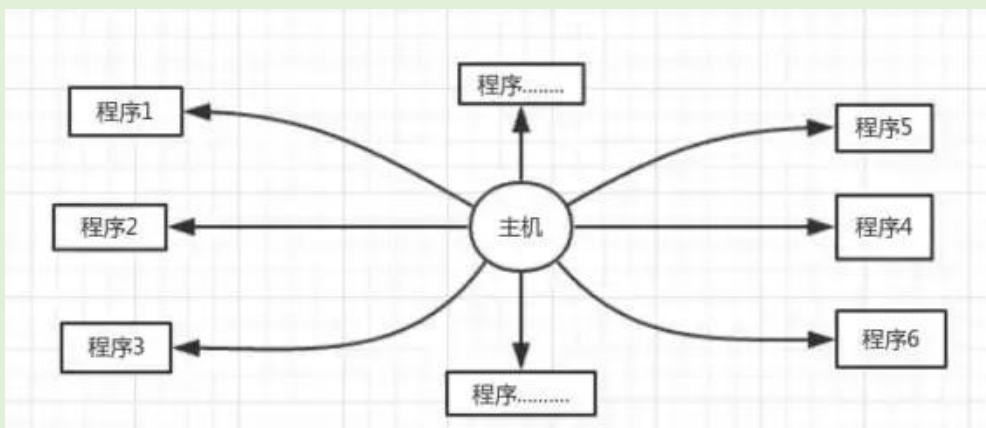
由多个同类型或至少担负同等功能的处理机组成，它们同时处理同一作业中能并行执行的多个任务。



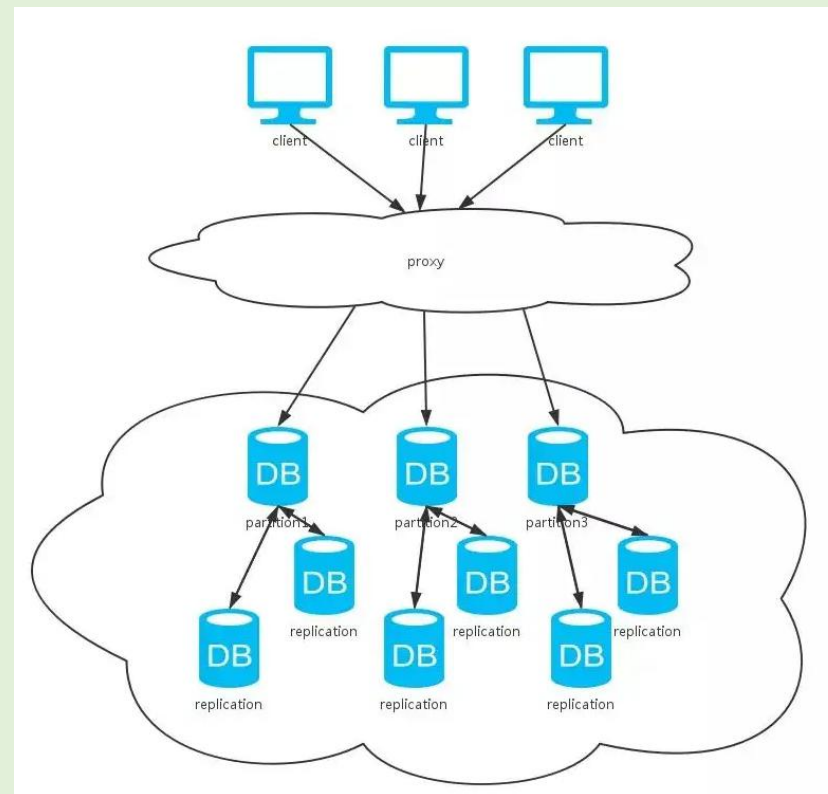
低位交叉多体并行

# 资源共享

- 单处理机中
  - 分时系统



- 多处理机中
  - 分布式系统



# 单机系统和多机系统中 并行性的发展

## 1.5.2 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。

## 1.5.3 重点关注常见情形

- 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。
- 例如：
  - 相对溢出，不溢出是经常性事件。针对不溢出的情况进行性能设计，无需花过多的时间和精力在解决溢出上。
  - 处理器的取值、译码比乘法器用的更加频繁，优先优化取指令和译码部件。

## 1.5.4 Amdahl定律

- 加速比 – 衡量系统的性能改进了多少

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

性能 = 1/执行时间

$$= \frac{\frac{1}{\text{总执行时间}_{\text{改进后}}}}{\frac{1}{\text{总执行时间}_{\text{改进前}}}}$$

$$= \frac{\text{总执行时间}_{\text{改进前}}}{\text{总执行时间}_{\text{改进后}}}$$

通常 大于 1

- 换个方式来看

$$\begin{aligned}
 \text{总执行时间}_{\text{改进后}} &= \text{不可改进部分的执行时间} + \text{可改进部分改进后执行时间} \\
 &= (1 - \text{可改进比例}) \times \text{总执行时间}_{\text{改进前}} + \frac{\text{可改进比例} \times \text{总执行时间}_{\text{改进前}}}{\text{部件加速比}} \\
 &= \text{总执行时间}_{\text{改进前}} \times \left[ (1 - \text{可改进比例}) + \frac{\text{可改进比例}}{\text{部件加速比}} \right]
 \end{aligned}$$


---

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

- **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。
- 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。一般情况下部件加速比是大于1的。
- 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。

考虑：可改进比例趋近于1。

考虑：部件加速比趋近于无穷大。



**Amdahl定律**表明：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。

**例** 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？

**例** 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：

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

采用此提高性能的方法后，能使整个系统的性能提高到原来的1.613倍。

- 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。
  - **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

$$1 / (1 - \text{可改进比例})$$

# 公式拓展

假设：

- $S_i$ 为第  $i$  种部件的加速比；
- $P_i$ 为第  $i$  种部件执行时间占总执行时间中的比例。

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

$$= \frac{1}{(1 - \sum P_i) + \sum \frac{P_i}{S_i}}$$

## 1.5.5 CPU性能公式

- 执行一个程序所需的CPU时间

$$\text{CPU时间} = \text{执行程序所需的时钟周期数} \times \text{时钟周期时间}$$

其中：

- 执行程序所需的时钟周期数 = 每条指令执行的平均时钟周期数  $\text{CPI}$   $\times$  所执行的指令条数  $\text{IC}$ ;
- 时钟周期时间是系统时钟频率的倒数。
- CPU时间公式也可以写为如下形式：

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

或：

$$\text{CPU时间} = \text{IC} \times \text{CPI} \times (1/\text{时钟频率})$$

请思考：如何降低CPU时间？

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

编译技术

指令集结构

计算机组成

硬件实现技术

CISC → RISC

IC↑

CPI↓

- CPI时间公式拓展

假设：计算机系统有n种指令，

$CPI_i$  : 第i种指令的处理时间（周期数）；

$IC_i$  : 在程序中第i种指令出现的次数；

则

$$\begin{aligned} \text{CPU时间} &= \text{执行程序所需的时钟周期数} \times \text{时钟周期时间} \\ &= \sum_{i=1}^n (CPI_i \times IC_i) \times \text{时钟周期时间} \end{aligned}$$

- CPI公式拓展

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

其中， $(IC_i/IC)$ 反映了第i种指令在程序中所占的比例。



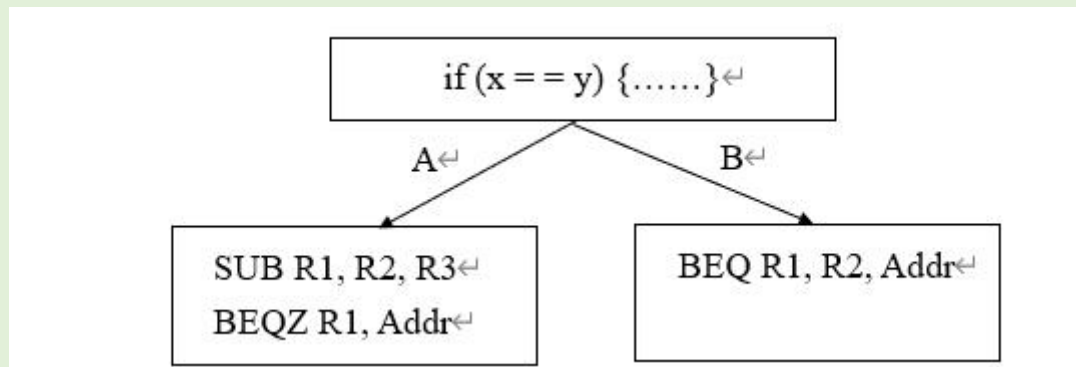
例 考虑条件分支指令的两种不同设计方法：

(1) CPU<sub>A</sub>：通过比较指令设置条件码，然后测试条件码进行分支。

(2) CPU<sub>B</sub>：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<sub>A</sub>，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<sub>A</sub>在分支时不需要比较，因此CPU<sub>B</sub>的时钟周期时间是CPU<sub>A</sub>的1.25倍。问：哪一个CPU更快？如果CPU<sub>B</sub>的时钟周期时间只是CPU<sub>A</sub>的1.1倍，哪一个CPU更快呢？

例如



# 分 析

CPU<sub>A</sub>:

- 需要两条指令完成分支，比较置条件码指令 + 分支指令。
- 假设完成某一功能F，使用CPU<sub>A</sub>，程序需100条指令；

程序指令构成：

指令类型	CPI <sub>i</sub>	指令数
比较置位	1	20
测试分支	2	20
其      他	1	60

CPU<sub>B</sub>:

- 需要一条指令完成比较和分支
- 假设完成同样功能F，使用CPU<sub>B</sub>，程序需80条指令；

程序指令构成：

指令类型	CPI <sub>i</sub>	指令数
比较并分支	2	20
其      他	1	60

解

我们不考虑所有系统问题，所以可用CPU性能公式。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<sub>A</sub>性能为

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

在CPU<sub>B</sub>中没有独立的比较指令，所以CPU<sub>B</sub>的程序量为CPU<sub>A</sub>的80%，分支指令的比例为

$$20\%/80\% = 25\%$$

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<sub>B</sub>不执行比较，故

$$IC_B = 0.8 \times IC_A$$

同时，根据假设，有

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

因此CPU<sub>B</sub>性能为

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

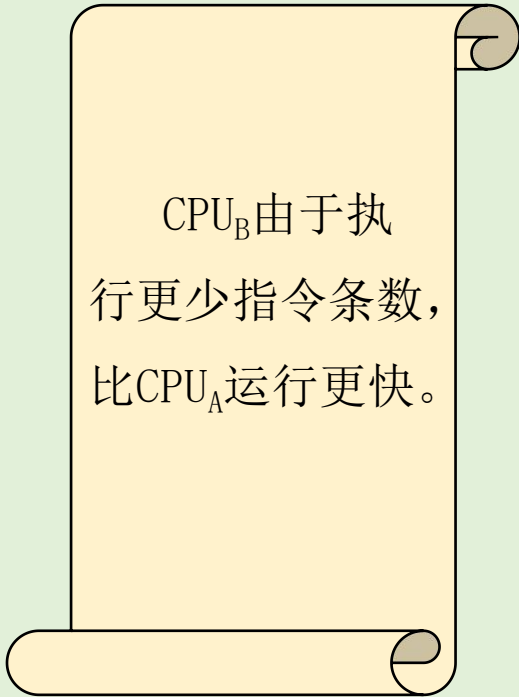
在这些假设之下，尽管CPU<sub>B</sub>执行指令条数较少，CPU<sub>A</sub>因为有着更短的时钟周期，所以比CPU<sub>B</sub>快。

如果CPU<sub>B</sub>的时钟周期时间仅仅是CPU<sub>A</sub>的1.1倍，则

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

CPU<sub>B</sub>的性能为

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



CPU<sub>B</sub>由于执行更少指令条数，比CPU<sub>A</sub>运行更快。

## 1.5.6 摩尔定律

摩尔定律是由英特尔(Intel)名誉董事长戈登·摩尔( Gordon moore)经过长期观察总结的经验。被称为计算机第一定律。摩尔定律是指单芯片上可容纳的晶体管数目，约每隔18个月便会增加一倍，性能也将提升一倍。

## 1.5.7 Dennard定律

晶体管面积的缩小使得其所消耗的电压以及电流会以差不多相同的比例缩小。也就是说，如果晶体管的大小减半，该晶体管的静态功耗将会降至四分之一（电压电流同时减半）。