计算机系统结构

上海大学计算机学院

教学安排和考核

- 学分：4，学时：50（课内学时20，研讨学时10，实验学时20）
- 上课时间：
  - **一 9-11 机房上机、研讨**

研讨课要求：每人准 备10分钟PPT

  - **三 5-6　计308**
  - 答疑时间：二3-4

教学安排和考核

- 学习资料下载：

  上海大学网络教学平台→资料

- 作业提交：

  上海大学网络教学平台→作业→网上填写提交

教学考核

- **总成绩 = 考试成绩60% + 平时成绩40%**
- **平时成绩** = 学习态度(出勤，作业)20% + 上机实验课成绩30%+研讨课成绩50%
- **考试成绩：线下闭卷考试**

联系

- 办公室：计算机大楼1011室
- E-mail: xuechen@shu.edu.cn

计算机系统结构 课程介绍

- 课程名称

  Computer Architecture

  计算机系统结构 计算机体系结构

  建筑物的设计或式样，通常指一个系统的外貌

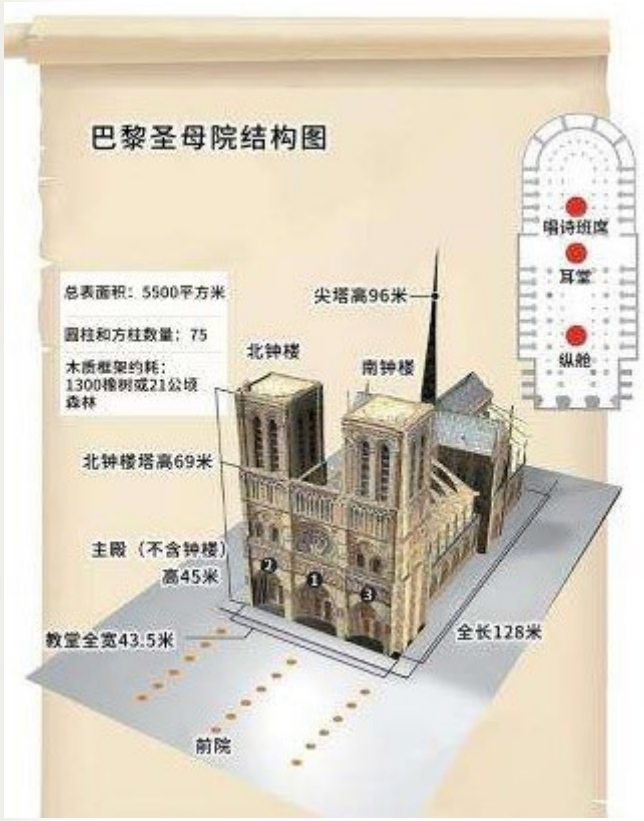
- 研究内容

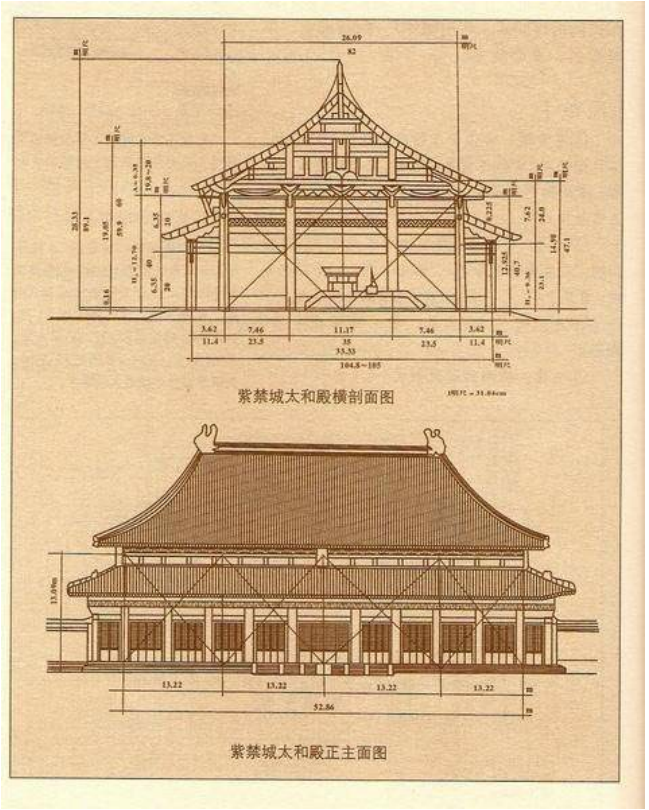
  从外部来研究计算机系统

  使用者所看到的物理计算机的抽象

  编写出能够在机器上正确运行的程序所必须了解到的计算机属性

计算机系统结构 课程介绍





- 学习目的
  - 建立计算机系统的完整概念
  - 学习计算机系统的分析方法和设计方法
  - 了解计算机系统的最新研究成果

- 与其它学科的交叉
  - 学科交叉：计算机组成、计算机操作系统、汇编语言、微计算机技术、计算机网络……
  - 新内容：超标量处理机、超流水线处理机、VLIW处理机、向量处理机、并行处理机、多处理机、互联网络……

教材

- 徐炜民、严允中，《计算机系统结构》（第3版），电子工业出版社，2003（国家教育部“普通高等教育十五国家级规划教材”）
- 郑纬民、汤志忠，《计算机系统结构》（第2版），清华大学出版社

参考书

- David A. Patterson and John L. Hennessy, Computer Architecture: A Quantitative Approach, 6 Ed., Morgan Kaufmann Publishers

  计算机系统结构：一种定量的方法(第五版)，清华大学出版社

- Kai Hwang, Advanced Computer Architecture: Parallelism, Scalability, Programmability

  高等计算机系统结构：并行性，可扩展性，可编程性，清华大学出版社，广西科学技术出版社

第一章 导论

1.1 计算机系统的基本概念

1.2 计算机系统的发展

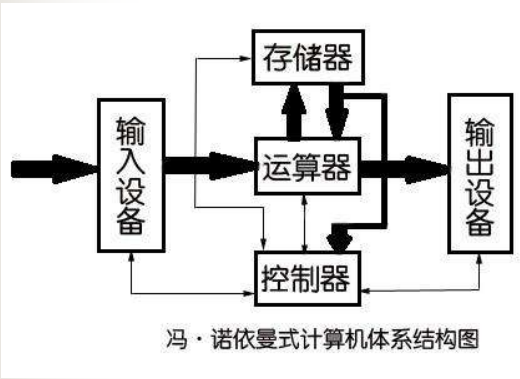
1.3 计算机系统的层次结构

1.4 计算机系统的设计方法

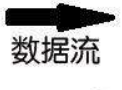
1.5 现代计算机系统的分类

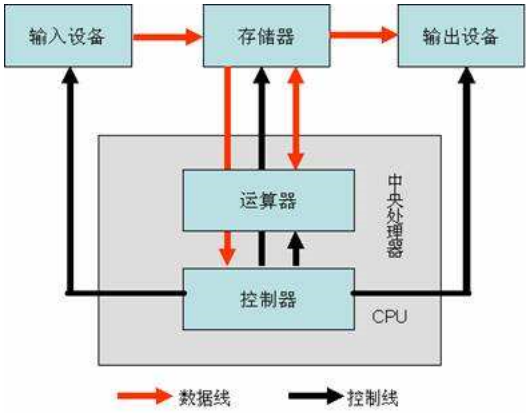
1.1计算机系统的基本概念

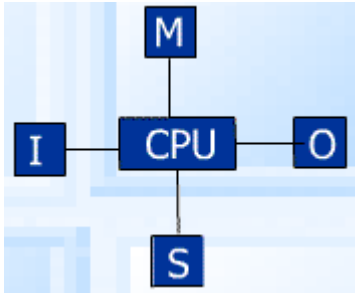
- 系统：为完成某个特定任务的多个功能部件相互配合而形成的有机整体
- 三种常见的说法：
  - 五大部件：运算器、控制器、存储器、输入输出设备



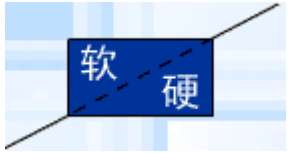
冯·诺依曼式计算机体系结构图



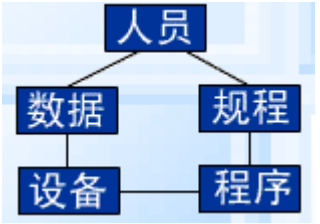




  - 软件和硬件组成



  - 由人员、数据、设备、程序和规程组成



1.2 计算机系统的发展

1.2.1 冯·诺依曼结构

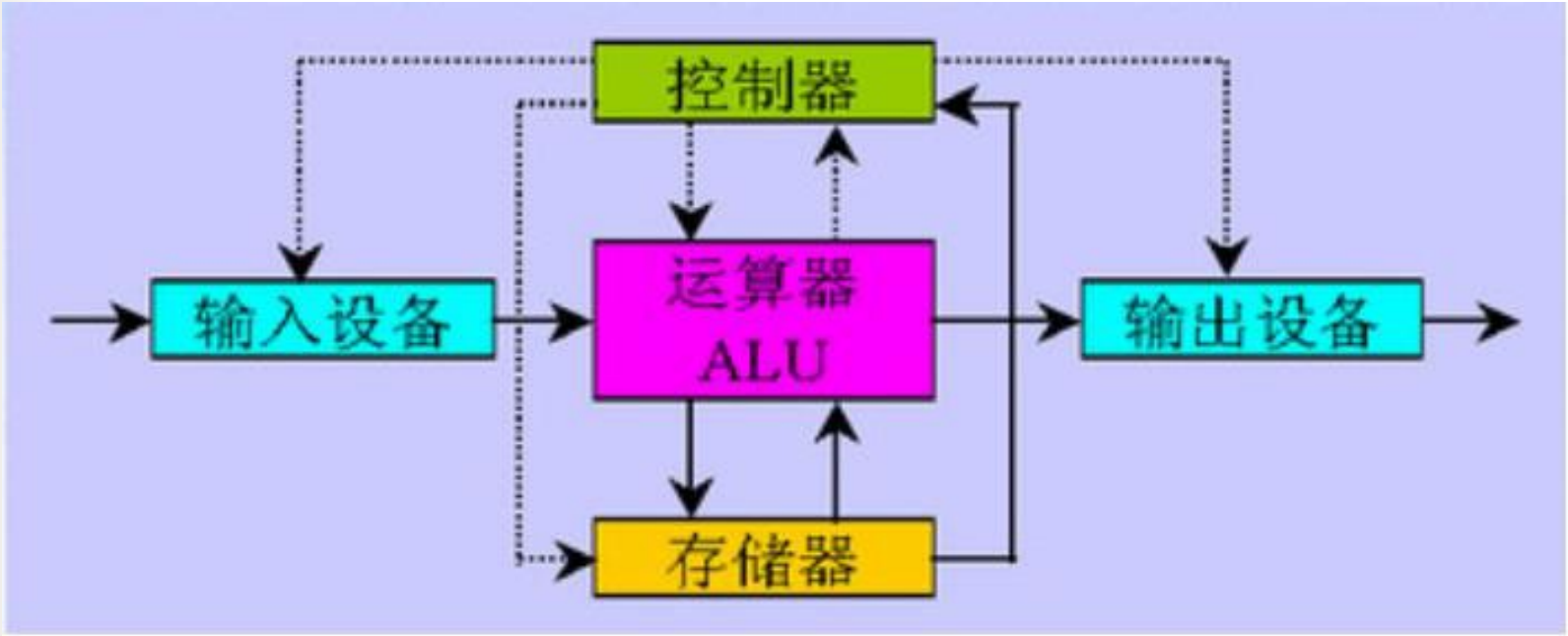
1.2.2 器件发展的影响

1.2.3 应用发展的影响

1.2.4 改进算法的影响

1.2.1 冯·诺依曼结构

- Van Nenmann 基本思想与1936年至1946年期间形成，有冯·诺依曼等人与1946年提出

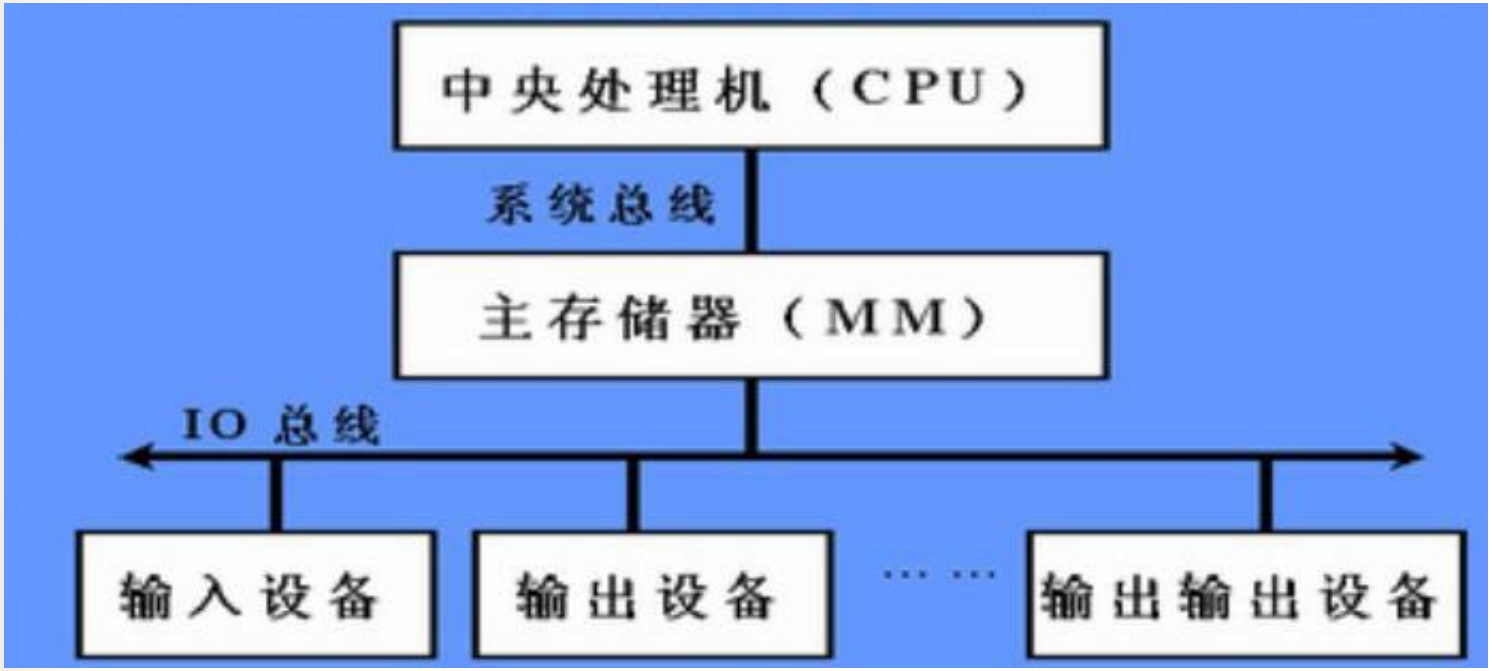


特点：存储程序、运算器为中心、集中控制

- 存储器字长固定，顺序线性编址的一维结构，每个地址是唯一定义的
- 指令有操作码和地址码组成
- 指令顺序执行，即一般按照指令在存储器中存放的顺序执行，程序分支有转移指令实现
- 运算器为中心，输入输出设备与存储器之间的数据传送都途径运算器，各部件的操作及它们之间的联系都有控制器集中控制
- 指令和数据同等对待，均以二进制码表示，采用二进制运算

现代处理机对冯·诺依曼结构的改进

- 不变的：存储程序
- 改变的：存储器为中心，总线结构，分散控制



1.2.2 器件发展的影响

- 第一代至第四代计算机以器件来划分
  - 第一代：电子管(Valve)
  - 第二代：晶体管(Transistor)
  - 第三代：集成电路(LSI)
  - 第四代：大规模集成电路(VLSI)
- 器件发展是提高计算机速度的主要途径
  - MIPS (Million Instructions Per Second) = Fz ×IPC (Instruction Per Cycle)
  - 器件发展对提高处理机主频起决定性作用
  - 需要研究新的器件来提高主频
  - 目前，既依靠提高Fz，也依靠提高IPC

器件发展的特点

- 集成度迅速提高: 芯片上的晶体管数量每18至24个月增加一倍
  - 每0.25平方英寸10^{10}个晶体管
  - 单芯片内可做大于1Gb存储器，单芯片内可以集成2个CPU+全部Cache
  - 还远没有达到集成度的极限
- 速度提高的余地不大
  - 依靠系统结构的发展，关键是并行编译和并行算法
- 价格直线下降：CPU价格每年下降大于80%
- 可靠性越来越高
  - 芯片可靠性达到10^{8}小时，连续使用一万年以上

1.2.3 应用发展的影响

- 应用需求
  - 高结构化数值计算：气象模型、流体流动、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
  - 海量存储和输入输出密集问题：数据库、事物处理
  - 图形学和设计系统：计算机辅助设计
- 三个时期
  - 通用计算机：通用科学计算
  - 专用计算机：科学计算、事物处理、实时控制
  - 高性能通用机：满足多种需求
  - 目前又开始多种专用处理机的研制

- 两个发展趋势
  - 维持价格不变，利用VLSI技术等，提高性能
  - 性能基本不变，价格迅速下降
- 三种设计思想
  - 最高性能价格比：商用机
  - 最高性能：国家安全需要，例如，银河计算机，神舟计算机等
  - 最低价格：家用学习机等

1.2.4 改进算法的影响

- 在多个层次上，算法影响系统结构，例如：
  - 快速乘法、除法、开平方等的实现
  - 记分板算法、Tomasulo算法提高指令级并行性
  - 消除名字相关、数据相关、控制相关的算法
  - 有些问题，如果算法上有突破，不需要高性能的系统结构，而在普通系统上就能得到解决
  - 许多算法还有改进的余地，通过算法的研究能够大幅度提高系统的性能

第一章 基本概念

1.1 计算机系统的基本概念

1.2 计算机系统的发展

1.3 计算机系统的结构

1.4 计算机系统的设计方法

1.5 现代计算机系统的研究领域

1.3 计算机系统的结构

1.3.1 计算机系统的层次结构

1.3.2 计算机系统的定义

1.3.3 计算机组成与实现

1.3.4 计算机系统结构、组成与实现三者之间的关系

1.3.5 计算机系统的特性

1.3.1 计算机系统的层次结构

- 虚拟计算机

  从不同角度所看到的计算机系统属性是不同的
- 主要观察者包括

  应用程序员

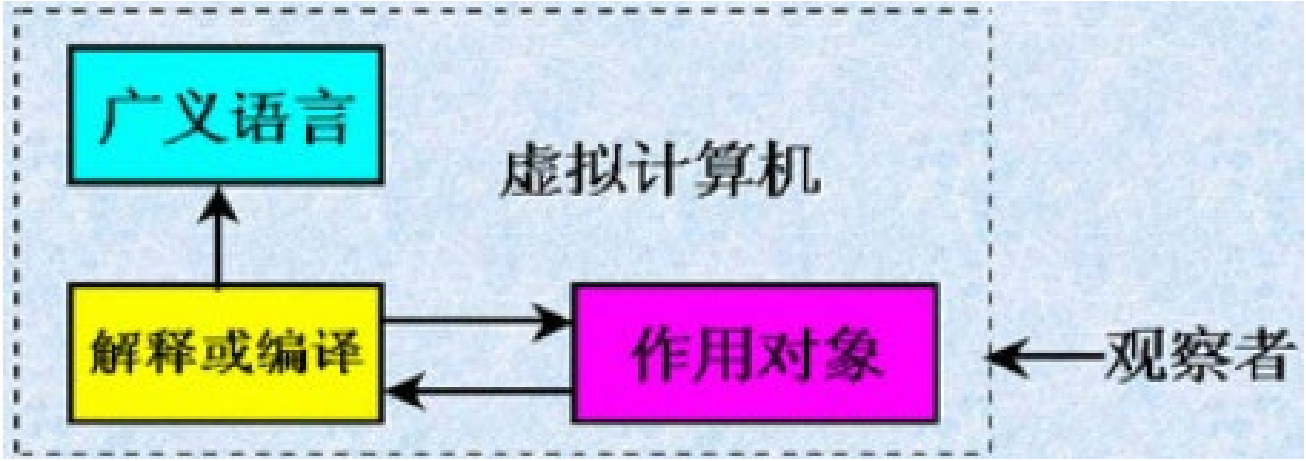
  高级语言程序员

  汇编语言程序员

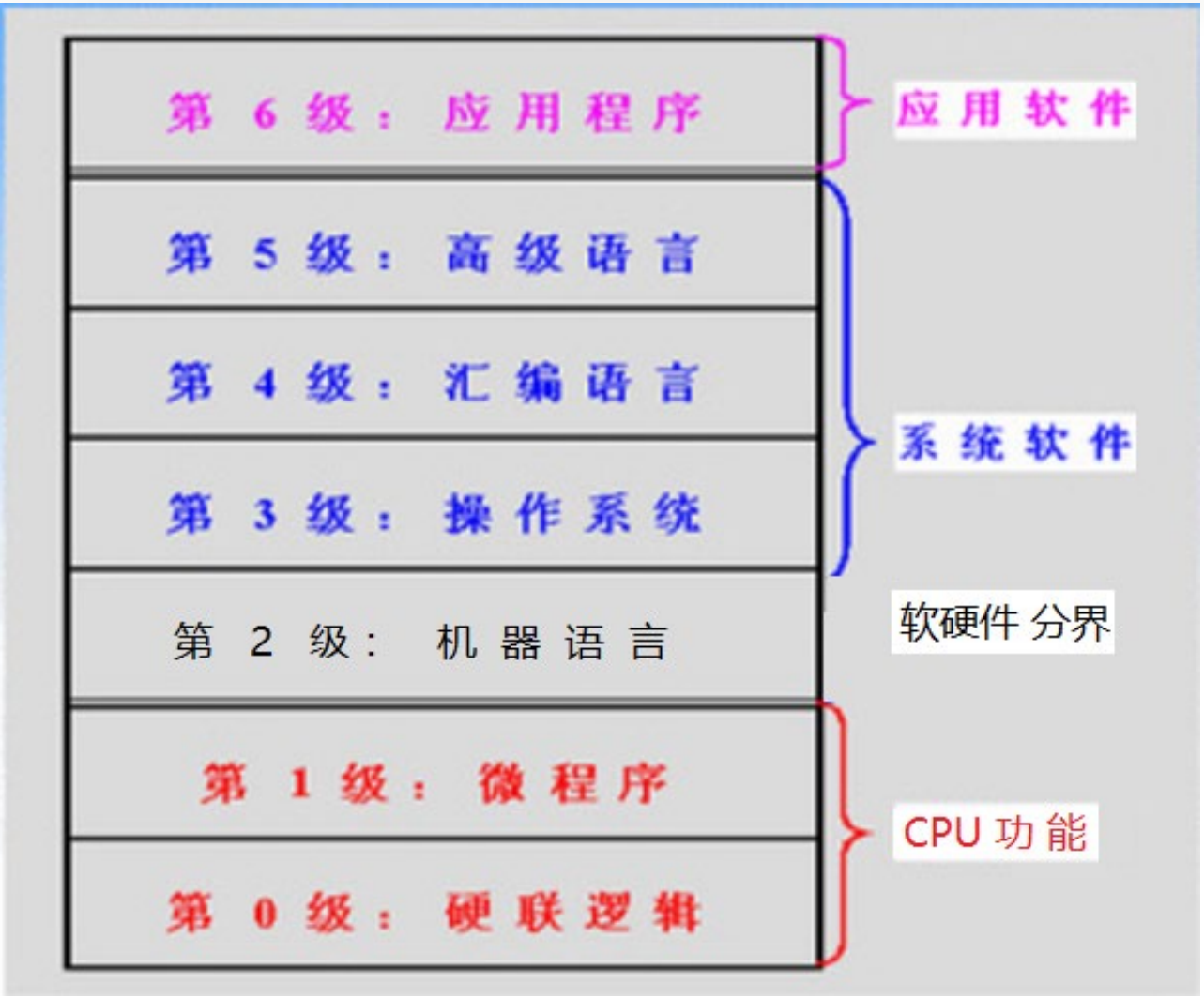
  系统管理员

  硬件设计人员
- 对计算机系统的认识是需要在某一个层次上

虚拟计算机系统

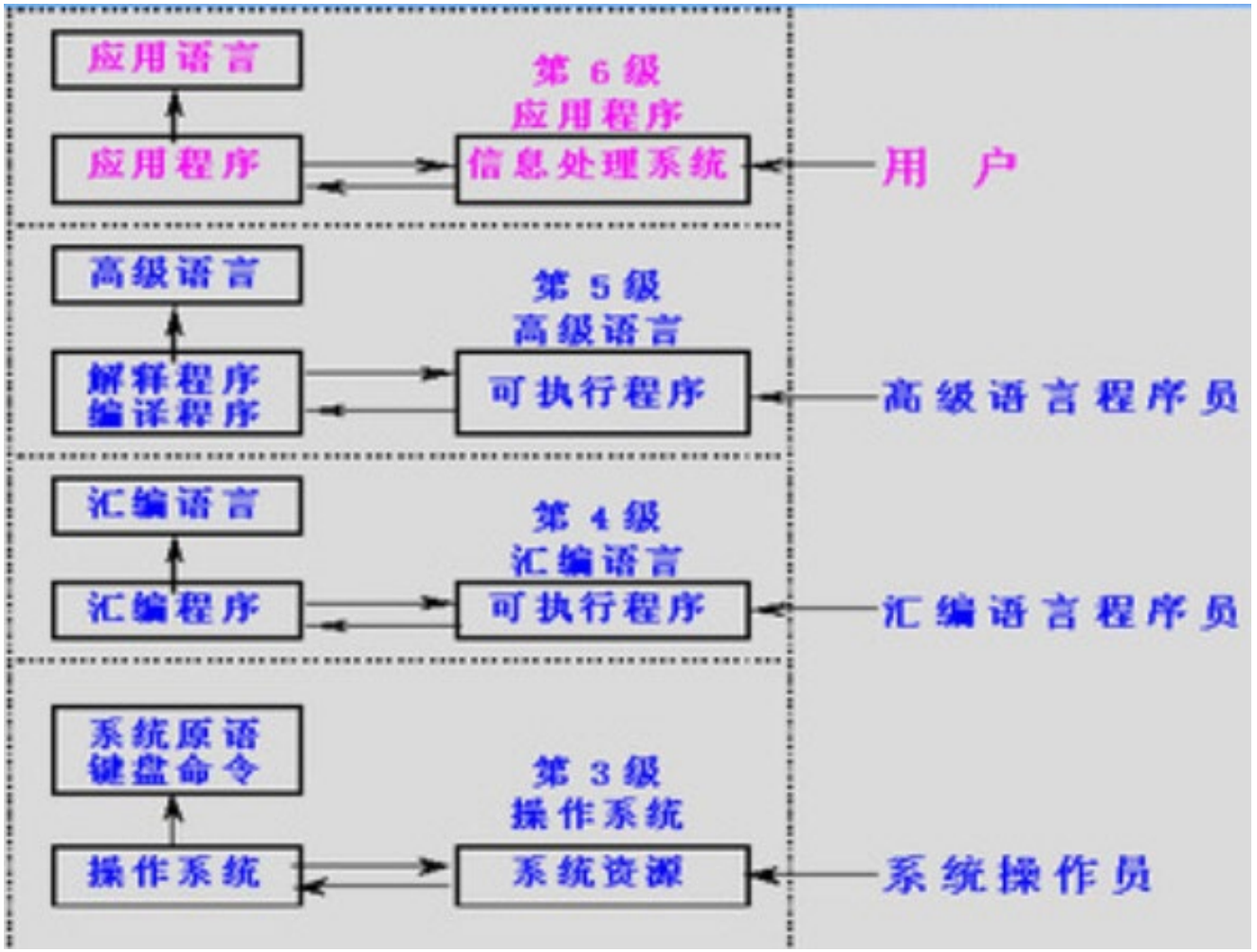


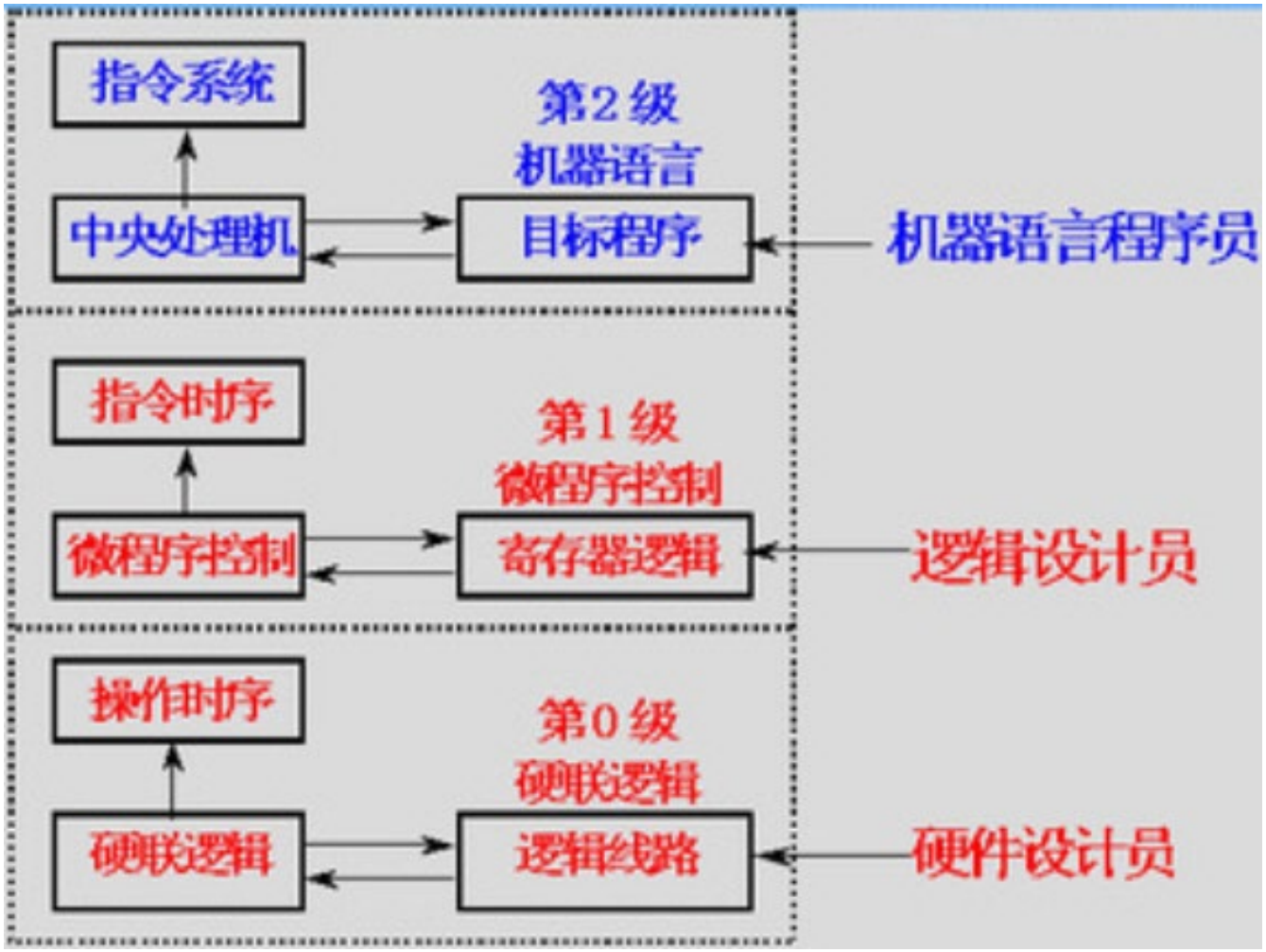
计算机系统的层次结构


第 6 级：应用程序
第 5 级：高级语言
第 4 级：汇编语言
第 3 级：操作系统
第 2 级：机器语言
第 1 级：微程序
第 0 级：硬联逻辑
应用软件
系统软件
软硬件分界
CPU 功能

层次结构

- 共分7个层次
  - 第0级到第1级由硬件实现
  - 第3级到第6级由软件实现，称为虚拟机
- 从学科领域来划分
  - 第0级和第1级属于计算机组成与系统结构
  - 第2级至第5级是系统软件
  - 第6级是应用软件
- 级之间有交叉
  - 例如：第3级必须依赖第4级和第5级来实现


应用语言
第 6 级
应用程序
应用程序
信息处理系统
用 户
高级语言
第 5 级
高级语言
解释程序
编译程序
可执行程序
高级语言程序员
汇编语言
第 4 级
汇编语言
汇编程序
可执行程序
汇编语言程序员
系统原语
键盘命令
第 3 级
操作系统
操作系统
系统资源
系统操作员


指令系统
第2级
机器语言
中央处理机
目标程序
机器语言程序员
指令时序
第1级
微程序控制
微程序控制
寄存器逻辑
逻辑设计员
操作时序
第0级
硬联逻辑
硬联逻辑
逻辑线路
硬件设计员

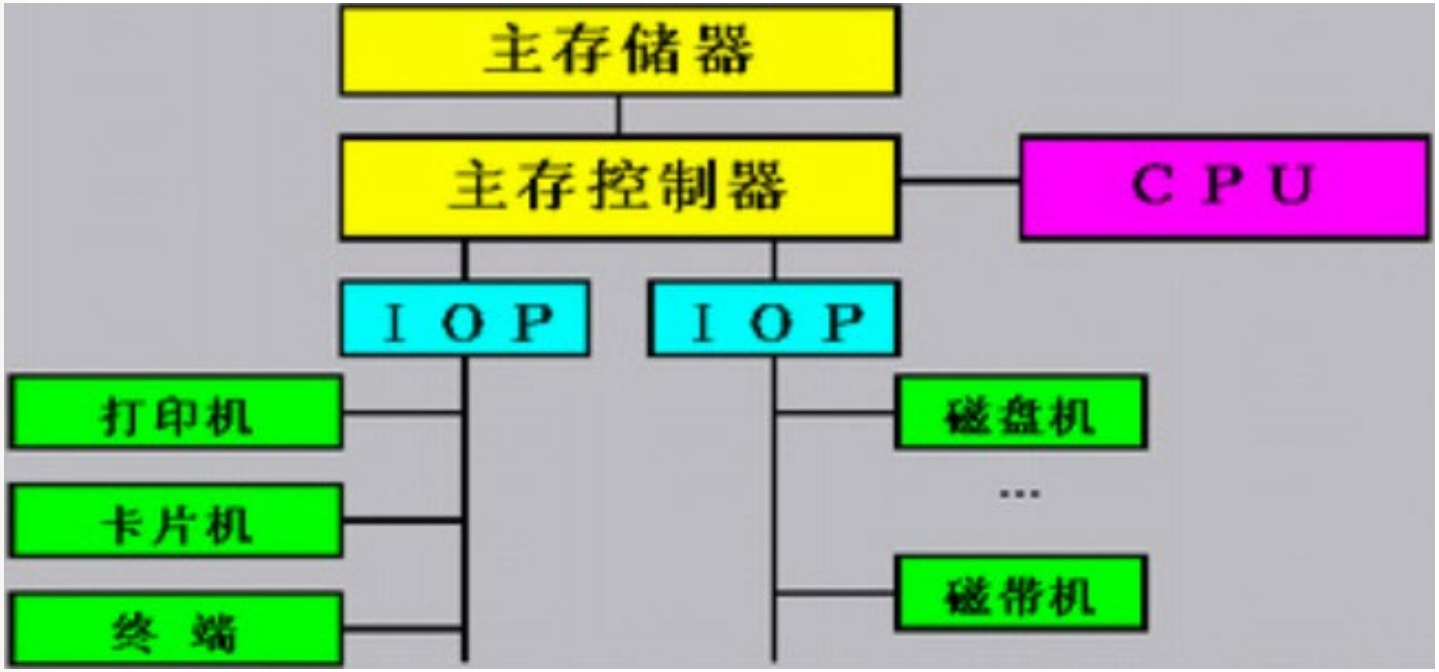
透明性

- 透明性定义
  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　　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

1.3.2 计算机系统结构的定义之一

- Amdahl于1964年推出IBM360系列计算机时提出：
- 程序员所看到的计算机系统的属性，即概念性结构和功能特性
- 程序员：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到的

概念性结构



功能特性指令系统及其执行模式

- 数据表示：硬件能够直接认别和处理的数据类型；
- 寻址方式：寻址单位、寻址方式的种类和地址运算等；
- 寄存器组织：操作数寄存器，变址寄存器，控制寄存器及专用寄存器的定义、数量和使用规则等；
- 指令系统：操作类型、格式，指令间的排序控制等；
- 中断系统：中断类型、中断级别和中断响应方式等；
- 存储系统：编址单位、编址方式和最大寻址空间等；
- 处理机工作状态：定义和切换方式，如管态和目态等；
- 输入、输出系统： 数据交换方式、交换过程的控制等；
- 信息保护：信息保护方式和硬件对信息保护的支持等。

计算机系统结构的定义之二

- 研究软硬件功能分配和对软硬件界面的确定
- 计算机系统由软件、硬件和固件组成，它们在功能上是同等的
- 同一种功能可以用硬件实现，也可以用软件或固件实现
- 不同的组成只是性能和价格不同，它们的系统结构是相同的

计算机系统结构的定义----两个角度

- 使用者：外部，正确使用计算机所具备的知识（指令系统，输入/输出，存储、中断等）
- 设计者：外部，如何设计，具备什么样的功能，这些功能用软件还是硬件实现

1.3.3 计算机组成

- 课程名称 Computer Organization

  计算机组成、计算机组织、
  计算机原理、计算机组成原理

- 研究方法

  从内部研究计算机系统

  计算机组成是指计算机系统结构的逻辑实现

计算机组成的研究内容

- 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

计算机实现技术

- 计算机实现是指计算机组成的物理实现
- 主要包括：
  - 处理机、主存储器等部件的物理结构
  - 芯片的集成度和速度
  - 专用芯片的设计
  - 器件、模块、插件、底板的划分与连接
  - 信号传输技术
  - 电源、冷却及装配技术，制造工艺及技术等

1.3.4计算机系统结构、组成和实现之间的关系

- 计算机系统结构→确定软硬件界面
- 计算机组成→逻辑实现
- 计算机实现→物理实现

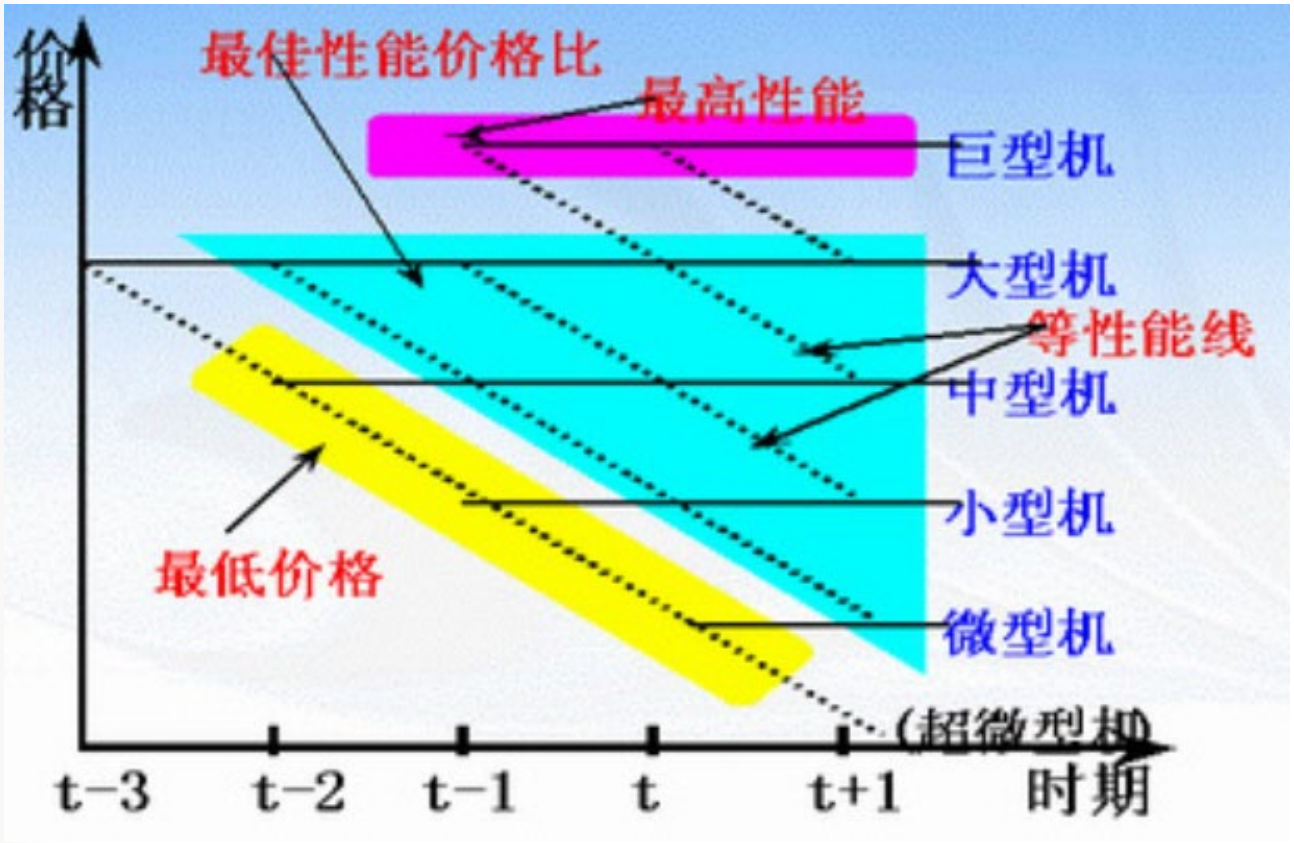
- 例1：确定指令系统的功能→系统结构；指令的逻辑实现→组成；指令功能的物理实现→实现
- 例2：确定是否需要乘除指令→系统结构；乘除指令用乘法器还是用加法器累加→组成；乘法器或加法器的物理实现→实现
- 随着技术、器件和应用的发展，三者之间的界限越来越模糊

三者的设计原则

- 系统结构设计不要对组成、实现有过多和不合理限制
- 组成设计应在系统结构指导下，以目前可实现技术为基础
- 实现应在组成的逻辑结构指导下，以目前器件技术为基础，以性能价格比优化为目标

1.3.5 计算机系统的特性

- 计算机等级：巨、大、中、小、微



系列机

- 定义：

  具有相同的系统结构，但组成和实现技术不同的一系列计算机系统

- 实现方法：
  - 在系统结构基本不变的基础上，根据不同的性能和不同的器件，研制出多种性能和价格不同的计算机系统
  - 一种系统结构可以有多种组成，一种组成也可以有多种物理实现
  - 如IBM370系列机：115，125，135，145，158，168等

软件兼容

- 软件相对于硬件成本越来越贵，已积累了大量成熟的系统软件和应用软件
- 软件兼容：同一个软件的目标程序可以不加修改地运行于系统结构相同的任何机器上
- 兼容种类：
  - 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
  - 向前兼容：
  - 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
  - 向下兼容：
- 向后兼容必须做到，向上兼容尽量做到
- 向前兼容和向下兼容，可以不考虑

- 采用系列机方法的主要优点：
  - 系列机之间软件兼容，可移植性好
  - 插件、接口等相互兼容
  - 便于实现机间通信
  - 便于维修、培训
  - 有利于提高产量、降低成本
- 采用系列机方法的主要缺点：
  - 限制了计算机系统结构的发展，如PC系列机，其系统结构非常落后，使用也最普及

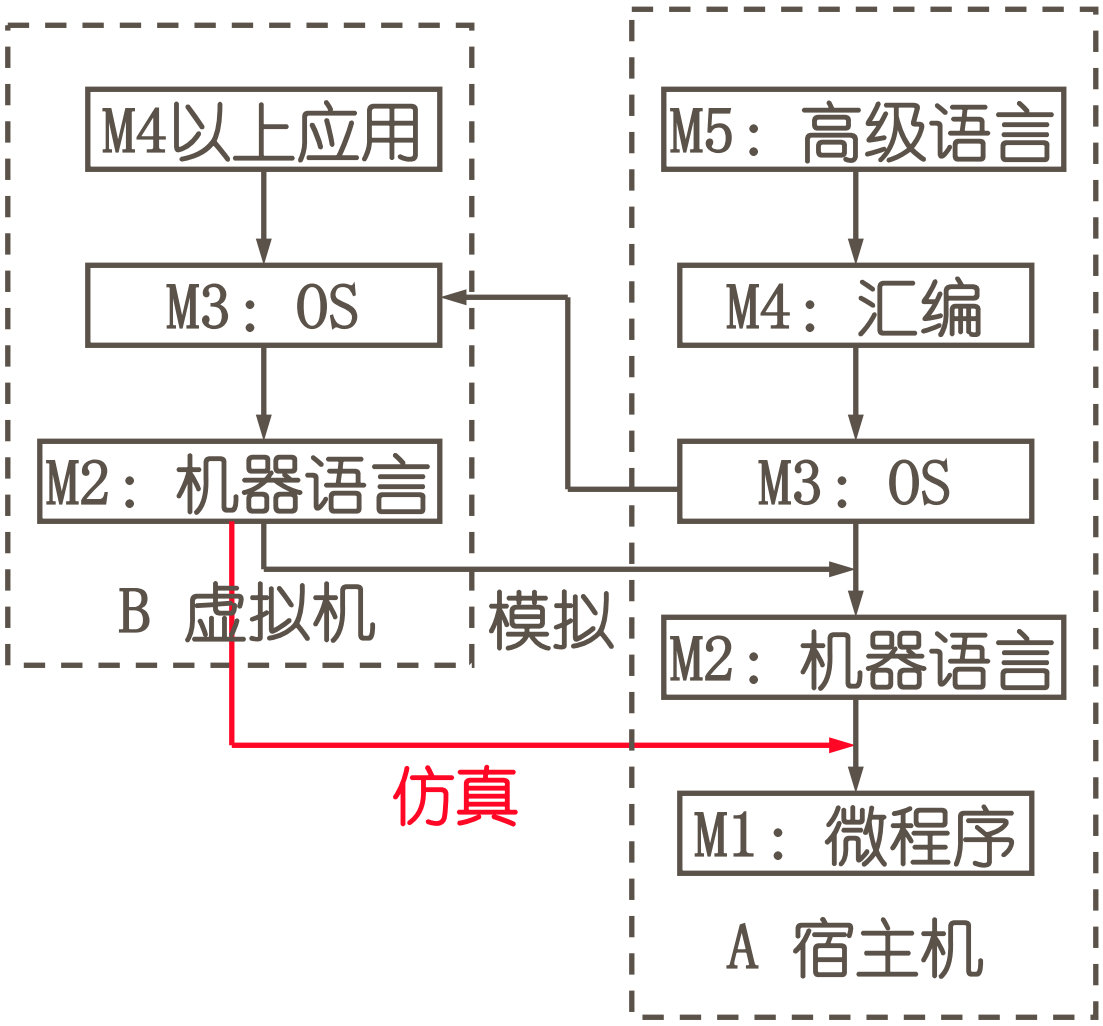
模拟与仿真

- 定义：在某个系统结构之上实现另一种系统结构。在一台现有的计算机上实现另一台计算机的指令系统。全部用软件实现的叫模拟，用软件、硬件、固件混合实现的叫仿真
- 模拟的实现方法

  在A计算机上通过解释或编译实现B计算机的指令系统。A机器称为宿主机，B机器称为虚拟机。
- 仿真的实现方法

  直接用A机器的一段微程序解释执行B机器的指令，A机器称为宿主机，B机器称为目标机

模拟与仿真


M4以上应用
M3：OS
M2：机器语言
B 虚拟机
模拟
M5：高级语言
M4：汇编
M3：OS
M2：机器语言
仿真
M1：微程序
A 宿主机

- 除了指令系统外，存储系统、操作系统、I/O系统等系统结构全貌

模拟与仿真优缺点比较

- 模拟方法灵活，可实现程序在任何机器之间的相互移植，但实现复杂、速度低，因此适合移植运行时间不长，使用次数少的程序
- 仿真速度快，但难以仿真存储系统和I/O系统，因此只适应于系统结构差别不大的机器之间的移植

第一章 导论

1.1 计算机系统的基本概念

1.2 计算机系统的发展

1.3 计算机系统的层次结构

1.4 计算机系统的设计方法

1.5 现代计算机系统的研究领域

1.4 计算机系统的设计方法

1.4.1 软、硬件取舍的基本原则

1.4.2 计算机系统设计的定量原则

1.4.3 计算机系统的设计任务

1.4.4 计算机系统的设计步骤

1.4.1 软硬件取舍

- 软硬件的关系
  - 理论上，两种极端实现方法：

    全硬件机器：操作系统、高级语言、应用等

    硬件只有1位加法和分支操作，其它都用软件
  - 软件与硬件实现的特点

    硬件实现：速度快、成本高、灵活性差、占用内存少

    软件实现：速度低、复制费用低、灵活性好、占用内存多
  - 关键问题：性能与价格的关系

从价格因素考虑的软硬件取舍

- 假设：硬件设计费为Dh，软件设计费为Ds，硬件拷贝费为Ch，软件拷贝费为Cs， C是软件重复设计的次数，R为软件重复出现次数(占用内存、占用介质)，当台数为V时，每台的硬件费用和软件费用之比为：

$$\left(\frac{Dh}{V} + Ch\right) : \left(\frac{C \cdot Ds}{V} + R \cdot Cs\right)$$

只有当 $(\frac{Dh}{V} + Ch) < (\frac{C \cdot Ds}{V} + R \cdot Cs)$ 时，适合用硬件实现

（1）设 $Dh = 100Ds$，$Ch = 100Cs$

$$(\frac{100Ds}{V} + 100Cs) < (\frac{C \cdot Ds}{V} + R \cdot Cs)$$

$C > 100$ 且 $R > 100$，经常用到的功能适宜用硬件实现

（2）设 $Ds = 10^4 Cs$

$$(\frac{10^6}{V} + 100) < (\frac{10^4 \cdot C}{V} + R)$$

$$10^6 + 100V < 10^4 \cdot C + R \cdot V$$

$$10^4(10^2 - C) < (R - 100) \cdot V$$

$C < 100$，$R > 100$，则当V较大时适宜用硬件实现

1.4.2 计算机系统设计的定量原则

- 原则：加快经常性事件的速度
- Amdahl定律：系统中某一部件由于采用更快的执行方式后，整个系统性能的提高与这种执行方式的使用频率或占总执行时间的比例有关

$$\text{加速比} = \frac{\text{采用改进措施后的性能}}{\text{未采用改进措施前的性能}} = \frac{\text{未采用改进措施前执行某任务时间} T_O}{\text{采用改进措施后执行某任务时间} T_n}$$

- 在Amdahl定律中，加速比与两个因素有关

可改进部分的比例：$Fe = \dfrac{\text{可改进部分的执行时间}}{\text{改进前整个任务的执行 时间}}$

改进部分的加速比：$Se = \dfrac{\text{改进前改进部分的执行 时间}}{\text{改进后改进部分的执行 时间}}$

改进后整个任务的执行时间为：$T_n = T_0 \times (1 - Fe + \dfrac{Fe}{Se})$

其中，T_0和T_n分别为改进前后整个任务的执行时间

改进后整个系统的加速比达到：$S_n = \dfrac{T_0}{T_n} = \dfrac{1}{(1 - Fe) + \dfrac{Fe}{Se}}$

其中$(1 - Fe)$表示不可改进部分

- 例1：设系统中某部件的处理时间占整个运行时间的40%，如果将该部件的处理速度加快到10倍，则采用加快措施后能使整个系统的性能提高多少？

$$Fe = 0.4 \qquad Se = 10$$

$$S_n = \frac{T_0}{T_n} = \frac{1}{(1-Fe) + \frac{Fe}{Se}} \approx 1.56$$

- 例2：设某个测试程序执行过程中，浮点数平方根FPSQR操作占总执行时间的20%，一种实现方法是用硬件实现FPSQR，使其速度加快到10倍，另一种实现方法是使所有浮点数指令FP速度加快到2倍，同时，设FP指令占整个测试程序执行时间的50%，比较两种方案的优劣。

$$FPSQR: \quad Fe_1 = 0.2 \qquad Se_1 = 10$$

$$S_1 = \frac{1}{1 - Fe_1 + Fe_1 / Se_1} = 1.22$$

$$FP: \quad Fe_2 = 0.2 \qquad Se_2 = 2$$

$$S_2 = \frac{1}{1 - Fe_2 + Fe_2 / Se_2} = 1.33$$

CPU性能公式

- CPU性能取决三个要素：时钟频率f；每条指令时钟周期数；指令条数IC。时钟周期T=1/f。
- 一个程序执行所需的CPU时间如下：

$$\text{CPU时间} = \frac{\text{CPU时钟周期总数}}{\text{时钟频率f}} = \text{CPU时钟周期总数} \times \text{时钟周期}T$$

如果一个程序的指令条数为IC,则每条指令的平均时钟周期数CPI

$$\text{CPI(Cycles Per Instruction)} = \frac{\text{CPU时钟周期总数}}{\text{IC}}$$

由此可得：

$$\text{CPU时间} = \text{IC} \times \text{CPI} \times \text{T} = \text{IC} \times \text{CPI} \times \frac{1}{\text{f}}$$

一个程序的CPU时钟周期总数也可表示为：

$$\text{CPU时钟周期总数} = \sum_{i=1}^{n}(\text{CPI}_i \times \text{I}_i)$$

I_i是i指令在程序中的条数，

CPI_i是i指令的平均时钟周期数，n为程序中指令的种类数

因此，$\text{CPU时间} = \text{T} \times \sum_{i=1}^{n}(\text{CPI}_i \times \text{I}_i)$

$$\text{CPI} = \frac{\sum_{i=1}^{n}(\text{CPI}_i \times \text{I}_i)}{\text{IC}} = \sum_{i=1}^{n}(\text{CPI}_i \times \frac{\text{I}_i}{\text{IC}}),$$

$\frac{\text{I}_i}{\text{IC}}$表示$i$指令在程序中所占的比例：操作的比例

- 例3：在一个程序中，如果浮点操作FP的比例为25%，FP的CPI为4，其他指令的CPI为1.33。浮点开平方根操作FPSQR比例为2%，FPSQR的CPI为20。如有两种方案，分别把FPSQR操作的CPI和所有FP操作的CPI均减至2，试利用CPU性能公式分析这两种方案的优劣

$$\text{加速比}S_0=\frac{\text{改进后的性能}}{\text{改进前的性能}}=\frac{\text{原}CPU\text{时间}T_0}{\text{改进后}CPU\text{时间}}$$

$$\text{原}CPU\text{时间}T_0=CPI*IC*T$$

$$FPSQR:\quad \text{改进后}CPU\text{时间}T_{n1}=CPI_1*IC*T$$

$$FP:\quad \text{改进后}CPU\text{时间}T_{n2}=CPI_2*IC*T$$

$$S_{n1}=\frac{T_0}{T_{n1}}=\frac{CPI*IC*T}{CPI_1*IC*T}=\frac{CPI}{CPI_1}\qquad S_{n2}=\frac{T_0}{T_{n2}}=\frac{CPI*IC*T}{CPI_2*IC*T}=\frac{CPI}{CPI_2}$$

$$CPI=4*0.25+1.33*0.75=2$$

$$CPI_1=2-20*0.02+2*0.02=1.64\qquad CPI_2=2-4*0.25+2*0.25=1.5$$

$$S_{n1}<S_{n2}$$

- 例4：设有两台机器A和B，对条件转移指令的处理采用不同方法。CPU_A采用一条比较指令来设置相应的条件码，由紧随其后的一条转移指令对此条件码进行测试，以确定是否进行转移。显然，实现一条条件转移要执行比较和测试两条指令。CPU_B采用比较功能和判别是否实现转移功能合在一条指令的方法，这样实现一条条件转移只需一条指令就可完成。
- 假设在这两台机器的指令系统中，执行条件转移指令均需2个时钟周期，而其他指令只需一个时钟周期。又假设在CPU_A中，条件转移指令占总执行指令条数的20%，由于每条转移指令都需要一条比较指令，所以比较指令也将占20%。CPU_B的时钟周期就比CPU_A要慢25%，现完成同一功能，比较CPU_A和CPU_B，哪个工作速度更快些？

CPU时间$=IC*CPI*T$

$CPU_A=IC_A*CPI_A*T_A$ $\qquad$ $CPU_B=IC_B*CPI_B*T_B$

$CPI_A=2*0.2+1*0.8=1.2$ $\qquad$ $CPU_A=IC_A*1.2*T_A$

A：20%条件转移　20%比较　60%其他

B：20%条件转移　60%其他

$CPI_B=\frac{20\%}{80\%}*2+\frac{60\%}{80\%}*1=1.25$ $\qquad$ $CPU_B=IC_B*1.25*T_B$

$T_B=1.25T_A$

$IC_B=0.8IC_A$

$CPU_B=0.8IC_A*1.25*1.25T_A=1.25IC_A*T_A$

$CPU_A=IC_A*1.2*T_A$

$CPU_B>CPU_A$

指令执行速度

- 一种经典的表示运算速度的方法
- MIPS (Million instructions Per Second), GIPS, TIPS

$$MIPS = \frac{\text{指令条数}}{\text{指令执行时间}\times 10^6} = \frac{\text{f}}{\text{CPI}\times 10^6} = \frac{\text{f}\times \text{IPC}}{10^6}$$

  - f为处理机的工作主频
  - CPI (Cycles Per Instruction)为每条指令所需的平均时钟周期数
  - IPC (Instruction Per Cycle)为每个时钟周期平均执行的指令条数

例4：计算PentiumIV 2GHz 的指令执行速度

解： 由于PentiumIV 2GHz处理机的

IPC = 4 (或CPI = 0.25)

f= 2000MHz

因此，MIPS = f×IPC = 8000MIPS

=8GIPS

- 主要优点：直观，方便。目前仍经常使用
- 主要缺点：
  - 不同指令的执行速度差别很大
  - 指令使用频度差别很大

多核时代的Amadahl定律

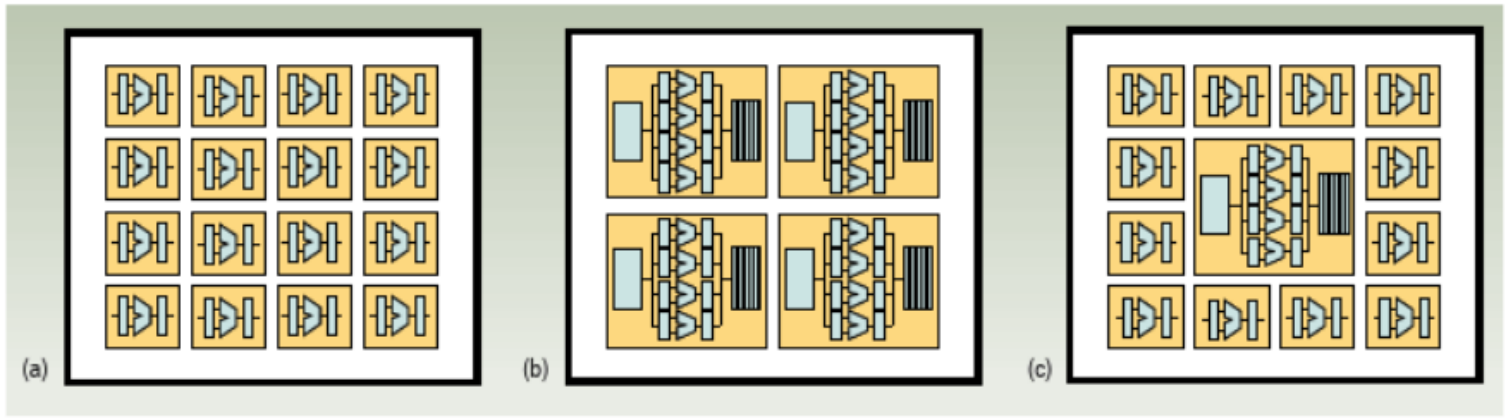


图1 多核芯片的不同种类：（a）内置16个单BCE内核的对称多核芯片；（b）内置4个四BCE内核的对称多核芯片；（c）内置1个四BCE内核和12个单BCE内核的非对称多核芯片。在图中，省略了一些重要结构，例如存储接口、共享缓存以及互连等。另外还假设面积是芯片的限制资源，而功耗不是。

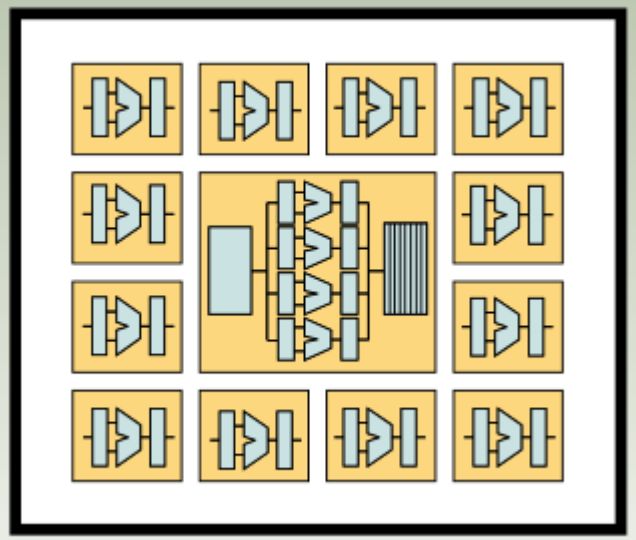
- BCE Base Core Equivalent：等价基本核
- n：BCE 总个数
- r：每个内核的BCE个数
- *perf(r)*：使用一个内核进行串行计算的性能
- *perf(r)*×*n/r*：使用n/r个内核进行并行计算，从而达到的性能

多核时代的Amadahl定律

根据Amadahl定律，对称多核芯片的加速比取决于可并行软件部分f

$$Speedup_{symmetric}(f,n,r)=\frac{1}{\frac{1-f}{perf(r)}+\frac{f\cdot r}{perf(r)\cdot n}}$$

多核时代的Amadahl定律



非对称（异构）多核芯片：一个或者多个内核比其它的内核功能要强大一些 。如图c所示，根据阿姆达尔定理的简单化假设，用额外的资源专门增强一个单核的能力会最为有益。

给定资源预算n=16个BCE，一个非对称多核芯片可以配置成一个内置4个BCE的内核加上12个内置1个BCE的内核。

一个较大的内核需要资源数r，剩下的资源数量n-r可全部配置成单BCE内核，所以可在这个芯片中配置数量为1+n-r的内核。

Amadal定律对非对称多核芯片有着不同的影响。一个较大内核的性能为perf（r）。但并行的部分性能来源于这个较大内核的perf（r）加上n-r个基本内核。整体上，得到：

$$Speedup_{asymmetric}(f,n,r)=\frac{1}{\frac{1-f}{perf(r)}+\frac{f}{perf(r)+n-r}}$$

1.4.3 计算机系统的设计任务

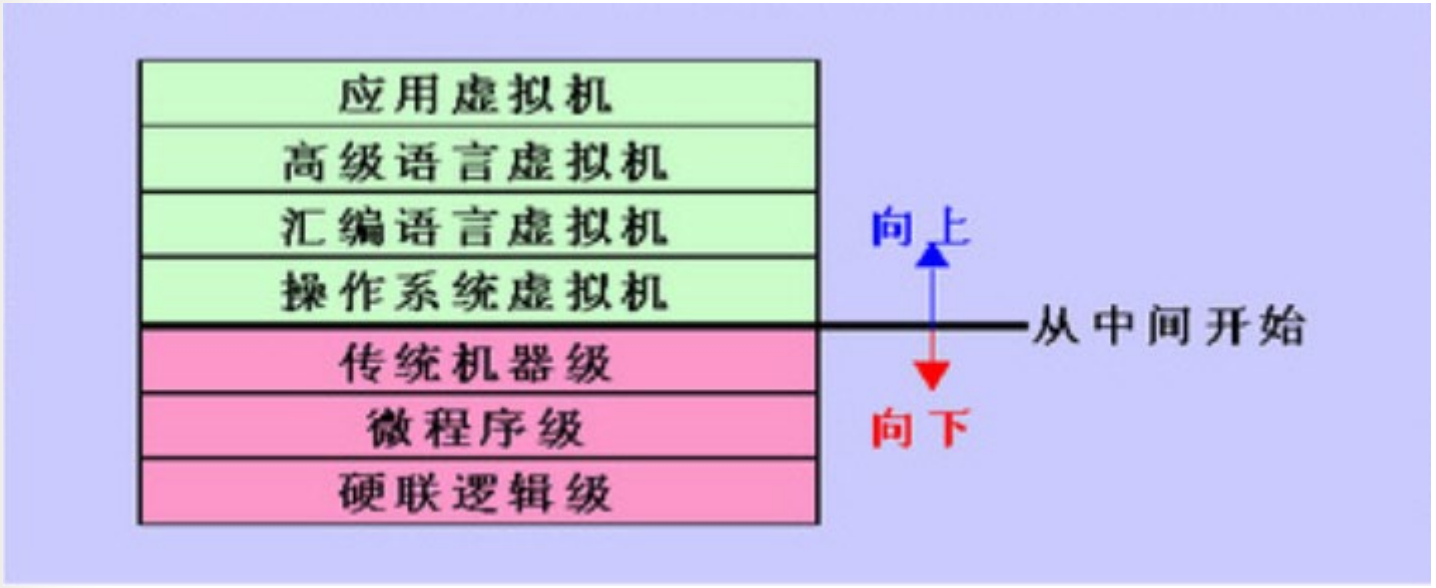
- 方法1：由上向下(Top-Down)
- 设计过程：

  面向应用的数学模型→面向应用的高级语言→面向这种应用的操作系统→面向操作系统和高级语言的机器语言→面向机器语言的微指令系统和硬件实现
- 应用场合：专用计算机的设计
- 特点：对于所面向的应用领域，性能和性能价格比很高，随着通用计算机价格降低，目前已经很少采用

- 方法2：由下向上(Bottom-up)
- 设计过程：

  根据当时的器件水平，设计微程序级和传统机器级→根据不同的应用领域涉及多种操作系统、汇编语言、高级语言编译器→最后设计面向应用的用户级
- 应用场合：通用计算机的一种设计方法，在计算机早期设计中（60至70年代）广为采用
- 特点：容易使软件和硬件脱节，整个计算机系统的效率降低

- 方法3：中间开始（Middle-Out）
- 用于系列计算机的设计过程中



- 方法3：中间开始（Middle-Out）
- 设计过程：

  首先定义软硬件的分界面（指令系统、存储系统、输入输出系统、中断系统、硬件对操作系统和编译系统的支持等）

  然后各个层次分别进行设计（软件设计人员设计操作系统、高级语言、汇编语言、应用程序等，硬件设计人员设计传统机器、微程序、硬联逻辑等）
- 应用场合：用于系列机的设计
- 特点：软硬件人员结合、同时设计，软硬件功能分配合理

第一章 基本概念

1.1 计算机系统的基本概念

1.2 计算机系统的发展

1.3 计算机系统的层次结构

1.4 计算机系统的设计方法

1.5 现代计算机系统的研究领域

1.5.1 计算机系统结构分类

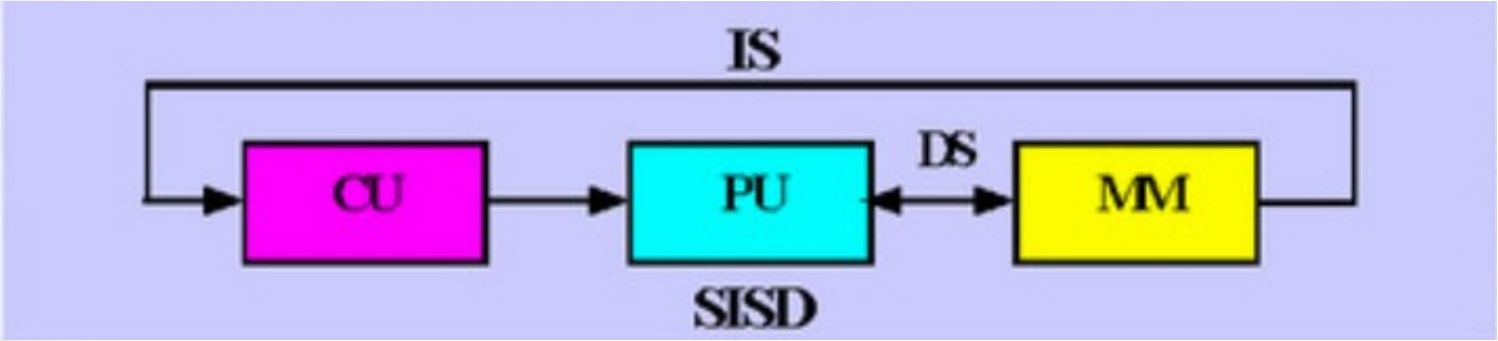
- Flynn(佛林)分类法
- 冯氏分类法
- Handler(汉德勒)分类法

Flynn分类法

- 1966年由Michael.J.Flynn提出
- 按照指令流和数据流的多倍型特征进行分类
  - 指令流：机器执行的指令序列
  - 数据流：由指令流调用的数据序列
  - 多倍性(multiplicity)：在系统性能瓶颈部件上同时处于同一执行阶段的指令或数据的最大可能个数

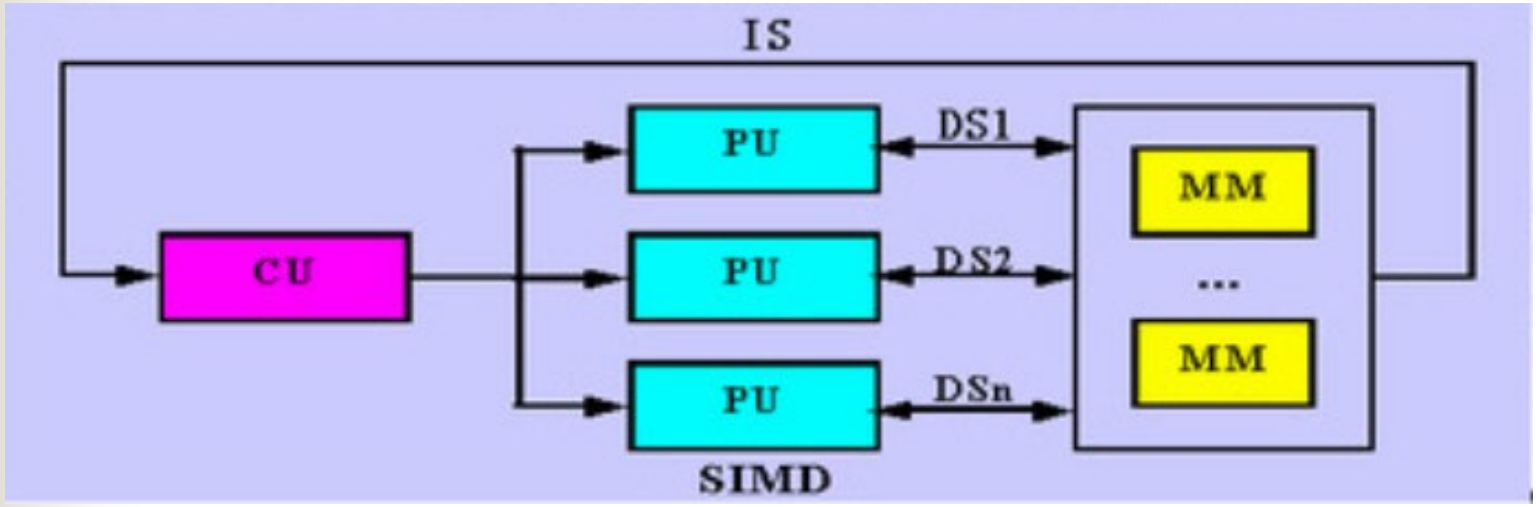
- 四种类型
  - 单指令流单数据流SISD(Single Instruction Single Datastream)
  - 单指令流多数据流SIMD(Single Instruction Multiple Datastream)
  - 多指令流单数据流MISD(Multiple Instruction Single Datastream)
  - 多指令流多数据流MIMD(Multiple Instruction Multiple Datastream)

- SISD典型单处理机，包括：
  - 单功能部件处理机：IBM1401,VAX-11
  - 多功能部件处理机：IBM360/91,370/168
  - 流水线处理机：标量流水处理机



SIMD

- 并行处理机、阵列处理机、向量处理机、相联处理机、超标量处理机、超流水线处理机
- 多个PU按一定方式互联，在同一个CU控制下，对各自的数据完成同一条指令规定的操作；从CU看指令顺序执行，从PU看数据并行执行



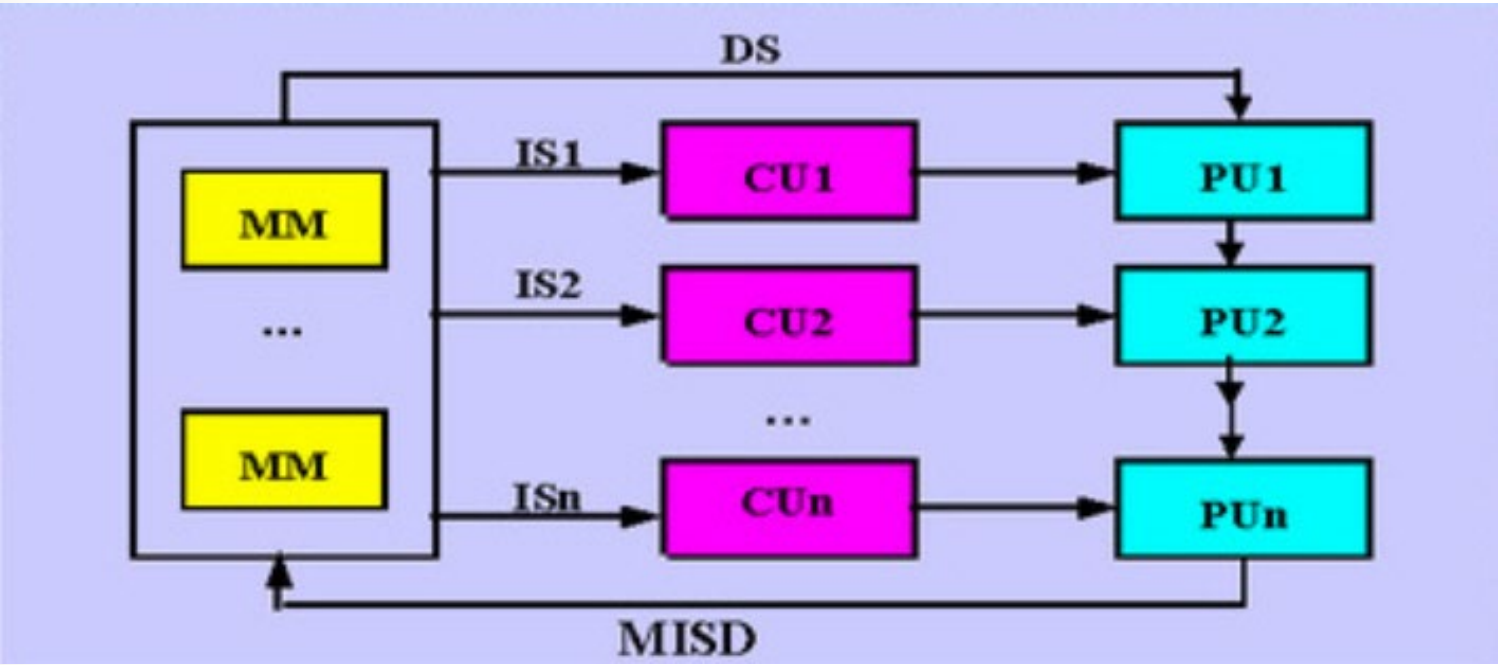
应用：在CPU、GPU等加速部件中广泛应用

- MMX（MultiMedia eXtensions): Intel MM0~MM7
- SSE（Streaming SIMD Extensions): SSE
- AVX（Advanced Vector eXtensions）
- ARM的SVE

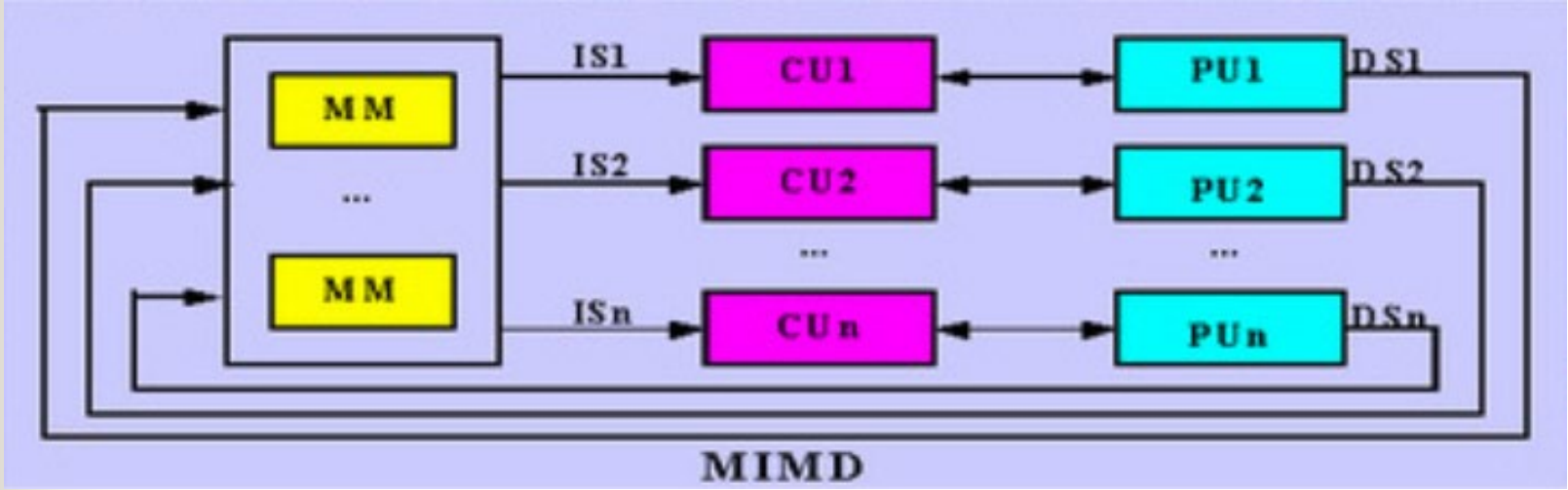
- SIMT Single Instruction Multiple Threads

  - Nvidia公司提出的，SIMD在GPU上的应用。
  - 针对批量的同种指令进行并行计算
  - SIMD 和SIMT架构类似，SIMT将一条指令并行应用于多个独立的进程，而非多个数据通道。

- MISD：几条指令对同一个数据进行不同的处理，实际上不存在

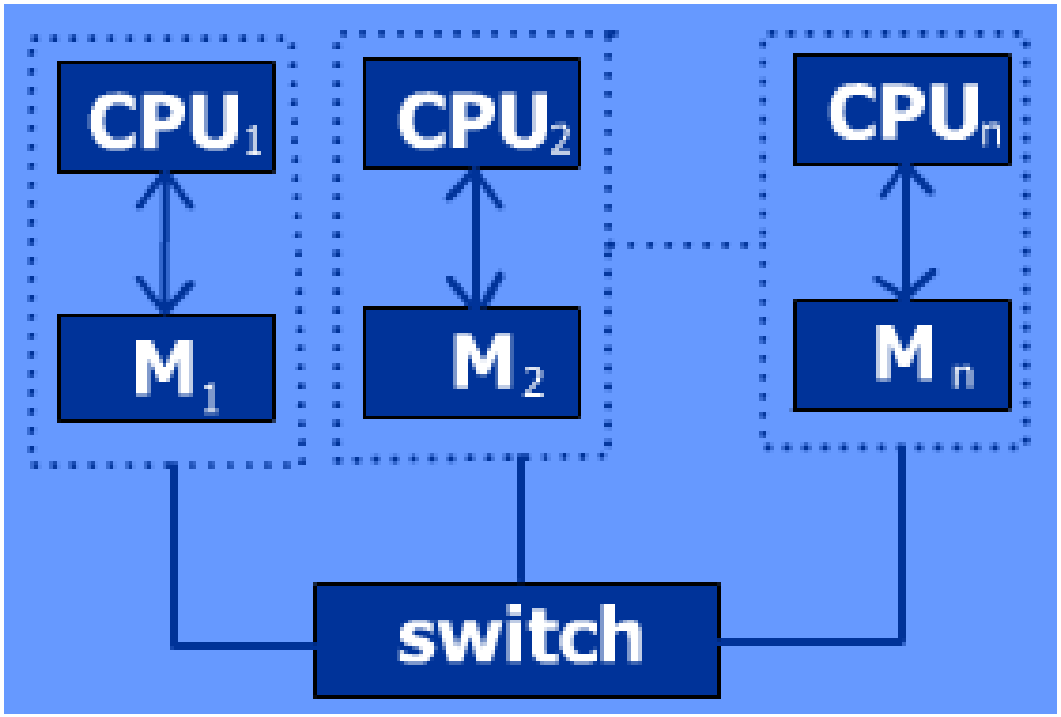


- MIMD多处理机系统，包括:
  - 紧密耦合：IBM3081、IBM3084
  - 松散耦合：D-825、Cmmp,CRAY-2



- SPMD:单程序多数据流，属于MIMD

  基于Cluster



- Flynn分类法反映了大多数计算机的并行性、工作方式和结构特点
- 主要缺点
  - 分类太粗

    在SIMD中包括有多种处理机

    对流水线处理机的划分不明确

    标量流水线为SISD，向量流水线为SIMD
  - 根本问题是把两个不同等级的功能并列对待

    数据流受指令流控制，造成MISD不存在
  - 非冯计算机的分类？其他新型计算机的分类

冯氏分类法

- 1972年美籍华人冯泽云提出
- 用最大并行度对计算机系统进行分类

单位时间能处理的最大二进制位数

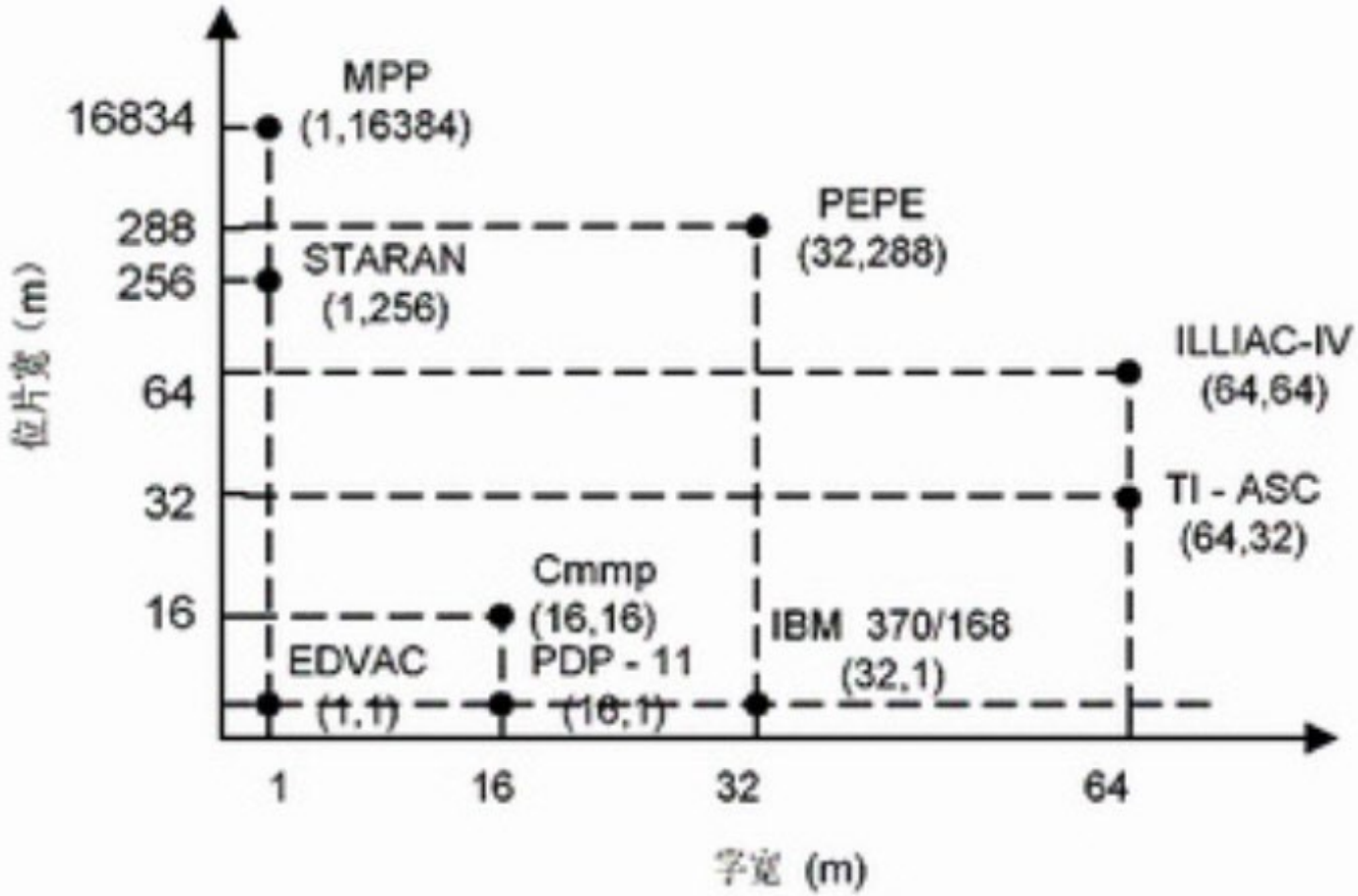
例如：同时处理的字宽为n,位宽为m,则最大并行度定义为：

$$P_m = m \times n$$

平均并行度：假设每个时钟周期t_i内能同时处理的二进制位数为B_i，则n个时钟周期内的平均并行度为：

$$P_n = \left(\sum_{i=1}^{n} B_i \Delta t_i \right) \Big/ n$$

- 表示方法：处理机名(m, n)


MPP
(1,16384)
16834
288
256
64
32
16
位片宽（m）
PEPE
(32,288)
STARAN
(1,256)
ILLIAC-IV
(64,64)
TI - ASC
(64,32)
Cmmp
(16,16)
IBM 370/168
(32,1)
EDVAC
(1,1)
PDP - 11
(16,1)
1
16
32
64
字宽 (m)

(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)

传统单处理机： m＝1, n > 1； 如：Pentium(32, 1)

(3)字串位并WSBP(Word Serial and Bit Parallel)

并行计算机、MPP、相联计算机：m > 1, n ＝1；

如：MPP(1, 16384), STARAN(1, 256)，DAP

(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)，

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

- 主要缺点：

  仅考虑数据并行，没有考虑指令, 任务, 作业的并行

Handler(汉德勒)分类法

- 由Wolfgan Handler于1977年提出
- 根据并行度和流水线分类
- 把计算机硬件结构分成三个层次，并分别考虑它们的可并行性和流水处理程度
  - 程序级k：程序控制部件(FCU)的个数
  - 操作级d：算术逻辑部件(ALU)或处理部件(PU)的个数
  - 逻辑级w：每个算术逻辑部件包含的逻辑线路(ELC)的套数

- 表示方法：

  t(系统型号)=(k，d，w)

- 例如：t(EDVAC)=(1，1，1)

  t(Pentium)=(1，1，32)

  t(STARAN)=(1，8192，1)

  t(ILLIAC Ⅳ)=(1，64，64)

  t(Cmmp)=(16，1，16)

- 为了表示流水线，采用：

  t(系统型号)=(k×k'，d×d'，w×w')

  其中：k' 表示宏流水线中程序控制部件的个数

  d' 表示流水线中算术逻辑部件的个数

  w' 表示流水线中基本逻辑线路的套数

- 例如：Cray1有1个CPU，12个相当于ALU或PE的处理部件，最多8级流水线，字长为64位，可以实现 1～14位流水线。

表示为：t(Cray1)=(1，12×8，64(1～14))

又例如：t(PEPE)=(1×3，288，32)

t(TI ASC)=(1，4，64×8)

1.5.2 现代计算机系统结构研究方向

计算机系统结构的研究工作正向着多种高性能方向发展,当前的研究主要集中在如下9方面：

① RISC结构
② Client/Server结构
③ Mpp矢量模式（Vector Mode）
④ 并行处理（Massively Parallel Processing）
⑤ 集群计算结构
⑥ Network结构
⑦ 数据流计算机结构
⑧ 分布计算环境结构，如云计算
⑨ AI计算