计算机系统结构

Computer Systems and Architecture

李峰
fli@sdu.edu.cn
https://funglee.github.io

第1章 基本概念

1.1 计算机系统结构简介

1.2 计算机系统的设计技术

1.3 计算机系统的评价标准

1.4 计算机系统结构的发展

1.1 计算机系统结构简介

1.1.1 计算机系统层次结构

1.1.2 计算机系统结构定义

1.1.3 计算机组成和实现

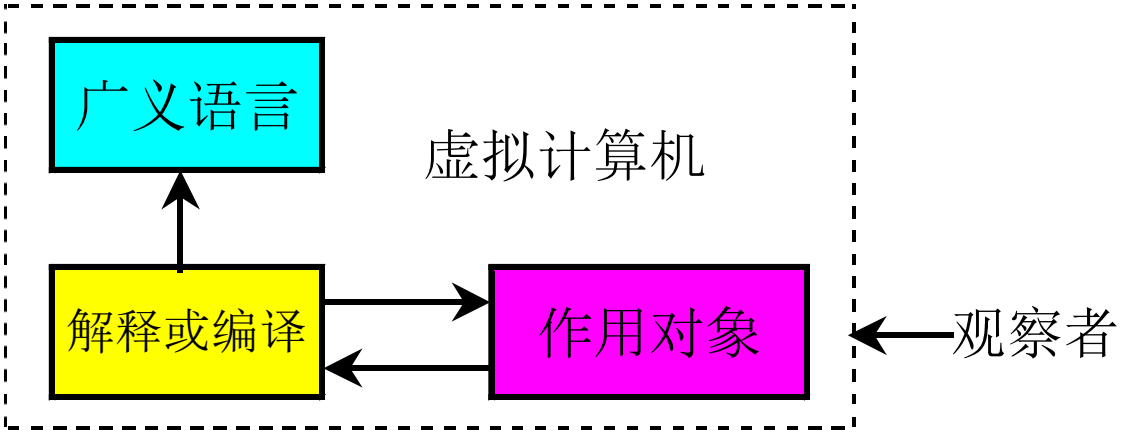
1.1.4 计算机系统结构的分类

1.1.1 计算机系统层次结构

1.虚拟计算机

- 定义：从不同角度所看到的计算机系统的属性是不同的
- 主要观察角度包括：
  - 应用程序员
  - 系统程序员
  - 硬件设计人员
- 对计算机系统的认识通常只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

计算机系统由硬件/器件和软件组成，按功能划分成多级层次结构

计算机系统的层次结构

- 计算机系统可分为7个层次
- 第3级至第6级由软件实现，称为虚拟机
- 从学科领域来划分：

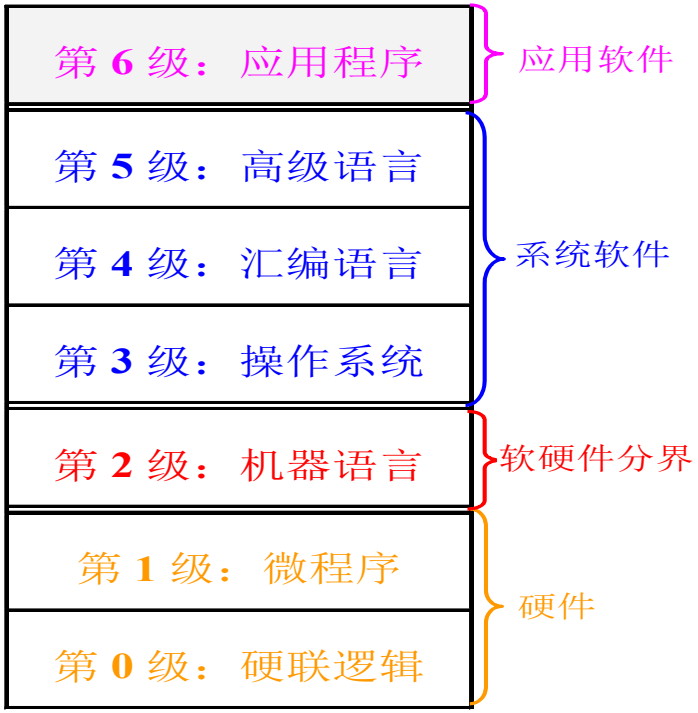
  第0级和第1级属于计算机组成原理

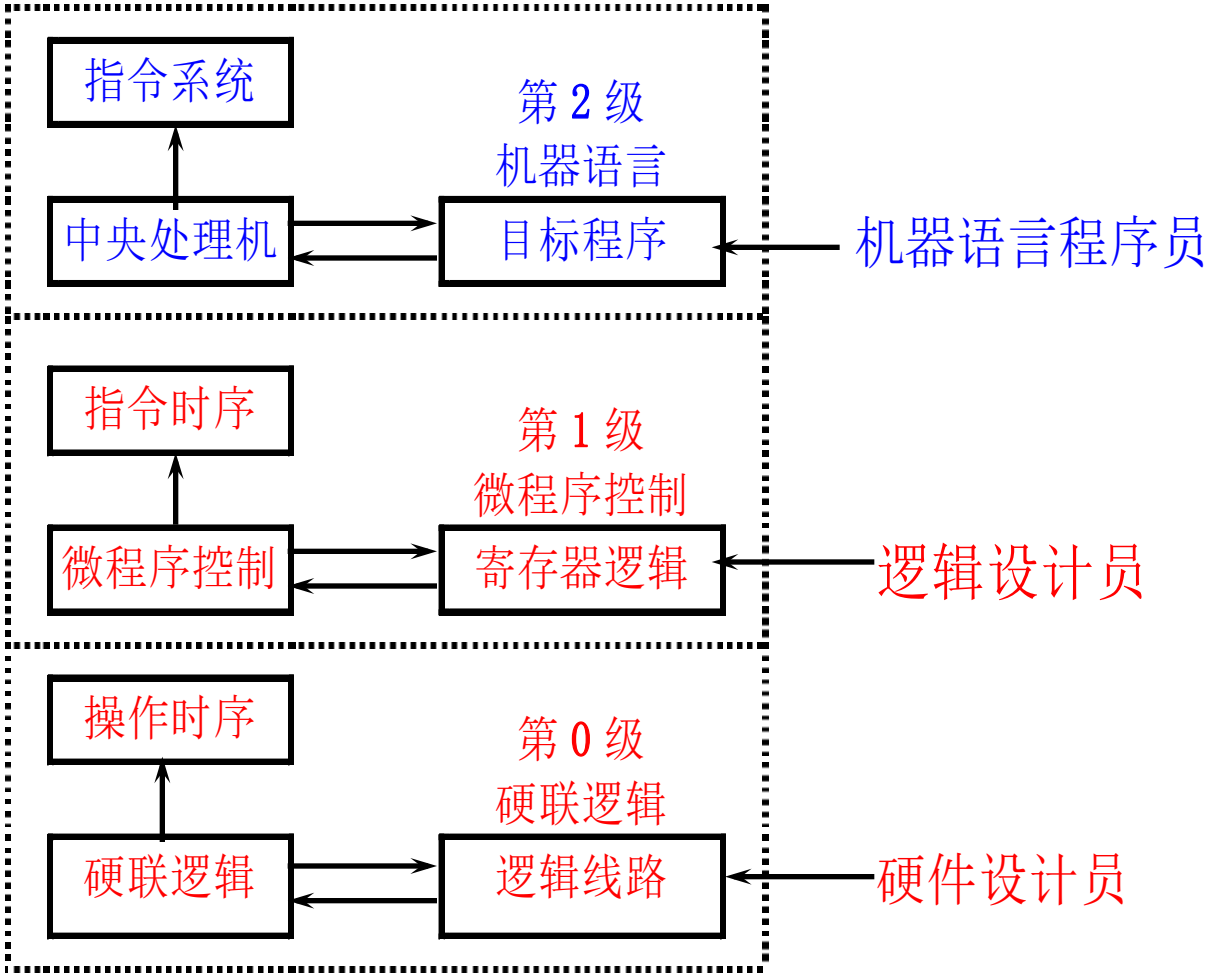
  第2级属于计算机系统结构

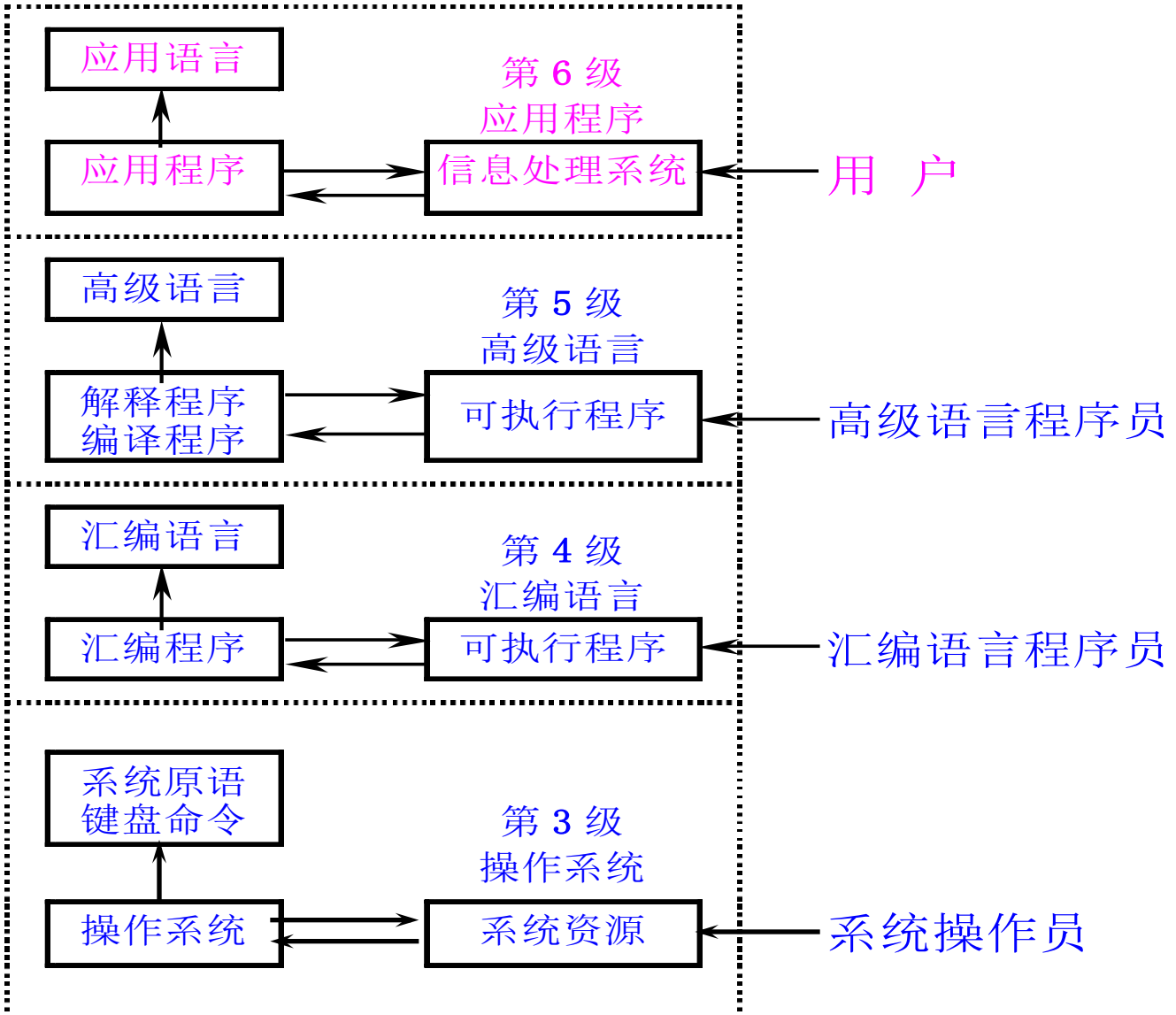
  第3至第5级属于系统软件

  第6级属于应用软件
- 它们之间有交叉

  例如：第3级必须依赖第4级和第5级来实现




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


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

透明性概念

- 定义：本来存在的事物或属性，从某种角度看似乎不存
- 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

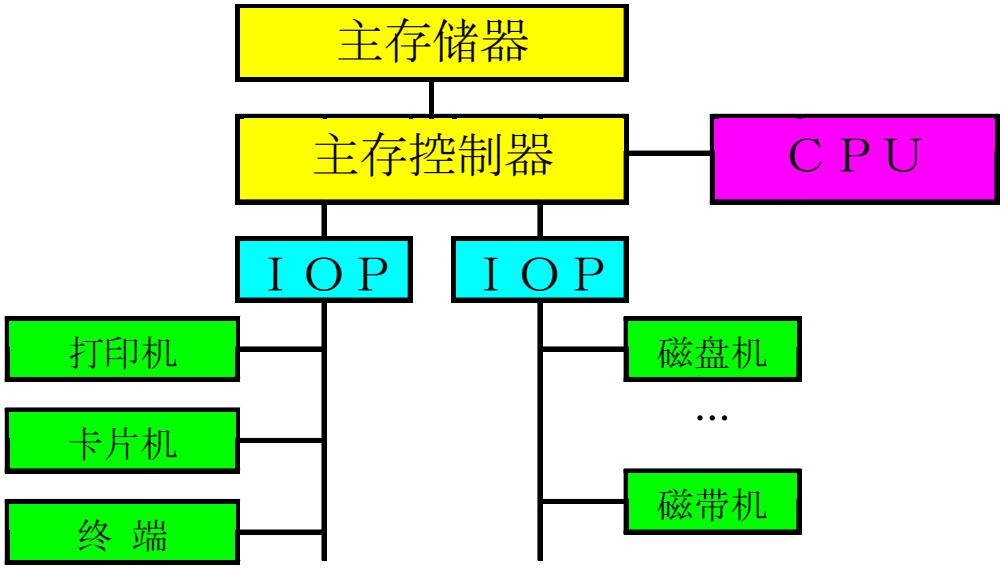
1.1.2 计算机系统结构的定义

1 计算机系统结构定义一

- **Amdahl于1964年在推出IBM360系列计算机时提出：**
- 程序设计者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：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到的

概念性结构

IBM360系列计算机的概念性结构



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；
- **寻址技术**：编址方式、寻址方式和定位方式等；
- **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**：寻址空间、虚拟存储器、Cache存储器等；
- **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。

2. 计算机系统结构定义二

- 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，他们的系统结构是相同的。
- 系列计算机概念：相同系统结构,不同组成和实现的一系列计算机系统。

1.1.3 计算机组成和实现

1. 计算机组成的主要任务

- 在计算机系统结构确定分配给硬件子系统的功能及其概念结构之后，研究各组成部分的内部构造和相互联系，以实现机器指令级的各种功能和特性

2. 计算机组成的研究方法

- 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

3. 计算机组成的主要研究内容

- 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。

4 计算机实现技术

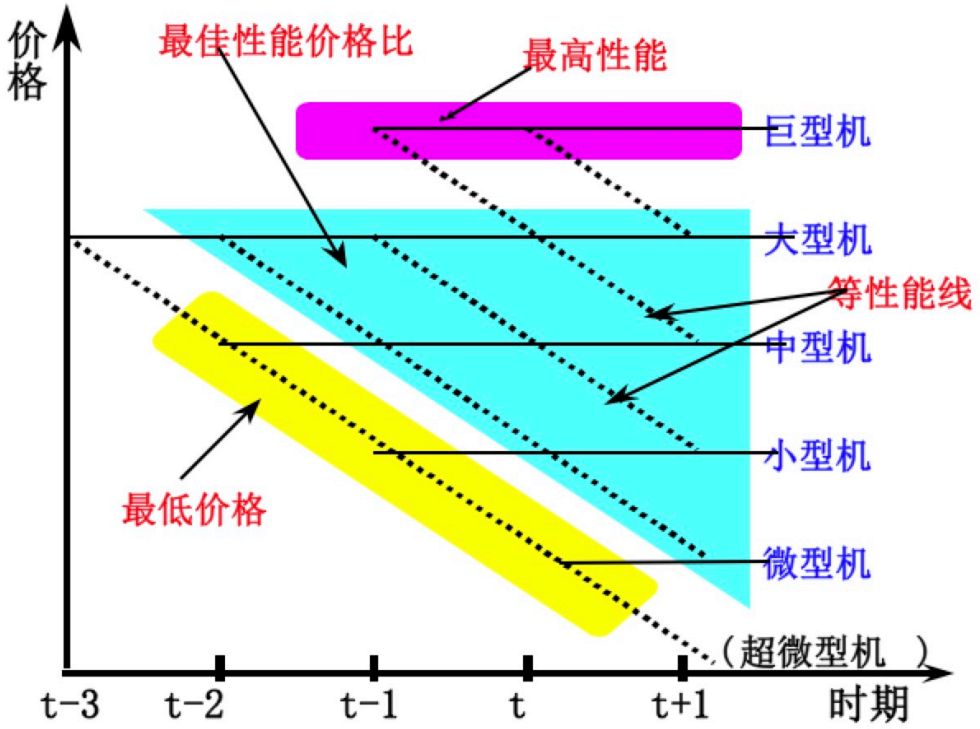
- 计算机实现是指计算机组成的物理实现主要包括：
  - 处理机、主存储器等部件的物理结构；
  - 器件的集成度和速度；
  - 专用器件的设计；
  - 器件、模块、插件、底版的划分与连接；
  - 信号传输技术；
  - 电源、冷却及装配技术，制造工艺及技术等。
- 随着技术、器件和应用的发展，三者之间的界限越来越模糊。

1.1.4 计算机系统的分类

1. 按处理机性能分类
2. 佛林（Flynn）分类法
3. 库克分类法
4. 冯泽云分类法
5. 汉德勒分类法

1 按处理机性能分类

- 按性能和价格的综合指标划分：巨型、大型、中型、小型、微型机

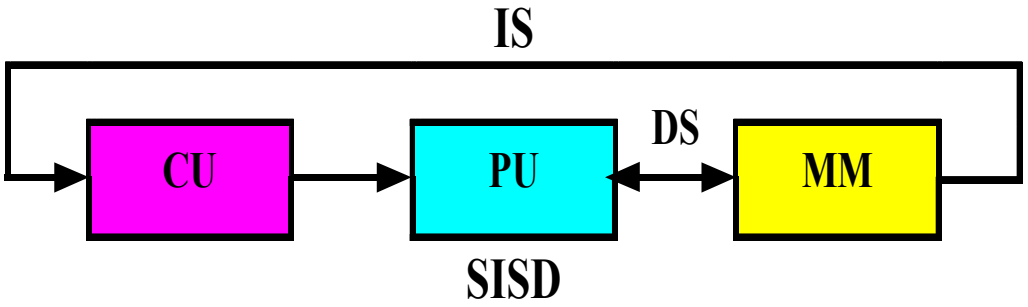


2 佛林（Flynn）分类法

- **1966年由Michael.J. Flynn 提出，按照指令流和数据流的多倍性特征进行分类**
  - 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
  1) 单指令流单数据流 **SISD(Single Instruction Single Datastream)**
  2) 单指令流多数据流 **SIMD(Single Instruction Multiple Datastream)**
  3) 多指令流单数据流 **MISD(Multiple Instruction Single Datastream)**
  4) 多指令流多数据流 **MIMS(Multiple Instruction Multiple Datastream)**

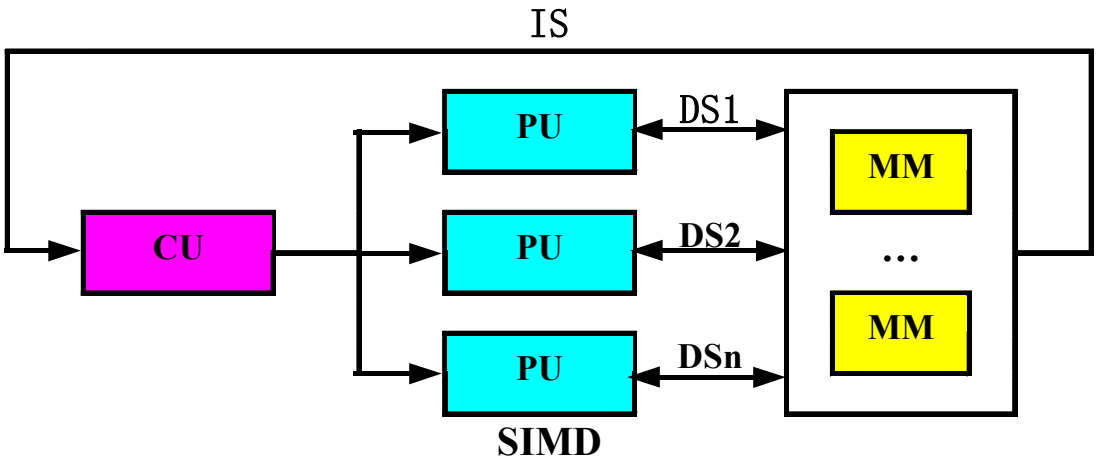
(1) SISD

- 典型顺序处理计算机



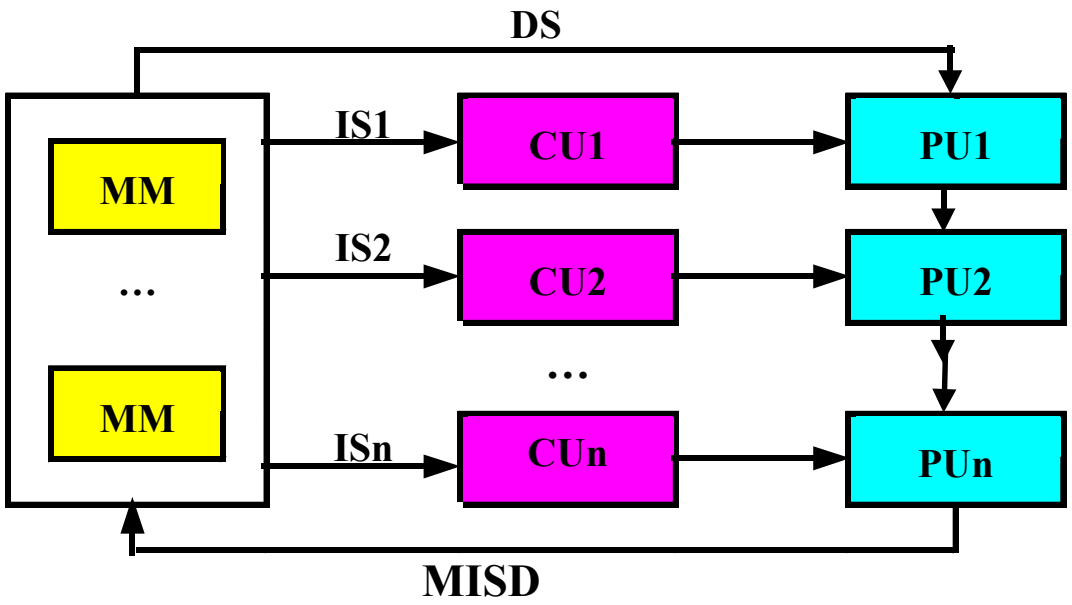
(2) SIMD：

- 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按一定方式互连，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。



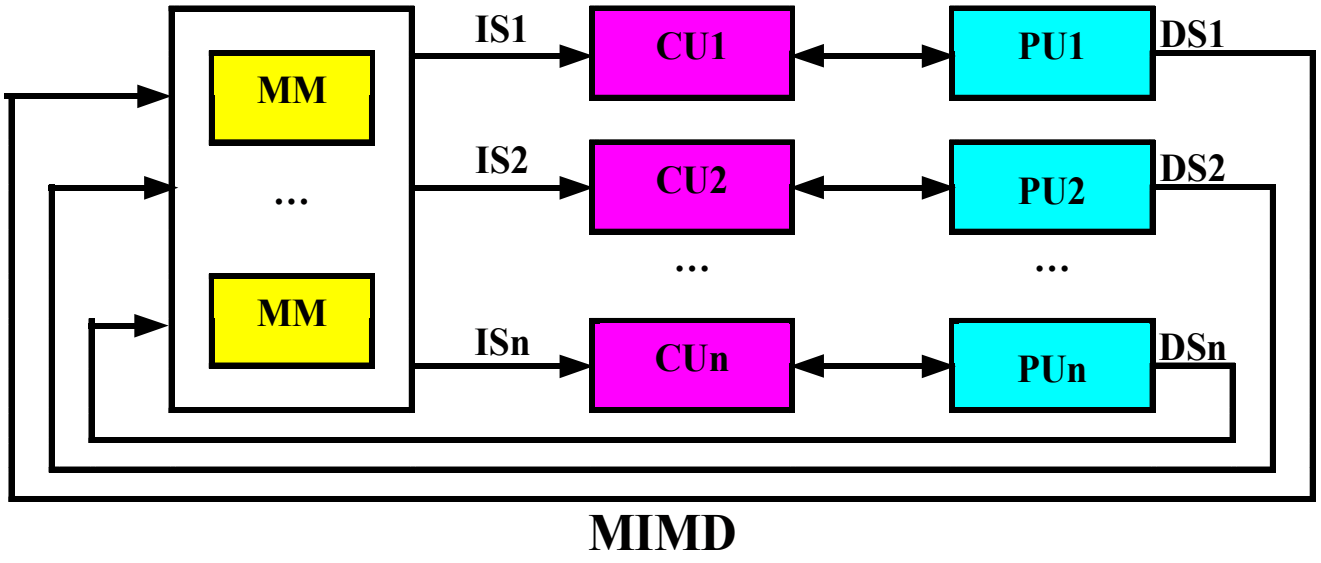
(3) MISD：

几条指令对同一个数据进行不同的处理



(4) MIMD 多处理机系统

- 紧密偶合：IBM3081、IBM3084、UNIVAC-1100/80
- 松散偶合：D-825, Cmmp, CRAY-2



3 库克分类法

- 1978年由 D. J. Kuck提出，按控制流和执行流分类
  1) 单指令流单执行流
     - SISE(Single Instruction Single Executionstream)
     - 典型的单处理机
  2) 单指令流多执行流
     - SIME(Single Instruction Multiple Executionstream)
     - 多功能部件处理机、相联处理机、向量处理机、流水线处理机、超流水线处理机、超标量处理机、SIMD并行处理机
  3) 多指令流单执行流
     - MISE，(Multiple Instruction Single Executionstream)
     - 多道程序系统
  4) 多指令流多执行流
     - MIME，(Multiple Instruction Multiple Executionstream)
     - 典型的多处理机

4 冯泽云分类法

- 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
- 最大并行度：单位时间内能处理的最大二进制位数
  - 字宽n：一个字中同时处理的二进位的位数
  - 位片宽度m：一个位片中能处理的字数
  - 例如：同时处理的字宽为n，位片宽为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位数为B_i，则T个时钟周期内的平均并行度为：

$$P_T = (\sum_{i=1}^{T} B_i \cdot t_i)/T$$

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

- 位串处理方式，每次只处理一个字中的一个位
- 串行计算机； $m=1,n=1$

(2)字并位串WPBS (Word Parallel and Bit Serial)

- 位（位片）处理方式，一次处理m个字中的1位
- 传统并行单处理机； $m>1$， $n=1$

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

- 字（字片）处理方式，每次处理一个字中的n位
- 并行计算机、MPP、相联计算机；$m=1, \mathrm{n}>1$

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

- 全并行处理方式，一次处理n个字，每个字为m位
- 全并行计算机；$m>1, \mathrm{n}>1$

5 汉德勒（Handler）分类法

- 由Wolfgan Handler于1977年提出，又称为ESC(Erlange Classification Scheme)分类法
- 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
  1) 程序级k：程序控制部件(PCU)的个数；
  2) 操作级d：算术逻辑部件(ALU)或处理部件（PU）的个数；
  3) 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)

1.2 计算机系统的设计技术

1.2.1 计算机系统的定量原理

1.2.2 计算机系统设计者的主要任务

1.2.3 计算机系统设计方法

1.2.1 计算机系统的定量原理

- 加快经常性事件的执行速度
  - **Amdahl**定律
  - **CPU**性能公式
  - 访问的局部性原理

- **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。**
- **在Amdahl定律中，加速比与两个因素有关：**

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

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

- 改进后整个任务的执行时间为：

$$T_n = T_0 \cdot (1 - F_e + \frac{F_e}{S_e})$$

其中：T_0为改进前的整个任务的执行时间。

- 改进后整个系统的加速比达到：

$$S_n = \frac{T_0}{T_n} = \frac{1}{(1 - F_e) + \frac{F_e}{S_e}}$$

其中：**Fe**表示可改进部分所占的百分比，

(1-Fe)表示不可改进部分所占的百分比，

Se表示改进后，可改进部分的加速比。

例：某部件的处理时间仅为整个运行时间的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？

解：由题意可知：Fe=0.4, Se=10，

根据Amdahl定律，加速比为：

$$S_n = \frac{1}{(1-0.4)+\frac{0.4}{10}} = \frac{1}{0.64} = 1.56$$

例：采用哪种实现技术来求浮点数平方根FPSQR的操作对系统的性能影响较大。假设FPSQR操作占整个测试程序执行时间的20%。一种实现方法是采用FPSQR硬件，使FPSQR操作的速度加快到10倍。另一种实现方法是使所有浮点数据指令的速度加快，使FP指令的速度加快到2倍，还假设FP指令占整个执行时间的50%。请比较这两种设计方案。

$$S_{FPSQR} = \frac{1}{(1-0.2)+0.2/10} = \frac{1}{0.82} = 1.22$$

$$S_{FP} = \frac{1}{(1-0.5)+0.5/2} = \frac{1}{0.75} = 1.33$$

CPU性能公式

- **CPU的程序执行时间T_{CPU}**
  - **程序执行的总指令条数IC（Instruction Counter）**
  - **平均每条指令的时钟周期数CPI（Cycles Per Instruction）**
  - **时钟主频f_c**

$$T_{CPU} = IC \times CPI \times \frac{1}{f_c}$$

- n种指令，每种指令的时钟周期数CPI_i，出现次数I_i

$$T_{CPU} = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{f_c}$$

- 平均指令时钟周期数

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{IC} = \sum_{i=1}^{n}(CPI_i \times \frac{I_i}{IC})$$

例：如果FP操作的比例为25%，FP操作的平均CPI=4.0，其它指令的平均CPI为1.33，FPSQR操作的比例为2%， FPSQR的CPI为20。假设有两种设计方案，分别把FPSQR操作的CPI和所有FP操作的CPI减为2。试利用CPU性能公式比较这两种设计方案哪一个更好(只改变CPI而时钟频率和指令条数保持不变)。

原系统的CPI=25%×4+75%×1.33=2

方案1（使FPSQR操作的CPI为2）系统

CPI=CPI原-2%×（20-2）=2-2%×18=1.64

方案2（提高所有FP指令的处理速度）系统

CPI=CPI原-25%×（4-2）=2-25%×2=1.5

显然，提高所有FP指令处理速度的方案要比提高FPSQR处理速度的方案要好。方案2的加速比=2/1.5= 1.33

程序访问的局部性规律

- 局部性分时间上的局部性和空间上的局部性
  - 时间局部性：程序中近期被访问的信息项很可能马上将被再次访问。
  - 空间局部性：指那些在访问地址上相邻近的信息项很可能会被一起访问。
- 存储器体系的构成就是以访问的局部性原理为基础的

1.2.2 计算机设计者的任务

- 确定用户对计算机系统的功能、价格和性能要求

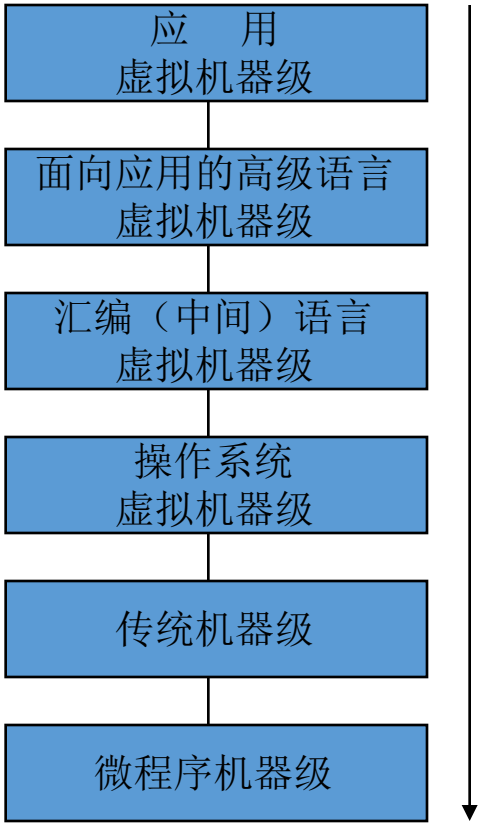
功能要求	应具备或支持的典型特性
应用领域 通用 科学计算 商用	**决定对计算机系统的性能要求** 对一系列任务有较好的性能 具有较好的浮点运算功能 支持COBOL、数据库、和事物处理等功能
软件兼容级别 编程语言级 目标代码级	**决定机器可以运行哪些软件** 设计者的自由度较大，但需要新的编译器 系统结构已经确定，无须投资软件
操作系统要求 地址空间大小 内存管理 安全保护	**为支持选定的操作系统所需要的特性** 非常重要的特性，可能限制程序的运行 页式或段式等管理方式，现代操作系统需要 操作系统和应用程序需要
标准 浮点 I/O总线 编程语言 网络	**市场上已有的，某种需要满足的标准** 格式和算法：IEEE、DEC、IBM等 I/O设备：VME、SCSI、PCI、光纤等 影响指令集：C、FORTAN、COBOL等 对不同网络的支持：内部互连网、Ethernet等

- 软硬件平衡
  - 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
- 设计出符合今后发展方向的系统结构
  - 把握硬件和软件的发展趋势

1.2.3 计算机系统设计方法

方法1：由上向下（Top-Down）

- **设计过程：**由上向下
  - 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



第一步：确定这一级的基本特性

第二步：设计或选择面向这种应用的高级语言

第三步：设计适于所用高级语言编译的中间语言

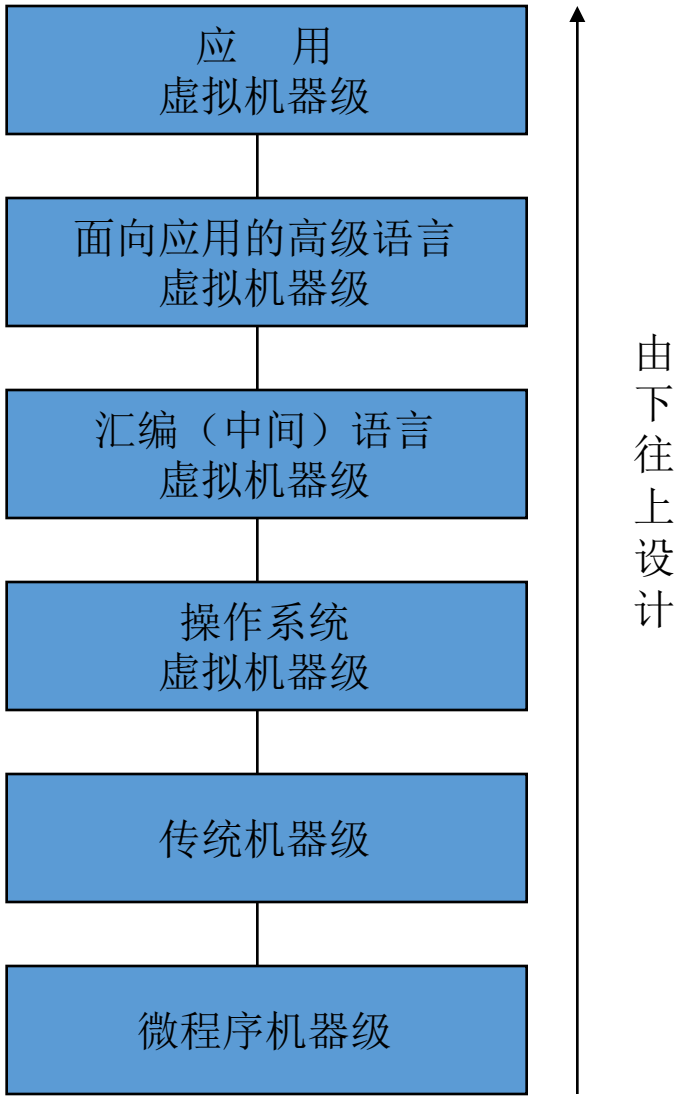
第四步：设计面向这种应用的操作系统

第五步：设计面向所用编译程序和操作系统的机器语言

第六步：设计面向机器语言的微指令机器硬件实现

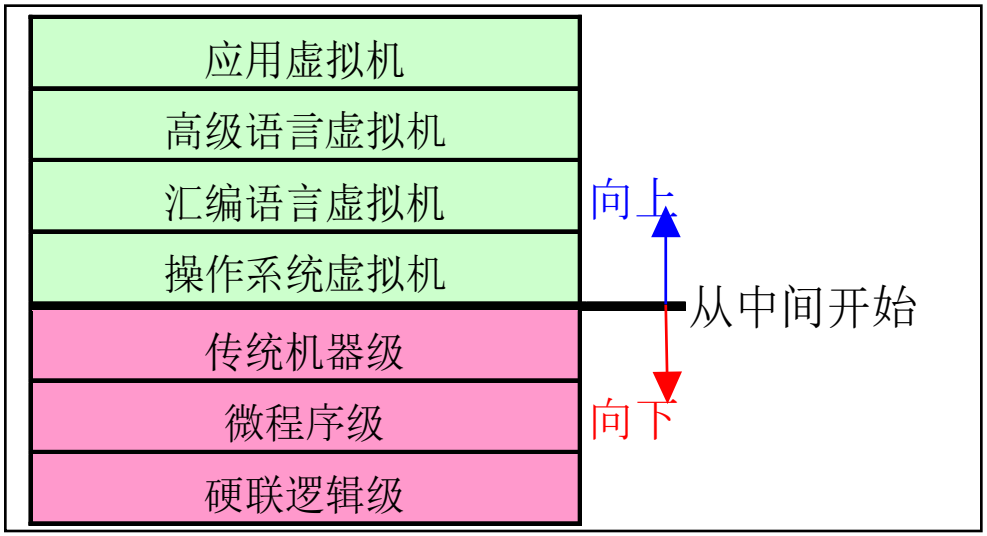
方法2：由下向上（Bottom-Up)

- **设计过程：**根据当时的器件水平，设计微程序机器级和传统机器级→根据不同的应用领域设计多种操作系统、汇编语言、高级语言编译器等→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（60～70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)

- **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。



1.3 计算机系统的评价标准

1.3.1 性能

- 主要标准：**MIPS**、**MFLOPS**、基准测试程序
- 性能比较

1.3.2 成本

1.3.1 性能

运算速度是表示处理机性能的主要指标。

1. MIPS (Million Instructions Per Second，每秒百万条指令数)

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

其中：Fz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：计算 PentiumIV 2GHz 处理机的指令执行速度。

解：由于 PentiumIV 2GHz 处理机的

IPC＝4（或CPI＝0.25），

Fz＝2000MHz

因此，$MIPS_{PentiumIV2G}$ ＝Fz×IPC＝2000×4

＝8000MIPS＝8GIPS

即**每秒钟80亿次**(平均每秒钟执行80亿条指令)

•主要优点：直观、方便。目前还经常使用

• 主要缺点：

1) MIPS依赖于指令集，用MIPS来比较指令集不同的机器的性能好坏是很不准确的
2) 在同一台机器上，由于指令使用频度差别很大，MIPS会因程序不同而变化
3) MIPS可能与性能相反
    - 例如，具有可选硬件浮点运算部件的机器，具有优化功能的编译器

2. MFLOPS（Million Floating Point Operations Per Second）

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

- 只能反映机器执行浮点操作的性能，并不能反映机器的整体性能
- 基于操作而非指令，可用来比较不同的机器
- 由于不同机器上的浮点运算集不同，MFLOPS并未完全可靠
- 依赖于操作类型

3. 用基准测试程序来测试评价机器的性能

- 采用实际的应用程序测试
  - 如：C语言的编译程序，CAD应用：Spice
- 采用核心程序测试
  - 从实际程序中抽出关键部分组合而成，例如循环部分
- 合成测试程序
  - 人为写的核心程序，规模小，结果预知
- 综合基准测试程序
  - 考虑了各种可能的操作和各种程序的比例，人为地平衡编制的基准测试程序

4 性能的比较

- 算术平均：$A_m = \frac{1}{n}\sum_{i=1}^{n} T_i$
- 调和平均(用速率来度量)：$H_m = n/\sum_{i=1}^{n}(1/R_i)$
  - $R_i = 1/T_i$
- 加权平均
  - 加权算术平均：$A_m = \sum_{i=1}^{n} W_i \cdot T_i$
  - 加权调和平均：$A_m = \left(\sum_{i=1}^{n} \frac{W_i}{R_i}\right)^{-1}$
  - 几何平均$\mathbf{G} = \sqrt[n]{\prod_{i=1}^{n} ETR_i}$
    - 其中，ETR（Execution Time Ratio）是程序标准化为参考机器后的时间

- **几何平均：$\mathbf{G} = \sqrt[n]{\prod_{i=1}^{n} \boldsymbol{ETR_i}}$**

  其中：n 指不同的程序，ETR(execution time ratio) i是第i个程序相对于参考机器正交化后的执行时间

- **几何平均速度与所参考的机器无关，有如下性质：**

$$\frac{X_i\text{的几何平均值}}{Y_i\text{的几何平均值}} = \left(\frac{X_i}{Y_i}\right)\text{的几何平均值}$$

几何平均值的比率等于比率的几何平均值

例 4：两个程序在三台机器上的执行时间

	机器 A	机器 B	机器 C
程序 P1（秒）	1	10	20
程序 P2（秒）	1000	100	20

运行程序P1时，A的速度是B的10倍；

运行程序P2时，B的速度是A的10倍；

运行程序P1时，A的速度是C的20倍；

运行程序P2时，C的速度是A的50倍；

运行程序P1时，B的速度是C的2倍；

运行程序P2时，C的速度是B的5倍。

算术平均：

- 程序P1和P2各执行1次，B的速度是A的9.1倍；
- 程序P1和P2各执行1次，C的速度是A的25倍；
- 程序P1和P2各执行1次，C的速度是B的2.75倍。

结论：

- 执行程序P1和P2相同次数，**机器A最慢，机器C最快**
- **算术平均速度：**三台机器的速度之比为：

A：B：C = 1：9.1：25

加权算术平均

- 加权算术平均**W1**三台机器的速度：**A < B < C**
- 加权算术平均**W2**三台机器的速度：**A < C < B**
- 加权算术平均**W3**三台机器的速度：**C < B < A**

	机器A	机器B	机器C
程序P1执行时间(s)	1	10	20
程序P2执行时间(s)	1000	100	20
加权W1 (0.5, 0.5)	**500.50**	**55.00**	**20.00**
加权W2 (0.909, 0.091)	**91.91**	**18.19**	**20.00**
加权W3 (0.999, 0.001)	**2.00**	**10.09**	**20.00**

几何平均

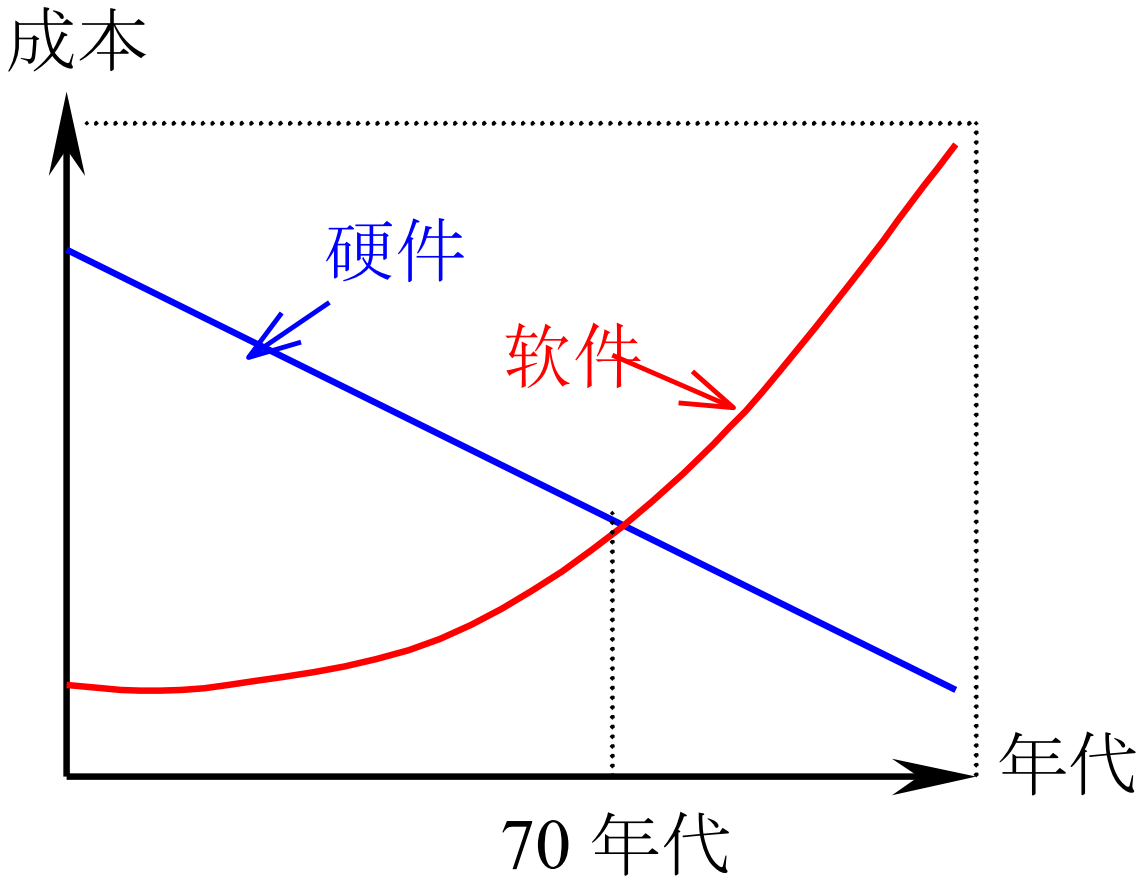
- 几何平均值与所参考的机器无关
  - 机器A与机器B的性能相同
  - 机器C的执行时间是机器A或机器B 的0.63倍。
- 执行程序P1和P2的总时间，机器A几乎是机器B的10倍。

执行时间正交化	与A正交			与B正交			与C正交		
	A	B	C	A	B	C	A	B	C
程序P1	1.0	10.0	20.0	0.1	1.0	2.0	0.05	0.5	1.0
程序P2	1.0	0.1	0.02	10.0	1.0	0.2	50.0	5.0	1.0
算术平均	1.0	5.05	10.01	5.05	1.0	1.1	25.03	2.75	1.0
几何平均	1.0	1.0	0.63	1.0	1.0	0.63	1.58	1.58	1.0
总时间比	1.0	0.11	0.04	9.1	1.0	0.36	25.03	2.75	1.0

1.2.4 价格标准

1. 价格与性能的关系：
   - 摩尔定理：速度每10年左右提高100倍，但价格基本维持不变
   - 用当前同样的价格，在5年之后能买到性能高出10倍的计算机
2. 硬件与软件的价格比例：
   - 硬件在整个计算机系统价格中所占的比例在下降，软件所占的比例在上升
   - 目前软件价格已经超过硬件价格

软件所占的成本越来越高

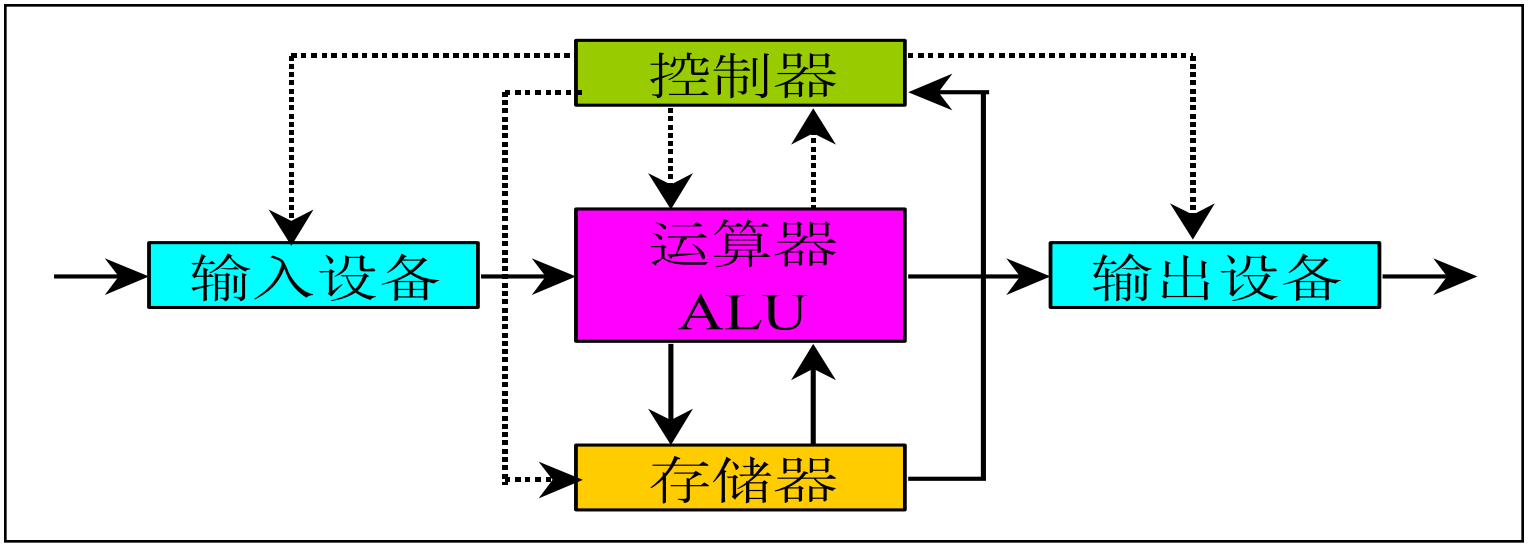


1.4 计算机系统的发展

1.4.1 冯·诺依曼结构
1.4.2 软件对系统结构的影响
1.4.3 价格对系统结构的影响 （课后阅读）
1.4.4 应用对系统结构的影响（课后阅读）
1.4.5 VLSI对系统结构的影响（课后阅读）
1.4.6 技术的发展对系统结构的影响（课后阅读）
1.4.7 算法和系统结构（课后阅读）

1.4.1 冯·诺依曼结构

Van Nenmann基本思想于1936年～1946年期间形成，由冯·诺依曼等人于**1946**年提出



1. 特点: 存储程序、运算器为中心、集中控制

- 存储器是字长固定的、顺序线性编址的一维结构，每个地址是唯一定义的。
- 由指令形式的低级机器语言驱动。
- 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，程序分支由转移指令实现。
- 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经运算器。
- 运算器、存储器、输入输出设备的操作以及它们之间的联系都由控制器集中控制。

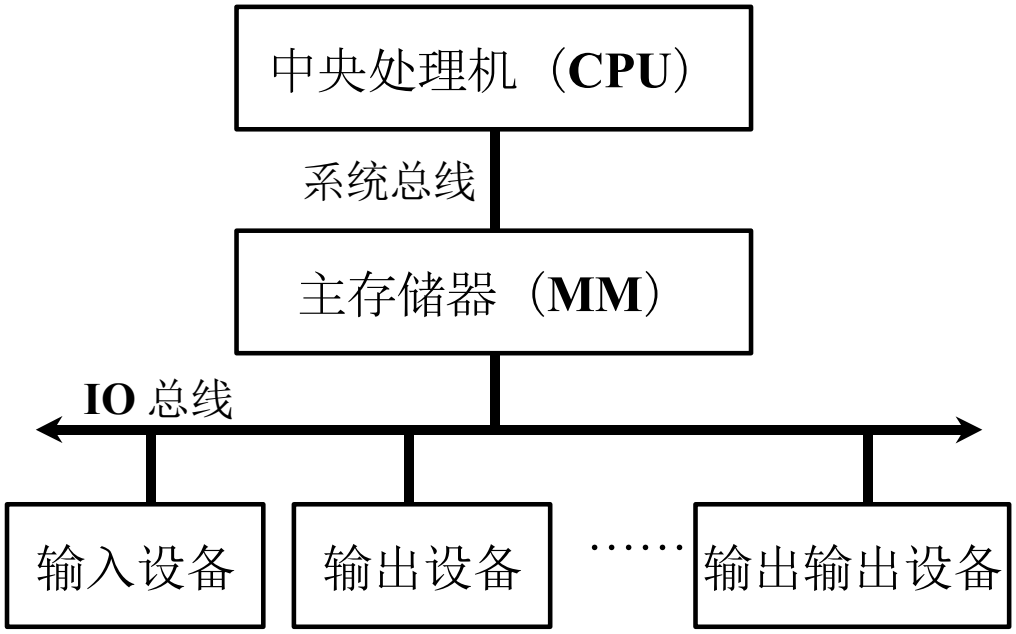
2. 现代处理机对冯·诺依曼结构的改进

不变的：存储程序

改变的：存储器为中心, 总线结构, 分散控制

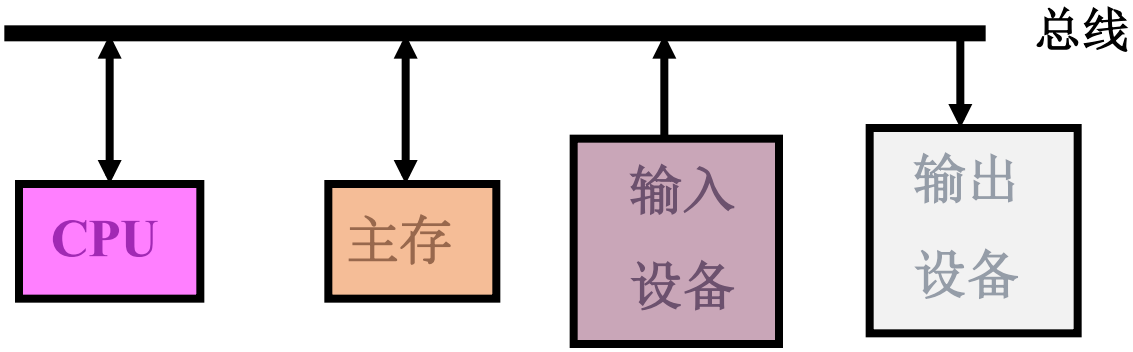
- 从基于串行算法变为适应并行算法，出现了向量计算机，并行计算机、多处理机等
- 高级语言与机器语言的语义距离缩小，从而出现了面向高级语言机器和直接执行高级语言机器
- 硬件子系统与操作系统和数据库管理系统软件相适应，出现了面向操作系统机器和数据库计算机
- 计算机系统结构从传统的指令驱动型改变为数据驱动型和需求驱动型，从而出现了数据流机器和归约机
- 为了适应特定应用环境而出现了各种专用计算机，如FFT变换机、过程控制计算机
- 为获得高可靠性而研制容错计算机
- 功能分散化、专业化，出现了各种分布计算机、外围处理机、通信处理机等
- 出现了与大规模、超大规模集成电路相适应的计算机系统结构
- 出现了处理非数值化信息的智能计算机

存储器为中心、分散控制



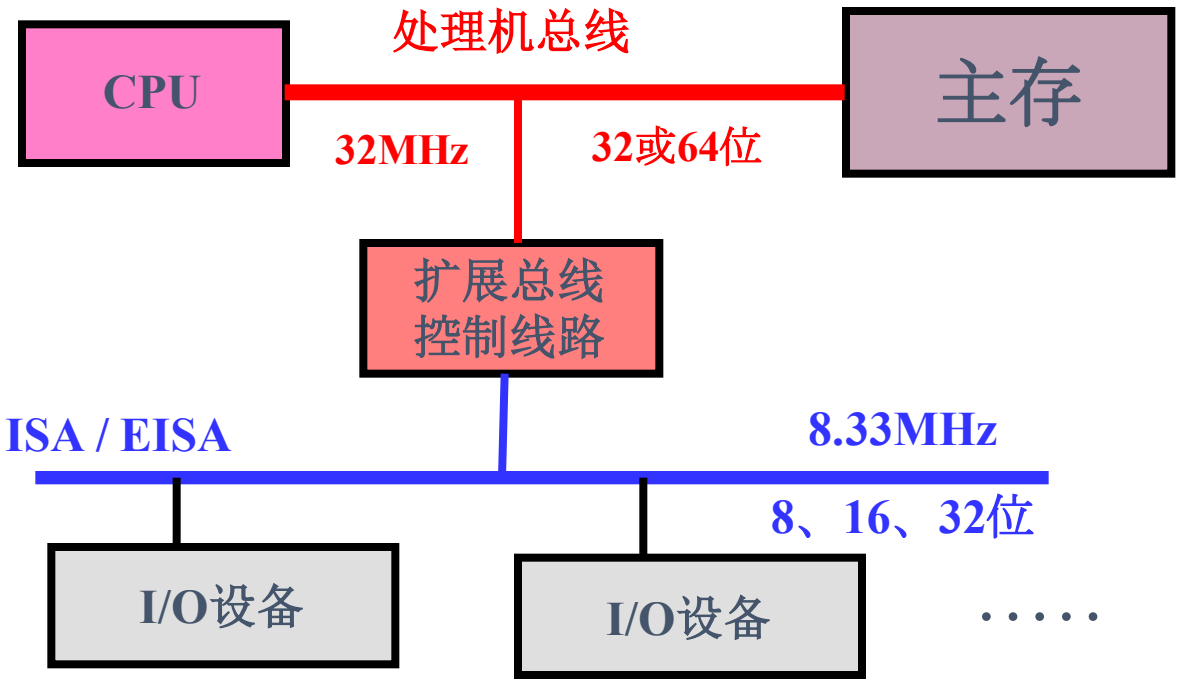
总线结构，分散控制

- 总线： 连接计算机各功能部件的连线和管理信息传输规则的逻辑电路称为总线。
- 特点：在任何时刻，只能有一个部件向总线上发送信息，可以有多个部件同时接收信息。
- 组成： 数据总线、地址总线、控制总线。

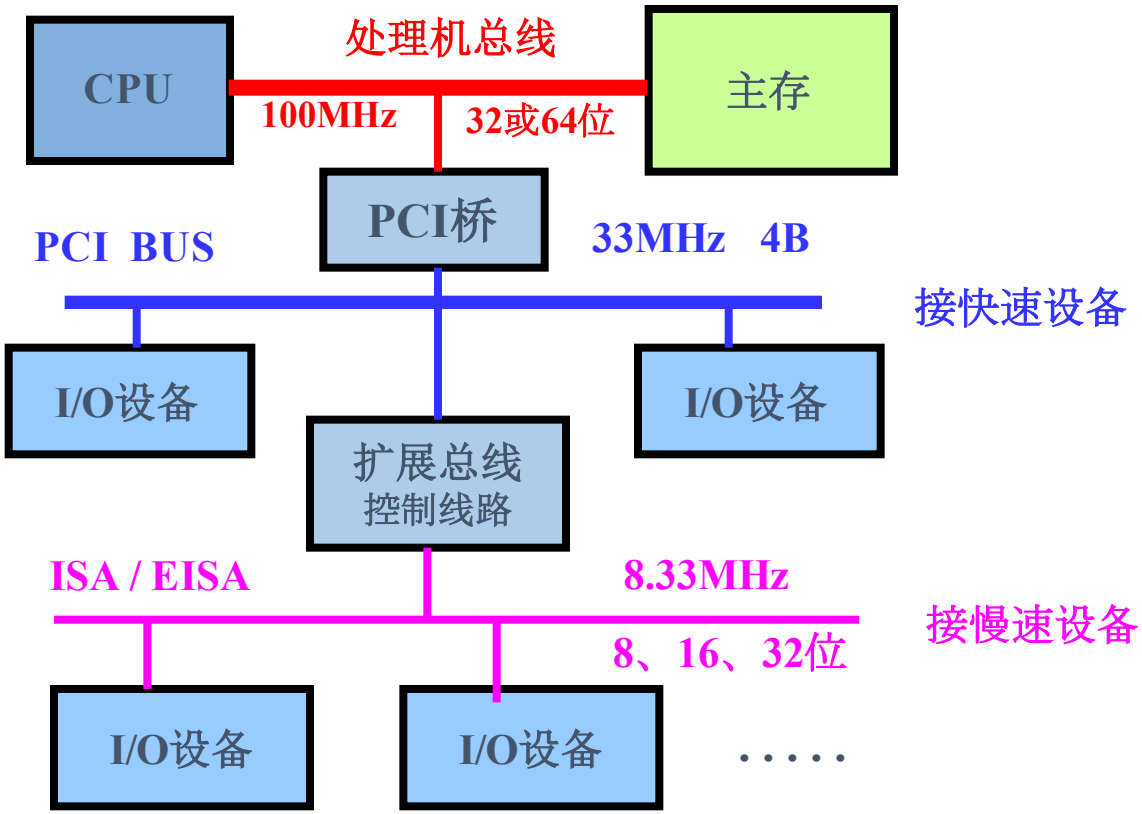


单总线结构

双总线结构


处理机总线
CPU
主存
32MHz
32或64位
扩展总线
控制线路
ISA / EISA
8.33MHz
8、16、32位
I/O设备
I/O设备
. . . . .

三总线结构


CPU
处理机总线
主存
100MHz
32或64位
PCI桥
PCI BUS
33MHz 4B
接快速设备
I/O设备
I/O设备
扩展总线
控制线路
ISA / EISA
8.33MHz
接慢速设备
8、16、32位
I/O设备
I/O设备
.....

1.4.2 软件对系统结构的影响

软件相对于硬件的成本越来越贵，已积累了大量成熟的系统软件和应用软件。人们希望在新的计算机系统结构出台后，软件能够继续在新的系统结构上运行，即要求软件具有可兼容性（或可移植性）

- 兼容种类
  1) 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。
  2) 向前兼容：
  3) 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。
  4) 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

方法一：系列机方法

- 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

- 相同的指令系统，采用顺序执行、重迭、流水和并行处理方式
- 相同的32位字长，数据通路宽度为8位、16位、32位、64位。
- 如PC系列机有：
  - 不同主频：4.7MHz，500MHz，1GHz，2.4GHz, 3GHz, …
  - 不同扩展：Pentium、Pentium Pro、Pentium MMX、Pentium SSE、Pentium SSE2
  - 不同Cache：Pentium、Celeron、Xeon
  - 不同字长：8位、16位、32位、64位

- 采用系列机方法的主要优点：

  1) 系列机之间软件兼容，可移植性好

  2) 插件、接口等相互兼容

  3) 便于实现机间通信

  4) 便于维修、培训

  5) 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

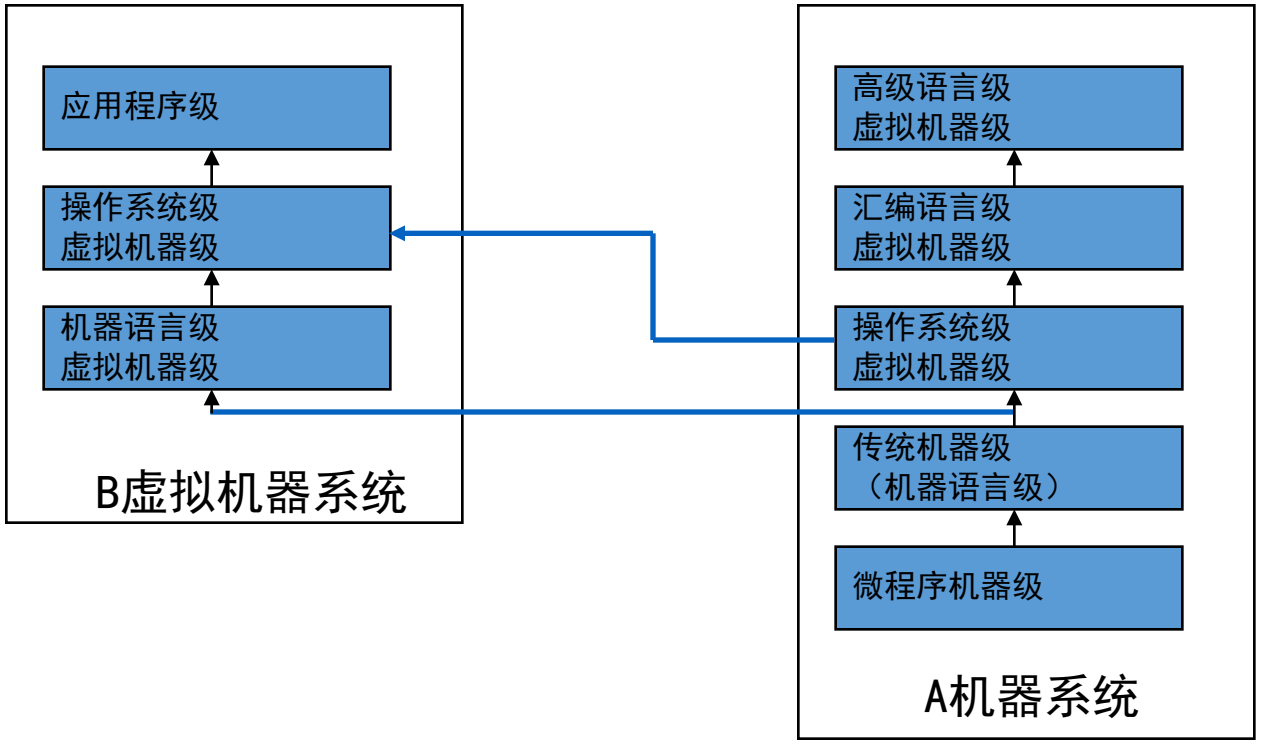
方法二：模拟和仿真

- 从指令系统来看，是指要在一种机器的系统结构上实现另一种机器的指令系统

模拟 Simulation

- 用机器语言程序解释实现软件移植的方法。
  - 进行模拟工作的A机称为宿主机（**Host Machine**）
  - 被模拟的B机称为虚拟机（**Virtual Machine**）
  - 所有为各种模拟所编制的解释程序通称为模拟程序，编制非常复杂和费时
  - 只适合于移植运行时间短，使用次数少，而且在时间关系上没有约束和限制的软件；

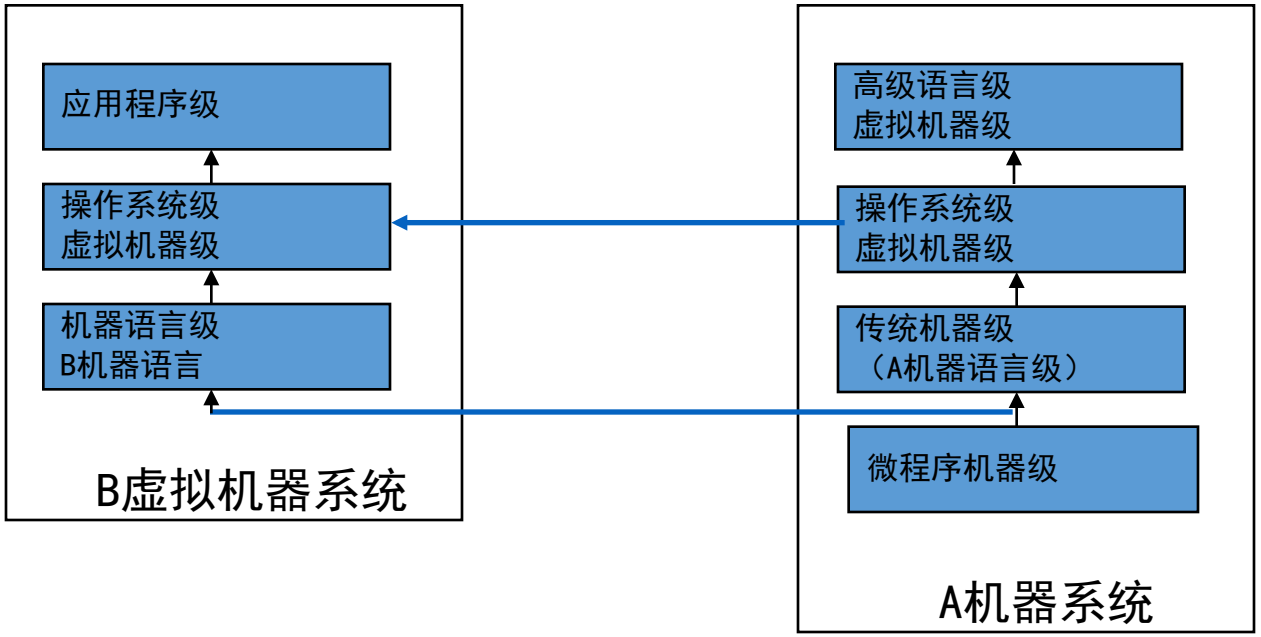
用模拟方法实现应用软件的移植



仿真 Emulation

- 用微程序直接解释另一种机器指令的方法。
  - 进行仿真工作的**A**机称为宿主机
  - 被仿真的**B**机称为目标机（**Target Machine**）
  - 所有为仿真所编制的解释微程序通称为仿真微程序；

用仿真方法实现应用软件的移植



仿真与模拟的区别

- 解释用的语言不同
- 解释程序所存的位置不同：仿真存在控制寄存器，模拟存在主存中
- 说明：
  - 模拟适用于运行时间不长、使用次数不多的程序
  - 仿真提高速度，但难以仿真存储系统、I/O系统，只能适用于系统结构差异不大的机器间；
  - 在开发系统中，两种方法共用

模拟与仿真的比较

项目	模拟	仿真
优点	可实现结构差别大的机器间软件移植	速度较快
缺点	运行速度低，实时性差，模拟程序复杂	机器结构差别大时，仿真困难
适用场合	运行时间短，使用次数少，无时间关系约束的软件	频繁使用且易于仿真的指令

方法三：统一高级语言

- 实现方法：采用同一种不依赖于任何具体机器的高级语言编写系统软件和应用软件
- 困难：至今还没有这样一种高级语言，短期内很难实现。C、Ada、Java、……

三种方法比较：

- 采用统一高级语言最好，是努力的目标
- 系列机是暂时性方法，也是目前最好的方法
- 仿真的速度低，芯片设计的负担重，目前用于同一系列机内的兼容，1/10～1/2的芯片面积用于仿真

本章重点：

1. 计算机系统的层次结构
2. 计算机系统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