第一章　计算机系统结构的基础知识

一、系统结构的相关概念

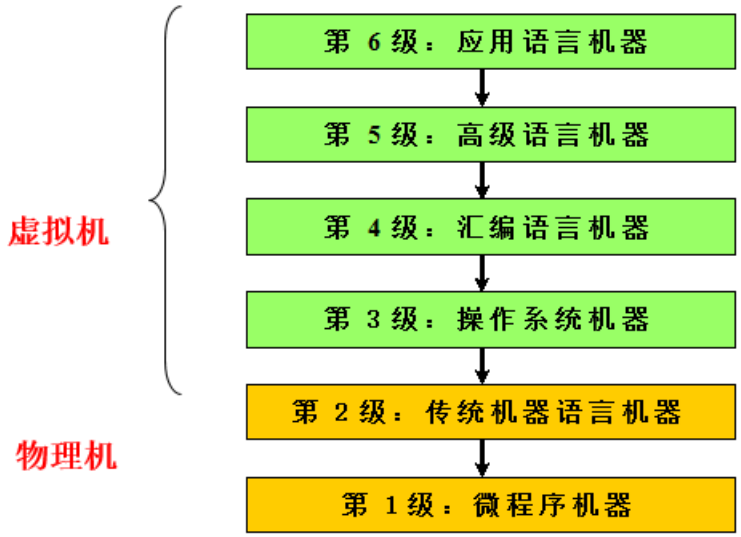
- **计算机系统的层次结构**

  第 0 层：硬件
  第 1 层：微程序（固件）
  第 2 层：机器语言机器
  第 3 层：操作系统机器
  第 4 层：汇编语言机器
  第 5 层：高级语言机器
  第 6 层：应用语言机器

1.1.1 计算机系统的层次结构

1. 计算机系统＝硬件/固件＋软件
2. 计算机语言从低级向高级发展

   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。
3. 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



- **计算机系统结构基本概念(广义机器、透明性、编译)**

  广义机器定义：执行和存储程序的算法和数据结构的集合体。

翻译：转换程序把高一级机器上的程序转换为低一级机器上等效的程序，然后在低一级机器上运行，实现程序的功能。

特点：速度快， 占用的存储空间较大（ L4 级以上）。

解释：把高一级机器上程序的每一条语句，转换为低一级机器上的一段等效程序并执行。执行完后，去高一级机器再取下一条语句或指令解释执行，直到解释执行完整个程序 。

特点：速度慢， 占用的存储空间较小（ L3 级以下）

透明性：在计算机技术中，一种本来存在的事物或属性，但从某种角度看又好像不存在的概念称为透明性。通常，低层机器的属性对高层机器程序员往往是透明的。

一般在一个计算机系统中，底层机器的属性对于高层机器的程序员来说是透明的。

- **计算机系统结构、组织和实现的定义**

计算机系统结构的实质：

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。

1.1.3 计算机组成和计算机实现

1. 计算机系统结构：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：计算机系统结构的逻辑实现

   - 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。
   - 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。

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，模块、插件、底板的划分与连接，信号传输，电源、冷却及整机装配技术等。
   - 着眼于：器件技术（起主导作用）、微组装技术。

具有相同系统结构的计算机可以采用不同的计算机组成。

同一种计算机组成又可以采用多种不同的计算机实现。

举例：乘法指令 、主存容量与编址方式

计算机组成的设计任务

- 数据通路：数据总线的宽度
- 专用部件设置：乘除法部件、浮点运算部件等
- 各种操作对部件的共享程度

- 功能部件的并行度：控制和处理的顺序方式、流水方式和分布处理方式
- 控制机构的组成方式：硬接线控制、微程序控制；单处理机、多理机或功能分布处理
- 缓冲和排队：缓冲技术用于平衡各部件之间的速度差异；排队技术用于安排等待处理事件的顺序关系，如随机、队列、堆栈等
- 容错技术：提高系统的可信性、可靠性、可用性。
- 预测和评估：优化性能

计算机实现的设计内容

–专用芯片（ASIC）的设计
–处理机、Cache 和主存的物理结构
–器件、模块、插件和底板的逻辑划分和连接
–信号传输
–电源与冷却
–微组装和整机组装技术

重点
–VLSI 器件的设计、测试与验证
–微组装技术

实例：

① 机器指令集的确定——计算机体系结构。

② 指令实现方式，如取指令、取操作数、运算、送结果等具体操作及其排序方式——计算机组成

③ 实现指令集中所有指令功能的具体电路、器件的设计、装配技术——计算机实现

① 确定是否有乘法指令——属计算机体系结构

② 乘法指令是用专门的乘法器实现，还是经加法器用重复的相加和右移操作来实现——属计算机组成。

③ 乘法器、加法器的物理实现，如器件的选定(器件集成度、类型、数量、价格)及所用微组装技术等——计算机实现

- **计算机系统分类方法\ Flynn 分类法**

1.1.4 计算机系统结构的分类

常见的计算机系统结构分类法有3种：

Flynn分类法、冯氏分类法和Handler分类法

1. Flynn分类法

Flynn分类法.

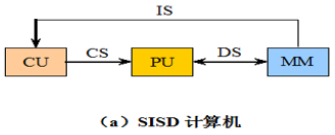
- 按照指令流和数据流的多倍性进行分类。
  - 指令流：计算机执行的指令序列
  - 数据流：由指令流调用的数据序列
  - 多倍性：在系统最受限的部件上，同时处于同一执行阶段的指令或数据的最大数目。

同时处于同一执行阶段的指令或数据的最大数目.

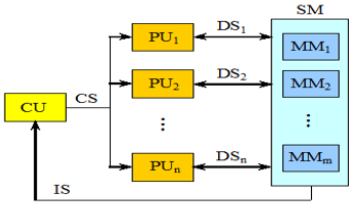
单指令单数据
单指令多数据
多指令单数据
多指令多数据.

按指令流和数据流的组织方式分类(1966 年)

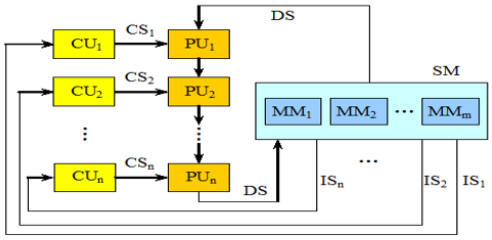
- 单指令流单数据流（SISD）计算机系统
- 单指令流多数据流（SIMD）计算机系统
- 多指令流单数据流（MISD）计算机系统
- 多指令流多数据流（MIMD）计算机系统



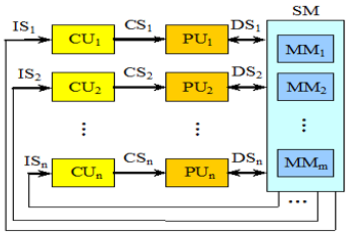
（a）SISD 计算机



（b）SIMD 计算机



（c）MISD 计算机



（d）MIMD 计算机

- 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

Handler 根据并行度和流水线提出了另一种分类法。 把计算机硬件结构分成三个层次
处理控制器 PCU
算术逻辑部件 ALU(或运算部件 PE)
运算部件所包含的位级电路 BLC

分别考虑各层并行度和流水处理程度，将计算机用 3 对整数表示：
T(体系型号)=〈k×k′,d×d′,w×w′〉

- k ——处理控制器 PCU 的数目；
- k′——可组成流水线的控制部件的 P 数目；
- d ——每个 PCU 所控制的 ALU(或 PE)数目；
- d′——可组成流水线的 ALU 部件的数目；
- w ——ALU 或 PE 的字长；
- w′——在所有 ALU 或一个 PE 中的流水段数目。

【例】 CDC6600 计算机体系有 1 个 CPU，它的运算器 ALU 有 10 个功能部件，所有的功能部件可连成一条流水线，字长 60 位。则 CDC6600 体系可描述为：
T(CDC6600)=T〈1,1×10,60〉

〈1, 1×10, 60〉

【例】 CRAY-1 计算机有 1 个 CPU，12 个相当于 ALU 或 PE 的处理部件，最多可以实现 8 级流水线。字长为 64 位，可以实现 1～14 位流水线处理。所以 CRAY-1 的体系结构可表示为:

T(CRAY-1)=〈1，12×8,64×(1～14)〉

注意:如 1 对参数的第二个元素值为 1，则省略不写。

冯氏分类法：用最大并行度对计算机分类

最大并行度 Pm 定义：在单位时间内能够处理的最大的二进制位数。

用最大并行度对计算机进行分类，得出 4 种不同的计算机结构:

1 字串位串 WSBS(Word Serial and Bit Serial)　　n = m = 1。
　　纯串行处理机

2　字串位并 WSBP(Word Serial and Bit Parallel)　　n = 1,m > 1
　　传统单处理机

3 字并位串 WPBS(Word Parallel and Bit Serial)　　n > 1,m = 1。
　　同时处理多个字的 同一位。

4　字并位并 WPBP(Word Parallel and Bit Parallel)　　n，m > 1。
　　同时处理多个字的 多个位。

二、基本原理和性能公式

- **大概率事件优先**

  大概率事件优先原理（经常性事件原则）：优先加速使用频率高的部件。

  对大概率事件（经常性事件），赋予优先处理权和资源使用权，能获得明显的系统性能。

  优先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。

- **Amdahl 定律**

  2. Amdahl 定律:加快某部件执行速度所获得的系统性能(加速比)，与该部件在系统中的总执行时间的比例（重要性）有关。

  系统性能加速比

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

  系统性能加速比与该部件在系统中的总执行时间有关。

  加速比依赖于两个因素

  可改进比例（Fe）：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。

  它总是小于等于 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（Se）：可改进部分(改进以后)性能提高的倍数。即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。

– 改进后程序的总执行时间T_n

$$T_n = T_0\left(1 - Fe + \frac{Fe}{Se}\right)$$

- T_0：改进前整个程序的执行时间
- $1-F_e$：不可改进比例

系统加速比S_n 为改进前与改进后总执行时间之比:

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

- **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。

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

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。

- **CPU 性能计算**

- **加速比公式应用**

拓展的 Amdahl 公式：

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

三、性能评价标准

- **性能指标（CPU 时间， CPI， MIPS,MFLOPS)**

CPU时间 = 时钟周期数 × 周期时间. 十

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 t(ns)是系统率时钟频率 f (MHz)的倒数。即

t=1/f

CPI 每条指令平均周期

指令的平均时钟周期数 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

周期数/指令数.

程序执行的 CPU 时间可以写成

CPU 时间 = IC ×CPI × t

CPU = CPI × IC × t.

每秒（百万条）指令数 MIPS(million instrctions per second)

MIPS=IN/(TE×1000000)

=IN/(IN×CPI×t×1000000)

MIPS = IN/TE × 1000000

=f/(CPI×1000000)

= f/CPI × 1000000

IN----程序的指令总数，TE--- 执行程序的时间，t 为时钟周期，f 为频率，CPI ----程序中指令的平均时钟周期数。

每秒（百万次）浮点运算 MFLOPS(million floating point operations per second)

三个评判标准：

1、 总执行时间

2、 调和平均值

3、 几何平均值