计算机
体系结构

计算机体系结构是计算机相关专业本科生必修的专业课。也称为计算机系统结构。

本课程先修的课程：计算机组成原理。

本课程是一门从计算机组成和结构的角度上学习和领会计算机系统的课程。

计算机系统是由软件和硬件组合的一个复杂的综合体。本课程主要研究如何对计算机系统软件和硬件的功能进行更合理的分配；研究如何更好、更合理地实现分配给硬件的那部分功能，使系统有尽可能高的性能价格比。

本课程学习的内容是计算机系统结构设计、硬件设计、高层次应用系统开发和系统软件开发所必须了解和掌握的基本知识。

通过本课程的学习，能进一步树立计算机系统的整体概念，熟悉有关计算机系统结构的概念、原理，了解常用的基本结构，领会结构设计的思想和方法，提高分析和解决问题的能力。

本课程的地位和重要性

1. 计算机系统结构是计算机科学与技术一级学科中的三个二级学科之一。

2. 计算机系统结构是国家同等学历申请硕士学位考试科目之一。

3. 硕士研究生入学考试“计算机组成原理”部分的试题中有体系结构方面的内容。

教材与参考书

1. 蒋本珊、马忠梅、郑宏．计算机体系结构简明教程．北京：清华大学出版社，2015
2. 郑纬民、汤志忠．计算机系统结构．北京：清华大学出版社，2001.2
3. 李学干．计算机系统的体系结构（第5版）．西安：西安电子科技大学出版社，2011
4. John L.Hennessy等．计算机系统结构——量化研究方法．北京：机械工业出版社（英文版），电子工业出版社（中文版），2004
5. 张晨曦．计算机系统结构教程．北京：清华大学出版社，2009
6. 李学干．《计算机系统结构》学习指导与题解．西安：西安电子科技大学出版社，2001

第1章 计算机系统结构的基本概念

1.1 计算机系统的多级层次结构

1.2 计算机系统结构、组成与实现

1.3 软硬取舍与计算机系统的设计思路

1.4 计算机设计的量化准则

1.5 对系统结构的影响因素

1.6 系统结构中的并行性

1.7 计算机系统分类

- 计算机系统＝软件＋硬件/固件
- 可以从多个角度考察计算机系统的结构
- 一种观点：从使用语言的角度，可以将计算机系统按功能划分为多级层次结构

图 1.1 计算机系统的多级层次结构

从学科领域来划分

- 第0和第1级属于计算机组成与系统结构
- 第2至第4级是系统软件
- 第5级是应用软件

它们之间仍有交叉

- 第1级涉及汇编语言程序设计的内容
- 第2级与计算机系统结构密切相关

在特殊的计算机系统中，有些级别可能不存在。

虚拟机概念

在计算机系统的多层次结构中，除第0、1级外，上面四级均为虚拟机。

虚拟计算机是指这个计算机只对该级的观察者存在。对某一层次的观察者来说，它只能是通过该层次的语言来了解和使用计算机，至于下层是如何工作和实现的就不必关心了。简而言之，虚拟机即是由软件实现的机器。

从使用语言的角度上，将计算机系统看成按功能划分的多级层次结构。

机器语言　　低级

汇编语言

高级语言

应用语言　　高级

特点：后者以前者为基础；比前者功能强、使用更方便。

计算机只能直接识别和执行机器语言。

汇编语言是一种符号式程序设计语言。

可以想象在使用机器指令的实际机器上出现了用汇编语言作为机器语言的“虚拟”机器。

翻译(Translation)：先用转换程序将高一级机器级上的程序整个地变换成低一级机器级上可运行的等效程序，然后再在低一级机器级上去实现的技术。（先翻译后执行）

例：英语翻译

解释（Interpretation）：在低一级机器级上用它的一串语句或指令来等效高一级机器上的一条语句或指令的功能，通过对高一级机器语言程序中的每条语句或指令逐条解释来实现的技术 。（边解释边执行）

例：解释一件事

翻译和解释是语言实现的两种基本技术。一般来说，解释执行比翻译花的时间多，但占用存储空间较少。

在多层次结构中，通常第1、2级是用解释方法实现的，而第3级或更高级则用翻译方法实现。

计算机系统结构主要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。

固件(Firmware)是指那些存储在能永久保存信息的器件（如ROM）中的程序，是具有软件功能的硬件。固件的性能指标介于硬件与软件之间，吸收了软、硬件各自的优点，其执行速度快于软件，灵活性优于硬件，是软、硬件结合的产物，计算机功能的固件化将成为计算机发展中的一个趋势。

软件和硬件实现在逻辑功能上等效。

计算机系统结构设计者的主要任务就是要确定软硬件的分界；软件、硬件和固件的功能分配。

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。

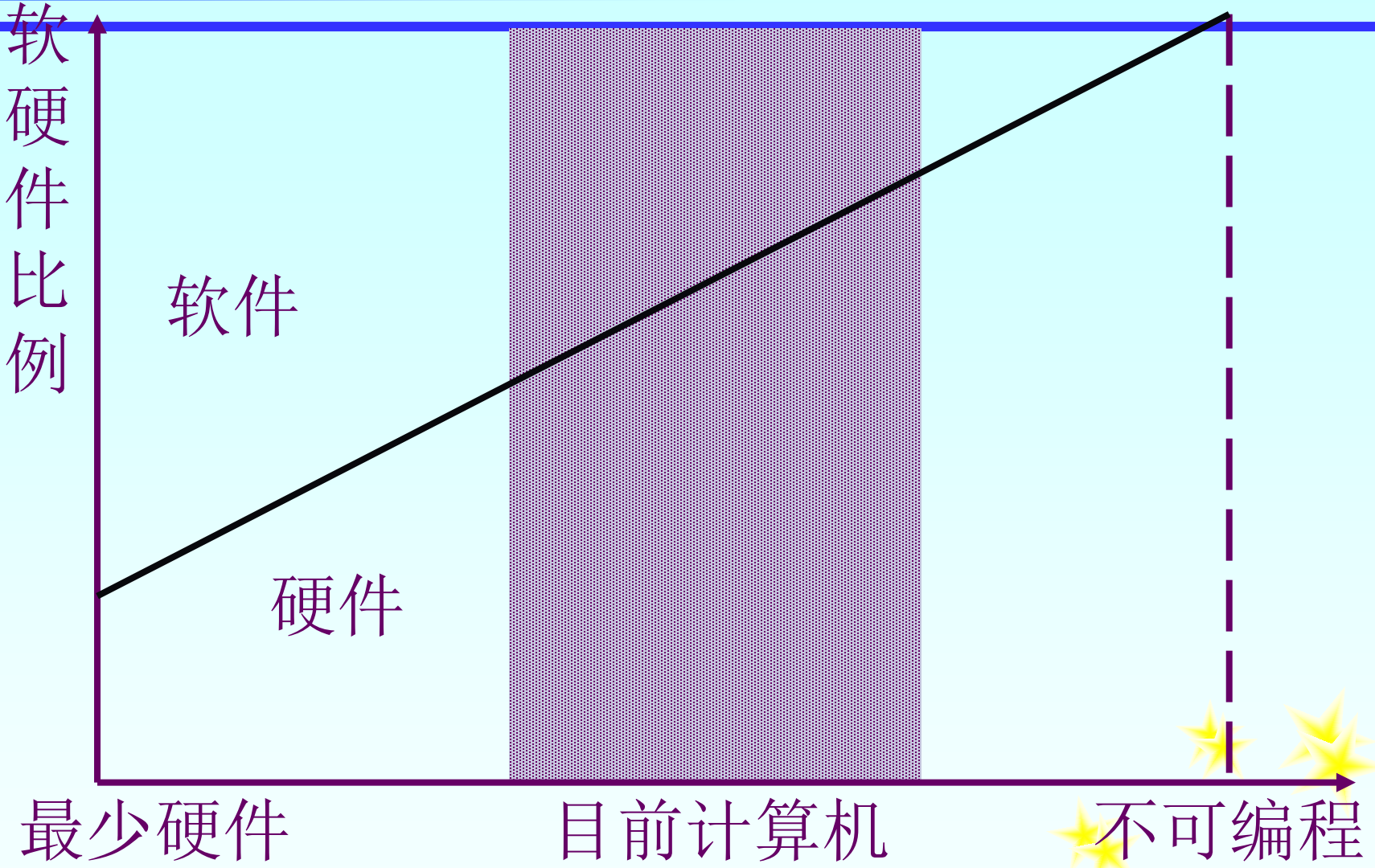


图 1.2 计算机系统的软、硬件功能分配

1.2.1 系统结构、组成与实现

我们这里所称的计算机系统结构或计算机体系结构(Computer Architecture) 指的是层次结构中传统机器级的系统结构，其界面之上的功能包括操作系统级、汇编语言级、高级语言级和应用语言级中所有软件的功能。界面之下的功能包括所有硬件和固件的功能。

从计算机系统的层次结构定义：

计算机系统结构是对计算机系统中各级界面的划分、定义及其上下的功能分配。

计算机系统结构定义

Amdahl于1964年在推出IBM360系列计算机时提出：程序员所看到的一个计算机系统的属性，即概念性结构和功能特性。

所谓概念性结构与功能特性，实际上就是计算机系统的外特性。

程序员：机器语言、操作系统、汇编语言、编译程序的程序员。

看到的：编写出能够在机器上正确运行的程序所必须了解到的东西。

对于这一定义，计算机界是有争议的，主要争议点基于这样一个事实，即由于计算机系统是包括软、硬件乃至固件资源的较复杂系统，因此处于不同级别的使用者（各级程序员）所看到的计算机具有不同的属性。例如，用高级语言编程的程序员，可以把**IBM PC**与**RS6000**两种机器看成是同一属性的机器；但对使用汇编语言编程的程序员来说，**IBM PC**与**RS6000**是两种截然不同的机器。

事实上，Amdahl提出的系统结构定义中的程序员指的是传统机器语言程序员。他们所看到的计算机属性，是硬件子系统的概念性结构和功能特性。包括：

(1)硬件能直接识别和处理的数据类型和格式等的数据表示；

(2)最小可寻址单位、 寻址种类、地址计算等的寻址方式；

(3)通用/专用寄存器的设置、数量、字长、使用约定等的寄存器组织；

(4)二进制或汇编级指令的操作类型、 格式、 排序方式、 控制机构等的指令系统；

(5)内存的最小编址单位、编址方式、容量、最大可编址空间等的存储系统组织；

(6)中断的分类与分级、中断处理程序功能及入口地址等的中断机构；

(7)系统机器级的管态和用户态的定义和切换；

(8)输入输出设备的连接、 使用方式、 流量、 操作结束、 出错指示等的机器级I/O结构；

(9)系统各部分的信息保护方式和保护机构。

计算机组成(Computer Organization)指的是计算机系统结构的逻辑实现，包括机器级内的数据流和控制流的组成以及逻辑设计等。它着眼于机器级内各事件的排序方式与控制机构、各部件的功能及各部件间的联系。计算机组成设计要解决的问题是在所希望达到的性能和价格下，怎样最佳、最合理地把各种设备和部件组织成计算机，以实现所确定的系统结构。

计算机组成设计要确定的方面一般应包括：

(1)数据通路宽度

(2)专用部件的设置

(3)各种操作对部件的共享程度

(4)功能部件的并行度

(5)控制机构的组成方式

(6)缓冲和排队技术

(7)预估、预判技术

计算机实现是指计算机组成的物理实现。它主要着眼于器件技术和微组装技术。

计算机组成和实现都属于计算机系统的内特性，这些特性对程序员来说是透明的（即程序员是看不到的）。

透明性概念：本来存在的事物或属性，从某种角度看似乎不存在。

这与日常生活中的“透明”的含义正好相反。日常生活中的“透明”是要公开，让大家看得到，而计算机中的“透明”，则是指看不到的意思。

所谓透明实际上就是指那些不属于自己管的部分。对于计算机系统结构而言，前述内容都是不透明的；而全部由硬件实现的，或是在机器语言、汇编语言编程中不会出现和不需要了解的部分都是透明的。

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。

再例如：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，对硬件设计者、计算机维修人员不透明。

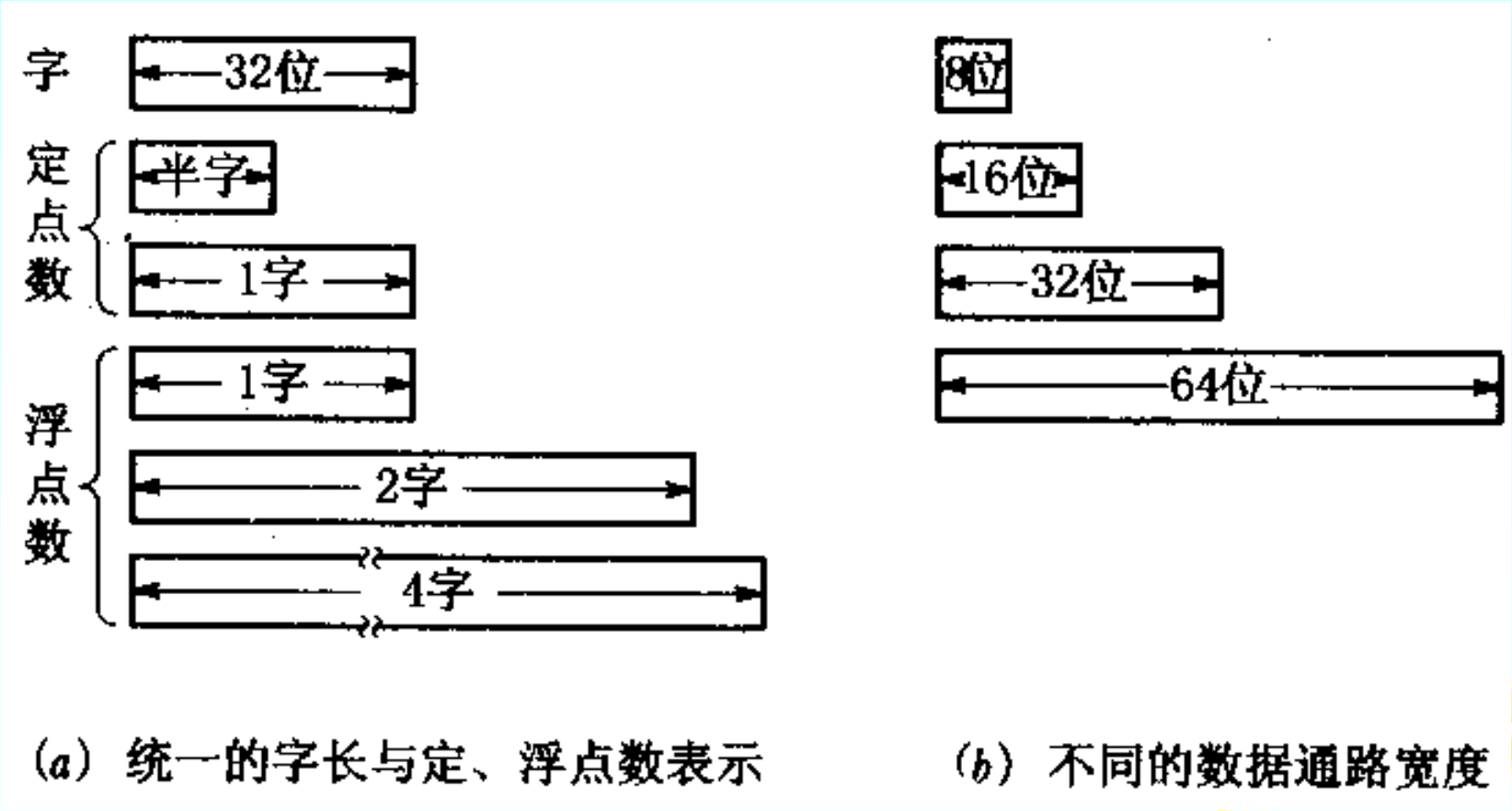


图 IBM 370系列机字长、数的表示和数据通路宽度

计算机系统结构、计算机组成和计算机实现是三个不同的概念。同一系统结构可因速度要求不同采用不同组成，一种组成可以采用多种不同的实现。

区分计算机系统结构与计算机组成这两个概念是十分重要的。

指令系统的确定属于计算机系统结构；而指令的实现，如取指令、分析指令、取操作数、运算、送结果等的操作安排和排序属于计算机组成。

确定指令系统中是否要设置乘法指令属于计算机系统结构；乘法指令是采用专门的高速乘法器实现，还是靠用加法器和移位器经时序信号控制其相加和移位来实现属于计算机组成。

主存容量与编址方式（按位、按字节还是字编址等）的确定属于计算机系统结构；而为达到性能价格要求，主存速度应选多快，采用何种逻辑结构等则属于计算机组成。

计算机制造商可能会向用户提供一系列系统结构相同的计算机，而它们的组成却有相当大的差别，即使是同一系列不同型号的机器，其价格和性能也是有极大差异的。

例如，有3台计算机：

1.没有Cache；

2.有单级CPU片外Cache；

3.第三台计算机既有CPU片内Cache，又有片外Cache。

这3台计算机具有不同的组成，但它们的系统结构可能是相同的。因此，只知其结构，不知其组成，就选不好性能价格比最合适的机器。此外，一种机器的系统结构可能维持许多年，但机器的组成却会随着计算机技术的发展而不断变化。

如果两台计算机具有不同的计算机组成和相同的计算机系统结构，那么在其中一台计算机上编译后的目标程序，拿到另一台计算机上也能运行，但两者的运行时间可能不同。

1.2.2 计算机系统结构、组成和实现的相互关系

1、系统结构要考虑组成和实现的发展，不要有过多或不合理的限制；

2、组成要考虑系统结构和实现。
决定于系统结构，受限于实现；

3、组成与实现不是被动的。
折中权衡；

4、实现是物质基础。

“计算机体系结构”学科=系统结构+组成

研究软硬件功能分配，最佳、最合理地实现分配给硬件的功能。

分为：

从程序设计者看——机器级界面

从计算机设计者看——分配给硬件的功能

1.3.1 软硬取舍的基本原则

确定软、硬件功能分配的第一个基本原则是，在现有硬件和器件条件下，系统要有高的性能价格比。

- 性能是一个综合指标，主要包括：
  - 实现费用
  - 速度
  - 其它性能等

- 实现费用 = 设计费用(D_s+D_h) + 重复生产费(M_s+M_h)
- 硬件设计费用(D_h) = 软件设计费用(D_s) * 100
- 硬件重复生产费(M_h) = 软件重复生产费(M_s) * 100
- 软件设计费用(D_s) = 软件重复生产费(M_s) * 10000

R为软件重复出现次数（占用内存、占用介质）

C为该功能在软件实现时需重新设计的次数

当台数为V时，每台的硬件费用和软件费用

$D_h/V + M_h < C \times Ds/V + R \times Ms$

$100\ Ds/V + 100Ms < C \times Ds/V + R \times Ms$

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

结论1：当R很大时，即经常使用的基本功能适宜用硬件实现。

结论2：当V很大时，即生产台数很多时适宜用硬件实现。

确定软、硬件功能分配的第二个基本原则是，要考虑到准备采用和可能采用的组成技术，使它尽可能不要过多或不合理地限制各种组成、 实现技术的采用。

确定软、硬件功能分配的第三个基本原则是，不能仅从“硬”的角度去考虑如何便于应用组成技术的成果和发挥器件技术的进展，还应从“软”的角度把为编译和操作系统的实现，以至高级语言程序的设计提供更多更好的硬件支持放在首位。

1.3.2 计算机系统的设计思路

- 出发点：多级层次结构
- 三种设计思路：

方法1：由上向下（Top-Down）

方法2：由下向上（Bottom-Up)

方法3：中间开始（Middle-Out)

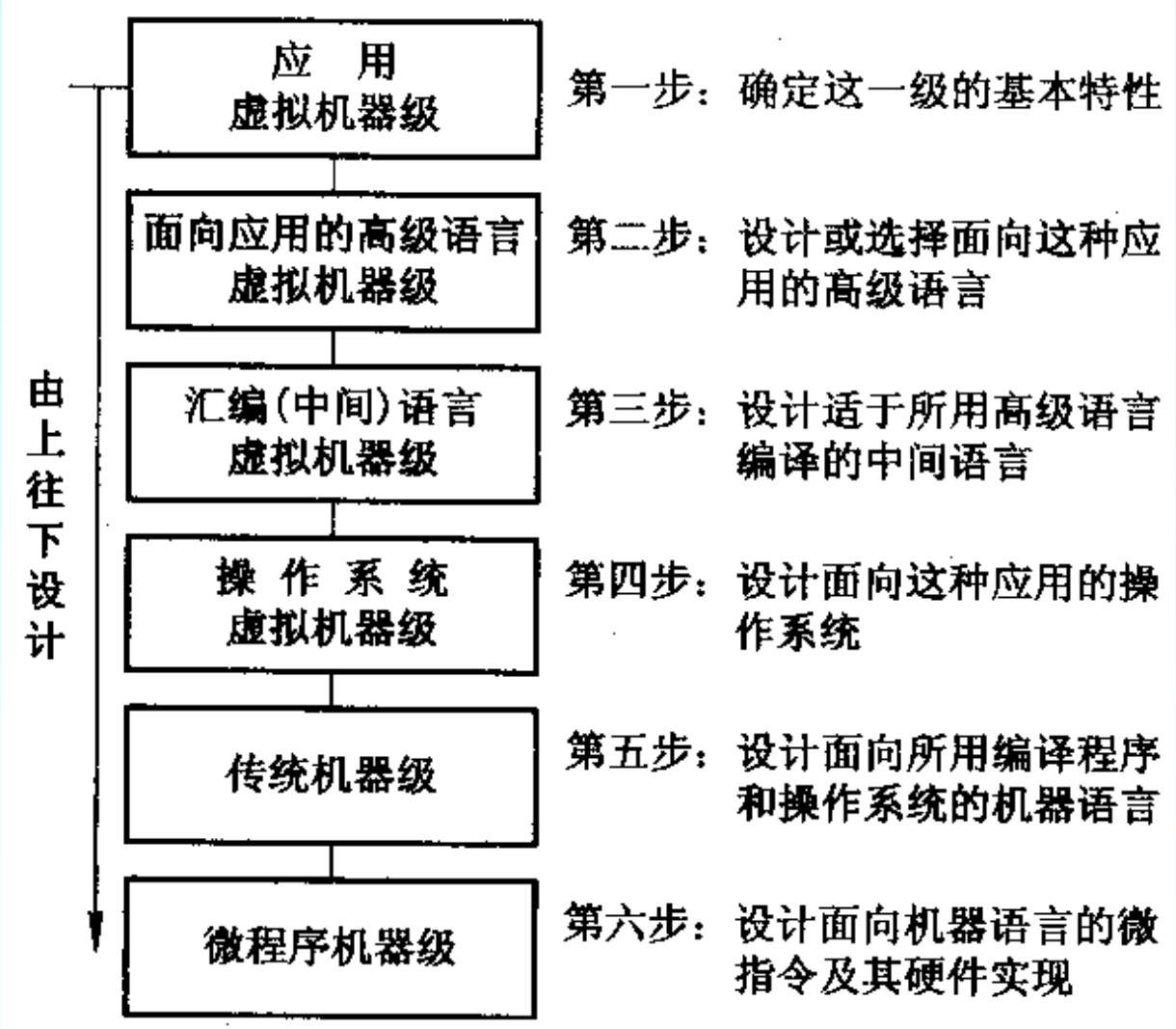


图 1.5 计算机系统“由上往下”设计的方法

由上往下设计

特点：从应用开始，逐级往下。

优点：运行效率高，软硬分配合理，适用于专用机的设计。

缺点：适应性差，周期长。

解决方法：不完全优化，不专门设计机器级“选型”。

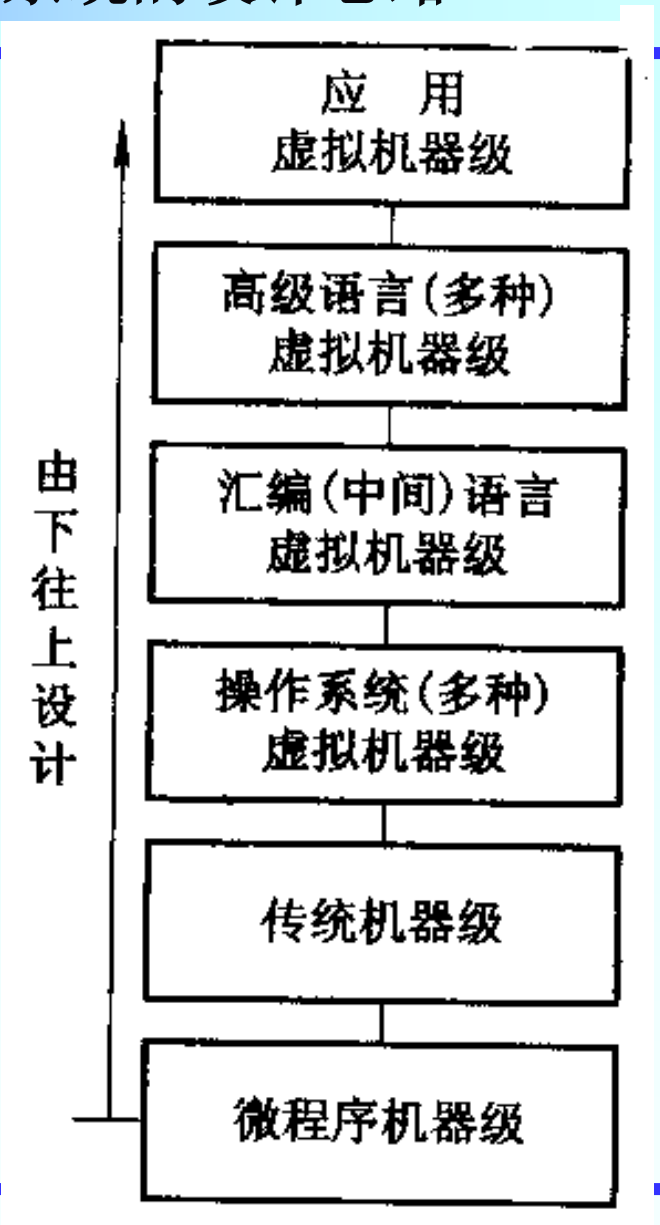


图 1.6 计算机系统“由下往上”设计的方法

由下往上设计

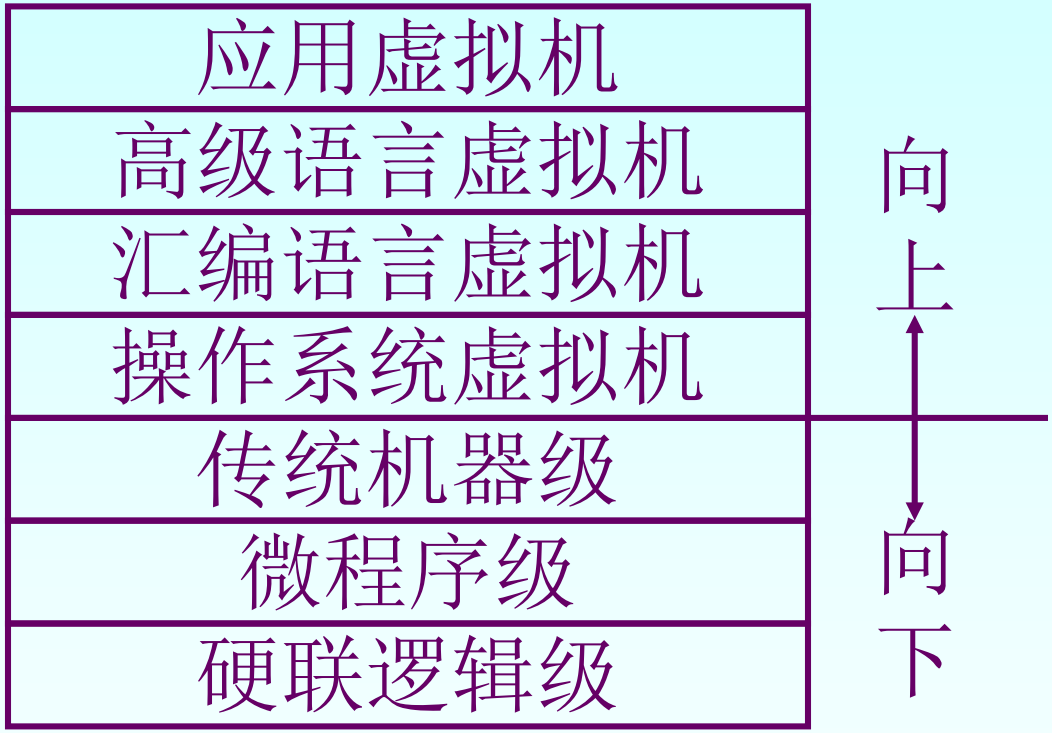
特点：根据器件等情况研制硬件，根据要求配置软件。

优点：可设计通用计算机。

缺点：软硬脱节、分离；硬件无法改变，某些性能指标是虚假的。

——很少使用——

由中间开始



由中间开始设计

特点：从软硬界面开始，同时进行软硬件设计。

优点：软硬件功能分配比较合理，缩短了研制周期，有利于硬件和软件设计人员之间的交流协调，解决软硬设计分离和脱节的问题。

“中间”指的是层次结构中的软硬交界面，目前多数是在传统机器级与操作系统机器级之间。

计算方向的变化导致三个不同的计算机市场：

1、桌面电脑：性价比

2、服务器：可用性、可扩展性、有效的吞吐量

3、嵌入式计算机：实时需求、最小化存储器需求、最小化功耗需求

四种飞速发展的实现技术对现代计算机实现的影响深远：

1.集成电路逻辑技术

2.半导体DRAM（动态随机访问存储器）

3.磁盘存储技术

4.网络技术

1.4.1计算机系统设计的定量原理

1.Amdahl定律

系统中对某一部件采用某种更快执行方式所能获得的系统性能改进程度，取决于这种执行方式被使用的频率或所占总执行时间的比例。

首先，**Amdahl**定律定义了加速比的概念。假设对机器进行某种改进，那么机器系统的加速比为：

$$\text{加速比} = \frac{\text{改进后的性能}}{\text{改进前的性能}}$$

或

$$\text{加速比} = \frac{\text{改进前的总执行时间}}{\text{改进后的总执行时间}}$$

系统加速比告诉我们改进后的机器比改进前快多少。Amdahl定律使我们能快速得出改进所获得的效益。系统加速比依赖于两个因素：

可改进比例（F_e），它总是小于1的。

性能提高比（S_e），它总是大于1的。

$$Fe = \frac{\text{可改进部分占用的时间}}{\text{改进前整个任务的执行时间}},$$

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

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为：

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

其中（1-Fe)表示不可改进部分。

实际上，**Amdahl**定律还表达了一种性能增加的递减规则：如果仅仅对计算机中的一部分做性能改进，则改进越多，系统获得的效果越小。**Amdahl**定律的一个重要推论是：如果只针对整个任务的一部分进行优化，那么所获得的加速比不大于：

$$\frac{1}{(1-Fe)}$$

例1：假设将某一部件的处理速度加快到10倍，该部件的原处理时间仅为整个运行时间的40%，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2：某计算机系统采用浮点运算部件后，浮点运算速度提高到原来的25倍，而系统运行某一程序的整体性能提高到原来的4倍，试计算该程序中浮点操作所占的比例。

- 解：由题意可知：Se=25，Sn=4，根据Amdahl定律：

$$4 = \frac{1}{(1-Fe)+\frac{Fe}{25}}$$

- 由此可得：Fe≈78.1%。

例3：求浮点数（FP）平方根的不同实现方法在性能上可能有很大差异。假设在程序中求浮点平方根（FPSQR）操作占总执行时间的20%，一种方法是增加专门的FPSQR 硬件，可提高速度10倍；另一种方法是提高所有FP指令的速度，FP指令占总执行时间的50%，FP指令的速度提高为原来的1.6倍。试比较这两种方法。

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

$$S_{n(FP)} = \frac{1}{(1-0.5)+\frac{0.5}{1.6}} = \frac{1}{0.8125} = 1.23$$

2. CPU性能公式

程序执行的**CPU**时间为：

$$\text{CPU时间} = \frac{\text{CPU时钟周期数}}{\text{时钟频率}}$$

若将程序执行过程中所处理的指令数，记为**IC**。这样可以获得一个与计算机系统结构有关的参数，即“指令时钟数**CPI**”。

$$\text{CPI} = \frac{\text{CPU时钟周期数}}{\text{IC}}$$

主频和CPU时钟周期

主频是衡量CPU速度的重要参数。CPU的主频又称为时钟频率，表示在CPU内数字脉冲信号振荡的速度，与CPU实际的运算能力并没有直接关系。主频的倒数就是CPU时钟周期，这是CPU中最小的时间元素。每个动作至少需要一个时钟周期。

CPI

CPI（Cycles per Instruction）就是每条指令执行所用的时钟周期数。由于不同指令的功能不同，造成指令执行时间不同，也即指令执行所用的时钟数不同，所以CPI是一个平均值。在现代高性能计算机中，由于采用各种并行技术，使指令执行高度并行化，常常是一个系统时钟周期内可以处理若干条指令，所以CPI参数经常用IPC（Instructions per Cycle）表示，即每个时钟周期执行的指令数。

$$\text{CPU时间} = \frac{\text{IC} \times \text{CPI}}{\text{时钟频率}}$$

这个公式通常称为**CPU**性能公式。它的三个参数反映了与系统结构相关的三种技术：

① 时钟频率：反映了计算机实现技术、生产工艺和计算机组织。

② **CPI**：反映了计算机实现技术、计算机指令系统的结构和组织。

③ **IC**：反映了计算机指令级的结构和编译技术。

假设计算机系统有n种指令，其中第i种指令的处理时间为CPI_i，在程序中第i种指令出现的次数为I_i，则有：

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

1.4.2 衡量计算机系统性能的主要标准

1. 吞吐量和响应时间

吞吐量和响应时间是描述计算机系统性能常用的参数，也是用户所关心的。吞吐量是指系统在单位时间内处理请求的数量。响应时间是指系统对请求作出响应的时间，响应时间包括CPU时间（运行一个程序所花费的时间）与等待时间（用于磁盘访问、存储器访问、I/O操作、操作系统开销等时间）的总和。

2. 运算速度

（1） 时钟频率（主频）：用于同类处理机之间　　如：**PentiumII/450** 比 **PentiumII/300**快**50%**，…

（2） 指令执行速度　一种很经典的表示方法　**MIPS (Million Instructions Per Second), KIPS, GIPS, TIPS**

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

主要缺点：

（1）不同指令的速度差别很大

（2）指令使用频度差别很大

（3）有相当多的非功能性指令

功能性指令：加、减、乘、除等

另一种替代标准

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

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4：计算Pentium II 450处理机的运算速度。

解：由于PentiumII 450处理机的IPC＝2 (或CPI＝0.5), Fz＝450MHz，因此，

$MIPS_{Pentium\ II\ 450}$＝Fz×IPC＝450×2＝900(MIPS)

例5：微机A和B是采用不同主频的CPU芯片，片内逻辑电路完全相同。

（1）若A机的CPU主频为8MHz，B机为12MHz，则A机的CPU时钟周期为多少？

（2）如A机的平均指令执行速度为0.4MIPS，那么A机的平均指令周期为多少？

（3）B机的平均指令执行速度为多少？

解：（1）A机的CPU主频为8MHz，所以A机的CPU时钟周期=1 ÷ 8MHz = 0.125μs。

（2）A机的平均指令执行速度为0.4MIPS，所以A机的平均指令周期= 1 ÷ 0.4MIPS = 2.5μs。

（3）A机平均每条指令时钟周期数 = 2.5μs ÷ 0.125μs= 20

而微机A和B片内逻辑电路完全相同，所以B机平均每条指令的时钟周期数也为20。

B机的平均指令执行速度 =主频÷ CPI=12 ÷20 MIPS = 0.6MIPS。

3、峰值速度

峰值指令速度MIPS、GIPS、TIPS

Pentium III 500有3条指令流水线，则其峰值指令速度为：

3×500MHz＝1500 (MIPS)

即每秒15亿次

例8：一个由8台机器组成的Cluster系统，每台机器是4个PentiumIII 500组成的SMP系统；计算这个Cluster系统的指令峰值速度。

解：

峰值指令速度：

500MHz × 8×4×3＝ 48(GIPS)

即每秒480亿次。

4. 等效指令速度：吉普森（Gibson）法

$$\text{等效指令执行时间} T = \sum_{i=1}^{n} (W_i \times T_i)$$

$$\text{等效指令速度} MIPS = 1 / \sum_{i=1}^{n} \frac{W_i}{MIPS_i}$$

$$\text{等效} CPI = \sum_{i=1}^{n} (CPI_i \times W_i)$$

其中，

Wi：指令使用频度，**i**：指令种类

静态指令使用频度：在程序中直接统计

动态指令使用频度：在程序执行过程中统计

在计算机发展的早期，用加法指令的运算速度来衡量计算机的速度。通常：加、减法**50%**，乘法**15%**，除法**5%**，程序控制**15%**，其他**15%**。

例6：我国最早研制的小型计算机DJS-130，定点16位，加法每秒50万次，但没有硬件乘法和除法指令，用软件实现乘法和除法，速度低100倍左右。求等效速度。

解：

定点等效速度为：

$$\text{等效指令速度 MIPS} = 1/\left(\frac{0.80}{0.5}+\frac{0.20}{0.5/100}\right) = 0.02\,\text{MIPS}$$

即每秒2万次，由于乘法和除法用软件实现，等效速度降低了25倍。

例7：假设在程序中浮点开平方操作FPSQR的比例为2%，它的CPI为100；其他浮点操作FP的比例为23%，它的CPI＝4.0；其余75%指令的CPI＝1.33，计算该处理机的等效CPI。如果FPSQR操作的CPI也为4.0，重新计算等效CPI。

解：

等效CPI_1＝100×2%＋4×23%＋1.33×75%

＝3.92

等效CPI_2＝4×25%＋1.33×75%

＝2.00

由于改进了仅占2%的FPSQR操作的CPI，使等效速度提高了近一倍。

1.4.3 计算机性能的比较

	计算机A	计算机B	计算机C
程序P1（s）	1	10	20
程序P2（s）	1000	100	20
总计（s）	1001	110	40

总执行时间

最简单的相对性能综合评价方法

算术平均速度

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

加权执行时间

为每一个程序赋予一个权重值**wi**，将各个程序的权重值与执行时间的乘积加起来。

	A	B	C	**w1**	**w2**	**w3**
程序P1（s）	1	10	20	0.50	0.909	0.999
程序P2（s）	1000	100	20	0.50	0.091	0.001
平均执行时间**w1**	500.5	55	20			
平均执行时间**w2**	91.91	18.19	20			
平均执行时间**w3**	2	10.09	20			

加权平均速度

$$\sum_{i=1}^{n} W_i \bullet T_i$$

归一化执行时间

将执行时间对一台参考机器归一化，然后取归一化执行时间的平均值。平均归一化执行时间可以表示成算术平均值，也可以表示成几何平均值。

几何平均速度：

$$G = \sqrt[n]{\prod_{i=1}^{n} ETR_i}$$

算术平均值因参考机器不同而不同，
几何平均值不随参考机器的变化而变化。

	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
算术平均值	1.0	5.05	10.01	5.05	1.0	1.1	25.3	2.75	1.0
几何平均值	1.0	1.0	0.63	1.0	1.0	0.63	1.58	1.58	1.0
总时间	1.0	0.11	0.04	9.1	1.0	0.36	25.03	2.75	1.0

其中，**ETR(execution time ratio)**中的**n** 指不同的程序。

几何平均速度与机器无关，与程序的执行时间无关。

1.4.4 计算机系统的性能评价

用于评价计算机系统性能的程序称为评测程序（**benchmark**）。评测程序能够揭示计算机系统对于某类应用的优势或不足。五种评测程序，评测的准确度依次递减。

真实的程序

修改过的程序

程序内核

小型基准程序

综合基准程序

把应用程序中用得最频繁的那部分核心程序作为评价计算机性能的标准程序。称为基准程序 (benchmark)。

整数测试程序：Dhrystone

用C语言编写，100条语句。包括：各种赋值语句，各种数据类型和数据区，各种控制语句，过程调用和参数传送，整数运算和逻辑操作。

VAX-11/780的测试结果为每秒1757个Dhrystones，即：1VAX MIPS＝1757 Dhrystones/Second

浮点测试程序：Linpack

用FORTRAN语言编写，主要是浮点加法和浮点乘法操作。

用 MFLOPS（Million Floating Point Operations Per Second）表示；GFLOPS、TFLOPS

Whetstone基准测试程序

- 用FORTRAN语言编写的综合性测试程序，主要包括：浮点运算、整数算术运算、功能调用、数组变址、条件转移、超越函数。
- 测试结果用Kwips表示。

SPEC基准测试程序 (System performance evaluation Cooperative)

由30个左右世界知名计算机大厂商所支持的非盈利的合作组织，包括：IBM、AT&T、 BELL、Compaq、CDC、 DG、DEC、 Fujitsu、 HP、 Intel、 MIPS、Motolola、SGI、SUN、Unisys等；

SPEC能够全面反映机器的性能，具有很高的参考价值；

以VAX-11/780的测试结果作为基数。

SPEC1.0 1989年10月宣布，程序量超过15万行，包含10个测试程序，4个定点程序，6个浮点程序；测试结果用SPECint'89和SPECfp'89表示。

1992年，又增加10个测试程序，共有6个定点程序和14个浮点程序，测试结果用SPECint'92和SPECfp'92表示。

1995年，推出SPECint'95和SPECfp'95。

2000年，推出SPECint2000（11个整数基准程序）和SPECfp2000（14个浮点基准程序）。

处理机	SPECint'95	SPECfp'95
PentiumII 400	18.5	13.3
PentiumII 450	18.7	13.7
PentiumIII 500	20.6	14.7
PientiumIII 550	22.3	15.6
Celeron 300A	12.0	9.66
Celeron 333	13.1	10.20
Celeron 366	14.1	10.70
Celeron 400	15.1	11.20
Celeron 433	16.1	11.60
Celeron 466	17.0	12.00

TPC基准程序

Transaction Processing Council（事务处理委员会）

成立于**1988**年，已有**40**多个成员；

用于评测计算机的事务处理、数据库处理、企业管理与决策支持等方面的性能。

1989年**10**月、**1990**年**8**月和**1992**年**7**月发表了**TPC-A**、**TPC-B**和**TPC-C**。

1.5.1 软件对系统结构的影响

- 问题的提出
  - 软件成本越来越高
  - 软件产量和可靠性的提高困难
    - ➔ 重新研究合理的软、硬件功能分配
  - 积累了大量成熟的软件
  - 排错比编写困难、软件生产率低
    - ➔ 不希望重新编写软件
- 因此，在新的系统中必须解决软件的可移植性问题

软件可移植性的定义

软件不用修改或只需少量加工就能由一台机器搬到另一台机器上运行，即同一软件可以应用于不同的环境。

实现软件可移植性的几种技术

技术一：统一高级语言

技术二：采用系列机思想

技术三：模拟与仿真

1.采用统一的高级语言方法

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、……

2.采用系列机方法

系列机定义：

同一厂家生产的具有相同的系统结构，不同组成和实现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，根据不同性能的要求和当时的器件发展情况，设计出各种性能、价格不同的计算机系统。一种系统结构可以有多种组成，一种组成可以有多种物理实现。

如**IBM370系列机**：

370/115、125、135、145、158、168等各种型号。

它们相同的系统结构，不同的组成和实现技术，不同的性能和价格。

相同的指令系统，分别采用顺序执行、重叠、流水和并行处理方式。

相同的**32**位字长，数据通道的宽度分别为**8**位、**16**位、**32**位、**64**位。

PC系列机：

8088、8086、80186、80286、80386、80486、Pentium、PentiumII、PentiumIII

不同工作主频；

不同扩展功能：**Pentium、Pentium Pro、Pentium MMX**；

不同的**Cache**：**PentiumII、Celeron、Xeon**；

不同的机器字长：**8位（8088）、16位（80286）、32位（80386）、64位。**

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。

软件兼容性设计方法

原因：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。

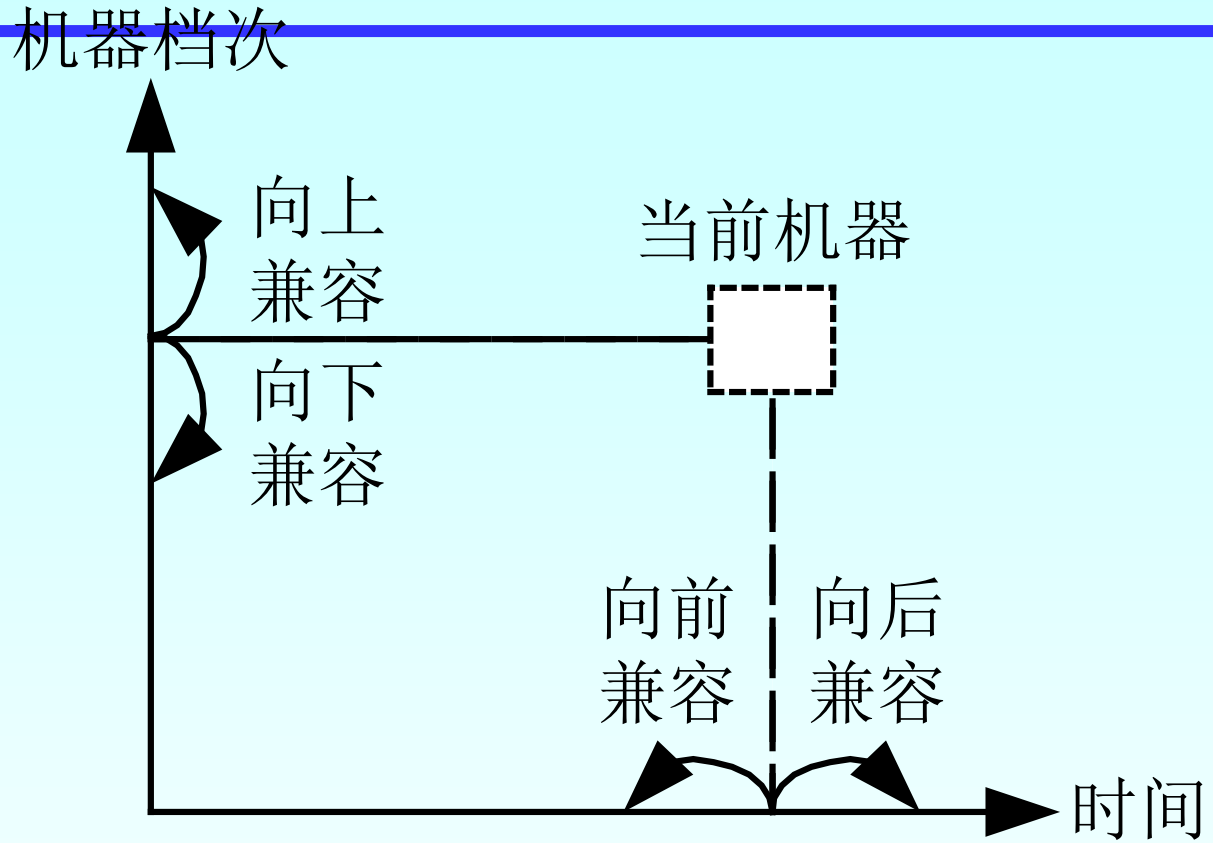
兼容种类

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



其中**向后兼容最重要**，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。

兼容机定义：

不同厂家生产的具有相同的系统结构的计算机系统。

3.采用模拟与仿真方法

定义：

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。

用硬件、固件或软件、硬件、固件混合实现的叫仿真。

模拟的实现方法：

在A计算机上通过解释方法实现B计算机的指令系统，即B机器的每一条指令用一段A机器的程序进行解释执行。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机器的每条指令。A机器称为宿主机，B机称为目标机。

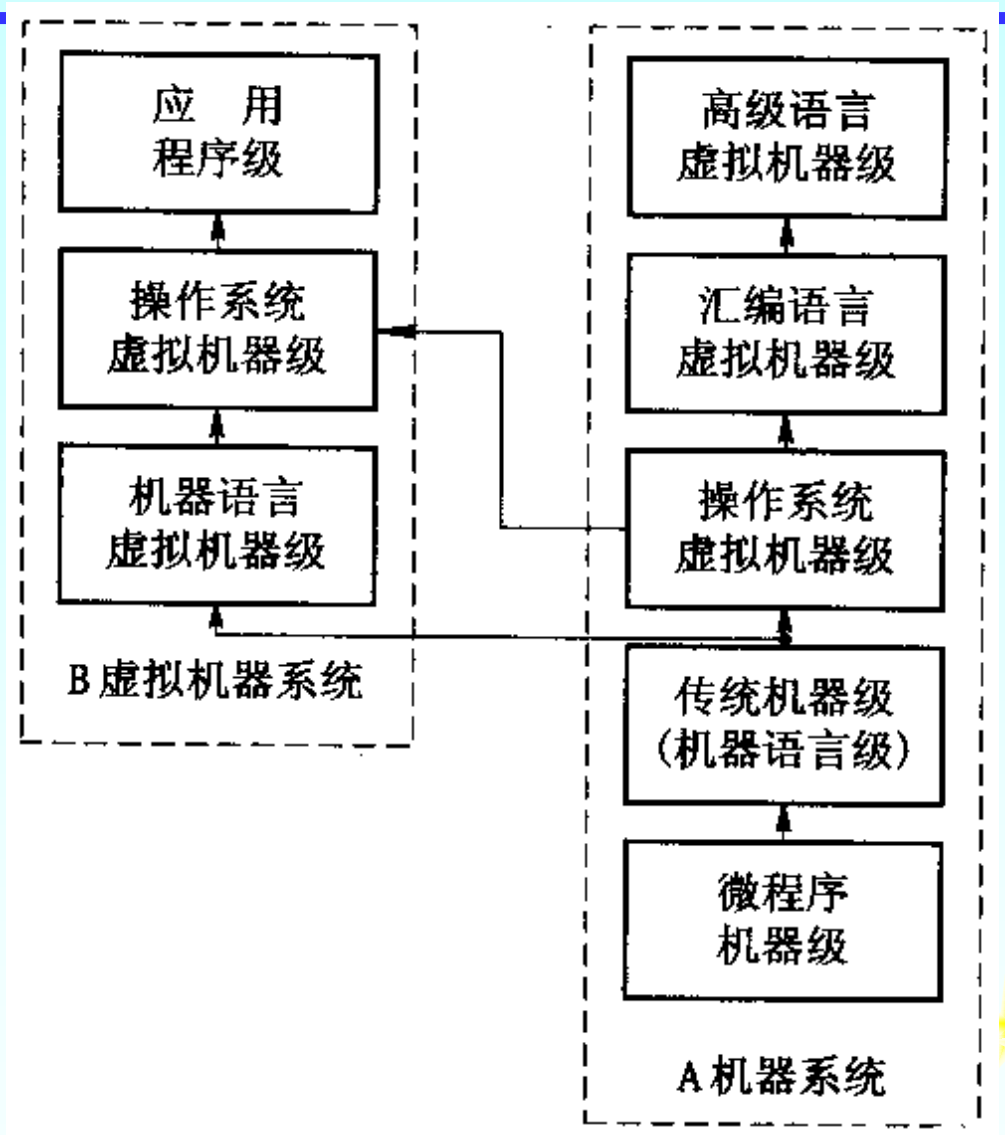


图 1.5 用模拟方法实现应用软件的移植

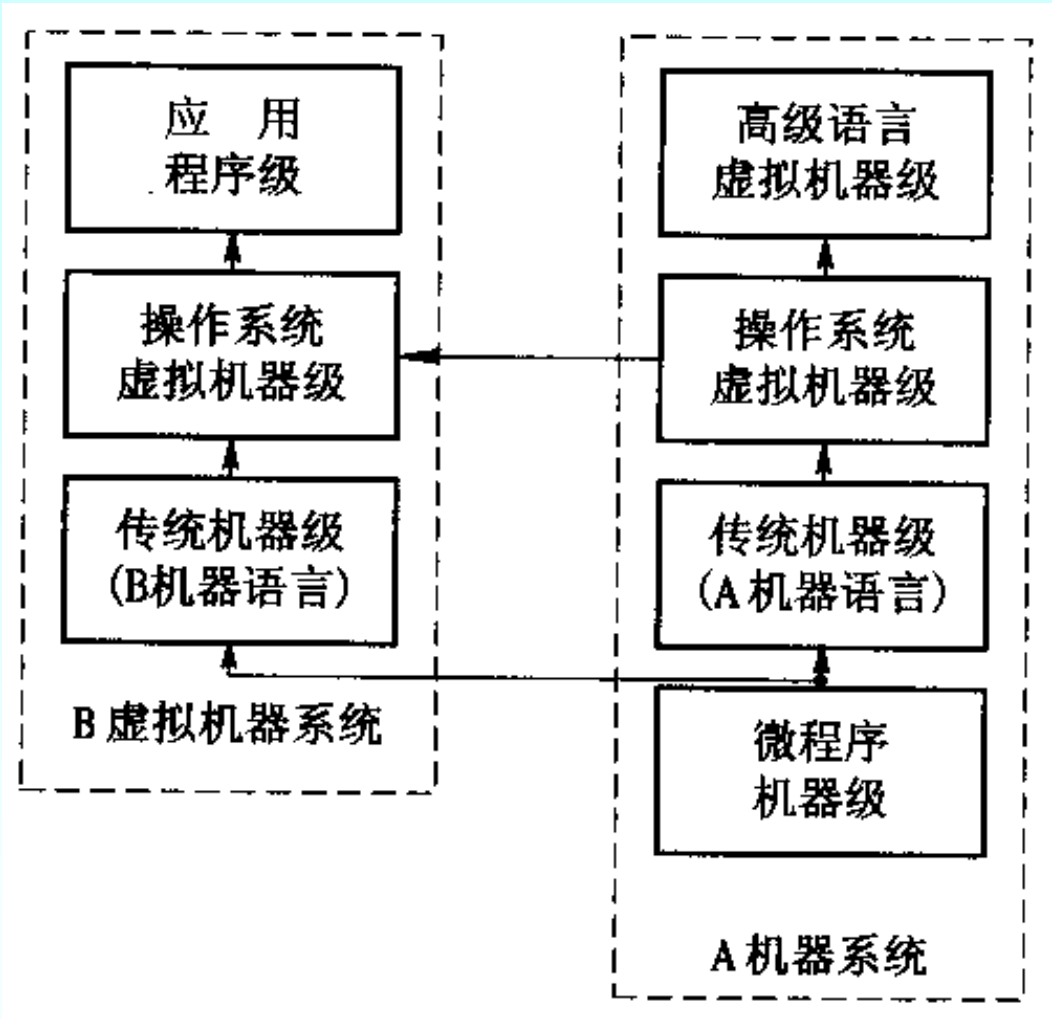


图 1.6 用仿真方法实现应用软件的移植

优缺点比较

模拟方法运行速度低，仿真方法速度高。

仿真需要较多的硬件（包括控制存储器）。

系统结构差别大的机器难于完全用仿真方法来实现。

仿真——微程序——控存中

模拟—— 机器语言程序——主存中

通常将模拟和仿真混合使用。

除了指令系统之外，还有存储系统、I/O系统、中断系统、控制台的操作等的模拟/仿真。

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。

一种新的设想：目标代码的兼容性研究。

一种机器的目标代码到另一种机器的目标代码的编译。

目标代码的并行重编译。

软件移植技术小结：

A 统一高级语言

解决结构相同或完全不同的各种机器上的软件移植，是重要方向。

问题：语言标准化很重要，短期很难，只能相对统一。

B 系列机

普遍采用，只解决同一系列结构内的软件兼容。

问题：兼容的约束阻碍系统结构取得突破进展。

C 模拟

灵活性较大，可实现不同系统间的软件移植。

问题：结构差别大时，效率和速度急剧下降。

D 仿真

速度损失小，可实现不同系统间的软件移植。

问题：灵活性较小，只能在结构差别不大的机器间采用。需结合模拟。

1.5.2 应用对系统结构的影响

- 应用对系统的结构的发展有重要的影响
- 其中一些要求是共同的，如程序可移植性，高性能价格比，高可靠性，便于使用等

40~50年代初	科学计算	简单通用机
50年代中/末	商业、事务处理(字符处理，I/O量大)、工业控制（中断、实时）	专用机
60年代中期	同时支持商业、事务处理、工业控制等应用	多功能通用机 良性循环

60年代中期	小型机、微型机多功能通用化	
60年代末 70年代初	特高可靠性应用，数据处理	容错技术
70年代初中期	特高速应用，数据处理	阵列机，向量机 价格昂贵
70年代中后期	高速阵列处理部件，一台功能很强的专用处理机(数组处理机)	数组处理机作为外设连接到通用机
80年代	非数据处理应用，主要为数据和信息的管理	
90年代中后期	知识处理，智能处理	支持高速并行，自然语言理解等

1.5 对系统结构的影响因素

- 处理性能和价格的两种途径：
  - 维持价格不变，充分利用器件等技术的进展，不断提高机器的性能
  - 在性能基本不变的基础上，利用器件等技术的进展不断降低机器的价格

- 从系统结构的观点看，实质上就是在低档(型)机器上引用，甚至照搬高档(型)的系统结构和组成

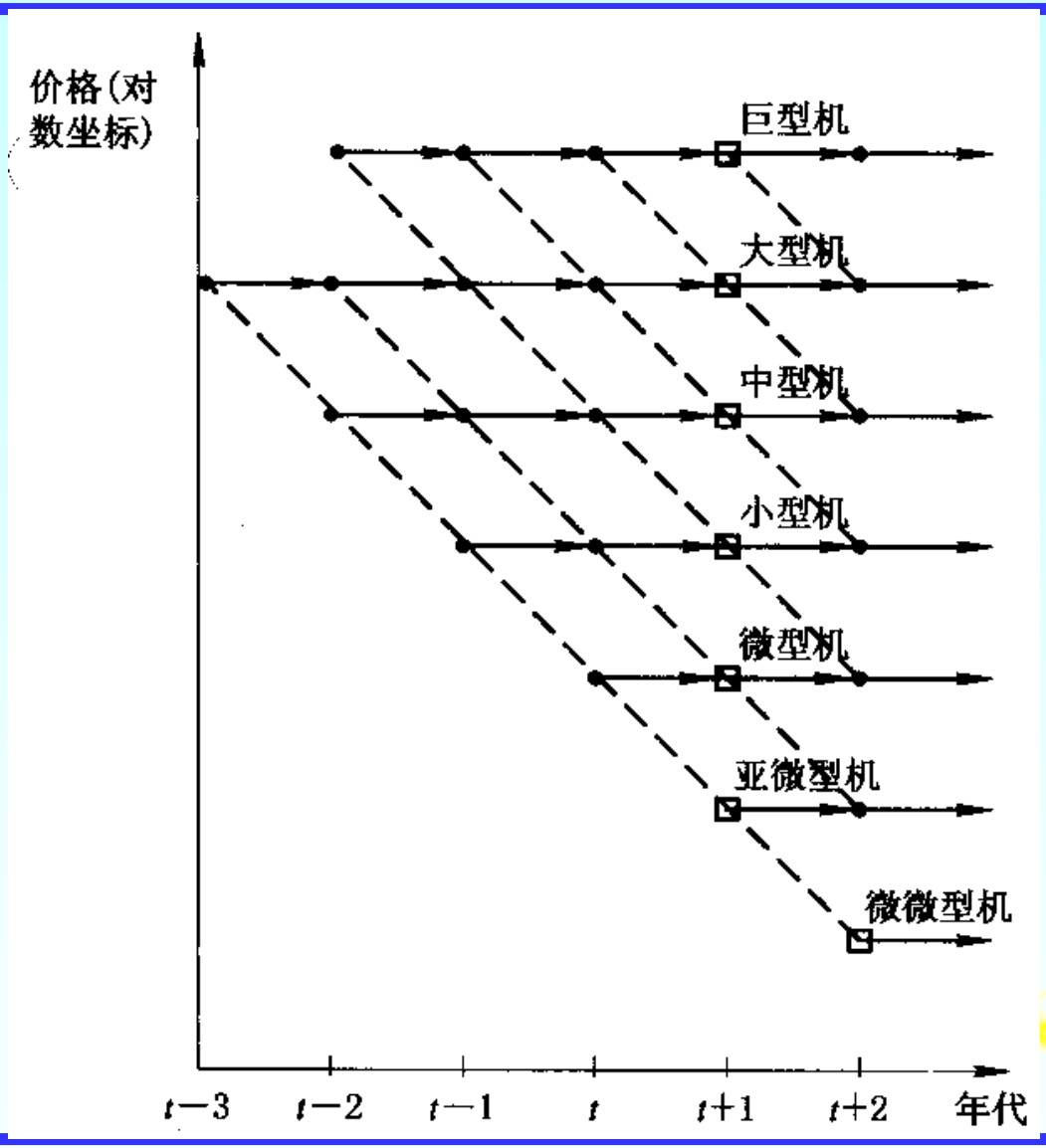


图 各型机器价格性能随时间变化的趋势

1.5.3 器件对系统结构的影响

- 器件的发展是推动系统结构和组成前进的关键因素和主要动力
- 计算机使用的基本器件：
  - 经历了电子管→晶体管→小规模IC→LSI和VLSI IC

- 集成电路逻辑技术，其晶体管密度每年以35%增长，4年翻2番。芯片尺寸每年增长10~20%。使得每个芯片上晶体管每年增长50%
- 半导体DRAM，其晶体密度每年增长40~60%，访问时间平均每十年减少1/3。每片的带宽随着延迟时间缩短而以其2倍的速度增长

- 网络技术，取决于交换和传输系统。延迟和带宽都能改进
- 磁盘存储技术，存储密度最近每年100%增长，访问时间过去10年缩短了1/3

- 计算机已经发展了五代
- 这五代计算机分别具有明显的器件、体系结构技术和软件技术的特征

60年代末 70年代初	非用户片 存储类器件发展	用存储器件取代逻辑器件	微程序
70年代中期	现场片	改变器件的内容和功能	**PROM, FPLA**
	用户片	按用户要求生产的**VLSI**	
	同一系列各档机可分别用通用片、现场片和用户片实现		

1.5 对系统结构的影响因素

第一代 (1945-1954)	电子管和继电器	存储程序计算机、程序控制I/O	机器语言和汇编语言	普林斯顿ISA、ENIAC、IBM701
第二代 (1955-1964)	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 LARC、CDC1604、IBM7030
第三代 (1965-1974)	SSI和MSI、多层印刷电路、微程序	流水线、Cache、先行处理、系列计算机	多道程序和分时操作系统	IBM360/370、CDC6600/7600、DEC PDP-8
第四代 (1974-1990)	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 VAX9000、Convax-1
第五代 (1991-)	高性能微处理器、高密度电路	超标量、超流水、SMP、MP、MPP	大规模、可扩展并行与分布处理	SGI Cray T3E、IBM SP2、DEC AlphaServer8400

- 改变了逻辑设计的传统方法
  - 逻辑简化 ➔ 充分利用**VLSI**，获得更高的性能价格比
  - 缩短周期，提高效能，使用大批量生产的**VLSI**
  - 硬的逻辑设计方法 ➔ 微程序、微高级语言、**CAD**等软的设计方法

- 使系统结构“下移”的速度加快
  - 大型机的数据表示、指令系统、OS等很快出现在小型微型机上
  - 多个CPU的分布处理
- 促进了算法、语言和软件的发展
  - 并行处理机/网络 ➔ 并行算法、并行语言、并行/分布式操作等

- 器件的发展是推动系统结构和组成前进的关键因素和主要动力
- 系统结构设计者要密切了解器件的现状和发展趋势，关注和分析新器件的出现和集成度的提高会给系统结构的发展带来什么样的新途径和新方向

总之，软件、应用、器件对系统结构的发展是有着很大影响的， 反过来，系统结构的发展又会对软件、应用、器件的发展提出新的要求，促使其有更大的发展。计算机系统结构设计者不仅要了解结构、组成、实现的关系，还要充分了解掌握软件、应用、器件发展的现状、趋势和发展要求，只有这样，才能对系统的结构进行有成效的设计、研究和探索。

1.6.1 并行性概念

1. 并行性的含义与并行性级别

并行性包含同时性和并发性二重含义。

同时性—两个或多个事件在同一时刻发生。

并发性—两个或多个事件在同一时间间隔内发生。

从计算机系统中执行程序的角度来看，并行性等级从低到高可以分为四级。它们分别是：

指令内部——一条指令内部各个微操作之间的并行。

指令之间——多条指令的并行执行。

任务或进程之间——多个任务或程序段的并行执行。

作业或程序之间——多个作业或多道程序的并行。

从计算机系统中处理数据的并行性来看，并行性等级从低到高可以分为：

位串字串—同时只对一个字的一位进行处理，这通常是指传统的串行单处理机，没有并行性。

位并字串—同时对一个字的全部位进行处理，这通常是指传统的并行单处理机，开始出现并行性。

位片串字并—同时对许多字的同一位(称位片)进行处理， 开始进入并行处理领域。

全并行——同时对许多字的全部或部分位组进行处理。

并行性是贯穿于计算机信息加工的各个步骤和阶段的，从这个角度来看，并行性等级又可分为：

存储器操作并行—可以采用单体多字、多体单字或多体多字方式在一个存储周期内访问多个字，进而采用按内容访问方式在一个存储周期内用位片串字并或全并行方式实现对存储器中大量字的高速并行比较、检索、更新、变换等操作。典型的例子就是并行存储器系统和以相联存储器为核心构成的相联处理机。

处理器操作步骤并行——处理器操作步骤可以指一条指令的取指、 分析、 执行等操作步骤，也可指如浮点加法的求阶差、对阶、尾加、舍入、规格化等具体操作的执行步骤。处理器操作步骤并行是将操作步骤或具体操作的执行步骤在时间上重叠流水地进行。典型的例子就是流水线处理机。

处理器操作并行——为支持向量、数组运算，可以通过重复设置大量处理单元，让它们在同一控制器的控制下，按照同一条指令的要求对多个数据组同时操作。典型的例子就是阵列处理机。

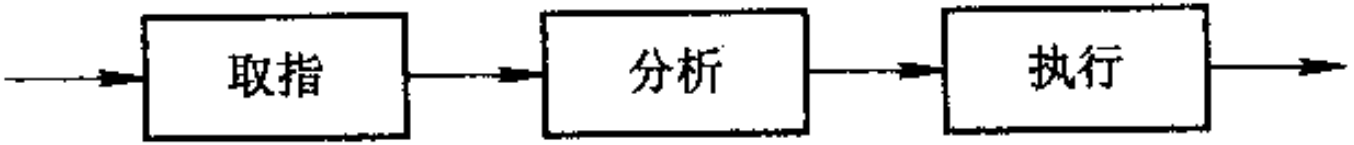
指令、任务、作业并行—这是较高级的并行，虽然它也可包含如操作、操作步骤等较低等级的并行，但原则上与操作级并行是不同的。 指令级以上的并行是多个处理机同时对多条指令及有关的多数据组进行处理，而操作级并行是对同一条指令及其有关的多数据组进行处理。因此，前者构成的是多指令流多数据流计算机，后者构成的则是单指令流多数据流计算机。 典型的例子是多处理机。

2. 并行性开发的途径

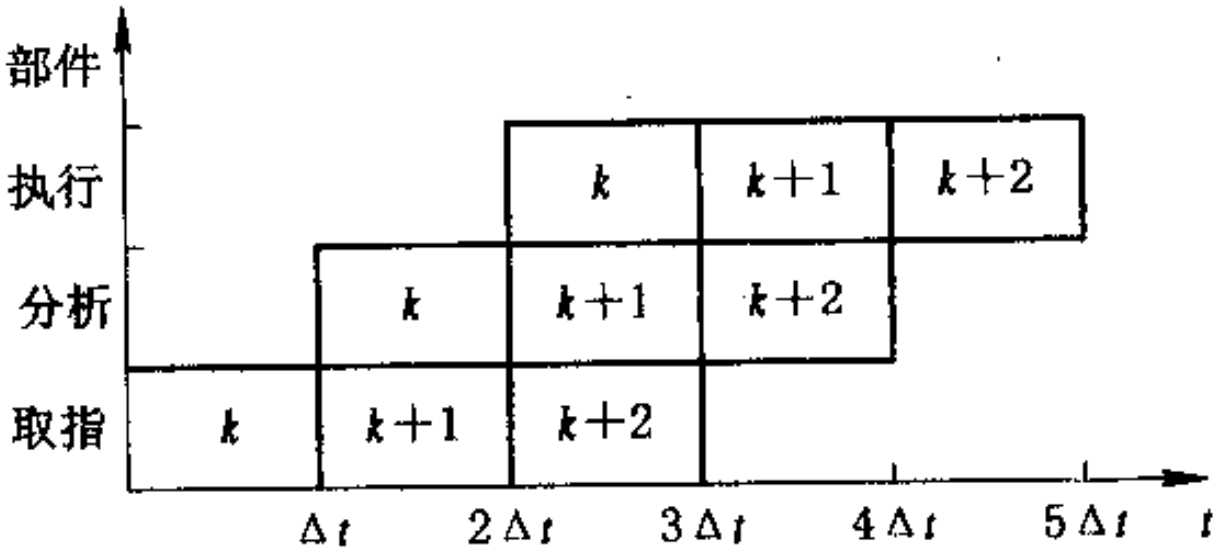
- 时间重叠
- 资源重复
- 资源共享

时间重叠

在并行性概念中引入时间因素，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，以加快硬件周期而赢得速度。



(a) 指令流水线



(b) 指令在流水线各部件中流过的时间关系

图 1.7 时间重叠的例子

资源重复

在并行性概念中引入空间因素，通过重复设置硬件资源来提高可靠性或性能。

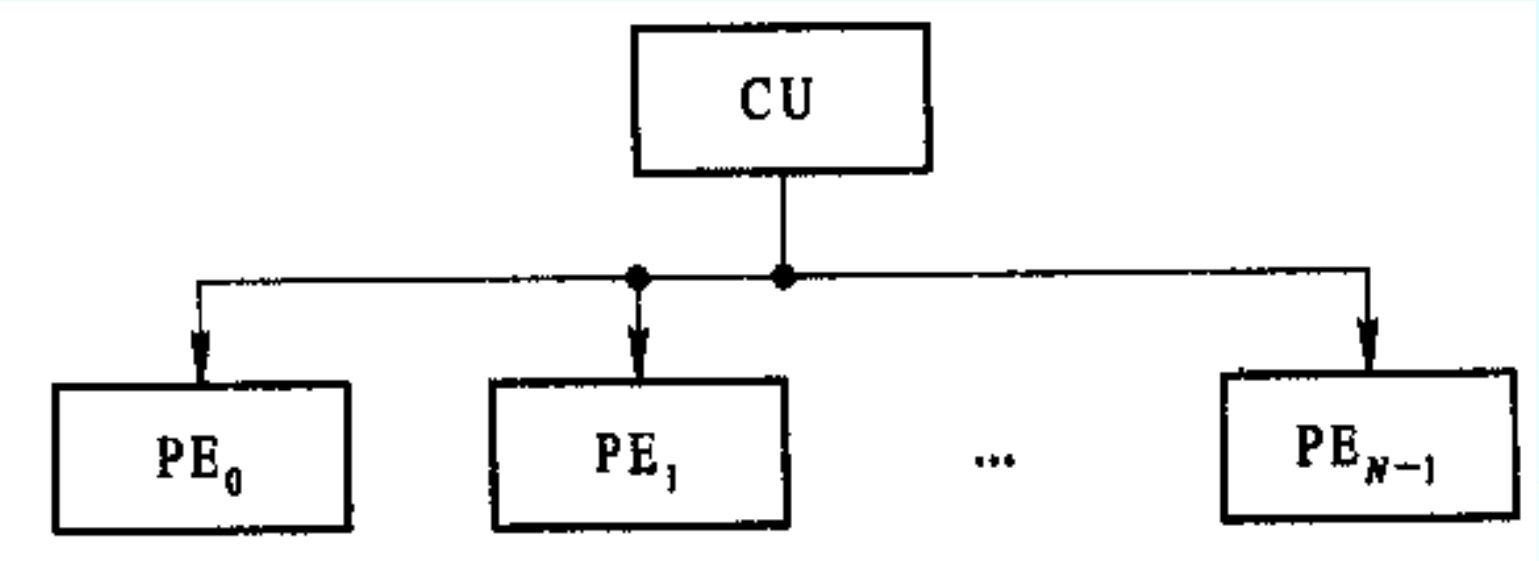


图 1.8 资源重复的例子

资源共享

利用软件的方法让多个用户按一定时间顺序轮流地使用同一套资源，以提高利用率，这样也可以提高整个系统的性能。

例如：多道程序分时系统。

再例如：共享主存、外设、通信线路的多处理机，计算机网络，以及分布处理系统。

资源共享不只限于硬件资源的共享，也包括软件、信息资源的共享。

1.6.2 并行处理系统的结构与多机系统的耦合度

1. 并行处理计算机的结构

并行处理计算机是强调并行处理的系统，除了分布处理系统外，按其基本结构特征，可以分成流水线计算机、阵列处理机、多处理机系统和数据流计算机等 4 种不同的结构。

流水线计算机主要通过时间重叠，让多个部件在时间上交错重叠地并行执行运算和处理，以实现时间上的并行。

阵列处理机主要通过资源重复，设置大量算术逻辑单元，在同一控制部件作用下同时运算和处理，以实现空间上的并行。相联处理机也可归属这一类。由于各个处理器(机)是同类型的且完成同样的功能，所以主要是一种对称、同构型多处理器(机)系统。 阵列处理机上主要要解决好处理单元间的灵活而有规律的互连模式及互连网络的设计、存储器组织、数据在存储器中的分布，以及研制对具体题目的高效并行算法等问题。

多处理机系统主要通过资源共享，让共享输入/输出子系统、数据库资源及共享或不共享主存的一组处理机在统一的操作系统全盘控制下，实现软件和硬件各级上相互作用，达到时间和空间上的异步并行。它可以改善系统的吞吐量、可靠性、灵活性和可用性。多处理机系统根据各处理机是否共享主存， 可分为紧耦合和松耦合两种不同类别。多处理机系统主要解决的问题是，处理机机间的互连、存储器组织等硬件结构，存储管理、 资源分配、任务分解、系统死锁的防止、进程间的通信和同步、多处理机的调度、系统保护等操作系统，高效并行算法和并行语言的设计等问题。

2. 多机系统的耦合度

多机系统指的是多处理机系统和多计算机系统。多处理机系统与多计算机系统是有差别的。多处理机系统是由多台处理机组成的单一计算机系统，各处理机都可有自己的控制部件，可带自己的局部存储器，能执行各自的程序，它们都受逻辑上统一的操作系统控制，处理机间以文件、单一数据或向量、数组等形式交互作用，全面实现作业、任务、指令、数据各级的并行。多计算机系统则是由多台独立的计算机组成的系统，各计算机分别在逻辑上独立的操作系统控制下运行，机间可以互不通信，即使通信也只是经通道或通信线路以文件或数据集形式进行，实现多个作业间的并行。

为了反映多机系统中各机器之间物理连接的紧密程度和交叉作用能力的强弱，引入耦合度概念。多机系统的耦合度，可以分为最低耦合、松散耦合和紧密耦合等。

各种脱机处理系统是最低耦合系统(Least Coupled System)，其耦合度最低，除通过某种中间存储介质之外，各计算机之间并无物理连接，也无共享的联机硬件资源。例如，独立外围计算机系统由主机和外围计算机组成，后者脱机工作，只通过磁带、 软盘或纸带等对主机的输入/输出提供支持。

如果多台计算机通过通道或通信线路实现互连，共享某些如磁带、磁盘等外设，则称为松散耦合系统。

如果多台计算机之间通过总线或高速开关互连，共享主存，则称为紧密耦合系统。

1.弗林（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对计算机系统进行分类。

指令流：机器执行的指令序列。

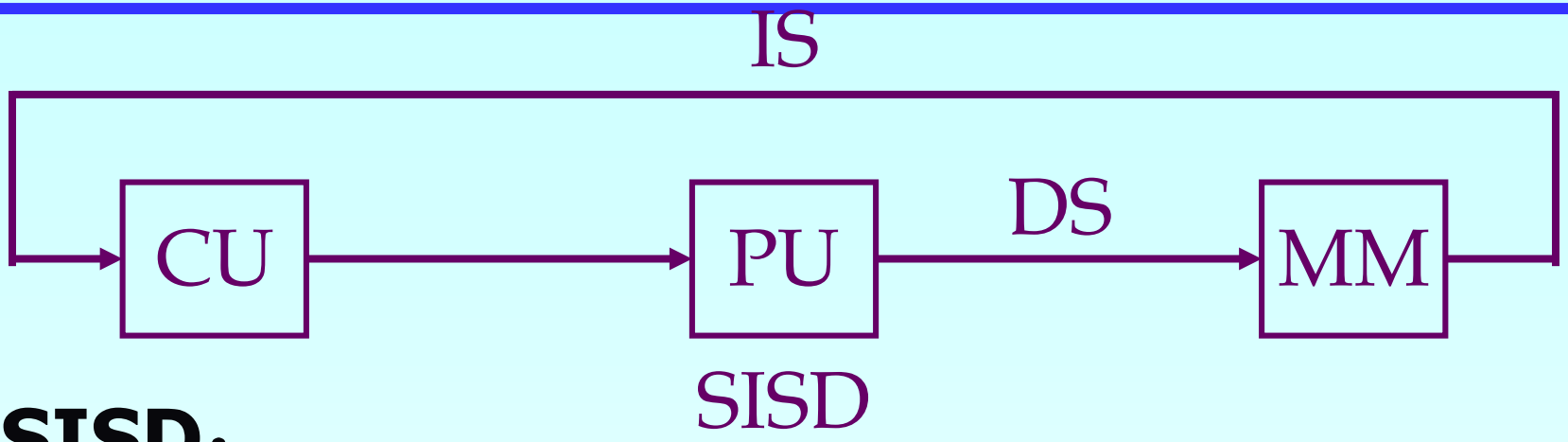
数据流：由指令流调用的数据序列，包括输入数据和中间结果。

多倍性：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。

数据流 / 指令流	单	多
单	SISD	SIMD
多	MISD	MIMD

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)



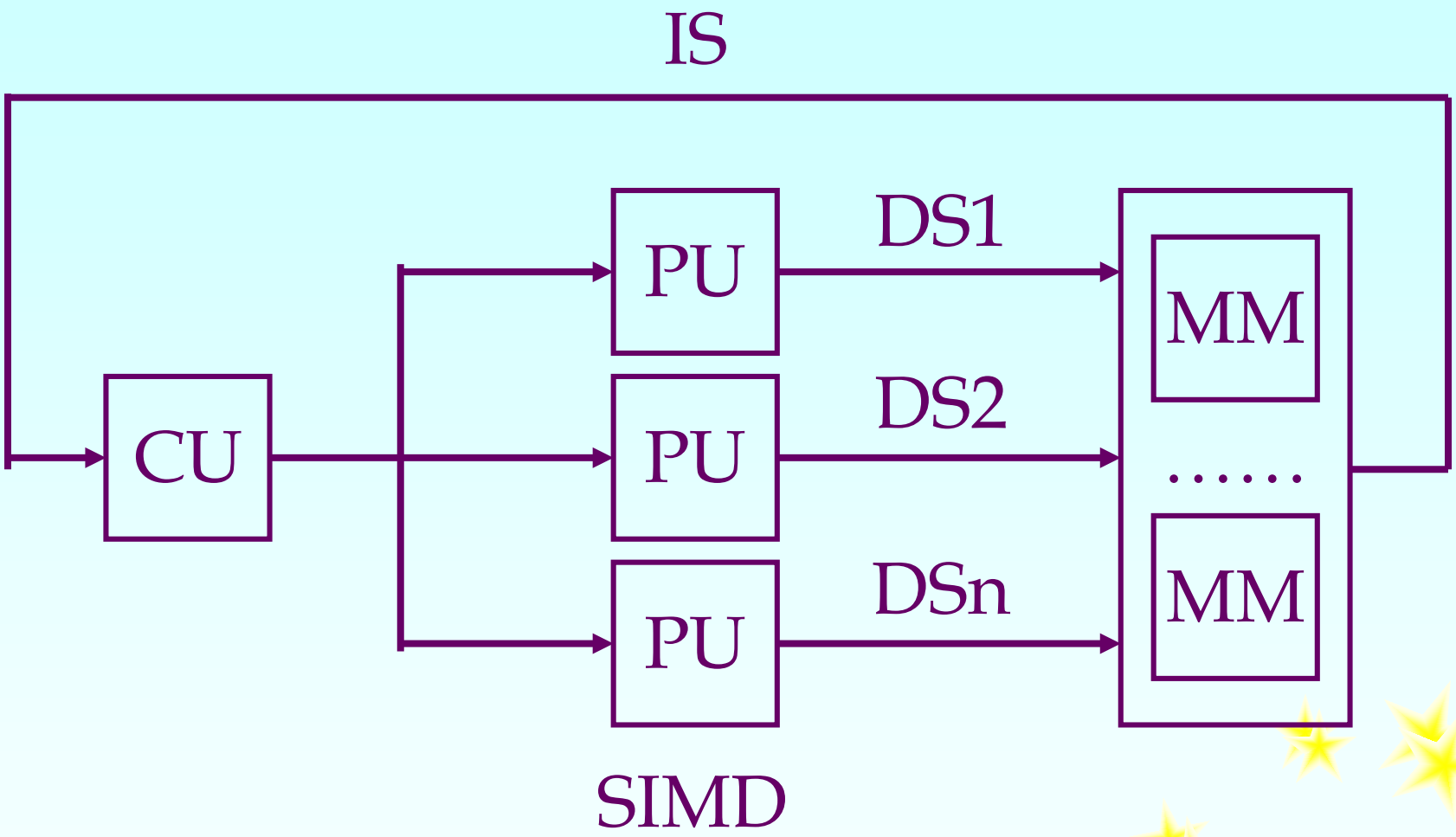
SISD：

它每次只对一条指令译码，并只对一个操作部件分配数据。

典型单处理机，包括：

单功能部件处理机：**IBM 1401，VAX-11**

多功能部件处理机：**IBM360/91，370/168，CDC6600**


IS
CU
PU
PU
PU
DS1
DS2
DSn
MM
……
MM
SIMD

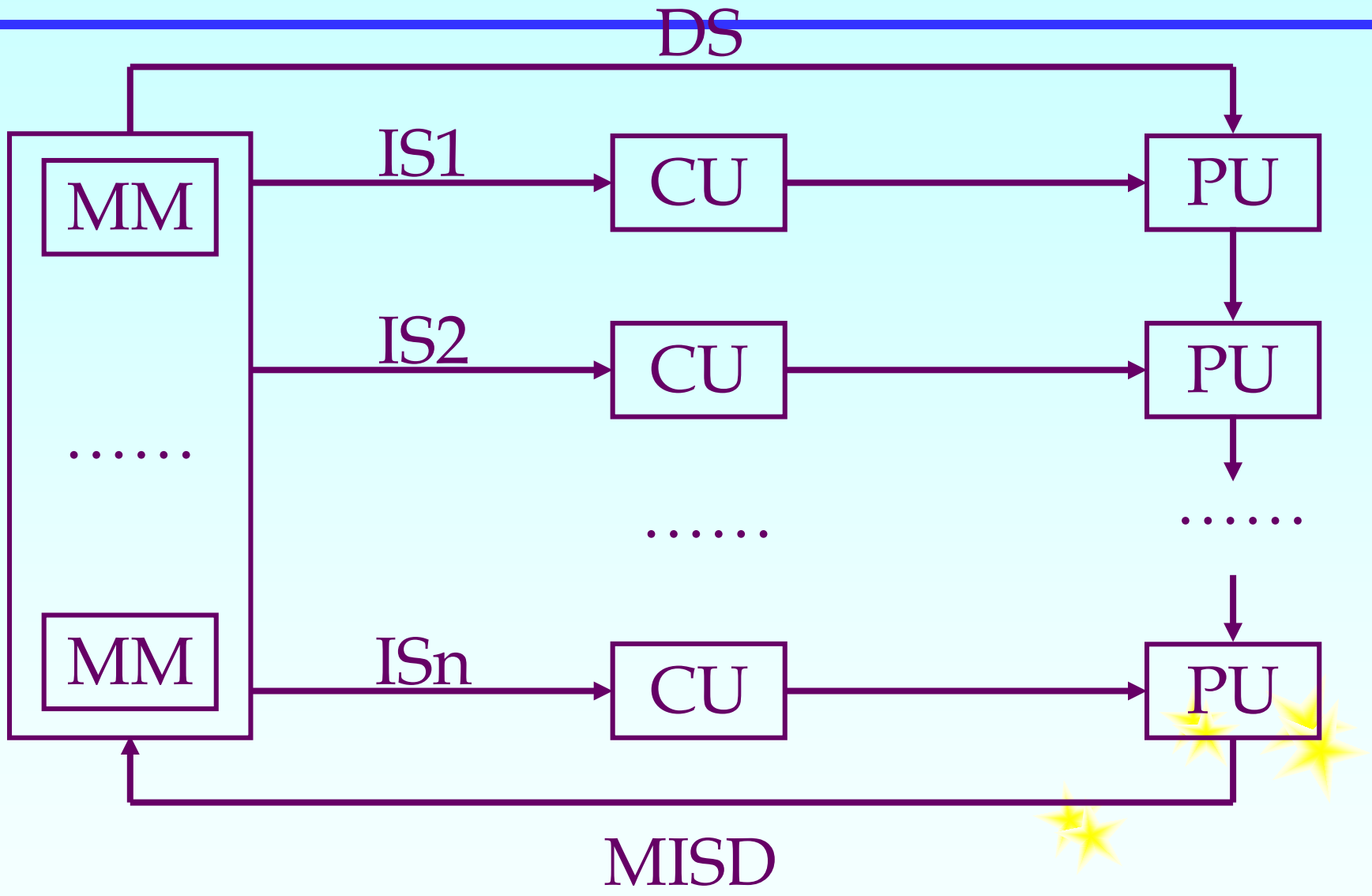
SIMD **：**

多个**PU**按一定方式互连，在同一个**CU**控制下，多个各自的数据完成同一条指令规定的操作；从**CU**看，指令顺序（串行）执行，从**PU**看，数据并行执行。

并行处理机、阵列处理机、向量处理机、相联处理机、….。

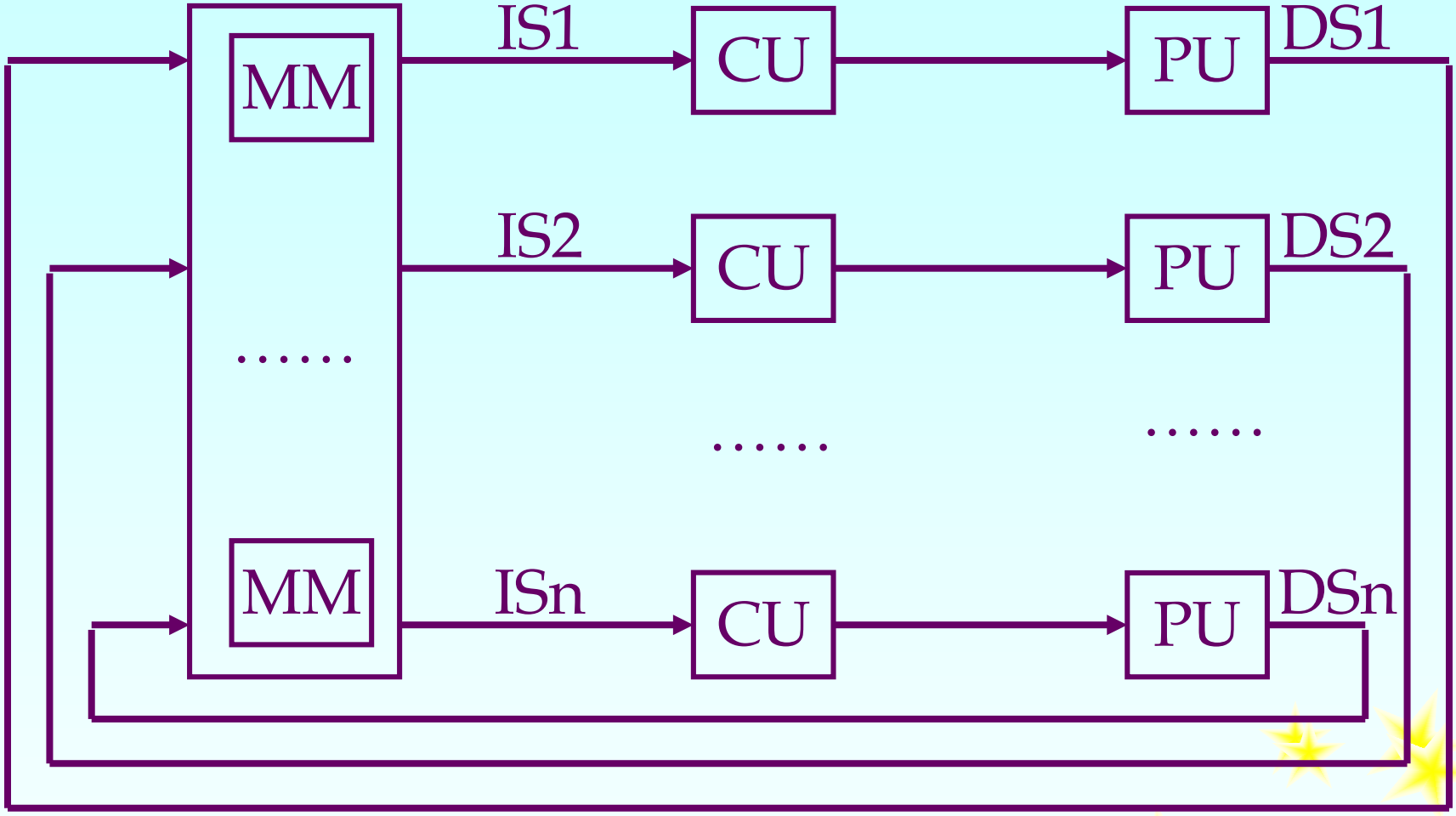
数据全并行：**ILLIAC IV**、**PEPE**、**STAR100**、**TI-ASC**、**CRAY-1**。

数据字并位串：**STARAN**、**MPP**、**DAP**。


DS
MM
IS1
CU
PU
IS2
CU
PU
……
……
……
MM
ISn
CU
PU
MISD

MISD：

几条指令对同一个数据进行不同处理，因为它要求系统在指令级上并行，而在数据级上又不并行，这是不太现实的。但现在也有些学者有不同的看法，在有些文献中将超级标量机以及超长指令字计算机等看作是**MISD**类型。



MIMD

MIMD：

能实现作业、任务、指令、数组各级全面并行的多机系统，它包括了大多数多处理机及多计算机系统。

松散耦合：**IBM3081**、**IBM3084**、**UNIVAC-1100/80**、**Cm***

紧密耦合：**D-825**、**Cmmp**、**CRAY-2**

表 按Flynn法对计算机分类举例

类型		计算机的型号
SISD	单功能部件	IBM 701，IBM 1401，IBM 1620，IBM 7090，PDP－11，VAX－11/780
	多功能部件	IBM 360/91，IBM 370/168 UP，CDC 6600
SIMD	数据全并行	ILLIAC－Ⅳ，PEPE，BSP，CDC STAR－100，TI－ASC，FPS AP－120B，FPS－164，IBM 3838，CRAY－1，CYBER 205，Fujitsu VP－200，CDC－NASF，B－5000
	数据位片串字并	STARAN，MPP，DAP
MIMD	松耦合	IBM 370/168 MP，UNIVAC 1100/80，IBM 3081/3084，Cm*
	紧耦合	Burroughs D－825，C. mmp，CRAY－2，S－1，CRAY－XMP，Denelcor HEP

Flynn分类法得到广泛应用

SISD、MIMD、SIMD…

Flynn分类法的主要缺点：

(1) 分类太粗：例如，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。

(2) 根本问题是把两个不同等级的功能 并列对待；通常，数据流受指令流控制，从而造成MISD不存在。

(3) 非冯计算机的分类？其他新型计算机的分类？

2. 库克分类法

1978年由D. J. Kuck提出。

按指令流和执行流分类。

四种类型：

(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。

主要缺点：

有些系统，如分布处理机等，没有总控制器。

分类级别太低，没有处理机级和机器级。

分类太粗，如**SIME**中包含了多种类型的处理机。

3. 冯泽云分类法

1972年美籍华人冯泽云提出。

用最大并行度来对计算机系统进行分类。

最大并行度：计算机系统在单位时间内能够处理的最大二进制位数。假设同时处理的字宽为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ti内能同时处理的二进位数为Bi，则T个时钟周期内的平均并行度定义为：

$$P_n = \frac{\sum_{i=1}^{T} B_i \cdot t_i}{T}$$

表示方法：处理机名 (n, m)

P_m=等于整数n和m确定的矩形面积。

四种类型

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

每次只处理一个字中的一位。

串行计算机：m=1，n=1；

例如：EDVAC(1，1)

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

每次只处理一个字中的n位。

传统的单处理机; m=1, n>1;

例如：Pentium(32,1)

(3) 字并位串WPBS

(Word Parallel and Bit Serial)

一次处理m个字中的1位。（位片处理）

并行计算机、MPP、相联计算机；

m>1, n=1;

例如：**MPP(1, 16384), STARAN(1, 256), DAP(1,4096)**

(4) 字并位并WPBP

(Word Parallel and Bit Parallel)

一次处理m个字，每个字为n位。

全并行计算机：**m>1, n>1;**

例如：**ILLIAC IV(64,64), ASC(64,32), PEPE(32,288), Cmmp(16,16)**

主要缺点：

仅考虑了数据的并行性，没有考虑指令、任务、作业的并行。

4. 汉德勒分类法

由Wolfgan Hindler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，计算机的硬件结构分成三个层次（程序级、操作级、逻辑级），并分别考虑它们的可并行性和流水处理程度。

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’ 表示指令流水线中算术逻辑部件的个数

w’ 表示操作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)

5. 其他分类法

如从对执行程序或指令的控制方式上分类

控制驱动的控制流方式

数据驱动的数据流方式

需求驱动的规约方式

模式驱动的匹配方式

本 章 重 点

计算机系统的层次结构

计算机系统结构的定义及研究对象

计算机系统结构、组成、实现三者的区别与相互联系。

计算机系统设计的主要方法（由中间开始设计）

Amdahl定律

CPU性能公式

解决软件可移植性的三种方法

透明性、系列机、兼容性、模拟与仿真等基本概念

系统结构中的并行性

并行性开发的途径

并行处理系统的结构和多机系统的耦合度

计算机系统的分类方法（弗林分类法）

术语解释

翻译、解释、计算机系统结构、透明性 、软件兼容、模拟、仿真、并行性、系列机、兼容机、时间重叠、紧耦合系统

作业

- 1-1
- 1-6
- 1-8
- 1-9
- 1-10
- 1-11