计算机系统结构

考试题型

客观题

选择题

填空题

问答题（术语解释）

主观题

计算题

1.1 计算机系统的多级层次结构

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

虚拟机概念

从不同角度所看到的计算机系统的属性是不同的，大部分人对计算机的认识只需要在某一个层次上。

虚拟计算机即是由软件实现的机器。

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

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）

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）

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。

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。

1.2 计算机系统结构、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。

计算机系统结构的定义

1. 定义一

Amdahl于1964年在推出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。

2.定义二

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。

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。

计算机组成

计算机组成是指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。

计算机实现

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。

计算机系统结构、组成与实现三者关系：

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。

1.3 软硬取舍与计算机系统的设计思路

软硬取舍的基本原则

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。

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。

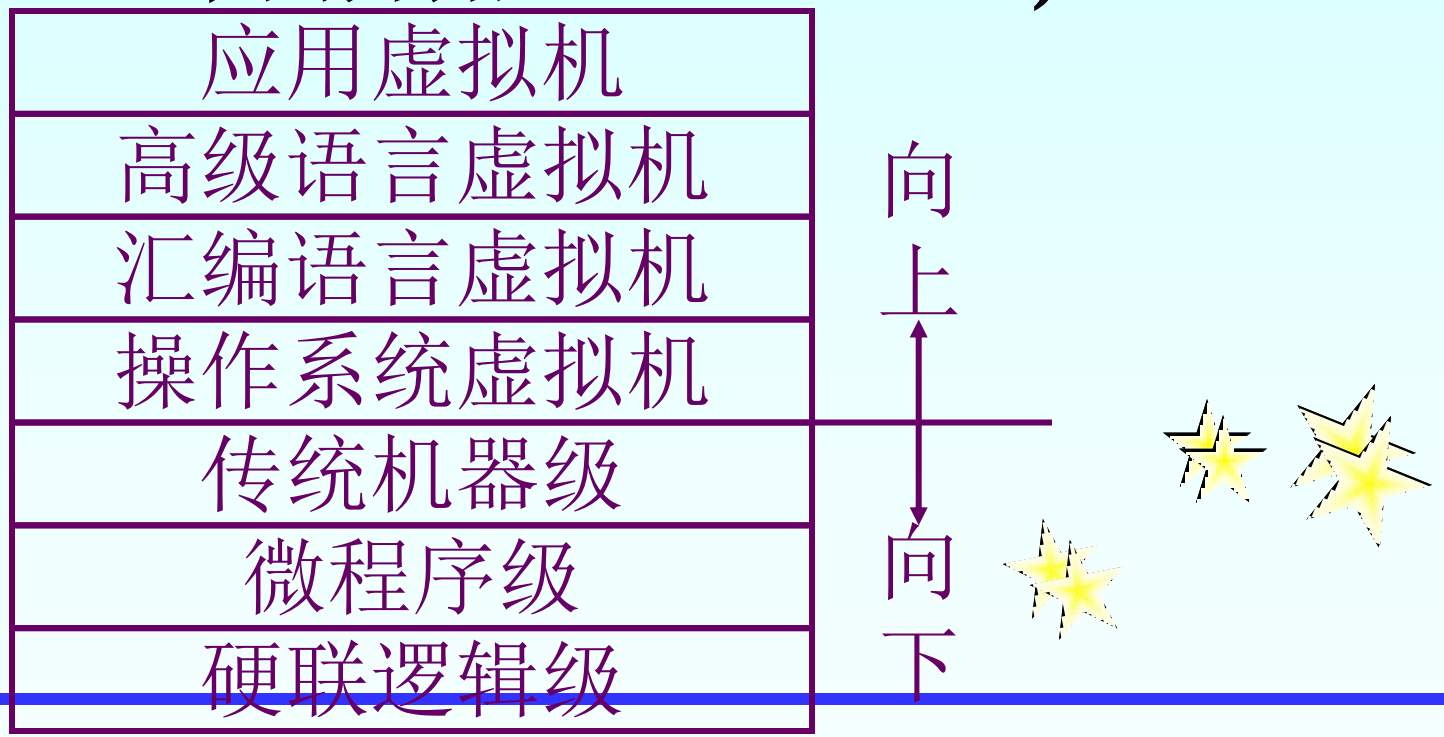
第三个基本原则是，不能仅从“硬”的角度去考虑如何便于应用组成技术的成果和发挥器件技术的进展，还应从“软”的角度把为编译和操作系统的实现，以至高级语言程序的设计提供更多更好的硬件支持放在首位。

计算机系统设计的主要方法

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)



1.4 计算机设计的量化准则

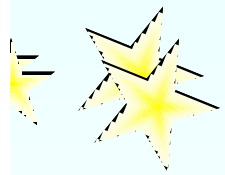
Amdahl定律

系统中某一部件由于采用某种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：

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

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



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。

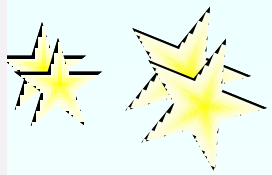
CPU性能公式

程序执行的CPU时间为：

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

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

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



系统结构的评价标准

1. 时钟频率（主频）：用于同类处理机之间。

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

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

3. 等效指令速度：吉普森（Gibson）法

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

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

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

1.5 对系统结构的影响因素

软件可移植性的定义

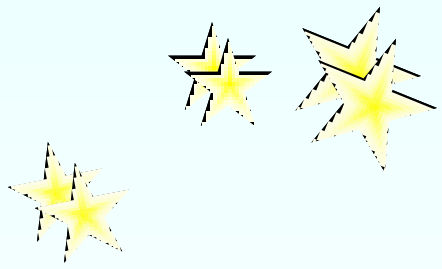
软件不用修改或只需少量加工就能由一台机器搬到另一台机器上运行，即同一软件可以应用于不同的环境。

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。

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。

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。

仿真——微程序——控存中

模拟—— 机器语言——主存中

软件移植方法区别：

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。

1.6 系统结构中的并行性及计算机系统的分类

并行性概念

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。

并行性开发的途径

- 时间重叠
- 资源重复
- 资源共享

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。

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。

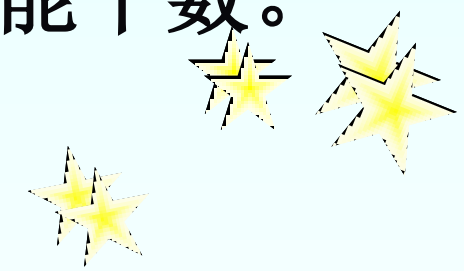
佛林（Flynn）分类法

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。



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)

2.1 数据表示

数据类型包括数据表示和数据结构。

数据表示的定义：数据表示是指计算机硬件能够直接识别，可以被指令系统直接调用的那些数据类型。

自定义数据表示

1、带标志符的数据表示法

标志符	数值

带有标志符的数据表示方式

2、数据描述符表示法

数据描述符与标志符的区别：

标志符与数据合存于一个存储单元中，用于描述单个数据的类型和属性（作用于一个数据）；而描述符则和数据分开存放，主要用于描述成块数据的特征（作用于一组数据）。

最高三位为101时表示数据描述符，最高三位为000时表示数据。

101	标志位	长度	地址

数据描述符

000	数值

数据

2.2 寻址方式

寻址方式：寻找操作数及数据存放单元的方法。

主要内容：寻址方式的设计思想和设计方法。

方法：分析各种寻址技术的优缺点，如何选择和确定寻址技术。

定位方式

程序需要定位的主要原因：程序的独立性；程序的模块化设计；数据结构在程序运行过程中，其大小往往是变化的；有些程序本身很大，大于分配给它的主存物理空间。

直接定位方式：在程序装入主存储器之前，程序中的指令和数据的主存物理地址就已经确定了的称为直接定位方式。

静态定位：在程序装入主存储器的过程中随即进行地址变换，确定指令和数据的主存物理地址的称为静态定位方式。

动态定位：在程序执行过程中，当访问到相应的指令或数据时才进行地址变换，确定指令和数据的主存物理地址的称为动态定位方式。

2.3 指令系统的设计和优化

主要目标：

节省程序的存储空间

指令格式尽量规整，便于译码

研究内容：

操作码的优化表示；地址码的优化表示

操作码的优化表示

操作码的三种编码方法：

固定长度，Huffman编码、扩展编码

Huffman编码法

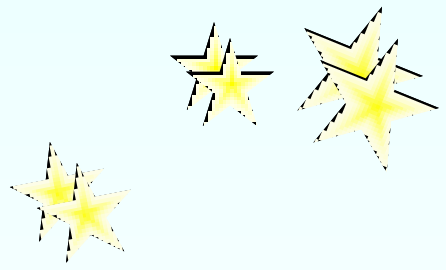
操作码的最短平均长度（理想情况），又称信息源熵，可通过下式计算：

$$H = -\sum_{i=1}^{n} p_i \cdot \log_2 p_i$$

其中：Pi表示第i种操作码在程序中出现的概率。

信息冗余量：

$$R = 1 - \frac{H}{\text{实际平均码长}}$$



Huffman编码的具体码值不惟一，但平均码长肯定是惟一的。

Huffman操作码的主要缺点：

操作码长度很不规整，硬件译码困难。

与地址码共同组成固定长的指令比较困难。

扩展编码法

由固定长操作码与Huffman编码法相结合形成的一种编码方式，操作码长度被限定使用有限的几种码长，仍体现高概率指令用短码，低概率指令用长码的思想，使操作码的平均码长虽大于Huffman编码，但小于等长编码，是一种实际可用的优化编码方法。

2.4 指令系统的发展和改进

进一步增强原有指令的功能以及设置更为复杂的新指令取代原先由软件子程序完成的功能，实现软件功能的硬化。复杂指令系统计算机，简称**CISC**。

通过减少指令种类和简化指令功能来降低硬件设计的复杂度，提高指令的执行速度。精简指令系统计算机，简称**RISC**。

指令系统的优化设计

有两个截然相反的方向：

1. **复杂指令系统计算机CISC**
   (Complex Instruction Set Computer)

   增强指令功能，设置功能复杂的指令
   面向目标代码、高级语言和操作系统
   用一条指令代替一串指令

2. **精简指令系统计算机RISC**
   (Reduced Instruction Set Computer)

   只保留功能简单的指令
   功能较复杂的指令用子程序来实现

CISC指令系统存在的问题：

(1) 指令系统庞大。

(2) 指令执行速度低。

(3) 编译程序本身太长、太复杂。

(4) 各种指令使用频度都不会太高，且差别很大。

20%与80%律

RISC的定义与特点

卡内基梅隆大学（Carnegie Mellon）论述

RISC的特点：

1、大多数指令在单周期内完成
2、LOAD/STORE结构
3、硬布线控制逻辑
4、减少指令和寻址方式的种类
5、固定的指令格式
6、注重编译优化技术

90年代初，IEEE的Michael Slater对RISC定义的描述：

1、RISC为使流水线高效率执行，应具有：

简单而统一格式的指令译码
大部分指令可以单周期执行完成
仅Load和Store指令可以访问存储器
简单的寻址方式
采用延迟转移技术
采用LOAD延迟技术

2、RISC为使优化编译器便于生成优化代码，应具有：三地址指令格式、较多的寄存器、对称的指令格式。

RISC的关键技术

1、延时转移技术

定义：为了使指令流水线不断流，在转移指令之后插入一条不相关的有效的指令，而转移指令被延迟执行，这种技术称为延迟转移技术。

采用指令延迟转移技术时，指令序列的调整由编译器自动进行。

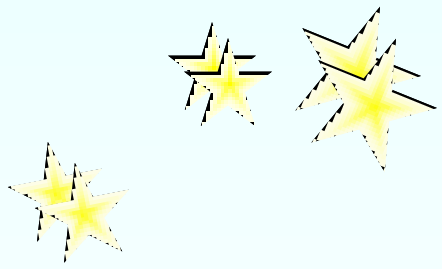
2、指令取消技术

采用指令延时技术，在许多情况下找不到可以用来调整的指令，故有些RISC采用指令取消技术，分为三种情况：

(1) 向后转移（循环程序）

(2) 向前转移(if-then)

(3) 隐含转移技术



3、重叠寄存器窗口技术（Overlapping Register Window）

原因：RISC中，子程序比CISC中多, 因传送参数而访问存储器的信息量很大。

重叠寄存器窗口技术由美国加洲大学伯克利分校的F .Baskett提出。

实现方法：

设置一个数量比较大的寄存器堆，并把它划分成很多个窗口。在每个过程使用的几个窗口中有一个窗口是与前一个过程共用，还有个窗口是与下一个过程共用。

4、指令流调整技术

目标：通过变量重新命名消除数据相关，提高流水线执行效率。

5、以硬件为主固件为辅

固件的主要缺点是：执行速度低。

主要优点是：便于实现复杂指令，便于修改指令系统。

以硬联逻辑为主来实现指令系统。

对于复杂指令，也使用微程序技术实现。

3.1 输入输出系统概述

输入输出系统的特点

输入输出系统涉及到机、光、电、磁、声、自动控制等多种学科。

用户无需了解输入输出系统和输入输出设备的具体细节就能使用输入输出设备。

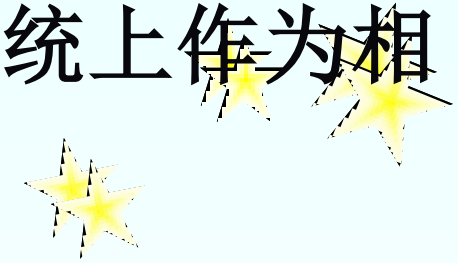
处理机的外部世界包括：本地和远程用户、系统操作员、操作控制台、输入输出设备、辅助存储器、其它处理机、各种通信设备和虚拟现实系统等。

1、异步性

2、实时性

3、与设备无关性

输入输出系统的发展经历了 3 个阶段，对应于 3 种方式，即程序控制输入输出(包括全软的、程序查询状态驱动的、中断驱动的几种)、直接存储器访问(DMA)和I/O处理机方式。这 3 种方式可以分别用在不同的计算机系统上，也可以用在同一个计算机系统上作为相互补充。



3.2 磁盘阵列

RAID是Redundent Array of Inexpensive Disks的缩写，直译为“廉价冗余磁盘阵列”，也简称为“磁盘阵列”。后来RAID中的字母I被改作为Independent，RAID就成了“独立冗余磁盘阵列”。

RAID级别	名称	数据磁盘数	可正常工作的最多失效磁盘数	检测磁盘数
RAID0	无冗余无校验的磁盘阵列	8	0	0
RAID1	镜象磁盘阵列	8	1	8
RAID2	纠错海明码磁盘阵列	8	1	4
RAID3	位交叉奇偶校验的磁盘阵列	8	1	1
RAID4	块交叉奇偶校验的磁盘阵列	8	1	1
RAID5	无独立校验盘的奇偶校验磁盘阵列	8	1	1
RAID6	双维无独立校验盘的奇偶校验磁盘阵列	8	2	2

3.3 总 线 设 计

总线的类型

就允许信息传送的方向来说，总线可以有单向传输和双向传输两种。双向传输又有半双向和全双向的不同。

总线按其用法可以分成专用的和非专用的。

总线的控制方式

集中式控制

分布式总线控制

优先次序裁决

(1)链式查询方式

(2)计数器定时查询方式

(3)独立请求方式

总线的通信技术

同步通信

异步通信

“数据宽度”指的是I/O设备取得I/O总线使用权后所传送数据的总量。

数据通路宽度指的是数据传送的物理宽度，即一个时钟周期所传送的信息量，它直接取决于数据总线的线数。

3.4 通道处理机

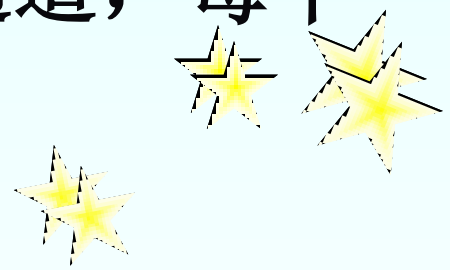
通道种类

通道分为三种类型：字节多路通道、选择通道和数组多路通道。

1、字节多路通道

为多台低速或中速的外围设备服务。

字节多路通道包含有多个子通道，每个子通道连接一个设备控制器。



2、选择通道

选择通道为高速外围设备服务。

每个选择通道只有一个以成组方式工作的子通道，逐个为多台高速外围设备服务。

3、数组多路通道

数组多路通道：把字节多路通道和选择通道的特性结合起来。

每次为一台高速设备传送一个数据块，并轮流为多台外围设备服务。

数组多路通道可以被看作是以成组方式工作的高速多路通道。

从磁盘存储器读出一个文件的的过程分为三步：定位、找扇区、读出数据。

通道流量分析

通道流量：单位时间内能够传送的最大数据量。又称为通道吞吐率，通道数据传输率等。

通道最大流量：通道在满负荷工作状态下的流量。

三种通道的最大流量计算公式如下：

$$f_{MAX.BYTE} = \frac{p \cdot n}{(T_S + T_D) \cdot p \cdot n} = \frac{1}{T_S + T_D} \text{字节/秒}$$

$$f_{MAX.SELETE} = \frac{p \cdot n}{(T_S / n + T_D) \cdot p \cdot n} = \frac{1}{T_S / n + T_D} \text{字节/秒}$$

$$f_{MAX.BLOCK} = \frac{p \cdot n}{(T_S / k + T_D) \cdot p \cdot n} = \frac{1}{T_S / k + T_D} \text{字节/秒}$$

通道流量与连接在这个通道上的所有设备的数据传输率的关系如下：

$$f_{BYTE}=\sum_{i=1}^{p} fi \qquad f_{SELETE}=\max_{i=1}^{p} fi \qquad f_{BLOCK}=\max_{i=1}^{p} fi$$

为了保证通道能够正常工作，不丢失数据，各种通道实际流量应该不大于通道最大流量，即满足下列不等式关系：

$f_{BYTE} \le f_{MAX\ BYTE}$，

$f_{SELETE} \le f_{MAX\ SELETE}$，

$f_{BLOCK} \le f_{MAX\ BLOCK}$

4.1 存储体系的概念和并行存储系统

存储器的主要性能：速度、容量、价格

速度用存储器的访问周期、读出时间、频带宽度等表示。

容量用字节B、千字节KB、兆字节MB和千兆字节GB等单位表示。

价格用单位容量的价格表示，如$/bit。

存储系统的关键是如何组织好速度、容量和价格均不相同的存储器，使这个存储器的速度接近速度最快的那个存储器，存储容量与容量最大的那个存储器相等，单位容量的价格接近最便宜的那个存储器。

1、并行访问存储器

方法：把m字w位的存储器改变成为m/n字n×w位的存储器。

逻辑实现：

把地址码分成两个部分，一部分作为存储器的地址，另一部分负责选择数据。

主要缺点：访问冲突大

(1) 取指令冲突
(2) 读操作数冲突
(3) 写数据冲突
(4) 读写冲突

2、高位交叉访问存储器

主要目的：扩大存储器容量

实现方法：用地址码的高位区分存储体号

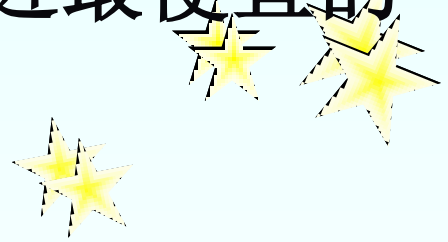
3、低位交叉访问存储器

主要目的：提高存储器访问速度

实现方法：用地址码的低位区分存储体号

存储系统(存储体系、存储层次)的定义

两个或两个以上速度、容量和价格各不相同的存储器用硬件、软件、或软件与硬件相结合的方法连接起来成为一个存储系统。这个系统对应用程序员透明，并且，从应用程序员看，它是一个存储器，这个存储器的速度接近速度最快的那个存储器，存储容量与容量最大的那个存储器相等，单位容量的价格接近最便宜的那个存储器。



主存—辅存层次

又称虚拟存储系统，由主存储器和磁盘存储器构成。

主要目的：扩大存储器容量，弥补主存容量的不足。

在主存和辅存之间，增加辅助的软硬件，让它们构成一个整体。从CPU看，速度接近主存的速度，容量是虚拟的地址空间，每位价格是接近于辅存的价格。由于虚拟存储系统需要通过操作系统来调度，因此对系统程序员是不透明的，但对应用程序员是透明的。

Cache－主存层次：

又称**Cache**存储系统，由**Cache**和主存储器构成。

主要目的：提高存储器速度，弥补主存速度的不足。

在**Cache**和主存之间，增加辅助硬件，让它们构成一个整体。从**CPU**看，速度接近**Cache**的速度，容量是主存的容量，每位价格接近于主存的价格。由于**Cache**存储系统全部用硬件来调度，因此它对系统程序员和应用程序员都是透明的。

程序局部性原理

局部性分为时间上的局部性和空间上的局部性。时间上的局部性是指最近访问的代码是不久将被访问的代码，这是由程序循环造成的。空间上的局部性是指那些地址上相邻近的代码可能会被一起访问，这主要是由于指令通常是顺序执行的，以及数据一般是以向量、阵列等形式簇聚地存储所致。

所以，程序在执行时所用到的指令和数据的地址分布不会是随机的，而是相对簇聚的。

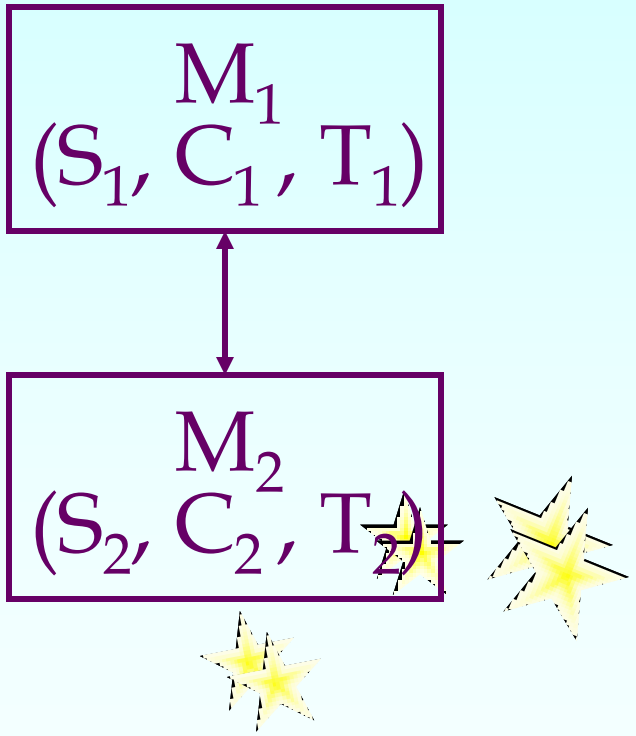
存储系统的单位容量平均价格

计算公式：

$$C = \frac{C_1 \cdot S_1 + C_2 \cdot S_2}{S_1 + S_2}$$

S2>>S1时, C≈C2, 但

S2与S1不能相差太大



存储系统的速度

表示方法：访问周期、存取周期、存储周期、存取时间、读出时间等。

命中率定义：在M1存储器中访问到的概率。

$$H = \frac{N_1}{N_1 + N_2}$$

N_1: M_1的访问次数

N_2: M_2的访问次数

访问周期与命中率的关系：

同时启动时：$T = HT_1 + (1-H)T_2$

不同时启动时：$T = T_1 + (1-H)T_2$

当命中率H→1时，T→T1

存储系统的访问效率：

$$e=\frac{T_1}{T}=\frac{T_1}{H\cdot T_1+(1-H)\cdot T_2}=\frac{1}{H+(1-H)\cdot\frac{T_2}{T_1}}=f(H,\frac{T_2}{T_1})$$

存储系统的访问效率主要与命中率和两级存储器的速度之比有关。

采用预取技术提高命中率

方法：不命中时，把M2存储器中相邻几个单元组成的一个数据块都取出来送入M1存储器中。

计算公式：

$$H' = \frac{H + n - 1}{n}$$

其中：H’是采用预取技术后的命中率；H是原来的命中率；n为数据块大小与数据重复使用次数的乘积。

提高存储系统速度的途径：

一是提高命中率**H**。

二是两个存储器的速度不要相差太大。

三加快内部地址映像及变换

4.2 虚拟存储器

虚拟存储器工作原理

虚拟存储器由主存储器和联机工作的外存储器（磁盘存储器）共同组成的。

把主存储器和虚拟存储器都划分成固定大小的页，主存储器的页称为实页，虚拟存储器中的页称为虚页。

一个**主存地址A**由两部分组成，实页号p和页内偏移d。

实页号p	页内偏移d

主存地址A的组成

一个**虚地址Av**由三部分组成，用户号U、虚页号P和页内偏移D。

用户号U	虚页号P	页内偏移D

多用户虚拟地址Av的组成

内部地址变换：

多用户虚拟地址Av变换成主存实地址A。

多用户虚拟地址中的页内偏移D直接作为主存实地址中的页内偏移d。

主存实页号p与它的页内偏移d直接拼接起来就得到主存实地址A。

外部地址变换：

首先查外页表得到磁盘存储器实地址。

把磁盘存储器实地址和主存储器实页号送入输入输出处理机。

把要访问数据所在的一整页都从磁盘存储器调入到主存储器。

地址的映像与变换

三种地址空间：虚拟地址空间，主存储器地址空间，辅存地址空间

地址映像：
把虚拟地址空间映像到主存地址空间

地址变换：在程序运行时，把虚地址变换成主存实地址

因地址映像和变换方法不同，有三种虚拟存储器：页式虚拟存储器、段式虚拟存储器、段页式虚拟存储器。

加快内部地址变换的方法

造成虚拟存储器速度降低的主要原因：

(1) 要访问主存储器须先查段表或页表

(2) 可能需要多级页表

页表级数的计算公式：

$$g = \left\lceil \frac{\log_2 Nv - \log_2 Np}{\log_2 Np - \log_2 Nd} \right\rceil$$

其中：

Np为页面的大小

Nv为虚拟存储空间大小

Nd为一个页表存储字的大小

页面替换算法的使用场合：

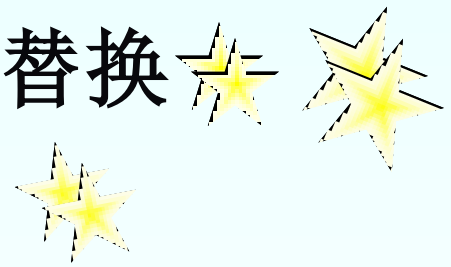
(1) 虚拟存储器中，主存页面的替换，一般用软件实现

(2) Cache块替换一般用硬件实现

(3) 虚拟存储器的快慢表中，快表存储字的替换，用硬件实现

(4) 虚拟存储器中，用户基地址寄存器的替换，用硬件实现

(5) 在有些虚拟存储器中目录表的替换



1、页面替换算法

(1) 随机算法(RAND Random algorithm)：

算法简单，容易实现；没有利用历史信息，没有反映程序的局部性，命中率低。

(2) 先进先出算法 (FIFO First-In First-Out algorithm)：

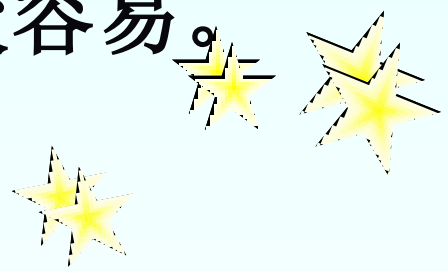
比较容易实现，利用了历史信息，没有反映程序的局部性。最先调入主存的页面，很可能也是经常要使用的页面。

(3) 近期最少使用算法 (LFU Least Frequently Used algorithm)：

既充分利用了历史信息，又反映了程序的局部性，实现起来非常困难。

(4) 最久没有使用算法 (LRU Least Recently Used algorithm)：

把LFU算法中的“多”与“少”简化成“有”与“无”，实现起来比较容易。



(5) 最优替换算法 (OPT OPTimal replacemant algorithm):

是一种理想化的算法。用来作为评价其它页面替换算法好坏的标准。

在虚拟存储器中，实际上有可能采用只有**FIFO**和**LRU**两种算法。

1、目录表

基本思想：用一个小容量高速存储器存放页表。

2、快慢表

快表TLB(Translation Lookaside Buffer)：小容量(几～几十个字)，高速硬件实现，采用相联方式访问。

慢表：当快表中查不到时，从存放在主存储器中的慢表中查找按地址访问，用软件实现。

快表与慢表也构成了一个两级存储系统。

3、散列函数

目的：把相联访问变成按地址访问，从而加大快表容量。

采用散列变换实现快表按地址访问

避免散列冲突：采用相等比较器

地址变换过程：相等比较与访问存储器同时进行。

4.3 高速缓冲存储器 (Cache)

地址映像：

把存放在主存中的程序按照某种规则装入到Cache中，并建立主存地址与Cache地址之间的对应关系。

地址变换：

当程序已经装入到Cache之后，在实际运行过程中，把主存地址变换成Cache地址。

在选取地址映像方法要考虑的主要因素：

地址变换的硬件容易实现；地址变换的速度要快；主存空间利用率要高；发生块冲突的概率要小。

1、全相联映像及其变换

映像规则：主存中的任意一块都可以映像到Cache中的任意一块。

如果Cache的块数为C_b，主存的块数为M_b，映像关系共有：$C_b \times M_b$种。

用硬件实现非常复杂。

在虚拟存储器中，全部用软件实现。

2、直接映像及其变换

映像规则： 主存中一块只能映像到Cache的一个特定的块中。

计算公式： b＝B mod Cb，其中：

b为Cache的块号，

B是主存的块号，

Cb是Cache的块数。

整个Cache地址与主存地址的低位部分完全相同。

3、组相联映像及其变换

组相联映像实际上是全相联映像和直接映像的折衷方案，所以其优点和缺点介于全相联和直接映像方式的优缺点之间。

映像规则（位选择映像算法）：

主存和Cache按同样大小划分成块；

Cache划分成大小相同的组，主存按照Cache组容量分区。主存每个分区中的块容量与Cache的组容量相等。

主存块到Cache组之间采用直接映像方式。在对应的组内部采用全相联映像方式，组内随便放。

Cache替换算法及其实现

Cache替换算法使用的时间：

发生块失效，且可以装入新调入块的几个Cache块都已经被装满时。

直接映像方式实际上不需要替换算法。

全相联映像方式的替换算法最复杂。

Cache的一致性问题

本节讨论的内容仅限于单处理机、单存储器。

造成Cache与主存的不一致的原因：

(1) 由于CPU写Cache，没有立即写主存。

(2) 由于IO处理机或IO设备写主存。

Cache的更新算法：

(1) 写直达法(写通过法), Write-through

CPU在执行写操作时，把数据同时写入Cache和主存。

(2) 写回法 (抵触修改法)Write-Back

CPU数据只写入Cache，不写入主存，仅当替换时，才把修改过的Cache块写回到主存。

写Cache的两种方法：

(1) 不按写分配法：在写Cache不命中时，只把所要写的字写入主存。

(2) 按写分配法：在写Cache不命中时，还把一个块从主存读入Cache。

目前，在写回法中采用按写分配法，在写直达法中采用不按写分配法。

Cache的预取算法

预取算法有如下几种：

(1) 按需取：在出现Cache不命中时，把一个块取到Cache中来。

(2) 恒预取：无论Cache是否命中，都把下一块取到Cache中。

(3) 不命中预取：当Cache不命中，把本块和下一块取到Cache中。

主要考虑因素：

命中率的提高；

Cache与主存之间通信量的增加。

5.1 重叠方式

指令的重叠执行方式

1、顺序执行方式

执行n条指令所用的时间为：

$$T=\sum_{i=1}^{n}(t_{取指令i}+t_{分析i}+t_{执行i})$$

如果每段时间都为t，则执行n条指令所用的时间为： **T=3nt**

取指k	分析k	执行k	取指k+1	分析k+1	执行k+1

主要优点：

控制简单，节省设备。

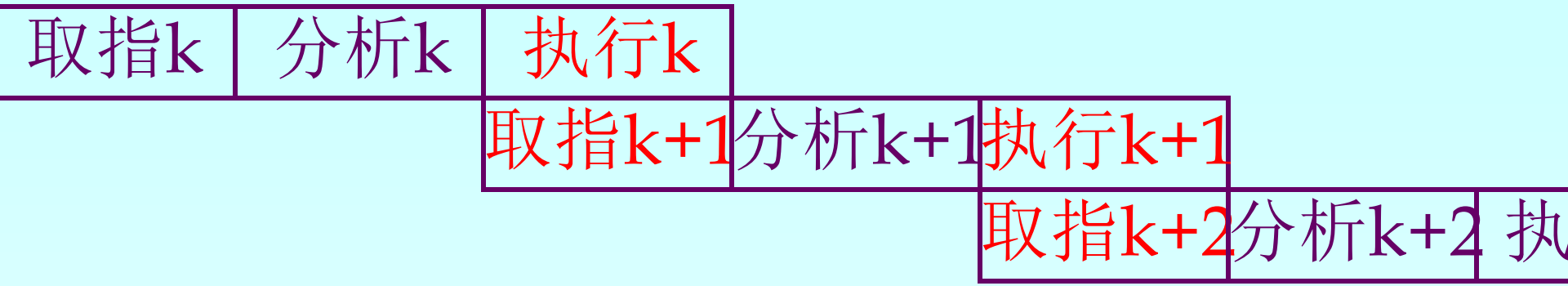
主要缺点：

执行指令的速度慢，功能部件的利用率很低。

2、一次重叠执行方式

一种最简单的流水线方式。

如果两个过程的时间相等，则执行n条指令的时间为：$T=(1+2n)t$



主要优点：

指令的执行时间缩短。

功能部件的利用率明显提高。

主要缺点：

需要增加一些硬件。

控制过程稍复杂。

3、二次重叠执行方式

如果三过程的时间相等，执行n条指令的时间为：$T=(2+n)t$

理想情况下同时有三条指令在执行。

处理机的结构要作比较大的改变，必须采用先行控制方式。

先行控制方式的原理

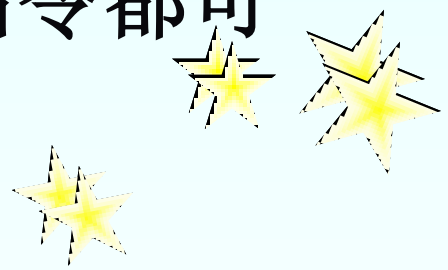
1、采用二次重叠执行方式，必须解决两个问题：

(1) 有独立的取指令部件、指令分析部件和指令执行部件。

独立的控制器：存储控制器、指令控制器、运算控制器。

(2) 要解决访问主存储器的冲突问题

取指令、分析指令、执行指令都可能要访问存储器。



2、解决访存冲突的方法：

(1) 采用低位交叉存取方式：

这种方法不能根本解决冲突问题。

取指令、读操作数、写结果。

(2) 两个独立的存储器：独立的指令存储器和数据存储器。

如果再规定，执行指令所需要的操作数和执行结果只写到通用寄存器，那么，取指令、分析指令和执行指令就可以同时进行。

在许多高性能处理机中，有独立的指令**Cache**和数据**Cache**。这种结构被称为哈佛结构。

(3) 采用先行控制技术

先行控制技术的关键是缓冲技术和预处理技术。

缓冲技术是在工作速度不固定的两个功能部件之间设置缓冲栈，用以平滑它们的工作。

在采用了缓冲技术和预处理技术之后，运算器能够专心于数据的运算，从而大幅度提高程序的执行速度。

相关处理方法：

推后读

设置相关专用通路

5.2 流水方式

空间并行性：

设置多个独立的操作部件。

多操作部件处理机。

超标量处理机。

时间并行性：

采用流水线技术。

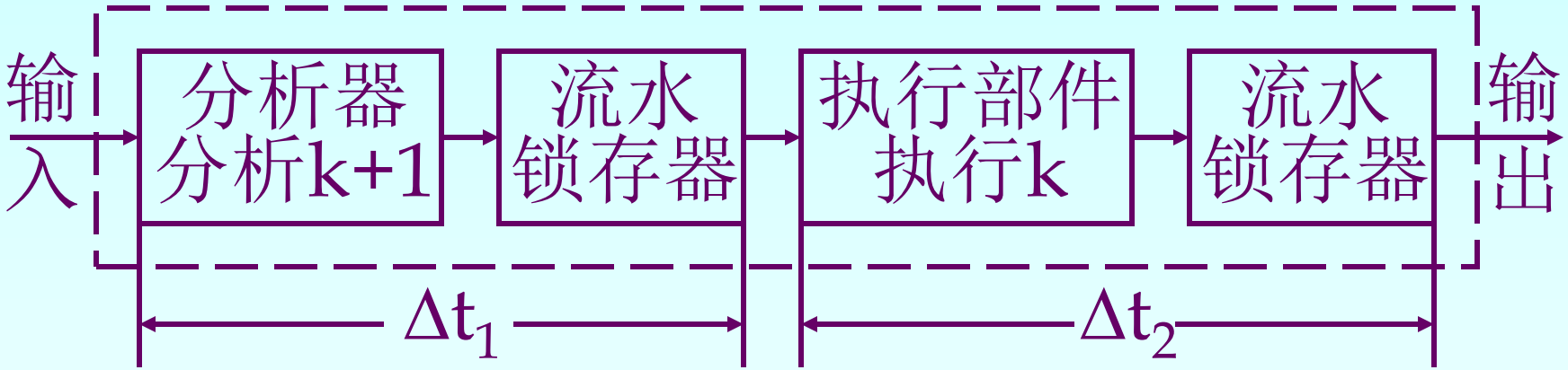
不增加或只增加少量硬件就能使运算速度提高几倍。

流水线处理机。

超流水线处理机。

流水线工作原理

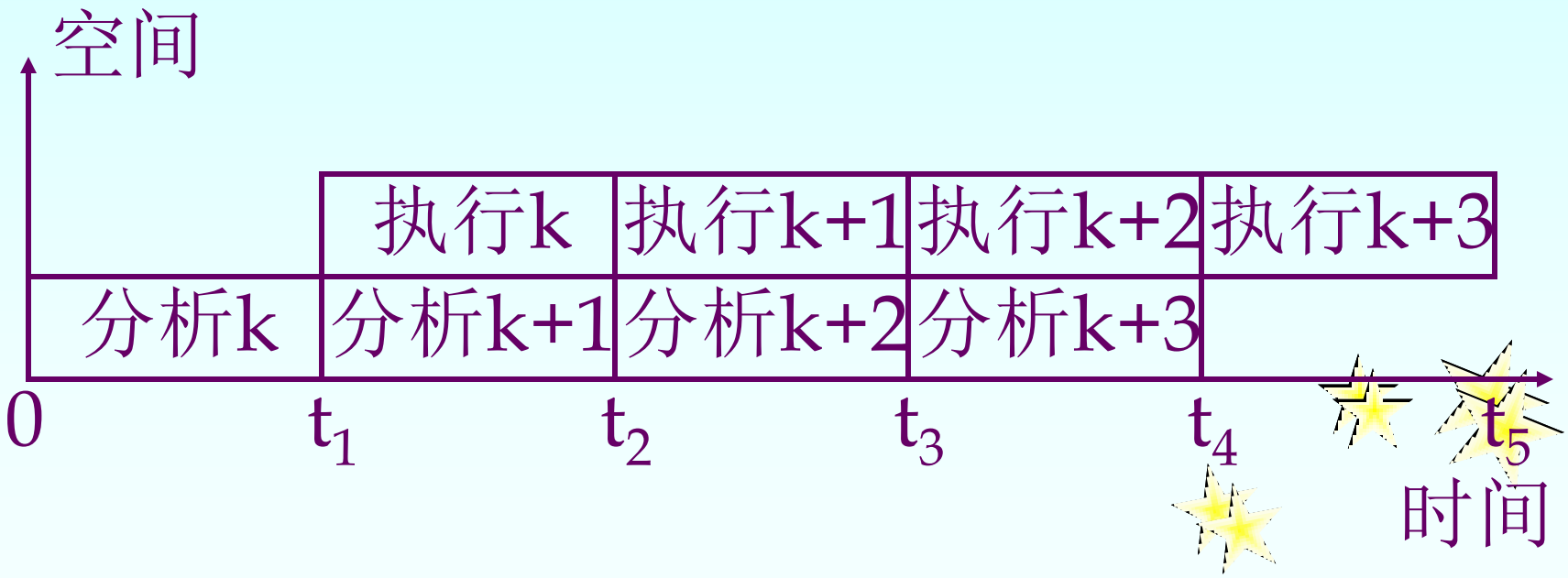
简单流水线



流水线的每一个阶段称为流水步、流水步骤、流水段、流水线阶段、流水功能段、功能段、流水级、流水节拍等。

流水线的时空图

一条简单流水线的时空图：



流水线的主要特点

(1)只有连续提供同类任务才能充分发挥流水线的效率

对于指令流水线：要尽量减少因条件分支造成的“断流”。

对于操作部件：主要通过编译技术，尽量提供连续的同类操作。

(2)在流水线的每一个流水线段中都要设置一个流水锁存器

时间开销：流水线的执行时间加长

是流水线中需要增加的主要硬件之一。

(3)各流水段的时间应尽量相等

流水线处理机的基本时钟周期等于时间最长的流水段的时间长度。

(4)流水线需要有“装入时间”和“排空时间”

流水线的分类

1、线性流水线与非线性流水线

流水线的各个流水段之间是否有反馈信号

线性流水线(Linear Pipelining)

每个流水段都流过一次，且仅流过一次。

非线性流水线(Nonlinear Pipelining)

在流水线的某些流水段之间有反馈回路或前馈回路。

线性流水线能够用流水线连接图唯一表示。

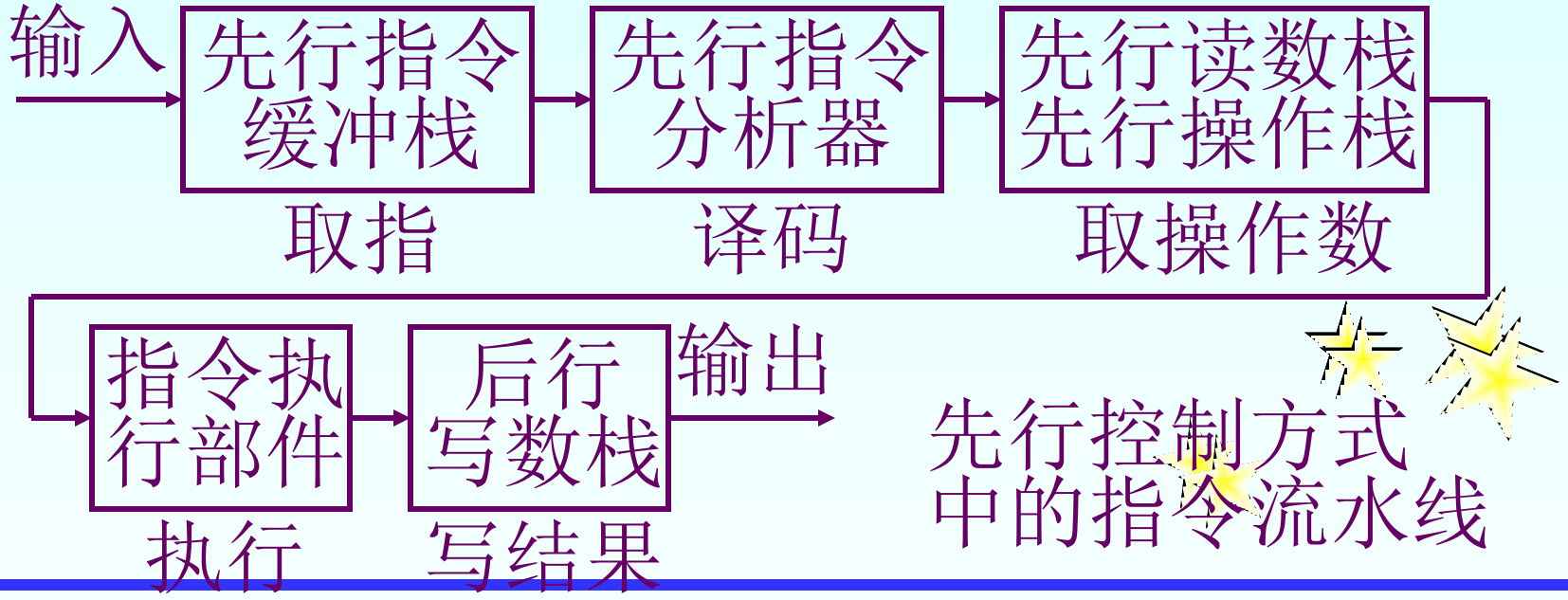
非线性流水线必须用流水线连接图+流水线预约表等共同表示。

2、按照流水线的级别来分

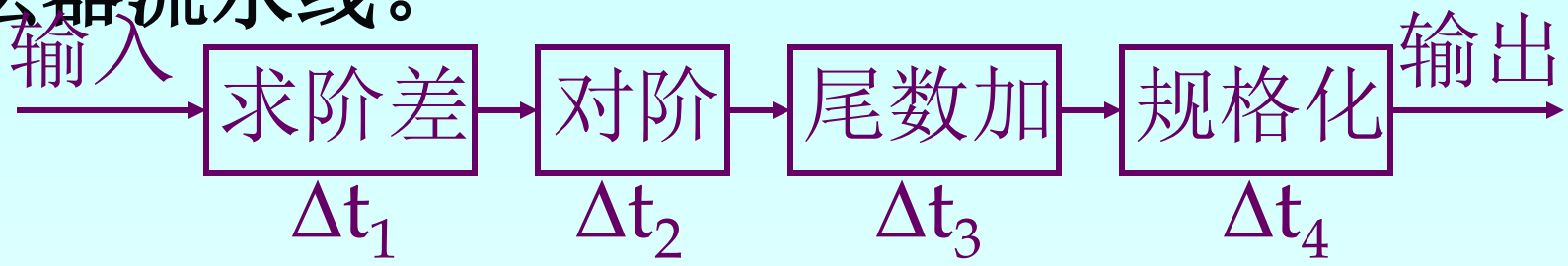
处理机级流水线, 又称为指令流水线。

(Instruction Pipelining)

例如：在采用先行控制器的处理机中，各功能部件之间的流水线。

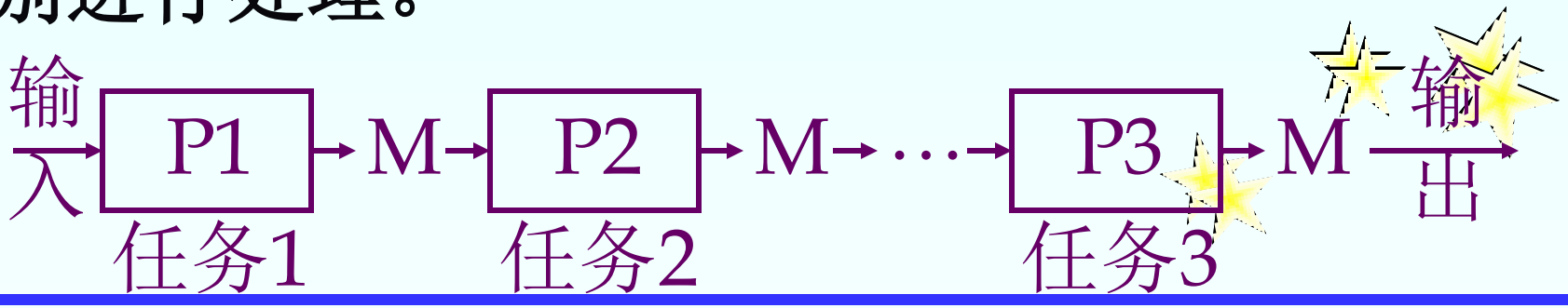


部件级流水线（操作流水线），如浮点加法器流水线。



处理机之间的流水线称为宏流水线 (Macro Pipelining)。

每个处理机对同一个数据流的不同部分分别进行处理。



3、单功能流水线与多功能流水线

单功能流水线：

只能完成一种固定功能的流水线。

多功能流水线：

流水线的各段通过不同连接实现不同功能。

4、静态流水线与动态流水线

静态流水线：

同一段时间内，多功能流水线中的各个功能段只能按照一种固定的方式连接，实现一种固定的功能。

只有连续出现同一种运算时，流水线的效率才能得到充分的发挥。

动态流水线：

在同一段时间内，多功能流水线中的各段可以按照不同的方式连接，同时执行多种功能。

线性流水线的性能分析

衡量流水线性能的主要指标有：
吞吐率、加速比和效率

1、吞吐率（Though Put）

求流水线吞吐率的最基本公式：$TP = n / T_k$

n为任务数，T_k为完成n个任务所用时间。

各段执行时间相等，输入连续任务情况下完成n个连续任务需要的总时间为：

$$T_k = (k+n-1)\,\Delta t$$

k为流水线的段数，Δt为时钟周期

吞吐率：

$$TP = \frac{n}{(k+n-1)\Delta t}$$

最大吞吐率为：

$$TP_{\max} = \lim_{n\to\infty} \frac{n}{(k+n-1)\Delta t} = \frac{1}{\Delta t}$$

各段执行时间不相等、输入连续任务情况下：

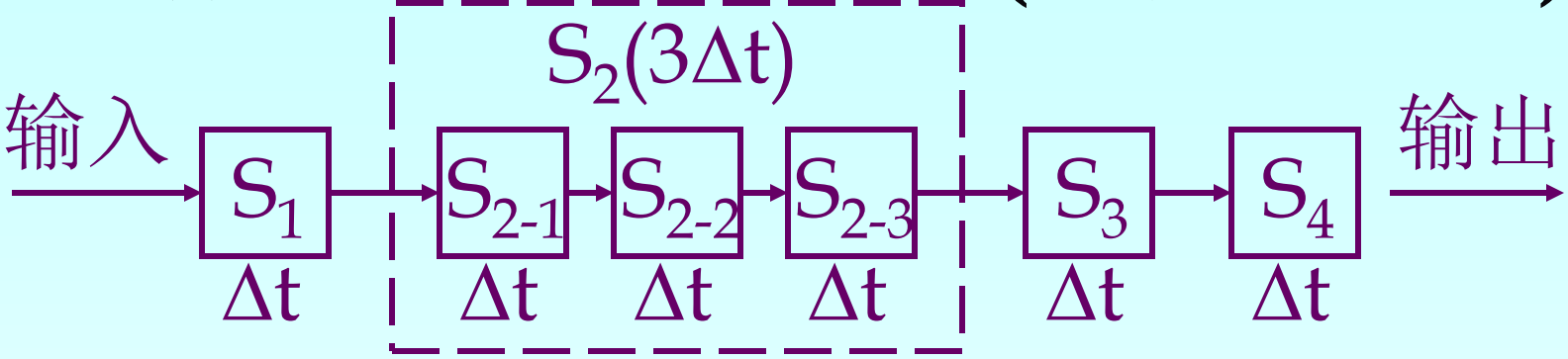
吞吐率为：

$$TP = \frac{n}{\sum_{i=1}^{k} t_i + (n-1)\max(\Delta t_1, \Delta t_2, \cdots, \Delta t_k)}$$

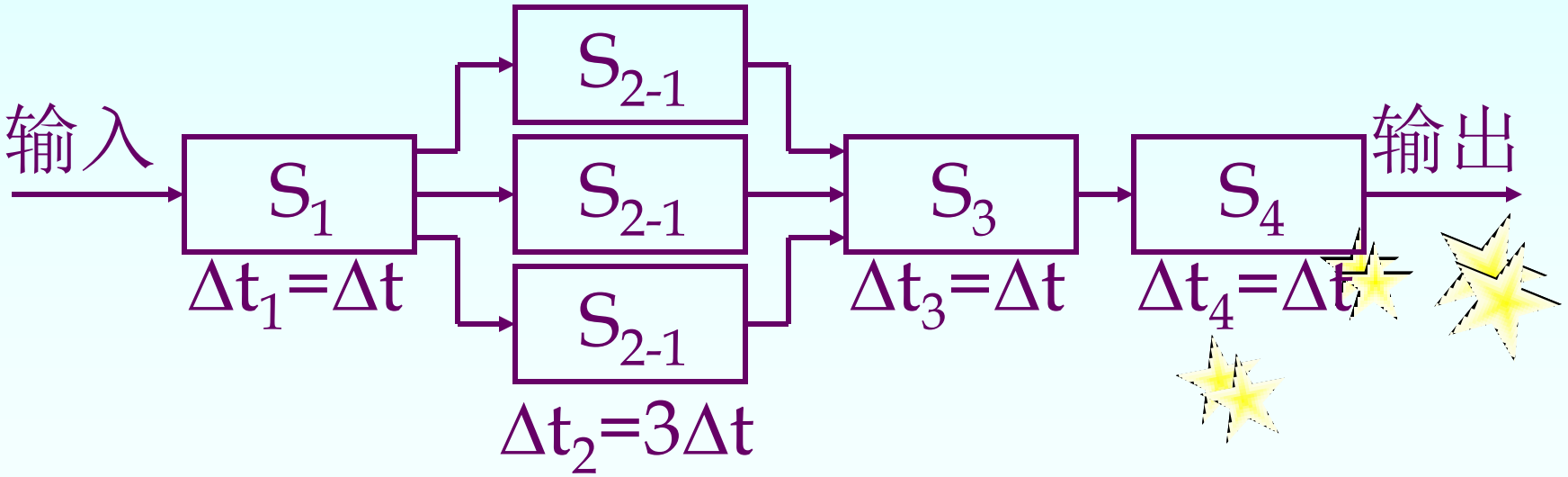
最大吞吐率为：

$$TP = \frac{1}{\max(\Delta t_1, \Delta t_2, \cdots, \Delta t_k)}$$

一是将“瓶颈”流水段细分(如果可分的话):



二是将“瓶颈”流水段重复设置:



2、加速比（Speedup）

计算流水线加速比的基本公式：

S = 顺序执行时间T_0 / 流水线执行时间T_k

各段执行时间相等，输入连续任务情况下

加速比为：

$$S = \frac{k \cdot n \cdot \Delta t}{(k+n-1)\Delta t} = \frac{k \cdot n}{k+n-1}$$

最大加速比为：

$$S_{\max} = \lim_{n \to \infty} \frac{k \cdot n}{k+n-1} = k$$

3、效率（Efficiency）

计算流水线效率的一般公式：

$$E=\frac{n\text{个任务占用的时空区}}{k\text{个流水段的总的时空区}}=\frac{T_0}{k\cdot T_k}$$

各流水段执行时间相等，输入n个连续任务流水线的效率为：

$$E=\frac{k\cdot n\cdot \Delta t}{k\cdot (k+n-1)\cdot \Delta t}=\frac{n}{k+n-1}$$

流水线的最高效率为：

$$E_{\max}=\lim_{n\to\infty}\frac{n}{k+n-1}=1$$

流水线的吞吐率、加速比与效率的关系：

因为

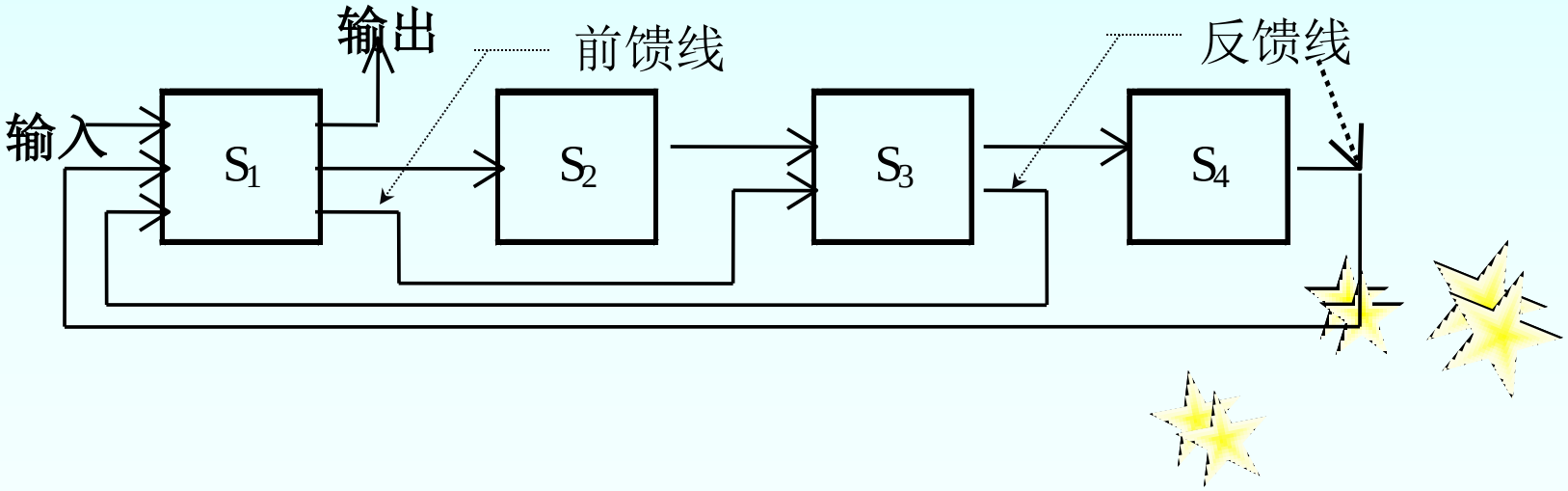
$$TP=\frac{n}{(k+n-1)\Delta t} \qquad S=\frac{k\cdot n}{k+n-1} \qquad E=\frac{n}{k+n-1}$$

因此：**E=TP·Δt，S=k·E**

非线性流水线的调度技术

线性流水线能够用流水线连接图唯一表示。

连接图不能唯一表示非线性流水线的工作流程，因此，引入流水线预约表。



构造状态图：把冲突向量送入一个m位逻辑右移移位器；如果移位器移出0，用移位器中的值与初始冲突向量作“按位或”运算，得到一个新的冲突向量；否则不作任何处理；如此重复m次。

对于中间形成的每一个新的冲突向量，也要按照这一方法进行处理。

在初始冲突向量和所有的新形成的冲突向量之间用带箭头的线连接，当新形成的冲突向量出现重复时可以合并到一起。

5.3 向量的流水处理与向量流水处理机

向量的流水处理

例如，要计算$D=A*(B+C)$，其中，A、B、C、D都是具有N个元素的向量，应该采用什么样的处理方式才能最充分发挥流水线的效能呢?

。

向量处理方式

1．横向处理方式，又称为水平处理方式，横向加工方式等。向量计算是按行的方式从左至右横向地进行。

2．纵向处理方式，又称为垂直处理方式，纵向加工方式等。向量计算是按列的方式自上而下纵向地进行。

3．纵横处理方式，又称为分组处理方式，纵横向加工方式等。横向处理和纵向处理相结合的方式。

5.4 指令级高度并行的超级处理机

超标量处理机

基本结构

一般流水线处理机：

一条指令流水线，一个多功能操作部件，每个时钟周期平均执行指令的条数小于1。

多操作部件处理机：

一条指令流水线，多个独立的操作部件，操作部件可以采用流水线，也可以不流水。多操作部件处理机的指令级并行度小于1。

单发射与多发射

单发射处理机：

每个周期只取一条指令、只译码一条指令，只执行一条指令，只写回一个运算结果。

取指部件和译码部件各设置一套。

可以只设置一个多功能操作部件，也可以设置多个独立的操作部件。

操作部件中可以采用流水线结构，也可以不采用流水线结构。

设计目标是每个时钟周期平均执行一条指令，ILP的期望值1。

多发射处理机：

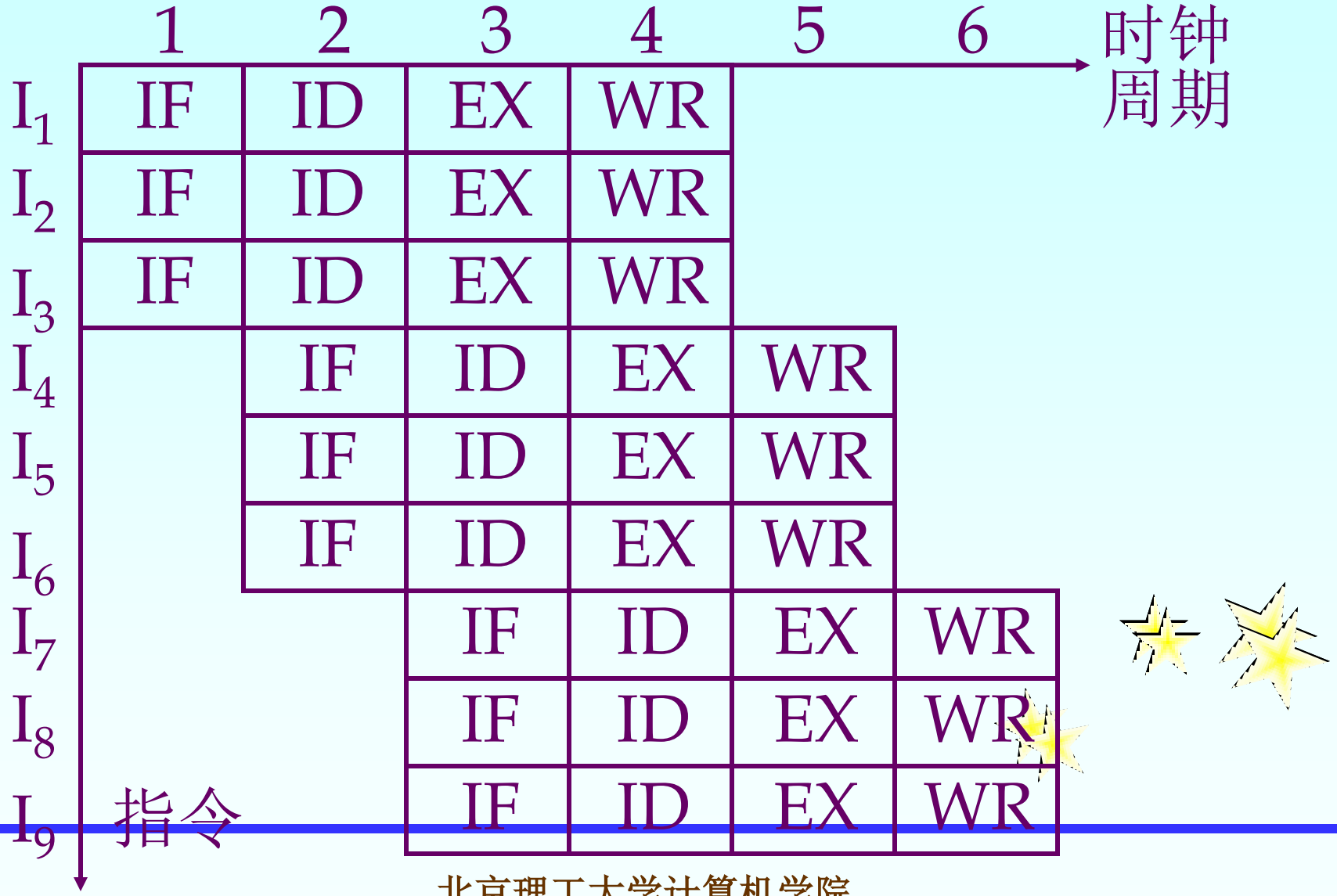
每个周期同时取多条指令、同时译码多条指令，同时执行多条指令，同时写回多个运算结果。

需要多个取指令部件，多个指令译码部件和多个写结果部件。

设置多个指令执行部件，复杂的指令执行部件一般采用流水线结构。

设计目标是每个时钟周期平均执行多条指令，ILP的期望值大于1。

多发射处理机的指令流水线时空图



超标量处理机：

一个时钟周期内能够同时发射多条指令的处理机称为超标量处理机。

必须有两条或两条以上能够同时工作的指令流水线。

超标量处理机性能

指令级并行度为(1,n)的超标量处理机，执行N条指令所的时间为：

$$T(m,1) = (k + \frac{N-m}{m})\Delta t$$

超标量处理机相对于单流水线普通标量处理机的加速比为：

$$S(m,1) = \frac{T(1,1)}{T(m,1)} = \frac{(k+N-1)\Delta t}{(k+\frac{N-m}{m})\Delta t}$$

即：

$$S(m,1)=\frac{m(k+N-1)}{mk+N-m}$$

超流水线处理机的加速比的最大值为：

$$S(m,1)_{MAX}=m$$

超流水线处理机

两种定义：

一个周期内能够分时发射多条指令的处理机称为超流水线处理机。

指令流水线有8个或更多功能段的流水线处理机称为超流水线处理机。

提高处理机性能的不同方法：

超标量处理机是通过增加硬件资源为代价来换取处理机性能的。

超流水线处理机则通过各硬件部件充分重叠工作来提高处理机性能。

两种不同并行性：

超标量处理机采用的是空间并行性（同时性）。

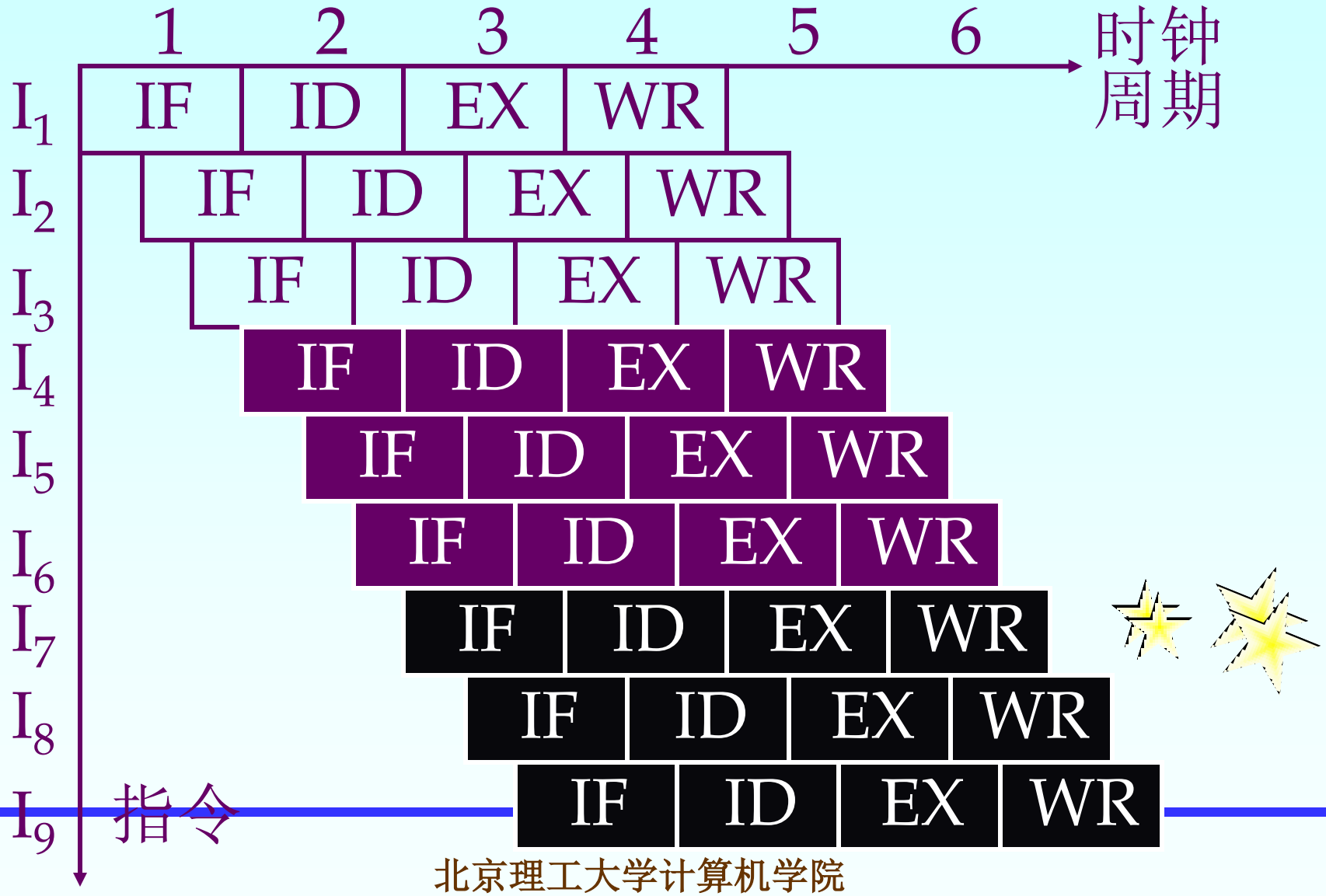
超流水线处理机采用的是时间并行性（并发性）。

指令执行时序

每隔1/n个时钟周期发射一条指令，流水线周期为1/n个时钟周期。

在超标量处理机中，流水线的有些功能段还可以进一步细分。

每个时钟周期分时发送3条指令的超流水线



超流水线处理机性能

指令级并行度为(1,n)的超流水线处理机，执行N条指令所的时间为：

$$T(1,n)=(k+\frac{N-1}{n})\Delta t$$

超流水线处理机相对于单流水线普通标量处理机的加速比为：

$$S(1,n)=\frac{T(1,1)}{T(1,n)}=\frac{(k+N-1)\Delta t}{(k+\frac{N-1}{n})\Delta t}$$

即：

$$S(1,n)=\frac{n(k+N-1)}{nk+N-1}$$

超流水线处理机的加速比的最大值为：

$S(1,n)_{MAX} = n$

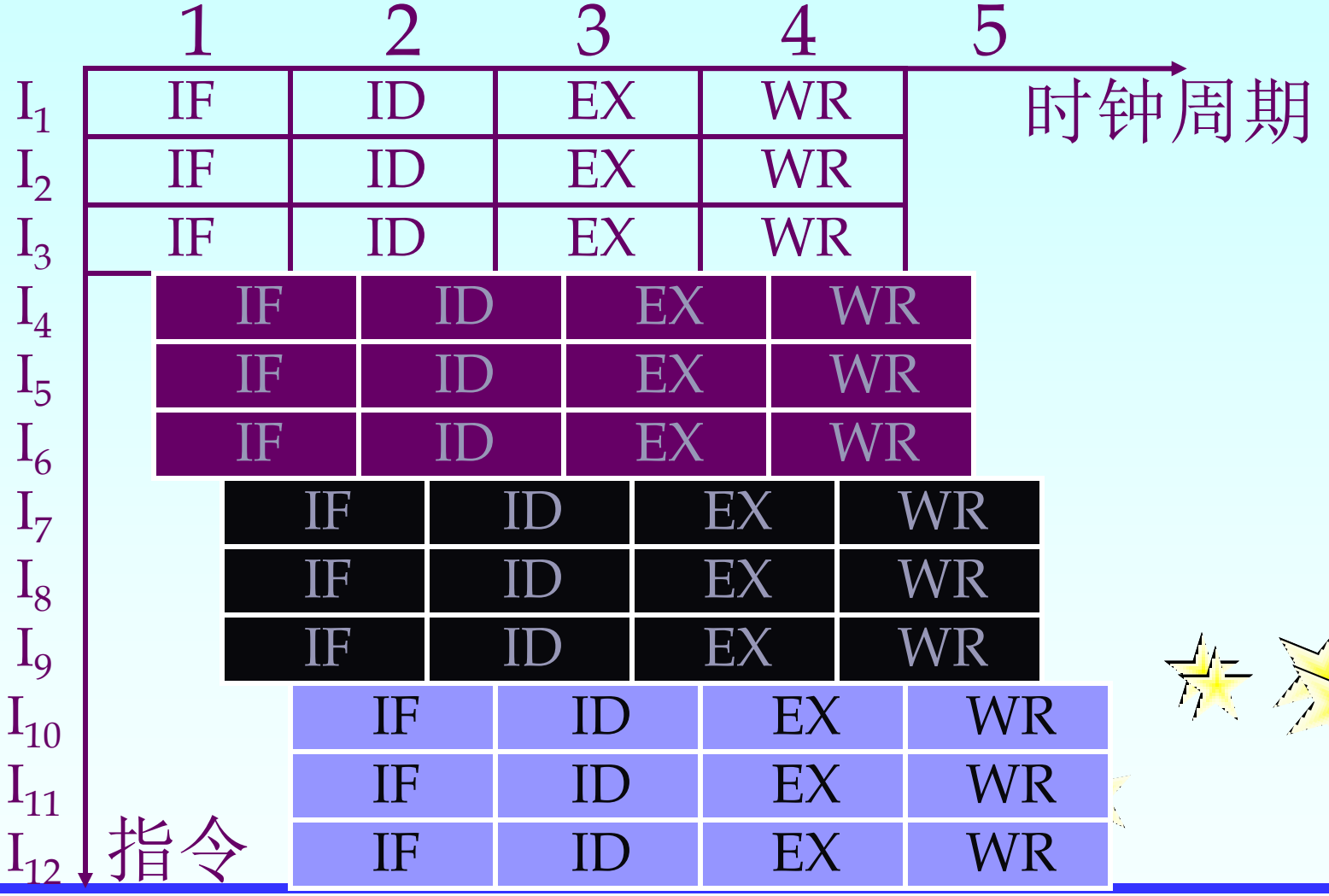
超标量超流水线处理机

把超标量与超流水线技术结合在一起，就成为超标量超流水线处理机。

指令执行时序

超标量超流水线处理机在一个时钟周期内分时发射指令n次，每次同时发射指令m条，每个时钟周期总共发射指令$m \cdot n$条。

每时钟周期发射3次,每次3条指令



超标量超流水线处理机性能

指令级并行度为(m,n)的超标量超流水线处理机，连续执行N条指令所需要的时间为：

$$T(m,n)=(k+\frac{N-m}{m\cdot n})\Delta t$$

超标量超流水线处理机相对于单流水线标量处理机的加速比为：

$$S(m,n)=\frac{S(1,1)}{S(m,n)}=\frac{(k+N-1)\Delta t}{(k+\frac{N-m}{mn})\Delta t}$$

$$S(m,n)=\frac{m\cdot n\cdot(k+N-1)}{m\cdot n\cdot k+N-m}$$

在理想情况下，超标量超流水线处理机加速比的最大值为：

$$S(m,n)_{MAX}=m\,n$$

并行处理机定义：

多个PU按照一定方式互连，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，指令是串行执行的，从PU看，数据是并行处理的。

并行处理机也称为阵列处理机。按照佛林分类法，它属于SIMD计算机。

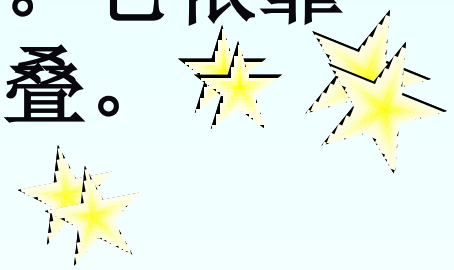
并行处理机的主要特点如下：

速度快，而且潜力大。

模块性好，生产和维护方便。

可靠性高，容易实现容错和重构。

效率低（与流水线处理机、向量处理机等比较）。通常作为专用计算机，因此，在很大程度上依赖于并行算法。它依靠的是资源重复，而不是时间重叠。



6.1 并行处理机原理

阵列处理机的基本构形

分布式存储器的阵列处理机

集中式共享存储器的阵列处理机

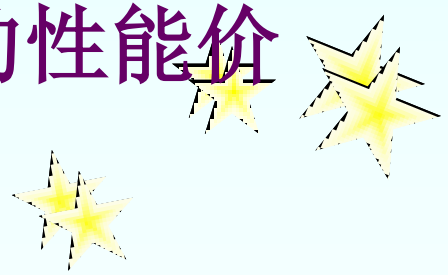
6.3 互连网络的基本概念

互连网络的作用

用来实现计算机系统内部多个处理机或多个功能部件之间的相互连接。

互连网络已成为并行处理系统的核心组成部分。

互连网络对整个计算机系统的性能价格比有着决定性的影响。



互连网络的表示方法

为了在输入结点与输出结点之间建立对应关系，互连网络有三种表示方法：

(1) 互连函数表示法

(2) 图形表示法

(3) 输入输出对应表示法

互连网络的特性

互连网络通常是用有向边或无向边连接有限个结点的组成。

互连网络的主要特性有：

(1) 网络规模：网络中结点的个数

(2) 结点度：与结点相连接的边数称为结点度。包括入度和出度。进入结点的边数叫入度，从结点出来的边数则叫出度

(3) 距离：两个结点之间相连的最少边数

(4) 网络直径：网络中任意两个结点间距离的最大值。用结点间的连接边数表示

(5) 结点间的线长：两个结点间连线的长度。用米、公里等表示

(6) 对称性：从任何结点看到拓扑结构都是一样的网络称为对称网络。对称网络比较易实现，编程也较容易。

互连网络在传输方面的主要性能参数：

(1) 频带宽度(Bandwidth)：互连网络传输信息的最大速率。

(2) 传输时间(Transmission time)：等于消息长度除以频宽。

(3) 飞行时间(Time of flight)：第一位信息到达接收方所花费的时间。

(4) 传输时延(Transport latency)：等于飞行时间与传输时间之和。

(5) 发送方开销(Sender overhead)：处理器把消息放到互连网络的时间。

(6) 接收方开销(Receiver overhead)：处理器把消息从网络取出来的时间。

一个消息的总时延可以用下面公式表示：

总时延＝发送方开销＋飞行时间＋消息长度/频宽＋接收方开销

基本的单级互连网络

1、 立方体单级网络

推广到n维的情形，N个节点的立方体单级网络共有$n=\log_2 N$种互连函数， 即

$$Cube_i(P_{n-1}\cdots P_i\cdots P_1P_0)=P_{n-1}\cdots \overline{P}_i\cdots P_1P_0$$

2、PM2I单级网络

PM2I单级网络是“加减2^i”(Plus-Minus 2^i)单级网络的简称。能实现与j号处理单元直接相连的是号为$j\pm2^i$的处理单元，即

$$\begin{cases} PM2_{+i} = j + 2^i \bmod N \\ PM2_{-i} = j - 2^i \bmod N \end{cases}$$

式中，$0 \le j \le N-1$, $0 \le i \le n-1$, $n=\log_2 N$。

3、混洗交换单级网络

用互连函数表示为

$$Shuffle(P_{n-1}P_{n-2}\cdots P_1P_0)=P_{n-2}\cdots P_1P_0P_{n-1}$$

式中，$n=\log_2 N$，$P_{n-1}P_{n-2}\cdots P_1P_0$为入端编号的二进制码。

7.1 多处理机概念

两个或两个以上处理机(包括PU和CU)，通过高速互连网络连接起来，在统一的操作系统管理下，实现指令以上级（任务级、作业级）并行。

按照Flynn分类法，多处理机系统属于MIMD计算机。

多处理机系统有多种分类方法：

按照处理机之间的连接程度：紧密耦合多处理机和松散耦合多处理机。

按照是否共享主存储器：共享存储器多处理机和分布存储器多处理机。

按照处理机是否相同：同构型多处理机和异构型多处理机。

多处理机系统的特点

1、结构灵活

并行处理机：专用，PE数很多（几千个），固定有限的通信。

多处理机： 通用，几十个，高速灵活的通信。

2、程序并行性

并行处理机的并行性存在于指令内部，识别比较容易。

多处理机的并行性存在于指令外部，在多个任务之间，识别难度较大。

3、并行任务派生

并行处理机把同种操作集中在一起，由指令直接启动各PE同时工作。

多处理机用专门的指令来表示并发关系，一个任务开始执行时能够派生出与它并行执行的另一些任务。如果任务数多于处理机数，多余的任务进入排队器等待。

4、进程同步

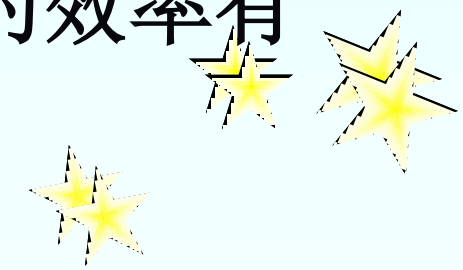
并行处理机仅一个CU，自然是同步的。

多处理机执行不同的指令，工作进度不会也不必保持相同。先做完的要停下来等待。有数据相关和控制相关也要停下来等待。要采取特殊的同步措施来保持程序所要求的正确顺序。

5、资源分配和进程调度

并行处理机的PE是固定的，采用屏蔽手段改变实际参加操作的PE数目。

多处理机执行并发任务，需用处理机的数目不固定，各个处理机进入或退出任务的时刻不相同，所需共享资源的品种、数量又随时变化。提出资源分配和进程调度问题，它对整个系统的效率有很大的影响。



多处理机结构：

从存储器的分布和使用上看，分为：

共享存储器结构：

物理共享：UMA（SMP）；

逻辑共享：NUMA，ccNUMA，COMA；

通过共享存储器通信，紧耦合。

分布式存储器结构(NoRMA)：

MPP，COW/NOW；

通过消息传递机制通信，松耦合。

多核处理器：

集成两个或以上独立处理单元（核）的处理器。

多核处理器结构：

同构多核，异构多核；

Cache设置与访问：典型4种结构

专用L1 Cache，专用片内L2 Cache；

专用L1 Cache，共享片外L2 Cache；

专用L1 Cache，共享片内L2 Cache；

专用L1 Cache，专用L2 Cache，共享片内L3 Cache；

多处理机的Cache一致性

为解决多个Cache之间内容不一致问题。

原因：

共享可写数据；

进程迁移；

I/O传输

多处理机的Cache一致性

解决方法：主要有两类

一类是以硬件为基础的做法

Cache一致性协议：

监听协议；

基于目录的协议；

另一类则主要是以软件为基础的做法。

监听协议：

适用于基于总线的共享存储器结构的多处理机。

写无效：

写共享数据时，使其他Cache中的共享数据的副本无效，独占数据，然后写入或更新。

写更新：（播写法）

通过总线将更新后的数据分发给所有其他处理机，更新所有Cache中的副本。

基于目录的协议：

适用于非总线互连的多处理机。

目录存放： 两种形式

集中式：集中存放在某地。

分布式：分布存放在不同处理机中。

目录表项：

记录数据块被处理机更新的情况。

目录形式：

全映射目录，有限目录，链式目录

程序并行性：

并行算法分类：多种

数值，非数值；

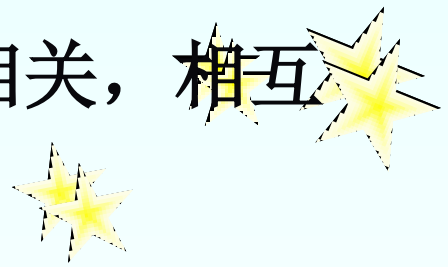
同步，异步；

粗粒度，中粒度，细粒度；

多级并行，多线程并行

程序段见的相关：

数据相关，数据反相关，数据输出相关，相互交换等。



多处理机性能模型：

性能和效率在很大程度上依赖于**R/C**比值，其中：**R**代表程序执行时间，**C**代表通信开销。

R/C比值是衡量任务粒度**(Granularity)**大小的尺度

在粗粒度（**Coarsegrain**）并行情况下，**R/C**比值比较大，通信开销小；

在细粒度（**Finegrain**）并行情况下，**R/C**比值比较小，通信开销大。

多处理机操作系统：

多台处理机协同工作完成所要求任务的操作系统。

三种类型：

主从型；

各自独立型；

浮动型；