计算机体系结构
复习

郑宏 副教授

计算机学院

北京理工大学

关于考试

- 考题类型：
  - 术语解释
  - 填空，判断对错
  - 选择
  - 简述回答
  - 计算
- 考题范围：
  - 第1章 – 第7章
  - 重点：第1章 – 第5章

- 复习：
  - 书 + 课件 + 作业 + 例题

第1章

概论

第1章 概论

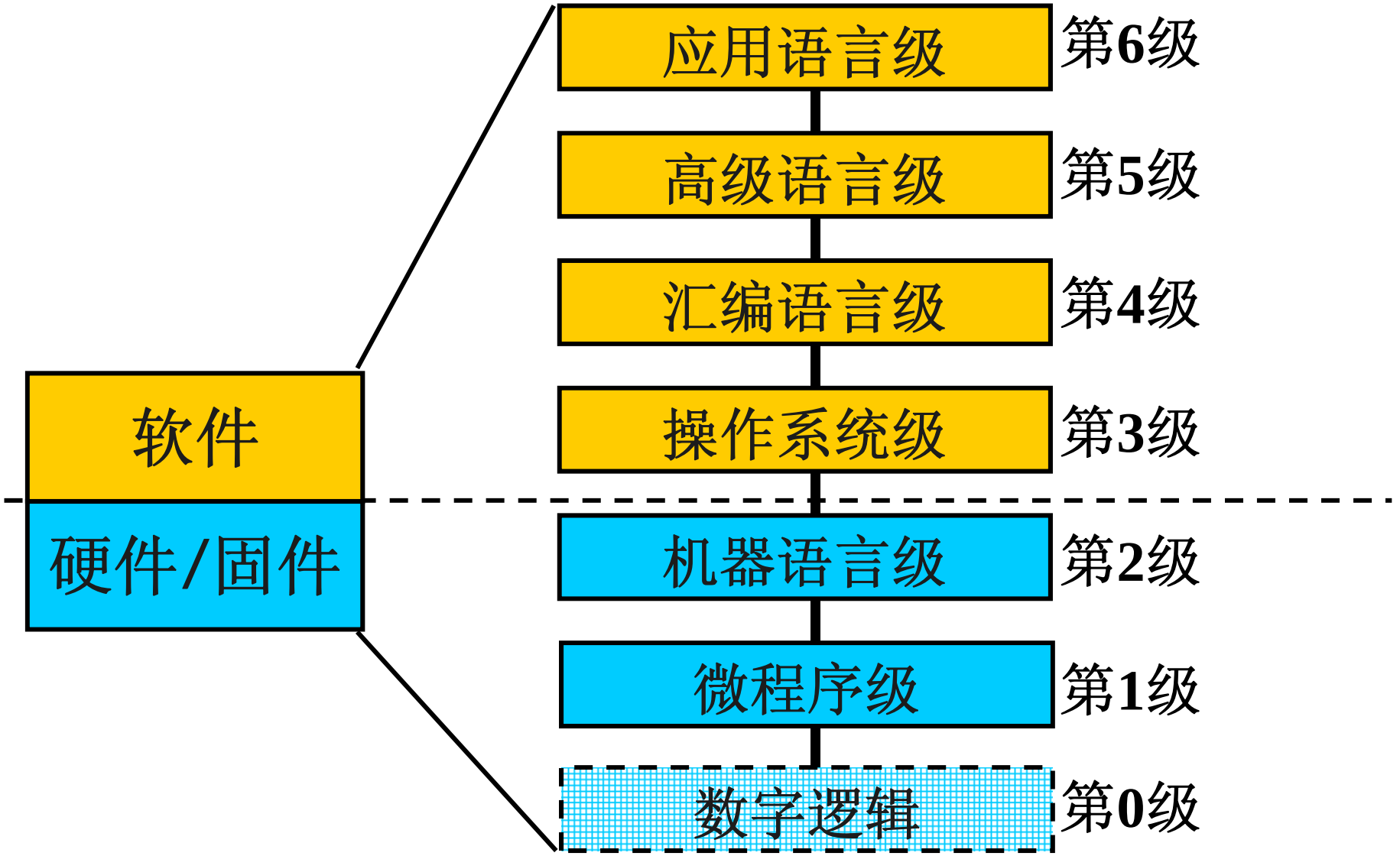
- 本章是基础，概念、术语较多
- 计算：CPU性能，Amdahl定律 等
- 重点：
  - 多级层次概念和机器级实现技术
  - 计算机体系结构的概念，设计内容和目标
  - 软硬件取舍与计算机系统设计思路
  - 影响系统结构的因素：软件的影响
  - 计算机设计的量化准则：CPU性能，Amdahl定律
  - 并行性，方法，分类

多级层次结构和机器级的实现技术

一，多级层次结构概念

- 对计算机系统的抽象
- 从使用语言的角度，将计算机系统按功能划分为多级层次结构，以对整个系统进行描述、分析、设计和使用

计算机系统的多级层次结构



多级层次结构和机器级的实现技术

二，机器级的实现技术

- 只有二进制机器指令（即机器语言）可以直接被硬件识别和执行，其上面各层机器级所使用的语言都必须被翻译或解释为较低层机器级上语言，由较低层机器实现
- 翻译和解释或这两者的结合是各机器级实现的主要方法
- 掌握翻译或解释概念和方法，注意两者的区别。

多级层次结构和机器级的实现技术

硬件实现，还是软件实现？

- 在逻辑功能上，软件和硬件是等效的。
  - 原理上，软件实现的功能完全可以由硬件实现，硬件实现的功能也完全可以由软件模拟完成。
- 但软件和硬件的性能价格比是不等效的。

系统结构、组成与实现及相互关系

- 系统结构定义

  计算机体系结构是对各机器级界面的划分、定义及上下级功能分配。

- 按照计算机系统的多级层次结构，不同机器级的界面有很大不同，意味着每个机器级都有其系统结构

- 各机器级都有它们自己的系统结构

系统结构、组成与实现及相互关系

- 计算机体系结构概念的**实质**是计算机系统中**软硬件界面**的确定，其界面之上的是软件的功能，界面之下的是硬件和固件的功能。
- **“指令集结构”**就是软硬件之间的界面之一
- **传统机器级界面(属性)**：机器语言程序员所看到的计算机系统的属性。

传统机器级界面（软硬件界面）

- 数据表示
- 寻址方式
- 寄存器组织
- 指令集
- 存储系统
- 中断系统
- 机器工作状态的定义和切换
- 输入输出系统
- 信息保护

系统结构、组成与实现及相互关系

- **透明性概念**

  本来存在的事物或属性，从某个角度看却好象不存在。

- 传统机器级的属性对高级语言程序员来说是**透明**的。

系统结构、组成与实现及相互关系

- 计算机组成的定义

  对系统结构的逻辑实现。包括：机器级内部的数据流、控制流的组成及逻辑设计；部件功能、部件间的联系等。

- 所解决的问题：
  - 在所希望达到的性能价格下，如何最佳、最合理地把各种设备和部件组织在一起，以实现所确定的系统结构

系统结构、组成与实现及相互关系

- 计算机组成的设计内容：
  - 数据通路宽度
  - 专用部件设置
  - 操作对部件的共享程度
  - 功能部件的并行度
  - 控制机构的组成方式
  - 缓冲和排队技术
  - 预估、预判技术
  - 可靠性技术 等

系统结构、组成与实现及相互关系

- 计算机实现的定义

  对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
  - 器件、模块、插件、底板的划分和连接
  - 专用器件设计
  - 信号传输；
  - 电源、冷却、微组装技术、整机装配技术等

- 其中，器件技术在实现技术中起主导作用

系统结构、组成与实现及相互关系

- 系统结构、组成和实现的关系
  - 三个不同的概念，即互相联系，又互相影响。
  - 不同系统结构使用不同组成技术。
  - 相同系统结构可以采用不同的组成。
  - 一种组成可以有不同的实现。
  - 组成也会影响系统结构。

软硬件取舍原则

- 在逻辑功能上，软件和硬件是等效的。
- 三原则：
  - 原则1：在现有硬件和器件条件下，系统要有高的性能价格比。
  - 原则2：要考虑到准备采用和可能采用的组成技术，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。
  - 原则3：为软件(如OS、编译、高级语言、应用软件等)的设计和实现提供更多更好的硬件支持。

计算机系统的设计思路

- 从多级层次结构出发，计算机系统的设计有三种思路：
  - 由上往下
  - 由下往上
  - 由中间开始
    - 中间取在哪里？有何优点？

实现软件的可移植性的方法

- 软件的可移植性定义。
- 实现软件可移植性的几种技术：
  - 统一高级语言
  - 采用系列机思想
  - 模拟与仿真

  - 掌握这些技术的主要特点

实现软件的可移植性的方法

- 软件兼容的定义。
- 软件兼容的分类：
  - 向上(下)兼容
  - 向前(后)兼容
- 系列机的定义。
- 系列机软件的兼容性要求：
  - 系列机软件必须保证向后兼容，力争做到向上兼容！
- 兼容机的定义。
- 软件兼容的缺点。
- 仿真与模拟的定义与区别。

计算机系统设计的量化准则

- **CPU性能格式**

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

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

$$T_{CPU} = \frac{CLK}{f_c} = CLK \times T_{CLK}$$

CPU性能公式

- 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IC_i。

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

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

CPU性能公式

- **MIPS与CPU时间**

$$MIPS = \frac{IC}{T_{CPU} \times 10^6} = \frac{IC}{IC \times CPI \times \frac{1}{f_c} \times 10^6}$$

$$= \frac{f_c}{CPI} \times 10^{-6} = IPC \times f_c \times 10^{-6}$$

CPU性能公式

- **等效指令速度：吉普森（Gibson）法**

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

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

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

W_i ——第i种指令的使用频度　　**T_i——第i种指令的执行时间**

CPI_i——第i种指令的CPI　　**$MIPS_i$——第i种指令的MIPS**

Amdahl定律

- 加速比 S_n:

假设对机器进行某种改进，那么机器系统的加速比为：

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

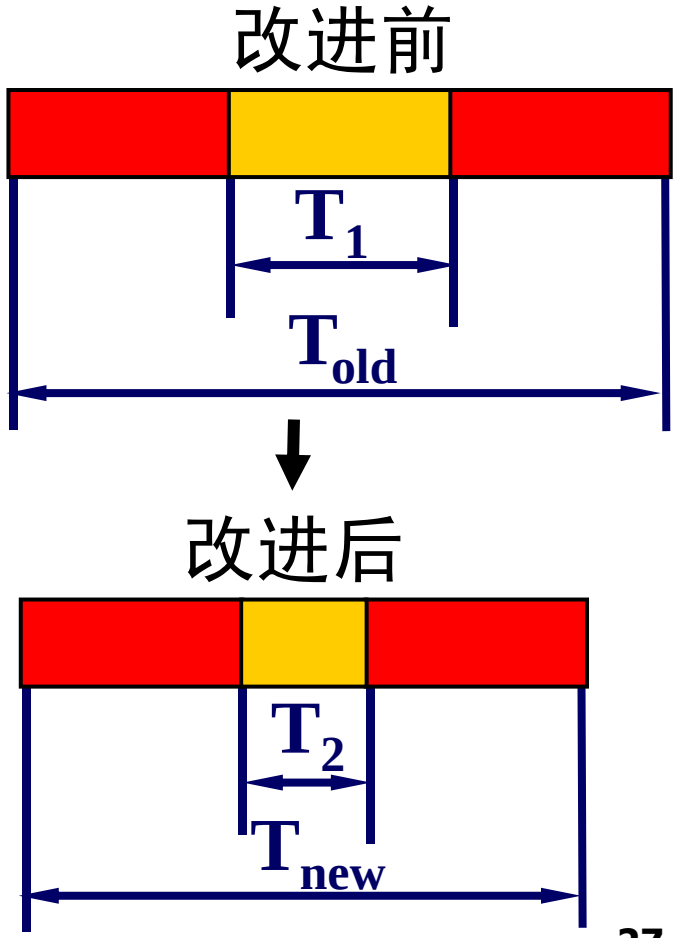
- 可改进比例 Fe
  - 可改进部分在原系统计算时间中所占的比例，0≤ Fe ≤1。
  - 1 – Fe 表示不可改进部分。
- 部件加速比Se
  - 可改进部分改进以后的性能提高，一般情况下，Se ＞1。

Amdahl定律

$$Fe = T_1 / T_{old} \qquad Se = T_1 / T_2$$

$$T_{new} = T_{old}\left[(1-Fe) + \frac{Fe}{Se}\right]$$

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



并行性及计算机系统分类

- **并行性定义**

  可以同时进行运算或操作的特性，包含两重含义：

  - **同时性(simultaneity)**：同一时刻发生
  - **并发性(concurrency)**：同一时间间隔内发生

- 开发性等级：不同角度不同分级

并行性及计算机系统分类

- 开发并行性的三种途径：
  - 时间重叠
    - 引入时间概念，让多个处理过程轮流使用同一套硬件设备的各个部分，以加快硬件周转而赢得速度
  - 资源重复
    - 引入空间因素，重复设置多套硬件
    - 提高速度，同时提高可靠性
  - 资源共享
    - 利用软件让多个用户轮流使用同一套资源
  - 要求掌握各自的特点，能举例说明

并行性及计算机系统分类

- 多机系统包括：
  - 多处理机系统
    - 由多台处理机组成的单一计算机系统
  - 多计算机系统
    - 由多台独立的计算机组成的系统

- 要求：了解它们的定义，之间的差别，实现的并行性，能举出例子

并行性及计算机系统分类

- 多机系统的耦合度的定义
- 耦合度分为：
  - 最低耦合
  - 松散耦合
  - 紧密耦合

- 要求：了解，能举例说明耦合度

并行性及计算机系统分类

- 弗林(Flynn)分类法
  - 指令流
  - 数据流
  - 多倍性
- 它将计算机系统分为四类：
  - SISD (单指令流，单数据流)
  - SIMD (单指令流，多数据流)
  - MISD (多指令流，单数据流)
  - MIMD (多指令流，多数据流)
  - 要求：能举例说明

第2章

数据表示与指令系统

第2章 数据表示与指令系统

- **计算：** 信息熵，平均码长，操作码编码（定长，哈夫曼，等长扩展）等
- **重点：**
  - 自定义数据表示
  - 指令操作码编码
  - 指令格式优化方法
  - 指令系统的改进

自定义数据表示

- 数据类型
  - 数据值的集合以及该集合的操作的集合。
- 数据表示
  - 可以被硬件直接识别和指令系统直接调用的数据类型。
- 计算机系统结构研究的首要问题是：
  - 在所有的数据类型中，哪些用硬件实现，哪些用软件实现，并研究它们的实现方法。
- 注意：数据表示与数据结构的关系

自定义数据表示

- 自定义数据表示的定义

  由数据本身来表明数据类型，使计算机内的数据具有自定义能力。
- 带标志符的数据表示
- 数据描述符
- 注意它们之间的的区别
- 其他数据表示

寻址、编址和程序定位方法

- 寻址技术
  - 寻找操作数及其他信息的地址的技术
- 编址方式
  - 指对各种存储设备进行编码的方法
- 部件的编址方式主要有三种：
  - 统一编址
  - 分类编址
  - 隐含编址
- 大多数计算机采用分类编址方式
  - 通常将主存、通用寄存器和堆栈分类编址

寻址、编址和程序定位方法

- 定位方式
  - 何时确定程序在主存中的物理地址？
  - 以何种方法确定程序在主存中的物理地址？
- 程序定位方法
  - 相关概念：
    - 逻辑地址
    - 主存物理地址
    - 程序定位
  - 方法：
    - 静态再定位
    - 动态再定位

指令格式优化

- 优化的目的
- 指令的组成
- 操作码的优化方法：
  - 定长编码
  - 全哈夫曼编码
  - 扩展编码，等长扩展编码
- 要求：
  - 能根据题目要求设计操作码
  - 会构造哈夫曼树，会分配编码
  - 会计算

操作码优化

- 信息源熵
  - 信息源包含的平均信息量
  - 对操作码而言就是操作码的**最短平均码长（理想情况）**。
  - 计算公式：

$$H = -\sum p_i \log_2 p_i$$

其中：H—信息源熵

p_i—第i个操作码出现的概率

操作码优化

- 信息冗余量

$$R = 1 - \frac{H}{\text{实际平均码长}}$$

- 实际平均码长

$$L = \sum p_i l_i$$

其中：p_i —第i个操作码出现的概率

l_i —第i个操作码的长度

指令系统的发展与改进

- 指令系统的设计、发展和改进有两种不同的途径和方向：
  - 增强指令功能。基本思路
  - 简化指令系统。基本思路

指令系统的发展与改进

- **CISC定义**
- **CISC结构追求的目标：**
  - 强化指令功能，减少程序的指令条数，以达到提高性能的目的。
- **从以下几个方面考虑：**
  - 面向目标程序的优化实现
  - 面向高级语言的优化实现
  - 面向操作系统的优化实现
- **CISC问题： 20％与80％规律**

指令系统的发展与改进

- RISC定义与特点
- 设计RISC机器应当遵循的一般原则
- RISC采用的基本技术、关键技术等：
  - 简单的寻址方式
  - 重叠寄存器窗口技术
  - 流水线技术与延时转移技术
  - 指令取消技术：向前转移，先后转移，隐含转移
  - 指令流调整技术
  - 优化编译系统设计的技术 等

指令系统的改进

- 了解RISC结构存在的不足：
  - 指令减少，加重了汇编程序员的负担，增加了机器语言程序的长度，占用了较大的存贮空间，加大了指令的信息流量
  - RISC上的编译程序要比CISC上的难写

第3章

输入输出系统

第3章 输入输出系统

- 计算：通道流量，画图，分析
- 重点：
  - I/O系统的控制方式
  - RAID
  - 通道处理机工作原理，通道类型，通道流量分析

I/O系统的控制方式

- 在计算机系统中，把处理机与主存储器之外的部分统称为**I/O系统**。
- **I/O系统特点**
  - 异步性
  - 实时性
  - 与设备无关性

I/O系统的控制方式

- 在计算机系统中，把处理机与主存储器之外的部分统称为**I/O系统**。
- **I/O系统功能**
  - 对指定外设进行**I/O**操，同时完成许多其它的管理和控制任务，如：编址、准备信息通路、信息传送、格式变换、中断请求等。
  - 上述功能由三个部分协同完成：
    - 输入输出机器指令
    - 输入输出设备及其控制器硬件
    - 操作系统

I/O系统的控制方式

- I/O系统性能：
  - 输入输出速度；
  - 用户从输入到输出的等待时间；
  - CPU和主存的利用率；
  - I/O系统的兼容能力，可扩展能力，综合处理能力，性能价格比等。

I/O系统的控制方式

- 三种控制方式：
  - 程序控制方式
  - DMA方式
  - I/O处理机方式
- 掌握每种方式的特点和优缺点

I/O系统的控制方式

- I/O处理机方式有两种形式：
  - 通道方式
  - 外围处理机方式
- 掌握通道处理机的工作原理

RAID

RAID级别	名称	最少数据磁盘数	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	2	0	0
RAID1	镜象磁盘阵列	2	1	0
RAID2	纠错海明码磁盘阵列	2	1	1
RAID3	位交叉奇偶校验的磁盘阵列	2	1	1
RAID4	块交叉奇偶校验的磁盘阵列	2	1	1
RAID5	无独立校验盘的奇偶校验磁盘阵列	2	1	1
RAID6	双维无独立校验盘的奇偶校验磁盘阵列	2	2	2

总线设计

- 总线特点
- 总线类型：多种分类方法
- 总线性能：
  - 宽度，带宽，负载等
- 总线定时控制：
  - 同步，异步
- 总线通信：
  - 同步，异步

总线设计

- 总线仲裁：
  - 集中式
    - 链式查询方式
    - 计数器定时查询方式
    - 独立请求方式
  - 分布式
    - 自举分布式
    - 冲突检测分布式
    - 并行竞争分布式

通道处理机

- 特点：
  - 它拥有通道指令和通道程序
  - 可与CPU并行工作
  - 由通道统一管理外设
- 通道处理机的工作有三个阶段：
  - 通道开始选择设备期
  - 通道数据传送期
  - 通道数据传送结束期
  - 完成一次I/O操作需要两次进管

通道处理机

- 通道类型。按信息传送的方式，分为：
  - 字节多路通道
  - 数组多路通道
  - 选择多路通道
- 它们的速度和容量是不同的。
- 注意掌握每种通道的特点。

通道处理机

- 通道流量分析
  - 通道极限流量
  - 实际最大流量
- 掌握三种通道的通道极限流量的计算方法
- 掌握三种通道的实际最大流量的计算方法

通道处理机

- 字节多路通道的极限流量

$$f_{max \bullet byte} = 1/(T_S + T_D)$$

- 字节多路通道的实际最大流量（求和）

$$f_{byte \bullet j} = \sum_{i=1}^{p_j} f_{i \bullet j}$$

通道处理机

- 数组多路通道的极限流量

$$f_{max \bullet block} = K/(T_S + KT_D)$$

- 数组多路通道的实际最大流量（求最大）

$$f_{block \bullet j} = \max_{i=1}^{p_j} f_{i \bullet j}$$

通道处理机

- 选择多路通道的极限流量

$$f_{max \bullet select} = N/(T_S + NT_D)$$

- 选择多路通道的实际最大流量（求最大）

$$f_{select \bullet j} = \max_{i=1}^{p_j} f_{i \bullet j}$$

其中：

fi, j— 第 **j** 号通道上第 **i** 个设备的数据传输率

Pj — 第 **j** 号通道上的设备数

通道处理机

- 掌握流量设计的基本原则

  极限流量≥实际最大流量

$$f_{max} \geq f$$

- 要求：
  - 会进行流量的计算设计
  - 正确安排优先级
  - 正确画出通道工作的示意图
  - 正确解决出现的问题

第4章

存贮体系

第4章 存贮体系

- 计算：地址映像，替换算法，命中率
- 重点：
  - 存贮体系及其性能，程序局部性
  - 虚拟存贮器及管理方式
  - 页式虚拟存贮器
    - 地址地址映象与变换，替换算法，堆栈型替换算法；
    - 提高虚拟存贮器性能的措施
  - 高速缓冲存贮器(Cache)
    - 特点，地址映象与变换，替换算法及实现，比较对法
    - 提高性能的措施
    - Cache一致性

存贮体系及其性能

- 对存贮器的基本性能要求是：
  - 大容量
  - 高速度
  - 低价格
- 但容量、速度和价格是相互矛盾的

存贮体系及其性能

- 解决矛盾的措施
  - 改进工艺和技术，降低成本、提高速度
  - 构成并行主存系统：
    - 单体多字，多体单字，多体多字
    - 低位交叉：用于提高存储器访问速度
    - 高位交叉：用于扩大存储器容量
  - 使用存贮器系统
    - 例如：至少有主存和辅存两种存贮器
  - 采用存贮体系

存贮体系及其性能

- 掌握存贮体系
  - 定义。多级存贮层次。
  - 虚拟存储器：主存—辅存存贮层次
    - 主要解决容量问题
    - 软硬件实现
    - 对应用程序员透明
  - Cache和主存存贮层次
    - 主要解决速度问题
    - 硬件实现
    - 对应用程序员和系统程序员都透明

存贮体系及其性能

- 程序的局部性定义
  - 时间局部性
  - 空间局部性
- **程序的局部性是预判的基础**
- 预判的准确性是存贮层次设计好坏的主要标志，很大程度上取决于所使用的算法和地址映像与变换方式

存贮体系及其性能

- 存贮层次的性能：对于两级存贮层次
  - 每位价格c
  - 命中率H：第一级的命中率
  - 存贮层次的等效访问时间T_A

同时启动或访问时：

掌握

$$T_A = H T_1 + (1 - H) T_2$$

不同时启动或访问时：

$$T_A = T_1 + (1 - H) T_2$$

  - 访问效率

$$e = \frac{T_{A1}}{T_A} = \frac{T_{A1}}{HT_{A1} + (1-H)T_{A2}} = \frac{1}{H + (1-H) \times \frac{T_{A2}}{T_{A1}}}$$

存贮体系及其性能

- 采用预取提高命中率
  - 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。

掌握 预取后的命中率：$$H' = \frac{H + n - 1}{n}$$

其中：

H – 原来的命中率

n – 数据块大小×数据重复使用次数

虚拟存贮器

- 虚拟存贮器是一个存贮体系
- 它对应用程序员透明，对系统程序员基本不透明
- 它允许应用程序员使用比实际主存空间大得多的程序（虚存）空间访问主存
- 每次访存时都要进行虚地址到实地址转换

虚拟存贮器

- 虚拟存贮器的管理
  - 段式管理
  - 页式管理
  - 段页式管理
- 虚拟存贮器采用地址映象方法、通过地址映象表机构实现程序的定位

页式虚拟存贮器

- 一般原理
- 地址映象算法
  - 典型：全相联
- 地址变换：页表
- 注意装入位
- 注意如何查页表
- 会计算虚地址的虚页号

页式虚拟存贮器

- 地址映象
- 地址变换
- 页面失效
- 页面争用或实页冲突
- 外页表和内页表的区别
- 快表和慢表的区别及关系

页式虚拟存贮器

- 替换算法
  - 先进先出算法
  - LRU算法
  - 优化替换算法
- 堆栈型算法及其特点

页式虚拟存贮器

- 重要结论：
  - LRU、OPT为堆栈型算法，FIFO不是。
  - 堆栈型算法的命中率随分配实页数的增加而单调上升，至少不会下降。
  - FIFO为非堆栈型算法，实页数的增加有时可能会降低命中率。
  - 可以用堆栈实现堆栈型替换算法。

页式虚拟存贮器

- 提高虚拟存贮器的等效访问速度的措施：
  - 提高主存命中率
    - 程序地址流、页面调度策略、替换算法、页面大小、以及分配给程序的实页数（主存容量）等都会影响主存命中率。
  - 两个存储器的速度不要相差太大
  - 加快内部地址变换
    - 相联目录表
    - 增加快表
    - 散列

页式虚拟存贮器

- 加快内部地址变换
  - 相联目录表：
    - 按内容访问的小容量相联存储器构造目录表
  - 快表：用快速硬件构成小容量的“相联目录表”，存放当前正在使用的虚实地址映象关系
    - 查表时，同时查快表和慢表，并将快表中不存在的内容从慢表中调入
    - 显然，这里也要用到替换算法，一般也采用LRU
  - 散列函数
    - 把相联访问变成按地址访问，从而加大快表容量

高速缓冲存贮器Cache

- Cache特点
- 地址映象与变换，掌握：
  - 全相联映象与变换
  - 直接映象与变换
  - 组相联映象与变换
- 替换算法：LRU
- Cache全部采用硬件实现
- Cache一致性

高速缓冲存贮器Cache

全相联映象与变换

- 特点
  - 主存中的任意一块均可以装入到Cache内的任意一块位置上
- 实现
  - 采用目录表硬件方式实现

高速缓冲存贮器Cache

直接映象与变换

- 特点
  - 主存中的每一块只能装入到Cache内唯一一个指定的块位置上
- 实现
  - 设置一个按地址访问的表存贮器（区号表），存放Cache中每一块目前被主存中哪个区的对应块所占用

高速缓冲存贮器Cache

组相联映象与变换：(重点 ： 位选择组相联)

- 特点
  - 将主存和Cache划分成同样大小的块
  - 将Cache分成Q组，每个组有S块
  - 将主存分成E个分区，每个区有Q个块（主存每个分区中的块容量与Cache中的组容量相等）。
  - 主存块到Cache组采用直接映像方式；由于组内有多个Cache块，该主存块在对应的组内部采用全相联映像方式，即组内随便放。
- 组相联映象介于全相联映象和直接映象之间
  - 若S = Cache的块数，就变成了全相联映象
  - 若S = 1，就变成了直接映象

高速缓冲存贮器Cache

- Cache替换算法
  - 典型：LRU
  - 硬件实现：
    - 堆栈法
    - 比较对法：会写表达式
- 直接映像方式需要使用替换算法？
- 何时进行替换？
  - 块失效 且 块冲突 时

高速缓冲存贮器Cache

- Cache的透明性
- 实现Cache一致性的两种“写”策略
  - 写直达法
  - 写回法
- 两种 “写”调块方法
  - 不按写分配法
  - 按写分配法

高速缓冲存贮器Cache

- Cache的预取算法
  - 按需取：在出现Cache不命中时，把一个块取到Cache中来
  - 恒预取：无论Cache是否命中，都把下一块取到Cache中
  - 不命中预取：当Cache不命中，把本块和下一块取到Cache中

基本Cache优化方法

- 降低失效率
  - 1、增加Cache块的大小
  - 2、增大Cache容量
  - 3、提高相联度
- 减少失效开销
  - 4、多级Cache
  - 5、使读失效优先于写失效
- 缩短命中时间
  - 6、避免在索引缓存期间进行地址转换

Cache - 主存 - 辅存三级层次

可以有如下几种做法：

- 1、两个存储系统组织方式。
- 2、一个存储系统组织方式。
- 3、全Cache系统。

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。

第5章

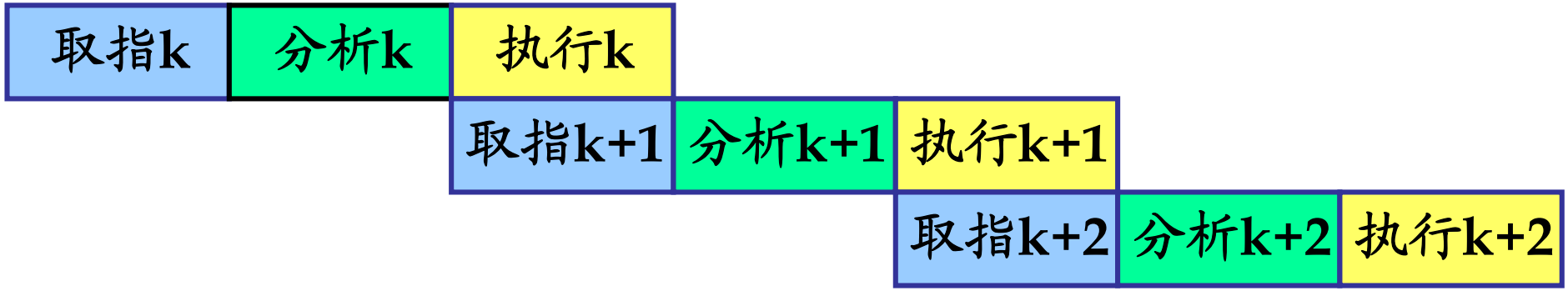
流水线和向量处理机

第5章 流水线和向量处理机

- **计算：** 加速比，吞吐率，效率，时空图，调度 等
- **重点：**
  - 一次重叠及相关处理
  - 流水方式的基本概念，流水线处理机的主要性能，相关处理
  - 非线性流水线的调度，性能
  - 向量的处理方式

重叠及相关处理

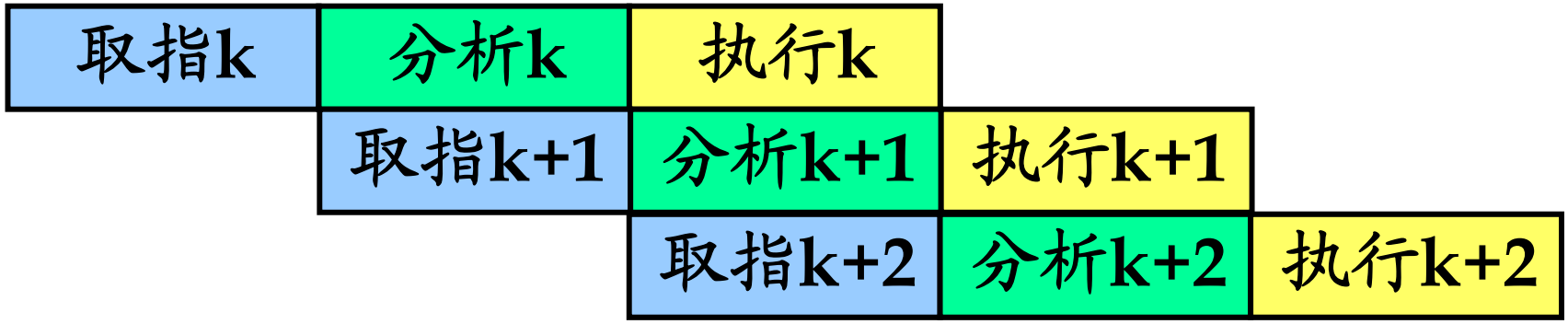
- 一次重叠定义及特点
  - 在第K条指令完成之前就开始执行第K+1条指令
  - 任何时间，指令分析部件和指令执行部件都只有相邻的两条指令在重叠解释



如果三个过程的时间相等，都为t，则执行n条指令的时间为：T=(1+2n)t

重叠及相关处理

- 二次重叠
  - 每次重叠执行三条指令



如果三过程的时间相等，执行n条指令的时间为：T=(2+n)t

重叠及相关处理

- 重叠对计算机组成的要求
  - 访存冲突
  - 功能部件冲突
  - 同步
  - 转移
  - 相关

重叠及相关处理

- 先行控制
  - 解决了分析与执行时间不等长问题。
  - 分析和执行部件可同时处理两条不相邻指令。
  - 采用技术：缓冲技术 + 预处理技术
  - 硬件要求
    - 增设指令缓冲栈，消除取指过程；
    - 增设数据缓冲栈，保证不同指令的读、写操作并行；
    - 增设先行操作栈，保证执行部件能连续执行。

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。

重叠及相关处理

- 相关
  - 定义：邻近指令之间出现了某种关联，为了避免出错而使得它们不能被同时解释执行的现象。
  - 原因：先写后读。
  - 类型：
    - 指令相关
    - 主存数相关
    - 寄存器组相关：二次相关

重叠及相关处理

- 相关解决方法：根据相关类型，采取不同的方法
- 指令相关：
  - 不允许修改指令；
  - 允许修改指令，但改变指令的执行方式，将指令相关转换为数相关。
- 主存数相关：
  - 推后读
- 寄存器组数和变址值相关（注意不同之处）
  - 推后读
  - 设置相关专用通路

流水方式

- 流水方式的特点
  - 将指令的执行过程分解得更细，分为更多的子过程，并让每个子过程分别由独立的子部件完成
  - 按时间重叠方法，可以同时处理多条指令
  - 机器的最大吞吐率成倍提高

流水方式

- 流水线的分类：
  - 单功能流水线，多功能流水线
  - 多功能：静态流水线，动态流水线
  - 线性流水线，非线性流水线
  - 等

流水线性能

- 流水线的性能：
  - 吞吐率（会计算）
  - 加速比（会计算）
  - 效率（会计算）
- 吞吐率
  - 最大吞吐率：满负荷时达到
  - 实际吞吐率

流水线性能

- 最大吞吐率
  - 它受限于流水线中最慢子过程所需要的时间
  - 称流水线中经过时间最长的子过程为瓶颈子过程
  - 消除瓶颈子过程的方法有：
    - 细分瓶颈过程
    - 重复设置瓶颈功能段

流水线性能

- 实际吞吐率TP
  - 指流水线单位时间内实际处理的指令条数或输出的结果数
  - 若流水线在$T_{流水}$时间内处理了n条指令，则 $TP = n/T_{流水}$

流水线性能

- 掌握线性流水线的性能计算方法

$$TP_{max} = 1/\max\{\Delta t_1, \Delta t_2, \Delta t_3, \Delta t_4, \ldots\}$$

$$TP = n/T_{流水}$$

$$\eta = \frac{\eta_1 + \eta_2 + \ldots + \eta_m}{m} = \frac{m \cdot \eta_0}{m} = \frac{m \cdot n \cdot \Delta t_0}{m \cdot T}$$

$$加速比 = T_{顺序执行}/T_{流水}$$

流水线性能

- 从时—空图上来看，效率实际上就是n个任务所占用的时空区面积与m个段所占用的总的时空区的面积之比

$$TP = \frac{\text{任务数}}{\text{从开始流入到n个任务全部流出的时间}}$$

$$\eta = \frac{\text{n个任务的加权时—空区}}{\text{m个段的总的加权时—空区}}$$

流水线性能

- 要求：
  - 熟练掌握时空图的画法，合理安排任务输入，进行正确的计算，包括吞吐率，效率，加速比等

流水线相关与中断

- 流水线相关分为两类：
  - 局部性相关
  - 全局性相关
- 局部性相关解决方法有两种：
  - 推后读
  - 设置相关专用通路

流水线相关与中断

- 全局性相关解决方法：
  - 关键是能正确判断判断转移分支。方法：
  - 猜测法
  - 加快和提前形成条件码
  - 采用延迟转移
  - 加快短循环程序的处理

流水线相关与中断

- 中断处理
  - 不精确断点法
  - 精确断点法
  - 正确理解“不精确”和“精确”

流水线的调度

1，线性流水线的调度

- 线性流水线无反馈，每个任务只通过每个段一次，不会发生冲突，因此只需每隔 **Δt** 输入一个任务即可。

2，非线性流水线的调度

- 非线性流水线段间有反馈回路，如果每拍向流水线输入任务，将会发生资源冲突

流水线的调度

3，二维预约表调度方法（掌握）

- 用于非线性流水线的调度
- 它利用类似时—空图的方法，得到一个任务使用流水线各段的时间关系表
- 如果任务在第n拍用到K段，就在相应第n列和第K行的交叉点处用√表示

流水线的调度

3，二维预约表调度方法（掌握）

- 5个步骤：
  - ① 构建延迟禁止向量F
  - ② 构建初始冲突向量 C
  - ③ 计算流水线的新的冲突向量
  - ④ 构造用冲突向量表示的流水线状态图
  - ⑤ 确定调度方案

流水线的调度

- 根据预约表可以得出一个任务使用各段所需间隔的拍数。间隔这些拍数就会产生争用
- 将流水线中所有各段对一个任务流过时会产生争用的节拍间隔数汇集在一起，构成**延迟禁止向量F**

流水线的调度

- 根据延迟禁止向量F 可以得到任务刚进入流水线时的初始冲突向量

  $C（C_n\ C_{n-1}…C_i…C_1）$

  - n=禁止向量F中的最大值
  - $C_i = 1$ 表示间隔i拍会产生冲突
  - $C_i = 0$ 表示间隔i拍不会产生冲突

流水线的调度

- 随着流水线中的任务每拍向前推进一个功能段，原先禁止后续任务进入流水线的间隔拍数应相应地减1，这意味着可以将冲突向量放在一个移位器中，每拍右移1位，让左面移出的空位补“0”

流水线的调度

- 若让第三个任务流入流水线后，即不与第一个任务发生功能段冲突，也不与第二个任务发生功能段冲突，新的冲突向量就应当是第一个任务当前的冲突向量与第二个任务初始的冲突向量按位“或”

流水线的调度

- 按照这样的思路，选择各种可能的拍数输入信任务，又可以产生新的冲突向量，一直到不再产生不同的冲突向量为止，这样就考虑了所有的可能性
- 由此可以画出用冲突向量表示的流水线状态转移图

流水线的调度

- 只要从状态图中的初始状态出发，找到一个间隔拍数呈周期性重复的方案，并按此方案进行调度，就不会发生功能段冲突。
- 显然，可以找到多个调度方案
- 但哪一种调度方案最佳呢？
- 所谓最佳调度方案是指：
  - 流水线吞吐率最高
  - 控制简单

流水线的调度

- 根据上述状态图，可以找出所有的调度方案，并计算出每种方案的平均间隔拍数。平均间隔拍数最小的就是吞吐率最高的
- 为了简化控制，也可以采用等间隔调度，但常常会使吞吐率和效率下降。

向量的流水处理

- 向量处理机定义
- 向量的处理方式主要有三种：
  - 水平（横向）处理方式
  - 垂直（纵向）处理方式
  - 分组（纵横）处理方式
- 哪种适合流水处理？

指令级高度并行的超级处理机

- 代表性的例子是：
  - 超标量（Superscalar）处理机
  - 超流水线（Superpipelining）处理机
  - 超标量超流水处理机
  - 超长指令字（VLIW）处理机
- 要求：会画图，会计算（加速比等）

指令级高度并行的超级处理机

- 常规的标量流水线单处理机是在每个△t时间内解释完一条指令。称这种流水机的度为1。
- 超标量处理机采用多指令流水线，在每个△t时间内同时解释完m条指令。称这种流水机的度为m。

指令级高度并行的超级处理机

机器类型	k段流水线基准标量处理机	m度超标量	n度超流水线	(m,n)度超标量超流水
机器流水线周期	1个时钟周期	1	$1/n$	$1/n$
同时发射指令条数	1条	m	1	m
指令发射等待时间	1个时钟周期	1	$1/n$	$1/n$
指令级并行度ILP	1	m	n	$m\times n$

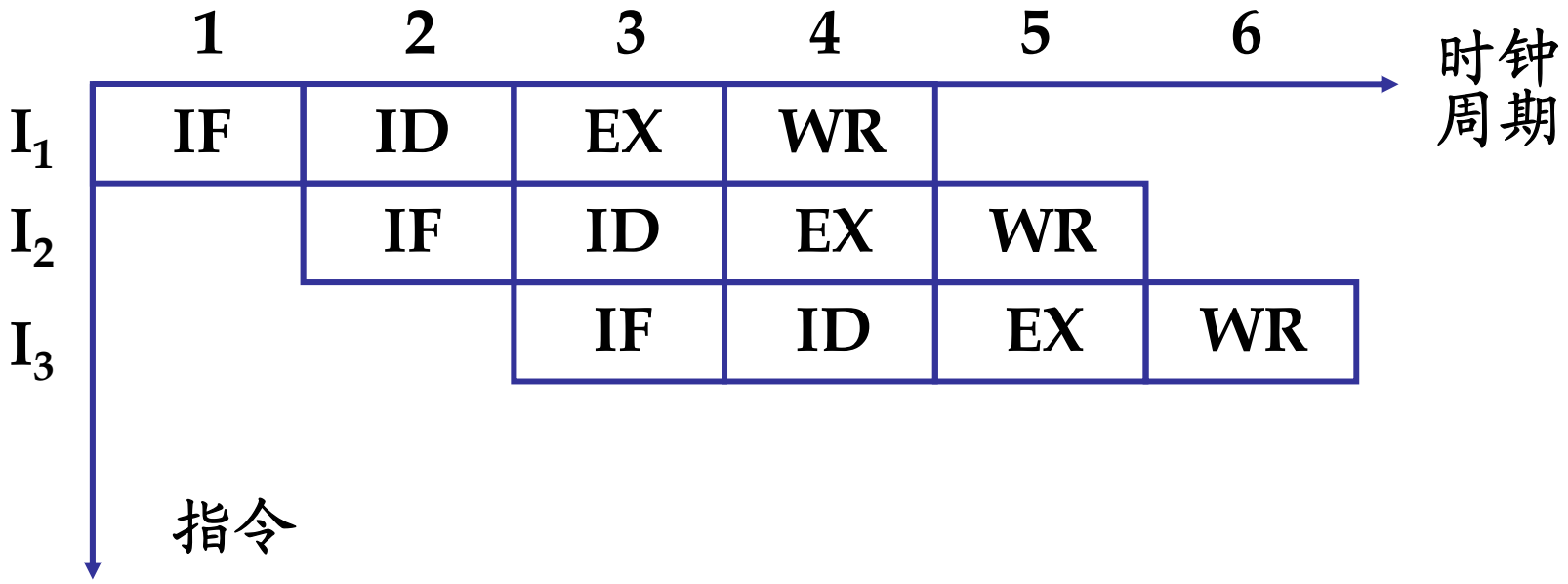
超标量、超流水、超标量超流水处理机的主要性能

指令级高度并行的超级处理机

- 单发射
  - 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。

指令级高度并行的超级处理机

单发射处理机的指令流水线时空图



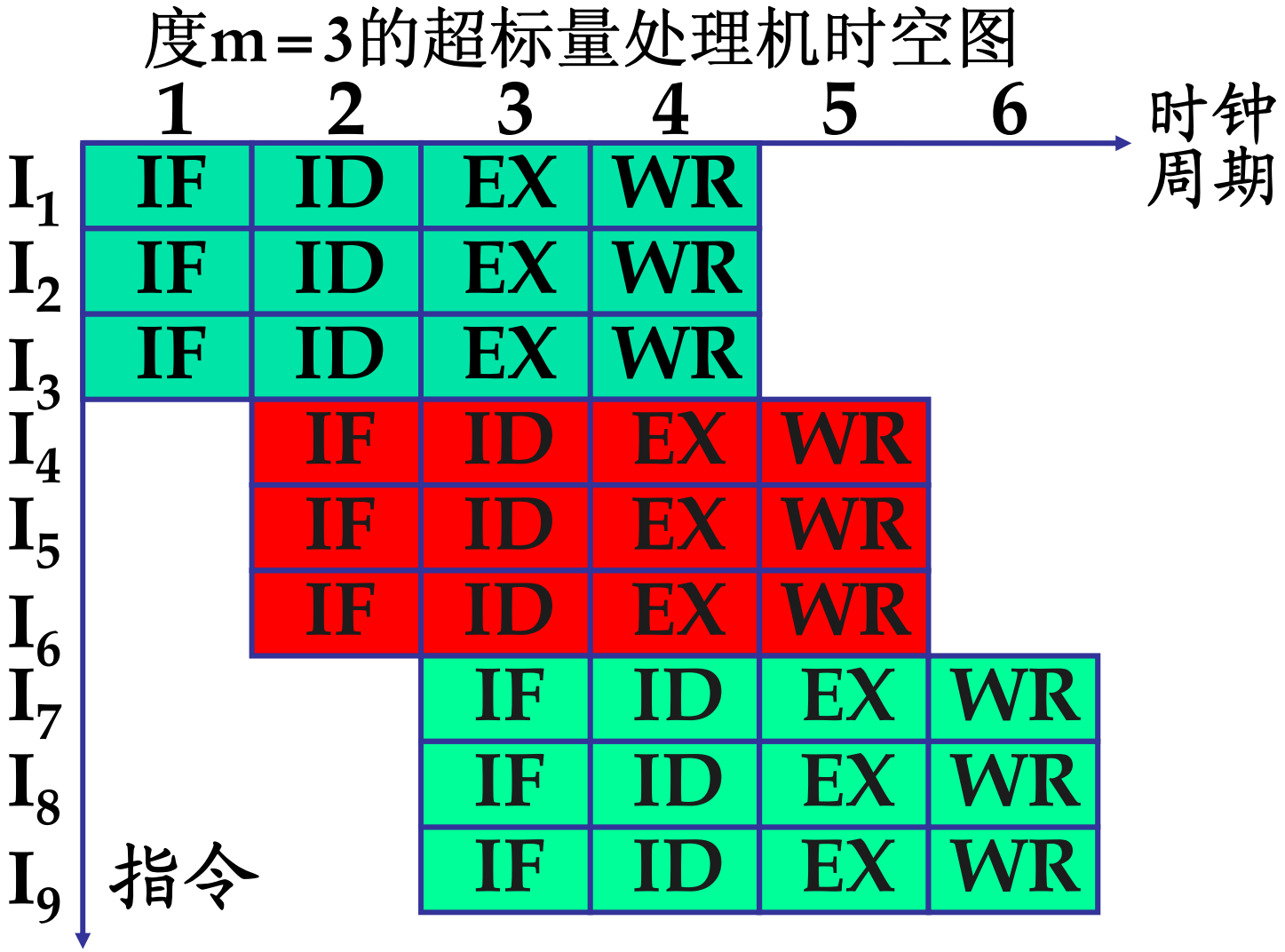
指令级高度并行的超级处理机

- 多发射
  - 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。

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。

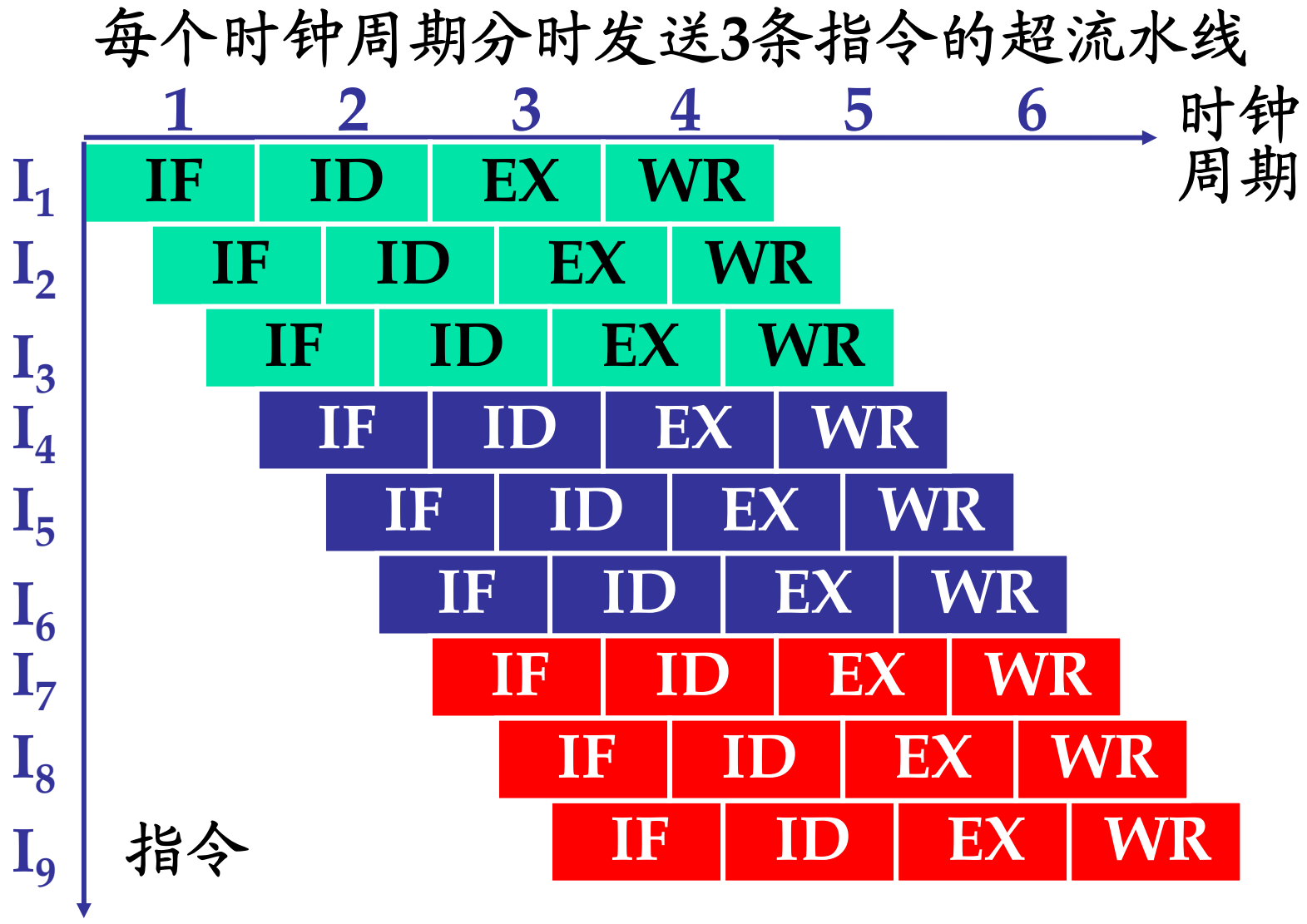
超标量处理机


度m=3的超标量处理机时空图
1
2
3
4
5
6
时钟周期
I1
I2
I3
I4
I5
I6
I7
I8
I9
指令
IF
ID
EX
WR

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。
- 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

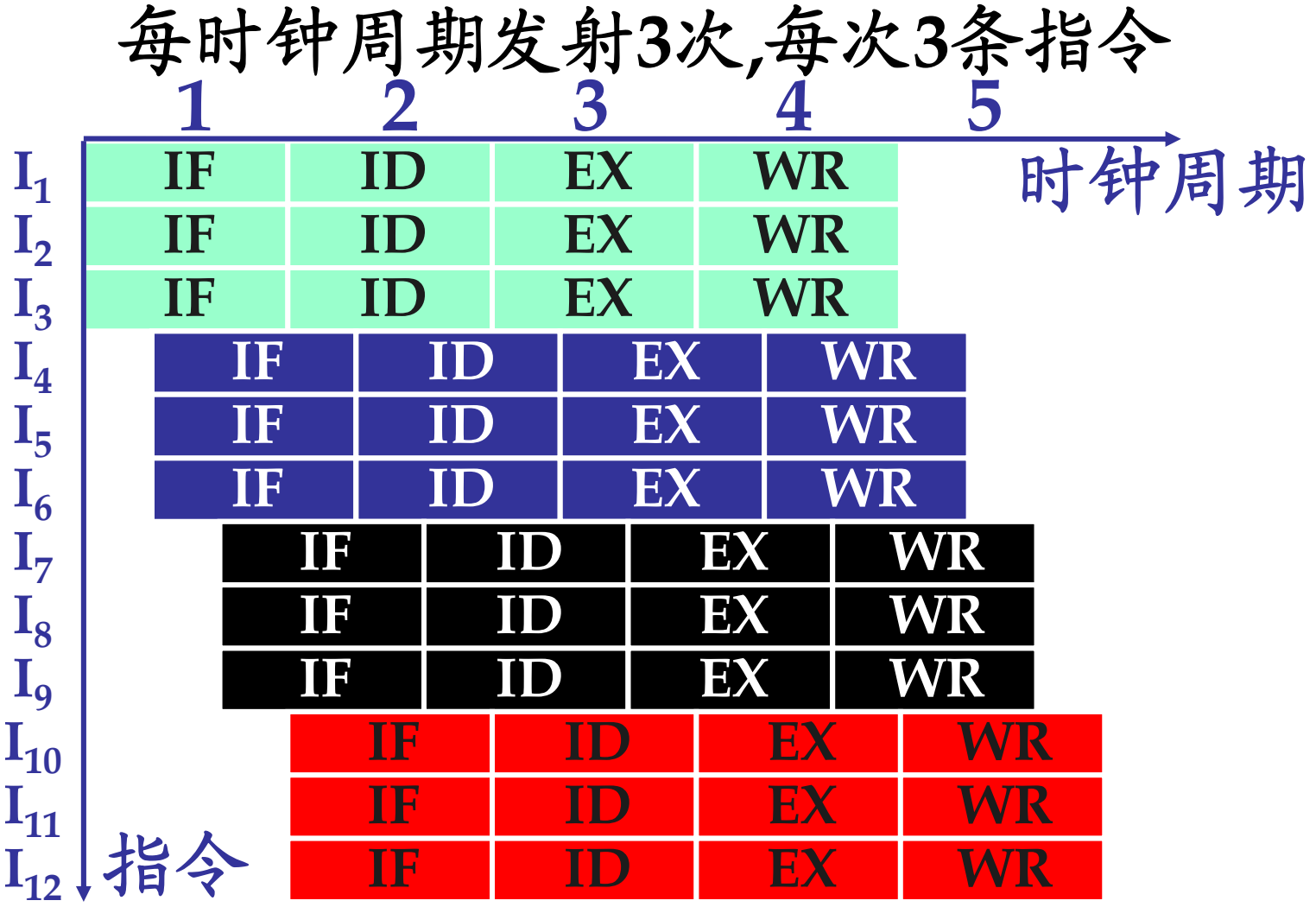
超流水线处理机


每个时钟周期分时发送3条指令的超流水线
1
2
3
4
5
6
时钟周期
I1
I2
I3
I4
I5
I6
I7
I8
I9
指令
IF
ID
EX
WR

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。

超标量超流水线处理机


每时钟周期发射3次,每次3条指令
1
2
3
4
5
时钟周期
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
I12
IF
ID
EX
WR
指令

超长指令字处理机(VLIW)

- **VLIW**将多个能并行执行的不相关或无关的操作先行压缩，组合在一起，形成一条有多个操作段的超长指令
- 运行时，这条超长指令控制多个相互独立的功能部件并行操作

第6章

并行处理机和相联处理机

第六章 并行处理机和相联处理机

重点：

- 并行处理机的定义、构形与特点
- SIMD计算机的基本单级互连网络
- 多级互连网络

并行处理机原理

- 并行（阵列）计算机是并行处理计算机中的一种重要结构
- 主要通过资源重复实现并行
- 一，并行处理机的定义：

  并行处理机也称为阵列处理机，它是一种重复设置了多个同样的处理单元（PE），按照一定方式把这些PE互相连接，在统一的控制部件（CU）作用下，PE各自对分配的数据并行地完成同一条指令所规定的操作，实现操作级并行的SIMD计算机

并行处理机原理

二，并行处理机的基本构形（掌握）

- 根据存贮器组成方式的不同，并行处理有**两种基本构形**：
  - 分布式存贮器的并行处理机
  - 集中式共享存贮器的并行处理机

并行处理机原理

三，并行处理机的特点（掌握）

- 与同样擅长向量处理的流水线处理机相比，它具有以下一些特点：
  - 利用的是资源重复，而不是时间重叠
  - 利用的是并行性中的同时性，而不是并发行
  - 使用简单而规整的互联网络ICN来确定PE之间的连接模式
  - PE之间互连灵活，在某些专门问题上的工作效率要比流水线处理机高
  - 它的专用性更强。如果习惯上把流水线处理机归属为通用计算机的话，并行处理机则被看成专用机
  - 并行处理机的结构与所采用的并行算法紧密结合

并行处理机原理

四， SIMD计算机的互联网络（了解）

- 在SIMD计算机中，PE之间以及PE与存贮体都要通过互联网络（ICN）进行信息交换
- ICN限定了并行处理机适用的解题算法的类型，也对整个系统的性能产生明显影响
- 因此，ICN是设计重点

并行处理机原理

1，SIMD互联网络的设计目标

- 结构不要复杂，以降低成本
- 互连要灵活，以满足算法和应用需要
- PE之间传送信息的步数尽可能少，以提高速度
- 可以用一系列规整单一的基本构件或其组合实现，模块性好，便于用VLSI实现，并满足系统的可扩充性

并行处理机原理

2，确定PE之间通信的互联网络的因素包括：

- 操作方式
- 控制策略
- 交换方法
- 网络拓扑结构等

设计互连网络时应考虑的问题

操作方式
- 同步：SIMD采用
- 异步：多处理机采用
- 同步异步组合：多处理机采用

控制方式
- 集中式：SIMD采用。集中控制全部开关
- 分布式：多处理机采用？

交换方式
- 线路交换：SIMD采用硬联线路交换，大批数据传输
- 包交换：多处理机和计算机网络采用
- 线路/包交换组合：ATM

设计互连网络时应考虑的问题

拓扑结构

- 静态：连接固定，灵活性差，适应性差
- 动态：
  - 单级：有限几种连接，须多次循环使用
  - 多级：多个单级网路串联而成，实现任意连接
    多级网络循环使用，实现复杂连接

并行处理机原理

3，网络拓扑结构

- 网络拓扑结构指互连网络输入、输入端可以实现连接的模式
- 有两种模式：
  - 静态
  - 动态

并行处理机原理

- 在静态连接模式中，两个PE之间的连接是固定的，不能重新配置成与其他PE连接
- 在动态连接模式中，PE之间的连接可以通过设置开关单元的状态重新配置

并行处理机原理

- 动态网络拓扑结构有两类：
  - 单级网络
  - 多级网络
- 动态单级网络的一级只有有限几种连接，因此必须经过多次循环通过，才能实现任意两个PE之间的通信，故也称其为循环网络
- 动态多级网络由多个动态单级网络串联组合而成，以实现任意两个PE之间的通信
- 在此基础之上，还可以将多级网络循环使用，以实现更复杂的互连

互连网络特性

- 网络规模
- 结点度
- 距离
- 网络直径
- 结点间的线长
- 等分宽度
- 对称性

互连网络性能

- 频带宽度(Bandwidth)
- 传输时间(Transmission time)
- 飞行时间(Time of flight)
- 传输时延(Transport latency)
- 发送方开销(Sender overhead)
- 接收方开销(Receiver overhead)

总时延＝发送方开销＋飞行时间＋
消息长度/频宽＋接收方开销

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**3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**

互连函数

- 为了反映不同互连网络的连接特性，可以用一组互连函数来定义互连网络
- 设互连网络有N（$N=2^n$）个入端和N个出端，则可以用0，1，2，... N-1标识每个端点。互连函数就表示了出端号和入端号的一一对应关系，即

  出端号i = f(入端号j) i,j=0, 1, 2, ... N-1

互连函数

- 互连函数可以直接用节点之间的连线图来表示，但很繁琐，也难以体现连接上的内在规律，因此常把入端**x**和出端**f(x)**用二进制数编码，从二者的二进制数编码上找出规律

互连函数

- 基本的单级互连网络
  - 立方体
  - **PM2I**
  - 混洗交换

互连函数

1，立方体单级互连网络

- 立方体单级互连网络源于立方体结构
- 8个结点的立方体有3个互连函数：
  - $Cube_0$
  - $Cube_1$
  - $Cube_2$

互连函数

- 推广到n维情况。N个节点的立方体单级互连网络共有$n=\log_2 N$种互连函数，即

$$Cube_i(P_{n-1}\ldots P_i\ldots P_1P_0)=P_{n-1}\ldots\overline{P_i}\ldots P_1P_0$$

- 当n>3时，称之为超级立方体网络
- 超级立方体网络的最大距离为n，即反复使用单级网络，最多经过n次就可以实现任意一对入端和出端的连接，而且任意两个节点之间至少有n条不同的路径，容错性很强

互连函数

2，PM2I单级互连网络

- **PM2I是Plus-Minus 2（加减2）单级网络的简称，能实现j号PE直接与j±2号PE连接，即**

$$PM2_{+i}(j) = j + 2^i \bmod N$$

$$PM2_{-i}(j) = j + 2^i \bmod N$$

$$0 \le j \le N, 0 \le i \le n, n = \log 2N$$

- **因此，它共有2n-1个互连函数**

互连函数

- PM2I比立方体单级网络灵活
- PM2I单级网络的最大距离为[n/2]
- 对于n=3，最多使用两次，既可以实现任意一对入端出端的连接

互连函数

3，混洗交换单级互连网络

- 混洗交换(shuffle-Exchange)单级互连网络包含两个互连函数：
  - 全混(Perfect Shuffle)
  - 交换(Exchange)

互连函数

- 全混的连接规律是：
  - 把全部按编码顺序排列的PE从中间分为两半，前一半和后一半在连接至输出端时正好一一隔开，如同洗扑克牌一样
  - 其互连函数为（左移）：

$$Shuffle(P_{n-1}P_{n-2}\cdots P_1P_0)=(P_{n-2}P_{n-3}\cdots P_1P_0P_{n-1})$$

其中 $n=\log_2 N$

互连函数

- **Shuffle函数有两个重要特征：**
  - **与Cube不同，shuffle函数是不可逆的**
    - 若把入端当出端，出端当入端，则原网络变成另外一个网络
  - **为实现全0和全1 PE与其他PE的任意连接，还必须增加$Cube_0$交换函数，这样就得到了全混交换单级网络**
  - **全混交换单级网络的最大距离为2n-1**

互连函数

- 多级互连网络

单级互连网络
- 1. 立方体
- 2. PM2I
- 3. 混洗
- 4. 蝶形

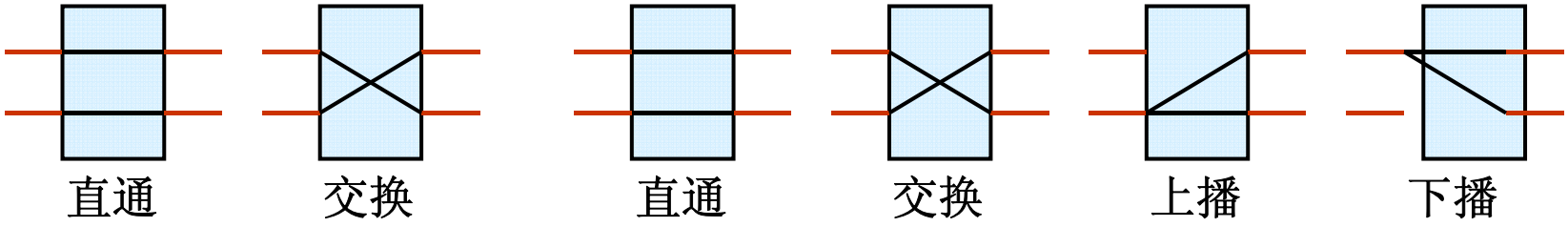
多级互连网络
- 1. 多级立方体
- 2. 多级PM2I
- 3. 多级混洗(Omega)

互连函数

- 多级互连网络
  - 立方体 单级互连网络
  - **PM2I** 单级互连网络
  - 混洗交换 单级互连网络
  - 蝶形 单级互连网络
- 多级互连网络采用的关键技术：
  - **(1)** 交换开关
  - **(2)** 交换开关之间的拓扑连接
  - **(3)** 对交换开关的控制方式

交换开关

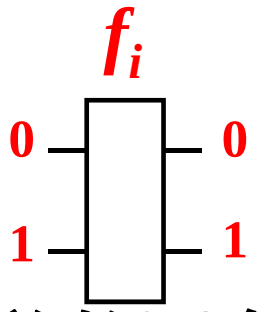
- 具有直通和交换两种功能的交换开关称为二功能开关，或交换开关。用一位控制信号控制。
- 具有所有四种功能的交换开关称为四功能开关，用两位控制信号控制。



(a) 1位控制信号　　(b) 2位控制信号

直通实现恒等置换

交换开关的不同控制方式



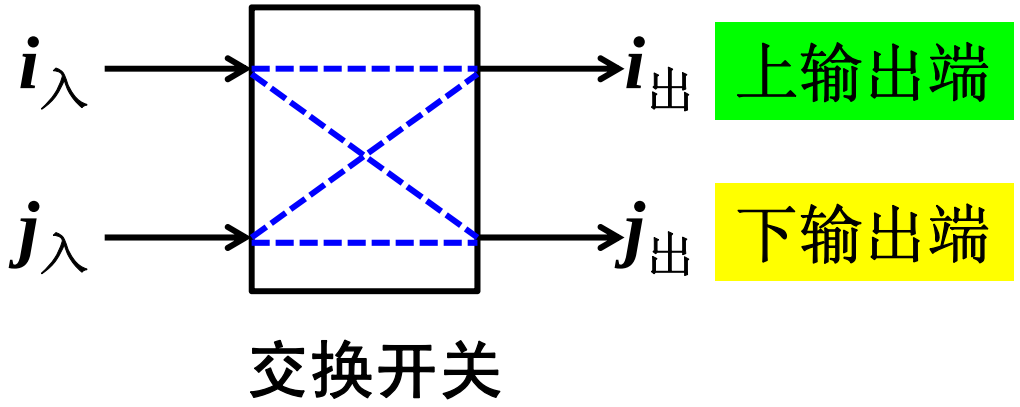
- 按级控制和交换置换(1/2):
  - 在右上图有一个开关控制示意图。假定开关的两个输入端分别标注以“0”和“1”，同样也给输出端标上“0”和“1”的标注。在直送方式下，0→0，1→1；而在交叉方式下，则有0→1，1→0。如果将这种传输情况看作二进制运算，那末控制信号 fi 就是参与逻辑运算的一个变量，其逻辑关系可以表示为：

$$E(x_i) = x_i \oplus f_i$$

fi =0时，表示直送； fi = 1 时，表示交叉。

交换开关的不同控制方式

- **按级控制和交换设置(2/2)：**
  - 采用单元控制方式时，控制信号为0时，2×2开关的输入端与上输出端连接；为1时，输入端与下输出端连接。



交换开关的不同控制方式

- 多级立方体网络
  - 会画图
  - 会根据级控制信号找到入端和出端的对应关系
- 多级混洗交换网络(Ω网)
  - 会画图
  - 会根据级控制信号找到入端和出端的对应关系

第7章

多处理机

第七章 多处理机

重点：

- 多处理机概念与特点，结构，与并行处理机的差别，主要技术问题
- 多核处理器
- 多Cache一致性
- 多处理机操作系统的特点与类型

多处理机的定义与特点

1，多处理机定义（掌握）

- 多处理机具有两台以上的处理机。这些多处理机在逻辑上统一的操作系统控制下，通过共享主存或输入输出子系统或高速通信网络进行通信，实现作业、任务、指令、数据各个级别的并行

多处理机的定义与特点

2，类型（掌握）

- 由于应用的目的和结构的不同，多处理机系统有3种类型：
  - 同构型多处理机系统
  - 异构型多处理机系统
  - 分布式多处理机系统

多处理机的定义与特点

3，特点（掌握）

- 多处理机属于MIMD系统，实现的是作业、任务之间的并行

多处理机的定义与特点

4，与并行处理机的差别（了解）

- 多处理机与属于SIMD的并行处理机有很大差别
- 这种差别是由于并行性等级不同造成的
- 主要差别有以下几个方面，见下表

多处理机的定义与特点

	多处理机	并行处理机
结构灵活性	可以适应多种算法，结构更灵活，以实现复杂的机间互连	主要针对向量和数组处理，有一定的专用性，互连通路有限且相对固定
程序并行性	实现作业、任务、指令的并行，并行性存在于指令外部，程序并行性较难识别	实现操作级并行，一条指令即可处理整个数组，并行性存在于指令内部，识别容易

多处理机的定义与特点

	多处理机	并行处理机
并行任务派生	需要专门的指令或语句指明程序段之间的并发关系，一个任务执行时，可以派生出与它并行的另一些任务	通过指令反映数据之间的并行，由指令直接启动多个PE工作

多处理机的定义与特点

	多处理机	并行处理机
进程同步	由于实现的是作业、任务、指令的并行，同一时刻，不同处理机执行不同指令，进度不一。当然若并发程序之间有相关和控制以来，就要采取同步措施	由于实现的是指令内部对数据操作的并行，所有活跃的PE在同一个CU控制下，同时执行同一条指令，工作自然是同步的

多处理机的定义与特点

	多处理机	并行处理机
资源分配和任务调度	执行并发任务时，所需要的处理机数目不固定，需要解决好资源分配和任务调度，均衡负载，提高硬件利用率，防止死锁等	执行向量数组运算，PE数目固定。程序员编写程序时，可以利用屏蔽手段设置PE的活跃状态，这样就可以改变参加并行执行的PE数目

多处理机主要技术（了解）

- 结构灵活性与通用性
- 进程通信方法：共享内存、互连网络、消息传递机制等
- 运行模型：数据并行 + 处理并行
  - 基于共享内存的运行模型
  - 基于消息传递机制的运行模型
- 并行性表达：
- 并行算法

多处理机的硬件结构（了解）

- 多处理机主要有两种硬件结构：
  - 紧耦合硬件结构
    - 紧耦合多处理机
    - 通过共享主存实现处理机之间的通信
    - 通信速率受限于主存的频宽
  - 松耦合硬件结构
    - 松耦合多处理机
    - 每台处理机都有一个容量较大的局部存贮器，用于存贮常用的指令和数据
    - 不同处理机之间通过通道或者通过消息传送系统实现通信，交换信息

多处理机的硬件结构

- 从存储器的分布和使用上看，多处理机系统分为两种结构：
  - 共享存储器结构
  - 分布式存储器结构

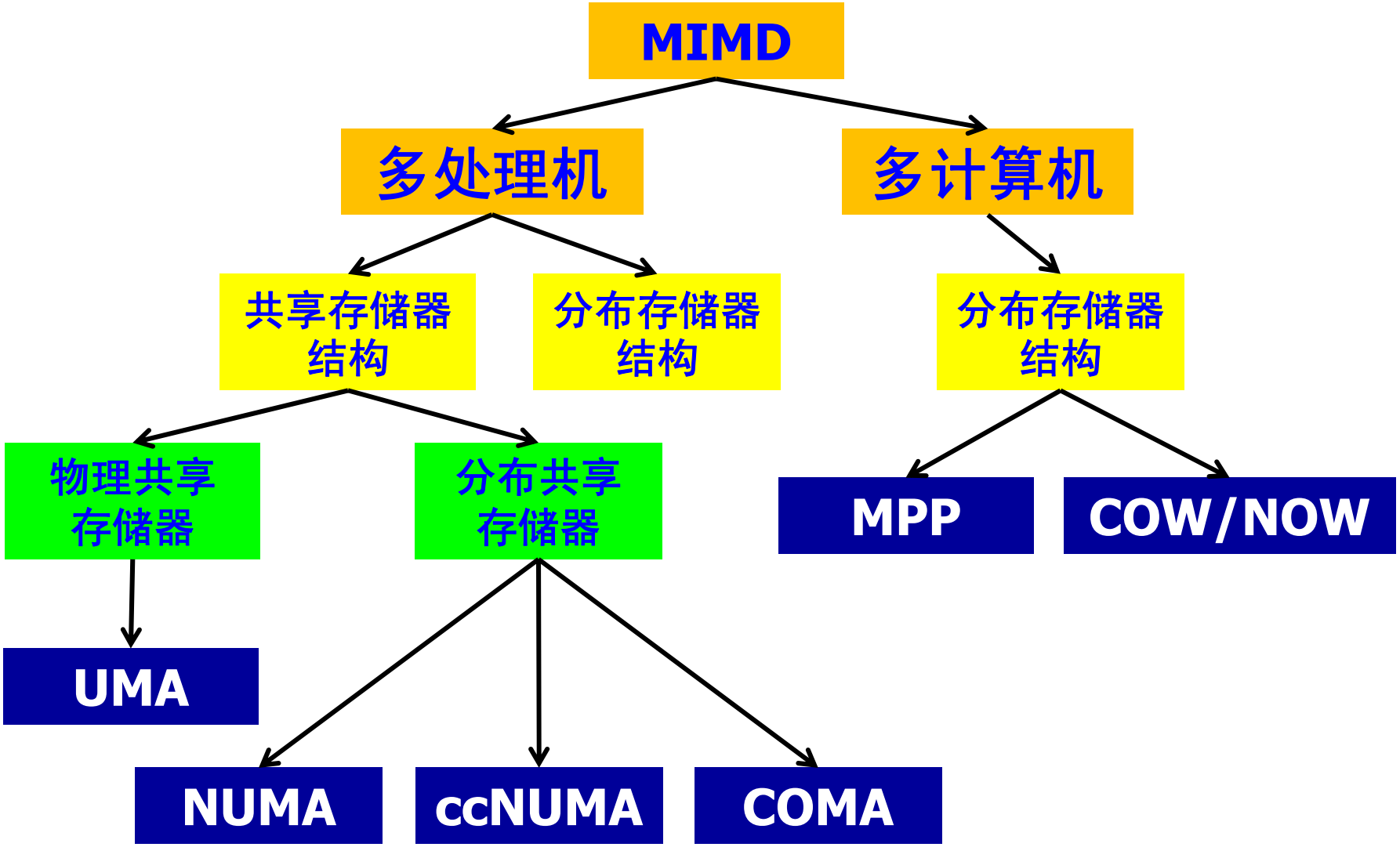
共享存储器结构（了解）

- 根据共享存储器实现方式不同又细分为：
  - UMA结构
  - NUMA结构
    - ccNUMA
    - NCC-NUMA
  - COMA结构

分布式存储器结构（了解）

- 也被称为非远程存储访问 (NORMA，No-Remote Memory Access) 模型
- 各处理机拥有自己的本地存储器，在本地操作系统控制下独立工作。
- 各处理机的本地存储器是私有的，不能被其他处理机访问。
- 各处理机借助互连网络、通过消息传递机制相互通信，实现数据共享。
- 大规模并行处理机（MPP）、机群（cluster）等采用了这种结构。

MIMD计算机分类


MIMD
多处理机
多计算机
共享存储器结构
分布存储器结构
分布存储器结构
物理共享存储器
分布共享存储器
MPP
COW/NOW
UMA
NUMA
ccNUMA
COMA

多核处理器（了解）

- 定义：
  - 集成两个或以上独立处理单元（核）的处理器
- 结构：
  - 同构多核，异构多核；
  - Cache设置与访问 等
  - 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Cache一致性（了解）

- 不一致的三种原因：
  - 共享可写数据
  - 进程迁移
  - I/O传输
- 解决方法：2类
  - 基于硬件的方法
    - 思想：跟踪共享数据的状态
    - Cache一致性协议：监听协议和基于目录的协议
  - 基于软件的方法
    - 由编译和操作系统检测并解决

两种Cache一致性协议（了解）

1. 监听协议(Snooping Protocol) — 拥有数据副本的Cache各自记录数据块的状态，无集中的状态。

2. 基于目录的协议(Directory based Protocol) — 将每个数据块的共享状态记录保存在某个地点 — 目录

监听协议 (Snoopy Protocol)

- 适用于：具有广播能力的总线结构多机系统。但只适用于小规模的多处理机系统
- 分布式算法：分散到所有Cache控制器。各Cache控制器自行产生状态变化及相应操作
- 两种策略：
  - 写无效（Write-Invalidate）（或 写作废）
  - 写更新（Write-Update）（或 播写法）

基于目录的协议

- 在每个结点增加目录存储器，用于存放目录。
- 目录必须跟踪记录每个共享数据块的状态。
- 目录里存放：
  - 有关Cache副本驻留在哪里的信息：所有共享数据块的所有Cache副本的地址表。
- 目录的存放方式：2种
  - 集中式目录方式（中心目录）
  - 分布式目录方式
- 目录形式：
  - 全映射目录，有限目录，链式目录

多处理机之间的互连形式（了解）

- 处理机之间的互连形式是决定处理机性能的一个重要因素
- 对互连的要求是：
  - 通信速率高
  - 成本低
  - 灵活多样，以实现各种复杂的、甚至不规则的互连模式，不发生冲突

多处理机之间的互连形式

主要有：

- 总线形式
- 环形形式
- 交叉开关形式
- 多端口寄存器形式
- 开关枢纽形式 等
- 当处理机数量较多时，也有类似SIMD的多级网络形式

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；
- 程序段间的相关：
  - 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

- 任务粒度的大小会显著影响多处理机的性能和效率
- 并行性在很大程度上依赖于R/C比值
  - R — 程序用于有效计算的执行时间
  - C — 处理机间通信等辅助开销时间
- 通常：
  - R/C比值小，细粒度并行，并行性低
  - R/C比值大，粗粒度并行，并行性高

为获得最佳性能，应对并行性和额外开销大小进行权衡，也要与应用问题的粒度取得适配。

基本性能

- 假设有一个包含 M 个任务的应用程序，希望在一个由 N 台处理机组成的系统上以最快的速度执行这个程序。
- 假设：
  - （1）每个任务的执行时间为R个单位；
  - （2）当两个任务不在同一台处理机上时，其通信所需的额外开销为C个单位时间。当两个任务在同一台处理机上时，通信所需的额外开销为0。

总处理时间＝R•max(M－K，K) ＋ C•(M－K)•K

总处理时间＝执行时间 + 通信和其它额外开销的时间

多处理机操作系统（了解）

一，多处理机操作系统的特点

- 程序执行的并行性
- 分布性
- 机器间通信与同步性
- 系统的容错性

多处理机操作系统

二，多处理机操作系统的3种类型：

- 主从型
  - 管理程序运行在一台指定的处理机上（主处理机）
- 各自独立型
  - 将控制功能分散给多台处理机，共同完成对整个系统的控制
- 浮动型
  - 这是一种介于主从型和各自独立型的折衷方式
  - 管理程序可以在处理机之间浮动。主控程序可以从一台处理机转移到另一台处理机