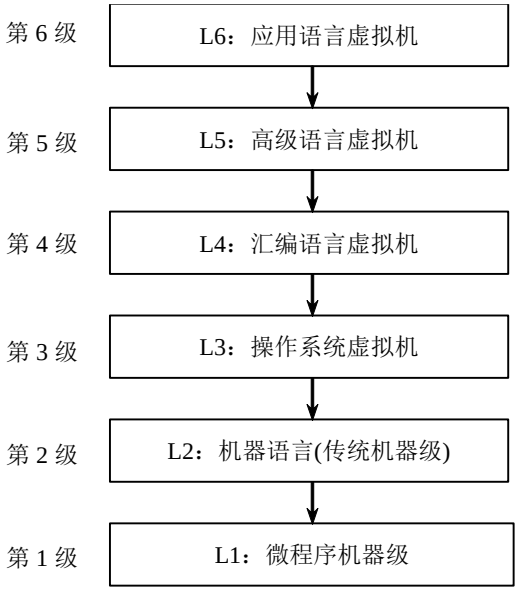
目录

第 1 章 计算机系统结构的基本概念

多级层次结构

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。



虚拟机器

由软件实现的机器。

解释

语言实现的一种基本技术。每当一条 N+1 级指令被译码后，就直接去执行一串等效的 N 级指令，然后再去取下一条 N+1 级的指令，依此重复进行。

翻译

语言实现的一种基本技术。先把 N+1 级程序全部变换成 N 级程序后，再去执行新产生的 N 级程序，在执行过程中 N+1 级程序不再被访问。

计算机系统结构

程序员所看到的计算机的属性，即概念性结构与功能特性。

经典计算机系统结构概念的实质

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。

透明性

在计算机技术中，对本来存在的事物或属性，但从某种角度看又好象不存在的概念称为透明

性。

计算机组成

计算机系统的逻辑实现。

计算机实现

计算机系统的物理实现。

冯氏分类法

冯氏分类法是用系统的最大并行度对计算机进行分类。它是由冯泽云先生于 1972 年提出的。

最大并行度

计算机系统在单位时间内能够处理的最大的二进制位数。可以用平面直角坐标系中的一个点代表一个计算机系统，其横坐标表示字宽（n 位），纵坐标表示一次能同时处理的字数（m 字）。$m \times n$ 就表示了其最大并行度。

Flynn 分类法

按照指令流和数据流的多倍性进行分类，它是 M.J.Flynn 于 1966 年提出的。

指令流

机器执行的指令序列。

数据流

由指令流调用的数据序列。

多倍性

在系统受限的部件上，同时处于同一执行阶段的指令或数据的最大数目。

以经常性事件为重点

对于大概率事件(最常见的事件)，赋予它优先的处理权和资源使用权，以获得全局的最优结果。

系统的加速比

对系统中的某些部件进行改进，改进后的系统性能与改进前的系统性能之比。

Amdahl 定律
加快某部件执行速度所获得的系统性能加速比，受限于该部件在系统中所占的重要性。

可改进比例
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。

部件加速比
可改进部分改进以后性能提高的倍数。它是改进前所需的执行时间与改进后执行时间的比。

程序的局部性原理
程序在执行时所访问地址的分布不是随机的，而是相对地簇聚；这种簇聚包括指令和数据两部分。

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。

程序的空间局部性
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 性能公式
程序执行的 CPU 时间= CPI × IC / 时钟频率

IC
程序执行过程中所处理的指令数。
反映了计算机指令集的结构和编译技术。

CPI
指令时钟数。
CPI = 总时钟周期数 / IC
反映了计算机实现技术、计算机指令集的结构和计算机组织。

响应时间
从事件开始到结束之间的时间，也称为执行时间。即计算机完成某一任务所花费的全部时间，包括访问磁盘、访问存储器、输入/输出、操作系统开销等。

流量
在单位时间内所能完成的工作量。

CPU 时间

CPU 为用户程序工作的时间，不包含 I/O 等待时间及运行其他程序的时间。可细分为用户 CPU 时间及系统 CPU 时间。

核心测试程序

由从真实程序中提取的较短但很关键的代码构成。

小测试程序

通常是指代码在几十行到 100 行的具有一些特定目的的测试程序。用户可以随时编写一些这样的程序来测试系统的各种功能，并产生用户已预知的输出结果，如皇后问题、迷宫问题、快速排序、求素数、计算 π 等。

合成测试程序

首先对大量的应用程序中的操作进行统计，得到各种操作比例，再按这个比例构造测试程序。Whetstone 与 Dhrystone 是最流行的合成测试程序。

基准测试程序套件

为了能比较全面地反映计算机在各个方面的处理性能，通常采用整套测试程序。这组程序称为基准测试程序套件，它是由各种不同的真实应用程序构成的。目前最成功和最常见的测试程序套件是 SPEC 系列。

事务处理测试程序

主要测试在线事务处理（On-Line Transaction Processing，OLTP）系统的性能，包括数据库访问和更新等。

存储程序计算机

冯·诺依曼结构计算机

输入/输出方式

程序控制（程序等待、程序中断）、DMA、通道、I/O 处理机

相联存储器 CAM

可按内容访问的存储器。

相联处理机

以相联存储器为核心的处理机。相联存储器除了完成信息检索任务外，还能进行一些算术逻

辑运算。

系列机

由同一厂家生产的具有相同的系统结构，但具有不同组成和实现的一系列不同型号的机器。

软件兼容

同一个软件可以不加修改地运行于系统结构相同的各档机器，而且它们所获得的结果一样，差别只在于运行时间不同。

兼容机

不同厂家生产的具有相同系统结构的计算机。

向上(下)兼容
按某档计算机编制的程序，不加修改的就能运行于比它高（低）档的计算机。

向前(后)兼容
按某个时期投入市场的某种型号机器编制的程序，不加修改地就能运行于在它之前（后）投入市场的机器。

模拟

用软件的方法在一台现有的机器（称为宿主机 host）上实现另一台机器（称为虚拟机）的指令集。

仿真

用一台现有机器（称为宿主机）上的微程序去解释实现另一台机器（称为目标机）的指令集。

并行性

在同一时刻或是同一时间间隔内完成两种或两种以上性质相同或不相同的工作。只要时间上互相重叠，就存在并行性。

同时性

两个或多个事件在同一时刻发生的并行性。

并发性

两个或多个事件在同一时间间隔内发生的并行性。

字串位串

每次只对一个字的一位进行处理。这是最基本的串行处理方式。

字串位并

同时对一个字的全部位进行处理，不同字之间是串行的。

字并位串

同时对许多字的同一位（称为位片）进行处理。

全并行

同时对许多字的全部位或部分位进行处理。

指令内部并行

单条指令中各微操作之间的并行。

指令级并行

并行执行两条或两条以上的指令。

线程级并行

并行执行两个或两个以上的线程，通常是以一个进程内派生的多个线程为调度单位。

任务级或过程级并行

并行执行两个或两个以上的过程或任务（程序段），以子程序或进程为调度单元。

作业或程序级并行

并行执行两个或两个以上的作业或程序。

时间重叠

多个处理过程在时间上相互错开，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

通过重复设置资源，尤其是硬件资源，大幅度提高计算机系统的性能。

资源共享

是一种软件方法，它使多个任务按一定时间顺序轮流使用同一套硬件设备。

同构型（对称型）多处理机

由多个同类型，至少担负同等功能的处理机组成，同时处理同一作业中能并行执行的多个任务。

异构型（非对称型）多处理机

由多个不同类型，至少担负不同功能的处理机组成，按照作业要求的顺序，利用时间重叠原理，依次对它们的多个任务进行加工，各自完成规定的功能动作。

分布处理系统

把若干台具有独立功能的处理机（或计算机）相互连接起来，在操作系统的全盘控制下，统一协调地工作，而最少依赖集中的程序、数据或硬件。

耦合度

反映多机系统各机器之间物理连接的紧密程度和交互作用能力的强弱。

松散耦合

通过通道或通信线路实现计算机间互连，共享某些外围设备，机间的相互作用是在文件或数据集一级进行。

紧密耦合

机间物理连接的频带较高，它们往往通过总线或高速开关实现互连，可以共享主存。

第 2 章 计算机指令集结构设计

堆栈型机器
其 CPU 中存储操作数的主要单元是堆栈。

累加器型机器
其 CPU 中存储操作数的主要单元是累加器。

通用寄存器型机器
CPU 中存储操作数的主要单元是通用寄存器。

三种类型指令集结构
根据 CPU 内部存储单元类型，将指令集结构分为堆栈型指令集结构、累加器型指令集结构和通用寄存器型指令集结构。

通用寄存器型指令集结构的三种类型
寄存器－寄存器型（RR：Register-Register）
寄存器－存储器型（RM：Register-Memory）
存储器－存储器型（MM：Memory-Memory）

CISC
复杂指令集计算机。

RISC
精简指令集计算机。

指令集结构的完整性
在一个有限可用的存储空间内，对于任何可解的问题，编制计算程序时，指令集所提供的指令足够使用。

指令集结构的规整性
没有或尽可能减少例外的情况和特殊的应用，以及所有运算都能对称、均匀地在存储器单元或寄存器单元之间进行。规整性主要包括对称性和均匀性。

对称性
指所有与指令集有关的存储单元的使用、操作码的设置等都是对称的。

均匀性
指对于各种不同的操作数类型、字长、操作种类和数据存储单元，指令的设置都要同等对待。

面向高级语言(HL)的机器

采用各种对高级语言和编译程序提供支持的措施，使机器语言和高级语言的语义差距比传统的冯·诺依曼型机器缩小许多。这种机器统称为面向高级语言(HL)的机器。

间接执行型高级语言机器

使高级语言成为机器的汇编语言。即高级语言和机器语言是一一对应的，这种机器称为间接执行型高级语言机器。

直接执行型高级语言机器

高级语言机器本身没有机器语言，或者说高级语言就作为机器语言。它可以直接由硬件或固件对高级语言源程序的语句逐条进行解释并执行。这种机器称为直接执行型高级语言机器。

跳转

当控制指令为无条件改变控制流时，称之为跳转。

分支

当控制指令是有条件改变控制流时，称之为分支。

位置无关

代码在执行时与它被载入的位置无关。

操作数类型

面向应用、面向软件系统所处理的各种数据结构。

操作数表示

硬件结构能够识别、指令系统可以直接调用的那些数据结构。

操作数的类型

主要有：整数（定点）、浮点、十进制、字符、字符串、向量、堆栈等。

变长编码格式

指令的长度是可变的。

定长编码格式

将操作类型和寻址方式组合编码在操作码中，所有指令的长度是固定唯一的。

混合型编码格式

通过提供一定类型的指令字长，期望能够兼顾降低目标代码长度和降低译码复杂度两个目标。

第 3 章 流水线技术

一次重叠执行方式

把执行第 k 条指令与取第 $k+1$ 条指令同时进行。

二次重叠执行方式

为了进一步提高执行速度，可以增加指令重叠执行的程度。把取第 $k+1$ 条指令提前到与分析第 k 条指令同时进行，把分析第 $k+1$ 条指令与执行第 k 条指令同时进行。

哈佛结构

程序空间和数据空间相互独立，因而具有独立的指令总线和数据总线的系统结构。

先行控制技术

缓冲技术和预处理技术的结合。

缓冲技术

在工作速度不固定的两个功能部件之间设置缓冲器，用以平滑它们的工作。

预处理技术
指预取指令、对指令进行加工以及预取操作数等。

流水线技术

将一个重复的时序过程分解成为若干个子过程，而每一个子过程都可有效地在其专用功能段上与其他子过程同时执行。

时(间)空(间)图
用来描述流水线的工作，横坐标表示时间，纵坐标代表流水线的各段。

流水线的深度

流水线的段数。

通过时间

流水线中第一个任务流出结果所需的时间。

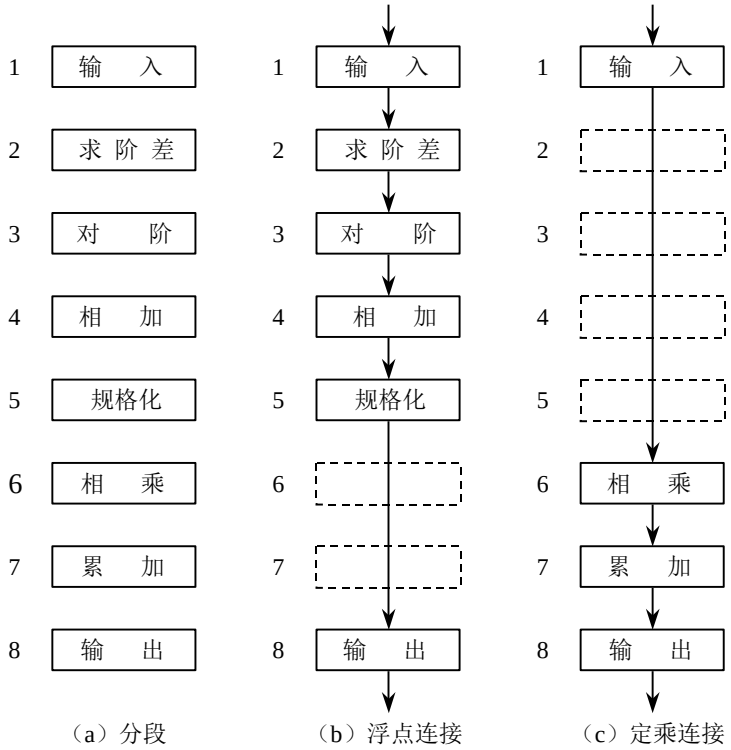
单功能流水线

只能完成一种固定功能的流水线。

功能流水线

流水线的各段可以进行不同的连接，从而使流水线在不同的时间，或者在同一时间完成不同的功能。

TI ASC 的多功能流水线



静态流水线
在同一时间内，流水线的各段只能按同一种功能的连接方式工作。

动态流水线

在同一时间内，当某些段正在实现某种运算时，另一些段却在实现另一种运算。

部件级流水线（运算操作流水线）

把处理机的算术逻辑部件分段，以便为各种数据类型进行流水操作。

处理机级流水线（指令流水线）

把解释指令的过程按照流水方式处理。

处理机间流水线（宏流水线）

由两个以上的处理机串行地对同一数据流进行处理，每个处理机完成一项任务。

标量流水处理机

不具有向量数据表示，仅对标量数据进行流水处理的处理机。

向量流水处理机

具有向量数据表示，并通过向量指令对向量的各元素进行处理的流水处理机。

线性流水线

流水线的各段串行连接，没有反馈回路。

非线性流水线

流水线中除有串行连接的通路外，还有反馈回路。

非线性流水线的调度

在非线性流水线中，确定什么时候向流水线引进新的任务，才能使该任务不会与先前进入流水线的任务发生冲突——争用流水段。

顺序流水线

流水线输出端任务流出的顺序与输入端任务流入的顺序完全相同。每一个任务在流水线的各段中是一个跟着一个顺序流动的。

乱序流水线

流水线输出端任务流出的顺序与输入端任务流入的顺序可以不同，允许后进入流水线的任务先完成（从输出端流出）。又称为无序流水线、错序流水线、异步流水线。

吞吐率

在单位时间内流水线所完成的任务数或输出结果的数量。

最大吞吐率

流水线在连续流动达到稳定状态后所得到的吞吐率。

流水线的瓶颈

流水线中最慢的一段。

消除瓶颈段的两种方法

细分瓶颈段、重复设置瓶颈段

加速比

流水线的速度与等功能的非流水线的速度之比。

效率

流水线的设备利用率。

排空时间

流水线中最后一个任务通过流水线所需的时间。

流水寄存器建立时间

在触发写操作的时钟信号到达之前，寄存器输入必须保持稳定的时间。

流水寄存器传输延迟

时钟信号到达后到寄存器输出可用的时间。

时钟偏移开销

流水线中，时钟到达各流水寄存器的最大差值时间（时钟到达各流水寄存器的时间不是完全相同）。

相关

指两条指令之间存在某种依赖关系。确定程序中指令之间存在什么样的相关，对于充分发挥流水线的效率有重要的意义。

数据相关

对于两条指令 i（在前）和 j（在后），如果下述条件之一成立，则称指令 j 与指令 i 数据相关：
（1）指令 j 使用指令 i 产生的结果；
（2）指令 j 与指令 k 数据相关，而指令 k 又与指令 i 数据相关。

名

指指令所访问的寄存器或存储器单元的名称。

名相关

如果两条指令使用相同的名，但是它们之间并没有数据流动，则称这两条指令存在名相关。

反相关

如果指令 j（在后）写的名与指令 i（在前）读的名相同，则称指令 i 和 j 发生了反相关。反相关指令之间的执行顺序是必须严格遵守的，以保证 i 读的值是正确的。

输出相关

如果指令 j（在后）和指令 i（在前）写相同的名，则称指令 i 和 j 发生了输出相关。输出相关指令的执行顺序是不能颠倒的，以保证最后的结果是指令 j 写进去的。

换名技术

通过改变指令中操作数的名来消除名相关。

寄存器换名

对于寄存器操作数进行换名称为寄存器换名。这个过程既可以用编译器静态实现，也可以用硬件动态完成。

控制相关

由分支指令引起的相关。它需要根据分支指令的执行结果来确定后续指令是否执行。

流水线冲突

指对于具体的流水线来说，由于相关的存在，使得指令流中的下一条指令不能在指定的时钟周期执行。

结构冲突

因硬件资源满足不了指令重叠执行的要求而发生的冲突。

数据冲突

当相关的指令靠得足够近时，它们在流水线中的重叠执行或者重新排序会改变指令读/写操作数的顺序，使之不同于它们非流水实现时的顺序，则发生了数据冲突。

控制冲突

流水线遇到分支指令和其他会改变 PC 值的指令所引起的冲突。

流水线气泡

流水线中插入的暂停周期。

定向技术

当流水线中出现数据冲突时，可以将计算结果从其产生的地方直接送到其他指令中需要它的地方，或所有需要它的功能单元，避免暂停。

写后读冲突（RAW）

考虑流水线中的两条指令 i 和 j，且 i 在 j 之前进入流水线中，j 的执行要用到 i 的计算结果，当它们在流水线中重叠执行时，j 可能在 i 写入其计算结果之前就先行对保存该结果的寄存器进行读操作，从而得到错误的值。

写后写冲突（WAW）

考虑流水线中的两条指令 i 和 j，且 i 在 j 之前进入流水线中，j 和 i 的目的寄存器相同，当它们在流水线中重叠执行时，j 可能在 i 写入其计算结果之前就先行对该结果寄存器进行写操作，从而导致写入顺序错误，在目的寄存器中留下的是 i 写入的值，而不是 j 写入的值。

读后写冲突（WAR）

考虑流水线中的两条指令 i 和 j，且 i 在 j 之前进入流水线中，j 可能在 i 读取某个源寄存器的内容之前就先对该寄存器进行写操作，导致 i 后来读取到的值是错误的。

流水线调度或指令调度

当流水线中出现冲突时，编译器通过重新排列代码的顺序来消除流水线中的暂停，这种技术称为流水线调度。

冻结或排空流水线

在流水线中，处理分支最简单的方法，保持或清除流水线在分支指令之后读入的任何指令，直到知道分支指令的目标地址以及分支转移是否成功为止。

分支延迟

由分支指令引起的延迟。

预测分支失败的方法

当流水线译码到一条分支指令时，流水线继续取指令，并允许该分支指令后的指令继续在流水线中流动。当流水线确定分支转移成功与否以及分支的目标地址之后，如果分支转移成功，流水线必须将在分支指令之后取出的所有指令转化为空操作，并在分支的目标地址处重新取出有效的指令；如果分支转移失败，那么可以将分支指令看作是一条普通指令，流水线正常流动，无需将在分支指令之后取出的所有指令转化为空操作。

预测分支成功的方法

一旦流水线译码到一条指令是分支指令，且完成了分支目标地址的计算，我们就假设分支转移成功，并开始在分支目标地址处取指令执行。

“延迟分支”方法

其主要思想是从逻辑上“延长”分支指令的执行时间。设延迟长度为 n 的分支指令后面有 n 个分支延迟槽，选择 n 条有效和有用的指令放入分支延迟槽中，无论分支成功与否，流水线都会执行这些指令。处于分支延迟槽中的指令“掩盖”了流水线原来必须插入的暂停周期。

水平(横向)处理方式

在横向处理方式中，向量计算是按行的方式从左到右横向地进行。若向量长度为 N，则水平处理方式相当于执行 N 次循环。若使用流水线，在每次循环中可能出现数据相关和功能转换，不适合对向量进行流水处理。

垂直(纵向)处理方式

将整个向量按相同的运算处理完毕之后，再去执行其他运算。

存储器－存储器型操作的运算流水线

向量运算指令的源/目向量都放在存储器内，使得流水线运算部件的输入、输出端直接与存储器相联，构成 MM 型的运算流水线。

分组(纵横)处理方式

把长度为 N 的向量分为若干组，每组长度为 n，组内按纵向方式处理，依次处理各组，组数为「N/n」，适合流水处理。

寄存器－寄存器型操作的运算流水线

可设长度为 n 的向量寄存器，使每组向量运算的源/目向量都在向量寄存器中，流水线的运算部件输入、输出端与向量寄存器相联，构成 RR 型运算流水线。

V_i冲突

并行工作的各向量指令的源向量或结果向量的 V_i有相同的。

功能部件冲突

向量功能部件冲突指的是同一个向量功能部件被一条以上的并行工作向量指令所使用。

链接技术

两条向量指令出现“写后读”相关时，若它们不存在功能部件冲突和向量寄存器（源或目的）冲突，就有可能把它们所用的功能部件头尾相接，形成一条链接流水线，进行流水处理。

链接流水线的流水时间

在链接流水线中，从第一个操作数开始流动到第一个结果产生并存入向量寄存器所需的时间。

向量循环或分段开采技术

当向量的长度大于向量寄存器的长度时，就必须把长向量分成固定长度的段，然后循环分段处理，一次循环只处理一个向量段。

向量处理机的峰值性能 R_∞：

当向量长度为无穷大时，向量处理机的最高性能，也称为峰值性能。

半性能向量长度 $n_{1/2}$

向量处理机的运行性能达到其峰值性能 R_∞ 的一半时所必须满足的向量长度。

向量长度临界值

对于某一计算任务而言，向量方式的处理速度优于标量串行方式处理速度时所需的最小向量长度。

第 5 章 存储层次

存储器的三个主要指标

从用户的角度来看，存储器的三个主要指标是：容量、速度和价格。

多级存储层次

由若干个采用不同实现技术的存储器构成的存储器系统。各存储器处在离 CPU 不同距离的层次上。其目标是速度接近于离 CPU 最近的存储器的速度，容量达到离 CPU 最远的存储器的容量。

命中率 *H*

CPU 在第一级存储器中找到所需数据的概率。

不命中率或失效率 *F*

CPU 在第一级存储器中找不到所需数据的概率。

失效开销

CPU 向第二级存储器发出访问请求到把这个数据块调入第一级存储器所需的时间。

平均访问时间 T_A

T_A = 命中时间 + 失效率 × 失效开销

“Cache－主存”层次

在 CPU 和主存之间增加一级速度快、但容量较小而每位价格较贵的高速缓冲存储器。借助于辅助软硬件，它与主存构成一个有机的整体，以弥补主存速度的不足。

“主存－辅存”层次

“主存—辅存”层次的目的是为了弥补主存容量的不足。它是在主存外面增加一个容量更大、每位价格更便宜、但速度更慢的存储器。它们依靠辅助软硬件的作用，构成一个整体。

全相联映像

当把一个块从主存调入 Cache 时，它可以被放置到 Cache 中的任意一个位置。

直接映像

当把一个块从主存调入 Cache 时，它只能被放置到 Cache 中唯一的一个位置。

组相联映像

当把一个块从主存调入 Cache 时，它可以被放置到 Cache 中唯一的一个组中的任何一个位置（Cache 被等分为若干组，每组由若干个块构成）。

n 路组相联
在组相联映像中，如果每组中有 n 个块，则称该映像规则为 n 路组相联。

相联度

组相联映像中每组中的块数。

目录表

目录表所包含的项数与 Cache 的块数相同，每一项对应于 Cache 中的一个块，用于指出当前该块中存放的信息是哪个主存块的。

候选位置

一个主存块可能映像到 Cache 中的一个或多个 Cache 块位置，这些 Cache 块位置称为候选位置。

随机法

随机地选择被替换的块。

先进先出法(FIFO)
选择最早调入的块作为被替换的块。

最近最少使用法(LRU)
选择近期最少被访问的块作为被替换的块。

写直达法

在执行“写”操作时，不仅把信息写入 Cache 中相应的块，而且也写入下一级存储器中相应的块。

写回法

在执行“写”操作时，只把信息写入 Cache 中相应的块。该块只有在被替换时，才被写回主存。

按写分配法

写失效时，先把所写单元所在的块调入 Cache，然后再进行写入。

不按写分配法

写失效时，直接写入下一级存储器中，而不把相应的块调入 Cache。

分离 Cache

将单一的 Cache 分为两个 Cache：一个专门存放指令，另一个专门存放数据。

混合 Cache

将指令和数据放在一个统一的 Cache 中。

强制性失效

当第一次访问一个块时，该块不在 Cache 中，需从下一级存储器中调入 Cache，这就是强制性失效。

容量失效

如果程序执行时所需的块不能全部调入 Cache 中，则当某些块被替换后，若又重新被访问，就会发生失效。这种失效称为容量失效。

冲突失效

在组相联或直接映像 Cache 中，若太多的块映像到同一组（块）中，则会出现该组中某个块被别的块替换（即使别的组或块有空闲位置），然后又被重新访问的情况。这就发生了冲突失效。

2:1 的 Cache 经验规则

大小为 N 的直接映像 Cache 的失效率约等于大小为 $N/2$ 的两路组相联 Cache 的失效率。

Victim Cache

在 Cache 与下一级存储器的数据通路之间增设一个全相联的小 Cache，用来存放由于失效而被丢弃（替换）的那些块。

伪相联

一种既能获得多路组相联 Cache 的低失效率，又能获得直接映像 Cache 的命中速度的相联办法。采用这种方法时，在命中情况下，访问 Cache 的过程和直接映像 Cache 中的情况相同；而发生失效时，在访问下一级存储器之前，会先检查 Cache 另一个位置（块），看是否匹配。确定这个“另一块”的一种简单的方法是将索引字段的最高位取反，然后按照新索引去寻找“伪相联组”中的对应块。如果这一块的标识匹配，则称发生了“伪命中”。否则，就只好访问下一级存储器。

寄存器预取

预取时，把数据取到寄存器中。

Cache 预取
预取时，只将数据取到 Cache 中，不放入寄存器。

故障性预取

在预取时，若出现虚地址故障或违反保护权限，则会发生异常。

非故障性预取或非绑定预取

预取时，若出现虚地址故障或违反保护权限，则不会发生异常。

非阻塞 Cache 或非锁定 Cache
Cache 在等待预取数据返回的同时，还能继续提供指令和数据。

子块放置技术

把一个 Cache 块划分为若干个小块，称为子块。为每一个子块赋一位有效位，用于说明该子块中的数据是否有效。失效时只从下一级存储器调入一个子块。

请求字

当从存储器向 CPU 调入一块时，块中只有一个字是 CPU 立即需要的，这个字称为请求字。

尽早重启动

在请求字没有到达时，CPU 处于等待状态。一旦请求字到达，就立即发送给 CPU，让等待的 CPU 尽早重启动，继续执行。

请求字优先

调块时，首先向存储器请求 CPU 所要的请求字。请求字一旦到达，就立刻送往 CPU，让 CPU 继续执行，同时从存储器调入该块的其余部分。

失效下的命中

Cache 在失效时仍允许 CPU 进行其它的命中访问。

局部失效率

对于某一级 Cache 来说：
局部失效率 = 该级 Cache 的失效次数/到达该级 Cache 的访存次数

全局失效率

对于某一级 Cache 来说：
全局失效率 = 该级 Cache 的失效次数/CPU 发出的访存总次数

虚拟 Cache
访问 Cache 的索引和标识都是虚拟地址的一部分。

物理 Cache
访问 Cache 的索引和标识都是物理地址的一部分。

进程标识符字段（PID）
虚拟 Cache 中，为了减少清空 Cache 的次数，在地址标识中增加一个进程标识符字段，指出 Cache 中各块的数据是属于哪个程序的。

同义或别名

虚拟 Cache 中，操作系统和用户程序对于同一个物理地址可能采用两种以上不同形式的虚拟地址来访问，这些地址称为同义或别名。

虚拟索引 + 物理标识方法
直接用虚地址中的页内位移作为访问 Cache 的索引，但标识却是物理地址。CPU 发出访存请求后，在进行虚实地址转换的同时，可并行进行标识的读取。在完成地址变换之后，再把得到的物理地址与标识进行比较。它既能得到虚拟 Cache 的好处，又能得到物理 Cache 的优点。

多字宽存储器结构

这是提高存储器带宽的最简单的方法。把主存的宽度增加为原来的若干倍。

多体交叉存储器

把存储芯片组织为多个体，让它们并行工作，从而能一次读或写多个字。存储器的各个体是按字交叉的。

独立存储体

将存储器分为若干个独立的存储体，每个体有独立的地址线、独立的数据总线，有多个存储控制器，以允许多个体独立操作。

体冲突

多个请求要访问同一个体。

DRAM 专用交叉结构
Nibble 方式、Page 方式、Static column 方式。

TLB

一个专用的高速缓冲器，用于存放近期经常使用的页表项，其内容是页表部分内容的一个副本。

第 6 章 输入/输出系统

输入/输出系统

简称 I/O 系统，它包括 I/O 设备以及 I/O 设备与处理机的连接。

存储外设可靠性能的参数

可靠性、可用性和可信性

系统可靠性

指系统从初始状态开始一直提供服务的能力。通常用平均无故障时间 MTTF（Mean Time To Failure）来衡量。

系统的失效率

平均无故障时间 MTTF 的倒数。

系统可用性

系统正常工作时间在连续两次正常服务间隔时间中所占的比率。

系统的可信性

指服务的质量，即在多大程度上可以合理地认为服务是可靠的。

有效构建方法

在构建系统的过程中消除故障隐患，这样建立起来的系统就不会出现故障。

纠错方法

在系统构建中设计容错部件，即使出现故障，也可以通过容错信息保证系统正常工作。

RAID

廉价磁盘冗余阵列或独立磁盘冗余阵列。

RAID0

采用数据分块技术，把数据分布在多个盘上，无冗余信息。

RAID1

镜像盘。每当数据写入一个磁盘时，也将该数据写到另一个冗余盘（镜像盘），形成数据的两个备份（数据镜像）。如果一个磁盘失效，系统可以到镜像盘中获得所需要的数据。

RAID2
位交叉式汉明编码阵列。数据字以位交叉方式分别记录在各个磁盘上，编码位被存放在多个校验（Ecc）磁盘的对应位上。

RAID3
位交叉奇偶校验盘阵列。数据以位或字节交叉的方式存于各盘，冗余的奇偶校验信息存储在专用的冗余盘上。特点是可以获得非常高的数据传输率。缺点是一次只能执行一个 I/O 请求。

RAID4
专用奇偶校验独立存取盘阵列。数据以块（块大小可变）交叉的方式存于各盘，冗余的奇偶校验信息存放在一个专用盘上。

RAID5
块交叉分布式奇偶校验盘阵列，是旋转奇偶校验独立存取的阵列。即数据以块交叉的方式存于各盘，但无专用的冗余盘，而是把冗余的奇偶校验信息均匀地分布在所有磁盘上。

RAID6
双维奇偶校验独立存取盘阵列。即数据以块（块大小可变）交叉的方式存于各盘，冗余的检、纠错信息均匀地分布在所有磁盘上。可容忍双盘出错。

分离事务总线
将总线事务分成请求和应答两部分。在请求和应答之间的空闲时间内，总线可以供其他的 I/O 使用，这样就不必在整个 I/O 过程中都独占总线。又称为流水总线、悬挂总线或者包交换总线。

I/O 总线标准
定义如何将设备与计算机进行连接的文档。

通道处理机
能够执行有限 I/O 指令，并且能够被多台外围设备共享的小型 DMA 专用处理机。

字节多路通道
一种简单的共享通道，主要为多台低速或中速的外围设备服务。当多台设备同时连接到一个字节多路通道上时，通道每连接一个外围设备，只传送一个字节，然后又与另一台设备连接，并传送一个字节。依次循环工作。

数组多路通道
适于为高速设备服务。通道每连接一台高速设备，传送一个数据块，传送完成后，又与另一台高速设备连接，再传送一个数据块。依次循环工作。

选择通道

适于为多台高速外围设备服务。在传送数据期间，该通道只能为一台高速外围设备服务，但在不同的时间内可以选择不同的设备。

通道流量

一个通道在数据传送期间，单位时间内能够传送的最大数据量。

通道最大流量

一个通道在满负荷工作状态下的流量。

虚拟 DMA

允许 DMA 设备直接使用虚拟地址，在 DMA 期间由硬件将虚拟地址映射到物理地址。

异步 I/O

允许进程在发出 I/O 请求后继续执行，直到该进程需要使用请求的数据。异步 I/O 允许多个 I/O 请求同时处理以最大限度地利用带宽。

第 7 章 互连网络

互连网络

一种由开关元件按照一定的拓扑结构和控制方式构成的网络，用来实现计算机系统中结点之间的相互连接。这些结点可以是处理器、存储模块或其他设备。

线路交换

源结点和目的结点之间的物理通路在整个数据传送期间一直保持连接。

分组交换

把信息分割成许多组（又称为包），将它们分别送入互连网络。这些数据包可以通过不同的路径传送，到目的结点后再拼合出原来的数据。在分组交换中，结点之间不存在固定连接的物理通路。

集中控制方式

集中控制方式中，有一个全局的控制器接收所有的通信请求，并由它设置互连网络的开关连接。

分散控制方式

分散控制方式中，不存在全局的控制器，通信请求的处理和开关的设置由互连网络分散地进行。

静态拓扑结构

在各结点之间有专用的连接通路，且在运行过程中不能改变。

动态拓扑结构

根据需要设置互连网络中的开关，从而对结点之间的连接通路进行重新组合，实现所要求的通信模式。

互连函数

用变量 x 表示输入（设 x=0，1，…，$N-1$），用函数 $f(x)$表示输出，通过数学表达式建立输入端与输出端的一一对应关系。即在互连函数 f 的作用下，输入端 x 连接到输出端 $f(x)$。也称为置换函数或排列函数。

循环

互连函数 $f(x)$有时可以采用循环表示，即：$(x_0\ x_1\ x_2\ \cdots\ x_{j-1})$。它表示

$$f(x_0)=x_1,\ f(x_1)=x_2,\ \cdots,\ f(x_{j-1})=x_0$$

j 称为该循环的长度。

交换函数

实现二进制地址编码中第 k 位互反的输入端与输出端之间的连接。其表达式为

$$E(x_{n-1}x_{n-2}\cdots x_{k+1}x_k x_{k-1}\cdots x_1x_0)=x_{n-1}x_{n-2}\cdots x_{k+1}\bar{x}_k x_{k-1}\cdots x_1x_0$$

均匀洗牌函数

将输入端分成数目相等的两半，前一半和后一半按类似均匀混洗扑克牌的方式交叉地连接到输出端（输出端相当于混洗的结果）。其函数关系可表示为

$$S(x_{n-1}x_{n-2}\cdots x_1x_0)=x_{n-2}x_{n-3}\cdots x_1x_0x_{n-1}$$

即把输入端的二进制编号循环左移一位。

逆均匀洗牌函数

将输入端的二进制编号循环右移一位而得到所连接的输出端编号。其互连函数为

$$S^{-1}(x_{n-1}x_{n-2}\cdots x_1x_0)=x_0x_{n-1}x_{n-2}\cdots x_1$$

逆均匀洗牌是均匀洗牌的逆函数。

蝶式互连函数

把输入端的二进制编号的最高位与最低位互换位置，便得到了输出端的编号。
定义为

$$B(x_{n-1}x_{n-2}\cdots x_1x_0)=x_0x_{n-2}\cdots x_1x_{n-1}$$

反位序函数

将输入端二进制编号的位序颠倒过来求得相应输出端的编号。其互连函数为

$$R(x_{n-1}x_{n-2}\cdots x_1x_0)=x_0x_1\cdots x_{n-2}x_{n-1}$$

PM2I 函数

一种移数函数，它是将各输入端都循环移动一定的位置连到输出端。其函数为

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

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

其中，$0\leqslant x\leqslant N-1$，$0\leqslant i\leqslant n-1$，$n=\log_2 N$，$N$ 为结点数。

网络规模

一般说来，网络用图来表示。这种图由用有向边或无向边连接的有限个结点构成。其结点数称为网络规模。

结点度

与结点相连接的边的数目。

入度

在单向通道的情况下，进入结点的通道数。

出度

在单向通道的情况下，从结点出来的通道数。

距离

对于网络中的任意两个结点，从一个结点出发到另一个结点终止所需要跨越的边数的最小值。

网络直径

网络中任意两个结点间最短路径长度的最大值。

等分宽度

在将某一网络切成相等两半的各种切法中，沿切口的最小通道边数。

结点之间的线长

两个结点之间连线的长度，用米、千米等表示。

对称网络

对于一个网络，如果从其中的任何一个结点看，拓扑结构都是一样的，则称此网络为对称网络。

线性阵列

一种一维的线性网络，其中 N 个结点用 N-1 个链路连成一行。内部结点度为 2，端结点度为 1，直径为 N-1，等分宽度 b=1。

环

用一条附加链路将线性阵列的两个端点连接起来而构成的。可以单向工作，也可以双向工作。它是对称的，结点度是常数 2。双向环的直径为 $N/2$，单向环的直径是 N。

带弦环

在环的基础上，给每个结点增加一条或两条链路。增加的链路愈多，结点度愈高，网络直径

就愈小。

全连接网络

一种环网。其中任何两个结点之间都有链路相连。

循环移数网络

通过在环上每个结点到所有与其距离为 2 的整数幂的结点之间都增加一条附加链而构成的。这就是说，如果 $|j-i|=2^r$，$r=0,1,2,\cdots,n-1$，网络规模 $N=2^n$，则结点 i 与结点 j 连接。这种循环移数网络的结点度为 $d=2n-1$，直径 $D=n/2$。

超立方体

一种二元 n 立方体结构。一般说来，一个 n 立方体由 $N=2^n$ 个结点组成，它们分布在 n 维上，每维有两个结点。

交叉开关网络

每个输入端通过一个交叉点开关无阻塞地与一个空闲输出端相连。它是单级无阻塞置换网络，带宽和互连特性最好。

第 8 章 多处理机

集中式共享存储器结构

由几个到几十个处理器构成的 MIMD 机器。各处理器通过大容量的 Cache 和总线互连，共享一个单独的物理存储器。又称为对称式共享存储器结构机器或者 UMA 机器。

分布式存储器结构

处理器的规模较大，存储器分布到各个处理器上，而非采用集中式。系统中每个结点包含了处理器、存储器、I / O 以及互连网络接口。

通信延迟

通信延迟 = 发送开销 + 跨越时间 + 传输延迟 + 接收开销

跨越时间

数字信号从发送方的线路端传送到接收方的线路端所经过的时间。

传输时间

全部的消息量除以线路带宽。

分布式共享存储器或可缩放共享存储器体系结构

物理上分离的多个存储器作为一个逻辑上共享的存储空间进行编址，如果一个处理器具有访问权，就可以访问任何一个其他的局部存储器。

共享存储器机器

共享地址空间的机器。利用 load 和 store 指令中的地址隐含地进行数据通信。

消息传递机器

多个地址空间的机器。数据通信要通过处理器间显式地传递消息来完成。

私有数据

供一个单独的处理器使用的数据。

共享数据

供多个处理器使用的数据。

共享数据的迁移

把远程的共享数据项副本放在本处理器局部的 Cache 中使用，从而降低了对远程共享数据的访问延迟。

共享数据的复制

把多个处理器需要同时读取的共享数据项的副本放在各自局部 Cache 中使用。复制不仅降低了访存的延迟，也减少了访问共享数据所产生的冲突。

Cache 一致性协议

对多个处理器维护 Cache 一致性的协议。

第 9 章 机群

PVP
并行向量处理机

SMP
对称多处理机

MPP
大规模并行处理机

机群

机群是一种价格低廉、易于构建、可扩放性极强的并行计算机系统。它由多台同构或异构的独立计算机通过高性能网络或局域网互连在一起，协同完成特定的并行计算任务。从用户的角度来看，机群就是一个单一、集中的计算资源。

单一系统映像

简称 SSI。包含四重含义。首先是“单一系统”，尽管系统中有多个处理器，用户仍然把整个机群视为一个单一的计算系统来使用。其次是“单一控制”，逻辑上，最终用户或系统用户使用的服务都来自机群中唯一一个位置。三是“对称性”，用户可以从任一个结点上获得机群服务，也就是说，对于所有结点和所有用户，除了那些具有特定访问权限的服务与功能外，所有机群服务与功能都是对称的。最后则是“位置透明”，用户不必了解真正提供服务的物理设备的具体位置。

MPI
Message Passing Interface。目前最重要的一个基于消息传递的并行编程工具，它具有可移植性好、功能强大、效率高等许多优点，而且有许多不同的免费、高效、实用的实现版本。

PVM
Parallel Virtual Machine。一种常用的基于消息传递的并行编程环境，它把工作站网络构建成一个虚拟的并行机系统，为并行应用提供了运行平台。

HPF
High Performance FORTRAN。一个支持数据并行的并行语言标准。

OpenMP
Open Multi-Processing。一个共享存储并行系统上的应用编程接口，它规范了一系列的编译制导、运行库例程和环境变量，并为 C/C++和 FORTRAN 等高级语言提供了应用编程接口。

高可用性机群

目的是在当系统中某些结点出现故障的情况下，仍能继续对外提供服务。它采用冗余机制，当系统中某个结点由于软、硬件故障而失效时，该结点上的任务将在最短的时间内被迁移到机群内另一个具有相同功能与结构的结点上继续执行。

负载均衡机群

主要目的是提供与结点个数成正比的负载能力。机群能够根据系统中各个结点的负载情况实时地进行任务分配。为此，它专门设置了一个重要的监控结点，负责监控其余每个工作结点的负载和状态，并根据监控结果将任务分派到不同的结点上。

高性能计算机群

主要目的是降低高性能计算的成本。它通过高速的商用互连网络，将数十台乃至上千台 PC 或工作站连接在一起，可以提供接近甚至超过传统并行计算机系统的计算能力，但其价格却仅是具有相同计算能力的传统并行计算机系统的几十分之一。

专用机群

是为代替传统的大中型机或巨型机而设计的，装置比较紧凑，一般都装在比较小的机架内，放在机房中使用，因此它的吞吐率较高，响应时间也较短。专用机群的结点往往是同构的，一般采用集中控制，由一个（或一组）管理员统一管理，而且用户一般需要通过一台终端机来访问它。

企业机群

它是为了充分利用各个结点的空闲资源而设计的，各个结点分散安放，各结点之间一般通过标准的 LAN 或 WAN 互连，通信开销较大，延迟较长。结点一般是异构的，并由不同的个人拥有或使用，机群管理者只能对各个结点进行有限的管理。结点拥有者可以随意地进行关机、重新配置或者升级，而且对一个结点而言，它的拥有者或使用者的任务应该具有最高优先级，高于企业的其他用户。