软件设计师教程

第5版

褚华 霍秋艳 主编

清华大学出版社
北京

内 容 简 介

本书作为中级职称的全国计算机技术与软件专业技术资格（水平）考试（简称“软考”）指定教材，具有比较权威的指导意义。本书根据《软件设计师考试大纲》（2018年审定通过）的重点内容，组织了12章的内容，考生在学习教材内容的同时，还须对照考试大纲，认真学习和复习大纲的知识点。

本书是在《软件设计师考试大纲》的指导下，对《软件设计师教程（第4版）》进行了认真修编，部分章节是重写后形成的。在本书中，强化了软件工程部分的知识，增加了Web应用系统分析与设计知识。

本书适合参加本级别考试的考生和大学在校生作为教材。

图书在版编目（CIP）数据

软件设计师教程/褚华，霍秋艳主编. —5版. —北京：清华大学出版社，2018（2018.3重印）
（全国计算机技术与软件专业技术资格（水平）考试指定用书）
ISBN 978-7-302-49122-4

Ⅰ. ①软… Ⅱ. ①褚… ②霍… Ⅲ. ①软件设计-资格考试-自学参考资料 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字（2017）第313210号

责任编辑：杨如林　柴文强
封面设计：常雪影
责任校对：徐俊伟
责任印制：沈　露

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×230mm　　**印　张：**42.75　　**防伪页：**1　　**字　数：**930千字
版　　次：2004年7月第1版　2018年2月第5版　　**印　次：**2018年3月第2次印刷
印　　数：5001～10000
定　　价：119.00元

产品编号：075520-01

前　言

全国计算机技术与软件专业技术资格（水平）考试实施至今已经历了二十余年，在社会上产生了很大的影响，对我国软件产业的形成和发展做出了重要的贡献。为了适应我国计算机信息技术发展的需求，人力资源和社会保障部、工业和信息化部决定将考试的级别拓展到计算机信息技术行业的各个方面，以满足社会上对各种计算机信息技术人才的需要。

编者受全国计算机专业技术资格考试办公室委托，对《软件设计师教程（第 4 版）》进行改写，以适应新的考试大纲要求。在考试大纲中，要求考生掌握的知识面很广，每个章节的内容都能构成相关领域的一门甚至多门课程，因此编写的难度很高。考虑到参加考试的人员已有一定的基础，所以本书中只对考试大纲中所涉及的知识领域的要点加以阐述，但限于篇幅所限，不能详细地展开，请读者谅解。

全书共分 12 章，各章节内容安排如下：

第 1 章主要介绍计算机系统基础知识、计算机体系结构以及安全性、可靠性和系统性能评测基础。

第 2 章主要介绍程序设计语言的基本概念与基本成分，阐述了汇编程序、编译程序与解释程序的基本原理。

第 3 章主要介绍数据结构的基础知识，包括线性结构、数组、广义表、树和图，以及查找和排序的基本算法。

第 4 章主要介绍操作系统基本概念与分类及特点、进程管理、存储管理、设备管理、文件管理、作业管理等。

第 5 章主要介绍软件工程中软件过程与过程模型、需求分析与需求工程、系统设计、系统测试、系统运行与维护、软件项目管理、软件质量、软件度量、软件工具与软件开发环境基础知识。

第 6 章主要介绍系统分析与设计、结构化分析与设计、Web 应用系统分析与设计、用户界面设计基础知识。

第 7 章主要介绍面向对象的基本概念和面向对象开发技术，包括面向对象的分析与设计方法，UML 以及设计模式的概念和应用。

第 8 章主要介绍算法设计与分析的基本概念，包括分治法、动态规划法、贪心法、回溯法、分支界限法、概率算法、近似算法、数据挖掘算法及智能优化算法。

第9章主要介绍数据库的基本概念、数据模型、关系代数、SQL语言、规范化理论和事务处理等控制功能。

第10章主要介绍网络与信息安全基础知识，包括网络体系结构、网络互连设备、网络构件、网络协议、网络应用、信息安全和网络安全方面的基础知识。

第11章主要介绍标准化与知识产权基础知识。

第12章主要介绍结构化分析与设计、数据库分析与设计、面向对象分析与设计、算法分析与设计以及面向过程、面向对象的程序设计与实现。

本书第1章由张淑平、马志欣编写，第2章由张淑平编写，第3章由张淑平、陈静玉、宋胜利编写，第4章由王亚平编写，第5章、第6章、第7章由霍秋艳、褚华编写，第8章由覃桂敏、褚华编写，第9章由王亚平编写，第10章由严体华编写，第11章由刘强编写，第12章由王亚平、褚华、霍秋艳、覃桂敏、张淑平编写，最后由霍秋艳、褚华统稿。

在本书的编写过程中，参考了许多相关的书籍和资料，编者在此对这些参考文献的作者表示感谢。同时感谢清华大学出版社在本书出版过程中所给予的支持和帮助。

因水平有限，书中难免存在欠妥之处，望读者指正，以利改进和提高。

编　者

2018年1月

目 录

第 1 章　计算机网络概论

1.1　计算机系统基础知识

1.1.1　计算机系统硬件基本组成

计算机系统是由硬件和软件组成的，它们协同工作来运行程序。计算机的基本硬件系统由运算器、控制器、存储器、输入设备和输出设备 5 大部件组成。运算器、控制器等部件被集成在一起统称为中央处理单元（Central Processing Unit，CPU）。CPU 是硬件系统的核心，用于数据的加工处理，能完成各种算术、逻辑运算及控制功能。存储器是计算机系统中的记忆设备，分为内部存储器和外部存储器。前者速度高、容量小，一般用于临时存放程序、数据及中间结果。而后者容量大、速度慢，可以长期保存程序和数据。输入设备和输出设备合称为外部设备（简称外设），输入设备用于输入原始数据及各种命令，而输出设备则用于输出计算机运行的结果。

1.1.2　中央处理单元

中央处理单元（CPU）是计算机系统的核心部件，它负责获取程序指令、对指令进行译码并加以执行。

1．CPU 的功能

（1）程序控制。CPU 通过执行指令来控制程序的执行顺序，这是 CPU 的重要功能。

（2）操作控制。一条指令功能的实现需要若干操作信号配合来完成，CPU 产生每条指令的操作信号并将操作信号送往对应的部件，控制相应的部件按指令的功能要求进行操作。

（3）时间控制。CPU 对各种操作进行时间上的控制，即指令执行过程中操作信号的出现时间、持续时间及出现的时间顺序都需要进行严格控制。

（4）数据处理。CPU 通过对数据进行算术运算及逻辑运算等方式进行加工处理，数据加工处理的结果被人们所利用。所以，对数据的加工处理也是 CPU 最根本的任务。

此外，CPU 还需要对系统内部和外部的中断（异常）做出响应，进行相应的处理。

2. CPU 的组成

CPU 主要由运算器、控制器、寄存器组和内部总线等部件组成，如图 1-1 所示。

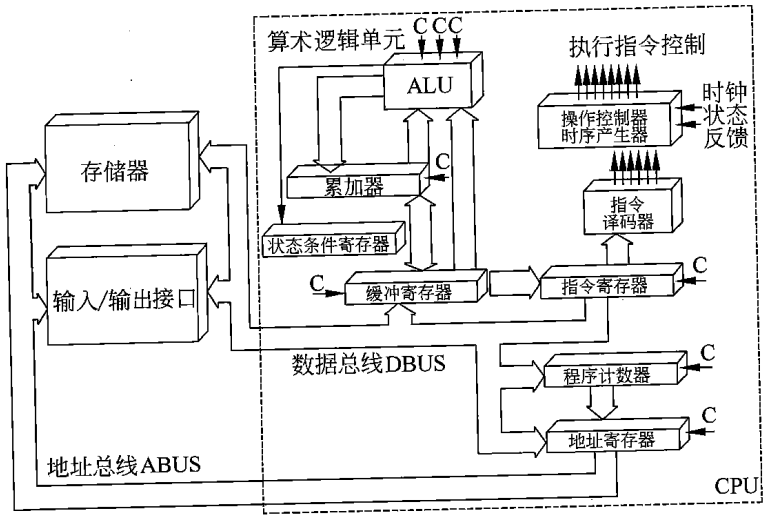


图 1-1 CPU 基本组成结构示意图

1）运算器

运算器由算术逻辑单元（Arithmetic and Logic Unit，ALU）、累加寄存器、数据缓冲寄存器和状态条件寄存器等组成，它是数据加工处理部件，用于完成计算机的各种算术和逻辑运算。相对控制器而言，运算器接受控制器的命令而进行动作，即运算器所进行的全部操作都是由控制器发出的控制信号来指挥的，所以它是执行部件。运算器有如下两个主要功能。

（1）执行所有的算术运算，例如加、减、乘、除等基本运算及附加运算。

（2）执行所有的逻辑运算并进行逻辑测试，例如与、或、非、零值测试或两个值的比较等。

下面简要介绍运算器中各组成部件的功能。

（1）算术逻辑单元（ALU）。ALU 是运算器的重要组成部件，负责处理数据，实现对数据的算术运算和逻辑运算。

（2）累加寄存器（AC）。AC 通常简称为累加器，它是一个通用寄存器，其功能是当运算器的算术逻辑单元执行算术或逻辑运算时，为 ALU 提供一个工作区。例如，在执行一个减法运算前，先将被减数取出暂存在 AC 中，再从内存储器中取出减数，然后同 AC 的内容相减，将所得的结果送回 AC 中。运算的结果是放在累加器中的，运算器中至少要有一个累加寄存器。

（3）数据缓冲寄存器（DR）。在对内存储器进行读/写操作时，用 DR 暂时存放由内存储器读/写的一条指令或一个数据字，将不同时间段内读/写的数据隔离开来。DR 的主要作用为：作为 CPU 和内存、外部设备之间数据传送的中转站；作为 CPU 和内存、外围设备之间在操作速度上的缓冲；在单累加器结构的运算器中，数据缓冲寄存器还可兼作为操作数寄存器。

（4）状态条件寄存器（PSW）。PSW 保存由算术指令和逻辑指令运行或测试的结果建立的各种条件码内容，主要分为状态标志和控制标志，例如运算结果进位标志（C）、运算结果溢出标志（V）、运算结果为 0 标志（Z）、运算结果为负标志（N）、中断标志（I）、方向标志（D）和单步标志等。这些标志通常分别由 1 位触发器保存，保存了当前指令执行完成之后的状态。通常，一个算术操作产生一个运算结果，而一个逻辑操作产生一个判决。

2）控制器

运算器只能完成运算，而控制器用于控制整个 CPU 的工作，它决定了计算机运行过程的自动化。它不仅要保证程序的正确执行，而且要能够处理异常事件。控制器一般包括指令控制逻辑、时序控制逻辑、总线控制逻辑和中断控制逻辑等几个部分。

指令控制逻辑要完成取指令、分析指令和执行指令的操作，其过程分为取指令、指令译码、按指令操作码执行、形成下一条指令地址等步骤。

（1）指令寄存器（IR）。当 CPU 执行一条指令时，先把它从内存储器取到缓冲寄存器中，再送入 IR 暂存，指令译码器根据 IR 的内容产生各种微操作指令，控制其他的组成部件工作，完成所需的功能。

（2）程序计数器（PC）。PC 具有寄存信息和计数两种功能，又称为指令计数器。程序的执行分两种情况，一是顺序执行，二是转移执行。在程序开始执行前，将程序的起始地址送入 PC，该地址在程序加载到内存时确定，因此 PC 的内容即是程序第一条指令的地址。执行指令时，CPU 自动修改 PC 的内容，以便使其保持的总是将要执行的下一条指令的地址。由于大多数指令都是按顺序来执行的，所以修改的过程通常只是简单地对 PC 加 1。当遇到转移指令时，后继指令的地址根据当前指令的地址加上一个向前或向后转移的位移量得到，或者根据转移指令给出的直接转移的地址得到。

（3）地址寄存器（AR）。AR 保存当前 CPU 所访问的内存单元的地址。由于内存和 CPU 存在着操作速度上的差异，所以需要使用 AR 保持地址信息，直到内存的读/写操作完成为止。

（4）指令译码器（ID）。指令包含操作码和地址码两部分，为了能执行任何给定的指令，必须对操作码进行分析，以便识别所完成的操作。指令译码器就是对指令中的操作码字段进行分析解释，识别该指令规定的操作，向操作控制器发出具体的控制信号，控制各部件工作，完成所需的功能。

时序控制逻辑要为每条指令按时间顺序提供应有的控制信号。总线逻辑是为多个功能部件

服务的信息通路的控制电路。中断控制逻辑用于控制各种中断请求，并根据优先级的高低对中断请求进行排队，逐个交给CPU处理。

3）寄存器组

寄存器组可分为专用寄存器和通用寄存器。运算器和控制器中的寄存器是专用寄存器，其作用是固定的。通用寄存器用途广泛并可由程序员规定其用途，其数目因处理器不同有所差异。

3. 多核CPU

核心又称为内核，是CPU最重要的组成部分。CPU中心那块隆起的芯片就是核心，是由单晶硅以一定的生产工艺制造出来的，CPU所有的计算、接收/存储命令、处理数据都由核心执行。各种CPU核心都具有固定的逻辑结构，一级缓存、二级缓存、执行单元、指令级单元和总线接口等逻辑单元都会有合理的布局。

多核即在一个单芯片上面集成两个甚至更多个处理器内核，其中，每个内核都有自己的逻辑单元、控制单元、中断处理器、运算单元，一级Cache、二级Cache共享或独有，其部件的完整性和单核处理器内核相比完全一致。

CPU的主要厂商AMD和Intel的双核技术在物理结构上有所不同。AMD将两个内核做在一个Die（晶元）上，通过直连架构连接起来，集成度更高。Intel则是将放在不同核心上的两个内核封装在一起，因此将Intel的方案称为“双芯”，将AMD的方案称为“双核”。从用户端的角度来看，AMD的方案能够使双核CPU的管脚、功耗等指标跟单核CPU保持一致，从单核升级到双核，不需要更换电源、芯片组、散热系统和主板，只需要刷新BIOS软件即可。

多核CPU系统最大的优点（也是开发的最主要目的）是可满足用户同时进行多任务处理的要求。

单核多线程CPU是交替地转换执行多个任务，只不过交替转换的时间很短，用户一般感觉不出来。如果同时执行的任务太多，就会感觉到“慢”或者“卡”。而多核在理论上则是在任何时间内每个核执行各自的任务，不存在交替问题。因此，单核多线程和多核（一般每核也是多线程的）虽然都可以执行多任务，但多核的速度更快。

虽然采用了Intel超线程技术的单核可以视为是双核，4核可以视为是8核。然而，视为是8核一般比不上实际是8核的CPU性能。

要发挥CPU的多核性能，就需要操作系统能够及时、合理地给各个核分配任务和资源（如缓存、总线、内存等），也需要应用软件在运行时可以把并行的线程同时交付给多个核心分别处理。

1.1.3 数据表示

各种数值在计算机中表示的形式称为机器数，其特点是采用二进制计数制，数的符号用0

和 1 表示，小数点则隐含，表示不占位置。机器数对应的实际数值称为数的真值。

机器数有无符号数和带符号数之分。无符号数表示正数，在机器数中没有符号位。对于无符号数，若约定小数点的位置在机器数的最低位之后，则是纯整数；若约定小数点的位置在机器数的最高位之前，则是纯小数。对于带符号数，机器数的最高位是表示正、负的符号位，其余位则表示数值。

为了便于运算，带符号的机器数可采用原码、反码和补码等不同的编码方法，机器数的这些编码方法称为码制。

1）原码、反码、补码和移码

（1）原码表示法。数值 X 的原码记为 $[X]_{原}$，如果机器字长为 n（即采用 n 个二进制位表示数据），则原码的定义如下：

若 X 是纯整数，则 $[X]_{原}=\begin{cases} X & 0 \leqslant X \leqslant 2^{n-1}-1 \\ 2^{n-1}+|X| & -(2^{n-1}-1) \leqslant X \leqslant 0 \end{cases}$

若 X 是纯小数，则 $[X]_{原}=\begin{cases} X & 0 \leqslant X < 1 \\ 2^{0}+|X| & -1 < X \leqslant 0 \end{cases}$

【例 1.1】 若机器字长 n 等于 8，分别给出+1，–1，+127，–127，+45，–45，+0.5，–0.5 的原码表示。

$[+1]_{原}$=0 0000001　　$[-1]_{原}$=1 0000001

$[+127]_{原}$=0 1111111　　$[-127]_{原}$=1 1111111

$[+45]_{原}$=0 0101101　　$[-45]_{原}$=1 0101101

$[+0.5]_{原}$=0 ◊1000000　　$[-0.5]_{原}$=1 ◊1000000（其中，◊ 是小数点的位置）

在原码表示法中，最高位是符号位，0 表示正号，1 表示负号，其余的 n–1 位表示数值的绝对值。数值 0 的原码表示有两种形式：$[+0]_{原}$=0 0000000，$[-0]_{原}$=1 0000000 。

（2）反码表示法。数值 X 的反码记作 $[X]_{反}$，如果机器字长为 n，则反码的定义如下：

若 X 是纯整数，则 $[X]_{反}=\begin{cases} X & 0 \leqslant X \leqslant 2^{n-1}-1 \\ 2^{n}-1+X & -(2^{n-1}-1) \leqslant X \leqslant 0 \end{cases}$

若 X 是纯小数，则 $[X]_{反}=\begin{cases} X & 0 \leqslant X < 1 \\ 2-2^{-(n-1)}+X & -1 < X \leqslant 0 \end{cases}$

【例 1.2】 若机器字长 n 等于 8，分别给出+1，–1，+127，–127，+45，–45，+0.5，–0.5 的反码表示。

$[+1]_{反}$=0 0000001　　$[-1]_{反}$=1 1111110

$[+127]_{反}$=0 1111111　　$[-127]_{反}$=1 0000000

$[+45]_{反}$=0 0101101　　$[-45]_{反}$=1 1010010

$[+0.5]_{反}$=0 ◊1000000　　$[-0.5]_{反}$=1◊ 0111111（其中，◊是小数点的位置）

在反码表示中，最高位是符号位，0 表示正号，1 表示负号，正数的反码与原码相同，负数的反码则是其绝对值按位求反。数值 0 的反码表示有两种形式：$[+0]_{反}$=0 0000000，$[-0]_{反}$=1 1111111。

（3）补码表示法。数值 X 的补码记作$[X]_{补}$，如果机器字长为 n，则补码的定义如下：

若 X 是纯整数，则 $[X]_{补}=\begin{cases} X & 0 \leqslant X \leqslant 2^{n-1}-1 \\ 2^n+X & -2^{n-1} \leqslant X \leqslant 0 \end{cases}$

若 X 是纯小数，则 $[X]_{补}=\begin{cases} X & 0 \leqslant X < 1 \\ 2+X & -1 \leqslant X < 0 \end{cases}$

【例 1.3】 若机器字长 n 等于 8，分别给出+1，–1，+127，–127，+45，–45，+0.5，–0.5 的补码表示。

$[+1]_{补}$=0 0000001　　$[-1]_{补}$=1 1111111

$[+127]_{补}$=0 1111111　　$[-127]_{补}$=1 0000001

$[+45]_{补}$=0 0101101　　$[-45]_{补}$=1 1010011

$[+0.5]_{补}$=0 ◊1000000　　$[-0.5]_{补}$=1◊ 1000000（其中，◊是小数点的位置）

在补码表示中，最高位为符号位，0 表示正号，1 表示负号，正数的补码与其原码和反码相同，负数的补码则等于其反码的末位加 1。在补码表示中，0 有唯一的编码：$[+0]_{补}$=0 0000000，$[-0]_{补}$=00000000。

（4）移码表示法。移码表示法是在数 X 上增加一个偏移量来定义的，常用于表示浮点数中的阶码。如果机器字长为 n，规定偏移量为 2^{n-1}，则移码的定义如下：

若 X 是纯整数，则 $[X]_{移}=2^{n-1}+X$ $(-2^{n-1} \leqslant X < 2^{n-1})$；若 X 是纯小数，则 $[X]_{补}=1+X$ $(-1 \leqslant X < 1)$。

【例 1.4】 若机器字长 n 等于 8，分别给出+1，–1，+127，–127，+45，–45，+0，–0 的移码表示。

$[+1]_{移}$=1 0000001　　$[-1]_{移}$=0 1111111

$[+127]_{移}$=1 1111111　　$[-127]_{移}$=0 0000001

$[+45]_{移}$=1 0101101　　$[-45]_{移}$=0 1010011

$[+0]_{移}$=1 0000000　　$[-0]_{移}$=10000000

实际上，在偏移 2^{n-1} 的情况下，只要将补码的符号位取反便可获得相应的移码表示。

2）定点数和浮点数

（1）定点数。所谓定点数，就是小数点的位置固定不变的数。小数点的位置通常有两种约定方式：定点整数（纯整数，小数点在最低有效数值位之后）和定点小数（纯小数，小数点在

最高有效数值位之前）。

设机器字长为 n，各种码制下带符号数的范围如表 1-1 所示。

表 1-1　机器字长为 n 时各种码制表示的带符号数的范围

码　制	定 点 整 数	定 点 小 数
原码	$-(2^{n-1}-1)\sim+(2^{n-1}-1)$	$-(1-2^{-(n-1)})\sim+(1-2^{-(n-1)})$
反码	$-(2^{n-1}-1)\sim+(2^{n-1}-1)$	$-(1-2^{-(n-1)})\sim+(1-2^{-(n-1)})$
补码	$-2^{n-1}\sim+(2^{n-1}-1)$	$-1\sim+(1-2^{-(n-1)})$
移码	$-2^{n-1}\sim+(2^{n-1}-1)$	$-1\sim+(1-2^{-(n-1)})$

（2）浮点数。当机器字长为 n 时，定点数的补码和移码可表示 2^n 个数，而其原码和反码只能表示 2^n-1 个数（0 的表示占用了两个编码），因此，定点数所能表示的数值范围比较小，在运算中很容易因结果超出范围而溢出。浮点数是小数点位置不固定的数，它能表示更大范围的数。

在十进制中，一个数可以写成多种表示形式。例如，83.125 可写成 $10^3\times 0.083125$ 或 $10^4\times 0.0083125$ 等。同样，一个二进制数也可以写成多种表示形式。例如，二进制数1011.10101 可以写成 $2^4\times 0.101110101$、$2^5\times 0.0101110101$ 或 $2^6\times 0.00101110101$ 等。由此可知，一个二进制数 N 可以表示为更一般的形式 $N=2^E\times F$，其中 E 称为阶码，F 称为尾数。用阶码和尾数表示的数称为浮点数，这种表示数的方法称为浮点表示法。

在浮点表示法中，阶码为带符号的纯整数，尾数为带符号的纯小数。浮点数的表示格式如下：

阶符	阶码	数符	尾数

很明显，一个数的浮点表示不是唯一的。当小数点的位置改变时，阶码也随着相应改变，因此可以用多个浮点形式表示同一个数。

浮点数所能表示的数值范围主要由阶码决定，所表示数值的精度则由尾数决定。为了充分利用尾数来表示更多的有效数字，通常采用规格化浮点数。规格化就是将尾数的绝对值限定在区间[0.5, 1]。当尾数用补码表示时，需要注意如下问题。

① 若尾数 $M\geq 0$，则其规格化的尾数形式为 M=0.1×××…×，其中，×可为 0，也可为 1，即将尾数限定在区间[0.5, 1]。

② 若尾数 $M<0$，则其规格化的尾数形式为 M=1.0×××…×，其中，×可为 0，也可为 1，即将尾数 M 的范围限定在区间[–1, –0.5]。

如果浮点数的阶码（包括 1 位阶符）用 R 位的移码表示，尾数（包括 1 位数符）用 M 位的补码表示，则这种浮点数所能表示的数值范围如下。

最大的正数：$+(1-2^{-M+1})\times 2^{(2^{R-1}-1)}$，最小的负数：$-1\times 2^{(2^{R-1}-1)}$

（3）工业标准 IEEE 754。IEEE 754 是由 IEEE 制定的有关浮点数的工业标准，被广泛采用。该标准的表示形式如下：

$$(-1)^S 2^E \left(b_0 b_1 b_2 b_3 \cdots b_{p-1}\right)$$

其中，$(-1)^S$ 为该浮点数的数符，当 S 为 0 时表示正数，S 为 1 时表示负数；E 为指数（阶码），用移码表示；$\left(b_0 b_1 b_2 b_3 \cdots b_{p-1}\right)$ 为尾数，其长度为 P 位，用原码表示。

目前，计算机中主要使用 3 种形式的 IEEE 754 浮点数，如表 1-2 所示。

表 1-2 3 种形式的 IEEE 754 浮点数格式

参 数	单精度浮点数	双精度浮点数	扩充精度浮点数
浮点数字长	32	64	80
尾数长度 P	23	52	64
符号位 S	1	1	1
指数长度 E	8	11	15
最大指数	+127	+1023	+16 383
最小指数	−126	−1022	−16 382
指数偏移量	+127	+1023	+16 383
可表示的实数范围	$10^{-38}\sim 10^{38}$	$10^{-308}\sim 10^{308}$	$10^{-4932}\sim 10^{4932}$

根据 IEEE 754 标准，被编码的值分为 3 种不同的情况：规格化的值、非规格化的值和特殊值，规格化的值为最普遍的情形。

① 规格化的值。

当阶码部分的二进制值不全为 0 也不全为 1 时，所表示的是规格化的值。例如，在单精度浮点格式下，阶码为 10110011 时，偏移量为+127（01111111），则其表示的真值为 10110011−01111111=00110100，转换为十进制后为 52。

对于尾数部分，由于约定小数点左边隐含有一位，通常这位数就是 1，因此单精度浮点数尾数的有效位数为24位，即尾数为1.××…×。也就是说，不溢出的情况下尾数M的值在1≤M<2之中，这是一种获得一个额外精度位的表示技巧。

例如，单精度浮点数格式下，$b_0 b_1 \cdots b_{22}=01001001100010001001011$ 时，其对应的尾数真值为 $1+2^{-2}+2^{-5}+2^{-8}+2^{-9}+2^{-13}+2^{-17}+2^{-20}+2^{-22}+2^{-23}$ 即尾数的真值为 1.28724038600921630859375（在程序中以十进制方式输出时，由于精度的原因不能完全给出此值）。

【例 1.5】 利用 IEEE 754 标准将数 176.0625 表示为单精度浮点数。

解：首先将该十进制转换成二进制数。

$(176.0625)_{10}=(10110000.0001)_2$

其次对二进制数进行规格化处理：10110000.0001＝1◊01100000001×2^7。这就保证了使 b_0 为 1，而且小数点应当在◊位置上。将 b_0 去掉并扩展为单精度浮点数所规定的 23 位尾数 01100000001000000000000。

然后求阶码，上述表示中的指数为 7，而单精度浮点数规定指数的偏移量为 127（注意，不是前面移码描述中所提到的 128），即在指数 7 上加 127。那么，E=7+127=134，则指数的移码表示为 10000110。

最后，可得到$(176.0625)_{10}$的单精度浮点数表示形式：

0 10000110 01100000001000000000000

② 非规格化的值。

当阶码部分的二进制值全为 0 时，所表示的数是非规格化的。在这种情况下，指数的真值为 1−偏移量（对于单精度浮点数为−126，对于双精度浮点数为−1022），尾数的值就是二进制形式对应的小数，不包含隐含的 1。

非规格化数有两个用途：一是用来表示数值 0，二是表示那些非常接近于 0 的数。因为在规格化表示方式下，必须使尾数大于等于 1，因此不能表示出 0。实际上，+0.0 的浮点表示是符号、阶码和尾数的二进制表示都全为 0。需要注意的是，符号位为 1 而阶码和尾数部分全为 0 时表示−0.0。也就是说，+0.0 和−0.0 在浮点表示时有所不同。

③ 特殊值。

当阶码部分的二进制值全为 1 时，表示特殊的值。当尾数部分全部为 0 时表示无穷大，当符号位为 0 时表示+∞，当符号位为 1 时表示−∞。当浮点运算溢出时，用无穷来表示。当尾数部分不全为 0 时，称为“NaN”，即“不是一个数”。当运算结果不是实数或者无穷，就表示为 NaN。

（4）浮点数的运算。设有浮点数 $X=M\times 2^i$，$Y=N\times 2^j$，求 $X\pm Y$ 的运算过程要经过对阶、求尾数和（差）、结果规格化并判溢出、舍入处理和溢出判别等步骤。

① 对阶。使两个数的阶码相同。令 $K=|i-j|$，把阶码小的数的尾数右移 K 位，使其阶码加上 K。

② 求尾数和（差）。

③ 结果规格化并判溢出。若运算结果所得的尾数不是规格化的数，则需要进行规格化处理。当尾数溢出时，需要调整阶码。

④ 舍入处理。在对结果右规时，尾数的最低位将因移出而丢掉。另外，在对阶过程中也会将尾数右移使最低位丢掉。这就需要进行舍入处理，以求得最小的运算误差。

⑤ 溢出判别。以阶码为准，若阶码溢出，则运算结果溢出；若阶码下溢（小于最小值），

则结果为0；否则结果正确，无溢出。

浮点数相乘，其积的阶码等于两乘数的阶码相加，积的尾数等于两乘数的尾数相乘。浮点数相除，其商的阶码等于被除数的阶码减去除数的阶码，商的尾数等于被除数的尾数除以除数的尾数。乘除运算的结果都需要进行规格化处理并判断阶码是否溢出。

1.1.4 校验码

计算机系统运行时，为了确保数据在传送过程中正确无误，一是提高硬件电路的可靠性，二是提高代码的校验能力，包括查错和纠错。通常使用校验码的方法来检测传送的数据是否出错。其基本思想是把数据可能出现的编码分为两类：合法编码和错误编码。合法编码用于传送数据，错误编码是不允许在数据中出现的编码。合理地设计错误编码以及编码规则，使得数据在传送中出现某种错误时会变成错误编码，这样就可以检测出接收到的数据是否有错。

所谓码距，是指一个编码系统中任意两个合法编码之间至少有多少个二进制位不同。例如，4位8421码的码距为1，在传输过程中，该代码的一位或多位发生错误，都将变成另外一个合法的编码，因此这种代码无检错能力。下面简要介绍常用的3种校验码：奇偶校验码、海明码和循环冗余校验码。

1．奇偶校验码

奇偶校验（Parity Codes）是一种简单有效的校验方法。这种方法通过在编码中增加一位校验位来使编码中1的个数为奇数（奇校验）或者为偶数（偶校验），从而使码距变为2。对于奇校验，它可以检测代码中奇数位出错的编码，但不能发现偶数位出错的情况，即当合法编码中的奇数位发生了错误时，即编码中的1变成0或0变成1，则该编码中1的个数的奇偶性就发生了变化，从而可以发现错误。

常用的奇偶校验码有3种：水平奇偶校验码、垂直奇偶校验码和水平垂直校验码。

2．海明码

海明码（Hamming Code）是由贝尔实验室的Richard Hamming设计的，是一种利用奇偶性来检错和纠错的校验方法。海明码的构成方法是在数据位之间的特定位置上插入 k 个校验位，通过扩大码距来实现检错和纠错。

设数据位是 n 位，校验位是 k 位，则 n 和 k 必须满足以下关系：

$$2^k - 1 \geqslant n + k$$

海明码的编码规则如下。

设 k 个校验位为 $P_k,P_{k-1},\cdots,P_1$，n 个数据位为 $D_{n-1},D_{n-2},\cdots,D_1,D_0$，对应的海明码为 $H_{n+k},H_{n+k-1},\cdots,H_1$，那么：

（1）P_i 在海明码的第 2^{i-1} 位置，即 $H_j=P_i$，且 $j=2^{i-1}$，数据位则依序从低到高占据海明码中剩下的位置。

（2）海明码中的任何一位都是由若干个校验位来校验的。其对应关系如下：被校验的海明位的下标等于所有参与校验该位的校验位的下标之和，而校验位由自身校验。

对于 8 位的数据位，进行海明校验需要 4 个校验位（$2^3-1=7$, $2^4-1=15>8+4$）。令数据位为 $D_7,D_6,D_5,D_4,D_3,D_2,D_1,D_0$，校验位为 P_4,P_3,P_2,P_1，形成的海明码为 $H_{12}, H_{11},\cdots,H_3,H_2,H_1$，则编码过程如下。

（1）确定 D 与 P 在海明码中的位置，如下所示：

H_{12}	H_{11}	H_{10}	H_9	H_8	H_7	H_6	H_5	H_4	H_3	H_2	H_1
D_7	D_6	D_5	D_4	P_4	D_3	D_2	D_1	P_3	D_0	P_2	P_1

（2）确定校验关系，如表 1-3 所示。

表 1-3　海明码的校验关系表

海　明　码	海明码的下标	校 验 位 组	说明（偶校验）
$H_1(P_1)$	1	P_1	P_1 校验：P_1、D_0、D_1、D_3、D_4、D_6 即 $P_1=D_0\oplus D_1\oplus D_3\oplus D_4\oplus D_6$
$H_2(P_2)$	2	P_2	P_2 校验：P_2、D_0、D_2、D_3、D_5、D_6 即 $P_2=D_0\oplus D_2\oplus D_3\oplus D_5\oplus D_6$
$H_3(D_0)$	3 = 1+2	P_1, P_2	
$H_4(P_3)$	4	P_3	P_3 校验：P_3、D_1、D_2、D_3、D_7 即 $P_3=D_1\oplus D_2\oplus D_3\oplus D_7$
$H_5(D_1)$	5 = 1+4	P_1, P_3	
$H_6(D_2)$	6 = 2+4	P_2, P_3	
$H_7(D_3)$	7 = 1+2+4	P_1, P_2, P_3	
$H_8(P_4)$	8	P_4	P_4 校验：P_4、D_4、D_5、D_6、D_7 即 $P_4=D_4\oplus D_5\oplus D_6\oplus D_7$
$H_9(D_4)$	9 = 1+8	P_1, P_4	
$H_{10}(D_5)$	10 = 2+8	P_2, P_4	
$H_{11}(D_6)$	11 = 1+2+8	P_1, P_2, P_4	
$H_{12}(D_7)$	12 = 4+8	P_3, P_4	

若采用奇校验，则将各校验位的偶校验值取反即可。

（3）检测错误。对使用海明编码的数据进行差错检测很简单，只需做以下计算：

$G_1=P_1\oplus D_0\oplus D_1\oplus D_3\oplus D_4\oplus D_6$

$G_2=P_2\oplus D_0\oplus D_2\oplus D_3\oplus D_5\oplus D_6$

$G_3=P_3\oplus D_1\oplus D_2\oplus D_3\oplus D_7$

$G_4=P_4\oplus D_4\oplus D_5\oplus D_6\oplus D_7$

若采用偶校验，则 $G_4G_3G_2G_1$ 全为 0 时表示接收到的数据无错误（奇校验应全为 1）。当

$G_4G_3G_2G_1$不全为0时说明发生了差错，而且$G_4G_3G_2G_1$的十进制值指出了发生错误的位置，例如$G_4G_3G_2G_1$=1010，说明$H_{10}(D_5)$出错了，将其取反即可纠正错误。

【例1.6】 设数据为01101001，试采用4个校验位求其偶校验方式的海明码。

解：$D_7D_6D_5D_4D_3D_2D_1D_0$=01101001，根据公式

$P_1 = D_0 \oplus D_1 \oplus D_3 \oplus D_4 \oplus D_6 = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 1$

$P_2 = D_0 \oplus D_2 \oplus D_3 \oplus D_5 \oplus D_6 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 0$

$P_3 = D_1 \oplus D_2 \oplus D_3 \oplus D_7 = 0 \oplus 0 \oplus 1 \oplus 0 = 1$

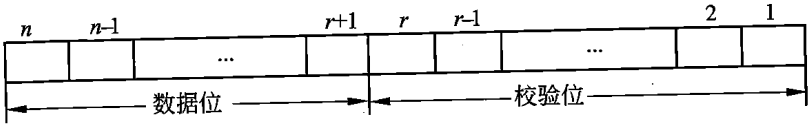
$P_4 = D_4 \oplus D_5 \oplus D_6 \oplus D_7 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$

求得的海明码为：

H_{12}	H_{11}	H_{10}	H_9	H_8	H_7	H_6	H_5	H_4	H_3	H_2	H_1
D_7	D_6	D_5	D_4	P_4	D_3	D_2	D_1	P_3	D_0	P_2	P_1
0	1	1	0	0	1	0	0	1	1	0	1

3. 循环冗余校验码

循环冗余校验码（Cyclic Redundancy Check，CRC）广泛应用于数据通信领域和磁介质存储系统中。它利用生成多项式为k个数据位产生r个校验位来进行编码，其编码长度为$k+r$。CRC的代码格式为：



由此可知，循环冗余校验码是由两部分组成的，左边为信息码（数据），右边为校验码。若信息码占k位，则校验码就占$n-k$位。其中，n为CRC码的字长，所以又称为（n，k）码。校验码是由信息码产生的，校验码位数越多，该代码的校验能力就越强。在求CRC编码时，采用的是模2运算。模2加减运算的规则是按位运算，不发生借位和进位。

1.2 计算机体系结构

1.2.1 计算机体系结构的发展

1. 计算机系统结构概述

1964年，阿姆达尔（G.M.Amdahl）在介绍IBM360系统时指出：计算机体系结构是站在

程序员的角度所看到的计算机属性，即程序员要能编写出可在机器上正确运行的程序所必须了解的概念性结构和功能特性。

1982 年，梅尔斯（G.J.Myers）在其所著的《计算机体系结构的进展》（Advances in Computer Architecture）一书中定义了组成计算机系统的若干层次，每一层都提供一定的功能支持它上面的一层，并把不同层之间的界面定义为某种类型的体系结构。Myers 的定义发展了Amdahl 的概念性结构思想，明确了传统体系结构就是指硬件与软件之间的界面，即指令集体系结构。

1984 年，拜尔（J.L.Baer）在一篇题为“计算机体系结构（Computer Architecture）”的文章中给出了一个含义更加广泛的定义：体系结构由结构、组织、实现、性能 4 个基本方面组成。其中，结构指计算机系统各种硬件的互连，组织指各种部件的动态联系与管理，实现指各模块设计的组装完成，性能指计算机系统的行为表现。这个定义发展了Amdahl 的功能特性思想。显然，这里的计算机系统组织又成为体系结构的一个子集。

计算机体系结构、计算机组织和计算机实现三者的关系如下。

（1）计算机体系结构（Computer Architecture）是指计算机的概念性结构和功能属性。

（2）计算机组织（Computer Organization）是指计算机体系结构的逻辑实现，包括机器内的数据流和控制流的组成以及逻辑设计等（常称为计算机组成原理）。

（3）计算机实现（Computer Implementation）是指计算机组织的物理实现。

2．计算机体系结构分类

（1）从宏观上按处理机的数量进行分类，分为单处理系统、并行处理与多处理系统和分布式处理系统。

- 单处理系统（Uni-processing System）。利用一个处理单元与其他外部设备结合起来，实现存储、计算、通信、输入与输出等功能的系统。
- 并行处理与多处理系统（Parallel Processing and Multiprocessing System）。为了充分发挥问题求解过程中处理的并行性，将两个以上的处理机互连起来，彼此进行通信协调，以便共同求解一个大问题的计算机系统。
- 分布式处理系统（Distributed Processing System）。指物理上远距离而松耦合的多计算机系统。其中，物理上的远距离意味着通信时间与处理时间相比已不可忽略，在通信线路上的数据传输速率要比在处理机内部总线上传输慢得多，这也正是松耦合的含义。

（2）从微观上按并行程度分类，有 Flynn 分类法、冯泽云分类法、Handler 分类法和 Kuck 分类法。

- Flynn 分类法。1966 年，M.J.Flynn 提出按指令流和数据流的多少进行分类。指令流为机器执行的指令序列，数据流是由指令调用的数据序列。Flynn 把计算机系统的结构分为单指令流、单数据流（Single Instruction stream Single Data stream，SISD），单指

令流、多数据流（Single Instruction stream Multiple Data stream，SIMD），多指令流、单数据流（Multiple Instruction stream Single Data stream，MISD）和多指令流、多数据流（Multiple Instruction stream Multiple Data stream，MIMD）4类。

- 冯泽云分类法。1972年，美籍华人冯泽云（Tse-yun Feng）提出按并行度对各种计算机系统进行结构分类。所谓最大并行度Pm是指计算机系统在单位时间内能够处理的最大二进制位数。冯泽云把计算机系统分成字串行位串行（WSBS）计算机、字并行位串行（WPBS）计算机、字串行位并行（WSBP）计算机和字并行位并行（WPBP）计算机4类。
- Handler分类法。1977年，德国的汉德勒（Wolfgang Handler）提出一个基于硬件并行程度计算并行度的方法，把计算机的硬件结构分为3个层次：处理机级、每个处理机中的算逻单元级、每个算逻单元中的逻辑门电路级。分别计算这三级中可以并行或流水处理的程序，即可算出某系统的并行度。
- Kuck分类法。1978年，美国的库克（David J.Kuck）提出与Flynn分类法类似的方法，用指令流和执行流（Execution Stream）及其多重性来描述计算机系统控制结构的特征。Kuck把系统结构分为单指令流单执行流（SISE）、单指令流多执行流（SIME）、多指令流单执行流（MISE）和多指令流多执行流（MIME）4类。

3．指令系统

一个处理器支持的指令和指令的字节级编码称为其指令集体系结构（Instruction Set Architecture，ISA），不同的处理器族支持不同的指令集体系结构，因此，一个程序被编译在一种机器上运行，往往不能在另一种机器上运行。

1）指令集体系结构的分类

从体系结构的观点对指令集进行分类，可以根据下述5个方面。

（1）操作数在CPU中的存储方式，即操作数从主存中取出后保存在什么地方。

（2）显式操作数的数量，即在典型的指令中有多少个显式命名的操作数。

（3）操作数的位置，即任一个ALU指令的操作数能否放在主存中，如何定位。

（4）指令的操作，即在指令集中提供哪些操作。

（5）操作数的类型与大小。

按暂存机制分类，根据在CPU内部存储操作数的区别，可以把指令集体系分为3类：堆栈（Stack）、累加器（Accumulator）和寄存器组（a set of Registers）。

通用寄存器（General-Purpose Register Machines，GPR）的关键性优点是编译程序能有效地使用寄存器，无论是计算表达式的值，还是从全局的角度使用寄存器来保存变量的值。在求解表达式时，寄存器比堆栈或者累加器能提供更加灵活的次序。更重要的是，寄存器能用来保

存变量。当变量分配给寄存器时，访存流量（Memory Traffic）就会减少，程序运行就会加速，而且代码密度也会得到改善。用户可以用指令集的两个主要特征来区分 GPR 体系结构。第一个是 ALU 指令有两个或 3 个操作数。在三操作数格式中，指令包括两个源操作数和一个目的操作数；在二操作数格式中，有一个操作数既是源操作数又是目的操作数。第二个是 ALU 指令中有几个操作数是存储器地址，对于典型的 ALU 指令，这个数可能在 1～3 之间。

2）CISC 和 RISC

CISC 和 RISC 是指令集发展的两种途径。

（1）CISC（Complex Instruction Set Computer，复杂指令集计算机）的基本思想是进一步增强原有指令的功能，用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，导致机器的指令系统越来越庞大、复杂。事实上，目前使用的绝大多数计算机都属于 CISC 类型。

CISC 的主要弊端如下。

① 指令集过分庞杂。

② 微程序技术是 CISC 的重要支柱，每条复杂指令都要通过执行一段解释性微程序才能完成，这就需要多个 CPU 周期，从而降低了机器的处理速度。

③ 由于指令系统过分庞大，使高级语言编译程序选择目标指令的范围很大，并使编译程序本身冗长、复杂，从而难以优化编译使之生成真正高效的目标代码。

④ CISC 强调完善的中断控制，势必导致动作繁多、设计复杂、研制周期长。

⑤ CISC 给芯片设计带来很多困难，使芯片种类增多，出错几率增大，成本提高而成品率降低。

（2）RISC（Reduced Instruction Set Computer，精简指令集计算机）的基本思想是通过减少指令总数和简化指令功能降低硬件设计的复杂度，使指令能单周期执行，并通过优化编译提高指令的执行速度，采用硬布线控制逻辑优化编译程序。RISC 在 20 世纪 70 年代末开始兴起，导致机器的指令系统进一步精炼而简单。

RISC 的关键技术如下。

① 重叠寄存器窗口技术。在伯克利的 RISC 项目中首先采用了重叠寄存器窗口（Overlapping Register Windows）技术。其基本思想是在处理机中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。每个过程使用其中相邻的 3 个窗口和一个公共的窗口，而在这些窗口中有一个窗口是与前一个过程共用，还有一个窗口是与下一个过程共用的。与前一过程共用的窗口可以用来存放前一过程传送给本过程的参数，同时也存放本过程传送给前一过程的计算结果。同样，与下一过程共用窗口可以用来存放本过程传送给下一过程的参数和存放下一过程传送给本过程的计算结果。

② 优化编译技术。RISC使用了大量的寄存器，如何合理地分配寄存器、提高寄存器的使用效率及减少访存次数等，都应通过编译技术的优化来实现。

③ 超流水及超标量技术。为了进一步提高流水线速度而采用的技术。

④ 硬布线逻辑与微程序相结合在微程序技术中。

（3）优化。

为了提高目标程序的实现效率，人们对大量的机器语言目标代码及其执行情况进行了统计。对程序中出现的各种指令以及指令串进行统计得到的百分比称为静态使用频度。在程序执行过程中，对出现的各种指令以及指令串进行统计得到的百分比称为动态使用频度。按静态使用频度来改进目标代码可减少目标程序所占的存储空间，按动态使用频度来改进目标代码可减少目标程序运行的执行时间。大量统计表明，动态和静态使用频度两者非常接近，最常用的指令是存、取、条件转移等。对它们加以优化，既可以减少程序所需的存储空间，又可以提高程序的执行速度。

面向高级程序语言的优化思路是尽可能缩小高级语言与机器语言之间的语义差距，以利于支持高级语言编译系统，缩短编译程序的长度和编译所需的时间。

面向操作系统的优化思路是进一步缩小操作系统与体系结构之间的语义差距，以利于减少操作系统运行所需的辅助时间，节省操作系统软件所占用的存储空间。操作系统的实现依赖于体系结构对它的支持。许多传统机器指令，例如算术逻辑指令、字符编辑指令、移位指令和控制转移指令等，都可用于操作系统的实现。此外，还有相当一部分指令是专门为实现操作系统的各种功能而设计的。

3）指令的流水处理

（1）指令控制方式。指令控制方式有顺序方式、重叠方式和流水方式3种。

① 顺序方式。顺序方式是指各条机器指令之间顺序串行地执行，执行完一条指令后才取下一条指令，而且每条机器指令内部的各个微操作也是顺序串行地执行。这种方式的优点是控制简单。缺点是速度慢，机器各部件的利用率低。

② 重叠方式。重叠方式是指在解释第K条指令的操作完成之前就可以开始解释第K+1条指令，如图1-2所示。通常采用的是一次重叠，即在任何时候，指令分析部件和指令执行部件都只有相邻两条指令在重叠解释。这种方式的优点是速度有所提高，控制也不太复杂。缺点是会出现冲突、转移和相关等问题，在设计时必须想办法解决。

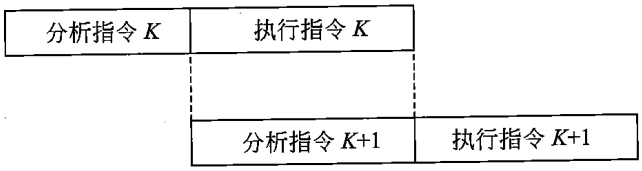


图1-2 一次重叠处理

③ 流水方式。流水方式是模仿工业生产过程的流水线（如汽车装配线）而提出的一种指令控制方式。流水（Pipelining）技术是把并行性或并发性嵌入到计算机系统里的一种形式，它把重复的顺序处理过程分解为若干子过程，每个子过程能在专用的独立模块上有效地并发工作，如图 1-3 所示。

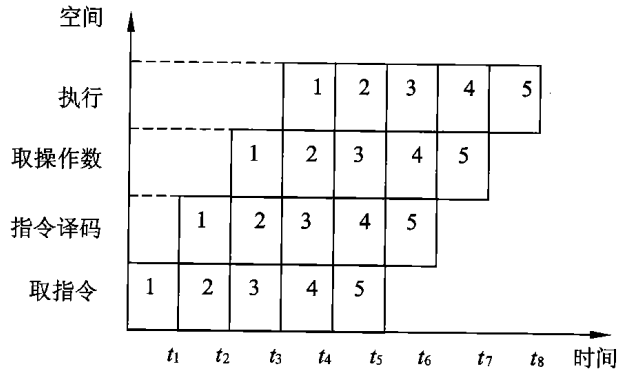


图 1-3　流水处理的时空图

在概念上，“流水”可以看成是“重叠”的延伸。差别仅在于“一次重叠”是把一条指令解释分解为两个子过程，而“流水”则是分解为更多的子过程。

（2）流水线的种类。

① 从流水的级别上，可分为部件级、处理机级以及系统级的流水。

② 从流水的功能上，可分为单功能流水线和多功能流水线。

③ 从流水的连接上，可分为静态流水线和动态流水线。

④ 从流水是否有反馈回路，可分为线性流水线和非线性流水线。

⑤ 从流水的流动顺序上，可分为同步流水线和异步流水线。

⑥ 从流水线的数据表示上，可分为标量流水线和向量流水线。

（3）流水的相关处理。

由于流水时机器同时解释多条指令，这些指令可能有对同一主存单元或同一寄存器的“先写后读”的要求，这时就出现了相关。这种相关包括指令相关、访存操作数相关以及通用寄存器组相关等，它只影响相关的两条或几条指令，而且最多影响流水线的某些段推后工作，并不会改动指令缓冲器中预取到的指令内容，影响是局部的，所以称为局部性相关。解决局部性相关有两种方法：推后法和通路法。推后法是推后对相关单元的读，直至写入完成。通路法设置相关专用通路，使得不必先把运算结果写入相关存储单元，再从这里读出后才能使用，而是经过相关专用通路直接使用运算结果，以加快速度。

转移指令（尤其是条件转移指令）与它后面的指令之间存在关联，使之不能同时解释。执行转移指令时，可能会改动指令缓冲器中预取到的指令内容，从而会造成流水线吞吐率和效率下降，比局部性相关的影响要严重得多，所以称为全局性相关。

解决全局性相关有3种方法：猜测转移分支、加快和提前形成条件码、加快短循环程序的处理。

条件转移指令有两个分支，一个分支是按原来的顺序继续执行下去，称为转移不成功分支；另一个分支是按转移后的新指令序列执行，称为转移成功分支。许多流水机器都猜选转移不成功分支，若猜对的几率很大，流水线的吞吐率和效率就会比不采用猜测法时高得多。

尽早获得条件码以便对流水线简化条件转移的处理。例如，一个乘法运算所需的时间较长，但在运算之前就能知道其结果为正或为负，或者是否为0，因此，加快单条指令内部条件码的形成，或者在一段程序内提前形成条件码，对转移问题的顺利解决是很有好处的。

由于程序中广泛采用循环结构，因此流水线大多采用特殊措施以加快循环程序的处理。例如，使整个循环程序都放入指令缓冲存储器中，对提高流水效率和吞吐率均有明显效果。中断和转移一样，也会引起流水线断流。好在中断出现的概率要比条件转移出现的概率低得多，因此只要处理好断点现场保护及中断后的恢复，尽量缩短断流时间即可。

RISC中采用的流水技术有3种：超流水线、超标量以及超长指令字。

① 超流水线（Super Pipe Line）技术。它通过细化流水、增加级数和提高主频，使得在每个机器周期内能完成一个甚至两个浮点操作。其实质是以时间换取空间。超流水机器的特征是在所有的功能单元都采用流水，并有更高的时钟频率和更深的流水深度。由于它只限于指令级的并行，所以超流水机器的CPI（Clock Cycles Per Instruction，每个指令需要的机器周期数）值稍高。

② 超标量（Super Scalar）技术。它通过内装多条流水线来同时执行多个处理，其时钟频率虽然与一般流水接近，却有更小的CPI。其实质是以空间换取时间。

③ 超长指令字（Very Long Instruction Word，VLIW）技术。VLIW和超标量都是20世纪80年代出现的概念，其共同点是要同时执行多条指令，其不同在于超标量依靠硬件来实现并行处理的调度，VLIW则充分发挥软件的作用，而使硬件简化，性能提高。VLIW有更小的CPI值，但需要有足够高的时钟频率。

（4）吞吐率和流水建立时间。

吞吐率是指单位时间内流水线处理机流出的结果数。对指令而言，就是单位时间内执行的指令数。如果流水线的子过程所用时间不一样，则吞吐率 p 应为最长子过程的倒数，即

$$p = 1/\max\{\Delta t_1, \Delta t_2, \cdots, \Delta t_m\}$$

流水线开始工作，需经过一定时间才能达到最大吞吐率，这就是建立时间。若 m 个子过程

所用时间一样，均为 Δt_0，则建立时间 $T_0 = m\Delta t_0$。

4．阵列处理机、并行处理机和多处理机

并行性包括同时性和并发性。其中，同时性是指两个或两个以上的事件在同一时刻发生，并发性是指两个或两个以上的事件在同一时间间隔内连续发生。

从计算机信息处理的步骤和阶段的角度看，并行处理可分为如下几类。

（1）存储器操作并行。

（2）处理器操作步骤并行（流水线处理机）。

（3）处理器操作并行（阵列处理机）。

（4）指令、任务、作业并行（多处理机、分布处理系统、计算机网络）。

1）阵列处理机

阵列处理机将重复设置的多个处理单元（PU）按一定方式连成阵列，在单个控制部件（CU）控制下，对分配给自己的数据进行处理，并行地完成一条指令所规定的操作。这是一种单指令流多数据流计算机，通过资源重复实现并行性。

2）并行处理机

SIMD 和 MIMD 是典型的并行计算机，SIMD 有共享存储器和分布存储器两种形式。

在具有共享存储器的 SIMD 结构（如图 1-4 所示）中，将若干个存储器构成统一的并行处理机存储器，通过互联网络 ICN 为整个并行系统的所有处理单元共享。其中，PE 为处理单元，CU 为控制部件，M 为共享存储器，ICN 为互联网络。

分布式存储器的 SIMD 结构如图 1-5 所示，其中，PE 为处理单元，CU 为控制部件，PEM 为局部存储器，ICN 为互联网络。含有多个同样结构的处理单元，通过寻径网络 ICN 以一定的方式互相连接。

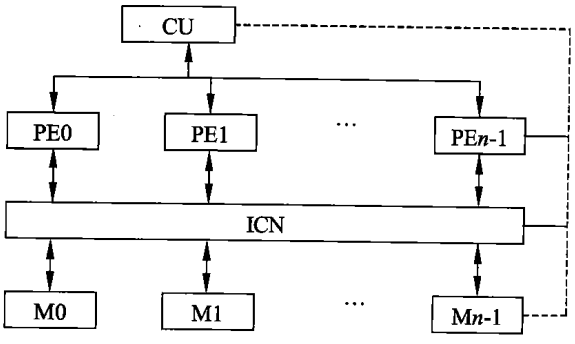


图 1-4　具有共享存储器的 SIMD 结构

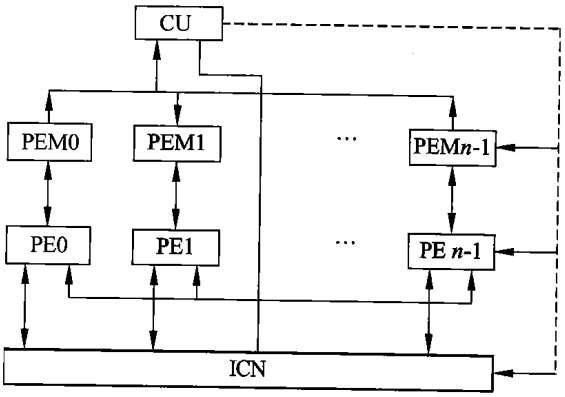


图 1-5　具有分布存储器的 SIMD 结构

分布存储器的并行处理机结构中有两类存储器，一类存储器附属于主处理机，主处理机实现整个并行处理机的管理，在其附属的存储器内常驻操作系统。另一类是分布在各个处理单元上的存储器（即PEM），这类存储器用来保存程序和数据。在阵列控制部件的统一指挥下，实现并行操作。程序和数据通过主机装入控制存储器。通过控制部件的是单指令流，所以指令的执行顺序还是和单处理机一样，基本上是串行处理。指令送到控制部件进行译码。划分后的数据集合通过向量数据总线分布到所有PE的本地存储器PEM。PE通过数据寻径网络互连。数据寻径网络执行PE间的通信。控制部件通过执行程序来控制寻径网络。PE的同步由控制部件的硬件实现。

3）多处理机

多处理机系统是由多台处理机组成的系统，每台处理机有属于自己的控制部件，可以执行独立的程序，共享一个主存储器和所有的外部设备。它是多指令流多数据流计算机。在多处理机系统中，机间的互连技术决定了多处理机的性能。多处理机之间的互连要满足高频带、低成本、连接方式的多样性以及在不规则通信情况下连接的无冲突性。

4）其他计算机

集群一般是指连接在一起的两个或多个计算机（结点）。集群计算机是一种并行或分布式处理系统，由很多连接在一起的独立计算机组成，像一个单集成的计算机资源一样协同工作，主要用来解决大型计算问题。计算机结点可以是一个单处理器或多处理器的系统，拥有内存、I/O设备和操作系统。连接在一起的计算机集群对用户和应用程序来说像一个单一的系统，这样的系统可以提供一种价格合理的且可获得所需性能和快速而可靠的服务的解决方案。

1.2.2 存储系统

1. 存储器的层次结构

计算机系统中可能包括各种存储器，如CPU内部的通用寄存器组、CPU内的Cache（高速缓存）、CPU外部的Cache、主板上的主存储器、主板外的联机（在线）磁盘存储器以及脱机（离线）的磁带存储器和光盘存储器等。不同特点的存储器通过适当的硬件、软件有机地组合在一起形成计算机的存储体系结构，如图1-6所示。其中，Cache和主存之间的交互功能全部由硬件实现，而主存与辅存之间的交互功能可由硬件和软件结合起来实现。

2. 存储器的分类

1）按存储器所处的位置分类

按存储器所处的位置可分为内存和外存。

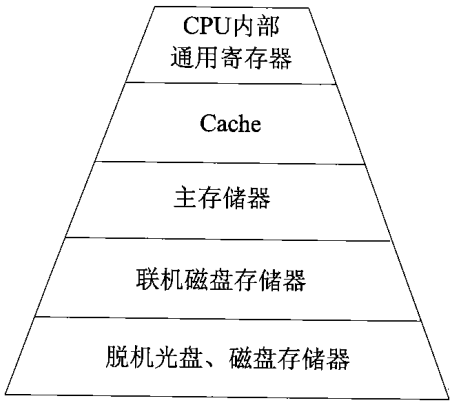


图 1-6　存储系统的层次结构

（1）内存。也称为主存，设在主机内或主机板上，用来存放机器当前运行所需要的程序和数据，以便向 CPU 提供信息。相对于外存，其特点是容量小、速度快。

（2）外存。也称为辅存，如磁盘、磁带和光盘等，用来存放当前不参加运行的大量信息，而在需要时调入内存。

2）按存储器的构成材料分类

按构成存储器的材料可分为磁存储器、半导体存储器和光存储器。

（1）磁存储器。磁存储器是用磁性介质做成的，如磁芯、磁泡、磁膜、磁鼓、磁带及磁盘等。

（2）半导体存储器。根据所用元件又可分为双极型和 MOS 型；根据数据是否需要刷新又可分为静态（Static Memory）和动态（Dynamic Memory）两类。

（3）光存储器。利用光学方法读/写数据的存储器，如光盘（Optical Disk）。

3）按存储器的工作方式分类

按存储器的工作方式可分为读/写存储器和只读存储器。

（1）读/写存储器（Random Access Memory，RAM）。它指既能读取数据也能存入数据的存储器。

（2）只读存储器。工作过程中仅能读取的存储器，根据数据的写入方式，这种存储器又可细分为 ROM、PROM、EPROM 和 EEPROM 等类型。

① 固定只读存储器（Read Only Memory，ROM）。这种存储器是在厂家生产时就写好数据的，其内容只能读出，不能改变。一般用于存放系统程序 BIOS 和用于微程序控制。

② 可编程的只读存储器（Programmable Read Only Memory，PROM）。其中的内容可以由用户一次性地写入，写入后不能再修改。

③ 可擦除可编程的只读存储器（Erasable Programmable Read Only Memory，EPROM）。其中的内容既可以读出，也可以由用户写入，写入后还可以修改。改写的方法是写入之前先用紫外线照射15～20分钟以擦去所有信息，然后再用特殊的电子设备写入信息。

④ 电擦除可编程的只读存储器（Electrically Erasable Programmable Read Only Memory，EEPROM）。与EPROM相似，EEPROM中的内容既可以读出，也可以进行改写。只不过这种存储器是用电擦除的方法进行数据的改写。

⑤ 闪速存储器（Flash Memory）。简称闪存，闪存的特性介于EPROM和EEPROM之间，类似于EEPROM，也可使用电信号进行信息的擦除操作。整块闪存可以在数秒内删除，速度远快于EPROM。

4）按访问方式分类

按访问方式可分为按地址访问的存储器和按内容访问的存储器。

5）按寻址方式分类

按寻址方式可分为随机存储器、顺序存储器和直接存储器。

（1）随机存储器（Random Access Memory，RAM）。这种存储器可对任何存储单元存入或读取数据，访问任何一个存储单元所需的时间是相同的。

（2）顺序存储器（Sequentially Addressed Memory，SAM）。访问数据所需要的时间与数据所在的存储位置相关，磁带是典型的顺序存储器。

（3）直接存储器（Direct Addressed Memory，DAM）。介于随机存取和顺序存取之间的一种寻址方式。磁盘是一种直接存取存储器，它对磁道的寻址是随机的，而在一个磁道内则是顺序寻址。

3．相联存储器

相联存储器是一种按内容访问的存储器。其工作原理就是把数据或数据的某一部分作为关键字，按顺序写入信息，读出时并行地将该关键字与存储器中的每一单元进行比较，找出存储器中所有与关键字相同的数据字，特别适合于信息的检索和更新。

相联存储器的结构如图1-7所示，其中，输入检索寄存器用来存放要检索的内容（关键字），屏蔽寄存器用来屏蔽那些不参与检索的字段，比较器将检索的关键字与存储体的每一单元进行比较。为了提高速度，比较器的数量应很大。对于位比较器，应每位对应一个，应有$2^m \times N$个，对于字比较器应有2^m个。匹配寄存器用来记录比较的结果，它应有2^m个二进制位，用来记录2^m个比较器的结果，1为相等（匹配），0为不相等（不匹配）。

相联存储器可用在高速缓冲存储器中，在虚拟存储器中用来作为段表、页表或快表存储器，用在数据库和知识库中。

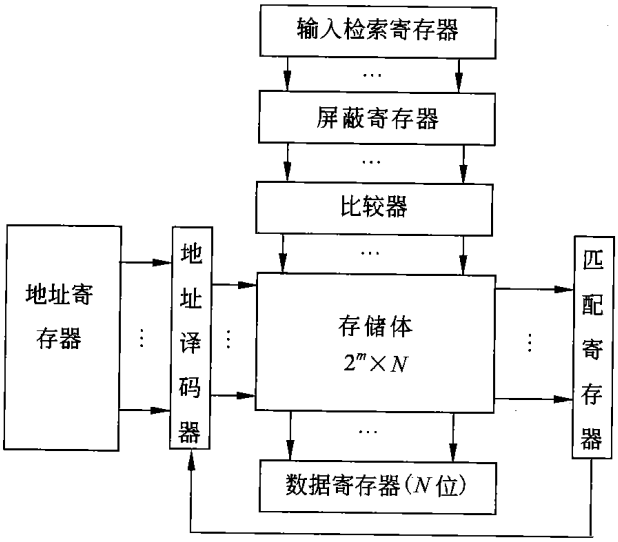


图 1-7　相联存储器的结构框图

4．高速缓存

高速缓存用来存放当前最活跃的程序和数据，其特点是：位于 CPU 与主存之间；容量一般在几千字节到几兆字节之间；速度一般比主存快 5～10 倍，由快速半导体存储器构成；其内容是主存局部域的副本，对程序员来说是透明的。

1）高速缓存的组成

高速缓存（Cache）、主存（Main Memory）与 CPU 的关系如图 1-8 所示。

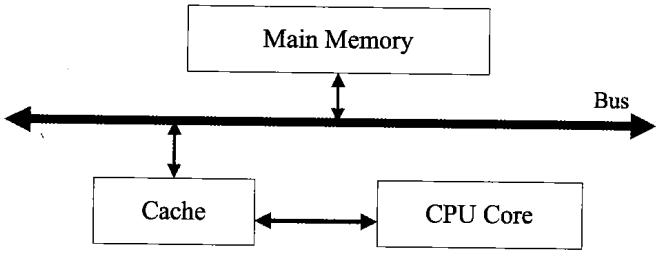


图 1-8　高速缓存、主存和 CPU 的关系示意图

Cache 存储器部分用来存放主存的部分拷贝（副本）信息。控制部分的功能是判断 CPU 要访问的信息是否在 Cache 存储器中，若在即为命中，若不在则没有命中。命中时直接对 Cache 存储器寻址；未命中时，要按照替换原则决定主存的一块信息放到 Cache 存储器的哪一块里。

现代 CPU 中 Cache 分为了多个层级，如图 1-9 所示。

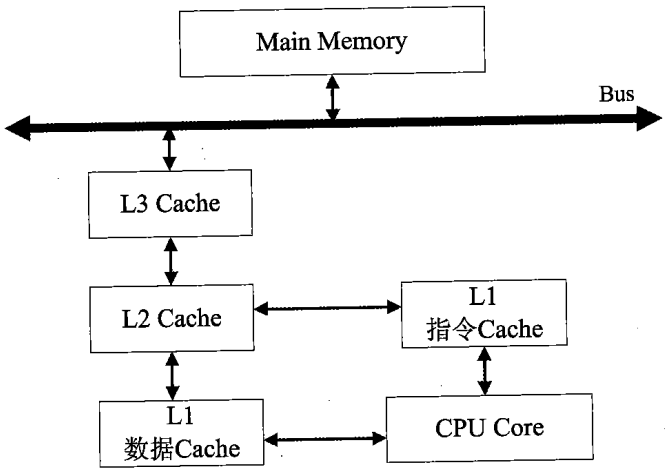


图 1-9　三级 Cache 示意图

2）高速缓存中的地址映像方法

在 CPU 工作时，送出的是主存单元的地址，而应从 Cache 存储器中读/写信息。这就需要将主存地址转换成 Cache 存储器的地址，这种地址的转换称为地址映像。Cache 的地址映像有如下 3 种方法。

（1）直接映像。直接映像是指主存的块与 Cache 块的对应关系是固定的，如图 1-10 所示。

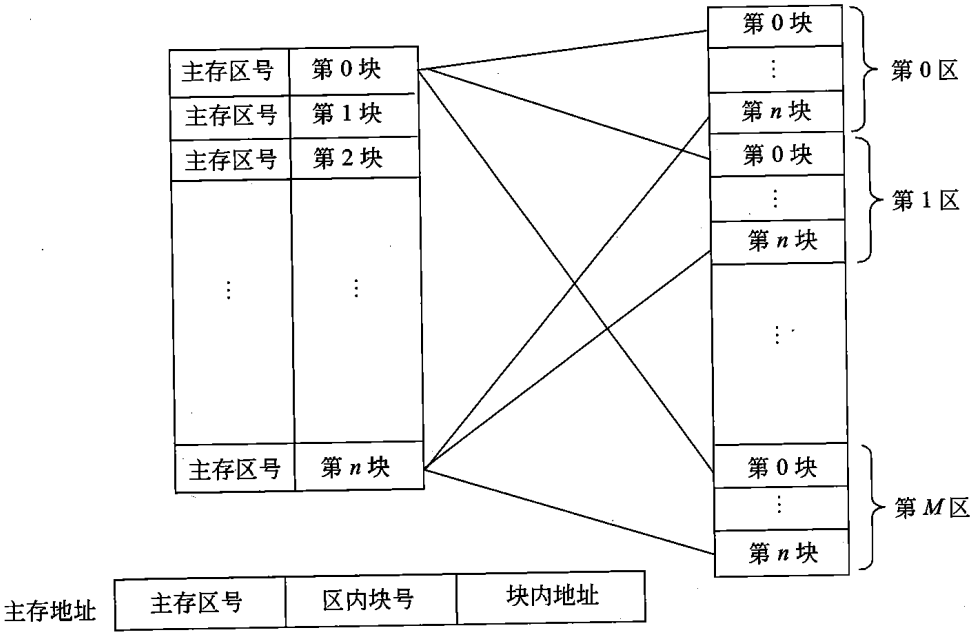


图 1-10　直接映像示意图

在这种映像方式下，由于主存中的块只能存放在 Cache 存储器的相同块号中，因此，只要主存地址中的主存区号与 Cache 中记录的主存区号相同，则表明访问 Cache 命中。一旦命中，由主存地址中的区内块号立即可得到要访问的 Cache 存储器中的块，而块内地址就是主存地址中给出的低位地址。

直接映像方式的优点是地址变换很简单，缺点是灵活性差。例如，不同区号中块号相同的块无法同时调入 Cache 存储器，即使 Cache 存储器中有空着的块也不能利用。

（2）全相联映像。全相联映像如图 1-11 所示。同样，主存与 Cache 存储器均分成大小相同的块。这种映像方式允许主存的任一块可以调入 Cache 存储器的任何一个块的空间中。

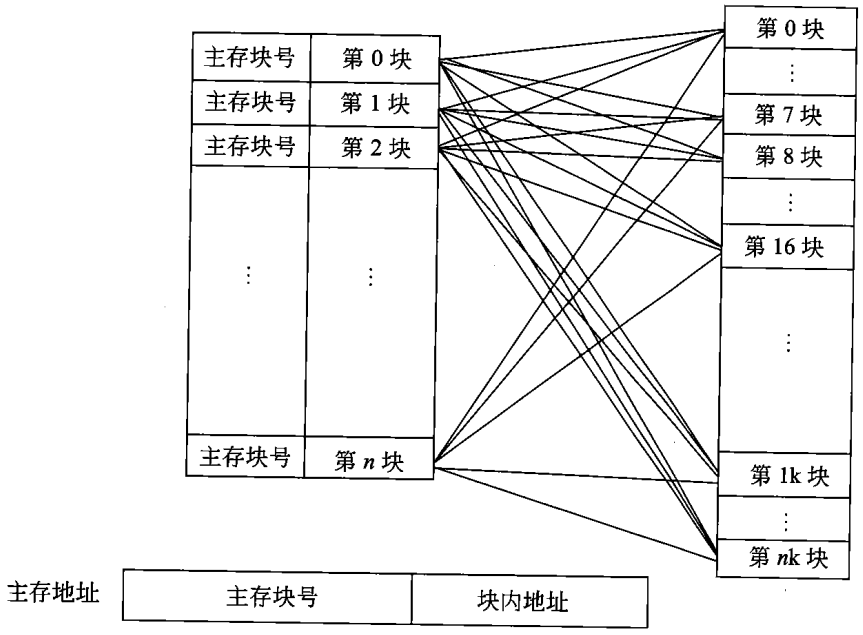


图 1-11　全相联映像示意图

例如，主存为 64MB，Cache 为 32KB，块的大小为 4KB（块内地址需要 12 位），因此主存分为 16384 块，块号从 0～16383，表示块号需要 14 位，Cache 分为 8 块，块号为 0～7，表示块号需 3 位。存放主存块号的相联存储器需要有 Cache 块个数相同数目的单元（该例中为 8），相联存储器中每个单元记录所存储的主存块的块号，该例中相联存储器每个单元应为 14 位，共 8 个单元。

在地址变换时，利用主存地址高位表示的主存块号与 Cache 中相联存储器所有单元中记录的主存块号进行比较，若相同即为命中。这时相联存储器单元的编号就对应要访问 Cache 的块号，从而在相应的 Cache 块中根据块内地址（上例中块内地址是 12 位，Cache 与主存的块内地

址是相同的）访问到相应的存储单元。

全相联映像的主要优点是主存的块调入Cache的位置不受限制，十分灵活。其主要缺点是无法从主存块号中直接获得Cache的块号，变换比较复杂，速度比较慢。

（3）组相联映像。这种方式是前面两种方式的折中。具体方法是将Cache中的块再分成组。例如，假定Cache有16块，再将每两块分为1组，则Cache就分为8组。主存同样分区，每区16块，再将每两块分为1组，则每区就分为8组。

组相联映像就是规定组采用直接映像方式而块采用全相联映像方式。也就是说，主存任何区的0组只能存到Cache的0组中，1组只能存到Cache的1组中，依此类推。组内的块则采用全相联映像方式，即一组内的块可以任意存放。也就是说，主存一组中的任一块可以存入Cache相应组的任一块中。

在这种方式下，通过直接映像方式来决定组号，在一组内再用全相联映像方式来决定Cache中的块号。由主存地址高位决定的主存区号与Cache中区号比较可决定是否命中。主存后面的地址即为组号。

3）替换算法

替换算法的目标就是使Cache获得尽可能高的命中率。常用算法有如下几种。

（1）随机替换算法。就是用随机数发生器产生一个要替换的块号，将该块替换出去。

（2）先进先出算法。就是将最先进入Cache的信息块替换出去。

（3）近期最少使用算法。这种方法是将近期最少使用的Cache中的信息块替换出去。

（4）优化替换算法。这种方法必须先执行一次程序，统计Cache的替换情况。有了这样的先验信息，在第二次执行该程序时便可以用最有效的方式来替换。

4）Cache的性能分析

Cache的性能是计算机系统性能的重要方面。命中率是Cache的一个重要指标，但不是最主要的指标。Cache设计的目标是在成本允许的条件下达到较高的命中率，使存储系统具有最短的平均访问时间。设H_c为Cache的命中率，t_c为Cache的存取时间，t_m为主存的访问时间，则Cache存储器的等效加权平均访问时间t_a为：

$$t_a = H_c t_c + (1-H_c)t_m = t_c + (1-H_c)(t_m - t_c)$$

这里假设Cache访问和主存访问是同时启动的，其中，t_c为Cache命中时的访问时间，$(t_m - t_c)$为失效访问时间。如果在Cache不命中时才启动主存，则

$$t_a = t_c + (1-H_c)t_m$$

在指令流水线中，Cache访问作为流水线中的一个操作阶段，Cache失效将影响指令的流水。因此，降低Cache的失效率是提高Cache性能的一项重要措施。当Cache容量比较小时，

容量因素在 Cache 失效中占有比较大的比例。降低 Cache 失效率的方法主要有选择恰当的块容量、提高 Cache 的容量和提高 Cache 的相联度等。

Cache 的命中率与 Cache 容量的关系如图 1-12 所示。Cache 容量越大，则命中率越高，随着 Cache 容量的增加，其失效率接近 0%（命中率逐渐接近 100%）。但是，增加 Cache 容量意味着增加 Cache 的成本和增加 Cache 的命中时间。

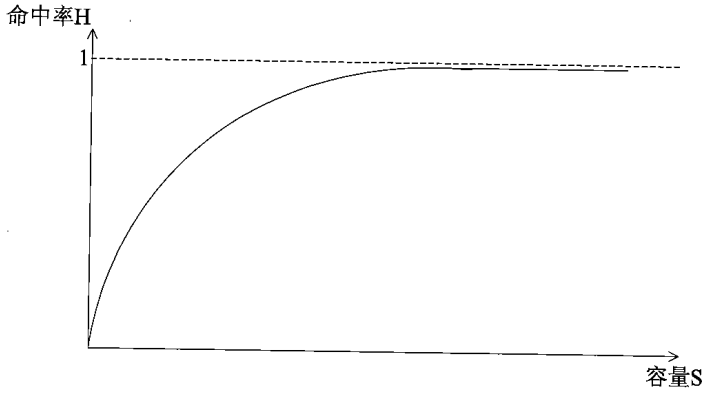


图 1-12　Cache 容量与命中率的关系

5）多级 Cache

在多级 Cache 的计算机中，Cache 分为一级（L1 Cache）、二级（L2 Cache）、三级（L3 Cache）等，CPU 访存时首先查找 L1 Cache，如果不命中，则访问 L2 Cache，直到所有级别的 Cache 都不命中，才访问主存。通常要求 L1 Cache 的速度足够快，以赶上 CPU 的主频。如果 Cache 为两级，则 L1 Cache 的容量一般都比较小，为几千字节到几十千字节；L2 Cache 则具有较高的容量，一般为几百字节到几兆字节，以使高速缓存具有足够高的命中率。

5．虚拟存储器

在概念上，可以将主存存储器看作一个由若干个字节构成的存储空间，每个字节（称为一个存储单元）有一个地址编号，主存单元的该地址称为物理地址（Physical Address）。当需要访问主存中的数据时，由 CPU 给出要访问数据所在的存储单元地址，然后由主存的读写控制部件定位对应的存储单元，对其进行读（或写）操作来完成访问操作。

现代系统提供了一种对主存的抽象，称为虚拟存储（Virtual Memory），使用虚拟地址（Virtual Address，由 CPU 生成）的概念来访问主存，使用专门的 MMU（Memory Management Unit）将虚拟地址转换为物理地址后访问主存。设主存容量为 4GB，则其简化后的访问操作和内存模型如图 1-13 所示。

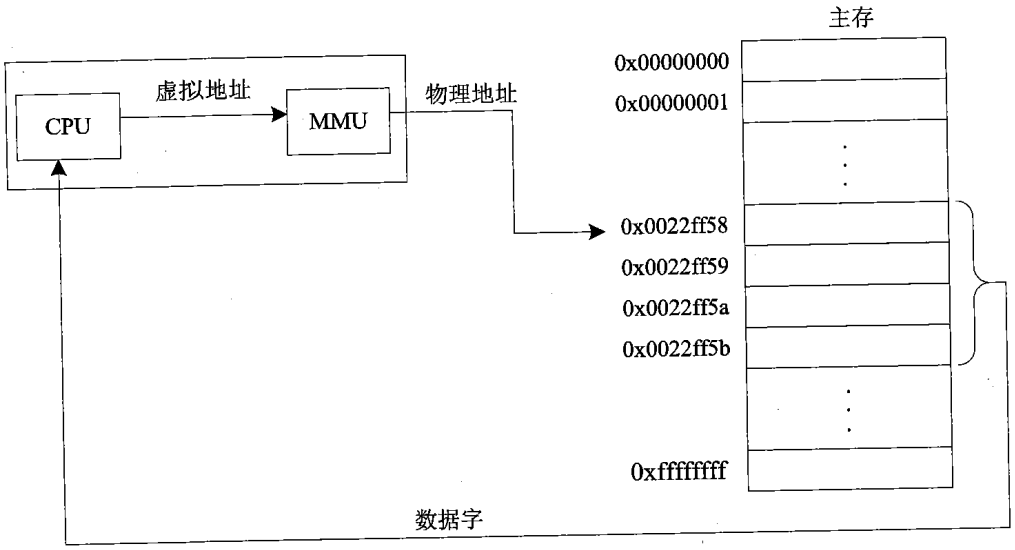


图 1-13 内存模型及使用虚拟地址访存示意图

虚拟存储器实际上是一种逻辑存储器，实质是对物理存储设备进行逻辑化的处理，并将统一的逻辑视图呈现给用户。因此，用户在使用时，操作的是虚拟设备，无需关心底层的物理环境，从而可以充分利用基于异构平台的存储空间，达到最优化的使用效率。

6．外存储器

外存储器用来存放暂时不用的程序和数据，并且以文件的形式存储。CPU 不能直接访问外存中的程序和数据，只有将其以文件为单位调入主存才可访问。外存储器主要由磁表面存储器（如磁盘、磁带）、光盘存储器及固态硬盘（采用 Flash 芯片或 DRAM 作为存储介质的存储器）构成。

1）磁表面存储器

在磁表面存储器中，磁盘的存取速度较快，且具有较大的存储容量，是目前广泛使用的外存储器。磁盘存储器由盘片、驱动器、控制器和接口组成。盘片用来存储信息。驱动器用于驱动磁头沿盘面径向运动以寻找目标磁道位置，驱动盘片以额定速率稳定旋转，并且控制数据的写入和读出。控制器接收主机发来的命令，将它转换成磁盘驱动器的控制命令，并实现主机和驱动器之间数据格式的转换及数据传送，以控制驱动器的读/写操作。一个控制器可以控制一台或多台驱动器。接口是主机和磁盘存储器之间的连接逻辑。

硬盘是最常见的外存储器。一个硬盘驱动器内可装有多个盘片，组成盘片组，每个盘片都配有一个独立的磁头。所有记录面上相同序号的磁道构成一个圆柱面，其编号与磁道编号相同。

将文件存储在硬盘上时尽可能放在同一圆柱面上，或者放在相邻柱面上，这样可以缩短寻道时间。

为了正确地存储信息，将盘片划成许多同心圆，称为磁道，从外到里编号，最外一圈为 0 道，往内道号依次增加。沿径向的单位距离的磁道数称为道密度，单位为 tpi（每英寸磁道数）。将一个磁道沿圆周等分为若干段，每段称为一个扇段或扇区，每个扇区内可存放一个固定长度的数据块，如 512B。磁道上单位距离可记录的位数称为位密度，单位为 bpi（每英寸位数）。因为每条磁道上的扇区数相同，而每个扇区的大小又一样，所以每条磁道都记录同样多的信息。又因为里圈磁道圆周比外圈磁道的圆周小，所以里圈磁道的位密度要比外圈磁道的位密度高。最内圈的位密度称为最大位密度。

硬盘的寻址信息由硬盘驱动号、圆柱面号、磁头号（记录面号）、数据块号（或扇区号）以及交换量组成。

磁盘容量有两种指标：一种是非格式化容量，它是指一个磁盘所能存储的总位数；另一种是格式化容量，它是指各扇区中数据区容量的总和。计算公式分别如下：

非格式化容量=面数×（磁道数/面）×内圆周长×最大位密度

格式化容量=面数×（磁道数/面）×（扇区数/道）×（字节数/扇区）

按盘片是否固定、磁头是否移动等指标，硬盘可分为移动磁头固定盘片的磁盘存储器、固定磁头的磁盘存储器、移动磁头可换盘片的磁盘存储器和温彻斯特磁盘存储器（简称温盘）。

2）光盘存储器

光盘存储器是一种采用聚焦激光束在盘式介质上非接触地记录高密度信息的新型存储装置。

根据性能和用途，光盘存储器可分为只读型光盘（CD-ROM）、只写一次型光盘（WORM）和可擦除型光盘。只读型光盘是由生产厂家预先用激光在盘片上蚀刻不能再改写的各种信息，目前这类光盘的使用很普遍。只写一次型光盘是指由用户一次写入、可多次读出但不能擦除的光盘，写入方法是利用聚焦激光束的热能，使光盘表面发生永久性变化而实现的。可擦除型光盘是读/写型光盘，它是利用激光照射引起介质的可逆性物理变化来记录信息。

光盘存储器由光学、电学和机械部件等组成。其特点是记录密度高、存储容量大、采用非接触式读/写信息（光头距离光盘通常为 2mm）、信息可长期保存（其寿命达 10 年以上）、采用多通道记录时数据传送率可超过 200Mb/s、制造成本低、对机械结构的精度要求不高、存取时间较长。

3）固态硬盘

固态硬盘的存储介质分为两种，一种是采用闪存（FLASH 芯片）作为存储介质，另外一种是采用 DRAM 作为存储介质。

基于闪存的固态硬盘是固态硬盘的主要类别，其主体是一块 PCB 板，板上最基本的配件就是控制芯片、缓存芯片和用于存储数据的闪存芯片。主控芯片是固态硬盘的大脑，其作用有两个：一是合理调配数据在各个闪存芯片上的负荷，二则是承担数据中转的作用，连接闪存芯片和外部 SATA 接口。不同主控芯片的能力相差非常大，在数据处理能力、算法，对闪存芯片的读取写入控制上会有非常大的不同，直接会导致固态硬盘产品在性能上差距很大。

固态硬盘的接口规范和定义、功能及使用方法上与普通硬盘基本相同，外形和尺寸也基本与普通的 2.5 英寸硬盘一致。

固态硬盘具有传统机械硬盘不具备的读写快速、质量轻、能耗低以及体积小等特点，但其价格仍较为昂贵，容量较低，一旦硬件损坏，数据较难恢复。

7．磁盘阵列技术

磁盘阵列是由多台磁盘存储器组成的一个快速、大容量、高可靠的外存子系统。现在常见的磁盘阵列称为廉价冗余磁盘阵列（Redundant Array of Independent Disk，RAID）。目前，常见的 RAID 如表 1-4 所示。

表 1-4 廉价冗余磁盘阵列

RAID 级别	说 明
RAID-0	RAID-0 是一种不具备容错能力的磁盘阵列。由 N 个磁盘存储器组成的 0 级阵列，其平均故障间隔时间（MTBF）是单个磁盘存储器的 N 分之一，但数据传输率是单个磁盘存储器的 N 倍
RAID-1	RAID-1 是采用镜像容错改善可靠性的一种磁盘阵列
RAID-2	RAID-2 是采用海明码进行错误检测的一种磁盘阵列
RAID-3	RAID-3 减少了用于检验的磁盘存储器的个数，从而提高了磁盘阵列的有效容量。一般只有一个检验盘
RAID-4	RAID-4 是一种可独立地对组内各磁盘进行读/写的磁盘阵列，该阵列也只用一个检验盘
RAID-5	RAID-5 是对 RAID-4 的一种改进，它不设置专门的检验盘。同一个磁盘上既记录数据，也记录检验信息，这就解决了前面多个磁盘机争用一个检验盘的问题
RAID-6	RAID-6 磁盘阵列采用两级数据冗余和新的数据编码以解决数据恢复问题，使在两个磁盘出现故障时仍然能够正常工作。在进行写操作时，RAID-6 分别进行两个独立的校验运算，形成两个独立的冗余数据，写入两个不同的磁盘

除此之外，上述各种类型的 RAID 还可以组合起来，构成复合型的 RAID，此处不再赘述。

8．存储域网络

在大型服务器系统的背后都有一个网络，把一个或多个服务器与多个存储设备连接起来，

每个存储设备可以是 RAID、磁带备份系统、磁带库和 CD-ROM 库等，构成了存储域网络（Storage Area Network，SAN）。这样的网络不仅解决了服务器对存储容量的要求，还可以使多个服务器之间共享文件系统和辅助存储空间，避免数据和程序代码的重复存储，提高辅助存储器的利用率。另外，SAN 还实现了分布式存储系统的集中管理，降低了大容量存储系统的管理成本，提高了管理效率。存储域网络是连接服务器与存储设备的网络，它能够将多个分布在不同地点的 RAID 组织成一个逻辑存储设备，供多个服务器共享访问，如图 1-14 所示。

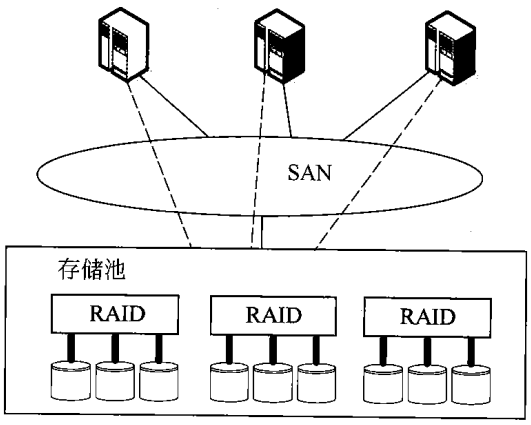


图 1-14　SAN 的结构

1.2.3　输入/输出技术

1. 微型计算机中最常用的内存与接口的编址方法

计算机系统中存在多种内存与接口地址的编址方法，常见的是下面两种：内存与接口地址独立编址和内存与接口地址统一编址。

1）内存与接口地址独立编址方法

在内存与接口地址独立编址方法下，内存地址和接口地址是完全独立的两个地址空间，它们是完全独立的并且是相互隔离的。访问数据时所使用的指令也完全不同，用于接口的指令只用于接口的读/写，其余的指令全都是用于内存的。因此，在编程序或读程序时很易使用和辨认。

这种编址方法的缺点是用于接口的指令太少、功能太弱。

2）内存与接口地址统一编址方法

在这种编址方法中，内存地址和接口地址统一在一个公共的地址空间里，即内存单元和接口共用地址空间。在这些地址空间里划分出一部分地址分配给接口使用，其余地址归内存单元使用。分配给内存的地址区间只能用于内存单元，接口绝不允许使用。同样，分配给接口的地址区间内存单元也绝不能使用。

这种编址方法的优点是原则上用于内存的指令全都可以用于接口，这就大大地增强了对接口的操作功能，而且在指令上也不再区分内存或接口指令。

该编址方法的缺点就在于整个地址空间被分成两部分，其中一部分分配给接口使用，剩余的为内存所用，这经常会导致内存地址不连续。由于用于内存的指令和用于接口的指令是完全

一样的，维护程序时就需要根据参数定义表仔细加以辨认。

2．直接程序控制

直接程序控制是指外设数据的输入/输出过程是在CPU执行程序的控制下完成的。这种方式分为无条件传送和程序查询方式两种情况。

1）无条件传送

在此情况下，外设总是准备好的，它可以无条件地随时接收CPU发来的输出数据，也能够无条件地随时向CPU提供需要输入的数据。

2）程序查询方式

在这种方式下，利用查询方式进行输入/输出，就是通过CPU执行程序来查询外设的状态，判断外设是否准备好接收数据或准备好了向CPU输入的数据。根据这种状态，CPU有针对性地为外设的输入/输出服务。

通常，一个计算机系统中可以存在着多种不同的外设，如果这些外设是用查询方式工作，则CPU应对这些外设逐一进行查询，发现哪个外设准备就绪就对该外设服务。这种工作方式有如下两大缺点。

（1）降低了CPU的效率。在这种工作方式下，CPU不做别的事，只是不停地对外设的状态进行查询。在实际的工程应用中，对于那些慢速的外设，在不影响外设工作的情况下，CPU应可以执行其他任务。

（2）对外部的突发事件无法做出实时响应。

3．中断方式

由程序控制I/O的方法，其主要缺点在于CPU必须等待I/O系统完成数据的传输任务，在此期间CPU需定期地查询I/O系统的状态，以确认传输是否完成。因此，整个系统的性能严重下降。

利用中断方式完成数据的输入/输出过程为：当I/O系统与外设交换数据时，CPU无须等待也不必去查询I/O的状态，而可以抽身出来处理其他任务。当I/O系统准备好以后，则发出中断请求信号通知CPU，CPU接到中断请求信号后，保存正在执行程序的现场，转入I/O中断服务程序的执行，完成与I/O系统的数据交换，然后再返回被打断的程序继续执行。与程序控制方式相比，中断方式因为CPU无须等待而提高了效率。

1）中断处理方法

在系统中具有多个中断源的情况下，常用的处理方法有多中断信号线法（Multiple Interrupt Lines）、中断软件查询法（Software Poll）、菊花链法（Daisy Chain）、总线仲裁法和中断向

量表法。

（1）多中断信号线法。每个中断源都有属于自己的一根中断请求信号线向 CPU 提出中断请求。

（2）中断软件查询法。当 CPU 检测到一个中断请求信号以后，即转入到中断服务程序去轮询每个中断源以确定是谁发出了中断请求信号。对各个设备的响应优先级由软件设定。

（3）菊花链法。软件查询的缺陷在于花费的时间太多。菊花链法实际上是一种硬件查询法。所有的 I/O 模块共享一根共同的中断请求线，而中断确认信号则以链式在各模块间相连。当 CPU 检测到中断请求信号时，则发出中断确认信号。中断确认信号依次在 I/O 模块间传递，直到发出请求的模块，该模块则把它的 ID 送往数据线由 CPU 读取。

（4）总线仲裁法。一个 I/O 设备在发出中断请求之前，必须先获得总线控制权，所以可由总线仲裁机制来裁定谁可以发出中断请求信号。当 CPU 发出中断响应信号后，该设备即把自己的 ID 发往数据线。

（5）中断向量表法。中断向量表用来保存各个中断源的中断服务程序的入口地址。当外设发出中断请求信号（INTR）以后，由中断控制器（INTC）确定其中断号，并根据中断号查找中断向量表来取得其中断服务程序的入口地址，同时 INTC 把中断请求信号提交给 CPU，如图 1-15 所示。中断源的优先级由 INTC 来控制。

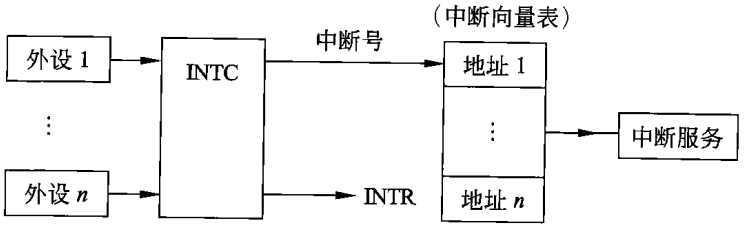


图 1-15　中断向量表法

2）中断优先级控制

在具有多个中断源的计算机系统中，各中断源对服务的要求紧迫程度可能不同。在这样的计算机系统中，就需要按中断源的轻重缓急来安排对它们的服务。

在中断优先级控制系统中，给最紧迫的中断源分配高的优先级，而给那些要求相对不紧迫（例如几百微秒到几毫秒）的中断源分配低一些的优先级。在进行优先级控制时解决以下两种情况。

（1）当不同优先级的多个中断源同时提出中断请求时，CPU 应优先响应优先级最高的中断源。

（2）当 CPU 正在对某一个中断源服务时，又有比它优先级更高的中断源提出中断请求，CPU 应能暂时中断正在执行的中断服务程序而转去对优先级更高的中断源服务，服务结束后再

回到原先被中断的优先级较低的中断服务程序继续执行，这种情况称为中断嵌套，即一个中断服务程序中嵌套着另一个中断服务程序。

4. 直接存储器存取方式

在计算机与外设交换数据的过程中，无论是无条件传送、利用查询方式传送还是利用中断方式传送，都需要由CPU通过执行程序来实现，这就限制了数据的传送速度。

直接内存存取（Direct Memory Access，DMA）是指数据在内存与I/O设备间的直接成块传送，即在内存与I/O设备间传送一个数据块的过程中，不需要CPU的任何干涉，只需要CPU在过程开始启动（即向设备发出“传送一块数据”的命令）与过程结束（CPU通过轮询或中断得知过程是否结束和下次操作是否准备就绪）时的处理，实际操作由DMA硬件直接执行完成，CPU在此传送过程中可做别的事情。

DMA传送的一般过程如图1-16所示。

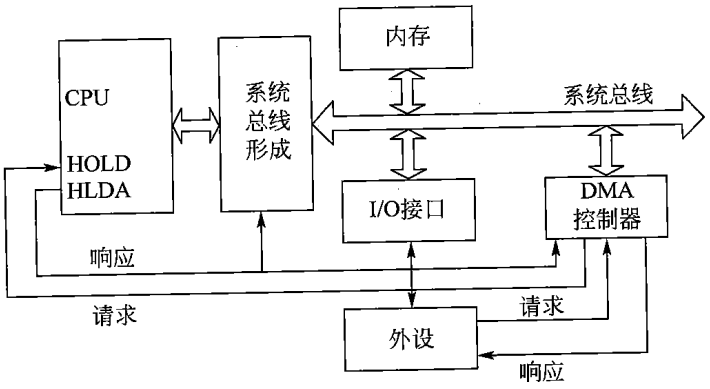


图1-16 DMA过程示意图

（1）外设向DMA控制器（DMAC）提出DMA传送的请求。

（2）DMA控制器向CPU提出请求，其请求信号通常加到CPU的保持请求输入端HOLD上。

（3）CPU在完成当前的总线周期后立即对此请求作出响应，CPU的响应包括两个方面的内容：一方面，CPU将有效的保持响应信号HLDA输出加到DMAC上，告诉DMAC它的请求已得到响应；另一方面，CPU将其输出的总线信号置为高阻，这就意味着CPU放弃了对总线的控制权。

（4）此时，DMAC获得了对系统总线的控制权，开始实施对系统总线的控制。同时向提出请求的外设送出DMAC的响应信号，告诉外设其请求已得到响应，现在准备开始进行数据的传送。

（5）DMAC 送出地址信号和控制信号，实现数据的高速传送。

（6）当 DMAC 将规定的字节数传送完时，它就将 HOLD 信号变为无效并加到 CPU 上，撤销对 CPU 的请求。CPU 检测到无效的 HOLD 就知道 DMAC 已传送结束，CPU 就送出无效的 HLDA 响应信号，同时重新获得系统总线的控制权，接着 DMA 前的总线周期继续执行下面的总线周期。

在此再强调说明，在 DMA 传送过程中无须 CPU 的干预，整个系统总线完全交给了 DMAC，由它控制系统总线完成数据传送。在 DMA 传送数据时要占用系统总线，根据占用总线方法的不同，DMA可以分为中央处理器停止法、总线周期分时法和总线周期挪用法等。无论采用哪种方法，在 DMA 传送数据期间，CPU 不能使用总线。

5．输入/输出处理机（IOP）

DMA 方式的出现减轻了 CPU 对 I/O 操作的控制，使得 CPU 的效率显著提高，而通道的出现则进一步提高了 CPU 的效率。

通道是一个具有特殊功能的处理器，又称为输入输出处理器（Input/Output Processor，IOP），它分担了 CPU 的一部分功能，可以实现对外围设备的统一管理，完成外围设备与主存之间的数据传送。

通道方式大大提高了 CPU 的工作效率，然而这种效率的提高是以增加更多的硬件为代价的。

外围处理机（Peripheral Processor Unit，PPU）方式是通道方式的进一步发展。PPU 是专用处理机，它根据主机的 I/O 命令，完成对外设数据的输入输出。在一些系统中，设置了多台 PPU，分别承担 I/O 控制、通信、维护诊断等任务。从某种意义上说，这种系统已变成分布式的多机系统。

1.2.4　总线结构

所谓总线（Bus），是指计算机设备和设备之间传输信息的公共数据通道。总线是连接计算机硬件系统内多种设备的通信线路，它的一个重要特征是由总线上的所有设备共享，因此可以将计算机系统内的多种设备连接到总线上。

1．总线的分类

微机中的总线分为数据总线、地址总线和控制总线 3 类。不同型号的 CPU 芯片，其数据总线、地址总线和控制总线的条数可能不同。

数据总线（Data Bus，DB）用来传送数据信息，是双向的。CPU 既可通过 DB 从内存或输入设备读入数据，也可通过 DB 将内部数据送至内存或输出设备。DB 的宽度决定了 CPU

和计算机其他设备之间每次交换数据的位数。

地址总线（Address Bus，AB）用于传送CPU发出的地址信息，是单向的。传送地址信息的目的是指明与CPU交换信息的内存单元或I/O设备。存储器是按地址访问的，所以每个存储单元都有一个固定地址，要访问1MB存储器中的任一单元，需要给出2^20个地址，即需要20位地址（2^20=1M）。因此，地址总线的宽度决定了CPU 的最大寻址能力。

控制总线（Control Bus，CB）用来传送控制信号、时序信号和状态信息等。其中有的信号是CPU向内存或外部设备发出的信息，有的是内存或外部设备向CPU发出的信息。显然，CB中的每一条线的信息传送方向是单方向且确定的，但CB作为一个整体则是双向的。所以，在各种结构框图中，凡涉及到控制总线CB，均是以双向线表示。

总线的性能直接影响到整机系统的性能，而且任何系统的研制和外围模块的开发都必须依从所采用的总线规范。总线技术随着微机结构的改进而不断发展与完善。

2．常见总线

（1）ISA总线。ISA是工业标准总线，只能支持16位的I/O设备，数据传输率大约是16Mb/s，也称为AT标准。

（2）EISA总线。EISA是在ISA总线的基础上发展起来的32位总线。该总线定义32位地址线、32位数据线以及其他控制信号线、电源线、地线等共196个接点。总线传输速率达33Mb/s。

（3）PCI总线。PCI总线是目前微型机上广泛采用的内总线，采用并行传输方式。PCI总线有适于32位机的124个信号的标准和适于64位机的188个信号的标准。PCI总线的传输速率至少为133Mb/s，64位PCI总线的传输速率为266Mb/s。PCI总线的工作与CPU的工作是相互独立的，也就是说，PCI总线时钟与处理器时钟是独立的、非同步的。PCI总线上的设备是即插即用的。接在PCI总线上的设备均可以提出总线请求，通过PCI管理器中的仲裁机构允许该设备成为主控设备，主控设备与从属设备间可以进行点对点的数据传输。PCI总线能够对所传输的地址和数据信号进行奇偶校验检测。

（4）PCI Express总线。PCI Express简称为PCI-E，采用点对点串行连接，每个设备都有自己的专用连接，不需要向整个总线请求带宽，而且可以把数据传输率提高到一个很高的频率。相对于传统PCI总线在单一时间周期内只能实现单向传输，PCI Express的双单工连接能提供更高的传输速率和质量。

PCI Express的接口根据总线位宽不同而有所差异，包括X1、X4、X8以及X16（X2模式将用于内部接口而非插槽模式），其中X1的传输速度为250Mb/s，而X16就是等于16倍于X1的速度，即是4Gb/s。较短的PCI Express卡可以插入较长的PCI Express插槽中使用。PCI Express接口能够支持热拔插。同时，PCI Express总线支持双向传输模式，还可以运行全双工

模式，它的双单工连接能提供更高的传输速率和质量，它们之间的差异与半双工和全双工类似。因此连接的每个装置都可以使用最大带宽。

（5）前端总线。微机系统中，前端总线（Front Side Bus，FSB）是将 CPU 连接到北桥芯片的总线。选购主板和 CPU 时，要注意两者的搭配问题，一般来说，如果 CPU 不超频，那么前端总线是由 CPU 决定的，如果主板不支持 CPU 所需要的前端总线，系统就无法工作。也就是说，需要主板和 CPU 都支持某个前端总线，系统才能工作。通常情况下，一个 CPU 默认的前端总线是唯一的。北桥芯片负责联系内存、显卡等数据吞吐量最大的部件，并与南桥芯片连接。CPU 通过前端总线（FSB）连接到北桥芯片，进而通过北桥芯片与内存、显卡交换数据。FSB 是 CPU 和外界交换数据的最主要通道，因此 FSB 的数据传输能力对计算机整体性能作用很大，如果没足够快的 FSB，再强的 CPU 也不能明显提高计算机整体速度。

（6）RS-232C。RS-232C 是一条串行外总线，其主要特点是所需传输线比较少，最少只需三条线（一条发、一条收、一条地线）即可实现全双工通信。传送距离远，用电平传送为 15m，电流环传送可达千米。有多种可供选择的传送速率。采用非归零码负逻辑工作，电平≤–3V 为逻辑 1，而电平≥+3V 为逻辑 0，具有较好的抗干扰性。

（7）SCSI 总线。小型计算机系统接口（SCSI）是一条并行外总线，广泛用于连接软硬磁盘、光盘、扫描仪等。该接口总线早期是 8 位的，后来发展到 16 位。传输速率由 SCSI–1 的 5Mb/s 到 16 位的 Ultra2 SCSI 的 80Mb/s。今天的传输速率已高达 320Mb/s。该总线上最多可接 63 种外设，传输距离可达 20m（差分传送）。

（8）SATA。SATA 是 Serial ATA 的缩写，即串行 ATA。它主要用作主板和大量存储设备（如硬盘及光盘驱动器）之间的数据传输之用。SATA 总线使用嵌入式时钟信号，具备了更强的纠错能力，与以往相比其最大的区别在于能对传输指令（不仅仅是数据）进行检查，如果发现错误会自动校正，这在很大程度上提高了数据传输的可靠性。串行接口还具有结构简单、支持热插拔的优点。

（9）USB。通用串行总线（USB）当前风头正劲，近几年得到十分广泛的应用。USB 由 4 条信号线组成，其中两条用于传送数据，另外两条传送+5V 容量为 500mA 的电源。可以经过集线器（Hub）进行树状连接，最多可达 5 层。该总线上可接 127 个设备。USB 1.0 有两种传送速率：低速为 1.5Mb/s，高速为 12Mb/s。USB 2.0 的传送速率为 480Mb/s。USB 总线最大的优点还在于它支持即插即用，并支持热插拔。

（10）IEEE-1394。IEEE-1394 是高速串行外总线，近几年得到广泛应用。IEEE-1394 也支持外设热插拔，可为外设提供电源，省去了外设自带的电源，能连接多个不同设备，支持同步和异步数据传输。IEEE-1394 由 6 条信号线组成，其中两条用于传送数据，两条传送控制信号，

另外两条传送8～40V容量为1500mA的电源，IEEE-1394总线理论上可接63个设备。IEEE-1394的传送速率从400Mb/s、800Mb/s、1600Mb/s直到3.2Gb/s。

（11）IEEE-488总线。IEEE-488是并行总线接口标准。微计算机、数字电压表、数码显示器等设备及其他仪器仪表均可用IEEE-488总线连接装配，它按照位并行、字节串行双向异步方式传输信号，连接方式为总线方式，仪器设备不需中介单元直接并联于总线上。总线上最多可连接15台设备。最大传输距离为20m，信号传输速度一般为500Kb/s，最大传输速度为1Mb/s。

1.3 安全性、可靠性与系统性能评测基础知识

1.3.1 计算机安全概述

计算机安全是一个涵盖非常广的课题，既包括硬件、软件和技术，又包括安全规划、安全管理和安全监督。计算机安全可包括安全管理、通信与网络安全、密码学、安全体系及模型、容错与容灾、涉及安全的应用程序及系统开发、法律、犯罪及道德规范等领域。

其中安全管理是非常重要的，作为信息系统的管理部门应根据管理原则和该系统处理数据的保密性，制定相应的管理制度或规范。例如，根据工作的重要程度确定系统的安全等级，根据确定的安全等级确定安全管理的范围，制定相应的机房管理制度、操作规程、系统维护措施以及应急措施等。

1. 计算机的安全等级

计算机系统中的三类安全性是指技术安全性、管理安全性和政策法律安全性。但是，一个安全产品的购买者如何知道产品的设计是否符合规范，是否能解决计算机网络的安全问题，不同的组织机构各自制定了一套安全评估准则。一些重要的安全评估准则如下。

（1）美国国防部和国家标准局推出的《可信计算机系统评估准则》（TCSEC）。

（2）加拿大的《可信计算机产品评估准则》（CTCPEC）。

（3）美国制定的《联邦（最低安全要求）评估准则》（FC）。

（4）欧洲英、法、德、荷四国国防部门信息安全机构联合制定的《信息技术安全评估准则》（ITSEC），该准则事实上已成为欧盟各国使用的共同评估标准。

（5）美国制定的《信息技术安全通用评估准则》（简称CC标准），国际标准组织（ISO）于1996年批准CC标准以ISO/IEC 15408—1999名称正式列入国际标准系列。

其中，美国国防部和国家标准局的《可信计算机系统评测标准》TCSEC/TDI将系统划分为

4 组 7 个等级，如表 1-5 所示。

表 1-5　安全性的级别

组	安全级别	定　义
1	A1	可验证安全设计。提供 B3 级保护，同时给出系统的形式化隐秘通道分析、非形式化代码一致性验证
2	B3	安全域。该级的 TCB 必须满足访问监控器的要求，提供系统恢复过程
	B2	结构化安全保护。建立形式化的安全策略模型，并对系统内的所有主体和客体实施自主访问和强制访问控制
	B1	标记安全保护。对系统的数据加以标记，并对标记的主体和客体实施强制存取控制
3	C2	受控访问控制。实际上是安全产品的最低档次，提供受控的存取保护，存取控制以用户为单位
	C1	只提供了非常初级的自主安全保护，能实现对用户和数据的分离，进行自主存取控制，数据的保护以用户组为单位
4	D	最低级别，保护措施很小，没有安全功能

2．安全威胁

随着信息交换的激增，安全威胁所造成的危害越来越被受到重视，因此对信息保密的需求也从军事、政治和外交等领域迅速扩展到民用和商用领域。所谓安全威胁，是指某个人、物、事件对某一资源的机密性、完整性、可用性或合法性所造成的危害。某种攻击就是威胁的具体实现。安全威胁分为两类：故意（如黑客渗透）和偶然（如信息发往错误的地址）。

典型的安全威胁举例如表 1-6 所示。

表 1-6　典型的安全威胁

威　胁	说　明
授权侵犯	为某一特权使用一个系统的人却将该系统用作其他未授权的目的
拒绝服务	对信息或其他资源的合法访问被无条件地拒绝，或推迟与时间密切相关的操作
窃听	信息从被监视的通信过程中泄漏出去
信息泄露	信息被泄漏或暴露给某个未授权的实体
截获/修改	某一通信数据项在传输过程中被改变、删除或替代
假冒	一个实体（人或系统）假装成另一个实体
否认	参与某次通信交换的一方否认曾发生过此次交换
非法使用	资源被某个未授权的人或者未授权的方式使用
人员疏忽	一个授权的人为了金钱或利益，或由于粗心将信息泄露给未授权的人
完整性破坏	通过对数据进行未授权的创建、修改或破坏，使数据的一致性受到损坏