



全国计算机技术与软件专业技术资格（水平）考试指定用书

软件设计师教程

（第4版）

褚华 主编

全国计算机专业技术资格考试办公室组编

清华大学出版社

本书仅提供部分阅读，如需完整版，请联系QQ: 461573687

提供各种书籍pdf下载，如有需要，请联系 QQ: 461573687

PDF制作说明：

本人可以提供各种PDF电子书资料，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我 QQ: 461573687, 或者 QQ: 2404062482。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ。因PDF电子书都有版权，请不要随意传播，最近pdf也越来越难做了，希望大家尊重下个人劳动，谢谢！

备用QQ:2404062482

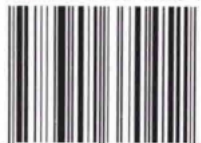
全国计算机技术与软件专业技术资格（水平）考试指定用书

根据人力资源和社会保障部、工业和信息化部文件，计算机技术与软件专业技术资格（水平）考试纳入全国专业技术人员职业资格证书制度的统一规划。通过考试获得证书的人员，表明其已具备从事相应专业岗位工作的水平和能力，用人单位可根据工作需要从获得证书的人员中择优聘任相应专业技术职务（技术员、助理工程师、工程师、高级工程师）。计算机技术与软件专业实施全国统一考试后，不再进行相应专业技术职务任职资格的评审工作。

清华大学出版社数字出版网站

WQBook  
www.wqbook.com

ISBN 978-7-302-37045-1



9 787302 370451 >

定价：79.00元

全国计算机技术与软件专业技术资格（水平）考

软件设计师教程

（第4版）

褚华 主编

全国计算机专业技术资格考试办公室组编

清华大学出版社

内 容 简 介

本书作为中级职称的软考指定教材,具有比较权威的指导意义。本书根据《软件设计师考试大纲》的重点内容,阐述了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(2014版),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(第三版)(修订版)》进行了认真修编,部分章节重写后形成的。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。

本书适合参加本书考试的考生和大学在校生作为教材。

本书扉页为防伪页,封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

软件设计师教程/褚华主编. —4版. —北京:清华大学出版社,2014

(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 978-7-302-37045-1

I. ①软… II. ①褚… III. ①软件设计-工程技术人员-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(2014)第143031号

责任编辑:柴文强 王冰飞

封面设计:傅瑞学

责任校对:徐俊伟

责任印制:宋 林

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:三河市新茂装订有限公司

经 销:全国新华书店

开 本:185mm×230mm 印 张:44.5 防伪页:1 字 数:967千字

版 次:2004年7月第1版 2014年9月第4版 印 次:2014年9月第1次印刷

印 数:1~20000

定 价:79.00元

产品编号:060347-01

序 言

由人力资源和社会保障部、工业和信息化部共同组织的“全国计算机技术与软件专业技术资格（水平）考试”（简称软考），肩负着科学评价选拔软件专业技术人才的光荣使命，肩负着正确引导软件行业专业技术人员潜心钻研、提高能力、加强创新的光荣使命，肩负着加强软件行业专业技术人员队伍建设的光荣使命。自 1991 年开考以来，软考坚持专业化、国际化、品牌化的发展方向，全国累计报名人数 330 万人，培养选拔软件行业专业技术人才 64 万人，部分考试标准与日本、韩国互认，为全国计算机和软件专业技术人员（包括香港、澳门和台湾地区来大陆就业的人员）提供了科学的评价体系和评价机制，为推动“两化”深度融合，提高工业信息化水平，走新型工业化道路提供了有力支撑。

党中央、国务院一直高度重视信息技术产业发展。以 2000 年的《国务院关于印发鼓励软件产业和集成电路产业发展的若干政策的通知》（国发【2000】18 号文件）和 2011 年的《国务院关于印发进一步鼓励软件产业和集成电路产业发展的若干政策的通知》（国发【2011】4 号文件）为重要标志的一系列政策措施，为软件产业和集成电路产业乃至整个信息技术产业发展提供了强劲动力。2011 年，我国软件产业实现业务收入超过 1.84 万亿元，产业规模是 2005 年的 4.7 倍，同比增长 32.4%，超过“十一五”期间平均增速 4.4 个百分点，实现了“十二五”的良好开局。软件产业占电子信息产业比重从 2000 年的 5.8% 上升到 19.9%。软件企业数量超过 3 万家，从业人数超过 300 万人。2012 年上半年，我国软件产业实现软件业务收入 10988 亿元，同比增长 26.2%。软件和信息服务业的持续快速发展，国民经济和社会信息化建设的深入开展，使软件人才和信息技术人才供给不足的问题依旧突出。按照国发【2011】4 号文件提出的“努力培养国际化、复合型、实用性人才”的要求，工业和信息化部教育与考试中心组织一批理论水平高、经验丰富的专家学者和业界精英，结合考试大纲和软件产业技术发展趋势，对原有的“全国计算机技术与软件专业技术资格（水平）考试教材和辅导用书”进行了更新，为广大软件行业从业人员提高学习能力、实践能力、创新能力和职业道德水平提供了依据。

当前，我国正处在全面建成小康社会的决定性阶段。坚持走中国特色新型工业化、信息化、城镇化、农业现代化道路，推动信息化和工业化深度融合、工业化和城镇化良性互动、城镇化和农业现代化相互协调，促进工业化、信息化、城镇化、农业现代化同步发展，是党中央的重要战略部署。造就规模宏大、素质优良的人才队伍，推动我国由人才大国迈向人才强国，既是

构成这一重要战略部署的紧迫任务,也是实施这一重要战略部署的关键措施。从现在起至全面建成小康社会的这一历史时期,信息技术仍然是走中国特色新型工业化、信息化、城镇化、农业现代化道路的先导性技术;全国计算机技术与软件专业技术资格(水平)考试也应该看做是落实党的十八大关于“推进各类人才队伍建设,实施重大人才工程,加大创新创业人才培养支持力度,重视实用人才培养”指示的重要组成部分。好雨知时节,当春乃发生——我相信,全国计算机技术与软件专业技术资格(水平)考试教材和辅导用书的及时更新必将为我国信息技术人才队伍发展壮大、为软件和信息服务业做大做强、为服务经济转型升级做出更大的贡献;同时我们也要注意,近年来,以云计算、物联网、移动互联网和大数据技术等为热点的新一代信息技术,正在对软件和信息服务业带来一系列深刻变化,也对软件和信息技术在各个领域的应用产生重要影响,我希望,在保持这套教材和辅导用书在一个时期内相对稳定的同时,也要注意及时反映信息技术的新变化、新进展,以跟上软件和信息服务业蓬勃发展的需要,跟上信息化以及新型工业化、城镇化和农业现代化建设蓬勃发展的需要。



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（第三版）修订版》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 章主要介绍操作系统中处理机管理、存储管理、设备管理、文件管理、作业管理以及网络与嵌入式操作系统基础知识。

第 4 章主要介绍软件工程中软件过程与过程模型、软件开发方法、软件工具与软件开发环境、软件项目管理、风险管理、软件度量基础知识。

第 5 章主要介绍系统分析与设计、需求分析与需求工程、结构化分析与设计、Web 应用系统分析与设计、用户界面设计、测试以及系统运行与维护基础知识。

第 6 章主要介绍网络与媒体基础知识，包括网络体系结构、网络互连设备、网络构件、网络应用和安全，以及多媒体声音、图形图像、动画与食品、多媒体网络与多媒体计算机方面的基础知识。

第 7 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。

第 8 章主要介绍数据结构的基础知识，包括线性结构、数组、矩阵、广义表、树和图，以及查找和排序的基本方法和算法。

第9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、分支界限法及其他算法。

第10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,以及设计模式的概念和应用。

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、第2章由张淑平编写,第3章由王亚平编写,第4章、第5章由褚华、霍秋艳编写,第6章由严体华、马志新、强编写,第7章由王亚平编写,第8章由张淑平、陈静玉编写,第9章由覃桂敏、褚华编写,第10章由霍秋艳、褚华编写,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,书中难免存在错漏和不妥之处,望读者指正,以利改进和提高。

编 者

2014年5月

目 录

第 1 章 计算机系统知识1	
1.1 计算机系统基础知识.....1	
1.1.1 计算机系统硬件基本组成.....1	
1.1.2 中央处理单元.....1	
1.1.3 数据表示.....4	
1.1.4 校验码.....9	
1.2 计算机体系结构.....11	
1.2.1 计算机体系结构的发展.....11	
1.2.2 存储系统.....19	
1.2.3 输入/输出技术.....29	
1.2.4 总线结构.....33	
1.3 安全性、可靠性与系统性能评测基础知识.....34	
1.3.1 计算机安全概述.....34	
1.3.2 加密技术和认证技术.....37	
1.3.3 计算机可靠性.....45	
1.3.4 计算机系统的性能评价.....47	
第 2 章 程序设计语言基础知识52	
2.1 程序设计语言概述.....52	
2.1.1 程序设计语言的基本概念.....52	
2.1.2 程序设计语言的基本成分.....56	
2.2 语言处理程序基础.....62	
2.2.1 汇编程序基本原理.....62	
2.2.2 编译程序基本原理.....64	
2.2.3 解释程序基本原理.....91	
第 3 章 操作系统知识94	
3.1 操作系统基础知识.....94	
3.1.1 操作系统的定义与作用.....94	
3.1.2 操作系统的特征与功能.....94	
3.1.3 操作系统的发展及分类.....95	
3.2 处理机管理.....98	
3.2.1 基本概念.....98	
3.2.2 进程的控制.....102	
3.2.3 进程间的通信.....102	
3.2.4 管程.....106	
3.2.5 进程调度.....108	
3.2.6 死锁.....110	
3.2.7 线程.....114	
3.3 存储管理.....115	
3.3.1 基本概念.....115	
3.3.2 存储管理方案.....116	
3.3.3 分页存储管理.....118	
3.3.4 分段存储管理.....120	
3.3.5 段页式存储管理.....122	
3.3.6 虚拟存储管理.....122	
3.4 设备管理.....128	
3.4.1 设备管理概述.....128	
3.4.2 I/O 软件.....129	
3.4.3 设备管理采用的相关缓冲技术.....130	
3.4.4 磁盘调度.....132	
3.5 文件管理.....134	
3.5.1 文件与文件系统.....135	
3.5.2 文件的结构和组织.....136	
3.5.3 文件目录.....137	
3.5.4 存取方法和存储空间的管理.....139	
3.5.5 文件的使用.....141	
3.5.6 文件的共享和保护.....141	
3.5.7 系统的安全与可靠性.....143	
3.6 作业管理.....145	
3.6.1 作业与作业控制.....145	
3.6.2 作业调度.....146	
3.6.3 用户界面.....147	
3.7 网络与嵌入式操作系统基础知识.....148	

3.7.1	网络操作系统	148	4.5.3	进度管理	185
3.7.2	嵌入式操作系统	149	4.5.4	软件项目的组织	188
3.8	UNIX 操作系统实例	150	4.5.5	软件质量管理	190
3.8.1	UNIX 操作系统	150	4.5.6	软件配置管理	197
3.8.2	UNIX 文件系统	150	4.6	风险管理	199
3.8.3	UNIX 进程与存储管理	152	4.6.1	软件风险	199
3.8.4	UNIX 设备管理	152	4.6.2	风险识别	200
3.8.5	shell 程序	154	4.6.3	风险预测	201
第 4 章	软件工程基础知识	159	4.6.4	风险评估	202
4.1	软件工程概述	159	4.6.5	风险控制	202
4.1.1	计算机软件	160	4.7	软件度量	203
4.1.2	软件工程基本原理	161	4.7.1	软件度量分类	203
4.1.3	软件生存周期	163	4.7.2	软件复杂性度量	205
4.1.4	软件过程	164	第 5 章	系统开发与运行	207
4.2	软件过程模型	168	5.1	系统分析与设计概述	207
4.2.1	瀑布模型 (Waterfall Model)	168	5.1.1	系统分析概述	207
4.2.2	增量模型 (Incremental Model)	170	5.1.2	系统设计的基本原理	208
4.2.3	演化模型 (Evolutionary Model)	170	5.1.3	系统设计的内容和步骤	211
4.2.4	喷泉模型 (Water Fountain Model)	173	5.1.4	系统总体结构设计	212
4.2.5	基于构件的开发模型 (Component-based Development Model)	173	5.1.5	系统文档	217
4.2.6	形式化方法模型 (Formal Methods Model)	174	5.2	需求分析与需求工程	218
4.3	软件开发方法	174	5.2.1	软件需求	218
4.3.1	结构化方法	174	5.2.2	需求分析原则	219
4.3.2	Jackson 方法	175	5.2.3	需求工程	219
4.3.3	原型方法	175	5.2.4	需求建模	220
4.3.4	面向对象方法	175	5.2.5	需求规约与验证	221
4.3.5	敏捷方法	176	5.2.6	需求管理	222
4.4	软件工具与软件开发环境	177	5.3	结构化分析方法	223
4.4.1	软件工具	177	5.3.1	结构化分析方法概述	223
4.4.2	软件开发环境	179	5.3.2	数据流图	223
4.5	软件项目管理	180	5.3.3	数据字典 (DD)	233
4.5.1	软件项目管理设计的范围	180	5.4	结构化设计方法	235
4.5.2	软件项目估算	182	5.4.1	结构图	235
			5.4.2	结构化设计的步骤	235
			5.4.3	数据流图到软件体系结构的映射	236

5.5 Web 应用系统分析与设计	238	6.5 网络安全	316
5.5.1 WebApp 的特性	238	6.5.1 网络安全概述	316
5.5.2 Web 应用系统分析模型	239	6.5.2 网络的信息安全	318
5.6 用户界面设计	242	6.5.3 防火墙技术	322
5.6.1 用户界面设计的黄金原则	242	6.6 多媒体的基本概念	328
5.6.2 用户界面的分析与设计	244	6.6.1 虚拟现实基本概念	330
5.6.3 用户界面设计问题	245	6.6.2 声音	332
5.7 测试基础知识	247	6.6.3 图形和图像	337
5.7.1 系统测试与调试	247	6.6.4 动画和视频	344
5.7.2 传统软件的测试策略	249	6.7 多媒体网络	351
5.7.3 测试方法	256	6.7.1 超文本与超媒体	352
5.7.4 调试	259	6.7.2 流媒体	353
5.7.5 测试 Web 应用系统	261	6.8 多媒体计算机系统	355
5.8 系统运行和维护知识	262	6.8.1 多媒体计算机硬件系统	355
5.8.1 系统转换	262	6.8.2 多媒体计算机软件系统	356
5.8.2 系统维护概述	264	第 7 章 数据库技术基础	359
5.8.3 系统评价	267	7.1 基本概念	359
第 6 章 网络与多媒体基础知识	269	7.1.1 数据库与数据库管理系统	359
6.1 网络概述	269	7.1.2 DBMS 的功能	360
6.1.1 计算机网络的概念	269	7.1.3 DBMS 的特征及分类	361
6.1.2 计算机网络的分类	272	7.1.4 数据库系统的体系结构	362
6.1.3 网络的拓扑结构	273	7.1.5 数据库的三级模式结构	365
6.1.4 ISO/OSI 网络体系结构	275	7.1.6 大数据	367
6.2 网络互连硬件	278	7.2 数据模型	370
6.2.1 网络的设备	278	7.2.1 数据模型的基本概念	370
6.2.2 网络的传输介质	281	7.2.2 数据模型的三要素	370
6.2.3 组建网络	283	7.2.3 E-R 模型	370
6.3 网络的协议与标准	286	7.2.4 层次模型	376
6.3.1 网络的标准	286	7.2.5 网状模型	378
6.3.2 局域网协议	288	7.2.6 关系模型	380
6.3.3 广域网协议	293	7.2.7 面向对象模型	381
6.3.4 TCP/IP 协议族	296	7.3 关系代数	382
6.4 Internet 及应用	301	7.3.1 关系数据库的基本概念	382
6.4.1 Internet 概述	302	7.3.2 5 种基本的关系代数运算	387
6.4.2 Internet 地址	302	7.3.3 扩展的关系代数运算	389
6.4.3 Internet 服务	310	7.4 关系数据库 SQL 语言简介	397

7.4.1 SQL 数据库体系结构	398	8.4.5 最短路径	471
7.4.2 SQL 的基本组成	398	8.5 查找	473
7.4.3 SQL 数据定义	399	8.5.1 查找的基本概念	473
7.4.4 SQL 数据查询	404	8.5.2 静态查找表的查找方法	474
7.4.5 SQL 数据更新	412	8.5.3 动态查找表	478
7.4.6 SQL 访问控制	413	8.5.4 哈希表	485
7.4.7 嵌入式 SQL	415	8.6 排序	489
7.5 关系数据库的规范化	416	8.6.1 排序的基本概念	489
7.5.1 函数依赖	416	8.6.2 简单排序	490
7.5.2 规范化	417	8.6.3 希尔排序	491
7.5.3 模式分解及分解应具有的特性	419	8.6.4 快速排序	493
7.6 数据库的控制功能	420	8.6.5 堆排序	494
7.6.1 事务管理	420	8.6.6 归并排序	496
7.6.2 数据库的备份与恢复	421	8.6.7 基数排序	497
7.6.3 并发控制	422	8.6.8 内部排序方法小结	498
第8章 数据结构	425	8.6.9 外部排序	499
8.1 线性结构	425	第9章 算法设计与分析	503
8.1.1 线性表	425	9.1 算法设计与分析的基本概念	503
8.1.2 栈和队列	430	9.1.1 算法	503
8.1.3 串	434	9.1.2 算法设计	503
8.2 数组、矩阵和广义表	439	9.1.3 算法分析	504
8.2.1 数组	439	9.1.4 算法的表示	504
8.2.2 矩阵	441	9.2 算法分析基础	504
8.2.3 广义表	442	9.2.1 时间复杂度	504
8.3 树	443	9.2.2 渐进符号	505
8.3.1 树与二叉树的定义	444	9.2.3 递归式	506
8.3.2 二叉树的性质与存储结构	445	9.3 分治法	509
8.3.3 二叉树的遍历	448	9.3.1 递归的概念	509
8.3.4 线索二叉树	451	9.3.2 分治法的基本思想	510
8.3.5 最优二叉树	452	9.3.3 分治法的典型实例	510
8.3.6 树和森林	456	9.4 动态规划法	514
8.4 图	459	9.4.1 动态规划法的基本思想	514
8.4.1 图的定义与存储	459	9.4.2 动态规划法的典型实例	515
8.4.2 图的遍历	463	9.5 贪心法	520
8.4.3 生成树及最小生成树	466	9.5.1 贪心法的基本思想	520
8.4.4 拓扑排序和关键路径	468	9.5.2 贪心法的典型实例	521

9.6 回溯法	524	11.2.1 知识产权基本概念	612
9.6.1 回溯法的算法框架	524	11.2.2 计算机软件著作权	616
9.6.2 回溯法的典型实例	527	11.2.3 计算机软件的商业秘密权	627
9.7 其他算法	532	11.2.4 专利权概述	629
9.7.1 分支限界法	532	11.2.5 企业知识产权的保护	634
9.7.2 概率算法	533	第12章 软件系统分析与设计	637
9.7.3 近似算法	535	12.1 结构化分析与设计	637
第10章 面向对象技术	536	12.1.1 需求说明	639
10.1 面向对象基础	536	12.1.2 结构化分析	639
10.1.1 面向对象的基本概念	536	12.1.3 总体设计	641
10.1.2 面向对象分析	539	12.1.4 详细设计	642
10.1.3 面向对象设计	540	12.2 数据库分析与设计	643
10.1.4 面向对象程序设计	541	12.2.1 数据库设计的步骤	643
10.1.5 面向对象测试	546	12.2.2 需求分析	643
10.2 UML	547	12.2.3 概念结构设计	645
10.2.1 事物	547	12.2.4 逻辑结构设计	648
10.2.2 关系	549	12.2.5 数据库的物理设计	650
10.2.3 UML 中的图	550	12.2.6 数据库的实施与维护	653
10.3 设计模式	561	12.2.7 案例分析	656
10.3.1 设计模式的要素	561	12.3 面向对象分析与设计	661
10.3.2 创建型设计模式	562	12.3.1 面向对象分析与设计的步骤	661
10.3.3 结构型设计模式	567	12.3.2 需求说明	662
10.3.4 行为设计模式	576	12.3.3 建模用例	663
10.3.5 应用举例	590	12.3.4 建模活动	664
第11章 标准化和软件知识产权基础知识	594	12.3.5 设计类图	666
11.1 标准化基础知识	594	12.3.6 建模对象状态	666
11.1.1 标准化的基本概念	594	12.3.7 建模序列图	668
11.1.2 信息技术标准化	602	12.4 算法分析与设计	669
11.1.3 标准化组织	604	12.4.1 C 程序设计语言与实现	670
11.1.4 ISO 9000 标准简介	608	12.4.2 算法设计与实现	683
11.1.5 ISO/IEC 15504 过程评估标准简介	610	12.5 面向对象的程序设计与实现	695
11.2 知识产权基础知识	612	12.5.1 设计与实现方法	695
		12.5.2 设计模式的应用	696

第 1 章 计算机系统知识

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五大部件组成。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处理结果。

1.1.2 中央处理单元

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不同的部件,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。

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。

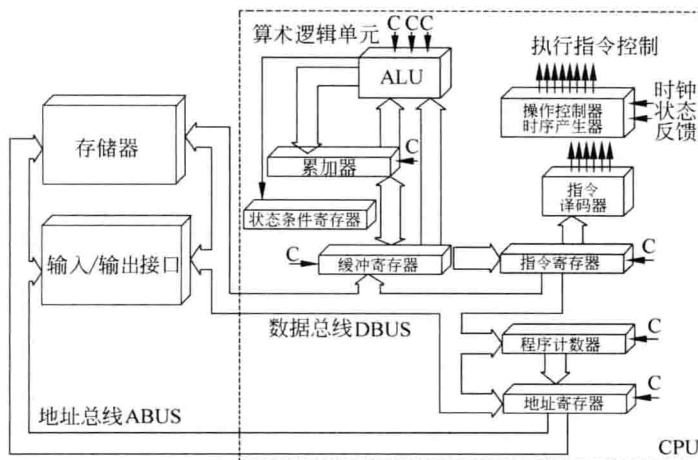


图 1-1 CPU 基本组成结构示意图

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，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，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 是运算器的重要组成部分，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 将自动修改 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)。指令分为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很大不同。AMD 将两个内核做一个 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,因此有人将 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。

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。若约定小数点的位置在机器数的最低数值位之后,则是纯整数;若约定小数点的位置在机器数的最高数值位之前(符号位之后),则是纯小数。

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 $[X]_{\text{原}}$,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:

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

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

【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。

$$\begin{aligned}
 [+1]_{\text{原}} &= 0\ 0000001 & [-1]_{\text{原}} &= 1\ 0000001 \\
 [+127]_{\text{原}} &= 0\ 1111111 & [-127]_{\text{原}} &= 1\ 1111111 \\
 [+45]_{\text{原}} &= 0\ 0101101 & [-45]_{\text{原}} &= 1\ 0101101 \\
 [+0.5]_{\text{原}} &= 0\ \diamond 1000000 & [-0.5]_{\text{原}} &= 1\ \diamond 1000000 \quad (\text{其中, } \diamond \text{ 是小数点的位置})
 \end{aligned}$$

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: $[+0]_{\text{原}}=0\ 0000000$, $[-0]_{\text{原}}=1\ 0000000$ 。

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

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

【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。

$$\begin{aligned}
 [+1]_{\text{反}} &= 0\ 0000001 & [-1]_{\text{反}} &= 1\ 1111110 \\
 [+127]_{\text{反}} &= 0\ 1111111 & [-127]_{\text{反}} &= 1\ 0000000 \\
 [+45]_{\text{反}} &= 0\ 0101101 & [-45]_{\text{反}} &= 1\ 1010010 \\
 [+0.5]_{\text{反}} &= 0\ \diamond 1000000 & [-0.5]_{\text{反}} &= 1\ \diamond 0111111 \quad (\text{其中, } \diamond \text{ 是小数点的位置})
 \end{aligned}$$

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: $[+0]_{\text{反}}=0\ 0000000$, $[-0]_{\text{反}}=1\ 1111111$ 。

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

$$\begin{aligned}
 \text{若 } X \text{ 是纯整数, 则 } [X]_{\text{补}} &= \begin{cases} X & 0 \leq X \leq 2^{n-1} - 1 \\ 2^n + X & -2^{n-1} \leq X \leq 0 \end{cases} \\
 \text{若 } X \text{ 是纯小数, 则 } [X]_{\text{补}} &= \begin{cases} X & 0 \leq X < 1 \\ 2 + X & -1 \leq X < 0 \end{cases}
 \end{aligned}$$

【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。

$$\begin{aligned}
 [+1]_{\text{补}} &= 0\ 0000001 & [-1]_{\text{补}} &= 1\ 1111111 \\
 [+127]_{\text{补}} &= 0\ 1111111 & [-127]_{\text{补}} &= 1\ 0000001 \\
 [+45]_{\text{补}} &= 0\ 0101101 & [-45]_{\text{补}} &= 1\ 1010011 \\
 [+0.5]_{\text{补}} &= 0\ \diamond 1000000 & [-0.5]_{\text{补}} &= 1\ \diamond 1000000 \quad (\text{其中, } \diamond \text{ 是小数点的位置})
 \end{aligned}$$

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尾加1。在补码表示中,0有唯一的编码: $[+0]_{\text{补}}=0\ 0000000$, $[-0]_{\text{补}}=00000000$ 。

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

【例 1.4】 若机器字长 n 等于 8,分别给出 +1, -1, +127, -127, +45, -45, +0.5, -0.5, +0, -0 的移码表示。

$[+1]_{\text{移}}=1\ 0000001$ $[-1]_{\text{移}}=0\ 1111111$
 $[+127]_{\text{移}}=1\ 1111111$ $[-127]_{\text{移}}=0\ 0000001$
 $[+45]_{\text{移}}=1\ 0101101$ $[-45]_{\text{移}}=0\ 1010011$
 $[+0.5]_{\text{移}}=1\ 0100000$ $[-0.5]_{\text{移}}=0\ 0100000$ (其中, \diamond 是小数点的位置)
 $[+0]_{\text{移}}=1\ 0000000$ $[-0]_{\text{移}}=10000000$

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-1} 个数,而其原码和反码只能表示 $2^{n-1}-1$ 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。因此引入浮点数,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, 阶码通常为带符号的纯整数, 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, 因此可以用多种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 $M=0.1 \times \times \times \cdots \times$, 其中, \times 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 $M=1.0 \times \times \times \cdots \times$, 其中, \times 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:

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

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; $(b_0 b_1 b_2 b_3 \cdots b_{p-1})$ 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

续表

参 数	单精度浮点数	双精度浮点数	扩充精度浮点数
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	$10^{-38} \sim 10^{38}$	$10^{-308} \sim 10^{308}$	$10^{-4932} \sim 10^{4932}$

在 IEEE 754 标准中, 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 $1.\times\times\cdots\times$ 。

【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: $10110000.0001 = 1 \diamond 01100000001 \times 2^7$ 。这就保证了使 b_0 为 1, 而且小数点应当在 \diamond 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 10001110。

最后, 可得到 $(176.0625)_{10}$ 的单精度浮点数表示形式:

0 10001110 01100000001000000000000

(4) 浮点数的运算。设有浮点数 $X = M \times 2^j$, $Y = N \times 2^i$, 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,各个部件之间要进行数据交换,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,它也是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确定位置上插入 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:

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

海明码的编码规则如下。

设 k 个校验位为 P_k, P_{k-1}, \dots, P_1 , n 个数据位为 $D_{n-1}, D_{n-2}, \dots, D_1, D_0$, 对应的海明码为 $H_{n+k}, H_{n+k-1}, \dots, H_1$, 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 $H_{12}, H_{11}, \dots, H_3, H_2, H_1$, 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	
$H_3 (D_0)$	$3 = 1+2$	P_1, P_2	
$H_4 (P_3)$	4	P_3	P_2 校验: $P_2, D_0, D_2, D_3, D_5, D_6$ 即 $P_2 = D_0 \oplus D_2 \oplus D_3 \oplus D_5 \oplus D_6$
$H_5 (D_1)$	$5 = 1+4$	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	P_3 校验: P_3, D_1, D_2, D_3, D_7 即 $P_3 = D_1 \oplus D_2 \oplus D_3 \oplus D_7$
$H_8 (P_4)$	8	P_4	
$H_9 (D_4)$	$9 = 1+8$	P_1, P_4	
$H_{10} (D_5)$	$10 = 2+8$	P_2, P_4	P_4 校验: P_4, D_4, D_5, D_6, D_7 即 $P_4 = D_4 \oplus D_5 \oplus D_6 \oplus D_7$
$H_{11} (D_6)$	$11 = 1+2+8$	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：D₇D₆D₅D₄D₃D₂D₁D₀=01101001，根据公式

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

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

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

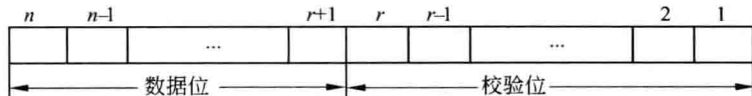
$$P_4 = D_4 \oplus D_5 \oplus D_6 \oplus D_7 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

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，校验码位数越长，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:体系结构是由结构、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 P_m 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

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)。

通用寄存器机(General-Purpose Register Machines, GPR机)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 是指令集发展的两个途径。

(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。

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。

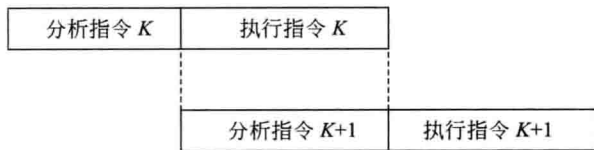


图 1-2 一次重叠处理

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。

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。

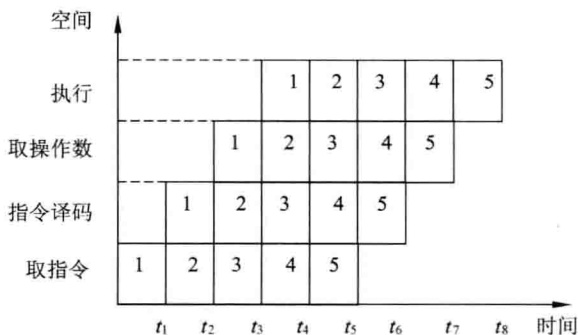


图 1-3 流水处理的时空图

(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 and 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

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

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,通过互连网络 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 为互连网络。

分布式存储器的 SIMD 处理机如图 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 为互连网络。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。

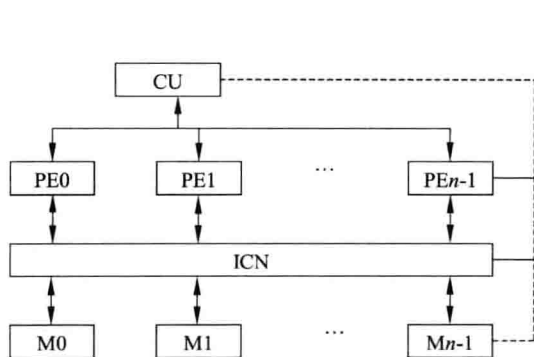


图 1-4 具有共享存储器的 SIMD 结构

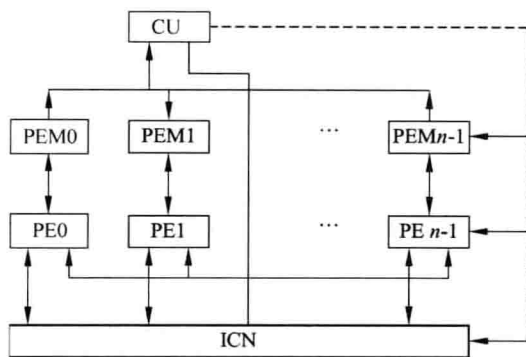


图 1-5 具有分布存储器的 SIMD 结构

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。

网格计算是伴随着因特网技术迅速发展起来的、专门用于复杂科学计算的新型计算模式。这种计算模式利用因特网把分散在不同地理位置的计算机组织成一个“虚拟的超级计算机”,其中每台参与计算的计算机就是一个“结点”,整个计算是由成千上万个“结点”组成的“一张网格”,所以这种计算方式叫网格计算。网格类似于集群,但是与集群相比,结点更加分散,结点的组织方式也更加灵活,相应地,平均每个结点所能提供的计算能力一般要比集群低,并且结点间的通信效率也较低。

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。

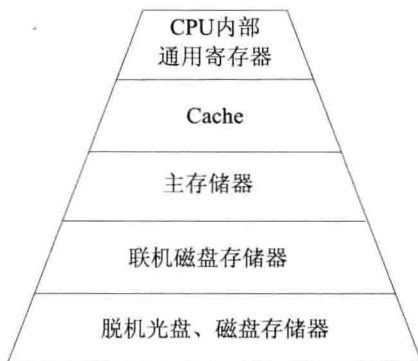


图 1-6 存储系统的层次结构

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。

(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)。

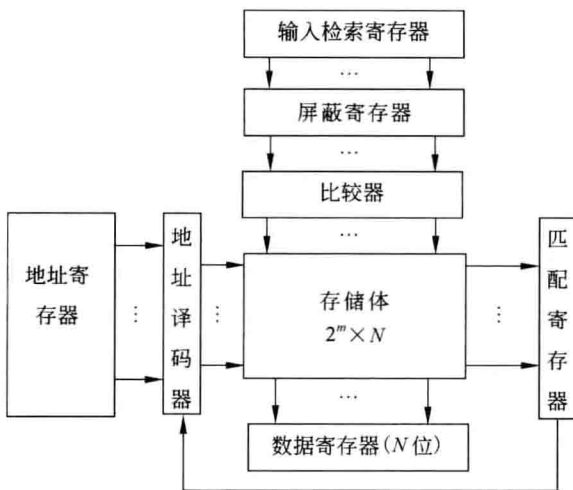


图 1-7 相联存储器的结构框图

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。

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：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

高速缓存的组成如图 1-8 所示。Cache 由两部分组成：控制部分和存储器部分。

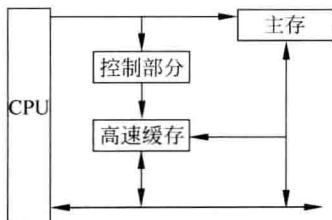


图 1-8 高速缓存的构成框图

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。

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 中块的对应关系是固定的, 如图 1-9 所示。

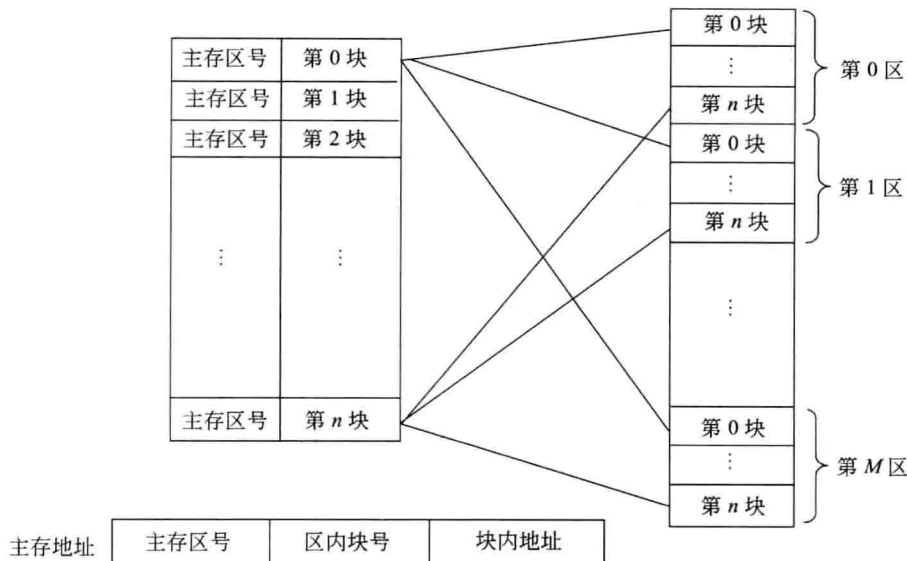


图 1-9 直接映像示意图

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 中的主存区号相同, 则表明访问 Cache 命中。一旦命中, 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 1-10 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。

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。

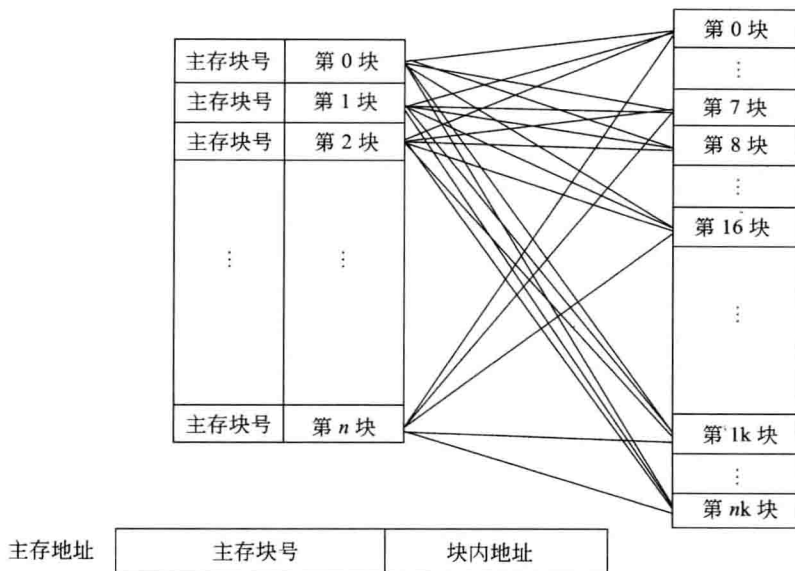


图 1-10 全相联映像示意图

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 容量的关系是: 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。

5) 多级 Cache

在多级 Cache 的计算机中, Cache 分为一级 (L1 Cache)、二级 (L2 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。目前, CPU 内的 Cache 通常为二级结构。通常要求 L1 Cache 的速度足够快, 以赶上 CPU 的主频。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, 以具有足够高的命中率。

5. 虚拟存储器

虚拟存储 (virtual memory) 技术是把很大的程序 (数据) 分成许多较小的块, 全部存储在

辅存中。运行时,把要用到的程序(数据)块先调入主存,并且把马上就要用到的程序块从主存调入高速缓存。这样,一边运行程序,一边进行所需程序(数据)块的调进/调出。只要及时供应所需处理的程序与数据,程序就能顺利而高速地运行下去。因此,对于应用程序员来说就好像有一个比实际主存大得多且可以放下整个程序的虚拟主存空间。当辅存中的程序块调入主存时,必须使程序在主存中定位,该工作由系统自动完成,应用程序员不用考虑如何把程序地址映像和变换成实际主存的物理地址,因此,虚存技术对于应用程序员来说是透明的。

虚拟存储器管理方式分为如下几种。

(1) 页式虚拟存储器。以页为信息传送单位的虚拟存储器。通常一页为几百字节到几千字节。为实现页式管理,需建立虚页与实页间的关系表,称为页表。在页表及变换软件的控制下,可将程序的虚拟地址变换为主存的实地址。页式管理的优点是页表硬件少,查表速度快,主存碎片少。其缺点是分页无逻辑意义,不利于存储保护。

(2) 段式虚拟存储器。以程序的逻辑结构形成的段(如某一独立程序模块、子程序等)作为主存分配依据的一种段式虚拟存储器的管理方法。为实现段式管理,需建立段表。在段地址变换机构及软件的控制下,可将程序的虚拟地址变换为主存的实地址。段式管理的优点是段的界限分明;支持程序的模块化设计;易于对程序段的编译、修改和保护;便于多道程序的共享。主要缺点是因为段的长度不一,主存利用率不高,产生大量内存碎片,造成浪费;段表庞大,查表速度慢。

(3) 段页式虚拟存储器。它是页式虚拟存储器和段式虚拟存储器相结合的一种管理方式。在这种虚拟存储器中,程序按逻辑结构分段,每一段再分成若干大小固定的页。程序的调入/调出是按页进行的,而程序又可按段实现保护。因此,这种管理方式兼有前两者的优点,只是地址变换速度比较慢。

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)及光盘存储器构成。下面介绍两种常用的外存储器。

1) 磁盘存储器

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片盘片,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 \times (磁道数/面) \times 内圆周长 \times 最大位密度

格式化容量 = 面数 \times (磁道数/面) \times (扇区数/道) \times (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输率可超过 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。

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可以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, 它能够把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 1-11 所示。

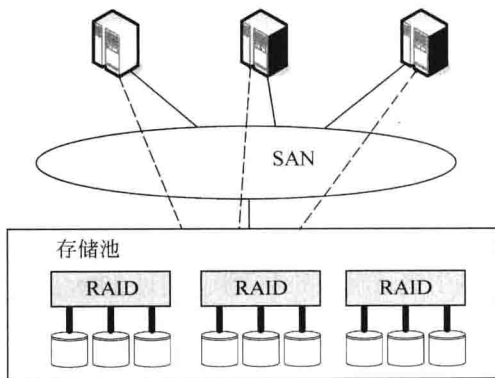


图 1-11 SAN 的结构

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,在编程或读程序中很易使用和辨认。

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,应在不影响外设工作的情况下, CPU 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 1-12 所示。中断源的优先级由 INTC 来控制。

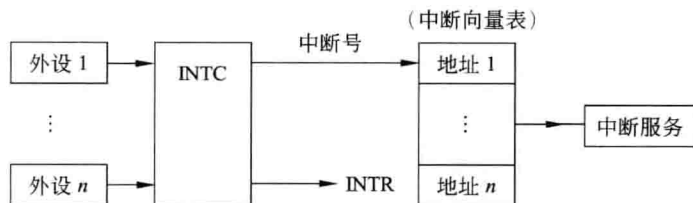


图 1-12 中断向量表法

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 1-13 所示。

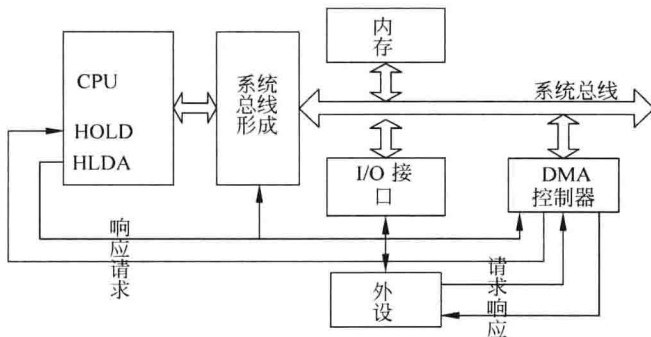


图 1-13 DMA 过程示意图

- (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)

上面所描述的输入/输出方式适合于外设不多、速度不很高的小型或微型机中。在大型计算机中, 外设很多, 要求计算机的速度很高, 采用程序传送、查询、中断或 DMA 都会因输入/

输出而造成过大的开销,影响计算机的整体性能,为此提出采用输入/输出处理机。

输入/输出处理机是一个专用处理机,接在主计算机上,主机的输入/输出操作由它来完成。它根据主机的 I/O 命令完成对外设数据的输入/输出。既然输入/输出由 IOP 来完成了,主机的工作效率必然提高了。

输入/输出处理机的数据传送方式有如下 3 种:字节多路方式、选择传送方式和数组多路方式。

1.2.4 总线结构

计算机系统的硬部件以总线方式进行互连,以便于部件和设备的扩充,并制定统一的总线标准。随着微电子技术和计算机技术的发展,总线技术也在不断地发展和完善,因此,计算机总线技术种类繁多、各具特色。

1. 总线的定义与分类

广义地讲,任何连接两个以上电子元器件的导线都可以称为总线。通常分为如下三类总线。

(1) 内部总线。用于芯片一级的互连,分为芯片内总线和元件级总线。芯片内总线用于集成电路芯片内部各部分的连接,元件级总线用于一块电路板内各元器件的连接。

(2) 系统总线。用于插件板一级的互连,用于构成计算机各组成部分(CPU、内存和接口等)的连接。

(3) 外部总线。又称通信总线,用于设备一级的互连,通过该总线和其他设备进行信息与数据交换。

2. 系统总线

系统总线有时也称为内总线,其性能直接影响到计算机的性能。常见的内总线标准如下。

(1) ISA 总线。ISA 是工业标准总线,它可以兼容更早的 PC 总线,在 PC 总线 62 个插座信号的基础上再扩充一个具有 36 个信号的插座构成 ISA 总线。

ISA 总线主要包括 24 条地址线,16 条数据线,以及控制总线(内存读/写、接口读/写、中断请求、中断响应、DMA 请求和 DMA 响应等), $\pm 5V$ 、 $\pm 12V$ 电源和地线等。

(2) EISA 总线。该总线是在 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。该总线利用总线插座与 ISA 总线相兼容,插板插在上层为 ISA 总线信号,将插板插在下层便是 EISA 总线。

(3) PCI 总线。PCI 总线是目前微型机上广泛采用的内总线。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, 具有很高的传输速率。PCI 总线的工作与处理器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。

3. 外部总线

外总线的标准有七八十种之多, 下面简要介绍几种。

(1) RS-232C。RS-232C 是一种串行外总线, 其主要特点是所需传输线比较少, 最少只需 3 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 $\leq -3V$ 为逻辑 1, 电平 $\geq +3V$ 为逻辑 0, 具有较好的抗干扰性。

(2) SCSI 总线。小型计算机系统接口(SCSI)是一种并行外总线, 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。目前的传输速率已高达 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)。

(3) 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 总线最大的优点在于它支持即插即用, 并支持热插拔。

(4) 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。相关资料介绍, IEEE-1394 总线上可接 63 个设备。IEEE-1394 的传送速率从 400Mb/s、800Mb/s、1600Mb/s 直到 3.2Gb/s。

(5) IEEE-488。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。其最大传输距离为 20m, 信号传输速度一般为 500Kbps, 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

计算机安全是指计算机资产的安全, 是要保证这些计算机资产不受自然和人为的有害因素的威胁和危害。计算机资产由系统资源和信息资源两大部分组成。系统资源包括硬件、软件、

配套设备设施、有关文件资料,还可以包括有关的服务系统和业务工作人员。信息资源包括计算机系统中存储、处理和传输的大量各种各样的信息。

1. 信息安全的基本要素

信息安全的5个基本要素为机密性、完整性、可用性、可控性和可审查性。

- (1) 机密性。确保信息不暴露给未授权的实体或进程。
- (2) 完整性。只有得到允许的人才能修改数据,并能够判别出数据是否已被篡改。
- (3) 可用性。得到授权的实体在需要时可访问数据。
- (4) 可控性。可以控制授权范围内的信息流向及行为方式。
- (5) 可审查性。对出现的安全问题提供调查的依据和手段。

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。

2. 计算机的安全等级

计算机系统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、荷四国防务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

3. 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
媒体清理	信息被从废弃的或打印过的媒体中获得
物理入侵	一个入侵者通过绕过物理控制而获得对系统的访问
资源耗尽	某一资源(如访问端口)被故意超负荷地使用, 导致其他用户的服务被中断

4. 影响数据安全的因素

影响数据安全的因素有内部和外部两类。

(1) 内部因素。可采用多种技术对数据加密;制定数据安全规划;建立安全存储体系,包括容量、容错数据保护和数据备份等;建立事故应急计划和容灾措施;重视安全管理,制定数据安全规范。

(2) 外部因素。可将数据分成不同的密级,规定外部使用人员的权限;设置身份认证、密码、设置口令、设置指纹和声纹笔迹等多种认证;设置防火墙,为计算机建立一道屏障,防止外部入侵破坏数据;建立入侵检测、审计和追踪,对计算机进行防卫。同时,也包括计算机物理环境的保障、防辐射、防水和防火等外部防灾措施。

1.3.2 加密技术和认证技术

1. 加密技术

加密技术是最常用的安全保密手段,数据加密技术的关键在于加密/解密算法和密钥管理。数据加密的基本过程就是对原来为明文的文件或数据按某种加密算法进行处理,使其成为不可读的一段代码,通常称为“密文”。“密文”只能在输入相应的密钥之后才能显示出原来的内容,通过这样的途径使数据不被窃取。

数据加密和数据解密是一对逆过程。数据加密是用加密算法 E 和加密密钥 K_1 将明文 P 变换成密文 C , 记为

$$C = E_{K_1}(P)$$

数据解密是数据加密的逆过程,解密算法 D 和解密密钥 K_2 将密文 C 变换成明文 P , 记为

$$P = D_{K_2}(C)$$

在安全保密中,可通过适当的密钥加密技术和管理机制来保证网络信息的通信安全。密钥加密技术的密码体制分为对称密钥体制和非对称密钥体制两种。相应地,对数据加密的技术分为两类,即对称加密(私人密钥加密)和非对称加密(公开密钥加密)。

1) 对称加密技术

对称加密采用了对称密码编码技术,其特点是文件加密和解密使用相同的密钥,这种方法在密码学中称为对称加密算法。

常用的对称加密算法有如下几种。

(1) 数据加密标准(Digital Encryption Standard, DES)算法。DES 主要采用替换和移位的方法加密。它用 56 位密钥对 64 位二进制数据块进行加密,每次加密可对 64 位的输入数据进

行 16 轮编码, 经一系列替换和移位后, 输入的 64 位原始数据转换成完全不同的 64 位输出数据。DES 算法运算速度快, 密钥生产容易, 适合于在当前大多数计算机上用软件方法实现, 同时也适合于在专用芯片上实现。

(2) 三重 DES (3DES, 或称 TDEA)。在 DES 的基础上采用三重 DES, 即用两个 56 位的密钥 K_1 和 K_2 , 发送方用 K_1 加密, K_2 解密, 再使用 K_1 加密。接收方则使用 K_1 解密, K_2 加密, 再使用 K_1 解密, 其效果相当于将密钥长度加倍。

(3) RC-5 (Rivest Cipher 5)。RC-5 是由 Ron Rivest (公钥算法的创始人之一) 在 1994 年开发出来的。RC-5 是在 RCF2040 中定义的, RSA 数据安全公司的很多产品都使用了 RC-5。

(4) 国际数据加密算法 (International Data Encryption Adleman, IDEA)。IDEA 是在 DES 算法的基础上发展起来的, 类似于三重 DES。IDEA 的密钥为 128 位, 这么长的密钥在今后若干年内应该是安全的。类似于 DES, IDEA 算法也是一种数据块加密算法, 它设计了一系列加密轮次, 每轮加密都使用从完整的加密密钥中生成的一个子密钥。IDEA 加密标准由 PGP (Pretty Good Privacy) 系统使用。

(5) 高级加密标准 (Advanced Encryption Standard, AES) 算法。AES 算法基于排列和置换运算。排列是对数据重新进行安排, 置换是将一个数据单元替换为另一个。AES 使用几种不同的方法来执行排列和置换运算。

AES 是一个迭代的、对称密钥分组的密码, 它可以使用 128、192 和 256 位密钥, 并且用 128 位 (16 字节) 分组加密和解密数据。

2) 非对称加密技术

与对称加密算法不同, 非对称加密算法需要两个密钥: 公开密钥 (publickey) 和私有密钥 (privatekey)。公开密钥与私有密钥是一对, 如果用公开密钥对数据进行加密, 只有用对应的私有密钥才能解密; 如果用私有密钥对数据进行加密, 那么只有用对应的公开密钥才能解密。因为加密和解密使用的是两个不同的密钥, 所以这种算法称为非对称加密算法。

非对称加密有两个不同的体制, 如图 1-14 所示。

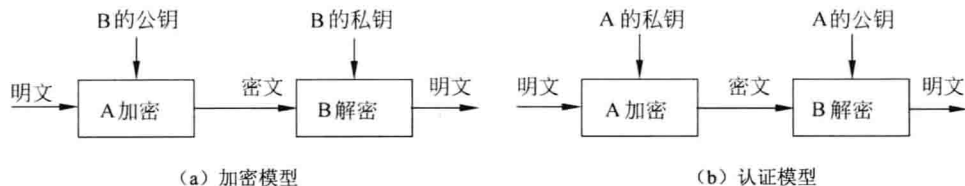
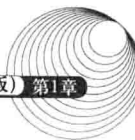


图 1-14 非对称加密体制模型

非对称加密算法实现机密信息交换的基本过程是: 甲方生成一对密钥并将其中的一把作为



公用密钥向其他方公开；得到该公用密钥的乙方使用该密钥对机密信息进行加密后再发送给甲方；甲方再用自己保存的另一把专用密钥对加密后的信息进行解密。甲方只能用其专用密钥解密由其公用密钥加密后的任何信息。

非对称加密算法的保密性比较好，它消除了最终用户交换密钥的需要，但加密和解密花费的时间长、速度慢，不适用于对文件加密，而只适用于对少量数据进行加密。

RSA (Rivest, Shamir and Adleman) 算法是一种公钥加密算法，它按照下面的要求选择公钥和密钥。

(1) 选择两个大素数 p 和 q (大于 10^{100})。

(2) 令 $n=p \times q$ 和 $z=(p-1) \times (q-1)$ 。

(3) 选择 d 与 z 互质。

(4) 选择 e ，使 $e \times d \equiv 1 \pmod{z}$ 。

明文 P 被分成 k 位的块， k 是满足 $2^k < n$ 的最大整数，于是有 $0 \leq P < n$ 。加密时计算

$$C = P^e \pmod{n}$$

这样公钥为 (e, n) 。解密时计算

$$P = C^d \pmod{n}$$

即私钥为 (d, n) 。

例如，设 $p=2$ ， $q=11$ ， $n=33$ ， $z=20$ ， $d=7$ ， $e=3$ ， $C=P^3 \pmod{33}$ ， $P=C^7 \pmod{33}$ ，则有

$$C=2^3 \pmod{33}=8 \pmod{33}=8$$

$$P=8^7 \pmod{33}=2097152 \pmod{33}=2$$

RSA 算法的安全性是基于大素数分解的困难性。攻击者可以分解已知的 n ，得到 p 和 q ，然后可得到 z ，最后用 Euclid 算法，由 e 和 z 得到 d 。但是要分解 200 位的数，需要 40 亿年；分解 500 位的数，则需要 10^{25} 年。

3) 密钥管理

密钥是有生命周期的，它包括密钥和证书的有效时间，以及已撤销密钥和证书的维护时间等。密钥既然要求保密，这就涉及密钥的管理问题，管理不好，密钥同样可能被无意识地泄露，并不是有了密钥就可以高枕无忧，任何保密也只是相对的，是有时效的。密钥管理主要是指密钥对的安全管理，包括密钥产生、密钥备份、密钥恢复和密钥更新等。

(1) 密钥产生。密钥对的产生是证书申请过程中重要的一步，其中产生的私钥由用户保留，公钥和其他信息则交给 CA 中心进行签名，从而产生证书。根据证书类型和应用的不同，密钥对的产生也有不同的形式和方法。对于普通证书和测试证书，一般由浏览器或固定的终端应用来产生，这样产生的密钥强度较小，不适合应用于比较重要的安全网络交易。而对于比较重要的证书，如商家证书和服务器证书等，密钥对一般由专用应用程序或 CA 中心直接产生，这样产生的密钥强度大，适合于重要的应用场合。

另外,根据密钥的应用不同,也可能会有不同的产生方式。例如,签名密钥可能在客户端或 RA(注册管理机构)中心产生,而加密密钥则需要 CA 中心直接产生。

(2) 密钥备份和恢复。在一个 PKI(Public Key Infrastructure, 公开密钥体系)系统中,维护密钥对的备份至关重要,如果没有这种措施,当密钥丢失后,将意味着加密数据的完全丢失,对于一些重要数据,这将是灾难性的。所以,密钥的备份和恢复也是 PKI 密钥管理中重要的一环。换句话说,即使密钥丢失,使用 PKI 的企业和组织必须仍能够得到确认,受密钥加密保护的重要信息也必须能够恢复。当然,不能让一个独立的个人完全控制最重要的主密钥,否则将引起严重后果。

企业级的 PKI 产品至少应该支持用于加密的安全密钥的存储、备份和恢复。密钥一般用口令进行保护,而口令丢失则是管理员最常见的安全疏漏之一。所以,PKI 产品应该能够备份密钥,即使口令丢失,它也能够让用户在一定条件下恢复该密钥,并设置新的口令。

(3) 密钥更新。如果用户可以一次又一次地使用同样的密钥与别人交换信息,那么密钥也同其他任何密码一样存在着一定的安全性,虽然说用户的私钥是不对外公开的,但是也很难保证私钥长期的保密性,很难保证长期以来不被泄露。如果某人偶然地知道了用户的密钥,那么用户曾经和另一个人交换的每一条消息都不再是保密的了。另外,使用一个特定密钥加密的信息越多,提供给窃听者的材料也就越多,从某种意义上讲也就越不安全了。

对每一个由 CA 颁发的证书都会有有效期,密钥对生命周期的长短由签发证书的 CA 中心来确定,各 CA 系统的证书的有效期限有所不同,一般为 2~3 年。当用户的私钥被泄漏或证书的有效快到时,用户应该更新私钥。这时用户可以废除证书,产生新的密钥对,申请新的证书。

(4) 多密钥的管理。假设在某机构中有 100 个人,如果他们任意两人之间可以进行秘密对话,那么总共需要多少密钥呢?每个人需要知道多少密钥呢?如果任何两个人之间要不同的密钥,则总共需要 4950 个密钥,而且每个人应记住 99 个密钥。如果机构的人数是 1000、10000 或更多,这种办法显然就过于愚蠢了,管理密钥将是一件非常困难的事情。为此需要研究并开发用于创建和分发密钥的加密安全的方法。

Kerberos 提供了一种解决这个问题的较好方案,它是由 MIT 发明的,使保密密钥的管理和分发变得十分容易,但这种方法本身还存在一定的缺点。为了能在因特网上提供一个实用的解决方案,Kerberos 建立了一个安全的、可信任的密钥分发中心(Key Distribution Center, KDC),每个用户只要知道一个和 KDC 进行会话的密钥就可以了,而不需要知道成百上千个不同的密钥。

2. 认证技术

认证技术主要解决网络通信过程中通信双方的身份认可。认证的过程涉及加密和密钥交换。通常,加密可使用对称加密、不对称加密及两种加密方法的混合方法。认证方一般有账户名/口令认证、使用摘要算法认证和基于 PKI 的认证。

一个有效的 PKI 系统必须是安全的和透明的,用户在获得加密和数字签名服务时不需要详细地了解 PKI 的内部运作机制。在一个典型、完整和有效的 PKI 系统中,除了具有证书的创建和发布,特别是证书的撤销功能外,一个可用的 PKI 产品还必须提供相应的密钥管理服务,包括密钥的备份、恢复和更新等。没有一个好的密钥管理系统,将极大地影响一个 PKI 系统的规模、可伸缩性和在协同网络中的运行成本。在一个企业中,PKI 系统必须有能力为一个用户管理多对密钥和证书;能够提供安全策略编辑和管理工具,如密钥周期和密钥用途等。

PKI 是一种遵循既定标准的密钥管理平台,能够为所有网络应用提供加密和数字签名等密码服务及所必需的密钥和证书管理体系。简单来说,PKI 就是利用公钥理论和技术建立的提供安全服务的基础设施。PKI 技术是信息安全技术的核心,也是电子商务的关键和基础技术。PKI 的基础技术包括加密、数字签名、数据完整性机制、数字信封和双重数字签名等。完整的 PKI 系统必须具有权威认证机构(CA)、数字证书库、密钥备份及恢复系统、证书作废系统、应用接口(Application Programming Interface, API)等基本构成部分。

(1) 认证机构。即数字证书的申请及签发机关,CA 必须具备权威性的特征。

(2) 数字证书库。用于存储已签发的数字证书及公钥,用户可由此获得所需的其他用户的证书及公钥。

(3) 密钥备份及恢复系统。如果用户丢失了用于解密数据的密钥,则数据将无法被解密,这将造成合法数据丢失。为了避免这种情况,PKI 提供了备份与恢复密钥的机制。但须注意,密钥的备份与恢复必须由可信的机构来完成。并且,密钥备份与恢复只能针对解密密钥,签名私钥为确保其唯一性而不能够作备份。

(4) 证书作废系统。证书作废处理系统是 PKI 的一个必备组件。与日常生活中的各种身份证件一样,证书在有效期以内也可能需要作废,原因可能是密钥介质丢失或用户身份变更等。为实现这一点,PKI 必须提供作废证书的一系列机制。

(5) 应用接口。PKI 的价值在于使用户能够方便地使用加密、数字签名等安全服务,因此一个完整的 PKI 必须提供良好的应用接口系统,使得各种各样的应用能够以安全、一致、可信的方式与 PKI 交互,确保安全网络环境的完整性和易用性。

PKI 采用证书进行公钥管理,通过第三方的可信任机构(认证中心,即 CA)把用户的公钥和用户的其他标识信息捆绑在一起,其中包括用户名和电子邮件地址等信息,以在 Internet

上验证用户的身份。PKI 把公钥密码和对称密码结合起来,在 Internet 上实现密钥的自动管理,保证网上数据的安全传输。

因此,从大的方面来说,所有提供公钥加密和数字签名服务的系统都可归结为 PKI 系统的一部分,PKI 的主要目的是通过自动管理密钥和证书为用户建立起一个安全的网络运行环境,使用户可以在多种应用环境下方便地使用加密和数字签名技术,从而保证网上数据的机密性、完整性和有效性。数据的机密性是指数据在传输过程中不能被非授权者偷看;数据的完整性是指数据在传输过程中不能被非法篡改;数据的有效性是指数据不能被否认。

PKI 发展的一个重要方面就是标准化问题,它也是建立互操作性的基础。目前,PKI 标准化主要有两个方面:一是 RSA 公司的公钥加密标准(Public Key Cryptography Standards, PKCS),它定义了许多基本 PKI 部件,包括数字签名和证书请求格式等;二是由 Internet 工程任务组(Internet Engineering Task Force, IETF)和 PKI 工作组(Public Key Infrastructure Working Group, PKIX)所定义的一组具有互操作性的公钥基础设施协议。在今后很长的一段时间内,PKCS 和 PKIX 将会并存,大部分的 PKI 产品为保持兼容性,也将会对这两种标准进行支持。

1) Hash 函数与信息摘要(Message Digest)

Hash(哈希)函数提供了这样一种计算过程:输入一个长度不固定的字符串,返回一串固定长度的字符串,又称 Hash 值。单向 Hash 函数用于产生信息摘要。Hash 函数主要可以解决以下两个问题:在某一特定的时间内,无法查找经 Hash 操作后生成特定 Hash 值的原报文;也无法查找两个经 Hash 操作后生成相同 Hash 值的不同报文。这样,在数字签名中就可以解决验证签名和用户身份验证、不可抵赖性的问题。

信息摘要简要地描述了一份较长的信息或文件,它可以被看作一份长文件的“数字指纹”。信息摘要用于创建数字签名,对于特定的文件而言,信息摘要是唯一的。信息摘要可以被公开,它不会透露相应文件的任何内容。MD2、MD4 和 MD5(MD 表示信息摘要)是由 Ron Rivest 设计的专门用于加密处理的并被广泛使用的 Hash 函数,它们产生一种 128 位信息摘要,除彻底地搜寻外,没有更快的方法对其加以攻击,而其搜索时间一般需要 1025 年之久。

2) 数字签名

数字签名主要经过以下几个过程。

- (1) 信息发送者使用一个单向散列函数(Hash 函数)对信息生成信息摘要。
- (2) 信息发送者使用自己的私钥签名信息摘要。
- (3) 信息发送者把信息本身和已签名的信息摘要一起发送出去。
- (4) 信息接收者通过使用与信息发送者使用的同一个单向散列函数(Hash 函数)对接收的信息本身生成新的信息摘要,再使用信息发送者的公钥对信息摘要进行验证,以确认信息发送者的身份和信息是否被修改过。

数字加密主要经过以下几个过程。

(1) 当信息发送者需要发送信息时, 首先生成一个对称密钥, 用该对称密钥加密要发送的报文。

(2) 信息发送者用信息接收者的公钥加密上述对称密钥。

(3) 信息发送者将第(1)步和第(2)步的结果结合在一起传给信息接收者, 称为数字信封。

(4) 信息接收者使用自己的私钥解密被加密的对称密钥, 再用此对称密钥解密被发送方加密的密文, 得到真正的原文。

数字签名和数字加密的过程虽然都使用公开密钥体系, 但实现的过程正好相反, 使用的密钥对也不同。数字签名使用的是发送方的密钥对, 发送方用自己的私有密钥进行加密, 接收方用发送方的公开密钥进行解密, 这是一个一对多的关系, 任何拥有发送方公开密钥的人都可以验证数字签名的正确性。数字加密则使用的是接收方的密钥对, 这是多对一的关系, 任何知道接收方公开密钥的人都可以向接收方发送加密信息, 只有唯一拥有接收方私有密钥的人才能对信息解密。另外, 数字签名只采用了非对称密钥加密算法, 它能保证发送信息的完整性、身份认证和不可否认性; 而数字加密采用了对称密钥加密算法和非对称密钥加密算法相结合的方法, 它能保证发送信息的保密性。

3) SSL 协议

SSL (Secure Sockets Layer, 安全套接层) 协议最初是由 Netscape Communication 公司设计开发的, 主要用于提高应用程序之间数据的安全系数。SSL 协议的整个概念可以被总结为: 一个保证任何安装了安全套接字的客户端和服务端间事务安全的协议, 它涉及所有 TC/IP 应用程序。SSL 协议主要提供如下三方面的服务。

(1) 用户和服务器的合法性认证。认证用户和服务器的合法性, 使得它们能够确信数据将被发送到正确的客户端和服务端上。客户端和服务端都有各自的识别号, 这些识别号由公开密钥进行编号, 为了验证用户是否合法, 安全套接层协议要求在握手交换数据时进行数字认证, 以此来确保用户的合法性。

(2) 加密数据以隐藏被传送的数据。安全套接层协议所采用的加密技术既有对称密钥技术, 也有公开密钥技术。在客户端与服务端进行数据交换之前, 交换 SSL 初始握手信息, 在 SSL 握手信息中采用了各种加密技术对其加密, 以保证其机密性和数据的完整性, 并且用数字证书进行鉴别, 这样就可以防止非法用户进行破译。

(3) 保护数据的完整性。安全套接层协议采用 Hash 函数和机密共享的方法来提供信息的完整性服务, 建立客户端与服务端之间的安全通道, 使所有经过安全套接层协议处理的业务在传输过程中能全部完整、准确无误地到达目的地。

安全套接层协议是一个保证计算机通信安全的协议, 对通信对话过程进行安全保护, 其实

现过程主要经过如下几个阶段。

- (1) 接通阶段。客户端通过网络向服务器打招呼, 服务器回应。
- (2) 密码交换阶段。客户端与服务器之间交换双方认可的密码, 一般选用 RSA 密码算法, 也有的选用 Diffie-Hellman 和 Fortezza-KEA 密码算法。
- (3) 会谈密码阶段。客户端与服务器间产生彼此交谈的会谈密码。
- (4) 检验阶段。客户端检验服务器取得的密码。
- (5) 客户认证阶段。服务器验证客户端的可信度。
- (6) 结束阶段。客户端与服务器之间相互交换结束的信息。

当上述动作完成之后, 两者间的资料传送就会加密, 另外一方收到资料后, 再将编码资料还原。即使盗窃者在网络上取得编码后的资料, 如果没有原先编制的密码算法, 也不能获得可读的有用资料。

发送时, 信息用对称密钥加密, 对称密钥用非对称算法加密, 再把两个捆绑在一起传送过去。接收的过程与发送正好相反, 先打开有对称密钥的加密包, 再用对称密钥解密。

在电子商务交易过程中, 由于有银行参与, 按照 SSL 协议, 客户的购买信息首先发往商家, 商家再将信息转发银行, 银行验证客户信息的合法性后, 通知商家付款成功, 商家再通知客户购买成功, 并将商品寄送客户。

4) 数字时间戳技术

数字时间戳是数字签名技术的一种变种应用。在电子商务交易文件中, 时间是十分重要的信息。在书面合同中, 文件签署的日期和签名一样均是十分重要的防止文件被伪造和篡改的关键性内容。数字时间戳服务 (Digital Time Stamp Service, DTS) 是网上电子商务安全服务项目之一, 能提供电子文件的日期和时间信息的安全保护。

时间戳是一个经加密后形成的凭证文档, 包括如下 3 个部分。

- (1) 需加时间戳的文件的摘要 (digest)。
- (2) DTS 收到文件的日期和时间。
- (3) DTS 的数字签名。

一般来说, 时间戳产生的过程为: 用户首先将需要加时间戳的文件用 Hash 编码加密形成摘要, 然后将该摘要发送到 DTS, DTS 在加入了收到文件摘要的日期和时间信息后再对该文件加密 (数字签名), 然后送回用户。

书面签署文件的时间是由签署人自己写上的, 而数字时间戳则是由认证单位 DTS 来加入的, 以 DTS 收到文件的时间为依据。

1.3.3 计算机可靠性

1. 计算机可靠性概述

计算机系统的硬件故障通常是由元器件的失效引起的。对元器件进行寿命试验并根据实际资料统计得知,元器件的可靠性可分成3个阶段,在开始阶段,元器件的工作处于不稳定期,失效率较高;在第二阶段,元器件进入正常工作期,失效率最低,基本保持常数;在第三阶段,元器件开始老化,失效率又重新提高,这就是所谓的“浴盆曲线”。因此,应保证在计算机中使用的元器件处于第二阶段。在第一阶段应对元器件进行老化筛选,而到了第3个阶段,则淘汰该计算机。

计算机系统的可靠性是指从它开始运行($t=0$)到某时刻 t 这段时间内能正常运行的概率,用 $R(t)$ 表示。所谓失效率,是指单位时间内失效的元件数与元件总数的比例,用 λ 表示,当 λ 为常数时,可靠性与失效率的关系为

$$R(t) = e^{-\lambda t}$$

典型的失效率与时间的关系曲线如图 1-15 所示。

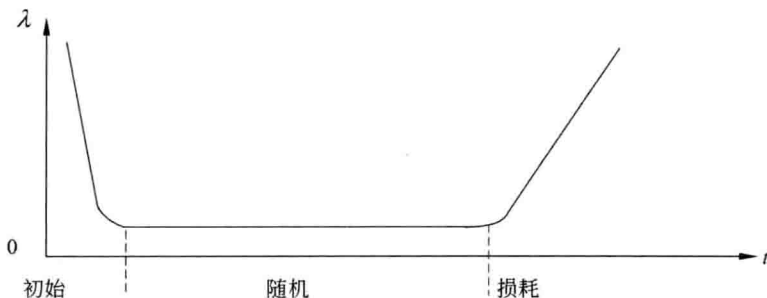


图 1-15 失效率特性

两次故障之间系统能正常工作的时间的平均值称为平均无故障时间 (MTBF), 即

$$\text{MTBF} = 1/\lambda$$

通常用平均修复时间 (MTRF) 来表示计算机的可维修性, 即计算机的维修效率, 指从故障发生到机器修复平均所需要的时间。计算机的可用性是指计算机的使用效率, 它以系统在执行任务的任意时刻能正常工作的概率 A 来表示, 即

$$A = \frac{\text{MTBF}}{\text{MTBF} + \text{MTRF}}$$

计算机的 RAS 是指用可靠性 R 、可用性 A 和可维修性 S 这 3 个指标衡量一个计算机系统。

但在实际应用中,引起计算机故障的原因除了元器件以外还有组装工艺、逻辑设计等因素。因此,不同厂家生产的兼容机即使采用相同的元器件,其可靠性及 MTBF 也可能相差很大。

2. 计算机可靠性模型

计算机系统是一个复杂的系统,而且影响其可靠性的因素非常复杂,很难直接对其进行可靠性分析。但通过建立适当的数学模型,把大系统分割成若干子系统,可以简化其分析过程。常见的系统可靠性数学模型有以下 3 种。

(1) 串联系统。假设一个系统由 N 个子系统组成,当且仅当所有的子系统都能正常工作时系统才能正常工作,这种系统称为串联系统,如图 1-16 所示。

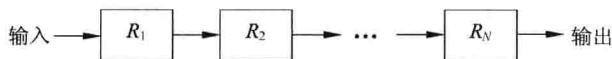


图 1-16 串联系统的可靠性模型

设系统中各个子系统的可靠性分别用 R_1, R_2, \dots, R_N 来表示,则系统的可靠性 R 可由下式求得。

$$R = R_1 R_2 \cdots R_N$$

如果系统的各个子系统的失效率分别用 $\lambda_1, \lambda_2, \dots, \lambda_N$ 来表示,则系统的失效率 λ 可由下式求得。

$$\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_N$$

【例 1.7】 设计计算机系统由 CPU、存储器、I/O 三部分组成,其可靠性分别为 0.95、0.90 和 0.85,求计算机系统的可靠性。

解: $R = R_1 \cdot R_2 \cdot R_3 = 0.95 \times 0.90 \times 0.85 = 0.73$

计算机系统的可靠性为 0.73。

(2) 并联系统。假如一个系统由 N 个子系统组成,只要有一个子系统正常工作,系统就能正常工作,这样的系统称为并联系统,如图 1-17 所示。设每个子系统的可靠性分别以 R_1, R_2, \dots, R_N 表示,整个系统的可靠性可由下式求得。

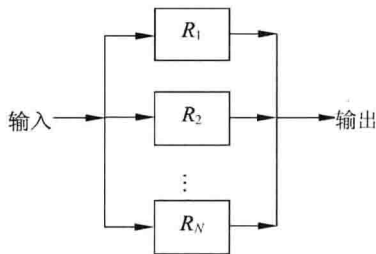


图 1-17 并联系统的可靠性模型

$$R = 1 - (1 - R_1)(1 - R_2) \cdots (1 - R_N)$$

假如所有子系统的失效率均为 λ ,则系统的失效率 μ 为

$$\mu = \frac{1}{\frac{1}{\lambda} \sum_{j=1}^N \frac{1}{j}}$$

在并联系统中只有一个子系统是真正需要的,其余 $N-1$ 个子系统称为冗余子系统,随着冗余子系统数量的增加,系统的平均无故障时间也增加了。

【例 1.8】 设一个系统由 3 个相同的子系统构成,其可靠性为 0.9,平均无故障时间为 10 000 小时,求系统的可靠性和平均无故障时间。

解: $R_1 = R_2 = R_3 = 0.9$ $\lambda_1 = \lambda_2 = \lambda_3 = 1/10\,000 = 1 \times 10^{-4}$ (小时)

系统可靠性 $R = 1 - (1 - R_1)^3 = 0.999$

系统平均无故障时间为

$$MTBF = \frac{1}{\mu} = \frac{1}{\lambda} \sum_{j=1}^3 \frac{1}{j} = \frac{1}{\lambda} \times (1 + \frac{1}{2} + \frac{1}{3}) = 18\,333 \text{ (小时)}$$

(3) N 模冗余系统。 N 模冗余系统由 N 个 ($N=2n+1$) 相同的子系统和一个表决器组成,表决器把 N 个子系统中占多数相同结果的输出作为系统的输出,如图 1-18 所示。

在 N 个子系统中,只要有 $n+1$ 个或 $n+1$ 个以上的子系统能正常工作,系统就能正常工作,输出正确的结果。假设表决器是完全可靠的,每个子系统的可靠性为 R_0 ,则 N 模冗余系统的可靠性为

$$R = \sum_{i=n+1}^N \binom{N}{i} \times R_0^i (1 - R_0)^{N-i}$$

其中, $\binom{N}{i}$ 表示从 N 个元素中取 i 个元素的组合数。

提高计算机的可靠性一般采取如下两项措施。

(1) 提高元器件质量,改进加工工艺与工艺结构,完善电路设计。

(2) 发展容错技术,使得在计算机硬件有故障的情况下,计算机仍能继续运行,得出正确的结果。

1.3.4 计算机系统的性能评价

无论是生产计算机的厂商还是使用计算机的用户,都需要有某种方法来衡量计算机的性能,作为设计、生产、购买和使用的依据。但是,由于计算机系统是一个极复杂的系统,其体系结构、组成和实现都有若干种策略,而且其应用领域也千差万别,所以很难找到统一的规则或标准去评测所有的计算机。

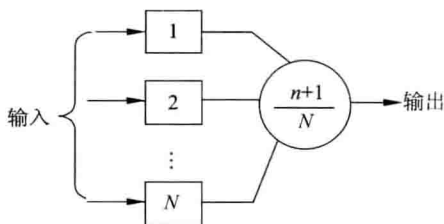


图 1-18 N 模冗余系统

1. 性能评测的常用方法

(1) 时钟频率。计算机的时钟频率在一定程度上反映了机器速度, 一般来讲, 主频越高, 速度越快。但是, 相同频率、不同体系结构的机器, 其速度可能会相差很多, 因此还需要用其他方法来测定机器性能。

(2) 指令执行速度。在计算机发展初期, 曾用加法指令的运算速度来衡量计算机的速度, 速度是计算机的主要性能指标之一。因为加法指令的运算速度大体上可反映出乘法、除法等其他算术运算的速度, 而且逻辑运算、转移指令等简单指令的执行时间往往被设计成与加法指令相同, 因此加法指令的运算速度有一定的代表性。当时表征机器运算速度的单位是 KIPS (每秒千条指令), 后来随着机器运算速度的提高, 计量单位发展到 MIPS (每秒百万条指令)。

另一种描述计算机指令执行速度的指标是每秒钟执行浮点数的百万次操作的数量 MFLOPS。

(3) 等效指令速度法。随着计算机指令系统的发展, 指令的种类大大增加, 用单种指令的 MIPS 值来表征机器的运算速度的局限性日益暴露, 因此出现了吉普森 (Gibson) 混合法或等效指令速度法等改进的办法。

等效指令速度法统计各类指令在程序中所占的比例, 并进行折算。设某类指令 i 在程序中所占的比例为 w_i , 执行时间为 t_i , 则等效指令的执行时间为

$$T = \sum_{i=1}^n (w_i \times t_i)$$

其中, n 为指令的种类数。

(4) 数据处理速率 (Processing Data Rate, PDR) 法。因为在不同程序中, 各类指令的使用频率是不同的, 所以固定比例方法存在着很大的局限性, 而且数据长度与指令功能的强弱对解题的速度影响极大。同时, 这种方法也不能反映现代计算机中高速缓冲存储器、流水线和交叉存储等结构的影响。具有这种结构的计算机的性能不仅与指令的执行频率有关, 而且与指令的执行顺序与地址分布有关。

PDR 法采用计算 PDR 值的方法来衡量机器性能, PDR 值越大, 机器性能越好。PDR 与每条指令和每个操作数的平均位数以及每条指令的平均运算速度有关, 其计算方法如下:

$$\text{PDR} = L / R$$

其中, $L = 0.85G + 0.15H + 0.4J + 0.15K$, $R = 0.85M + 0.09N + 0.06P$ 。

式中: G —— 每条定点指令的位数;

M —— 平均定点加法时间;

H —— 每条浮点指令的位数;

N —— 平均浮点加法时间;

J ——定点操作数的位数;

P ——平均浮点乘法时间;

K ——浮点操作数的位数。

此外,还做了如下规定: $G > 20$ 位, $H > 30$ 位;从主存取一条指令的时间等于取一个字的时间;指令与操作数存放在主存,无变址或间址操作;允许有并行或先行取指令功能,此时选择平均取指令时间。 PDR 值主要对 CPU 和主存储器的速度进行度量,但不适合衡量机器的整体速度,因为它没有涉及 Cache、多功能部件等技术对性能的影响。

(5) 核心程序法。上述性能评价方法主要是针对 CPU (有时包括主存),它没有考虑诸如 I/O 结构、操作系统、编译程序的效率等系统性能的影响,因此难以准确评价计算机的实际工作能力。

核心程序法是研究较多的一种方法,它把应用程序中用得最频繁的那部分核心程序作为评价计算机性能的标准程序,在不同的机器上运行,测得其执行时间,作为各类机器性能评价的依据。机器软/硬件结构的特点能在核心程序中得到反映,但是核心程序各部分之间的联系较小。由于程序短,所以访问存储器的局部性特征很明显,以至于 Cache 的命中率比一般程序高。

2. 基准测试程序

基准程序法(Benchmark)是目前被用户一致承认的测试性能的较好方法,有多种多样的基准程序,例如主要测试整数性能的基准程序、测试浮点性能的基准程序等。

(1) 整数测试程序。Dhrystone 是一个综合性的基准测试程序,它是为了测试编译器及 CPU 处理整数指令和控制功能的有效性,人为地选择一些“典型指令”综合起来形成的测试程序。

Dhrystone 程序测试的结果由每秒多少个 Dhrystones 来表示机器的性能,这个数值越大,性能越好。VAX11/780 的测试结果每秒 1757Dhrystones。为便于比较,人们假设 1VAX MIPS = 每秒 1757Dhrystones,将被测机器的结果除以 1757,就得到被测机器相对 VAX11/780 的 MIPS 值。有些厂家在宣布机器性能时就用 Dhrystone MIPS 值作为机器的 MIPS 值。

不过不同的厂家在测试 MIPS 值时,使用的基准程序一般是不一样的,因此不同厂家机器的 MIPS 值有时虽然是相同的,但其性能却可能差别很大,那是因为各厂家在设计计算机时针对不同的应用领域,如科学和工程应用、商业管理应用、图形处理应用等,而采用了不同的体系结构和实现方法。同一厂家的机器,采用相同的体系结构,用相同的基准程序测试,得到的 MIPS 值越大,一般说明机器速度越快。

(2) 浮点测试程序。在科学计算和工程应用领域内,浮点计算工作量占很大比例,因此机器的浮点性能对系统的应用有很大的影响。有些机器只标出单个浮点操作性能,如浮点加法、浮点乘法时间,而大部分工作站则标出用 Linpack 和 Whetstone 基准程序测得的浮点性能。

Linpack 主要测试向量性能和高速缓存性能。Whetstone 是一个综合性测试程序,除测试浮点操作外,还测试整数计算和功能调用等性能。

① 理论峰值浮点速度。巨型机和小巨型机在说明书中经常给出“理论峰值速度”的 MFLOPS 值,它不是机器实际执行程序时的速度,而是机器在理论上最大能完成的浮点处理速度。它不仅与处理机时钟周期有关,而且还与一个处理机里能并行执行操作的流水线功能部件数目和处理机的数目有关。多个 CPU 机器的峰值速度是单个 CPU 的峰值速度与 CPU 个数的乘积。

② Linpack 基准测试程序。Linpack 基准程序是一个用 FORTRAN 语言写成的子程序软件包,称为基本线性代数子程序包,此程序完成的主要操作是浮点加法和浮点乘法操作。在测量计算机系统的 Linpack 性能时,让机器运行 Linpack 程序,测量运行时间,将结果用 MFLOPS 表示。

当解 n 阶线性代数方程组时, n 越大,向量化程度越高。其关系如表 1-7 所示。

表 1-7 矩阵的向量化程度

矩阵规模	100×100	300×300	1000×1000
向量化百分比	80%	95%	98%

向量化百分比指含向量成分的计算量占整个程序计算量的百分比。在同一台机器中,向量化程度越高,机器的运算速度越快,因为不管 n 的大小,求解方程时花在非向量操作上的时间差不多是相等的。

③ Whetstone 基准测试程序。Whetstone 是用 FORTRAN 语言编写的综合性测试程序,主要由执行浮点运算、整数算术运算、功能调用、数组变址、条件转移和超越函数的程序组成。Whetstone 的测试结果用 Kwips 表示,1Kwips 表示机器每秒钟能执行 1000 条 Whetstone 指令。

(3) SPEC 基准程序(SPEC Benchmark)。SPEC(System Performance Evaluation Cooperation)是由几十家世界知名的计算机厂商所支持的非盈利的合作组织,旨在开发共同认可的标准基准程序,目前已更名为 Standard Performance Evaluation Cooperation。

SPEC 最初于 1989 年建立了重点面向处理器性能的基准程序集(现在称为 SPEC89),主要版本有 SPEC CPU89、SPEC CPU92、SPEC CPU95、SPEC CPU2000、SPEC CPU2006 等,SPEC CPU2006 包括 12 个整数基准程序集(CINT2006)和 17 个浮点基准程序集(CFP2006)。CINT2006 包括 C 编译程序、量子计算机仿真、下象棋程序等,CFP2006 包括有限元模型结构化网格法、分子动力学质点法、流体动力学稀疏线性代数法等。

为了简化测试结果,SPEC 决定使用单一的数字来归纳 12 种整数基准程序。具体方法是将被测计算机的执行时间标准化,即将被测计算机的执行时间除以一个参考处理器的执行时间,

结果称为 SPECratio。SPECratio 值越大,表示性能越快(因为 SPECratio 是执行时间的倒数)。CINT2006 或 CFP2006 的综合测试结果是取 SPECratio 的几何平均值。

SPEC 原来主要测试 CPU 性能,现在则强调开发能反映真实应用的基准测试程序集,并已推广至测试高性能计算机系统、网络服务器上商业应用服务器等。

(4) TPC 基准程序。TPC (Transaction Processing Council, 事务处理委员会) 基准程序是由 TPC 开发的评价计算机事务处理性能的测试程序,用于评测计算机在事务处理、数据库处理、企业管理与决策支持系统等方面的性能。其中,TPC-C 是在线事务处理 (On-line Transaction Processing, OLTP) 的基准程序,TPC-D 是决策支持的基准程序。TPC-E 作为大型企业信息服务的基准程序。与 TPC-C 一样,TPC-E 的测试结果也主要有两个指标:性能指标 (tpsE, transactions per second E) 和性价比 (美元/tpsE)。其中,前者是指系统在执行多种交易时,每秒钟可以处理多少交易,其指标值越大越好;后者则是指系统价格与前一指标的比值,数值越小越好。

TPC 基准测试程序在商业界范围内建立了用于衡量机器性能以及性能价格比的标准。但是,任何一种测试程序都有一定的适用范围,TPC 也不例外。

本书仅提供部分阅读，如需完整版，请联系QQ: 461573687

提供各种书籍pdf下载，如有需要，请联系 QQ: 461573687

PDF制作说明：

本人可以提供各种PDF电子书资料，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我 QQ: 461573687, 或者 QQ: 2404062482。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ。因PDF电子书都有版权，请不要随意传播，最近pdf也越来越难做了，希望大家尊重下个人劳动，谢谢！

备用QQ:2404062482