



全国计算机技术与软件专业技术资格（水平）考试指定用书

程序员教程

（第4版）

张淑平 霍秋艳 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

提供各种书籍的pd电子版代找服务，如果你找不到自己想要的书的pdf电子版，我们可以帮您找到，如有需要，请联系QQ1779903665.

PDF代找说明：

本人可以帮助你找到你要的PDF电子书，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%都带书签索引和目录，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QQ1779903665。

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，大部分我都可以找到，而且每本100%带书签索引目录。因PDF电子书都有版权，请不要随意传播，如果您有经济购买能力，请尽量购买正版。

声明：本人只提供代找服务，每本100%索引书签和目录，因寻找pdf电子书有一定难度，仅收取代找费用。如因PDF产生的版权纠纷，与本人无关，我们仅仅只是帮助你寻找到你要的pdf而已。

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  书文
局泉
www.wqbook.com

ISBN 978-7-302-36804-5



9 787302 368045 >

定价：59.00元

全国计算机技术与

平) 考试指定用书

程序员教程

(第4版)

张淑平 霍秋艳 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
北京

内 容 简 介

本书作为初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《程序员考试大纲》的重点内容,阐述了共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《程序员考试大纲》的指导下,对《程序员教程(第三版)(修订版)》进行再编后完成的。本书适合参加相关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

程序员教程/张淑平,霍秋艳主编. —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 978-7-302-36804-5

I. ①程… II. ①张…②霍… III. ①程序设计-工程技术人员-资格考试-教材 IV. ①TP311.1

中国版本图书馆CIP数据核字(2014)第124327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:刘海龙

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:32 防伪页:1 字 数:697千字

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:1~10000

定 价:59.00元

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才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，也对软件和信息服务业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。

苏.波

前 言

全国计算机技术与软件专业技术资格（水平）考试从实施至今已有二十余年，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《程序员教程（第三版）（修订版）》一书进行再编，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，因此编写本书的难度很高。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，各章的内容安排如下。

第 1 章 计算机系统基础知识：主要介绍计算机系统硬件组成、数据在计算机中的表示和运算、校验码基础知识和指令系统基础知识。

第 2 章 操作系统基础知识：主要介绍操作系统的类型和功能等基本概念，进程管理、存储管理、设备管理、文件管理和作业管理等基础知识。

第 3 章 数据库基础知识：主要介绍数据库管理系统的主要功能和特征、数据库模式、数据模型和 ER 图、关系运算和 SQL 等基础知识。

第 4 章 多媒体基础知识：主要介绍多媒体的基本概念、音频、图形和图像、动画和视频、网络多媒体等基础知识。

第 5 章 网络基础知识：主要介绍网络的功能、分类、组成和拓扑结构，基本的网络协议与标准，常用网络设备与网络通信设备的作用和特点、局域网（LAN）和互联网（Internet）基础知识。

第 6 章 程序设计语言基础知识：主要介绍程序设计语言的类型和特点、程序设计语言的基本成分以及编译、解释等基本的语言翻译基础知识。

第 7 章 软件工程基础知识：主要介绍软件工程和项目管理基础、面向对象分析与设计方法、软件需求分析、软件设计、编码和测试、软件系统运行与维护、软件质量管理等基础知识。

第 8 章 数据结构与算法：主要介绍线性表和链表、栈、队列、数组、树、图等基本数据结构以及查找、排序等常用算法。

第9章 标准化和知识产权基础知识: 主要介绍标准化的基本概念, 标准分类、标准的代号及编号等方面的基础知识; 知识产权的概念与特点、计算机软件著作权等方面的基础知识。

第10章 安全性基础知识: 主要介绍计算机病毒、网络安全、访问控制和加解密基础知识。

第11章 C/C++程序设计: 主要介绍 C/C++程序基础、类与对象、继承与多态、输入与输出流库、异常处理和常用 STL 模板库, 以及 C 程序中常见的问题和错误。

第12章 Java 语言程序设计: 主要介绍 Java 程序语言基础和特点、类与接口、异常、文件和输入/输出流、Java 小应用程序和 Java 类库等基础知识。

本书第1章由张淑平编写, 第2章、第3章由王亚平编写, 第4章由马志欣编写, 第5章由严体华编写, 第6章由张淑平编写, 第7章由褚华编写, 第8章由张淑平、陈静玉编写, 第9章由刘强编写, 第10章由严体华、张淑平编写, 第11章由霍秋艳、张淑平编写, 第12章由霍秋艳编写, 全书由张淑平、霍秋艳统稿。

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2月

目 录

第 1 章 计算机系统基础知识 1

- 1.1 计算机系统的基本组成 1
- 1.2 数据的表示及运算 3
 - 1.2.1 计算机中数据的表示 3
 - 1.2.2 校验码 12
 - 1.2.3 逻辑代数及逻辑运算 16
 - 1.2.4 机器数的运算 18
- 1.3 计算机的基本组成及工作原理 21
 - 1.3.1 总线的基本概念 22
 - 1.3.2 中央处理单元 24
 - 1.3.3 存储系统 28
 - 1.3.4 输入/输出技术 35
- 1.4 指令系统简介 38

第 2 章 操作系统基础知识 44

- 2.1 操作系统概述 44
- 2.2 处理机管理 47
 - 2.2.1 基本概念 47
 - 2.2.2 进程控制 50
 - 2.2.3 进程通信 50
 - 2.2.4 进程调度 55
 - 2.2.5 死锁 56
 - 2.2.6 线程 58
- 2.3 存储管理 58
 - 2.3.1 基本概念 59
 - 2.3.2 存储管理方案 60
 - 2.3.3 分页存储管理 61
 - 2.3.4 分段存储管理 63
 - 2.3.5 虚拟存储管理 64
- 2.4 设备管理 67
 - 2.4.1 设备管理概述 67
 - 2.4.2 设备管理技术 69

- 2.4.3 磁盘调度 71
- 2.5 文件管理 72
 - 2.5.1 基本概念 72
 - 2.5.2 文件的结构和组织 73
 - 2.5.3 文件目录 75
 - 2.5.4 存取方法、存取控制 76
 - 2.5.5 文件的使用 77
 - 2.5.6 文件的共享和保护 78
 - 2.5.7 系统的安全与可靠性 79
- 2.6 作业管理 80
 - 2.6.1 基本概念 81
 - 2.6.2 作业调度 82
 - 2.6.3 人机界面 85

第 3 章 数据库基础知识 87

- 3.1 基本概念 87
 - 3.1.1 数据库系统 87
 - 3.1.2 数据库管理技术的发展 88
 - 3.1.3 大数据 90
- 3.2 数据模型 93
 - 3.2.1 基本概念 93
 - 3.2.2 数据模型的三要素 94
 - 3.2.3 E-R 模型 94
 - 3.2.4 基本的数据模型 98
- 3.3 DBMS 的功能和特征 101
 - 3.3.1 DBMS 的功能 101
 - 3.3.2 DBMS 的特征与分类 102
- 3.4 数据库模式 104
 - 3.4.1 模式 104
 - 3.4.2 三级模式两级映像 105
- 3.5 关系数据库与关系运算 106
 - 3.5.1 关系数据库的基本概念 106

3.5.2 关系数据库模式	109	5.3 TCP/IP 协议体系结构	177
3.5.3 完整性约束	109	5.3.1 OSI/ISO 参考模型与 TCP/IP 体系结构	177
3.5.4 关系代数运算	110	5.3.2 TCP/IP 协议	179
3.6 关系数据库 SQL 语言简介	113	5.3.3 IP 地址	182
3.6.1 SQL 概述	114	5.4 Internet 基础知识	186
3.6.2 SQL 数据定义	115	5.4.1 Internet 服务	186
3.6.3 SQL 数据查询	119	5.4.2 因特网接入方式	190
3.6.4 SQL 数据更新	127	5.4.3 TCP/IP 的配置	192
3.6.5 SQL 的访问控制	129	5.4.4 浏览器的设置与使用	193
3.6.6 嵌入式 SQL	130	5.5 局域网基础	196
3.7 数据库设计	131	5.6 网络安全基本概念	201
第4章 多媒体基础知识	134	第6章 程序设计语言基础知识	206
4.1 多媒体的基本概念	134	6.1 程序设计语言概述	206
4.1.1 媒体的分类和特征	134	6.1.1 程序设计语言的基本概念	206
4.1.2 多媒体计算机系统	135	6.1.2 程序设计语言的分类和特点	207
4.2 音频	140	6.1.3 程序设计语言的基本成分	211
4.2.1 数字声音基础	140	6.2 语言处理程序基础	217
4.2.2 声音文件格式	145	6.2.1 汇编程序基本原理	217
4.3 图形和图像	146	6.2.2 编译程序基本原理	219
4.3.1 图像的基础知识	146	6.2.3 解释程序基本原理	228
4.3.2 图形与图像信息的表示和获取	148	第7章 软件工程基础知识	231
4.3.3 图形图像编码	151	7.1 软件工程概述	231
4.4 动画和视频	154	7.1.1 软件生存周期	231
4.4.1 动画的基本概念	154	7.1.2 软件生存周期模型	233
4.4.2 模拟视频和数字视频	156	7.1.3 软件过程	237
4.4.3 视频文件格式	160	7.1.4 软件工具	239
4.5 网络多媒体	161	7.1.5 软件开发环境	242
4.5.1 超文本与超媒体	161	7.2 软件需求分析	243
4.5.2 流媒体的基本概念	162	7.2.1 软件需求的定义	243
第5章 网络基础知识	164	7.2.2 软件需求分析的基本任务	243
5.1 计算机网络概述	164	7.2.3 需求建模	244
5.1.1 计算机网络的组成	164	7.3 软件设计	244
5.1.2 计算机网络的分类	165	7.3.1 软件设计的基本任务	245
5.2 计算机网络硬件	168	7.3.2 软件设计原则	246
5.2.1 计算机网络互连设备	168	7.4 结构化分析与设计方法	249
5.2.2 计算机网络传输媒体	173		

7.4.1 结构化分析方法	249	9.1.1 基本概念	337
7.4.2 结构化设计方法	251	9.1.2 信息技术标准化	342
7.4.3 结构化程序设计方法	254	9.1.3 标准化组织	345
7.5 面向对象分析与设计方法	254	9.1.4 ISO 9000 标准简介	347
7.5.1 面向对象的基本概念	254	9.1.5 能力成熟度模型简介	349
7.5.2 面向对象分析与设计	256	9.2 知识产权基础知识	351
7.5.3 UML 概述	257	9.2.1 基本概念	351
7.5.4 设计模式	259	9.2.2 计算机软件著作权	353
7.6 软件测试与运行	261	9.2.3 计算机软件的商业秘密权	364
7.6.1 软件测试的目的及原则	261	第 10 章 安全性基础知识	367
7.6.2 软件测试方法	263	10.1 安全性概述	367
7.6.3 软件测试过程	266	10.2 计算机病毒和计算机犯罪概述	369
7.6.4 软件测试设计和管理	268	10.3 网络安全	375
7.6.5 软件调试	268	10.4 访问控制	378
7.6.6 软件运行与维护	269	10.5 加密与解密	380
7.7 软件项目管理	271	第 11 章 C/C++ 程序设计	385
7.7.1 软件项目管理概述	271	11.1 C/C++ 程序基础	385
7.7.2 软件质量与软件质量保证	274	11.1.1 C/C++ 程序基本结构	386
第 8 章 数据结构与算法	278	11.1.2 数据类型和运算符	387
8.1 线性结构	278	11.1.3 基本输入/输出	395
8.1.1 线性表	278	11.1.4 控制语句	399
8.1.2 栈和队列	285	11.1.5 函数	403
8.1.3 串	291	11.1.6 指针与引用	407
8.2 数组和矩阵	292	11.2 类与对象	412
8.3 树和图	296	11.3 继承与多态	420
8.3.1 树	296	11.4 输入与输出流库	425
8.3.2 图	303	11.5 异常处理	428
8.4 常用算法	307	11.6 类库	431
8.4.1 算法概述	307	11.6.1 string	431
8.4.2 排序算法	312	11.6.2 STL	435
8.4.3 查找算法	319	11.6.3 vector	436
8.4.4 字符串处理	328	11.7 C 程序设计要点	440
8.4.5 递归算法	331	11.7.1 指针与链表	440
8.4.6 图的相关算法	332	11.7.2 递归函数	441
第 9 章 标准化和知识产权基础知识	337	11.7.3 程序错误	442
9.1 标准化的基本知识	337		

第 12 章 Java 语言程序设计	452	12.3.4 继承	472
12.1 Java 语言概述.....	452	12.3.5 抽象类与接口	474
12.1.1 Java 语言的特点	452	12.4 异常	480
12.1.2 Java 开发环境	453	12.4.1 异常的处理	480
12.2 Java 语言基础.....	453	12.4.2 自定义异常	482
12.2.1 Java 基本数据类型	454	12.5 文件输入、输出和流	484
12.2.2 控制结构	460	12.5.1 字节流	484
12.2.3 Java 数组与字符串	462	12.5.2 字符流	487
12.3 类与接口	466	12.5.3 标准输入/输出流	488
12.3.1 类的定义与使用	466	12.6 Java 小应用程序	490
12.3.2 对象的初始化	468	12.7 Java 类库的使用	493
12.3.3 包	470		

第 1 章 计算机系统基础知识

本章主要包括计算机系统的组成、计算机中数据的表示和运算、计算机系统硬件基础组成及指令系统等基础知识。

1.1 计算机系统的基本组成

计算机系统是由硬件系统和软件系统组成的, 计算机硬件是计算机系统中看得见、摸得着的物理装置, 计算机软件是程序、数据和相关文档的集合。计算机系统的主要组成如图 1-1 所示。

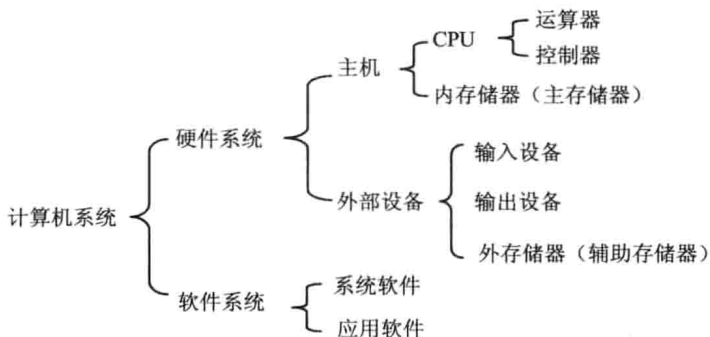


图 1-1 计算机系统的组成示意图

1. 计算机系统的硬件组成

传统概念上, 基本的计算机硬件系统由运算器、控制器、存储器、输入设备和输出设备 5 大部件组成, 随着网络技术的发展, 通信部件也逐渐成为其基本构件。运算器和控制器及其相关部件已被集成在一起, 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。

运算器是对数据进行加工处理的部件, 它主要完成算术和逻辑运算。控制器的主要功

能则是从主存中取出指令并进行分析，控制计算机的各个部件有条不紊地完成指令的功能。

存储器是计算机系统记忆设备，分为内部存储器（Main Memory，MM，简称内存、主存）和外部存储器（简称外存）。内存速度快、容量小，一般用来临时存储计算机运行时所需的程序、数据及中间结果。外存容量大、速度慢，可用于长期保存信息。寄存器是 CPU 中的记忆器件，用来临时存放指令、数据及运算结果。与内部存储器相比，寄存器的速度要快得多。

习惯上将 CPU 和主存储器的有机组合称为主机。输入/输出（I/O）设备位于主机之外，是计算机系统与外界交换信息的装置。所谓输入和输出，都是相对于主机而言的。输入设备的作用是把转换成二进制形式的信息输入到计算机的存储器中，输出设备的作用是把运算结果按照人们所要求的形式输出到外部设备或存储介质上。

2. 计算机软件

计算机软件是指为管理、运行、维护及应用计算机系统所开发的程序和相关文档的集合。如果计算机系统中仅有硬件系统，则只具备了计算的基础，并不能真正运算，只有将解决问题的步骤编制成程序并加载到计算机内存开始运行，才能完成运算。软件系统是计算机系统中的重要组成部分，通常可将软件分为系统软件和应用软件两大类。

3. 计算机的类型

计算机技术的发展异常迅速，将更多的元件集成到单一的半导体芯片上，使得计算机变得更小，功耗更低，速度更快。

（1）按照体积和工作能力，计算机分为巨型机、大型机、小型机和微型机。微型机有多种形式，如台式机（Desktop）、膝上型计算机（Laptop）或笔记本式计算机（Notebook）、工作站（Workstation）、掌上型计算机和个人数字助理（Personal Digital Assistant，PDA）等。

（2）按照功能是否专一，计算机分为通用计算机和专用（嵌入式）计算机。

（3）按照 CPU 的指令系统架构，计算机分为复杂指令系统计算机（Complex Instruction Set Computer，CISC）和精简指令系统计算机（Reduced Instruction Set Computer，RISC）。

（4）按体系结构及指令处理方式，计算机分为单指令流单数据流计算机（Single Instruction Single Data，SISD）、单指令流多数据流计算机（Single Instruction Multiple Data，SIMD）、多指令流单数据流计算机（Multiple Instruction Single Data，MISD）和多指令流多数据流计算机（Multiple Instruction Multiple Data，MIMD）。

1.2 数据的表示及运算

1.2.1 计算机中数据的表示

在计算机内部,数值、文字、声音、图形图像等各种信息都必须经过数字化编码后才能被传送、存储和处理。所谓编码,就是采用少量的基本符号,选用一定的组合原则,来表示大量复杂多样的信息。基本符号的种类和这些符号的组合规则是一切信息编码的两大要素。例如,用10个阿拉伯数码表示数字,用26个英文字母表示英文词汇等,都是编码的典型例子。

1. 进位计数制及其转换

在采用进位计数的数字系统中,如果只用 r 个基本符号表示数值,则称其为 r 进制(Radix- r Number System), r 称为该数制的基数(Radix)。不同数制的共同特点如下。

(1) 每一种数制都有固定的符号集。例如,十进制数制的基本符号有十个:0, 1, 2, ..., 9。二进制数制的基本符号有两个:0和1。

(2) 每一种数制都使用位置表示法。即处于不同位置的数符所代表的值不同,与它所在位置的权值有关。

例如,十进制数1234.55可表示为

$$1234.55 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1} + 5 \times 10^{-2}$$

可以看出,各种进位计数制中权的值恰好是基数的某次幂。因此,对任何一种进位计数制表示的数都可以写成按权展开的多项式。计算机中常用的几种进位计数制如表1-1所示。

表 1-1 计算机中常用的进位计数制的表示

进 位 制	二 进 制	八 进 制	十 进 制	十 六 进 制
规则	逢二进一	逢八进一	逢十进一	逢十六进一
基数	$r=2$	$r=8$	$r=10$	$r=16$
数符	0, 1	0, 1, 2, ..., 7	0, 1, 2, ..., 9	0, 1, 2, ..., 9, A, B, ..., F
权	2^i	8^i	10^i	16^i
形式表示符	B	O	D	H

1) 十进制计数法与二进制计数法的相互转换

在十进制计数制中, $r=10$,基本符号为0, 1, 2, ..., 9。无论多大的数,都是用这10个符号的组合来表示,故称为十进制计数法。

在二进制计数制中, $r=2$,基本符号为0和1。二进制数中的一个0或1称为1位(bit)。

二进制数转换成十进制数的方法是：将二进制数的每一位数乘以它的权，然后相加，即可求得对应的十进制数值。

【例 1-1】 把二进制数 100110.101 转换成相应的十进制数。

$$\begin{aligned}(100110.101)_2 &= 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\ &= 32 + 0 + 0 + 4 + 2 + 0 + 0.5 + 0 + 0.125 \\ &= 38.625\end{aligned}$$

将十进制数转换成二进制数时，整数部分和小数部分分别转换，然后再合并。十进制整数转换为二进制整数的方法是“除 2 取余”；十进制小数转换为二进制小数的方法是“乘 2 取整”。

十进制数转换成二进制数还有一种简便的方法：把一个十进制数写成按二进制数权的大小展开的多项式，按权值从高到低依次取各项的系数就可得到相应的二进制数。

【例 1-2】 把十进制数 175.71875 转换为相应的二进制数。

$$\begin{aligned}(175.71875)_{10} &= 2^7 + 2^5 + 2^3 + 2^2 + 2^1 + 2^0 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-5} \\ &= 10101111.10111_2\end{aligned}$$

2) 八进制计数法与十进制、二进制计数法的相互转换

八进制计数制中的基本符号为 0, 1, 2, ..., 7。

十进制数转换为八进制数的方法是：对于十进制整数采用“除 8 取余”的方法转换为八进制整数；对于十进制小数则采用“乘 8 取整”的方法转换为八进制小数。

二进制数转换成八进制数的方法是：从小数点起，把二进制数每三位分成一组，然后写出每一组的等值八进制数，顺序排列起来就得到所要求的八进制数。

依照同样的思想，将一位八进制数用三位二进制数表示，就可以直接将八进制数转换成二进制数。

二进制与八进制数之间的对应关系如表 1-2 所示。

表 1-2 二进制、八进制和十六进制数之间的对应关系

二进制	八进制	二进制	十六进制	二进制	十六进制
000	0	0000	0	1000	8
001	1	0001	1	1001	9
010	2	0010	2	1010	A
011	3	0011	3	1011	B
100	4	0100	4	1100	C
101	5	0101	5	1101	D
110	6	0110	6	1110	E
111	7	0111	7	1111	F

【例 1-3】 把二进制数 10101111.10111 转换为相应的八进制数。

$$(10\ 101\ 111.101\ 11)_2 = 257.56_8$$

3) 十六进制计数法十进制、二进制计数法的相互转换

在十六进制计数制中, $r=16$, 基本符号为 0, 1, 2, ..., 9, A, B, ..., F。

十进制数可以转换为十六进制数的方法是: 十进制数的整数部分“除 16 取余”, 十进制数的小数部分“乘 16 取整”。

由于一位十六进制数可以用 4 位二进制数来表示, 因此二进制数与十六进制数的相互转换就比较容易。二进制数转换成十六进制数的方法是: 从小数点开始, 每 4 位二进制数为一组, 将每一组用相应的十六进制数符来表示, 即可得到正确的十六进制数。

二进制与十六进制数之间的对应关系如表 1-2 所示。

【例 1-4】 把二进制数 10101111.10111 转换为相应的十六进制数。

$$(1010\ 1111.1011\ 1)_2 = \text{AF.B8}_{16}$$

2. 二进制运算规则

(1) 加法: 二进制加法的进位规则是“逢二进一”。

$$0+0=0 \quad 1+0=1 \quad 0+1=1 \quad 1+1=0 \text{ (有进位)}$$

(2) 减法: 二进制减法的借位规则是“借一当二”。

$$0-0=0 \quad 1-0=1 \quad 1-1=0 \quad 0-1=1 \text{ (有借位)}$$

(3) 乘法:

$$0 \times 0=0 \quad 1 \times 0=0 \quad 0 \times 1=0 \quad 1 \times 1=1$$

3. 机器数和码制

各种数据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、1 表示, 小数点隐含表示而不占位置。机器数对应的实际数值称为数的真值。

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。无符号数是针对二进制来讲的, 无符号数的表数范围是非负数, 即全部二进制位均代表数值, 没有符号位。

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。

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。数值零的原码表示有两种形式: $[+0]_{\text{原}}=00000000$, $[-0]_{\text{原}}=10000000$ 。

【例 1-5】 若机器字长 n 等于 8, 则

$[+1]_{\text{原}}=00000001$	$[-1]_{\text{原}}=10000001$
$[+127]_{\text{原}}=01111111$	$[-127]_{\text{原}}=11111111$
$[+45]_{\text{原}}=00101101$	$[-45]_{\text{原}}=10101101$
$[+0.5]_{\text{原}}=0 \diamond 1000000$	$[-0.5]_{\text{原}}=1 \diamond 1000000$ (其中 \diamond 是小数点的位置)

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。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: $[+0]_{\text{反}}=00000000$, $[-0]_{\text{反}}=11111111$ 。

【例 1-6】 若机器字长 n 等于 8, 则

$[+1]_{\text{反}}=00000001$	$[-1]_{\text{反}}=11111110$
$[+127]_{\text{反}}=01111111$	$[-127]_{\text{反}}=10000000$
$[+45]_{\text{反}}=00101101$	$[-45]_{\text{反}}=11010010$
$[+0.5]_{\text{反}}=0 \diamond 1000000$	$[-0.5]_{\text{反}}=1 \diamond 01111111$ (其中 \diamond 是小数点的位置)

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 , 则最高位为符号位, 0 表示正号, 1 表示负号, 其余的 $n-1$ 位表示数值。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: $[+0]_{\text{补}}=00000000$, $[-0]_{\text{补}}=00000000$ 。

【例 1-7】 若机器字长 n 等于 8, 则

$[+1]_{\text{补}}=00000001$	$[-1]_{\text{补}}=11111111$
$[+127]_{\text{补}}=01111111$	$[-127]_{\text{补}}=10000001$
$[+45]_{\text{补}}=00101101$	$[-45]_{\text{补}}=11010011$
$[+0.5]_{\text{补}}=0 \diamond 1000000$	$[-0.5]_{\text{补}}=1 \diamond 1000000$ (其中 \diamond 是小数点的位置)

相对于原码和反码表示, 补码表示法有一个例外, 当符号位为 1 而数值位全部为 0 时, 它表示整数 2^{n-1} , 即此时符号位的 1 既表示负数又表示数值。

设计补码时, 有意识的引用了模运算在数理上对符号位的自动处理, 利用模的自动丢弃实现了符号位的自然处理。

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

【例 1-8】 若机器字长 n 等于 8, 则

$[+1]_{\text{移}}=10000001$	$[-1]_{\text{移}}=01111111$
$[+127]_{\text{移}}=11111111$	$[-127]_{\text{移}}=00000001$
$[+45]_{\text{移}}=10101101$	$[-45]_{\text{移}}=01010011$

$[+0]_{\text{移}} = 10000000$
 $[-0]_{\text{移}} = 10000000$

4. 定点数和浮点数

1) 定点数

所谓定点数,就是表示数据时小数点的位置固定不变。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 , 各种码制表示下的带符号数的范围如表 1-3 所示。

表 1-3 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	$-2^{n-1} \sim +(2^{n-1}-1)$	$-1 \sim +(1-2^{-(n-1)})$

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 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, 运算中很容易因结果超出范围而溢出。

2) 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, 一个实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。

一个含小数点的二进制数 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 为尾数, 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。若不对浮点数的表示作出明确规定，同一个浮点数的表示就不是唯一的。

为了提高数据的表示精度，当尾数的值不为 0 时，规定尾数域的最高有效位应为 1，这称为浮点数的规格化表示。否则修改阶码同时左右移小数点位置的，使其变为规格化数的形式。

规格化就是将尾数的绝对值限定在区间[0.5, 1)。

(1) 若尾数 $F \geq 0$ ，则其规格化的尾数形式为 $F=01 \times \times \times \cdots \times$ ，其中 \times 可为 0，也可为 1，即将尾数 F 的范围限定在区间[0.5, 1)。

(2) 若尾数 $F < 0$ ，则其规格化的尾数形式为 $F=10 \times \times \times \cdots \times$ ，其中 \times 可为 0，也可为 1，即将尾数 F 的范围限定在区间 $(-1, -0.5]$ 。

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：

S	P	M
-----	-----	-----

其中， S 为数的符号位，为 0 时表示正数，为 1 时表示负数； P 为指数（阶码），用移码表示（偏移值为 $2^{p-1}-1$ ， p 为阶码的位数）； M 为尾数，用原码表示。

对于阶码为 0 或 255 的情况，IEEE 754 标准有特别的规定：

若 P 为 0 且 M 为 0，则表示真值 ± 0 （正负号和数符位有关）。如果 $P=255$ 并且 M 是 0，则这个数的真值为 $\pm \infty$ （与符号位有关）；如果 $P=255$ 并且 M 不是 0，则这不是一个数（NaN）。

目前，计算机中主要使用三种形式的 IEEE 754 浮点数，如表 1-4 所示。

表 1-4 三种形式的 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	23	52	64
符号位长度	1	1	1
阶码长度	8	11	15
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，对于单精度浮点数和双精度浮点数，约定小数点左边隐含有一位，通常这位数就是 1，因此尾数为 $1.\times \times \cdots \times$ 。

【例 1-9】 利用 IEEE 754 标准将数 176.0625 表示为单精度浮点数。

解：首先将该十进制数转换成二进制数，即 $(176.0625)_{10} = (10110000.0001)_2$ ，其次对二进制数进行规格化处理，即 $10110000.0001 = 1 \diamond 01100000001 \times 2^7$ 。这就保证了最高位为 1，而且小数点应当在 \diamond 位置上，将最高位去掉并扩展为单精度浮点数所规定的 23 位尾数，得到 01100000001000000000000。

然后求阶码，上述表示中指数为 7，用移码表示为 10000110（偏移量是 127，因此偏移后的指数值为 $7+127=134$ ）。

最后，可得到 $(176.0625)_{10}$ 的单精度浮点数表示形式：

0 10000110 01100000001000000000000

5. 十进制数与字符的编码表示

数值、文字和英文字母等都被认为是字符，任何字符进入计算机时，都必须转换成二进制表示形式，称为字符编码。

用 4 位二进制代码表示一位十进制数，称为二-十进制编码，简称 BCD 编码。因为 $2^4=16$ ，而十进制数只有 0~9 这 10 个不同的数符，故有多种 BCD 编码。根据 4 位代码中每一位是否有确定的权来划分，可分为有权码和无权码两类。

应用最多的有权码是 8421 码，即 4 个二进制位的权从高到低分别为 8、4、2 和 1。无权码中常用余 3 码和格雷码。余 3 码是在 8421 码的基础上，把每个数的代码加上 0011 后构成的。格雷码的编码规则是相邻的两个代码之间只有 1 位不同。

常用的 8421BCD 码、余 3 码、格雷码与十进制数的对应关系如表 1-5 所示。

表 1-5 8421BCD 码、余 3 码、格雷码与十进制数的对应关系

十 进 制 数	8421BCD 码	余 3BCD 码	格 雷 码
0	0000	0011	0000
1	0001	0100	0001
2	0010	0101	0011
3	0011	0110	0010
4	0100	0111	0110
5	0101	1000	1110
6	0110	1001	1010
7	0111	1010	1000
8	1000	1011	1100
9	1001	1100	0100

6. ASCII 码

ASCII 码（American Standard Code for Information Interchange，美国标准信息交换代码）已被国际标准化组织 ISO 采纳，成为一种国际通用的信息交换用标准代码。基本的 ASCII 码采用 7 个二进制位，即 $d_6d_5d_4d_3d_2d_1d_0$ 对字符进行编码：低 4 位组 $d_3d_2d_1d_0$ 用作行编码，高 3 位组 $d_6d_5d_4$ 用作列编码。基本的 ASCII 字符代码表如表 1-6 所示。

表 1-6 7 位 ASCII 代码表

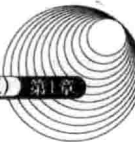
$d_3d_2d_1d_0$ 位 (低 4 位)	$d_6d_5d_4$ 位 (高 3 位)							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	,	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	,	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	'	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	↑	n	~
1111	DI	US	/	?	O	↓	o	Del

根据 ASCII 码的构成格式，可以很方便地从对应的代码表中查出每一个字符的编码。例如，字符 0 的 ASCII 码值为 0110000 ($2^5+2^4=48$)，字符 a 的 ASCII 码值为 1100001 ($2^6+2^5+2^0=97$)。

7. 汉字编码

在计算机中处理汉字，必须先将汉字代码化，即对汉字进行编码。汉字处理包括汉字的编码输入、汉字的存储和汉字的输出等环节。

西文是拼音文字，基本符号比较少，编码比较容易，而且在一个计算机系统中，输入、内



部处理、存储和输出都可以使用同一代码。汉字种类繁多,编码比拼音文字困难,而且在一个汉字处理系统中,输入、内部处理、存储和输出对汉字代码的要求不尽相同,所以采用的编码也不同。汉字信息处理系统在处理汉字和词语时,关键的问题是要进行一系列的汉字代码转换。

1) 输入码

中文的字数繁多,字形复杂,字音多变,常用汉字就有 7000 个左右。为了能直接使用西文标准键盘输入汉字,必须为汉字设计相应的编码方法,汉字的输入码主要分为三类:数字编码、拼音码和字形码。

(1) 数字编码。数字编码就是用数字串代表一个汉字的输入,常用的是国标区位码。国标区位码将国家标准局公布的 6763 个两级汉字分成 94 个区,每个区 94 位,区码和位码各两位十进制数字。例如,“中”字位于第 54 区 48 位,区位码为 5448。

汉字在区位码表的排列是有规律的。在 94 个分区中,1~15 区用来表示字母、数字和符号,16~87 区为一级和二级汉字。一级汉字以汉语拼音为序排列,二级汉字以偏旁部首进行排列。使用区位码方法输入汉字时,必须先是在表中查找汉字对应的代码,才能输入。数字编码输入的优点是重码率低,而且输入码和内部编码的转换比较方便,但是数字码难以记忆。

(2) 拼音码。拼音码是以汉语读音为基础的输入方法。由于汉字同音字太多,输入重码率很高,因此,按拼音输入后还必须进行同音字选择,会影响输入速度。

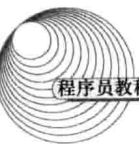
(3) 字形编码。字形编码是以汉字的形状确定的编码。汉字总数虽多,但都是由一笔一划组成,全部汉字的部件和笔划是有限的。因此,把汉字的笔划部件用字母或数字进行编码,按笔划书写的顺序依次输入,就能表示一个汉字,五笔字型、表形码等便是这种编码法。五笔字型编码是最常见的输入码。

2) 内部码

汉字内部码(简称汉字内码)是汉字在设备和信息处理系统内部存储、处理、传输汉字用的代码。汉字数量多,用一个字节无法区分,采用国家标准局 GB2312—80 中规定的汉字国标码,两个字节存放一个汉字的内码,每个字节的最高位置 1,作为汉字机内码。由于两个字节各用 7 位,因此可表示 16 384 个可区别的机内码。以汉字“大”为例,国标码为 3473H,两个字节的高位置 1,得到的机内码为 B4F3H。

为了统一地表示世界各国的文字,国际标准化组织 1993 年公布了“通用多八位编码字符集”国际标准 ISO/IEC 10646,简称 UCS (Universal Code Set)。UCS 包含了中、日、韩等国的文字,这一标准为包括汉字在内的各种正在使用的文字规定了统一的编码方案。该标准是用 4 个 8 位码(4 个字节)来表示每个字符,并相应地指定组、平面、行和字位。

- 组: 用一个 8 位二进制来编码组,最高位不用,剩下 7 位,能表示 128 个组。
- 平面: 用一个 8 位二进制来编码平面,能表示 256 个平面。
- 行: 用一个 8 位二进制来编码行,能表示 256 个行。
- 字位: 用一个 8 位二进制来编码字位,能表示 256 个字位。



一个字符就被安排在这个编码空间的一个字位上。4 个 8 位码 32 位足以包容世界上所有的字符,同时也符合现代处理系统的体系结构。

例如,ASCII 字符“A”,其 ASCII 码用十六进制表示为 41H。它在 UCS 中的编码为 00000041H,即在 00 组,00 面,00 行,第 41H 字位上。汉字“大”在 GB 2312 中的编码为 3474H,它在 UCS 中的编码为 00005927H,即在 00 组,00 面,59H 行,第 27H 字位上。

我国相应的国家标准为 GB 13000。详细内容请查阅网址 <http://www.unicode.org>。

3) 字形码

汉字字形码是表示汉字字形的字模数据,通常用点阵、矢量函数等方式表示,用点阵表示字形时,汉字字形码指的就是这个汉字字形点阵的代码。字形码也称字模码,是用点阵表示的汉字字形码,它是汉字的输出方式,根据输出汉字的要求不同,点阵的多少也不同。简易型汉字为 16×16 点阵,高精度型汉字为 24×24 点阵、 32×32 点阵、 48×48 点阵等。

字模点阵的信息量是很大的,所占存储空间也很大,以 16×16 点阵为例,每个汉字就需要 32 字节用于机内存储。字库中存储了每个汉字的点阵代码,当显示输出时才检索字库,输出字模点阵得到字形。

汉字的矢量表示法是将汉字看作由笔画组成的图形,提取每个笔画的坐标值,这些坐标值就可以决定每一笔画的位置,将每一个汉字的所有坐标值信息组合起来就是该汉字字形的矢量信息。显然,汉字的字形不同,其矢量信息也就不同,每个汉字都有自己的矢量信息。由于汉字的笔画不同,则矢量信息就不同。所以,每个汉字矢量信息所占的内存大小不一样。这与点阵表示法不一样。

同样,将每一个汉字的矢量信息集中在一起就构成了汉字库。当需要汉字输出时,利用汉字字形检索程序根据汉字内码从字模库中找到相应的字形码。

1.2.2 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就会变成错误编码,这样就可以检测出接收到的数据是否有错。

码距是校验码中的一个重要概念。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编码,因此这种代码无差错检验能力。下面简单介绍常用的三种校验码:奇偶校验码 (Parity Codes)、海明码 (Hamming Code) 和循环冗余校验 (Cyclic Redundancy Check, CRC) 码。

1. 奇偶校验码

奇偶校验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个校验位来使编码中1的个数为奇数(奇校验)或者偶数(偶校验),从而使码距变为2。对于奇偶校验,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,即当合法编码中奇数位发生了错误,也就是编码中的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。

表 1-7 8421 码的奇偶校验码

十 进 制 数	8421BCD 码	带奇校验位的 8421 码	带偶校验位的 8421 码
0	0000	0000 1	0000 0
1	0001	0001 0	0001 1
2	0010	0010 0	0010 1
3	0011	0011 1	0011 0
4	0100	0100 0	0100 1
5	0101	0101 1	0101 0
6	0110	0110 1	0110 0
7	0111	0111 0	0111 1
8	1000	1000 0	1000 1
9	1001	1001 1	1001 0

从表 1-7 可知,带奇偶校验位的 8421 码由 4 位信息位和 1 位校验位组成,码距为 2,能检查出代码信息中奇数位出错的情况,而错在哪些位是检查不出来的。也就是说,它只能发现错误,而不能校正错误。

常用的奇偶校验码有三种:水平奇偶校验码、垂直奇偶校验码和水平垂直校验码。

(1) 水平奇偶校验码。对每一个数据的编码添加校验位,使信息位与校验位处于同一行。

(2) 垂直奇偶校验码。这种校验码把数据分成若干组,一组数据占一行,排列整齐,再加一行校验码,针对每一列采用奇校验或偶校验。

【例 1-10】 对于 32 位数据 10100101 00110110 11001100 10101011,其垂直奇校验和垂直偶校验如下所示。

编码分类	垂直奇校验码	垂直偶校验码
数据	1 0 1 0 0 1 0 1	1 0 1 0 0 1 0 1
	0 0 1 1 0 1 1 0	0 0 1 1 0 1 1 0
	1 1 0 0 1 1 0 0	1 1 0 0 1 1 0 0
	1 0 1 0 1 0 1 1	1 0 1 0 1 0 1 1
校验位	0 0 0 0 1 0 1 1	1 1 1 1 0 1 0 0

(3) 水平垂直校验码。在垂直校验码的基础上, 对每个数据再增加一位水平校验位, 便构成水平垂直校验码。

【例 1-11】对于 32 位数据 10100101 00110110 11001100 10101011, 其水平垂直奇校验和偶校验如下所示。

奇偶类	水平垂直奇校验码			水平垂直偶校验码		
分类	水平校验位	数据		水平校验位	数据	
数据	1	1	0	1	0	1
	1	0	0	1	1	0
	1	1	1	0	0	1
	0	1	0	1	0	1
垂直校验位	0	0	0	0	1	1

2. 海明码

海明码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: 在数据位之间插入 k 个校验位, 通过扩大码距来实现检错和纠错。

例如, 对于 8 位的数据位, 进行海明校验需要 4 个校验位。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) 首先确定数据位与校验位在海明码中的位置, 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

校验位设置在 2^i 位置, 因此 P_1 对应 H_1 , P_2 对应 H_2 , P_3 对应 H_4 , P_4 对应 H_8 。

每个校验位只校验数据位中位置号的二进制编码和自身位置号的二进制编码相匹配的数据位。例如, D_3 (H_7) 的位置号为 7 ($=4+2+1$), 因此该数据位由 P_1 、 P_2 和 P_3 校验。

(2) 通过校验关系, 确定各校验位的值。

P_1 偶校验: P_1 、 D_0 、 D_1 、 D_3 、 D_4 、 D_6

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

P_2 偶校验: P_2 、 D_0 、 D_2 、 D_3 、 D_5 、 D_6

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

P_3 偶校验: P_3 、 D_1 、 D_2 、 D_3 、 D_7

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

P_4 偶校验: P_4 、 D_4 、 D_5 、 D_6 、 D_7

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

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, 只需作以下计算:

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

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

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

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

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 (奇校验则应全为 1)。当 $G_4G_3G_2G_1$ 不全为 0 说明发生了差错, 而且 $G_4G_3G_2G_1$ 的十进制值指出发生错误的位置。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 1-12】 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

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

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

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

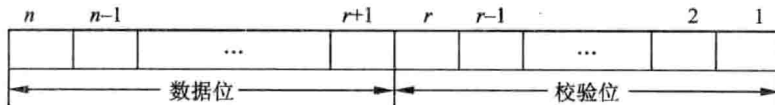
$$P_4 = D_4 \oplus D_5 \oplus D_6 \oplus D_7 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

因此, 求得的海明码为:

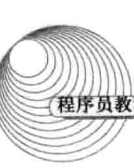
H_{12}	H_{11}	H_{10}	H_9	H_8	H_7	H_6	H_5	H_4	H_3	H_2	H_1
D_7	D_6	D_5	D_4	P_4	D_3	D_2	D_1	P_3	D_0	P_2	P_1
0	1	1	0	0	1	0	0	1	1	0	1

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，如下所示：

$$0+0=0 \quad 1+0=1 \quad 0+1=1 \quad 1+1=0$$

$$0-0=0 \quad 1-0=1 \quad 0-1=1 \quad 1-1=0$$

1.2.3 逻辑代数及逻辑运算

逻辑代数是 1849 年英国数学家乔治·布尔提出的，它是用代数的方式对逻辑变量进行描述和分析的数学工具，也称为布尔代数。逻辑变量的取值只有“真”和“假”，通常以 1 表示“真”，0 表示“假”。

1. 基本的逻辑运算

在逻辑代数中有三种最基本的运算：“与”运算、“或”运算、“非”运算，其他逻辑运算可由这三种基本运算进行组合来表示。

1) “与”运算

“与”运算又称为逻辑乘，其运算符号常用 AND、 \cap 、 \wedge 或 \cdot 表示。设 A 和 B 为两个逻辑变量，当且仅当 A 和 B 的取值都为“真”时， A “与” B 的值为“真”；否则 A “与” B 的值为“假”，如表 1-8 所示。

2) “或”运算

“或”运算也称为逻辑加，其运算符号常用 OR、 \cup 、 \vee 或 $+$ 表示。设 A 和 B 为两个逻辑变量，当且仅当 A 和 B 的取值都为“假”时， A “或” B 的值为“假”；否则 A “或” B 的值为“真”，如表 1-9 所示。

表 1-8 “与”运算规则

A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

表 1-9 “或”运算规则

A	B	$A+B$
0	0	0
0	1	1
1	0	1
1	1	1

3) “非”运算

“非”运算也称为逻辑求反运算，常用 \bar{A} 表示对变量 A 的值求反。其运算规则很简单： $\bar{1}=0$ ， $\bar{0}=1$ 。

4) “异或”运算

常用的逻辑运算还有“异或”运算，又称为半加运算，其运算符号常用 XOR 或 \oplus 表示。设 A 和 B 为两个逻辑变量，当且仅当 A 、 B 的值不同时， A “异或” B 为真。 A “异或” B 的运

算可由前三种基本运算表示, 即 $A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$ 。

2. 常用的逻辑公式

常用的逻辑公式如表 1-10 所示。

表 1-10 常用的逻辑公式

交换律	$A+B=B+A$ $A \cdot B=B \cdot A$	重叠律	$A+A=A$ $A \cdot A=A$
结合律	$A+(B+C)=(A+B)+C$ $A \cdot (B \cdot C) = (A \cdot B) \cdot C$	互补律	$\overline{\overline{A}}+A=1$ $\overline{\overline{A}} \cdot A=0$
		吸收律	$A + \overline{A}B = A + B$
分配律	$A \cdot (B+C)=A \cdot B + A \cdot C$ $A+(B \cdot C) = (A+B) \cdot (A+C)$	0-1 律	$0+A=A$ $0 \cdot A=0$ $1+A=1$ $1 \cdot A=A$
反演律	$\overline{A+B} = \overline{A} \cdot \overline{B}$ $\overline{A \cdot B} = \overline{A} + \overline{B}$	对合律	$\overline{\overline{A}} = A$
其他公式	$AB + \overline{A}\overline{B} = A$ $A + \overline{A}B = A$ $AB + \overline{A}C + BC = AB + \overline{A}C$ $\overline{A \oplus B} = \overline{A} \oplus B = A \oplus \overline{B}$		

3. 逻辑表达式及其化简

1) 逻辑表达式与真值表

逻辑表达式就是用逻辑运算符把逻辑变量(或常量)连接在一起表示某种逻辑关系的表达式。常用表格来描述一个逻辑表达式与其变量之间的关系, 也就是把变量和表达式的各种取值都一一对应列举出来, 称之为真值表。

【例 1-13】 用真值表证明 $AB + \overline{A}\overline{B} = A$ 。

A	B	AB	$\overline{A}\overline{B}$	$AB + \overline{A}\overline{B}$
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	1	0	1

从表中可以看出, 无论 B 取何值, $AB + \overline{A}\overline{B}$ 的值和 A 的值都是相同的, 所以 $AB + \overline{A}\overline{B} = A$ 。

2) 逻辑表达式的化简

利用逻辑运算的规律和一些常用的逻辑恒等式可以对一个逻辑表达式进行化简。

【例 1-14】 化简逻辑表达式 $(\overline{A}BC + \overline{A}\overline{B}C + \overline{A}BC + \overline{A}\overline{B}C + \overline{A}BC + \overline{A}\overline{B}C)$ 。

解: $(\overline{A}BC + \overline{A}\overline{B}C + \overline{A}BC + \overline{A}\overline{B}C + \overline{A}BC + \overline{A}\overline{B}C)$

$$= (\overline{A}(\overline{B}C + C)) + (\overline{A} + A)BC + \overline{A}B(\overline{C} + C) \quad (\text{结合律、分配律})$$

$$\begin{aligned}
 &= (\overline{AB} + BC + \overline{AB}) && (\text{互补律}) \\
 &= ((\overline{A} + A)\overline{B} + BC) && (\text{结合律、分配律}) \\
 &= (\overline{B} + BC) && (\text{互补律}) \\
 &= (\overline{B} + C) && (\text{吸收律})
 \end{aligned}$$

1.2.4 机器数的运算

1. 机器数的加减运算

在计算机中, 可以只设置加法器, 而将减法运算转换为加法运算来实现。

1) 原码加、减法

当两个相同符号的原码数相加时, 只需将数值部分直接相加, 运算结果的符号与两个加数的符号相同。若两个加数的符号相异, 则应进行减法运算。其方法是: 先比较两个数绝对值的大小, 然后用绝对值大者的绝对值减去绝对值小者的绝对值, 结果的符号取绝对值大者的符号。因此, 原码表示的机器数进行减法运算是很麻烦的, 所以在计算机中很少被采用。

2) 补码加、减法

① 补码加法的运算法则是: 和的补码等于补码求和, 即 $[X+Y]_{\#} = [X]_{\#} + [Y]_{\#}$ 。

② 补码减法的方法是: 差的补码等于被减数的补码加上减数取负后的补码。因此, 在补码表示中, 可将减法运算转换为加法运算, 即 $[X-Y]_{\#} = [X]_{\#} + [-Y]_{\#}$ 。

③ 由 $[X]_{\#}$ 求 $[-X]_{\#}$ 的方法是: $[X]_{\#}$ 的各位取反 (包括符号位), 末尾加 1。

【例 1-15】 设二进制整数 $X = +1000100$, $Y = +1110$, 求 $X+Y$ 、 $X-Y$ 的值。

解: 设用 8 位补码表示带符号数据, 由于 X 和 Y 都是正数, 所以 $[X]_{\#} = 01000100$, $[Y]_{\#} = 00001110$, 那么 $[-Y]_{\#} = 11110010$ 。

$ \begin{array}{r} 01000100 \\ + 00001110 \\ \hline 01010010 \end{array} $	$ \begin{array}{r} 01000100 \\ - 00001110 \\ \hline 00110110 \end{array} $	$ \begin{array}{r} 01000100 \\ + 11110010 \\ \hline 00110110 \end{array} $
(a) $[X]_{\#} + [Y]_{\#}$	(b) $X - Y$	(c) $[X]_{\#} + [-Y]_{\#}$

由于 X 和 Y 均是正数, 所以 $X+Y$ 的值就等于 $[X]_{\#} + [Y]_{\#}$, 即 $X+Y = +1010010$; 由于 X 的绝对值大于 Y 的绝对值, 所以 $X-Y$ 的值就等于 $[X]_{\#} + [-Y]_{\#}$, 即 $X-Y = +110110$ 。

【例 1-16】 设二进制整数 $X = +110110$, $Y = -110011$, 求 $X+Y$ 、 $X-Y$ 的值。

解: 设用 8 位补码表示带符号数据, 那么, $[X]_{\#} = 00110110$, $[Y]_{\#} = 11001101$, $[-Y]_{\#} = 00110011$ 。

$\begin{array}{r} 00110110 \\ + 11001101 \\ \hline 110000011 \end{array}$	$\begin{array}{r} 00110110 \\ - 00110011 \\ \hline 00000011 \end{array}$	$\begin{array}{r} 00110110 \\ + 00110011 \\ \hline 01101001 \end{array}$
<p>自然丢弃 \swarrow (a) $[X]_{\text{补}} + [Y]_{\text{补}}$</p>	<p>(b) $X+Y=X- Y$</p>	<p>(c) $[X]_{\text{补}} + [-Y]_{\text{补}}$</p>

由于 X 是正数、 Y 是负数，且 X 的绝对值大于 Y 的绝对值，所以 $X+Y$ 的值就等于 $X-|Y|$ ，也等于 $[X]_{\text{补}} + [Y]_{\text{补}}$ ，即 $+11$ 。因此， $X-Y$ 的值就等于 $X+|Y|$ ，也等于 $[X]_{\text{补}} + [-Y]_{\text{补}}$ ，即 $+1101001$ 。

因此，补码加减运算的规则如下。

(1) 参加运算的操作数用补码表示。

(2) 符号位参加运算。

(3) 若进行相加运算，则两个数的补码直接相加；若进行相减运算，则将减数连同其符号位一起变反加 1 后与被减数相加。

(4) 运算结果用补码表示。

与原码减运算相比，补码减运算的过程要简便得多。在补码加减运算中，符号位和数值位一样参加运算，无须作特殊处理。因此，多数计算机都采用补码加减运算法。

3) 溢出及判定

在确定了运算的字长和数据的表示方法后，数据的范围也就确定了。一旦运算结果超出所能表示的数据范围，就会发生溢出。发生溢出时，运算结果肯定是错误的。

只有当两个同符号的数相加（或者是相异符号数相减）时，运算结果才有可能溢出。

【例 1-17】 设正整数 $X=+1000001$ ， $Y=+1000011$ ，若用 8 位补码表示，则 $[X]_{\text{补}}=01000001$ ， $[Y]_{\text{补}}=01000011$ ，求 $[X+Y]_{\text{补}}$ 。

解：计算 $[X]_{\text{补}} + [Y]_{\text{补}}$

$$\begin{array}{r} 01000001 \\ + 01000011 \\ \hline 10000100 \end{array}$$

两个正数相加的结果为一个负数，结果显然是荒谬的。产生错误的原因就是溢出。

【例 1-18】 设负整数 $X=-1111000$ ， $Y=-10010$ ，字长为 8，则 $[X]_{\text{补}}=10001000$ ， $[Y]_{\text{补}}=11101110$ ，求 $[X+Y]_{\text{补}}$ 。

解：计算 $[X]_{\text{补}} + [Y]_{\text{补}}$

$$\begin{array}{r} 10001000 \\ + 11101110 \\ \hline 101110110 \end{array}$$

两个负数相加，结果为一个正数，显然也是错误的。

常用的溢出检测机制主要有进位判决法和双符号位判决法等如下几种方法。

(1) 双符号位判决法。若采用两位表示符号, 即 00 表示正号、11 表示负号, 则溢出时两个符号位就不一致了, 从而可以判定发生了溢出。

若运算结果两符号分别用 S_2 和 S_1 表示, 则判别溢出的逻辑表示式为 $VF=S_2 \oplus S_1$ 。

【例 1-19】 设正整数 $X=+1000001$, $Y=+1000011$, 若用 8 位补码表示, 则 $[X]_{\text{补}}=00\ 1000001$, $[Y]_{\text{补}}=00\ 1000011$, 求 $[X+Y]_{\text{补}}$ 。

解: 计算 $[X]_{\text{补}}+[Y]_{\text{补}}$

$$\begin{array}{r} 00\ 1000001 \\ +\quad 00\ 1000011 \\ \hline 01\ 0000100 \end{array}$$

式中, 结果的 S_2 和 S_1 不一致, 说明运算过程中有溢出。

(2) 进位判决法。令 C_{n-1} 表示最高数值位向最高位的进位, C_n 表示符号位的进位, 则 $C_{n-1} \oplus C_n = 1$ 表示溢出。

(3) 根据运算结果的符号位和进位标志判别。该方法适用于两同号数求和或异号数求差时判别溢出。根据运算结果的符号位和进位标志, 溢出的逻辑表达式为 $VF=SF \oplus CF$ 。

(4) 根据运算前后的符号位进行判别。若用 X_s 、 Y_s 、 Z_s 分别表示两个操作数及运算结果的符号位, 当两个同号数求和或异号数求差时, 就有可能发生溢出。溢出是否发生可根据运算前后的符号位进行判别, 其逻辑表达式为 $VF=X_s \cdot Y_s \cdot \overline{Z_s} + \overline{X_s} \cdot \overline{Y_s} \cdot Z_s$ 。

2. 机器数的乘除运算

在计算机中实现乘除法运算, 通常有如下三种方式。

① 纯软件方案, 在只有加法器的低档计算机中, 没有乘、除法指令, 乘除运算是用程序来完成的。这种方案的硬件结构简单, 但作乘除运算时速度很慢。

② 在现有的能够完成加减运算的算术逻辑单元 ALU 的基础上, 通过增加少量的实现左、右移位的逻辑电路, 来实现乘除运算。与纯软件方案相比, 这种方案增加硬件不多, 而乘除运算的速度有了较大提高。

③ 设置专用的硬件阵列乘法器 (或除法器), 完成乘 (除) 法运算。该方案需付出较高的硬件代价, 可获得最快的执行速度。

3. 浮点运算

1) 浮点加减运算

设有浮点数 $X=M \times 2^i$, $Y=N \times 2^j$, 求 $X \pm Y$ 的运算过程如下。

(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。在对结果进行右移时,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。舍入处理的方法如下。

① 截断法。将要保留的数据末位右边的数据全都截去,不管数据是0还是1。

② 末位恒1法。将要保留的末位数据恒置1,不管右移丢掉的数据是0还是1。

③ 0舍1入法。舍去的数据为0时,保持末位原始状态。若舍去的数据为1,则将末位加1。这类似于十进制中的四舍五入。但当数据为0.1111...1,即在尾数全为1的特殊情况下,这种舍入会再次产生溢出。遇到这种情况可用硬件判断,并在舍去1时末位不再加1。

(5) 溢出判别。以阶码为准。若阶码溢出(超过最大值),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。

2) 浮点乘除运算

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。

1.3 计算机的基本组成及工作原理

计算机硬件的基本组成包括运算器、控制器、存储器、输入设备和输出设备五大部分。其中,集成在一起的运算器和控制器称为CPU。

运算器(Arithmetic and Logic Unit, ALU)是对数据进行加工处理的部件,它既能完成算术运算又能完成逻辑运算,所以称为算术逻辑单元。

控制器的主要功能是从主存中取出指令并进行分析,以控制计算机的各个部件有条不紊地完成指令的功能。

存储器主要由称为内存和外存的存储部件组成,为了提高整个系统的运行速度,计算机中往往还要设置寄存器、高速缓存等存储器。

输入/输出设备是计算机系统与外界交换信息的装置,一般通过总线和接口将主机与I/O设备有机地组合在一起。

1.3.1 总线的基本概念

1. 总线的定义与分类

总线是连接多个设备的信息传送通道,实际上是一组信号线。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。总线通常分为以下几类。

- 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。
- 系统总线,又称内总线。用于计算机各组成部分(CPU、内存和接口等)的连接。
- 外总线,又称通信总线。用于计算机与外设或计算机与计算机之间的连接或通信。

2. 系统总线

系统总线(System Bus)是微机系统中最重要的总线,对整个计算机系统的性能有重要影响。CPU通过系统总线对存储器的内容进行读写,同样通过系统总线,实现将CPU内数据写入外设,或由外设读入CPU。按照传递信息的功能来分,系统总线分为地址总线、数据总线和控制总线。

系统总线的性能指标主要有带宽、位宽和工作频率等。

系统总线的带宽指的是单位时间内总线上传送的数据量,即每秒钟传送的最大稳态数据传输率。系统总线的位宽指的是总线能同时传送的二进制数据的位数,或数据总线的位数,即32位、64位等总线宽度的概念。总线的位宽越宽,每秒钟数据传输率越大,总线的带宽越宽。总线的工作时钟频率以MHz为单位,工作频率越高,总线工作速度越快,总线带宽越宽。

它们之间的关系是:总线的带宽=总线的工作频率*总线的位宽/8。

常见的传统系统总线有ISA、EISA、PCI/AGP等。

(1) ISA总线。ISA是工业标准总线,它可以与更早的PC总线兼容。ISA总线是在PC总线62个插座信号的基础上,再扩充另一个36个信号的插座而构成的。

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),±5V、±12V电源和地线等。

(2) EISA总线。EISA总线是在ISA总线的基础上发展起来的32位总线。该总线定义了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。总线传输率可达33Mb/s。EISA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;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总线有两种标准: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。PCI 总线的发展遇到了并行总线的技术瓶颈。

(4) AGP (Accelerated Graphics Port, 图形加速端口) 是 Intel 公司推出的图形显示卡专用局部总线, AGP 总线直接与主板的北桥芯片相连, 且通过该接口让显示芯片与系统主内存直接相连, 避免了窄带宽的 PCI 总线形成的系统瓶颈, 增加 3D 图形数据传输速度, 同时在显存不足的情况下还可以调用系统主内存。所以它拥有很高的传输速率, 这是 PCI 等总线无法与其相比拟的。

AGP 标准工作在 32 位总线时有 66MHz 和 133MHz 两种工作频率, 最高数据传输率为 266Mb/s 和 533Mb/s, 而 PCI 总线理论上的最大传输率仅为 133Mb/s。目前最高规格的 AGP 8X 模式下, 数据传输速度达到了 2.1Gb/s。

(5) PCI Express 总线。PCI Express (以下简称 PCI-E) 有 PCI Express X1、X2、X4、X8、X12、X16 和 X32 等多种规格, 采用点对点串行连接, 与 PCI 以及更早期的计算机总线的共享并行架构不同, 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。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-E 的双单工连接能提供更高的传输速率和质量。

目前, PCI-E X1 和 PCI-E X16 已成为 PCI-E 主流规格, 同时很多芯片组厂商在南桥芯片当中添加对 PCI-E X1 的支持, 在北桥芯片当中添加对 PCI-E X16 的支持。除去提供极高数据传输带宽之外, PCI-E 因为采用串行数据包方式传递数据, 所以 PCI-E 接口每个针脚可以获得比传统 I/O 标准更多的带宽, 这样就可以降低 PCI-E 设备生产成本和体积。另外, PCI-E 也支持高阶电源管理, 支持热插拔, 支持数据同步传输, 为优先传输数据进行带宽优化。

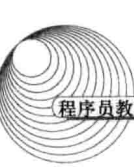
3. 外总线

外总线的标准有七八十种之多, 此处简单介绍几种。

(1) RS-232C。RS-232C 是一种串行外总线, 其主要特点是: 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, 具有较好的抗干扰性。

(2) RS-485。RS-485 采用平衡发送和差分接收, 因此具有抑制共模干扰的能力。要求通信距离为几十米到上千米时, 广泛采用 RS-485 串行总线标准。

(3) SCSI。小型计算机系统接口 (Small Computer System Interface, SCSI) 是一种并行外总线, 广泛用于连接软硬磁盘、光盘和扫描仪等。该接口总线早期是 8 位的, 后来发展到 16



位、32 位。Ultra320 SCSI 单通道的数据传输速率最大可达 320Mb/s, 如果采用双通道 SCSI 控制器可以达到 640Mb/s。

(4) USB。通用串行总线是 1994 年底由 Compaq、IBM 和 Microsoft 等众多公司联合提出的, 目前得到广泛应用, USB 接口已经成为计算机硬件系统的基本配置。USB 1.0 有两种传送速率: 低速为 1.5Mb/s, 高速为 12Mb/s。USB2.0 的传送速率为 480Mb/s。USB 最大的优点是支持即插即用并支持热插拔。

(5) IEEE-1394。IEEE-1394 也是一种串行数据传输协议, 支持即插即用并支持热插拔, 与 USB 相比速度更快, 主要用于音频、视频等数据的传输。IEEE-1394 理论上可以连接 64 台设备, 传输速率有 100Mb/s、400Mb/s、800Mb/s、1600Mb/s、3.2Gb/s 等规格。

1.3.2 中央处理单元

CPU 是 Central Process Unit (中央处理单元) 的缩写, 简称为微处理器 (Microprocessor), 常被称为处理器 (Processor)。

1. CPU 的功能

CPU 是计算机工作的核心部件, 用于控制并协调各个部件, 其基本功能如下所述。

(1) 指令控制。CPU 通过执行指令来控制程序的执行顺序, 这是 CPU 的重要职能。

(2) 操作控制。一条指令功能的实现需要若干操作信号来完成, 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) 时序控制。CPU 通过时序电路产生的时钟信号进行定时, 以控制各种操作按照指定的时序进行。

(4) 数据处理。在 CPU 的控制下完成对数据的加工处理是其最根本的任务。

另外, CPU 还需要对内部或外部的中断 (异常) 以及 DMA 请求做出响应, 进行相应的处理。

2. CPU 的组成

CPU 主要由运算器、控制器 (Control Unit, CU)、寄存器组和内部总线组成, 如图 1-2 所示。

1) 运算器

运算器 (简称为 ALU) 主要完成算术运算和逻辑运算, 实现对数据的加工与处理。不同的计算机的运算器结构不同, 但基本都包括算术和逻辑运算单元、累加器 (AC)、状态字寄存器 (PSW)、寄存器组及多路转换器等逻辑部件。

在运算过程中, 寄存器组用于暂存操作数或数据的地址。标志寄存器也称为状态寄存器,

用于存放算术、逻辑运算过程中产生的状态信息。

累加器是运算器中的主要寄存器之一，用于暂存运算结果以及向 ALU 提供运算对象。

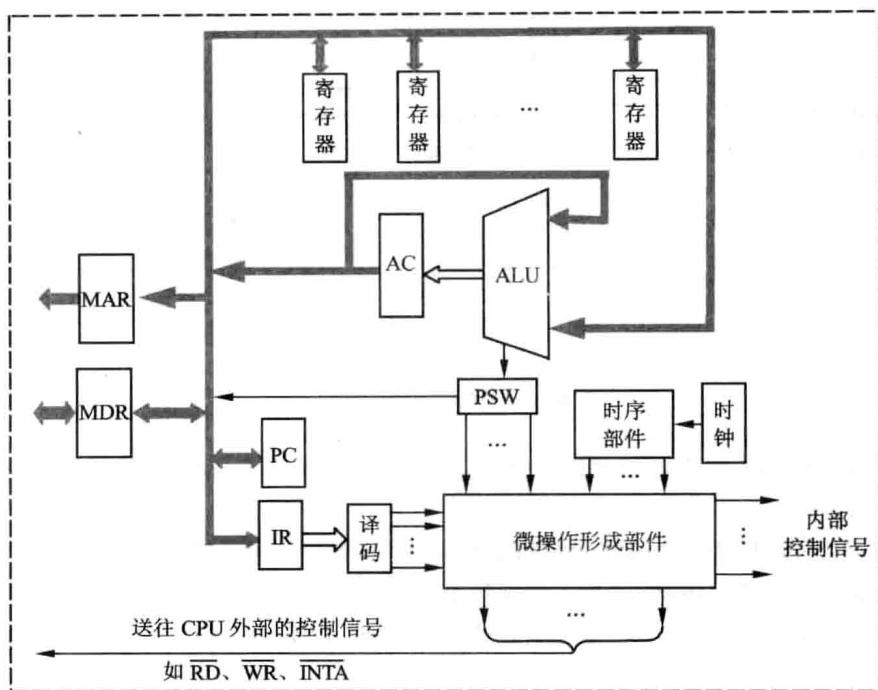


图 1-2 CPU 主要组成框图

2) 控制器

控制器的主要功能是从内存中取出指令，并指出下一条指令在内存中的位置，将取出的指令送入指令寄存器，启动指令译码器对指令进行分析，最后发出相应的控制信号和定时信息，控制和协调计算机的各个部件有条不紊地工作，以完成指令所规定的操作。

控制器由程序计数器（简称 PC）、指令寄存器（IR）、指令译码器、状态字寄存器（PSW）、时序产生器和微操作信号发生器组成，如图 1-3 所示。

控制器各部分的主要作用如下。

- 程序计数器：当程序顺序执行时，每取出一条指令，PC 内容自动增加一个值，指向下一条要取的指令。当程序出现转移时，则将转移地址送入 PC，然后由 PC 指出新的指令地址。
- 指令寄存器：存放正在执行的指令。

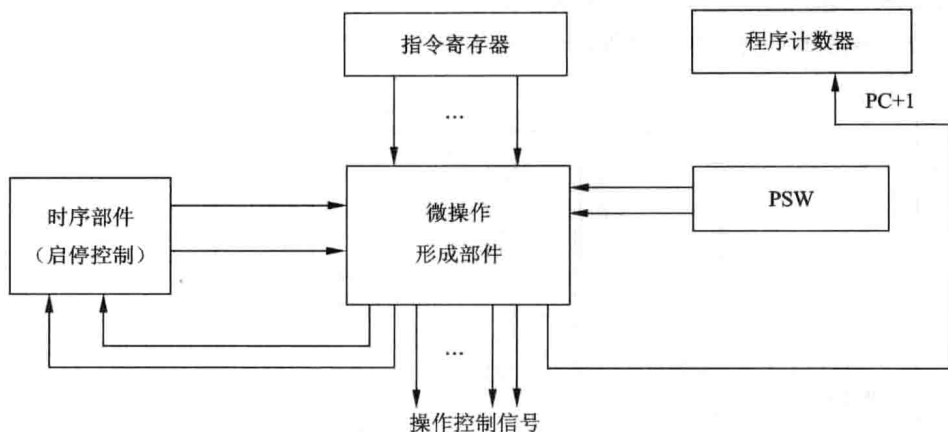


图 1-3 控制器组成框图

- 指令译码器：对现行指令进行分析，确定指令类型和指令所要完成的操作以及寻址方式。
- 时序部件：用于产生时序脉冲和节拍电位以控制计算机各部件有序地工作。
- 状态字寄存器（PSW）：用于保存指令执行完成后产生的条件码，例如运算是否有溢出，结果为正还是为负，是否有进位等。此外，PSW 还保存中断和系统工作状态等信息。
- 微操作信号发生器：根据指令提供的操作信号、时序产生器提供的时序信号，以及各功能部件反馈的状态信号等综合成特定的操作序列，从而完成对指令的执行控制。

控制器的作用是控制整个计算机的各个部件有条不紊地工作，它的基本功能就是从内存取指令和执行指令。

执行指令的过程分为如下几个步骤。

(1) 取指令。控制器首先按程序计数器所指出的指令地址从内存中取出一条指令。

(2) 指令译码。将指令的操作码部分送指令译码器进行分析，然后根据指令的功能向有关部件发出控制命令。

(3) 按指令操作码执行。根据指令译码器分析指令产生的操作控制命令以及程序状态字寄存器的状态，控制微操作形成部件产生一系列 CPU 内部的控制信号和输出到 CPU 外部的控制信号。在这一系列控制信号的控制下，实现指令的具体功能。

(4) 形成下一条指令地址。若非转移类指令，则修改程序计数器的内容；若是转移类指令，则根据转移条件修改程序计数器的内容。

通过上述步骤逐一执行一系列指令,就使计算机能够按照这一系列指令组成的程序的要求自动完成各项任务。

3) 寄存器组

寄存器是 CPU 中的一个重要组成部分,它是 CPU 内部的临时存储单元。寄存器既可以用来存放数据和地址,也可以存放控制信息或 CPU 工作时的状态。在 CPU 中增加寄存器的数量,可以使 CPU 把执行程序时所需的数据尽可能地放在寄存器中,从而减少访问内存的次数,提高其运行速度。但是,寄存器的数目也不能太多,除了增加成本外,寄存器地址编码增加还会增加指令的长度。CPU 中的寄存器通常分为存放数据的寄存器、存放地址的寄存器、存放控制信息的寄存器、存放状态信息的寄存器和其他寄存器等类型。

- 累加器 (accumulator): 累加器是一个数据寄存器,在运算过程中暂时存放操作数和中间运算结果,不能用于长时间地保存一个数据。
- 通用寄存器组: 通用寄存器组是 CPU 中的一组工作寄存器,运算时用于暂存操作数或地址。在程序中使用通用寄存器可以减少访问内存的次数,提高运算速度。
- 标志寄存器: 标志寄存器也称为状态字寄存器,用于记录运算中产生的标志信息。状态寄存器中的每一位单独使用,称为标志位。标志位的取值反映了 ALU 当前的工作状态,可以作为条件转移指令的转移条件。典型的标志位有以下几种。
 - ◆ 进位标志位 (C): 当运算结果最高位产生进位时将该位置 1。
 - ◆ 零标志位 (Z): 当运算结果为零时置“1”。
 - ◆ 符号标志位 (S): 当运算结果为负时置“1”。
 - ◆ 溢出标志位 (V): 当运算结果产生溢出时置“1”。
 - ◆ 奇偶标志位 (P): 当运算结果中“1”的个数为偶数时置“1”。
- 指令寄存器: 指令寄存器用于存放正在执行的指令,指令从内存取出后送入指令寄存器。其操作码部分经指令译码器送微操作信号发生器,其地址码部分指明参加运算的操作数的地址形成方式。在指令执行过程中,指令寄存器中的内容保持不变。
- 数据缓冲寄存器 (MDR): 用来暂时存放由内存储器读出的一条指令或一个数据字;反之,当向内存存入一个数据字时,也暂时将它们存放在数据缓冲寄存器中。
- 地址寄存器 (MAR): 用来保存当前 CPU 所访问的内存单元的地址。由于在内存和 CPU 之间存在着操作速度上的差别,所以必须使用地址寄存器来保持地址信息,直到内存的读/写操作完成为止。
- 其他寄存器: 根据 CPU 的结构特点还有一些其他寄存器,例如堆栈指示器、变址寄存器和段地址寄存器等。

4) 内部总线

CPU 内部总线将运算器、控制器和寄存器组等连接在一起。

3. 双核和多核处理器

推动微处理器性能不断提高的因素主要有两个：半导体工艺技术的飞速进步和体系结构的不断发展。半导体工艺技术的每一次进步都为微处理器体系结构的研究提出了新的问题，开辟了新的领域；体系结构的进展又在半导体工艺技术发展的基础上进一步提高了微处理器的性能。这两个因素是相互影响，相互促进的。多核的出现是技术发展和应用需求的必然产物。

CPU 中最重要的组成部分称为内核或核心 (Die)，核心是由单晶硅以一定的生产工艺制造出来的，CPU 中所有的计算、接收/存储命令、处理数据都由核心执行。

双核处理器是指在一个处理器上集成两个运算核心，从而提高计算能力。“双核”的概念最早是由 IBM、HP、Sun 等支持 RISC (精简指令系统) 架构的高端服务器厂商提出的。双核心处理器技术的引入是提高处理器性能的有效方法。因为处理器实际性能是处理器在每个时钟周期内所能处理的指令数的总量，因此增加一个内核，处理器每个时钟周期内可执行的指令数将增加一倍。

CPU 制造商 AMD 和 Intel 的双核技术在物理结构上有很大不同。AMD 将两个内核做在一个晶元上，通过直连架构进行连接，集成度更高。Intel 则是将放在不同晶元上的两个内核封装在一起，因此人们将 Intel 的方案称为“双芯”，而将 AMD 的方案称为“双核”。

由于仅仅提高单核芯片的速度会产生过多热量且无法带来相应的性能改善，而且即便是没有热量问题，其性价比也令人难以接受，速度稍快的处理器价格要高很多。因此，英特尔工程师们开发了多核芯片，即在一个处理器中集成两个或多个完整的计算引擎 (内核)，采用分治策略，通过划分任务，线程应用能够充分利用多个执行内核，并可在特定的时间内执行更多任务。

1.3.3 存储系统

1. 存储器的分类

(1) 按存储器所处的位置可分为内存和外存。

(2) 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。

- 磁存储器：用磁性介质做成的，如磁芯、磁泡、磁膜、磁鼓、磁带及磁盘等。
- 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。
- 光存储器：如光盘 (Optical Disk) 存储器。

(3) 按工作方式可分为读写存储器和只读存储器。

- 读写存储器 (Random Access Memory, RAM): 既能读取数据也能存入数据的存储器。这类存储器的特点是它存储信息的易失性, 即一旦去掉存储器的供电电源, 则存储器所存信息也随之丢失。
 - 只读存储器: 只读存储器所存信息是非易失的, 也就是它存储的信息去掉供电电源后不会丢失, 当电源恢复后它所存储的信息依然存在。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。
 - ◆ 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, 故这种存储器又称为掩膜 ROM。这类存储器一般用于存放系统程序 BIOS 和用于微程序控制。
 - ◆ 可编程的只读存储器 (Programmable Read Only Memory, PROM): 其内容可以由用户一次性地写入, 写入后不能再修改。
 - ◆ 可擦除可编程只读存储器 (Erasable Programmable Read Only Memory, EPROM): 其内容既可以读出, 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。
 - ◆ 电擦除的可编程只读存储器 (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。
 - ◆ 闪速存储器 (Flash Memory): 简称闪存, 其特性介于 EPROM 和 EEPROM 之间, 类似于 EEPROM, 闪存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) 按寻址方式分类可分为随机存储器、顺序存储器和直接存储器。
- 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。
 - 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。
 - 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。

2. 存储系统的层次结构

不同的存储器, 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。一般情

况下, 计算机的存储体系结构可用图 1-4 所示的三级结构进行描述。其中高速缓存 (Cache) 的速度最快, 其次是主存储器 (MM), 处于最底层的辅助存储器 (外存储器) 速度最慢。若将 CPU 内部的寄存器也看作是存储器的一个层次, 则可将存储系统分为 4 层结构。

存储系统中采用高速缓存可显著地提高计算机系统的工作速度。Cache 在功能上并不是必需的部件, 因此在一些简单的计算机中, 没有设置高速缓存, 那么这种计算机的存储体系就由主存和辅存两级存储器构成。

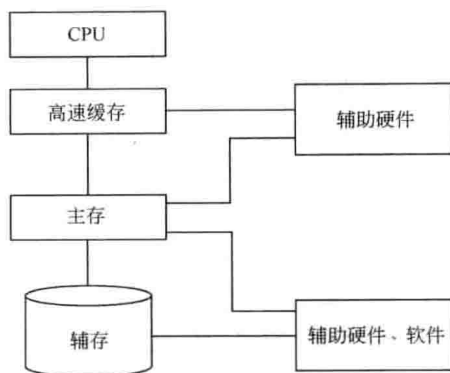


图 1-4 存储器层次结构示意图

3. 主存储器

主存储器简称为主存、内存, 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, 以便向 CPU 提供信息。相对于外存, 其特点是容量小速度快。

1) 主存的种类

主存一般由 RAM 和 ROM 这两种工作方式的存储器组成, 其绝大部分存储空间由 RAM 构成。计算机系统中常见的 SDRAM (Synchronous Dynamic Random Access Memory, 同步动态随机存取存储器) 的发展经历了四代, 分别是第一代 SDR SDRAM、第二代 DDR SDRAM (双倍速率 SDRAM)、第三代 DDR2 SDRAM (更高的工作频率) 和第四代 DDR3 SDRAM (更低的工作电压)。

2) 主存的组成

主存储器主要由存储体、控制线路、地址寄存器、数据寄存器和地址译码电路等部分组成, 如图 1-5 所示。

- 地址寄存器: 用来存放由地址总线提供的将要访问的存储单元的地址码。地址寄存器的位数 N 决定了其可寻址的存储单元的个数 M , 即 $M=2^N$ 。
- 数据寄存器: 用来存放要写入存储体的数据或从存储体中读取的数据。
- 存储体: 存放程序和数据的存储空间。
- 译码电路: 根据存放在地址寄存器中的地址码, 在存储体中找到相应的存储单元。
- 控制线路: 根据读写命令, 控制主存储器的各部分协作完成相应的操作。

对主存的操作分为读操作和写操作。读出时, CPU 把要读取的存储单元的地址送入地址寄存器, 经地址译码线路分析后选中主存的对应存储单元, 在控制线路的作用下, 将被选存储单

元的内容读取到数据寄存器中, 读操作完成; 写入时, CPU 将要写入的存储单元的地址送入地址寄存器, 经地址译码线路分析后选中主存的对应存储单元, 在控制线路的作用下, 将数据寄存器的内容写入到指定的存储单元中, 写操作完成。

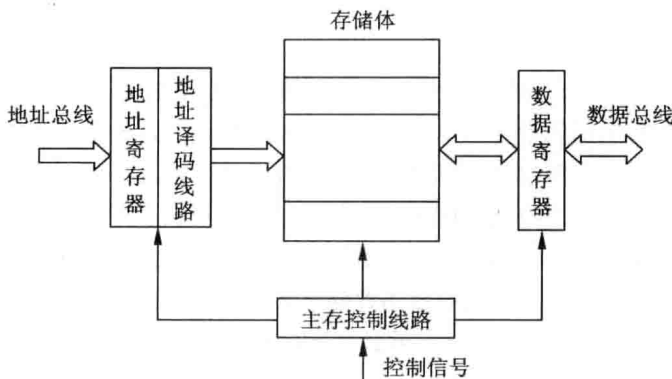


图 1-5 主存储器组成框图

3) 主存性能指标

(1) 内存容量。内存容量常以字节数表示, 目前计算机系统常见的内存容量为 512MB、1GB 和 2GB 等, 计算机系统中常见的容量单位为 KB、MB、GB、TB, 其关系如下:

$$1\text{KB}=1024\text{B}=1024\text{B}$$

$$1\text{MB}=1024\text{KB}=1\,048\,576\text{B}$$

$$1\text{GB}=1024\text{MB}=1\,073\,741\,824\text{B}$$

$$1\text{TB}=1024\text{GB}=1\,099\,511\,627\,777\text{B}$$

更大的容量单位为 PB、EB 等。

$$1\text{PB}=1024\text{TB}=1\,125\,899\,906\,842\,624\text{B}$$

$$1\text{EB}=1024\text{PB}=1\,152\,921\,504\,606\,846\,976\text{B}$$

(2) 存储时间。存储时间是指存储器从接到读或写的命令起, 到读写操作完成为止所需要的时间, 分为取数时间和存取周期。取数时间是指存储器从接收读出命令到被读出信息稳定在数据寄存器的输出端为止的时间间隔; 存取周期是指两次独立的存或取操作之间所需的最短时间。半导体存储器的存取周期一般为 60~100ns, 是指连续两次存储器访问的最小时间间隔。

(3) 带宽 (Band Width)。带宽是指存储器的数据传送速率, 即每秒传送的数据位数, 记作 B_m 。假设存储器传送的数据宽度为 W 位 (即一个存储周期中读取或写入的位数), 那么

$$B_m = \frac{W}{T_m} \quad (\text{位/秒})。$$

(4) 可靠性。存储器的可靠性用平均故障间隔时间 MTBF 来衡量。MTBF 可以理解为两次故障之间的平均时间间隔。MTBF 越长, 表示可靠性越高, 即保持正确工作能力越强。

4. 高速缓存

1) 高速缓存的特点

高速缓存 (Cache) 的出现首先是由于 CPU 的速度和性能提高很快而主存速度较低且价格高, 其次就是程序执行的局部性特点, 也就是即在一段时间内, 整个程序的执行仅限于程序中的某一部分。相应地, 执行所访问的存储空间也局限于某个内存区域。具体表现为时间局部性和空间局部性。时间局部性是指如果程序中的某条指令一旦执行, 则不久之后该指令可能再次被执行; 如果某数据被访问, 则不久之后该数据可能再次被访问。空间局部性是指一旦程序访问了某个存储单元, 则不久之后, 其附近的存储单元也将被访问。因此, 将速度比较快而容量有限的静态存储器芯片构成 Cache, 以尽可能发挥 CPU 的高速度, 并且用硬件来实现 Cache 的全部功能。

Cache 的主要特点为:

- (1) Cache 位于 CPU 和主存之间, 容量较小, 一般在几千字节到几兆字节之间。
- (2) 速度一般比主存快 5~10 倍, 由快速半导体存储器制成。
- (3) 其内容是主存内容的副本 (频繁使用的 RAM 位置的内容及这些数据项的存储地址), 对程序员来说是透明的。

2) Cache 的组成

Cache 主要由两部分组成: 控制部分和存储器部分。控制部分的功能是判断 CPU 要访问的数据是否在 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, 若是读取操作, 则从主存中读取数据, 并按照确定的替换原则把该数据写入 Cache 存储器中; 若是写入操作, 则将数据写入主存即可。

5. 外存储器

外存储器用来存放暂时不用的程序和数据, 外存上的信息以文件的形式存储, 相对于内存, 外存的容量大、速度慢。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方可访问。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。下面介绍几种常用的外存储器。

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。

(1) 磁盘存储器的构成。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作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。控制器接受主机发来的命令,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多台驱动器。接口是主机和磁盘存储器之间的连接部件。

(2) 硬盘。硬盘中可记录信息的磁介质表面叫做记录面。每一个记录面上都分布着若干同心的闭合圆环,称为磁道。数据就记录在磁道上。使用时要对磁道进行编号,按照半径递减的次序从外到里编号,最外一圈为0道,往内道号依次增加。最内圈的位密度称为最大位密度。为了便于记录信息,磁盘上的每个磁道又分成若干段,每一段称为一个扇区。例如,每个磁道分为16段、32段等。

硬盘按盘片是否固定、磁头是否移动等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)。按盘片的直径,分为14英寸、8英寸、5.25英寸、3.5英寸等几种。

一个硬盘驱动器内可装有多片盘片,组成盘片组,每个盘片可以提供两个记录面,每个记录面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。文件存储在硬盘上时会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。

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。

硬磁盘的主要技术指标如下。

- 道密度。为了减少干扰,磁道之间要保持一定的间隔,沿磁盘半径方向,单位长度内磁道的数目称为道密度。常用的道密度单位是:道/毫米,或道/英寸。
- 位密度。位密度是指在磁道圆周上单位长度内存储的二进制位的个数。常用的位密度单位是:b/mm,或b/in。为了简化电路设计,规定每个磁道上记录的位数是相同的。由于磁盘中各个磁道的圆周长度不同,因此不同磁道上的位密度是不一样的,越靠近盘心的磁道位密度就越高,并用最内圈磁道的位密度来定义磁盘的位密度。
- 存储容量。存储容量是指整个磁盘所能存储的二进制位信息的总量。磁盘的容量有非格式化容量和格式化容量之分。一般情况下,磁盘容量是指格式化容量。

非格式化容量=位密度×内圈磁道周长×每个记录面上的磁道数×记录面数

格式化容量=每个扇区的字节数×每道的扇区数×每个记录面的磁道数×记录面数

- 平均存取时间。平均存取时间是指从发出读写命令开始,磁头从某一位置移动到指定位置并开始读写数据所需时间。它包括寻道时间和等待时间,是两者之和。

- 寻道时间 (seek time)。寻道时间是指磁头移动到目标磁道 (或柱面) 所需要的时间, 由驱动器的性能决定, 是个常数, 由厂家给出。
- 等待时间 (rotational latency)。等待时间是指等待读写的扇区旋转到磁头下方所用的时间, 一般选用磁道旋转一周所用时间的一半作为平均等待时间。可见, 提高磁盘转速可缩短这个时间。
- 数据传输率。数据传输率是指磁头找到数据的地址后, 单位时间内写入或读出的字节数。

数据传输率=每个扇区的字节数×每道扇区数×磁盘的转速

2) 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存储装置。

(1) 光盘存储器的类型。根据性能和用途, 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。只读型光盘上的信息由生产厂家预先用激光在盘片上蚀刻, 用户不能改写。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, 它利用激光照射引起介质的可逆性物理变化来记录信息。

(2) 光盘存储器的组成及特点。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) 光盘存储器与磁盘存储器的比较如下。

① 光盘是非接触式读写信息, 比磁盘的头盘间距大 1 万倍左右, 所以光盘的耐用性高, 使用寿命长。

② 光盘可靠性高, 对使用环境要求不高, 机械振动上的问题较少, 不需要特殊的防震与除尘设备。

③ 光盘的记录密度为磁盘的 10~100 倍, 但读取时间比磁盘慢, 其读/写速度只有磁盘的几分之一。

④ 光盘易于更换, 可做成自动换盘装置。

3) USB 移动硬盘和 USB 闪存盘

USB 移动硬盘的容量大, 支持热插拔, 即插即用, 可像使用本地硬盘一样存取文件。当工作完成后, 停止设备, 拔下数据线即可。USB 移动硬盘通常由一块 2.5 英寸的笔记本硬盘或普通 3.5 英寸硬盘与相应大小的硬盘盒组成。硬盘盒的作用是将硬盘的数据接口标准 (通常是 IDE

接口) 转换为 USB 接口标准。USB 移动硬盘的容量等于其内部硬盘的容量, 传输速率则与采用的 USB 接口标准相关。

USB 闪存盘又称为优盘, 是使用闪存 (Flash Memory) 作为存储介质的一种半导体存储设备, 采用 USB 接口标准。闪存盘具备比软盘容量更大 (1GB 和 2GB 是目前常见的优盘容量)、速度更快、体积更小、寿命更长等优点, 而且容量不断增加、价格不断下降。根据不同的使用要求, 优盘还具有基本型、加密型和启动型等类型, 在移动存储领域已经取代了软盘。

1.3.4 输入/输出技术

输入/输出 (Input/Output, I/O) 系统是计算机与外界进行数据交换的通道。主机和 I/O 设备间不是简单地用系统总线连接起来就可以, 还需要进行控制。随着计算机技术的发展, I/O 设备的种类越来越多, 其控制方式各不相同, 很难做到由 CPU 来统一控制和管理, 各设备的数据格式和传输率差异较大, 所以需要有一个 I/O 系统负责协调和控制 CPU、存储器和各种外部设备之间的数据通信。

1. 接口的功能及分类

1) 接口

广义上讲, 接口是指两个相对独立子系统之间的相连部分, 也常被称为界面。由于主机与各种 I/O 设备的相对独立性, 它们一般是无法直接相连的, 必须经过一个转换机构。用于连接主机与 I/O 设备的这个转换机构就是 I/O 接口电路, 简称为 I/O 接口, 如图 1-6 所示。

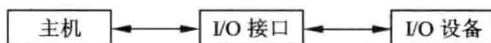


图 1-6 主机、I/O 接口、I/O 设备之间的关系

显然, I/O 接口并非仅仅完成设备间物理上的连接, 一般来说它还应具有下述主要功能。

(1) 地址译码功能。由于一个计算机系统中连接有多台 I/O 设备, 相应的接口也有多个, 为了能够进行区别和选择, 必须给它们分配不同的地址码, 这与存储器中对存储单元编址的道理是一样的。

(2) 在主机与 I/O 设备间交换数据、控制命令及状态信息等。

(3) 支持主机采用程序查询、中断和 DMA 等访问方式。

(4) 提供主机和 I/O 设备所需的缓冲、暂存、驱动能力, 满足一定的负载要求和时序要求。

(5) 进行数据的类型、格式等方面的转换。

2) 接口的分类

(1) 按数据传送的格式可分为并行接口和串行接口。

并行接口采用并行传送方式,即一次把一个字节(或一个字)的所有位同时输入或输出,同时(并行)传送若干位。并行接口一般指主机与 I/O 设备之间、接口与 I/O 设备之间均以并行方式传送数据。

串行接口采用串行传送方式,数据的所有位按顺序逐位输入或输出。一般情况下,接口与 I/O 设备之间采用串行传送方式,而串行接口与主机之间则采用并行方式。

一般来说,并行接口适用于传输距离较近、速度相对较高的场合,接口电路相对简单;串行接口则适用于传输距离较远、速度相对较低的场合。

(2) 按主机访问 I/O 设备的控制方式,可分为程序查询接口、中断接口、DMA 接口,以及更复杂一些的通道控制器、I/O 处理机等。

(3) 按时序控制方式可分为同步接口和异步接口。

还可从其他角度进行分类,这里不再详述。

需要说明的是,一个完整的 I/O 接口不仅包括一些硬件电路,也可能包括相关的软件驱动程序模块。这些软件模块有的放在接口的 ROM 中,有的放在主机系统上的 ROM 中,也有的存储在外存中,需要时再装入内存执行。

2. 主机与外设间的连接方式

在不同的计算机系统中,主机与 I/O 设备之间的连接模式可能不同,常见的有总线型、星型、通道方式和 I/O 处理机方式等,其中总线方式是互连的基本方式,也是其他连接模式的基础。

总线是一组能为多个部件分时共享的信息传送线,用来连接多个部件并为之提供信息交换通路。所谓共享,是指连接到总线上的所有部件都可通过它传递信息;分时性是指某一时刻只允许一个部件将数据发送到总线上。因此,共享是利用分时实现的。

总线不仅是一组信号线,还包括相关的协议。由于要实现分时共享,所以必须制定相应的规则,称之为总线协议。总线协议一般包括信号线定义、数据格式、时序关系、信号电平和控制逻辑等。

3. I/O 接口的编址方式

尽管在微型计算机中存在着许多种内存与接口地址的编址方法,但最常见的还是以下两种方式。

(1) 与内存单元统一编址。将 I/O 接口中有关的寄存器或存储部件看作存储器单元,与主存中的存储单元统一编址。这样,内存地址和接口地址统一在一个公共的地址空间里,对 I/O

接口的访问就如同对主存单元的访问一样。

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,这就增强了对接口的操作功能,也无须设置专门的 I/O 操作指令。其缺点是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,会导致内存地址不连续。另外,由于对内存操作的指令和对接口操作的指令不加区分,读程序时就需根据参数定义表仔细加以辨认,才能区分指令是对内存操作还是对接口操作。

(2) I/O 接口单独编址。通过设置单独的 I/O 地址空间,为接口中的有关寄存器或存储部件分配地址码,需要设置专门的 I/O 指令进行访问。这种编址方式的优点是不占用主存的地址空间,访问主存的指令和访问接口的指令不同,在程序中很容易使用和辨认。

4. CPU 与外设之间交换数据的方式

1) 直接程序控制

直接程序控制方式的主要特点是 CPU 直接通过 I/O 指令对 I/O 接口进行访问操作,主机与外设之间交换信息的每个步骤均在程序中表示出来,整个的输入/输出过程是由 CPU 执行程序来完成的,具体实现时可分为两种方式:立即程序传送方式和程序查询方式。

(1) 立即程序传送方式。在这种方式下, I/O 接口总是准备好接收来自主机的数据,或随时准备向主机输入数据, CPU 无须查看接口的状态,就执行输入/输出指令进行数据传送。这种传送方式又称为无条件传送或同步传送。

(2) 程序查询方式。在这种方式下, CPU 通过执行程序查询外设的状态,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。

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。

程序查询方式的优点是简单且容易实现,缺点是降低了 CPU 的利用率, CPU 的大量时间消耗在查询外设的状态上;对外部的突发事件无法做出实时响应。

2) 中断方式

中断是计算机系统中的一个重要概念。

(1) 中断的定义。

中断是这样一个过程:在 CPU 执行程序的过程中,由于某一个外部的或 CPU 内部事件的发生,使 CPU 暂时中止正在执行的程序,转去处理这一事件,当事件处理完毕后又回到原先被中止的程序,接着中止前的状态继续向下执行。这一过程就称为中断。

引起中断的事件就称为中断源。若中断是由 CPU 内部发生的事件引起的,这类中断源就称为内部中断源;若中断是由 CPU 外部的的事件引起的,则称为外部中断源。

(2) 中断方式下的数据传送。

当 I/O 接口准备好接收数据或准备好向 CPU 传送数据时,就发出中断信号通知 CPU。对中断信号进行确认后,CPU 保存正在执行的程序的现场,转而执行提前设置好的 I/O 中断服务程序,完成一次数据传送的处理。这样,CPU 就不需要主动查询外设的状态,在等待数据期间可以执行其他程序,从而提高了 CPU 的利用率。采用中断方式管理 I/O 设备,CPU 和外设可以并行地工作。

虽然中断方式可以提高 CPU 的利用率,能处理随机事件和实时任务,但一次中断处理过程需要经历保存现场、中断处理和恢复现场等阶段,需要执行若干条指令才能处理一次中断事件。因此,这种方式无法满足高速的批量数据传送要求,所以引入 DMA 方式。

3) 直接存储器存取方式

直接存储器存取 (Direct Memory Access, DMA) 方式的基本思想是通过硬件控制实现主存与 I/O 设备间的直接数据传送,数据的传送过程由 DMA 控制器 (DMAC) 进行控制,不需要 CPU 的干预。在 DMA 方式下,由 CPU 启动传送过程,即向设备发出“传送一块数据”的命令,在传送过程结束时,DMAC 通过中断方式通知 CPU 进行一些后续处理工作。

DMA 方式简化了 CPU 对数据传送的控制,提高了主机与外设并行工作的程度,实现了快速外设和主存之间成批的数据传送,使系统的效率明显提高。但 DMA 方式也有局限性,由于 DMA 控制器只能控制简单的数据传送操作,因此对外设的管理和某些控制操作仍由 CPU 承担,因此,在外设数量较多、输入/输出频繁的大、中型机中,通常设置通道,使 CPU 摆脱管理和控制外设的沉重负担。

4) 通道控制方式

通道是一种专用控制器,它通过执行通道程序进行 I/O 操作的管理,为主机与 I/O 设备提供一种数据传输通道。用通道指令编制的程序存放在存储器中,当需要进行 I/O 操作时,CPU 只要按约定格式准备好命令和数据,然后启动通道即可,通道则执行相应的通道程序,完成所要求的操作。用通道程序也可完成较复杂的 I/O 管理和预处理,从而在很大程度上将 CPU 从繁重的 I/O 管理工作中解脱出来,提高了系统的效率。

随着通道的进一步发展,其结构越来越复杂,功能逐渐变得通用,发展为现在广泛使用的输入/输出处理器 (I/O Processor, IOP),这里不再赘述。

1.4 指令系统简介

CPU 所能完成的操作是由其执行的指令决定的,这些指令称为机器指令。CPU 能执行的所有机器指令的集合称为该 CPU 的指令系统。指令系统设计的好坏、功能的强弱,对整个计算机会产生很大的影响,指令系统是计算机中硬件与软件之间的接口。

1. 指令格式

指令是指挥计算机完成各种操作的基本命令。一般来说,一条指令包括两个基本组成部分:操作码和地址码。基本格式如下:

操作码字段 OP	操作数地址码字段 Addr
----------	---------------

操作码说明指令的功能及操作性质。地址码用来指出指令的操作对象,它指出操作数或操作数的地址及指令执行结果的地址。

操作码用二进制编码来表示,该字段越长,所能表示的指令就越多。若操作码的长度为 n ,则可表示的指令为 2^n 条。根据指令中地址码的数量,指令格式分为以下几种。

(1) 三地址指令格式。三地址指令格式为:

OP	A_1	A_2	A_3
----	-------	-------	-------

其中,OP 为操作码; A_1 、 A_2 、 A_3 分别是源操作数 1、源操作数 2 和目的操作数的地址。该类指令实现的操作是: $(A_1) OP (A_2) \rightarrow (A_3)$ 。

(2) 二地址指令格式。二地址指令格式为:

OP	A_1	A_2
----	-------	-------

二地址指令实现如下操作: $(A_1) OP (A_2) \rightarrow (A_1)$ 。

(3) 一地址指令格式。一地址指令格式为:

OP	A
----	---

在一地址指令中,只给出一个操作数的地址。若操作是针对一个操作数的指令,其操作为: $OP (A) \rightarrow (A)$;若操作是针对两个操作数的一地址指令,通常另一个操作数是隐含的(另一个操作数在累加器 AC 中),其操作为: $(AC) OP (A) \rightarrow (AC)$ 。

(4) 零地址指令格式。零地址指令只有操作码,不含操作数地址。其格式为:

OP

零地址指令在操作上分两种情况:一种是无操作数的控制操作,如空操作指令 NOP、停机指令 HLT 等;另一种是隐含有操作数,在指令中不体现。

2. 寻址方式

寻址方式就是如何对指令中的地址字段进行解释,以获得操作数的方法或获得程序转移地

址的方法,操作数的位置可能在指令中、寄存器中、存储器中或 I/O 端口中。常用的寻址方式有立即寻址、直接寻址、寄存器寻址、寄存器间接寻址等。

(1) 立即寻址。操作数就包含在指令中。在形成指令的机器代码形式时,立即数就跟在指令操作码的后面,取出指令时即可得到操作数。例如,8086 指令系统中,指令 `ADD AX, 3048H` 的功能是将寄存器 `AX` 中的内容和十六进制数值 `3048` 相加,结果送入 `AX` 寄存器。指令中的 `3048H` 是一个操作数,采用立即寻址方式取得该操作数。再如,ARM 指令系统中,`MOV R0, #0xFF000` 就是将 `0xFF000` 存储到 `R0` 中,其中十六进制数值 `FF000` 就是立即数。

(2) 直接寻址。操作数存放在内存单元中,指令中直接给出操作数所在存储单元的地址。例如,8086 指令系统中,指令 `ADD AX, [2000H]` 的功能是将寄存器 `AX` 中的内容和数据段中偏移地址为 `2000H` 的存储单元中的内容相加,结果送入寄存器 `AX`。存储单元 `2000H` 的内容是操作数。

(3) 寄存器寻址。操作数存放在某一寄存器中,指令中给出存放操作数的寄存器名。例如指令 `ADD AX, 3048H`,其中第一个操作数放在寄存器 `AX` 中,取得第一个操作数的寻址方式为寄存器寻址。再如,ARM 指令系统中,`MOV R0, #0xFF000` 中的 `R0` 采用寄存器寻址。

(4) 寄存器间接寻址。操作数存放在内存单元中,操作数所在存储单元的地址在某个寄存器中。例如,8086 指令系统中,指令 `ADD AX, [BX]` 的功能是从寄存器 `BX` 中取得第二个操作数在数据段的偏移地址,然后访问内存读取操作数,再与寄存器 `AX` 中的数据相加,结果存入 `AX` 寄存器。

(5) 间接寻址。操作数存放在内存单元中。这种寻址方式下,指令中给出操作数地址的地址,取出操作数要进行两次访问内存的操作。

(6) 基址寻址。操作数存放在内存单元中。指令中操作数地址码给出基址寄存器和一个偏移量(可正可负),操作数的有效地址为基址寄存器的内容加上偏移量。例如,8086 指令系统中,指令 `MOV AX, [BX+100H]` 的源操作数的地址为数据段中偏移量为 `BX` 的内容加上 `100H`。再如,指令 `MOV [BP-08H], DI` 中目的操作数的地址为堆栈段中偏移量为 `BP` 的内容减去 `08H`。

(7) 变址寻址。操作数存放在内存单元中。操作数的有效地址等于变址寄存器的内容加偏移量。例如,指令 `ADD AX, [DI+100H]`,其中第二个操作数采用变址寻址方式, `DI` 是变址寄存器。

3. 指令的种类

尽管为不同 CPU 所设计的指令系统各不相同,但基本上所有的指令系统都包含以下几种类型的指令。

1) 数据传送类指令

这类指令将数据从一个地方传送到另一个地方, 主要包括如下指令。

(1) 数据传送指令。这类指令中一般有两个操作数地址: 源操作数地址和目的操作数地址。

传送方式一般包括:

- ① 将立即数传送到寄存器。
- ② 将立即数传送到存储单元。
- ③ 将一个寄存器的内容传送到另一个寄存器。
- ④ 将寄存器的内容传送到存储单元。
- ⑤ 将数据从一个存储单元传送到另一个存储单元。
- ⑥ 将数据由存储单元传送到寄存器。

(2) 数据交换指令。主要包括:

- ① 寄存器与寄存器之间的数据交换。
- ② 存储器单元与寄存器之间的数据交换。
- ③ 存储器单元与存储器单元之间的数据交换。

(3) 堆栈操作指令。主要包括压入堆栈指令和弹出堆栈指令。

2) 输入/输出 (I/O) 类指令

这类指令用于实现主机与外设间的信息传送, 包括数据的输入/输出、主机向外设发出控制命令以及输入外设的状态。通常有三种方法来实现输入/输出。

3) 算术运算类指令

这类指令支持 CPU 实现加、减、乘、除等算术运算。主要包括加法、减法、乘法、除法、求补、加 1、减 1 和比较等指令。

4) 逻辑运算指令

这类指令支持 CPU 实现各种逻辑运算。一般的 CPU 都会设置逻辑运算指令, 主要包括与、或、异或、取反等指令。

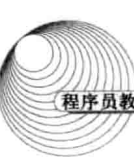
5) 移位操作指令

根据移位的方向, 当操作数的各位顺序向左移动一位称为左移, 同样, 当操作数的各位顺序向右移动一位称为右移。移位指令一般可分为算术移位、逻辑移位和循环移位三种类型。

(1) 算术移位。算术移位指令对带符号操作数进行移位, 其执行过程如图 1-7 所示。



图 1-7 算术移位操作示意图



左移时从最低位依次向最高位移动，最低位补 0，最高位移入“进位”位 C 中。右移时从最高位向最低位依序移动，最低位移入“进位”位 C，而最高位（即符号位）保持不变。

(2) 逻辑移位。逻辑移位指令对无符号操作数进行移位，其执行过程如图 1-8 所示。

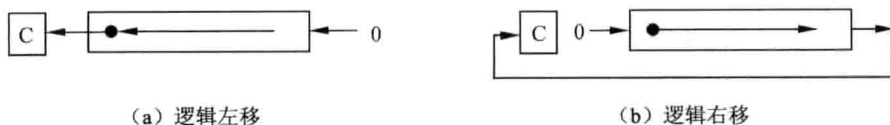


图 1-8 逻辑移位操作示意图

逻辑左移指令的执行过程与算术左移相同，低位向高位移动，最低位补 0；而逻辑右移指令与算术右移指令不同，是用 0 填补最高位。

(3) 循环移位。循环移位指令分为不带进位的循环移位和带进位的循环移位两种。

不带进位的循环左移指令每做一次移位，总是将操作数的最高位移入进位标志位 C 中，并且还将操作数的最高位移入最低位，从而构成一个环，如图 1-9 (a) 所示。

不带进位的循环右移指令每做一次移位，总是将操作数的最低位移入进位标志位 C 中，另外还将操作数的最低位移入最高位，从而构成一个环，如图 1-9 (b) 所示。



图 1-9 不带进位的循环移位

带进位的循环移位是将进位标志位 C 包含在内进行循环移位，如图 1-10 所示。



图 1-10 带进位的循环移位

6) 程序控制类指令

程序控制类指令用于改变指令执行的顺序和控制流的方向，主要包括以下几种。

(1) 跳转指令。跳转指令又可分为无条件跳转指令和条件跳转指令。无条件跳转指令直接使程序的控制流转移到指定的目标，而条件跳转指令则需先判断指令中规定的跳转条件是否满足，当满足规定的条件时，则使程序的控制流转移到指定的目标；否则不跳转。

(2) 子程序调用和返回指令。在程序设计时,通常把完成某种独立功能、多次重复使用的代码段定义为一个子程序。编程时,根据需要可调用子程序完成相应的功能。子程序调用指令使控制流从一个程序段跳转到另一个程序段。子程序执行结束后,必须返回到调用它的程序段,因此子程序返回指令根据所记录的返回地址将控制流转回调用它的程序段。

(3) 陷阱指令。陷阱是一种意外事件所引起的中断。通常陷阱指令是隐含的,不提供给用户使用。当意外事件引发中断后,由 CPU 自动执行。但也有些指令系统提供了用户可以使用的陷阱指令,使用户可以用它进行系统调用,从而可以利用操作系统提供的各项功能。

7) 串操作类指令

现代计算机经常需要处理大量的字符串信息。因此,一般都会设置字符串操作指令。常见的串操作指令包括串传送指令、串比较指令、串搜索指令、串替换指令、串转换指令和串抽取指令等。

8) 处理机控制类指令

这类指令用于对 CPU 实施控制。例如,对 PSW 中的标志实现置位或清零、停机指令、开中断指令、关中断指令、空操作指令等。

9) 数据转换类指令

有的指令系统中还会设置数据转换指令。例如,将十进制数转换为二进制数、二进制数转换为十进制数、定点数转换为浮点数、浮点数转换为定点数等指令。

第2章 操作系统基础知识

2.1 操作系统概述

操作系统（Operating System, OS）是计算机系统中必不可少的核心系统软件，其他软件（如编辑程序、汇编程序、编译程序、数据库管理系统等系统软件，以及大量的应用软件）是建立在操作系统的基础上，并在操作系统的统一管理和支持下运行。操作系统是用户与计算机之间的桥梁，用户可以通过操作系统提供的功能访问计算机系统软硬件资源。

1. 操作系统的作用、特征与功能

操作系统的作用是通过资源管理提高计算机系统的效率，改善人机界面为用户提供友好的工作环境。操作系统能有效地组织和管理系统中的各种软、硬件资源，合理地组织计算机系统工作流程，控制程序的执行，并且向用户提供一个良好的工作环境和友好的接口。

操作系统的4个特征是并发性、共享性、虚拟性和不确定性。从资源管理的观点来看，操作系统的功能可分为5大部分：处理机管理、文件管理、存储管理、设备管理和作业管理。

（1）处理机管理。实质上是对处理机的执行“时间”进行管理，采用多道程序等技术将CPU的时间合理地分配给每个任务。主要包括进程控制、进程同步、进程通信和进程调度。

（2）文件管理。主要包括文件存储空间管理、目录管理、文件的读写管理和存取控制。

（3）存储管理。是对主存储器“空间”进行管理，主要包括存储分配与回收、存储保护、地址映射（变换）和主存扩充。

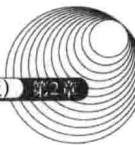
（4）设备管理。实质是对硬件设备的管理，包括对输入/输出设备的分配、启动、完成和回收。

（5）作业管理。包括任务、界面管理、人机交互、图形界面、语音控制和虚拟现实等。

操作系统提供系统命令级的接口，供用户组织和控制自己的作业运行。操作系统还提供编程一级接口，供用户程序和系统程序调用操作系统功能。

2. 操作系统的类型

操作系统分为批处理操作系统、分时操作系统、实时操作系统、网络操作系统、分布式操作系统、微机操作系统和嵌入式操作系统等。



1) 批处理操作系统

批处理操作系统分为单道批处理和多道批处理。

单道批处理操作系统是一种早期的操作系统,该系统可以提交多个作业,“单道”的含义是指一次只有一个作业装入内存执行。作业由用户程序、数据和作业说明书(作业控制语言)三部分组成。当一个作业运行结束后,随即自动调入同批的下一个作业,从而节省了作业之间的人工干预时间,提高了资源的利用率。

多道批处理操作系统允许多个作业装入内存执行,在任意一个时刻,作业都处于开始点和终止点之间。每当运行中的一个作业由于输入/输出操作需要调用外部设备时,就把 CPU 交给另一道等待运行的作业,从而将主机与外部设备的工作由串行改变为并行,进一步避免了因主机等待外设完成任务而浪费宝贵的 CPU 时间。多道批处理系统主要有三个特点:多道、宏观上并行运行、微观上串行运行。

2) 分时操作系统

在分时操作系统中,一个计算机系统与多个终端设备连接。分时操作系统是将 CPU 的工作时间划分为许多很短的时间片,轮流为各个终端的用户服务。例如,一个带 20 个终端的分时系统,若每个用户每次分配一个 50 ms 的时间片,则每隔 1s 即可为所有的用户服务一遍。因此,尽管各个终端上的作业是断续地运行的,但由于操作系统每次对用户程序都能做出及时的响应,因此用户感觉整个系统均归其一人占用。

分时系统主要有 4 个特点:多路性、独立性、交互性和及时性。

3) 实时操作系统

实时是指计算机对于外来信息能够以足够快的速度进行处理,并在被控对象允许的时间范围内做出快速反应。实时系统对交互能力要求不高,但要求可靠性有保障。为了提高系统的响应时间,对随机发生的外部事件应及时做出响应并对其进行处理。

实时系统分为实时控制系统和实时信息处理系统。实时控制系统主要用于生产过程的自动控制,例如数据自动采集、武器控制、火炮自动控制、飞机自动驾驶和导弹的制导系统等。实时信息处理系统主要用于实时信息处理,如飞机订票系统、情报检索系统等。实时系统与分时系统除了应用的环境不同,主要有以下三点区别。

(1) 系统的设计目标不同。分时系统是设计成一个多用户的通用系统,交互能力强;而实时系统大都是专用系统。

(2) 交互性的强弱不同。分时系统是多用户的通用系统,交互能力强;而实时系统是专用系统,仅允许操作并访问有限的专用程序,不能随便修改,且交互能力差。

(3) 响应时间的敏感程度不同。分时系统是以用户能接收的等待时间为系统的设计依据,而实时系统是以被测物体所能接受的延迟为系统设计依据。因此,实时系统对响应时间的敏感

程度更强。

4) 网络操作系统

网络操作系统是使联网计算机能方便而有效地共享网络资源, 为网络用户提供各种服务的软件和有关协议的集合。因此, 网络操作系统的功能主要包括高效、可靠的网络通信; 对网络中共享资源 (在 LAN 中有硬盘、打印机等) 的有效管理; 提供电子邮件、文件传输、共享硬盘和打印机等服务; 网络安全管理; 提供互操作能力。

主要的网络操作系统有 UNIX、Linux 和各种版本的 Windows Server 系统。

5) 分布式操作系统

分布式计算机系统是由多个分散的计算机经连接而成的计算机系统, 系统中的计算机无主、次之分, 任意两台计算机可以通过通信交换信息。通常, 为分布式计算机系统配置的操作系统称为分布式操作系统。

分布式操作系统能直接对系统中各类资源进行动态分配和调度、任务划分、信息传输协调工作, 并为用户提供一个统一的界面, 标准的接口, 用户通过这一界面实现所需要的操作和使用系统资源, 使系统中若干台计算机相互协作完成共同的任务, 有效控制和协调诸任务的并行执行, 并向系统提供统一、有效的接口的软件集合。

分布式操作系统是网络操作系统的更高级形式, 它保持网络系统所拥有的全部功能, 同时又有透明性、可靠性和高性能等特性。

6) 微机操作系统

常用的微机操作系统有各种版本的 Windows、Linux 等。Windows 操作系统是 Microsoft 公司开发的图形用户界面、多任务、多线程操作系统。Linux 是一套免费使用和自由传播的类 UNIX 操作系统, 由世界各地成千上万的程序员设计和实现, 其目的是建立不受任何商品化软件版权制约的、全世界都能自由使用的 UNIX 兼容产品。

7) 嵌入式操作系统

嵌入式操作系统运行在嵌入式智能芯片环境中, 对整个智能芯片以及它所操作、控制的各种部件装置等资源进行统一协调、处理、指挥和控制。其主要特点:

(1) 微型化。从性能和成本角度考虑, 希望占用资源和系统代码量少, 如内存少、字长短、运行速度有限、能源少 (用微小型电池)。

(2) 可定制。从减少成本和缩短研发周期考虑, 要求嵌入式操作系统能运行在不同的微处理器平台上, 能针对硬件变化进行结构与功能上的配置, 以满足不同应用需要。

(3) 实时性。嵌入式操作系统主要应用于过程控制、数据采集、传输通信、多媒体信息及关键要害领域需要迅速响应的场合, 所以对实时性要求高。

(4) 可靠性。系统构件、模块和体系结构必须达到应有的可靠性, 对关键要害应用还要提

供容错和防故障措施。

(5) 易移植性。为了提高系统的易移植性,通常采用硬件抽象层(Hardware Abstraction Level, HAL)和板级支撑包(Board Support Package, BSP)的底层设计技术。

嵌入式实时操作系统有很多,常见的有 VxWorks、 μ Clinux、PalmOS、WindowsCE、 μ C/OS-II 和 eCos 等。

3. 促使操作系统发展的因素

促使操作系统发展的因素主要有三个方面:第一,硬件的不断地升级与新的硬件产品出现,需要操作系统提供更多、更复杂的支持;第二,新的服务需求,操作系统为了满足系统管理者和用户需求,需要不断扩大服务范围;第三,修补操作系统自身的错误,操作系统在运行的过程中其自身的错误也会不断地被发现,因此需要不断地修补操作系统自身的错误(即所谓的“补丁”)。需要说明的是,在修补的过程中也可能会产生新的错误。

2.2 处理机管理

处理机管理也称为进程管理,其核心是如何合理地分配处理机的时间,提高系统的效率。在计算机系统中有多个并发执行的程序,采用“程序”这个静态的概念已经不能描述程序执行时动态变化的过程,所以引入了“进程”。

2.2.1 基本概念

1. 程序执行时的特征

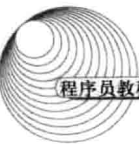
前趋图是一个有向无循环图,由结点和有向边组成,结点代表各程序段的操作,而结点间的有向边表示两个程序段操作之间存在的前趋关系(\rightarrow)。程序段 P_i 和 P_j 的前趋关系表示成 $P_i \rightarrow P_j$, 其中 P_i 是 P_j 的前趋, P_j 是 P_i 的后继,其含义是 P_i 执行结束后 P_j 才能执行。例如,图 2-1 为三个程序段,其中输入是计算的前驱,计算是输出的前驱。



图 2-1 三个结点的前驱图

程序顺序执行时的主要特征如下。

- (1) 顺序性。程序的各程序段严格按照规定的顺序执行。
- (2) 封闭性。程序运行时系统内的资源只受该程序控制而改变,执行结果不受外界因素的影响。
- (3) 可再现性。只要程序执行环境和初始条件相同,程序多次执行的结果相同。



若在计算机系统中采用多道程序设计技术，则主存中的多道程序可处于并发执行状态。对于图 2-1 中有三个程序段的作业，虽然其中有前趋关系的各程序段不能在 CPU 和输入/输出各部件上并行执行，但是同一个作业内没有前趋关系的程序段或不同作业的程序段可以分别在 CPU 和各输入/输出部件上并行执行。

例如，某系统中有一个 CPU、一台输入设备和一台输出设备，每个作业具有三个程序段：输入 I_i 、计算 C_i 和输出 P_i ($i=1,2,3$)。图 2-2 为三个作业的各程序段并发执行的前驱图。从图 2-2 中可以看出， I_2 与 C_1 并行执行， I_3 、 C_2 与 P_1 并行执行， C_3 与 P_2 并行执行。其中， I_2 、 I_3 受到 I_1 的间接制约， C_2 、 C_3 受到 C_1 的间接制约， P_2 、 P_3 受到 P_1 的间接制约；而 C_1 、 P_1 受到 I_1 的直接制约， C_2 、 P_2 受到 I_2 的直接制约……

程序并发执行时的主要特征如下。

- (1) 失去了程序的封闭性。
- (2) 程序和机器执行程序的活动不再一一对应。

- (3) 并发程序间具有相互制约性。

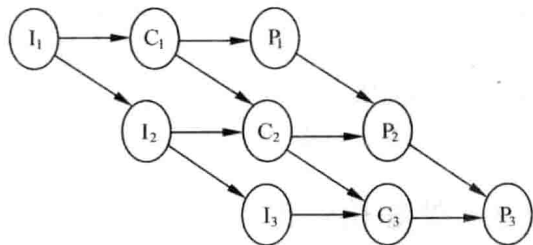


图 2-2 程序并发执行的前驱图

2. 进程的组成

进程 (Process) 是程序的一次执行。进程通常由程序、数据和进程控制块 (Process Control Block, PCB) 组成。其中，程序部分描述了进程需要完成的功能。假如，一个程序能被多个进程同时共享执行，那么这部分就应该以可再入码的形式编制，它是程序执行时不可修改的部分。数据部分包括程序执行时所需的数据及工作区，这部分只能为一个进程所专用，是进程的可修改部分。进程控制块是进程存在的唯一标志，其主要内容如表 2-1 所示。

表 2-1 PCB 的内容

信 息	含 义
进程标识符	标明系统中的各个进程
状态	说明进程当前的状态
位置信息	指明程序及数据在内存或外存的物理位置
控制信息	参数、信号量和消息等
队列指针	链接同一状态的进程
优先级	进程调度的依据
现场保护区	将处理机的现场保护到该区域以便再次调度时能继续正确运行

3. 进程的状态及其状态间的切换

1) 三态模型

在多道程序系统中, 进程的运行是走走停停, 在处理器上交替运行, 状态也不不断地发生变化, 因此进程一般有三种基本状态: 运行、就绪和阻塞, 如图 2-3 所示, 也称三态模型。

- 运行: 当一个进程在处理器上运行时, 称该进程处于运行状态。显然, 对于单处理机系统, 处于运行状态的进程只有一个。
- 就绪: 一个进程获得了除处理机外的一切所需资源, 一旦得到处理机即可运行, 则称此进程处于就绪状态。
- 阻塞: 也称等待或睡眠状态, 一个进程正在等待某一事件发生 (例如, 请求 I/O 而等待 I/O 完成等) 而暂时停止运行, 这时即使把处理机分配给该进程, 它也无法运行, 故称该进程处于阻塞状态。

2) 五态模型

事实上, 对于一个实际的系统, 进程的状态及其转换将更复杂。例如, 引入新建态和终止态构成了五态模型, 如图 2-4 所示。

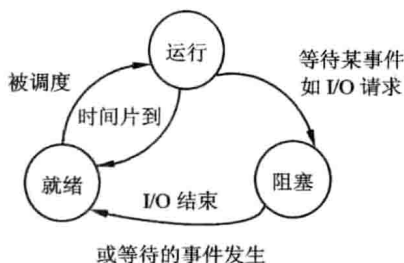


图 2-3 进程三态模型

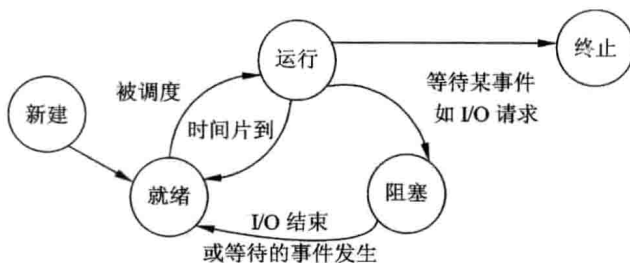


图 2-4 进程五态模型

其中, 新建态对应于进程刚刚被创建且没有被提交的状态, 并等待系统完成创建进程的所有必要信息。因为创建进程时分为两个阶段, 第一个阶段为一个新进程创建必要的管理信息, 第二个阶段让该进程进入就绪状态。由于有了新建态可以增加调度的灵活性, 即操作系统可以根据系统的性能和内存容量的限制推迟新建态进程的提交。

类似地, 进程的终止也可分为两个阶段, 第一个阶段等待操作系统进行善后处理, 第二个阶段释放内存。

提供各种书籍的pd电子版代找服务，如果你找不到自己想要的书的pdf电子版，我们可以帮您找到，如有需要，请联系QQ1779903665.

PDF代找说明：

本人可以帮助你找到你要的PDF电子书，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%都带书签索引和目录，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QQ1779903665。

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，大部分我都可以找到，而且每本100%带书签索引目录。因PDF电子书都有版权，请不要随意传播，如果您有经济购买能力，请尽量购买正版。

声明：本人只提供代找服务，每本100%索引书签和目录，因寻找pdf电子书有一定难度，仅收取代找费用。如因PDF产生的版权纠纷，与本人无关，我们仅仅只是帮助你寻找到你要的pdf而已。