高等院校计算机专业教材

计算机原理

张道光　编著　　周 兵　主审

西安电子科技大学出版社

西安

内　容　简　介

本书围绕计算机系统的组成，详细论述了各个部分的组成及工作原理，结合新的计算机技术应用，内容更具有创新性。全书共 11 章，其内容包括：计算机系统概论、数据的表示与校验、运算方法与运算器、存储器系统、指令系统、控制器、接口与输入输出、外围设备、总线、并行处理与互连网络、多处理机与机群系统。

本书可作为高等院校计算机及相关专业本科生的教材，也可以作为计算机专业研究生及计算机工程技术人员的参考书。

前　言

计算机技术的迅速发展对社会的进步带来了巨大的推动作用并产生了深远的影响，它正潜移默化地改变着人们的生产方式、工作方式、生活方式和学习方式，掌握计算机基本知识和应用技术已成为当今社会人们的迫切要求和参与社会竞争的必要条件，是衡量个人素质的重要标志之一。

《计算机原理》是计算机科学与技术学科一门重要的专业基础课程，在计算机硬件课程群中起着承上启下的作用。为适应计算机的发展和教学改革的需要，编者结合多年的教学实践经验，首先进行内容“优化”，将计算机硬件课程群中相近的内容进行整合，使该教材既能体现原课程教材的主体面貌，又赋予了新的结构内容；其次突出内容“关联”，将内容之间的联系勾画出来，展现在读者面前，以达到“纲举目张”之目的；最后采用“案例教学”，针对目前学生普遍存在的“对设计无从下手，对问题束手无策”等现象，采用“案例教学”是培养学生分析解决、解决问题行之有效方法。

全书共 11 章，第 1 章计算机概述；第 2 章计算机中数据的表示；第 3 章运算方法与运算器；第 4 章指令系统；第 5 章存储器系统；第 6 章控制器；第 7 章接口与输入输出；第 8 章外围设备；第 9 章总线；第 10 章并行处理与互连网络；第 11 章多处理机与机群系统。

本书由张道光主编，并编写了第 2 章、第 4 章、第 5 章；第 7 章和第 8 章由副主编刘卫光编写；第 10 章由副主编夏冰编写；第 1 章和第 3 章由王鼎媛编写；第 6 章由张书钦编写；第 9 章和第 11 章由董跃钧编写。全书由张道光、周兵统稿。

本书承蒙周兵教授审阅了全部书稿，并提出了许多宝贵意见，在此表示诚挚的感谢。由于时间仓促及编者水平有限，书中难免存在疏漏和错误，敬请广大读者批评指正。

编　者

2008-10-25

目录

第 1 章　计算机系统概论 1

1.1　计算机的发展与应用 1
1.1.1 计算机发展阶段和发展趋势 1
1.1.2 计算机的应用 3
1.2　计算机系统的组成 4
1.2.1 计算机系统的硬件 4
1.2.2 计算机系统的软件 4
1.2.3 计算机系统的层次结构 5
1.3　计算机系统结构 6
1.3.1 传统冯·诺依曼计算机系统结构 6
1.3.2 现代计算机系统结构 6
关　联 7
习　题 8

第 2 章　计算机中数据的表示 9

2.1　进位计数制及其之间的转换 9
2.1.1 进位计数制 9
2.1.2 进位计数制之间的相互转换 10
2.2　定点数的表示 11
2.2.1 符号的表示 11
2.2.2 小数点的表示 12
2.2.3 几种机器数形式 12
2.3　浮点数的表示 16
2.3.1 浮点数的格式 16
2.3.2 浮点数的规格化 17
2.3.3 浮点数的表示范围 18
2.3.4 浮点数的机器零 18
2.4　非数值数据的表示 19
2.4.1 ASCII 码与字符串 19
2.4.2 BCD 码与十进制数串 19
2.5　数据校验码 21
2.5.1 奇偶校验码 21
2.5.2 海明校验码 22
2.5.3 循环冗余校验码 24
关　联 26
习　题 27

第 3 章　运算方法与运算器 30

3.1　定点数的算术运算与实现 30
3.1.1 定点数加减运算 30
3.1.2 定点数乘法运算 35

3.1.3 定点数除法运算.....40
3.2 逻辑运算和移位操作.....43
3.2.1 逻辑运算.....43
3.2.2 移位操作.....44
3.3 定点运算器.....44
3.3.1 算术逻辑单元.....44
3.3.2 定点运算器的基本结构.....48
3.3.3 定点运算器模型.....49
3.4 浮点数的算术运算与浮点运算器.....50
3.4.1 浮点数的加减运算.....50
3.4.2 浮点数的乘法和除法运算.....51
3.4.3 浮点运算器.....53
关 联.....54
习 题.....55
第 4 章 存储器系统.....59
4.1 存储器概述.....59
4.1.1 存储器分类.....59
4.1.2 存储器系统结构.....60
4.1.3 主存储器的技术指标.....60
4.2 半导体存储器.....61
4.2.1 半导体存储器分类.....61
4.2.2 存储元电路.....62
4.2.3 存储器芯片.....64
4.2.4 存储器的扩展与应用.....73
4.2.5 并行存储器.....78
4.3 高速缓冲存储器.....80
4.3.1 Cache 基本原理.....80
4.3.2 Cache 的结构.....81
4.3.3 Cache 的读写过程.....85
4.4 虚拟存储器.....85
4.4.1 概述.....85
4.4.2 页式虚拟存储器.....86
4.4.3 段式虚拟存储器.....87
4.4.4 段页式虚拟存储器.....88
4.4.5 替换算法.....89
关 联.....90
习 题.....91
第 5 章 指令系统.....94
5.1 指令系统与性能.....94
5.1.1 指令与指令系统.....94
5.1.2 指令系统的性能.....94
5.2 机器指令.....95
5.2.1 机器指令的格式.....95

5.2.2 指令字的长度.....95
5.2.3 机器指令的分类.....95
5.3 操作码的编码方法.....96
5.3.1 定长编码.....96
5.3.2 变长编码.....97
5.4 地址码的寻址方式.....99
5.4.1 指令寻址方式.....99
5.4.2 操作数寻址方式.....100
5.5 典型的指令系统.....103
5.5.1 复杂指令系统.....103
5.5.2 精简指令系统.....105
关 联.....107
习 题.....108
第 6 章 控制器.....110
6.1 CPU 功能和组成.....110
6.1.1 CPU 的功能.....110
6.1.2 CPU 的基本组成.....111
6.2 控制器的时序系统和控制方式.....113
6.2.1 有关周期的基本概念.....113
6.2.2 时序信号与体制.....113
6.2.3 时序信号发生器.....114
6.2.4 控制方式.....117
6.3 指令流程图.....118
6.3.1 五类典型指令的指令周期分析.....118
6.3.2 指令周期流程.....125
6.4 微程序控制器.....127
6.4.1 基本概念.....127
6.4.2 微程序控制器基本原理.....129
6.4.3 微程序设计.....130
6.4.4 微程序设计举例.....134
6.4.5 微程序控制器设计步骤.....137
6.5 组合逻辑控制器.....138
6.6 门阵列控制器.....140
6.6.1 可编程逻辑阵列 PLA.....141
6.6.2 基本思想.....141
6.7 流水线处理技术.....141
6.7.1 指令执行方式.....141
6.7.2 流水线的分类.....143
6.7.3 线性流水线.....143
6.7.4 流水线中的相关问题.....145
6.8 多媒体技术.....146
6.9 典型 CPU 简介.....147
6.9.1 8086CPU.....147
6.9.2 Pentium 微处理器.....148

关　联.....150
习　题.....151
第 7 章　接口与输入输出.....155
7.1　接口概述.....155
7.1.1 接口的功能与组成.....155
7.1.2 I/O 端口的编址方式.....156
7.1.3 I/O 端口地址的译码.....156
7.2　输入输出方式.....158
7.2.1 程序控制传送方式.....158
7.2.2 中断方式.....161
7.2.3 直接存储器方式.....165
7.2.4 通道方式.....170
关　联.....171
习　题.....172
第 8 章　外围设备.....176
8.1　外围设备概述.....176
8.1.1 外围设备的概念.....176
8.1.2 外围设备的分类.....177
8.1.3 外围设备的功能.....177
8.2　输入设备.....177
8.2.1 键盘.....178
8.2.2 鼠标.....178
8.2.3 其他输入设备.....179
8.3　输出设备.....181
8.3.1 显示器.....181
8.3.2 打印机.....183
8.4　外存储设备.....186
8.4.1 磁表面存储器的原理.....186
8.4.2 磁记录方式.....187
8.4.3 硬磁盘存储器.....189
8.4.4 光盘存储设备.....191
8.4.4 闪存.....193
关　联.....194
习　题.....195
第 9 章　总线.....197
9.1　总线技术概述.....197
9.2　总线系统结构.....199
9.2.1 总线通道组成.....199
9.2.2 总线结构类型.....199
9.3　总线信息传送方式及定时.....201
9.3.1 总线信息传送方式：.....201
9.3.2 总线定时.....202

9.4 总线的仲裁.....202
9.4.1 集中式仲裁.....202
9.4.2 分布式仲裁.....204
9.5 计算机中的总线.....204
9.5.1 内部总线.....204
9.5.2 外部通信总线.....207
9.6 新一代总线.....211
9.6.1 PCI Express 总线.....211
9.6.2 USB 总线.....213
关 联.....216
习 题.....217
第 10 章 并行处理和互连网络.....219
10.1 并行处理的概念.....219
10.1.1 并行性.....219
11.1.2 并行性的等级和分类.....219
10.1.3 开发并行性的途径.....220
10.2 并行处理机基本结构.....221
10.2.1 并行处理机的两种典型结构.....221
11.2.2 并行处理机的特点.....222
10.3 SIMD 计算机基本结构.....223
10.3.1 SIMD 计算机模型.....223
10.3.2 SIMD 计算机发展过程.....223
10.3.3 Illiac IV 计算机.....224
10.3.4 Burroughs BSP 计算机.....226
10.3.5 CM-2 计算机.....228
10.4 SIMD 计算机的应用.....230
10.4.1 计算模型及有限差分.....230
10.4.2 阵列处理机的几种基本算法.....231
10.5 互连网络的概念.....231
10.5.1 基本概念和作用.....232
10.5.2 主要特性和性能参数.....233
10.5.3 互连函数.....236
10.6 静态互连网络.....239
10.6.1 静态互连网络结构.....239
10.6.2 静态互连网络特性.....241
10.7 动态互连网络.....242
10.7.1 动态互连网络的互连形式.....242
10.7.2 动态网络互连方式的比较.....244
10.7.3 多级互连网络.....245
10.8 互连网络的消息传递机制.....245
10.8.1 消息寻径.....245
10.8.2 死锁和虚拟通道.....249
10.8.3 单播方式下的寻径.....252
10.8.4 广播方式下的寻径.....254

关　联.....254
习　题.....255
第 11 章　多处理机与机群系统.....256
11.1　多处理机系统特点与分类.....256
11.1.1 基本结构.....256
11.1.2 多处理机系统特点.....257
11.1.3 多处理机系统的 Cache 一致性问题.....257
11.2　多处理机软件和典型的多处理机系统.....259
11.2.1 并行算法.....259
11.2.2 程序并行性分析.....260
11.2.3 并行程序设计语言.....261
11.2.4 MPP 和 SMP.....261
11.2.5 CM-5 系统.....264
11.3.3 SGI Origin 2000 系列服务器.....266
11.3　机群系统.....267
11.3.1 机群系统的结构特点.....267
11.3.2 机群系统的关键技术.....268
11.3.3 提高通信系统的性能.....269
11.3.4 几种典型系统.....271
关　联.....272
习　题.....273

第 1 章　计算机系统概论

【内容摘要】

纵观计算机的发展长河，去认知计算机组成和计算机系统结构，正确理解计算机的一些基本概念，为后续内容的学习打下良好的基础。

【学习要点】

- 计算机系统的硬件组成及其基本功能
- 计算机系统的软件组成与作用
- 计算机系统层次结构

1.1　计算机的发展与应用

纵观计算机 60 多年的发展长河，我们可以明显看到计算机迅猛发展对人类社会的进步带来的巨大推动作用，尤其是微型计算机的出现和快速普及，极大地开拓了计算机更为广阔的应用领域，正潜移默化地改变人们的生产方式、工作方式、生活方式和学习方式。下面从计算机的硬件和软件两个方面来简单介绍计算机的发展历程。

1.1.1 计算机发展阶段和发展趋势

1、按逻辑部件划分计算机发展阶段

1946 年 2 月，在美国的宾夕法尼亚大学研制成功了第一台电子计算机，称为“ENIAC”（Electronic Numerical Intergrator and Calculator，电子数字积分器），它是一个重 30 吨、占地 150 平方米、每小时耗电 150 千瓦的庞然大物，其内部采用了 18800 只电子管，1500 个继电器，与当今的微型计算机相比，相形见绌，只能送博物馆展览了，但它毕竟是计算机的鼻祖，是一个新生事物，自诞生之日起，便具有强大的生命力，随着主要电子器件的演变，计算机的发展按“代”划分为五个阶段。

（1）电子管时代计算机（1946 年 ~ 1959 年）

主要特点：逻辑器件———电子管
　　主　　存———磁鼓
　　辅　　存———磁带
　　软　　件———机器语言、汇编语言
　　应　　用———科学计算

代表机型：IBM 700 系列计算机

（2）晶体管时代计算机（1959 年 ~ 1964 年）

主要特点：逻辑器件———晶体管
　　主　　存———磁芯
　　辅　　存———磁盘
　　软　　件———高级语言、编译系统
　　应　　用———除科学计算外，已开始应用于数据处理、过程控制

代表机型：IBM 7000 系列计算机

（3）集成电路时代计算机（1964 年 ~ 1975 年）

主要特点：逻辑器件———小规模集成芯片
　　主　　存———磁芯
　　辅　　存———磁盘
　　软　　件———高级语言、操作系统
　　应　　用———科学计算、数据处理、过程控制

代表机型：IBM 360 系列计算机、DEC 生产的 PDP-8 小型商用计算机

（4）大规模/超大规模集成电路时代计算机(1975 年 ~ 1990 年)

主要特点：逻辑器件————大规模/超大规模集成芯片

主　　存————半导体存储芯片

辅　　存————磁盘、光盘

软　　件————高级语言、操作系统

应　　用————科学计算、数据处理、过程控制，并进入网络应用时代

代表机型：微型计算机

（5）超级规模集成电路时代计算机（1990 年 ~ 现在）

随着集成电路的集成度进一步提高，出现了极大、甚大规模集成电路，推动计算机进入了第五个发展阶段。在此阶段，推出了 32 位、64 位微处理器芯片，如 Pentiun　、Athlon 64 等，使微机的性能更上了一个台阶。同时，采用大规模并行计算和高性能机群计算技术的超级计算机也得到迅速发展，如 IBM 公司的“深蓝”超级并行计算机、我国 2004 年研制开发的“曙光”超级计算机。

2、按应用特点划分计算机发展阶段

计算机发展的每一个阶段与其应用密切相关，计算机按其应用特点可以划分三个阶段。

（1）超、大、中、小型计算机阶段（1946 年 ~ 1980 年）

计算机代替人的脑力劳动，提高工作效率，解决较复杂的数学计算和数据处理。

（2）微型计算机阶段（1981 年 ~ 1990 年）

随着微型计算机的普及和计算机技术的日新月异，计算机的功能日益强大，应用领域无所不在，无处不用，对世界科技和经济的发展起到了重要的推动作用。

（3）计算机网络阶段（1981 年以后）

计算机技术与通信技术融合而聚变的网络技术在 20 世纪 80 年代以后发展极为迅速，由简单的远程终端，发展到今天遍布全球的因特网，实现了信息资源共享，推动了信息化社会的进程。

3、计算机的发展趋势

随着科学技术的不断进步，未来计算机将会朝着高性能、网络化、个性化等方向发展，具体体现以下几个方面：

（1）两极发展方向

当今计算机正朝着微型计算机和巨型计算机方向发展，微型计算机的发展和普及程度标志着计算机应用水平，巨型计算机的发展代表了计算机科学的发展水平。

（2）智能化计算机

它采用人工智能方法和技术，系统设计中充分考虑建造知识库管理系统，根据所存储的知识进行推理和判断，在某种程度上具有模仿人的思维功能，并具有声音和图像识别能力。

（3）多媒体计算机

它采用多媒体技术，充分考虑到人的听觉、视觉效果，将大量信息用数值、文字、声音、图形、图像、视频等进行展现，从而极大地改善人们生活、学习环境。

（4）网络计算机

计算机网络技术在压缩时空方面的出色表现而流行于世，已成为现代人们生活中必不可少的组成部分，它正朝着高速化、综合化、智能化方面发展。

（5）非冯 · 诺依曼体系结构计算机

冯 · 诺依曼（johe Von Neumman）在 1946 年提出了“存储程序”的计算机设计方案，人们把按照这一原理设计的计算机称之为冯 · 诺依曼计算机，该计算机“集中顺序控制”的串行机制严重制约了计算机的性能，为进一步提高计算机性能，非冯 · 诺依曼体系结构的计算机理论产生了，相继出现“并行处理机”、“神经网络计算机”、“生物晶体计算机”等。

1.1.2 计算机的应用

随着计算机的普及和计算机技术日新月异，计算机应用范围从科学计算、数据处理等传统领域扩展到办公自动化、人工智能、电子商务、远程教育等，涉及到政治、经济、军事、科技以及社会生活的各个方面。归纳起来，计算机的应用有以下几个方面：

1、科学计算

在科学研究和工程技术中遇到的各类数学问题的计算统称为科学计算。科学计算问题复杂，计算量大，有些用人工计算甚至无法完成。例如地图着色的“四色问题”，需要上百亿次的计算，如果人工计算，一个人既便昼夜不停，也需要几万年！又如海域气象预报，如果用人工计算，算出结果时早已失去了实际意义。在工程预算方面，为选择一个理想的方案，往往需要计算几十个甚至上百个方案，只有使用计算机才能很好地解决上述问题。

2、数据处理

数据处理是指对数据进行收集、分析和加工等。据统计，世界上 70%以上的计算机主要用于数据处理，因此，计算机不在是传统意义上的计算工具了，已成为数据处理领域最强有力的工具，被广泛用于信息传递、情报检索、企事业管理、金融、物流、办公自动化等。

3、实时控制

实时控制又称过程控制，要求及时地检测和收集被控对象的有关数据，并能按最佳状态进行自动调节和控制。利用计算机可以提高自动控制的准确性，实时控制广泛应用于冶金、机械、纺织、化工、电力等行业中。

在军事上，导弹的发射、先进的防空系统等现代化军事设施通常都是计算机构成的控制系统。例如利用卫星定位系统控制导弹实际飞行轨道，直接袭击目标，其命中率几乎接近100%；美国在海湾战争和伊拉克战争中，计算机实时控制技术发挥了淋漓尽致的作用。

4、计算机辅助系统

计算机辅助设计 CAD(Computer Aided Design)是人们借助计算机进行设计的一项专门技术，广泛应用于航空、制造、建筑及微电子技术等方面。首先按照设计任务书的要求提出设计方案，然后进行各种方案的比较，确定产品结构、外形尺寸、材料选择，进行模拟组装，再对模拟组装设备进行各种性能测试，根据测试结果进行修正，最后设计出产品，产品设计完成后再将其分解为零件、分装部件，并给出零件图、分装部件图、总体装配图等，上述全部工作都由计算机完成，大大降低了产品的设计成本，缩短了产品设计周期，因此，CAD技术被各制造业广泛应用。

计算机辅助制造 CAM(Computer Aided Manufacturing)是利用计算机代替人去完成制造过程相关工作，包括生产工艺控制、物料流控制、生产过程控制与仿真、质量控制与检测。目前人们将数控、物料流控制及储存、机器人、柔性制造、生产过程仿真等计算机相关控制技术统称为计算机辅助制造。利用计算机参与大脑的辅助工作是一个不断开拓的新领域，计算机辅助工艺规划 CAPP（Computer Aided Process Planning）、计算机辅助工程 CAE(Computer Aided Engineering)等越来越得到广泛的应用。

5、人工智能

人工智能研究如何让计算机模仿人类的高级思维活动，该领域是近年来重点开发的新兴领域，有着广阔的应用前景，被成功地用于机器人的研制和各类专家系统的开发，以及智能翻译系统、语音图像识别、密码分析、指纹鉴定等。

6、远程教育

Internet 和 WWW 技术的发展带来了教育事业大变革，日渐兴起的远程教育使教学资源通过互联网穿越时空，学生受教育可以不受时间、空间和地域的限制，在全球的每一个角落通过网络自由学习，每一个学生都可以获得第一流老师的指导，都可以向世界最权威的专家请教，都可以从世界的任何角落获得最新的信息和资料。

7、电子商务

电子商务是指以电子形式进行的商品交易活动和服务。电子商务以其公平、快捷、方便、高效、中间环节少、24 小时交易和服务等巨大优势赢得了人们的青睐，在短短几年时间，电子商务得到了突飞猛进的发展，电子商务已成为一股不可阻挡的潮流，改变整个未来世界的面貌，推动全球经济一体化的进程。

1.2 计算机系统的组成

一个计算机系统是由硬件、软件两大部分组成。硬件是计算机系统的物质基础，没有硬件，再好的软件也无法运行；没有强有力的硬件支持，就不可能编制出高质量、高效率的软件；没有好的硬件环境，一些先进的软件也无法运行。同样，软件是计算机系统的灵魂，没有软件，再好的硬件也毫无用途，犹如一堆废物；没有高质量的软件，硬件也不可能充分发挥其效率。

1.2.1 计算机系统的硬件

计算机的硬件由运算器、控制器、存储器、输入设备和输出设备五大组成部分，其中运算器和控制器是计算机的核心，统称为中央处理单元 CPU（Central Processing Unit）。

1、输入设备（Input Device）

输入设备用来向计算机输入程序和原始数据。可分为字符输入设备、图形输入设备和语音输入设备等，常用的输入设备有键盘、鼠标、扫描仪、光笔等。

2、输出设备（Output Device）

输出设备用来输出计算机的处理结果及程序，处理结果可以是数据、字符、表格、图形等，常用的输出设备有显示器、打印机、绘图仪等。

3、存储器（Memory）

存储器用来存放程序和数据，在控制器的控制下，可以与输入设备、输出设备、运算器、控制器等进行信息交换。计算机中的存储器分为三类：主存储器、辅助存储器和高速缓冲存储器。主存储器和高速缓冲存储器统称为内存储器，简称内存，由半导体存储器构成，是可以被 CPU 直接访问。辅助存储器又称为外存储器，简称外存，它不能被 CPU 直接访问。

4、运算器（Arithmetic Logic Unit 简称 ALU）

运算器是对数据进行运算的部件，其主要功能是对二进制数据进行算术运算（加、减、乘、除）和逻辑运算（与、或、非、移位），故又称为算术逻辑单元（ALU）。

5、控制器（Controller）

控制器是整个计算机的控制中心，其功能是控制计算机各个部件自动协调工作，具体而言，控制器的功能包括顺序控制、操作控制和时间控制。顺序控制是对程序中指令执行顺序的控制，换句话讲，如何保证计算机执行完一条指令后能够正确地取下一条指令并执行。操作控制是指计算机在执行一条指令时怎样产生这条指令所需的所有控制信号。时间控制是将指令所需的所有控制信号按照一定的时间顺序发送给相应部件，控制各个部件去完成相应动作，进而实现指令的功能。

1.2.2 计算机系统的软件

计算机系统中各种软件的有机组合构成了计算机的软件系统，基本的软件系统包括系统软件和应用软件两大类。

1、系统软件

（1）操作系统

操作系统是系统软件的核心，负责管理和控制计算机的硬件资源、软件资源和程序的运行，具体包括并发控制、内存管理、进程调度、I/O 及文件管理等，它是用户与计算机之间的接口，提供了软件的开发环境和运行环境。

（2）语言处理程序

由于计算机本身只能识别和处理用二进制的“0”或“1” 形式表示的机器语言，因此任何用其他语言编写的都必须翻译为机器语言程序后才能由计算机去执行和处理。语言处理程序就是完成这种翻译工作的程序，其翻译方式有两种：一种称为解释，通过解释程序对用程序设计语言编写的源程序边解释边执行。另一种称为编译，通过编译程序将源程序全部翻译为机器语言的目标程序后再执行。

（3）数据库管理系统

数据管理系统被广泛应用在信息处理、情报检索和各种管理系统中，极大地方便了用户根据需要建立数据库，查询、显示、修改数据库的内容，打印各种表格等。

（4）分布式软件系统

分布式软件系统包括分布式操作系统、分布式编译系统、分布式数据库系统、分布式算法及软件包等，主要用于分布式计算环境、管理分布式计算资源、控制分布式程序的运行、提供分布式程序开发与设计工具等。

（5）网络软件系统

网络软件系统包括网络操作系统、网络协议、通信软件、网络应用程序等，支持网络活动和数据通信。人类借助计算机网络这个平台实现远程教育、网络聊天、视频点播、电子邮件、电子商务等。

（6）各种服务程序

服务程序是指为方便用户使用和维护计算机所编制的程序。这类程序包含的内容很广泛，如装入程序、编辑程序、调试程序、诊断程序等，一些通用的应用软件也可以作为服务程序，如文字处理程序、表格处理程序、图形处理软件等。

2、应用软件

应用软件是指用户为解决某个应用领域中的各类问题而编写开发的程序。由于计算机的应用极其广泛，所以应用软件种类繁多，极其丰富。目前应用软件正向标准化、集成化方向发展，许多通用的应用软件根据其功能组成不同的应用软件包，方便供用户选择使用。

1.2.3 计算机系统的层次结构

由上述内容可知，计算机是硬件与软件相结合的一个整体，对于不同的应用、不同的对象、不同的设计者，计算机的复杂程度各不相同，从而使计算机具有了不同的属性。为了使计算机硬件和软件之间、系统与使用者之间更好地协调与配合，以便构成合理、高效的计算机系统，因此提出了计算机系统层次结构，如图 1-1 所示。

1、微程序设计层

此层可以进行微程序设计，由机器硬件直接执行微指令编写的微程序。

2、指令系统层

此层可以进行机器语言程序设计，该程序由第 1 层的微程序负责解释执行。

3、操作系统层

它由操作系统程序实现。操作系统程序是机器指令和广义指令组成，广义指令是对操作系统定义和解释的软件指令，因此，此层属于混合层。

4、汇编语言层

此层可以进行汇编语言程序设计，软件上需要汇编程序的支持。

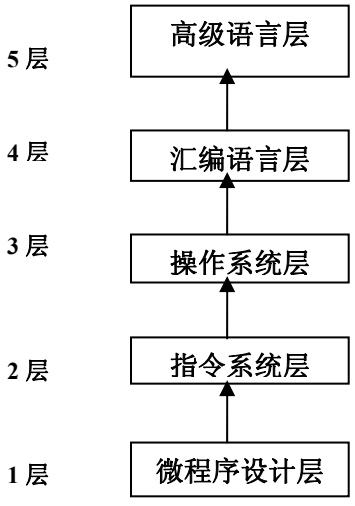


图 1-1　计算机系统层次结构示意图

5、高级语言层

此层可以进行高级语言程序设计，是面向用户的，软件上需要各种高级语言编译程序的支持。

由图 1-1 可以看到不同层次之间的关系：上面的一层是建立在下一层的基础上实现出来的，实现的功能更强大，更接近人解决问题的思维方式和处理问题的具体过程，对使用人员更方便，使用这一层提供的功能时，不必关心下一层的实现细节；下面一层是实现上一层的基础，更接近计算机硬件实现的细节，实现的功能相对简单，人们使用这些功能更感到困难。在实现这一层的功能时，可能尚无法了解其上一层的目标和将要解决的问题，也不必理解其更下一层实现中的有关细节问题，只要使用下一层所提供出来的功能来完成本层次的功能处理即可。采用这种分层次的方法来分析和解决某些问题，有利于简化处理问题的难度，在某一段时间，在处理某一层中的问题时，只需集中精力解决当前最需要关心的核心问题即可，而不必牵扯各上下层中的其他问题。

1.3 计算机系统结构

1.3.1 传统冯·诺依曼计算机系统结构

传统冯·诺依曼计算机系统结构如图 1-2 所示。其硬件由运算器、控制器、存储器、输入设备和输出设备五大部分组成。特点是以运算器为中心，采用存储程序原理，将编写好的程序预先存储到存储器中，然后启动系统自动执行程序。构成程序的指令是串行执行的，在控制器的控制下自动地、连续地从存储器中依此取出指令并进行分析和执行。存储器是按顺序的一维线性空间，按地址访问存储器。指令和数据采用二进制形式。

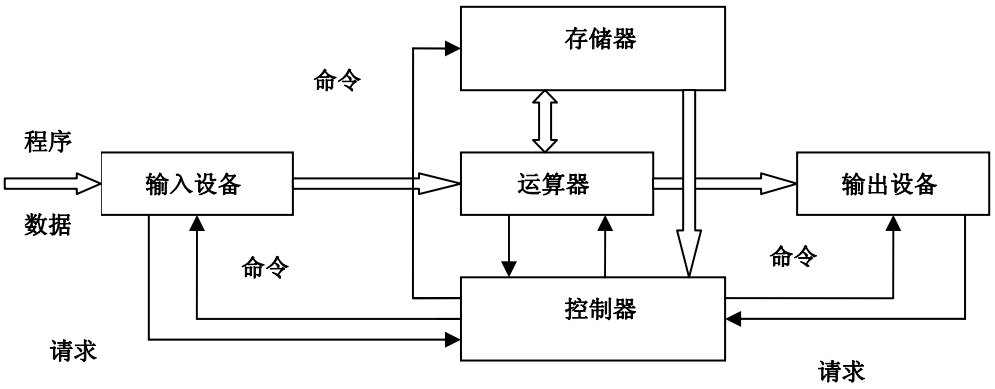


图 1-2　　传统的计算机基本结构框图

1.3.2 现代计算机系统结构

由于传统冯·诺依曼计算机系统结构存在着很多缺点，首先是 CPU 与存储器之间的瓶颈和串行执行指令都会严重影响 CPU 功效的发挥；其次是高级语言与机器语言的巨大差异会带来较为繁重的编译工作；再者是较复杂的数据结构导致必须使用地址映射才能解决。以 CPU 为中心的传统冯·诺依曼计算机系统结构已不能计算机发展的需要，甚至会影响计算机的性能，因此，必须改变传统冯·诺依曼计算机系统结构，以适用计算机发展的需要。现代计算机系统结构孕育而生，新的计算机技术不断涌现。图 1-3 所示是以存储器为中心的现代计算机系统结构。现代计算机系统结构的计算机与传统冯·诺依曼计算机相比较，不同之处主要体现在三个方面：第一，现代计算机系统结构以存储器为中心，I/O 设备与 CPU 可以并行工作，进一步改进了系统的性能。第二，现代计算机采用先行控制技术和流水线技术，改变传统的串行执行程序为并行，从而提高系统作业的吞吐量。第三，现代计算机中的存储器采用多体交叉存储器，可以在一个存储器访问周期中同时对多个存储单元进行访问，实现多字的一次性存取，增加存储带宽。

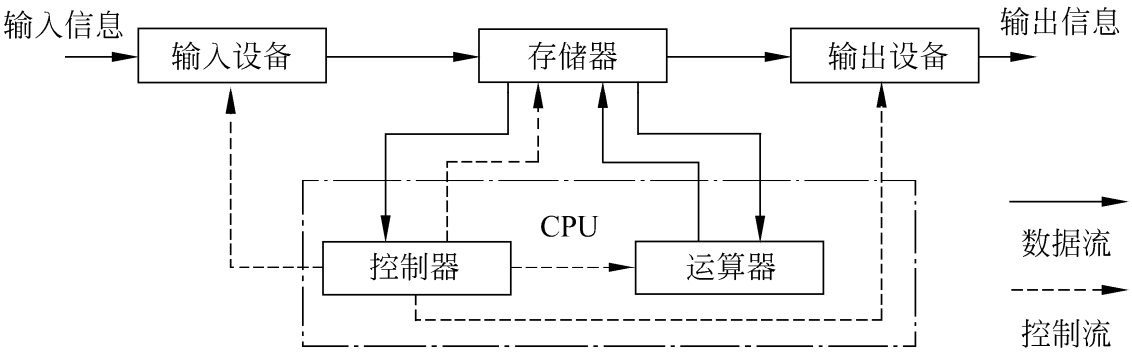
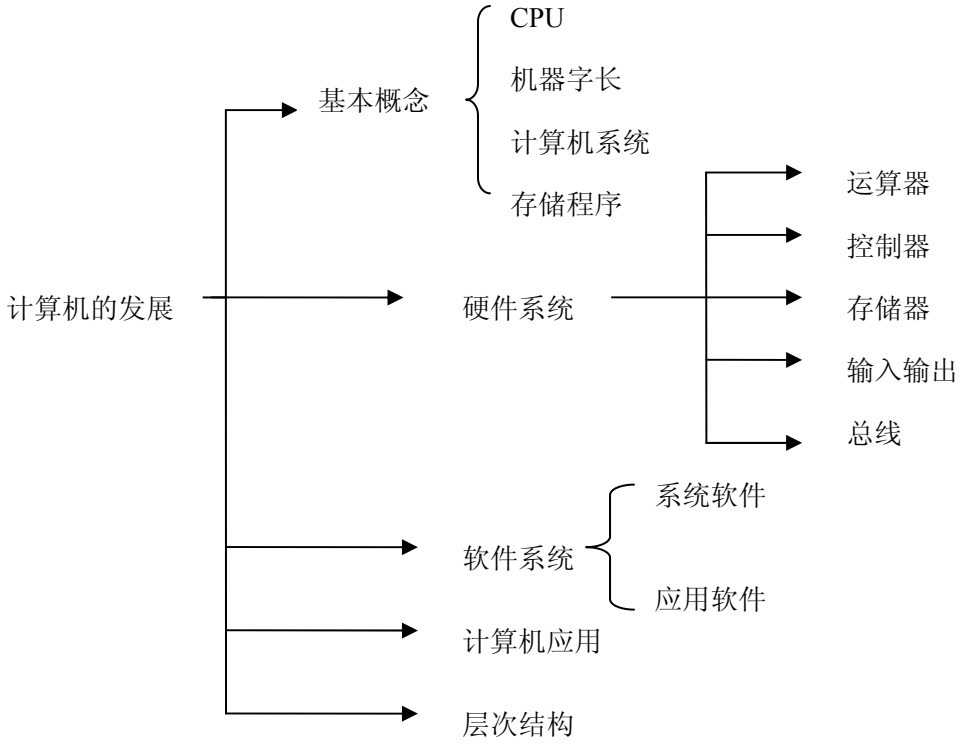


图 1-3　　现代的计算机基本结构框图

关　联



习　题

1.1 冯·诺依曼计算机由哪几个部分组成？各部分的功能是什么？

1.2 计算机的发展经历了哪几个时代？计算机有哪些方面的应用？

1.3 传统的计算机系统结构的特点？现代计算机系统结构为什么以存储器为中心？

1.4 解释下面术语的含义

计算机系统、系统软件、计算机系统的层次结构、硬件、软件、微处理器、机器字长

1.5 单选题

（1）1946 年美国推出的世界上第一台计算机称为（　）。

A、ENIAC　　B、UNIVAC-I

C、ILLIAC-IV　　D、EDVAC

（2）完整的计算机系统包括两大部分，它们是（　）。

A、运算器和控制器　　B、主机与外设

C、硬件与软件　　D、硬件与操作系统

（3）现代计算机系统结构是以（　）为中心的。

A、运算器　　B、控制器

C、存储器　　D、寄存器

1.6 填空题

（1）CPU 主要包括_____和_____两个部分

（2）计算机系统的硬件包括_____、_____、_____、_____、_____等五大部分

（3）从软件、硬件的交界面看，计算机层次结构包括_____和_____两大部分。

第 2 章　计算机中数据的表示

【内容摘要】

数据是计算机处理的对象，它在计算机内部是用二进制信息来表示的，不仅表示形式最简单，而且，物理上最可靠。数据包括数值型数据和非数值型数据两种，数值型数据用于表示整数、小数和实数，其表示方式将会涉及数的位权、基数、符号、小数点等问题；非数值型数据用于表示字符、声音、图形、图像等，其表示方式主要是代码的约定。

【学习要点】

- 进位计数制及其之间的相互转换
- 数据的定点表示法和数据的浮点表示法
- 机器数与真值
- 非数值数据的编码
- 数据的校验

2. 1　进位计数制及其之间的转换

2.1.1 进位计数制

进位计数制是一种按进位进行计数的制式。在日常工作生活中，我们习惯使用十进制数，而计算机内部则只能识别二进制数，但在程序设计时，数据往往用十进制数或十六进制数表示，而很少用二进制数，因为用二进制数表示数据或地址时，位数太长，书写不方便，易出错。

进位计数制有两个特征：

基数 R（Radix）：是指进制数中数码所允许取值的个数，且计数规则是“逢 R 进一”。

位权 W（Weight）：是指基数 R 的 i 次幂（R^i），表示进制数中第 i 位的位权。

1、十进制数(Decimal)

十进制数是我们日常工作生活中最常用的数，数中的任一数码 $d_i \in \{0，1，2，\cdots，9\}$，所以十进制数的基数为 10，且逢十进一；十进制数的位权为 10^i。任何一个十进制数都可以用式（2-1）写成一个按权展开的多项式和的形式。

$$\begin{aligned} D&=d_nd_{n-1}\cdots d_1d_0.d_{-1}\cdots d_{-m} \\ &=d_n\times10^n+d_{n-1}\times10^{n-1}+\cdots+d_1\times10^1+d_0\times10^0+d_{-1}\times10^{-1}+\cdots+d_{-m}\times10^{-m} \\ &=\sum_{i=n}^{-m} d_i\times10^i \end{aligned} \qquad (2\text{-}1)$$

十进制数后缀为 D，可省略。如 78D，179.26D 或 78，179.26 等。

2、二进制数（Binary）

二进制数中的任一数码 $b_i \in \{0，1\}$，所以二进制数的基数为 2，且逢二进一；二进制数的位权为 2^i。任何一个二进制数都可以用式（2-2）写成一个按权展开的多项式和的形式。

$$\begin{aligned} B&=b_nb_{n-1}\cdots b_1b_0.b_{-1}\cdots b_{-m} \\ &=b_n\times2^n+b_{n-1}\times2^{n-1}+\cdots+b_1\times2^1+b_0\times2^0+b_{-1}\times2^{-1}+\cdots+b_{-m}\times2^{-m} \\ &=\sum_{i=n}^{-m} b_i\times2^i \end{aligned} \qquad (2\text{-}2)$$

二进制数后缀为 B，如 1001B，10011101.1101B 等。

3、十六进制数(Hexadecimal)

十六进制数中的任一数码 $h_i \in \{0，1，2，\cdots，9，A，B，C，D，E，F\}$，所以十六进制

数的基数为 16，且逢十六进一；十六进制数的位权为 16^i。任何一个十六进制数都可以用式（2-3）写成一个按权展开的多项式和的形式。

$$H=h_nh_{n-1}\cdots h_1h_0.h_{-1}\cdots h_{-m}$$

$$=h_n\times16^n+h_{n-1}\times16^{n-1}+\cdots+h_1\times16^1+h_0\times16^0+h_{-1}\times16^{-1}+\cdots+h_{-m}\times16^{-m}$$

$$=\sum_{i=n}^{-m}h_i\times16^i \qquad (2\text{-}3)$$

十六进制数后缀为 H，如 23AH，9C78.1B3H 等，A~F 相当于十进制数的 10~15。为了区分十六进制数和标识符（标号、变量等），当十六进制数首位为 A~ F 时，其前必须加“0”，如 0F08H，0C57.2H。

2.1.2 进位计数制之间的相互转换

1、二进制数与十进制数之间的转换

（1）二进制数转换为十进制数

直接按式（2-1）展开并求和即可。

【例 2-1】将 101110.101B 转换为十进制数。

解：$101110.101B=1\times2^5+0\times2^4+1\times2^3+1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}=46.625$

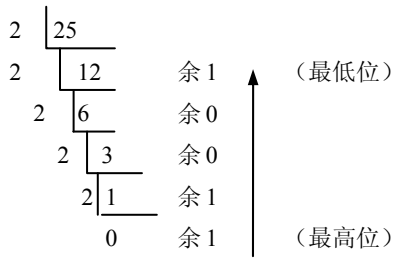
（2）十进制数转换为二进制数

十进制数转换为二进制数有两种方法：直接法和查表法

直接法：对于整数部分采用“除 2 取余法”，直到商为零，而余数（由低位到高位）即为转换成的二进制数整数部分；对于小数部分采用“乘 2 取整法”，而积的整数部分（由高位到低位）即为转换成的二进制数小数部分。最后将转换结果合起来便得到相应的二进制数。

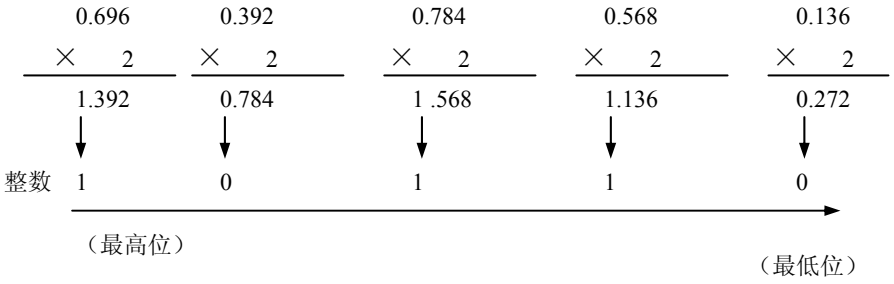
【例 2-2】将 25.696 转换为二进制数。

解：对整数部分，采用“除 2 取余法”，计算如下：



因此 25D=11001B

对于小数部分,采用“乘 2 取整法”，计算如下：



因此 0.696D≈0.10110B

25.696D≈11001.10110B

查表法：利用十进制与二进制数对照表，把十进制数分解成 2^i 多项式和的形式，然后查表求得对应的二进制数。十进制转换为二进制数对应关系如表 2-1 所示。

表 2-1　　十进制与 2 的整次幂之间的对应关系

0.125	0.25	0.5	1	2	4	8	16	32	64	128	256	512	1024
2^{-3}	2^{-2}	2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}

具体方法：把十进制数（整数和小数）分解成 2 的整次幂项的和，对于出现 2 的整次幂

项相应的位置数码取“1”，否则取“0”。

【例 2-3】将 133. 625D 转换为二进制数。

解：133.625D=128+4+1+0.5+0.125=$2^7+2^2+2^0+2^{-1}+2^{-3}$=10000101.101B

2、二进制数与十六进制数之间的转换

由于四位二进制数的编码与一位十六进制数的数码之间存在着一一对应的关系，如表 2-2 所示。因此，二进制数与十六进制数之间的转换十分简单、方便。

表 2-2　　二进制和十六进制之间的对应关系

H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
B	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

（1）二进制数转换为十六进制数

以小数点为分界线，分别向左、向右按四位进行分组，不足四位者，在最前面或最后面补 0，使之成为四位，然后，每四位按表 2-2 的对应关系用一位十六进制数来表示。

【例 2-4】将 1111000111.10011B 、110110001101B 分别转换为十六进制数。

解：1111000111.10011B=0011 1100 0111.1001 1000B=3C7.98H

110110001101B=1101 1000 1101B=0D8DH

（2）十六进制数转换为二进制数

每一位十六进制数按表 2-2 的对应关系转换成四位的二进制数即可，小数点位置不变。

【例 2-5】将 3F.75H 转换为二进制数。

解：3F.75H=0011 1111.0111 0101B=111111.01110101B

3、十进制数与十六进制数之间的转换

（1）十六进制数转换为十进制数

直接按式(2-3)展开并求和即可。

【例 2-6】将 7B9.62H 转换为十进制数。

解：7B9.3CH=$7\times16^2+11\times16^1+9\times16^0+3\times16^{-1}+12\times16^{-2}$=1977.234375

（2）十进制数转换为十六进制数

采用类似于十进制转换为二进制数的方法，对于整数部分采用“除 16 取余法”，直到商为零，而余数（由低位到高位）即为转换成的十六进制数整数部分；对于小数部分采用“乘 16 取整法”，而积的整数部分（由高位到低位）即为转换成的十六进制数小数部分。最后将转换结果合起来便得到相应的十六进制数。这种方法因乘、除 16 比较复杂，一般采用下面间接方法。

十进制数 $\xrightarrow{\text{转换}}$ 二进制数 $\xrightarrow{\text{转换}}$ 十六进制数

【例 2-7】将 105.75 转换为十六进制数。

解：105.75=1101001.11B=0110 1001.1100B=69.CH

2.2　定点数的表示

2.2.1 符号的表示

计算机本身无法识别数据的符号(正号“+”、负号“-”)，为了让计算机能够识别数的符号，必须用“0”和“1”来表示。因此规定：数据字的最高位为符号位，并且用“0”表示正（+)；用“1”表示负（-)。于是一个数据字的书写形式和机器内的存储形式存在差异，为了区别这种不同，我们把书写形式（正、负符号加绝对值）的数据称为真值，机器内的存储形式（符号位加二进制数值）的数据称为机器数。一个数据的机器数究竟采用多少位表示，与机器的字长有关，若机器的字长是 16 位，则表示数据的机器数也是 16 位。机器数有一定的长度限制，相应的真值便会存在一定的范围要求，以 8 位机为例，8 位的机器数相应的真值范围为-128~+127。

2.2.2 小数点的表示

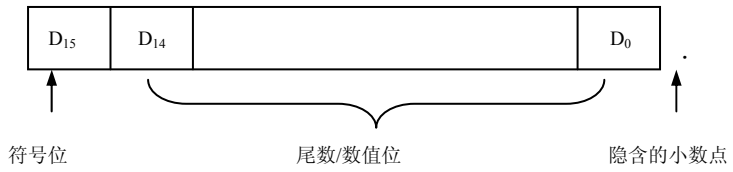
数据中的小数点在计算机中有两种表示方法，即定点表示法和浮点表示法。

1、定点表示法

定点表示法所表示的数据为定点数。定点数指小数点在数中的位置是固定不变的。因此，计算机中数的小数点不用表示，或者隐含表示，不占用存储空间。下面以 16 位机为例说明定点表示法。

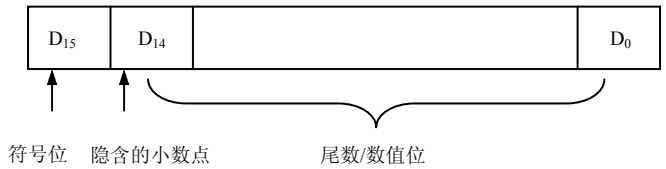
（1）定点整数

小数点被固定在数值位的最低有效位之后，其格式如下：



（2）定点小数

小数点被固定在符号位与尾数之间，其格式如下：



定点整数表示的数只能是整数，而定点小数所表示的数只能是小数。在本书中，我们更侧重于定点小数。

2、浮点表示法

定点表示法难以表示数值很大的数据和数值很小的数据，为了表示更大范围的数据，数学上通常采用科学计数法，将数据表示成三部分：第一部分表示数据的符号；第二部分表示数据的有效值；第三部分表示数据中的小数点位置。改变了第三部分的数值，相当于改变了小数点位置，这种表示小数点的方法称为浮点表示法。浮点表示法所表示的数据是浮点数，浮点数 x 通常表示为：

$$x = (-1)^S M \times R^e$$

其中，S(Sign)——符号，0 表示正（+），1 表示负（-）。

M(Mantissa)——尾数，为定点小数，表示浮点数 X 的有效数字，决定浮点数的精度。

e（exponent）——阶码，为定点整数，表示浮点数 x 中小数点的实际位置，是影响浮点数大小的主要因素，换一句话讲，决定了浮点数的表示范围。

R（Radix）——基数，是一个系统中约定的常数，通常取值为 2。

由于 R 是一个常数，所以，浮点数在计算机中存储形式为 S、E 、M，其中 E 是阶码的移码形式。由此可见，浮点数的表示又归结到定点整数和定点小数表示问题。

关于浮点数的表示更详细的内容将在 **2.3** 节中讨论。

2.2.3 几种机器数形式

对于数据定点表示的符号问题，计算机如何处理符号位？符号位能否同数值位一样参加运算？为了妥善地处理好这个问题，下面以 n 位机为例，介绍几种定点数的机器数形式（机器数为 n 位，其中符号位占 1 位，数值位占 n-1 位，而小数点隐含）。

1、原码

原码是一种最简单、最直观的机器数表示形式，与真值的形式最为接近。

（1）原码的定义

设 x 为 n 位的二进制数据，式（2-4）和式（2-5）分别给出了 x 为定点小数 $\pm 0.x_1x_2\cdots x_{n-1}$ 和 x 为定点整数 $\pm x_1x_2\cdots x_{n-1}$ 的原码定义。

定点小数原码的定义：

$$[x]_{原}=\begin{cases} x & 0\le x\le +(1-2^{-(n-1)}) \\ 1-x=1+|x| & -(1-2^{-(n-1)})\le x\le 0 \end{cases} \quad (2\text{-}4)$$

定点整数原码的定义：

$$[x]_{原}=\begin{cases} x & 0\le x\le +(2^{n-1}-1) \\ 2^{n-1}-x & -(2^{n-1}-1)\le x\le 0 \end{cases} \quad (2\text{-}5)$$

【例 2-8】已知二进制数的真值 x，求$[x]_{原}$。

①x=+0.0010110　②x=-0.0010110　③x=+0010110　④x=-0010110

解：由定义可得

① ∵x≥0，∴$[x]_{原}$= x = 0.0010110

② ∵x＜0，∴$[x]_{原}$= 1-x = 1.0010110

③ ∵x≥0，∴$[x]_{原}$= x = 00010110

④ ∵x＜0，∴$[x]_{原}$= 2^7-x = 10010110

（2）原码所表示的数据大小

以 n 位定点小数为例，分析原码所表示的数据范围。

n 位定点正小数的原码：0.00…0 (+0) ⋮ 0.11…1 (+(1-2^-(n-1)))

n 位定点负小数的原码：1.00…0 (-0) ⋮ 1.11…1 (-(1-2^-(n-1)))

所以，n 位定点小数的原码所表示的数据范围为[-（$1-2^{-(n-1)}$），+（$1-2^{-(n-1)}$）]。0 的原码形式有两种：+0 的原码为 0.00…0，-0 的原码为 1.00…0。

（3）原码的运算特点

1）原码的移位

原码移位规则：符号位保持不变，数值位进行左移或右移，移出后出现的空位进行补 0。

【例 2-9】已知 x 的原码[x]原，求[2x]原、[1/2x]原

①　$[x]_{原}$ = 0.0101001　　②$[x]_{原}$ = 1.0101001

解：　①$[2x]_{原}$ = 0.1010010　　$[1/2x]_{原}$ = 0.0010100

②$[2x]_{原}$ = 1.1010010　　$[1/2x]_{原}$ = 1.0010100

2）原码的加减运算

原码的加减运算较复杂，不仅要判断两个数的符号，确定是进行加法还是进行减罚运算，还要比较两个数的绝对值大小，来决定运算结果的符号。可见，原码不便于进行加减运算，对原码的符号位处理的硬件较复杂。为了简化运算，人们提出机器数的补码形式，从此，不再分别处理符号位和数值位，把符号位作为数值的一部分参与运算，并能将减法运算转换成加法运算，从而简化了硬件及结构，降低了成本。

2、补码

（1）互补数

计数制中，为了简化计数，常采用一种计满归零的方法。例如钟表的计时，计满 12 归 0，即 12=0，于是有：13 点=1 点，14 点=2 点，…，23 点=11 点，24 点=0 点。因此，钟表是以 12 为模的计数方式，其数学表达式为：12=0　（mod 12）

对于钟表的表盘，若时针指向 12 点，顺时针方向拨时针 8 格，即表示的时间为 8 点，而按逆时针方向拨时针 4 格，表示的时间仍为 8 点，因此，两种不同方向的拨法其结果是一样的，于是，8 和-4 是模 12 的互补数，记为：-4=8　（mod 12）

对于任意一个数 x，若模为 M，则数 x 的补数$[x]_{补数}$可由式（2-6）进行计算。

$$[x]_{补数} = M + x \pmod{M} \quad (2\text{-}6)$$

根据式（2-6）可知：

1)当 x≥0 时，M + x ≥ M，把 M 丢掉，$[x]_{补数}$ = x，即正数的补数等于他本身；

2)当 x＜0 时，$[x]_{补数}$ = M + x = M-｜x｜ 即负数的补数等于模与该数绝对值之差。

【例 2-10】求在模 M=2 的条件下，二进制数 x 的补数$[x]_{补数}$

①x= +0.0101001　②x= -0.0101001

解：①∵x≥0，∴$[x]_{补数}$ = 2 + x =x = 0.0101001

②∵x＜0，∴$[x]_{补数}$ = 2 + x = 2-｜x｜= 2-0.0101001=1.1010111

（2）补码的定义

补码实际上是对模的补数，由于机器字长的限制，数据在计算机中的运算是有模运算。设 x 为 n 位的二进制数据，式（2-7）和式（2-8）分别给出了 x 为定点小数$\pm 0.x_1x_2\cdots x_{n-1}$和 x 为定点整数$\pm x_1x_2\cdots x_{n-1}$的补码定义。

定点小数补码的定义：

$$[x]_{补}=\begin{cases} x & 0\le x\le +(1-2^{-(n-1)}) \\ 2+x=1-|x| & -1\le x\le 0 \end{cases} \quad (\text{mod } 2) \qquad (2\text{-}7)$$

定点整数补码的定义：

$$[x]_{补}=\begin{cases} x & 0\le x\le 2^{n-1}-1 \\ 2^n+x & -2^{n-1}\le x\le 0 \end{cases} \quad (\text{mod } 2^n) \qquad (2\text{-}8)$$

【例 2-11】已知 x，求 x 的补码$[x]_{补}$。

①x=+0.0010110　②x=-0.0010110　③x=+0010110　④x=-0010110

解：由定义可得

① ∵x≥0，∴$[x]_{补}$= x = 0.0010110

② ∵x＜0，∴$[x]_{补}=2^8+x$ = 1.1101010

③ ∵x≥0，∴$[x]_{补}$= x = 00010110

④ ∵x＜0，∴$[x]_{补}= 2^8+x$ = 11101010

（3）补码所表示的数据大小

以 n 位定点小数的补码为例，分析补码所表示的数据范围。

n 位定点正小数的补码：0.00…0 +0 ⋮ 0.11…1 +（1-$2^{-(n-1)}$）

n 位定点负小数的补码：1.00…0 -1 ⋮ 1.11…1 $-2^{-(n-1)}$

所以，n 位定点小数的补码所表示的数据范围为[-1，+（1-$2^{-(n-1)}$）]。小数 0 的补码形式只有一种：$[+0]_{补}$=0.00…0，$[-0]_{补}$=2+0. 00…0=0.00…0。

（4）补码的运算特点

1）$[x]_{补}$的移位

补码的移位规则：补码的左移时，符号位保持不变，数值位进行左移，最底位出现的空位进行补 0；补码的右移时，符号位保持不变，数值位进行右移，最高位出现的空位填补符号位。

【例 2-12】已知 x 的补码[x]补，求[2x]补、[1/2x]补

①$[x]_{补}$ = 0.0101001　②$[x]_{补}$ = 1.0101001

解：① $[2x]_{补}$ = 0.1010010　$[1/2x]_{补}$ = 0.0010100

② $[2x]_{补}$ = 1.1010010　$[1/2x]_{补}$ = 1.0010100

在补码左移过程中，注意不要把高位的数值位移出，否则将会出错。例如，将 8 位定点正小数补码$[x]_{补}$=0.1010011 进行左移时，需要将最高数值位的 1 移出，如果将 1 移入符号位，正数的补码变成了负数的补码，造成符号出错；如果将 1 丢掉，又会失去最高位的有效数值

而导致出错。同理，对于 8 位定点负小数补码$[x]_{补}$=1.0010011 进行左移时，也会出现同样的错误。

2）$[x]_{补}$与$[x]_{原}$的转换

若 x≥0，则$[x]_{原}=[x]_{补}$。若 x＜0，则将$[x]_{原}$除符号位以外的各位取反后，再在最底位上加 1，即得到$[x]_{补}$；反之，将$[x]_{补}$除符号位以外的各位取反后，再在最底位上加 1，即得到$[x]_{原}$。

【例 2-13】已知$[x]_{原}$，求$[x]_{补}$。

①$[x]_{原}$=0.0010110　②$[x]_{原}$=1.0010110　③$[x]_{原}$=00010110　④$[x]_{原}$=10010110

解：由定义可得

①∵x≥0，∴$[x]_{补}=[x]_{原}$= 0.0010110　②∵x＜0，∴$[x]_{补}$=1.1101010

③∵x≥0，∴$[x]_{补}=[x]_{原}$= 00010110　④∵x＜0，∴$[x]_{补}$=11101010

从**【例 2-13】**的结果中我们还可以看出一个规律：当 x＜0 时，保持原码的符号位不变，从原码的最底位开始向高位扫描，在遇到第一个 1 之后，保持该位 1 和比他低的各位不变，将其余位取反，即得到$[x]_{原}$对应的补码。

【例 2-14】 已知$[x]_{补}$，求$[x]_{原}$。

①$[x]_{补}$=0.0010110　②$[x]_{补}$=1.0010110　③$[x]_{补}$=00010110　④$[x]_{补}$=10010110

解：由定义可得

①∵x≥0，∴$[x]_{原}=[x]_{补}$= 0.0010110　②∵x＜0，∴$[x]_{原}$=1.1101010

③∵x≥0，∴$[x]_{原}=[x]_{补}$= 00010110　④∵x＜0，∴$[x]_{原}$=11101010

3）$[x]_{补}$与$[-x]_{补}$的关系

已知$[x]_{补}$求$[-x]_{补}$称为对$[x]_{补}$求补或变补。其规则是：将$[x]_{补}$的各位取反（包括符号位），然后在最低位加 1，即得到$[-x]_{补}$。反之亦然。

【例 2-15】已知$[x]_{补}$，求$[-x]_{补}$。

①$[x]_{补}$=0.0010110　②$[x]_{补}$=1.0010110　③$[x]_{补}$=00010110　④[x]=10010110

解：由定义可得

①$[-x]_{补}$=1.1101010　②$[-x]_{补}$=0.1101010

③$[-x]_{补}$=11101010　④$[-x]_{补}$=01101010

3、反码

反码实质上是一种特殊的补码，其特殊性在于反码的模比补码的模小 2^{n-1}。

（1）反码的定义

设 x 为 n 位的二进制数据，式（2-9）和式（2-10）给出了 x 为定点小数$\pm 0.x_1x_2\cdots x_{n-1}$和 x 为定点整数$\pm x_1x_2\cdots x_{n-1}$的反码定义。

定点小数反码的定义：

$$[x]_{反}=\begin{cases} x & 0\le x\le +(1-2^{-(n-1)}) \\ (2-2^{-(n-1)})+x & -(1-2^{-(n-1)})\le x\le 0) \end{cases} \pmod{(2-2^{-(n-1)})} \quad (2\text{-}9)$$

定点整数反码的定义：

$$[x]_{反}=\begin{cases} x & 0\le x\le 2^{n-1}-1 \\ (2^n-1)+x & -(2^{n-1}-1)\le x\le 0) \end{cases} \pmod{(2^n-1)} \quad (2\text{-}10)$$

【例 2-16】已知 x，求 x 的反码$[x]_{反}$。

①x=+0.0010110　②x=-0.0010110　③x=+0010110　④x=-0010110

解：由定义可得

①$[x]_{反}$=0.0010110　②$[x]_{反}$=1.1101001　③$[x]_{反}$=00010110　④$[x]_{反}$=11101001

（2）反码所表示的数据大小

以 n 位定点小数为例，分析反码所表示的数据范围。

n 位定点正小数的反码：0.00…0 +0 ⋮ 0.11…1 +（$1-2^{-(n-1)}$）

n 位定点负小数的反码：1.00…0 -（$1-2^{-(n-1)}$）⋮ 1.11…1 -0

所以，n 位定点小数的反码所表示的数据范围为[-（$1-2^{-(n-1)}$），+（$1-2^{-(n-1)}$）]。小数 0 的反码形式有两种：+0 的反码为 0.00…0，-0 的反码为 1.11…1。

4、移码

移码通常用来表示浮点数的阶码 e。我们知道阶码是整数，若采用定点整数补码形式表示，补码的符号位可以作为数值位，这样一来，负数补码的值总大于正数补码的值，在进行对阶时，需要进行比较两个浮点数的阶码大小，比较起来就不直观和方便。因此，人们提出了移码形式。浮点数阶码 e 的移码形式我们记为“E”。移码 E 实质上是将阶码 e 的真值映像到一个正数域，这样移码 E 的大小可以直观地反映出真值的大小，便于比较数值的大小。若阶码 e 的真值范围为[-128，+127]，则 E=128+e，移码 E 的范围是[0，255]。

（1）移码的定义

设 x 为 n 位的二进制数据，式（2-11）给出了 x 为定点整数$\pm x_1x_2\cdots x_{n-1}$的移码定义。

定点整数移码的定义：

$$[x]_{移}= 2^{n-1}+x \qquad -2^{n-1}\le x\le 2^{n-1}-1 \qquad (2\text{-}11)$$

由式（2-11）知，移码是把真值 x 在数轴上正向平移了 2^{n-1}，所以，移码也称为增量或余码。

（2）移码所表示的数据大小

对于 n 位定点整数的移码，其表示的数据范围。

n 位定点整数的移码：100…0 0 ⋮ 011…1 2^n-1

所以，n 位定点整数的移码所表示的数据范围为[0，2^n-1]，0 的移码形式只有一种，即 100…0。

（3）移码与补码的关系

1）当 $0\le x\le 2^{n-1}-1$ 时，∵$[x]_{补}=x$，$[x]_{移}=2^{n-1}+x$，∴$[x]_{移}=2^{n-1}+[x]_{补}$

2）当 $-2^{n-1}\le x<0$ 时，∵$[x]_{补}=2^n+x$，$[x]_{移}=2^{n-1}+x$，∴$[x]_{移}=2^{n-1}+[x]_{补}-2^n=[x]_{补}-2^{n-1}$

【例 2-17】已知二进制数 x，求[x]补和[x]移

①x = +0101001　②x = -0101001

解：①∵x≥0，∴$[x]_{补}$=00101001　$[x]_{移}$=10101001

②∵x＜0，∴$[x]_{补}$ =11010111　$[x]_{移}$=01010111

2.3 浮点数的表示

2.3.1 浮点数的格式

根据浮点表示法可知，计算机中的一个浮点数由阶码 E、尾数 M 和符号 S 三个部分组成。其中 E 为定点整数的移码形式，M 为定点小数的补码形式，S 为数符。由于三个部分所占的位置和长度不同，浮点数的格式也不相同，为此，美国 IEEE（电气电子工程师协会）提出了一个从系统结构角度支持浮点数的表示方法，称为 IEEE754 标准，是目前计算机普遍采用的标准。

（1）32 位单精度浮点数格式（IEEE754 标准）

32 位单精度浮点数格式如图 2-1 所示。

符号位 S（1 位）	阶码 E（8 位）	尾数 M（23 位）

图 2-1　IEEE754 标准 32 位单精度浮点数格式

S：数符，占 1 为。S 为 0 表示“+”，S 为 1 表示“-”。

E：阶码 e 的移码形式，占据 8 位，包括 1 位阶符和 7 位数值。将阶码 e 的真值平移 127 便转换成移码 E，即 E=127+e。移码 E 的取值范围为[1，254]，0 和 255 用于表示特殊含义的数值。

M：尾数，占 23 位。由于尾数的规格化要求，IEEE754 标准约定小数点左边隐含一位“1”，从而使尾数的实际有效位为 24 位，即尾数的有效值为 1.M。

根据上述规定，32 位单精度浮点数所表示的数值 x 为：

$$x = (-1)^S \times 2^{E-127} \times 1.M$$

若 E=0，且 M=0，则 x 为 0

若 E=0，且 M≠0，则 $x = (-1)^S \times 2^{-127} \times 0.M$，为非规格化浮点数

若 1≤E≤254，则 $x = (-1)^S \times 2^{E-127} \times 1.M$，为规格化浮点数

若 E=255，且 M≠0，则 x 为“非数值”

若 E=255，且 M=0，则 $x = (-1)^S \times \infty$

值得注意的是：非规格化浮点数和 0 的尾数 M 前的隐含位是“0”而不是“1”。

【例 2-18】将 5/32 和-69.625 表示成 IEEE754 单精度浮点数的格式。

解：①$5/32=0.00101B=1.01B\times 2^{-3}$，按 IEEE754 单精度浮点数的要求，

∵x≥0，∴S=0

∵尾数的有效值为 1.M，∴M=01000000000000000000000B

∵E=127+e，∴E=127+（-3）=124=01111100B

5/32 表示成 IEEE754 单精度浮点数的格式为：

0	01111100	01000000000000000000000

②∵-69.625= -1000101.101B= $-1.000101101B\times 2^6$　∴S=1

M=00010110100000000000000B

E=127+6=133=10000101B，其浮点数格式如下：

1	10000101	00010110100000000000000

【例 2-19】将 IEEE754 单精度浮点数 42E48000H 转换成真值十进制数。

解：按 IEEE754 定义的单精度浮点数格式

单精度浮点数 42E48000H 可表示为：

0	10000101	11001001000000000000000

∴S=1，E=10000101B=133，M=11001001000000000000000B=0.78515625，其浮点数对应的真值为：$(-1)^S \times 2^{E-127} \times 1.M=(-1)^0 \times 2^{133-127} \times 1.78515625=1.78515625\times 2^6=114.25$

（2）64 位双精度浮点数格式（IEEE754 标准）

64 位双精度浮点数格式如图 2-2 所示。

符号位 S（1 位）	阶码 E（8 位）	尾数 M（23 位）

图 2-2　IEEE754 标准 64 位双精度浮点数格式

64 位双精度浮点数所表示的数值 x 为：

$$x = (-1)^S \times 2^{E-1023} \times 1.M$$

2.3.2 浮点数的规格化

浮点数规格化的目的在于：一方面为了提高运算精度，尽可能占满尾数的位数，以保留更多的有效数字；另一方面保证了浮点数的唯一性。例如，对于浮点数 0.001011×2^6，由于 $0．001011\times 2^6=0.101100\times 2^4=0.000001011\times 2^9=\cdots\cdots$，所以，同一个数 0.001011×2^6 就有多种表示，无法保证其唯一性。另外，如果规定尾数为 6 位，0.000001011×2^9 就成了 0.000001×2^9，从而丢掉了有效数字，降低了数的精度。对于基数 R 为 2 的浮点数，其规格化条件是：0.5≤｜M｜＜1。在计算机中，浮点数通常是以规格化形式存储和参加运算的。

如果运算结果出现了非规格化浮点数，则必须对结果进行规格化处理。

2.3.3 浮点数的表示范围

浮点数的格式被确定后，其所表示的数据范围也就确定了。求浮点数的表示范围，实质上是分析出浮点数所表示的最大数、最小数等。下面针对基数 R 为 2 的浮点数（阶码为 m+1 位（包括 1 位阶符），尾数为 n+1 位（包括 1 位数符）），分别讨论不同形式的机器数所表示的规格化和非规格化的最大、最小数。

1、阶码和尾数均采用原码表示

阶码和尾数均采用原码表示时，典型浮点数的机器数和真值如表 2-3 所示。

表 2-3　　阶码和尾数均用原码表示时机器数的规格化和非规格化最大、最小数

典型浮点数	机器数	真值
非规格化最小正数	0　1 11…1　　00…01	$+2^{-n}\times 2^{-(2m-1)}$
规格化最小正数	0　1 11…1　　10…00	$+2^{-1}\times 2^{-(2m-1)}$
最大正数	0　0 11…1　　11…11	$+(1-2^{-n})\times 2^{(2m-1)}$
非规格化最大负数	1　1 11…1　　00…01	$-2^{-n}\times 2^{-(2m-1)}$
规格化最大负数	1　1 11…1　　10…00	$-2^{-1}\times 2^{-(2m-1)}$
最小负数	1　0 11…1　　11…11	$-(1-2^{-n})\times 2^{(2m-1)}$

2、阶码和尾数均采用补码表示

阶码和尾数均采用补码表示时，典型浮点数的机器数和真值如表 2-4 所示。

表 2-4　　阶码和尾数均用补码表示时机器数的规格化和非规格化最大、最小数

典型浮点数	机器数	真值
非规格化最小正数	0　1 00…0　　00…01	$+2^{-n}\times 2^{-2m}$
规格化最小正数	0　1 00…0　　10…00	$+2^{-1}\times 2^{-2m}$
最大正数	0　0 11…1　　11…11	$+(1-2^{-n})\times 2^{(2m-1)}$
非规格化最大负数	1　1 00…0　　11…11	$-2^{-n}\times 2^{-2m}$
规格化最大负数	1　1 00…0　　01…11	$-(2^{-1}+2^{-n})\times 2^{-2m}$
最小负数	1　0 11…1　　00…00	$-1\times 2^{(2m-1)}$

3、阶码和尾数分别采用移码和补码表示

阶码和尾数分别采用移码和补码表示时，典型浮点数的机器数和真值如表 2-5 所示。

表 2-5 阶码用移码尾数用补码表示时机器数的规格化和非规格化最大、最小数

典型浮点数	机器数	真值
非规格化最小正数	0　0 00…0　　00…01	$+2^{-n}\times 2^{-2m}$
规格化最小正数	0　0 00…0　　10…00	$+2^{-1}\times 2^{-2m}$
最大正数	0　1 11…1　　11…11	$+(1-2^{-n})\times 2^{(2m-1)}$
非规格化最大负数	1　0 00…0　　11…11	$-2^{-n}\times 2^{-2m}$
规格化最大负数	1　0 00…0　　01…11	$-(2^{-1}+2^{-n})\times 2^{-2m}$
最小负数	1　0 11…1　　00…00	$-1\times 2^{(2m-1)}$

2.3.4 浮点数的机器零

计算机在对浮点数处理的过程中，首先是判“0”，因为，当一个浮点数是“0”时，运算可以简化。机器零是指如果一个浮点数的尾数为全 0，则不论阶码为何值；或者如果一个浮点数的阶码小于它所能表示的最小值，则不论其尾数为何值，计算机都会把这种浮点数当作零看待。特别是当阶码用移码表示、尾数用补码表示时，如果阶码为它所表示的最小数-2^m(阶码为 m 位)且尾数为 0 时，其阶码的移码形式为全 0，尾数的补码形式也为全 0，这时的浮点数的机器数形式也为全 0，从而有利于简化机器的判 0 电路。

2.4 非数值数据的表示

2.4.1 ASCII 码与字符串

字符和字符串是计算机中用得最多的非数值数据，人们利用字符和字符串编写程序、表达文字及各类信息，以便于与计算机进行交流。为了使计算机硬件能够识别和处理字符，必须对字符按一定规则用二进制进行编码。

1、ASCII 码

ASCII 码（American Standard Code for Information Interchange 美国标准信息交换码）是目前国际上广泛使用的字符编码。ASCII 码为 7 位二进制编码，可表示 128 个字符，包括 10 个数字字符（0~9）、52 个英文字母（大、小写各 26 个）、34 个常用符号和 32 个控制字符（如空格符 NUL、回车符 CR、换行符 LF 等）。表 2-6 所示为 ASCII 编码。7 位 ASCII 码用 $b_6b_5b_4b_3b_2b_1b_0$ 表示，其中 $b_6b_5b_4$ 为 ASCII 码的高 3 位，是字符所在表中列的编码；$b_3b_2b_1b_0$ 为 ASCII 码的低 4 位，是字符所在表中行的编码。

表 2-6　　ASCII 码字符表

$b_6b_5b_4$ / $b_3b_2b_1b_0$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P		p
0001	SOH	DC_1	!	1	A	Q	a	q
0010	STX	DC_2	″	2	B	R	b	r
0011	ETX	DC_3	#	3	C	S	c	s
0100	EOT	DC_4	$	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	SI	US	/	?	O	_	o	DEL

在计算机中，一个字符用一个字节表示，由于 ASCII 码只有 7 位，因此，表示字符的字节的最高位 b_7 有以下用法：

（1）b_7 置 0，用于表示字符的 ASCII 码

（2）b_7 置 1，用于表示汉字的编码

（3）b_7 用作奇偶校验位

2、字符串

字符串是指连续的一串字符。由于一个字符占一个字节单元，所以字符串在存储时要占连续的多个字节单元，至于存储顺序不同的机器可以有不同的规定，字符串中的字符既可以从低位字节向高位字节顺序存放，也可以从高位字节向低位字节顺序存放。

2.4.2 BCD 码与十进制数串

1、8421BCD 码

BCD（Binary Coded Decimal）码是对一位十进制数所进行的编码，其目的在于：一方

面方便快捷地将十进制数转换成二进制形式的数据；另一方面直接实现对十进制数的运算。BCD 码的编码方法有多种，较常用的有 8421 BCD 码、2421 BCD 和余 3 码。常见的 BCD 码如表 2-7 所示。

表 2-7　　BCD 码

一位十进制数	8421BCD 码	2421BCD 码	余 3 码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

8421BCD 码是 4 位二进制编码，“8421”是指 4 个位置的位权分别为 8、4、2、1，有且只有 10 个编码。由于 4 位二进制的编码有且只有 16 个编码，所以，8421BCD 码与 4 位的二进制编码之间并不存在一一对应的关系。在对十进制数直接进行运算时，其结果可能会出现非 8421BCD 码，此时需要及时进行加 6（0110）修正，否则，运算的结果会出错。

另外需要注意的是 BCD 码是对一位十进制数所进行的编码，对于多位十进制数，则应用多个 BCD 码组合起来表示。例如十进制数 147，对应的 8421BCD 码为 0001 0100 0111。

2、十进制数

（1）非压缩型十进制数

非压缩型十进制数主要应用于显示、打印等非数值处理过程中，是将十进制数以字符串的形式进行表示，即把十进制数中的每一位数字以及数的正、负符号都当作一个字符，用一个字节表示其 ASCII 码。根据数的符号（+、-）的 ASCII 码所存储的位置不同，非压缩型十进制数又可分为前分隔非压缩型十进制数和后嵌入非压缩型十进制数。

1）前分隔非压缩型十进制数

前分隔非压缩型十进制数的表示方法是：数的符号占一个字节，正号“+”的 ASCII 码为 2BH，负号“-”的 ASCII 码为 2DH，位于十进制数的数字位之前。

【例 2-20】采用前分隔非压缩型十进制数，写出十进制数+256 和-1716 在内存的存储形式。

解：+256 在内存的存储形式为：

2B	32	35	36
“+”	“2”	“5”	“6”

十进制数+256 在内存中共占 4 个字节单元。

-1716 在内存的存储形式为：

2D	31	37	31	36
“-”	“1”	“7”	“1”	“6”

十进制数-1716 在内存中共占 5 个字节单元。

2）后嵌入非压缩型十进制数

后嵌入非压缩型十进制数的表示方法是：数的符号不单独占一个字节，而是嵌入到最低位数字的编码中。其嵌入规则是：若数的符号为正号“+”，则最低位数字的 ASCII 码保持不变；若数的符号为负号“-”，则最低位数字的 ASCII 码加上 40H。

【例 2-21】采用后嵌入非压缩型十进制数，写出十进制数+256 和-1716 在内存的存储形式。

解：+256 在内存的存储形式为：

32	35	36
“2”	“5”	“6”

十进制数+256 在内存中共占 3 个字节单元

-1716 在内存的存储形式为：

31	37	31	76
“1”	“7”	“1”	“6”

十进制数-1716 在内存中共占 4 个字节单元

（2）压缩型十进制数

压缩型十进制数既能节省存储空间，又便于直接进行十进制运算，是被广泛采用的十进制数表示方法。此方法是用一个字节存放两位十进制数的 BCD 码，数的符号（+、-）占半个字节，并存放在最低位数字的 BCD 码之后。数的符号所占的半个字节规定为：“1100”表示正号（+），“1101”表示负号（-）。为了保证此表示方法不出现只有半个字节的情况，规定压缩型十进制数数字个数和 1 位符号之和必须是偶数，否则，在最高位数字之前补一个数字 0。

【例 2-22】采用压缩型十进制数，写出十进制数+256 和-1716 在内存的存储形式。

解：+256 在内存的存储形式为：

0010	0101	0110	1100
“2”	“5”	“6”	“+”

十进制数+256 在内存中共占 2 个字节单元

-1716 在内存的存储形式为：

0001	0111	0001	0110	1101
“1”	“7”	“1”	“6”	“-”

十进制数-1716 在内存中共占 3 个字节单元

2.5 数据校验码

数据在存储和传送的过程中，难免不会出现错误，为了减少和避免错误的发生，一方面从硬件方面采取措施，提高硬件本身的抗干扰能力和可靠性；另一方面在数据编码上采取检错纠错的措施，使得机器能够自动发现错误，甚至能纠正错误。我们把这种具有检测错误或带有自动纠错能力的数据编码称为数据校验码。其原理是在数据中加入一些校验位，组成数据校验码，通过检查数据校验码的合法性来判断是否出错或进行纠错。常用的数据校验码有奇偶校验码、海明校验码和循环冗余校验码（CRC）等。

2.5.1 奇偶校验码

奇偶校验码是一种最简单、最常用的校验码，被广泛应用于内存的读写校验和串行通信。

1、奇偶校验码的编码

对于 n 位二进制数 $D=d_{n-1}d_{n-2}\cdots d_1d_0$，添加一位校验位 P，P 的位置可以在数据 D 的最高位 d_{n-1} 之前，也可以在数据 D 的最低位 d_0 之后，并且，P 是数据 D 的函数，即 $\mathbf{P=}f\mathbf{(D)}$。于是，n 位数据 D 和 1 位校验位 P 便构成了一个 n+1 位的奇偶校验码。奇偶校验码根据 P ~ D 之间的函数不同，可分为奇校验和偶校验。

偶校验（Even）：加入一位校验位 P_{Even} 后，使得 n+1 位的奇偶校验码中“1”的个数为偶数，因此，$P_{Even}=d_{n-1}\oplus d_{n-2}\oplus\cdots d_1\oplus d_0$。

奇校验（Odd）：加入一位校验位 P_{Odd} 后，使得 n+1 位的奇偶校验码中“1”的个数为奇数，因此，$P_{Odd}=\overline{P_{Even}}$。

2、奇偶校验码的校验

发送端将一个奇偶校验码发送后，接收端需要对接收到信息进行校验，判断所接收到的数据是否有误，以决定其取舍。具体校验方法为：如果接收端接收到一个 1 的个数为偶数的奇校验码，或接收端接收到一个 1 的个数为奇数的偶校验码，则表示接收端所接收的是一个有错的校验码。通过设置出错标志 E（E=0，表示无误；E=1，表示有误），很容易实现校验。

偶校验的出错标志 E_{Even}：$E_{Even}= d_{n-1} \oplus d_{n-2} \oplus \cdots d_1 \oplus d_0 \oplus P_{Even}$

奇校验的出错标志 E_{Odd}：$E_{Odd}= d_{n-1} \oplus d_{n-2} \oplus \cdots d_1 \oplus d_0 \oplus P_{Odd}$

3、奇偶校验码的纠错能力

根据奇偶校验码的校验可知，奇偶校验码只能发现一位出错或奇数位同时出错，至于出错的位置是无法确定的，因此，奇偶校验码无法实现纠错功能。尽管如此，由于一位出错的概率远远高于多位同时出错的概率，奇偶校验码能够满足一般可靠性的要求，因此，奇偶校验码一种最简单、最常用的数据校验码，它被广泛应用于对存储器中数据的检查或传送数据的检查。

2.5.2 海明校验码

海明校验码是由 Richard Hamming 于 1950 年提出的，到目前还被广泛应用。它是在奇偶校验的基础上，通过合理增加校验位的位数，组成海明校验码，不仅能够实现发现多位出错，而且能够对一位出错进行自动纠正。

1、海明校验码中校验位的位数

设数据的位数为 n，校验位的位数为 k，则组成的海明校验码为 n+k 位。校验时 k 位校验位的编码共有 2^k 种状态，其中只有一种状态用来表示数据无误，其余 2^k -1 种状态用来表示数据有措。由于海明校验码共 n+k 位，所以校验位的位数 k 与数据的位数 n 应满足：

$$2^k -1 \geq n+k \qquad (2\text{-}12)$$

根据式(2-12)可计算出具有检测一位出错并能纠正一位错误能力的海明校验码中 k 与 n 的具体对应关系，如表 2-8 所示。

表 2-8　　海明校验码中校验位的位数 k 与数据的位数 n 的关系

n	最小的 k
1 ~ 4	4
5 ~ 11	5
12 ~ 26	6
27 ~ 57	7
58 ~ 120	8

2、海明校验码的编码

设 n 位的数据为 $D_{n-1}\cdots D_1D_0$，k 位的校验位为 $P_{k-1}\cdots P_1P_0$，组成的海明校验码为 $H_mH_{m-1}\cdots H_1$，其中 m=n+k，海明校验码的编码规则为：

（1）校验位 Pi 在海明校验码 $H_mH_{m-1}\ldots H_1$ 中的位置

每一个校验位 P_i 在海明校验码中被安置在位号为 2^i 的位置（$i \in \{0，1，\cdots，k\text{-}1\}$），校验位 P_{k-1} 可能不满足这个关系，将其安置在海明校验码的最高位。其余各位为数据位，并按从低位到高位依次排列。

（2）海明校验码 $H_mH_{m-1}\ldots H_1$ 的校验

海明校验码 $H_mH_{m-1}\cdots H_1$ 中的每一位 H_j 是由多个校验位 P_i 来校验的（$i \in \{0,1,\cdots,k\text{-}1\}$），其关系是被校验的每一位位号 j 等于校验它的各个校验位的位号之和。

（3）校验位 P_i 的形成

根据规则（2），找出校验位 P_i 参与了对哪些数据位的校验，反过来，被校验的这些数据位按照奇偶校验原理形成校验位 P_i，下面以一个字节的数据为例来讨论海明校验码。由于数据为 8 位，即 n=8，按式（2-12）可求出校验位的位数 k=5，海明校验码的总位数为 13

位，表示为 $H_{13}H_{12}\cdots H_1$。

按规则（1）知，海明校验码为 $P_4D_7D_6D_5D_4P_3D_3D_2D_1P_2D_0P_1P_0$

按规则（2）可以得到表 2-9 所示的结果。

表 2-9　　海明校验码与校验位的关系

海明校验码	数据位/校验位	参与校验的校验位位号	被校验的海明校验码的位号等于校验它的各个校验位位号之和
H_1	P_0	1	1=1
H_2	P_1	2	2=2
H_3	D_0	1，2	3=1+2
H_4	P_2	4	4=4
H_5	D_1	1，4	5=1+4
H_6	D_2	2，4	6=2+4
H_7	D_3	1，2，4	7=1+2+4
H_8	P_3	8	8=8
H_9	D_4	1，8	9=1+8
H_{10}	D_5	2，8	10=2+8
H_{11}	D_6	1，2，8	11=1+2+8
H_{12}	D_7	4，8	12=4+8
H_{13}	P_4	13	13=13

按规则（3）可以将形成校验位 P_i 的数据位进行分组，即校验组的分组。如果需要区分是两位出错还是一位出错，可以补充一个总校验位 P_4

P_0：D_0，D_1，D_3，D_4，D_6

P_1：D_0，D_2，D_3，D_5，D_6

P_2：D_1，D_2，D_3，D_7

P_3：D_4，D_5，D_6，D_7

P_4：D_0，D_1，D_2，D_3，D_4，D_5，D_6，D_7，P_0，P_1，P_2，P_3

若采用偶校验，则各个校验位的形成方法为：

$P_0=D_0\oplus D_1\oplus D_3\oplus D_4\oplus D_6$

$P_1=D_0\oplus D_2\oplus D_3\oplus D_5\oplus D_6$

$P_2=D_1\oplus D_2\oplus D_3\oplus D_7$

$P_3=D_4\oplus D_5\oplus D_6\oplus D_7$

$P_4=D_0\oplus D_1\oplus D_2\oplus D_3\oplus D_4\oplus D_5\oplus D_6\oplus D_7\oplus P_0\oplus P_1\oplus P_2\oplus P_3$

3、海明校验码的校验

在接收端收到海明校验码后，需对 k 个校验位分别进行 k 组奇偶校验，以判断数据传输是否出错。分组校验后，校验结果形成 k 位的状态字 $S_{k-1}\cdots S_1S_0$，若状态字 $S_{k-1}\cdots S_1S_0$ 全 0，则表示数据传输无误；否则，则表示数据传输有误，状态字 $S_{k-1}\cdots S_1S_0$ 所对应的十进制数值就是出错位的位号，将该位取反，即可实现纠错。下面以一个字节数据的海明校验码为例来说明校验过程。校验时按偶校验，状态字 $S_4\cdots S_1S_0$ 形成如下：

$S_0= P_0\oplus D_0\oplus D_1\oplus D_3\oplus D_4\oplus D_6$

$S_1= P_1\oplus D_0\oplus D_2\oplus D_3\oplus D_5\oplus D_6$

$S_2= P_2\oplus D_1\oplus D_2\oplus D_3\oplus D_7$

$S_3= P_3\oplus D_4\oplus D_5\oplus D_6\oplus D_7$

$S_4= P_4\oplus D_0\oplus D_1\oplus D_2\oplus D_3\oplus D_4\oplus D_5\oplus D_6\oplus D_7\oplus P_0\oplus P_1\oplus P_2\oplus P_3$

状态字 $S_4\cdots S_1S_0$ 能够反映 13 位海明校验码的出错情况，错误个数为偶数时 S_4 为 0，错误个数为奇数时 S_4 为 1，以此可以判断是一位出错还是两位出错。

2.5.3 循环冗余校验码

在一位出错的概率远远大于多位同时出错概率的情况下，奇偶校验是一种简单有效的检查方法，但在串行通信中，常常会遇到瞬间干扰，造成多位数据会同时发生错误，此时，使用奇偶校验码意义不大，通常采用循环冗余校验码，即 CRC 码（Cyclic Redundancy Check）。它是一种具有很强检错纠错能力的校验码。

1、循环冗余校验思想

用 n 位的数据 $d_{n-1}\cdots d_1d_0$ 构成一个 x 的 n-1 次幂的多项式 M(ｘ), 即 $M(x)= d_{n-1}x^{n-1}+d_{n-2}x^{n-2}+\cdots+d_1x^1+d_0$，再用一个约定的多项式 G（ｘ）去除 M（ｘ），可得式（2-13)所示的关系式：

$$\frac{M(x)}{G(x)}= Q(x)+\frac{R(x)}{G(x)} \qquad (2\text{-}13)$$

其中 Q（ｘ）为商数，R（ｘ）为余数。

由式（2-13)可得：M（ｘ）- R（ｘ）=Q（ｘ）×G（ｘ） （2-14）

发送端将 M（ｘ）- R（ｘ）作为校验码进行发送，接收端收到校验码后，用双方约定的多项式 G（ｘ）去除，如果能够被整除，即余数为 0，则表示数据传输无误，否则，则表示数据传输有误。

2、模 2 运算

由式（2-14）可知，M（ｘ）- R（ｘ）是减法运算，需要涉及到借位，难以用简单的拼装方法实现编码，为了回避借位运算，CRC 码采用了模 2 运算。所谓模 2 运算，是指以按位模 2 加运算为基础的二进制四则运算。简单地讲，模 2 运算是一种不考虑进位和借位的运算。

（1）模 2 加减

模 2 加减是按异或规则实现按位加，不进位。其运算规则是：

0±0=0，0±1=1，1±0=1，1±1=0

【例 2-23】按模 2 加减规则，计算 1101+1011，1101-1011

解：根据模 2 加减规则可得：

①1101+1011=0110 ②1101-1011=0110

由此可见，模 2 的加法运算等于模 2 的减法运算。

（2）模 2 乘

模 2 乘就是在进行乘法运算时，按模 2 加的运算规则，对各个位积进行模 2 加法。

例如 1010×1011=1001110。

（3）模 2 除

模 2 除就是在进行除法运算时，按模 2 减的运算求部分余数，计算时不进行借位。

例如 1011001÷1101=1100，余数为 101。

3、CRC 码的编码

设 n 位数据 $d_{n-1}\cdots d_1d_0$ 所构成一个 x 的 n-1 次幂的多项式 M(ｘ), 即 $M(x)= d_{n-1}x^{n-1}+d_{n-2}x^{n-2}+\cdots+d_1x^1+d_0$，约定一个生成多项式 G（ｘ）为 x 的 k 次幂的多项式，即 $G(x)= C_kx^k+C_{k-1}x^{k-1}+\cdots+C_1x^1+C_0$，G（ｘ）作为除数的余数多项式 R（ｘ）最高次幂为 x^{k-1}，因此 R（ｘ）相当于 k 位余数形成的多项式。把 k 位余数拼接在 n 位数据位之后，便构成 n＋k 位的 CRC 码。CRC 码的编码只是简单地拼接，关键是如何根据数据去产生 k 位余数。

由于 n 位数据位后附加了 k 位余数，n 位数据所构成一个 x 的 n-1 次幂的多项式 M（ｘ）

相当于提高了 x^{k-1} 阶，即 M（x）• x^{k-1}。然后按模 2 除法，用 M（x）• x^{k-1} 除以 G（x），便得到余数多项式 R（x），该多项式中 x 的各个次幂的系数就是 k 位余数。

$$\frac{M(x)\cdot x^{k-1}}{G(x)}=Q(x)+\frac{R(x)}{G(x)} \qquad (2\text{-}15)$$

由式（2-15)可得：M（x）• x^{k-1}–R（x）=Q（x）×G（x）

根据模 2 加减运算规则可得：M（x）• x^{k-1}+R（x）=Q（x）×G（x） （2-16)

【例 2-24】设生成多项式 G（x）= x^3+x^1+1，求 4 位数据 1101 的 CRC 码。

解：∵G（x）= x^3+x^1+1，是 4 位的多项式

∴余数是 3 位

根据式（2-16）可得 3 位的余数为 101，因此，4 位数据 1101 的 CRC 码为 1101101。

在 CRC 码中，由 n 位数据和 k 位校验构成的 n+k 位的 CRC 码，也称为（n+k，n）码。在**【例 2-24】**中，由于 n=4，n+k=7，故称（7，4）码。

4、CRC 码的校验

接收端收到 CRC 码后，用双方约定的生成多项式 G（x）做模 2 除，如果除得的余数为 0，则表示接收到的信息中没有错误，否则，则表示接收到的信息中某一位发生错误。出错的位置不同相应的余数也不同，因此，可以根据余数来确定出错的位置，进而实现纠错功能。

在**【例 2-24】**中，对于 4 位数据 1101 的（7，4）码出错情况如表 2-10 所示。

表 2-10　　4 位数据 1101 基于 G（x）= x^3+x^1+1　（7，4）码的校验

错误状态	F_7 F_6 F_5 F_4 F_3 F_2 F_1 D_3 D_2 D_1 D_0 R_2 R_1 R_0	余数	错误位置
无误	1 1 0 1 0 0 1	000	无
有误	1 1 0 1 0 0 **0**	001	1
有误	1 1 0 1 0 **1** 1	010	2
有误	1 1 0 1 **1** 0 1	100	3
有误	1 1 0 **0** 0 0 1	011	4
有误	1 1 **1** 1 0 0 1	110	5
有误	1 **0** 0 1 0 0 1	111	6
有误	**0** 1 0 1 0 0 1	101	7

以表 2-10 中的第二行为例，把余数 001 补 0 后再除以 G（x），得到的第二次余数为 010，第二次余数为 010 再补 0 后除以 G（x），得到的第三次余数为 100，按此继续做除法，不难发现所得的余数依次为 011、110、111、101，最后又回到 001，这就是为什么将此校验码称之为循环校验码的原因。根据循环码的这一特点，当接收端收到 CRC 码与生成多项式 G（x）做模 2 除得到的余数不为 0 时，可以一边对余数补 0 继续做模 2 除，同时使被检测的 CRC 码循环左移，当出现余数为 101 时，原来出错的位已移到 F_7 的位置，将其取反进行纠错后，在下次移位时送回 F_1，将 CRC 码继续循环左移，移满一个循环后，可得到一个纠错后的 CRC 码。

例如，若接收端收到 CRC 码字为 1010111，用 G（x）= x^3+x^1+1 做模 2 除，得到的余数为 100，说明接收到的信息中某一位发生错误。将此余数补 0 后再除以 G（x），同时使 CRC 码循环左移。进行如此运算 4 次后，得到的余数为本 101，此时，CRC 码也循环左移了 4 位，变成了 1111010。出错位已移至 F_7，将其取反可得 0111010。再将它循环左移 3 位，补足 7 次（一个循环），出错位又回到 F_3，便得到一个正确的码字 1010011。

5、生成多项式 G（x）

由前面的内容可知，在 CRC 码的形成和校验过程中，生成多项式 G（x）起着非常重要的作用。采用的生成多项式 G（x）不同，所形成的 CRC 码字、码距不同，其检错和纠错能力也不同。关于生成多项式 G（x），并非任何一个 x 的 k 次幂多项式都可以作为生成多项式，它应满足以下要求：

（1）任何一位发生错误都应使余数不为 0

（2）不同位发生错误时，其余数应不同

（3）对余数做模 2 除，应能使余数循环

为便于选择满足以上要求的生成多项式 G（x），表 2-11 列出了常用的生成多项式。

表 2-11　　常用的生成多项式 G（x）

CRC 码长	数据位数	码距	多项式 G（x）
7	4	3	x^3+x+1
7	4	3	x^3+x^2+1
7	3	4	$x^4+x^3+x^2+1$
7	3	4	x^4+x^2+x+1
15	11	3	x^4+x+1
15	7	5	$x^8+x^7+x^6+x^4+1$
15	5	7	$x^{10}+x^8+x^5+x^4+x^2+x+1$
31	26	3	x^5+x^2+1
31	21	5	$x^{10}+x^9+x^8+x^6+x^5+x^3+1$
63	57	3	x^6+x+1
63	51	5	$x^{12}+x^{10}+x^5+x^4+x^2+1$

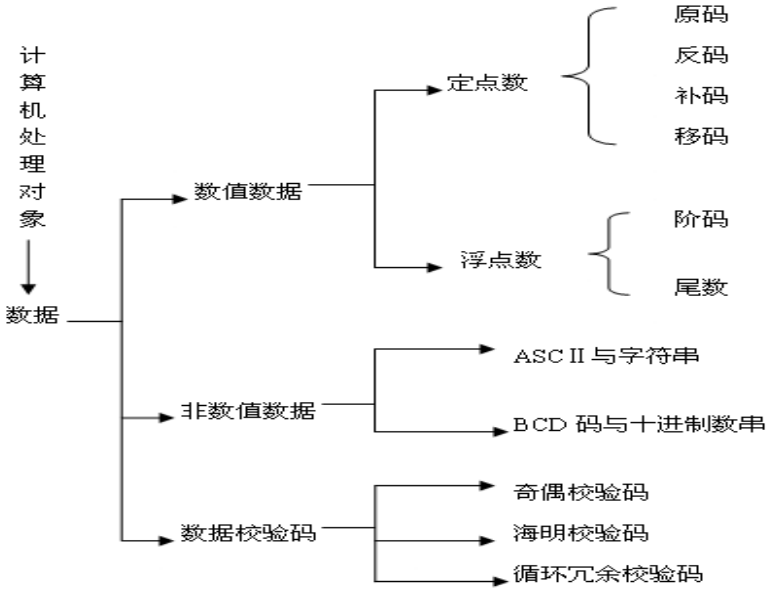
在数据通信与网络中，由于数据位数多，上千位二进制数据位构成一帧，为检测信息传输的正确与错误，广泛采用 CRC 码进行校验。这时所使用的生成多项式的次幂比较高，常用的生成多项式有：

$G(x)=x^{16}+x^{15}+x^2+1$

$G(x)=x^{16}+x^{12}+x^5+1$

$G(x)=x^{32}+x^{26}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

关　联



习 题

2.1 数制转换

（1）213=（　　）B=（　　）H

（2）0F8EH=（　　）B=（　　）D

（3）69.75=（　　）B=（　　）H

（4）10111011B =(　　)D=（　　）H

（5）110000010.0101 B = (　　)D =(　　)H

2.2 设机器字长为 8 位（含一位符号位），分别求出$[x]_{原}$、$[x]_{反}$、$[x]_{移}$、$[x]_{补}$、$[-x]_{补}$、$[\frac{1}{2}x]_{补}$。

（1）x=+119　（2）x=-56　（3）x=-1　（4）x=+25/128　（5）-37/64

2.3 已知 x 二进制真值，试求$[x]_{补}$、$[-x]_{补}$、$[\frac{1}{2}x]_{补}$、$[\frac{1}{4}x]_{补}$、$[2x]_{补}$、$[4x]_{补}$、$[-2x]_{补}$、$[-\frac{1}{4}x]_{补}$。

（1）x=+0.0101101　（2）x=-0.0001011　（3）x=-1

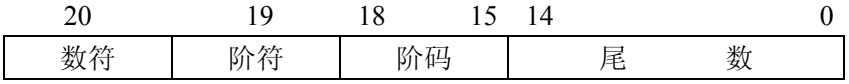
2.4 已知$[x]_{补}$，求 x 的真值。

（1）$[x]_{补}$=0.1010111（2）$[x]_{补}$=1.1110010（3）$[x]_{补}$=0.0101101　（4）$[x]_{补}$=1.1111111

2.5 设十进制数 $x=(+124.625)\times 2^{-10}$。

（1）写出 x 对应的二进制定点小数表示形式。

（2）若机器的浮点数表示格式为：



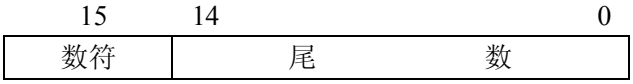
其中阶码和尾数的基数均为 2。

1）写出阶码和尾数均采用原码表示时的机器数形式。

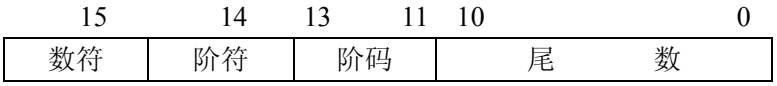
2）写出阶码和尾数均采用补码表示时的机器数形式。

2.6 设某机字长为 16 位，数据表示格式如下。

定点数：



浮点数：



分别写出下列数据表示形式中所能表示的最小正数、最大正数、最大负数、最小负数（绝对值最大的负数）以及浮点规格化最小正数、最大负数所对应的十进制真值。

（1）原码表示的定点整数

（2）补码表示的定点整数

（3）阶码与尾数均用原码表示的浮点数

（4）阶码与尾数均用补码表示的浮点数

（5）阶码为移码、尾数用补码表示的浮点数

2.7 设 2.6 题的浮点数格式中，阶码与尾数均用补码表示，分别写出下面用十六进制书写的浮点数所对应的十进制真值。

（1）FFFFH　　（2）C400H

2.8 写出下列十进制数的 IEEE754 标准 32 位单精度浮点数的机器数表示形式。

（1）0.15625　　（2）-0.15625

2.9 写出 IEEE754 标准 32 位单精度浮点数所能表示的最小规格化正数和最大规格化负数的机器数表示形式。

2.10 写出下列十六进制的 IEEE754 单精度浮点数所代表的十进制数真值。

（1）42E48000H （2）3F880000H （3）00800000H （4）C7F00000H

2.11 设有两个正浮点数：$N1=M_1\times2^{e1}$, $N2=M_2\times2^{e2}$

（1）若 e1＞e2 ,是否有 $N_1>N_2$？

（2）若 M_1、M_2 均为规格化数，上述结论是否正确？

2.12 设一个 6 位二进制小数 $x=0.a_1a_2a_3a_4a_5a_6$,x≥0,请回答：

（1）若要 $x\geq\frac{1}{8}$, $a_1a_2a_3a_4a_5a_6$ 需要满足什么条件？

（2）若要 $x>\frac{1}{2}$, $a_1a_2a_3a_4a_5a_6$ 需要满足什么条件？

（3）若要 $\frac{1}{4}\geq x>\frac{1}{16}$, $a_1a_2a_3a_4a_5a_6$ 需要满足什么条件？

2.13 分别用前分隔数字串、后嵌入数字串和压缩的十进制数串形式表示下列十进制数。

（1）+79 （2）-635 （3）+2008 （4）-8510

2.14 下面是两个字符（ASCII 码）的海明校验码（偶校验），请检测它们是否有错？如果有错，请加以改正，并写出相应的正确 ASCII 码所代表的字符。

（1）10111010011 （2）10001010110

2.15 设数据信息是 1010110010001111，选择生成多项式为 G(x)=100101,求出数据的 CRC 码。

2.16 完成下列 8421 BCD 码与其它数制/码制的转换。

（1）（1001 0011）BCD= ( )D =( ) B

（2）(0011 0111 0110.011)BCD = ( )D =( ) B

（3）11010011B=( )BCD

（4）110000010.0101B = ( )BCD

（5）151.625 = ( ) B = ( )BCD

2.17 计算下列 BCD 码的和，并按规则进行十进制调整。

（1）96+87 （2）1556+298

2.18 选择题

（1）某机器字长 64 位，1 位符号位，63 位尾数。若采用定点小数表示，则最大正小数为（ ）。

A、$+(1-2^{-64})$ B、$+(1-2^{-63})$ C、2^{-64} D、2^{-63}

（2）设[x]补=$1.x_1x_2x_3x_4x_5x_6x_7x_8$，当满足（ ）时，$x>-\frac{1}{2}$ 成立。

A、$x_1=1$, $x_2x_3x_4x_5x_6x_7x_8$ 至少有一个为 1 B、$x_1=0$, $x_2x_3x_4x_5x_6x_7x_8$ 至少有一个为 1

C、$x_1=1$, $x_2x_3x_4x_5x_6x_7x_8$ 任意 D、$x_1=0$, $x_2x_3x_4x_5x_6x_7x_8$ 任意

（3）在下列机器数中，哪种表示方式下零的表示形式是惟一的（ ）。

A、原码 B、补码 C、反码 D、都不是

（4）下列论述中，正确的是（ ）。

A、已知$[x]_{原}$求$[x]_{补}$的方法是在$[x]_{原}$的末位加 1

B、已知$[x]_{补}$求$[-x]_{补}$的方法是在$[x]_{补}$的末位加 1

C、已知$[x]_{原}$求$[x]_{补}$的方法是将尾数连同符号位一起取反，再在末位加 1

D、已知$[x]_{补}$求$[-x]_{补}$的方法是将尾数连同符号位一起取反，再在末位加 1

（5）IEEE754 标准规定的 32 位浮点数格式中，符号位为 1 位，阶码为 8 位，尾数为 23 位，则它所能表示的最大规格化正数为（ ）。

A、$+(2-2^{-23})\times 2^{+127}$　B、$+(1-2^{-23})\times 2^{+127}$　C、$+(2-2^{-23})\times 2^{+255}$　D、$2^{+127}\times 2^{-23}$

（6）浮点数的表示范围取决于（　）。

A、阶码的位数　　B、尾数的位数

C、阶码采用的编码　　D、尾数采用的编码

（7）假定下列字符码中有奇偶校验位，但没有数据错误，采用奇校验位的编码是（　）。

A、10000010　　B、11010110　　C、11010111　　D、10111011

2.19 填空题

（1）设某机器字长为 8 位（含一符号位），若$[x]_{补}$=11001001，则 x 所表示的十进制数的真值为_____，$[\frac{1}{4}x]_{补}$=_____；若$[y]_{移}$=11001001，则 y 所表示的十进制数的真值为_____，y 的原码$[y]_{原}$=_____。

（2）在带符号数的编码方式中，零的表示是惟一的有_____和_____。

（3）若$[x_1]_{补}$=10110111，$[x_2]_{原}$=1.01101，则数 x_1 的十进制数真值是_____，x_2 的十进制数真值是_____。

（4）设某浮点数的阶码为 8 位，用移码表示；尾数为 24 位，采用规格化补码表示。则该浮点数能表示的最大正数的阶码为_____，尾数为_____；规格化最小负数的阶码为_____，尾数为_____。

（5）设有效信息位的位数为 k，校验位的位数为 r，则能够检测出一位出错并能自动纠错的海明校验码应满足的关系是_____。

第 3 章　运算方法与运算器

【内容摘要】

计算机最基本的功能是对数据进行处理。计算机对数据的处理可以归纳为两种基本运算：一是算术运算，二是逻辑运算。算术运算是指加、减、乘、除运算，逻辑运算是指逻辑移位和逻辑与、逻辑或、逻辑非等运算。无论哪一种运算，都会涉及到运算对象（数据），结合数据的不同表示方法，本章将侧重于以定点小数的补码来讨论计算机中各类数据的运算方法与硬件实现问题。

【学习要点】

- 定点小数补码的四则运算与实现
- 浮点数的四则运算与实现
- 十进制数的加减运算与实现
- 逻辑运算与 ALU
- 定点运算器与浮点运算器

3.1　定点数的算术运算与实现

定点数的机器数有原码、反码、补码等形式，在本节中我们重点是以定点小数的补码来讨论定点数的加、减、乘、除运算。只有补码运算可以将减法转换为加法，补码的符号位同补码的数值位一样参与运算，运算规则简单，易于实现，所以现代计算机中的数据普遍采用补码。

3.1.1 定点数加减运算

1、补码加减运算规则

假设定点小数 x 的补码为$[x]_{补}$，定点小数 y 的补码为$[y]_{补}$。

则有：　$[x+y]_{补}= [x]_{补}+[y]_{补}$　（mod 2）　　(3-1)

$[x-y]_{补}= [x]_{补}- [y]_{补}=[x]_{补}+[-y]_{补}$　(mod 2)　　(3-2)

根据式（3-1）和式（3-2），补码加减运算的规则如下：

规则 1，参与运算的数是补码，运算结果也是补码；

规则 2，符号位同数值位一样参与运算，其结果的符号取决于运算结果的补码；

规则 3，加法运算时，直接将两个的补码相加；减法运算时，需要将减数变补（即$[y]_{补}$求$[-y]_{补}$），然后再与被减数的补码相加。这是由于数据采用补码形式后，减法运算转换成了加法运算，所以计算机实现加减运算只需加法器，无需减法器。我们在进行减法运算时也只能按加法进行运算；

规则 4，补码的运算是有模的运算，如果运算结果超出了模（即符号位的运算产生了进位），则将模自动丢失。这样处理的结果并非是错误的，还需要进一步进行溢出判断。

2、溢出与溢出判断

溢出是指运算结果超出了机器所能表示的数的范围。如果运算结果超出了机器所能表示的最大正数，称为上溢（正溢出）；如果运算结果超出了机器所能表示的最小负数，称为下溢（负溢出）。一旦出现溢出，机器将无法表示正确结果，因此，计算机在运算的过程中必须正确判断溢出并及时处理。

（1）单符号判别法

符号判别法是根据参与运算数的符号和运算结果的符号进行判断溢出。设$[x]_{补}=x_0.x_1x_2\cdots x_{n-1}$，$[y]_{补}=y_0.y_1y_2\cdots y_{n-1}$，　$[z]_{补}= [x+y]_{补}=z_0.z_1z_2\cdots z_{n-1}$，溢出标志为 V，则溢出标志 V 与符号位（$x_0$、$y_0$、$z_0$）之间的关系如表 3-1 所示。根据真值表 3-1，可得出溢出的判断条件为：

$$V= x_0y_0\overline{z_0} + \overline{x_0}\ \overline{y_0}z_0=（x_0 \oplus y_0）（x_0 \oplus y_0）$$

表 3-1　　溢出标志（V）与符号位（x_0、y_0、z_0）之间的真值表

x_0	y_0	z_0	V
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

（2）双符号判别法

双符号判别法也称为变形补码法，是根据结果的变形补码来进行判断溢出。变形补码的定义如下：设 x 为 n 位的二进制数据，式（3-3）和式（3-4）分别给出了 x 为定点小数 $\pm 0.x_1x_2\cdots x_{n-1}$ 和 x 为定点整数 $\pm x_1x_2\cdots x_{n-1}$ 的变形补码定义。

定点小数变形补码的定义：

$$[x]_{变补}= \begin{cases} x & 0\le x<1 \\ 4+x & -1\le x<0 \end{cases} \pmod 4 \qquad (3\text{-}3)$$

定点整数变形补码的定义：

$$[x]_{变补}= \begin{cases} x & 0\leqslant x<2^{n-1} \\ 2^{n+1}+x & -2^{n-1}\leqslant x<0 \end{cases} \pmod{2^{n+1}} \qquad (3\text{-}4)$$

由定义可知：

1）变形补码只是比普通补码多了一位符号位，所以，变形补码也称双符号位补码。如：$[x]_{补}=x_0.x_1x_2\cdots x_{n-1}$，则$[x]_{变补}= x_0x_0.x_1x_2\cdots x_{n-1}$，其中，$x_0x_0$ 为 00 表示正数，x_0x_0 为 11 表示负数。

2）变形补码的加减运算与普通补码的加减运算相同，只不过两个符号位都参与了运算，运算结果为：$z_0'z_0.z_1z_2\cdots z_{n-1}$，其中，$z_0'z_0$ 是结果的两个符号位，其含义为：

$z_0'z_0$=00，表示结果为正，无溢出

$z_0'z_0$=11，表示结果为负，无溢出

$z_0'z_0$=01，表示结果正溢出

$z_0'z_0$=10，表示结果负溢出

由此可见，当 $z_0'z_0$ 不一致时，结果便产生溢出。因此，采用变形补码进行运算时，溢出的判断条件为：$V= z_0'\overline{z_0} + \overline{z_0}'z_0= z_0' \oplus z_0$。

（3）进位判别法

当最高有效位产生进位而符号位无进位时，产生正溢；当最高有效位无进位而符号位产

生进位时，产生负溢。故溢出的判断条件为：$V=C_s \oplus C_0$，其中 C_s 为符号位产生的进位，C_0 为最高有效位产生的进位。

3、补码加减运算的步骤

补码的加减运算虽然十分简单，但一定要注意运算步骤的完整。为此归纳出补码的加减运算步骤：

步骤 1，求$[x]_{补}$、$[y]_{补}$、$[-y]_{补}$；

步骤 2，按式（3-1）和式（3-2）进行二进制加法运算。注意：只能按加法进行运算，即使是减法运算，也要将减数变补（即由$[y]_{补}$求$[-y]_{补}$），然后再与被减数相加；

步骤 3，利用溢出判断条件，判断运算结果的正确性。

【例 3-1】已知 x 的真值，用补码的加减运算，计算 x±y。

①x=+0.1010　y=+0.0011　②x=-0.1101　y=-0.1011

解：①$[x]_{补}$=**0**.1010，$[y]_{补}$=**0**.0011，$[-y]_{补}$=**1**.1101

$[x+y]_{补}=[x]_{补}+[y]_{补}$=**0**.1010+**0**.0011=**0**.1101

$[x- y]_{补}=[x]_{补}- [y]_{补}=[x]_{补}+[-y]_{补}$=**0**.1010+**1**.1101=**0**.0111

具体运算过程如下：

$$\begin{array}{r} \mathbf{0}.1010 \\ +\mathbf{0}.0011 \\ \hline \mathbf{0}.1101 \end{array} \qquad \begin{array}{l}\text{进位位}\\ \text{自然丢失}\end{array} \begin{array}{r} \mathbf{0}.1010 \\ +\mathbf{1}.1101 \\ \hline 1\mathbf{0}.0111 \end{array}$$

由单符号判别法可知，x±y 的结果没有发生溢出。

②$[x]_{补}$=**1**.0011，$[y]_{补}$=**1**.0101，$[-y]_{补}$=**0**.1011

$[x+y]_{补}=[x]_{补}+[y]_{补}$=**1**.0011+**1**.0101=**0**.1000

$[x-y]_{补}=[x]_{补}-[y]_{补}=[x]_{补}+[-y]_{补}$=**1**.0011+**0**.1011=**1**.1110

具体运算过程如下：

$$\begin{array}{l}\text{进位位}\\ \text{自然丢失}\end{array} \begin{array}{r} \mathbf{1}.0011 \\ +\mathbf{1}.0101 \\ \hline 1\mathbf{0}.1000 \end{array} \qquad \begin{array}{r} \mathbf{1}.0011 \\ +\mathbf{0}.1011 \\ \hline \mathbf{1}.1110 \end{array}$$

由单符号判别法可知，x+y 的结果为正，而参与运算的两个数是负数，所以结果发生了溢出。而 x-y 的结果不会发生溢出。

【例 3-2】已知 x 的真值，利用变形补码求 x±y。

① x=+0.1010　y=+0. 0011　② x=-0.1101　y=-0.1011

解：①$[x]_{变补}$=**00**.1010，$[y]_{变补}$=**00**.0011，$[-y]_{变补}$=**11**.1101

$[x+y]_{变补}=[x]_{变补}+[y]_{变补}$=00.1010+00.0011=00.1101

$[x-y]_{变补}=[x]_{变补}-[y]_{变补}=[x]_{变补}+[-y]_{变补}$=00.1010+11.1101=00.0111

具体运算过程如下：

$$\begin{array}{r} \mathbf{00}.1010 \\ +\mathbf{00}.0011 \\ \hline \mathbf{00}.1101 \end{array} \qquad \begin{array}{l}\text{进位位}\\ \text{自然丢失}\end{array} \begin{array}{r} \mathbf{00}.1010 \\ +\mathbf{11}.1101 \\ \hline 1\mathbf{00}.0111 \end{array}$$

由双符号判别法可知，x±y 结果的双符号相同，所以没有发生溢出。

②$[x]_{变补}$=**11**.0011，$[y]_{变补}$=**11**.0101，$[-y]_{变补}$=**00**.1011

$[x+y]_{变补}=[x]_{变补}+[y]_{变补}$=11.0011+11.0101=10.1000

$[x-y]_{变补}=[x]_{变补}-[y]_{变补}=[x]_{变补}+[-y]_{变补}$=**11**.0011+**00**.1011=**11**.1110

具体运算过程如下：

$$\begin{array}{l}\text{进位位}\\ \text{自然丢失}\end{array} \begin{array}{r} \mathbf{11}.0011 \\ +\mathbf{11}.0101 \\ \hline 1\mathbf{10}.1000 \end{array} \qquad \begin{array}{r} \mathbf{11}.0011 \\ +\mathbf{00}.1011 \\ \hline \mathbf{11}.1110 \end{array}$$

由双符号判别法可知，x+y 结果的双符号为 **10**，所以运算结果发生了溢出。而 x-y 结果的双符号为 **11**，不会发生溢出。

【例 3-3】已知 x =-0.10111，y=-0.10001，求 $\frac{1}{2}$（x+y）。

解： ∵$[x]_{变补}$=**11**.01001，$[y]_{变补}$=**11**.01111

∴$[\frac{1}{2}x]_{变补}$=**11**.10100，$[\frac{1}{2}y]_{变补}$=**11**.10111

$[\frac{1}{2}(x+y)]_{变补}=[\frac{1}{2}x]_{变补}+[\frac{1}{2}y]_{变补}$=**11**.10100+**11**.10111=**11**.01011

溢出判断：由于结果的双符号位相同，未产生溢出，运算结果正确。

∴ $\frac{1}{2}$(x+y)的真值为-0.10101

4、补码加减运算的逻辑实现

（1）一位全加器

设一位全加器 FA（Full Addition）的输入为 A_i 和 B_i，进位输入为 C_i，结果输出为 S_i，进位输出为 C_{i+1}，则一位全加器的真值表如表 3-2 所示。

表 3-2　　一位全加器的真值表

输入			输出	
A_i	B_i	C_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

根据表 3-2 可得输出与输入之间的逻辑表达式：

$$S_i=(A_i \oplus B_i) \oplus C_i \qquad (3\text{-}5)$$

$$C_{i+1}=A_iB_i+(A_i \oplus B_i) \cdot C_i \qquad (3\text{-}6)$$

由式（3-5）和式（3-6），一位全加器逻辑电路如图 3-1(b)所示。假设单位逻辑门电路(与门、与非门、或门、或非门)的时间延迟为 T，则一个异或门的时间延迟为 3T，一位全加器产生 zi 的时间延迟为 6T，进位传递（由 Ci 到 Ci+1）时间为 2T。

（2）n 位补码加减运算的逻辑电路

n 位补码加减运算的逻辑电路需要 n 个一位全加器 FA、外加控制信号 M（M=0，实现加法；M=1，实现减法）组成。其逻辑电路如图 3-1(a)所示。

为了进一步讨论进位关系，将式(3-5) 中进位输出 C_{i+1} 的关系式进行改变如下形式：

$$C_{i+1}=A_iB_i+(A_i \oplus B_i) \cdot C_i=G_i+P_i \cdot C_i$$

其中，$G_i= A_iB_i$，为进位生成函数；$P_i= A_i \oplus B_i$，为进位传递函数。下面讨论 n 位补码加减运算的进位关系。

$$\begin{aligned} C_1 &= G_0+P_0 \cdot C_0 \\ C_2 &= G_1+P_1 \cdot C_1 \\ &\vdots \\ C_{n-1} &= G_{n-2}+P_{n-2} \cdot C_{n-2} \end{aligned} \qquad (3\text{-}7)$$

由式（3-7）可知，高位的进位 C_i 仅仅是它的低一位进位 C_{i-1} 的函数，也就是说，只有

产生出 C_{i-1} 后才能产生 C_i，这种进位关系称为行波进位或串行进位。行波进位的补码加减运算时间主要取决于进位的传递时间。为了提高补码的加减运算速度，进位关系往往采用并行进位，即高位的进位 C_i 都是 C_0 的函数。

$$C_1 = G_0 + P_0 \cdot C_0$$

$$C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1G_0 + P_1P_0 \cdot C_0$$

$$\vdots \qquad (3\text{-}8)$$

$$C_{n-1} = g_{n-2} + P_{n-2} \cdot C_{n-2} = G_{n-2} + P_{n-2}G_{n-3} + \cdots + P_{n-3}P_{n-4}\cdots P_1G_0 + P_{n-3}P_{n-4}\cdots P_1 P_0 \cdot C_0$$

由式（3-8）可知，n 越大，进位逻辑电路越复杂，因此，往往将 n 位按 4 位进行分组，不难证明，n/4-1 组间的进位信号 G^* 和传递信号 P^*，的表达式。

$$P^*_0 = P_3P_2P_1P_0 \qquad G^*_0 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$$

$$P^*_1 = P_7P_6P_5P_4 \qquad G^*_1 = G_7 + P_7G_6 + P_7P_6G_6 + P_7P_6P_5G_4$$

$$\vdots \qquad \vdots$$

$$P^*_{n/4-1} = P_{n-1}P_{n-2}P_{n-3}P_{n-4} \qquad G^*_{n/4-1} = G_{n-1} + P_{n-1}G_{n-2} + P_{n-1}P_{n-2}G_{n-3} + P_{n-1}P_{n-2}P_{n-3}G_{n-4}$$

它们同样满足 $C^*_{i+1} = G^*_i + P^*_i C^*_i$

各组的进位为：

$$C^*_1 = G^*_0 + P^*_0 \cdot C_0$$

$$C^*_2 = G^*_1 + P^*_1G^*_0 + P^*_1 P^*_0 \cdot C_0$$

$$\vdots$$

$$C^*_{n/4-1} = G^*_{n/4-2} + P^*_{n/4-2}G^*_{n/4-3} + \cdots + P^*_{n/4-2}P^*_{n/4-3}\cdots P^*_1G^*_0 + P^*_{n/4-2}P^*_{n/4-3}\cdots P^*_1 P^*_0 \cdot C_0$$

n 位定点数的加减运算进行分组后，组内可以按上述并行进位关系进行加法，而组间既可以是串行进位关系，也可以是组间并行进位关系，在这里我们不再进行婺述，感兴趣的同学可以看有关的参考书。

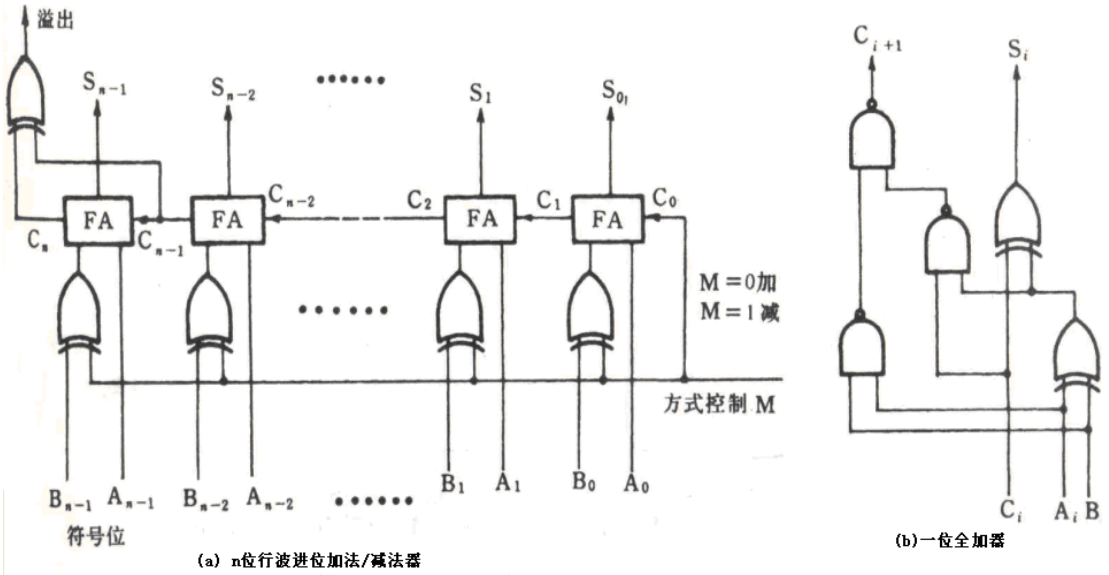


图 3-1　n 位补码加减运算电路

在实际的运算器中，参与运算的数据和运算结果通常存放在寄存器中，控制器控制将数据送入加法器，并进行加法运算，再将运算结果送回寄存器。如图 3-2 所示，给出了带有寄存器的实现补码加减运算的逻辑电路。其中寄存器 A、B 分别存放参与运算的两个补码数据，运算结果送回寄存器 A 保存。

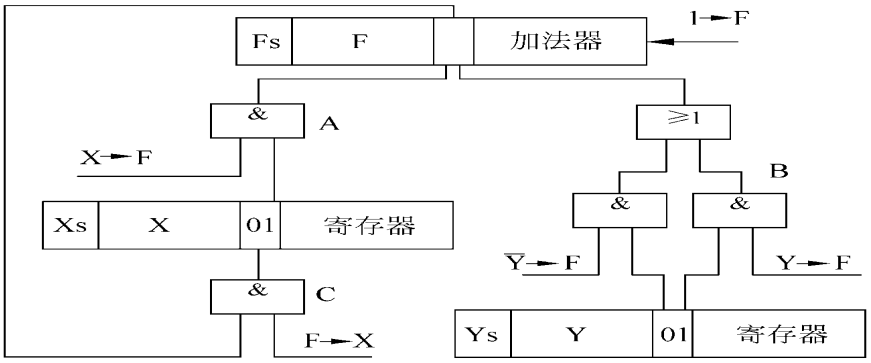


图 3-2　　实现补码加减运算的逻辑电路

3.1.2 定点数乘法运算

1、补码与真值的关系

设$[x]_{补}=x_0.x_1x_2\cdots x_{n-1}$

当 x≥0 时，则$[x]_{补}=0.x_1x_2\cdots x_{n-1}=\sum_{i=1}^{n-1} x_i\times 2^{-i}$

当 x＜0 时，则$[x]_{补}=1.x_1x_2\cdots x_{n-1}=2+x$

$\therefore x=[x]_{补}-2=-1+0.x_1x_2\cdots x_{n-1}=-1+\sum_{i=1}^{n-1} x_i\times 2^{-i}$

所以，对于任何数 x，其补码与真值的关系如式（3-9）所示。

$$x=-x_0+\sum_{i=1}^{n-1} x_i\times 2^{-i}=-x_0+0.x_1x_2\cdots x_{n-1} \qquad (3\text{-}9)$$

2、补码的右移

$\because x=-x_0+\sum_{i=1}^{n-1} x_i\times 2^{-i}=-x_0+0.x_1x_2\cdots x_{n-1}$

$$\frac{1}{2}x=-\frac{1}{2}x_0+\frac{1}{2}\sum_{i=1}^{n-1} x_i\times 2^{-i}=-x_0+\frac{1}{2}x_0+\frac{1}{2}\sum_{i=1}^{n-1} x_i\times 2^{-i}$$

$$=-x_0+\sum_{i=0}^{n-1} x_i\times 2^{-(i+1)}=-x_0+0.x_0x_1x_2\cdots x_{n-1}=x_0.x_0x_1x_2\cdots x_{n-1}$$

$\therefore [\frac{1}{2}x]_{补}=x_0.x_0x_1x_2\cdots x_{n-1}$

3、补码的一位乘法

设被乘数$[x]_{补}=x_0.x_1x_2\cdots x_{n-1}$，乘数$[y]_{补}=y_0.y_1y_2\cdots y_{n-1}$，则：

$$[x\cdot y]_{补}=[x]_{补}\times(-y_0+\sum_{i=1}^{n-1} y_i\times 2^{-i})$$

$$=[x]_{补}\times[-y_0+y_12^{-1}+y_22^{-2}+\cdots+y_{n-1}2^{-(n-1)}]$$

$$=[x]_{补}\times[-y_0+(y_1-y_12^{-1})+(y_22^{-1}-y_22^{-2})+\cdots+(y_{n-1}2^{-(n-2)}-y_{n-1}2^{-(n-1)}]$$

$$=[x]_{补}\times[(y_1-y_0)+(y_2-y_1)2^{-1}+\cdots+(y_{n-1}-y_{n-2})2^{-(n-2)}+(0-y_{n-1})2^{-(n-1)}]$$

在乘数$[y]_{补}$的最低位 y_{n-1} 之后再添加一位 y_n，并令 $y_n=0$，在乘法运算开始时令部分积 $Z_0=0$，便可以得到布斯（Booth）递推公式：

$[Z_0]_{补}=0$

$[Z_1]_{补}=\{[Z_0]_{补}+(y_n-y_{n-1})\ [x]_{补}\}\times 2^{-1}$，$y_n=0$

$[Z_2]_{补}=\{[Z_1]_{补}+(y_{n-1}-y_{n-2})\ [x]_{补}\}\times 2^{-1}$

⋮

$[Z_i]_{补}=\{[Z_{i-1}]_{补}+(y_{n-i+1}-y_{n-i})\ [x]_{补}\}\times 2^{-1}$

⋮

$[Z_{n-1}]_{补}=\{[Z_{n-2}]_{补}+(y_2-y_1)\ [x]_{补}\}\times 2^{-1}$

$[Z_n]_{补}=\{[Z_{n-1}]_{补}+(y_1-y_0)\ [x]_{补}\}$

Booth 算法规则如表 3-3 所示，其操作流程如图 3-3 所示，最后一步（i=n）时不进行右移操作，即得到$[x\cdot y]_{补}$

表 3-3　　Booth 算法规则

$y_{i+1}y_i$	$(y_{i+1}-y_i)\ [x]_{补}$	对上次部分积的操作
00	0	上次部分积加 0，右移一位
01	$-[x]_{补}$	上次部分积加$[-x]_{补}$，右移一位
10	$[x]_{补}$	上次部分积加$[x]_{补}$，右移一位
11	0	上次部分积加 0，右移一位

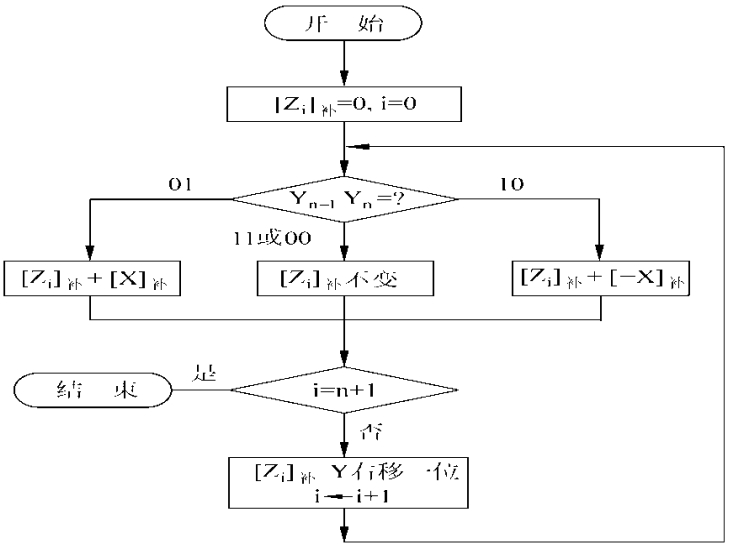


图 3-3　　补码一位乘 Booth 算法流程图

【例 3-4】已知 x 和 y 的真值，x=-0.1101　　y=0.1011，利用补码的一位乘求 x・y。

解：$[x]_{补}$=11.0011，$[y]_{补}$=00.1011，$[-x]_{补}$=00.1101

	部分积	乘数	判别位	说明
	0 0. 0 0 0 0	0 1 0 1	1 0	开始，Y_n=0 判别位为10
$+[-X]_{补}$	0 0. 1 1 0 1			部分积$+[-X]_{补}$
	0 0. 1 1 0 1			
	0 0. 0 1 1 0	1 0 1 0	1 1	右移1位，判别位为11
	0 0. 0 0 1 1	0 1 0 1	0 1	右移1位，判别位为01
$+[X]_{补}$	1 1. 0 0 1 1			$+[X]_{补}$
	1 1. 0 1 1 0			
	1 1. 1 0 1 1	1 0 1 0	1 0	右移1位，判别位为10
$+[-X]_{补}$	0 0. 1 1 0 1			$+[-X]_{补}$
	0 0. 1 0 0 0			
	0 0. 0 1 0 0	0 0 0 1	0 1	右移1位，判别位为01
$+[X]_{补}$	1 1. 0 0 1 1			$+[X]_{补}$
	1 1. 0 1 1 1	0 0 0 1		最后一位不移位

∴$[x\cdot y]_{补}$=11.01110001，x・y 的真值为-0.10001111B

通过【例 3-4】补码一位乘法的运算过程分析，用硬件实现补码一位乘法时，只需三个寄存器和一个 n 位加法器，实现补码一位乘法的硬件逻辑电路如图 3-4 所示。图中 A、B、C 为三个寄存器，在运算开始时，寄存器 A 用于存放乘积和部分积的高位部分，初始内容为 0；寄存器 C 用于存放乘数和部分积的低位部分，初始内容为乘数；y_{n-1} 和 y_n 用于控制进行+[x]$_{补}$还是+[-x]$_{补}$。寄存器 B 用于存放被乘数，可以在 y_{n-1} 和 y_n 控制下输出原码和反码，当执行+[x]$_{补}$时，输出原码；当执行+[-x]$_{补}$时，输出反码。C_x 是乘法控制触发器，C_x=1，允许产生移位脉冲，以控制进行乘法运算；否则，停止进行乘法运算。计数器 i 用于记录乘法次数，在运算开始时，计数器 i 清 0，每进行一次运算，计数器 i 进行加 1，当计数到 n+1 时，结束运算。另外，当 i=n 时，将计数器 i 清 0，使得在进行第 n+1 次运算时，不再进行移位。

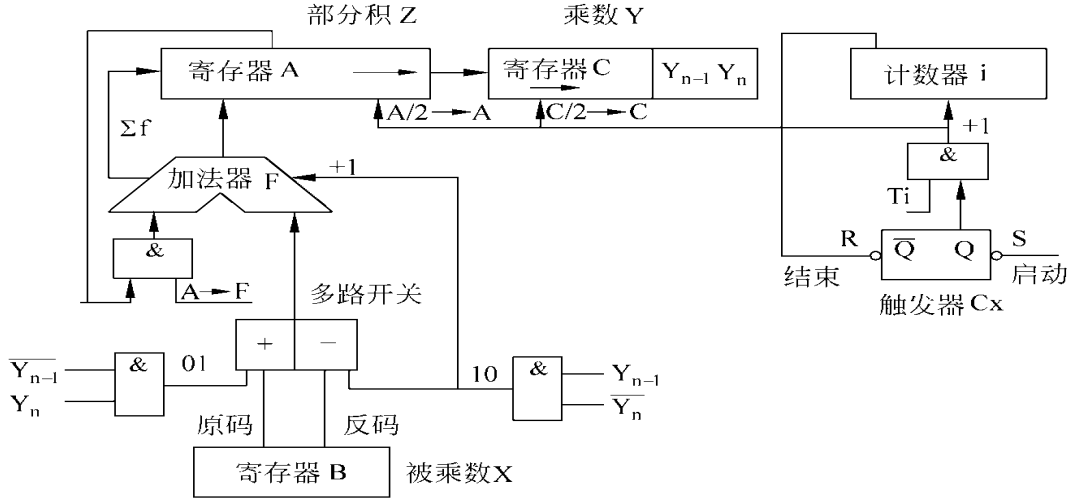


图 3-4　　实现补码一位乘的逻辑电路

4、补码的二位乘法

根据布斯递推公式，将两步合并为一步，即可导出补码的两位乘法公式。

设上次部分积为[Z_i]$_{补}$，本次的部分积则为：

$$[Z_{i+1}]_{补}=\{[Z_i]_{补}+(y_{n-i}-y_{n-i-1})\,[x]_{补}\}\times 2^{-1}$$

下次的部分积为：

$$[Z_{i+2}]_{补}=\{[Z_{i+1}]_{补}+(y_{n-i-1}-y_{n-i-2})\,[x]_{补}\}\times 2^{-1}$$

把[Z_{i+1}]$_{补}$带入[Z_{i+2}]$_{补}$中，可得：

$$[Z_{i+2}]_{补}=\{\{[Z_i]_{补}+(y_{n-i}-y_{n-i-1})\,[x]_{补}\}\times 2^{-1}+(y_{n-i-1}-y_{n-i-2})\,[x]_{补}\}\times 2^{-1}$$

$$=\{[Z_i]_{补}+(y_{n-i}-2y_{n-i-2}+y_{n-i-1})\,[x]_{补}\}\times 2^{-2}$$

$(y_{n-i}-2y_{n-i-2}+y_{n-i-1})$式中 y_{n-i}、 y_{n-i-2}、 y_{n-i-1} 的初始值分别为附加位 y_n=0、y_{n-1}、y_{n-2}。其值及其对应的操作如表 3-4 所示。

表 3-4　　$(y_{n-i}-2y_{n-i-2}+y_{n-i-1})$与[$P_{i+2}$]$_{补}$的关系

y_{n-i}、 y_{n-i-2}、 y_{n-i-1}	$(y_{n-i}-2y_{n-i-2}+y_{n-i-1})\,[x]_{补}$	[Z_{i+2}]$_{补}$
0　0　0	0	$\{[Z_i]_{补}+0\}\times 2^{-2}$
0　0　1	$+[x]_{补}$	$\{[Z_i]_{补}+[x]_{补}\}\times 2^{-2}$
0　1　0	$-2[x]_{补}$	$\{[Z_i]_{补}+2[-x]_{补}\}\times 2^{-2}$
0　1　1	$-[x]_{补}$	$\{[Z_i]_{补}+[-x]_{补}\}\times 2^{-2}$
1　0　0	$+[x]_{补}$	$\{[Z_i]_{补}+[x]_{补}\}\times 2^{-2}$
1　0　1	$+2[x]_{补}$	$\{[Z_i]_{补}+2[x]_{补}\}\times 2^{-2}$
1　1　0	$-[x]_{补}$	$\{[Z_i]_{补}+[-x]_{补}\}\times 2^{-2}$
1　1　1	0	$\{[Z_i]_{补}+0\}\times 2^{-2}$

【例 3-5】 已知 x=-0.1101　　y=-0.0101，利用补码的二位乘求 x・y。

解：$[x]_{补}$=11.0011，$[y]_{补}$=11.1011，$[-x]_{补}$=00.1101

	部分积	乘数 附加位	判别位	说　明
		↓		
	00.0000	111011 0	110	组合值为 −1,加$[-X]_{补}$
$+[-X]_{补}$	00.1101			右移 2 位
	00.1101	111011 0		
	00.0011	0111 101	101	组合值为 −1,加$[-X]_{补}$
$+[-X]_{补}$	00.1101			右移 2 位
	01.0000	011110 1		
	00.0100	000111 1	111	组合值为 0,加 0

∴$[x \cdot y]_{补}$=00.01000001，x・y 的真值为+0.01000001B

5、并行乘法运算

由于乘法运算大约占全部算术运算的 1/3 左右，因此采用高速乘法器件，无论从速度上还是从效益上都是十分必要的。随着大规模集成电路的迅速发展，出现了各种形式的流水阵列乘法器，它们属于并行乘法器。在此只简单介绍阵列乘法器的基本原理。

（1）无符号数阵列乘法器

设 A 为 m 位的无符号二进制整数($A=a_{m-1}\cdots a_1a_0$)，B 为 n 位的无符号二进制整数($B=b_{n-1}\cdots b_1b_0$)，A、B 的数值（真值）分别是 a、b，其乘积为 P（P = A·B），是一个 m+n 位无符号二进制整数，即 $P=p_{m+n-1}\cdots p_1p_0$，数 P 的真值为 p，则：

$$a=\sum_{i=0}^{m-1} a_i\times 2^i \qquad b=\sum_{j=0}^{n-1} b_j\times 2^j$$

$$p=a\cdot b=\left(\sum_{i=0}^{m-1} a_i\times 2^i\right)\cdot\left(\sum_{j=0}^{n-1} b_j\times 2^j\right)=\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(a_ib_j)\times 2^{i+j}$$

$$=\sum_{k=0}^{m+n-1} p_k\times 2^k$$

位积 $a_ib_j\{\ a_ib_j \mid 0\le i\le m-1$ 和 $0\le j\le n-1\}$可以用 m×n 个逻辑与门并行产生，乘积 p_k 可由相同位权（2^k）的位积相加，再加来自低位的进位产生。这样利用位积相加来实现乘法运算的器件称为阵列乘法器，如图 3-5 所示。该阵列乘法器从根本上避免了一位乘和两位乘中所需的大量重复的相加和移位操作，从而提高了乘法运算的速度。

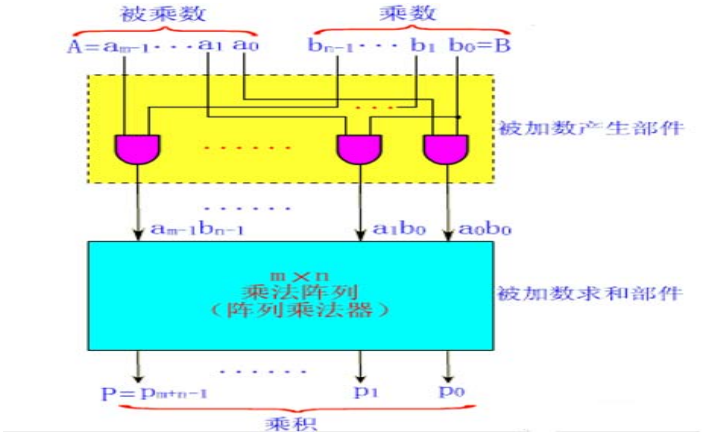


图 3-5　　m×n 位不带符号的阵列乘法器

（2）带符号数阵列乘法器

带符号数阵列乘法器是在无符号数阵列乘法器的基础上增加了符号处理和求补电路，所以也称为符号求补的阵列乘法器。

1）求补器

按照扫描来进行的求补操作，对于 n 位带符号定点整数 $A=a_{n-1}\cdots a_1a_0$，从 A 的最低位 a_0 开始向高位进行扫描，直到找到第一个“1”（例如 $a_i=1$，$0\le i\le n-1$），则该位置的 1 以及比 a_i 低的所有位都保持不变，其余的位（比比 a_i 高的所有位，不包括 a_i 位）都求反，即实现了对 A 的求补。第 i 扫描的输出为 C_i，则 $C_i=a_i+C_{i-1}$，$C_{-1}=0$。求补器的输出 $a_i^*=a_i\oplus EC_{i-1}$，$0\le i\le n-1$，E 为控制信号，当 E=1 时，启动求补器进行求补；当 E=0 时，$a_i^*=a_i$，显然，数的符号位可以作为控制信号 E。如图 3-6 所示，给出了一个 4 位的对 2 求补器电路。

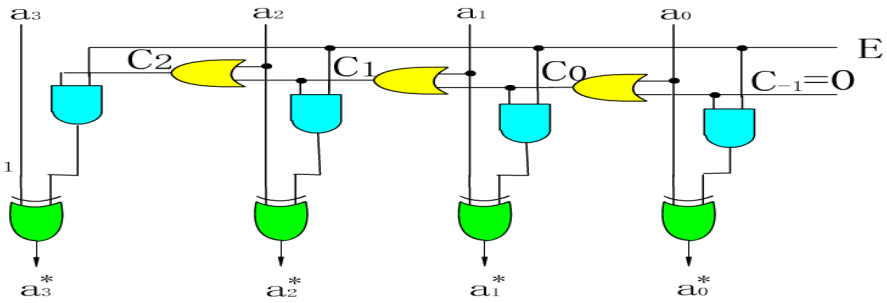


图 3-6　对 2 求补器电路

2）带求补器的阵列乘法器

设 $A=a_na_{n-1}\cdots a_1a_0$，$B=b_nb_{n-1}\cdots b_1b_0$，A 和 B 为 n+1 位带符号定点整数。则 A 与 B 的乘积为 P，P 为一个 2n 位带符号数。

$P=A\cdot B=P_{2n-1}P_{2n-2}\cdots P_1P_0$

$P_{2n}=a_n\oplus b_n$

其中 P_{2n} 为乘积的符号位。如图 3-7 所示，给出了带求补器的阵列乘法器。该带求补器的阵列乘法器使用了三个求补器，两个算前求补器是对 A 和 B 求绝对值，然后，将两个数的绝对值送入无符号数阵列乘法器中进行相乘，即得到 2n 位乘积的绝对值 $P_{2n-1}P_{2n-2}\cdots P_1P_0$。最后，再根据异或门输出的符号位 P_{2n}，控制算后求补器对 $P_{2n-1}P_{2n-2}\cdots P_1P_0$ 求补，即得到 2n+1 位的乘积 $P_{2n}P_{2n-1}P_{2n-2}\cdots P_1P_0$。

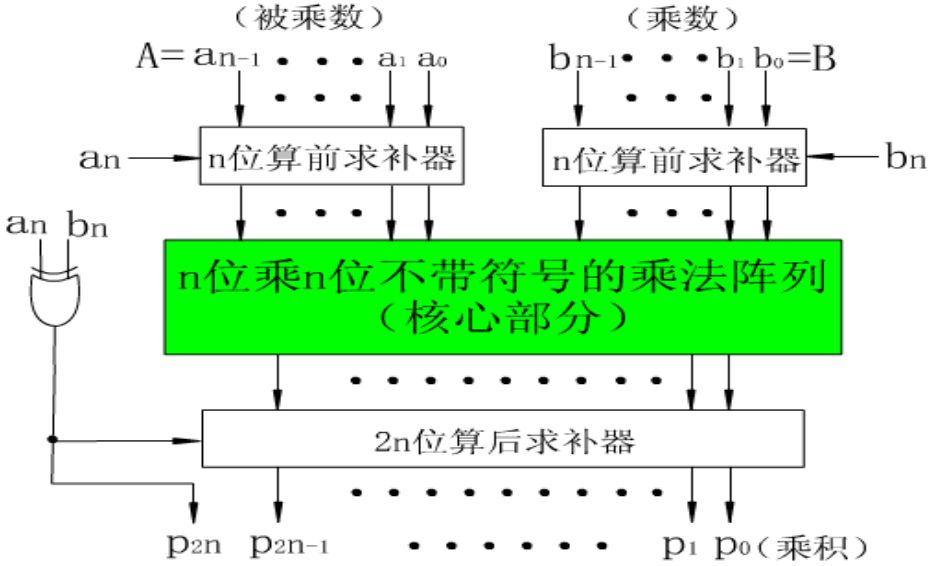


图 3-7 （n+1）×（n+1）位带求补器的阵列乘法器

3.1.3 定点数除法运算

定点数除法可以分为恢复余数法和不恢复余数两种，我们在定点数原码除法中讨论恢复余数的除法。由于后者采用得较多，在定点数补码除法中讨论不恢复余数的除法（加减交替法）。

1、原码的恢复余数法

设被除数为 x，其原码为$[x]_{原}=x_0.x_1x_2\cdots x_{n-1}$，除数为 y，其原码为$[y]_{原}=y_0.y_1y_2\cdots y_{n-1}$，则有：

商 q=x/y，其原码为$[q]_{原}=(x_0 \oplus y_0)+(0.x_1x_2\cdots x_{n-1}/0.y_1y_2\cdots y_{n-1})$

商的符号 $q_0=(x_0 \oplus y_0)$，商的数值部分是$(0.x_1x_2\cdots x_{n-1}/0.y_1y_2\cdots y_{n-1})$，实质上是两个正数求商的运算，类似于十进制数的除法运算，其算法：

（1）判断 x<y?

如果 x<y，则商的整数位上 0，x 的低位补 0，得余数 r_0

（2）比较 r_0 与 $2^{-1}y$

如果 $r_0 \geq 2^{-1}y$，表示够减，则小数点后的第一位商上 1，做 $r_0-2^{-1}y$，得余数 r_1；如果 $r_0<2^{-1}y$，表示不够减，则小数点后的第一位商上 0，不做 $r_0-2^{-1}y$，其余数 $r_1=r_0$

（3）比较 r_1 与 $2^{-2}y$

如果 $r_1 \geq 2^{-2}y$，表示够减，则小数点后的第二位商上 1，做 $r_1-2^{-2}y$，得余数 r_2；如果 $r_1<2^{-2}y$，表示不够减，则小数点后的第二位商上 0，不做 $r_1-2^{-2}y$，其余数 $r_2=r_1$

（4）最后比较 r_{n-2} 与 $2^{-(n-1)}y$

如果 $r_{n-2} \geq 2^{-(n-1)}y$，表示够减，则小数点后的第 n-1 位商上 1，做 $r_{n-2} \geq 2^{-(n-1)}y$，得余数 r_{n-1}；如果 $r_{n-2}<2^{-(n-1)}y$，表示不够减，则小数点后的第 n-1 位商上 0，不做 $r_{n-2}-2^{-(n-1)}y$，其余数 $r_{n-1}=r_{n-2}$

机器的比较必须先做减法，若余数为正，表示够减；否则，表示不够减。不够减时，由于机器已经做过了减法运算，因此，必须恢复原来的余数后再继续往下运算，所以这种除法称为恢复余数的除法。事实上，不够减时不必要恢复余数，可以根据余数的符号继续往下进行运算，这种除法称为不恢复余数的除法，也叫加减交替法。

【例 3-6】 已知 x=0.1011　　y=0.1101，利用恢复余数法求 x÷y。

解：$[-y]_{补}=11.0011$

	被除数	商	操作说明
	1011	0000	X
	10110	0000	被除数左移一位，即 2×X
+	110011	0000	减除数，即 $+[-Y]_{补}$
	001001	0001	余数 R_i 为正，商 1
	010010	0010	左移 1 位，成 $2R_i$
+	110011	0010	减除数，即 $+[-Y]_{补}$
	000101	0011	余数 R_i 为正，商 1
	001010	0110	左移 1 位，成 $2R_i$
+	110011	0110	减除数，即 $+[-Y]_{补}$
	111101	0110	余数 R_i 为负，商 0
+	1101	0110	+Y，即恢复余数
	001010	0110	得到余数 $2R_i$
	010100	1100	$2\times(2R_i)$（左移 1 位）
+	110011	1100	减除数，即 $+[-Y]_{补}$
	00.0111	1101	余数 R_i 为正，商 1

x÷y 的商为 0.1101B，余数为 0.0111B

2、补码的加减交替法

在进行补码的除法运算时，由于被除数和除数都采用的补码形式，在判断是否够减时，需要比较它们的绝对值的大小，同符号采用加法，不同符号采用减法。在被除数的绝对值小于除数的绝对值（商不溢出）的情况下，补码的加减交替法算法如下：

（1）判符号位

首先判断被除数与除数的大小，来确定是进行加法还是减法运算。如果被除数与除数同符号，用被除数减去除数；否则，被除数加上除数。然后，判断余数与除数的符号是否相同，来决定商的符号位是 1 还是 0。如果余数与除数的符号相同，上商“1”；否则，上商“0”。所上的商即为结果的符号位。

（2）求商的数值位

如果上次上商 1，将余数左移一位后减去除数；如果上次上商 0，将余数左移一位后加上除数。然后判断本次操作后的余数，如果余数与除数的符号相同，上商“1”；否则，上商“0”。如此重复进行 n-1 次（设数值位是 n 位）。

（3）商的最后一位一般采用恒值 1 法

对于定点小数而言，最低位置的 1 相当于 2^{-n}，因此，恒值 1 法所产生的最大误差为$\pm 2^{-n}$。

若要求商的精度较高，可按第（2）步再进行一次操作以求得商的第 n 位。当除不尽时，若商为负，则在商的最低位加 1，使商从反码转换成补码；若商为正，商的最低位不加 1。

【例 3-7】 已知 x=0.1011　y=0.1101，利用补码的加减交替法求 x÷y。

解：$[x]_{补}$=00.1011，$[y]_{补}$=00.1101，$[-y]_{补}$=11.0011

被除数／（余数）	商	操作说明
0 0 1 0 1 1	0 0 0 0 0	开始
+ 1 1 0 0 1 1		两数同号，+$[-Y]_{补}$
1 1 1 1 1 0	0 0 0 0 0	余数与除数异号，上商 0
1 1 1 1 0 0	0 0 0 0	左移一位
+ 0 0 1 1 0 1		因上次商 0，+$[Y]_{补}$
0 0 1 0 0 1	0 0 0 0 1	余数与除数同号，上商 1
0 1 0 0 1 0	0 0 0 1	左移一位
+ 1 1 0 0 1 1		因上次商 1，+$[-Y]_{补}$
0 0 0 1 0 1	0 0 0 1 1	余数与除数同号，上商 1
0 0 1 0 1 0	0 0 1 1	左移一位
+ 1 1 0 0 1 1		因上次商 1，+$[-Y]_{补}$
1 1 1 1 0 1	0 0 1 1 0	余数与除数异号，上商 0
1 1 1 0 1 0	0 1 1 0	左移一位
+ 0 0 1 1 0 1		因上次商 0，+$[Y]_{补}$
0 0 0 1 1 1	0 1 1 0 1	余数与除数同号，上商 1

x÷y 商的补码为 00.1101，x÷y 余数的补码为 00.0111。

3、并行除法运算

并行除法运算类似于并行乘法运算，采用了大规模集成电路——阵列除法器。阵列除法器有多种，如不恢复余数的阵列除法器、补码阵列除法器等。这里以不恢复余数的阵列除法器为例，来介绍并行除法器的组成。

（1）可控的加/减单元（CAS）

可控的加/减单元（CAS）的逻辑电路如图 3-8 所示。它具有 4 个输入端和 4 个输出端，

当输入线 P=0 时，CAS 做加法运算；当输入线 P=1 时，CAS 做减法运算。

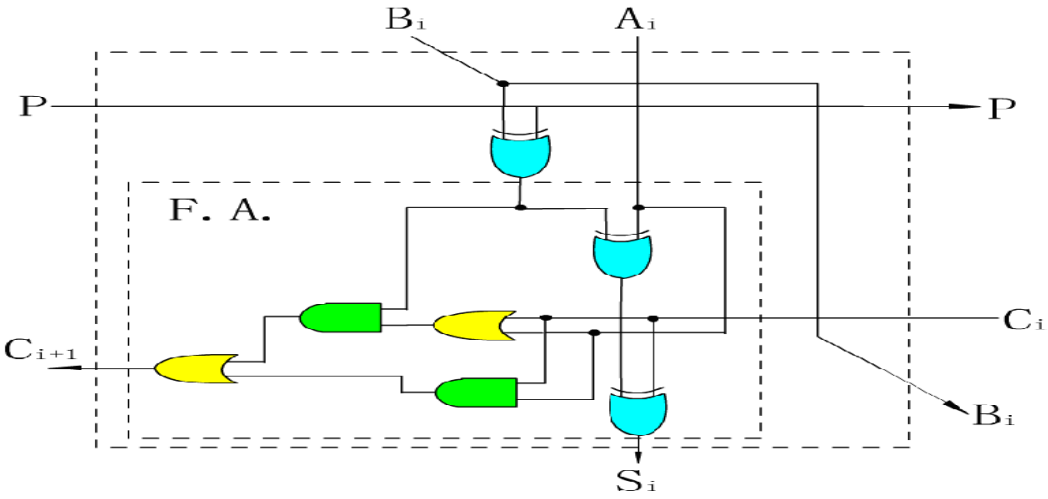


图 3-8　可控的加/减单元逻辑电路

CAS 单元输入与输出之间逻辑关系表达式如式（3-10）所示。

$S_i=A_i \oplus (B_i \oplus P) \oplus C_i$

$C_{i+1}=(A_i+C_i) \cdot (B_i \oplus P)+A_iC_i$　　（3-10）

当 P=0 时，逻辑关系表达式（3-10）等同于式（3-11），这是我们很熟悉的一位全加器的逻辑表达式。

$S_i=A_i \oplus B_i \oplus C_i$

$C_{i+1}=A_iB_i+B_iC_i+A_iC_i$　　（3-11）

当 P=1 时，逻辑关系表达式（3-10）等同于式（3-12）。

$S_i=A_i \oplus \overline{B_i} \oplus C_i$

$C_{i+1}=A_i\overline{B_i}+\overline{B_i}C_i+A_iC_i$　　（3-12）

其中，$\overline{B_i}=B_i \oplus 1$，$C_i$ 为借位输入，C_{i+1} 为借位输出。

（2）不恢复余数的阵列除法器

在不恢复余数的阵列除法器中，每一行所进行的操作是加法还是减法，取决于前一行输出的符号与被除数的符号是否一致。当出现不够减时，部分余数相对于被除数要改变符号，此时，产生一个商位“0”，除数首先沿对角线右移，然后，加到下一行的部分余数上。当够减时，部分余数不改变符号，此时，产生一个商位“1”，下一行的操作是减法。下面以 4 位不恢复余数的阵列除法器为例来说明其原理。

4 位不恢复余数的阵列除法器逻辑原理图如图 3-9 所示。其中

被除数 $x=0.x_1x_2x_3x_4x_5x_6$（双倍长度于除数）

除数 $y=0.y_1y_2y_3$

商 $q=0.q_1q_2q_3$

余数 $r=0.00r_3r_4r_5r_6$

阵列除法器由 4×4 个 CAS 单元组成，单元之间的互联是用 n=3 的阵列来表示的。被除数 x 是沿最上面一行和最右边对角线的垂直线进行输入，除数 y 是沿最上边一行的对角线进行输入；而商 q 是最左边进行输出，余数 r 是最下面一行的垂直线进行输出。最上面一行所进行的初始操作是减法，所以，控制信号 P 置“1”。减法是采用 2 的求补器来实现的，右

端各个 CAS 单元上的反馈线作为初始的进位输入，即最低位加上“1”。每一行最左边单元的进位输出决定商的取值，当前的商反馈到下一行，以确定下一行所进行的操作。由于进位输出信号表示当前部分余数的符号，因此，它将决定下一行所进行的操作是加法还是减法。

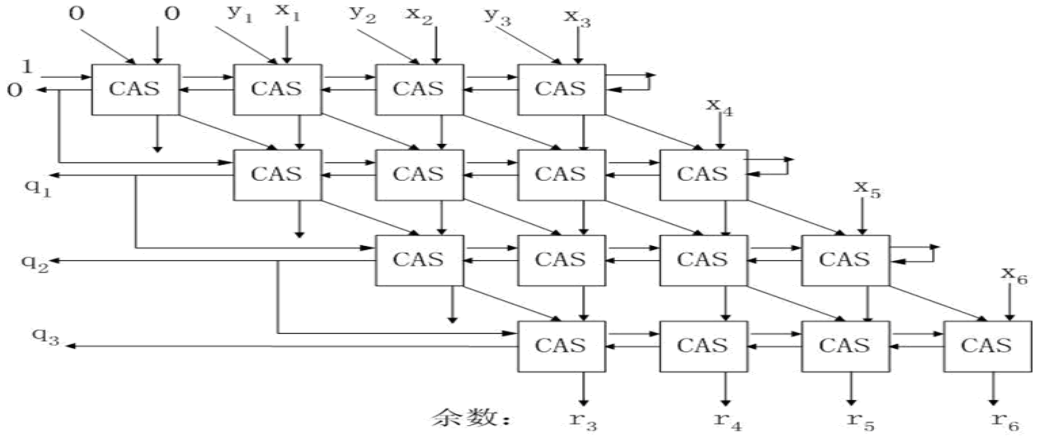


图 3-9　4 位不恢复余数的阵列除法器

【例 3-8】 x=0.101001，　y=0.111　求 x÷y

解：在计算机中，减法运算通过加补码实现，因此先求出：　[-y]补 =1.001

被除数	0.1 0 1 0 0 1		
减 y	1.0 0 1		
余数为负	1.1 1 0 0 0 1	< 0	q_0=0
余数左移	1.1 0 0 0 1		
加 y	0.1 1 1		
余数为正	0.0 1 1 0 1	> 0	q_1=1
余数左移	0.1 1 0 1		
减 y	1.0 0 1		
余数为负	1.1 1 1 1	< 0	q_2=0
余数左移	1.1 1 1		
加 y	1.0 0 1		
余数为正	0.1 1 0	> 0	q_3=1

所以：　商 q=0.101　　余数 r =0.000110

3.2　逻辑运算和移位操作

计算机除了能进行加、减、乘、除等基本算术运算以外，还可以对逻辑数进行逻辑运算。逻辑数是指无符号的二进制数。逻辑运算主要指逻辑与、逻辑或、逻辑异或等。

3.2.1 逻辑运算

设参与逻辑运算的两个数 x 和 y：

$x=x_0x_1x_2\cdots x_{n-1}$

$y=y_0y_1y_2\cdots y_{n-1}$

1、逻辑与（逻辑乘）：记作“∧”或“ • ”　　$z=x\wedge y=z_0z_1z_2\cdots z_{n-1}$

则　　$z_i=x_i\wedge y_i \quad i\in\{0，1，2，\cdots，n-1\}$

2、逻辑或（逻辑加）：记作“∨”或“ + ”　　$z=x\vee y=z_0z_1z_2\cdots z_{n-1}$

则　　$z_i=x_i\vee y_i \quad i\in\{0，1，2，\cdots，n-1\}$

3、逻辑异或（逻辑异）：记作“⊕”　　$z=x\oplus y=z_0z_1z_2\cdots z_{n-1}$

则　　$z_i=x_i\oplus y_i \quad i\in\{0，1，2，\cdots，n-1\}$

【例 3-9】 已知 x=01101011B，y=11101001B， 分别求 x∧y、x∨y、x⊕y。

解：

01101011	01101011	01101011
∧11101001	∨11101001	⊕11101001
01101001	**11101011**	**10000010**

3.2.2 移位操作

在定点数的运算中有左移和右移操作，在浮点数的运算中也用到移位操作，可见移位操作的重要性。由于机器字长的限制，当机器数进行左移或右移时，必然会使机器数产生空位和移出，移出的位到进位标志，而所产生的空位是填“0”、填“1”、还是填符号位或原来的进位标志，因此，移位操作分为逻辑移位、算术移位和循环移位，如图 3-10 所示。

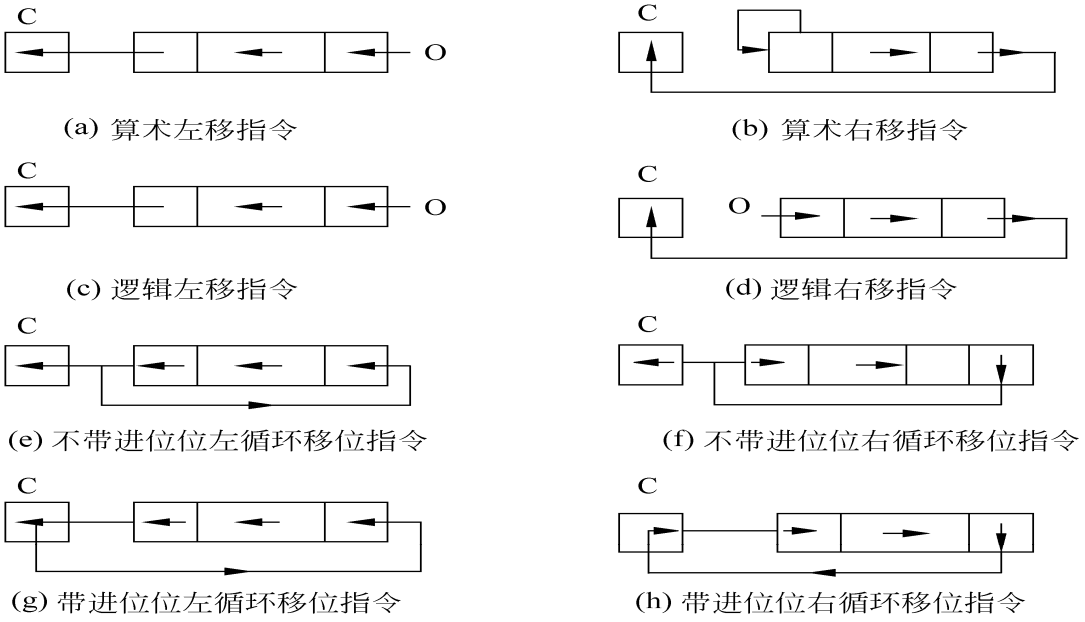


图 3-10　移位操作示意图

3.3　定点运算器

运算器是 CPU 的重要组成部分，是专门加工处理数据的部件，根据所能处理的数据不同，运算器可以分为定点运算器和浮点运算器。运算器的核心是算术逻辑单元（ALU），除此之外，还有各类寄存器（通用寄存器、累加器、数据缓冲寄存器、标志寄存器等）、数据多路选择器和数据总线等。

3.3.1 算术逻辑单元

1、一位算术逻辑单元

已知 X_i 和 Y_i 是一位二进制数，其算术运算的和为 F_i，则 $F_i=X_i+Y_i$。表面上看，F_i 只能是算术运算的结果。如果 X_i 或 Y_i 是 A_i、B_i 的逻辑函数，例如 $X=A_i \cdot B_i$ 或 A_i+B_i，当 $Y_i=0$ 时，则 $F_i=X_i+Y_i=X_i=A_i \cdot B_i$ 或 A_i+B_i。F_i 便是 A_i、B_i 逻辑运算的结果。因此，一位算术逻辑单元是在一位加法器的基础上再加一个逻辑函数发生器构成，如图 3-11 所示。

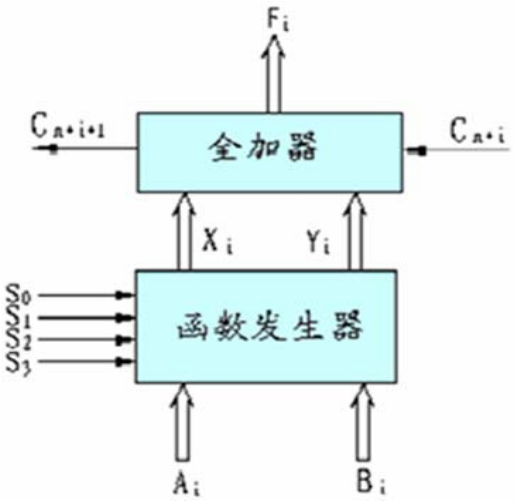


图 3-11　一位算术逻辑单元

因此，一位算术逻辑单元的逻辑表达式为：

$F_i=X_i \oplus Y_i \oplus C_{n+i}$

$C_{n+i+1}=X_iY_i+Y_iC_{n+i}+C_{n+i}X_i$

S_0、S_1、S_2、S_3 是用来控制逻辑函数发生器的输入 A_i 和 B_i，逻辑函数发生器的输出 X_i 和 Y_i 分别是受 S_2S_3 控制的 A_i 和 B_i 的组合函数和受 S_0S_1 控制的 A_i 和 B_i 的组合函数，其函数关系如表 3-5 所示。

表 3-5　　X_i、Y_i 与控制信号（S_0、S_1、S_2、S_3）和数据输入（A_i、B_i）的关系

S_0　S_1	Y_i	S_2　S_3	X_i
0　0	$\overline{A_i}$	0　0	1
0　1	$\overline{A_i}\,B_i$	0　1	$\overline{A_i}+\overline{B_i}$
1　0	$\overline{A_i}\,\overline{B_i}$	1　0	$\overline{A_i}+B_i$
1　1	0	1　1	A_i

由表 3-5，可写出 X_i 和 Y_i 的逻辑表达式

$X_i=S_2S_3A_i+S_2\overline{S_3}\ (\overline{A_i}+B_i)+\overline{S_2}\,S_3\,(\overline{A_i}+\overline{B_i})+\overline{S_2}\ \overline{S_3}$

$=\ \overline{S_3A_iB_i+S_2A_i\overline{B_i}}$

$Yi=\overline{S_0}\ \overline{S_1}\ \overline{A_i}+\overline{S_0}\,S1\,\overline{A_i}\,Bi+S0\,\overline{S_1}\ \overline{A_i}\ \overline{B_i}$

$=\overline{A_i+S_0B_i+S_1\overline{B_i}}$

结合一位全加器 FA 输出与输入之间的逻辑表达式，可得一位算术逻辑单元的一组逻辑表达式：

$$X_i=\overline{S_3A_iB_i+S_2A_i\overline{B_i}}$$

$$Y_i=\overline{A_i+S_0B_i+S_1\overline{B_i}} \qquad (3\text{-}13)$$

$F_i=Y_i \oplus X_i \oplus C_{n+i}$

$C_{n+i+1}=Y_i+X_iC_{n+i}$

2、四位算术逻辑单元

四位算术逻辑单元是由 4 个一位算术逻辑单元组成，其内部的一位算术逻辑单元之间的进位关系可以是串行进位，也可以是并行进位。图 3-12 给出了串行进位的四位算术逻辑单元。

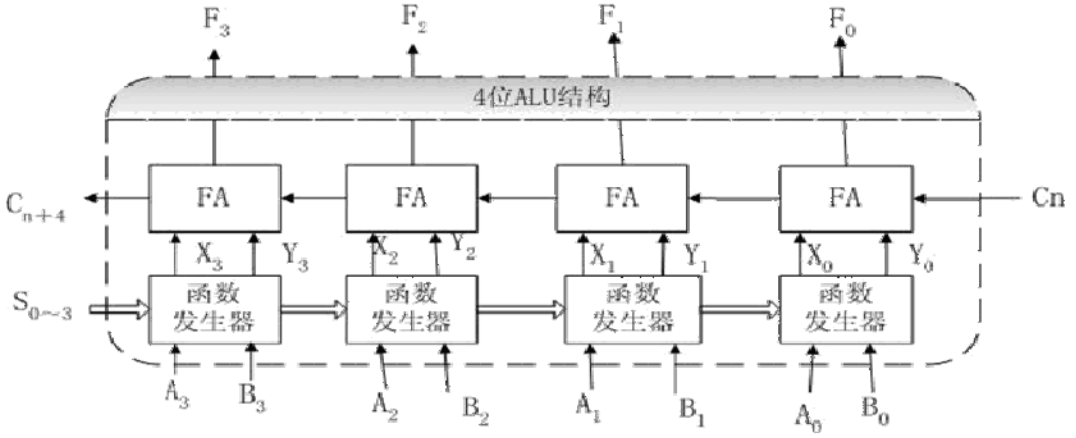


图 3-12　四位 ALU

先行进位的四位算术逻辑单元（74181ALU）除产生 4 位的数据输出（$F_0F_1F_2F_3$）和 1

位进位输出 C_{n+4} 外，又多产生 2 个输出，一个是 G，进位发生输出；另一个是 P，进位传递输出。根据式(3-13)，74181ALU 内部采用先行进位的进位关系如下：

第 0 位向第 1 位的进位公式为：$C_{n+1}=Y_0+X_0C_n$　　C_n 是向第 0 位的进位

第 1 位向第 2 位的进位公式为：$C_{n+2}=Y_1+X_1C_{n+1}= Y_1+Y_0X_1+X_1X_0C_n$

第 2 位向第 3 位的进位公式为：$C_{n+3}=Y_2+X_2C_{n+2}= Y_2+Y_1X_2+Y_0X_2X_1+X_2X_1X_0C_n$

第 3 位的进位输出公式为：$C_{n+4}=Y_3+X_3C_{n+3}= Y_3+Y_2X_3+Y_1X_3X_2+Y_0X_3X_2X_1+X_3X_2X_1X_0C_n$

令 $G= Y_3+Y_2X_3+Y_1X_3X_2+Y_0X_3X_2X_1$

$P= X_3X_2X_1X_0$

则　　$C_{n+4}=G+P\,C_n$　　（3-14）

综上所述，用 TTL 电路实现的四位算术逻辑单元（74181ALU）的逻辑电路如图 3-13 所示。M 为逻辑运算与算术运算的控制信号。当 M=1 时，封锁了各位的进位输出，进行的是逻辑运算；当 M=0 时，所进行的是算术运算。74181ALU 有两种工作方式：一种是采用正逻辑工作方式，另一种是采用负逻辑工作方式。无论哪一种工作方式都是等效的，都有 16 种逻辑运算和 16 种算术运算。表 3-6 列出了 74181ALU 正逻辑方式各种运算功能。74181ALU 正、负逻辑的引脚图如图 3-14 所示。

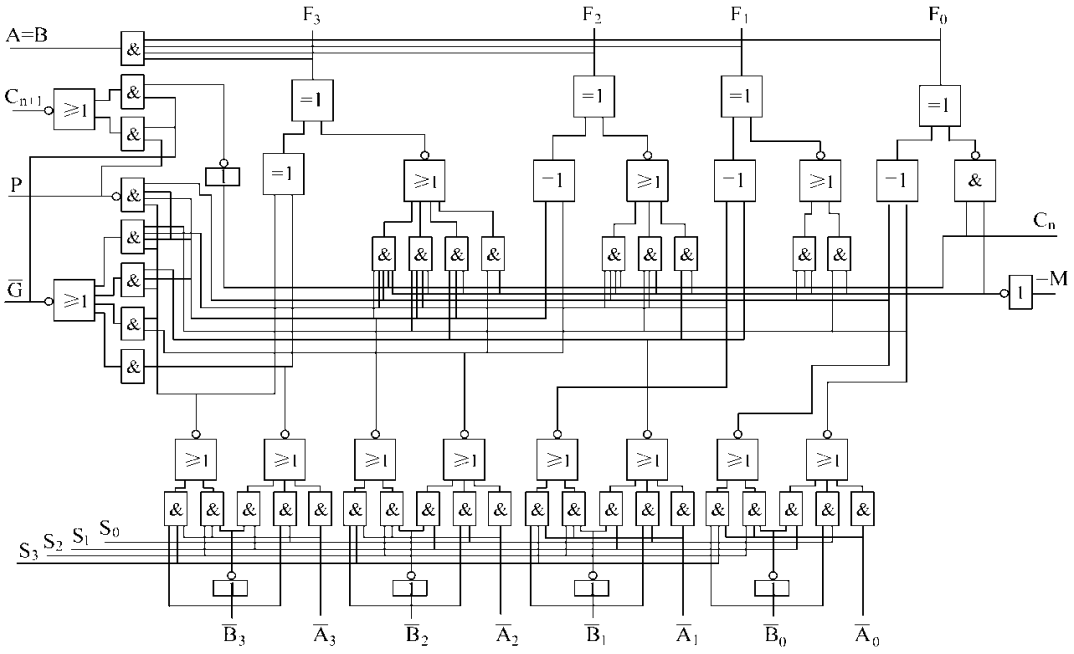


图 3-13　　74181ALU 逻辑电路

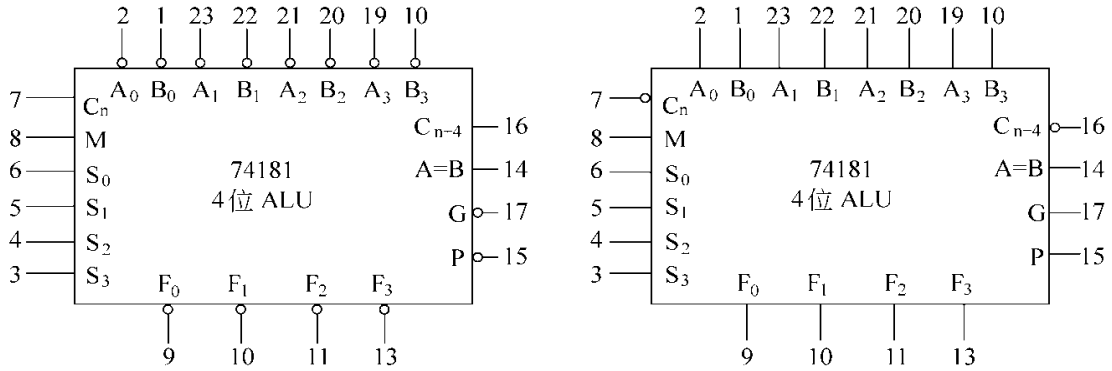


图 3-14　74181ALU 正、负逻辑的引脚图

表 3-6　　正逻辑 74181ALU 算术/逻辑运算功能表

S_3 S_2 S_1 S_0	逻辑运算（M=1）	算术运算（M=0，C_n=1）
0 0 0 0	$\overline{A}$	A
0 0 0 1	$\overline{A+B}$	A+B
0 0 1 0	$\overline{A}B$	$A+\overline{B}$
0 0 1 1	逻辑 0	减 1
0 1 0 0	$\overline{AB}$	$A+A\overline{B}$
0 1 0 1	$\overline{B}$	(A+B) 加 $A\overline{B}$
0 1 1 0	$A \oplus B$	A 减 B 减 1
0 1 1 1	$A\overline{B}$	$A\overline{B}$ 减 1
1 0 0 0	$\overline{A}+B$	A 加 AB
1 0 0 1	$\overline{A \oplus B}$	A 加 B
1 0 1 0	B	$(A+\overline{B})$加 AB
1 0 1 1	AB	AB 减 1
1 1 0 0	逻辑 1	A 加 2A
1 1 0 1	$A+\overline{B}$	(A+B)加 A
1 1 1 0	A+B	（$A+\overline{B}$）加 A
1 1 1 1	A	A 减 1

3、两级先行进位的 ALU

74181ALU 为了方便多个 74181ALU 的连接，特设置了 P 和 G 两个先行进位输出端。如果将 4 个 74181ALU 的先行进位输出端（P_0、G_0，P_1、G_1，P_2、G_2，P_3、G_3）送到先行进位部件 74182CLA，又可实现第二级的先行进位，即组与组之间的先行进位。

按照式（3-14）可得先行进位部件 74182CLA 的进位逻辑关系：

$C_{n+x}=G_0+P_0C_n$

$C_{n+y}=G_1+P_1C_{n+x}= G_1+G_0P_1+P_1P_0C_n$

$C_{n+z}=G_2+P_2C_{n+y}= G_2+G_1P_2+G_0P_2P_1+P_2P_1P_0C_n$

$C_{n+4}=G_3+P_3C_{n+z}= G_3+G_2P_3+G_1P_3P_2+G_0P_3P_2P_1+P_3P_2P_1P_0C_n$

令 $G^*= G_3+G_2P_3+G_1P_3P_2+G_0P_3P_2P_1$

$P^*= P_3P_2P_1P_0$

则　　$C_{n+4}=G^*+P^*C_n$

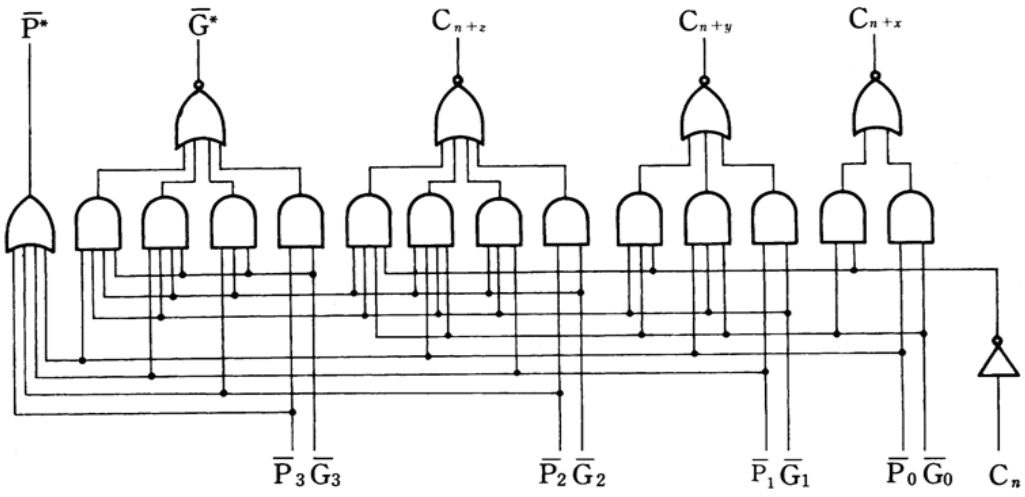


图 3-15　74182CLA 的逻辑电路

用 TTL 电路实现的先行进位部件 74182CLA 的逻辑电路如图 3-15 所示。其中，G*称为成组进位发生输出，P*称为成组进位传递输出。

由此可见，一个先行进位部件 74182CLA 可以连接 4 个 74181ALU，实现一个 16 位的组间、组内都是先行进位的 ALU，如图 3-16 所示。

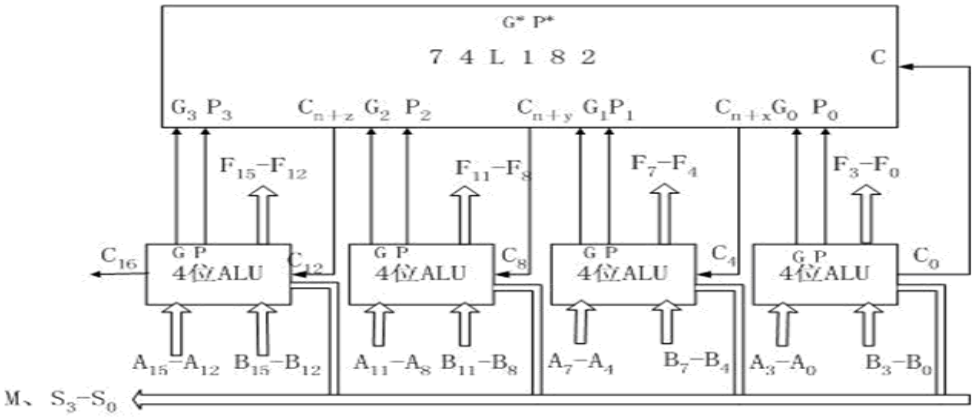


图 3-16　16 位全先行进位的 ALU

显然，用两个 16 位全先行进位的 ALU 进行级联，便可以组成了一个 32 位的 ALU。两个 16 位全先行进位 ALU 之间的进位采用的是串行进位。如图 3-17 所示。

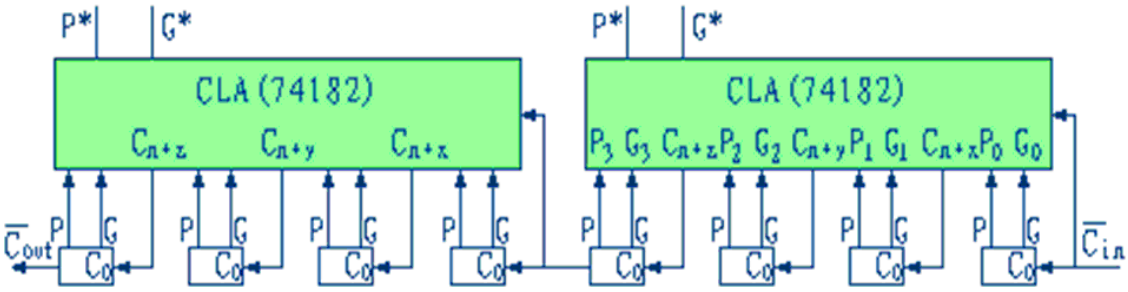


图 3-17　32 位 ALU

3.3.2 定点运算器的基本结构

1、单总线结构运算器

单总线结构运算器的特点是所有部件都连在一组总线上，在同一时间内，只能允许有一个数据放在总线上，参与运算的两个数据无法同时送到 ALU，因此，在 ALU 的两个输入端分别设置了 A、B 两个缓冲器。运算结束时，再通过单总线将结果存于目的寄存器。单总线结构运算器的主要缺点是操作速度慢。单总线结构运算器如图 3-18 所示。

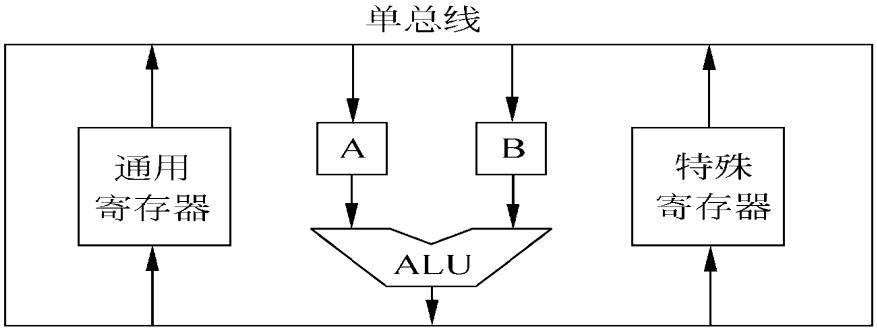


图 3-18　单总线结构运算器

2、双总线结构运算器

双总线结构运算器的特点是操作部件连接在两组总线上，可以通过这两组总线同时传输数据，从而保证了在同一时间内将参与运算的两个数据送到 ALU 并进行运算，但运算结果不能直接到总线，因为，此时的两组总线都被参与运算的两个数据占用着，所以，在 ALU 的输出端设置一个数据缓冲寄存器，运算结果通过缓冲寄存器再送到总线。显然，双总线结构运算器的执行速度比单总线结构运算器的执行速度快。双总线结构运算器如图 3-19 所示。

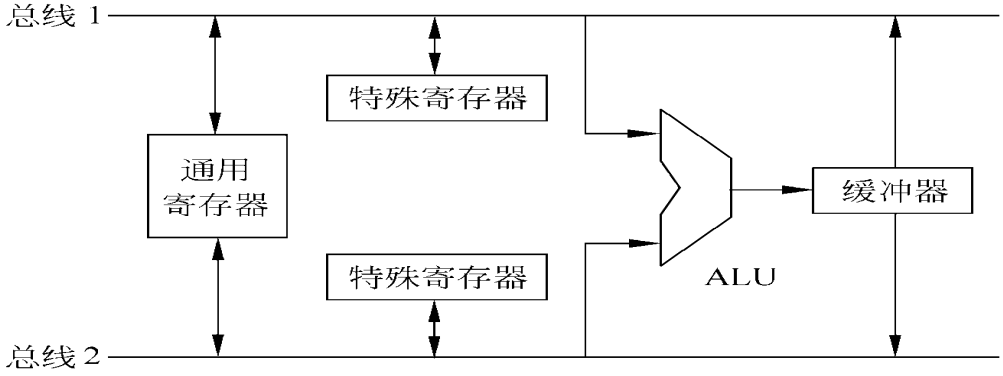


图 3-19　双总线结构运算器

3、三总线结构运算器

三总线结构运算器的特点是操作部件连接在三组总线上，可以通过这三组总线同时传输数据，不仅保证了在同一时间内将参与运算的两个数据送到 ALU 并进行运算，还可以将运算结果直接送到总线上。与前两种结构的运算器相比较，三总线结构运算器的执行速度最快，不过其控制也更复杂，在三总线结构运算器中，还设置一个总线旁路器，其目的是：当一组总线上的数据不需操作时，可以通过总线旁路器将此数据直接送到其它总线上，而不必经过 ALU。三总线结构运算器如图 3-20 所示。

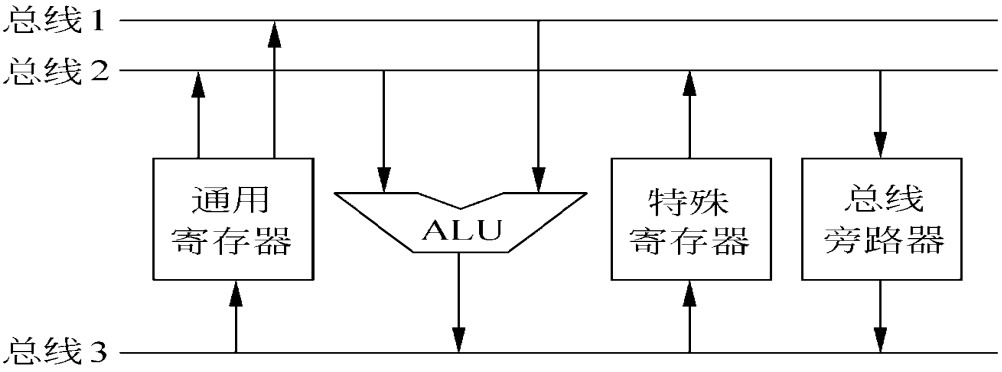


图 3-20　三总线结构运算器

3.3.3 定点运算器模型

运算器是由算术逻辑单元（ALU）、寄存器（通用寄存器、累加器、数据缓冲寄存器、标志寄存器等）、数据多路选择器和数据总线等组成。如图 3-21 所示，给出了单累加器运算器模型。该模型的核心是 ALU，实现对数据的算术运算和逻辑运算。AC 是累加器，实际上它也是寄存器，只不过使用得较频繁而已，不仅来存放参与运算的数据，还用来存放 ALU 所运算的结果。在此模型中所有运算结果只能存放于 AC 中。STR 为状态寄存器，其内设置有多个状态标志，例如零标志、进位标志、溢出标志、符号标志等，数据在 ALU 中运算的过程中会对 STR 中的标志产生影响。DR 为数据缓冲积存器，运算器与存储器之间的数据交换必需通过 DR，设置 DR 主要是解决速度匹配问题。

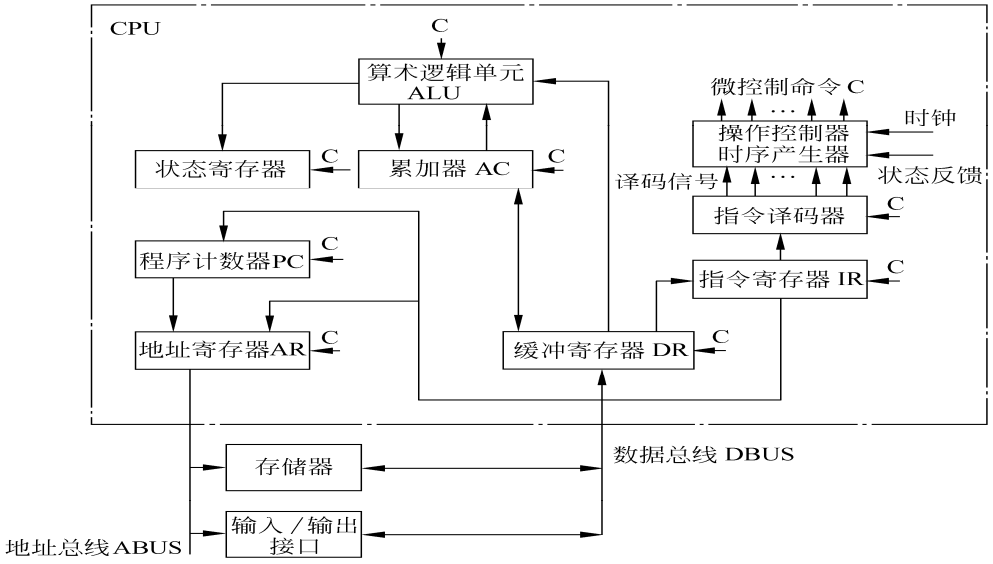


图 3-21　定点运算器模型

3.4　浮点数的算术运算与浮点运算器

3.4.1 浮点数的加减运算

设两个浮点数 x 和 y，它们分别为

$x=M_x\times 2^{ex}$　　x 表示为 E_x M_x

$x=M_y\times 2^{ey}$　　y 表示为 E_y M_y

其中，E_x、E_y 分别是浮点数 x 和 y 的阶码机器数形式，通常采用移码；M_x、M_y 为 x 和 y 的尾数机器数形式，通常采用补码。令 $z=x\pm y$，则

$$z=x\pm y=(M_x\times 2^{ex-ey}\pm M_y)\times 2^{ey} \qquad ex\leq ey$$

浮点数加减运算步骤如下：

（1）0 浮点数检查

如果加减运算中的两个浮点数 x 和 y 有一个为 0，则浮点数的加减运算就没有必要严格按照上述步骤进行，从而节约了运算时间，因此，0 浮点数检查是十分必要的。

（2）对阶

阶码表示的是小数点的位置，对阶实际上是使小数点对齐。当两个浮点数的阶码不相同时，将阶码小的浮点数的尾数右移△E（△E=｜ex-ey｜位，其阶码值增加△E，使两个浮点数的阶码值相同，此操作称位对阶。因为浮点数的尾数右移时丢失的是低位数值位，这样造成的浮点数误差较小。

（3）尾数加减

对阶完成后，即可进行尾数的加减。不论是加法还是减法，都必须按加法进行运算。

（4）结果规格化

尾数的加减运算结果未必满足浮点数的规格化条件，有必要进行左规或右规。下面以变补形式的尾数为例来讨论左规、右规和舍入处理。

当尾数加减运算结果的双符号位不同时，并不表示运算结果的溢出，因为影响浮点数大小的主要因素是阶码而不是尾数，仅表示尾数加减运算结果的绝对值大于 1，向左破坏了规格化，因此，需要将尾数加减运算结果右移并相应地提高阶码来实现规格化，这种实现规格化的方法称为右规；当尾数加减运算结果的双符号位相同，且符号位与最高数值位相同时，尾数的加减运算结果不符合浮点数的规格化条件，因此，需要将尾数加减运算结果左移并相应地降低阶码来实现规格化，这种实现规格化的方法称为左规；当尾数加减运算结果的双符

号位相同，且符号位与最高数值位不相同时，尾数的加减运算结果符合浮点数的规格化条件，因此，不需要左规或右规。

（5）舍入处理

在对阶和右规时，需要将浮点数的尾数和尾数加减运算结果进行右移，被右移的低位部分会因此而丢失，从而造成一定的误差，因此，需要进行舍入处理。最简单的舍入处理方法有：

1）“0 舍 1 入”法：如果右移时被丢掉的数值位的最高位为 0，则舍去；否则，在数值的最低位加 1。

2）“恒 1”法：只要有数值位被丢掉，则在数值的最低位置“1”。

（6）溢出判断

浮点数加减运算结果的溢出判断取决于阶码的溢出。如果阶码没有溢出，浮点数加减运算结果不会溢出；如果阶码溢出，则需要进行相应地处理。

阶码上溢：是指阶码超出了其所能表示的最大正数，一般认为结果是±∞。

阶码下溢：是指阶码超出了其所能表示的最小负数，一般认为结果是 0。

【例 3-10】 已知 $x=2^{010}\times(-0.110100)$　　$y=2^{100}\times(0.101011)$，求 x+y。

解：（1）对阶(阶码和尾数均采用补码表示)

$[x]_{浮}$=**00**.010；**11**.001100

$[y]_{浮}$=**00**.100；**00**.101011

浮点数 x 的阶码小，将 x 的尾数右移 2 位，x 的阶码增 2。即

$[x]_{浮}$=**00**.100；**11**.110011

（2）尾数求和

　　11.110011

+　**00**.101011

　　————————

　1**00**.011110

（3）规格化

∵尾数求和的结果为 **00**.011110，符号位与数值最高位相同。

∴需进行左规，即尾数向左移 1 位，阶码减 1，可得：**00**.011；**00**.111100

（4）舍入处理

采用“0 舍 1 入”法，由于左规时移出一位 0，所以，舍去 0。

（5）溢出判别

由于运算结果的阶码未发生溢出，所以运算结果未溢出。

$x+y=2^{011}\times(+0.111100)$

3.4.2 浮点数的乘法和除法运算

1、浮点数的乘法和除法运算规则

设两个浮点数 x 和 y，它们分别为

$x=M_x\times2^{ex}$　　x 表示为 E_x M_x

$y=M_y\times2^{ey}$　　y 表示为 E_y M_y

其中，E_x、E_y 分别是浮点数 x 和 y 的阶码机器数形式，通常采用移码，M_x、M_y 为 x 和 y 的尾数机器数形式，通常采用补码。则浮点数的乘法和除法运算规则：

浮点数的乘法运算规则：$x\times y=(M_x+M_y)\cdot2^{(ex+ey)}$

浮点数的除法运算规则：$x\div y=(M_x\div M_y)\cdot2^{(ex-ey)}$

2、浮点数的乘法和除法运算步骤

浮点数的乘法和除法运算大体分为六步：第一步，0 浮点数检查；第二步，阶码的加/

减运算；第三步，尾数乘/除运算；第四步，结果规格化；第五步，舍入处理；第六步，溢出判断。由于浮点数加减结果规格化、舍入处理也适用于浮点数的乘法和除法运算，尾数乘/除运算与定点数的乘/除类似，所以在此只讨论阶码的加/减运算及溢出判断和舍入处理。

（1）阶码运算

对阶码的运算有+1、-1、求和、求差四种，运算的同时还需要检查阶码是否溢出。由于阶码通常采用补码、移码形式，而补码的运算和溢出判别在前面已讲过，所以在此只讨论阶码的加/减运算及溢出判断。

1）移码运算

根据移码定义，对于 n+1 位的移码（1 位符号和 n 位数值），则：$[x]_{移}=2^n+x$，$-2^n\le x<2^n$，

$[x]_{移}+[y]_{移}=2^n+x+2^n+y$

$=2^n+(2^n+(x+y))$

$=2^n+[x+y]_{移}$

即直接用移码求阶码之和时，结果的最高位多了一位 1，要得到正确的移码形式结果，必须对结果的符号位取反。

当混合使用移码和补码时，由于$[y]_{补}=2^{n+1}+y \pmod{2^{n+1}}$，则：

$[x]_{移}+[y]_{补}=2^n+x+2^{n+1}+y$

$=2^{n+1}+(2^n+(x+y))$

$=2^{n+1}+[x+y]_{移}$

即 $[x+y]_{移}=[x]_{移}+[y]_{补} \pmod{2^{n+1}}$

同理

$[x-y]_{移}=[x]_{移}+[-y]_{补} \pmod{2^{n+1}}$

2）溢出判断

如果阶码运算的结果溢出，则上述各式均不成立。为了便于判断阶码运算的结果溢出，使用双符号位的阶码加法器，并规定移码的第二个符号位，即最高符号位恒用 0 参与运算，则溢出条件是结果的最高符号位为 1。此时，若两符号位为 10，表示结果上溢；若两符号位为 11，表示结果下溢；若两符号位为 00，表示结果为负；若两符号位为 01，表示结果为正。

【例 3-11】 已知 x=+011　y=+110，求$[x+y]_{移}$和$[x-y]_{移}$，并判断是否溢出。

解：$[x]_{移}=$**01**011，$[y]_{补}=$**00**110，$[-y]_{补}=$**11**010，

$[x+y]_{移}=[x]_{移}+[y]_{补}=$**01**011+**00**110=**10**001，双符号为 **10**，结果上溢。

$[x-y]_{移}=[x]_{移}+[-y]_{补}=$**01**011+**11**010=**00**101，双符号为 **00**，结果正确。

（2）舍入处理

舍入处理的目的是尽量减少误差，其方法有截断处理和修正处理。截断处理也称为恒舍法，是无条件地丢掉正常尾数最低位之后的全部数值。修正处理是保留右移过程中移出的若干高位的值，然后再按某种规则，用所保留的高位值来修正尾数。其舍入规则有：

1）只要尾数的最低位为 1 或移出丢掉的几位中有 1，把尾数的最低位置 1，否则保持原来的 0

2）最低位恒置 1

3）0 舍 1 入 当失去的最高位的值为 1 时，把该 1 加到最低数值位上进行修正，否则，舍去丢掉的各位的值。

【例 3-12】 已知$[x]_{补}=11.01100000$，$[x_2]_{补}=11.01100001$，$[x_3]_{补}=11.01111001$， 求执行只保留小数点后 4 位有效数字的舍入操作值。

解：执行舍入处理后，其结果分别为$[x]_{补}=11.0110$（不舍不入）

$[x_2]_{补}=11.0110$（舍）

$[x_3]_{补}=11.1000$（入）

【例 3-13】 已知 $x=2^{-5}\times0.0110011$　　$y=2^{3}\times(-0.1110010)$，阶码用 4 位移码表示，尾数（含数符）用 8 位补码表示，求 x×y。要求用补码完成尾数乘法运算，运算结果的尾数保留高 8 位，并用尾数低位字长的值处理舍入操作。

解：阶码采用双符号移码，尾数采用双符号补码，则

$[M_x]_{补}$=**00**.0110011，$[M_y]_{补}$=**11**.0001110

$[E_x]_{移}$=**00**011，$[E_y]_{移}$=**01**.011，$[E_y]_{补}$=**00**011

$[x]_{浮}$=**00**011，**00**.0110011，$[y]_{移}$=**01**.011，**11**.0001110

（1）求阶码和

$[E_x+E_y]_{移}=[E_x]_{移}+[E_y]_{补}$=**00**011+**00**011=**00**110，其真值为-2。

（2）尾数乘法运算可采用补码一位乘实现，即

$[M_x]_{补}\times[M_y]_{补}$=**00**.0110011×**11**.0001110=**11**.10100101001010

（3）规格化处理

乘积的尾数符号位与最高数值位相同，不符合规格化要求，需进行左规，即尾数左移一位，阶码减 1，此时，阶码为：**00**101（-3），尾数为：**11**.01001010010100。

（4）舍入处理

尾数为负数，取尾数高位字长，按舍入规则，舍去低位字长，故尾数为：**1**.0100101。

$[x\times y]_{浮}$=**00**101，**11**.0100101

x×y 的真值为：$2^{3}\times(-0.1011011)$

3.4.3 浮点运算器

由于浮点数的运算是对阶码和尾数分别进行的运算，阶码是定点整数，尾数是定点小数，所以，浮点运算器实质上是两个定点运算器构成的，其工作原理结构图如图 3-22 所示。

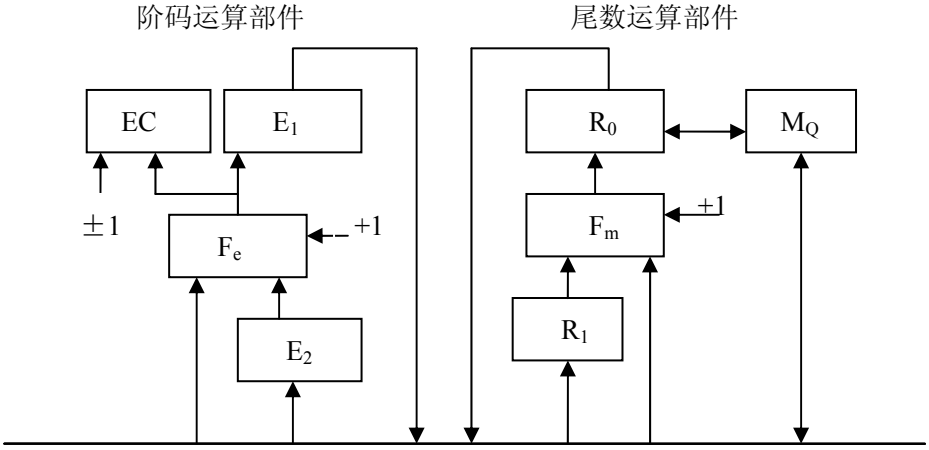


图 3-22　浮点运算器

阶码运算部件用来对阶码进行加减运算，由寄存器（E_1、E_2）、计数器 EC、并行加法器 F_e 组成。其中 E_1、E_2 用于存放与 R_0、R_1 中尾数相对应的阶码。

1、尾数运算部件

尾数运算部件用来对尾数进行加、减、乘、除运算，由寄存器（R_0、R_1、M_Q）、并行加法器 F_m 组成。其中 R_0、R_1 用于存放操作数，R_0 还用来存放运算结果；M_Q 是乘商寄存器，用来进行乘除运算；R_0、M_Q 具有联合左移、右移功能；R_1 具有右移功能，用于实现对阶的移位。

2、工作原理

（1）浮点数的加减运算

1）阶码运算部件求出阶差△E=E_1-E_2，并存放于 EC 中，EC 根据符号判断出哪个浮点数的阶码小，控制将其相应的尾数（R_0 或 R_1）进行右移。若△E 为正，则 E_2 小，控制 R_1 进行右移，且每右移一位，EC-1；若△E 为负，则 E_1 小，控制 R_0 进行右移，且每右移一位，EC-1。一直控制移位到 EC=0，完成浮点数的对阶操作；

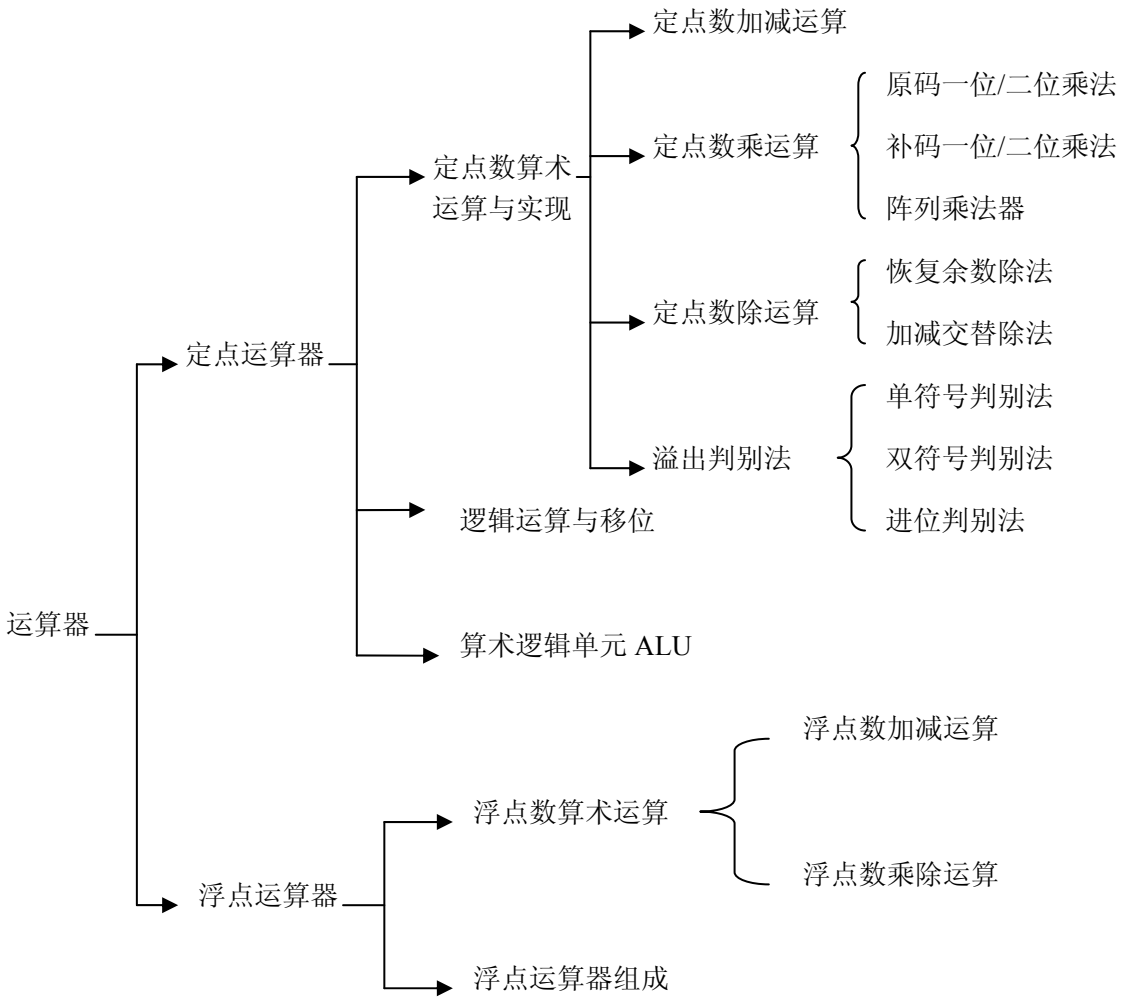
2）尾数运算部件完成对尾数的加减，并将结果存于 R_0 中；

3）判别运算结果，进行规格化。在规格化处理过程中，每将 R_0 左移（或右移）一位，应将 E_1、E_2 中的较大者减 1（加 1）。规格化处理结束后，将其作为结果的阶码。

（2）浮点数的乘除运算

在进行浮点数的乘除运算时，阶码运算部件和尾数运算部件独立工作，阶码运算部件只做阶码的加减运算，尾数运算部件完成尾数的乘除运算，运算结束后对结果进行规格化处理。

关　联



习　题

3.1 已知$[x]_{补}$、$[y]_{补}$、计算$[x+y]_{补}$和$[x-y]_{补}$。

（1）[x]补=0.11011，[y]补=0.00011 （2）[x]补=0.10111，[y]补=1.00101（3）[x]补=1.01010，[y]补=1.10001

3.2 已知$[x]_{补}$、$[y]_{补}$、计算$[x+y]_{变补}$和$[x-y]_{变补}$。

（1）$[x]_{补}$=100111，$[y]_{补}$=111100 （2）$[x]_{补}$=011011，$[y]_{补}$=110100（3）$[x]_{补}$=101111，$[y]_{补}$=011000

3.3 设某机字长为 8 位，给定十进制数：x=+49，y=-74，试按补码运算规则计算下列各题。

（1）$[x]_{补}+[y]_{补}$　　（2）$[x]_{补}-[y]_{补}$　　（3）$[-x]_{补}+[\frac{1}{2}y]_{补}$

（4）$[2x-\frac{1}{2}y]_{补}$　　（5）$[\frac{1}{2}x+\frac{1}{2}y]_{补}$　　（6）$[-x]_{补}+[2y]_{补}$

3.4 分别用原码一位乘法和补码一位乘法计算$[x\times y]_{原}$和$[x\times y]_{补}$。

（1）x=0.11001，y=0.10001　　（2）x=0.01101，y=-0.10100

（3）x=-0.10111，y=0.11011　　（4）x=-0.01011，y=-0.11010

3.5 用补码两位乘法计算$[x\times y]_{原}$和$[x\times y]_{补}$。

（1）x=0.11001，y=0.10001　　（2）x=0.10101，y=-0.01101

（3）x=-0.01111，y=0.11101　　（4）x=-0.01001，y=-0.10010

3.6 分别用原码不恢复余数法和补码不恢复余数法计算$[x/y]_{原}$和$[x/y]_{补}$。

（1）x=0.01011，y=0.10110　　（2）x=0.10011，y=-0.11101

（3）x=-0.10111，y=-0.11011　　（4）x=+10110，y=-00110

3.7 在进行浮点加减运算时，为什么要进行对阶？说明对阶的方法和理由。

3.8 已知某模型机的浮点数据表示格式如下：

15	14	13 … 8	7 … 0
数符	阶符	阶　码	尾　　数

其中，浮点数尾数和阶码的基值均为 2，均采用补码表示。

（1）求该机所能表示的规格化最小正数和非规格化最小负数及其所对应的十进制真值。

（2）已知两个浮点数的机器数表示为 EF80H 和 FFFFH，求它们所对应的十进制真值。

（3）已知浮点数的机器数表示为：

$[x]_{补}$=1111100100100101，$[y]_{补}$=1111011100110100

试按浮点加减运算，计算$[x\pm y]_{补}$。

3.9 已知某机浮点数表示格式如下：

11	10	9 … 7	6 … 0
数符	阶符	阶　码	尾　　数

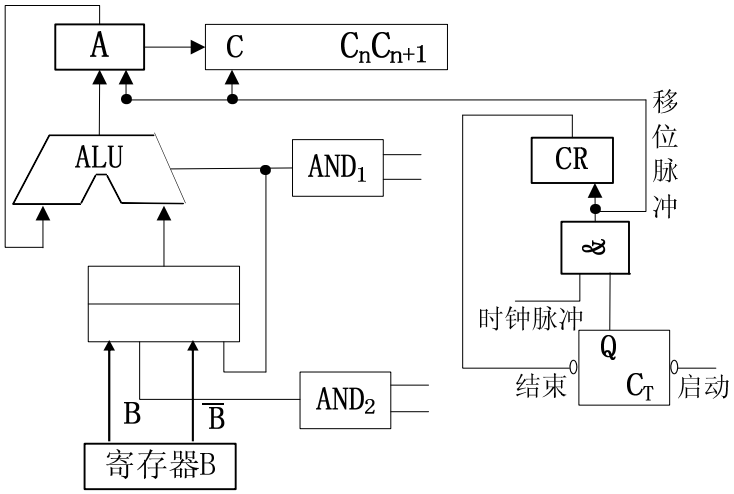
其中，浮点数尾数和阶码的基值均为 2，阶码用移码表示，尾数用补码表示。设：

$x=0.110101\times 2^{-001}$　　$y=-0.100101\times 2^{+001}$

用浮点运算规则，计算 x+y、x-y、x×y、x/y （要求写出详细运算步骤，并进行规格化）。

3.10 下图给出了实现补码乘法的部分硬件框图。

（1）请将图中逻辑门 AND_1 和 AND_2 的输入信号填写正确。



题图 3.10　补码乘法部分框图

（2）按补码乘法规则，将下列乘法运算算式完成，写出 x×y 的真值。

00.00000	1001100
→00.00000	0100110
00.11001	
00.11001	
→00.01100	1010011
→00.00110	0101001

（3）根据（2）中的乘法算式，将乘法运算初始和结束时，三个寄存器中的数据填入下列表格中。

寄存器	A	B	C
运算初始			
运算结束			

3.11 说明定点补码和浮点补码加减运算的溢出判断方法。

3.12 设有一个 16 位定点补码运算器，数据最低位的序号为 1。运算器可实现下述功能：

（1）A±B→A　（2）B×C→A、C（乘积高位在 A 中）　（3）A÷B→C（商在 C 中）

请设计并画出运算器第 4 位及 A、C 寄存器第 4 位输入逻辑。加法器本身逻辑可以不画。

3.13 设一个 8 位寄存器的内容为十六进制数 0C5H，连续经过一次算术右移、一次逻辑左移、一次大循环右移、一次小循环左移。写出每次移位后寄存器的内容和进位标志 C 的状态。

3.14 选择题

（1）运算器的核心部分是（　）。

A、数据总线　　B、累加寄存器

C、算术逻辑运算单元　　D、多路开关

（2）在浮点运算中下面的论述正确的是（　）。

A、对阶时应采用向左规格化

B、对阶时可以使小阶向大阶对齐，也可以使大阶向小阶对齐

C、尾数相加后可能会出现溢出，但可采用向右规格化的方法得出正确结论

D、尾数相加后不可能得出规格化的数

（3）当采用双符号位进行数据运算时，若运算结果的双符号位为 01，则表明运算（　）。

A、无溢出　B、正溢出　C、负溢出　D、不能判别是否溢出

（4）补码加法运算的规则是（　）。

A、操作数用补码表示，符号位单独处理

B、操作数用补码表示，连同符号位一起相加

C、操作数用补码表示，将加数变补，然后相加

D、操作数用补码表示，将被加数变补，然后相加

（5）原码乘除法运算要求（　）。

A、操作数必须都是正数　　B、操作数必须具有相同的符号位

C、对操作数符号没有限制　　D、以上都不对

（6）进行补码一位乘法时，被乘数和乘数均用补码表示，运算时（　）。

A、首先在乘数 y_n 后增设附加位 y_{n+1}，且初始 $y_{n+1}=0$，依照 y_ny_{n+1} 的值确定下面的运算

B、首先在乘数 y_n 后增设附加位 y_{n+1}，且初始 $y_{n+1}=1$，依照 y_ny_{n+1} 的值确定下面的运算

C、根据乘数符号位，决定乘数 y_n 后附加位 y_{n+1} 的值，依照 y_ny_{n+1} 的值确定下面的运算

D、不在乘数 y_n 后增设附加位 y_{n+1}，而直接根据乘数的末两位 $y_{n-1}y_n$ 确定下面的运算

（7）下面对浮点运算器的描述中正确的是（　）。

A、浮点运算器由阶码部件和尾数部件实现

B、阶码部件可实现加、减、乘、除四种运算

C、阶码部件只能进行阶码的移位操作

D、尾数部件只能进行乘法和加法运算

（8）若浮点数的阶码和尾数都用补码表示，则判断运算结果是否为规格化数的方法是（　）。

A、阶符与数符相同为规格化数

B、阶符与数符相异为规格化数

C、数符与尾数小数点后第一位数字相异为规格化数

D、数符与尾数小数点后第一位数字相同为规格化数

（9）已知$[x]_{补}$=1.01010，$[y]_{补}$=1.10001,下列答案正确的是（　）。

A.、$[x]_{补}+[y]_{补}$=1.11011　　B、$[x]_{补}+[y]_{补}$=0.11011

C、$[x]_{补}-[y]_{补}$=0.11011　　D、$[x]_{补}-[y]_{补}$=1.11001

（10）下列叙述中概念正确的是（　）。

A、定点补码运算时，其符号位不参加运算

B、浮点运算中，尾数部分只进行乘法和除法运算

C、浮点数的正负由阶码的正负符号决定

D、在定点小数一位除法中，为了避免溢出，被除数的绝对值一定小于除数的绝对值

3.15 填空题

（1）在补码加减运算中，符号位与数据_____参加运算，符号位产生的进位_____。

（2）在采用变形补码进行加减运算时，若运算结果中两个符号位_____，表示发生了溢出。若结果的两个符号位为_____，表示发生正溢出；为_____，表示发生负溢出。

（3）浮点乘除法运算的运算步骤包括：_____、_____、_____、_____和_____。

（4）在浮点运算过程中，如果运算结果的尾数部分不是_____形式，则需要进行规格化处理。设尾数采用补码表示形式，当运算结果_____时，需要进行右规操作；当运算结果_____时，需要进行左规操作。

（5）浮点运算器由_____和_____两部分组成，它们本身都是定点运算器。

3.16 是非题

（1）运算器的主要功能是进行加法运算。（　）

（2）加法器是构成运算器的主要部件，为了提高运算速度，运算器中通常都采用并行加法器。（　）

（3）在定点整数除法中，为了避免运算结果的溢出，要求｜被除数｜＜｜除数｜。（　）

（4）浮点运算器中的阶码部件可实现加、减、乘、除运算。（　）

（5）根据数据的传递过程和运算控制过程来看，阵列乘法器实现的是全并行运算。（　）

（6）逻辑右移执行的操作是进位标志位移入符号位，其余数据位依次右移 1 位，最低位移入进位标志位。（　）

第 4 章　存储器系统

【内容摘要】

存储器是计算机的重要组成部分之一，是用来存放程序和数据的。为了解决存储器的容量大、速度快、价格低三方面的矛盾，计算机往往采用多级存储体系结构（即 Cache、主存、辅存结构），从而使各种存储器构成一个有机整体，称之为存储器系统。其中主存是一种半导体存储器，包括只读存储器、随机存储器、闪速存储器和并行存储器等，它是本章的一个重点；Cache 是一种高速缓冲存储器，是为了解决 CPU 与主存之间的速度匹配问题而采用的一种硬件技术，并发展为多级 Cache 体系，分为指令 Cache 和数据 Cache；辅存是一种虚拟存储器，可以分为段式、页式、段页式虚拟存储器。

【学习要点】

- 存储器的分类和技术指标
- 半导体存储器工作原理与扩展方法
- 闪速存储器、双端存储器和多体交叉存储器
- Cache 地址映射方法
- 虚拟存储器基本概念及其地址变换过程

4.1　存储器概述

4.1.1 存储器分类

存储器随着计算机的发展进步较快，种类繁多，其分类方法也有多种。

1、按存储器在系统中的作用进行分类

（1）主存储器

主存储器是用来存放当前运行的程序和数据的，可以被 CPU 直接访问的半导体存储器。它位于主机内部，又称内存储器，简称内存或主存。

（2）辅助存储器

辅助存储器是为解决主存容量不足而设置的存储器，是用来存放 CPU 暂不执行的程序和数据，它可以是硬盘、U 盘、光盘等，其特点是存储容量大。辅助存储器又称外存储器，简称外存或辅存。辅存是不能被 CPU 直接访问的，当需要运行存放在辅存中的程序时，必须将辅存中的程序调入内存，然后再由 CPU 去执行。近年来，大容量半导体存储器如 FLASH 存储器的价格迅速下降，用闪存制成的“优盘”成为了一种很受欢迎的外存。

（3）高速缓冲存储器

高速缓冲存储器是介于 CPU 与主存之间，用来解决 CPU 与主存之间的速度匹配问题而设置的高速小容量的存储器，简称 Cache。它可以做在 CPU 内部，称内部 Cache，也可以位于 CPU 之外，称为外部 Cache。

2、按存取方式进行分类

（1）随机存储器

随机存储器简称为 RAM（Random　Access　Memory），是指存储单元既能被 CPU 读，又能被 CPU 写，CPU 对存储单元的读、写都是随机的，且读、写时间与存储单元的物理位置无关。一般主存主要是由 RAM 组成。

（2）只读存储器

只读存储器简称为 ROM（Read　Only　Memory），是指存储单元只能被 CPU 随机地进行读，而不能进行写。只读存储器可以作为主存的一部分，用来存放不变的程序和数据，例如计算机的加电诊断程序、系统引导程序等。

（3）顺序存储器

顺序存储器简称 SAM（Sequential Access Memory），是指存储器的内部信息排列有序的，CPU 对存储器的读或写是按顺序进行的，并且 CPU 对存储器的读或写时间与信息在存储器中的物理位置有关。

2、按存储介质进行分类

（1）半导体存储器

半导体存储器是一种利用半导体器件来进行存储二进制信息的存储器。计算机的内存一般是由半导体存储器组成，根据半导体存储器的制造工艺不同，可以将半导体存储器分为双极型和 MOS 型。

（2）磁表面存储器

磁表面存储器是利用涂在基体表面上的一层磁性材料来存储二进制信息的存储器，例如磁盘等。

（3）光存储器

光存储器是采用光学原理制成的存储器，它是通过能量高度集中的激光束照在基体表面而引起的物理或化学的变化来记忆二进制信息。

4.1.2 存储器系统结构

不管计算机的主存容量有多大，总是无法满足人们的期望。为了解决存储器的容量大、速度快、价格低三方面的矛盾，计算机往往采用多级存储体系结构，如图 4-1 所示。最上层（即第一层）是 CPU 的内部寄存器，数量是有限的，其访问时间是几个 *ns*；第二层是 Cache，存储容量在 32*KB*~几十 *MB*，其访问时间是十几个 *ns*；第三层是主存，存储容量几十 *MB*~几个 *GB*，其访问时间是几十个 *ns*；第四层是辅存，存储容量在几个 *GB* ~几十 *GB*，其访问时间是 10*ms* 以上，如果是光盘，其驱动时间加加上访问时间就需要用秒来衡量了。

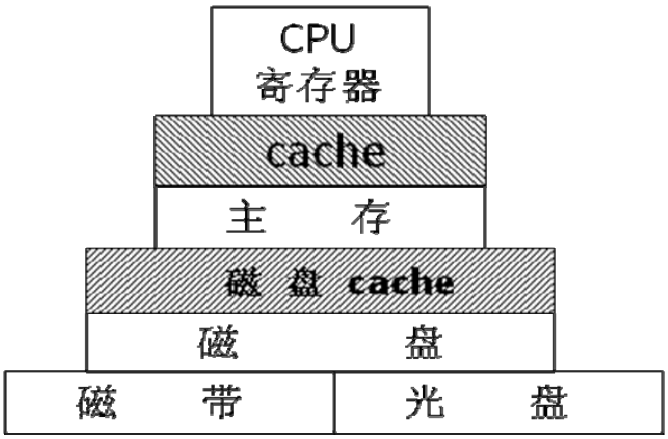


图 4-1 多级存储体系结构

4.1.3 主存储器的技术指标

主存储器一般是半导体存储器，主存储器的技术指标也是指半导体存储器的技术指标。半导体存储器的组成结构如图 4-2 所示。存储体是存储二进制信息的主体，是由很多存储单元组成，为了区别不同的存储单元，就需要对存储体中每一个存储单元进行统一编号，这个编号我们称之为存储单元的地址。于是，存储单元与其地址之间建立了一一对应的关系，一旦给出一个单元的地址就能唯一确定一个存储单元。存储单元所存储的二进制信息称之为存储单元的内容，由此可见，存储单元的地址与存储单元的内容是两个不同的概念，但两者又存在一定的关系，即存储单元的内容可以用存储单元的地址来表示。这种通过存储单元的地址来访问存储单元的方法称为编址方法，可以分为按字节编址和按字编址。按字节编址是指与存储单元的地址相对应的存储单元内容是一个字节，即最小寻址单位是一个字节，目前大多数计算机采用的是按字节编址。按字编址是指与存储单元地址相对应的存储单元内容是一个字。

1、存储容量

存储容量是存储器主要性能指标，存储容量越大，所能存储的信息量就越多。存储容量的大小常用 *B*、*KB*、*MB*、*GB* 和 *TB* 为单位表示。其中，$1KB=2^{10}B=1024B$；$1MB=1024KB$

$=2^{20}B$；$1GB=1024MB=2^{30}B$；$1TB=1024GB=2^{40}B$。

2、存取时间

存取时间是指从启动一次存储器操作到完成该操作所经历的时间。例如，读出时间是指从 CPU 向存储器发出有效地址和读命令开始，直到将被选单元的内容读出为止所用的时间。显然，存取时间越小，存取速度越快。

3、存储周期

连续启动两次独立的存储器操作(如连续两次读操作)所需要的最短间隔时间称为存储周期。它是衡量主存储器工作速度的重要指标。一般情况下，存储周期略大于存取时间。

4、存储器带宽

存储器带宽是指在单位时间内从存储器中所存取的信息量，是衡量数据传输速率的重要技术指标，通常以位/秒或字节/秒做度量单位。

5、可靠性

可靠性一般指存储器对外界电磁场及温度等变化的抗干扰能力。存储器的可靠性用平均故障间隔时间 MTBF(Mean Time Between Failures)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，可靠性越高，存储器正常工作能力越强。

6、性能/价格比

性能/价格比(简称性价比)是衡量存储器经济性能好坏的综合指标，它关系到存储器的实用价值。性能包括前述的各项指标，其中存取时间、存储周期、存储器带宽都反映了主存速度的指标，而价格是指存储单元本身和外围电路的总价格。

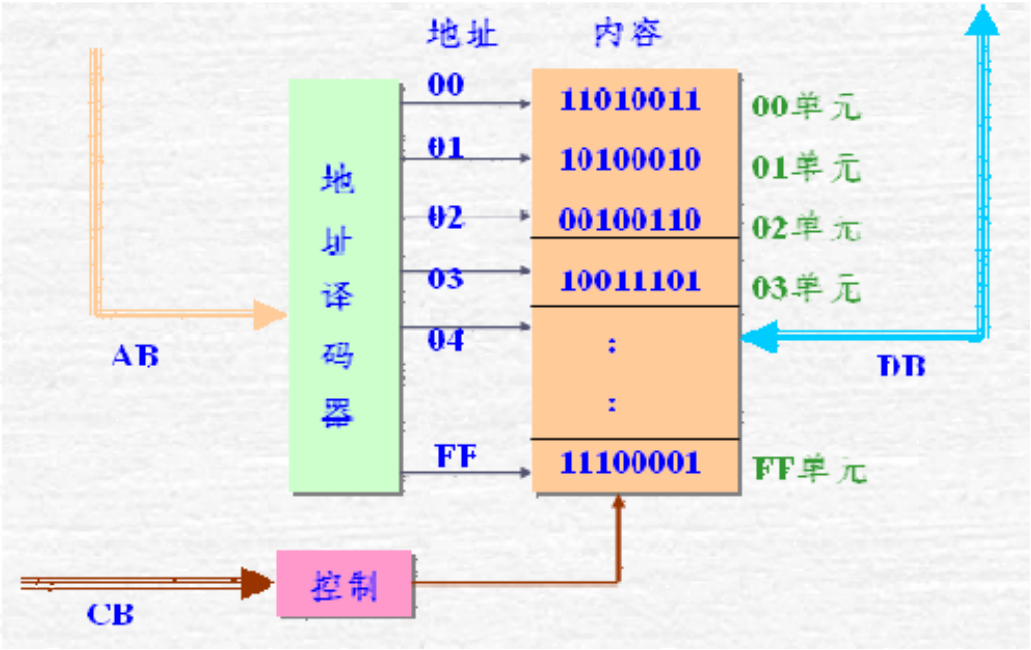


图 4-2　存储器组成结构

4.2　半导体存储器

4.2.1 半导体存储器分类

半导体存储器是目前被广泛应用于主存的一种存储器，按其读写性能可分为：随机读写存储器（RAM）和只读存储器（ROM）两大类，如图 4-3 所示。

1、RAM

RAM 是可读、可写的存储器，又称为读写存储器，其特点是：系统断电后会自动丢失其中存储的信息。根据制造工艺，RAM 可分为双极型和 MOS 型两种，其中 MOS 型 RAM

按信息存放方式不同，可分为静态 RAM(Static RAM，简称 SRAM)和动态 RAM(Dynamic RAM，简称 DRAM)。

2、ROM

只读存储器 ROM 是非易失性存储器，其特点是：系统断电后其中所存储的信息不会丢失。只读存储器种类繁多，在此只能对几种 ROM 进行简单地介绍。

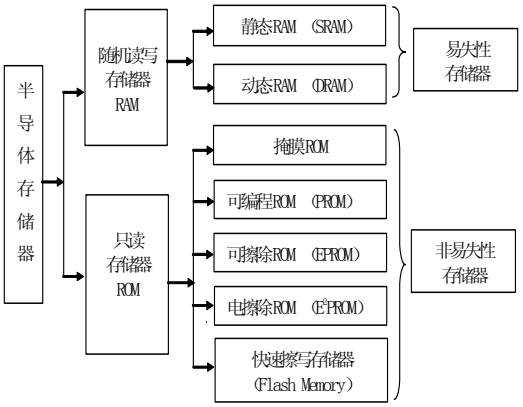


图 4-3 半导体存储器分类

（1）PROM

PROM（Programmable ROM）称为可编程 ROM，是由用户把要写入的信息“烧”入 PROM 中，对 PROM 的“烧”入操作，需要一个 ROM 编程器的特殊设备。

（2）EPROM

EPROM(Erasable PROM)称为紫外线擦除 PROM，用紫外光照射 EPROM，可实现对 EPROM 中信息的擦除，所有 EPROM 芯片都有一个窗口用于接收照射它的紫外线。

（3）EEPROM

EEPROM（Electrically EPROM）称为电擦除 PROM，它与 EPROM 相比，存在很多优势：其一，它是采用电擦除，可以实现瞬间擦除，而 EPROM 需要 20 分钟左右的擦除时间；其二，用户可以对 EEPROM 进行有选择地擦除，而 EPROM 是对整个芯片所有内容进行擦除；其三，用户可以直接在电路板上对 EEPROM 进行擦除和编程，而不需要额外的设备。

（4）FE^2PROM

FE^2PROM（Flash EEPROM）称为闪烁可编程可擦除 ROM，简称闪存，也是半导体存储器，它既吸收了 EPROM 结构简单、编程可靠的优点，又保留了 E^2PROM 用隧道效应擦除快捷的特性，而且集成度可以做得很高。

4.2.2 存储元电路

存储元电路是指存储一位二进制信息“1”或“0”的电路，又称存储细胞或基本单元电路。结合半导体存储器的分类，下面分别介绍几种存储元电路。

1、六管静态存储元

六管静态存储元是由六只 NMOS 管（T_1～T_6）组成，如图 4-4 所示。其中 T_1 与 T_2 构成一个反相器，T_3 与 T_4 构成另一个反相器，两个反相器的输入与输出交叉连接构成双稳态触发器，利用稳态来存储一位二进制信息“1”或“0”。当 T_1 导通、T_3 截止时为 0 状态；当 T_3 导通、T_1 截止时为 1 状态。在不掉电的情况下，存储元所存储的一位二进制信息是不会改变的。T_5、T_6 是门控管（行选通管），由 X_i 线控制其导通或截止。当 X_i=1 时，T_5、T_6 导通；当 X_i=0 时，T_5、T_6 截止。T_7、T_8 也是门控管（列选通管），其导通与截止受 Y_i 线控制。当 Y_j=1 时，T_7、T_8 导通；当 Y_j=0 时，T_7、T_8 截止。T_7、T_8 是用来控制位线与数据线之间连接状态，并不是每个存储元都需要这两只管子，所以称为六管 NMOS 静态存储元。

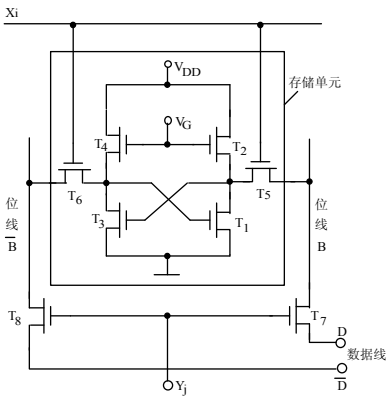


图 4-4 六管静态存储元

只有当存储元所在的行、列对应的 X_i、Y_j 线均为 1 时，该存储元才与数据线接通，才能对它进行读或写，这种状态称为选中状态。

2、单管动态存储元

单管动态存储元只有一个电容和一个 MOS 管组成，如图 4-5 所示。一位二进制信息存储依靠的是 MOS 管栅极与源极之间的极间电容，若极间电容有电荷，表示所储存的信息为“1”；否则，表示所储存的信息为“0”。在保持状态下，行选择信号线为低电平，V 管截止，电容 C 不存在充放电回路(当然还有一定的泄漏)，其上的电荷状态将保持不变（有电荷表示存“1”，无电荷表示存“0”)。

（1）读出

在对存储元进行读操作时，行选择线为高电平，使 V 管导通，于是刷新放大器读取对应电容 C 上的电压值，只有当列选择信号有效时，存储元才可以输出信息。刷新放大器的灵敏度很高，放大倍数很大，并且能将读取电容上的电压值转换为逻辑“0”或者逻辑“1”。因此在读出的过程中，存储元中的电容将会受到影响，为了在读出信息之后存储元仍能保持原有的信息，刷新放大器在读取电容上的电压值之后又立即进行重写，使每次读出后电容 C 上的电荷保持不变，这就是所谓的“再生”或“刷新”。

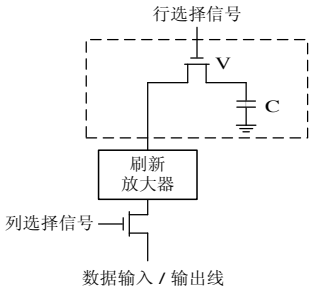


图4-5　单管动态存储元

（2）写入

在对存储元进行写操作时，行选择线为高电平，使 V 管导通，如果列选择信号也为高电平，则存储元被选中，于是由数据输入/输出线送来的信息通过刷新放大器和 T 管送到电容 C。

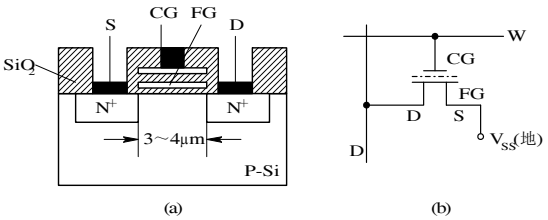
（3）刷新

由于晶体管 V 存在漏电流，平时电容 C 上的电荷将逐渐泄漏掉，不能长期保存，将使存入的信息消失。为此，需要周期性地对电容进行充电，以补充泄漏的电荷，通常把这种补充电荷的过程叫刷新或再生。随着器件工作温度的增高，放电速度会变快。刷新时间间隔一般要求在 1～100 *mS*。工作温度为 70℃时，典型的刷新时间间隔为 2*mS*，因此，2*mS* 内必须对存储的信息刷新一遍。

3、只读存储器存储元

（1）EPROM 存储元

初期的 EPROM 存储元是浮栅雪崩注入 MOS，记为 FAMOS。它的集成度低，用户使用不方便，速度慢，因此很快被性能和结构更好的叠栅注入 MOS（SIMOS）取代。



(a)　SIMOS 管结构；　(b) SIMOS EPROM 存储元

图 4-6　SIMOS 型 EPROM

SIMOS 管结构如图 4-6(a)所示。它属于 NMOS，与普通 NMOS 不同的是：它有两个栅极，一个是控制栅 CG，另一个是浮栅 FG。FG 在 CG 的下面，被 S_iO_2 所包围，与四周绝缘。单个 SIMOS 管构成一个 EPROM 存储元，如图 4-6(b)所示。

与 CG 连接的线 W 称为字线，读出和编程时作选址用。漏极与位线 D 相连接，读出或编程时输出、输入信息。源极接 V_{SS}(接地)。当 FG 上没有电子驻留时，CG 开启电压为正常值 Vcc，若 W 线上加高电平，源、漏间也加高电平，SIMOS 形成沟道并导通，称此状态为“1”；当 FG 上有电子驻留，CG 开启电压升高超过 Vcc，这时若 W 线加高电平，源、漏间仍加高电平，SIMOS 不导通，称此状态为“0”。人们就是利用 SIMOS 管 FG 上有无电子驻留来存储二进制信息“1”或“0”。因 FG 上电子被绝缘材料包围，不获得足够能量

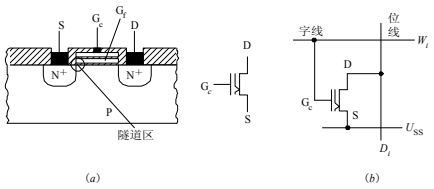
很难跑掉，所以可以长期保存信息，即使断电也不丢失。

SIMOS EPROM 芯片出厂时 FG 上是没有电子的，即都是“1”信息。对它编程就是在 CG 和漏极都加高电压，向某些元件的 FG 注入一定数量的电子，把它们写为“0”。EPROM 封装方法与一般集成电路不同，需要有一个能通过紫外线的石英窗口，擦除时将芯片放入擦除器的小盒中，用紫外灯照射约 20 分钟，若读出各单元内容均为 FFH，说明原信息已被全部擦除，恢复到出厂状态。写好信息的 EPROM 为了防止光线长期照射而引起的信息破坏，常用遮光胶纸贴于石英窗口上。EPROM 的擦除是对整个芯片进行的，不能只擦除个别单元或个别位，擦除时间较长，且擦和写均需离线操作，使用起来不方便，因此，能够在线擦写的 E^2PROM 芯片近年来得到广泛应用。

（2）FE^{2}PROM 存储元

闪存是新一代电信号擦除的可编程 ROM，它既吸收了 EPROM 结构简单、编程可靠的优点，又保留了 E^2PROM 用隧道效应擦除快捷的特性，而且集成度可以做得很高。

图 4-7(a)是闪存采用的叠栅 MOS 管示意图。其结构与 EPROM 中的 SIMOS 管相似，两者区别在于浮栅与衬底间氧化层的厚度不同，在 EPROM 中氧化层的厚度一般为 30～40*nm*，在闪存中仅为 10～14*nm*, 而且浮栅和源区重叠的部分是源区的横向扩散形成的，面积极小, 因而浮栅与源区之间的电容很小，当 Gc 和 S 之间加电压时，大部分电压将降在浮栅与源区之间的电容上，闪存的存储元就是用这样一只单管组成的，如图 4-7(b)所示。



(*a*) 叠栅 MOS 管； (*b*) 存储单元

图 4-7 快闪存储器

闪存存储元的写入方法和 EPROM 相同，即利用雪崩注入的方法使浮栅充电。在读出状态下，字线加上+5*V*，若浮栅上没有电荷，则叠栅 MOS 管导通，位线输出低电平；如果浮栅上充有电荷，则叠栅管截止，位线输出高电平。擦除方法是利用隧道效应进行的，类似于 E^2PROM 写 0 操作。在擦除状态下，控制栅处等于 0 电平，同时在源极加入幅度为 12*V* 左右、宽度为 100*mS* 的正脉冲，在浮栅和源区间极小的重叠部分产生隧道效应，使浮栅上的电荷经隧道释放，但由于片内所有叠栅 MOS 管的源极连在一起，所以擦除时是将全部存储单元同时擦除，这是不同于 E^2PROM 的一个特点。

4.2.3 存储器芯片

1、半导体存储器芯片组成与结构

每一个存储器芯片都具有一定的存储容量，通常表示为 N×m（bit），其中 N 是存储器芯片的字数，一般为 2 的 n 次幂($N=2^n$)，m 是存储器芯片的位数，一般为 1、4、8 等。由此可见，一个存储器芯片内部包含 N×m 个存储元电路，将 N×m 个存储元电路排列成矩阵，即构成存储矩阵，也称之为存储体。存储体是存储器芯片的核心，它与外围电路（地址译码电路、读/写控制电路、输入/输出控制电路等）集成在一块硅片上，称为存储器组件。

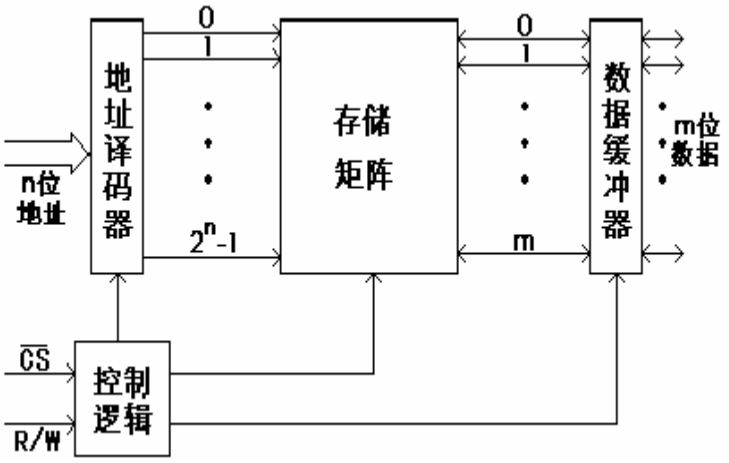


图 4-8 半导体存储器芯片基本结构

存储器组件经过各种形式的封装，引出地址线、数据线、控制线和电源与地线等，即制成了半导体存储器芯片。半导体存储器芯片基本结构如图 4-8 所示。半导体存储器芯片根据位数 m 的不同，可以将存储器芯片分为字片式结构（m≠1）和位片式结构（m=1）。

（1）字片式结构半导体存储器芯片

图 4-9 所示是 16×8 位的字片式结构半导体存储器芯片结构图。图中每一个小方块表示一个存储元电路，存储矩阵的每一行由 8 个存储元电路组成一个存储单元，存放一个 8 位的存储字。一行中所有存储元电路的字线连在一起，与地址译码器的某一个输出端相连；所有存储单元相同的位组成一列，一列中所有存储元电路的两条位线分别连在一起，并使用同一个读写放大电路，读写放大电路与双向数据线相连。若存储器芯片接到地址信息为 $A_3A_2A_1A_0$=1111 时，$A_3A_2A_1A_0$ 经地址译码器译码后字线 15 有效，即选中 15 号字线相应的存储单元，从而实现对该单元中所有的存储元电路同时进行读/写。这种对地址仅进行一个方向上的译码方式称为单译码方式或一维译码方式。由于字片式结构半导体存储器芯片采用单译码方式，芯片内有多少个存储单元就需要多少个译码驱动电路，所需译码驱动电路较多，电路复杂，为此，大多数存储器芯片都采用双译码方式，既位片式结构。

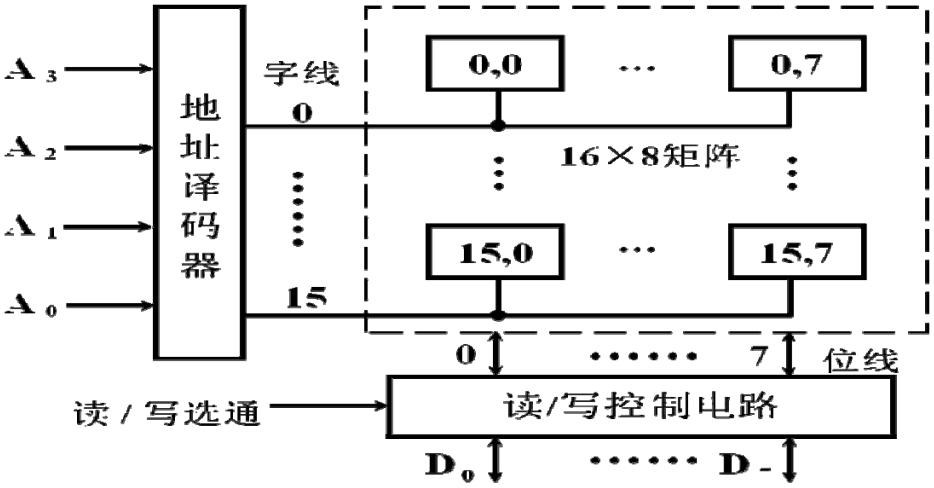


图 4-9　字片式结构存储器芯片

（2）位片式半导体存储器芯片

图 4-10 所示是 1K×1 位的位片式结构半导体存储器芯片结构图。采用多字 1 位结构，即 1024 个字排列成 32×32 的矩阵，中间每一个小方块代表一个存储元电路。

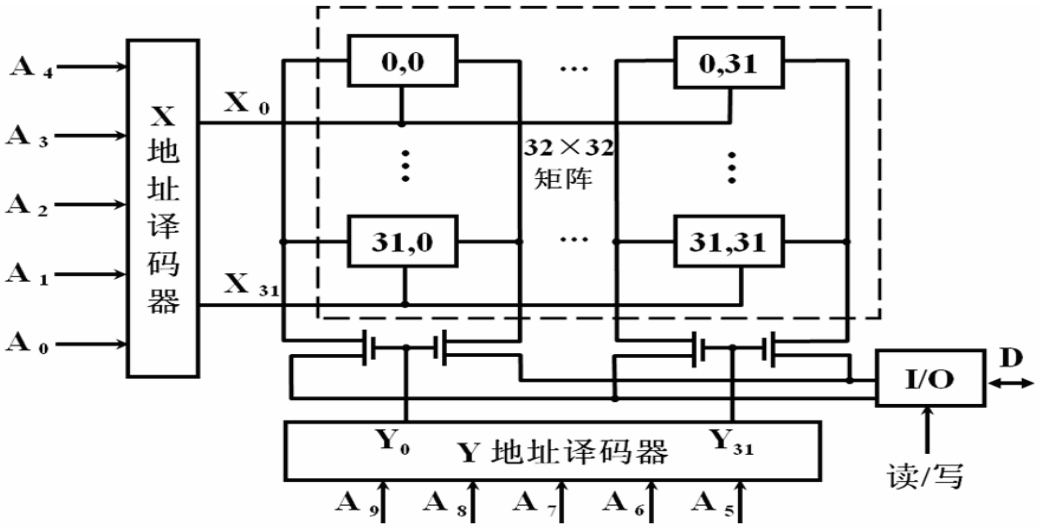


图 4-10　位片式结构存储器芯片

为了方便存取，给它们编上号，32 行的编号为 X_0、X_1、…、X_{31}，32 列的编号为 Y_0、Y_1、…、Y_{31}。这样每一个存储元电路便组成一个存储单元，都有一个唯一的固定编号（X_i 行、Y_j 列），这个编号称为存储单元的地址。地址译码器是将地址信息转换成有效的行选信号(X_i)和列选信号(Y_j)，从而选中某一存储单元。对于图 4-10 所示双译码方式的存储器芯片，行地址译码器采用 5∶32 译码器，即 5 条地址线 A_0、A_1 、…、A_4 作为译码器的输入，译码器的输出为 X_0、X_1、…、X_{31}；列地址译码器也采用 5∶32 译码器，地址线 A_5、A_6、…、A_9 作为译码器的输入，译码器输出为 Y_0、Y_1、…、Y_{31}，这样共有 10 条地址线用来寻址 1K 字（2^{10}=1K）。例如，输入地址为 $A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0$=0000000001，则行选信号 X_1＝1 和列选信号 Y_0＝1，所以选中第 X_1 行、第 Y_0 列存储单元中的 1 个存储元电路，从而实现对该单元中的这一个存储元电路进行读/写。

2、半导体存储器芯片工作原理

（1）半导体存储器芯片内部控制电路

1）读/写控制

对于被选中的存储单元，究竟进行读操作还是写操作，是由读/写控制逻辑电路进行控制。如果是读操作，则被选中存储单元中的数据经数据线、输入/输出线传送出去；如果是写操作，则将数据经过输入/输出线、数据线存入被选中单元中的各个存储元电路。

2）输入/输出控制

被选中的存储单元通过输入/输出端进行交换数据，读出时它是输出端，写入时它是输入端，即一线二用，由读/写控制信号控制。图 4-11 给出了一个简单的输入/输出控制电路。

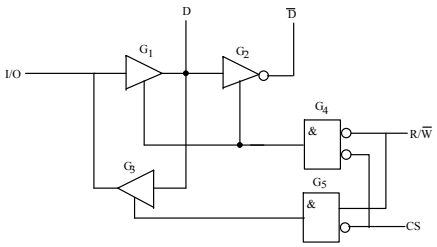


图 4-11　输入/输出控制电路

当选片信号 $\overline{CS}$＝1 时，G_5、G_4 输出为 0，三态门 G_1、G_2、G_3 均处于高阻状态，输入/输出（I/O）端与存储器内部完全隔离，存储器禁止读/写操作，即不工作。

当选片信号 $\overline{CS}$＝0 时，芯片被选通，如果 $R/\overline{W}$＝1 时，G_5 输出高电平，G_3 被打开，于是被选中单元中所存储的数据出现在 I/O 端，存储器执行读操作；如果 $R/\overline{W}$＝0 时，G_4 输出高电平，G_1、G_2 被打开，此时加在 I/O 端的数据以互补的形式出现在内部数据线上，并被存入到所选中的存储单元中，存储器执行写操作。

输入/输出端数据线的条数与被选中存储单元所存储的位数相同，例如 1024×1 位的位片式结构半导体存储器芯片，一个地址只能选中由 1 个存储元电路组成的存储单元，因此只有 1 条输入/输出线；对于 256×4 位的半导体存储器芯片，一个地址只能选中由 4 个存储元电路组成的存储单元，所以有 4 条输入/输出线。但也有半导体存储器芯片的数据输入线和输出线是分开的。

3）片选控制

一个半导体存储器芯片的存储容量总是有限的，计算机的主存往往是由一定数量的半导体存储器芯片按某种方式进行连接组合而成。当访问存储器时，一次只能访问主存中的某一片（或几片）半导体存储器芯片，为了方便实现对存储器芯片选择的控制，半导体存储器芯片往往设置有一条或几条片选信号（$\overline{CS}$或$\overline{CE}$）线。当芯片的片选信号有效时，该芯片被选中；当芯片的片选信号无效时，该芯片未被选中。存储器芯片的片选信号通常是由地址译码器的输出信号与

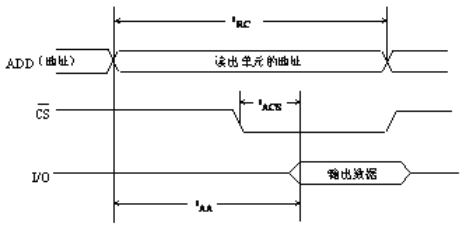


图 4-12　存储器读操作时序

一些控制信号（读写命令）来形成。

（2）半导体存储器芯片工作时序

1）读操作时序

读操作时序如图 4-12 所示。具体过程如下：

①欲读出单元的地址送地址总线 AB

②待地址稳定后形成有效的选片信号 $\overline{CS}$

③在 R / $\overline{W}$ 线上加高电平，经过一段延时后，所选择单元的内容出现在 I/O 端。

④使选片信号 $\overline{CS}$ 无效，I/O 端呈高阻态，本次读出过程结束。

由于地址缓冲器、译码器及输入/输出电路存在延时，所以在地址信号加到存储器上之后，必须等待一段时间，数据才能稳定地传输到数据输出端，这段时间称为地址存取时间，记为 t_{AA}。如果在存储器芯片的地址输入端已经有稳定地址的条件下，加入选片信号，从选片信号有效到数据稳定输出，这段时间间隔记为 t_{ACS}。显然在进行存储器读操作时，只有在地址和选片信号加入，且分别等待 t_{AA} 和 t_{ACS} 以后，被读单元的内容才能稳定地出现在数据输出端。图中 t_{RC} 为读周期，它表示该芯片连续进行两次读操作必须的时间间隔。

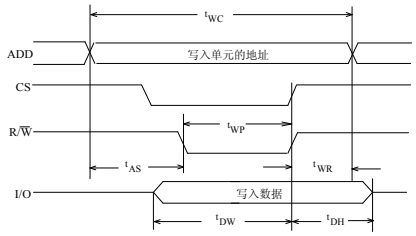


图 4-13 存储器写操作时序

2）写操作时序

写操作时序如图 4-13 所示。具体过程如下：

①将欲写入单元的地址送地址总线 AB

②在选片信号 $\overline{CS}$ 端加上有效电平，选中 SRAM 芯片

③将待写入的数据加到数据输入端

④在 R / $\overline{W}$ 线上加入低电平，进入写工作状态

⑤使选片信号无效，数据输入线回到高阻状态

由于地址改变时，新地址的稳定需要经过一段时间，如果在这段时间内加入写控制信号（即 R / $\overline{W}$ 变低），就可能将数据错误地写入其他单元。为防止这种情况出现，在写控制信号有效前，地址必须稳定一段时间，这段时间称为地址建立时间，记为 t_{AS}。同时在写信号失效后，地址信号至少还要维持一段写恢复时间（t_{WR}），为了保证速度最慢的存储器芯片的写入，写信号有效的时间不得小于写脉冲宽度（t_{WP}）。此外，对于写入的数据，应在写信号 t_{DW} 时间内保持稳定，且在写信号失效后继续保持 t_{DH} 时间。在时序图中还给出了写周期 t_{WC}，它反应了连续进行两次写操作所需要的最小时间间隔。对大多数静态半导体存储器来说，读周期和写周期是相等的，一般为十几到几十 *nS*。

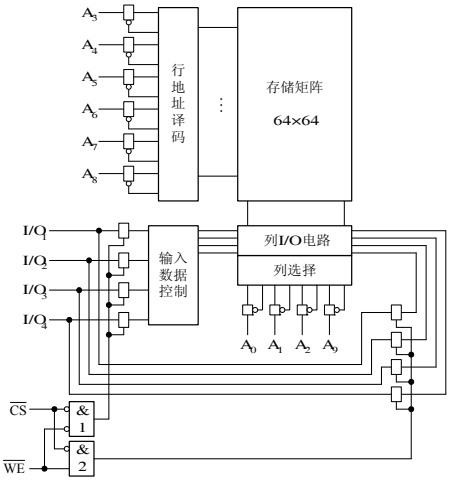
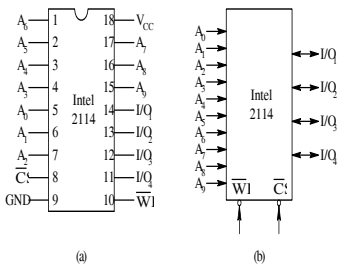


图 4-14　Intel 2114 内部结构

3、半导体存储器芯片实例

（1）静态存储器芯片（SRAM）

Intel 2114 SRAM 芯片的容量为 1K×4 位，



(a)　引脚；(b) 逻辑符号

图 4-15　Intel 2114 引脚及逻辑符号

芯片内部结构如图 4-14 所示。该芯片采用 18 脚封装，+5*V* 电源，芯片的实际引脚图和逻辑符号如图 4-15 所示。

由于 1K×4＝4096，所以 Intel 2114 SRAM 芯片有 4096 个基本存储电路，将 4096 个基本存储电路排成 64 行、64 列的存储矩阵，每根列选择线同时连接 4 位列线，对应于并行的 4 位(位于同一行的 4 位应作为同一单元的内容被同时选中)，从而构成了 64 行 16 列=1*K* 个存储单元，每个单元存储 4 位二进制信息。由于有 1K 个存储单元(2^{10}=1K)，所以 Intel 2114 SRAM 芯片应有 10 条地址输入信号线 A_0…A_9。由于芯片采用双译码方式，6 条地址 A_3…A_8 作为行地址译码输入，经行译码产生 64 条行选择线，其余 4 条 A_0、A_1、A_2 和 A_9 用于列地址译码输入，经过列译码产生 16 条列选择线。10 条地址线 A_0…A_9 送来的地址信号分别送到行、列地址译码器，经译码后选中一个存储单元(有 4 个存储元电路)。当片选信号 $\overline{CS}$＝0，$\overline{WE}$＝0 时，数据输入三态门打开，I/O 电路对被选中单元的 4 个存储元电路进行写入操作；当 $\overline{CS}$＝0，$\overline{WE}$＝1 时，数据输入三态门关闭，而数据输出三态门打开，I/O 电路将被选中单元内的 4 个存储元电路所存储的 4 位信息被读出并送数据线；当 $\overline{CS}$＝1，即 $\overline{CS}$ 无效时，不论 $\overline{WE}$ 为何种状态，各三态门均为高阻状态，芯片不工作。

（2）动态存储器芯片（DRAM）

DRAM 芯片的结构大体与 SRAM 芯片相似，是由存储矩阵和外围电路构成。不过 DRAM 芯片集成度高，存储容量大，导致芯片的地址引脚多，给制造芯片带来较大的难度，为此，DRAM 芯片的地址采用分时复用技术，即地址分两次送的方法，从而将地址线减少一半；另外 DRAM 芯片还需要刷新电路。

1）Intel 2164A DRAM 芯片

DRAM 芯片 Intel 2164A 芯片的存储容量为 64K×1 位，采用单管动态存储元电路，每个存储单元只有一个存储元电路，即一个存储单元只能存储 1 位数据，其内部结构如图 4-16 所示。Intel 2164A 芯片的存储体本应构成一个 256×256 的存储矩阵，为提高工作速度(需减少行列线上的分布电容)，将存储矩阵分为 4 个 128×128 矩阵，每个 128×128 矩阵配有 128 个读出放大器，各有一套 I/O 控制(读/写控制)电路。64K 字(2^{16}=64K)容量本需 16 条地址线，由于采用分时复用技术，芯片的地址线只需 8 条 A_7…A_0，其引脚如图 4-17 所示。在行地址选通信号 $\overline{RAS}$ 控制下，先将 8 位行地址 A_7…A_0 送入行地址锁存器，经译码后产生两组行选择线，每组 128 根。然后在列地址选通信号 $\overline{CAS}$ 控制下，将 8 位列地址 A_7…A_0 送入列地址锁存器，经译码后产生两组列选择线，每组 128 根。行地址与列地址选择 4 个 128×128 矩阵之一，因此，16 位地址是分成两次送入芯片的，对于某一地址码，只有一个 128×128 矩阵和它的 I/O 控制电路被选中。A_7～A_0 这 8 根地址线还用于在刷新时提供行地址，因为刷新是一行一行进行的。

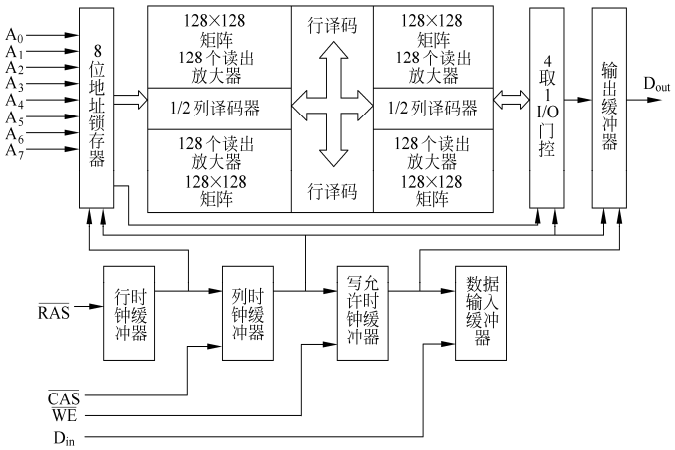


图 4-16　Intel 2164A 芯片内部结构

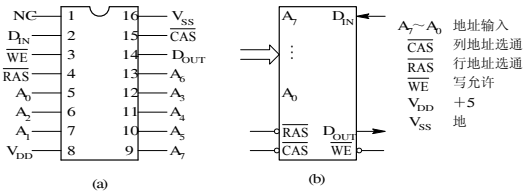


图 4-17　Intel 2164A 引脚与逻辑符号

(a) 引脚；(b) 逻辑符号

Intel 2164A 读/写操作由$\overline{WE}$信号来控制。读操作时，$\overline{WE}$为高电平，选中单元的内容经三态输出缓冲器从D_{OUT}引脚输出；写操作时，$\overline{WE}$为低电平，D_{IN}引脚上的信息经数据输入缓冲器写入选中单元。Intel 2164A 没有片选信号，实际上用行地址和列地址选通信号$\overline{RAS}$和$\overline{CAS}$作为片选信号，可见，片选信号已分解为行选信号与列选信号两部分。

2）动态存储器刷新方式

动态存储器的刷新方式有集中式、分散式和异步式三种。如图 4-18 所示。

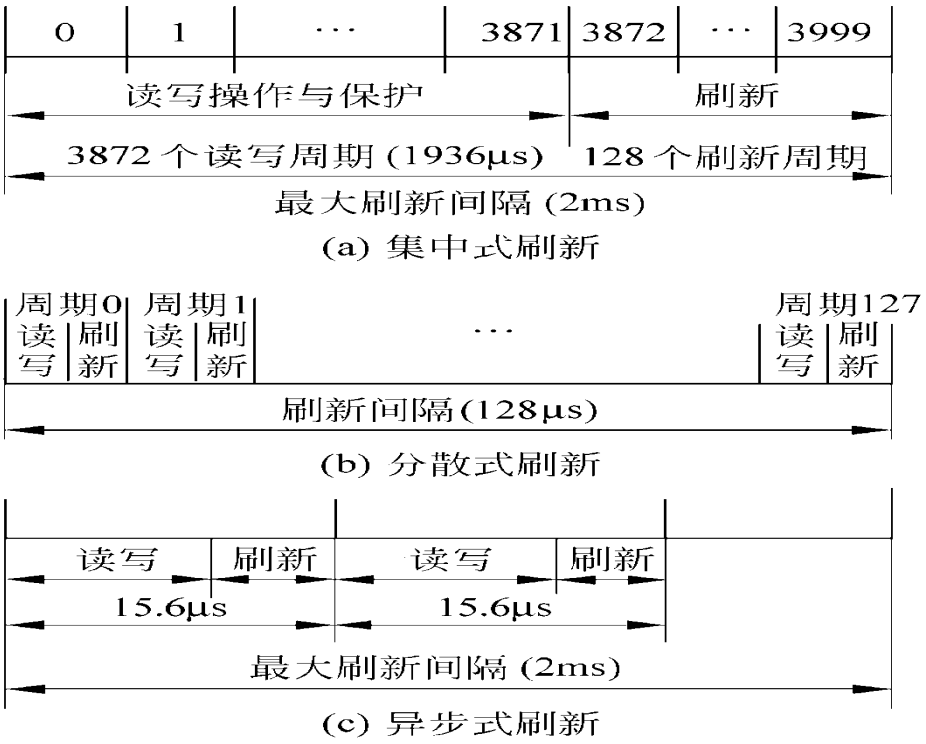


图 4-18　动态存储器刷新时间分配图

①集中式刷新

所谓集中式刷新，是指在允许的最大刷新周期内，根据存储容量的大小和存取周期的长短，集中安排一段刷新时间，在刷新时间内停止读写操作。例如，某一动态 RAM 由 128×128 存储矩阵组成，存取周期为 0.5μs，连续刷新 128 行，共需 128 个读周期，即一次刷新的总时间为 64μs。若刷新周期为 2ms，那么，2ms 内有 4000 个读写操作。在这 4000 个读写操作内，前面 3872 个周期用来进行读写或维持信息，后面 128 个周期用来刷新。集中式刷新时间分配图如图 4-18（a）所示。由此图可以看出，在读写操作时，不进行刷新操作，因此，读写操作不受刷新操作影响，读写速度较高。但在刷新时，必须停止读写操作。这段不能进行读写操作的时间称为“死区”。在本例中，这段“死区”占 4000 个周期中的 128 个，故死时间率为 3.2%。“死区”随存储矩阵行数的增加而增加，对于 256×256 存储矩阵来说，死时间率增加一倍。为了减少“死区”的时间，对于大容量的动态 RAM 芯片，可以采用在一个刷新周期内同时刷新多行的方法，以减少刷新周期数。

②分散式刷新

分散式刷新是指把每行存储单元的刷新分散到每个读写周期内进行，即把系统对存储器的访问周期分为两段，前一段用来读写数据或使存储器处于保持状态，后一段用来对存储矩阵的一行进行刷新。分散式刷新时间分配图如图 4-18（b）所示。这种刷新方式增加了系统对存储器的存储时间，如动态存储器芯片的存储时间为 0.5μs，则系统对存储器的存储时间为 1μs。对于前述 128×128 存储矩阵的芯片来说，这个存储器刷新一遍需要 128μs，就是以 128μs 作为间隔时间。这种刷新方法避免了“死区”，但加长了存储器的存储时间，降低了

整机的处理速度。而且刷新时间过于频繁，没有充分利用所允许的最大刷新间隔时间。这种方式不适用于高速存储器。

③异步式刷新

异步式刷新是上述两种方法的结合，它充分利用最大间隔时间并使“死区”缩短。对于128×128 存储矩阵的芯片来说，每行的刷新间隔时间是 2*ms*/128，即每隔 15.6*μs* 刷新一行。在 2*ms* 内分散地对 128 行轮流刷新一遍，刷新一行是只停止一次读写操作时间。分散式刷新时间分配图如图 4-18（c）所示。这样，对每一行来说，刷新时间仍为 2*ms*，而“死区”的长度则缩短为 0.5*μs*。

消除“死区”的方法，还可以采用不定期的刷新方法。即可以把刷新时间安排在 CPU 不访问内存的时间内进行。这种刷新方法没有单独占用 CPU 的时间，也没有“死区”，效率最高，但是，刷新的控制线路较复杂。

3）DRAM 芯片的存取模式

①标准模式的 DRAM 芯片

由 Intel 2164A DRAM 芯片可知，标准模式 DRAM 芯片的访问步骤：先给出所要访问存储单元的行地址并保持稳定，然后给出有效的行地址选通信号 $\overline{\text{RAS}}$，将行地址锁存到行地址译码器，此后再给出所要访问存储单元的列地址并保持稳定，然后给出有效的列地址选通信号 $\overline{\text{CAS}}$，将列地址锁存到列地址译码器，通过行、列地址译码器的译码，找到相应的存储单元，在读/写控制信号 $\text{R}/\overline{\text{W}}$ 作用下，实现对该存储单元的读或写操作。标准模式 DRAM 芯片的访问时间是指从芯片地址引脚上给出行地址开始，到可以使用出现在芯片数据引脚上的数据为止所需的时间。由于 $\overline{\text{RAS}}$ 信号失效后 DRAM 芯片尚需一个预充时间 t_{RP}，以便为下次访问做准备。所以，DRAM 芯片存取周期比访问时间要长，至少要长 t_{RP} 时间，这也是 DRAM 芯片与 SRAM 芯片的不同之处。为了消除 DRAM 芯片的预充时间所带来的负面影响，采用交错内存连接方法，即将两个内存条安排在一起使用，交替地访问两个内存条。当访问一个内存条的同时，另一个内存条执行预充操作，从而将预充时间隐藏在访问时间之中。

②页模式 DRAM 芯片

芯片内存储元电路组成的存储矩阵是芯片的核心，存储矩阵中的一行所包含的存储元电路的个数称为一页。由于绝大多数情况下对存储器的访问是连续的，所以没有必要像对标准模式 DRAM 芯片的访问，每次都要给出行地址和列地址。对页模式 DRAM 芯片的访问，如果所访问的存储元电路与上次访问的存储元电路在同一页中，只给出列地址，行地址保持不变，便可以选中并访问该存储元电路，与标准模式 DRAM 芯片比较，可以看出，第二次访问时间比第一次访问时间要短了许多。

③静态列模式 DRAM 芯片

静态列模式 DRAM 芯片与页模式 DRAM 芯片相似，访问某一行的第一个存储元电路所需要的时间是标准的 $\overline{\text{RAS}}$ 访问时间。当给出所要访问存储单元的行地址并保持稳定，然后给出有效的行地址选通信号 $\overline{\text{RAS}}$，将行地址锁存到行地址译码器，行地址在访问本行中的存储元电路的过程中保持不变，接着给出列地址，并给出有效的片选信号 $\overline{\text{CS}}$，然后，列地址在存储器芯片之外的一个增量寄存器中不断增量，并将每次增量后的地址信号送往存储器芯片，作为列地址译码器的输入，列地址译码器不断译码以确定要访问的存储单元。这样，只要 $\overline{\text{RAS}}$ 和 $\overline{\text{CS}}$ 始终保持低电平，同一行中各个存储元电路所存储的数据就连续地出现在静态列模式 DRAM 芯片的数据输出端，直到这一行中的最后一个存储元电路所存储的数据出现在数据输出端为止。

④半字节模式 DRAM 芯片

在对半字节模式 DRAM 芯片访问时，先给出行地址，并辅以有效的 $\overline{\text{RAS}}$ 信号，将行地址锁存，然后再给出列地址，同时 $\overline{\text{CAS}}$ 信号有效，将第一个列地址锁存。然后 $\overline{\text{RAS}}$ 信号保

持有效，行地址不再变化，而$\overline{CAS}$信号在有效和无效之间不停地切换，连续读出一行中的四位。可见半字节模式 DRAM 芯片类似于页模式 DRAM 芯片，只是页模式 DRAM 芯片一次要连续读出一行中的所有位，而半字节模式 DRAM 芯片只是读出一行中连续的 4 位。除此之外，半字节模式 DRAM 芯片与页模式 DRAM 芯片和静态列模式 DRAM 芯片的不同之处是：半字节模式 DRAM 芯片不需要设置列地址计数器电路。

⑤EDO DRAM 芯片

EDO DRAM 芯片是后期研发的一种高速存储器芯片，EDO DRAM（Extended Data-Out DRAM）称为扩展数据输出 DRAM，有时也称超级页模式 DRAM，事实上它是页模式 DRAM 的超级版本，为解决页模式 DRAM 芯片的局限性而研制开发的。

⑥同步 DRAM 芯片（SDRAM）

当 CPU 总线速度超过 75*Hz* 时，即使是 EDO DRAM 芯片也不能满足 CPU 速度的要求，所以人们开发了同步 DRAM。在所有传统 DRAM 芯片中（包括页模式 DRAM 芯片、EDO DRAM 芯片），DRAM 的时序与 CPU 的时序是不同步的，即 CPU 和 DRAM 芯片间没有一个公共的参考时钟。如果 CPU 访问 DRAM 时，DRAM 不能及时给出数据，它会发出 NOT READY 信号告知 CPU，CPU 通过在总线时序中插入等待周期来响应 NOT READY 信号。而 SDRAM 芯片，CPU 与 SDRAM 芯片间存在一个公共时钟信号，任何操作都与公共时钟信号同步，CPU 无需等待，从而实现突发模式操作。突发模式既可用于读操作，也可以用于写操作，为简便起见，在此只讨论突发模式的读操作。在突发读模式中，CPU 像正常情况一样提供第一个欲访问单元的地址，先给出$\overline{RAS}$信号，接着给出$\overline{CAS}$信号。由于 CPU 读 SDRAM 的内容用于填充 Cache，所以要一次读几个连续的单元（所读单元的个数取决于 Cache 结构）。因此，CPU 在给出第一个单元的地址后，后续单元的地址就无需再给出了，从而节省了建立地址和保持信息的时间。即可简单地通过编程把 SDRAM 设置为突发模式，告诉它一次要连续读的单元个数就可以了。每次突发读出的单元个数称为突发长度（Burst Length），可以是 1、2、4、8、16、256（整页）。为了进一步提高性能，SDRAM 芯片内部采用交错连接，即存储元电路的安排是遵循交错方式，从而实现了在访问一组存储元电路的同时刷新另一组存储元电路。如果 SDRAM 芯片融合了突发模式和交错连接两种技术，那么由 SDRAM 芯片构成的内存可用于总线频率高达 124*MHz*，若总线频率超高 125*MHz*，SDRAM 也不能满足性能要求了，只能采用速度更高的 DDR SDRAM（Double Data Rate SDRAM）。由于篇幅的限制，在此不再讨论，感兴趣的同学可以看相关的参考书。

（3）只读存储器芯片

半导体只读存储器芯片种类较多，就 EPROM 芯片而言，存在多种型号，常用的有 2716(2K×8)、2732(4K×8)、2764(8K×8)、27128(16K×8)、27246(32K×8)等。

1）Intel 2716 EPROM 芯片

①Intel 2716 芯片的内部结构和外部引脚

Intel 2716 EPROM 芯片采用 NMOS 工艺制造，双列直插式 24 引脚封装。其引脚、逻辑符号及内部结构如图 4-19 所示。11 条地址输入线 $A_{10}\cdots A_0$，其中 7 条用于行译码，4 条用于列译码。

$O_7\cdots O_0$：8 位数据线。编程写入时是输入线，正常读出时是输出线。

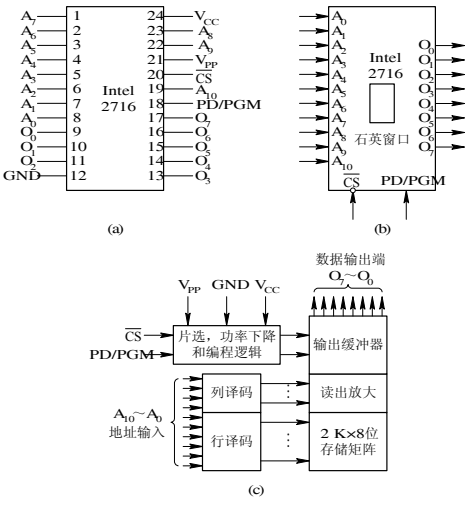


图 4-19　Intel 2716 的引脚、逻辑符号及内部结构

(a)引脚；(b) 逻辑符号；(c) 内部结构

$\overline{CS}$：片选信号。当$\overline{CS}$＝0 时，允许 2716 读出。

PD/PGM：待机/编程控制信号，输入。

VPP：编程电源。在编程写入时，VPP＝+25*V*；正常读出时，

VPP＝+5*V*。

VCC：工作电源，为+5*V*。

②Intel 2716 芯片的工作方式

Intel 2716 芯片的工作方式如表 4-1 所示。

读出方式：当$\overline{CS}$＝0 时，此方式可以将选中存储单元的内容读出。

未选中：当$\overline{CS}$＝l 时，不论 PD/PGM 状态如何，Intel 2716 芯片均未被选中，数据线呈高阻态。

待机(备用)方式：当 PD/PGM＝1 时，Intel 2716 芯片处于待机方式。这种方式和未选中方式类似，但其功耗由 525*mW* 下降到 132*mW*，下降了 75%，所以又称为功率下降方式，此时数据线呈高阻态。

编程方式：当 VPP＝+25*V*，$\overline{CS}$＝l，并在 PD/PGM 端加上 52*mS* 宽的正脉冲时，可以将数据线上的信息写入指定的地址单元。数据线为输入状态。

校验编程内容方式：此方式与读出方式基本相同，只是 VPP＝+25*V*。在编程后，可将 Intel 2716 芯片中的信息读出，与写入的内容进行比较，以验证写入内容是否正确。数据线为输出状态。

禁止编程方式：此方式禁止将数据总线上的信息写入 Intel 2716 芯片。

表 4-1　　Intel 2716 工作方式

工作方式	PD/PGM	$\overline{CS}$	电压（*V*）	数据线
读出	0	0	+5	输出
未选中	×	1	+5	高阻
待机	1	×	+5	高阻
编程	正脉冲	1	+25	输入
校验编程	0	0	+25	输出
禁止编程	0	1	+25	高阻

2）Intel 2816 E^2PROM 芯片

Intel 2816 芯片是 2K×8 位的 E^2PROM 芯片，有 24 条引脚，单一+5*V* 电源。其引脚如图 4-20 所示。其工作方式如表 4-2 所示。

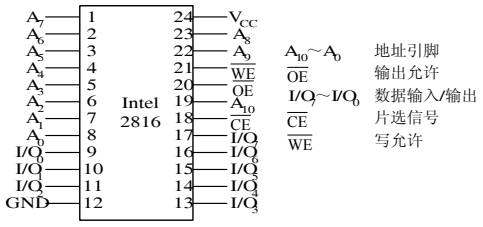


图 4-20　**Intel 2816** 的引脚

读出方式：当$\overline{CE}$=0，$\overline{OE}$=0，并且 VPP 端加+4～+6*V* 电压时，Intel 2816 芯片处于正常的读工作方式，此时数据线为输出状态。

待机(备用)方式：当$\overline{CE}$=1，$\overline{OE}$ 为任意状态，且 VPP 端加+4～+6*V* 电压时，Intel 2816 芯片处于待机状态。与 Intel 2716 芯片一样，待机状态下芯片的功耗将下降。

字节擦除方式：当$\overline{CE}$=0，$\overline{OE}$=1，数据线(I/O_0～I/O_7)都加高电平且 VPP 加幅度为+21*V*、宽度为 9～15*mS* 的脉冲时，Intel 2816 芯片处于以字节为单位的擦除方式。

整片擦除方式：当$\overline{CE}$=0，数据线(I/O_0～I/O_7)都为高电平，$\overline{OE}$ 端加+9～+15*V* 电压及 VPP 加 21*V*、9～15*mS* 的脉冲时，约经 10*mS* 可擦除整片的内容。

字节写入方式：当$\overline{CE}$=0，$\overline{OE}$=1，VPP 加幅度为+21*V*、宽度为 9～15 *mS* 的脉冲时，来自数据线(I/O_7～I/O_0)的数据字节可写入 Intel 2816 芯片的存储单元中。可见，字节写入和字节擦除方式实际是同一种操作，只是在字节擦除方式中，写入的信息为全“1”而已。

禁止方式：当$\overline{CE}$=1，VPP 为+4～+22V 时，不管$\overline{OE}$是高电平还是低电平，Intel 2816 芯片都将进入禁止状态，其数据线(I/O_0～I/O_7)呈高阻态，内部存储单元与外界隔离。

表 4-2　　2816 的工作方式

	$\overline{CE}$	$\overline{OE}$	V_{pp}/V	数据线状态
读出	0	0	4V~6V	输出
备用	1	X	4V~6V	高阻
字节擦除	0	1	21V	输入全 1
字节写入	0	1	21V	输入
整片擦除	0	+9~+14	21V	输入全 1
擦写禁止	1	X	4V~22V	高阻

4.2.4 存储器的扩展与应用

无论哪种存储器芯片，CPU 对存储器的访问，首先通过地址总线给出所要访问存储单元的地址，经地址译码器译码后，选中所要访问的单元，然后发出相应的读/写控制信号，最后才是数据在数据总线上进行传送。所以，存储器芯片与 CPU 的连接主要是存储器芯片上地址信号线、数据信号线和控制信号线与系统总线（地址总线 AB、数据总线 DB、控制总线 CB）的连接。由于一个存储器芯片的容量（N×M 位）总是有限的，因此，内存总是由一定数量的存储器芯片构成。面对种类繁多的存储器芯片，首先要考虑如何选择合理的芯片和需要多少芯片问题，其次是如何把这些芯片连接起来，最后是与系统总线的连接问题。

存储器芯片的选择通常要考虑存储器芯片的存取速度、存储容量、电源电压、功耗及成本等多个因素。确定某一存储器芯片后，根据内存容量大小，计算出所需存储器芯片的数量。假设存储器芯片的容量为 N×M 位，内存的容量为 K×L 位，由于内存容量要比一个存储器芯片的容量大得多，所以，K⩾N、L⩾M，所需存储器芯片的数量可按式（4-1）进行计算。

$$\text{芯片数} = \frac{K}{N} \times \frac{L}{M} \qquad (4\text{-}1)$$

式（4-1）中，K 和 N 分别是内存的字数（单元数）和芯片的字数（单元数），L 和 M 分别是内存中一个字的位数和芯片中一个字的位数。字数 K 和 N 一般为 2 的整次幂，且 K⩾N；内存中一个字的位数 L 与机器的字长相关，一般是字节的整数倍，而芯片中一个字的位数 N 可以是 1、4、8 等不等。所以，使用存储器芯片构成内存时，需要在字、位两方面进行扩展，具体扩展方法可以分为位扩展、字扩展和字位扩展。

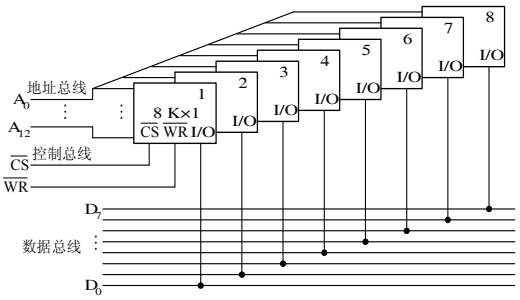


图 4-21 位扩展连接方式

1、存储芯片的扩展

（1）位扩展

位扩展是指存储器芯片的字(单元)数满足要求而位数不够，即 K=N、L>M，需对每个存储单元的位数进行扩展。图 4-21 给出了使用 8 片 8K×1 位的 RAM 芯片位扩展构成 8K×8 位存储器系统的连线图。由于存储器的

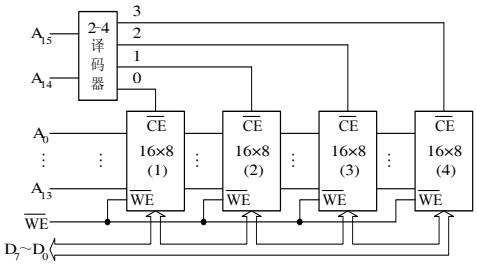


图 4-22　字扩展连接方式

字数与存储器芯片的字数一致，8K=2^{13}，故需 13 根地址线 $A_{12}\cdots A_0$ 对各芯片内的存储单元进行寻址。每一个芯片只有一条数据线，所以需要 8 片这样的芯片，将它们的数据线分别接到数据总线 $D_7\cdots D_0$ 上。因此，每一条地址线有 8 个负载，每一条数据线有一个负载。

位扩展法中，所有芯片都应同时被选中，各芯片$\overline{CS}$端可直接接地，也可并联在一起，根据地址范围的要求，与高位地址线译码产生的片选信号相连。若地址线 $A_{12}\cdots A_0$ 为全 0，即选中了存储器 0 号单元，该单元的 8 位信息是由各芯片 0 号单元的 1 位信息共同构成的。可以看出，位扩展的连接方式是将各芯片的地址线、片选$\overline{CS}$、读/写控制线进行并联，而数据线分别接到数据总线 $D_7\cdots D_0$ 上。

（2）字扩展

字扩展用于存储芯片的位数满足要求而字数不够的情况，即 K>N、L=M，需对存储单元数量的扩展。图 4-22 给出了用 4 个 16K×8 位的存储芯片构成一个 64K×8 位存储器连接图。

图 4-22 中 4 个芯片的数据线与数据总线 $D_7\cdots D_0$ 相并连；地址总线的低位地址 $A_{13}\cdots A_0$ 与各芯片的 14 位地址线连接，用于进行片内寻址；为了区分 4 个芯片的地址范围，还需要 2 根高位地址线 A_{14}、A_{15}，经 2：4 译码器产生 4 个输出信号，分别与 4 个芯片的片选端相连。采用如此连接后，各芯片的地址范围如表 4-3 所示。

表 4-3　各个芯片的地址空间分配情况

芯片	$A_{15}A_{14}$	$A_{13}A_{12}\cdots\cdots A_1A_0$	地址范围
1	**0　0**	0　0……0 0 ⋮　⋮　　⋮ 1　1　　1 1	0000H ⋮ 03FFH
2	**0　1**	0　0……0 0 ⋮　⋮　　⋮ 1　1　　1 1	4000h ⋮ 07FFH
3	**1　0**	0　0……0 0 ⋮　⋮　　⋮ 1　1　　1 1	8000h ⋮ 0BFFFH
4	**1　1**	0　0……0 0 ⋮　⋮　　⋮ 1　1　　1 1	0C000h ⋮ 0FFFFH

可以看出，字扩展的连接方式是将各芯片的地址线、数据线、读/写控制线进行并联，将地址总线中的低位地址线直接与各芯片地址线相连，以选择片内的某个单元，即实现片内寻址；而剩余的高位地址线用来产生片选信号，连接到各芯片的片选端，以选择某一个芯片，即实现对芯片的寻址。剩余的高位地址线如何用来产生片选信号呢？具体方法有：全译码法、部分译码法和线选法。

1）全译码法

所谓全译码法是指剩余的所有高位地址线都作为译码器的输入，其输出作为片选信号。也就是说，除了已经连接到存储器芯片上的地址线外，其他高位的地址线都被送入译码电路，以产生片选信号。采用全译码法所构成的存储器，存储单元的地址将是唯一的，不会出现地址重叠。

2）部分译码法

所谓部分译码法是指剩余的一部分高位地址线作为译码器的输入，其输出作为片选信

号。由于有些地址线未参与译码，则该地址线是“0”或“1”将不影响芯片的选中与否。采用部分译码法所构成的存储器，存储单元的地址将不再是唯一的，会出现一定程度的地址重叠。

3）线选法

所谓线选法是指剩余的高位地址线直接作为芯片的片选信号。也就是说，不在存储器芯片上的高位地址线直接作为存储器芯片的片选信号。使用线选法的好处是译码电路简单。但线选不仅导致一个存储单元有多个地址，还有可能一个地址同时选种多个单元，会引起数据总线的冲突。

（3）字位扩展

在实际应用中，往往会遇到字数和位数都需要扩展的情况。若使用 N×M 位的存储器芯片构成一个容量为 K×L 位 (K>N，L>M)的存储器，那么这个存储器共需要(N/K)×(M/L)个存储器芯片。连接时可将这些芯片分成 N/K 个组，每组有 M/L 个芯片。组内采用位扩展法，组间采用字扩展法。例如用 Intel 2114 RAM 芯片(1K×4)构成 4K×8 存储器，其存储器连接方法如图 4-23 所示。其中 8 片 Intel 2114 芯片分成 4 组(RAM_1、RAM_2、RAM_3 和 RAM_4)，每组 2 片。组内采用位扩展法构成 1K×8 的存储模块，4 个存储模块用字扩展法连接成 4K×8 的存储器。用 10 根地址线 $A_9\cdots A_0$ 对每组芯片进行片内寻址，同组芯片应被同时选中，因此，同组芯片的片选信号应并联在一起。2：4 译码器对两根高位地址线 $A_{11}A_{10}$ 进行译码，产生 4 个输出信号分别与各组芯片的片选端相连。

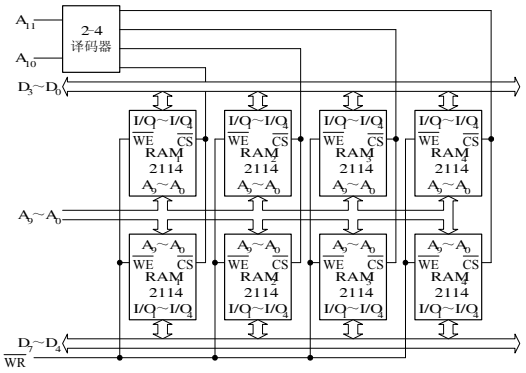


图 4-23　字位扩展连接方式

（4）用不同规格的存储器芯片扩展存储器

用不同规格的存储器芯片扩展存储器，地址范围的分配问题较为复杂，除了考虑字扩展和位扩展外，还要考虑如何选择合理的芯片和采用怎样的结构，关于结构问题将在 4.2.5 并行存储器中进行讨论，在此以具体的例子来说明地址译码器和存储器片选信号的连接。

【例 4-1】试用现有存储器芯片 EPROM（8K×8 位）和 SRAM（16K×1 位、2K×8 位、4K×8 位、8K×8 位）去构成主存储器，具体要求是：0～8191（8*KB*）为系统程序区，由只读存储器芯片组成；8192～32767（24*KB*）为用户程序区；最后 2*KB* 地址空间为系统程序工作区。若系统的地址总线 16 条 $A_{15}\cdots A_0$，双向数据总线 8 条（$D_7\cdots D_0$），控制总线与主存有关的信号有 $\overline{\text{MERQ}}$、R / $\overline{\text{W}}$。试从上述芯片中选择适当芯片设计该计算机主存储器，并画出主存储器逻辑框图。

解：根据主存储器的具体要求，8*KB* 的系统程序区可用 1 片 8K×8 位的 EPROM 芯片组成；24*KB* 的用户程序区可以用 SRAM 芯片组成，由于 SRAM 芯片种类较多，究竟选用哪一种芯片较合理，不仅要看芯片的位数与系统的数据总线是否匹配，还要看组成存储器的芯片所需的片选信号是否最少，只有这样，存储器的译码电路才能是最简单的，因此，这 24*KB* 的用户程序区可选用 3 片 8K×8 位的 SRAM 芯片组成；最后 2*KB* 的系统程序工作区可选用 1 片 2K×8 位的 SRAM 芯片组成。各个芯片的地址空间范围如表 4-4 所示。

表 4-4　　存储器各个芯片地址范围

$A_{15}A_{14}A_{13}$	$A_{12}A_{11}$	$A_{10}\cdots\cdots A_0$	各个芯片的地址范围
0　0　0	0　0 ⋮ 1　1	0……0 ⋮ 1……1	0000H～1FFFH （8*KB* EPROM）
	0　0	0……0	

0 0 1	┆ 1 1	┆ 1……1	2000H～3FFFH （8*KB* SRAM）
0 1 0	0 0 ┆ 1 1	0……0 ┆ 1……1	4000H～5FFFH （8*KB* SRAM）
0 1 1	0 0 ┆ 1 1	0……0 ┆ 1……1	6000H～7FFFH （8K×8 SRAM）
1 1 1	1 1	0……0 ┆ 1……1	F800H～FFFFH （2*KB* SRAM）

结合表 4-4 的地址分配情况，考虑到多数芯片是 8K×8 位的，所以选用地址总线中的低 16 位地址线 $A_{12}\cdots A_0$ 作为片内寻址，高 3 位地址线 $A_{15}A_{14}A_{13}$ 作为译码器的输入，其输出作为片选信号。对于 2K×8 的 SRAM 芯片片选信号需要另加门电路解决。主存储器连接图如图 4-24 所示。

【例 4-2】试用 4 片 4K×4 位的芯片和 3 片 8K×8 位的芯片扩展 32*KB* 主存储器。若系统的地址总线 16 条 $A_{15}\cdots A_0$，双向数据总线 8 条（$D_7\cdots D_0$），控制总线与主存有关的信号有：$\overline{\text{MERQ}}$、$\text{R}/\overline{\text{W}}$。

解：扩展 32*KB* 的主存储器需要 15 条地址线 $A_{14}\cdots A_0$、8 条数据线 $D_7\cdots D_0$ 和相关控制信号线。4 片 4K×4 位的存储器芯片构成 8*KB* 的主存，采用字位扩展法，每两片为一组，每组构成 4*KB* 的主存；3 片 8K×8 的存储器芯片构成 24*KB* 的主存储器，采用字扩展法。这 32*KB* 的主存储器的地址范围分配如表 4-5 所示。

存储器的具体连接要考虑控制线、数据线和地址线的连接。具体而言：

控制线的连接：控制线的连接较简单，只要将控制总线的读写控制信号 WE 与每一存储器芯片的读写控制线 WE 直接相连即可。

数据线的连接：3 片 8K×8 位 SRAM 芯片为 8 位数据宽度，将对应 8 位数据线与系统数据总线相连即可；4 片 4K×4 位 SRAM 芯片，分成两组，每组两片，每片数据宽度为 4 位。，将每组其中一片 4 位数据线连接到系统数据总线 $D_7\cdots D_4$ 上，另一片连接到 $D_3\cdots D_0$ 上。

地址线的连接：不同规格芯片的连接，关键在于译码器输入与输出的选择。对于输入线来说，通常以容量最小的芯片为基础。因此,容量大的芯片其片内高位地址线也要参加译码。

表 4-5　　32K×8 位主存储器地址范围

$A_{15}A_{14}A_{13}$	A_{12}	$A_{11}\cdots\cdots A_0$	各个芯片的地址范围
0 0 0	0 ┆ 1	0……0 ┆ 1……1	0000H～1FFFH （第三组 8*KB* 主存）
0 0 1	0 ┆ 1	0……0 ┆ 1……1	2000H～3FFFH （第四组 8*KB* 主存）
0 1 0	0 ┆ 1	0……0 ┆ 1……1	4000H～5FFFH （第五组 8*KB* 主存）
0 1 1	0	0……0 ┆ 1……1	6000H～6FFFH （第一组 4*KB* 主存）

0 1 1	1	0……0 ┇ 1……1	7000H～7FFFH （第二组 4*KB* 主存）

本例以最小容量 4K×4 位的存储器芯片为基础，32K 是 4K 的 8 倍，故选用 74LS138 译码器。用 3 条地址线 A_{14}、A_{13}、A_{12} 作为译码器的输入，其译码器的输出有 8 个信号，由上到下，000 选择第一组芯片，001 选择第二组芯片，第三组至第五组为 8K×8 位的芯片，有 8K 个单元，是第一、第二组芯片 4K 个单元的两倍，需用两条译码输出线选择，故各增加与门一个，由 010 和 011 共同选择第三组芯片，由 100 和 101 共同选择第四组芯片，110 和 111 共同选择第五组芯片。另外，由于系统地址总线有 16 条，而 32*KB* 主存储器需要 15 条地址线 A_{14}…A_0，A_{15} 地址线如果不参加译码，它可随机出现 0、1 两种情况，将其加于 A_{14}…A_0 决定的任何一个地址码上，可以出现两种不同的地址码，这种现象称为地址的重合，局部译码时必定会出现地址重合或覆盖现象。为避免地址重合或覆盖现象，必须采用全译码，实现全译码在此只需将 A_{15} 与 74LS138 的 G_1 相连，此时 A_{15} 为 1；或将 A_{15} 与 74LS138 的 G_{2a} 或 G_{2b} 相连，A_{15} 为 0，本例采用后者连接，32*KB* 主存储器具体连接如图 4-25 所示。

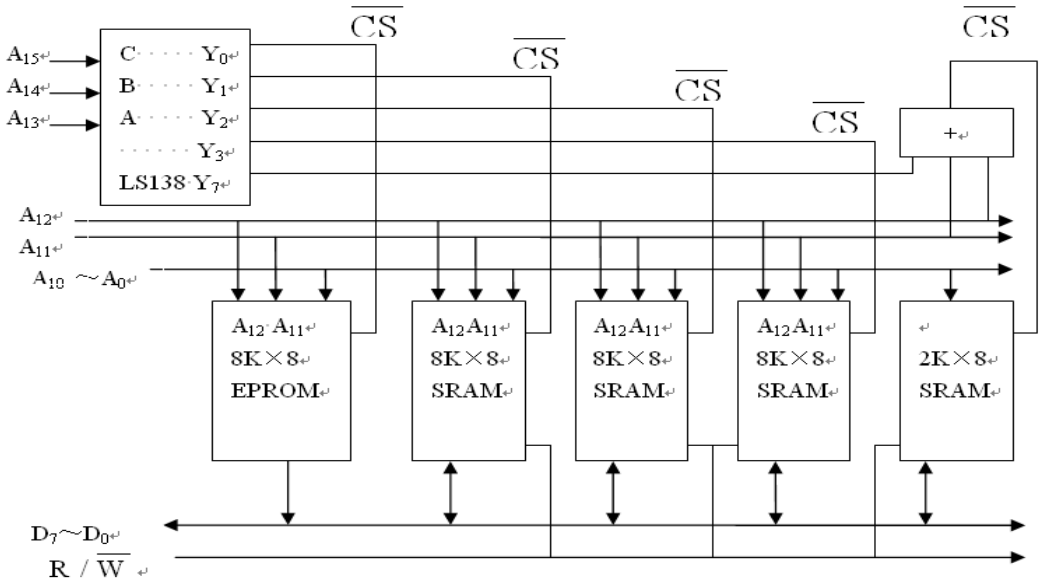


图 4-24 36*KB* 主存储器

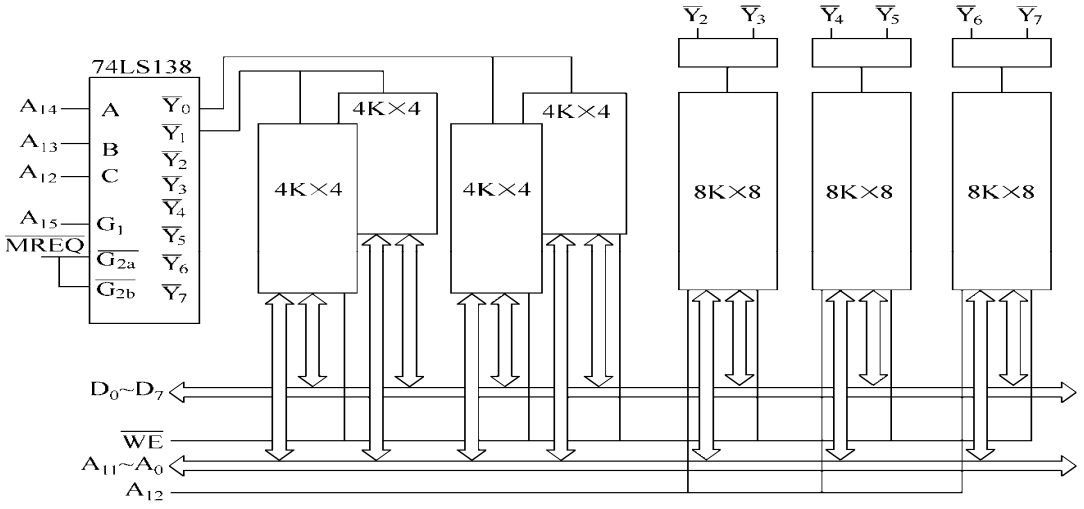


图 4-25　32*KB* 主存储器

4.2.5 并行存储器

随着软件规模的增大和系统性能要求的提高，对主存的要求是容量要大，速度要快。尽管主存的存取速度在不断地提高，但它的速度与 CPU 的速度相比，仍存在较大的差距，主存的存取速度是整个计算机系统速度的瓶颈。为了解决这个问题，存储器系统采用了层次结构，用虚拟存储器的方式扩大主存的存储容量，用高速缓冲存储器提高存取速度。除此以外，调整主存的组织结构来提高存取速度，也是一种行之有效的方法。在常规主存储器设计中，访问地址采用顺序方式，如图 4-26(a)所示。主存储器容量为 32 字，分成 4 个模块 M_0…M_3，每个模块存储 8 个字。访问地址按顺序分配给一个模块后，接着再按顺序为下一个模块分配访问地址。由此可见，在对顺序方式中的某个模块进行访问时，其他模块是不工作的，一旦某一模块出现故障，对其他模块不会造成影响。但由于各个模块是一个接一个串行工作，将会限制存储器的带宽。为了提高存储器的带宽，实现多模块流水式并行工作，访问地址采用交叉方式，如图 4-26(b)所示。地址的分配方法与顺序方式不同：先将 4 个线性地址 0，1，2，3 依次分配给 M_0，M_1，M_2，M_3 模块，再将线性地址 4，5，6，7 依次分配给 M_0，M_1，M_2，M_3 模块……直到全部线性地址分配完毕为止。可以看出，连续地址分布在相邻的不同模块内，而同一个模块内的地址都是不连续的，因此，对于连续字的块传送，交叉方式的存储器可以实现多模块流水式并行存取，从而大大提高存储器的带宽。

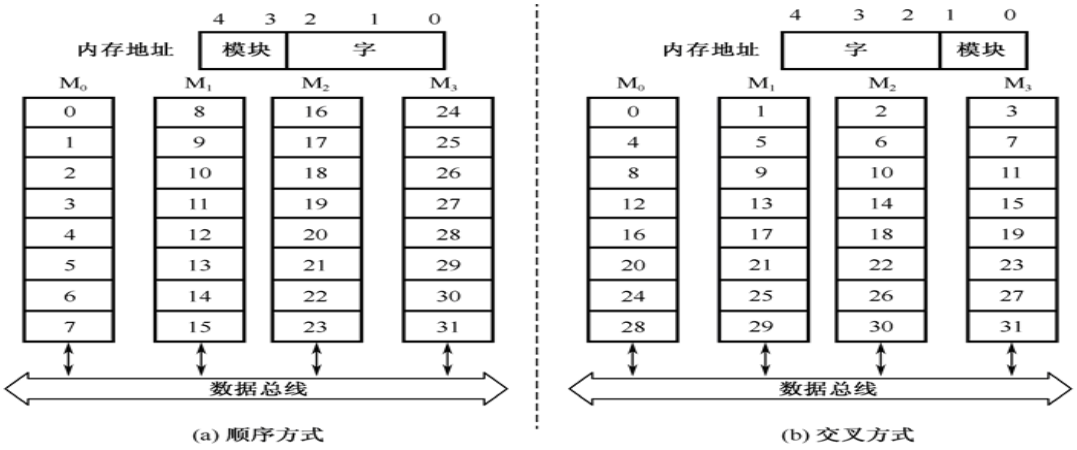


图 4-26　并行存储器

1、多体顺序并行存储器

前面所讨论的存储器，一个存取周期只能访问一个存储单元，无法实现并行操作，要想进行对存储器的并行处理，必须改变存储器的组织结构，将大容量的存储器分成若干个存储体，每个存储体都有自己的读写线路、地址寄存器和数据寄存器，并能以同等的方式与 CPU 交换信息。另外，每个存储体容量相等，它们既能同时工作又能独立编址。图 4-27 是多体顺序并行存储器原理图。

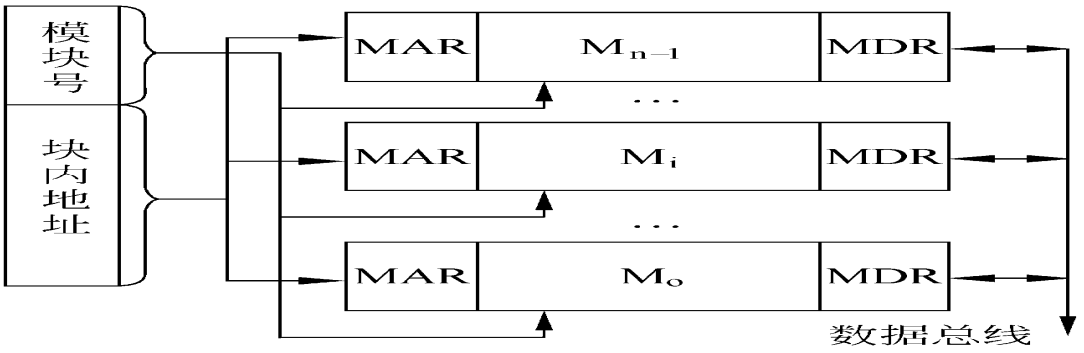


图 4-27　多体顺序并行存储器原理图

主存由 n 个容量相等的存储体组成，其中 MAR 为存储体地址寄存器，MDR 为存储体数据寄存器，主存地址寄存器的高位表示不同的存储体，低位表示存储体内单元地址，各体内地址寄存器指示存储单元的内容。这种结构使得各个存储体内相邻单元的地址是连续的，从而体现多体“顺序”这一特点，有利于并行处理，能够实现 n 个存储体的并行操作，即一次能访问并处理 n 个字。

2、多体交叉并行存储器

多体交叉并行存储器类似于多体顺序并行存储器，都是由多个相对独立的存储体组成，所不同的是多体交叉并行存储器的主存地址的高位表示存储体内单元地址，低位用于选择不同的存储体，这种组织形式使得各个存储体内相邻单元的地址是不连续的，而相邻存储体之间相同位置的单元地址是连续的，从而体现多体“交叉”这一特点。下面以一个四体交叉并行存储器为例，来说明多体交叉并行存储器的工作原理。如图 4-28 所示是一个四体交叉并行存储器，该存储器由 4 个独立的存储体 M_0、M_1、M_2、M_3 组成，低位地址用于选择不同的存储体，高位地址表示存储体内单元地址。

n 体交叉并行存储器的编址规则如下：

规则 1，地址连续的两个单元分布在相邻的两个存储体中，地址按存储体方向顺序编号。

规则 2，同一存储体内相邻的两个单元地址之差等于 n。例如在四体交叉并行存储器中，同一存储体内相邻的两个单元地址之差等于 4。

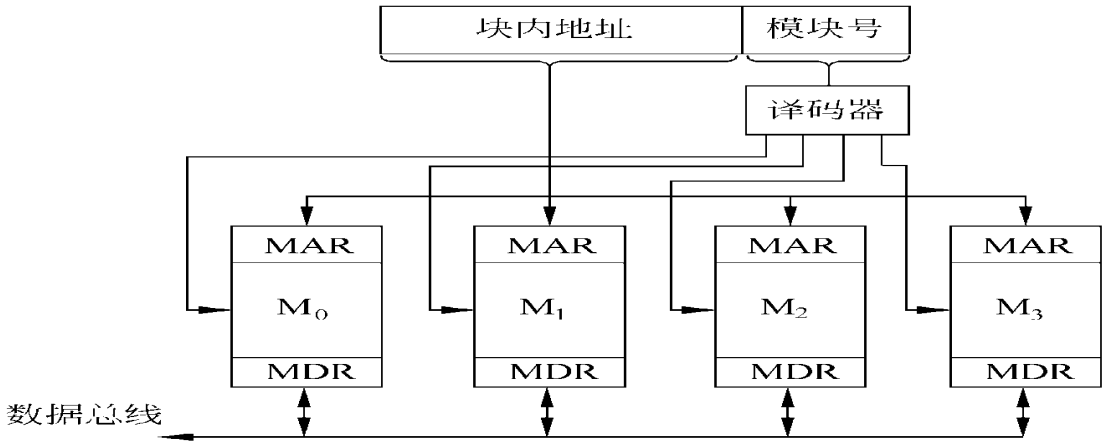


图 4-28　多体交叉并行存储器原理图

规则 3，任何一个存储单元地址编号的末 $\log_2 n$ 位正好指示该单元所属存储体的编号，访问主存时只要判断这几位就能决定访问的是哪个存储体。在四体交叉并行存储器中，M_0 存储体中每个单元地址的最后两位都是“00”，M_1 存储体中每个单元地址的最后两位都是“01”，M_2 存储体中每个单元地址的最后两位都是“10”，M_3 存储体中每个单元地址的最后两位都是“11”。

规则 4，同一存储体中每个单元地址除去存储体编号后的高位地址正好是存储体中单元的顺序号，由此就可决定访问单元在存储体中的位置。

多体交叉并行存储器地址交叉排列的目的是为了便于各个存储体同时工作。假设 CPU 要取 4 条长度为一个字长的指令，这 4 条指令分别存放在地址为 0、1、A、B 的 4 个单元中，这 4 个单元分别在 M_0、M_1、M_2、M_3 四不同的存储体中。为了在一个存取周期内能访问 4 个指令字，在多体并行主存系统中采用了分时工作的方法，目前普遍采用的是分时读出法，即每个存储体每次读写一个字，各存储体分时启动，即每隔四分之一存取周期启动一个存储体，其时序如图 4-29 所示，M_0 存储体在第一个主存周期开始读写，经过 $1/4T_M$ 启动 M_1 存储体，M_2 和 M_3 存储体分别在 $1/2T_M$、$3/4T_M$ 时刻开始它们各自的读写操作，4 个存储体以 $1/4T_M$ 的时间间隔进入并行工作状态。

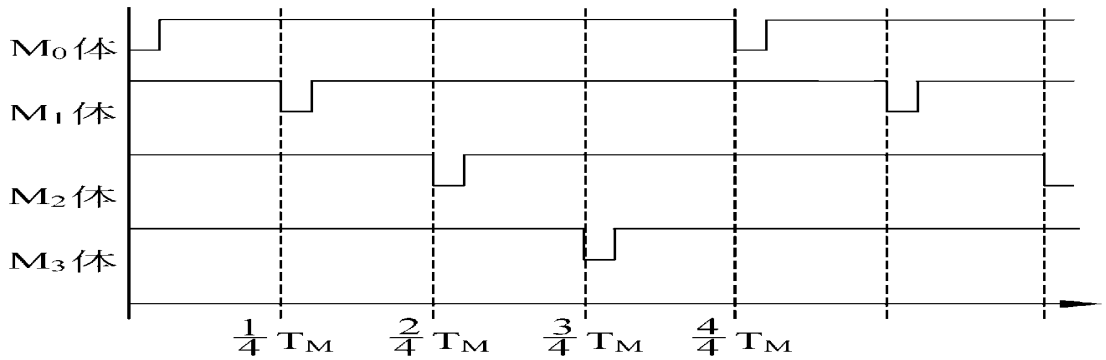


图 4-29　四体交叉并行存储器工作时序

4.3　高速缓冲存储器

4.3.1 Cache 基本原理

1、Cache 与程序访问的局部性

虽然 CPU 主频的提升会带动系统性能的改善，但系统性能的提高不仅仅取决于 CPU，还与系统架构、存储部件的存取速度、信息在各个部件之间的传送速度、指令结构等因素有关，特别是主存的存取速度。如果 CPU 工作速度较高，而主存的存取速度较低，必然会造成 CPU 的等待，从而降低了 CPU 的处理速度和效率。如 400*MHz* 的 PIII，一次指令执行时间为 2*nS*，与其相配的主存 SDRAM 存取时间为 10*nS*，比前者慢了 5 倍。如何解决 CPU 与主存之间的速度匹配问题？一种方法是在基本总线周期中插入等待，尽管能处理好速度匹配问题，但无法保证 CPU 较高的工作速度；另一种方法是采用存取速度较快的 SRAM 做主存，却又大幅度提升了系统成本。如何在不大增加系统成本的前提下又能保证系统性能的提升？人们开始对大量典型程序运行情况进行分析，结果表明：在一个较短的时间间隔内，由程序产生的地址往往集中在存储器逻辑地址空间的很小范围内。指令地址的分布本来就是连续的，再加上循环程序段和子程序段要重复执行多次。因此，对这些地址的访问就自然具有时间上集中分布的倾向。数据分布的集中倾向虽不如指令明显，但对数组的存储和访问以及工作单元的选择，都可以使存储器地址相对集中。这种对局部范围的存储器地址频繁访问，而对此范围以外的地址访问甚少的现象，称之为程序访问的局部性。根据程序访问的局部性原理，在主存和 CPU 之间设置一个高速的、容量相对较小的存储器，该存储器包括管理在内的全部功能都由硬件实现，如图 4-30 所示。把正在执行的指令地址附近的一部分指令或数据从主存调入这个存储器，起到缓冲作用，以实现速度匹配。这种方法既保证了系统成本增加不大，又保证了系统性能的有效提升，是目前计算机普遍采用的一种行之有效的技术。为此，我们把介于 CPU 和主存之间的高速、小容量、起缓冲作用的存储器称作高速缓冲存储器(Cache)。基于目前大规模集成电路技术和生产工艺，在 CPU 芯片内部做成一定容量的 Cache，称之为一级（L1）Cache，CPU 外部由 SRAM 构成的 Cache 称为二级（L2）Cache。最新的 CPU 内部已经出现二级甚至三级 Cache。

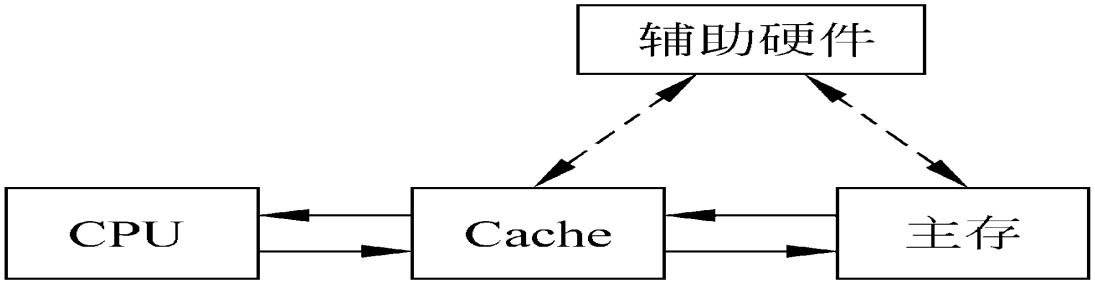


图 4-30 Cache 与主存的关系

2、Cache 的工作原理

CPU 与 Cache 之间的数据交换是以字为单位，而 Cache 与主存之间的数据交换是以块为单位，一个块由若干个字组成的。当 CPU 读取主存中一个字时，便发出此字的内存地址到 Cache 和主存，Cache 控制逻辑根据地址判断此字当前是否在 Cache 中。如果在 Cache 中，我们称之为命中，命中时 CPU 通过访问 Cache 把此字读出送给 CPU；如果不在 Cache 中，我们称之为不命中，不命中时 CPU 用主存读周期把此字从主存读出送到 CPU，与此同时，把含有这个字的整个数据块从主存读出送到 Cache 中。系统正是依据此原理，不断地与当前指令集相关联的一个不太大的后继指令集从内存读到 Cache，然后再与 CPU 高速传送，从而达到速度匹配。CPU 对存储器进行数据请求时，通常先访问 Cache，由于局部性原理不能保证所请求的数据百分之百地在 Cache 中，这里便存在一个命中率，即 CPU 在任一时刻从 Cache 中获取数据的几率。命中率越高，从 Cache 中获取数据的可能性就越大，一般来说，Cache 的存储容量比主存的容量小得多，但不能太小，太小会使命中率太低；也没有必要过大，过大不仅会增加成本，而且当容量超过一定值后，命中率随容量的增加将不会有明显地增长。只要 Cache 的空间与主存空间在一定范围内保持适当比例的映射关系，Cache 的命中率还是相当高的。一般规定 Cache 与主存空间比为 4：1000，即 128*KB* Cache 可映射 32*MB* 主存、256*KB* Cache 可映射 64*MB* 主存，在这种情况下，命中率都在 90％以上。至于没有命中的数据，CPU 只好直接从主存获取，获取的同时，也把它拷贝到 Cache，以备下次访问。

4.3.2 Cache 的结构

Cache 通常由相联存储器实现。相联存储器的每一个存储块都具有额外的存储信息，称为标签 Tag。当访问相联存储器时，将地址和每一个标签进行比较，从而对标签相同的存储块进行访问。为了说明 Cache 的结构，设块的大小为 2^b 个连续的字，将整个主存按块的大小进行分块，共分为 2^m 个块，其块号用 j 表示，其中 j=0，1，2，…，2^m -1；将 Cache 也按块的大小进行分块，共分为 2^c 个块，其块号用 i 表示，其中 i=0，1，2，…，2^c -1。地址映射是指把主存地址空间映射到 Cache 地址空间的规则，即主存块与 Cache 块之间的对应关系，此规则可以表示为 i 与 j 之间的函数关系，根据 i 与 j 之间的函数关系（映射方式）的不同，可以将 Cache 的结构分为全相联映射 Cache、直接映射 Cache、组相联映射 Cache 三种基本结构。不同结构的 Cache 必然有与之对应的地址变换方法，所谓地址变换是指程序在实际运行过程中，如何把主存地址变换成 Cache 地址的方法。

1、全相联映射

在全相联映射 Cache 中，地址映射函数关系为：i=j，即主存中的任意一块 j 可以映射到 Cache 中任何一块位置上。全相联映射方式如图 4-31 所示。全相联映射方式的地址变换如图 4-32 所示。主存地址包括主存块号 j 和块内地址 w。主存块号 j 作为相联查找的关键字段，与目录表中的主存块号字段进行比较，若找到一个与主存块号 j 相同的主存块号，并且有效位为 1 时，则取其 Cache 块号 i 与块内地址 w 拼接在一起，形成一个 Cache 地址。如果在相联查找中没有发现与主存块号 j 相同的主存块号，表明该主存块尚未装入 Cache，则调入该主存块到 Cache 中，并修改目录表。

由于任何一个主存块可以存放在任何一个空闲 Cache 块的位置上，因此，全相联映射方式发生两个主存块争用一个 Cache 块位置的冲突概率较低，Cache 的空间利用率也因此较高。但是由于全相联映射方式的地址变换需要容量为 C 个存储字的相联存储器来存放目录表，Cache 越大，要求相联存储器的容量也越来越大，过大的相联存储器不仅价格昂贵，而且也会降低地址变换的速度。

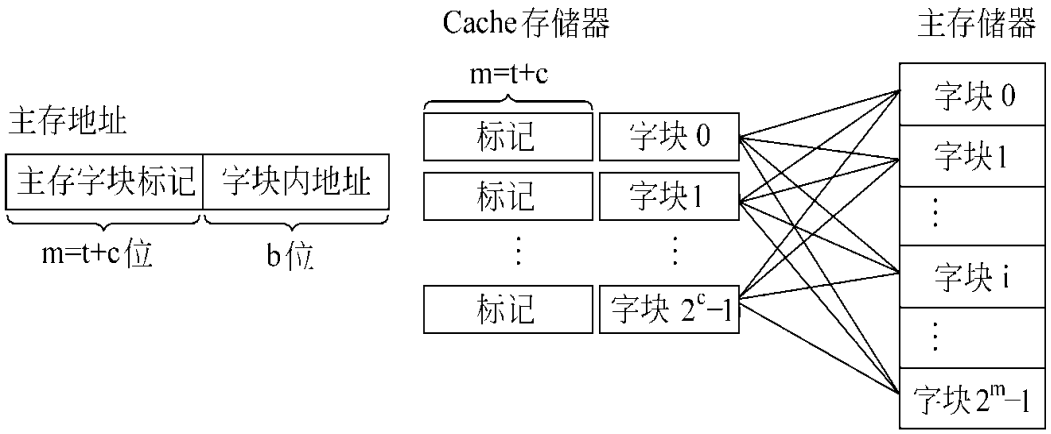


图 4-31　全相联映射方式

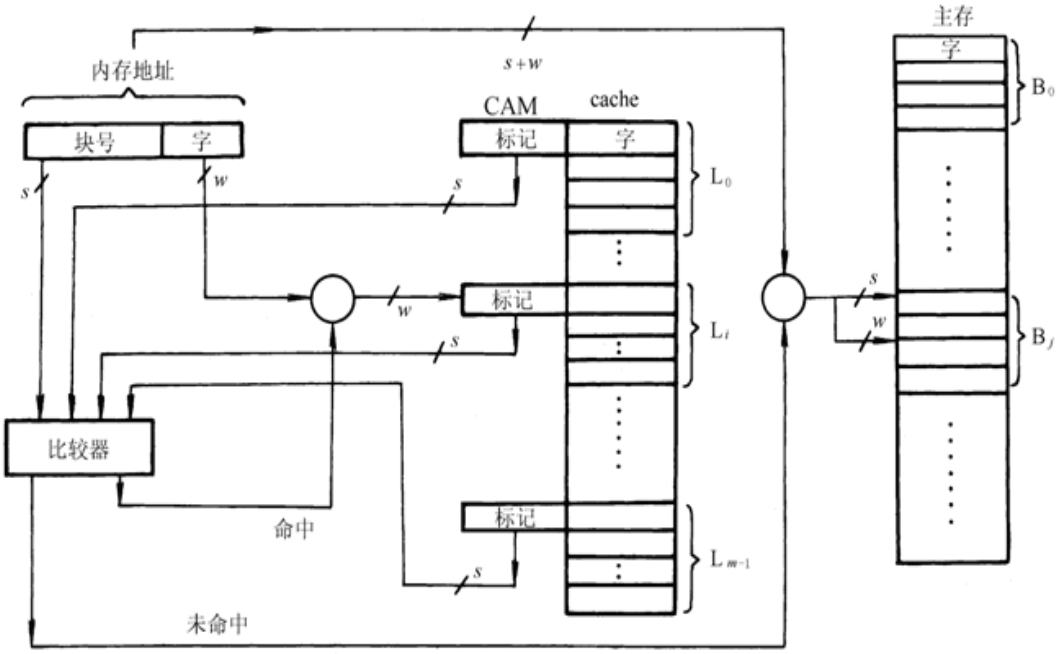


图 4-32 全相联映射方式的地址变换

【例 4-3】假设某计算机系统中 Cache 容量为 32*KB*，块大小是 16 个字节，主存容量为 1*MB*，地址映射为全相联映射方式。

（1）　主存地址多少位？如何分配？

（2）　Cache 地址多少位？如何分配？

（3）　目录表的格式和容量？

解：（1）主存地址共 20 位，1*MB*=2^{20}*B*。主存可分为 4*MB*/16*B*=2^{16} 个块，主存地址为：块号（16 位）、块内字地址（4 位）

（2）Cache 地址为 15 位：块号（11 位）、块内字地址（4 位）

（3）目录表的格式为：主存块号（16 位）+Cache 块号（11 位）+有效位（1 位）。目录表容量为 Cache 的块数，即 2*KB*。

2、直接映像

直接映像 Cache 中，地址映射函数关系为：i=j mod 2^c，即主存中的任意一块 j 可以映射到 Cache 中满足 i=j mod 2^c 的块位置上。C 为 Cache 的块容量，将主存按 Cache 块容量大小进行分区，可分为 P=M/C 个区，其区号用 k 表示，k=0，1，…，P-1。在直接映射方式中，

主存的一个区与整个 Cache 相对应，因此，Cache 的地址与主存地址中除区号外的低位地址完全相同。主存地址由区号 k、块号 j、块内地址 w 组成，用块号 j 去访问区表存储器，把从中读出的区号与主存地址中区号 k 进行比较，如果区号相同，并且有效位为 1 时，则命中，主存地址中块号 j 与块内地址 w 直接作为 Cache 地址，从而访问 Cache；如果区号相同，但有效位为 0 时，则表示 Cache 中虽然已有要访问的块，但该块与已经被修改的主存副本块内容不一致，是一个作废的块，需要再从主存中调入并重写该块，并且重写后将有效位置 1。如果区号不相同，并且有效位为 1 时，则表示没有命中，但该块为有效块，需要把该块调出，写入主存，并修改主存中相应的副本块；如果区号不相同，并且有效位为 0 时，则表示没有命中，可以直接调入所需要的新块。

直接映射方式的地址变换如图 4-33 所示。主存地址由三部分组成，末 b 位为字块内地址，中间 c 位为 Cache 字块地址，高 t 位（t=m-c）是主存字块标记，也就是记录在相应 Cache 字块标记中的内容，当有效位为“1”时，表明该数据块是主存中一块数据的副本。Cache 在接收到 CPU 送来的主存地址和读写命令后，用中间 c 位字段找到对应的 Cache 中字块，然后将其标记与主存地址的高 t 位比较，如果两者相等，而且有效位为“1”，则可根据 b 位块内地址，从 Cache 中取得所需的指令或数据；如果两者不相等，或者有效位为“0”，就从主存读出新的字块替换 Cache 中旧的字块，同时修改 Cache 中标记，并将数据送给 CPU。

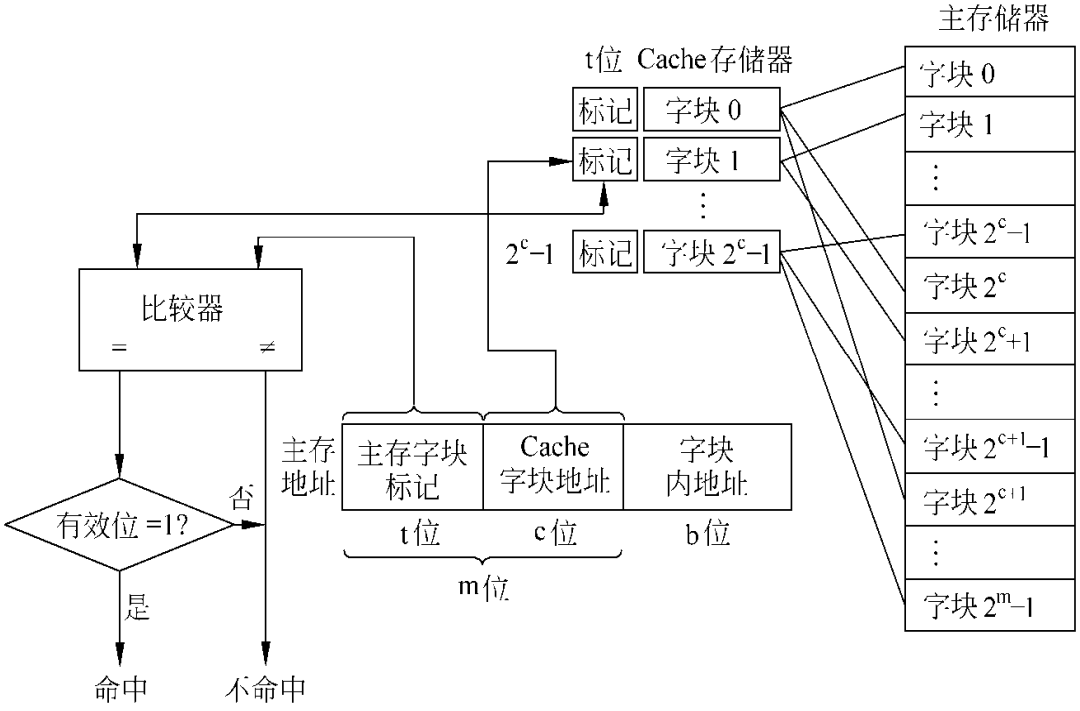


图 4-33　直接映射方式地址变换

直接映射方式映射函数简单，实现的硬件简单，地址变换速度快，一旦命中且有效，主存地址中除区号外的低位地址直接变换成 Cache 的地址。但也存在比较突出的缺点，Cache 的利用率不高，其原因是，当两个以上的主存块映射到相同的 Cache 块位置而发生冲突时，即使其他 Cache 块位置空闲也不能被利用。

【例 4-4】假设某计算机系统中 Cache 容量为 64*KB*，块大小是 16 个字节，主存容量为 4*MB*，地址映射为直接映射方式。

（1）　主存地址多少位？如何分配？

（2）　Cache 地址多少位？如何分配？

（3）　目录表的格式和容量？

解：（1）主存地址共 22 位，4*MB*=2^{22}*B*。主存可分为 4*MB*/64*KB*=64 个区，每个区的块

数为：64*KB*/16=4*KB*，因此，主存地址为：区号（6 位）、块号（12 位）、块内字地址（4 位）。

（2）Cache 地址为 16 位：块号（12 位）、块内字地址（4 位）

（3）目录表的格式为：区号（6 位）+有效位（1 位）。目录表容量与缓冲块容量相同，4*KB*。

3、组相联映射

组相联映射 Cache 是介于全相联映射 Cache 和直接映射 Cache 之间的一种结构，是目前采用较多的一种地址映射方式。组相联映射方式是把整个主存按 Cache 的容量 C 进行分区，区号为 k，每个区和 Cache 再按同样大小划分成数量相等的组，组号为 g，组内再划分成块，主存的组到 Cache 的组之间采用直接映射方式，对应组内各个块之间采用全相联映射方式。用于地址变换的块表容量与 Cache 的块容量相等，块表的字长包含主存地址的区号、组内块号、Cache 地址的组内块号、1 位有效位。组相联映射方式的地址变换如图 4-34 所示。主存地址由区号 k、组号 g、组内块号 j、块内地址 w 组成，用组号 g 去访问块表存储器，从中读出一组字，字数为组内的块容量。把这些字中的区号和块号与主存地址中区号 k 和块号 j 进行相联比较，如果发现有相等的，则表示 Cache 中已有要访问的块，若有效位为 1，则命中，从这个字中取出 Cache 的块号 i 与主存地址中的组号 g 与块内地址 w 拼接起来形成 Cache 地址；如果相联比较不等或有效位为 0，则不命中，应调入新块。

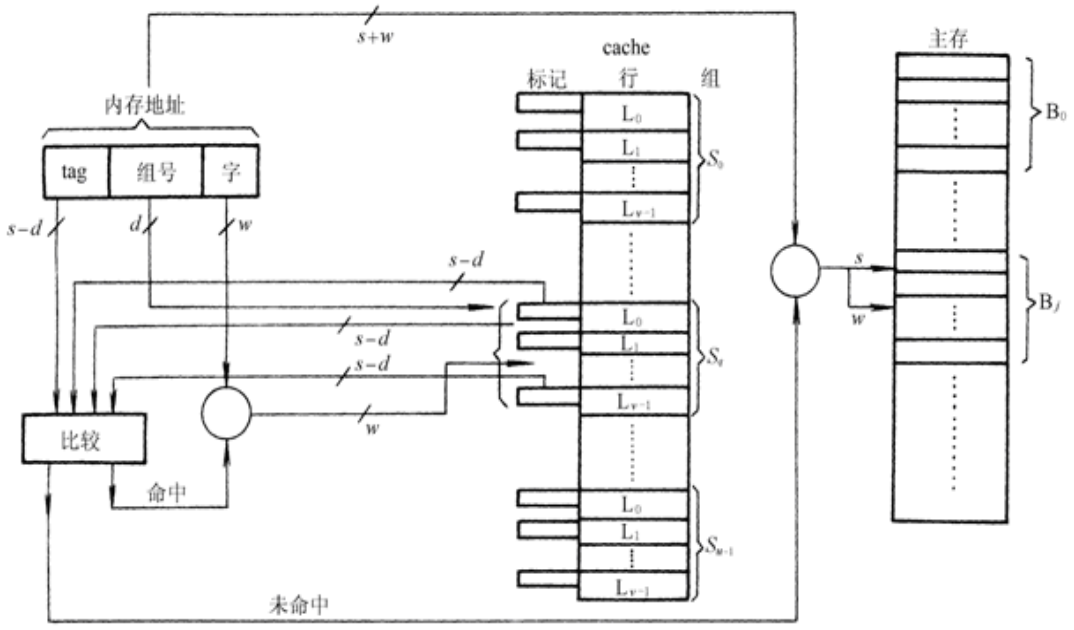


图 4-34 组相联映射方式的地址变换

【例 4-5】假设某计算机系统中 Cache 容量为 32*KB*，块大小是 64 个字节，缓存共分 128 个组，主存容量为 1*MB*，地址映射为组相联映射方式。

（1） 主存地址多少位？如何分配？

（2） Cache 地址多少位？如何分配？

（3） 块表的格式和容量？

解：（1）主存地址共 20 位，1*MB*=2^{20}*B*。主存可分为 1*MB*/32*KB*=2^5 个区，区号为 5 位；Cache 共分 128 个组，组号为 7 位；块内地址 w 为 6 位；块号为 32*KB*/128/64=2^2，即 2 位。

（2）Cache 地址为 16 位：组号（6 位）、块号为（2）、块内字地址（6 位）

（3）目录表的格式为：区号（5 位）+主存组内块号（2）+Cache 组内块号（2）+有效位（1 位）。块表容量与 Cache 的块容量相同，即：组数×组内块数=128×4=512。

4.3.3 Cache 的读写过程

1、Cache 的读过程

当 CPU 发出读请求时，如果 Cache 命中，通过地址变换，就可以直接对 Cache 进行读操作，不需要对主存进行读操作；如果 Cache 未命中，则 CPU 直接对主存进行读操作，同时将相应的块调入到 Cache，若此时 Cache 已满，如何将 Cache 中的某块调出，把新块调入？这就需要替换策略。常用的替换算法有以下四种。

（1）随机算法（Random，RAND）

随机算法是一种最简单的替换算法，不考虑 Cache 中块的过去、现在和将来的使用情况，随机地选择一块进行替换。

（2）先进先出算法（First-In First-Out，FIFO）

先进先出算法是按调入 Cache 的先后顺序，在需要替换时将最先调入 Cache 中的块替换掉。这种方法容易实现，而且系统开销较小。

（3）最近最少使用算法（Least Recently Used，LRU）

最近最少使用算法是根据块的使用情况，将 CPU 最近最少使用的块作为被替换的对象。这种替换方法需要随时记录 Cache 中各个块的使用情况，以便确定 Cache 中哪一个块是最近最少使用的块。LRU 算法相对比较合理，但实现起来较为复杂。为此，有人提出最久未使用算法（Least Frequently Used，LFU），将“使用的多少”问题转换为“有没有使用”，从而把算术问题转换成逻辑记录问题，简化了算法的实现过程，从而被广泛应用。

（4）最优化算法（OPTimization，OPT）

最优化算法是一种以将来使用最少作为替换目标的算法，该算法是一种理想化算法，现实中很少使用，它常被用来评价其他替换算法优劣的标准。

2、Cache 的写过程

当 CPU 发出写请求时，如果 Cache 未命中，则 CPU 直接对主存进行写操作，与 Cache 无关；如果 Cache 命中，就会出现如何保持 Cache 中的内容与主存中内容的一致性问题，一般的处理方法有如下两种。

（1）写直达法(Write-Through)

CPU 同时写主存和 Cache，以保证主存的数据能同步地更新。这种方法实现简单，但可能增加多次不必要的主存写入。

（2）回写法(Write-Back)

将 CPU 要写的信息暂时只写入 Cache，并用标志将该块注明，直到该块从 Cache 中替换出去时才一次写回主存。这种方法操作速度快，但因主存中的字块未经随时修改而有可能出错。

4.4 虚拟存储器

4.4.1 概述

1、虚拟存储器

虚拟存储器（Virtual Memory）是建立在主存—辅存的物理结构基础之上，不断发展完善，逐步形成的。早期具有辅存的存储系统并不是虚拟存储器，这是因为用户最多只允许使用主存容量，而实际上用户编程所用空间要比主存容量小的多。用户编程要想超出可使用的主存空间范围，就必须对程序进行对准分段。对于哪段存放在主存、哪段存放在辅存、何时从辅存调入主存、又何时从主存调出到辅存、存储空间如何分配、地址如何编写等一系列问题，都要求编程人员认真考虑并事先在程序中安排；在多道程序和多用户分时系统的运行

情况下，还往往会发生用户竞争存储空间的矛盾；随着计算机规模的扩大和复杂程度的增加，用户的负担日益加重。如何让用户从繁重的负担中解脱出来，不再去考虑主存空间大小、主存与辅存的差别等，用户编程可以随意使用存储空间，在用户心目中，计算机系统有一个大容量、高速度、可满足要求、使用方便的存储器，而没有主存、辅存之分，于是便产生了虚拟存储器的概念。所谓虚拟存储器是指主存—辅存层次，它以透明的方式给用户提供了一个比实际主存空间大得多的程序地址空间。

2、虚地址与实地址

用户编程时所用的地址称虚拟地址或逻辑地址，简称虚地址。虚地址的全部集合构成的地址空间称为虚拟地址空间或逻辑空间，实际的主存地址称为物理地址或实地址，实地址对应的空间为主存空间或物理空间，其容量为主存容量或实存容量。对于虚拟存储器，程序运行中每次访问主存时，都必须进行虚、实地址的变换。

3、虚拟存储器与 Cache 的比较

主存—辅存层次与 Cache—主存层次是两种不同的层次，有很多相似之处，他们所采用的地址变换、地址映射方式、替换策略等，在原理上看是相同的，都是基于程序局部性原理，而不同之处在于：Cache 用于弥补主存与 CPU 之间的速度差距，而虚拟存储器用来弥补主存与辅存之间的容量差距；Cache 与主存之间每次传送是以块为单位，块的大小只有几十个字节，而虚拟存储器的信息传送单位可以是页、段等，长度很大，几百至几百 K 字节；主存的访问速度比 Cache 慢 5~10 倍，而辅存比主存要慢上千倍，因此，CPU 访问未命中时，系统的相对性能损失有很大不同，在虚拟存储器中未命中时性能损失远远大于 Cache 中未命中性能损失。

4、虚拟存储器的管理方式

由于主存—辅存层次的基本信息传送单位可以采用段、页、段页几种不同方案，因此，虚拟存储器的存储管理方法也就相应存在段式管理、页式管理和段页式管理。

（1）段式管理

段式管理是把主存按段分配的存储管理方式。段是指逻辑结构相对独立的部分，例如子程序、数据表等。段作为独立的逻辑单位可以被其他程序段调用，形成规模较大的程序。因此，段作为基本信息单位在主存—辅存之间传送和定位是比较合理的。段的逻辑独立性易于编译、管理、修改和保护，也便于多道程序的共享；某些类型的段（堆栈、队列）具有动态可变长度，允许自由调度以便有效利用主存空间。但由于段的长度各不相同，段的起点和终点不定，给主存空间分配带来麻烦，而且容易在段间留下许多空余的零碎存储空间不好利用而造成浪费。各个段在主存的位置是由段表指示的，段表包括段起址、段长和控制位组成。

（2）页式管理

针对段式管理会在主存中各个段之间留下一些不好利用的空余零碎空间，有人提出了页式管理方式。所谓页式管理是指把主存的物理空间和辅存的逻辑空间按页（一定长度的区域）划分并进行管理。各个页在主存中的位置由页表指示，页表包括实页号和控制位组成。页作为主存—辅存之间基本信息传送单位，唯一可能造成浪费的是程序最后一页零头的页内空间。由于页不是逻辑上独立的实体，所以，处理、保护、共享都不及段式管理方便。

（3）段页式管理

段页式管理兼顾前两种管理方式的优点，是将程序按模块分段，段再分页，进入主存仍以页为基本信息传送单位，用段表和页表进行两级定位的管理方式。

4.4.2 页式虚拟存储器

在页式虚拟存储器中，设虚拟空间分成 m 个逻辑页，其逻辑页号为 0，1，2，…，m-1；主存空间也按同样大小分成 n 个物理页，其物理页号为 0，1，2，…，n-1。显然，m＞n，由于页的大小都为 2 的整数幂个字，所以，页的起点都落在低位字段为零的地址上。因此，

虚地址分为两个字段：高位字段为逻辑页号，低位字段为页内行地址。同样，实地址也分为两个字段：高位字段为物理页号，低位字段为页内行地址。由于页面大小相同，所以虚地址与实地址中页内行地址是相同的。

虚地址到实地址的变换是通过页表来实现的。在页表中，每一个逻辑页号都对应一个表目，表目内容包括该逻辑页所在主存的物理页号和一些控制位，用其中的物理页号与虚地址中的页内行地址拼接成实地址，据此来访问主存。页式虚拟存储器的地址变换如图 4-35 所示。

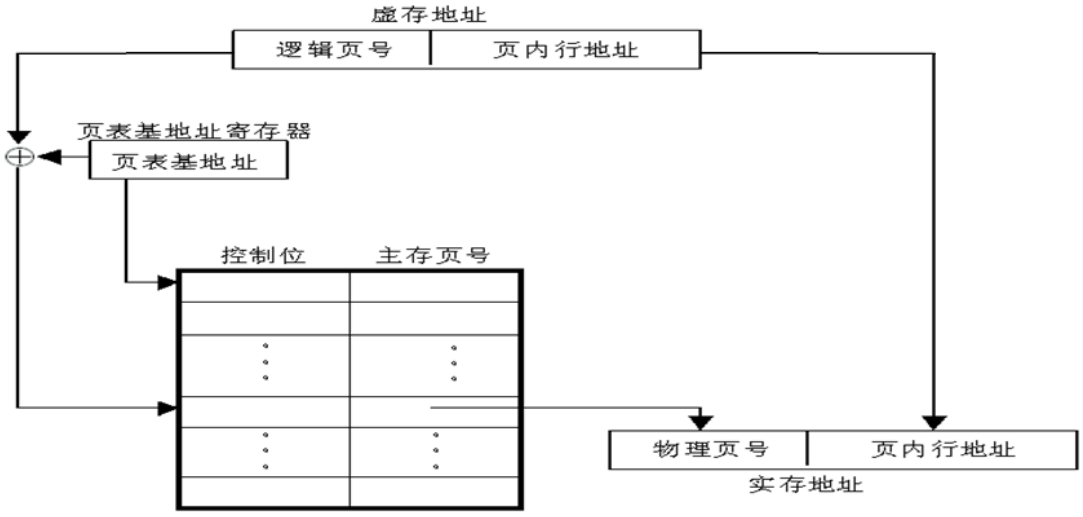


图 4-35　页式虚拟存储器的地址变换过程

通常页表中的一些控制位包括装入有效位、修改位、替换控制位等，其中如果装入有效位为 1，表示该逻辑页已在主存；如果装入有效位为 0，表示该逻辑页尚未调入主存，CPU 如果访问该页，就会页面失效中断，启动输入输出系统，将辅存中的逻辑页调入到主存；修改位用来指示主存页面中的内容是否被修改过，若修改位为 1，表示该物理页修改过，在该物理页被替换时必须写回主存；替换控制位是用来指出需替换的页。

假设页表已保存或已调入主存，在访问存储器时，首先要查页表，即使页表命中，也得先访问一次主存去查页表，形成实地址后再访问一次主存才能取出数据，相当于访问两次主存。如果页面失效，还要进行页面替换、页面修改，访问主存的次数将会更多。因此，把页表的最活跃部分存放在一个高速存储器中组成一个快表，这是减少时间开销的一种方法。一种由快表和慢表组合实现地址变换如图 4-36 所示。快表由硬件组成，它比页表小得多，仅仅是慢表的一个小小副本。查表时，由逻辑页号同时去查快表和慢表，当在快表中找到此逻辑页号时，就能很快地形成实地址，并使慢表的查找作废；如果在快表中找不到此逻辑页号，那就要访问一次主存去查慢表，形成实地址，同时将此逻辑页号和对应的物理页号送入快表，替换快表中的内容，这也要用到替换算法。

4.4.3 段式虚拟存储器

在段式虚拟存储器中，段是按程序的逻辑结构划分的，各个段的长度因程序而异，因此虚地址是由段号和段内地址组成。虚地址到实地址的变换是通过段表来实现的。在段表中，每一个段号都对应一个表目，表目内容包括段起址、段长和一些控制位，用其中段起址与虚地址中的段内地址相加成实地址，据此来访问主存。段式虚拟存储器的地址变换如图 4-37 所示。

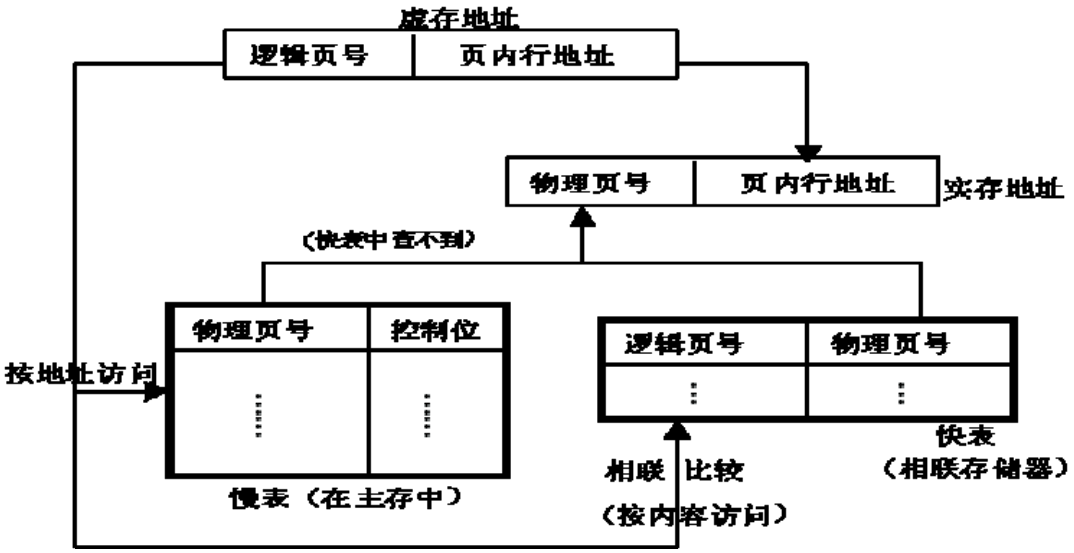


图 4-36 快表和慢表组合实现地址变换

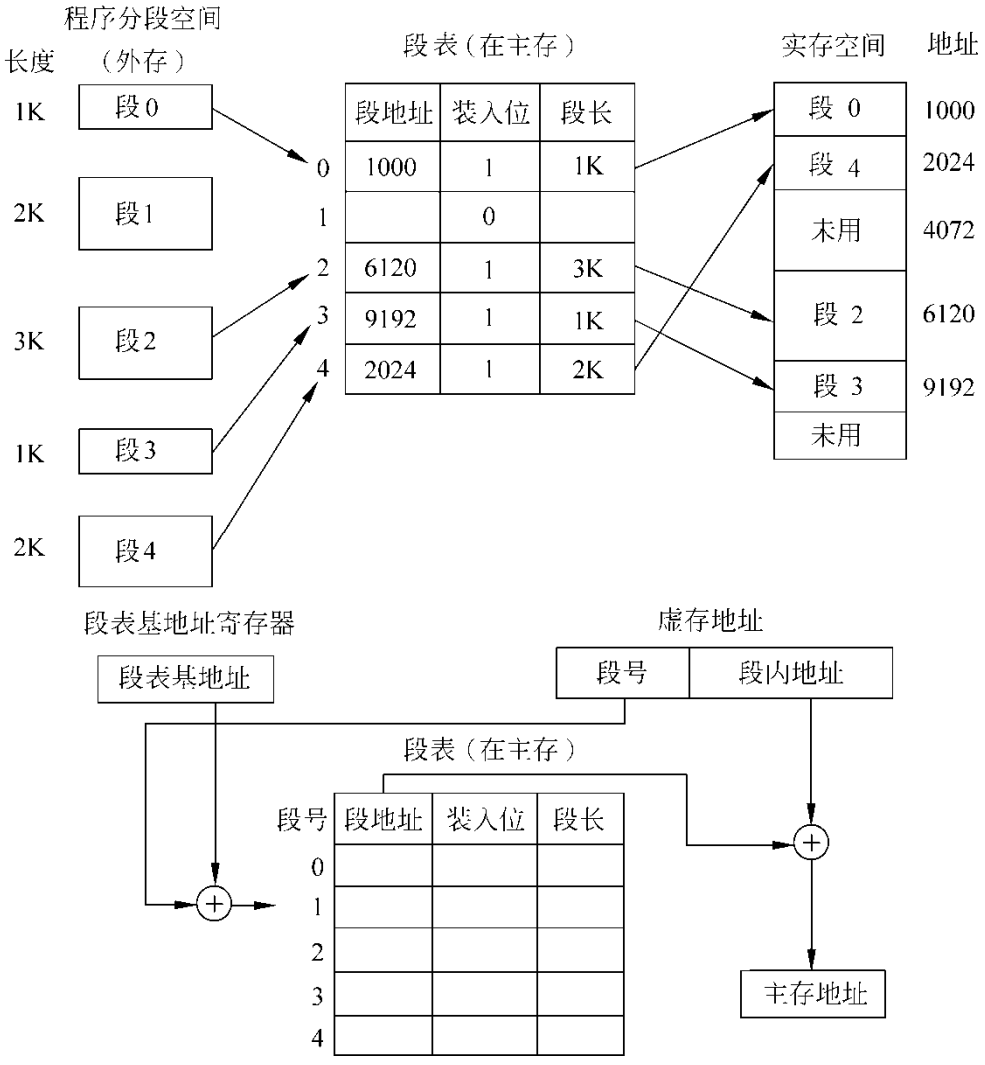


图 4-37 段式虚拟存储器地址变换过程

4.4.4 段页式虚拟存储器

段页式虚拟存储器是段式虚拟存储器和页式虚拟存储器的结合，是把程序按逻辑单位进

行分段，再把每个段分成固定大小的页。程序对主存的调入调出是按页面进行的，但它又可以按段实现共享和保护。因此，它兼备了段式虚拟存储器和页式虚拟存储器的优点。其缺点是地址变换过程中需要多次查表。在段页式虚拟存储器中，每道程序是通过一个段表和一组页表来进行定位的。段表中的每一个表目对应一个段，表目的内容包括该段的页表起始地址和该段的控制保护信息。由页表指明该段的各个页在主存中的位置以及是否装入、已修改等状态信息。

多道程序的每一道程序需要一个基号（用户标志号、程序标志号），由它指明该道程序的段表起始地址（存放在基址寄存器中），因此，虚地址由基号、段号、页号、页内地址组成。下面通过举例来说明虚地址到实地址的变换过程。

【例 4-6】假设有三道程序（用户标志号为 A、B、C），其基址寄存器内容分别为 S_A、S_B、S_C，虚地址到实地址的变换过程如图 4-38 所示。在主存中每道程序都有一张段表，A 程序有 4 段，C 程序有 3 段。每段应有一张页表，段表的每行就表示页表的起始位置，而页表的每行为相应的物理页号。

解：①根据基号 C，执行 S_C+段号（1）操作，得到段表相应的行地址，其内容为页表的起始地址 b

②执行 b+页号（2），得到物理页号的地址，其内容为物理页号（10）

③物理页号与页内地址拼接成物理地址

由此可见，段页式虚拟存储器由虚地址到实地址的变换至少需要查两次表，一次段表和一次页表。段、页表构成表层次，表层次并不只是段页式存在，页表也存在，这是因为整个页表是连续存储的，当一个页表的大小超过一个页面大小时，页表就可能分成几个表，分存于几个不连续的主存页面中，然后，将这些页表的起始地址又放入一个新的页表中，从而形成了二级表层次。一个大的程序可能需要多级页表层次。对于多级页表层次，在程序运行时，除了第一级页表需驻留在主存外，整个页表中只需有一部分是在主存中，大部分可存于外存，需要时再由第一级页表调入，从而减少每道程序占用的主存空间。

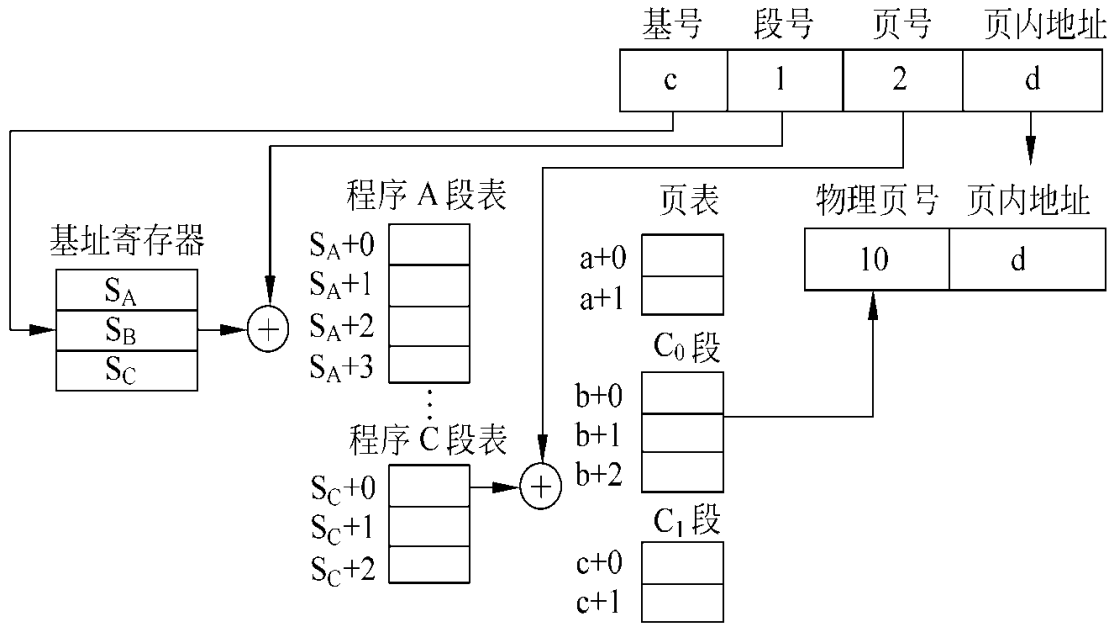


图 4-38　段页式虚拟存储器地址变换过程

4.4.5 替换算法

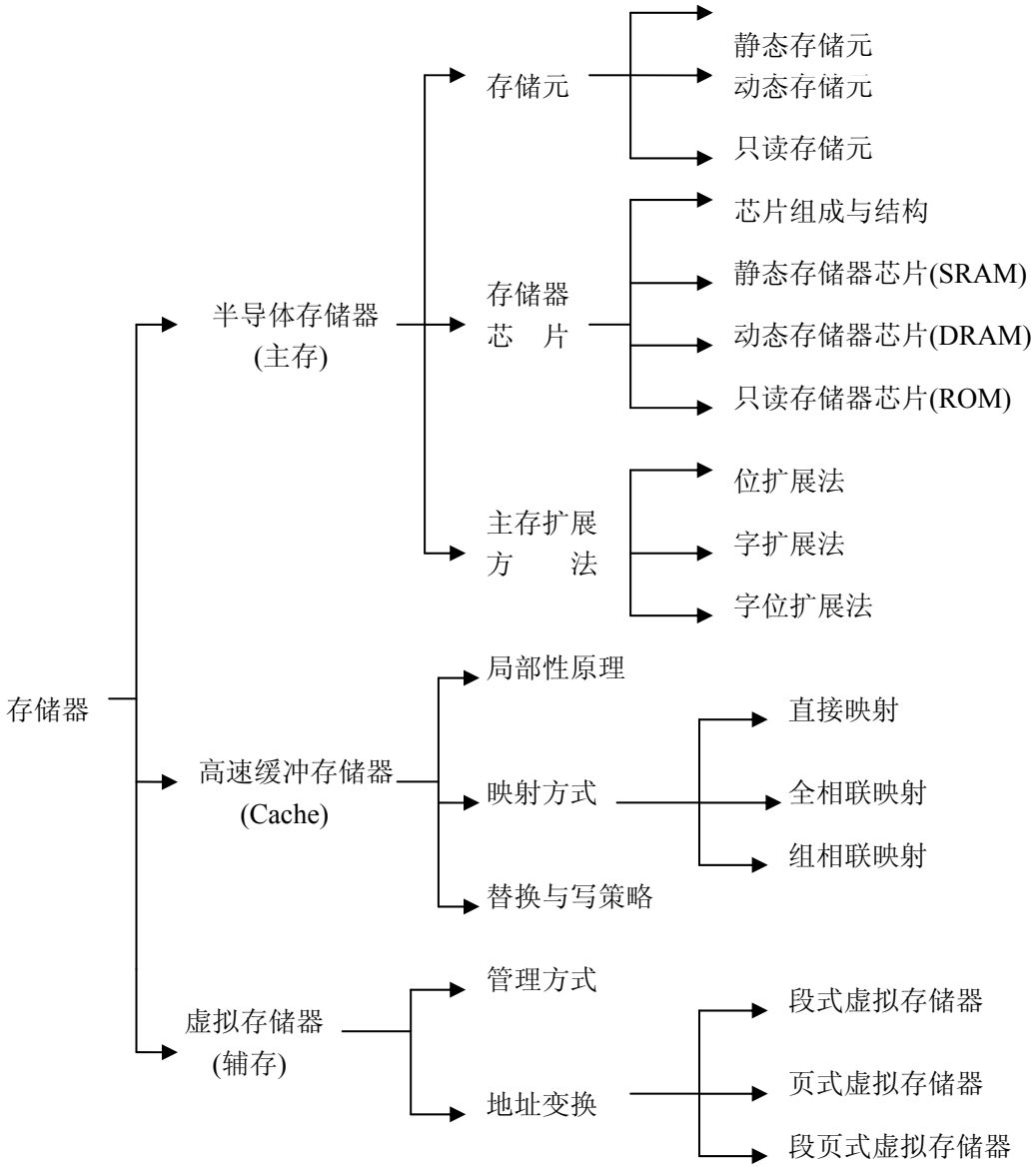
在 Cache 中我们讨论过替换算法，在虚拟存储器中仍然存在替换问题，当 CPU 访问主存而产生页面失效（缺页）时，若主存页面已全部被占满，必须采用替换算法将主存中的某页进行替换。虚拟存储器中的替换与 Cache 中的替换有很多相似之处，但也存在明显的不同，

主要表现在两个方面，一是缺页至少要涉及一次磁盘存取，以读取所缺的页，因此缺页使系统蒙受的损失比 Cache 未命中大得多；二是页面替换的选择余地大，属于一个进程的页面都可替换。为了换取更高的命中率，虚拟存储器中的替换策略一般采用 LRU 算法、LFU 算法、FIFO 算法。

对于将被替换出去的页面是否要进行某些处理？由于主存中的每一个页在辅存中都留有副本，假如该页调入主存后没有被修改过，那么就不必进行处理，否则，应该把该页重新写入辅存，以保证辅存中数据的正确性。为此，页表中专门设置有修改控制位，当该页刚调入主存时，此控制位为 0；当对该页进行写操作时，此控制位置 1。在该页被替换时，检查其修改控制位是否为 1，如果为 1，则先将该页从主存写入辅存，然后再从辅存接收新的一页；如果为 0，直接从辅存接收新的一页。

在虚拟存储器中，为了实现逻辑地址到物理地址的转换，并在页面失效时进行合理有效的管理，专门设置了由硬件实现的存储器管理部件 MMU。

关　联



习　题

4.1 判断题

（1）外存比内存的存储容量大、存取速度快。（　）

（2）动态 RAM 和静态 RAM 都是易失性半导体存储器。（　）

（3）Cache 是内存的一部分，它可由指令直接访问。（　）

（4）引入虚拟存储系统的目的，是为了加快外存的存取速度。（　）

（5）多体交叉存储器主要是为了解决扩充容量问题。（　）

（6）数据引脚和地址引脚越多，芯片的容量越大。（　）

（7）存储芯片的价格取决于芯片的容量和速度。（　）

（8）要访问 DRAM，应首先给出$\overline{\text{RAS}}$地址，之后再给出$\overline{\text{CAS}}$地址。（　）

4.2 选择题

（1）内存储器用来存放（　）。

A、 程序　B、 数据　C、 微程序　D、 程序和数据

（2）某一静态 RAM 芯片，其容量为 64K×1 位，则其地址线有（　）。

A、 64 条　B、 64000 条　C、 16 条　D、 64436 条

（3）需要定期刷新的存储芯片是（　）。

A、EPROM　　B、DRAM　　C、SRAM　　D、EEPROM

（4）（　）存储芯片是易失性的。

A、 SRAM　　B、 UV-EPROM　　C、 NV-RAM　　D、 EEPROM

（5） 有$\overline{\text{RAS}}$和$\overline{\text{CAS}}$引脚的存储芯片是（　）。

A、 EPROM　　B、 DRAM　　C、 SRAM　　D、三者都不是

（6）下面叙述不正确的是（　）。

A、半导体随机存储器可随时存取信息，掉电后信息丢失

B、在访问随机存储器时，访问时间与单元的物理位置无关

C、内存储器中存储的信息均是不可改变的

D、随机存储器和只读存储器可以统一编址

（7）动态 RAM 与静态 RAM 相比，其优点是（　）。

A、动态 RAM 的存储速度快

B、动态 RAM 不易丢失数据

C、在工艺上，比静态 RAM 的存储密度高

D、控制比静态 RAM 简单

（8）某 512×8 位 RAM 芯片采用一位读/写线控制读写，该芯片的引脚至少有（　）。

A、17 条　　B、19 条　　C、21 条　　D、22 条

（9）下列存储器中，存取速度最慢的是（　）。

A、 半导体存储器　　B、 光盘存储器

C、 磁带存储器　　D、 硬盘存储器

（10）在主存储器和 CPU 之间增加 Cache 的主要目的是（　）。

A、 降低整机系统的成本　　B、 解决 CPU 和主存之间的速度匹配问题

C、 扩大主存容量　　D、 替代 CPU 中的寄存器工作

（11）采用虚拟存储器的主要目的是（　）。

A、 提高主存储器的存取速度

B、 扩大主存储器的存储空间，并能进行自动管理和调度

C、 提高外存储器的存取速度

D、 扩大外存储器的存储空间

（12）某计算机的字长是 32 位，其存储容量是 32MB，若按字编址，它的寻址范围是（ ）。

A、 0～8MB B、 0～32M C、 0～32MB D、 0～8M

（13）在 Cache 的地址映射中，若主存储器中的任意一块均可映射到 Cache 内的任意一块的位置上，则这种方法称为（ ）。

A、 直接映射 B、 组相联映射

C、 全相联映射 D、 混合映射

4.3 填空题

（1)采用 4K×4 位的静态 RAM 存储芯片扩展 32KB 的存储模块，需要存储芯片数为_____片。

（2）存储器的技术指标有_____、_____、_____和_____等。

（3）CPU 能直接访问_____和_____，但不能直接访问_____。

（4）Cache 存储器的主要作用是解决_____。

（5）存储器的取数时间是衡量主存_____的重要指标，它是从_____到_____的时间。

（6）某存储器数据总线宽度为 32 位，存取周期为 250ns,则其带宽是_____

（7）Cache 的地址映像方式有_____、_____和_____三种。

（8）虚拟存储器处于_____层次，它给用户提供了一个比实际_____空间大得多的_____空间。

4.4 静态 MOS 存储器与动态 MOS 存储器存储信息的原理有何不同？为什么动态 MOS 存储器需要刷新？一般有哪几种刷新方式？

4.5 某一动态 RAM 芯片(64K×1 位)，采用地址复用技术，则除了电源和地引脚外，该芯片还有哪些引脚？各为多少位？

4.6 在页模式 DRAM 中，“打开一页”指什么？在打开一页的操作中，信号 $\overline{RAS}$ 和 $\overline{CAS}$ 的作用是什么？

4.7 DRAM 的 t_{RC} 和 t_{RAC} 指什么？两者有何不同？

4.8 假设某存储器地址为 22 位，存储器字长为 16 位，试问：

（1）该存储器能存储多少字节信息？

（2）若用 64K×4 位的 DRAM 芯片组成该存储器，则需多少片芯片？

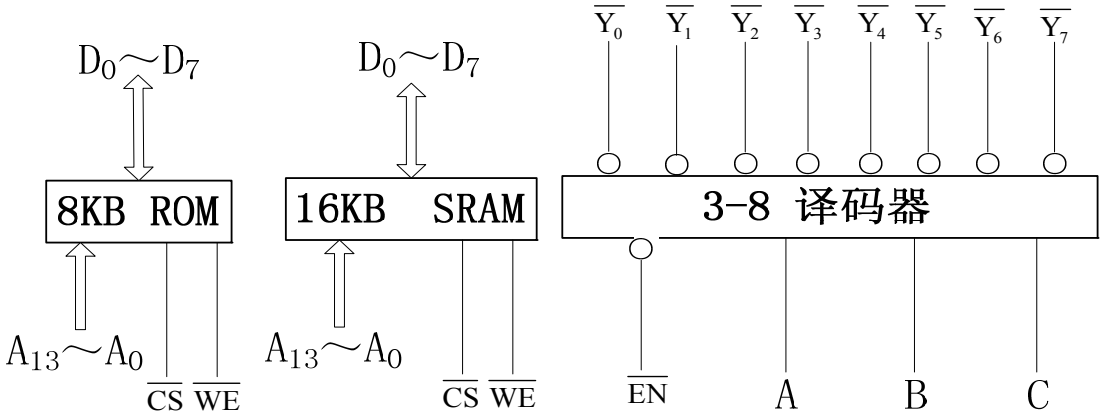
（3）在该存储器的 22 位地址中，多少位用于片间寻址？多少位用于片内寻址？

4.9 某 8 位计算机采用单总线结构，地址总线 17 根（$A_{16}\cdots A_0$,），数据总线 8 根，双向（$D_7\cdots D_0$），控制信号 $R/\overline{W}$（高电平为读，低电平为写）。已知该机存储器地址空间从 0 连续编址，其地址空间分配如下：最低 8*KB* 为系统程序区，由 ROM 芯片组成；紧接着 40*KB* 为备用区，暂不连接芯片；而后 78*KB* 为用户程序和数据空间，用静态 RAM 芯片组成；最后 2*KB* 用于 I/O 设备（与主存统一编址）。现有芯片如题图 4.9 所示。

SRAM：16K×8 位，其中 $\overline{CS}$ 为片选信号，低电平有效；$\overline{WE}$ 为写控制信号，低电平写，高电平读。

ROM：8K×8 位，其中 $\overline{CS}$ 为片选信号，低电平有效；$\overline{OE}$ 为读出控制，低电平读出有效。

译码器：3-8 译码器，输出低电平有效；$\overline{EN}$ 为使能信号，低电平时译码器功能有效。其他与、或等逻辑门电路自选。



题图　4.9

（1）请问该主存需多少 SRAM 芯片？

（2）试画出主存芯片与 CPU 的连接逻辑图。

（3）写出各芯片地址分配表。

4.10 已知某 8 位机的主存采用 4K×4 位的 SRAM 芯片构成该机所允许的最大主存空间，并选用模块板结构形式，该机地址总线为 18 位，问：

（1）若每个模块板为 32K×8 位，共需几个模块板？

（2）每个模块板内共有多少块 4K×4 位的 RAM 芯片？画出一个模块板内各芯片连接的逻辑框图。

（3）该主存共需要多少 4K×4 位的 RAM 芯片？CPU 如何选择各个模块板？

4.11 64K×1 位 DRAM 芯片通常制成两个独立的 128×256 阵列。若存储器的读/写周期为 0.5 μs，则对集中式刷新而言，其“死区”时间是多少？如果是一个 256K×1 位的 DRAM 芯片，希望能与上述 64K×1 位 DRAM 芯片有相同的刷新延时，则存储阵列应如何安排？

4.12 访问主存的地址是 20 位（$A_{19}\cdots A_0$），数据总线为 8 位，分别计算下列各种情况下标识 cache 和数据 cache 的大小，并画出对应的结构框图。

（1）全相连映像，内容 cache 大小为 1024

（2）直接映像，$A_{15}\cdots A_0$ 作为索引

（3）2 路组相连映像，$A_{14}\cdots A_0$ 作为索引

（4）4 路组相连映像，$A_{13}\cdots A_0$ 作为索引

（5）8 路组相连映像，$A_{12}\cdots A_0$ 作为索引

第 5 章　指令系统

【内容摘要】

指令系统是指计算机所能执行的全部指令的集合，也称为指令集，它不仅是计算机硬件设计的依据，还是软件设计的基础，因此，一台计算机指令系统的优劣直接影响着计算机系统的性能和功能。本章将围绕机器指令探讨指令的格式和长度、操作码编码技术、地址码寻址技术，结合指令的分类和功能，剖析复杂指令系统和精简指令系统。

【学习要点】

- 指令的基本格式
- 固定长度的操作码编码和可变长度的操作码编码
- 指令的寻址方式和操作数的寻址方式
- 复杂指令系统和精简指令系统

5.1　指令系统与性能

5.1.1 指令与指令系统

指令是指示计算机去执行某种操作的命令。如果命令是二进制代码形式，则相应的指令为机器指令，机器指令是计算机能直接识别并执行的指令，而用任何其他形式的指令或语言所编写的程序必须通过“翻译”或“编译”，编译成机器语言程序，才能在计算机中运行。指令系统则是指计算机所有机器指令的集合，也称指令集（Instruction Set）。它是面向机器的，不同的计算机具有不同的指令系统，反过来讲，不同的指令系统决定了计算机不同的性能和功能。随着计算机的发展和应用领域的不断扩大，指令系统越来越丰富，机器指令的数量成倍增长，多达成百上千条指令，如此庞大的指令系统我们称之为复杂指令系统（CIS），具有复杂指令系统的计算机称为复杂指令系统计算机（CISC）。CISC 就是采用复杂的指令系统，来达到增强计算机的功能、提高速度的目的。经过对 CISC 指令使用频率的测试分析，发现只有占指令系统 20%的指令是常用的，剩余 80%的指令在程序中出现的概率大约 20%。测试结果表明：花费巨大代价所增加的复杂指令只有 20%左右的使用率，这将造成计算机硬件资源的巨大浪费。为此，人们开始考虑能否用最常用的 20%左右的简单指令来组合实现不常用的 80 %的指令，于是，出现了精简指令系统（RIS）和精简指令系统计算机（RISC）。精简指令系统（RIS）是对复杂指令系统（CIS）进行简化和优化，使机器结构简化的同时，有效地提高机器的性能、速度和性能价格比。

5.1.2 指令系统的性能

指令系统的性能将决定计算机的基本功能，因此，指令系统的设计是计算机系统设计的一个核心问题，它不仅与计算机的硬件紧密相关，而且直接影响到用户的使用。一个完善的指令系统应满足以下几个方面的要求：

第一是完备性：要求机器的指令系统丰富，且功能齐全。

第二是有效性：有效性是指用指令系统中的指令所编写的程序能被高效率地运行。高效率表现在程序所占存储空间小、执行速度快。

第三是规整性：规整性包括指令系统的对称性、均匀性、一致性。对称性是指寄存器与存储器的对称，能对寄存器操作的指令同样能对存储器操作。均匀性是指各种数据类型适用于一种操作性质的指令。一致性是指指令长度和数据长度通常是字节的整数倍。

第四是兼容性：兼容性是指“向上兼容”，即抵挡机上运行的软件可以在高档机上运行。

第五是可扩展性：可扩展性是指保留一定余量的操作码空间，便于扩展指令使用。

5.2 机器指令

5.2.1 机器指令的格式

根据指令的概念，一条机器指令形式上是一串二进制代码，我们通常把表示机器指令的这一串二进制代码称为指令字。作为一个指令字，不仅要表示出指令所进行的某种具体操作，还要表示出指令对谁操作以及操作结果的去向，即指令的操作对象。因此，任何一条机器指令都有两部分组成：一部分是操作码，另一部分是地址码。其格式如图 5-1 所示。

操作码（OP）	地址码（A）

图 5-1　机器指令的基本格式

操作码 OP：是指令所进行的具体操作的编码。不同的指令相应的操作和功能不同，则需要用不同的操作码来表示。因此，操作码的长度与机器所能执行操作的多少有关。

地址码 A：是指令操作对象的地址编码。它是一个广义的概念，它不仅可以是操作对象的存储地址，还可以是操作对象本身。因此，地址码的长度与存储器的容量、寄存器的多少、机器的长度有关。

5.2.2 指令字的长度

指令字的长度是指一个指令字所包含的二进制代码的位数。结合机器指令的格式，指令字的长度取决与操作码的长度、地址码的长度和地址码的个数，因此，影响指令字长度的因素有多种，不同指令系统的指令长度各不相同，同一个指令系统的不同指令其长度也不尽相同，但为了便于存储，指令字的长度与机器字长之间具有一定的匹配关系，通常指令长度设计为字节的整数倍。例如 8086/8088CPU 的指令系统，最短的指令长度为 1 个字节，最长的指令长度为 6 个字节；再如奔腾系列机的指令系统，最短的指令长度为 1 个字节，最长的指令长度为 12 个字节。由于指令的长度不尽相同，我们不仅可以将指令按其长度进行分类，还可以将指令系统进行分类。如果指令的长度等于机器的字长，这样的指令称为单字长指令；如果指令的长度等于两倍的机器字长，这样的指令称为双字长指令；还可以分为更多倍字长的指令以及半字长指令等。如果指令系统中各种指令字的长度均固定，则称为定长指令字结构；如果指令系统中各种指令字的长度随指令功能而异，则称为可变长指令字结构。

定长指令字结构：结构简单，指令的译码时间短，有利于硬件控制系统的设计，多用于机器字长较长的大、中型计算机和精简指令系统计算机。但指令字的长度普遍较长，冗余现象严重，不利于指令扩展。

可变长指令字结构：结构灵活，能充分利用指令中的每一位，信息冗余少，平均指令长度短，易于指令的扩展。但指令格式不规整，硬件控制系统复杂。

5.2.3 机器指令的分类

机器指令不仅可以按其长度划分为单字指令、双字指令和多字指令，还可以按指令中地址码的多少将机器指令划分为：零地址指令、单地址指令、双地址指令和多地址指令。

1、零地址指令

零地址指令的指令字中只有操作码，而没有地址码。其格式为：

OP

零地址指令有以下两种情况：

第一是指令不需要操作对象，也就不需要地址码，例如停机指令、空操作指令等。

第二是指令需要一个操作对象，但这个操作对象可以隐含，从而指令的地址码也隐含，例如对堆栈、累加器操作的指令。指令功能为：OP（AC）→AC。

2、单地址指令

单地址指令的指令字中只有一个地址码 A，也称为一地址指令。其格式为：

OP	A

单地址指令有以下两种情况：

第一是指令有一个操作对象，并且这个操作对象不能隐含，必须用一个地址码表示，例如加 1 指令、减 1 指令等。指令功能为：OP（A）→A。

第二是指令有两个操作对象，其中一个操作对象隐含，通常指累加器（AC），另一个用一个地址码表示，例如加法指令、减法指令等。指令功能为：（AC）OP（A）→AC。

3、**双地址指令**

双地址指令的指令字中有二个地址码 A_1 和 A_2，也称为二地址指令。其格式为：

OP	A_1	A_2

双地址指令的指令功能为：（A_1）OP（A_2）→A_1 或（A_1）OP（A_2）→A_2

双地址指令可以分为三种指令类型：

（1）S—S 型指令

即存储器—存储器型指令，是指指令中两个操作对象都是存储器，用地址码 A_1 和 A_2 进行表示， A_1、A_2 为主存地址。

（2）R—R 型指令

即寄存器—寄存器型指令，是指指令中两个操作对象都是寄存器，用地址码 R_1 和 R_2 进行表示， R_1、R_2 为寄存器地址。

（3）R—S 型指令

即寄存器—存储器型指令，是指指令中两个操作对象一个是寄存器，一个是存储器，用地址码 R 和 A 进行表示， R 为寄存器地址，A 为主存地址。

在以上三种指令类型中，R—R 型指令在被执行过程中不需要访问存储器，执行速度最快，因此是双地址指令中最常用的一种指令格式，尤其在 RISC 中，所有运算指令均为 R—R 型指令。

4、**三地址指令**

三地址指令的指令字中有三个地址码 A_1、A_2 和 A_3，也称为多地址指令。其格式为：

OP	A_1	A_2	A_3

三地址指令的指令功能为：（A_1）OP（A_2）→A_3

三地址指令的指令字较长，多用于字长较长的大、中型计算机中，小型机和微型机很少使用。

5.3 操作码的编码方法

操作码是机器指令的重要组成部分，是对指令所执行操作的编码，指令不同其操作码也不同，具体表现在：操作码长度相同，操作码不同或操作码长度不同。所以，按指令字中操作码的长度是否相同或固定，可以将操作码的编码分为固定长度的定长编码和可变长度的变长编码两种。

5.3.1 定长编码

定长编码是最简单的操作码编码方法。由于操作码的长度和位置在指令字中是固定的，有利于简化指令结构和减少指令的译码时间，适用于字长较长的大、中型计算机。例如 IBM370 机，其字长是 32 位，指令分为半字长、单字长、1.5 倍字长指令类型，无论哪一种类型的指令，其操作码的长度和位置都是固定的，由于操作码的长度固定为 8 位，指令系统所能允许的指令条数最多为 2^8=256，实际上 IBM370 机只有 183 条指令，剩余的操作码编码为非法的操作码。如果操作码的编码采用定长编码，指令字中的操作码长度取决于指令系统中的全部指令条数。例如若指令系统中有 m 条指令，则操作码的长度 n 应满足式(5-1)

$$n \geqslant \log_2^m \qquad (5\text{-}1)$$

5.3.2 变长编码

变长编码是指指令字中操作码的长度和位置不固定，随指令的不同而存在差异。采用变长编码方法，可以有效地压缩操作码的平均长度，在小型机和微机中被广泛采用。如小型机PDP-11，机器字长为 16 位，指令分为单字长、双字长、三字长等指令类型，操作码占 5~6 位，且遍及整个指令字长度范围。但由于操作码长度和位置的不固定，增加了指令译码和分析的难度，使硬件设计复杂化。

1、操作码扩展技术

在不增加指令字长度的情况下，通过采用操作码扩展技术，使操作码的长度随地址码的减少而增加，这样一来，对于地址码个数不同的指令，可以具有不同长度的操作码，从而可以充分利用指令字的地址码，使有限的指令字长度可以表示更多的指令。

设机器指令字长度为 16 位。其中操作码为 4 位，地址码为 4 位。指令格式为：

OP	A_1	A_2	A_3

如果采用定长编码，4 位操作码只能表示 16 条三地址指令。如果指令系统中除三地址指令外，还有二地址、一地址和零地址指令，且要求有 15 条三地址指令、15 条二地址指令、15 条一地址指令和 16 条零地址指令，则采用定长编码方法是无法满足上述要求的，这就需要采用变长编码方法，操作码的长度随地址码的减少而增加，具体操作码扩展如下：

三地址指令操作码的编码
0000 ×××× ×××× ××××
0001 ×××× ×××× ××××
⋮
1110 ×××× ×××× ××××
} 15 条三地址指令

二地址指令操作码的编码
1111 0000 ×××× ××××
1111 0001 ×××× ××××
⋮
1111 1110 ×××× ××××
} 15 条二地址指令

一地址指令操作码的编码
1111 1111 0000 ××××
1111 1111 0001 ××××
⋮
1111 1111 1110 ××××
} 15 条一地址指令

零地址指令操作码的编码
1111 1111 1111 0000
1111 1111 1111 0001
⋮
1111 1111 1111 1111
} 16 条零地址指令

【例 5-1】某机器指令字长度为 16 位。其中地址码长度为 4 位。如果指令系统中三地址指令有 11 条、二地址指令有 72 条、零地址指令有 64 条。问最多还能设计多少条一地址指令？

解：对于三地址指令：地址码共占 12 位，指令字中还剩 4 位用于操作码，其编码最多为 2^4=16 个，由于实际只有 11 条三地址指令，所以还剩下 16-11=4 个编码，可用于二地址指令。

对于二地址指令：地址码共占 8 位，指令字中还剩 8 位用于操作码，扣除三地址指令用过的操作码，其编码最多为 5×2^4=80 个，由于实际只有 72 条二地址指令，所以还剩下 80-72=8 个编码，可用于一地址指令。

对于一地址指令：地址码共占 4 位，指令字中还剩 12 位用于操作码，扣除三地址指令

和二地址指令用过的操作码，其编码最多为 $8\times2^4=128$ 个，如果这 128 个编码都表示一地址指令，那么指令系统中就不存在零地址指令，实际上存在有零地址指令，所以，这 128 个编码并不都是一地址指令。究竟一地址指令还能设计多少条，这就取决于实际存在的零地址指令条数。由于系统要求有 64 条零地址指令，而 4 位操作码只能最多表示 16 条指令，所以，需要一地址指令提供 64/16=4 个操作码编码用于零地址指令，以保证零地址指令有 64 条。因此，指令系统中一地址指令最多还能设计 128-4=124 条指令。

2、Huffman 编码法

Huffman 编码法是根据各种指令使用的频率不同，采用优化技术，将使用频率最高的指令用最短的编码表示，而对于使用频率较低的指令用较长的编码表示，从而缩短操作码的平均长度。采用 Huffman 编码法所得到的操作码的平均长度为：

$$\overline{H}=\sum_{i=1}^{n} P_i\times L_i$$

其中 P_i 是第 i 条指令使用的频率，L_i 是第 i 条指令操作码的长度，n 是指令总数。

下面以 10 条指令的机器为例来说明 Huffman 编码过程。

已知 10 条指令的使用频率如表 5-1 所示。

表 5-1　　指令的使用频率

指令	使用频率（p_i）
I_0	**0.30**
I_1	**0.20**
I_2	**0.16**
I_3	**0.09**
I_4	**0.08**
I_5	**0.07**
I_6	**0.04**
I_7	**0.03**
I_8	**0.02**
I_9	**0.01**

（1）Huffman 树

根据指令的使用频率，画出 Huffman 树，具体步骤：

1）按频率大小进行排列，频率相同的可任意排列。

2）把出现频率最小的两项进行合并，将其频率相加，再把相加后的频率进行重新排序。

3）重复步骤 2），直至只剩下最后两个频率，再将其合并，便形成 Huffman 树。如图 5-2 所示。

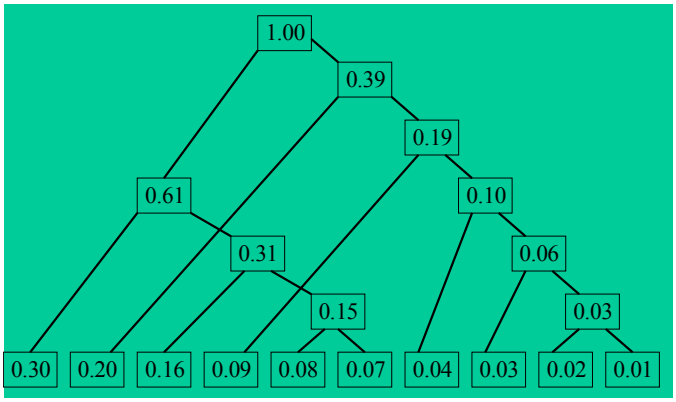


图 5-2　　Huffman 树

（2）Huffman 编码过程

根据 Huffman 树，从根节点（最后两个频率形成的节点）开始编码，左分支编码为“1”，右分支编码为“0”；各个子节点也同根节点一样，左分支编码为“1”，右分支编码为“0”，直到末节点为止。各条指令的编码便是从根节点开始到末节点结束的一个二进制代码。如表 5-2 所示。

表 5-2　　Huffman 编码结果

指令	Huffman 编码法	长度	2-5-6 扩展编码法	长度
I_0	**11**	2	**11**	2
I_1	**01**	2	**01**	2
I_2	**101**	3	**1010**	4
I_3	**001**	3	**0010**	4
I_4	**1001**	4	**1001**	4
I_5	**1000**	4	**1000**	4
I_6	**0001**	4	**0001**	4
I_7	**00001**	5	**000010**	6
I_8	**000001**	6	**000001**	6
I_9	**000000**	6	**000000**	6

则采用 Huffman 编码法所得到的操作码平均长度为：

$$\overline{H}=\sum_{i=1}^{n} P_i \times L_i = (0.30+0.20)\times 2 + (0.16+0.09+0.08+0.07+0.04)\times 4 + (0.03+0.02+0.01)\times 6$$

$$= 1 + 0.75 + 0.76 + 0.15 + 0.18$$

$$= 2.85$$

若采用定长编码，由于指令有 10 条，所以操作码的长度为 4。可见，采用 Huffman 编码法更有效地缩短了操作码的平均长度。

3、**扩展编码法**

采用 Huffman 编码法能更有效地缩短操作码的平均长度，但这种编码方法所形成的操作码很不规整，将会给译码造成极大的困难，不利于软件的编译。于是，采取了一种折中的方案，将 Huffman 编码法与定长编码法进行有效结合，在缩短操作码平均长度的基础上，使操作码更加规整。这种编码方法称为扩展编码法。扩展编码结果如表 5-2 所示。采用扩展编码法所得到的操作码平均长度：

$$\overline{H}=\sum_{i=1}^{n} P_i \times L_i = \mathbf{(0.30+0.20)\times 2+(0.16+0.09)\times 3+(0.08+0.07+0.04)\times 4+0.03\times 5+(0.02+0.01)\times 6}$$

$$\mathbf{= 1 + 1.76 + 0.36}$$

$$\mathbf{= 3.12}$$

5.4　地址码的寻址方式

寻址方式是指在指令被执行的过程中，根据指令字中的地址码去寻找操作对象的方式。如果地址码所表示的操作对象是数据信息，这样的操作对象称为操作数，对应的寻址方式为操作数寻址方式；如果地址码所表示的操作对象是指令信息，相应的寻址方式为指令寻址方式。

5.4.1 指令寻址方式

指令寻址方式又可分为顺序寻址方式和跳转寻址方式。如图 5-3 所示。

1、顺序寻址方式

顺序寻址方式是采用 PC 增量的方式形成下一条指令地址。具体寻址过程如图 5-3（a）所示，PC 称为程序计数器，是用来跟踪程序的执行并指向下一条将要被执行的指令。由于程序在内存中是连续存放的，当程序顺序执行时，PC 的内容加上一定的增量，便形成下一条将要被执行指令的地址，至于增量是多少，取决于指令所占存储单元的个数。

2、跳转寻址方式

跳转寻址方式是指当程序发生转移时，下一条将要执行的指令地址不再是 PC 的内容，而是根据转移指令字中的地址码 A 去形成下一条将要执行的指令地址。具体寻址过程如图 5-3（b）所示，如果是地址码直接形成 PC，记作 A→PC，这种跳转寻址方式也称为绝对跳转；如果是当前 PC 的内容（现行 PC 加上增量）加上地址码形成 PC，记作（PC）+A→PC，这种跳转寻址方式也称为相对跳转。

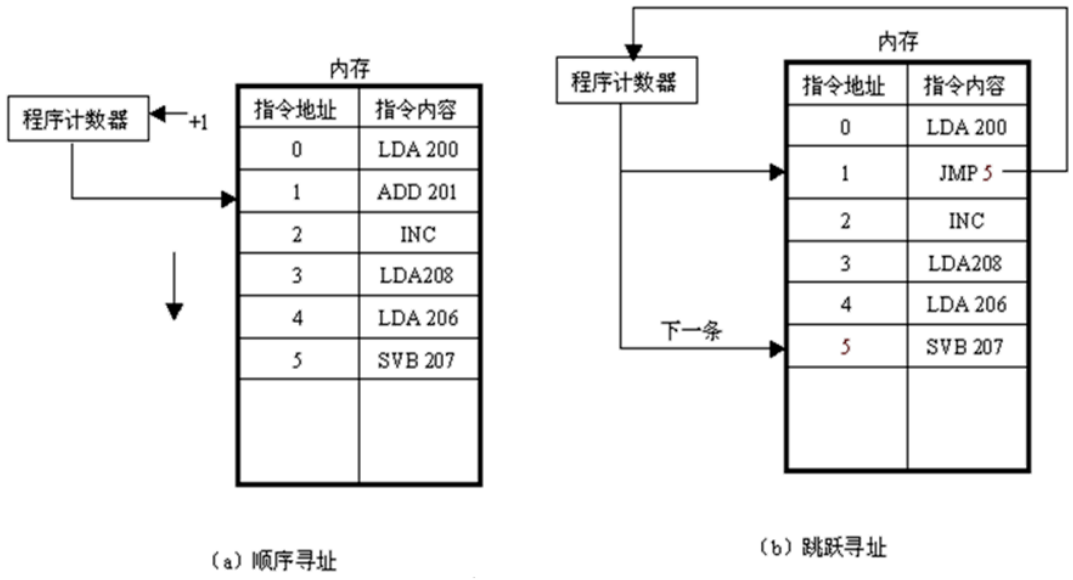


图 5-3 指令的寻址方式

5.4.2 操作数寻址方式

操作数的存放不象指令的存放有规律，它既可以存放在寄存器、存储器和外设中，也可以存放在指令中，因此表示操作数的地址码尽管形式上都是二进制代码，但所表示的含义各不相同，有时代表的是地址，有时代表的是数据，往往比较复杂，一般讨论寻址方式时，主要是讨论操作数寻址方式。下面以一地址指令为例，介绍几种最常用的基本寻址方式。一地址指令格式如图 5-4 所示。

OP	寻址方式 MOD	形式地址 A

图 5-4 一地址指令格式

由于操作数寻址方式种类较多，所以在指令字中设置一个寻址方式 MOD 字段，用来指明具体的寻址方式。当操作数存放在存储器中时，指令字中的地址码并不是内存的实际地址，因此我们把它称为形式地址。形式地址必须转换成内存的实际地址，然后才能访问存储器。内存的实际地址也称为有效地址（EA）。所以，EA 是 A 的一个函数，记为：EA=*f*（A）。函数*f*不同，其寻址方式也不同。

1、立即寻址

立即寻址方式是指指令字中的地址码本身就是操作数，即操作数是 D，D 也称为立即数。

OP	立即寻址	D

立即寻址的优点：由于从存储器中取出指令的同时，操作数也被取出了，因为操作数是

在指令字中，所以在执行指令时，不必再次访问存储器，从而提高了指令的执行速度。但由于指令字长度的限制，D 的位数限制了立即数的大小。

2、直接寻址

直接寻址方式是指指令字中的地址码 D（形式地址）就是操作数的有效地址，即 EA=D。如图 5-5 所示。

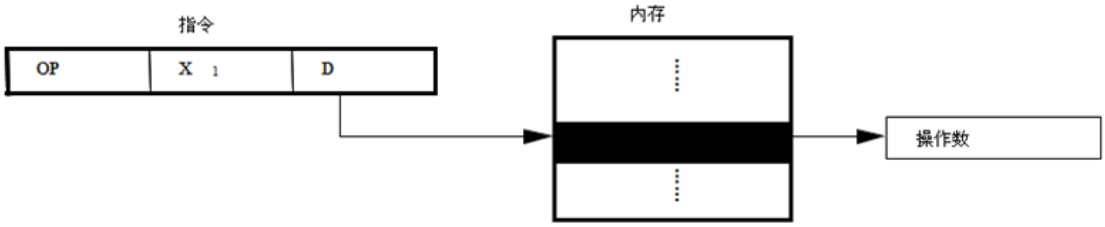


图 5-5 直接寻址方式

直接寻址简单直观，便于硬件实现。但由于指令字长度的限制，形式地址 D 限制了指令的寻址范围，随着内存容量不断扩大，要想寻址整个内存空间，势必造成指令字长度的增加。另外在编程时，如果使用了直接寻址指令，一旦操作数地址发生变化，就必须修改指令中的 D，给编程带来不便。

3、间接寻址

间接寻址方式是指指令字中的地址码 D（形式地址）不是操作数的有效地址，而是操作数的有效地址的有效地址，即 EA=（D）。如图 5-6 所示。

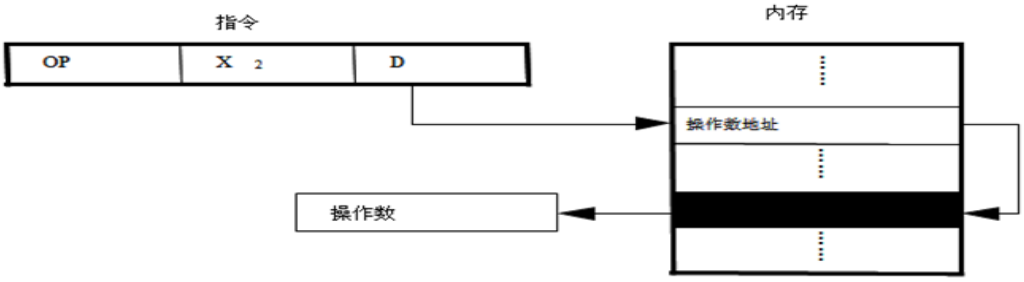


图 5-6 间接寻址方式

与直接寻址相比，间接寻址比直接寻址灵活，使用较短的地址码可以访问较大的存储空间。另外有利于编程，如果使用了间接寻址指令，当操作数地址发生变化时，可不必修改指令，只需修改存放有效地址的单元内容即可。其缺点比较明显，指令的执行时间较长，因为需要多次访问存储器。

4、寄存器直接寻址

寄存器直接寻址方式也称寄存器寻址方式，是指指令字中的地址码 R（形式地址）是某一寄存器的编号，其寄存器的内容为指令所需的操作数，即 EA=（R）。如图 5-7 所示。

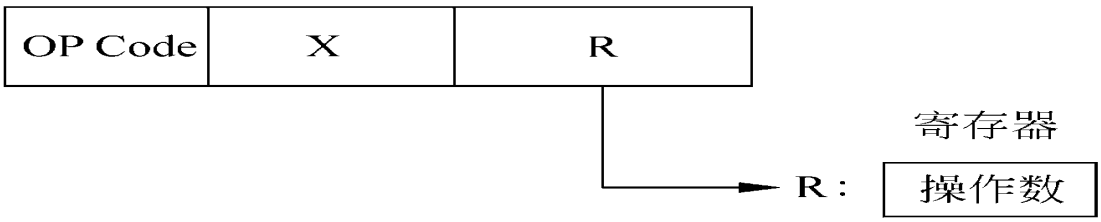


图 5-7 寄存器直接寻址方式

由于寄存器寻址方式的操作数位于寄存器中，采用寄存器寻址方式的指令在被执行时，无需访问内存，从而减少了指令的执行时间；另外，由于内部寄存器数量有限，指令字中的

地址码较短，有效地缩短了指令字长度。因此，寄存器寻址方式在计算机中得到了广泛的应用。

5、寄存器间接寻址

寄存器间接寻址方式是指指令字中的地址码 A（形式地址）是某一寄存器的编号，其寄存器的内容不是指令所需的操作数，而是指令所需操作数的有效地址，即 EA=（（R））。如图 5-8 所示。

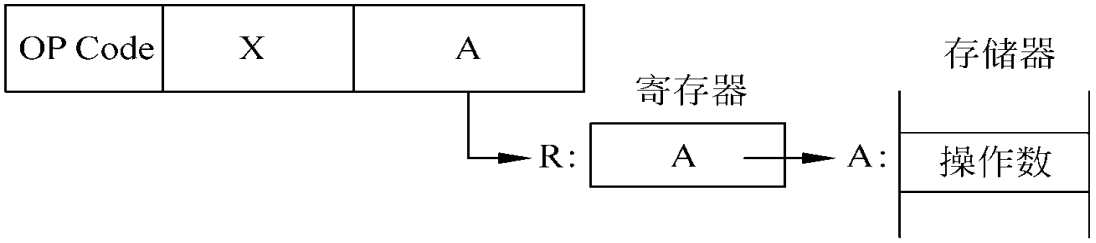


图 5-8 寄存器间接寻址方式

指令采用寄存器间接寻址方式时，由于操作数的有效地址存放在寄存器中，此指令在被执行时，只需访问两次存储器，比间接寻址速度快。

6、变址寻址

变址寻址方式是指指令字中的地址码 A（形式地址）加上指令中指定的变址寄存器的内容形成操作数的有效地址，即 EA=A+（R_i）。如图 5-9 所示。

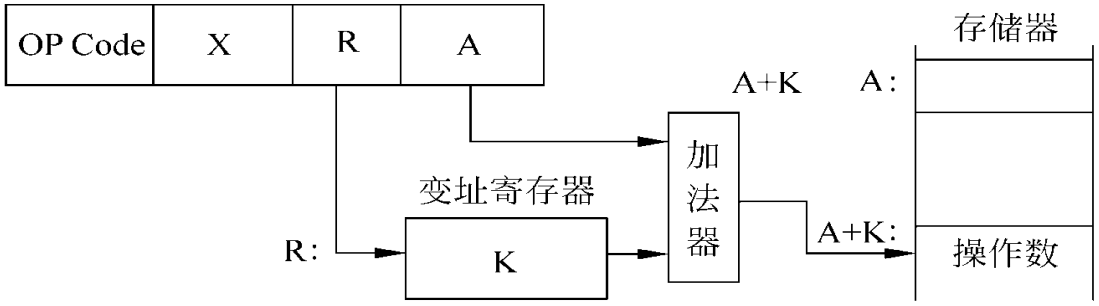


图 5-9　变址寻址方式

指令采用变址寻址方式时，如果操作数地址发生变化时，可不必修改指令，只需按一定增量修改变址寄存器的内容，方便实现循环程序设计。

7、基址寻址

基址寻址方式是指指令字中的地址码 D（形式地址）加上指令中指定的基址寄存器的内容形成操作数的有效地址，即 EA=D+（R_b），基址寄存器通常是专用的寄存器。如图 5-10 所示。

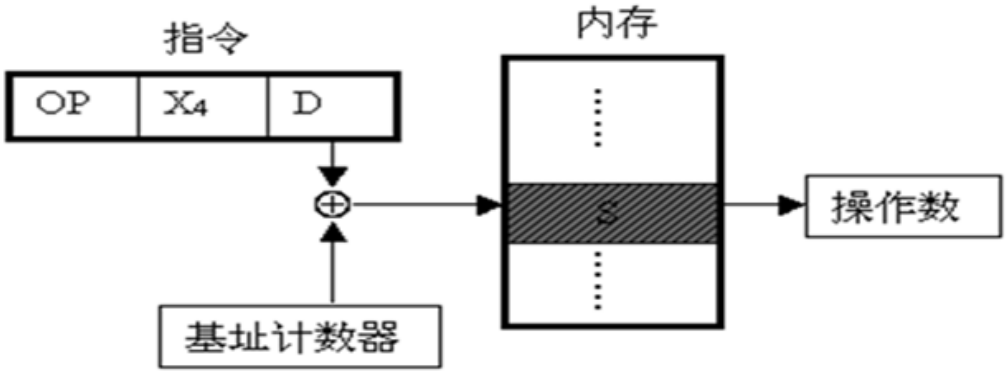


图 5-10 基址寻址方式

基址寻址与变址寻址有效地址的形成过程很相似，但两者的应用有着本质的区别。基址寻址是面向系统的，用于将用户程序的逻辑地址转换成物理地址，以便实现程序的再定位，基址寄存器的内容通常是由操作系统进行设置，不允许进行增量或减量，并对用户是透明的。

8、相对寻址

相对寻址方式是将当前 PC 的内容（现行 PC 加上增量）加上地址码（形式地址）形成操作数的有效地址，即 EA=（PC）+D→PC，D 称为位移量，是定点整数的补码。如图 5-11 所示。

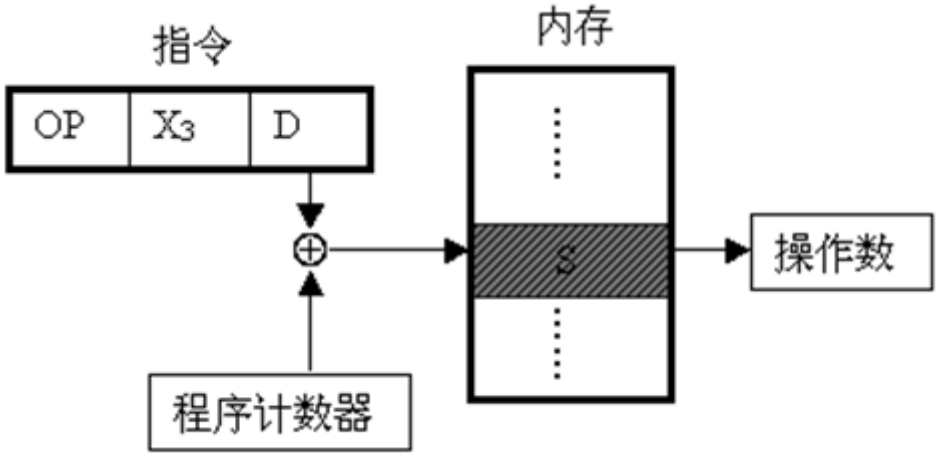


图 5-11 相对寻址方式示意图

指令采用相对寻址方式时，编程时不需要用指令的绝对地址，可以将程序放在内存的任何地方。

9、隐含寻址

隐含寻址是指指令字中的地址码 A 隐含的。对于一地址指令而言，指令格式中地址码 A 就缺省了，尽管没有地址码，但操作数是存在的，并规定为累加器 AC。

5.5 典型的指令系统

5.5.1 复杂指令系统

1、指令类型

不同机器的指令系统是各不相同的，CISC 的指令系统一般有上百种指令，按照指令的功能对指令进行归类，一个较完善的复杂指令系统应该包含以下 8 种指令类型。

（1）数据传送指令

数据传送指令主要包括取数指令、存数指令、传送指令、交换指令、对堆栈操作的指令等，用来实现存储器与寄存器、堆栈之间、寄存器与寄存器、堆栈之间的数据传送。

（2）算术运算指令

算术运算指令主要包括二进制定点数的四则运算指令、十进制数的四则运算指令、浮点数的四则运算指令、算术移位指令、比较指令、求补和求反指令等。这类指令主要用于定点数和浮点数的算术运算。

（3）逻辑运算指令

逻辑运算指令主要包括逻辑与、逻辑或、逻辑非、逻辑移位等指令，主要用于无符号数的位操作、代码转换、逻辑判断。

（4）串操作指令

串操作指令包括串传送指令、串比较指令、串搜索指令、串替换指令等。主要用于对字符串和数据串的处理。

（5）转移指令

转移指令也称程序控制指令，主要包括无条件转移指令、条件转移指令、调用与返回指令、中断与返回指令等，主要用来控制改变程序的执行顺序。

（6）输入输出指令

输入输出指令主要用来控制外设、检测外设的工作状态、实现对外设的数据传送。

（7）特权指令

特权指令是指具有特殊权限的指令，只用于操作系统，一般不会直接提供给用户。

（8）其它指令

其他指令主要指复位指令、停机指令、空操作指令、对状态寄存器的置位和复位指令。

综上所述，CISC 指令系统的特点有：

1）指令系统复杂庞大。表现在指令条数较多

2）指令格式多样。表现在指令字长不固定，寻址方式种类较多

3）指令的执行时间差别较大

4）大多数 CISC 采用微程序控制器

2、8086CPU 指令格式

8086CPU 指令长度可以是 1 ～ 6 字节，其指令格式如图 5-12 所示。其中 B_1 和 B_2 为基本字节，B_3 ～ B_6 将根据不同指令做不同的定义。

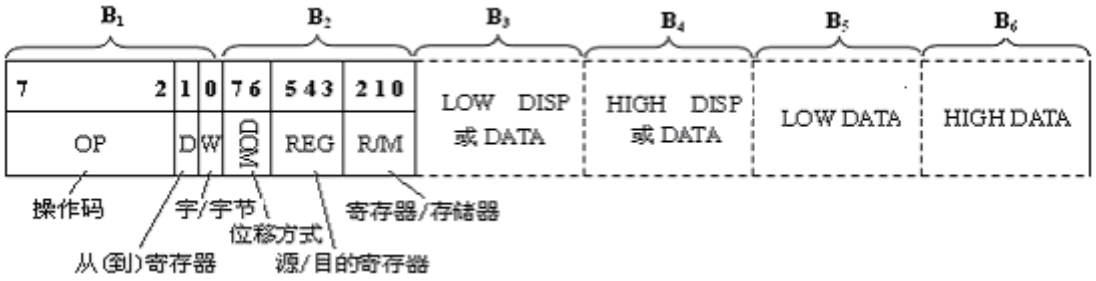


图 5-12 8086 指令格式

（1）B_1 字节各字段定义

1）OP：表示指令操作码

2）D：表示方向。D＝1 时寄存器为目的操作数；D＝0 时寄存器为源操作数。

3）W：表示字节或字处理方式。W＝0 表示字节处理指令；W＝1 表示字处理指令。

（2）B_2 字节各字段定义

1）MOD：表示指令的寻址方式。8086 的一条指令中，最多可使用两个操作数，它们不能同时位于存储器中，最多只能有一个是存储器操作数。当 MOD≠11 时为存储器方式，即有一个操作数位于存储器中；MOD＝00，没有位移量。MOD＝01，只有低 8 位位移量，需将符号扩展 8 位，形成 16 位。MOD＝10 有 16 位位移量。当 MOD＝11 时，为寄存器方式，两个操作数均为寄存器。

2）REG：表示指令中只有一个操作数，这个操作数为寄存器。

3）R/M：R/M 受 MOD 制约。当 MOD=11（即寄存器方式时），由此字段给出指令中第二个操作数所在的寄存器编码；当 MOD≠11 时，此字段用来指出应如何计算指令中使用的存储器操作数的有效地址。MOD 和 R/M 字段表示的有效地址 EA 计算方法。

（3）B_3 ～ B_6 字节

1）这四个字节一般是给出存储器操作数地址的位移量（即偏移量）或立即操作数。位移量可为 8 位，也可为 16 位，这由 MOD 来决定。8086 规定 16 位的字位移量的低位字节放于低地址单元，高位字节放于高地址单元。

2）若指令中只有 8 位位移量，8086 在计算有效地址时，自动用 8 位位移量的符号将其

扩展成一个 16 位的双字节数，以保证有效地址的计算不产生错误，实现正确的寻址。指令中的立即操作数位于位移量的后面。若 B_3B_4 有位移量，立即操作数就位于 B_5B_6。若指令中无位移量，立即操作数就位于 B_3B_4 字节。总之，指令中缺少的项将由后面存在的项向前顶替，以减少指令的长度。

5.5.2 精简指令系统

1、RISC 指令系统的特点

结合表 5-3 所列出的典型 RISC 指令系统的基本特征，我们可以看出 RISC 指令系统的最大特点：

第一是指令条数少，所选取的指令都是使用频率较高的简单指令；

第二是指令字长固定，且指令格式种类少；

第三是只有取数/存数指令可以访问存储器，其它指令的操作都在寄存器之间进行。

表 5-3　　典型 RISC 指令系统的基本特征

机器型号	指令数	寻址方式	指令格式	通用寄存器数	主频/MHz
RISC-1	31	2	2	78	8
MIPS	55	3	4	16	4
SPARC	75	5	3	120~136	25~33
MIPSR3000	91	3	3	32	25
i860	65	3	4	32	50

2、SPARC 机指令系统

SPARC 机是 RISC 最典型代表，机器的字长为 32 位，其指令系统有 75 条指令。

（1）指令类型和指令格式

1）SPARC 机指令类型

① 算术运算/逻辑运算/移位指令

② 取数（LOAD）/存数（STORE）指令

③ 控制转移指令

④ 读/写专用寄存器指令

⑤ 浮点运算指令

⑥ 协处理器指令

2）SPARC 机指令格式

SPARC 机有三种指令格式，如表 5-4 所示。

表 5-4　　SPARC 指令格式

<table>
<tr><td rowspan="2">格式 1
CALL 指令</td><td>31 30</td><td colspan="6">29　28　　0</td></tr>
<tr><td>OP</td><td colspan="6">Disp 30(偏移量)</td></tr>
<tr><td rowspan="3">格式 2
Branc 指令
SETH 指令</td><td>31 30</td><td>29</td><td>28　25</td><td>24 22</td><td colspan="3">21　　0</td></tr>
<tr><td>OP</td><td>a</td><td>Cond</td><td>OP_2</td><td colspan="3">Disp 22(偏移量)</td></tr>
<tr><td>OP</td><td colspan="2">R_d</td><td>OP_2</td><td colspan="3">imm 22(立即数)</td></tr>
<tr><td rowspan="4">格式 3

其他指令</td><td>31 30</td><td colspan="2">29　　25</td><td>24　　19</td><td>18　　15</td><td>13 12</td><td>5　4　　0</td></tr>
<tr><td>OP</td><td colspan="2">R_d</td><td>OP_3</td><td>R_{s1}</td><td>i</td><td>A_{s1}　R_{s2}</td></tr>
<tr><td>OP</td><td colspan="2">R_d</td><td>OP_3</td><td>R_{s1}</td><td>i</td><td>$Simm_{13}$</td></tr>
<tr><td>OP</td><td colspan="2">R_d</td><td>OP_3</td><td>R_{s1}</td><td colspan="2">OP_f　R_{s2}</td></tr>
</table>

其中 OP、OP_2、OP_3 为指令操作码，OP_f 为浮点指令操作码。为了增加立即数和偏移量的长度，调用指令 CALL、转移指令 BRANCH 和 SETHI 指令操作码缩短了，其中 SETHI 指令是将 22 位的立即数(imm22)左移 10 位，送到 R_d 所指示的寄存器中，然后再执行一条加法指令，以补充后面 10 位数据，从而形成 32 位字长的数据。

R_{s1}、R_{s2}为通用寄存器地址，用作源操作数寄存器地址或地址寄存器地址。

R_d为目的寄存器地址，用来保存运算结果或从存储器取来的数据。

$Simm_{13}$是13位扩展符号的立即数。运算时如果$Simm_{13}$最高位是1，则最高位前面的所有位都扩展为1；否则，最高位前面的所有位都扩展为0。

i是用来选择第二个操作数。当i=0，第二个操作数在R_{s2}中；当i=1，第二个操作数为$Simm_{13}$。

（2）指令功能与寻址方式

1）算术逻辑运算指令

功能：将R_{s1}、R_{s2}的内容（或$Simm_{13}$）按操作码规定的操作运算后结果送R_d。

当i=0时，（R_{s1}）OP（R_{s2}）→R_d

当i=1时，（R_{s1}）OP $Simm_{13}$→R_d

2）取数/存数指令

功能：取数指令LOAD将存储器中的数据送往R_d；存数指令STORE将R_d中的数据送往存储器。

存储器地址的计算方法如下（寄存器间接寻址）：

当i=0时，存储器地址=（R_{s1}）+（R_{s2}）

当i=1时，存储器地址（R_{s1}）+ $Simm_{13}$

3）控制转移指令

条件转移（BRANCH）：由Cond字段决定程序是否转移，用相对寻址方式形成转移地址。

转移并连接（JMPL）：将本指令的地址保存在R_d所指示的寄存器中，以备程序返回使用。用寄存器间接寻址形成转移地址。

调用（CALL）：采用相对寻址方式形成转移地址。

陷阱（TRAP）：采用寄存器间接寻址形成转移地址。

TRAP程序返回（RETT）：采用寄存器间接寻址形成转移地址。

4）读/写专用寄存器指令

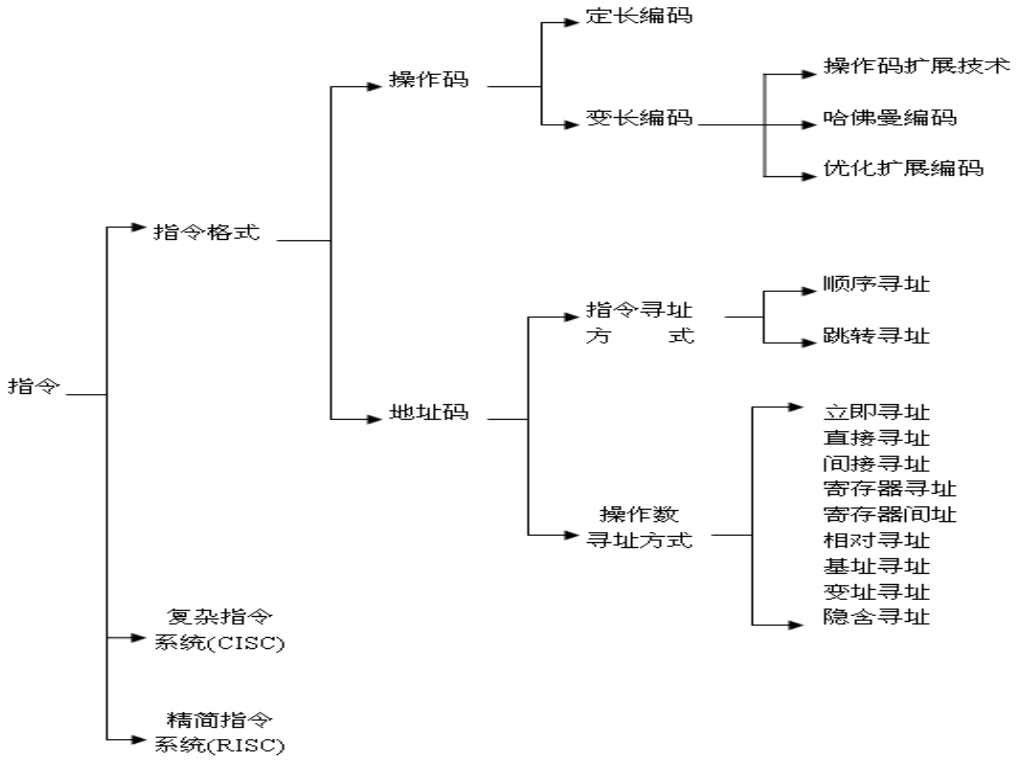
SPARC有4个专用寄存器（PSR、Y、WIM、TBR），其中PSR称为程序状态寄存器，其内容反映并控制机器的运行状态，非常重要，因此，读/写PSR指令一般是特权指令。

综上所述，对于SPARC机的指令系统，由于采用RIS技术，该机器的指令系统只设置了75条指令，还有一些指令并没有选入指令系统，但很容易用指令系统中的指令去替代实现。如表5-5所示。其中，表的左边列出了6条指令，表的右边给出了替代指令及实现方法。（SPARC约定R_0的内容恒为0）

表5-5　　RISC指令的替代与实现

指令	功能	替代指令	实现方法
MOVE	寄存器间数据传送	ADD	$R_s+R_0→R_d$
INC	寄存器内容加1	ADD	imm13=1，为操作数
DEC	寄存器内容减1	SUB	imm13=-1，为操作数
NEG	求补	SUB	$R_0-R_s→R_d$
NOT	逻辑非	XOR	imm13=-1，为操作数
CLR	清除寄存器	ADD	$R_0+R_0→R_d$

关 联


指令
指令格式
操作码
定长编码
变长编码
操作码扩展技术
哈佛曼编码
优化扩展编码
地址码
指令寻址
方 式
顺序寻址
跳转寻址
操作数
寻址方式
立即寻址
直接寻址
间接寻址
寄存器寻址
寄存器间址
相对寻址
基址寻址
变址寻址
隐含寻址
复杂指令
系统(CISC)
精简指令
系统(RISC)

习　题

5.1 什么叫指令？什么叫指令系统？指令通常有哪几种地址格式？

5.2 什么叫指令地址？什么叫形式地址？什么叫有效地址？

5.3 什么叫寻址方式？有哪些基本的寻址方式？简述其寻址过程。

5.4 基址寻址方式和变址寻址方式各有什么不同？

5.5 设某机指令长为 16 位，每个操作数的地址码为 6 位，指令分为单地址指令、双地址指令和零地址指令。若双地址指令为 K 条，零地址指令为 L 条，问最多可有多少条单地址指令？

5.6 设某机指令长为 16 位，每个地址码长为 4 位，试用扩展操作码方法设计指令格式。其中是三地址指令有 10 条，二地址指令为 90 条，单地址指令 32 条，还有若干零地址指令，问零地址指令最多有多少条？

5.7 设某机字长为 32 位，CPU 有 32 个 32 位通用寄存器，有 8 种寻址方式，包括直接寻址、间接寻址、立即寻址、变址寻址等，采用 R-S 型单字长指令格式，共有 120 条指令，试问：

（1）该机直接寻址的最大存储空间为多少？

（2）若采用间接寻址，则可寻址的最大存储空间为多少？如果采用变址寻址呢？

（3）若立即数为带符号的补码整数，试写出立即数范围。

5.8 一种单地址指令格式如下所示，其中 I 为间接特征，X 为寻址模式，D 为形式地址。I、X、D 组成该指令操作数的有效地址 E。设 R 为变址寄存器，R_1 为基址寄存器，PC 为程序计数器，请在下表中第一列位置填写适当的寻址方式。

OP	I	X	D

寻址方式	I	X	有效地址 E
①	0	00	E=D
②	0	01	E=（PC）+ D
③	0	10	E=（R）+ D
④	0	11	E=（R_1）+ D
⑤	1	00	E=（D）
⑥	1	11	E=（（R_1）+ D），D=0

5.9 简述 RISC 的主要特点。

5.10 选择题

（1）计算机系统中，硬件能够直接识别的指令是（　）。

A、机器指令　B、汇编语言指令　C、高级语言指令　D、特权指令

（2）指令系统中采用不同的寻址方式的主要目的是（　）。

A、增加内存的容量　　B、缩短指令长度，扩大寻址范围

C、提高访问内存的速度　　D、简化指令译码电路

（3）在相对寻址方式中，若指令中地址码为 X，则操作数的地址为（　）。

A、X　　B、(PC)+X　　C、X+段基址　　D、变址寄存器+X

（4）在指令的地址字段中直接指出操作数本身的寻址方式，称为（　）。

A、隐含地址　B、立即寻址　C、寄存器寻址　D、直接寻址

（5）在一地址指令格式中，下面论述正确的是（　）。

A、只能有一个操作数，它由地址码提供

B、一定有两个操作数，另一个是隐含的

C、可能有一个操作数，也可能有两个操作数

D、如果有两个操作数，另一个操作数一定在堆栈中

（6）在变址寄存器寻址方式中，若变址寄存器的内容是 4E3CH，给出的偏移量是 63H，则它对应的有效地址是（　）。

A、63H　　B、4D9FH　　C、4E3CH　　D、4F9FH

（7）程序控制类指令的功能是（　）。

A、进行算术运算和逻辑运算

B、进行主存与 CPU 之间的数据传送

C、进行 CPU 和 I/O 设备之间的数据传送

D、改变程序执行的顺序

（8）算术右移指令执行的操作是（　）。

A、符号位填 0，并顺次右移 1 位，最低位移至进位标志位

B、符号位不变，并顺次右移 1 位，最低位移至进位标志位

C、进位标志位移至符号位，顺次右移 1 位，最低位移至进位标志位

D、符号位填 1，并顺次右移 1 位，最低位移至进位标志位

（9）下列几项中，不符合 RISC 指令系统的特点是（　）。

A、指令长度固定，指令种类少

B、寻址方式种类尽量多，指令功能尽可能强

C、增加寄存器的数目，以尽量减少访存次数

D、选取使用频率最高的一些简单指令以及很有用但不复杂的指令

5.13 填空题

（1）一台计算机所具有所有机器指令的集合称为该计算机的_____。它是计算机与_____之间的接口。

（2）在指令编码中，操作码用于表示_____，n 位操作码最多可以表示_____条指令。地址码用于表示_____。

（3）在寄存器寻址方式中，指令的地址码部分给出的是_____，操作数存放在_____。

（4）采用存储器间接寻址方式的指令中，指令的地址码字段中给出的是_____所在的存储器单元地址，CPU 需要访问内存_____次才能获得操作数。

（5）操作数直接出现在指令的地址码字段中的寻址方式称为_____寻址；操作数所在的内存单元地址直接出现在指令的地址码字段中的寻址方式称为_____寻址。

5.14 判断下列各题的正误

（1）利用堆栈进行算术/逻辑运算的指令可以不设置地址码。（　）

（2)指令中地址码所指定寄存器的内容是操作数有效地址的寻址方式称为寄存器寻址。（　）

（3）一条单地址格式的双操作数加法指令，其中一个操作数来自指令中地址字段指定的存储单元，另一个操作数则采用间接寻址方式获得。（　）

（4）在计算机的指令系统中，真正必需的指令种类并不多，很多指令都是为了提高机器速度和便于编程而引入的。（　）

（5）RISC 系统的特征是使用了丰富的寻址方式。（　）

第 6 章　控制器

【内容摘要】

控制器是计算机的一个重要组成部分，是计算机的指挥和控制中心，它与 CPU 密不可分。本章我们从 CPU 功能出发，探讨其内部组成与结构，揭开 CPU 这个神秘的面纱，围绕控制功能的实现，重点介绍微程序控制器、组合逻辑控制器和门阵列控制器的设计思想和方法；结合典型 CPU 案例，体会采用新技术后的高性能微处理器。

【学习要点】

- 控制器的组成及结构
- 控制方式与指令周期
- 控制器的设计（逻辑控制器和微程序控制器）
- 新技术（指令流水线技术、MMS 技术）

6.1　CPU 功能和组成

6.1.1 CPU 的功能

大家知道 CPU 是由运算器和控制器两大部分组成，其基本任务是执行程序，而程序是指令的有序集合，因此，CPU 的基本功能体现在两个方面：一方面是如何保证程序中指令执行顺序的正确，另一方面是如何实现一条指令的功能。具体控制如下：

1、指令控制

按照冯·诺依曼“存储程序”思想，程序被装入主存后，计算机应能按其预先设定的要求有条不紊地执行指令，才可完成具体的任务。因此，严格控制程序的执行顺序，是 CPU 的首要任务。指令控制能实现对程序中指令的执行顺序进行控制。由于程序中的指令有两大类，一是顺序执行的指令，二是转移指令。对于顺序执行的指令，当 CPU 执行完此指令后，该指令下面的一条指令便是 CPU 要执行的下一条指令，由于程序的连续存放，可以通过设置一个程序计数器 PC 或指令指针（Program　counter）进行控制，且 PC 具有自动加 1 的功能，因此，PC 始终指向的是下一条 CPU 将要执行的指令。对于转移指令，当 CPU 执行完此指令后，将转移的目的指令所在的地址送 PC，便实现转移指令的正确转移。

2、操作控制

对于一条指令的执行，要涉及到计算机中的若干个部件。指令不同，控制完成指令的功能所涉及的部件不同。如何控制这些部件，就需要各种不同的操作控制信号。因此，一条指令对应一组操作控制信号。指令不同，指令的操作码不同，指令所对应的一组操作控制信号也不同。因此，操作控制是对指令的操作码进行译码来产生该指令所需要的一组操作控制信号。

3、时序控制

在操作控制的基础上，需要对各种操作控制信号的产生时间、稳定时间、撤销时间及相互之间的关系进行严格控制。因为指令所对应一组操作控制信号并不是同时作用于相应的各个部件。这种对操作控制信号施加时间上的控制，称为时序控制。实现时序控制需要设置时序产生器和操作控制器。只有严格地进行时序控制，才能保证各功能部件组合构成有机的计算机系统。

4、数据加工

数据加工是对数据进行算术运算和逻辑运算，它是 CPU 的根本任务，也是运算器的基本功能，而运算器是 CPU 组成的一个部分，因此，数据加工不可避免地会涉及到运算器和寄存器。

6.1.2 CPU 的基本组成

CPU 是运算器和控制器的总称，它既具有运算器的功能，又具有控制器的功能。欲实现控制器的功能，作为控制器应由程序计数器 PC（Program counter）、指令寄存器 IR（Instrack Register）、指令译码器、时序产生器和操作控制器组成。欲实现运算器的功能，作为运算器应由算术逻辑单元 ALU、累加寄存器简称累加器 AC（Accunulator）、数据缓冲寄存器 DR（Data Register）、状态寄存器 STR（Stru Register）组成。具体 CPU 模型如图 6-1 所示。

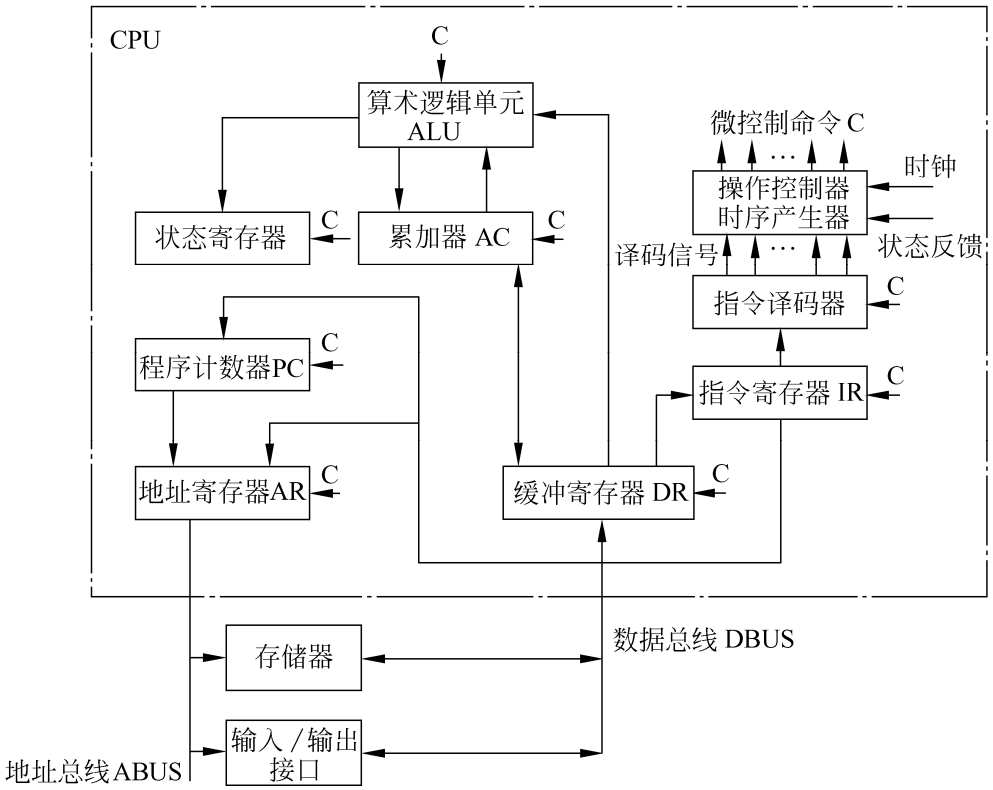


图 6-1 CPU 主要组成部件逻辑结构示意图

1、指令的执行过程

（1）取指令

根据指令所在存储器单元的地址（由程序计数器 PC 提供），将 PC 中的指令地址送地址寄存器 AR，此后，PC 自动加 1，地址寄存器 AR 的内容经地址总线和地址译码器选中指令所在的存储单元，CPU 发出读命令，将该指令从主存中取出，经数据总线送到数据缓冲寄存器 DR，由于所取的是指令而不是数据，因此，再将数据缓冲寄存器 DR 中的指令送往指令寄存器 IR，从而完成取指令。由此可见：

1）所有指令的取指令阶段是完全相同的

2）CPU 当前正在执行的指令是指令寄存器 IR 中的指令

3）程序计数器 PC 始终指向的是下一条 CPU 将要执行的指令

（2）分析指令

对指令寄存器 IR 中指令的操作码进行译码分析，产生该指令所需要的一组操作控制信号（译码信号），再经时序产生器和操作控制器形成时序控制信号（控制命令）；通过对指令寄存器 IR 中指令的地址码进行分析，形成操作对象（操作数）的有效地址，并按此地址去读取操作数，或形成转移地址，以实现程序的转移。

（3）执行指令

时序控制信号（控制命令）作用于相应的各个部件，使各个部件产生相应的动作，从而

完成指令的功能，并根据需要，保存操作结果。

一条指令执行结束，若没有异常情况和特殊请求，则按程序顺序，再去取出并执行下一条指令。控制器就是按取指令、分析指令、执行指令这样的步骤进行周而复始的控制过程，直到完成程序所规定的任务并停机为止。

2、CPU 内部主要寄存器

CPU 内部寄存器是用来保存运算和控制过程中的中间结果、最后结果和控制、状态信息。不同的 CPU，其内部寄存器可能有所差异，但不论哪一种 CPU，其内部寄存器至少要有一下 6 类寄存器。

（1）数据缓冲寄存器（DR）

数据缓冲寄存器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，当向内存存入一条指令或一个数据字时，也暂时将它们存放在数据缓冲寄存器中。 缓冲寄存器的作用是 ：

1）作为 CPU 和内存、外部设备之间信息传送的中转站

2）补偿 CPU 和内存、外围设备之间在操作速度上的差别

3）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

（2）指令寄存器（IR）

指令寄存器用来保存当前正在执行的一条指令。当执行一条指令时，先把它从内存取到缓冲寄存器中，然后再传送至指令寄存器。指令划分为操作码和地址码字段，由二进制数字组成。为了执行任何给定的指令，必须对操作码进行测试，以便识别所要求的操作。 指令译码器就是做这项工作的。指令寄存器中操作码字段的输出就是指令译码器的输入。操作码一经译码后，即可向操作控制器发出具体操作的特定信号。

（3）程序计数器（PC）

为了保证程序中指令的执行顺序的正确，CPU 必须设置一个计数器，用来指示 CPU 将要执行的下一条指令，起这种作用的计数器通常称为指令计数器，简称 PC。在程序开始执行前，必须将程序的起始地址装入 PC，即程序的一条指令所在内存单元地址送入 PC。当执行指令时，CPU 将自动修改 PC 的内容，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的对 PC 加 1。但是，当遇到转移指令时，那么后继指令的地址（即 PC 的内容)必须从指令的地址码取得，在这种情况下，下一条从内存取出的指令将由转移指令来规定，不同于顺序指令只是简单的对 PC 加 1。因此，程序计数器的结构应当是具有寄存和计数两种功能的结构。

（4）地址寄存器（AR）

地址寄存器用来保存当前 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。当 CPU 访问主存时，即 CPU 对主存存/取数据或者 CPU 从主存中读出指令时，都要使用地址寄存器和数据缓冲寄存器。同样，当 CPU 访问外设时，同样要使用地址寄存器和数据缓冲寄存器。

（5）累加器（AC）

累加器是累加寄存器的简称，它是一个通用寄存器。其功能是：当运算器进行算术或逻辑运算时，一方面作为 ALU 一个输入端，为 ALU 提供一个操作数；另一方面作为 ALU 一个输出端，用来存放 ALU 运算的结果。显然，运算器中至少要有一个累加器，当运算器内部只有一个累加器时，我们称为单累加器结构的运算器或 CPU。目前，CPU 中的累加器多达 16 个、32 个、甚至更多，当使用多个累加器时，就变成通用寄存器堆结构，其中任何一个既可存放源操作数，也可存放目的操作数。

（6）状态寄存器（SR）

状态寄存器也称为程序状态字，简称 PSW，用来保存 CPU 在执行算术指令和逻辑指令时建立的各种条件码内容，如运算结果进位标志(C)，运算结果溢出标志（V)，运算结果为零标志(Z)，运算结果为负标志(N)等等。这些标志位通常分别由 1 位触发器保存。除此之外，状态寄存器还保存中断和系统工作状态等信息，以便使 CPU 和系统能及时了解机器工作状态和程序运行状态。因此，状态寄存器是一个由各种状态条件标志拼凑而成的寄存器。

3、控制器的分类

根据时序控制信号或控制命令产生方法不同，可以将控制器分为组合逻辑控制器、微程序控制器和门阵列控制器三种。本章将重点讨论微程序控制器和组合逻辑控制器。

（1）组合逻辑控制器

组合逻辑控制器是采用组合逻辑技术来产生控制命令，它是分立元件时代的产物，其最大优点是速度快，但结构不规整，设计、调试和维修较困难，难以实现设计自动化，目前只有一些巨型机和 RISC 机为了追求高速度仍采用组合逻辑控制器。

（2）微程序控制器

微程序控制器是采用存储逻辑来产生控制命令，它是将控制命令代码化，机器指令转化成一段微程序，并存于控制存储器中，通过执行控存中的微程序来产生机器指令所需的一组控制命令。它与组合逻辑控制器的设计思想截然不同，微程序控制器设计规整，调试、维修以及更改、扩充指令方便，易于实现自动化设计，已成为当前控制器的主流。

（3）门阵列控制器

门阵列控制器实质上也是一种组合逻辑控制器，但它与传统的组合逻辑控制器不同，它是程序可编的，它吸收了前两种控制器的设计思想，被广泛应用于嵌入式系统。

6.2 控制器的时序系统和控制方式

6.2.1 有关周期的基本概念

1、指令周期

指令周期是指从取指令、分析指令到执行完该指令所需的全部时间。一个指令周期一般包含若干个 CPU 周期。由于指令的操作功能不同，有的简单，有的复杂，因此，不同指令的指令周期不尽相同。

2、CPU 周期

一个 CPU 周期又称一个机器周期，与机器的一个基本操作相对应，结合机器的基本操作，一般机器的 CPU 周期有取指周期、取数周期、执行周期、中断周期等。由于 CPU 内部的操作速度较快，而 CPU 访问一次内存所花的时间较长，因此，通常用从内存读取一条指令字的最短时间来规定 CPU 周期。

3、节拍

在一个 CPU 周期内要完成一个基本操作，由于一个基本操作又包含若干个微操作，这些微操作不但需要占用一定的时间，而且有一定的先后次序。因此，需要把一个 CPU 周期等分成若干个时间小段，每一小段称为一个节拍，一个节拍对应一个电位信号，控制一个或几个微操作的执行。

4、脉冲

在一个节拍内，有时还需要设置一个或几个工作脉冲，工作脉冲也称为时钟周期或 T 周期。

上述指令周期、CPU 周期、时钟周期之间的关系如图 6-2 所示。一个指令周期包含两个 CPU 周期，每个 CPU 周期包含四个时钟周期。

6.2.2 时序信号与体制

时序信号是对操作控制信号实施时间控制而形成的信号，是利用定时脉冲的顺序和不同

的脉冲间隔，有条理、有节奏地指挥机器的动作，规定在这个脉冲到来时做什么，在那个脉冲到来时又做什么，给计算机各部分提供工作所需的时间标志。为此，需要采用周期、节拍、脉冲多级时序体制。至于计算机内存中所存放的二进制形式的指令和数据的区分问题，我们可以从两个层面上进行区分，一是从时间上来说，取指令是发生在指令周期的第一个 CPU 周期中，即发生在“取指令”阶段，而取数据是发生在指令周期的后面几个 CPU 周期中，即发生在“执行指令”阶段。二是从空间上来说，如果取出的代码是指令，那么一定送往指令寄存器，如果取出的代码是数据，那么一定送往运算器。由此可见，时间控制对计算机来说是十分重要，计算机的协调动作需要时间标志，而时间标志则是用时序信号来体现的，控制器所产生的各种控制信号都是时间因素（时序信号）和空间因素（部件）的函数。

时序信号最基本的体制是电位—脉冲制，该体制下最容易理解的例子是寄存器之间的数据传送，数据加在触发器的电位输入端，用电位的高低来表示数据是“1”还是“0”；而打入数据的控制信号加在触发器的时钟输入端，且要求打入数据的控制信号到来之前，电位信号必须是稳定的，只有电位信号先建立，打入到寄存器中的数据才是可靠的。

在组合逻辑控制器中，时序信号的体制往往采用周期—节拍—脉冲三级体制，即一个 CPU 周期包含多个节拍，每个节拍又允许包含多个脉冲。在图 6-5 中，一个 CPU 周期包含 4 个节拍，每个节拍又包含 1 个脉冲。

在微程序控制器中，时序信号比较简单，一般采用节拍—脉冲二级体制。

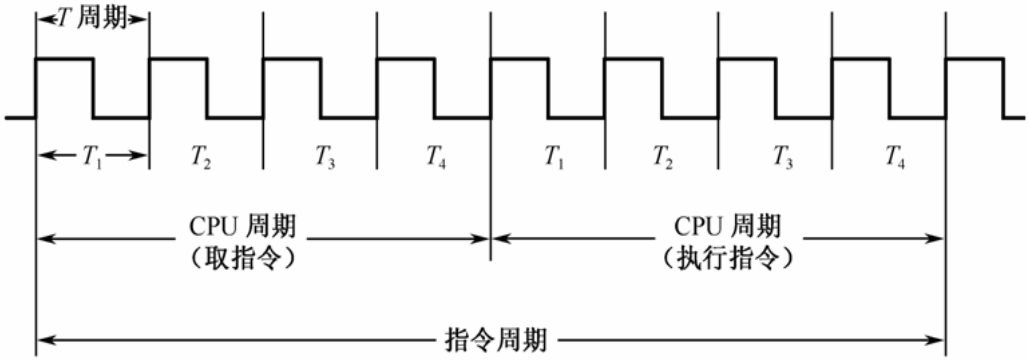


图 6-2　指令周期、CPU 周期、时钟周期间的关系

6.2.3 时序信号发生器

时序信号发生器的电路是不尽相同的，组合逻辑控制器的时序电路复杂，而微程序控制器的时序电路简单，但无论哪一种控制器，其内部的时序信号发生器最基本的构成是一样的，都是由时钟源、环形脉冲发生器、节拍脉冲和读写时序译码逻辑、启停控制逻辑等部分组成。如图 6-3 所示。

1、时钟源

时钟源用来为环形脉冲发生器提供频率稳定且电平匹配的方波时钟脉冲信号。它通常由石英晶体振荡器和与非门组成的正反馈振荡电路组成，其输出送至环形脉冲发生器。

2、环形脉冲发生器

环形脉冲发生器是用来产生一组有序的间隔相等或不等的脉冲序列，以便通过译码电路来产生最后所需的节拍脉冲。为了在节拍脉冲上不带干扰毛刺，环形脉冲发生器通常采用循环移位寄存器形式。典型的环形脉冲发生器及其译码如图 6-4 所示。

假设时钟源产生的时钟信号为 5*MHz*（时钟周期 200*ns*），当控制台发出总清信号（$\overline{\text{CLR}}$），使触发器 C_4 置“1”，门 3 打开，第一个正脉冲 Φ 通过门 3，使触发器 C_1~C_3 清“0”。经过半个脉冲周期（100*ns*）的延迟，触发器 C_4 由“1”状态翻到“0”状态，再经半个脉冲周期（100*ns*）的延迟，第二个正脉冲的上升沿（即第一个 $\overline{\Phi}$ 的后沿）作移位信号，使触发器 C_1~C_3

变为“100”状态。此后，第二个$\overline{\Phi}$、第三个$\overline{\Phi}$连续通过门 2 形成移位信号，使触发器 C_1~C_3 相继变为“110”、“111”状态。当 C_3 变为“1”状态时，对应第四个正脉冲，其状态便反映

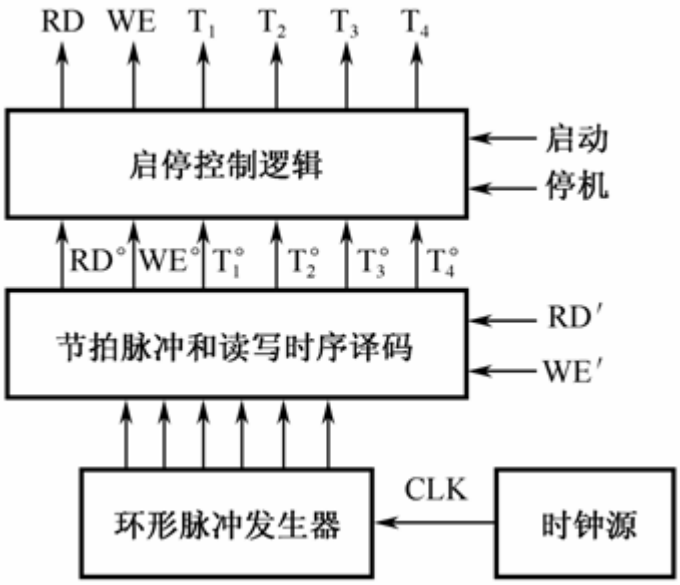


图 6-3　时序信号发生器

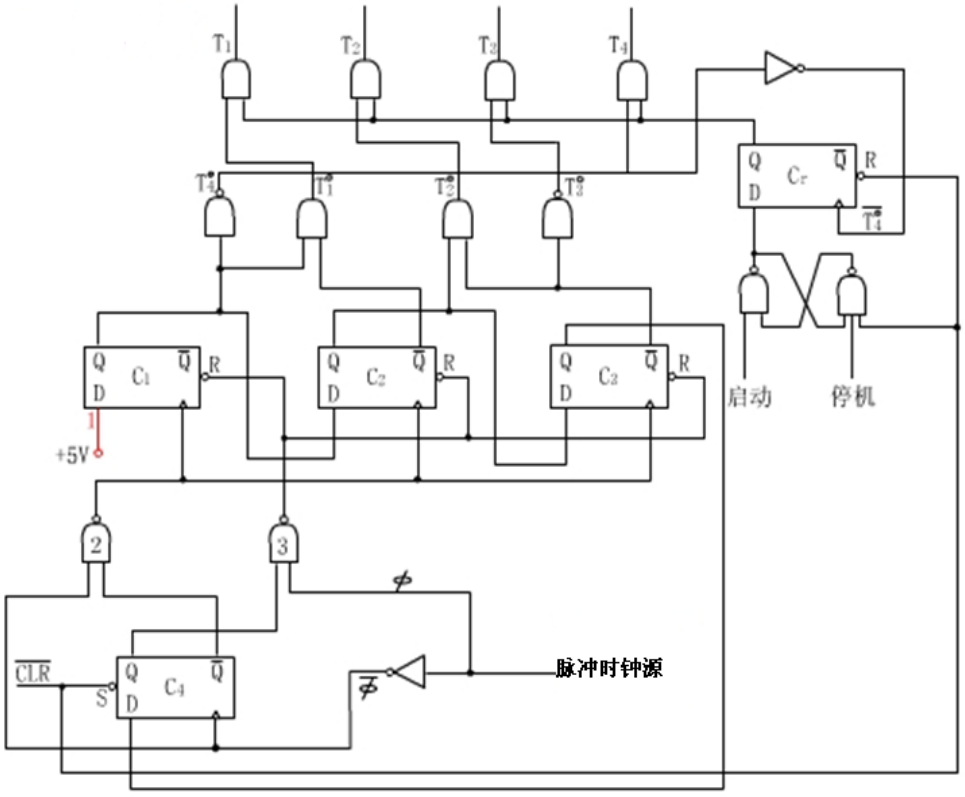


图 6-4　环形脉冲发生器及其译码逻辑

到触发器 C_4 的 D 端，因而第四个正脉冲的下降沿又将 C_4 置“1”，门 3 再次打开，第五个正脉冲便通过门 3，形成清“0”脉冲，使触发器 C_1~C_3 清“0”。于是下一个循环再度开始。其过程如图 6-5 所示。

3、节拍脉冲和读写时序的译码

节拍脉冲和读写时序的译码逻辑如图 6-4 的上半部。假设一个 CPU 周期产生 4 个等间

隔的节拍，根据图 6-5 节拍与脉冲之间的时序关系，其译码逻辑可表示为：

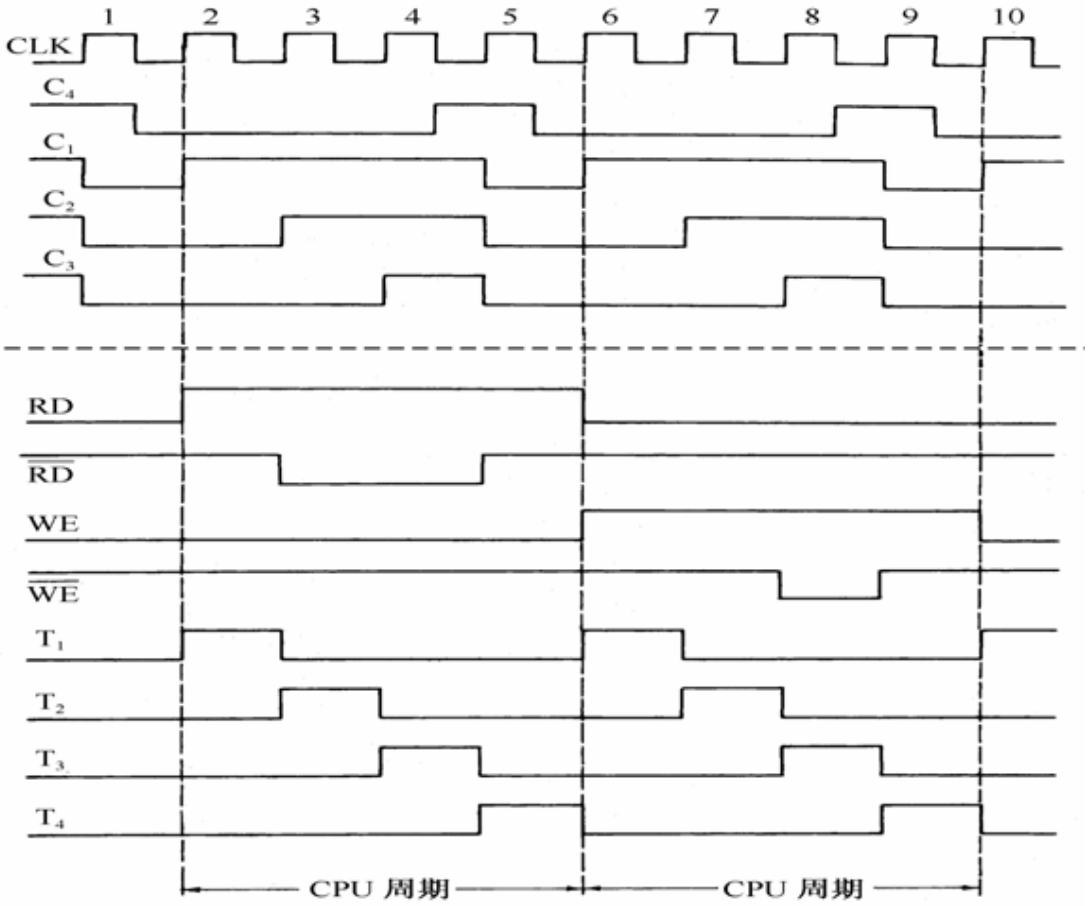


图 6-5　节拍与脉冲之间的时序关系

$T_1^0=C_1 \cdot \overline{C_2}$　　$T_2^0=C_2 \cdot \overline{C_3}$

$T_3^0=C_3$　　$T_4^0=\overline{C_1}$

RD°=C_2 · RD′　　WE°=C_3 · WE′

值得注意的是：

（1）节拍脉冲间逻辑关系

节拍 T_1^0、T_2^0、T_3^0、T_4^0 与图 6-5 中的节拍 T_1、T_2、T_3、T_4 在逻辑关系上是完全相同，只是后者是经过启停控制逻辑中的与门输出的。

（2）节拍脉冲宽度

一个 CPU 周期为 800*ns*，在周期—节拍—脉冲三级体制中，一个 CPU 周期包含 4 个节拍，每一个节拍包含一个脉冲（时钟周期），因此，节拍 T_1^0、T_2^0、T_3^0、T_4^0 的脉冲宽度均为 200*ns*。

（3）读写控制信号

由图 6-5 可以看出：信号 RD′/WE′ 的持续时间为一个 CPU 周期。根据图 6-4 可知，信号 RD°/WE° 受控于信号 RD′/WE′，只有当 RD′/WE′ 有效后才会产生 RD°/WE°，结合图 6-6，读写信号 $\overline{RD}/\overline{WE}$ 是 RD°/WE° 经过启停控制逻辑中的与非门输出的。

4、启停控制逻辑

启停控制逻辑电路如图 6-6 所示。其核心是触发器 Cr，当触发器 Cr 为“1”时，原始的节拍 T_1^0、T_2^0、T_3^0、T_4^0 和读/写时序信号 RD°/WE° 通过门电路转变为 CPU 真正需要的节拍 T_1、T_2、T_3、T_4 和读/写时序信号 $\overline{RD}/\overline{WE}$；否则，当触发器 Cr 为“0”时，将关闭时序信号发生器。为了保证时序信号发生器产生完整的脉冲，在 T_1 的前沿启动时序信号发生

器，在 T_4 的后沿关闭时序信号发生器，在触发器 Cr 的下面加上一个 RS 触发器，且用 $\overline{T_4^0}$ 作为触发器 Cr 的时钟控制端。

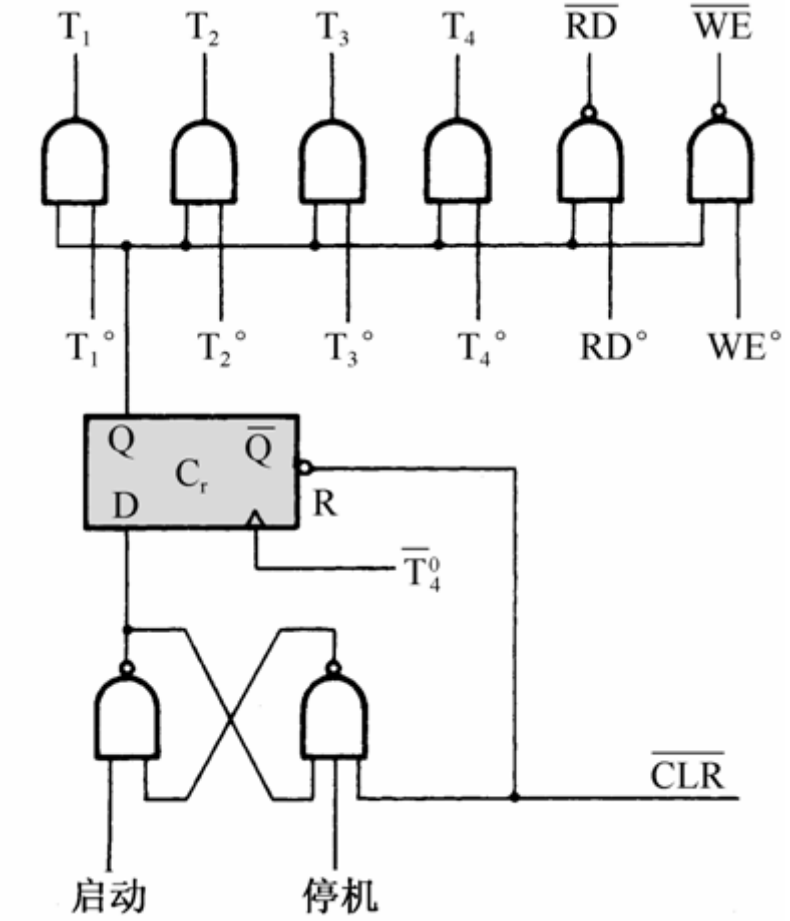


图 6-6　启停控制逻辑

6.2.4 控制方式

控制方式是指对各种操作在时间上所进行的控制，其实质反映了时序信号的定时方式。常用的控制方式有同步控制、异步控制、联合控制。

1、同步控制方式

同步控制方式是指任何指令的执行或指令中各个具体操作的执行均由确定统一的时间基准信号所控制，尽管指令功能不同，但在同步控制方式下，指令在执行时所需的机器周期数和时钟周期数都固定不变，即完全同步控制方式，此方式的优点是时序关系简单，控制方便，但会造成大量时间的浪费。这是因为大多数简单指令的很多节拍是不用的，处于等待状态。因此，在实际应用中往往采取折中的方案。常用的方法有：

（1）采用中央控制与局部控制相结合的方法

根据周期—节拍—脉冲三级体制，一个 CPU 周期包含多个节拍。大多数指令的 CPU 周期均采用统一的节拍，称为中央控制，对于少数在统一节拍内不能完成的指令，需要延长节拍或增加节拍，使之在延长节拍内完成指令，并再返回到中央控制，这种通过延长节拍或增加节拍来控制完成指令称为局部控制。

（2）采用不同的机器周期和延长节拍的方法

不同指令的指令周期可以划分为若干个 CPU 周期，如取指、取数、执行等周期，根据执行指令的需要，可选取不同的 CPU 周期数。在节拍安排上，每个 CPU 周期划分为固定的节拍，每个节拍都可根据需要延长一个节拍。

（3）采用分散节拍的方法

分散节拍是指根据操作的实际需要，需要多少节拍，时序发生器就产生多少节拍，这样可以做到完全避免节拍轮空，是提高指令运行速度的有效方法，但这种方法会使时序发生器复杂化，同时也无法解决节拍内那些简单操作因等待所浪费的时间。

2、异步控制方式

异步控制方式是根据每条指令或每个操作的实际需要，需要多少时间就分配多少时间。这意味着每条指令的指令周期可由多少不等的机器周期数组成；也可以是当控制器发出某一操作控制信号后，等待执行部件完成操作后发“回答”信号，再开始新的操作。显然，用这种方式形成的操作控制序列没有固定的 CPU 周期数(节拍电位)或严格的时钟周期(节拍脉冲)与之同步。

3、联合控制方式

联合控制方式是同步控制与异步控制相结合的方式。现代计算机中没有完全采用同步或完全采用异步的控制方式，大多数都采用联合控制方式。即在功能部件内部采用同步控制方式或以同步控制方式为主的控制方式，在功能部件之间采用异步控制方式。

6.3 指令流程图

指令流程图是控制器设计的基础，它是用方框图语言来表示的指令周期。结合图 6-1 CPU 的结构模型，分析常用指令的指令周期，并在此基础上划出指令流程图。

6.3.1 五类典型指令的指令周期分析

五类典型指令包括非访内指令（CLA 指令）、直接访内指令（ADD 指令）、写存指令（STA 指令）、转移指令（JMP 指令）和空操作指令（NOP 指令）。这些指令的存储见表 6-1。

表 6-1　　五条典型的指令

八进制地址	八进制内容	助记符
020	250 000	CLA
021	030 000	ADD 30
022	020 040	STA 40
023	000 000	NOP
024	140 021	JMP 21
⋮	⋮	⋮
030	000 006	数据
031	000 040	数据
⋮	⋮	⋮
040	存和数单元	数据

1、CLA 指令周期

CLA 指令是一条非访内指令，此类指令的指令周期需要两个 CPU 周期，第一个 CPU 周期完成取指和译码操作，第二个 CPU 周期用作指令的执行操作。CLA 指令周期如图 6-7 所示。非访内指令包括寄存器之间的数据传送指令、对累加器操作的指令和其他一些零地址指令，下面以 CLA 指令为例来说明非访内指令的两个 CPU 周期所进行的具体操作。

（1）取指周期

CLA 指令的第一个 CPU 周期为取指周期，此周期完成取指令并对指令的操作码进行译码。假设程序计数器 PC 的内容为 020Q，正指向 CLA 指令，则取指周期所进行的具体操作如图 6-8 所示。操作步骤如下：

①将程序计数器 PC 的内容 020Q 送入地址寄存器 AR　（PC→AR）

②程序计数器 PC 的值加 1，使 PC 指向下一条 CPU 将要执行的指令　（PC+1）

③将地址寄存器 AR 的内容送到地址总线上　（AR→ABUS）

④CPU 发出读命令，将所选主存地址为 020 的单元内容“250 000”读出，经过数据总线，传送给数据缓冲寄存器 DR　（$R/\overline{W}=1$）

⑤数据缓冲寄存器的内容“250 000”送给指令寄存器 IR　（DR→IR）

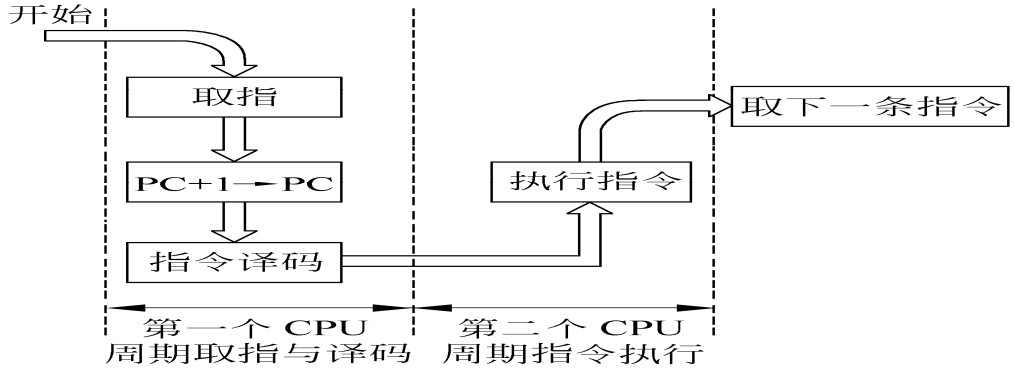


图 6-7　CLA 指令的指令周期

指令译码器 ID 对指令寄存器中操作码进行译码，产生出该指令所需的所有控制信号，以便执行指令操作。由于所有指令的指令周期中第一个 CPU 周期都是取指周期，该周期所进行的操作都相同，因此，取指周期作为指令流程图的公共部分，任何一条指令执行结束时必须返回到取指周期，以便继续取下一条指令。

（2）执行周期

CLA 指令的第二个 CPU 周期为执行周期，此周期完成对累加器 AC 的清 0。其具体操作如图 6-9 所示。操作步骤如下：

①操作控制器送 CLA 相应的控制信号给算术逻辑单元 ALU　（0→AC）

②ALU 响应该控制信号，将累加器 AC 的内容清零　（AC=0）

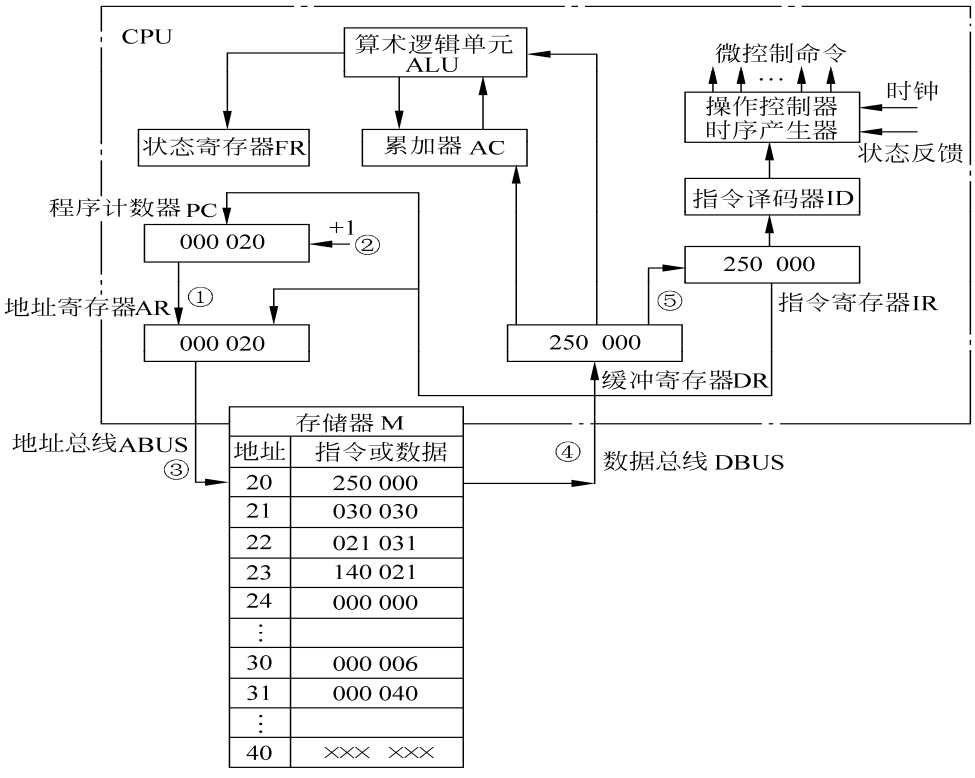


图 6-8　CLA 取指周期阶段

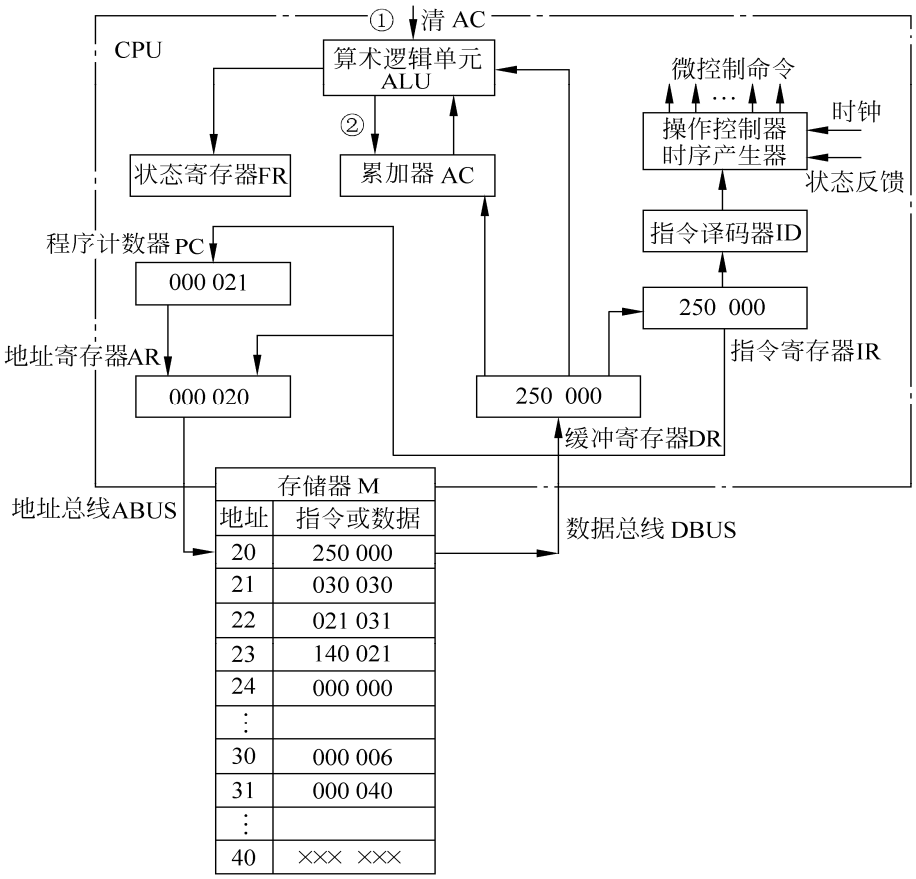


图 6-9　CLA 执行周期

2、ADD 指令周期分

ADD 指令是一条直接访内指令，此类指令的指令周期需要三个 CPU 周期，第一个 CPU 周期完成取指和译码操作，第二个 CPU 周期将指令寄存器 IR 中的地址码送往地址寄存器，第三个 CPU 周期从内存中取出操作数并完成相加操作。ADD 指令周期如图 6-10 所示。

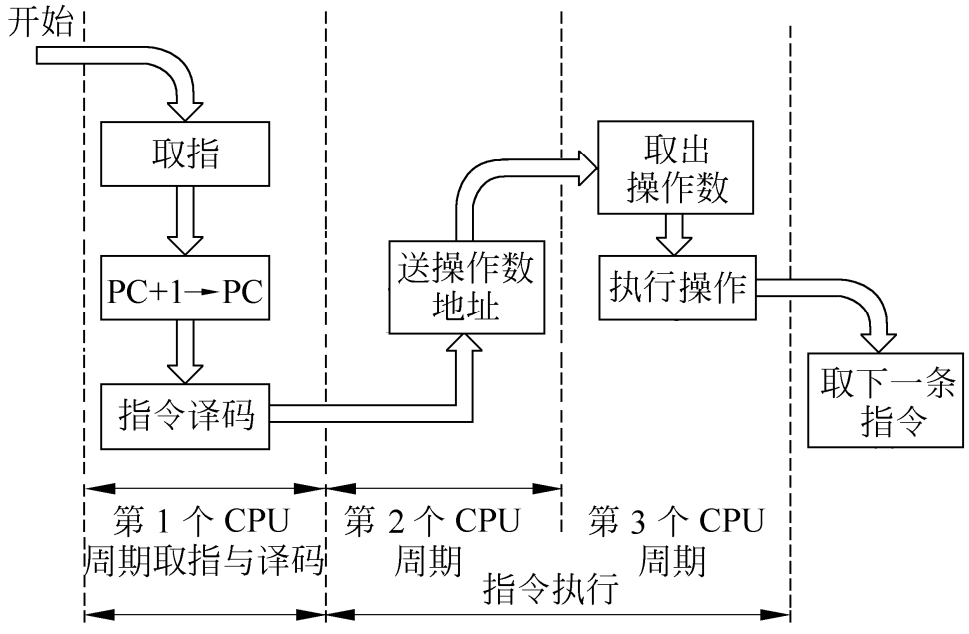


图 6-10 ADD 指令的指令周期

（1）取指周期

在取得前一条指令 CLA 后，程序计数器 PC 的值已经加 1，修改为 021Q，正指向当前 ADD 指令。经 ADD 指令的第一个 CPU 周期后，取出 021Q 单元的内容 “030 030”（ADD 30 指令）送指令寄存器 IR，PC 的值加 1，即 PC 的值修改为 22，指向下一条 CPU 将要执行的指令，经指令译码器得出该指令的功能，即将累加器的内容和主存 030 单元的内容相加。由于 ADD 指令的取指、译码操作过程与 CLA 指令的第一个 CPU 周期完全相同，在此不作重复。

（2）指令寄存器 IR 中的地址码装入地址寄存器 AR（IR_{Addr}→AR）

ADD 指令的第二个 CPU 周期完成指令寄存器 IR 中的地址码（030）装入地址寄存器 AR，表示为 030→AR，其中 030 为内存中存放操作数的地址。具体操作如图 6-11 所示。

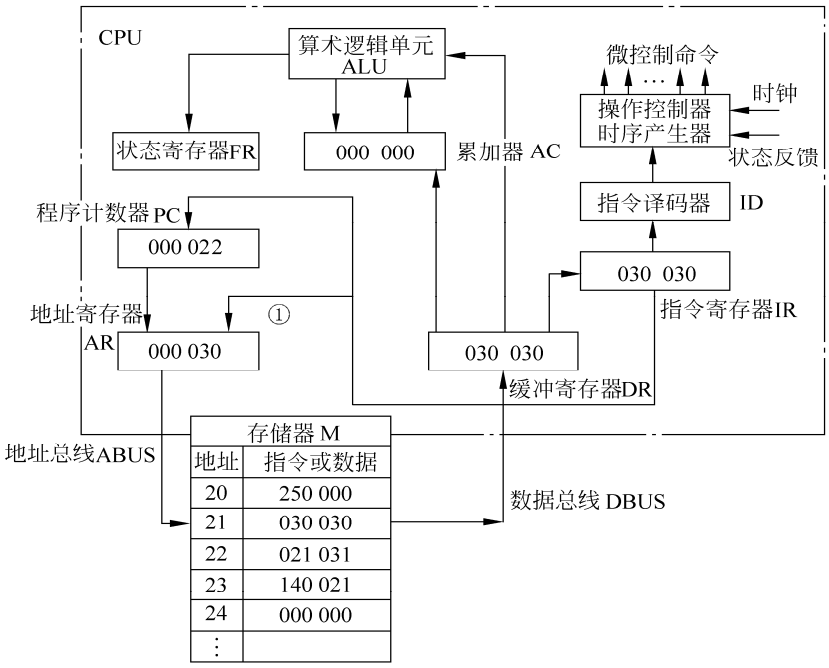


图 6-11　指令寄存器 IR 中的地址码（030）装入地址寄存器 AR

根据图 6-1CPU 模型，对于单累加器结构的 CPU，一个操作数必然隐含在累加器 AC 中，另一个操作数只能来源于数据缓冲寄存器 DR，因此，对于直接寻址指令而言，都会面临一个相同的操作，即将指令寄存器 IR 中的地址码装入地址寄存器 AR（IR_{Addr}→AR），在此，我们将此操作理解为一个机器周期或 CPU 周期。

（3）两操作数相加

ADD 指令的第三个 CPU 周期主要完成取操作数并执行加法操作。其具体操作如图 6-12 所示。操作步骤如下：

①把地址寄存器 AR 中的操作数的地址（30）送地址总线；　（AR→ABUS）

②CPU 发出读命令，将主存地址为 30 的单元内容 6 读出，经过数据总线送给数据缓冲寄存器 DR；　（$R/\overline{W}=1$）

③执行加法运算。数据缓冲寄存器的内容 6 为算术逻辑单元提供一个操作数，累加器为 ALU 提供另一个操作数，两个操作数经 ALU 相加，并把加法的结果送给累加器，此时，累加器的内容为 6。　（“+”）

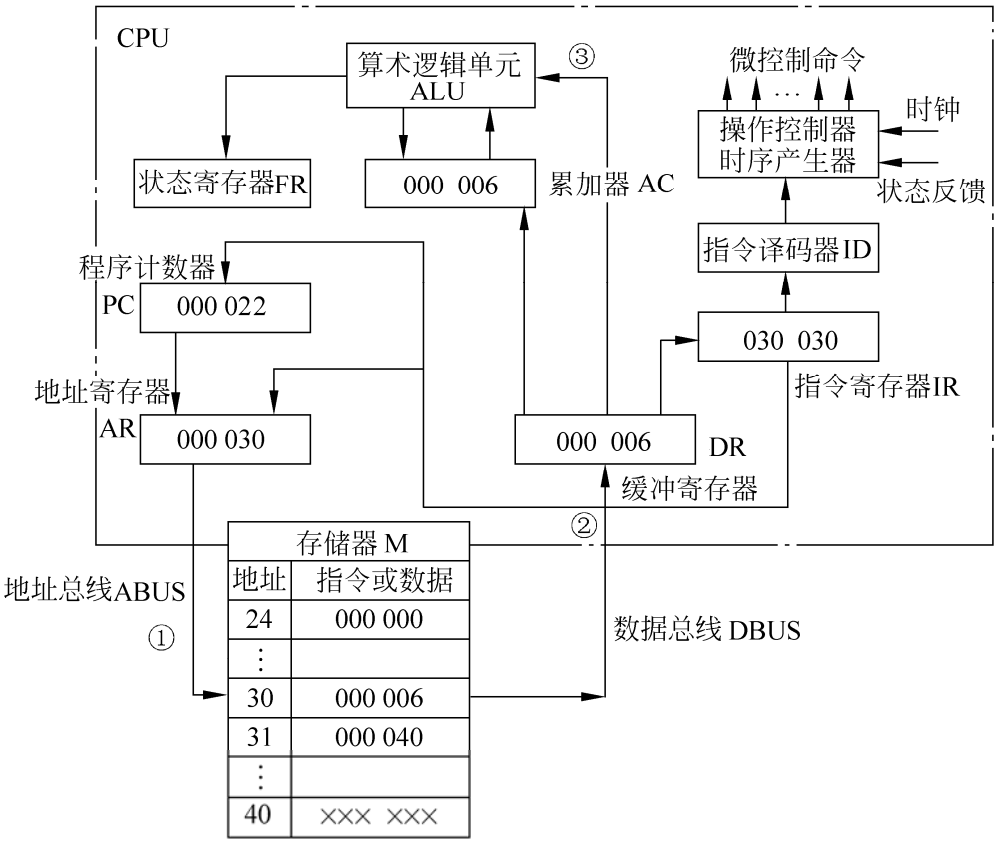


图 6-12　两操作数相加

3、STA 指令周期

STA 指令是一条间接访内指令，该指令的指令周期由 4 个 CPU 周期组成，如图 6-13 所示。第一个 CPU 周期完成取指和译码操作，第二个 CPU 周期将指令寄存器 IR 中指令的地址码送往地址寄存器，第三个 CPU 周期从内存中取出操作数的地址并送往地址寄存器，第四个 CPU 周期将累加器 AC 的内容写入内存单元。

（1）取指周期

STA 指令的第一个 CPU 周期仍为取指周期，将当前 PC 所指的主存单元中的指令“STA I 30”（机器码为“021031”）取出并译码，PC 的值加 1，指向下一条 CPU 将要执行的指令。由于 STA 指令的第一个 CPU 周期与 CLA、ADD 指令的第一个 CPU 周期的操作完全相同，所以在此不再重复。

（2）指令寄存器 IR 中的地址码装入地址寄存器 AR（IR_{Addr}→AR）

STA 指令的第二个 CPU 周期将指令寄存器 IR 中的地址码（031）装入地址寄存器 AR（031→AR）。由于 STA 指令的第二个 CPU 周期与 ADD 指令的第二个 CPU 周期的操作完全相同，所以在此也不再重复。只是强调：ADD 指令中的地址码 030 是操作数所在内存单元的地址，而 STA 指令中的地址码 31 并不是操作数所在内存单元的地址，而是操作数所在内存单元的地址所在的内存单元的地址。简单地讲，STA 指令中的地址码 31 是操作数所在内存单元地址的地址。

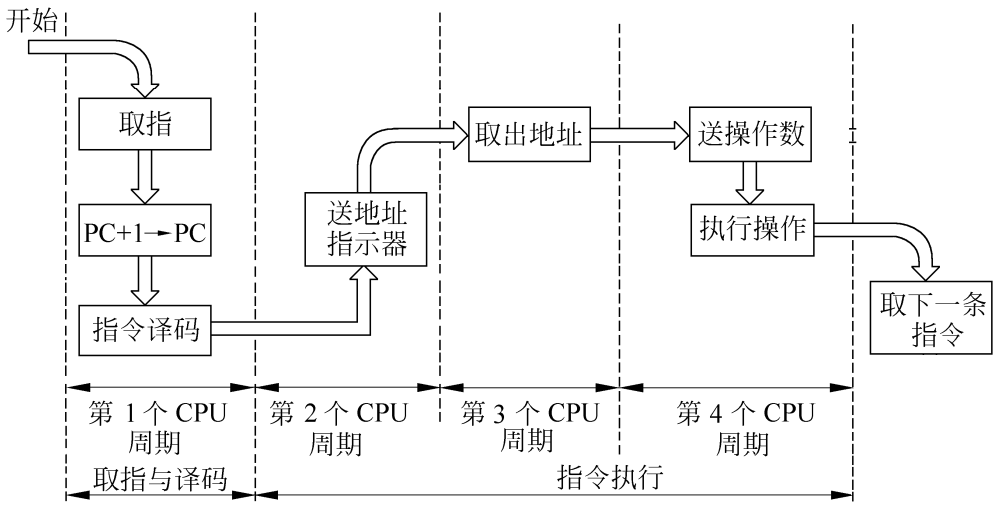


图 6-13　STA 指令的指令周期

（3）取操作数的地址并送往地址寄存器

STA 指令的第 3 个 CPU 周期主要完成从内存单元地址为 31 的单元中取出操作数地址 40，其具体操作如图 6-14 所示。操作步骤如下：

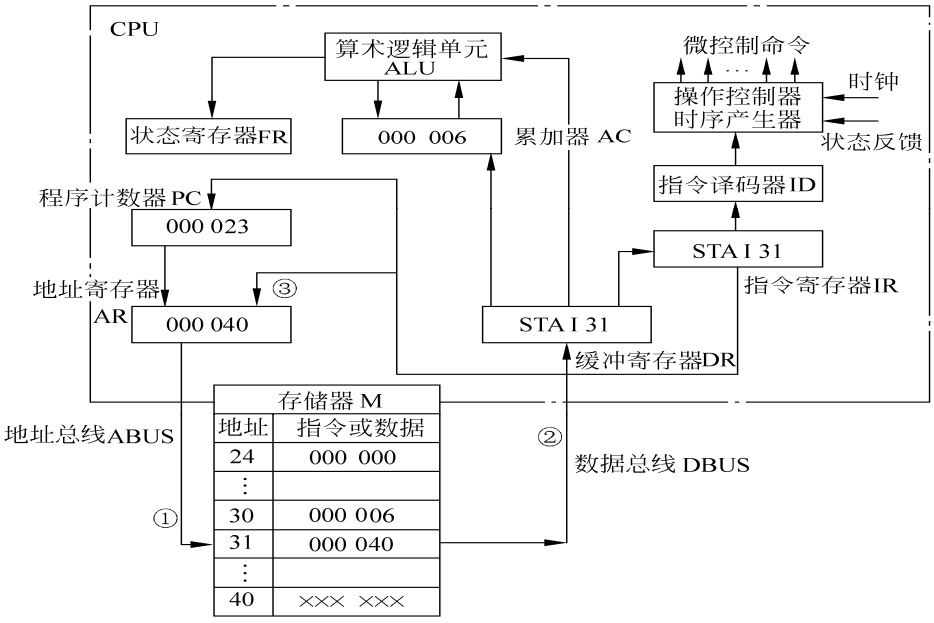


图 6-14　取操作数的地址并送往地址寄存器

①地址寄存器的内容 31 发送到地址总线；　（AR→ABUS）

②CPU 发出读命令，将选中的地址为 31 内存单元的内容 40 读到数据总线上；（$R/\overline{W}=1$）

③把数据总线上的内容装入地址寄存器 AR。于是 40 进入地址寄存器，替代原来的内容 31。（DBUS→AR）

（4）累加器 AC 的内容写入内存单元

STA 指令的第 4 个 CPU 周期主要完成累加器 AC 的内容写入内存单元。其具体操作如图 6-15 所示。操作步骤如下：

①累加器 AC 的内容 6 送给数据缓冲寄存器 DR；（AC→DR）

②把地址寄存器 AR 的内容 40 送到地址总线上；（AR→ABUS）

③把数据缓冲寄存器 DR 的内容 6 发送到数据总线；（DR→DBUS）

④ CPU 发出写命令，将数据总线上的内容写入到所选的主存单元中，即将数据 6 写入地

址为 40 的主存单元。（$R/\overline{W}=0$）

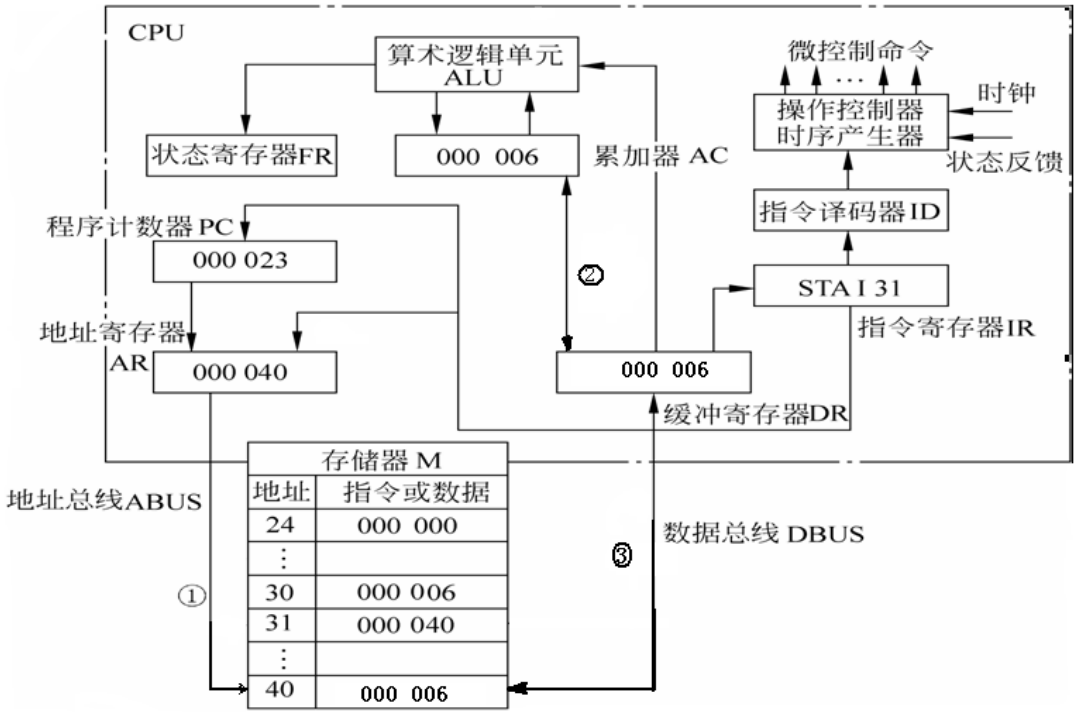


图 6-15　累加器 AC 的内容写入内存单元

4、JMP 指令周期

JMP 指令可以是直接寻址，也可以是间接寻址。在本例中，JMP 指令采用直接寻址，该指令的指令周期需要两个 CPU 周期，第一个 CPU 周期完成取指和译码操作，第二个 CPU 周期将指令寄存器 IR 中指令的地址码送往地址寄存器 AR 和程序计数器 PC。JMP 指令的指令周期如图 6-16 所示。

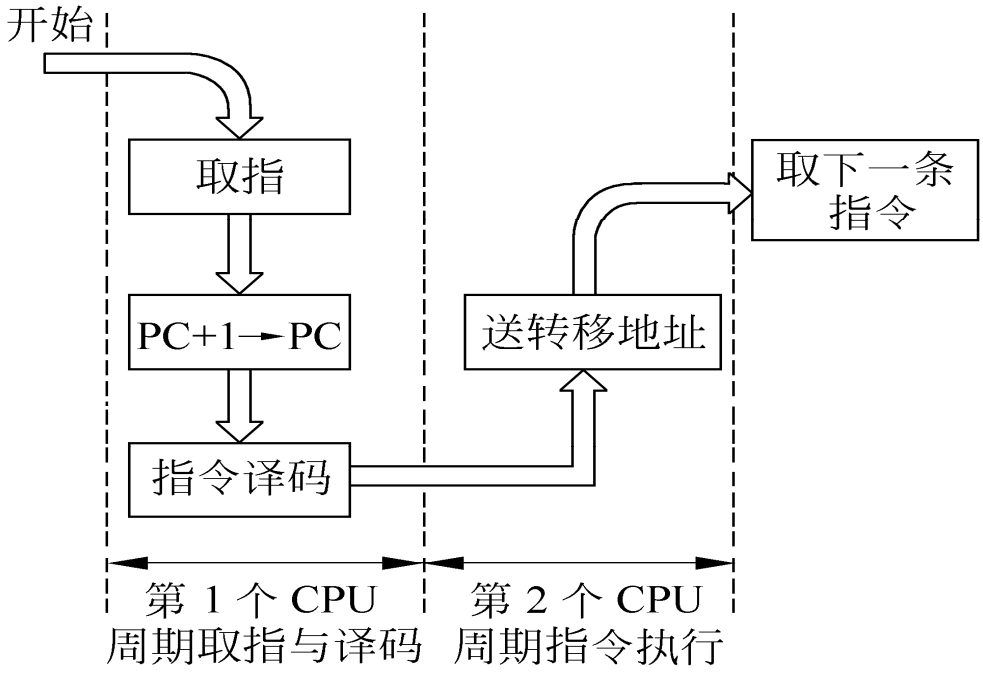


图 6-16　JMP 指令的指令周期

（1）取指周期

JMP 指令的第一个 CPU 周期仍为取指周期，即将当前 PC 所指的主存单元中的指令“JMP 21”（机器码为“140 021”）取出并译码，PC 的值加 1。由于 JMP 指令的第一个 CPU 周期与 CLA、ADD、STA 指令的第一个 CPU 周期的操作完全相同，所以在此不再重复。

（2）指令寄存器 IR 中的地址码装入地址寄存器 AR 和程序计数器 PC（IR_{Addr}→AR、PC）

JMP 指令的第二个 CPU 周期将指令寄存器 IR 中的地址码（021）装入地址寄存器 AR 和程序计数器 PC（IR_{Addr}→AR、PC）。类似于 ADD、STA 指令的第二个 CPU 周期，所不同的是，指令寄存器 IR 中的地址码（021）不仅装入地址寄存器 AR，还要装入程序计数器 PC，从而代替了 PC 原来的内容 24，这样，下一条 CPU 将要执行的指令不是 24 单元中的指令，而是 21 单元中的指令，从而改变了程序原来的执行顺序，实现了程序的转移。

5、NOP 指令周期

NOP 指令是一条空操作指令，也是非访内指令，此指令的指令周期由两个 CPU 周期组成，第一个 CPU 周期完成取指和译码操作，第二个 CPU 周期用作指令的执行操作。由于 NOP 指令是一条空操作指令，所以，在第二个 CPU 周期中不产生任何控制信号。NOP 指令的指令周期与 CLA 指令的指令周期相同，在此不再重复。

6.3.2 指令周期流程

通过分析典型指令的指令周期，对每一条指令的取指过程和执行过程有了较为深刻的印象。为了便于进行控制器的设计，指令周期可以采用类似程序流程图的形式进行描述。指令周期流程主要由方框、棱形框、有向线段和公共操作符～组成。其中，一个方框代表一个 CPU 周期，表示某些具体的操作；棱形框表示某种判断或测试，不单独占用一个 CPU 周期；有向线段表示时间的先后顺序；公共操作符～表示一条指令执行结束，转入公操作。由于所有指令的取指令阶段完全相同，并且是指令的第一个 CPU 周期，因此，取指令可以作为公操作。指令不同只是指令的执行阶段不同，可以根据指令的操作码转向不同指令的执行阶段。我们把 CLA、ADD、STA、JMP 四条典型指令的指令周期进行归纳，划出了指令周期流程图。如图 6-17 所示。

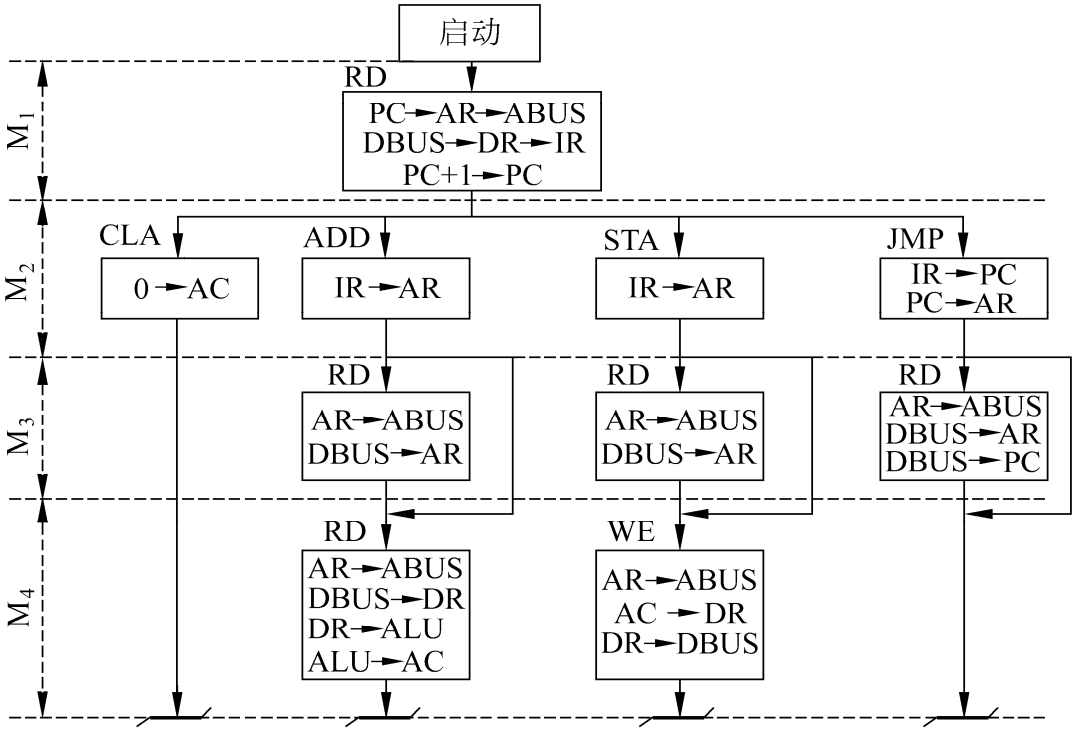


图 6-17 指令周期流程图

图 6-17 中，对于 ADD、STA、JMP 指令，由于指令的寻址方式不同，可以采用直接寻址，也可以采用间接寻址，因此，指令流程图出现了分支，请读者来分析哪一个分支是直接寻址，哪一个分支是间接寻址。

【例 6-1】图 6-18 所示为双总线结构机器的数据通路，IR 为指令寄存器，PC 为程序计数器（具有自增功能），M 为主存（受 R/$\overline{W}$ 信号控制），AR 为地址寄存器，DR 为数据缓冲寄存器，ALU 由加、减控制信号决定完成何种操作，控制信号 G 控制的是一个门电路。另外，线上标注有小圈表示有控制信号，例中 y_i 表示 y 寄存器的输入控制信号，R_{1o} 为寄存器 R_1 的输出控制信号，未标字符的线为直通线，不受控制。

（1）画出“ADD　R_0，R_2 ”的指令周期流程图，并列出相应的操作控制信号序列。假设该指令的地址已放入 PC 中。

（2）画出“SUB　R_1，R_3 ”的指令周期流程图，并列出相应的操作控制信号序列。假设该指令的地址已放入 PC 中。

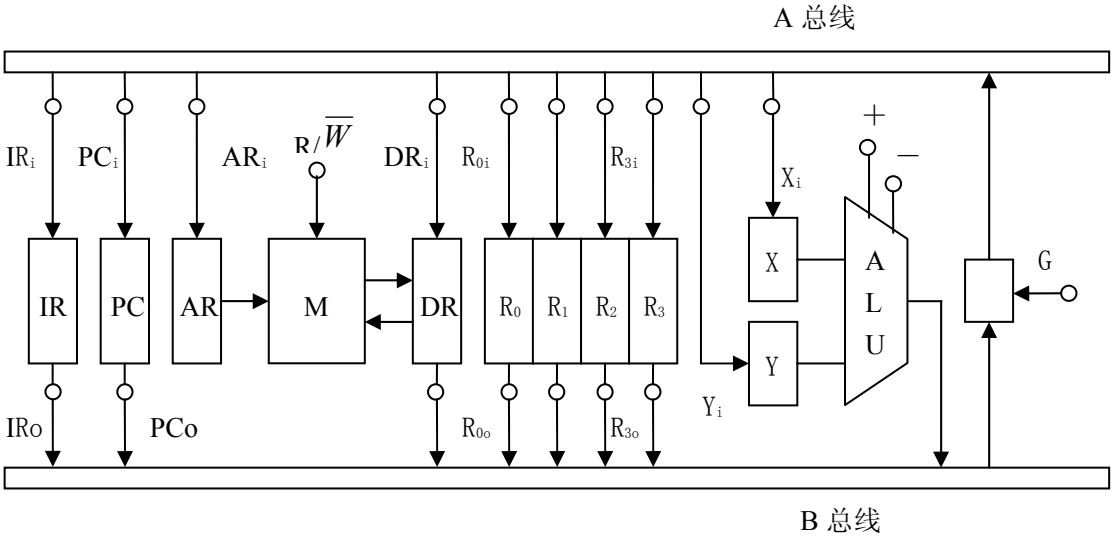


图 6-18 双总线结构机器的数据通路

解：（1）“ADD　R_0，R_2 ”指令是一条加法指令，其功能为（R_0）+（R_2）→R_0，根据给定的数据通路，“ADD　R_0，R_2 ”的指令周期流程如图 6-19（a）所示，该图的右侧注明了相应的操作控制信号序列。

（2）“SUB　R_1，R_3 ”指令是一条减法指令，其功能为（R_1）−（R_3）→R_3，其指令周期流程如图 6-19（b）所示，该图的右侧注明了相应的操作控制信号序列。

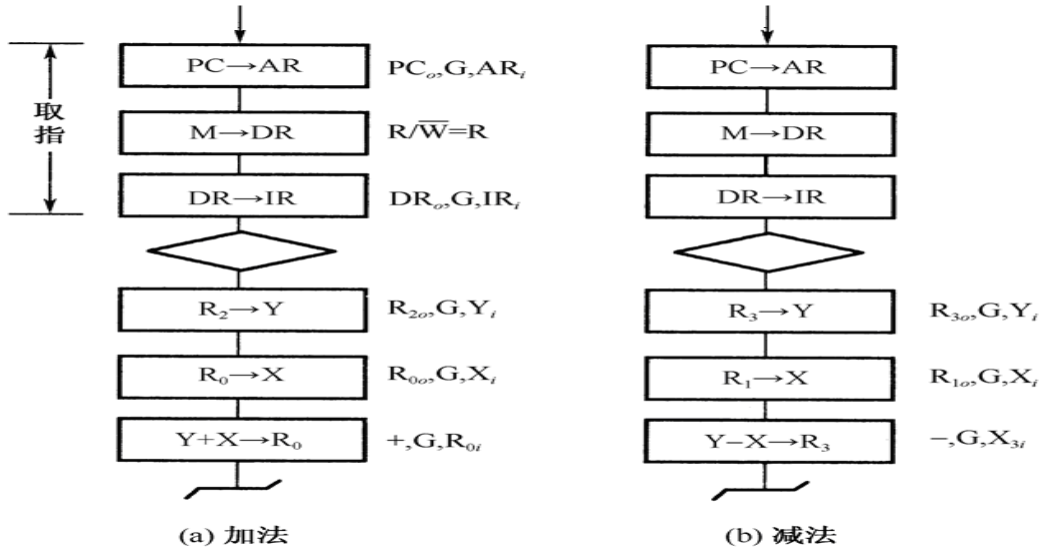


图 6-19 指令周期流程图

6.4 微程序控制器

微程序控制器具有规整性、灵活性、可维护性等一系列优点而被广泛应用。早在 1951 年，英国剑桥大学的 M.V.Wilkes 教授提出了微程序设计，微程序设计的实质是用程序设计的思想方法来组织操作控制逻辑，把各条指令的微操作序列编制成微程序，并存放在控制存储器中，执行机器指令时，通过读取并执行相应的微程序来实现这条机器指令的功能。

6.4.1 基本概念

1、微命令和微操作

我们可以把组成计算机的各个部分划分为两大类，一类是控制部件，另一类是执行部件。控制器就是控制部件，而运算器、存储器、外围设备相对控制器来讲，就是执行部件，控制部件与执行部件之间的联系依靠的是控制信号线，我们通常把控制部件通过控制信号线向执行部件发出各种控制命令称为微命令，由此可见，微命令是构成控制信号序列的最小单位。而执行部件接受微命令后所进行的操作称为微操作，微操作是执行部件中最基本的操作。微操作可以分为相容性和相斥性两种。所谓相容性的微操作，是指在同时或同一个 CPU 周期内可以并行执行的微操作。所谓相斥性的微操作，是指不能在同时或不能同一个 CPU 周期内可以并行执行的微操作。

图 6-20 所示给出了一个简单运算器模型，其中 ALU 为算术逻辑单元，R_1、R_2、R_3 为三个寄存器，三个寄存器的内容可以通过多路开关从 ALU 的 X 输入端或 Y 输入端送至 ALU，而 ALU 的输出可以送往任何一个寄存器或同时送往 R_1、R_2、R_3 三个寄存器。多路开关的每个控制门是一个常闭的开关，它的一个输入端代表来自寄存器的信息，而另一个输入端是控制端，一旦两个输入端都有输入信号，多路开关则处于开的状态，才会产生输出。图中每一个开关门都有相应的微命令来控制，例如，开关门 4 由编号为 4 的微命令控制，开关门 5 由编号为 5 的微命令控制，如此等等。三个寄存器 R_1、R_2、R_3 分别由 1、2、3 微命令控制，以便在 ALU 运算完毕时，将运算结果打入到某一寄存器。另外，ALU 只有+、-和 M 三种操作。Cy 为进位触发器，只有 ALU 产生进位时该触发器的状态为“1”。

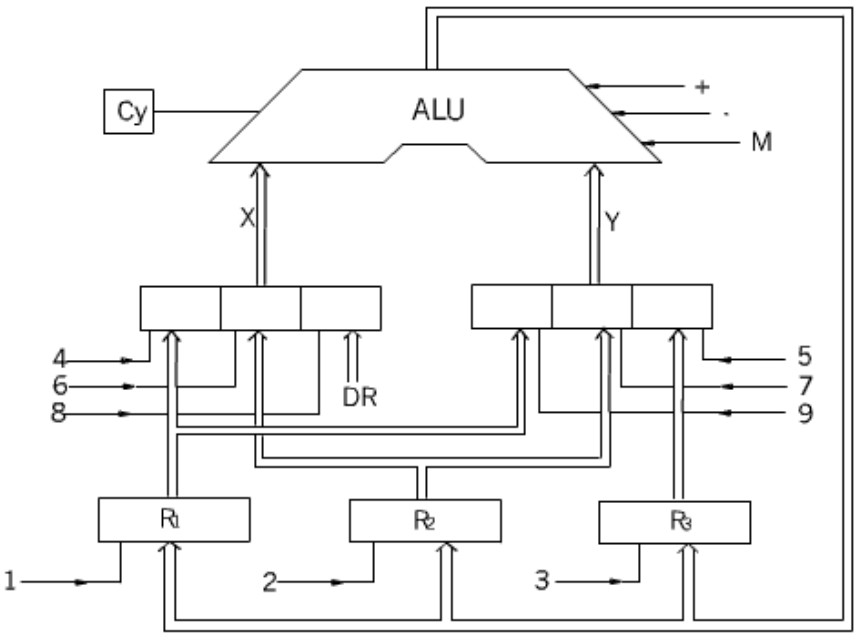


图 6-20　简单运算器模型

由此可见，ALU 的操作（+、-、M）在同一个 CPU 周期中只能选择一种，所以，+、-、M 三个微操作是相斥性的微操作。另外，4、6、8 三个微操作是相斥性的微操作，5、7、9 三个微操作也是相斥性的微操作。微操作 1、2、3 是可以同时进行的，所以是相容性的微操作。另外，ALU 的 X 输入微操作 4、6、8 分别与 Y 输入的微操作 5、7、9 任意两个微操作都是相容性的微操作。

2、微指令和微程序

微指令是指在一个 CPU 周期中能够实现一定操作功能的一组微命令。而微程序则是微指令序列。我们知道一条机器指令的指令周期包含多个 CPU 周期，根据微指令的概念，一个 CPU 周期对应一条微指令，因此，一条机器指令的功能是由多条微指令组成的序列来实现的。作为微指令，一方面要形成后继微地址，即执行完某一条微指令后，必须给出下一条微指令的地址，下一条微指令的地址也称为后继微地址，以便当前微指令执行完毕后能正确取出下一条微指令；另一方面要产生一组微命令。因此，微指令的格式应至少包含操作控制字段和顺序控制字段两部分。操作控制字段用来产生微命令，顺序控制字段用来形成后继微地址。某一具体微指令格式如图 6-21 所示。其微指令长度为 23 位，包含操作控制字段和顺序控制字段两部分。其中操作控制字段占 17 位，每一位表示一个微命令。当操作控制字段某一位为“1”时，则表示发出该位相应的微命令；否则，则表示不发出该位相应的微命令。例如，当微指令第一位为“1”时，表示发出 LDR'_1 微命令，运算器执行 ALU→R_1 的微操作。同样，当微指令第 10 位为“1”时，表示向 ALU 发出“+”微命令， ALU 则执行“+”微操作。

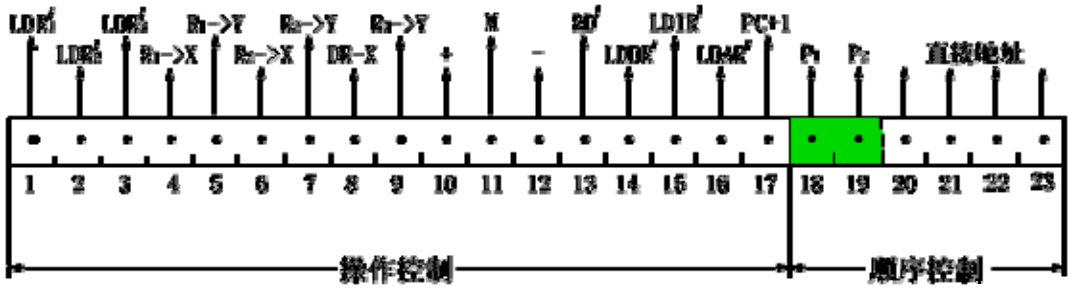


图 6-21 微指令格式

微指令的顺序控制字段占 6 位。其中 4 位（20～23）用来直接形成后继微地址，第 18、19 位两位作为判断测试标志。当此两位都为“0”时，表示不进行测试，直接将微指令中第 20～23 位作为后继微地址；当微指令的第 18 位或第 19 位为“1”时，表示要进行 P_1 或 P_2 的判断测试，根据测试结果，需要对微指令中第 20～23 位的某一位或某几位进行修改，将修改后的地址作为后继微地址。

3、机器指令与微指令

机器指令与微指令的关系如图 6-22 所示。

（1）机器指令与微指令所处的位置不同

机器指令在程序中，位于主存储器内，该机器指令的第一个 CPU 周期完成取指和译码，即将机器指令从主存中取出，并送到指令寄存器 IR 中，再对 IR 中的指令操作码进行译码，从而识别出是什么机器指令。而微指令在微程序中，位于控制存储器内，在一个微周期中完成从控制存储器取出微指令，并存放于微指令寄存器 μIR，再执行 μIR 中的微指令。

（2）机器指令与微程序的对应关系

一条机器指令对应一个微程序，也就是说，一条机器指令是由一个微程序负责解释执行的。

在控制存储器中存在着所有机器指令的微程序，如何实现不同的机器指令转去执行相应的微程序呢？在机器指令的第一个 CPU 周期完成取指和译码后，由于机器指令不同，其指令的操作码也不同，因此，可以根据指令寄存器 IR 中的指令操作码去产生该机器指令相应的微程序入口地址。具体而言，可以把机器指令的操作码直接作为该机器指令相应的微程序入口地址，也可以通过映像存储器来实现操作码与微程序入口地址之间的转换。产生出该机器指令相应的微程序入口地址后，将该微程序入口地址送到微地址寄存器 μAR，从而实现机器指令正确转移到相应的微程序。

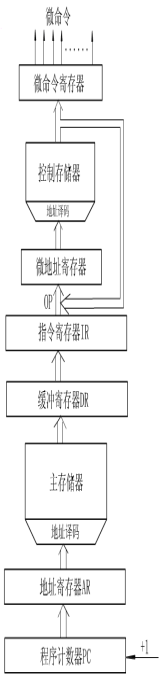


图 6-22　机器指令与微指令的关系

（3）公共的取指微指令

由于所有机器指令的第一个 CPU 周期所进行的操作都完全相同，结合微指令的定义可知，一个 CPU 周期对应一条微指令，因此，所有机器指令相应微程序的第一条微指令完全相同，为了节约控制存储器的存储容量，将微程序的第一条微指令（即取指微指令）设置为公共微指令，任何一个微程序执行完毕后都转到该公共的取指微指令，以完成下一条机器指令的读取操作。

6.4.2 微程序控制器基本原理

1、微程序控制器的组成

微程序控制器的组成与结构如图 6-23 所示。它主要由控制存储器、微指令寄存器和微指令地址形成部件三大部分组成。

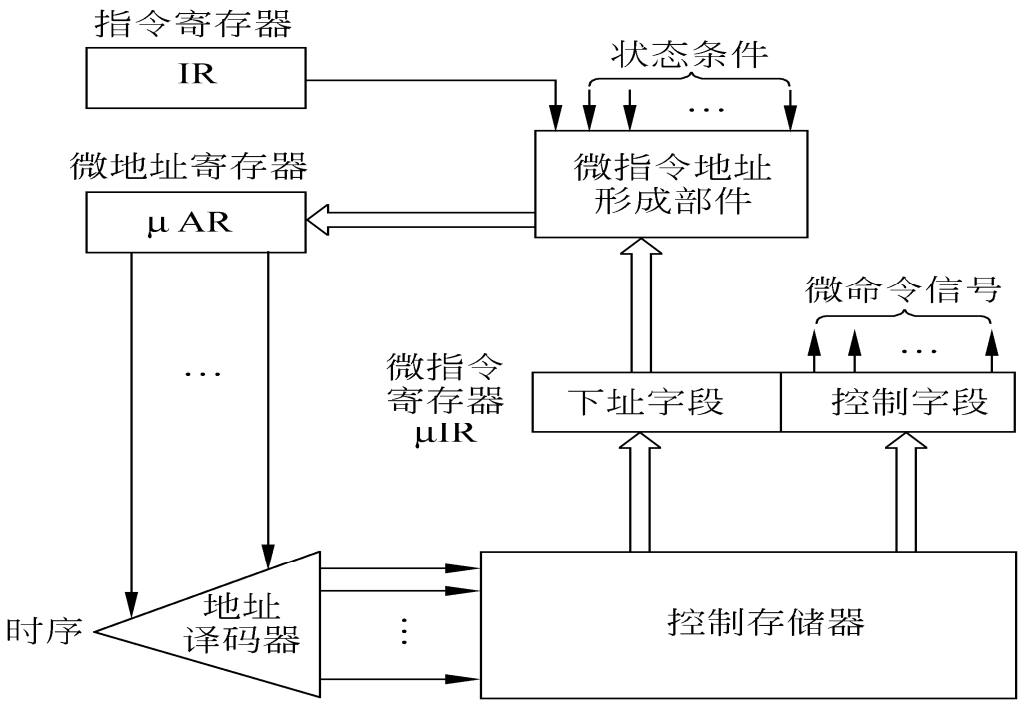


图 6-23 微程序控制器的组成与结构

（1）控制存储器（CM）

控制存储器是用来存放指令系统所对应的全部微程序，它是一种读出时间较快的只读存储器，其容量视指令系统而定，其字长由控制命令的多少、微指令的编码格式以及下址字段

的宽度而定。

（2）微指令寄存器（μIR）

微指令寄存器是用来存放从控制存储器读出的一条微指令。微指令由操作控制字段和顺序控制字段构成，其中顺序控制字段用来产生将要执行的下一条微指令的地址，操作控制字段则用来产生一组微命令。

（3）微指令地址形成部件

微指令地址形成部件又称微指令地址发生器或后继微地址形成部件，是用来形成将要执行的下一条微指令的地址（简称后继微地址）。一般情况下，下一条微指令的地址由上一条微指令的顺序控制字段直接决定。当微程序出现分支时，将由状态条件的反馈信息去形成转移地址；当取指令公共操作完成后，可以根据指令的操作码去产生微指令入口地址。图 6-24 仅给出了上述三种后继微地址的形成方式，根据计算机规模不同，还可以有更多的后继微地址形成方式。

2、微程序的执行过程

微程序的执行过程类似于 CPU 对程序的执行，一方面要控制微程序中微指令的执行顺序，另一方面要控制执行微指令，其具体过程如下：

（1）读取并执行公共的“取指令”微指令

从控制存储器中取出一条公共的“取指令”微指令，并送到微指令寄存器 μIR。由于这是一条公用的微指令，一般存放在控制存储器的 0 号或 1 号地址单元。CPU 在执行微指令寄存器 μIR 中的微指令时，操作控制字段产生相关的微命令，这些微命令实现从主存中读取机器指令并将其送到指令寄存器 IR。

（2）形成微程序入口地址

根据指令寄存器 IR 中的指令操作码，通过微地址形成线路产生相应的微程序入口地址，并将微程序入口地址送往微地址寄存器 μAR。

（3）执行微程序

根据微地址寄存器 μAR 中的微地址，取出微程序中第一条微指令并送入微指令寄存器 μIR，开始执行 μIR 中的微指令，操作控制字段产生微命令，顺序控制字段形成后继微地址。该微指令执行结束时，将顺序控制字段所形成的后继微地址再送入微地址寄存器 μAR，去读取下一条微指令。如此重复，直至微程序中的最后一条微指令。

（4）实现返回

执行完一条机器指令对应的一段微程序后，返回 0 号或 1 号微地址单元，读取并执行“取指令”微指令。

由此可见，微程序控制器的工作过程涉及到两个层面：一个层面是程序员所看到的传统机器级，包括指令、程序、主存储器；另一个层面是设计者所看到的微程序级，包括微指令、微程序、控制存储器（相对程序员是“透明”的）。

6.4.3 微程序设计

微程序设计的关键是如何设计微指令的结构，与微指令结构相关的因素较多，除机器硬件外，还要考虑如何缩短微指令字的长度，如何提高微程序的执行速度，如何有利于对微指令的修改，以便提高微程序设计的灵活性。

1、微命令编码

微命令编码是指对微指令中的操作控制字段所进行的编码，其编码方法有直接表示法、编码表示法、混合表示法。

（1）直接表示法

操作控制字段中的每一位代表一个微命令的编码方法称为微命令的直接表示法。直接表示法对微命令不需要译码。其优点是简单、直观，输出可直接用于控制，一条微指令可以定

义并执行多个并行的微命令。微指令的直接表示法如图 6-24 所示。显然，当微指令中的微命令增多时，会导致微指令字加长，使控制存储器的容量加大。因此，直接表示法只适用于微命令数量不多的 CPU。对于微命令较多的 CPU 而言，常采用编码表示法。

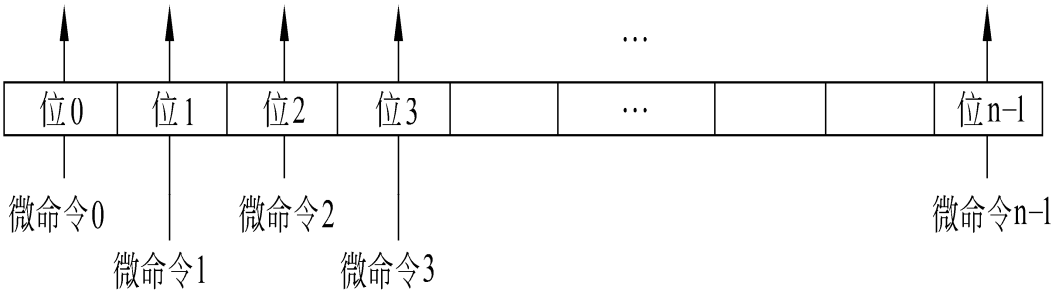


图 6-24 微命令直接表示法

（2）编码表示法

编码表示法把一组相斥性的微命令组成一个字段，然后通过微命令译码器对每一个字段进行译码，译码输出作为微命令。编码表示法的微指令结构如图 6-25 所示。采用编码表示法，可以用较少的二进制信息位表示较多的微命令。例如，某一个 3 位的二进制信息字段，经译码后可以表示 8 个微命令；4 位的二进制信息字段，经译码后可以表示 16 个微命令。与直接表示法相比，编码表示法可使微指令字大大缩短，但由于增加了译码电路，编码表示法比直接表示法的速度要慢一些。

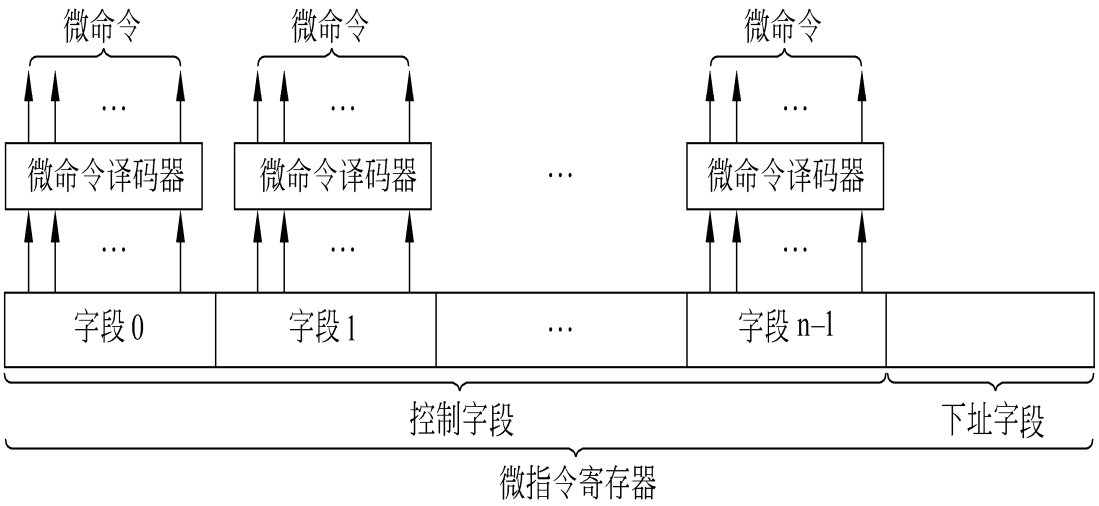


图 6-25 微命令的编码表示法

（3）混合表示法

混合表示法把直接表示法和编码表示法相混合使用，以便能综合考虑微指令字长、灵活性和执行速度等方面的要求。

另外，在微指令中还增加一个常数字段，该字段可作为操作数送入 ALU 运算，也可作为计数器的初值用来控制微程序循环次数。

2、后续微地址的形成方法

微指令执行的顺序控制问题，实际上是如何确定下一条微指令的地址问题。通常，产生后续微地址的方法有计数器方式、增量方式与断定方式结合、多路转移方式三种。

（1）计数器方式

这种方式与用程序计数器 PC 来产生后续指令地址的方法类似。计算机加电后执行的第一条微指令地址（微程序入口）来自专门的硬件电路，控制实现取指令操作，然后由指令操作码产生后续微地址。若是顺序执行微指令，则将现行微地址（在微地址计数器 μPC 中）

加 1 产生后续微地址；若遇到转移类微指令，则由 μPC 与形成转移微地址的逻辑电路组合成后续微地址，例如利用该逻辑电路的输出与 μPC 低位进行逻辑加，形成后续微地址。这种方式可使微指令的下址字段很短，仅起选择作用。其缺点是微程序转移不灵活，使得微程序在控制存储器中的空间分配较困难。计数器方式的原理如图 6-26 所示。

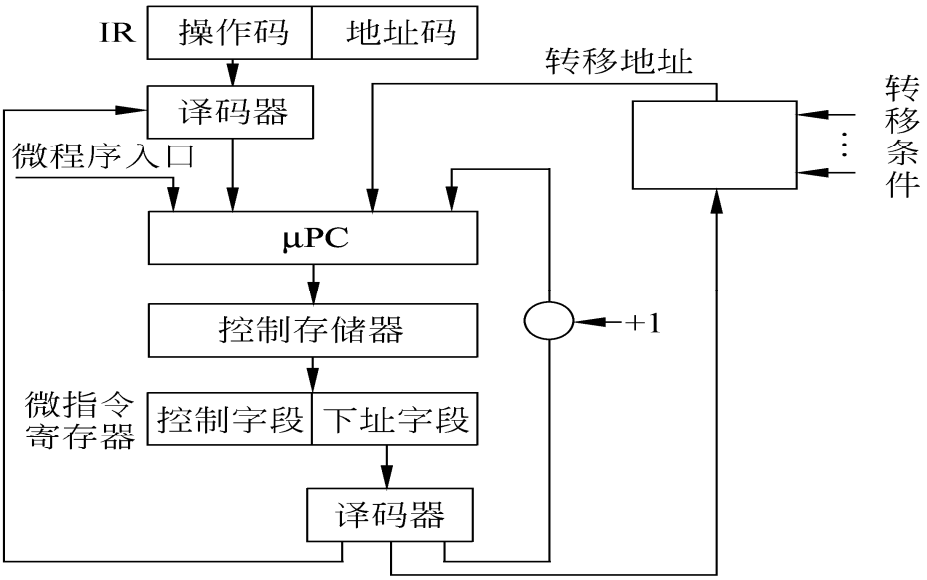


图 6-26 计数器方式

在图 6-26 中，μPC 兼作控制存储器的地址寄存器，其输入有 4 个。下址字段仅有两位，其功能是选择 3 个输入源的一个作为 μPC 的输入，而微程序入口是由专门的硬件电路产生，不受下址字段控制。

计数器方式的基本特点是：微指令的顺序控制字段较短，微地址产生机构简单。但是多路并行转移功能较弱，速度较慢，灵活性较差。

（2）增量方式与断定方式的结合

在这种方式中，微指令顺序控制字段又分为条件选择字段和转移地址字段两部分，其中条件选择字段用来规定“条件转移”微指令要测试的外部条件；转移地址字段可用作后继微地址，当转移条件满足时，用它作下一个微地址，即将“转移地址”送给微程序计数器 μPC；当转移条件不满足时，则使用微程序计数器 μPC 提供下一条微指令的地址。增量方式与断定方式结合形成后续微地址的原理图如图 6-27 所示。

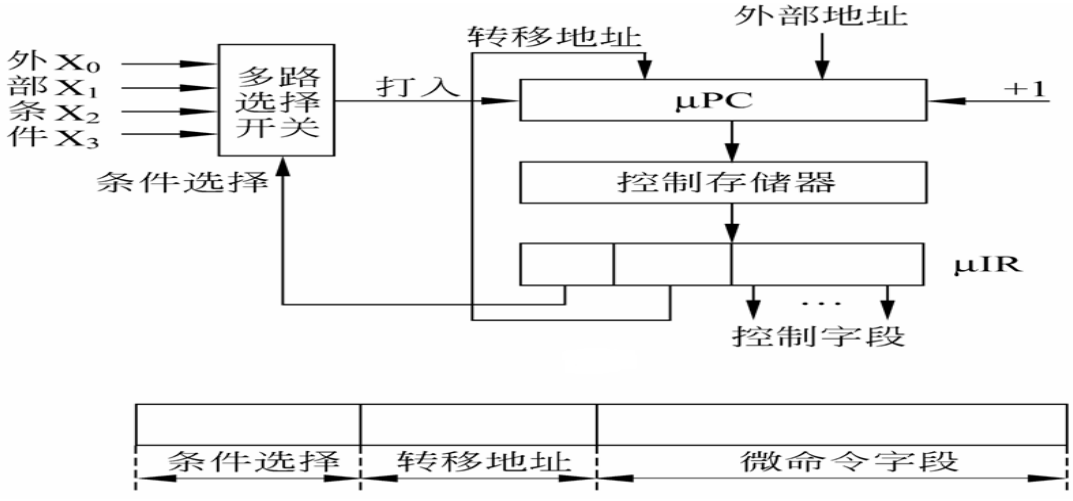


图 6-27 增量方式与断定方式结合形成后续微地址的原理图

μPC 是微程序计数器，具有计数和并行接受数据的功能；μIR 是微指令寄存器。当 μIR 中的转移条件（条件选择）字段指出一次转移时，微指令“转移地址”字段的内容就被送入 μPC。“条件选择”字段用来控制一个多路开关，根据外部的状态条件信息，多路开关将所选的某一路数据并行打入 μPC 中。

假设必须测试的两个状态条件变量为 V_1 和 V_2，故需使用一个 2 位的条件选择字段 S_1S_0：①当 S_1S_0 =00 时，微程序不转移。②当 S_1S_0 =01 时，如果 V_1=1 则转移，否则顺序执行。③当 S_1S_0 =10 时，如果 V_2=1 则转移，否则顺序执行。④当 S_1S_0 =11 时，无条件转移。与此对应，多路开关有 4 个输入 X_0、X_1、X_2、X_3，其中 X_0=0，X_1=V_1，X_2=V_2，X_3=1。因此，当 S_1S_0 =i 时，它选通多路开关输出 X_i。从而控制“转移地址”字段的内容送入或不送入 μPC。

（3）多路转移方式

在执行一条微指令时，可能会遇到从若干个微地址中选择一个作为后续微地址的情况，这种转移方式称为多路转移。例如，微指令在执行时，有时要根据某些硬件状态来决定后续微地址，属于这些状态的可以是根据运算结果所置的标志位、计数器状态、数据通道状态等。一种状态（0 或 1）可以用来选择两个微地址之一，即两路转移；而两种状态可以用来选择 4 个微地址之一，即四路选择。微程序设计实践表明，实现两路转移的情况较多，其次是四路转移，而四路以上转移的情况比较少见。

两路转移只涉及微地址的一位；四路转移涉及微地址的两位，一般就定在微地址的最后两位，也就是说，当执行转移微指令时，根据条件可转移到 4 个微地址中的一个，这 4 个微地址的高位部分相同，仅是最低两位不同。实现多路转移可以减少微指令的长度，对于一般条件转移微指令（相当于两路转移）来说，需要两条微指令来完成上述四路转移的功能。多路转移方式的特点是： 能与较短的顺序控制字段配合，实现多路并行转移，灵活性好，速度快，但转移地址逻辑需要用组合逻辑方法实现。

3、微指令的格式

微指令的格式大体分成两类：水平型微指令和垂直型微指令。

（1）水平型微指令

水平型微指令是指一次能定义并执行多个并行操作微命令的微指令。其一般格式如下：

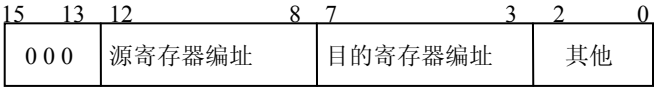
控制字段	条件测试字段	下址字段

按照控制字段的编码方法不同，水平型微指令可分为直接表示水平型微指令、字段译码水平型微指令和混合表示水平型微指令，这些微指令在介绍微命令表示法时已作叙述，在此不作重复。

（2）垂直型微指令

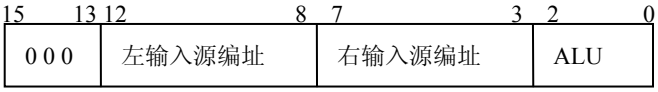
微指令中设置微操作码字段，采用微操作码编译法，由微操作码规定微指令的功能 ，称为垂直型微指令。其格式类似于机器指令，它有操作码，在一条微指令中只有 1～2 个微命令，每条微指令的功能简单，因此，实现一条机器指令的微程序要比水平型微指令编写的微程序长得多，它是采用较长的微程序结构去换取较短的微指令结构。下面介绍 4 条垂直型微指令。设微指令字长为 16 位，微操作码 3 位。

1）寄存器—寄存器传送型微指令



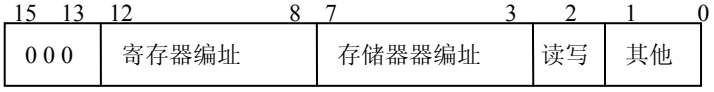
其功能是把源寄存器中的数据传送到目标寄存器。15～13 位为微操作码，源寄存器和目标寄存器编址各 5 位，可指定 32 个寄存器。

2）运算控制型微指令



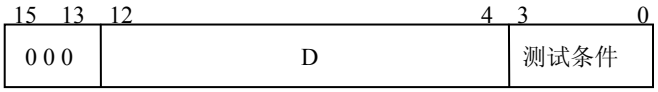
其功能是选择 ALU 的左、右两个输入源，按 ALU 字段所指定的运算功能(8 种操作)进行处理，并将结果送入暂存器中。左、右输入源编址可指定 32 种信息源之一。

3）访存微指令



其功能是将主存中一个单元的信息送入寄存器或者将寄存器的数据送往主存。存储器编址是指按规定的寻址方式进行编址。第 2 位指定读操作或写操作

4）条件转移微指令



其功能是根据测试条件决定是转移到D所指定的微地址单元，还是顺序执行下一条微指令。9 位 D 字段不足以表示一个完整的微地址，但可以用来替代现行 μPC 的低位地址。 测试条件字段有 4 位，可规定 16 种测试条件。

由此可见，水平型微指令和垂直型微指令各有所长。第一，水平型微指令并行操作能力强，效率高，灵活性强，而垂直型微指令则较差；第二，水平型微指令执行一条指令的时间短，而垂直型微指令执行时间长；第三，水平型微指令编写的微程序较短，而垂直型微指令编写的微程序较长；第四，水平型微指令用户难以掌握，而垂直型微指令与机器指令相似，相对来说，比较容易掌握。

6.4.4 微程序设计举例

我们知道微程序设计的关键是设计微指令的结构，而微指令的结构设计又取决于控制器和运算器的结构。在此我们以图 6-20 所示的运算器、图 6-21 所示的微指令格式为例，来具体说明 “十进制加法”指令的微程序设计过程。

1、十进制加法指令的功能

十进制加法指令的功能是用 BCD 码来完成一位十进制数的加法运算。当两个一位十进制数进行相加运算时，如果和数大于 9，

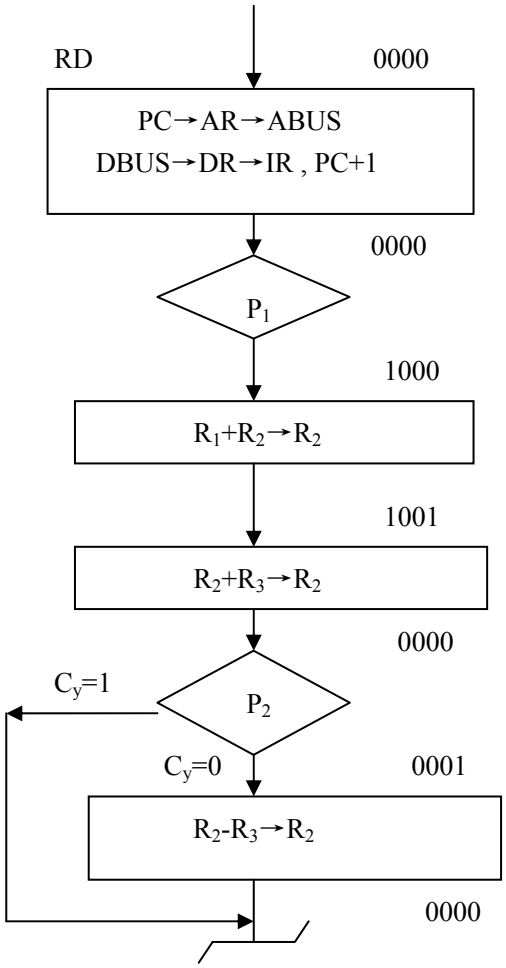


图 6-28 十进制加法指令的微程序流程

其运算结果是错误的，必须对错误的结果进行加 6 修正，方可得到正确的结果；如果和数小于或等于 9，十进制运算的结果是正确的，无须进行修正。

2、十进制加法指令的微程序流程图

假设两个一位十进制数 a 和 b 已存放在图 6-20 中的 R_1 和 R_2 寄存器中，数 6 存放在 R_3 寄存器中。其算法采用先进行 a+b+6 运算，再判断结果有无进位，若有进位 C_y=1，不减 6；若无进位 C_y=0，减去 6。因此，完成十进制加法指令的微程序流程如图 6-28 所示。

3、十进制加法指令的微程序

根据图 6-28 可知，十进制加法指令的微程序由 4 条微指令组成。

第一条微指令是完成取指操作，该微指令所在控制存储器该微指令所在控制存储器的地址为 0000, 其编码为：

000 000 000 000 11111	10	0000

这条微指令在执行时，一方面操作控制字段产生 5 个微命令，他们分别是 LDAR′、RD′、LDDR′、LDIR′、PC+1。LDAR′执行 PC→AR；PC+1 执行 PC+1→PC；RD′和 LDDR′执行 M→DR；LDIR′执行 DR→IR。假设十进制加法指令的操作码为 1000，那么指令寄存器 IR 中的指令操作码是 1000。另一方面，顺序控制字段指明下一条微指令的地址为 0000。由于判断字段中 P_1=1，表示测试的“状态条件”是指令寄存器 IR 中的指令操作码 1000，即用指令操作码 1000 作为下一条微指令的地址，因此，此微指令的下址 0000 并不是下一条微指令的真正地址。下一条微指令的真正地址是指令的操作码 1000，于是，微地址寄存器 μAR 的内容修改为 1000。

第二条微指令是完成 a+b 运算操作，该微指令的微地址为 1000，其编码为：

010 100 100 100 00000	00	1001

根据微地址寄存器 μAR 的内容 1000，取出第二条微指令并执行。执行时，一方面操作控制字段产生 4 个微命令，他们分别是 R_1→X、R_2→Y、+、LDR_2′。于是，运算器完成 R_1+R_2→R_2 的操作。另一方面，顺序控制字段中判断测试字段 P_1 和 P_2 均为 0，表示不进行测试，于是，该微指令的下址字段 1001 直接作为下一条微指令的地址，并传送到微地址寄存器 μAR。

第三条微指令是完成 a+b+6 运算操作，该微指令的微地址为 1001，其编码为：

010 001 001 100 00000	01	0000

根据微地址寄存器 μAR 的内容 1001，取出第三条微指令并执行。执行时，一方面操作控制字段产生 4 个微命令，他们分别是 R_2→X、R_3→Y、+、LDR_2′。于是，运算器完成 R_2+R_3→R_2 的操作。另一方面，顺序控制字段中判断测试字段 P_2 为 1，表示对 P_2 进行测试，测试的“状态条件”为进位标志 C_y，根据进位标志 C_y 的状态来修改微地址寄存器 μAR 的最后一位：若 C_y=0 时，下一条微指令的地址为 0001；若 C_y=1 时，下一条微指令的地址为 0000。由此可见，该微指令的下址 0000 并不是下一条微指令的真正地址。下一条微指令的真正地址是对下址 0000 的修改。在此假设 C_y=0，则要执行的下一条微指令的地址为 0001，送到微地址寄存器 μAR。

第四条微指令是完成 a+b-6 运算操作，该微指令的微地址为 0001，其编码为：

010 001 001 001 00000	00	0000

根据微地址寄存器 μAR 的内容 0001，取出第四条微指令并执行。执行时，一方面操作控制字段产生 4 个微命令，他们分别是 R_2→X、R_3→Y、-、LDR_2′。于是，运算器完成 R_2-R_3→R_2 的操作。另一方面，顺序控制字段中判断测试字段 P_1 和 P_2 均为 0，表示不进行测试，于是，该微指令的下址字段 0000 直接作为下一条微指令的地址，并传送到微地址寄存器 μAR，按该地址取出的微指令为“取指”微指令。

如果第三条微指令进行测试时 C_y=1，那么，下一条微指令的地址为 0000，将不执行第四条微指令，而是直接由第三条微指令就转向“取指”微指令，便开始从内存中取第二条机

器指令，再转去执行该机器指令相应的微程序。

【例 6-2】设某运算器的结构如图 6-29(a)所示，其中 ALU 为 16 位算术逻辑单元，S_A、S_B 为 16 位暂存器，4 个通用寄存器由 D 触发器组成其读写控制功能见表 6-2。机器采用串行微程序控制方式，其微指令周期如图 6-29(b)所示。其中读 ROM 是从控制存储器中读出一条微指令的时间，为 1μs；ALU 做加法运算时间为 500ns；m_1 是读寄存器的时间为 500ns；m_2 是写寄存器的工作脉冲宽度为 100ns。

表 6-2　　通用寄存器读写控制功能

读控制				写控制			
R	RA_0	RA_1	选择	W	WA_0	WA_1	选择
1	0	0	R_0	1	0	0	R_0
1	0	1	R_1	1	0	1	R_1
1	1	0	R_2	1	1	0	R_2
1	1	1	R_3	1	1	1	R_3
0	×	×	不读出	0	×	×	不写入

微指令字长为 12 位，其格式如下：（未考虑顺序控制字段）

0 1	2 3	4	5	6	7	8	9	10	11
RA_0RA_1	WA_0WA_1	R	W	LDS_A	LDS_B	S_B→ALU	$\overline{S_B}$ →ALU	Reset	～

其中 RA_0RA_1：读 R_0——R_3 的选择控制

WA_0WA_1：写 R_0——R_3 的选择控制

R：寄存器读命令

W：寄存器写命令

LDS_A：打入 S_A 的控制信号

LDS_B：打入 S_B 的控制信号

S_B→ALU：传送 S_B 的控制信号

$\overline{S_B}$ →ALU：传送 $\overline{S_B}$ 的控制信号并使 ALU 最低位加 1

Reset：清暂存器 S_B 为 0 的信号

～：一段微程序结束，转入取指令的控制信号

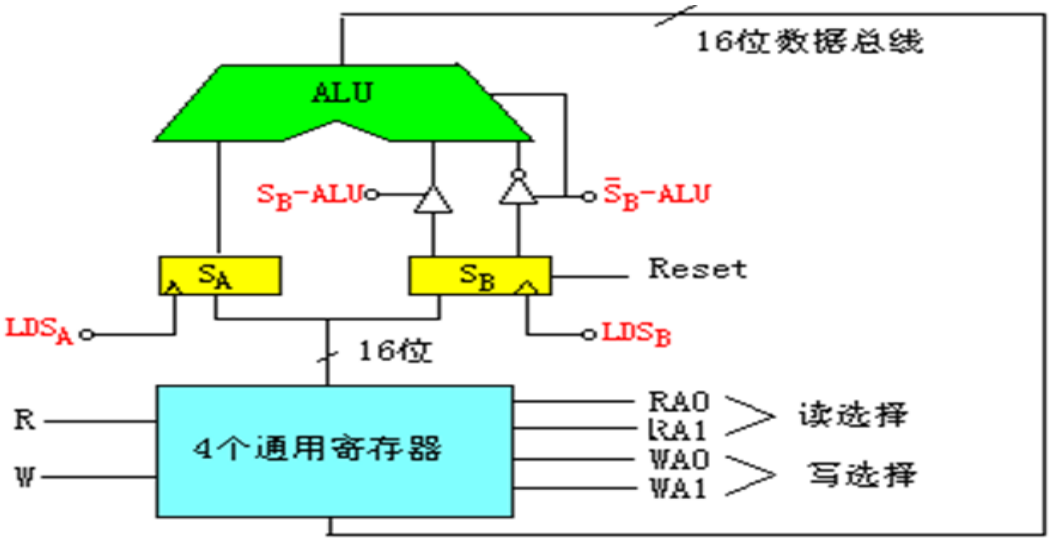


图 6-29　16 位的运算器

要求用二进制代码写出一下机器指令的微程序：

（1）ADD　R_0 ，R_1 ；（R_0）+（R_1）→R_1

（2）SUB　R_2 ，R_3 ；（R_3）—（R_2）→R_3

（3）MOV　R_2 ，R_3 ；（R_2）→（R_3）

解：首先根据机器指令的功能，画出三条机器指令的微程序流程图，如图 6-30 所示。其中未考虑公共的取指周期和顺序控制问题，在每方框的右上角仅用数字标号表示微指令的顺序。其次根据机器指令的微程序流程图，设计微程序，表 6-3 给出了三条机器指令的微程序见。其中×表示任意设置（0 或 1 均可）。可以看出：ADD 和 SUB 指令的微程序由 3 条微指令构成，MOV 指令的微程序由 2 条微指令构成。

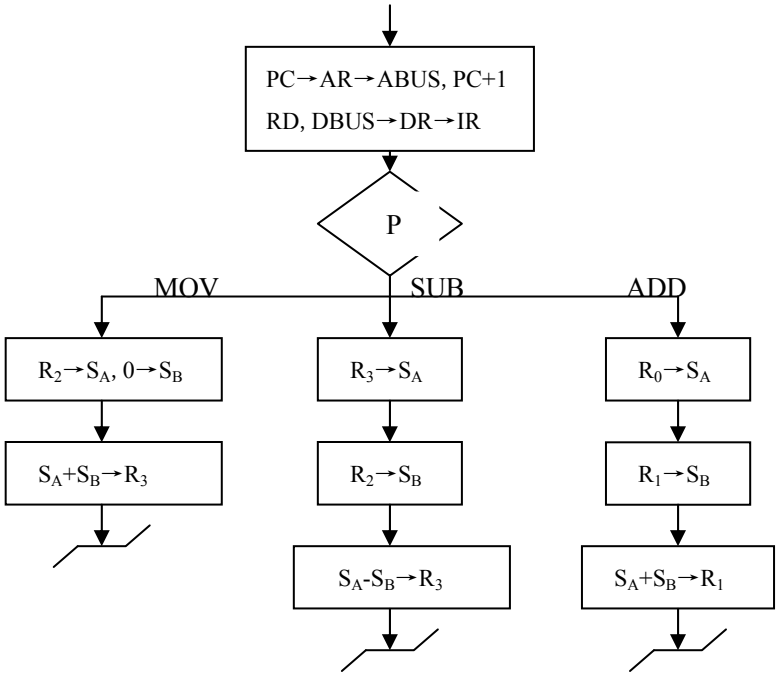


图 6-30 微程序流程图

表 6-3　　三条机器指令的微程序

机器指令	微程序的二进制代码
ADD　R_0 ，R_1	**1.00××10100000** **2.01××10010000** **3.××0101001001**
SUB　R_2 ，R_3	**4.11××10100000** **5.10××10010000** **6.××1101000101**
MOV　R_2 ，R_3	**7.10××10100000** **8.××1101001011**

6.4.5　微程序控制器设计步骤

微程序控制器设计的最主要任务是编写所有机器指令对应的微程序。具体微程序控制器设计步骤如下：

1、设计微指令的结构

设计微程序结构时，首先分析机器的组成结构，拟定微命令集，找出相容性和相斥性微命令，选择以微指令的执行方式（串行方式还是并行方式），确定微命令的编码方式和字段的划分；其次分析测试条件，以确定后续微地址的形成方法（增量方式、断定方式）。

2、画出微程序流程图

在分析所有机器指令的功能基础上，合理安排控制存储器中的微程序，画出微程序流程

图。

3、编制微程序

根据微指令格式和微程序流程图编写微程序，并对所有微程序进行优化和代码化。

4、写控制存储器

将指令系统中所有机器指令的微程序进行二进制编码，并写入控制存储器。如果控制存储器采用只读存储器，一旦写入，其内容将不会改变，此控制器为静态微程序控制器；如果控制存储器采用随机存储器，则为动态微程序控制器。

6.5 组合逻辑控制器

组合逻辑控制器也称为硬布线控制器，是早期设计计算机的一种方法，这种方法是把控制部件看作为产生专门固定时序控制信号的逻辑电路，而此逻辑电路以使用最少元件和取得最高操作速度为设计目标，一旦控制部件构成后，除非重新设计和物理上对它重新布线，否则要想增加新的控制功能是不可能的，这种缺陷使得组合逻辑控制器的设计和调试变得非常复杂，而且代价巨大，所以，组合逻辑控制器后来被微程序控制器所取代。但随着新一代计算机和 VLSI 技术的发展，组合逻辑控制器又得到了重视，如 RISC 精简指令系统计算机广泛使用这种控制器。

组合逻辑控制器由组合逻辑线路、模 k 时序产生器和 1/k 译码器等部件组成，其原理如图 6-31 所示。

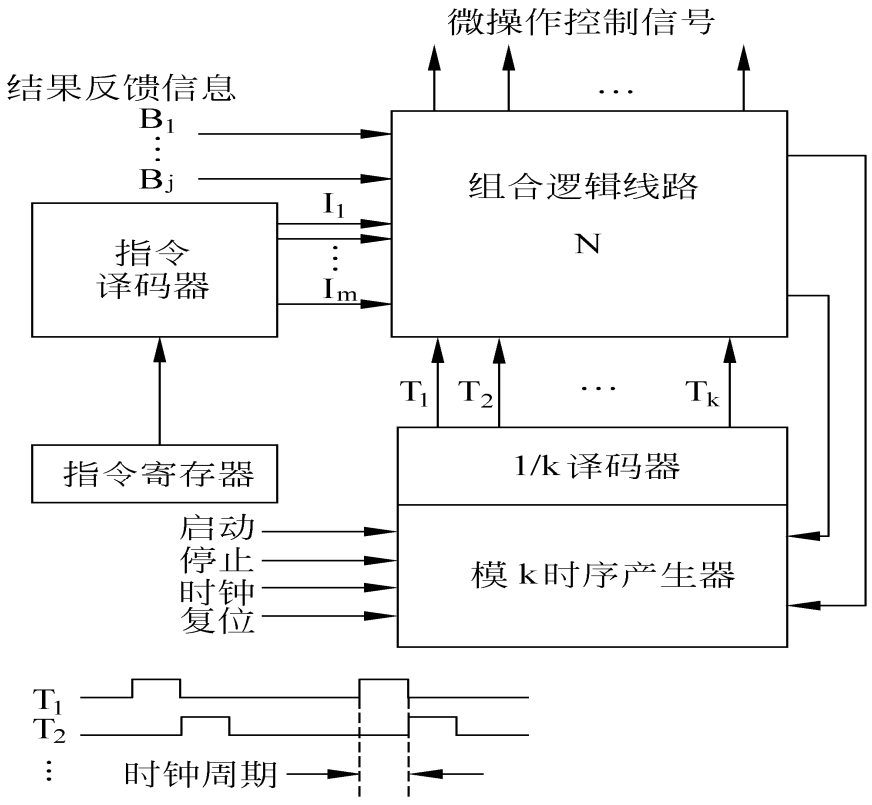


图 6-31 组合逻辑控制器原理框图

其中，组合逻辑线路的输入信号有：来自指令译码器的输出 I_m、来自执行部件的反馈信息 B_j、来自时序产生器的节拍脉冲信号 T_k；组合逻辑线路的输出信号就是微操作控制信号，用来对执行部件进行控制。某一微操作控制信号 C_n 是指令操码译码器的输出 I_m、节拍脉冲信号 T_k 和反馈信息 B_j 的逻辑函数，即 $C_n=f(I_m,T_k,B_j)$。为了说明组合逻辑控制器的工作原理，假设简化的 CPU 模型为如图 6-32 所示。指令的执行过程如图 6-33 所示。其中前 3

步为取指令，对所有指令来说是相同的，后 3 步为执行指令，随指令功能的差异而变化。根据指令 ADD、AND、STA、LDA、JMP、JMPZ、COM 的功能，我们可以画出指令的流程图，如图 6-34 所示。图中微操作决定了在 CPU 中所需的控制信号和控制点。

我们假定读主存和写主存的操作在两个单位时间内完成，其他微操作都可以在一个单位时间内执行完毕。由指令流程图可以看出，ADD、AND、STA、LDA 指令需要 8 个单位时间，其中取指周期需要 4 个单位时间，执行周期需要 4 个单位时间；而 JMP、JMPZ、COM 指令只需 5 个单位时间，4 个单位时间用于取指，一个单位时间用于执行。假定一个单位时间用一个节拍脉冲的时序信号来体现，因此需要一个模 8 计数器。

显然，根据指令流程图，可以确定在指令周期中各时刻必须激活某一指令的某些操作信号。例如，对引起一次内存读操作的操作信号 C_3 来说，当 T_2=1，取指令时被激活；而当 T_6=1，3 条指令（LDA、ADD、AND）取操作数时也被激活，此时，指令译码器的 LDA、ADD、AND 输出为“1”，因此，C_3 的逻辑表达式为：

C3=T2+T6（LDA+ADD+AND）

一般来说，控制信号 Cn=∑(Ti∑Im) 。其中,Im 为指令译码器的输出。在要求 i<k 步的一条指令的情况下（k 为时序产生器的模），时序产生器可以在第 i 步后清零复位，以便缩短指令的执行时间。实现上述指令的组合逻辑控制器如图 6-35 所示。

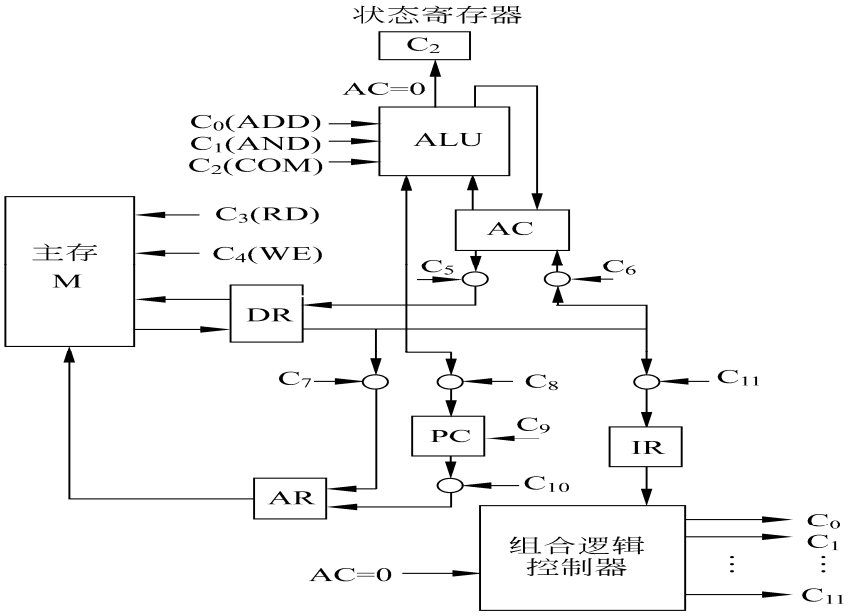


图 6-32　简化的 CPU 模型

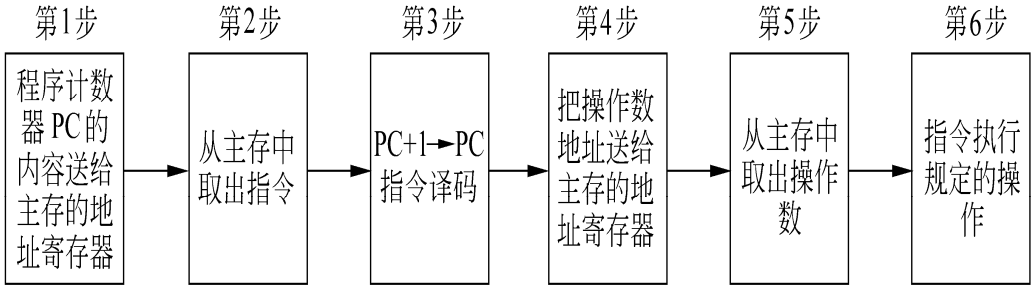


图 6-33　指令的执行过程

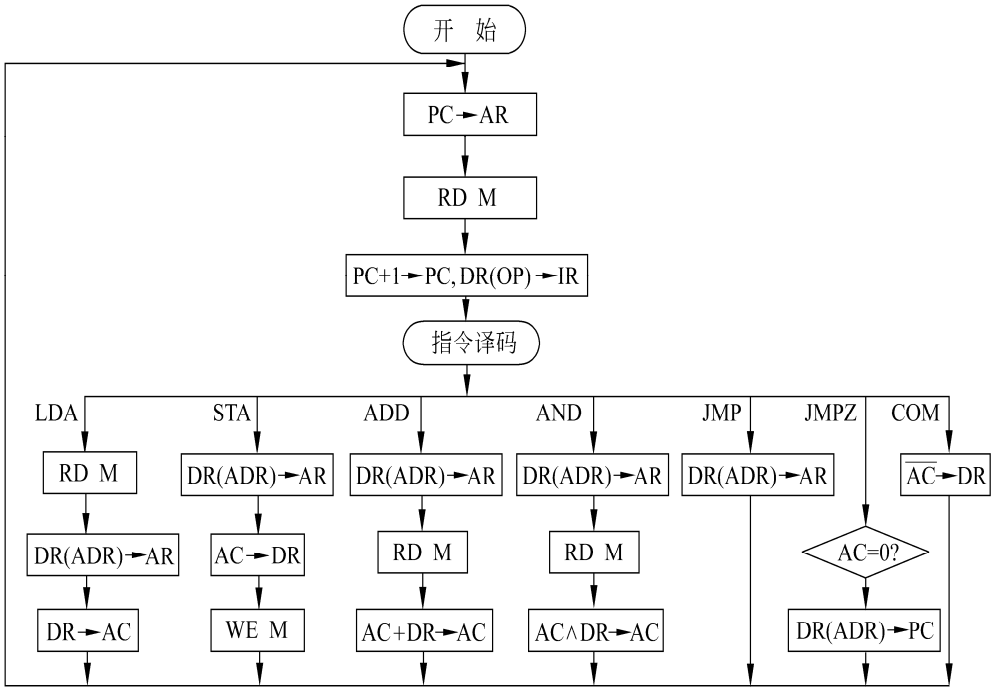


图 6-34 指令的流程图

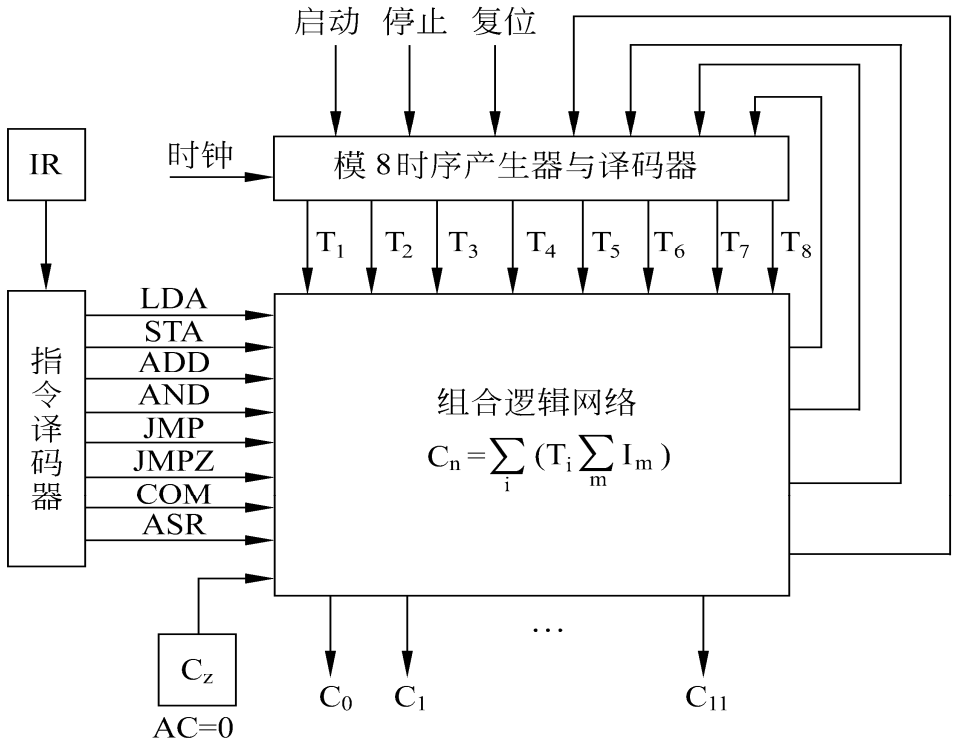


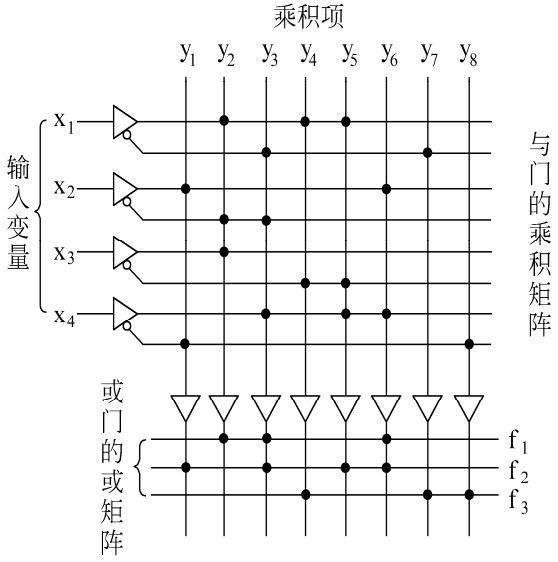
图 6-35 实现 7 条指令的组合逻辑控制器

6.6 门阵列控制器

采用通用可编程逻辑器件可以实现组合逻辑，也可以实现时序逻辑，从而可以满足计算机系统中对随机逻辑功能的需要，即可用来设计操作控制器。由于通用可编程逻辑器件由大量的与门、或门阵列等电路构成，所以简称为门阵列器件。为了与早期的组合逻辑控制器相区别，用门阵列设计的操作控制器称为门阵列控制器。

6.6.1 可编程逻辑阵列 PLA

PLA 逻辑结构如图 6-36 所示，它是多个“与”门电路的集合，并且它可以在任何一个输出端上进行“或”运算。PLA 器件有四个输入变量 x_1、x_2、x_3、x_4，每个变量有原码和反码两个输出，三个和项 f_1～f_3（“或”逻辑）中每一个可以包含 8 个乘积项 y_1～y_8（“与”逻辑），每一个和项 f_i 控制一个输出函数，它可用外界电脉冲编制程序。交叉线上的圆点在矩阵上部相当于“与”门，而在矩阵下部的则相当于“或”门。输入变量的每一行可以被地址矩阵的每一列识别为逻辑“1”、“0”或者任意值（用 d 表示）。



乘积项	输入变量				输出功能		
	x_1	x_2	x_3	x_4	f_1	f_2	f_3
y_1	d	1	d	0	0	1	0
y_2	1	0	1	d	1	0	0
y_3	0	0	d	1	1	1	0
y_4	1	d	0	d	0	0	1
y_5	1	d	0	1	0	1	0
y_6	d	1	d	1	1	1	0
y_7	0	d	1	d	0	0	1
y_8	d	d	d	0	0	0	1

图 6-36　PLA 逻辑电路

例如对于乘积项 y_2 有：x_1=1、x_2=0、x_3=1、x_4=d、f_1=1、f_2=0、f_3=0，对于输出函数 f_1 而言，由 3 个乘积项组成，分别是 y_2、y_3 和 y_6，因此，输出函数 f_1 为：

$$f_1=y_2+y_3+y_6=x_1\overline{x_2}x_3(x_4)+\overline{x_1}\,\overline{x_2}(x_3)x_4+(x_1)x_2(x_3)x_4$$

其中括号表示变量的真值可以为任意值，即可以为“1”，也可以为“0”

由此可见，PLA 在实现任意逻辑函数时与 PROM 相类似，但在逻辑结构上有着本质区别。第一，PLA 中的输出函数并不一定包含所有乘积项，它只对必要的乘积项进行编程，而在 PROM 中则应对所有乘积项进行编程；第二，在 PROM 中，要用固定地址译码器来进行寻址，而 PLA 利用可编程的地址矩阵来选择所需要的乘积项。因此，要实现同样的逻辑函数，用 PLA 要比 PROM 所需的位数少，译码逻辑也较少。

6.6.2 基本思想

我们知道微操作控制信号既是操作码的函数，又是节拍电位、节拍脉冲和反馈条件等因素的函数。因此，在设计门阵列控制器时，通常把指令的操作码、节拍电位、节拍脉冲和反馈条件作为门阵列的输入，再按一定的“与或”关系来编排逻辑阵列的输出，便是所需要的微操作控制信号。假设某一微操作控制信号 C_6 发生在指令 A（设 OP 为 I_1I_2）的节拍电位 M_2、节拍脉冲 T_4 时间，也发生在指令 B（设 OP 为 I_1I_2）的节拍电位 M_3、节拍脉冲 T_2 时间，且进位触发位 C_y 为“1”，那么 C_6 的逻辑表达式如下：

$$C_6=I_1I_2\cdot M_2\cdot T_4+I_1I_2\cdot M_3\cdot T_2\cdot C_y$$

将上述输入变量送入门阵列电路进行编排，就可产生所需要的微操作控制信号。

6.7　流水线处理技术

6.7.1 指令执行方式

指令执行方式是指令执行过程之间的衔接关系。由于一条指令的执行过程大体上可分为

取指令、分析指令和执行指令三个阶段，所以，指令执行方式可以分为顺序、重叠和流水三种方式。

1、顺序执行方式

顺序执行方式是指各条指令执行过程之间是顺序串行执行的。即只有一条指令执行完后，才能取下一条指令来执行。顺序执行方式控制简单，但执行速度慢，效率低。顺序执行指令的方式如图 6-37 所示。

取指令 K	分析 K	执行 K	取指令 K+1	分析 K+1	执行 K+1

图 6-37　顺序执行方式

2、重叠执行方式

重叠执行方式是指在前一条指令执行完成之前，就开始取下一条指令并执行。即相邻两条指令的执行过程在时间上发生重叠。重叠执行方式根据重叠程度，可分为一次重叠和二次重叠。如图 6-38 所示。

取指令 K	分析 K	执行 K		
		取指令 K+1	分析 K+1	执行 K+1

（a）一次重叠

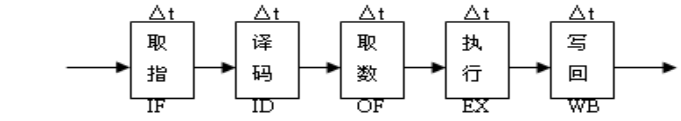
取指令 K	分析 K	执行 K		
	取指令 K+1	分析 K+1	执行 K+1	
		取指令 K+2	分析 K+2	执行 K+2

（b）二次重叠

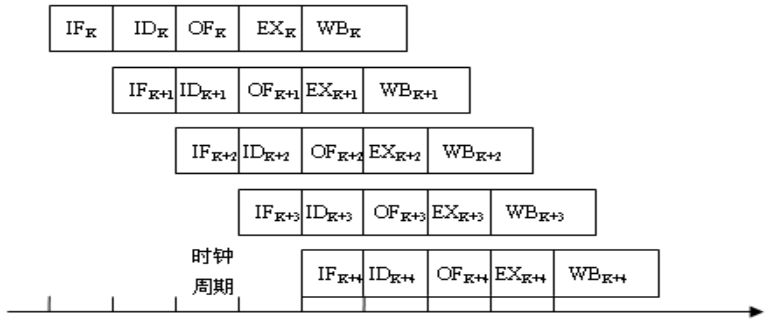
图 6-38 重叠执行方式

3、流水执行方式

流水执行方式是对重叠执行方式的进一步发展，采用类似生产流水线方式来控制指令的执行过程。它是把指令的执行工作划分为若干个复杂程度相当、处理时间大致相等的子任务，每一个子任务由一个独立的功能部件来完成。由于流水线上各个功能部件的并行工作，使得机器执行指令的速度大大提高。假设一条指令的指令周期包含五个子过程：取指（IF）、译码（ID）、取数（OF）、执行（EX）、写回（WB），而每一个子过程对应一个功能部件 S_i，这 5 个功能部件便组成一个五段指令流水线，如图 6-39（a）所示。其指令的流水处理过程如图 6-39（b）所示。



(a)五段指令流水线



(b)指令流水处理过程

图 6-39 五段指令流水方式

从五段指令流水执行方式可以看出，在一个时钟周期内同时有 5 条指令分别在不同的功能部件上进行解释执行，等流水线工作稳定后，每一个时钟周期都会有一条指令的执行结果从流水线流出。假设各个功能段所需时间相等，均为一个时钟周期△t，则在理想情况下，流水线吞吐率（单位时间内所处理的指令条数）为 1/△t。如果采用顺序执行方式，则一条指令的执行时间为 5△t，显然，流水线方式将会大大提高机器的吞吐率。

6.7.2 流水线的分类

1、按等级分类

算术流水线是将复杂的运算过程组成流水线工作方式。如浮点数的加法运算可分成 0 检查、对阶、尾数求和、规格化和溢出 5 个子过程，组成 5 级流水加法器；还有流水乘法器、流水除法器等。现代计算机中已广泛采用了流水的算术运算器

（1）指令流水线

指令流水线是将指令的整个执行过程分为若干个子过程，如取指、译码、执行、写回等几个并行处理的子过程。目前，几乎所有的高性能计算机都采用了指令流水线。

（2）宏流水线

处理机流水线又称为宏流水线，它是指程序步骤的并行，由一串级联的处理机 PE（Processor Element）构成流水线，每台处理机 PE 负责完成某一特定的任务，并将结果输出到共享存储器 SM（Shared Memory）。数据流从第一台处理机输入，经处理后被送入共享存储器 SM 中；第二台处理机从共享存储器 SM 中取出数据进行处理，然后送给共享存储器 SM 中，如此串联下去。随着高档微处理器芯片的出现，处理机流水线被广泛应用在多机系统中。

2、按流水线结构分类

根据流水线中各功能段之间是否存在反馈，可以将流水线分为线性流水线和非线性流水线两种。

（1）线性流水线

线性流水线是指流水线中的每一个功能部件在处理流水任务时，最多只执行一次，且没有反馈回路。

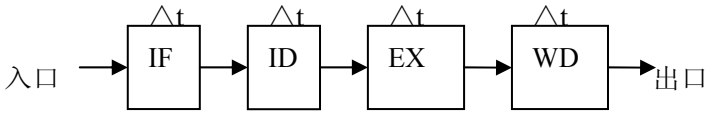
（2）非线性流水线

非线性流水线是指流水线中的一些功能部件在处理流水任务时，可以通过反馈回路多次被执行。

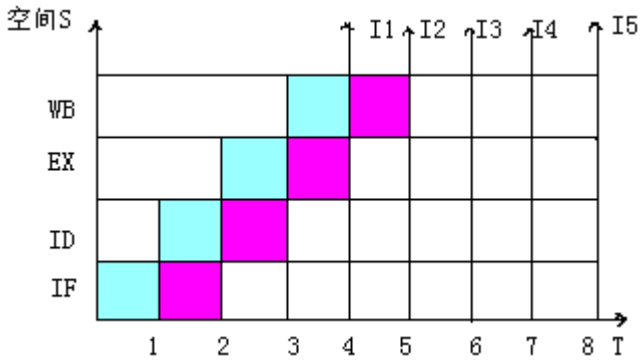
6.7.3 线性流水线

1、线性流水线时空图

线性流水线在执行多个任务时，多个任务从线性流水线的入口鱼贯而入，经各个功能部件处理，到达一定的时间后，线性流水线的出口就会每经过一个时间片完成一个任务。对于图 6-40（a）四段线性指令流水线而言，在经过 4 个 Δt 之后，每一个 Δt 就会执行完一条指令，这个过程可以用线性流水线时空图进行描述，如图 6-40（b）所示。



（a）四段指令流水线



（b）**指令流水线时空图**

图 6-40 线性流水线时空图

2、线性流水线主要技术指标

（1）吞吐率（T_p）

吞吐率是指单位时间内线性流水线所能完成的指令数、任务数或输出结果的数量。对于 m 段线性流水线，如果线性流水线中各个功能部件所占时间均为 Δt，那么，吞吐率 T_p 可按下式进行计算：

$$TP=\frac{n}{m\Delta t+(n-1)\Delta t}$$

（2）加速比（S_P）

$$SP=\frac{T_s}{T_c}=\frac{nm\Delta t}{m\Delta t+(n-1)\Delta t}=\frac{m}{1+(m-1)/n}$$

（3）效率（η）

$$\eta=\frac{n\text{ 个任务占用的时空区}}{m\text{ 段总的时空区}}=\frac{nm\Delta t}{m(m+n-1)\Delta t}=\frac{Sp}{m}$$

【例 6-3】设有 100 条指令的程序经过图 6-39（a）所示的 5 段指令流水线，试求出完成该程序的流水时间、流水线的实际吞吐率、加速比和效率（假设 Δt=10*ns*）。

解：流水总时间 $T_C=m\Delta t+(n-1)\Delta t=5\times10+99\times10=1040ns$

$T_P=100/T_C=100/1040$

非流水总时间 $T_S=n\times5\times\Delta t=100\times5\times10=5000ns$

∴$SP=T_S/T_C=5000/1040\approx5$

$\eta=T_P\times\Delta t\approx100\%$

2、高级流水线

为了提高流水线的处理速度，目前主要采取的措施有：超流水技术、超长指令字技术和超标量技术，在此只简要地介绍，有关详细内容请参考“计算机系统结构”课程的相关章节。

（1）超流水技术

超流水技术主要体现在时间上的进一步重叠，即进一步细化流水线的功能段，增加功能段数量。

（2）超长指令字技术

超长指令字技术简称 VLIW，用于指令系统的进一步重叠，即通过增加超长指令来改善流水性能。具体而言，超长指令字技术经过编译优化，将多条能够并行执行的指令合并成一条具有多个操作码的超长指令。

（3）超标量技术

超标量技术是通过重复设置流水线，来进一步加快流水处理速度。如 Pentium 微处理器，内部采用了 2 套流水线（U、V 流水线），从而进一步提高了运算器的运算速度。

6.7.4 流水线中的相关问题

流水线技术可以提高运行效率，但由于流水线中存在相关问题，将会严重影响流水线的运行速度。所谓相关，是指一段程序的相近指令之间存在某种依赖关系，这种依赖关系会影响指令的并行执行。流水线中的相关问题主要指资源相关、数据相关和控制相关。

1、资源相关

资源相关是指多条指令进入流水线后，在同一机器时钟周期内争用同一个功能部件所发生的冲突。假定一条指令流水线由五段组成。由表 6-5 可以看出，在时钟 4，I_1 与 I_4 两条指令发生争用存储器资源的相关冲突。

表 6-5　　两条指令同时访问内存发生资源相关冲突

指令＼时钟	1	2	3	4	5	6	7	8
I1(Load)	IF	ID	EX	MEM	WB			
I2		IF	ID	EX	MEM	WB		
I3			IF	ID	EX	MEM	WB	
I4				IF	ID	EX	MEM	WB
I5					IF	ID	EX	MEM

解决资源相关冲突的办法：一是第 I_4 条指令停顿一拍后再启动；二是增设一个存储器，将指令和数据分别放在两个存储器中。

2、数据相关

在一个程序中，如果必须等前一条指令执行完毕后，才能执行后一条指令，那么这两条指令就是数据相关的。在流水计算机中，指令的处理是重叠进行的，前一条指令还没有结束，第二、三条指令就陆续地开始工作。由于多条指令的重叠处理，当后继指令所需的操作数，刚好是前一指令的运算结果时，便发生数据相关冲突。如表表 6-6 所示，ADD 指令与 SUB 指令发生了数据相关冲突。

表 6-6　　两条指令发生数据相关冲突

指令＼时钟	1	2	3	4	5	6	7	8
ADD	IF	ID	EX	MEM	WB			
SUB		IF	ID	EX	MEM	WB		
AND			IF	ID	EX	MEM	WB	

解决数据相关冲突的办法：在流水 CPU 的运算器中设置若干运算结果缓冲寄存器，暂时保留运算结果，以便于后继指令直接使用，这称为“向前”或“定向”传送技术。

2、控制相关

控制相关是由转移指令引起的，所以也称控制转移相关。当 CPU 执行转移取指令时，依据转移条件的产生结果，可能为顺序取下条指令；也可能转移到新的目标地址取指令，从而使流水线发生断流。为了减小转移指令对流水线性能的影响，常用以下两种转移处理技术：

（1）延迟转移法

由编译程序重排指令序列来实现，其基本思想是“先执行再转移”，即发生转移时并不排空指令流水线，而是让紧跟在转移指令 I_b 之后已进入流水线的少数几条指令继续完成，如果这些指令是与 I_b 结果无关的有用指令，那么延迟损失时间片正好得到了有效的利用。

（2）转移预测法

用硬件方法来实现，依据指令过去的行为来预测将来的行为。通过使用转移取和顺序取两路指令预取队列器以及目标指令 cache，可将转移预测提前到取指阶段进行，以获得良好的效果。

6.8 多媒体技术

媒体一词在涉及信息传递的领域中是指传递信息的媒介，它包括存储信息的实体与传递信息的载体两部分。磁盘、光盘等皆属存储信息的实体，而载体则指用来表达信息的形体，如数值、文字、声音、图形与动静图像等。

多媒体技术是指计算机把各种不同的电子媒质集成起来，统一进行存储、处理和传输。这些电子媒质包括计算机屏幕显示、视频光盘、CD-ROM 以及语言和声音的综合，同时在这些部件之间建立逻辑连接，从而使整个系统具有交互性。显然，多媒体技术使计算机进一步摆脱了“计算工具”的传统观念，成为处理各种信息的强有力工具。

1、图像与声音的压缩技术

多媒体技术很重要的内容是对图像与声音进行操作、存储与传送。这就需要将每幅图像从模拟量转换成数字量，然后进行图像处理，与图形文字等复合，再存储在机器内。但是进行管理、操作、存储的图像并不只是数量很少的静止图像，而是符合视频标准的每秒 30 帧的彩色图像。如果由多媒体计算机存储器能演放 1 秒钟的音像制品，则信息量就高达 22.5 兆字节，而光盘 CD-ROM 容量只有 550 兆字节。可见如不对图像采用压缩技术，仅存储图像的要求这一点就无法达到，何况 CD-ROM 的数据传输率也只有 150*KB*/s，无法做到多幅图像的实时再现。图像数据如不压缩，则实现多媒体通信也就不可能。图像压缩是将图像用像素存储的方式，经过图像变换、量化、高效编码等处理，转换成特殊形式的编码。这样一来，计算机所需存储与实时传送的数据量就可大大降低。

2、适应多媒体技术的软件技术

为适应多媒体技术发展，一是需要开发具有多媒体功能 OS(操作系统)，二是开展以编辑工具为中心的软件技术研究。对第一个课题，Microsoft 开发的视窗 95 至视窗 2000 系列多媒体 OS 版获得了很大成功。对第二个课题，编辑工具必须将图形、文档、声音、图像、视像等多种媒质联系在一起，为实际应用提供方便。

3、计算机系统结构方面的技术

为了在算机系统中增加多媒体数据的获取功能、压缩解压功能、实时处理功能、多媒体数据的 I/O 与通信功能，在计算机系统结构领域需要做三方面的改进：

第一，选择专用芯片和专用插卡来扩充功能，如声卡、视频卡、网卡、内接或外接调制解调器；

第二，进一步改善总线的结构和性能，如加宽系统总线，提高时钟速率；

第三，将一些重要的多媒体技术融合到 CPU 芯片或设计全新的多媒体 CPU 芯片。

4、MMX 技术

MMX 技术是多媒体扩展结构技术，此技术的应用，极大地提高了计算机在多媒体和通信应用方面的功能。带有 MMX 技术的 CPU 特别适合于数据量很大的图形、图像数据处理，从而使三维图形、图画、运动图像为目标的 MPEG 视频、音乐合成、语音识别、虚拟现实等数据处理的速度有了很大提高。MMX 技术集成到新一代 pentium CPU 时，主要体现在以下 3 个方面：

（1）MMX 数据类型

MMX 技术定义了三种打包的数据类型和一种 64 位字长的数据类型。打包数据类型中的每个元素以及 64 位数都是带符号或不带符号的定点整数(字节、字、双字、四字)。四种

数据类型定义如下：

1）紧缩字节类型：8 个字节打包成一个 64 位数据

2）紧缩字类型：4 个字打包成一个 64 位数据

3）紧缩双字类型：两个 32 位的双字打包成一个 64 位数据

4）四字类型：一个 64 位数

（2）MMX 寄存器

8 个 MMX 寄存器 MM_0—MM_7 的宽度为 64 位，借用浮点处理单元中的 8 个(80 位)数据寄存器，它是通过使用“别名”的办法来实现的。这样，8 个字节或 4 个字或 2 个双字被打包装入一个 64 位的 MMX 寄存器，一旦执行一条 MMX 指令，将所有这些 8 个、4 个或 2 个的数据同时取出，进行数学运算或逻辑操作。事实上，这种运算处理过程是一种并行处理过程，故称为 SIMD(单指令多数据)的并行处理。

（3）MMX 指令集

MMX 指令的先进性体现在：

1）SIMD 结构

SIMD 结构是单指令多数据的系统结构，MMX 指令可以充分利用 64 位带宽的处理能力，一次可以并行处理 8 个 8 位数据、或 4 个 16 位数据、或 2 个 32 位数据。而以前我们遇到的计算机是 SISD(单指令单 数据)的系统结构。

2）饱和运算方式

饱和运算方式是运算发生溢出时使用的处理方法，由于不需要进行溢出处理，所以提高了处理能力。饱和运算适合于面向像素数据的处理。

6.9 典型 CPU 简介

6.9.1 8086CPU

8086CPU 是 40 条引脚的双列直插式组件，采用单一的＋5*V* 电源和 5*MHz* 的单相时钟，具有 20 位地址总线，可寻址的内存地址空间达 1*MB*，可寻址的 I/O 地址空间为 64*KB*，其内部结构如图 6-41 所示。

从图 6-41 中可看出，8086CPU 是由两个独立的功能部件构成，它们是指令执行部件 EU（Execution Unit）和总线接口部件 BIU(Bus Interface Unit)。执行单元 EU 由算术逻辑运算单元 ALU、16 位标志寄存器 FR、通用寄存器组和 EU 控制器等四个部件组成，其主要功能是执行指令。总线接口单元 BIU 由地址加法器、专用寄存器组、指令队列和总线控制电路等四个部件组成，其主要功能是形成访问存储器的物理地址以访问存储器，从存储器取出的指令暂存到指令队列中等待执行；配合 EU 部件访问存储器或 I/O 端口，读取操作数参加 EU 中的运算或存放运算结果。

指令执行单元 EU 和总线执行单元 BIU 的操作各自独立进行，两者也可并行工作，使取指令和执行指令两个操作过程重叠进行，从而有效加快了系统的运算速度。执行单元 EU 执行指令时，不必访问存储器去取指令，而是直接从指令队列中取得指令代码，并分析执行它。若在指令执行过程中需要访问存储器或 I/O 端口，EU 只须向 BIU 送出访问存储器的逻辑地址，BIU 根据 EU 的要求形成访问存储器的物理地址，然后根据物理地址去访问存储器或 I/O 端口，取得操作数送到 EU 中去参加运算。必要时，可将运算结果写回到存储器中去，所以 EU 单元实际上不与外界打交道，所有与外部的操作都是在 BIU 控制下完成。

1、指令执行单元 EU

指令执行单元 EU 只负责执行指令。一般情况下指令按顺序执行，EU 可源源不断地从指令队列中取得执行指令，而省去访问存储器取指令所需的时间。如果在执行指令过程中需要访问存储器取操作数，那么 EU 将访问地址送给 BIU 后，将要等待操作数到来后才能继续

操作，遇到转移类指令，要将指令队列中的后续指令作废，等待 BIU 重新从存储器取出目标地址中的指令代码进人指令队列后，EU 才能继续执行指令。这种情况下，EU 和 BIU 的并行操作会受到一定的影响。这是采用并行操作方式不可避免的现象，只要转移指令出现概率不是很高，两者的并行操作仍然会取得良好的效果。

EU 中的算术逻辑单元（ALU）可完成 16 位或 8 位的二进制运算，运算结果可通过内部总线送到通用寄存器组或 BIU 的内部寄存器中等待写入存储器。16 位暂存器用来暂存参加运算的操作数。经 ALU 运算后的结果特征送入标志寄存器 FR 中保存。

EU 控制器负责从 BIU 的指令队列中取指令，并对指令进行译码，根据指令要求向 EU 内部各部件发出控制命令以完成各条指令的功能。

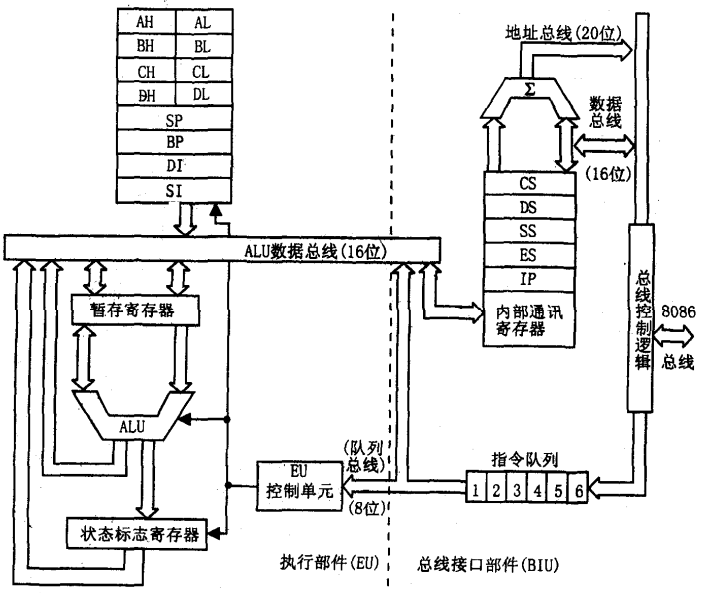


图 6-41 8086CPU 的内部结构图

2、总线接口单元 BIU

总线接口单元 BIU 负责与存储器或 I/O 端口打交道，正常情况下，BIU 通过地址加法器形成指令所在存储器中的物理地址后，启动存储器，从给定地址的存储器中取出指令代码送指令队列中等待执行，一旦指令队列中空出 2 个字节，BIU 将自动进人读指令操作以填满指令队列。只要收到 EU 送来的操作数地址，BIU 将立即形成操作数的物理地址，完成读/写操作数或运算结果的功能。遇到转移类指令，BIU 将指令队列中尚存的指令废除，重新从存储器目标地址中取指令并送到指令队列中。

BIU 的指令队列可存放 6 字节的指令代码，一般情况下应保证指令队列中总是填满指令，使 EU 连续不断地得到待执行的指令。16 位地址加法器专门用来完成逻辑地址变换成物理地址的功能，实际上是进行一次地址加法，将两个 16 位的逻辑地址变换成 20 位的物理地址，以达到寻址 1*MB* 的存储空间。

总线控制电路将 8086CPU 的内部总线与外部总线相连，是 8086CPU 与外部交换数据的通道，它包括 16 条数据总线，20 条地址总线和若干条控制总线，CPU 通过这些总线与外部设备取得联系，并与外部设备一起形成各种规模的 8086 微型计算机系统。

6.9.2 Pentium 微处理器

按照 80X86 的逻辑，80486 之后的下一个 CPU 名字应是 80586，但是 Intel 公司决定不再采用数字命名，原因是为了保护商标版权，所以 Intel 为了获得新一代 CPU 的商标专利，将其取名为 Pentium，它来源于希腊字“Pente”，其意思为 5，现在人们习惯上也将 Pentium 叫做 586，或称作奔腾 586，或称 P5（Pentium 未正式命名前的称呼）。

1、Pentium 微处理器内部结构

Pentium 微处理器是 1993 年投入使用的，与 80X86 系列保持完全兼容，其内部结构如图 4-42 所示。采用 0.8 微米的 Bi-CMOS 技术，使芯片集成度达到 310 万个晶体管。全部引脚被封装在一个大型的 237 针 PGA 中。时钟频率有 60*MHz*、66*MHz*、75*MHz*、90*MHz*、100*MHz*、120*MHz*、133*MHz*、150*MHz* 及 166*MHz* 等多种，其中 Pentium-66 的运算速度为 112*MIPS*。

业内人士认为 Pentium 是一个划时代的微处理器，它的性能已超过了原有的工作站及超级小型机。

2、Pentium 微处理器技术特点

（1）超标量双流水线结构

超标量流水线设计是 Pentium 微处理器技术的核心。所谓超标量就是处理器内部含有多个执行单元来完成多条指令的同时执行。Pentium 有两条分别称为 U 和 V 的指令流水线，各自有独立的算术逻辑单元 ALU 及高速缓存结构。这种双流水线并行作业的方式，使得 Pentium 在每个时钟周期内可同时执行两条指令。此外，还有一个执行单元，保证同时完成一条浮点运算指令。

（2）分支预测技术

为了减少由于转移导致流水线的效率损失，Pentium 采用分支预测技术来动态预测指令的目标地址，从而节省了 CPU 的执行时间。通常在用户程序中包含不少的条件转移指令，在流水线计算机中，这些转移指令由于产生分支可能使予取和予译码指令作废。Pentium 内部有两个予取指令缓冲队列，在执行条件转移指令前，一个以顺序方式予取指令，另一个以转移方式予取指令，后者也称作分支目标缓冲器 BTB（Branch Target Buffer），这是一个小的 cache，它基于转移指令，尤其是循环转移的固有特点，可以认为在大多数情况下，当一条转移指令被再次执行时，其成功与否及转移目标与上次相同。据此可构造动态的分支目标预测硬件。BTB 是一种效果较好的硬件机制，统计表明 BTB 的容量较大时（如超过 256 项）预测准确率可达 90%。通过这种动态分支预测技术，不管是否产生转移，所需指令都在执行前予取好。

（3）双 Cache

Pentium 内部有两个 Cache，每个 8*KB*。一个 Cache 用于指令高速缓存，另一个用于数据高速缓存。这两个高速缓存可同时存取，前者可提供多达 32 位的原始操作码，后者每个时钟周期内可以提供两次存取的数据。这种双路高速缓存结构减少了争用高速缓存所造成的冲突，改进了处理器性能。

（4）更快的浮点运算单元

浮点运算过程分为 8 个流水步级，前 4 步同整数流水线，接下来两步为二级浮点操作，最后两步为写结果、出错报告等。浮点运算单元对一些常用指令如 ADD、MUL 等不是采用微程序，而是由硬件实现，使浮点运算速度更快。

（5）固化常用指令

Pentium 对一些常用指令如 MOV、PUSH、POP、INC、DEC、$\overline{TEST}$ 及 JMP 等改用硬件实现，而不使用微操作，加快了指令的执行速度。

（6）增加总线宽度

Pentium 内部总线与 80386、80486 一样，数据线和地址线的宽度都为 32 位，但 CPU 和内存进行数据交换的外部数据总线为 64 位，提高了读写存储器的速度，使得一个总线周期内的数据传输量提高了一倍。例如，在主频 66*MHz* 时，内部 64 位总线使 CPU 与内存可一次传送 8 个字节数据，这样，Pentium 传送数据的速度达 528*MB*/S（8 字节×66*MHz*）。此外，Pentium 还支持一种叫突发式的总线周期，该模式可在一个总线周期内传送 4 个 64 位数。在存储管理中，Pentium 的页面大小除可采用 80386、80486 的 4*KB* 页面外，还可选用高达 4*MB* 的页面。页面尺寸由控制寄存器 CR_4 中的 PSE 位来选择。当选用 4*MB* 页面时，只需用页组目录项表来寻址 4*MB* 页面，无需页表，从而极大地减少了内存用量，并加快访问内存的速度。

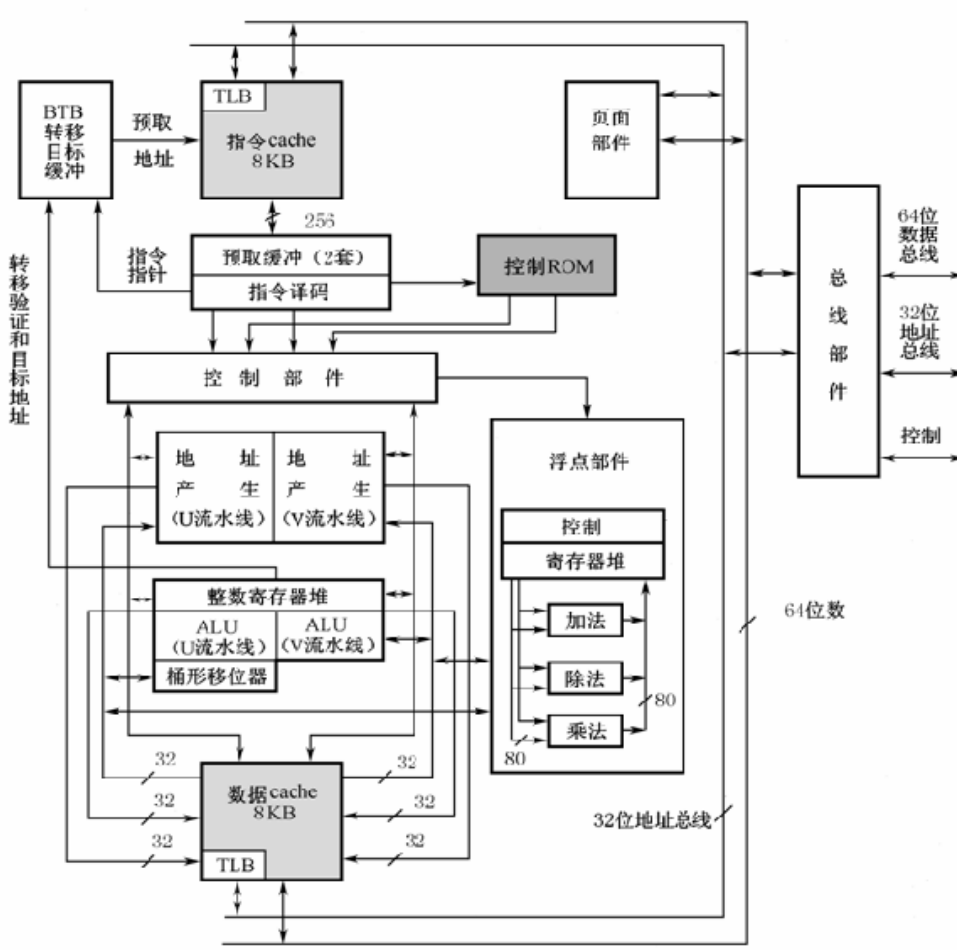
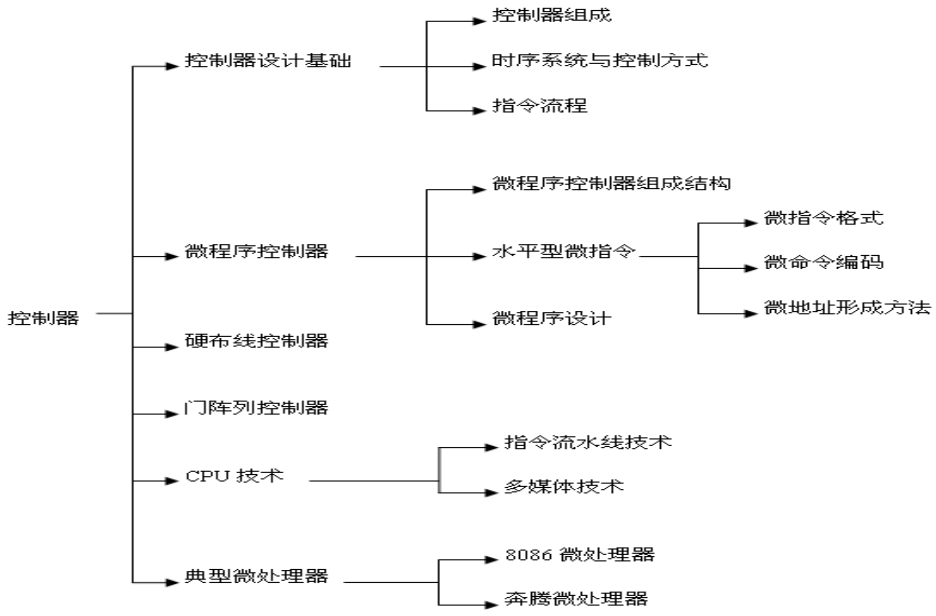


图 6-42 Pentium CPU 的内部结构

关 联



习　题

6.1 控制器的基本功能是什么？它由哪些基本部件组成？各部件作用是什么？

6.2 CPU 中有哪几个最主要的寄存器？它们的主要作用是什么？

6.3 什么是同步控制？什么是异步控制？什么是联合控制？在同步控制方式中，什么是三级时序系统？

6.4 试述指令周期、CPU 周期、节拍周期三者的关系。

6.5 按图 6-9 所示的 CPU 结构框图，试写出执行下面各条指令的控制信号序列。

（1）ADD　R_0，R_1　　（2）ADD　（R_0），R_1

注：指令中第一个地址为源地址，第二个地址为目标地址。

6.6 试分析在模型机中执行下列指令的操作流程

（1）ADD　（R_0），R_1

（2）SUB　X（R_0），R_1

（3）MOV　（R_0），R_1

6.7 试述组合逻辑控制器与微程序控制器的组成差别？

6.8 何谓微命令、微操作、微指令、微周期？

6.9 微指令编码有哪几种常用方式？在分段编码方法中，分段的原则是什么？

6.10 什么是起始微地址？什么是后继微地址？有哪几种形成方法？

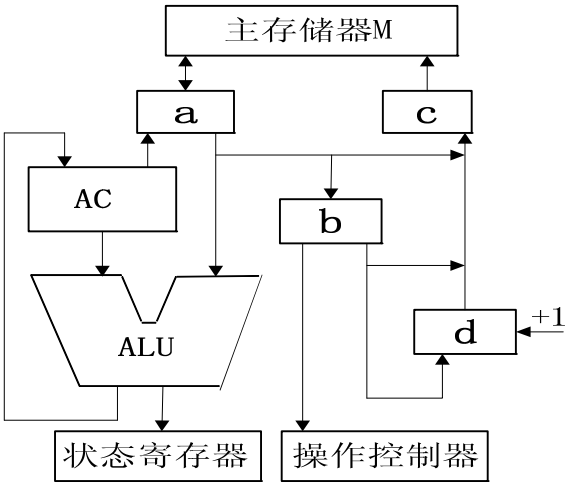
6.11 试写出在微程序控制的模型机中执行下列指令的微程序流程

（1）ADD　（R_0），R_1

（2）SUB　X（R_0），R_1

（3）MOV　（R_0），R_1

6.12 图题 6.12 为一 CPU 的结构框图

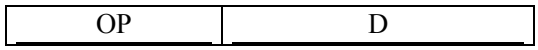


图题　6.12

（1）标明图中 a、b、c、d 四个寄存器的名称

（2）简述取指令的操作流程

（3）若加法指令格式与功能如下：



其功能为：（AC）+（D）→AC，试分析执行加法指令的操作流程。

6.13 某计算机有如下部件：ALU，移位寄存器，指令寄存器 IR，主存储器 M,主存数据寄存器 MDR，主存地址寄存器 MAR，通用寄存器 R_0～R_3，暂存器 C 和 D。

试将各逻辑部件组成一个数据通路，并标明数据流动方向。

6.14 设 R_1 、R_2 、R_3 、R_4 是 CPU 中的通用寄存器，请使用机器周期流程框图分别表示下列指令的执行流程。

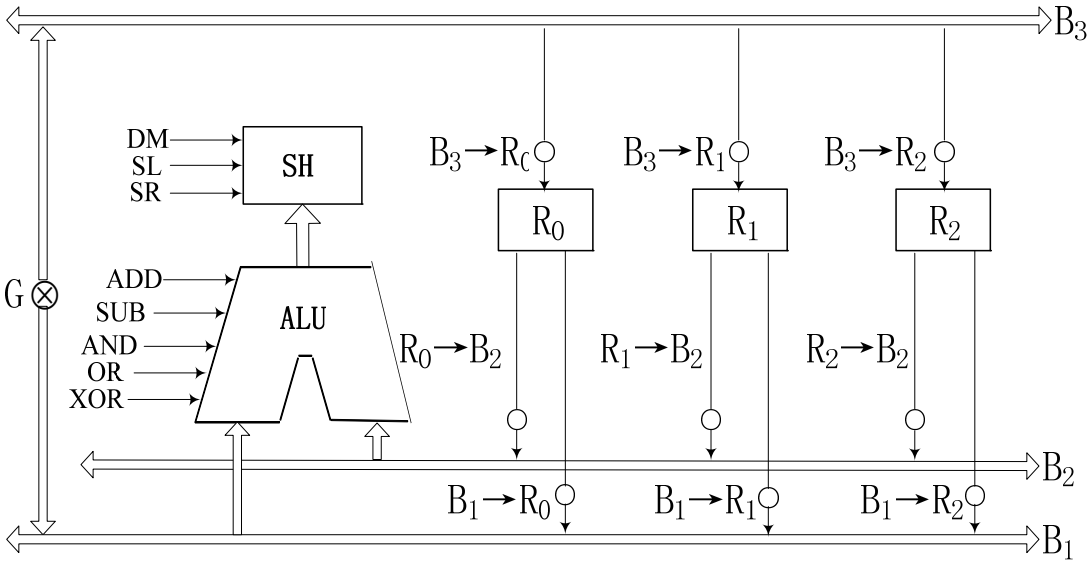
（1）取数指令：LDA （R_1），R_2

该指令是 S-R 型双操作数指令，R_1 为源操作数，R_2 为目的操作数。

（2）存数指令：STA R_3，（R_4）

该指令是 R-S 型双操作数指令，R_3 为源操作数，R_4 为目的操作数。

6.15 某计算机的运算器为三总线（B_1 、B_2 、B_3）结构，B_1 和 B_3 通过控制信号 G 连通。算术逻辑部件 ALU 具有 ADD、SUB、AND、OR、XOR 等 5 种运算功能，其中 SUB 运算时 ALU 输入端为 B_1 － B_2 模式，移位器 SH 可进行直送（DM）、左移一位（SL）、右移一位（SR）3 种操作。通用寄存器 R_0 、R_1 、R_2 都有输入输出控制信号，用于控制寄存器的接收与发送，如图题 6.15 所示。



图题 6.15

试分别写出实现下列功能所需的操作序列。

（1）4（R_0）+（R_1）→R_1

（2）[（R_2）-（R_1）]/2→R_1

（3）（R_0）→R_2

（4）（R_0）∧（R_1）→R_0

（5）（R_2）∨（R_1）→R_2

（6）（R_2）⊕（R_0）→R_0

（7）0→R_0

说明：∧表示与操作、∨表示或操作、⊕表示异或操作。

6.16 现给出 8 条微指令 I_1～I_8 及所涉及的微命令（如表题 6.16 所示）。请设计微指令控制字段格式，要求所使用的控制位最少，并其保持微指令自身内在的并行性。

表题 6.16

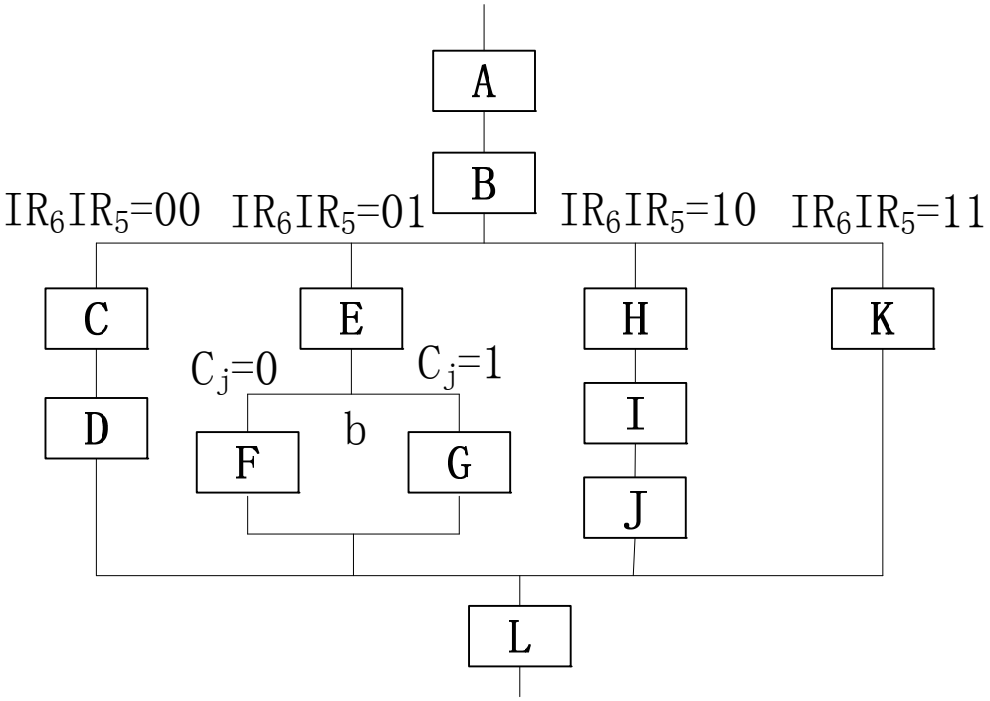
微指令	相关的微指令	微指令	相关的微命令
I_1	a,b,c,d,e	I_5	c,e,g,i
I_2	a,d,f,g	I_6	a,h,j
I_3	b,h	I_7	c,d,h
I_4	c	I_8	a,b,i

6.17 请按断定方式实现图题 6.17 的微程序流程的顺序控制，要求：

（1）给出微指令顺序控制字段格式（假定 μMAR 为 6 位）。

（2）给出各类微指令的二进制地址并编写实现此流程的微程序。

（3）画出地址修改逻辑电路。



图题 6.17

说明：图中每个方框代表一条微指令，分支点 a 由指令寄存器 IR_6 IR_5 两位决定，分支点 b 由进位标志 C_j 决定。

6.18 说明相关性对流水线的影响，并给出一些常用的解决方法。

6.19 假定某计算机的指令按取指、分析和执行三步骤处理，每步所需时间分别为 t_f、t_d、t_e，请分别计算满足下列要求时，执行 100 条指令所花费的时间。

（1）依次串行执行。

（2）仅（K+1）取指与 K 执行重叠。

（3）仅（K+2）取指、（K+1）译码、K 执行重叠。

6.20 在图 6-53 所示的流水线上处理下述程序段时会出现什么问题？如何解决这些问题？

（1）ADD R_1，R_2　　（2）MOV R_3，R_1

（3）ADD R_0，R_4　　（4）MOV （R_4），R_5,

说明：前一个操作数为目的数，后一个操作数为源数。

6.21 单选题

（1）程序计数器的功能是____。

A、存放微指令地址　　B、计算程序长度

C、存放指令　　D、存放下条机器指令的地址

（2）CPU 从主存取出一条指令并执行该指令的所有时间称为____。

A、时钟周期　　B、节拍　　C、机器周期　　D、指令周期

（3）主存中的程序被执行时，首先要将从内存中读出的指令存放到____。

A、程序计数器　B、地址寄存器　C、指令译码器　D、指令寄存器

（4）在下列的部件中，不属于控制器的是____。

A、程序计数器　B、数据缓冲器　C、指令译码器　D.指令寄存器

（5）为了确定下一条微指令的地址而采用的断定方式的基本思想是____。

A、用程序计数器 PC 来产生后继微指令地址

B、用微程序计数器 μPC 来产生后继微指令地址

C、通过微指令顺序控制字段由设计者指定或由设计者指定的判别字段控制产生后继微指令地址

D、通过指令中指定一个专门字段来控制产生后继微指令地址

（6）构成控制信号序列的最小单位是____。

A、微程序　B、微指令　C、微命令　D、机器指令

（7）微程序控制器中，机器指令与微指令的关系是____。

A、每一条机器指令由一条微指令来执行

B、每一条机器指令由一段用微指令编成的微程序来解释执行

C、一段机器指令组成的程序可由一条微指令来执行

D、一条微指令由若干条机器指令组成

6.22 填空题

（1）控制器的主要功能包括____、____、和____等三个功能。

（2）一般而言，CPU 中至少有____、____、____、____、____和____六个寄存器。

（3）微指令的编码方式有_____、_____、_____和_____等三种。

（4）CPU 周期也称为_____周期，一个 CPU 周期包括若干个_____。

（5）在程序执行过程中，控制器控制计算机的运行总是处于_____、分析指令和_____的循环之中。

（6）微程序控制器的核心部件是_____，它一般由_____构成。

（7）在同一微周期中_____的微命令被称为互斥微命令，而在同一微周期中_____的微命令被称为相容微命令。显然，_____的微命令不能放在一起译码。

（8）由于微程序设计的灵活性，只要简单改变_____，就可改变微程序控制的机器指令系统。

6.23 是非题

（1）在主机中，只有存储器能存放数据。（　）

（2）一个指令周期由若干个机器周期组成。（　）

（3）决定计算机运算精度的主要技术指标是计算机的字长。（　）

（4）微程序设计的字段直接编译原则是：同时出现在一条微指令中的微命令放在不同的字段里，而分时出现的微命令放在同一个字段里。（　）

（5）由于微程序控制器采用了存储逻辑，结构简单规整，电路延迟小，而组合逻辑控制器结构复杂，电路延迟大，所以微程序控制器比组合逻辑控制器的速度快。（　）

（6）在 CPU 中，译码器主要用在运算器中选多路输入数据中的一路数据送到 ALU。（　）

（7）控制存储器是用来存放微程序的存储器，它的速度应该比主存储器的速度快。（　）

（8）由于转移指令的出现而导致控制相关，因此 CPU 不能采用流水线技术。（　）

第 7 章　接口与输入输出

【内容摘要】

接口(Interface)是输入接口和输出接口的总称，也是计算机系统中必不可少的组成部分之一。这是因为任何输入或输出设备都必须通过接口与总线相连，随着计算机应用的不断深入，接口和接口技术将会显得尤其重要。它与计算机主机的速度、处理能力、实用性、兼容性等各项性能都有十分密切的关系。为了提高主机的工作效率，组织合理的 I/O 系统、配备先进的 I/O 技术及接口部件是充分发挥计算机系统性能必不可少的条件。

由于外围设备种类繁多、功能各异，因此与主机交换信息的方式也各不相同。本章着重介绍 I/O 系统的组成，主机和外围设备之间的五种数据传输方式——程序控制方式、中断方式、DMA 方式、通道方式和 I/O 处理机方式。

【学习要点】

- 接口的组成和编址方式
- 程序控制方式、中断方式、DMA 方式、通道方式

7.1　接口概述

7.1.1 接口的功能与组成

1、接口的主要功能

I/O 接口是介于主机和外设之间用于完成某些控制功能、速度匹配、信号转换的一种缓冲电路。其作用是：一方面将来自外设的信息传送给微处理机，另一方面是微处理机对信息进行加工后再通过 I/O 接口电路传回外设。I/O 系统如图 7-1 所示。由于数据在 CPU 中传送速度是纳秒级的，而外设的速度则是毫秒级的，最快是微秒级的，两者速度相差悬殊很大。况且 CPU 中的二进制数据是并行传输的，并且有标准的电位要求，而外设因其种类的不同，数据的传输方式有串行的，有并行的，还有串并行的。因此，I/O 接口的基本功能就是进行外设与 CPU 之间的信息转换，使其形式上能互相适应，速度上能互相匹配，以解决 CPU 与外设之间在数据形式、数据的传递方式以及传递速率上存在很大差异的矛盾。同时能根据 CPU 的控制要求，对 I/O 系统的工作进行控制与检测。

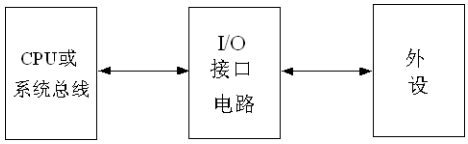


图 7-1　I/O 系统

2、I/O 接口的组成

I/O 接口由硬件和软件构成。I/O 接口硬件电路主要包括寄存器组、译码电路、总线接口和读写控制逻辑，如图 7-2 所示。I/O 接口的软件是指接口的驱动程序。因此，接口技术是一种将硬件和软件相结合的技术，不仅要设计设备的接口电路，还要设计设备的驱动程序，随着计算机应用的不断扩大，接口技术将显得越来越重要。

为了区别 CPU 内部的寄存器，我们把 I/O 接口电路中所包含一组寄存器称为 I/O 端口，简称为端口(Port)。正如每个存储单元都有一个物理地址一样，每个端口也有一个地址与之相对应，该地址称为端口地址。根据端口所存放的信息不同，可以将端口分为数据端口、状态端口和控制端口。具体而言，用来保存 CPU 和外设之间传送的数据(如数字、字符及某种特定的编码等)并对输入/输出数据起缓冲作用的数据寄存器称为数据

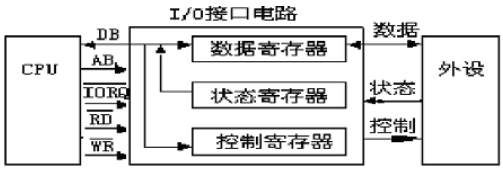


图 7-2　I/O 接口硬件结构

端口；用来存放外设或者接口部件本身状态的状态寄存器称为状态端口；用来存放 CPU 发往外设的控制命令的控制寄存器称为控制端口。由此可见，接口和端口是两个不同的概念，若干个端口加上相应的控制电路才构成接口。软件通常由接口初始化程序和接口工作程序构成。接口初始化程序用来对接口芯片设置工作方式和初始条件。接口工作程序是用来进行数据交换的程序。

7.1.2 I/O 端口的编址方式

I/O 端口都有自己的端口地址，供 CPU 向接口中寄存器发送命令、读取状态和传送数据。一个端口地址可以只对应一个端口，也可以多个端口地址对应一个端口。I/O 端口编址方式是为端口分配地址的方法，具体分配方法有两种：I/O 端口与内存单元统一编址和 I/O 端口与内存单元独立编址。

1、I/O 端口统一编址

这种编址方式是对 I/O 端口和存储单元按照存储单元的编址方法统一编排地址号，由 I/O 端口地址和存储单元地址共同构成一个统一的地址空间。例如，对于一个有 16 根地址线的微机系统，若采用统一编址方式，其地址空间的结构如图 7-3 所示。

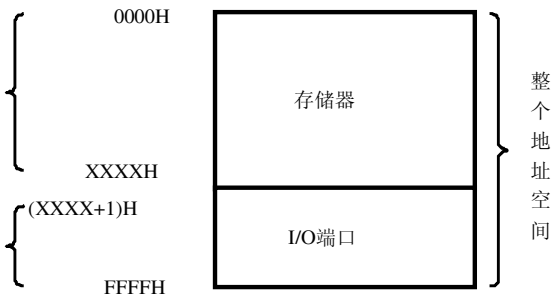


图 7-3　I/O 端口与内存单元统一编址

I/O 端口统一编址的优点是无需专门的 I/O 指令。对存储器的各种寻址方式也同样适用于对 I/O 端口的访问，给使用者提供了很大的方便。缺点是 I/O 端口占用了一部分存储器地址空间，因而相对减少了内存的地址可用范围。

2、I/O 端口独立编址

在这种编址方式中，建立了两个地址空间，一个为内存地址空间，一个为 I/O 地址空间。内存地址空间和 I/O 地址空间是相对独立的，通过控制总线来确定 CPU 到底要访问内存还是 I/O 端口，为确保控制总线发出正确的信号，除了要有访问内存的指令之外，系统还要提供用于 CPU 与 I/O 端口之间进行数据传输的输入/输出指令。80x86 CPU 组成的微机系统都采用独立编址方式。在 8086/8088 系统中，共有 20 根地址线对内存寻址，内存的地址范围是 00000H～0FFFFFH，用地址总线的低 16 位对 I/O 端口寻址，所以 I/O 端口的地址范围是 0000H～0FFFFH，如图 7-4 所示。其优点是存储器地址空间不受 I/O 端口地址空间的影响；其缺点是专用 I/O 指令增加了指令系统复杂性，且 I/O 指令类型少，程序设计灵活性较差。此外，还要求 CPU 提供专门的控制信号以区分对存储器和 I/O 端口的操作，增加了控制逻辑的复杂性。

图 7-4　I/O 端口与内存单元独立编址

7.1.3 I/O 端口地址的译码

当 CPU 执行 I/O 指令时，只能对选中的端口进行读写操作。如何识别被选中的端口，这就是端口地址的译码问题。端口地址的译码方法有多种，下面仅介绍常用的方法。

1、固定式地址译码

固定式译码的端口地址由硬件连线决定，不能更改，适用于不需改变端口地址的场合。例如设计一个“读 2F8H 端口”的译码电路，如图 7-5 所示。分析：2F8H 是一个输入端口（数据输入寄存器）地址，若地址信号 $A_9\cdots A_0$=2F8H 时，选中此端口，并进行数据输入。参与译码的除地址信号 $A_9\cdots A_0$ 外，还需要 AEN、$\overline{IOR}$。其中 AEN 信号用于 DMA 操作控制，只有当 AEN=0 时，即不是 DMA 操作时译码才有效；当 AEN=1 时，即是 DMA 操作时，使译码无效。$\overline{IOR}$ 信号用于输入输出控制，当 $\overline{IOR}$ = 0 时，进行输入；当 $\overline{IOR}$ =1 时，进行输出。当 CPU 执行指令 IN AL，DX ；DX=2F8H 时，该指令产生的总线信号为：$\overline{IOR}$ =0，AEN=0，地址信号 $A_9\cdots A_0$=2F8H，接口电路把 CPU 执行该指令产生的信号变为选通信号 $\overline{Y}$ =0，则选中 2F8H 端口，并读出该端口中的数据，经数据总线送到寄存器 AL。

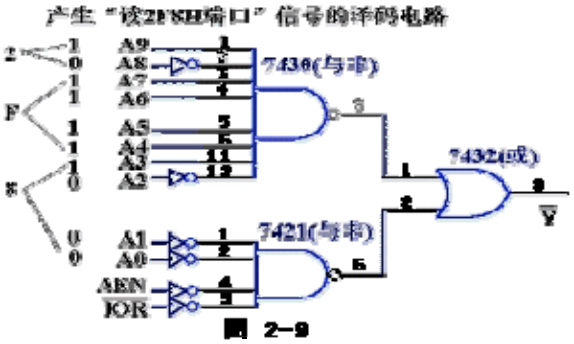


图 7-5　读操作 2F8H 端口地址译码电路

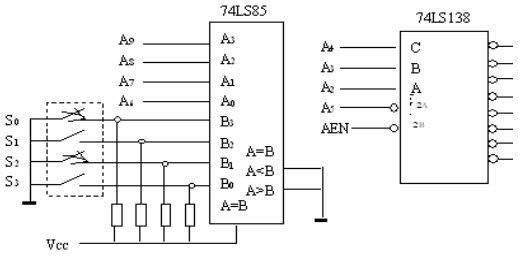


图 7-6　地址开关加比较器译码电路

2、开关式可选地址译码

这种译码方式可通过开关使接口卡的 I/O 端口地址根据要求加以改变而无需改动线路，其电路有以下几种形式。

（1）地址开关加比较器

如图 7-6 所示，当比较器输出有效（相等）时，译码输出有效。此时假设 S_0、S_2 闭合，其译码输出地址为：

$\overline{Y}_0$：140H～143H　　$\overline{Y}_1$：144H～147H

$\overline{Y}_6$：158H～15BH　　$\overline{Y}_7$：15CH～15FH

（2）异或门加地址开关

如图 7-7 所示，若要求异或门的输出 C=1，则两个输入端逻辑电平反相，即若开关 S 闭合，则 A_i=1；否则，A_i =0。若要求异或门的输出 C=0，则两个输入端逻辑电平相同，即若开关 S 闭合，则 A_i=0；否则，A_i =1。其典型应用如图 7-8 所示。读者自己分析端口地址。

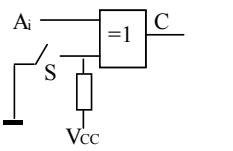


图 7-7 异或门加地址开关

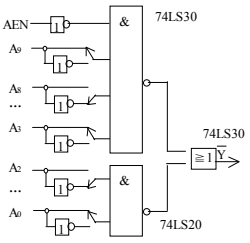


图 7-8 异或门加地址

（3）跳接开关

如图 7-9 所示，其地址由跳接开关决定。其中芯片 74LS138 内部包含 4 个异或门，各异或门与引脚的关系如图 7-9。各个端口地址分配为：

$\overline{Y}_0$ ～$\overline{Y}_7$ =170H～177H。

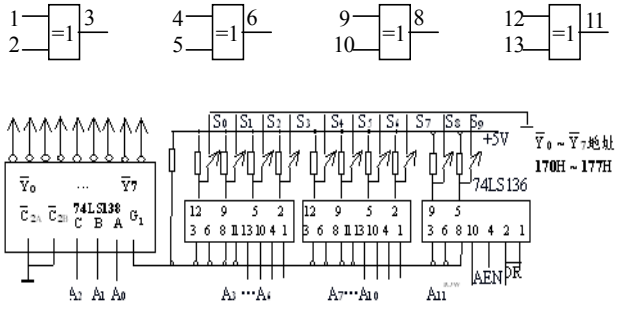


图 7-9 跳接开关译码电路

7.2 输入输出方式

研究主机与外围设备之间的数据传输方式主要围绕着两个问题：一是主机速度与外围设备速度的匹配问题，二是如何提高整机系统的性能问题。早期的输入输出系统是由程序控制的，即外围设备的启动、停止等工作全部由 CPU 执行程序来实现控制。通常而言，外围设备的工作速度比主机的工作速度要低得多，因此，在外围设备工作的时候，主机处于等待状态。在这种工作方式下，主机与外围设备不能同时工作，整个计算机系统的工作效率较低。若主机和外围设备能同时工作，整个计算机系统的工作效率就会明显提高。因而引入了中断的概念，即当外围设备需要 CPU 为其服务时，才向 CPU 请求服务，CPU 暂停当前的工作，转而为外围设备服务。当 CPU 为外围设备的服务结束后，继续返回到原来的工作。由于中断的辅助操作很多，特别是外围设备较多时，中断过于频繁，将使 CPU 应接不暇，导致整机的性能会受到很大的影响。如果外围设备和主存之间的信息传输不经过 CPU，而是外围设备和主存之间的信息直接传输，就成了直接存储器传输（DMA）方式。在 DMA 方式中，CPU 把部分输入输出的控制权交给了设备控制器，在外围设备与主存之间传输信息期间减少了中间环节，从而进一步提高了信息传输率高。如果设备控制器能执行自己的指令、程序来完成输入输出的功能，就形成了通道或 I/O 处理机工作方式。一个通道在执行输入输出过程的开始，由 CPU 进入管理程序，启动通道工作。以后的过程全部由通道按照通道程序自行处理，CPU 可同时继续进行原来的工作，直至整个输入输出过程结束才中断 CPU 作结束处理。不论一个输入输出过程交换多少信息，只需中断 CPU 两次，外围设备、通道和 CPU 可以同时工作。仅在硬件上的这些措施还不一定能充分发挥作用，必须要求操作系统具有管理多道程序运行的功能。现有的操作系统占有大量的 CPU 时间，把操作系统中有关外围设备管理部分的功能分散到输入输出通道中，就形成了 I/O 处理机。

综上所述，输入输出方式可以分为四种方式，即程序控制的输入输出方式、中断方式、DMA 方式和通道方式。程序控制方式和中断方式适用于数据传输量少、传输率较低的外围设备，DMA 方式、通道方式适用于数据传输率较高的外围设备。

7.2.1 程序控制传送方式

1、无条件传送方式

无条件传送方式又称同步传送方式。 微机系统中的一些简单的外设，如开关、继电器、数码管、发光二极管等，在它们工作时，可以认为输入设备已随时准备好向 CPU 提供数据，而输出设备也随时准备好接收 CPU 送来的数据，这样，在 CPU 需要与外设交换信息时，就能够用 IN 或 OUT 指令直接对这些外设进行输入/输出操作。由于在这种方式下 CPU 对外设进行输入/输出操作时无需考虑外设的状态，故称之为无条件传送方式。对于简单外设，若采用无条件传送方式，其接口电路也很简单。如简单外设作为输入设备时，输入数据保持时间相对于 CPU 的处理时间要长得多，所以可直接使用三态缓冲器和数据总线相连，如图 7-10(a)所示。当执行输入的指令时，读信号有效，选择信号 $M/\overline{IO}$ 处于低电平，因而三态缓冲器被选通，使其中早已准备好的输入数据送到数据总线上，再到达 CPU。所以要求 CPU 在执行输入指令时，外设的数据是准备好的，即数据已经存入三态缓

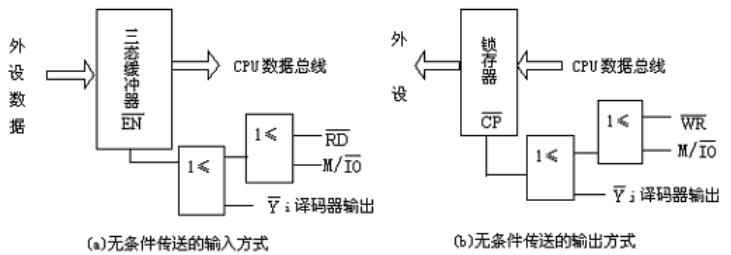


图 7-10 无条件传送方式接口

冲器中。简单外设为输出设备时由于外设取数的速度比较慢，要求 CPU 送出的数据在接口电路的输出端保持一段时间，因而一般都需要锁存器，如图 7-10(b)所示。CPU 执行输出指令时，M / $\overline{IO}$ 和 $\overline{WR}$ 信号有效，于是，接口中的输出锁存器被选中，CPU 输出的信息经过数据总线送入输出锁存器中，输出锁存器保持这个数据，直到外设取走。

无条件传送方式下，程序设计和接口电路都很简单，但是为了保证每一次数据传送时外设都能处于就绪状态，传送不能太频繁。对少量的数据传送来说，无条件传送方式是最经济实用的一种传送方法。

2、查询传送方式

查询传送也称为条件传送，是指在执行输入指令(IN)或输出指令(OUT)前，要先查询相应设备的状态，当输入设备处于准备好状态，输出设备处于空闲状态时，CPU 才执行输入/输出指令与外设交换信息。为此，接口电路中既要有数据端口，还要有状态端口。

查询传送方式的流程图见图 7-11。从图中可以看出，采用查询方式完成一次数据传送要经历如下过程：

第一步，CPU 从接口中读取状态字；

第二步，CPU 检测相应的状态位是否满足“就绪”条件；

第三步，如果不满足，则重复第一步、第二步；若外设已处于“就绪”状态，则传送数据。

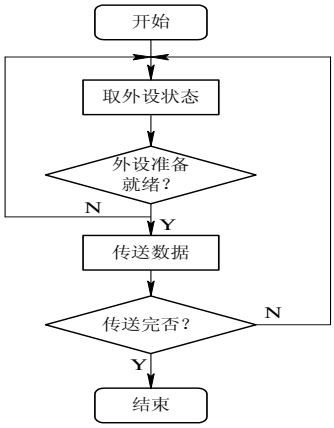


图 7-11　查询传送方式的流程图

（1）查询式输入

实现查询式输入的接口电路见图 7-12。当输入设备数据准备好，就发低电平有效的选通信号 $\overline{STB}$，其作用有：

1）作为 8 位锁存器的控制信号，当 $\overline{STB}$=0 时，输入设备的数据被送入锁存器；

2）使 D 触发器的输出端 Q 端变成高电平，表示外设已准备好，接口电路已有外设送来的数据。

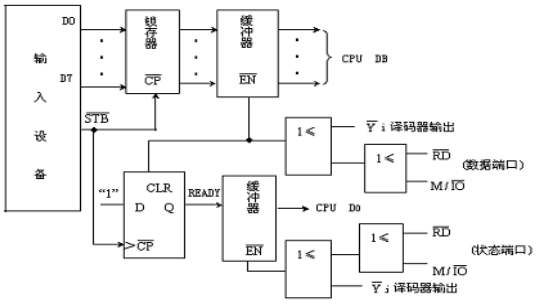


图 7-12　查询式输入的接口电路

当 CPU 要从外设输入数据时，先从状态口读 READY 状态（在 CPU 数据总线的 D_0 上），当 READY=1，从数据端口读入数据，同时把 D 触发器清零（即 READY=0），以准备接收下一个数据。

查询输入的编程如下：

①数据准备好，选通信号输出正跳变，将

数据 ⟶ 锁存器

D 触发器置 1，作为 Ready 信号，D_7=1

②查询状态信号，执行 IN AL，状态口

Ready(bit_7) ⟶ AL

③若 Ready=1，执行 IN AL，数据口

输入数据 ⟶ AL；

D 触发器复位，Ready=0

④程序如下：

WAIT_FOR: IN　　AL，STATUS_PORT

```
TEST    AL，80H
JE      WAIT_FOR
IN      AL，DATA_PORT
```

（2）查询式输出

当 CPU 将数据输出到外部设备时，由于 CPU 传送数据速度很快，如果外设不及时将数据取走，CPU 就不能再向外设输出数据，否则，数据会丢失。因此，外设取走一个数据就要发一个状态信息，告诉 CPU 数据已被取走，可以输出下一个数据。

实现查询式输出的接口电路见图 7-13。CPU 要不断地查询外设，当外设没有准备好时，CPU 要等待，由于许多外设的速度比 CPU 要慢得多，导致 CPU 的利用率不高。

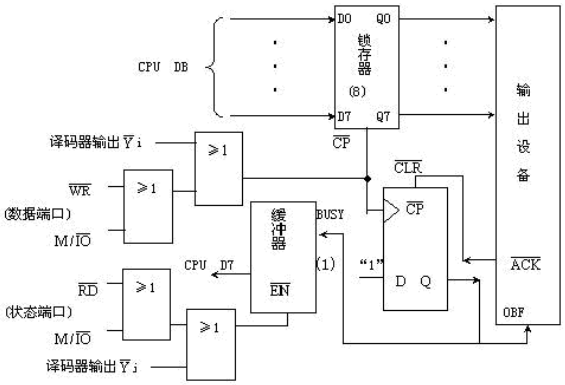


图 7-13 查询式输出的接口电路

查询输出的编程

1）上一数据处理结束，应答信号 $\overline{ACK}$ 的负跳变，使 D 触发器复“0”；输出装置启动信号为 0，缓冲器输出 Busy=0；

2）查询状态信号，执行 指令 IN AL，状态口；Busy(bit_7) ⟶ AL

3）若 Busy =0，执行指令 OUT 数据口，AL；

输出数据 AL ⟶ DB ⟶ 锁存器 ⟶ 输出装置； D 触发器置 1，启动输出装置 Busy=1，禁止输出。

4）程序如下：

```
WAIT_FOR：IN      AL，STATUS_PORT
          TEST    AL，80H
          JNE     WAIT_FOR
          MOV     AL，STORE      ；从数据区取数
          OUT     DATA_PORT，AL
```

【例 7-1】查询数据输入，有 8 个模拟电压，一个 8bit A/D 转换器。由 CPU 控制一个 8 选 1 多路开关，以及 A/D 转换启动。依次实现 8 个模拟电压的 A/D 转换，并把数据存储在 DSTORE 为首地址的内存单元中。

解：查询数据输入程序如下：

```
START：     MOV     DL，0F8H
            LEA     DI，DSTORE
AGAIN：     MOV     AL，DL
            AND     AL，0EFH      ；使 bit4=0
            OUT     Contol_port，AL       ；停止 A/D
            CALL    DELAY         ；延时等待 A/D
            MOV     AL，DL
            OUT     Contol_port，AL       ；结束
WAIT_FOR：  IN      AL，STATUS _PORT      ；输入状态信号
            SHR     AL，1         ；将状态信号
            JNC     WAIT_FOR      ；转 WAIT_FOR
            IN      AL， DATA_PORT
```

```
STORB                    ；(AL) → （DI）
INC      DL              ；改变模拟信号
JNE      AGAIN
```

7.2.2 中断方式

1、中断的基本概念

中断是指当 CPU 正在执行程序过程中，由于某一突发事件的发生，CPU 暂时中止正在执行的程序，转去处理突发事件，待处理完毕后，再返回到原来被中止的程序继续执行。可见中断是一个过程，能够引起中断的突发事件称为中断源，根据中断源不同，可以将中断分为硬件中断和软件中断。处理突发事件的程序称为中断服务程序，简称中服，是否需要中服，可以将中断分为程序中断和简单中断。从主程序转到中断服务程序称为中断响应，根据中断响应方法不同，可以将中断分为查询中断和向量中断。从中断服务程序返回到主程序称为中断返回，为了能够正确返回到被打断处继续执行原程序，中断响应时应自动保护断点。所谓断点就是 CPU 被打断处指令的下一条指令的地址。

可见“中断”是由外围设备或其他非预期的急需处理的事件引起的，外围设备处于“主动”状态，CPU 处于被动状态。例如，现有 1 号、2 号、3 号外围设备处于中断工作方式，它们分别在时刻 t_1、t_2 和 t_3 向 CPU 请求服务，其中断示意图如图 7-14 所示。

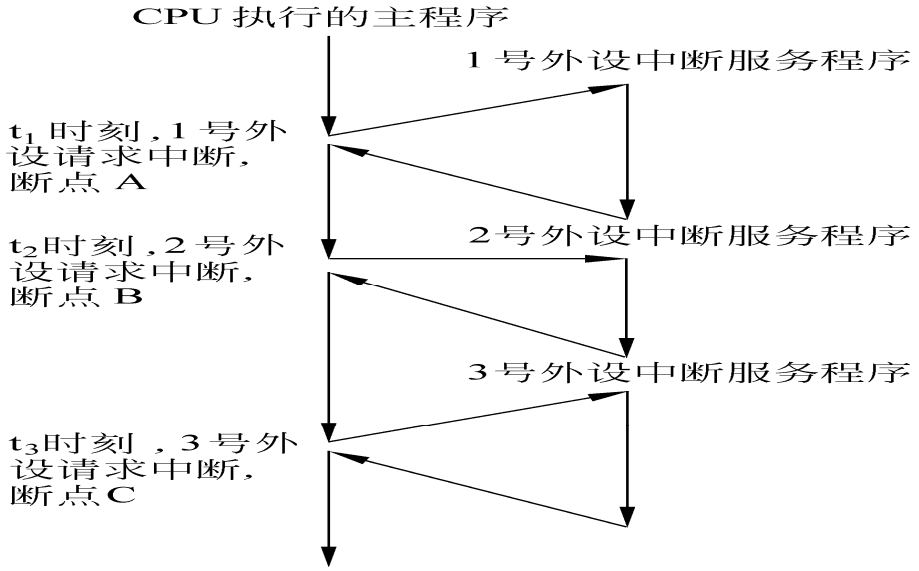


图 7-14 中断响应和处理示意图

2、中断系统

（1）中断系统的功能

1）CPU 和外围设备并行工作

外围设备大多是由微处理机控制工作的，当 CPU 和外围设备之间不需交换数据时，它们可以同时工作，只有当它们之间有数据要交换时，外围设备向 CPU 请求中断，CPU 中断正在执行的程序，转而执行中断服务程序，待中断服务程序执行完毕后，CPU 从断点处继续被中断程序的执行，这时，CPU 和外围设备又可并行工作。中断系统是外围设备和 CPU 联系的必要手段。

2）分时操作

中断系统是变更程序执行流程的有效手段，在多道程序工作的计算机系统中，CPU 执行的程序可以通过定时中断在各道程序之间切换，实现分时操作。对多道程序和分时操作的选择，没有中断系统是不可能的。

3）增强系统的可靠性

当处理机发生运算溢出、非法操作码等程序性错误或机器故障时，可以通过中断系统暂停现行程序的执行，保留现场，转入响应的中断服务程序，对错误或故障进行处理。

4）实时处理

所谓实时处理是指某个事件或现象出现时能及时地进行处理，而不是集中起来进行批处理。由于实时信息是随机的，只有通过中断系统及时响应和处理，才能避免信息的丢失和错误的操作。

5）人机交互

在计算机工作过程中，人要随机地干预机器，了解机器的工作状态，给机器下达临时性的命令等。在没有中断系统的计算机中，这些功能几乎是无法实现的。利用中断系统实现人机交互很方便、很有效。

总之，中断系统在计算机中具有很重要的作用，中断系统和操作系统是密切相关的，在很多方面，操作系统是借助中断系统来控制和管理计算机系统的。

（2）中断接口

CPU 从接受中断请求信号到中断服务结束，可分为两个阶段： 第一个阶段是中断响应；第二个阶段是中断处理。中断响应阶段主要解决三个问题：一是正确地找到对应的中断服务程序的入口地址；二是为中断返回做好准备；三是保证中断响应的完整性。在一个计算机系统中存在着多个中断源，每个中断源有其对应的中断服务程序的入口地址。在中断响应过程中，必须能识别当前请求中断的中断源，即 CPU 必须知道相应中断的中断号或中断向量。CPU 可以根据中断号或中断向量找到中断服务程序的入口地址，将此地址赋予程序计数器 PC，即可跳转到中断服务程序的入口处，从而开始执行中断服务程序。中断号或中断向量一般存放在中断源的接口电路中，如图 7-15 所示。CPU 获取中断号或中断向量的方法是：在中断响应期间，CPU 往接口发送中断响应信号 INTA，接口接收到 INTA 信号后，将中断号或中断向量通过数据总线传输给 CPU。中断服务程序结束后，必须能正确地返回到被中断的断点处继续原来程序的执行。要实现程序的正确返回，首先当外围设备请求中断时，CPU 待当前基本操作结束后，才响应中断；其次 CPU 必须将当前程序计数器 PC 的值（断点地址）及 CPU 的状态（包括各种标志的程序状态字）压入堆栈保护起来，这些操作叫做现场保护。由于中断请求是随机的，在一个中断响应过程中，可能有新的中断请求。为了不造成混乱，CPU 对新的中断请求应不予以响应。解决办法是： 在中断响应期间，置 CPU 内的中断允许标志为无效状态。整个中断响应过程如图 7-16 所示。

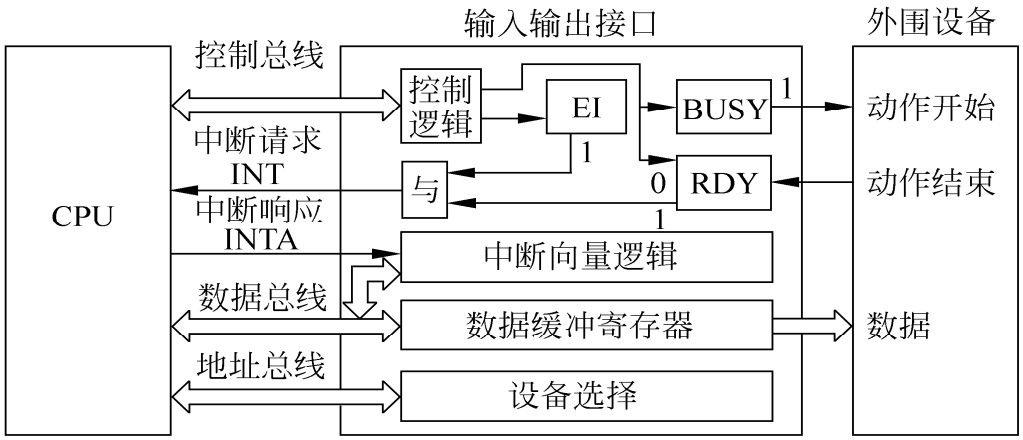


图 7-15 中断接口示意图

不同的外围设备，其功能要求也不同，因此，中断处理的具体内容是不同的。但中断服务程序有共同的结构模式，如图 7-17 所示。进入中断服务程序后，首先要进一步保护现场。

因为中断的发生是随机的，虽然在中断响应期间，CPU 已经将程序计数器 PC 及程序状态字压入堆栈保护，但要使中断返回后，程序能正确地执行下去，必须将中断服务程序用到的一些寄存器的内容压入堆栈保护，以免中断服务程序修改这些现场数据。

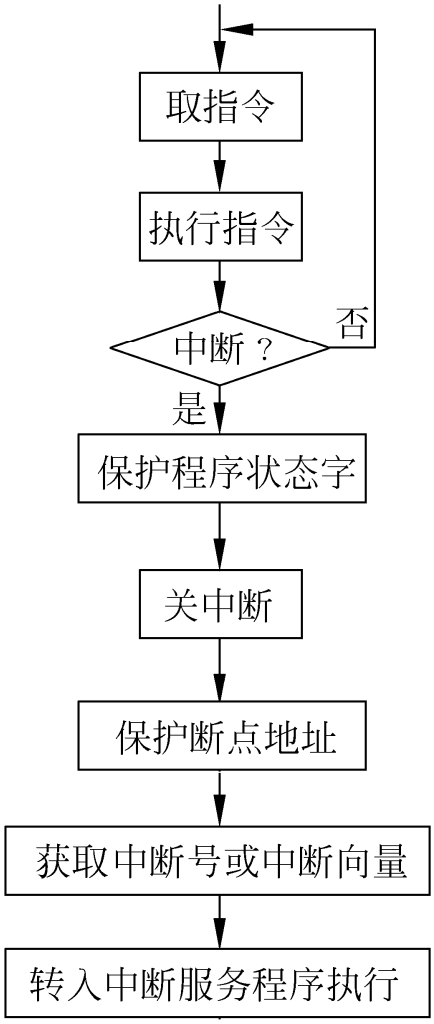


图 7-16　中断响应过程

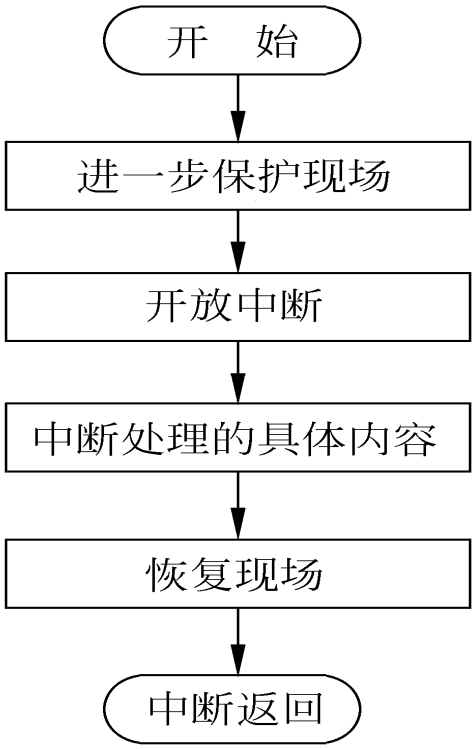


图 7-17 中断服务程序

CPU 在响应中断期间，已经将中断允许标志清“0”，即 CPU 处于禁止中断状态。为了使更高的中断进入，进入中断服务程序后，需将中断允许标志置“1”，开放中断，以实现中断嵌套。所谓中断嵌套，是指优先级别高的中断打断优先级别低的中断。中断服务程序的具体内容执行完成后，恢复现场，即将“进一步保护现场”时压入堆栈的内容从堆栈中弹出，传送给原来的那些寄存器，为中断返回做准备。由于中断的请求是随机的，在一个有多个中断源的计算机中，存在下列两种可能：一是两个中断源同时请求中断；二是当 CPU 正在处理一个中断时，又有新的中断请求。这两个问题的解决方案实质上是中断优先级的处理问题。

（3）中断优先权

中断优先级是根据各个中断事件的轻重缓急程度不同而分成的若干级别，每一个中断源分配给一个优先权。为了使各种中断处理情况都“合乎情理”地进行，中断的响应处理必须符合优先级原则。优先级原则有以下四条：第一，当只有一个中断源请求中断时，CPU 响应此中断；第二，当有两个以上中断源同时请求中断时，CPU 先响应优先级别高的中断源。待优先级别高的中断处理结束后，再响应和处理优先级别低的中断源；第三，当 CPU 正在处理一个中断时，又有一新的中断请求，且新的中断源的优先级比正在处理的中断源的优先

级高，则 CPU 暂停当前中断的处理，转而响应和处理优先级高的中断。待优先级别高的中断处理完毕后，才再继续原中断的处理；第四，当 CPU 正在处理一个中断时，有一新的中断请求，且新的中断源的优先级比正在处理的中断源的优先级低，则待 CPU 处理完当前中断后，才去响应和处理新的中断。

1）软件查询法

中断优先级的解决方法一般有两种：一是软件查询法，二是硬件电路法。软件查询法的程序流程图如图 7-18 所示。在这种情况下，CPU 在接到中断请求信号后，必须判断中断请求信号是哪个外围设备发来的，它可以用程序来查询中断源，以查询到中断源的先后次序来确定优先级，若改变查询次序就可以修改优先级。软件查询法在有些机器中可以用 I/O 指令依次查询，在确认了有请求的中断源后，转入到相应的中断服务程序。也有些机器用取回中断号按优先级次序逐位判定，若某位中断请求标志为“1”，就转去执行该中断源的中断服务程序。用软件查询法实现中断优先级的处理电路简单，但效率较低。

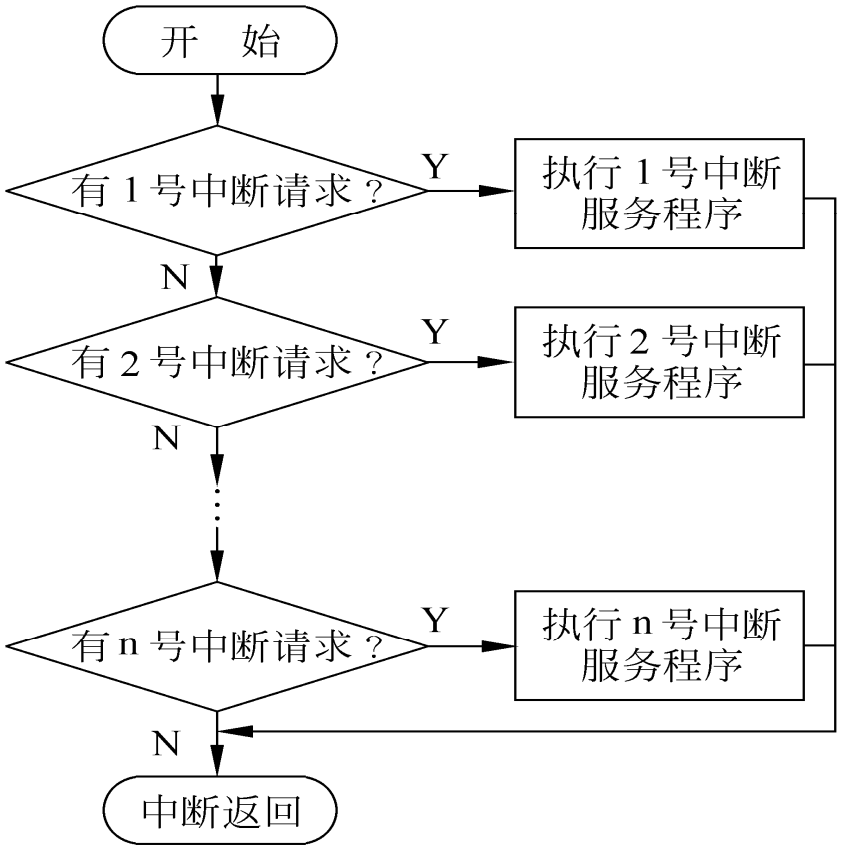


图 7-18 软件查询法的程序流程图

2）硬件电路法

硬件电路法又可以分为菊花链电路和判优逻辑电路两种。菊花链电路用于单线请求的计算机系统，各外围设备的中断请求用一条请求线来传送，当 CPU 接收到中断请求信号后，中断源的优先级不是用查询程序依次排队来确定，而是用硬件排队线路来替代软件排队，即用菊花链电路将中断应答信号（INTA）一级一级地往下传送，菊花链电路如图 7-19 所示。中断应答信号先传到链头，若优先级最高的外围设备没有提出中断请求，则中断应答信号就沿菊花链下传，直到某个提出中断请求的中断源截取中断应答信号后，中断应答信号就不再往下传，该中断源把自己的中断号传送给 CPU，CPU 接到中断号后，转入相应中断服务程序。菊花链电路实现中断优先级的处理速度快，但功能固定，不够灵活。

判优逻辑电路用于多线中断请求的计算机系统，它用可编程芯片实现，用户可灵活设置

中断优先级、中断号、屏蔽和开放中断等内容，目前计算机中都采用这种方式。每个外围设备都可以通过各自的中断请求信号线，将中断信号送给判优逻辑电路，判优逻辑电路如图 7-20 所示。当多个中断源同时请求中断或有中断嵌套时，判优逻辑电路根据预先设定的中断优先级，选出优先级最高的中断源，并将相应的中断号送给 CPU，CPU 立即为优先级最高的中断源服务。

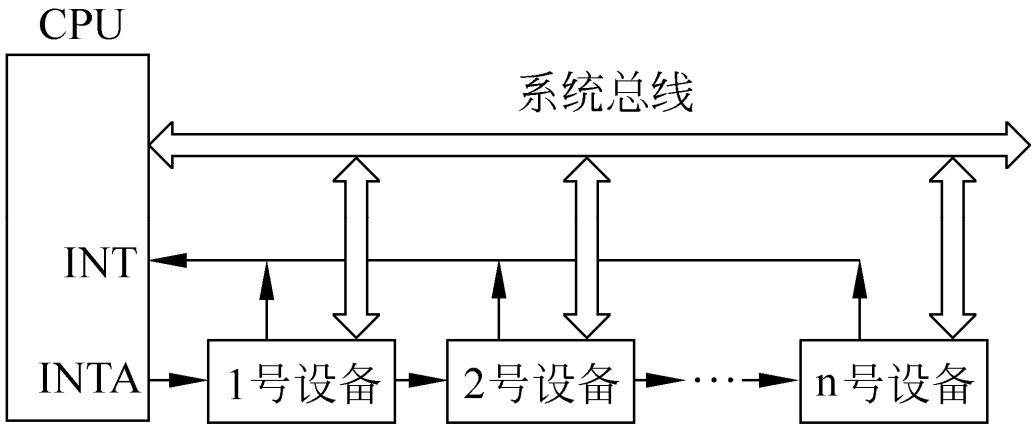


图 7-19 单线请求菊花链电路

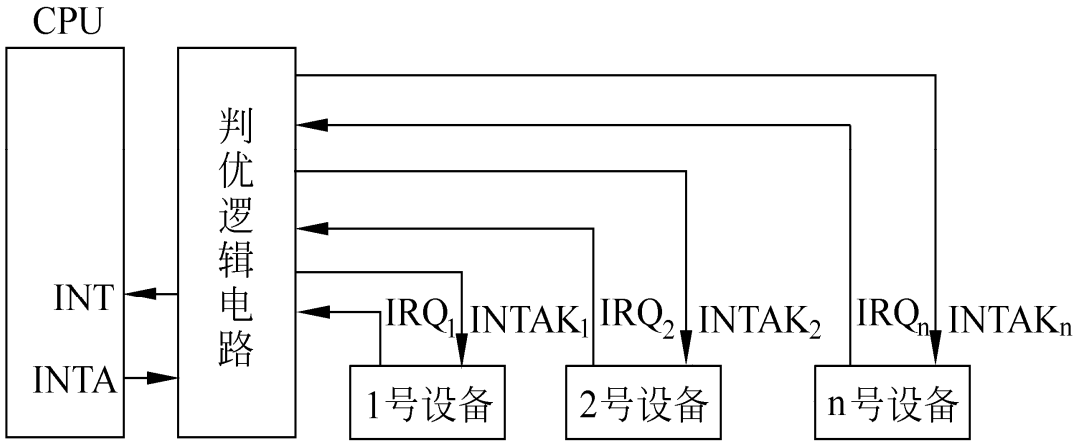


图 7-20 多线请求判优电路

3、中断方式与查询方式比较

查询传送方式是由 CPU 来查询外设的状态，CPU 处于主动地位，而外设处于被动地位。中断传送方式则是由外设主动向 CPU 发出请求，等候 CPU 处理，在没有发出请求时，CPU 和外设都可以独立进行各自的工作。目前的微处理器都具有中断功能，而且已经不仅仅局限于数据的输入/输出，而是在更多的方面有重要的应用。例如实时控制、故障处理以及 BIOS 和 DOS 功能调用等。

中断传送方式的优点是：CPU 不必查询等待，工作效率高，CPU 与外设可以并行工作；由于外设具有申请中断的主动权，故系统实时性比查询方式要好得多。但采用中断传送方式的接口电路相对复杂，而且每进行一次数据传送就要中断一次 CPU，CPU 每次响应中断后，都要转去执行中断处理程序，且都要进行断点和现场的保护和恢复，浪费了很多 CPU 的时间。故这种传送方式一般适合于少量的数据传送。对于大批量数据的输入/输出，可采用高速的直接存储器存取方式，即 DMA 方式。

7.2.3 直接存储器方式

直接存储器存取（DMA）方式，是一种完全由硬件控制的输入输出工作方式，用于实

现存储器和外设之间、存储器和存储器之间直接进行数据传送(如磁盘与内存间交换数据、高速数据采集、内存和内存间的高速数据块传送等)，传送过程无需 CPU 介入，这样，在传送时就不必进行保护现场等一系列额外操作，传输速度基本取决于存储器和外设的速度，如图 7-21 所示。DMA 传送方式需要一个专用接口芯片 DMA 控制器(DMAC)对传送过程加以控制和管理。在正常工作时，CPU 是计算机系统的主控部件，所有工作周期均用于执行 CPU 的程序。在 DMA 方式下，CPU 释放总线的控制权，DMA 控制器接管总线，由 DMAC 发出地址及读/写信号来实现高速数据传输。传送结束后 DMAC 再将总线控制权交还给 CPU。如图 7-22 所示。

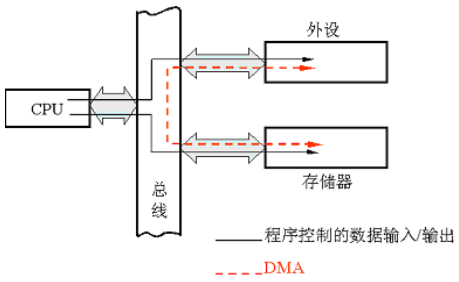


图 7-21　DMA 与程序控制数据传送路径比较

1、总线的分时使用

CPU 和 DMA 控制器都可以作为主控设备，它们可以分时控制总线，实现内存和外围设备之间的数据传输。DMA 控制器和 CPU 分时使用总线的方式有以下三种：停止 CPU 访问、周期挪用、DMA 控制器和 CPU 交替访问内存。

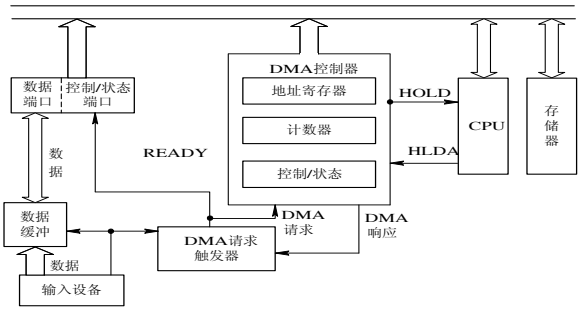


图 7-22　DMA 系统结构框图

（1）停止 CPU 访问

所谓停止 CPU 访问方式，是指在 DMA 传输过程中，CPU 释放总线的控制权，处于不工作状态或者叫保持状态。当外围设备要求传输一批数据时，由 DMA 控制器发一个停止信号给 CPU，要求 CPU 放弃对数据总线、地址总线和有关控制总线的使用权。DMA 控制器获得总线的控制权后，开始数据传输。在一批数据传输完毕后，DMA 控制器向 CPU 发一个 DMA 结束信号，释放总线的控制权，把总线控制权交还给 CPU。这种传送方式的优点是控制简单，它适用于高速的外围设备与内存之间实现成组的数据传输。由于外围设备和内存传输两个数据之间的间隔一般总是大于内存存储周期，因此，在 DMA 期间，一部分内存的工作周期处于空闲状态，内存的效能未得到充分发挥。

（2）周期挪用

在周期挪用方式中，当外围设备没有 DMA 请求时，CPU 按程序要求访问内存；当外围设备有 DMA 请求时，则由外围设备挪用一个或几个内存周期，实现外围设备和内存之间的数据传输。周期挪用方式在进行 DMA 传输时存在两种情况：一种是 CPU 不需要访问内存，如 CPU 正在执行乘法指令，由于乘法指令执行时间较长，此时外围设备访问内存和 CPU 访问内存没有冲突，即外围设备挪用一、二个内存周期对 CPU 执行程序没有任何影响。另一种情况是，在外围设备要求访问内存时，CPU 也要访问内存，这就产生了冲突。在这种情况下，外围设备访问内存的优先级比 CPU 要高，因为外围设备访问内存有时间要求，前一个数据必须在下一个访问内存请求之前存取完毕。显然，在这种情况下外围设备挪用一、二个内存周期，意味着 CPU 延缓了对指令的执行，或者更明确地说，在 CPU 执行访问内存指令的过程中插入 DMA 请求，挪用了一、二个内存周期。与停止 CPU 访问内存的方式比较，周期挪用的方式既实现了外围设备与内存之间的数据传送，又较好地发挥了内存和 CPU 的效率，是一种广泛使用的方法。但是，外围设备每次周期挪用都要申请总线的控制权、建立总线控制权和归还总线控制权等操作。所以，传送一个字对内存来说要占用一个周期，但对

DMA 控制器来说一般要占用 2~5 个内存周期。因此，周期挪用的方式适用于外围设备读写周期大于内存存取周期的情况。

（3）DMA 控制器和 CPU 交替访问内存

如果 CPU 的工作周期比内存存取周期长得多，此时采用交替访问内存的方式可以使 DMA 传输和 CPU 同时发挥最高的效率。这种方式不需要总线使用权的申请、建立和归还过程，总线控制权分两个周期分时由 DMA 控制器和 CPU 控制，DMA 控制器和 CPU 有各自的访问内存地址寄存器、数据寄存器和读写控制逻辑。在第一个周期，如果 DMA 控制器有访问内存的请求，可以在这个周期内传输地址、数据等信号。在第二个周期，如果 CPU 有访问内存的请求，同样在第二个周期内传输地址、数据等信号。对于总线，分别由两个独立的控制信号控制一个多路转换器在 DMA 控制器和 CPU 之间切换总线的控制权。这种控制权的转移几乎不需要什么时间，所以对 DMA 来说效率是最高的。交替访问内存的方式又称“透明的 DMA”方式，该名称的来由是这种 DMA 传送对 CPU 来说是透明的，没有任何感觉和影响。在透明的 DMA 方式下工作，CPU 既可不停止程序的运行，也不进入等待状态，是一种高效率的工作方式。当然，相应的硬件控制逻辑要更加复杂。

2、DMA 控制器

（1）DMA 控制器的组成

如前所述，DMA 控制器可以作为主控部件控制总线实现内存与外围设备之间的数据传输，因此，它具有总线请求和响应，总线控制，传送地址，对传送的字数据计数等功能。DMA 控制器的逻辑结构如图 7-23 所示。

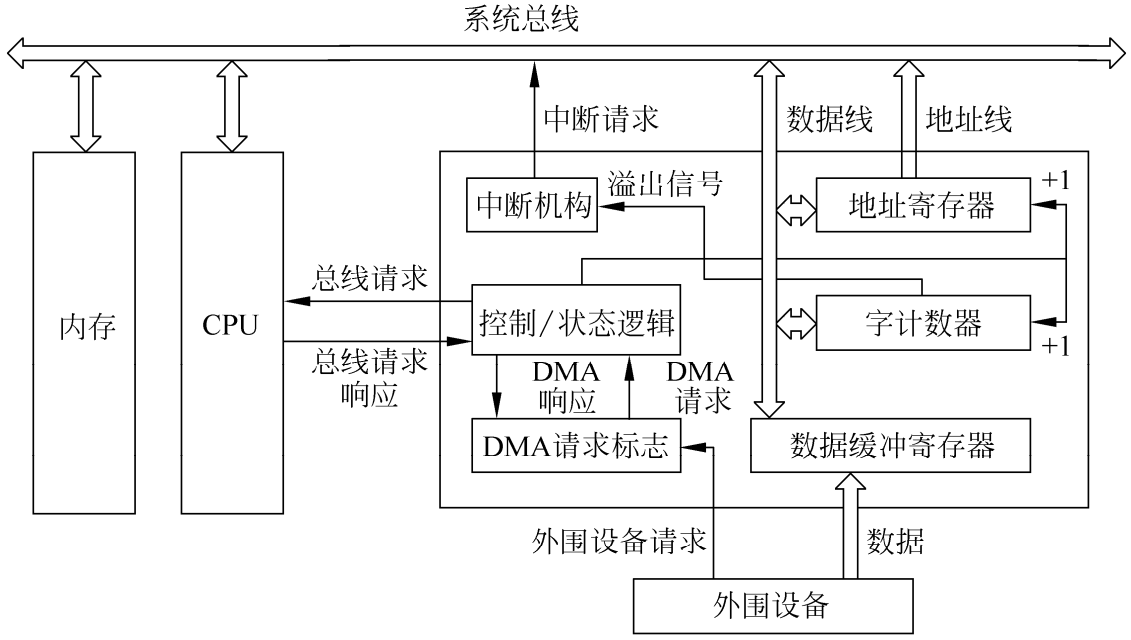


图 7-23 DMA 控制器的逻辑结构

1）地址寄存器

用来存放 DMA 过程中的内存地址。初始时地址寄存器的内容为内存块的首地址，每传输一个字数据，地址寄存器的内容自动加 1，以增量方式给出内存的下一个地址。

2）字计数器

用来记录被传输数据块的长度。初始内容在数据块传输前预置为要传输字数据的总长度的补码。在 DMA 过程中，每传输一个字，计数器加 1。当计数器溢出时，表示这批数据已经传输完毕，于是 DMA 控制器向 CPU 发 DMA 结束信号。

3）数据缓冲寄存器

用来暂存每次传输的数据。当 DMA 以内存—内存方式传输数据时，由 DMA 控制器先将内存中源单元的数据读取至数据缓冲器，再将数据缓冲器中的数据写入到目标内存单元。

4）DMA 请求标志

每当外围设备准备好一个数据字后，给出一个控制信号，使 DMA 请求标志置“1”。该标志置位后向控制/状态逻辑发出 DMA 请求，后者又向 CPU 发出总线请求，CPU 响应此信号后发回总线请求响应信号，控制/状态逻辑接收到此信号后发 DMA 响应信号，使 DMA 请求标志复位，为交换下一个数据做好准备。

5）控制/状态逻辑

它由控制和时序电路、状态标志等组成，用以对地址寄存器、字计数器等内容的修改进行控制，设置数据传输类型，并对 DMA 请求信号和 CPU 响应信号进行协调和同步。

6）中断机构

当字计数器溢出时，意味着一组数据传输完毕，由溢出信号触发中断机构，向 CPU 提出中断报告。这里的中断与上一节介绍的中断概念相同，但目的不同，前者是为了数据的输入或输出，后者是为了告知 CPU 一组数据传输的结束。

（2）DMA 控制器的工作方式

1）单字节传输方式

在该方式下，DMAC 每次控制总线后只传输一个字节，传输完后即释放总线控制权。这样 CPU 至少可以得到一个总线周期，并进行有关操作。

2）成组传输方式(块传输方式)

采用这种方式，DMAC 每次控制总线后都连续传送一组数据，待所有数据全部传送完后再释放总线控制权。显然，成组传输方式的数据传输率要比单字节传输方式高。但是，成组传输期间 CPU 无法进行任何需要使用系统总线的操作。

3）请求传输方式

在该方式下，每传输完一个字节，DMAC 都要检测 I/O 接口发来的 DMA 请求信号是否有效。若有效，则继续进行 DMA 传输；否则就暂停传输，将总线控制权交还给 CPU，直至 DMA 请求信号再次变为有效，再从刚才暂停的那一点继续传输。

3、DMA 的数据传输过程

一次 DMA 的数据传输过程可分为两个阶段：DMA 传输前的预处理阶段和 DMA 数据传输阶段。

（1）预处理

预处理是对 DMA 控制器的初始化操作。在初始化操作时，CPU 作为主控部件，DMA 控制器作为从控设备，根据 DMA 传输要求，CPU 测试外围设备的状态，设置 DMA 初始化命令字。初始化命令字主要包括下列 6 个方面：

1）设置 DMA 传输方式的数据传输方向

DMA 数据传输方向有三种选择：外围设备到内存的数据传输；内存到外围设备的数据传输；内存到内存的数据传输。

2）设置 DMA 的数据传输方式

决定当前要传输的 DMA 方式是停止 CPU 访问方式、周期挪用方式，还是 DMA 控制器和 CPU 交替访问内存方式。

3）设置 DMA 各通道的优先级

对于多个通道的 DMA 控制器来说，通过此命令决定各个通道的优先级。当两个以上的通道同时请求 DMA 传输时，DMA 控制器先响应优先级高的通道。

4）开放或屏蔽 DMA 通道

与中断方式类似，CPU 也可以通过设置命令字开放或屏蔽某个 DMA 通道。当某个通道

被屏蔽后，即使该通道有 DMA 请求，DMA 控制器也不予响应。

5）设置 DMA 传输的字数

字数通常以补码形式。设置 DMA 传输的字数时，CPU 将实际要传输的字数以字计数器的长度为模取补，并将取补后的数据传输给字计数器。例如，当前要传输的字数据块的长度为 8192（2000H），字计数器的长度为 16 位，则其补码为 57344（E000H），CPU 将 57344 传输给字计数器。

6）设置 DMA 传输的内存初始地址

若是内存到内存的 DMA 传输，则需两个 DMA 通道，一个 DMA 控制器通道的地址寄存器用来设置源数据区的初始地址，另一个用来设置目的数据区的初始地址。

（2）DMA 数据传输

在 DMA 数据传输阶段，DMA 控制器作为主控部件，控制总线实现数据传输。下面以外围设备向内存传输数据为例来说明 DMA 的数据传输过程。

1）外围设备向 DMA 控制器请求 DMA 传输

2）DMA 控制器向 CPU 发总线请求信号

若该通道未被屏蔽，则 DMA 控制器进行优先级裁决。如果无更高优先级的 DMA 通道正在进行数据传输或同时请求 DMA 传输，则 DMA 控制器向 CPU 发总线请求信号。

3）CPU 释放总线的控制权

CPU 结束当前正在进行的基本操作后，释放总线的控制权，并向 DMA 控制器发一个总线响应信号。

4）DMA 控制器获得总线的控制权

DMA 控制器接收总线响应信号后，获得总线的控制权，并将 DMA 响应信号传递给外围设备。

5）DMA 传送

DMA 控制器将地址寄存器的内容发往地址总线，同时发 I/O 读和存储器写等控制信号，以传输一个字数据。

6）修改寄存器和计数器内容

若为单字传输，地址寄存器的内容加 1，字计数器的内容加 1，DMA 过程结束；若为数据块传输，则判断字计数器是否溢出，如果未溢出，则继续第 5）步，否则，DMA 传输结束。

7）DMA 结束

DMA 结束时，DMA 控制器将总线控制权交还给 CPU，CPU 继续原来的处理。

在大型计算机系统中，如果仅仅采用前面介绍的程序查询、中断和 DMA 这三种输入输出方式来管理外围设备，还存在以下两个问题：第一，所有输入输出操作都要由 CPU 控制，CPU 的负担较重，整个计算机的性能势必降低。对于低速外围设备，每传输一个字数据都要由 CPU 执行程序来完成，而高速的外围设备虽然采用 DMA 方式降低了 CPU 的负担，但初始化等操作仍需要 CPU 通过执行程序来完成。在大型计算机系统中，这种输入输出操作对 CPU 的时间占用实际上是一种浪费。避免这种浪费的方法之一就是设置专用的输入输出处理机来分担全部或大部分的输入输出操作。第二，大型计算机系统中外围设备虽然很多，但是一般并不同时工作。如果为每一台外围设备都配置一个接口，显然是一种浪费。采用 DMA 方式传输数据，虽然提高了输入输出的速度，但它是以每一台外围设备都配置一个专用的 DMA 控制器为代价的。在微型或小型计算机系统中，由于快速外围设备的台数很少，因而采用 DMA 控制器的数量是有限的。而在大型计算机系统中，快速外围设备的数量较多，就存在着如何让 DMA 控制器能被多台外围设备共享的问题。

7.2.4 通道方式

为了使 CPU 摆脱繁重的输入输出操作，提高系统附加硬件的利用率，在大型计算机系统中采用通道方式传输数据是一种比较好的选择。通道处理机能够负担外围设备的大部分输入输出工作，包括所有按字节传输方式工作的低速和中速外围设备，按数据块传输方式工作的高速外围设备。在一台大型计算机系统中可以有多个通道，一个通道可以连接多个外围设备控制器，而一个设备控制器又可以管理一台或多台外围设备，这样就形成了非常典型的输入输出系统的四级层次结构。

1、通道的基本功能

通道的基本功能是执行通道指令，组织外围设备和内存之间的数据传输，按 I/O 指令要求启动外围设备，向 CPU 报告中断等，具体功能有：第一，接收 CPU 的 I/O 指令，按指令要求与指定的外围设备进行通信；第二，从内存取出属于该通道程序的通道指令，经译码后向设备控制器或外围设备发出各种命令；第三，组织外围设备与内存之间进行数据传输，并根据需要提供数据传输的缓存空间，提供数据存入内存的地址和传输的数据量；第四，从外围设备得到状态信息，形成并保存通道本身的状态信息，根据要求将这些状态信息送到内存的指定单元，供 CPU 使用；第五，将外围设备的中断请求和通道本身的中断请求，按次序向 CPU 报告。

2、通道的分类

根据通道的工作方式不同，可将通道分为字节多路通道、选择通道和数组多路通道。

（1）字节多路通道

字节多路通道是一种简单的共享通道，主要用于连接大量的低速设备。由于外围设备的工作速度较慢，通道在传输两个字节之间有很多空闲时间，利用这段空闲时间，字节多路通道可以为其他外围设备服务。因此，字节多路通道采用分时工作方式，依靠它与 CPU 之间的高速总线分时为多台外围设备服务。

（2）选择通道

选择通道用于连接高速的外围设备。高速外围设备需要很高的数据传输率，因此不能采用字节多路通道那样的控制方式。选择通道在物理上可以连接多台外围设备，但多台设备不能同时工作，也就是说在一段时间内，选择通道只能为一台外围设备服务，在不同的时间内可以选择不同的外围设备。一旦选中某一设备，通道就进入“忙”状态，直到该设备数据传输工作结束后，才能为其他设备服务。

（3）数组多路通道

数组多路通道是字节多路通道和选择通道的结合，它的基本思想是： 当某设备进行数据传输时，通道只为该设备服务；当设备在进行寻址等控制性操作时，通道暂时断开与该设备的连接，挂起该设备的通道程序，去为其他设备服务，即执行其他设备的通道程序。由于数组多路通道既保持了选择通道高速传输数据的优点，又充分利用了控制性操作的时间间隔为其他设备服务，使通道效率充分得到发挥，因此，数组多路通道在实际的计算机系统中应用得最多。

3、通道的工作过程

通道的工作过程可分为启动通道、数据传输、通道程序结束三个阶段，其工作示意图如图 7-24 所示。

（1）启动通道

在用户程序中使用访管指令进入管理程序，由 CPU 通过管理程序组织一个通道程序，并启动通道。广义指令由一条访管指令和若干个参数组成，访管指令的地址码部分实际上是这条访管指令要调用的管理程序入口地址。当用户程序执行到要求进行输入输出操作的访管指令时，产生自愿访管中断请求。CPU 响应这个中断请求后，转入管理程序入口。管理程

序根据广义指令提供的参数，如设备号、交换长度和主存起始地址等信息来编制通道程序，在通道程序的最后，用一条启动输入输出指令来启动通道开始工作。

（2）数据传输

通道处理机执行 CPU 为它组织的通道程序，完成指定的数据输入输出工作。通道被启动后，CPU 就可以退出操作系统的管理程序，返回到用户程序中继续执行原来的程序，而通道开始传输数据。

（3）通道程序结束

当通道处理机执行完通道程序的最后一条通道指令——“断开通道指令”时，通道的数据传输工作就全部结束了。通道程序结束后向 CPU 发出中断请求。CPU 响应这个中断请求后，第二次进入操作系统，调用管理程序对输入输出中断进行处理。如果是正常结束，管理程序进行必要的登记等工作；如果是故障、错误等异常情况，则进行例外情况处理。然后 CPU 返回到用户程序继续执行。

总之，每完成一次输入输出工作，CPU 只需要两次调用管理程序，大大减少了对用户程序的打扰。当系统中有多个通道同时工作时，CPU 与多种不同类型、不同工作速度的外围设备并行工作，可以充分发挥效能。

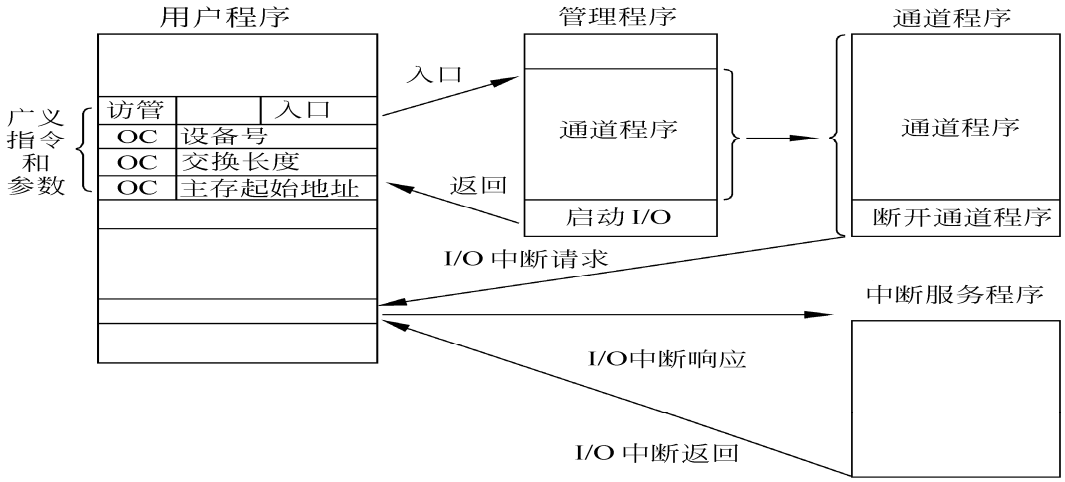
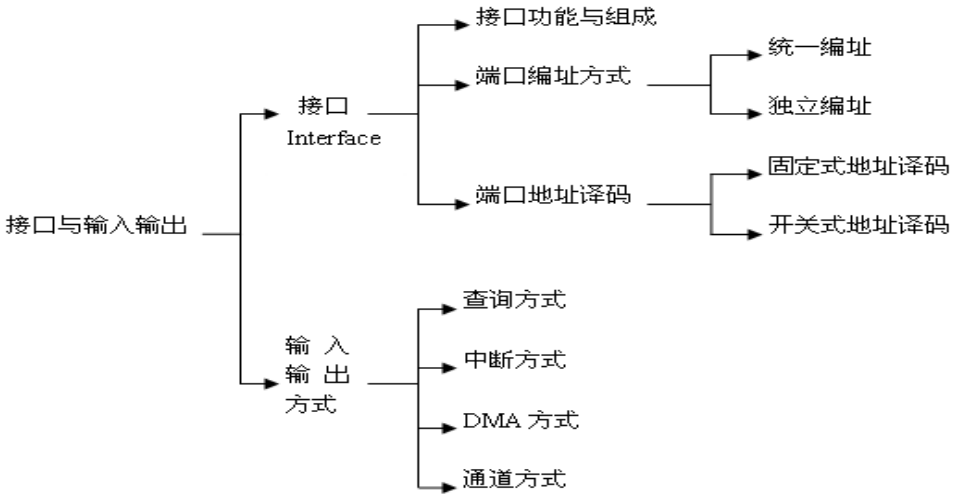


图 7-24 通道工作过程

关　联



习　题

7.1 什么是接口？为什么需要接口？接口有哪些主要功能？

7.2 什么是端口？端口有哪些编址方式？各有什么特点？

7.3 CPU 外设之间一般有哪些信息传输方式？各有什么特点？

7.4 简述以中断方式传送一个数据信息的过程。

7.5 直接程序控制方式如何控制主机与外设之间的信息交换？这种方式有哪些优缺点？

7.6 假设某计算机系统有 5 级中断，其优先次序为：$P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_5$,试问：

（1）若 CPU 执行正常程序过程中，有 P_2、P_4 请求中断，CPU 在现行指令结束响应中断，执行某一中断服务程序，在执行过程中又出现了 P_1、P_3 请求，画出 CPU 处理中断的过程示意图。

（2）若将中断处理次序改为：$P_2 \rightarrow P_4 \rightarrow P_1 \rightarrow P_5 \rightarrow P_3$,试给出中断屏蔽码表（参考表 7.7）。

7.7 某中断系统有 5 级中断，其优先级为 1→2→3→4→5。通过设置屏蔽的方法可以改变各级中断的处理次序。设中断屏蔽位为 0 表示屏蔽，现将中断处理次序改为 1→4→2→5→3，将中断屏蔽位的设置情况填入表题 7.7 内。

表题　7.7

中断处理程序级别	中断屏蔽位				
	1 级	2 级	3 级	4 级	5 级
第 1 级					
第 2 级					
第 3 级					
第 4 级					
第 5 级					

7.8 与程序中断方式相比，DMA 方式有哪些主要特点？

7.9 比较直接程序控制方式、程序中断方式、DMA 方式及通道方式进行信息传送控制的主要特点和适用场合。

7.10 说明 DMA 控制器中下面寄存器或部件的作用。

（1）数据缓冲寄存器 DBR

（2）主存地址寄存器 DMAR

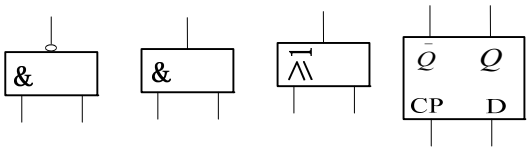
（3）中断控制逻辑

7.11 什么是 DMA 方式？在用 DMA 方式进行信息传送的预处理阶段应传送哪些初始参数？

7.12 某计算机中断系统设有 5 个中断源，分别为 P_1、P_2、P_3、P_4、P_5，它们的优先次序从高到低为 $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_5$，假设来自设备的中断请求信号分别为 $INTR_1$、$INTR_2$、$INTR_3$、$INTR_4$、$INTR_5$，经排队后送入向量地址产生电路的信号为 INT_1、INT_2、INT_3、INT_4、INT_5。已知各中断源的向量地址如下：

表题　7.12

中断源	中断信号	中断向量地址
P_1	INT_1	0005H
P_2	INT_2	0007H
P_3	INT_3	0009H
P_4	INT_4	000BH
P_5	INT_5	000DH



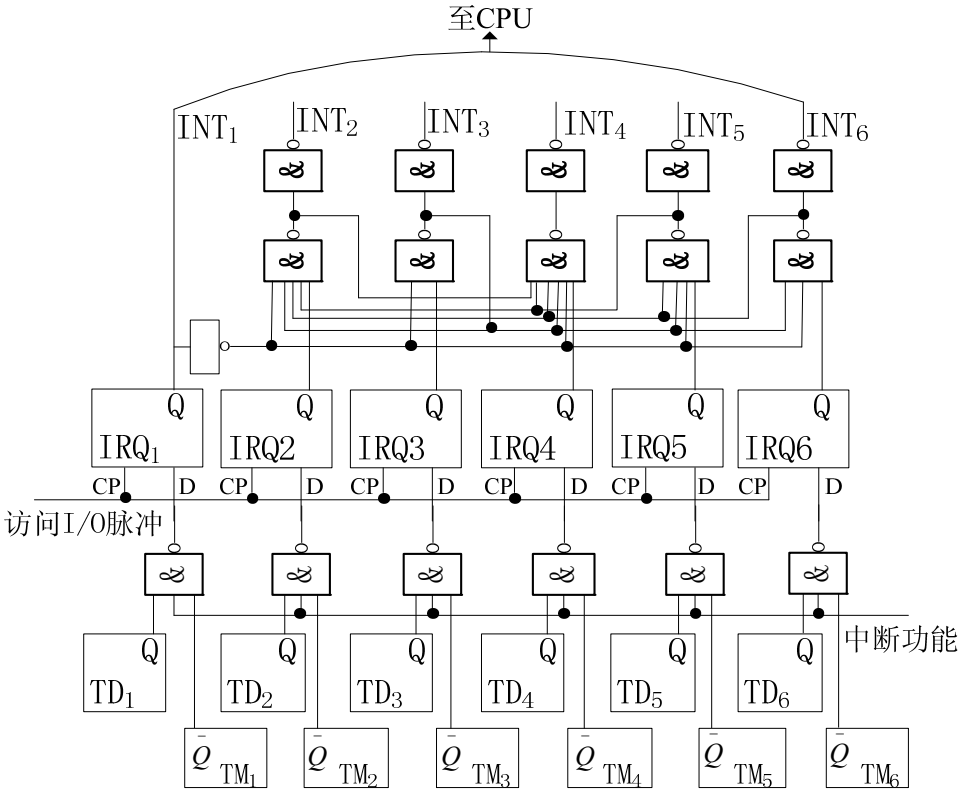
图题　7.12

设向量地址寄存器为 VMAR。当 CPU 给出中断响应信号 INTA 时，将目前已申请中断且优先级最高的中断源的向量地址送入 VMAR。试设计中断排队电路、向量地址形成电路及 VMAR 的第 4 位到第 0 位的逻辑电路。

7.13 图题 7.13 给出了一个中断控制逻辑线路。图中 T_{Mi} 为第 i 级设备的中断屏蔽触发器，T_{Mi}=1 表示屏蔽该设备的中断请求（i=1～6），T_{Di} 为第 i 级设备的完成（或就绪）触发器，T_{Di}=1 表示第 i 级设备工作完成或就绪，可以发出中断请求。IRQ_i 为第 i 级设备的中断请求触发器，IRQ_i=1 表示第 i 级设备发出中断请求，INT_i 为第 i 级设备向 CPU 或中断向量地址编码电路传送的中断请求信号。表题 7.13 中给出了某时刻各中断源（设备）的状态。

表题　7.13

设备状态	P_1	P_2	P_3	P_4	P_5	P_6
T_{Di}	1	1	1	1	1	1
T_{Mi}	1	0	0	0	0	1



图题　7.13

（1）根据表题 7.13 提供的信息，哪些设备可能发出中断请求信号，使相应的 IRQ_i=1？

（2）根据表题 7.13 提供的信息，CPU 应首先响应哪个设备的中断请求?

（3）若 CPU 响应了设备 6 的中断请求，问应向哪些设备的中断屏蔽触发器 T_{Mi} 发出中断屏蔽信号，使 T_{Mi}=1，以便实现多重中断的处理。

（4）如果在 CPU 执行某用户程序过程中，有了中断源 1、4 的中断请求，CPU 在处理中断源 4 的中断请求过程中，又有了中断源 2、3 的中断请求。请画出 CPU 处理各中断请求的过程。

7.14 判断题

（1）DMA 控制器和 CPU 可以同时使用总线工作。（ ）

（2）在计算机系统中，所有的数据传送都必须由 CPU 控制实现。（ ）

（3）一个更高优先级的中断请求可以中断另一个中断处理程序的执行。（ ）

（4）外围设备一旦申请中断，立刻能得到 CPU 的响应。（ ）

（5）通道程序是由通道控制字组成的，通道控制字也称通道指令。（ ）

（6）在直接程序控制方式下，CPU 启动 I/O 设备的指令开始执行后，直到数据传送完为止，CPU 不能执行别的程序。（ ）

（7）DMA 方式提高了 CPU 的效率，同时也提高了数据传送的速度。这是由于 DMA 方式在传送数据时不需要 CPU 干预，而且在一批数据传送完毕时，也完全不需要 CPU 干预。（ ）

（8）CPU 在执行当前指令最后所做的检查是否有各类中断请求的次序，即为 CPU 处理各类中断的次序。（ ）

7.15 选择题

（1）I/O 接口中的数据缓冲器的作用是（ ）。

A、用来暂存外围设备和 CPU 之间传送的数据

B、用来暂存外围设备的状态

C、用来暂存外围设备的地址

D、以上都不是

（2）在中断响应过程中，保护程序计数器的作用是（ ）。

A、使 CPU 能找到中断处理程序的入口地址

B、使中断返回后，能回到断点处继续原程序的执行

C、使 CPU 和外围设备能并行工作

D、为了实现中断嵌套

（3）DMA 方式用来实现（ ）。

A、CPU 和内存之间的数据传送

B、外围设备和外围设备之间的数据传送

C、CPU 和外围设备之间的数据传送

D、内存和外围设备之间的数据传送

（4）如果认为 CPU 查询设备的状态信号是处于非有效工作状态，那么，在下面几种主机与设备之间的数据传送方式中，（ ）主机与设备是串行工作的；（ ）主机与设备是并行工作的；（ ）主程序与外围设备是并行运行的。

A、程序查询方式 B、中断方式 C、DMA 方式 D、通道方式

（5）中断向量地址是（ ）。

A、子程序的入口地址 B、中断服务程序的入口地址

C、中断服务程序入口地址的地址 D、中断向量表的起始地址

（6）采用 DMA 方式传送数据时，每传送一个数据，就要占用（ ）的时间。

A、一个指令周期 B、一个 CPU 周期

C、一个存储周期 D、一个总线周期

（7）周期挪用方式常用于（ ）中。

A、直接存储器存取方式的输入输出

B、直接程序控制传送方式的输入输出

C、CPU 的某寄存器与存储器之间的直接程序控制传送

D、程序中断方式的输入输出

（8）在下面有关 DMA 概念的叙述中，正确的是（　）。

A、当 CPU 在执行指令时，CPU 与 DMA 控制器同时提出了对主存访问的要求，这时应首先满足 CPU 的要求，以免指令执行发生错误，而 DMA 传送数据是可等待的。

B、DMA 周期挪用方式是在 CPU 访问存储器总线周期结束时，插入一个 DMA 访问周期。在此期间，CPU 等待或执行不需要访问内存的操作。

C、因为 DMA 传送是在 DMA 控制器控制下内存与外设直接进行数据传送，因此在这种方式中，始终不需要 CPU 干预。

D、CPU 在接到 DMA 请求后，必须尽快地在一条指令执行后予以响应。

7.16 填空题

（1）CPU 对输入输出设备的访问，采用按地址访问的形式。对 I/O 设备编址的方法，目前采用方式主要有：_____和_____，其中_____需要有专门的 I/O 指令支持。

（2）主机与外围设备之间的数据交换方式有_____、_____、_____和_____等几种。

（3）接口接收到中断响应信号 INTA 后，要将_____传给 CPU。

（4）通道的种类有_____、_____和_____三种。

（5）通道的工作过程可分为_____、_____和_____三部分。

第 8 章　外围设备

【内容摘要】

在计算机中除了 CPU 和主存外，其余的每一部分都可作为一个外围设备来看待。其功能是在计算机和其他机器之间，以及计算机与用户之间提供联系。由于外围设备的地位越来越重要，本章介绍一些常用的输入输出设备和外存储设备。

【学习要点】

- 外围设备的功能及分类
- 输出设备
- 输入设备
- 外围存储设备

8.1　外围设备概述

外围设备是计算机系统不可缺少的组成部分，用户在使用计算机系统时，接触最多的就是外围设备。外围设备是计算机和外部世界联系的桥梁。

8.1.1 外围设备的概念

外围设备（Peripheral Device）又叫做外部设备（External Device），在计算机硬件系统中，外围设备是计算机与外界交换信息的全部附属设备。如图 8-1 所示。外围设备的主要作用是：将外界的信息输入计算机；取出计算机要输出的信息；存储需要保存的信息和编辑整理外界信息以便输入计算机。外围设备一般由媒体、设备和设备控制器组成。设备控制器是控制该设备进行操作的控制部件，它接受中央处理机通过接口传送来的各种信息，并按设备的不同要求把这些信息传送到设备或从设备读出信息传送到接口。因为不同设备有不同的要求，所以要求各种专门的设备控制器与之配合。

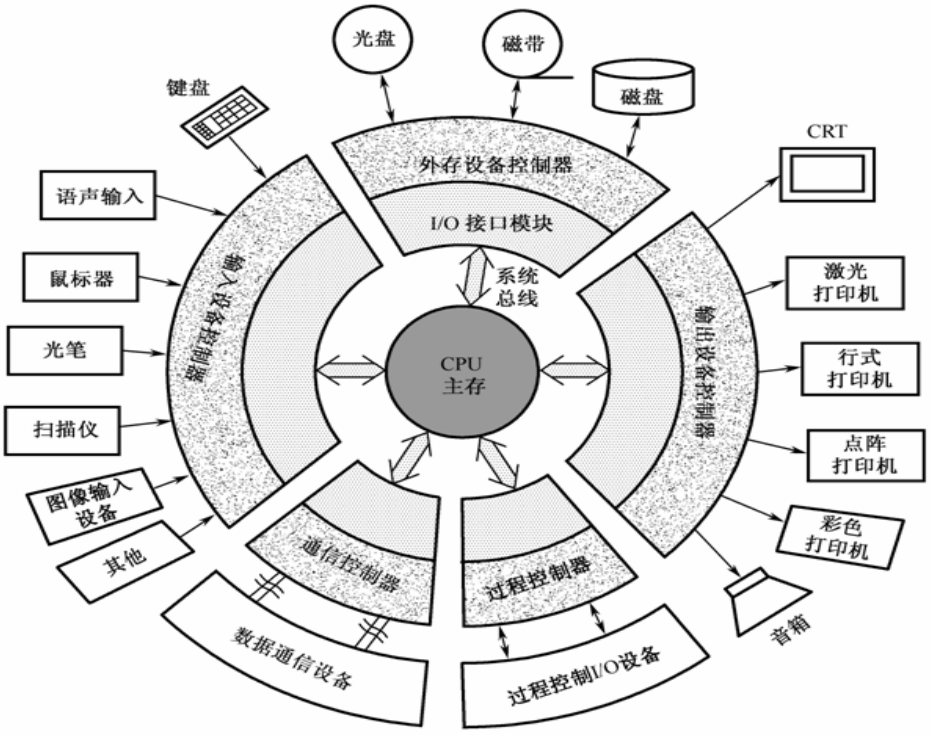


图 8-1　外围设备

8.1.2 外围设备的分类

外围设备大体分为输入设备、输出设备、外存设备、数据通信设备、过程控制设备五大类。每一种设备，都是在它自己的设备控制器控制下进行工作，而设备控制器则通过适配器（接口）和主机相连，并受主机控制。

1、输入设备

输入设备是人和计算机的最重要的接口，它的功能是把原始数据和处理这些数据的程序、命令通过输入接口输入到计算机中。因此，凡是能把程序、数据和命令送入计算机进行处理的设备都是输入设备。由于需要输入到计算机的信息多种多样，如字符、图形、图像、语音、光线、电流、电压等等，各种形式的输入信息都需要转换为二进制编码，才能为计算机所利用，因此不同输入设备在工作原理、工作速度上相差很大。输入设备包括字符输入设备（如键盘、条形码阅读器、磁卡机）、图形输入设备（如鼠标器、图形数字化仪、操纵杆）、图像输入设备（如扫描仪、传真机、摄像机）。

2、输出设备

输出设备同样是十分重要的人机接口，它的功能是用来输出人们所需要的计算机的处理结果。输出的形式可以是数字、字母、表格、图形、图像等。最常用的输出设备是各种类型的显示器、打印机和绘图仪等。

3、外存设备

在计算机系统中除了计算机主机中的内存储器（包括主存和高速缓冲存储器）外，还应有外存储器，简称“外存”。外存储器用来存储大量的暂时不参加运算或处理的数据和程序，因而允许速度较慢。一旦需要，可成批地与内存交换信息。它是主存储器的后备和补充，因此叫它为“辅助存储器”。外存的特点是存储容量大、可靠性高、价格低，在脱机情况下可以永久地保存信息，进行重复使用。外存按存储介质可分为磁表面存储器和光存储器。现在人们使用的磁表面存储器主要是磁盘和磁带。微机上使用的主要是硬磁盘存储器和软磁盘存储器。光盘存储器作为一种新型的信息存储设备已经在微机上普及使用。目前，可移动磁盘如移动硬盘和 U 盘已开始在微机系统中使用，为用户提供了很大的方便。

8.1.3 外围设备的功能

外围设备的功能是在计算机和其他机器之间以及计算机与用户之间提供联系。人操作计算机必须要进行人机对话，程序要输入，程序运行中所需要的数据也要输入，操作者要了解程序运行的情况，以便随时对出现的不正常情况进行干预和处理，计算机系统要把处理结果以操作者需要的方式输出，这些都要通过外围设备来实现。不少输入/输出设备，如键盘、显示器、软盘驱动器、打印机等就是提供这种手段的设备。外围设备就象计算机的五官和四肢一样，通过其接受外界的信息，并对处理的结果做出反应。

进入 21 世纪，随着计算机技术的飞速发展，对外围设备的要求也越来越高，外围设备正向电子化和智能化的方向发展。如尽量减少外围设备中机械部件的比重，采用伺服电机和大规模超大规模集成电路器件代替，使结构简化。外围设备的控制采用微处理器、单片机和专用大规模集成电路器件，使外围设备具有处理机的功能。这样不仅使外围设备的体积缩小，增强了功能，提高了可靠性，而且降低了成本。这些在打印机、显示器、磁盘存储器、扫描仪中反映十分明显。

8.2 输入设备

输入设备是向计算机输入数据和信息的设备，是计算机与用户或其他设备通信的桥梁，用于把原始数据和处理这些数的程序输入到计算机中。现在的计算机能够接收各种各样的数据，既可以是数值型的数据，也可以是各种非数值型的数据，如图形、图像、声音等都可以通过不同类型的输入设备输入到计算机中，进行存储、处理和输出。

8.2.1 键盘

键盘（Keyboard）是常用的输入设备，它由一组排列成阵列形式的按键组成，目前常用的键盘有 104 个按键，包括数字键、字母键、符号键、功能键及控制键等。每一个按键在计算机中都有它的惟一代码。当按下某个键时，键盘接口将该键的二进制代码送入计算机主机，并将按键字符显示在显示器上。当快速大量输入字符而主机来不及处理时，先将这些字符的代码送往内存的键盘缓冲区，然后再从该缓冲区中取出进行分析处理。键盘接口电路多采用单片微处理器，由它控制整个键盘的工作，如上电时对键盘的自检、键盘扫描、按键代码的产生、发送及与主机的通讯等。

1、键盘的分类

（1）触点式按键

早期的键盘几乎全部是机械式键盘，每个按键的下部有两个触点，平时两个触点没有接触，相当于断路，该键被按下后两触点导通。这种键盘手感差，易磨损，故障率较高。

（2）电容式按键

这种按键通过改变电容器电极之间的距离，产生电容的变化。每个按键内活动极、驱动极与检测极组成两个串联的电容器。键按下时，上下两极片靠近，极板间距离缩短，来自振荡器的脉冲信号被电容耦合后输出。反之，则无信号输出。这种键使用中不存在磨损、接触不良等问题，耐久性、灵敏度和稳定性比较好。为了避免电极间进入灰尘，电容式按键开关采用密封组装。电容式键盘手感好，寿命长，目前使用的计算机键盘多为电容式无触点键盘。

2、键盘接插件标准

目前 PC 上常用的键盘插口有 2 种，一是比较老式的直径 13mm 的 5 芯 PC 键盘插口，如图 8-2 右所示；第二种是最常用的直径 8mm 的 6 芯 PS / 2 键盘插口如图 8-2 左所示。

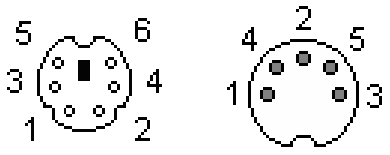


图 8-2　键盘接口图

8.2.2 鼠标

鼠标（Mouse）是一种手持式屏幕坐标定位设备，它是适应菜单操作的软件和图形处理环境而出现的一种输入设备，特别是在现今流行的 Windows 图形操作系统环境下应用鼠标方便快捷。常用的鼠标有两种，一种是机械式的，另一种是光电式的。

机械式鼠标的底座上装有一个可以滚动的橡胶球，鼠标内置了 3 个滚轴：X 方向滚轴和 Y 方向滚轴，另 1 个是空轴。这 3 个滚轴都与可以滚动的橡胶球接触，并随着橡胶球滚动一起转动。X，Y 滚轴上装有带孔的译码轮，它的转动会阻断或导通 LED 发出的光线，在光敏晶体管上产生表示位移的脉冲。光标和鼠标的移动方向是一致的，而且移动的距离成比例。

光电式鼠标的底部装有两个平行放置的小光源，需要与一块画满小方格的反射板配合使用鼠标在反射板上移动，光源发出的光经反射板反射后，由鼠标接收，并转换为电移动信号送入计算机，使屏幕的光标随之移动。其他方面与机械式鼠标一样。

鼠标有两个键的，也有三个键的。由于鼠标所配的软件系统不同，对上述三个键的定义有所不同。一般情况下，鼠标左键可在屏幕上确定某一位置，该位置在字符输入状态下是当前输入字符的显示点；在图形状态下是绘图的参考点。在菜单选择中，左键可选择菜单项，也可以选择绘图工具和命令。当作出选择后系统会自动执行所选择的命令。鼠标能够移动光标，选择各种操作和命令，并可方便地对图形进行编辑和修改，但却不能输入字符和数字。

鼠标接口有传统的 COM、PS/2 和新型的 USB 三种：

（1）串行通信口鼠标

串口鼠标使用 9 针 D 型接口，采用 RS-232C 标准进行通信。这种鼠标的电源由串口的 RTS 信号提供，GND 作为地线，使用 TXD 发送数据，DTR 作为联络信号线。大多数鼠标采用 7 位数据位、1 位停止位，无奇偶校验方式、以 1200-2400 b/s 的速率发送数据。如图 8-3 左所示。

（2）PS / 2 鼠标

PS / 2 鼠标最早用在 IBM PS / 2 系列微机上而得名。它使用专用的鼠标接插座（6 芯 DIN 型头），安装灵活方便，不占用串口资源。现在的 PC 机主板都有支持 PS / 2 鼠标接口的插座。它们的接插件及信号如图 8-3 右所示。

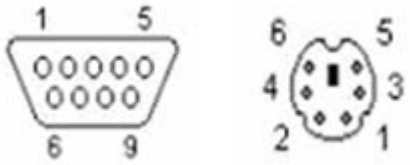


图 8-3　鼠标接口图

（3）USB 鼠标

由于 USB 接口的普及，USB 接口的鼠标使用越来越广泛。PS/2 接口不支持热拔插，因为它并不像 USB 接口那样具有保护电路，热拔插很容易 PS/2 接口容易电路中的电容被击穿，从而导致 PS/2 接口损坏，而 USB 鼠标却支持热插拔。另外 USB 还有支持即插即用的优点，所以 USB 接口已经成为计算机的最主要的接口方式。

8.2.3 其他输入设备

1、光学标记阅读机

光学标记阅读机是一种用光电原理，对已经印刷好或书写好的文字稿样进行扫描，获得文字图像信息，并通过计算机软件决策出识别的结果。首先，对输入原稿上的文字进行光学扫描。然后，对取样点的文字进行二值化；其次，对二值化图像进行预处理，包括消除污染、杂音和倾斜校正等。预处理后的信息经过特征抽取，得到图像的特征。它是识别的依据，如笔画长度、角度、端点、笔画分布、四周特征等。这些特征以多维数据的形式表示，作为识别标准的学习图形，也以多维矢量形式存于分类辞书和判定辞书中；最后，将输入汉字图像的特征与辞书中的标准图像特征比较，最接近的被认为是相匹配的，并作为判定的结果输出。

2、扫描仪

扫描仪是一种光、机、电一体化的产品，用于捕获影像(照片，文字页，图形和插画等)，并将之转换为电脑可以显示、编辑、储存和输入的数据格式，是继键盘和鼠标之后的第三代主要的电脑输入设备。扫描仪刚投放市场时曾因其过高的价格而使人望而却步，然而随着扫描技术的不断改进和发展，以及扫描仪价格的日趋大众化，扫描仪正逐步成为计算机不可缺少的外设产品之一。

感光器件的工作是将感应到的光信号转换成电信号。大功率日光灯管用于对文字或者图像进行强烈的照射以达到完全感应的作用。驱动马达的作用是带动日光灯管，将光一步步扫过需要的物品。驱动皮带是与驱动马达一起带动日光灯管的装置。模数信号转换器是将模拟电信号转换成数字信号以供电脑识别，其工作过程非常简单，电源发出的光被图样反射后，反射光（或透射光）被图样调制并被光电传感器转换成数字信号，数字信号经接口送到主机中可进行下一步处理。光源和传感器在步进电机的带动下可扫过整个图样。扫描仪的技术指

标主要有分辨率、色彩位数、幅面和接口方式。

（1）分辨率

分辨率是扫描仪所能够捕捉到的真实的分辨率，平板式扫描仪的光学分辨率为 CCD 的点数除以扫描仪的最大扫描宽度。分辨率是扫描仪最重要的技术指标，分辨率高，扫描精度自然会提高，但扫描速度会相应降低。一般来说，家用或者普通办公室文档使用 300 d/i×600 d/i 的扫描仪即可满足基本需求。

（2）色彩位数

色彩位数（bit）表示扫描仪在采样时，它捕捉的每个像素上检测出的最大颜色或灰度级。从理论上说，色彩位数增加，可以捕捉到的细节数量增加，密度范围增加。但是由于采用的 CCD（Charge Couple Device，电荷耦合器件）品质不同，信噪比不同，所以不能一概而论。对于专业扫描仪来说，应该有对应关系，36 位的扫描仪应该有 3.4D 的密度范围，而 42 位的扫描仪则应该有 3.7D 的密度范围，否则就不是真正的色彩位数。色彩位数越高，所表现的色彩种类就越丰富自然。常见的扫描仪色彩位数为 24 位，30 位，36 位等。普通用户可选择 24 位或 30 位的扫描仪。

（3）幅面

幅面表示扫描仪扫描普通文档或照片的大小。幅面大的扫描仪，对大面积图像的扫描比较方便，但价格也就高得多。家用扫描仪多用来扫描普通文档或者相片，所以幅面一般不超过 A4，因此 A4 或者 A4 加长的扫描仪即可满足家庭用户的使用。幅面达到 A1 或 A0 的扫描仪，一般称为大幅面扫描仪。

（4）接口方式

接口方式是指扫描仪与计算机之间采用的接口类型。常用的有并行打印机接口、USB 接口、SCSI 接口。SCSI 接口的传输速度快，而采用 USB 接口或火线接口则更简便。采用 SCSI 接口的扫描仪由于价格偏高，设置相对复杂，所以目前一般应用在设计领域。USB 接口大大方便了多平台用户，只要你的计算机有 USB 接口，只需要安装驱动程序就可以使用了。但是，由于受到 USB 接口速度的限制，故采用 USB 接口的扫描仪一般为民用型或普通办公型。

3、语音设备

语音不仅是人类交流信息最自然、最有效的手段，而且是人机通信最有效的一种方法，它摆脱了各种传统输入方法的弱点。

语音识别和文字识别都属于模式识别的范畴，有共同的基本原理，但输入技术和工具则大不相同。语音输入方式是利用声音的物理模型，通过语音分析手段，预先将一些语音的特征参数提取，并存于计算机系统中。当讲话者语音输入时，处理系统就对该信号进行特征参数抽取，然后和已储存的特征参数进行比较，通过逻辑判别或距离测量，对输入的语音进行识别。

通过话筒或录音输入设备输入的语音信号是模拟量，经放大、滤波和进行 A/D 转换后，得到这个语音信息的数字编码，才能送入计算机进行处理。精确地测定一个语音的起始点和结束点，是语音参量与样音参量比较的基础。为了便于测量，常按每 10*ms* 为一帧的时间顺序来进行，并由粗判和细判两个步骤来具体确定。首先进行粗判： 由某帧开始，若连续 NF 帧的参数满足粗判有声条件，则判为头部有声；此后，若连续 NR 帧满足粗判无声条件，则判为尾部无声。然后再进行细判： 从粗判起始点前 25 帧开始向后搜索，当从某帧开始连续 SNF 帧满足细判有声条件，该帧确定为起始点；尾部判定从粗判无声后的 15 帧开始向前搜索，连续 SNR 帧为无声，该帧被判定为结束点。

当进行语音识别时，时间的扭曲处理是极为重要的一步。准确地测定语音的起始点和终点，并从中提取了该区间有关语音参量之后，还不能立即和样音进行比较。这是因为，即使

是同一个人重复发同一个音，每次发音也不可能完全一致，即各帧参量不可能完全相同，为了消除各帧由离散性带来的影响，使被测语音尽可能地与样音对齐，就必须将它们中的一个进行时间扭曲处理。经过扭曲处理后，被测语音强度曲线和样音强度曲线的吻合程度得到改进。这时，就可以转入识别处理，即选出各参量最为接近的样音的语言信号作为最后输出。

此外，还有一种识别方法是按照一定规则把测得的语音特征进行计算，译成单词、短语和句子。这种方法较为复杂，但前途极广。

8.3 输出设备

输出设备是人与计算机交互的一种部件，用于数据的输出。它把各种计算结果数据或信息以数字、字符、图像、声音等形式表示出来。常见的有显示器、打印机、绘图仪、影像输出系统、语音输出系统、磁记录设备等。本节只简要介绍常用的输出设备显示器和打印机。

8.3.1 显示器

以可见光的形式传递和处理信息的设备叫显示设备，是目前计算机系统中应用最广泛的人机界面设备。按显示设备所用的显示器件分类，有阴极射线管(CRT)显示器、液晶显示器(LCD)、等离子显示器等。

阴极射线管显示器采用大家所熟悉的电视机显像管，有黑白和彩色两种。目前，大多数显示设备使用彩色 CRT 显示器，如图 **8-4** 所示。液晶和等离子显示器都是平板式显示器，它们的特点是体积小、功耗低，是很有发展前途的新型显示器。

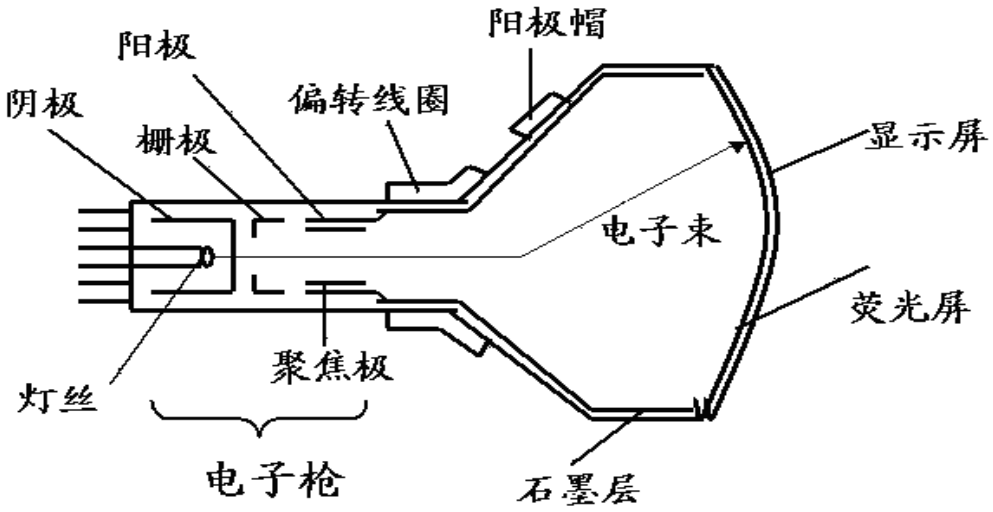


图 8-4 阴极射线管显示器结构图

1、显示器的分类

在阴极射线管显示器中，以扫描方式不同，分成光栅扫描和随机扫描两种显示器；以分辨率不同，分成高分辨率显示器和低分辨率显示器；以 CRT 荧光屏对角线的长度分类，有 12 英寸、14 英寸、16 英寸、19 英寸等多种。另外，阴极射线管显示器还可分为字符显示器和图形显示器。字符显示器只能显示字符，不能显示图形，一般只有两种颜色。图形显示器不仅可以显示字符，而且可以显示图形和图像。图形是指工程图，即由点、线、面、体组成的图形；图像是指景物图。不论图形还是图像在显示器上都是由像素（光点）所组成。显示器屏幕上的光点是由阴极电子枪发射的电子束打击荧光粉薄膜而产生的。彩色显示器的显像管的屏幕内侧是由红、绿、蓝三色磷光点构成的小三角形（像素）发光薄膜。由于接收的电子束强弱不同，像素的三原色发光强弱就不同，可以产生一个不同亮度和颜色的像素。当电

子束从左向右、从上而下地逐行扫描荧光屏，每扫描一遍，就显示一屏，称为刷新一次，只要两次刷新的时间间隔少于 0.01*s*，则人眼在屏幕上看到的就是一个稳定的画面。

2、显示器的技术指标

（1）逐行 / 隔行扫描

受扫描频率的限制，隔行扫描显示器在低分辨率下逐行扫描，在高分辨率下改为隔行扫描。此时，对同一屏幕的图像先扫描奇数行，再扫描偶数行。扫描奇数行时，电子束可能因偏移扫描到偶数行，反之，扫描偶数行时的电子束可能扫描到奇数行，造成水平线上的抖动。采用逐行扫描后，有效地避免了上述不足。

（2）分辨率与灰度级

分辨率是指显示器所能表示的像素个数。像素越密，分辨率越高，图像越清晰。分辨率取决于显像管荧光粉的粒度、荧光屏的尺寸和 CRT 电子束的聚焦能力。刷新存储器要有与显示像素数相对应的存储空间，用来存储每个像素的信息。

灰度级是指黑白显示器中所显示的像素点的亮暗差别程度，在彩色显示器中则表现为颜色的不同。灰度级越多，图像层次越清晰逼真。灰度级取决于每个像素对应刷新存储器单元的位数和 CRT 本身的性能。如果用 4 位表示一个像素，则只有 16 级灰度或颜色；如果用 8 位表示一个像素，则有 256 种灰度或颜色。字符显示器只用“0”、“1”两级灰度就可表示字符的有无，这种只有两级灰度的显示器称为单色显示器。具有多种灰度级的黑白显示器称为多灰度级黑白显示器。

（3）刷新频率

只有当 CRT 的电子束打到显示屏上的荧光粉时，荧光屏相应的点才会发亮。这段发亮的时间很短，一般只能维持几十毫秒。在这么短的时间内，眼睛是感觉不到的。为了使眼睛能感觉到稳定的图像显示，必须使电子束不停地重复扫描，这个重复扫描的过程叫做刷新。而刷新频率就是每秒屏幕刷新的次数。刷新频率越低，图像闪烁得就越明显。一般显示器要求在 1024×768 的分辨率下要能达到 75*Hz* 的刷新率。与这个指标有关的还有行频和带宽。行频是每秒扫描的行数，如 65*K/S*，85*K/S* 等。带宽是每秒扫描像素的点数，常见的是几十兆赫兹，高性能显示器的带宽在 100*MHz* 以上。

（4）随机扫描和光栅扫描

控制电子束在 CRT 屏幕上随机地运动，从而产生图形和字符的方式称为随机扫描。电子束只需在需要显示的地方扫描，而不必扫描全屏幕，因此这种扫描方式的速度快，图像清晰。高质量的图形显示设备一般采用随机扫描的方式。由于这种扫描的控制电路与电视标准不同，因此驱动电路较复杂，价格较贵。

光栅扫描同电视系统的扫描方式。在电视中图像充满整个画面，因此要求电子束扫过整个屏幕。光栅扫描是从上至下顺序扫描，采用逐行扫描和隔行扫描两种方式。光栅扫描的缺点是冗余时间多，分辨率不如随机扫描方式，但由于电视技术业已成熟，计算机系统中除高质量图形显示器外，大部分字符、图形、图像显示器都采用光栅扫描方式。光栅扫描的缺点是冗余时间多，分辨率不如随机扫描方式，但其扫描方式与电视系统相同，控制电路比较简单，技术比较成熟，因此成本较低。目前大部分字符、图像、图形显示器采用光栅扫描方式。

显示器是通过“显示接口”及总线与主机连接，待显示的信息（字符或图形图像）是从显示缓冲存储器（一般为内存的一个存储区，占 16*kB*）送入显示器接口的，经显示器接口的转换，形成控制电子束位置和强弱的信号。受控的电子束就会在荧光屏上描绘出能够区分出颜色不同、明暗层次的画面。显示器的两个重要技术指标是：屏幕上光点的多少，即像素的多少，称为分辨率；光点亮度的深浅变化层次，即灰度，可以用颜色来表示。分辨率和灰度的级别是衡量图像质量的标准。

常用的显示接口卡有多种，如 CGA 卡、VGA 卡、MGA 卡等。以 VGA(VideoGraphicsArray)

视频图形显示接口卡为例，标准 VGA 显示卡的分辨率为 640×480，灰度是 16 种颜色；增强型 VGA 显示卡的分辨率是 800×600、960×720，灰度可为 256 种颜色。所有的显示接口卡只有配上相应的显示器和显示软件，才能发挥它们的最高性能。

8.3.2 打印机

打印机是计算机最基本的输出设备之一、是一种复杂而精密的机械电子装置。它主要用于打印结果及程序，还可用于打印统计图表和描绘图形。随着计算机技术的发展，对打印机提出了更高的要求，如高速度、低噪声、高分辨率、输出图形图像化和彩色化等。

打印机按印字方式可分为击打式和非击打式两类。击打式打印机是利用机械动作将字体通过色带打印在纸上，根据印出字体的方式又可分为活字式打印机和点阵式打印机。活字式打印机是把每一个字刻在打字机构上，可以是球形、菊花瓣形、鼓轮形等各种形状。点阵式打印机是利用打印钢针按字符的点阵打印出字符。每一个字符可由 m 行×n 列的点阵组成。一般字符由 7×8 点阵组成，汉字由 24×24 点阵组成。点阵式打印机常用打印头的针数来命名，如 9 针打印机、24 针打印机等。

非击打式打印机是用各种物理或化学的方法印刷字符的，如静电感应，电灼、热敏效应，激光扫描和喷墨等。其中激光打印机和喷墨式打印机是目前最流行的两种打印机，它们都是以点阵的形式组成字符和各种图形。激光打印机接收来自 CPU 的信息，然后进行激光扫描，将要输出的信息在磁鼓上形成静电潜像，并转换成磁信号，使碳粉吸附到纸上，加热定影后输出。喷墨式打印机是将墨水通过精制的喷头喷到纸面上形成字符和图形的。下面分别介绍针式打印机、激光打印机和喷墨打印机。

1、针式打印机

针式打印机是通过打印针击打色带来进行工作的。对于针式打印机来说，最大的好处是对纸张来者不拒，如普通纸、旧挂历纸的背面、旧塑膜挂历的衬纸、平整的商品包装纸、街上散发的广告单背面等，尤其对打印多层介质，如多联发票等，它有着激光打印机与喷墨打印机无可取代的优势，还有其耗材省和维护简单的优点，使得针式打印机有着比较大的市场。

(1) 针式打印机组成

针式打印机主要由打印头单元、小车单元、送纸单元及控制单元组成。其中，打印头单元是针式打印机机械部分的核心，具有体积小、结构紧密、精度高的优点。打印头的种类很多，常用的有拍合式打印头和永磁式打印头。当打印单元接到打印命令时，打印针向外撞击，色带的墨迹打印到纸上，从而在打印纸上形成字符或图像。小车单元的任务是打印头送到指定位置，以使打印头的打印针打到正确位置。送纸单元主要是在打印过程中提供纸张输送或退出的装置。控制单元是打印机的控制中心。

(2) 针式打印机的印字原理

针式打印机的印字原理是将要输出的 ASCII 码字符、汉字或图形转换成对应的点阵码，再根据点阵码由驱动电路驱动钢针击打色带，从而印出字符或图形。点阵码用二进制数表示，“1”表示要打的点，“0”表示不打点。一个 ASCII 码字符的点阵格式有 5×7、7×7 和 9×9 几种。图 8-5 是字符 A 对应的点阵码。

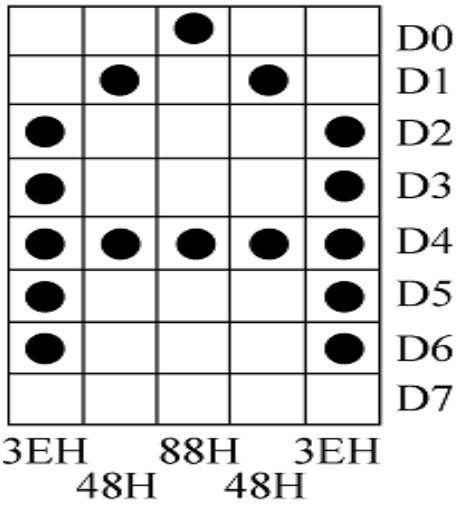


图 8-5　A 对应的点阵码

汉字点阵格式有 16×16、24×24、32×32 和 40×40 几种。当打印汉字钢针数不够时，可以采用多次扫描实现。点阵式打印机有两种打印方式，一是字符打印方式，主机将字符的编码（ASCII 码或汉字内码）通过接口传送给打印机，

在打印机的字符库中找出对应的点阵码，再根据点阵码驱动打印头冲击色带印出字符。二是图形打印方式，主机将图形的点阵码送给打印机，由驱动电路冲击色带印出图形的点阵。

2、喷墨打印机

喷墨打印机具有体积小、重量轻、工作噪声低的特征。目前，在与 PC 相配合的应用领域里，喷墨打印机已取代了针式打印机。其原理如图 **8-6** 所示。

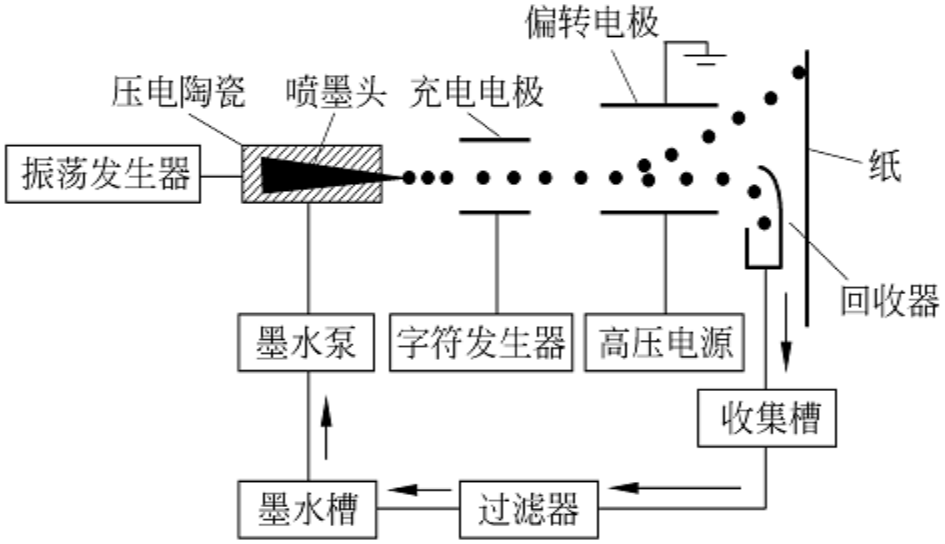


图 8-6　喷墨打印机原理图

喷墨打印机主要由喷头、墨盒、清洁单元、小车单元、送纸单元和控制单元组成。其中小车、送纸、控制三个单元与针式打印机大同小异；喷头与针式打印机打印头的作用相同。喷头和墨盒的结构可分为两类：一类是喷头和墨水盒是一体化结构；另一类是喷头和墨盒分离结构。前者整体结构简单，体积小，但打印机消耗费用大，墨盒中的墨水用完后要连同喷头一起更换；而后者则只需更换墨盒即可，这样就可大大节省费用。

工作时，导电的墨水在墨水泵的高压作用下进入喷嘴，通过喷嘴形成一束极细的高速射流，射流通过高频振荡发生器断裂成连续均匀的墨水滴流。在充电极上，施加一个静电场给墨水充电，所充电荷的多少随墨水喷在纸上的高低位置而变化。在充电电极上所加的电压越高，充电电荷就越多。电荷一直保持到墨点落到记录纸上为止。带不同电荷的墨点通过加有恒定高压偏转电极形成的电场后垂直偏转到所需的位置。若在垂直线上某处不需喷点，则相应的墨点不充电，这些墨点在偏转电场中不发生偏转而按原方向射入回收器。当一列字符印完后，喷墨头以一定的速度沿水平方向由左向右移动一列距离。依次下去，即可印刷出一个字符，并由若干个字符构成一个字符行。喷墨打印机有以下三个方面的指标：

（1）打印精度

打印精度的度量单位为每英寸可打印的点数 D/I(Dot Per Inch)。所谓 d/i，是指打印机在，是衡量打印质量的一个重要标准，也是一个最为基本的判断打印机分辨率的指标。至少应有 300 d/i 的分辨率，才能使打印效果得到基本保证。一般应用时，d/i 值也不一定是越大越好，720 d/i 左右就比较合适了。

（2）打印速度

打印速度的度量单位为 p/m(Page Per Minute)。p/m 是指打印机每分钟可打印多少页，是衡量打印机打印速度的一个重要指标。但要注意，打印机说明书上所标明的 p/m 值是指该机所能达到的最高打印速度，实际使用时，这个值会受到多种因素的影响，肯定会低于标称值。

（3）打印幅面

喷墨打印机的打印幅面一般以 A4 为主。普通家庭用户和中小型办公用户使用 A4 幅面的喷墨打印机，可以满足绝大部分应用要求。A3，A2 幅面的喷墨打印机一般用于 CAD，广

告制作，艺术设计，印刷出版等行业。

3、激光打印机

激光打印机是发展较快的一种非击打式打印机。它使用成熟的技术，具有很高的稳定性，打印速度快，安静，使用成本低廉，并且输出接近印刷的质量。其原理框图如图 **8-7** 所示。

当计算机主机向打印机发送数据时，打印机首先将接收到的数据暂存在缓存中，当接收到一段完整的数据后，再发送给打印机的处理器，处理器将这些数据组织成可以驱动打印引擎动作的信号流，对于激光打印机而言，这个信号流就是驱动激光头工作的一组脉冲信号。

激光打印机的核心技术就是所谓的电子成像技术，这种技术结合了影像学与电子学的原理和技术以生成图像，核心部件是一个可以感光的硒鼓。激光发射器所发射的激光照射在一个棱柱形反射镜上，随着反射镜的转动，光线从硒鼓的一端到另一端依次扫过（中途有各种聚焦透镜，使扫描到硒鼓表面的光点非常小），硒鼓以 1/300 英寸或 1/600 英寸的步幅转动，扫描又在接下来的一行进行。硒鼓是一只表面涂覆了有机材料的圆筒，预先带有电荷，当有光线照射时，受到照射的部位会发生电阻的变化。计算机所发送来的数据信号控制着激光的发射，扫描在硒鼓表面的光线不断变化，有的地方受到照射，电阻变小，电荷消失，也有的地方没有光线射到，仍保留有电荷，最终，硒鼓表面就形成了由电荷组成的潜影。

墨粉是一种带电荷的细微塑料颗粒，其电荷与硒鼓表面的电荷极性相反，当带有电荷的硒鼓表面经过显影辊时，有电荷的部位就吸附了墨粉颗粒，潜影就变成了真正的影像。硒鼓转动的同时，另一组传动系统将打印纸送进来，经过一组电极，打印纸带上了与硒鼓表面极性相同但强得多的电荷，随后纸张经过带有墨粉的硒鼓，硒鼓表面的墨粉被吸引到打印纸上，图像就在纸张表面形成了。此时，墨粉和打印机仅仅是靠电荷的引力结合在一起，在打印纸被送出打印机之前，经过高温加热，塑料质的墨粉被熔化，在冷却过程中固着在纸张表面。

将墨粉传给打印纸之后，硒鼓表面继续旋转，经过一个清洁器，将剩余的墨粉去掉，以便进入下一个打印循环。

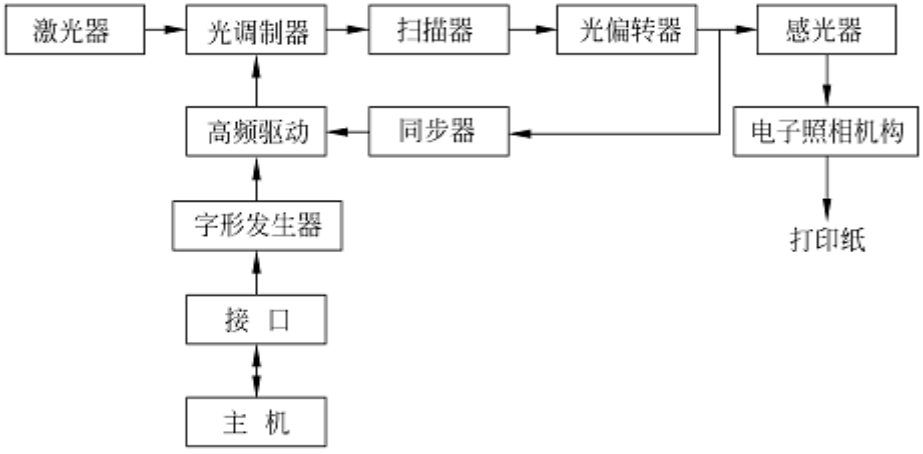


图 8-7　激光打印机原理框图

激光扫描控制系统包括激光器、光调制器、扫描器、光偏转器等部件。激光器是激光打印机的光源，它具有相干性高、方向性好、单色性强、能量集中等优点，并能进行高速调制和偏转控制。光调制器对光的传播方向和强度进行控制，控制的方法有机械、电光和声光等几种。由于声光调制器完全满足打印机调制频率带宽的要求，因而被普遍采用。要把调制过的激光束射到感光体上形成图形，必须使激光束沿光导鼓轴线方向产生横向移动和纵向移动，纵向移动利用光导鼓的旋转实现，横向移动则靠扫描器来实现。光偏转器配合扫描器，使激光束聚焦，并使聚焦轨迹在光导鼓表面上匀速直线运动。

电子照相转印系统应用电子照相技术记录激光扫描信息，经过显影转印到普通纸上。电子照相转印系统包括感光鼓、转印电极、充电极、显影器、清洁器及输纸机构。电子照相转

印系统的工作过程归纳为充电、曝光、显影、转印、定影和清洁等步骤。充电过程使感光鼓的光导层表面带电。根据所需印出的字符或图形，激光束的感光鼓表面上产生静电现象，称为曝光过程。由于光导层的光电效应，受激光照射的鼓面部分正电荷流向地，电荷消失后成为良导体。鼓面未被照射的部分，即字符或图像以外的地方，仍保留电荷，这样就生成不可见的静电现象。

由以上原理可以看出激光打印机与针式、喷墨打印机的一个本质的区别在于：激光打印机打印一次成像一整页，是逐页打印；而针式和喷墨打印机都是打印头一次来回打印一行，是逐行打印。因此，相同打印要求下，激光打印机的打印速度要比针式打印机和喷墨打印机要快，这也是激光打印机的一个优势所在。

由此可见：针式打印机由于采用的是机械击打式的打印头，因此穿透力很强，能打印多层复写纸，具备拷贝功能，另外还能打印不限长度的连续纸。使用的耗材是色带，在三种打印机中是最廉价的一种。其缺点就是体积、重量都较大，打印噪音大，精度低，速度慢，一般无打印彩色图像功能。适合有专门要求的专业应用场合，例如财务、税务、金融机构等等。常见的机型有 EPSON 的 LQ 系列，如 LQ－680K；STAR 的 AR、CR 系列，如 AR－6400 等。喷墨打印机打印精度高，通常都能打印彩色图像，而且体积及重量都可以做的非常小巧，甚至能随身携带打印，打印时的噪音也很小。但使用的消耗材料－墨水，是三种打印机中相对来说最为昂贵的！而且，想要打印精美的图像，还要使用同样昂贵的专用打印纸才能有很好的打印效果。因此喷墨打印机的使用成本很高！同时，也不具备拷贝和打连续纸功能。适合对打印质量要求高但数量较小的场合，如家庭，小型办公室等等。常见的机型有 EPSON 的 STYLUS PHOTO 系列，如 STYLUS PHOTO 900；CANON 的 i 系列，如 i6500、i355 等。激光打印机的打印精度也很高，基本上与喷墨打印机无太大区别。它使用的耗材－硒鼓，其成本介于针式打印机和喷墨打印机之间。同样也能打印彩色图像，且对打印介质的要求没有喷墨打印机那么高。打印的速度是三种打印机中最快的，而且噪音也很小。但体积和重量相对喷墨打印机要大。也只能逐页打印，无拷贝和打印连续纸功能。适合打印数量大，任务重的场合，如大型商务机构，设计、印刷领域等等。常见的机型有惠普 HP 的 LaserJet 系列，如 LaserJet1010，Color LaserJet 8550 等。

8.4 外存储设备

外存储设备的特点是容量大、成本低，可以永久保存数据，是主存的重要补充。外存目前主要有磁表面存储器和光存储器两类，如磁盘、光盘等。

8.4.1 磁表面存储器的原理

磁表面存储器是利用磁性材料在不同的磁场作用下具有的两个稳定剩磁状态来记录信息的。其利用一层仅有几微米甚至不到 1 微米的表面磁介质作为逻辑信息的媒体，以磁介质的两种不同的剩磁方向变化的规律来表示二进制数字信息，如图 **8-8** 所示。

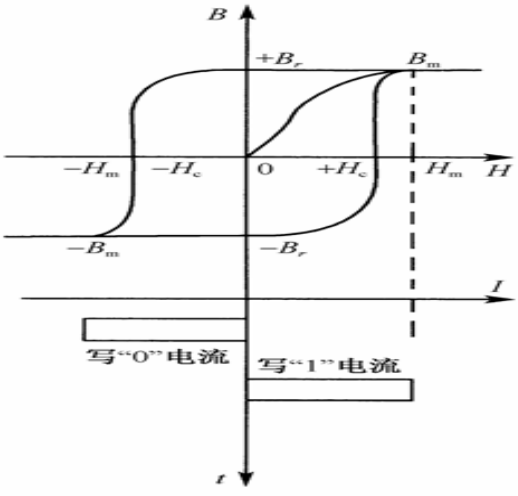


图 8-8 磁性材料的物理特性

磁记录的过程是一种电磁信息的转换过程。写入时，首先将数字代码信息以电流的形式输入到磁头线圈中去，然后转换成磁场去磁化磁介质，最终将数字代码信息以磁化状态的形式记录在磁介质上。读出时，首先通过磁头将磁化状态转换成电信号，最后还原成数字形式输出。所谓记录方式，是指以写入电

流波形的不同方式记录数字信息“0”和“1”，即按某种规律将一串二进制数字信息变换成磁层中相应的磁化元状态，用读写电路实现这种转换，如图 **8-9** 所示。

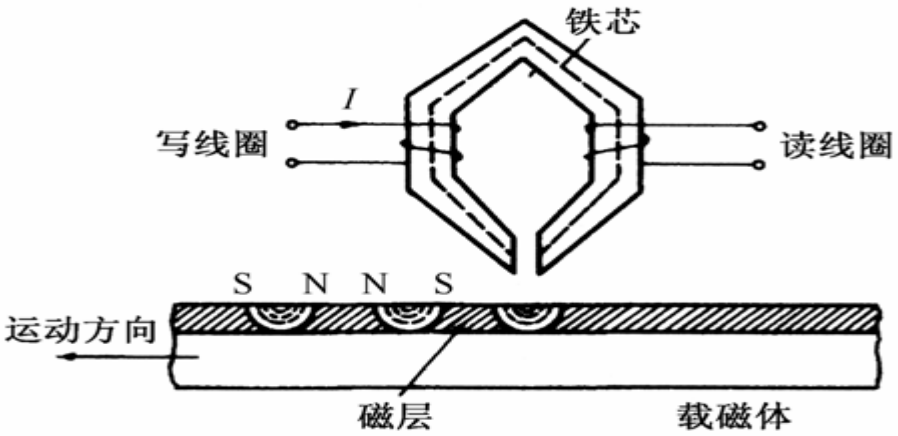


图 8-9 磁表面存储器的读写操作图

8.4.2 磁记录方式

在磁记录设备中，由于写入电流的幅度、相位、频率变化不同，从而形成了不同的记录方式，主要有归零制（RZ）、不归零制（NRZ）、调相制（PM）、调频制（FM）和改进的调频制（MFM）等几大类。下面以数字序列 10001110 为例来说明各种方式的记录原理。

1、归零制

归零制的电流波形如图 **8-10** 所示。

归零制记录方式的特点是： 正向电流代表“1”，负向电流代表“0”，不论某存储元记录的代码是“0”还是“1”，记录电流都要恢复到零电流（即没有电流）。

这种记录方式简单易行，但记录密度低，抗干扰能力差，往往把各种干扰的电流信号同时写入。

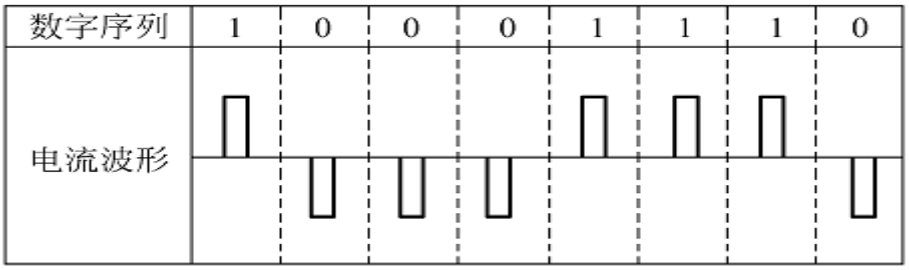


图 8-10 归零制波形图

2、不归零制

不归零制有两种方式，一种是一般的不归零制，另一种是见“1”就翻不归零制。一般的不归零制电流波形如图 **8-11** 所示。

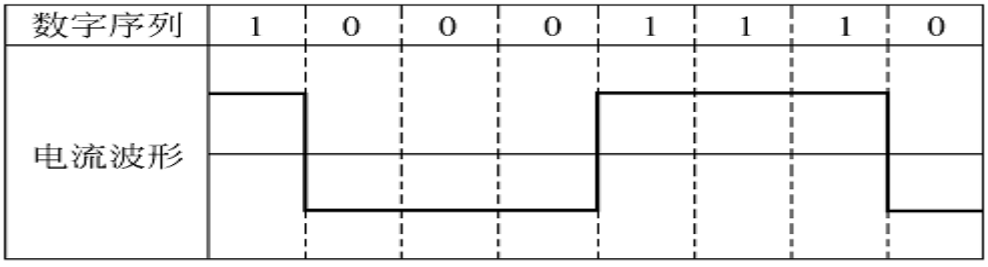


图 8-11 一般的不归零制波形图

这种方式在记录数据时，磁头线圈中不是有正向电流流过，就是有反向电流流过，即磁头线圈总是有电流流过。它的特点是： 对连续记录的“1”或“0”，写电流不改变方向。因此，这种记录方式比归零制减少了磁化翻转的次数。

见“1”就翻的不归零制的电流波形如图 **8-12** 所示。同样，这种方式在记录数据时，磁头线圈中始终有电流流过。和一般的不归零制的显著不同之处是： 流过磁头的电流在记录“1”时改变方向，在记录“0”时，电流方向不变。

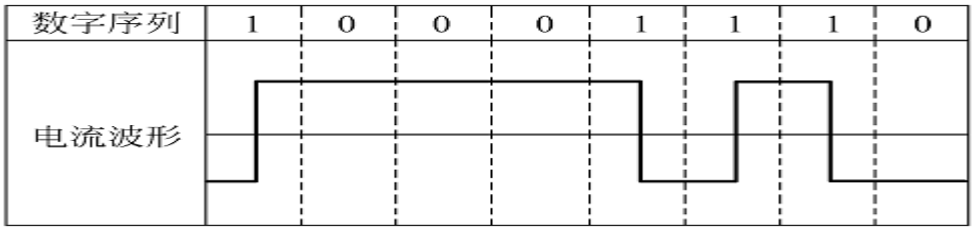


图 8-12 见“1”就翻不归零制波形图

3、调相制

调相制记录方式是利用电流两个相位相差 180° 来表示数字信息“0”和“1”。在一个位周期的中间位置，电流由负到正变化为“1”，由正到负为“0”，即利用电流相位的变化进行写入“1”或写入“0”的操作，所以通过磁头中的电流方向一定要改变一次。这种记录方式中“1”和“0”的读出信号相位不同，抗干扰能力较强。另外，读出信号经分离电路可提取自同步脉冲，因此具有自同步能力。调相制的电流波形如图 **8-13** 所示。

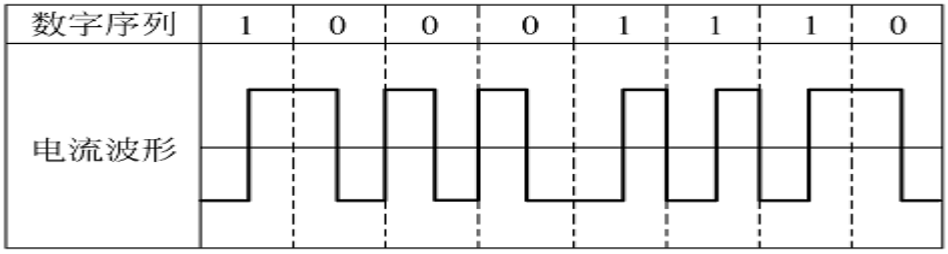


图 8-13 调相制波形图

4、调频制

调频制的电流波形如图 **8-14** 所示。这种方式在记录数字“1”时，电流不仅在位周期的中心位置翻转一次，而且在位与位之间也发生翻转。在记录数字“0”时，在位位置的中心不发生翻转，但在位与位之间的边界要翻转一次。调频制在记录数据“1”时，其翻转频率是记录“0”的两倍，所以又称为倍频制。

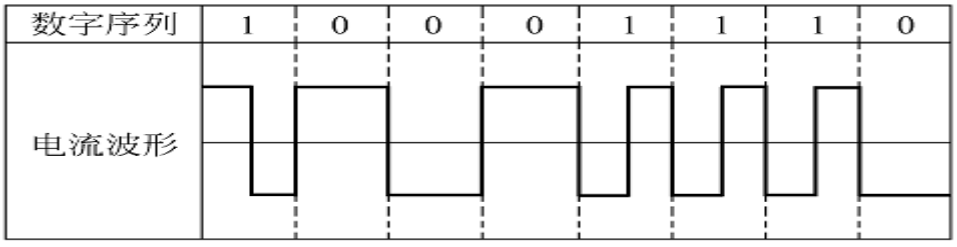


图 8-14 调频制波形图

5、改进的调频制

这种记录方式基本上与调频制相同，即在记录“1”时，电流在位中心位置翻转一次，在记录“0”时，电流方向不翻转。与调频制不同之处在于： 只有连续记录两个或两个以上“0”时，才在位周期的起始处翻转一次，而并非在每个位周期的起始位置都要翻转，因此进一步提高了记录密度。改进的调频制电流波形如图 **8-15** 所示。

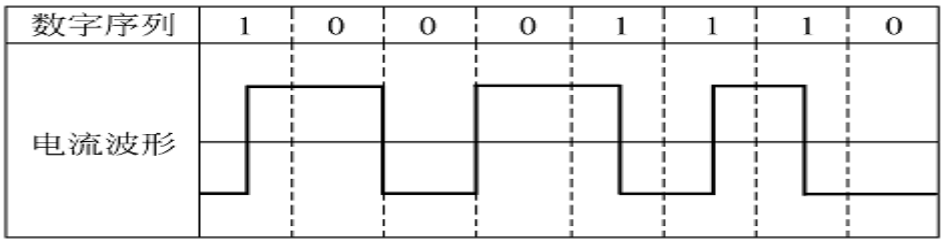


图 8-15　改进的调频制波形图

8.4.3 硬磁盘存储器

硬磁盘存储器 HDD（Hard Disk Driver）是磁表面存储设备，即用某些磁性材料薄薄地涂在金属铝或塑料表面作载磁体来存储信息。硬磁盘驱动器是记录介质为硬质圆形盘片的磁表面存储器。硬磁盘存储器具有速度快, 容量大，可以直接存取等优点，是计算机系统中的大容量辅助存储器，在辅存家族中担当着重要的角色。其外形及结构如图 **8-16** 所示。

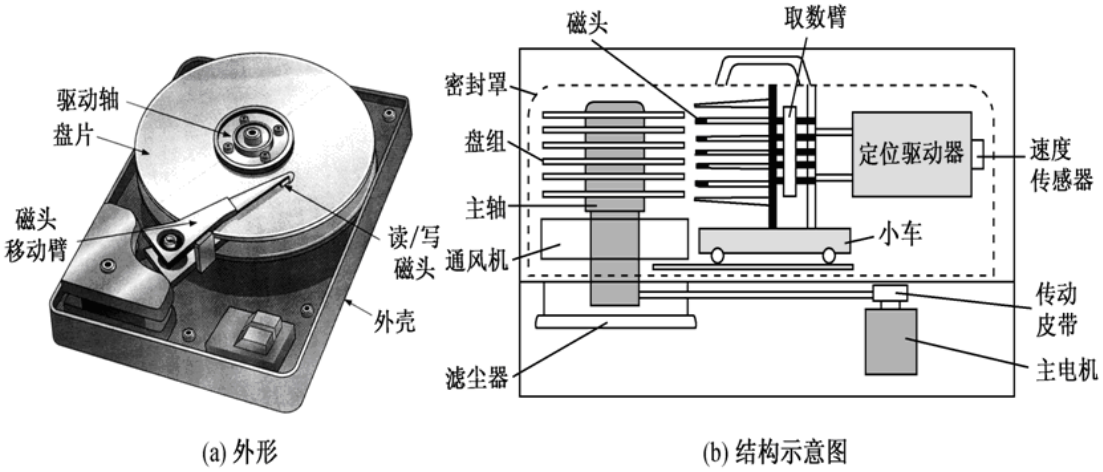


图 8-16 磁盘存储器的外形和结构示意图

1、硬磁盘存储器的分类

硬盘存储器根据磁头和盘片结构的不同可以分为固定磁头硬盘、活动磁头固定盘片硬盘、活动可换盘片硬盘以及温彻斯特硬磁盘等几种类型。

（1）固定磁头磁盘存储器

每一磁道均对应一个磁头，又称为每道一头磁盘。当要读写某一磁道时，通过译码器构成的磁头选择开关选取对应磁头工作。其特点是无需寻道操作，存取速度最快、但每道一头，磁头数量的增加使造价提高，且因机械装配限制了道密度的提高、仅用于少数要求快速存取而容量不大的场合。

（2）移动磁头固定盘片磁盘存储器

一片或一组磁盘片固定安装在主轴上，不可更换，每个盘面一个或两个磁头，所有盘面的磁头成梳状安装在磁头架上，通过磁头定位机构驱动磁头沿盘面径向位置移动，以定位所需磁道。其特点是大大减少了磁头数量，造价降低，可通过提高道密度来扩大存储容量。

（3）移动磁头可换盘片磁盘存储器

活动磁头可换盘片磁盘存储器不但磁头是移动式的，盘片也制成盘盒或者盘组的形式，可由用户装卸，并可在相同的磁盘机之间互换，读写时，需要有一套定位机构先将磁头沿径向移动到指定的磁道上方。

（4）温彻斯特磁盘存储器

微型计算机中使用的是温彻斯特硬磁盘，它把磁头、盘片、小车、导轨以及主轴等制作成一个整体，密封安装，简称为“温盘”。硬盘工作时，盘片高速旋转，通过浮在盘面上的磁头记录或读取信息。

2、硬盘驱动器结构与读写过程

硬盘存储器主要由磁头、盘片、硬盘驱动器和读/写控制电路组成。

（1）盘片

硬磁盘以铝合金材料作基片，它的表面涂敷一层磁介质作为记录媒体。盘片的上下两面都能记录信息，通常把磁盘片表面称为记录面。记录面上一系列同心圆称为磁道。每个盘片表面通常有几十到几百个磁道，每个磁道又分为若干个扇区。如图 **8-17** 所示。

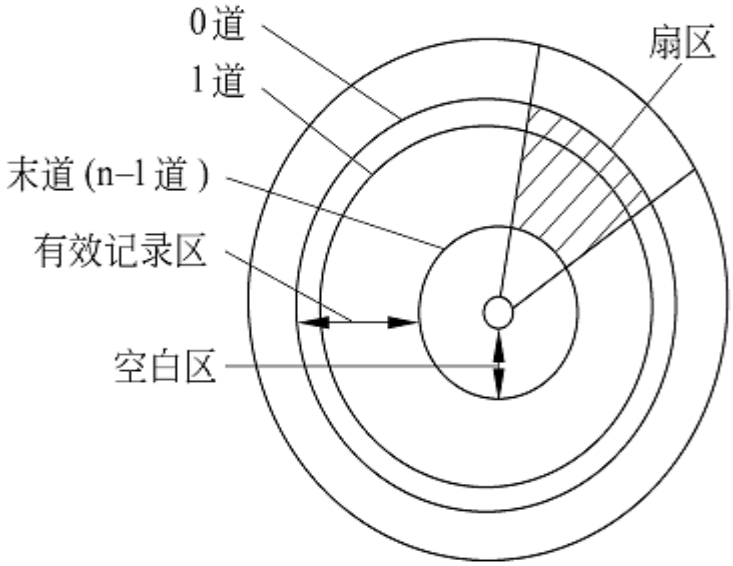


图 8-17 硬盘盘片示意图

磁道的编址是从外向内依次编号，最外一个同心圆叫 0 磁道，最里面的一个同心圆叫 n 磁道，n 磁道里面的圆面积并不用来记录信息。扇区的编号有多种方法，可以连续编号，也可间隔编号。磁盘记录面经这样编址后，就可用 n 磁道 m 扇区的磁盘地址找到实际磁盘上与之相对应的记录区。除了磁道号和扇区号之外，还有记录面的面号，以说明本次处理是在哪一个记录面上。例如对活动头磁盘组来说，磁盘地址是由记录面号(也称磁头号)、磁道号和扇区号三部分组成。

在磁道上，信息是按区存放的，每个区中存放一定数量的字或字节，各个区存放的字或字节数是相同的。为进行读/写操作，要求定出磁道的起始位置，这个起始位置称为索引。索引标志在传感器检索下可产生脉冲信号，再通过磁盘控制器处理，便可定出磁道起始位置。

磁盘存储器的每个扇区记录定长的数据，因此读/写操作是以扇区为单位一位一位串行进行的。每一个扇区记录一个记录块。数据在磁盘上的记录格式如下：每个扇区开始时由磁盘控制器产生一个扇标脉冲。扇标脉冲的出现即标志一个扇区的开始。两个扇标脉冲之间的一段磁道区域即为一个扇区(一记录块)。每个记录块由头部空白段、序标段、数据段、校验字段及尾部空白段组成。其中空白段用来留出一定的时间作为磁盘控制器的读写准备时间，序标被用来作为磁盘控制器的同步定时信号。序标之后即为本扇区所记录的数据。数据之后是校验字，它用来校验磁盘读出的数据是否正确。

盘片分为单面盘和双面盘两种，单面盘仅有一面能记录信息，双面盘两面都能记录信息。单片磁盘记录信息的容量毕竟有限，为了增大存储容量，硬盘一般由多个盘片组成盘片组。

（2）硬盘驱动器

磁头定位系统用来将磁头定位在准备进行读写的指定磁道上。它包括磁头架、磁头小车、

小车导轨、音圈电机及定位控制逻辑等几个部分。磁头安装在磁头臂上，每两个磁盘之间都配置一个读写臂，读写臂安有上下两个磁头，分别与上下两个盘面贴近。所有的读写臂连在一起，外形像梳子，故又称梳状读写臂。在音圈电机带动下，小滑架可前后平行地移动位置，以便磁头沿径向移动而找到目的磁道。当盘面作高速旋转时，依靠盘面的高速气流所产生的气动浮力使得磁头略微离开盘面，形成微小气隙，类似浮动式磁头。

（3）读/写控制电路

读写电路分写入与读出两个部分。写入电路包括编码电路、写入驱动电路。编码电路应根据记录方式的规定，把数据转换为相应的电流波形，驱动电路使写电流满足一定的幅度要求。读出电路主要包括前置放大器、主放大电路、鉴别电路和整形电路等。前置放大电路放大来自磁头线圈的感应信号，所以频带要宽，用差动放大器来排除外界的共模干扰。主放大电路把信号的幅度放大到鉴别电路能可靠鉴别的程度，鉴别电路应不受波形畸变及噪声影响，从信号中鉴别出有用的信号。不同的记录方式及不同的鉴别方式配合不同的鉴别电路。常用的鉴别方式有幅度鉴别、峰值鉴别和过零交叉点鉴别等。

硬盘的读写过程从查找磁道开始，驱动机构把磁头定位在目标磁道上方，目标扇区旋转到磁头下方时，读/写操作开始。写入时，数据经编码电路变换成相应的写电流，送到磁头写线圈，磁化盘面上的表面磁层，形成一个微小的磁化单元。读出时，磁化单元高速经过磁头，在磁头读线圈中感应出电压信号，经放大，整形和选通后输出。硬磁盘与软磁盘的存储原理与记录方式基本相同，但在结构和性能上存在较大差别。主要是：(1)硬盘转速高，存取速度快；软盘转速低，存取速度慢。(2)硬盘有固定头、固定盘、盘组等结构；软盘都是活动头，是可换盘片结构。(3)硬盘是浮动磁头读写；软盘磁头是接触式读写。(4)硬盘系统造价贵，大部分盘片不能互换；软盘系统造价低，盘片保管方便使用灵活，具有互换性。

8.4.4 光盘存储设备

光盘是近年发展起来的一种外存设备，是多媒体计算机不可缺少的设备。光盘采用聚焦激光束在盘式介质上非接触地记录高密度信息，以介质材料的光学性质(如反射率)的变化来表示所存储信息的“1”或“0”，如图 **8-18** 所示，通过将激光聚焦成极细光束对存储介质读写信息。光存储的本质不在于其记录过程，也不在于介质上信息的存储形式，而在于其读出过程。因此，无论采用哪一种方式存储信息，只要是用光学方式读出，即可认为是光存储器。

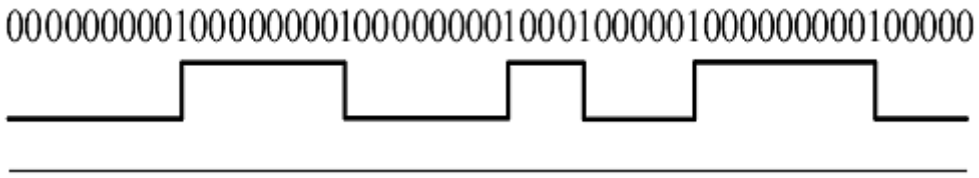


图 8-18 光盘数据记录格式

光盘的记录介质采用磁光材料，驱动器读写头是用半导体激光器和光路系统组成的光头。与其他存储器相比，光盘存储器具有记录密度高、存储容量大、信息保存寿命长、工作稳定可靠、环境要求低等特点。因此，它得到广泛的应用。

1、光盘存储器的组成

光盘存储器由光盘盘片、光盘驱动器 ODD(Optical Disk Driver)和光盘控制器 ODC(Optical Disk Controller)组成。驱动器同样有读/写头、寻道定位机构、主轴驱动机构等。与磁盘存储器不同的是，光盘存储器还有光学机构。

光盘盘片是记录信息的基体，由盘基、记录薄层和保护层组成。其中，盘基的材料为丙烯树酯、聚碳酸酯或硼硅酸玻璃，其热传导率低且非常耐热，用于记录信息的激光功率小。记录薄层分为形变型记录薄层、相变型记录薄层和磁光型记录薄层。形变型记录薄层用有机染料苏丹黑 B（硼），对激光有良好的吸收能力，熔点低且有较好的灵敏性和稳定性。相变

型记录薄层分不可逆相变：锑硒化合物(Sb-Se)、碲的低氧化物 TeOx，在激光照射下可实现从晶态至非晶态的转换；可逆相变：碲的低氧化物加锡(Sn)、锗(Ge)，只要激光加热过程的温度与时间适宜可实现晶态、非晶态的相互转换。磁光型记录薄层：磁化方向垂直于盘面，使用锰铋系晶体：MnCuBi、MnBi 等；柘榴石系单晶：$TbFeO_3$ 等；稀土类铁系非晶体：TbFe、GdFe 和 GdCo。保护层可以使记录层免受水蒸汽等的腐蚀，减少灰尘、划痕等对读出的影响，在记录薄层的表面直接覆盖一层厚度为 50—200μm 的金属薄膜或透明聚合物。

光盘驱动器的功能是驱动光盘转稳、转准，寻找光道并借助光头和激光器完成读写操作。光盘控制器由数据输入输出缓冲器、编码器、译码器和并/串转换电路等组成，通过 I/O 接口实现主机与 ODD 信息交换，控制 ODD 驱动盘片旋转、寻道和进行读写操作的功能。

2、光盘存储器的分类

按读写性质分，光盘有只读型光盘：记录的信息只能读出，不能被修改。一次型光盘：用户可在这种光盘上记录信息，但只能写一次，写后的信息不能再改变，只能读。重写型光盘：分磁光盘和相变盘两种，用户可对这类光盘进行随机写入、擦除或重写信息。

（1）只读光盘

只读光盘 CD-ROM（Compact Disc-Read Only Memory）是最常用的光盘，直径约 12cm，存储容量约 650MB。CD-ROM 的工作特点是： 采用激光调制方式记录信息，将信息以凹坑和凸区的形式记录在光道上。光盘是由母盘压模制成，一旦复制成型，永久不变，用户只能读出信息。

（2）一次写多次读光盘

一次写多次读 WORM（Write Once Read Many）光盘在使用前首先要进行格式化，格式化后形成信息区和逻辑目录区。WORM 光盘引入 DOS 文件分配表的概念，在光盘的根目录下面是用户定义的逻辑目录，逻辑目录对应文件管理区，文件管理区对应着相应的文件信息。

（3）可重写光盘

可重写 Rewriteable 光盘是最理想的光盘类型，也是最有应用前途的光盘类型。它像硬盘一样可读写。利用浮动磁光头在磁盘上进行磁场调制，可进行信息的高速重写。目前，由于价格较贵，普及受到限制。随着技术不断发展，成本将会不断降低，相信其前景会更好。

3、光盘存储器工作原理

（1）只读光盘读原理

只读光盘上的信息是沿着盘面螺旋形状的信息轨道以坑点的形式记录的，它既可以记录模拟信息，也可以记录数字信息。有坑点表示为“1”，无坑点表示为“0”，一系列的坑点(存储元)形成信息记录道。对数据存储用的光盘来讲，这种坑点分布作为数字“1”、“0”代码的写入或读出标志。由于只读型光盘是生产厂家制造，为了大量复制，需要制作母盘，为此将存储信息以表面坑点形式记录在母盘上。

光盘的记录信息以凹坑方式永久性存储。读出时，当激光束聚焦点照射在凹坑上时将发生衍射，反射率低；而聚焦点照射在凸面上时大部分光将返回。根据反射光的光强变化并进行光-电转换，即可读出记录信息。

（2）可重写光盘的擦写原理

要实现光盘信息的重写，必须将光盘介质的状态恢复到原来的性质，即擦去已有的信息，然后重新记录新的信息。磁光式光盘采用玻璃盘基上加 4 层膜结构组成，它是以稀土—过渡金属非晶体垂直磁化作为记录介质光学膜和保护膜的多层夹心结构。磁光写入信息的过程是用激光照射光盘垂直膜面磁化方向上的磁化物质，并对其垂直磁化。根据居里点热磁效应，用激光向需要存储信息“1”的单元区域加热，使其温度超过居里点，失去磁性。在盘的另一面的电磁线圈上施加一个外磁场，使被照单元反向磁化。这样，该单元区域磁化方向与其他未照射单元反向，从而产生一个信息状态“1”，而其他未照射单元相当于存储信息“0”。

信息擦去过程与写过程刚好相反，即恢复原来的磁化方向。读出原理是利用物理学中电磁感应效应，检测出磁盘单元磁化方向的不同，读出相应的信息“0”或“1”。

4、光盘存储器的发展

光存储器目前产品有 CD-ROM、DVD-ROM、DVD 机、DVD-RW(DVR)等，正产品化的技术有蓝光 DVD、HD-DVD、多波长存储、全息光存储等。其中蓝光的光波比红光短，存储密度可达到更高。2004，HP 推用蓝光技术 StorageWorks 超密度光盘库，存储容量 7.1*TB*，与磁光技术比，容量是三倍，成本下降 75%，开创了一个归档存储的新时代。Sony 在 ISOM04 大会上，发表了最新的蓝光光盘技术，允许单面记录层扩充到 8 层，记录容量有望提高到 200*GB*。目前，体全息存储、蓝色光盘是目前最有希望的两种新型存储技术。

8.4.4 闪存

1、闪存简介

闪存（Flash Memory）是一种长寿命的非易失性（在断电情况下仍能保持所存储的数据信息）的存储器，数据删除不是以单个的字节为单位而是以固定的区块为单位，区块大小一般为 256KB 到 20MB。闪存是电子可擦除只读存储器（EEPROM）的变种，EEPROM 与闪存不同的是，它能在字节水平上进行删除和重写而不是整个芯片擦写，这样闪存就比 EEPROM 的更新速度快。由于其断电时仍能保存数据，闪存通常被用来保存设置信息，如在电脑的 BIOS（基本输入输出程序）、PDA（个人数字助理）、数码相机中保存资料等。另一方面，闪存不像 RAM（随机存取存储器）一样以字节为单位改写数据，因此不能取代 RAM。

在 1984 年，东芝公司的发明人 Fujio Masuoka 首先提出了快速闪存存储器的概念。与传统电脑内存不同，Intel 是世界上第一个生产闪存并将其投放市场的公司。1988 年，公司推出了一款 256Kbit 闪存芯片。它如同鞋盒一样大小，并被内嵌于一个录音机里。後来，Intel 发明的这类闪存被统称为 NOR 闪存。它结合 EPROM(可擦除可编程只读存储器)和 EEPROM(电可擦除可编程只读存储器)两项技术，并拥有一个 SRAM 接口。第二种闪存称为 NAND 闪存。它由日立公司于 1989 年研制，并被认为是 NOR 闪存的理想替代者。NAND 闪存的写周期比 NOR 闪存短 90%，它的保存与删除处理的速度也相对较快。NAND 的存储单元只有 NOR 的一半，在更小的存储空间中 NAND 获得了更好的性能。

内存和 NOR 型闪存的基本存储单元是 bit，用户可以随机访问任何一个 bit 的信息。而 NAND 型闪存的基本存储单元是页（Page）（可以看到，NAND 型闪存的页就类似硬盘的扇区，硬盘的一个扇区也为 512 字节）。每一页的有效容量是 512 字节的倍数。所谓的有效容量是指用于数据存储的部分，实际上还要加上 16 字节的校验信息，因此我们可以在闪存厂商的技术资料当中看到“（512+16）Byte”的表示方式。目前 2*Gb* 以下容量的 NAND 型闪存绝大多数是（512+16）字节的页面容量，2Gb 以上容量的 NAND 型闪存则将页容量扩大到（2048+64）字节。

NAND 型闪存以块为单位进行擦除操作。闪存的写入操作必须在空白区域进行，如果目标区域已经有数据，必须先擦除后写入，因此擦除操作是闪存的基本操作。一般每个块包含 32 个 512 字节的页，容量 16*KB*；而大容量闪存采用 2*KB* 页时，则每个块包含 64 个页，容量 128*KB*。

每颗 NAND 型闪存的 I/O 接口一般是 8 条，每条数据线每次传输（512+16）*bit* 信息，8 条就是（512+16）×8*bit*，也就是前面说的 512 字节。但较大容量的 NAND 型闪存也越来越多地采用 16 条 I/O 线的设计，如三星编号 K9K1G16U0A 的芯片就是 64*M*×16*bit* 的 NAND 型闪存，容量 1*Gb*，基本数据单位是（256+8）×16*bit*，还是 512 字节。

寻址时，NAND 型闪存通过 8 条 I/O 接口数据线传输地址信息包，每包传送 8 位地址信息。由于闪存芯片容量比较大，一组 8 位地址只够寻址 256 个页，显然是不够的，因此通常一次地址传送需要分若干组，占用若干个时钟周期。NAND 的地址信息包括列地址(页面中

的起始操作地址)、块地址和相应的页面地址，传送时分别分组，至少需要三次，占用三个周期。随着容量的增大，地址信息会更多，需要占用更多的时钟周期传输，因此 NAND 型闪存的一个重要特点就是容量越大，寻址时间越长。而且，由于传送地址周期比其他存储介质长，因此 NAND 型闪存比其他存储介质更不适合大量的小容量读写请求。

2、闪存的应用及前景

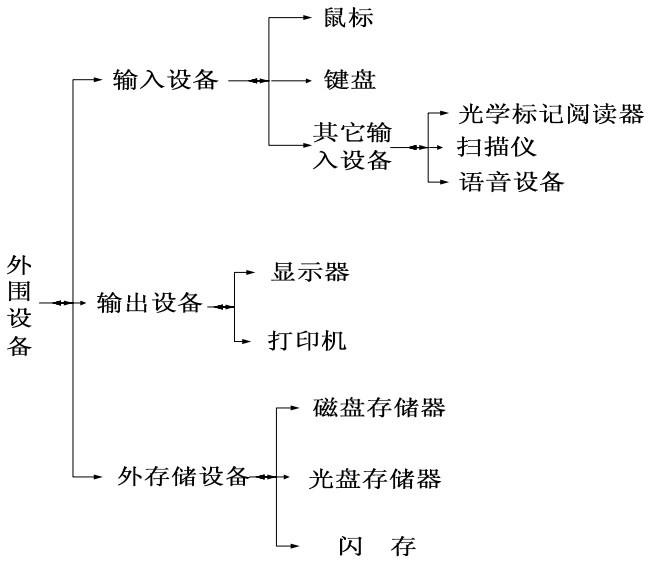
“优盘”是闪存走进日常生活的最明显写照，其实早在 U 盘之前，闪存已经出现在许多电子产品之中。传统的存储数据方式是采用 RAM 的易失存储，电池没电了数据就会丢失。采用闪存的产品，克服了这一毛病，使得数据存储更为可靠。除了闪存盘，闪存还被应用在计算机中的 BIOS、PDA、数码相机、录音笔、手机、数字电视、游戏机等电子产品中。

追溯到 1998 年，优盘进入市场。接口由 USB1.0 发展到 2.0，速度逐渐提高。U 盘的盛行还间接促进了 USB 接口的推广。为什么 U 盘这么受到人们欢迎呢？

闪存盘可用来在电脑之间交换数据。从容量上讲，闪存盘的容量从 16*MB* 到 2*GB* 可选，突破了软驱 1.44*MB* 的局限性。从读写速度上讲，闪存盘采用 USB 接口，读写速度比软盘高许多。从稳定性上讲，闪存盘没有机械读写装置，避免了移动硬盘容易碰伤、跌落等原因造成的损坏。部分款式闪存盘具有加密等功能，令用户使用更具个性化。闪存盘外形小巧，更易于携带。且采用支持热插拔的 USB 接口，使用非常方便。

目前，闪存正朝大容量、低功耗、低成本的方向发展。与传统硬盘相比，闪存的读写速度高、功耗较低，目前市场上已经出现了闪存硬盘。随着制造工艺的提高、成本的降低，闪存将更多地出现在日常生活之中。如果单从储存介质上来说 ，闪存比硬盘好 。但并不是音质上的好，是指数据传输的速度还有抗震度来说（闪存不存在抗震） 。要对比两者之间的优劣并不难， 首先理解什么是数码，知道什么是数码信号之后就该清楚数码信号通常是不受储存介质干扰的。(忽略音频流文件的误码，硬盘和闪存在这个方面可以忽略，光盘不同。)硬盘和闪存的数据准确性都很高 ，在同样的测试条件下（相同解码相同输出），两者音质肯定是一样的 。

关　联



习 题

8.1 判断题

(1)不带汉字库的点阵式打印机不能打印汉字。（ ）

(2)磁带存储器是记录模拟信号的设备。（ ）

(3)外围设备位于主机箱的外部。（ ）

(4)在各种数字磁记录方式中，改进式调频制的记录密度最高。（ ）

(5)随着半导体集成电路的发展，外围设备在计算机系统硬件的价格中所占的比重将越来越低。（ ）

(6)硬盘系统和软盘系统均可分为固定磁头和可移动磁头两种。（ ）

8.2 选择题

(1)主机从外部获取信息的设备称为（ ）。

A、外部存储器 B、外围设备 C、输入设备 D、输出设备

(2)在显示器的规格中，数据 640×480、1024×768 等表示（ ）。

A、显示器屏幕的大小

B、显示器显示字符的最大列数和行数

C、显示器的颜色指标

D、显示器的显示分辨率

(3)软盘存储器在读写数据时（ ）。

A、盘片转动，磁头不动 B、盘片不动，磁头移动

C、盘片磁头都不动 D、盘片转动，磁头移动

(4)下列外存中，属于顺序存取存储器的是（ ）。

A、软盘 B、磁带 C、硬盘 D、光盘

(5)有一种 CRT 显示器的分辨率为 1024×1024 像素，像素的颜色数为 256 色，则刷新存储器的容量是（ ）。

A、 1*MB* B、512*KB* C、2*MB* D、256*KB*

(6)磁盘驱动器向盘片磁层记录数据时，采用（ ）方式写入数据。

A、并行 B、并—串行 C、串行 D、串—并行

(7)一张 3.5 英寸的软盘片，其存储容量为（ ）*MB*，每个扇区存储的固定数据是（ ）。

A、1.44*MB*，512*B* B、1.2*MB* ，512*KB* C、1.76*MB*，512*B* D、360*KB*，512*KB*

(1)分辨率是显示器的主要性能参数之一，它的含义是（ ）。

A、显示屏幕的水平扫描率和垂直扫描率

B、显示屏幕上光栅的列数和行数

C、可显示的不同颜色的总数

D、同一幅画面允许显示不同颜色的最大数目

8.3 填空题

（1）字符显示器的控制逻辑电路的功能包括_____、_____ 、_____和_____等。

（2）按打印原理分类，打印机可分为_____打印机和_____打印机两类。

（3）磁盘的地址格式由_____、_____ 、_____和_____四部分组成。

（4）温彻斯特磁盘是一种采用先进技术研制的_____磁头、_____ 盘片的磁盘机，它将磁头、盘片、电机等驱动部件和读写电路等组装成一个_____的机电一体化整体，成为最有代表性的_____存储器。

（5）光盘存储器是近年来发展起来的一种_____设备，是_____必须配置的设备。按读写性质分，光盘可分为_____型、_____型和_____型三类。

（6）一个完整的磁盘存储器由三部分组成，其中，_____又叫磁盘机或磁盘子系统，是独立于主机的一个完整的设备_____是磁盘机与主机的接口部件；_____用于保存信息。

（7）闪速存储器能提供高性能、低功耗、高可靠性以及_____能力，为现有的_____体系结构带来了巨大的变化，因此作为_____用于笔记本电脑中。

（8）显示适配器作为 CRT 显示器和 CPU 的接口，由_____存储器、_____控制器、ROM BIOS 三部分组成。先进的_____控制器具有_____加速能力。

8.4 按功能分，计算机外围设备可分成哪几类？

8.5 鼠标有哪几种类型？简述光电式鼠标的工作原理。

8.6 简述显示器技术性能指标和显卡的种类。

第 9 章　总线

【内容摘要】

在计算机系统中，计算机内部各部件之间和计算机与外围设备之间的地址、数据、控制信息都是通过总线传送的，因此，总线是计算机系统的重要组成部分。本章介绍系统总线，包括系统总线结构、总线的控制和通信方式、信息在总线中的传送方式、计算机的总线标准以及常用总线举例。

【学习要点】

- 总线的分类、功能及特性
- 总线的结构类型
- 总线的信息传送方式及仲裁方式
- 计算机中的各种常用标准总线

9.1　总线技术概述

计算机可以看成是由三大部件构成，即中央处理器（CPU）、存储器（M）和输入输出系统（I/O），而计算机工作过程中信息的流动，就表现在这三大部件之间的通信。总线（BUS）是一组传输公共信息的信号线的集合，是在计算机系统各部件之间传输地址、数据和控制信息的公共通路。

总线能为多个部件服务，总线的基本工作方式通常是由发送信息的部件分时地将信息发往总线，再由总线将这些信息同时发往各个接收信息的部件。究竟由哪个部件接收信息，要由 CPU 给出的设备地址经译码产生的控制信号来决定。

1、工作原理

当总线空闲（其他器件都以高阻态形式连接在总线上）且一个器件要与目的器件通信时，发起通信的器件驱动总线，发出地址和数据。其他以高阻态形式连接在总线上的器件如果收到（或能够收到）与自己相符的地址信息后，即接收总线上的数据。发送器件完成通信，将总线让出（输出变为高阻态）。

2、总线的分类

（1）按照总线的位置分

1）片内总线　它是位于大规模、超大规模集成芯片内部各单元电路之间的总线，作为这些单元电路之间的信息通路。如 CPU 内部 ALU、寄存器组、控制器等部件之间的总线。

2）局部总线（也称内部总线）　通常指微机主板上各部件之间的信息通路。由于是一块电路板内部的总线，故又称在板局部总线。较典型的局部总线如：IBM-PC 总线，ISA 总线，EISA 总线和 PCI 总线等。

3）系统总线（也称外部总线）　是指微机底板上的总线，用来构成微机系统的各插件板、多处理器系统各 CPU 模块之间的信道。较典型的系统总线如 STD-BUS，MULTI-BUS，VME 等。

4）通信总线　它是微机系统与系统之间、微机系统与其它仪器仪表或设备之间的信息通路。这种总线往往不是计算机专有的，而是借用电子工业其它领域已有的总线标准并加以应用形成的。流行的通信总线如：EIA-RS-232C、RS-422A、RS-485，IEEE-488，VXI 等总线标准。

（2）按照系统的功能分

1）数据总线用于传送数据信息　数据总线是双向三态形式的总线，即他既可以把 CPU 的数据传送到存储器或 I / O 接口等其它部件，也可以将其它部件的数据传送到 CPU。数据总线的位数是微型计算机的一个重要指标，通常与微处理的字长相一致。例如 Intel 8086 微

处理器字长 16 位，其数据总线宽度也是 16 位。需要指出的是，数据的含义是广义的，它可以是真正的数据，也可以指令代码或状态信息，有时甚至是一个控制信息，因此，在实际工作中，数据总线上传送的并不一定仅仅是真正意义上的数据。

2)地址总线是专门用来传送地址　由于地址只能从 CPU 传向外部存储器或 I / O 端口，所以地址总线总是单向三态的，这与数据总线不同。地址总线的位数决定了 CPU 可直接寻址的内存空间大小，比如 8 位微机的地址总线为 16 位，则其最大可寻址空间为 216＝64*KB*，16 位微型机的地址总线为 20 位，其可寻址空间为 220＝1*MB*。一般来说，若地址总线为 n 位，则可寻址空间为 2n 字节。

3）控制总线用来传送控制信号和时序信号　控制信号中，有的是微处理器送往存储器和 I / O 接口电路的，如读 / 写信号，片选信号、中断响应信号等；也有是其它部件反馈给 CPU 的，比如：中断申请信号、复位信号、总线请求信号、限备就绪信号等。因此，控制总线的传送方向由具体控制信号而定，一般是双向的，控制总线的位数要根据系统的实际控制需要而定。实际上控制总线的具体情况主要取决于 CPU。

（3）按照传输数据的方式划分，可以分为串行总线和并行总线。串行总线中，二进制数据逐位通过一根数据线发送到目的器件；并行总线的数据线通常超过 2 根。常见的串行总线有 SPI、I2C、USB 及 RS232 等。

（4）按照时钟信号是否独立，可以分为同步总线和异步总线。同步总线的时钟信号独立于数据，而异步总线的时钟信号是从数据中提取出来的。SPI、I2C 是同步串行总线，RS232 采用异步串行总线。

3、总线功能

（1）数据传输功能

数据传输功能是总线的基本功能，用总线传输率来表示，即每秒传输的字节数，单位是 Mbps（兆字节每秒）。影响总线传输率的因素有：总线宽度、时钟频率等。

（2）多设备支持功能

多个设备使用一条总线，首先是总线占用权的问题，哪一个主设备申请占用总线，由总线仲裁器确定。

（3）中断

中断是计算机对紧急事务响应的机制，是计算机反应灵敏与否的关键。

（4）错误处理

错误处理包括奇偶校验错、系统错、电池失效等错误检测处理，以及提供相应的保护对策。

4、总线的特性

在总线层次中，CPU 总线、存储总线，因不同的计算机系统采用的芯片组不同，所以这些总线也不完全相同，互相没有互换性。而系统总线则不同，它是与 I/O 扩展插槽相连接的。I/O 插槽中可以插入各种扩冲板卡，作为各种外设的适配器与外设连接。因此要求系统总线必须有统一标准，它们必须在以下几方面做出规定：

（1）机械特性

物理特性指的是总线物理连接的方式。包括总线的根数、总线的插头、插座是什么形状、引脚是如何排列的等。例如：IBM-PC/XT 的总线共有 62 根线，分两列编号。

（2）电气特性

主要是定义每一根线上信号的传送方向、有效电平范围。一般规定送入 CPU 的信号称为输入信号，从 CPU 送出的信号称为输出信号。

（3）功能特性

确定引脚名称与功能，及其相互连接的协议。功能结构规范是总线的核心，通常以时序

和状态描述信息的交流、流向及管理规则。总线在功能结构方面的规范包括：
数据线、地址线、读/写及其它控制线、状态线、时钟线、电源线和地线等；中断机制；总线主控仲裁；应用逻辑：如联络（也称握手）线、复位、自启动、休眠维护等。

（4）时间特性

时间特性确定了每根线在什么时间有效，也就是每根线的时序。

5、总线的数据传送

（1）申请占用总线

需要使用总线的总线主设备向总线仲裁机构提出占用总线的请求，经总线仲裁机构判定，若满足响应条件，则发出响应信号，并把下一个总线传送周期的总线控制权授予申请者。

（2）寻址

获得总线控制权的总线主设备，通过地址总线发出本次要访问的存储器和 I/O 端口的地址，经地址译码选中被访问的模块并开始启动数据转换。

（3）传送数据

总线主设备也叫主模块，被访问的设备叫从模块。主模块和从模块之间的操作是由主模块控制在两个从模块之间通过数据总线进行数据传送。

（4）结束

主、从模块的信息均从总线上撤除，让出总线，以便其它主模块使用。

6、总线的性能指标

（1）总线宽度

数据总线一次能传送的数据位数（数据线根数）。总线宽度越宽则总线每次传输的数据量越大。

（2）总线带宽，又称为总线数据传输率

单位时间内总线上传送的数据量（MB／s）。带宽越大的总线性能越高。

（3）总线工作时钟频率以 MHz 为单位

工作频率越高则总线工作速度越快。

（4）其他指标

负载能力、总线仲裁时间。

9.2 总线系统结构

9.2.1 总线通道组成

早期总线实际上是处理器芯片引脚的延伸，是处理器与 I/O 设备适配器的通道。这种简单的总线一般由 50—100 条线组成，这些线按其功能可分为三类：地址线、数据线和控制线。

简单总线结构的不足之处在于：第一 CPU 是总线上的唯一主控者；第二总线信号是 CPU 引脚信号的延伸，故总线结构紧密与 CPU 相关，通用性较差。

当代流行的总线是一些标准总线，追求与结构、CPU、技术无关的开发标准，并满足包括多个 CPU 在内的主控者环境需求。在当代总线结构中，CPU 和它私有的 cache 一起作为一个模块与总线相连。系统中允许有多个这样的处理器模块。而总线控制器完成几个总线请求者之间的协调与仲裁。

整个总线分成如下四部分：第一部分： 数据传送总线，由地址线、数据线、控制线组成；第二部分：仲裁总线，包括总线请求线和总线授权线；第三部分：中断和同步总线，用于处理带优先级的中断操作，包括中断请求线和中断认可线。第四部分：公用线，包括时钟信号线、电源线、地线、系统复位线以及加电或断电的时序信号线等。

9.2.2 总线结构类型

1、单总线结构

单总线结构是用一组总线连接整个计算机系统的各大功能部件，计算机系统的所有设备都挂在这条总线上，各大部件之间所有的信息传送都通过这组总线。

早期的计算机，如美国 DEC 公司 PDP-11 机只使用一组总线，所有的部件和设备都接在这唯一的总线上，包括数据总线，地址总线，控制总线，其优点是结构简单，成本低廉，缺点是运行效率低。其结构如图 9-1 所示。

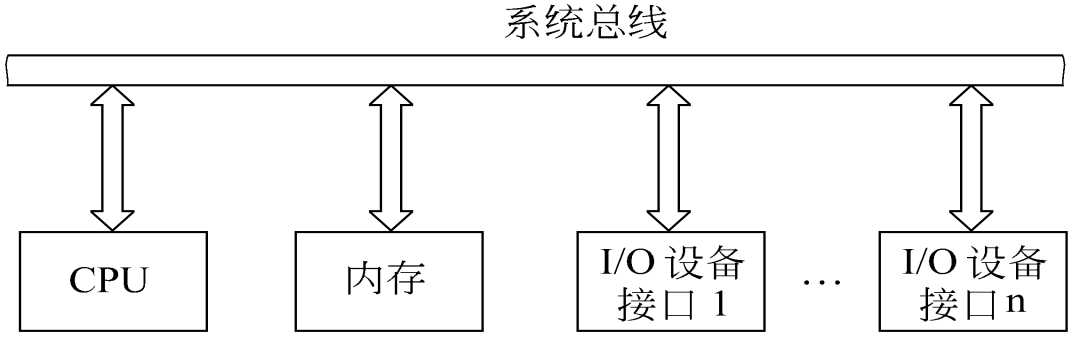


图 9-1 单总线结构

单总线结构能经同一总线实现 I/O 设备之间以及 I/O 设备与 CPU 之间的直接联系，这是一条共享总线，是微型机和小型机经常采用的一种总线结构。

单总线结构具有以下一些特点：特点一：所有连接到单总线上的计算机系统部件都共享同一地址空间。也就是说，主存储器的存储单元、各个子系统中所有能与总线实现通信的寄存器都可以统一编址。I/O 设备地址都采取存储器映射方式编址，因而指令系统中没有输入输出指令，任何访问存储器的指令都可以访问连接到总线上的任何设备。特点二：单总线采用异步通信方式，其传输速率只与设备固有速率有关，而与总线上其他子系统无关，与总线的物理长度无关。特点三：单总线不仅用在处理器级部件间互连，而且也可以用于各单元部件之间的连接。它们都具有标准总线的接口。特点四：与总线连接的所有部件是互相独立的，这种总线结构便于系统部件的扩充。

2、双总线结构

单总线结构的缺陷是系统效率和连接到总线上的各设备的利用率不高。这是因为单总线不允许多于两台的设备在同一时刻交换信息。为了克服这一缺陷，在有些小型机和大、中型机中，让 I/O 总线与内存总线分开，形成了双总线结构。

这种总线结构有两条总线，一条是内存总线，用于 CPU、内存和通道之间进行数据传送；另一条是 I/O 总线，用于多个外围设备与通道之间进行数据传送。其结构如图 9-2 所示。

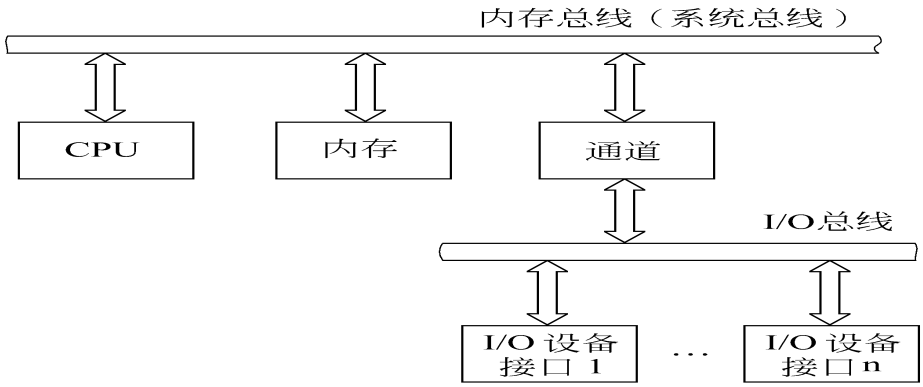


图 9-2 双总线结构

在双总线结构中，通道是计算机系统中的一个独立部件，使 CPU 的效率大为提高，并可以实现形式多样而更为复杂的数据传送。双总线的优点是以增加通道这一设备为代价的，

通道实际上是一台具有特殊功能的处理器，所以双总线通常在大、中型计算机中采用。

3、三总线结构

三总线结构计算机是基于上述思想，在 CPU 和主存之间设置了一条独立总线，以进一步提高计算机的效率。这种总线结构是在计算机系统的各部件之间采用三条各自独立的总线来构成信息通路。这三条总线是内存总线、输入/输出（I/O）总线和内存访问（DMA）总线，如图 9-3 所示。

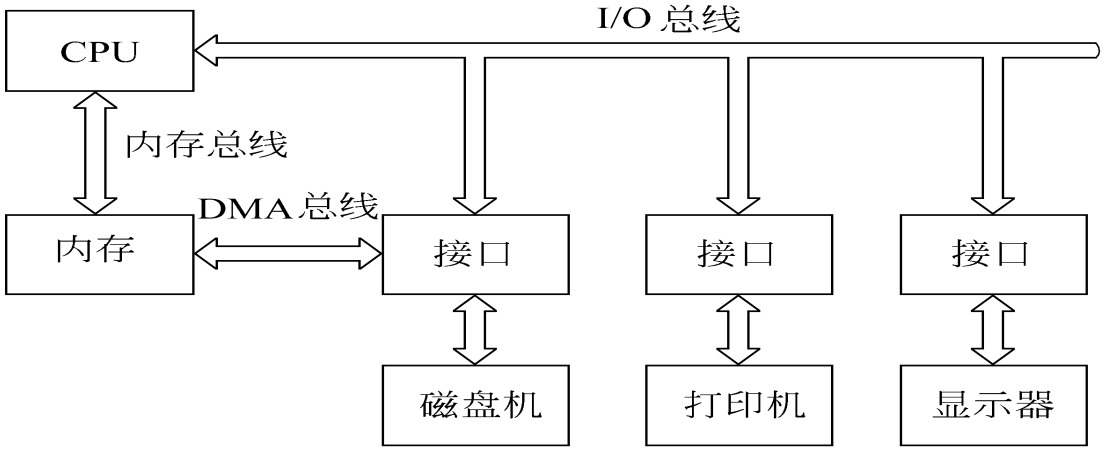


图 9-3 三总线结构

在三总线中，内存总线用于 CPU 和内存之间传送地址、数据和控制信息；I/O 总线供 CPU 和各类外设之间的通信；DMA 总线使内存和高速外设之间能够直接传送数据。一般来说，在三总线系统中，在任一时刻只使用一种总线，但若使用多入口存储器，内存总线可与 DMA 总线同时工作，此时，三总线系统可以比单总线运行得更快。但是三总线系统中，设备到设备不能直接进行信息传送，而必须经过 CPU 或内存间接传送，所以三总线系统总线的工作效率较低。

若再把不同速率的外部设备分类连接建立多条总线，则就是多总线结构了。

9.3 总线信息传送方式及定时

9.3.1 总线信息传送方式：

计算机系统中，传输信息采用三种方式：串行传送、并行传送和分时传送。但是出于速度和效率上的考虑，系统总线上传送的信息必须采用并行传送方式。

1、串行传送

当信息以串行方式传送时，只有一条传输线，且采用脉冲传送。在串行传送时，按顺序来传送表示一个数码的所有二进制位(bit)的脉冲信号，每次一位，通常以第一个脉冲信号表示数码的最低有效位，最后一个脉冲信号表示数码的最高有效位。在串行传送时，被传送的数据需要在发送部件进行并－－串变换，这称为拆卸；而在接收部件又需要进行串－－并变换，这称为装配。串行传送的主要优点是只需要一条传输线，这一点对长距离传输显得特别重要，不管传送的数据量有多少，只需要一条传输线，成本比较低廉。

2、并行传送

用并行方式传送二进制信息时，对每个数据位都需要单独一条传输线。信息有多少二进制位组成，就需要多少条传输线，从而使得二进制数“0”或“1”在不同的线上同时进行传送。并行传送一般采用电位传送。由于所有的位同时被传送，所以并行数据传送比串行数据传送快得多。

3、分时传送

分时传送有两种概念：一是采用总线复用方式，某个传输线上既传送地址信息，又传送

数据信息。为此必须划分时间片，以便在不同的时间间隔中完成传送地址和传送数据的任务。分时传送的另一种概念是共享总线的部件分时使用总线。

9.3.2 总线定时

获得总线使用权的设备或部件可以在总线上进行数据通信。总线通信方式是实现总线控制和数据传送的手段。信息在总线上的传送方式通常有同步和异步两种方式。

1、同步通信

总线上的部件通过总线进行信息传送时，用一个公共的时钟信号进行同步，部件之间按照约定时钟时间进行信息交换，这种方式称为同步通信。这个公共的时钟信号可以由总线控制部件发送到每一个部件或设备，也可以每个部件有自己的时钟发生器，但是，它们都必须由 CPU 发出的时钟信号进行同步。由于采用统一的时钟，所有总线信号和命令必须以总线时钟有效时开始，所有总线操作以总线时钟为基本时间单位，即总线所用时钟数必须是整数。

同步通信的优点是具有较高的传输速率，数据传输速度快，总线控制逻辑也比较简单。同步通信适用于总线长度较短，各部件存取时间比较接近的情况。因此带来的缺点是假如总线长度长了，势必降低数据的传输速率。由于总线长度较短，不能及时进行数据通信的有效性检验。

2、异步通信

任何一个事件都只能是前面一个或一些事件的结果，所有设备以信号“握手”的方式进行联系，从而完成总线操作等工作，这种数据传送的方式称为异步通信。

在异步通信最根本的特征是总线系统中没有统一的时间标志，允许总线上的各部件有各自的时钟，部件之间的通信不依靠公共的时间标准，而是利用应答方式的“握手”信号来实现。发送部件将数据放到总线上后，经过一定的时间延迟，便在控制线上发出“数据准备好”信号，而接收部件则应发“数据接收”信号来响应，把此信号送到源部件上，并接收数据。发送部件接收到响应信号后，去除原数据，本次传送结束。

异步通信方式的优点是，便于实现不同传输速率部件之间的数据传送，而且对总线长度也没有严格的要求，还能实现数据的有效性检验。缺点是速度一般不如同步通信方式高，而且总线控制逻辑也相对复杂一些。

9.4 总线的仲裁

由于存在多个设备或部件同时申请对总线的使用权，为保证在同一时间只能有一个设备获得总线使用权，必须具有总线仲裁部件，总线仲裁部件按照申请者的优先权选择设备。只有获得了总线使用权的设备或部件，才能开始数据传送。按照总线仲裁电路的位置不同，仲裁方式分为集中式仲裁和分布式仲裁两类。总线仲裁电路基本集中在一处的（如集中于 CPU 中），称为集中式仲裁；而总线仲裁电路分散在总线各部件的，称为分布式仲裁。

9.4.1 集中式仲裁

集中式仲裁是单总线、双总线和三总线结构计算机主要采用的方式，集中式总线的仲裁方式主要有以下三种：链式查询方式、计数器定时查询方式和独立请求总线控制方式。

1、链式查询方式

在链式查询方式电路中，除一般数据总线和地址总线外，在控制总线中有三根线用于总线的控制，它们分别是总线忙（BS）线、总线请求（BR）线和总线同意（BG）线。

BS：总线忙/闲状态线，当其有效时，表示总线正被某外设使用。

BR：总线请求线，当其有效时，表示至少有一个外部设备要求使用总线。

BG：总线同意，当其有效时，表示总线控制部件响应总线请求（BR）。

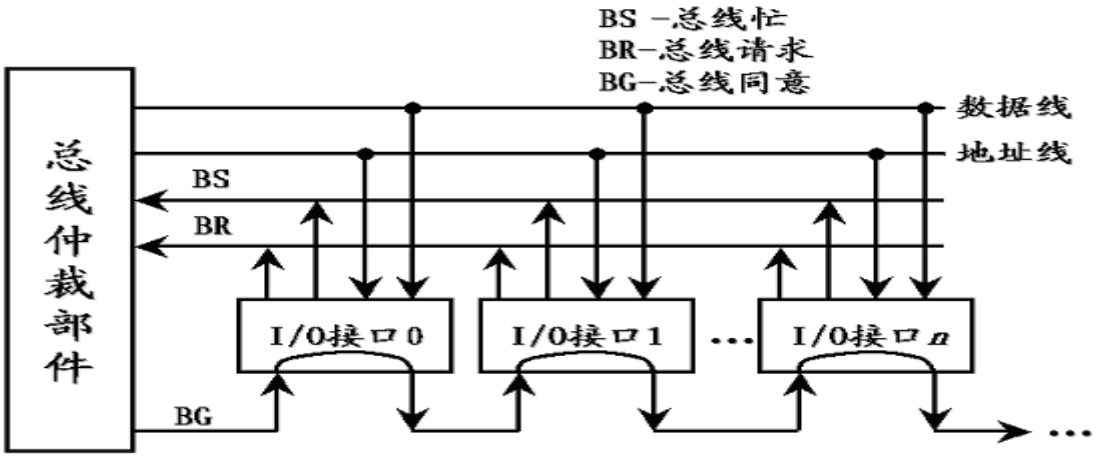


图 9-4 链式查询方式

如图 9-4 所示，总线同意信号（BG）是串行地从一个 I/O 接口送到下一个 I/O 接口，如果 BG 到达的接口无总线请求，则继续往下传；如果 BG 到达的接口有总线请求，BG 信号便不再往下传，这意味着该 I/O 接口获得了总线使用权。BG 信号就像一条链一样串联所有的设备接口，故这种总线控制方式称为链式查询方式。在查询链中，离总线控制器最近的设备具有最高优先权；离总线控制器越远的设备，优先权越低。这种方式的优点是只用很少几根线就能按一定优先次序实现总线仲裁，很容易扩充设备。缺点是对询问链的电路故障很敏感，如果第 i 个设备的接口中有关链的电路有故障，那么第 i 个以后的设备都不能进行工作。查询链的优先级是固定的，如果优先级高的设备出现频繁的请求时，优先级较低的设备可能长期不能使用总线。

2、计数器定时查询方式

计数器定时查询方式的工作原理是：总线上任一设备要求使用总线时，通过“总线请求”（BR）线发出总线请求信号，总线控制器接到请求信号后，在“总线忙”（BS）为复位的情况下，让计数器开始计数，计数值通过一组地址线发至各设备。每个设备接口都有一个设备地址判别电路，当地址线上的计数值与请求总线的设备地址一致时，该设备把“总线忙”（BS）置位，获得了总线控制权。此时，终止计数查询。每次计数可以从“0”开始，也可以从终止点开始。如果从“0”开始，各设备的优先次序与链式查询法相同，即优先次序是固定的。如果从终止点开始，则是一种循环方法，每个使用总线的优先级是相同的。如图 **9-5** 所示。

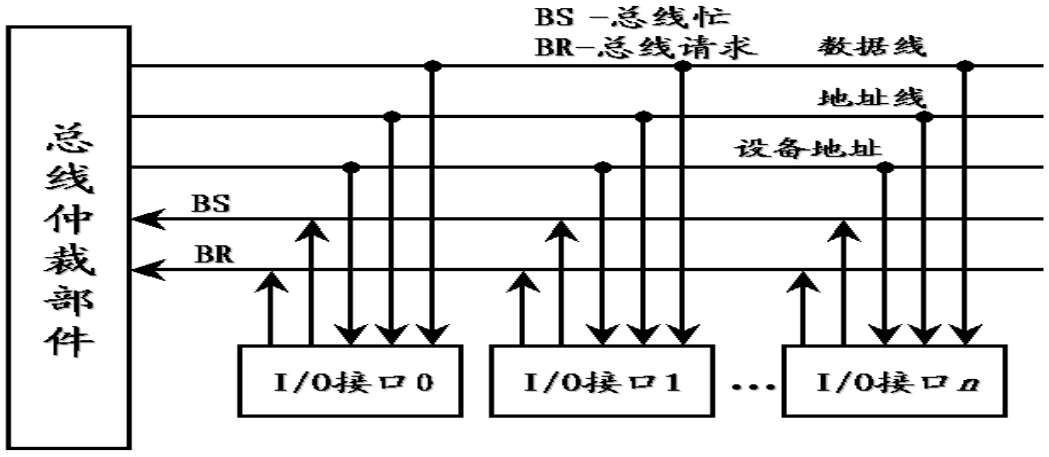


图 9-5 计数器定时查询方式

这种方式的优点是：由于查询可以被程序控制（计数器的初值可由程序设定），所以优先次序可以方便地改变。另外，这种查询方式不会出现链式查询那样当某个设备的接口中有关键的电路出现故障时，会影响其他设备使用总线。缺点是：要有一组设备地址线，从而增加了控制线的数量，而且控制也较为复杂。

3、独立请求方式

在独立请求方式中，每一个共享总线的设备均有一对“总线请求”（BR）和“总线同意”（BG）线。当设备要求使用总线时，便发出“总线请求”信号，总线控制部件中一般有一个排队电路，根据一定的优先次序决定首先响应哪个设备的请求，当请求的设备排上队，便收到“总线同意”（BG）信号，从而可以使用总线。如图 **9-6** 所示。

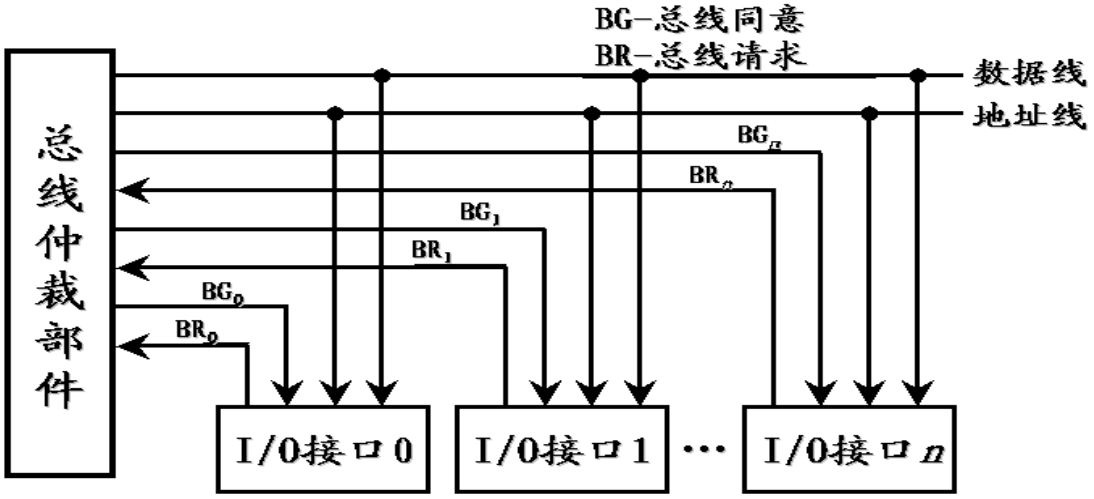


图 9-6 独立请求方式

独立请求方式的优点是：响应时间快，对优先次序的控制也相当地灵活，它可以预先固定优先次序，也可以通过程序来改变优先次序。并且在必要时屏蔽某些设备的请求。缺点是：控制线的数量多。比如要控制 n 个设备，必须有 n 根“总线请求”线和 n 根“总线同意”线，另外，独立请求方式的控制器也要复杂得多。

9.4.2 分布式仲裁

分布式仲裁不需要中央仲裁器，每个潜在的主方功能模块都有自己的仲裁号和仲裁器。当它们有总线请求时，把它们唯一的仲裁号发送到共享的仲裁总线上，每个仲裁器将仲裁总线上得到的号与自己的号进行比较。如果仲裁总线上的号大，则它的总线请求不予响应，并撤消它的仲裁号。最后，获胜者的仲裁号保留在仲裁总线上，分布式仲裁是以优先级仲裁策略为基础。

9.5 计算机中的总线

总线是构成计算机系统的基础，它将影响系统的灵活性、成本、性能和可靠性。由于超大规模集成电路工艺的发展，系统的复杂性也在不断地增加，总线则往往成为提高性能、可靠性和模块化的制约因素。因此在大多数微机系统中推行标准总线技术。标准总线使得总线接口部件标准化，简化了系统设计，缩短了开发时间，降低了开发成本，增加了系统配置的灵活性。下面就介绍几种常用标准总线。

9.5.1 内部总线

1、工业标准总线 ISA

最早的 PC 总线是 IBM 公司 1981 年在 PC/XT 电脑采用的系统总线，它基于 8bit 的 8088

处理器，被称为 PC 总线或者 PC/XT 总线。在 1984 年的时候，IBM 推出基于 16-bit Intel 80286 处理器的 PC/AT 电脑，系统总线也相应地扩展为 16bit，并被称呼为 PC/AT 总线。而为了开发与 IBM PC 兼容的外围设备，行业内便逐渐确立了以 IBM PC 总线规范为基础的 ISA（工业标准架构：Industry Standard Architecture ）总线。

ISA 总线最大传输速率仅为 8MB/s，但允许多个 CPU 共享系统资源。由于兼容性好，它在上个世纪 80 年代是最广泛采用的系统总线，不过它的弱点也是显而易见的，比如传输速率过低、CPU 占用率高、占用硬件中断资源等。使用 286 和 386SX 以下 CPU 的电脑似乎和 8/16bit ISA 总线还能够相处融洽，但当出现了 32-bit 外部总线的 386DX 处理器之后，总线的宽度就已经成为了严重的瓶颈，并影响到处理器性能的发挥。因此在 1988 年，康柏、惠普等 9 个厂商协同把 ISA 扩展到 32-bit，这就是著名的 EISA（Extended ISA，扩展 ISA）总线。

2、EISA 总线

EISA 总线的插槽的外形与 ISA 总线完全相同，但插槽为两层结构，第一层的引线定义与 ISA 的一样，共 98 根引线；第二层的引线是 EISA 的扩充部分，共 90 根引线。EISA 是 32 位总线，支持多处理器结构，具有较强的 I/0 扩展能力和负载能力，支持多总线主控，传输率为 33Mbps，适用于网络服务器、高速图像处理、多媒体等领域。由于 EISA 总线是兼容商共同推出的，技术标准公开，因而受到世界上众多厂家的欢迎。

EISA 是一种支持多处理器的高性能 32 位标准总线，但由于兼顾了 ISA 的电气特性，因而妨碍了 EISA 总线速度的进一步提高，但由于是 32-bit 总线的缘故，带宽提高了一倍，达到了 32MB/s。可惜的是，EISA 仍旧由于速度有限，并且成本过高，在 20 世纪 90 年代初的时候，被 PCI 总线给取代了。

3、PCI 总线

PCI 总线（Peripheral Component Interconnect，即外部设备互联总线），是 Intel 公司在 1992 年率先提出的。其引脚排列示意图如图 **9-7** 所示。

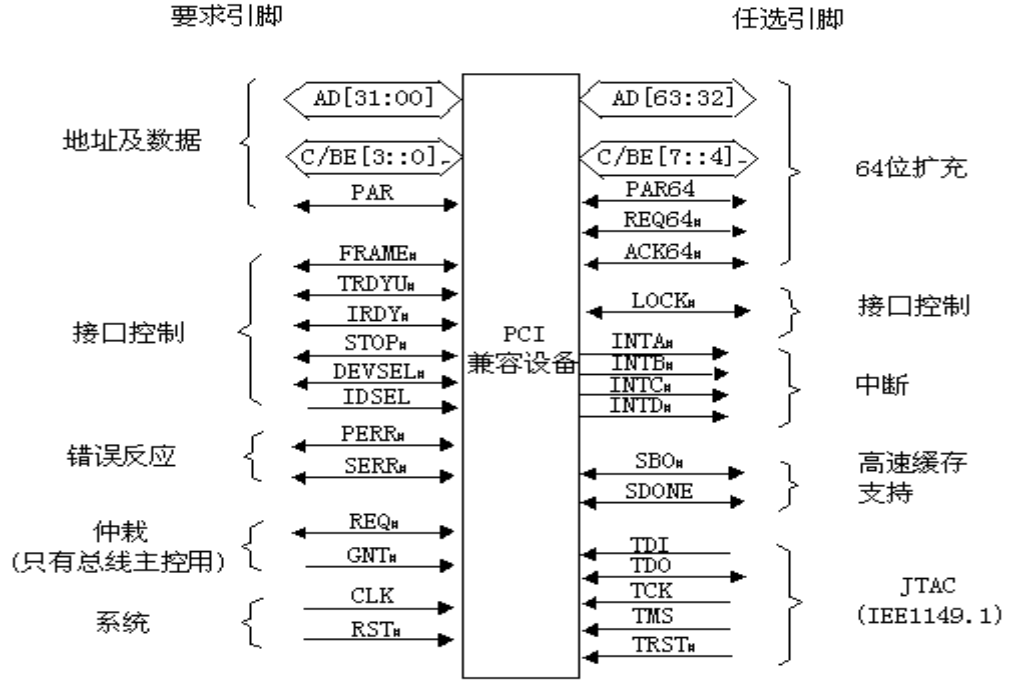


图 9-7　PCI 总线的引脚排列示意图

随着计算机技术的迅速发展和信息化水平的不断提高，对计算机外设的使用性能要求越来越高，特别是对图形显示的高要求，硬盘容量的增大和数据传输率的提高，要求有更高性能的总线。原有的 ISA 总线和 EISA 总线显然已不适应。当时 CPU 的速度甚至还高过总线的速度，造成硬盘、显示卡还有其它的外围设备只能通过慢速并且狭窄的瓶颈来发送和接受数据，使得整机的性能受到严重的影响。为了解决这个问题，1992 年 Intel 在发布 486 处理器的时候，也同时提出了 32-bit 的 PCI 总线。从数据宽度上看，PCI 总线有 32bit、64bit 之分；从总线速度上分，有 33MHz、66MHz 两种。最早提出的 PCI 总线工作在 33MHz 频率之下，传输带宽达到了 133MB/s（33MHz×32bit/8），比 ISA 总线有了极大的改善，基本上满足了当时处理器的发展需要。随着对更高性能的要求，1993 年提出了 64-bit 的 PCI 总线，后来又提出把 PCI 总线的频率提升到 66MHz。目前广泛采用的是 32-bit、33MHz 的 PCI 总线。PCI 总线是独立于 CPU 的系统总线，采用了独特的中间缓冲器设计，可将显示卡、声卡、网卡、硬盘控制器等高速的外围设备直接挂在 CPU 总线上，打破了瓶颈，使得 CPU 的性能得到充分的发挥。在 PCI 上包含有寄存器，上面带有配置所需的器件信息。

PCI 总线的特点：

（1）PCI 总线的地址总线与数据总线是分时复用

这样做的好处是，一方面可以节省接插件的管脚数，另一方面便于实现突发数据传输。在做数据传输时，由一个 PCI 设备做发起者(Master)，而另一个 PCI 设备做目标(Slave)。总线上的所有时序的产生与控制，都由 Master 来发起。PCI 总线在同一时刻只能供一对设备完成传输，这就要求有一个仲裁机构，来决定在谁有权力拿到总线的主控权。

（2）支持即插即用

当板卡插入系统时，系统会自动对板卡所需资源进行分配，如基地址、中断号等，并自动寻找相应的驱动程序。而不象旧的 ISA 板卡，需要进行复杂的手动配置。即插即用实际的实现远比说起来要复杂。在 PCI 板卡中，有一组寄存器，叫“配置空间”，用来存放基地址与内存地址，以及中断等信息。

以内存地址为例。当上电时，板卡从 ROM 里读取固定的值放到寄存器中，对应内存的地方放置的是需要分配的内存字节数等信息。操作系统要跟据这个信息分配内存，并在分配成功后把相应的寄存器中填入内存的起始地址。这样就不必手工设置开关来分配内存或基地址了。对于中断的分配也与此类似。

（3）实现中断共享

ISA 卡的一个重要局限在于中断是独占的，而我们知道计算机的中断号只有 16 个，系统又用掉了一些，这样当有多块 ISA 卡要用中断时就会有问题了。

PCI 总线的中断共享由硬件与软件两部分组成。硬件上，采用电平触发的办法：中断信号在系统一侧用电阻接高，而要产生中断的板卡上利用三极管的集电极将信号拉低。这样不管有几块板产生中断，中断信号都是低；而只有当所有板卡的中断都得到处理后，中断信号才会回复高电平。软件上，采用中断链的方法：假设系统启动时，发现板卡 A 用了中断 7，就会将中断 7 对应的内存区指向 A 卡对应的中断服务程序入口 ISR_A；然后系统发现板卡 B 也用中断 7，这时就会将中断 7 对应的内存区指向 ISR_B，同时将 ISR_B 的结束指向 ISR_A。以此类推，就会形成一个中断链。而当有中断发生时，系统跳转到中断 7 对应的内存，也就是 ISR_B。ISR_B 就要检查是不是 B 卡的中断，如果是，要处理，并将板卡上的拉低电路放开；如果不是，则呼叫 ISR_A。这样就完成了中断的共享。

4、AGP 总线

三维图形应用的发展，对显卡的计算速度提出了越来越高的要求，PCI 总线对于胃口越来越大的 3D 显卡却力不从心，并成为了制约显示子系统和整机性能的瓶颈。因此，PCI 总线的补充—AGP 总线就应运而生了。AGP 总线不同于通用的 PCI 局部总线，它是供图形加

速卡专用的。其地址和数据分离(PCI 为 49 根信号线，而 AGP 总线是 65 根)，可实现“流水线”处理；地址线和数据线分离，没有切换的“开销”，提高了系统实际数据传输速率和随机访问主内存时的性能。AGP 总线的首要目的是将纹理数据置于主内存，以减少图形存储器的容量，从而可以生产廉价、高性能的图形卡，开通主内存到图形卡的高速传输通道。

AGP 总线可以将系统主内存映射为 AGP 内存，用作图形卡上的专业显存的扩展，并通过直接内存执行方式提高系统的 3D 图形处理性能，减少图形设备对系统的占用。

AGP 总线最重要的特征是提高了数据传输的带宽。由于 AGP 总线宽为 32 位，基于 66*MHz* 时钟，并在时钟脉冲的“上升沿”和“下降沿”都能传输数据，因而可达到 533*Mbps* 的理论传输率，比普通 PCI 接口图形卡提高了 4 倍。而且由于 AGP 总线还借用了处理器的“流水线”技术，并有 8 条额外的“边际”(Sideband)数据请求线，支持对数据的“流水线”装入和预先读取，同时还可将需要的“边际数据”一起传输，从而大大增加了有效带宽。如果采用新潮的 AGP 4×模式，AGP 总线的时钟频率将增加到 133*MHz*，其数据传输率将突破 1*Gbps* 大关。

9.5.2 外部通信总线

1、RS-232C 总线

RS-232C 是一种串行通信总线标准，也是数据终端设备（DTE）和数据通信设备（DCE）之间的接口标准，是 1969 年由美国电子工业协会（EIA）从 CCITT 远程通信标准中导出的一个标准。RS-232C 标准包括机械指标和电气指标，其中机械指标规定：RS-232C 标准接口通向外部的连接器（插针和插座）是一个“D”型保护壳 25 针插头，如图 **9-8** 所示。

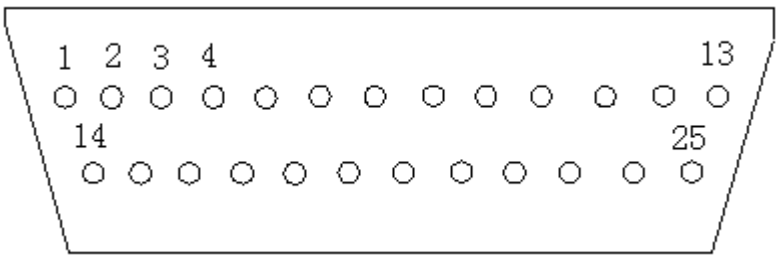


图 9-8 RS-232C 连接器

（1）R232C 的主要特点

1）信号线少

RS-232C 总线共有 25 根线，它包括有主副两个通道，用它可进行双工通信。实际应用中，多数只用主信号通道（即第一通道），并只使用其中几个信号（通常 3-9 根线）。

2）传输距离远

由于 RS-232C 采用串行传输方式，并将 TTL 电平转换成了 RS-232C 电平，在基带传输时，距离可达 30*m*。若是采用光电隔离 20*A* 电流环传送，其传输距离可达 1000*m*。

3）可供选择的传输速率多

RS-232C 规定的标准传送速率有：50，75，110，150，300，600，1200，2400，4800，9600，19200 波特。可以灵活地使用于不同速率的设备。

4）抗干扰能力强

RS-232C 采用负逻辑，空载时以+3—+25*V* 之间任意电压表示逻辑“0”，以-3—-25*V* 之间任意电压表示逻辑“1”，且它是无间隔不归零电平传送，从而大大提高了抗干扰能力。

（2）RS-232C 总线的功能规范

1）引脚分配

RS-232C 总线共有 25 根信号线，其中，2 根地线、4 根数据线、11 根控制线、3 根定时

线、5 根备用线。引脚分配及定义如表 **9-1** 所示。

表 9-1　　引脚分配及定义

引脚	说 明	缩写	引脚	说 明	缩写
*1	保护地	PG	14	第二数据发送，输出	TXD
*2	数据发送，输出	TXD	*15	发送码元定时，输出	
*3	数据接收，输入	RXD	16	第二数据接收，输入	RXD
*4	请求发送，输出	RTS	*17	接收码元定时，输出	
*5	允许发送，输入	CTS	*18	未定义	
*6	数据设备准备好，输入	DSR	19	第二请求发送，输出	RTS
*7	信号地	SG	*20	数据终端准备好，输出	DTR
*8	接收信号检出，输入	DCD	*21	信号质量检测，输出	
9	电流环发送返回，输出		*22	振铃指示。输入	RI
10	空	备用	*23	数据信号速率选择	
11	电流环发送数据，输出		*24	发送信号码元定时，输出	
12	第二接收信号检出，输入	DCD	25	未定义	
13	第二允许发送，输入	CTS			

注：带“*”表示主信道信号

2）引脚信号说明

在 RS-232C 总线中，虽然绝大多数信号线均已定义使用，但在一般的微型计算机串行通信中，经常使用的只有以下 9 个信号线，具体见表 **9-2**，它们都是主信道组的信号线。

表 9-2　　常用主信道组信号线

引脚号	符号	方向	功能
2	TXD	输出	发送数据
3	RXD	输入	接收数据
4	RTS	输出	请求发送
5	CTS	输入	允许发送
6	DSR	输入	数据设备准备好
7	GND		信号地
8	DCD	输入	数据载波检测
20	DTR	输出	数据终端准备好
22	RI	输入	振铃指示

这 9 根引脚分为两类：一类是基本的数据传送引脚，另一类是用于调制解调器（MODEM）的控制和反映它的状态的引脚。

基本的数据传送引脚：TXD，RXD，GND（2，3，7 号引脚）是基本数据传送引脚。

MODEM 的控制和状态引脚：从计算机通过 RS-232C 接口送给 MODEM 的控制引脚包括 DTR 和 RTS；从 MODEM 通过 RS-323C 接口送给计算机的状态信息引脚包括 DSR、CTS、

DCD 和 RI。

·DTR 数据终端准备完毕引脚，用于通知 MODEM 计算机准备好，可以通信了。

·RTS 为请求发送引脚，用于通知 MODEM 计算机请求发送数据。

·DSR 为数据通信设备准备就绪引脚，用于通知计算机，MODEM 准备好了。

·CTS 为允许发送引脚，用于通知计算机 MODEM 可以接收数据了。

·DCD 为数据载体检测引脚，用于通知计算机 MODEM 与电话线另一端的 MODEM 已经建立联系。

·RI 振铃信号指示引脚，用于通知计算机，有来自电话网的信号。

（3）RS-232C 电气规范

RS-232C 电气规范如表 **9-3** 所示。

表 9-3　RS-232C 总线的电气规范

带3~7kΩ负载时驱动器的输出电平	逻辑0：+5~+15V
	逻辑1：-5~-15V
不带负载时驱动器的输出电平	-25~+25V
驱动器断开时的输出阻抗	> 300Ω
输出短路电流	< 0.5A
驱动器转换速率	< 30V/μs
接收器输入阻抗	在3~7kΩ之间
接收器输入电压的允许范围	-25~+25V
输入开路时接收器的输出	逻辑1
输入经300Ω接地时接收器的输出	逻辑1
+3V输入时接收器的输出	逻辑0
-3V输入时接收器的输出	逻辑1
最大负载电容	2500Pf

2、IEEE1394

IEEE1394 是由 Apple 公司于 20 世纪 80 年代中期开始开发的一种串行总线，中文译名为火线接口（firewire），这个名称来源于它令人瞠目结舌的传输速度。该规范于 1995 年得到了标准化，名称也由 FireWire 改为了 IEEE 1394。此外，索尼公司对 IEEE 1394 进行了许多改进，并且推出了它自己的版本 dubbed i.Link，大部分的索尼计算机都使用了该版本的 IEEE 1394 规范。同 USB 一样，IEEE1394 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，支持同步数据传输。1394 标准的最新版本是 1394b。IEEE1394 接口如图 **9-9** 所示。

图 9-9　IEEE1394 接口

IEEE1394 分为两种传输方式：Backplane 模式和 Cable 模式。Backplane 模式最小的速率也比 USB1.1 最高速率高，分别为 12.5 *Mbps/s* 、25 *Mbps/s* 、50 *Mbps/s*，可以用于多数的高带宽应用。Cable 模式是速度非常快的模式，分为 100 *Mbps/s* 、200 *Mbps/s* .400 *Mbps/s* 和 800*Mbps* 几种，在 200*Mbps/s* 下即可传输不经压缩的高质量数据电影。

1394b 是 1394 技术的升级版本，是仅有的专门针对多媒体--视频、音频、控制及计算机

而设计的家庭网络标准。它通过低成本、安全的 CAT5（五类）实现了高性能家庭网络。1394a 自 1995 年就开始提供产品，1394b 是 1394a 技术的向下兼容性扩展。1394b 能提供 800 *Mbps/s* 或更高的传输速度。近年来随着成本的下降，1394 卡正迅速普及。虽然市面上还没有 1394b 接口的光储产品出现，但相信在不久之后也必然会出现在用户眼前。

1394 接口具有把一个输入信息源传来的数据向多个输出机器广播的功能，特别适用于家庭视听 AV(AUDIO-VISUAL)的连接。由于该接口具有等时间的传送功能，确保视听 AV 设备重播声音和图像数据质量，具有好的重播效果,严格的讲，IEEE1394 卡像 USB 一样只是通用接口，而不是视频捕捉卡。比如说，我们可以连接一个高速外接硬盘到 IEEE1394 卡上。不过因为 IEEE1394 卡的绝大多数用途是与 DV 数码摄像机相连采集数字视频信号，所以,我们通常把它看作捕捉卡了。目前市场上的 1394 卡可以简单的分成两类：带有硬件 DV 实时编码功能的 DV 卡和用软件实现压缩编码的 1394 卡。带有硬件编码功能的 DV 卡一般价格在数千元，带有硬件编码的 DV 卡可以大大提高 DV 编辑的速度，可以实时地处理一些特技转换，而且许多此类卡带有处理 MPEG-II 视频流的功能。

IEEE1394 和 USB 产生于相同的历史背景，要解决同一个问题，即如何使外设与计算机的连接变得更方便，更简洁。但二者的立足点有所不同，前者立足于高速设备，而后者立足于中低速设备。当然，新的 USB 标准(USB2.0)已提供对高速设备的支持。相比于 USB 接口，早期在 USB1.1 时代，1394a 接口在速度上占据了很大的优势，在 USB2.0 推出后，1394a 接口在速度上的优势不再那么明显。同时现在绝对多数主流的计算机并没有配置 1394 接口，要使用必须要购买相关的接口卡，增加额外的开支。

Apple 公司对 IEEE1394 的评价是：高速，高速，还是高速。这也说明了 IEEE1394 总线的速率和 USB 总线相比，确实要快许多，然而，其电气特性、接口电路和通信协议都要比 USB 复杂，因而价格也高出许多。

下面所列是 IEEE1394 总线的一些特点：

特点一，采用点对点模型，所有连接设备建立一种对等网络，设备之间可以互相通信而不通过主机。

特点二，单一总线最多连接 63 个物理节点，但一个计算机系统中最多可以有 1024 条 IEEE1394 总线。

特点三，支持三种速率模式：100*Mbps*、200*Mbps* 和 400*Mbps*。1394B 又定义了三种更高的速率：800*Mbps*、1.6*Gbps* 和 3.2*Gbps*。而速率的选择是通过在总线上加入不同的共模电流来实现的。

特点四，支持等时和异步两种传输方式。等时传输的概念是按一定的速率进行传输，拥有固定的带宽，和 USB 不同的是，除了点对点的传输外，还可以一对多，进行广播式传输。异步传输通过惟一地址指定响应节点，通信时请求方(即发送方)与响应方(即接收方)需要进行联络。响应方在收到请求时要作出应答表示已收到请求，而请求方在收到响应方对请求所作的响应信息时也要作出应答，表示已收到响应。

特点五，以 125μs 为循环周期。异步传输有至少 20％带宽可用，等时传输则至多 80％。

特点六，采用六线制，包括两对双绞线和一对电源线。一对双绞线传输数据，另一对传输选通信号，数据和选通进行“异或”运算后可得到时钟信号。

特点七，采用四层传输协议，由上至下依次为：总线管理层、事务层、链路层和物理层。总线管理层负责总线配置、电源和带宽管理、节点活动管理等。事务层为支持有关异步传输操作向上层提供服务。链路层负责传输包的生成和分解。物理层提供串行总线接口实现数据比特传输，并实现总线仲裁以确保同一时间上只有一个节点通过总线发送数据。

特点八，总线信号支持三种事件：总线配置、总线仲裁和数据传输。当系统加电或者有设备插入或拔出时会进行总线配置(总线配置无须主机干预)，配置完成后开始数据传输，但

节点在每次传输事务之前需首先通过总线仲裁事件获得总线控制权。

特点九，支持即插即用。

特点十，设备可以自供电或由总线供电。在自供电时还可以向总线供电。

IEEE1394 总线可以连接多种外部设备，其中包括：大容量存储器、视频输出设备、数码相机、高速打印机、娱乐设备、机顶盒、小型网络和视频会议设备等。当然，能够连接到IEEE1394 总线的设备必须符合 IEEE1394 总线规范，具有相应的 IEEE1394 总线接口。

9.6 新一代总线

9.6.1 PCI Express 总线

随着技术的发展，PCI 总线已经无法满足电脑性能提升的要求，必须由带宽更大、适应性更广、发展潜力更深的新一带总线取而代之，这就是 PCI Express 总线。PCI-Express 是最新的总线和接口标准，由于是第三代输入/输出总线，所以它原来的名称为“3GIO（Third-Generation Input/Output）”，是由英特尔提出的，很明显英特尔的意思是它代表着下一代 I/O 接口标准。交由 PCI-SIG（PCI 特殊兴趣组织）认证发布后才改名为“PCI-Express”。这个新标准将全面取代现行的 PCI 和 AGP，最终实现总线标准的统一。它的主要优势就是数据传输速率高，目前最高可达到 10GB/s 以上，而且还有相当大的发展潜力。PCI Express 也有多种规格，从 PCI Express 1X 到 PCI Express 16X，能满足现在和将来一定时间内出现的低速设备和高速设备的需求。能支持 PCI Express 的主要是英特尔的 i915 和 i925 系列芯片组。当然要实现全面取代 PCI 和 AGP 也需要一个相当长的过程，就象当初 PCI 取代 ISA 一样，都会有个过渡的过程。

PCI Express 采用了目前业内流行的点对点串行连接，比起 PCI 以及更早期的计算机总线的共享并行架构，每个设备都有自己的专用连接，不需要向整个总线请求带宽，而且可以把数据传输率提高到一个很高的频率，达到 PCI 所不能提供的高带宽。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 Express 的双单工连接能提供更高的传输速率和质量，它们之间的差异跟半双工和全双工类似。

尽管 PCI Express 技术规格允许实现 X1（250MB/秒），X2，X4，X8，X12，X16 和 X32 通道规格，但是依目前形式来看，PCI Express X1 和 PCI Express X16 将成为 PCI Express 主流规格，同时芯片组厂商将在南桥芯片当中添加对 PCI Express X1 的支持，在北桥芯片当中添加对 PCI Express X16 的支持。除去提供极高数据传输带宽之外，PCI Express 因为采用串行数据包方式传递数据，所以 PCI Express 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 Express 设备生产成本和体积。PCI Express 接口能够支持热拔插，这也是个不小的飞跃。PCI Express 卡支持的三种电压分别为+3.3*V*、3.3*Vaux* 以及+12*V*。另外，PCI Express 也支持高阶电源管理，支持数据同步传输，为优先传输数据进行带宽优化。

目前，PCI-E 3.0 规范也已经确定，其编码数据速率，比同等情况下的 PCI-E 2.0 规范提高了一倍，X32 端口的双向速率高达 320*Gbps*。

1、PCI Express 总线的技术优势

与 PCI 总线相比，PCI Express 总线主要有下面的技术优势：

优势一，是串行总线，进行点对点传输，每个传输通道独享带宽。

优势二，PCI Express 总线支持双向传输模式和数据分通道传输模式。其中数据分通道传输模式即 PCI Express 总线的 x1、x2、x4、x8、x12、x16 和 x32 多通道连接，x1 单向传输带宽即可达到 250*MB/s*，双向传输带宽更能够达到 500*MB/s*，这个已经不是普通 PCI 总线所能够相比的了。

优势四，PCI Express 总线充分利用先进的点到点互连、基于交换的技术、基于包的协议来实现新的总线性能和特征。电源管理、服务质量（QoS）、热插拔支持、数据完整性、

错误处理机制等也是 PCI Express 总线所支持的高级特征。

优势五，与 PCI 总线良好的继承性，可以保持软件的继承和可靠性。PCI Express 总线关键的 PCI 特征，比如应用模型、存储结构、软件接口等与传统 PCI 总线保持一致，但是并行的 PCI 总线被一种具有高度扩展性的、完全串行的总线所替代。

优势六，PCI Express 总线充分利用先进的点到点互连，降低了系统硬件平台设计的复杂性和难度，从而大大降低了系统的开发制造设计成本，极大地提高系统的性价比和健壮性。从下面表格可以看出，系统总线带宽提高同时，减少了硬件 PIN 的数量，硬件的成本直接下降。

2、PCI Express 的硬件协议

PCI-E 的连接是建立在一个双向的序列的(1-bit)点对点连接基础之上，这称之为“传输通道”。与 PCI 连接形成鲜明对比的是 PCI 是基于总线控制，所有设备共同分享的单向 32 位并行总线。PCI-E 是一个多层协议，由一个对话层，一个数据交换层和一个物理层构成。物理层又可进一步分为逻辑子层和电气子层。逻辑子层又可分为物理代码子层（PCS）和介质访问控制子层(MAC)。

（1）物理层

于使用电力方面，每组流水线使用两个单向的低电压微分信号（LVDS）合计达到 2.5 兆波特。传送及接收不同数据会使用不同的传输通道，每一通道可运作四项资料。两个 PCI-E 设备之间的连接成为“链接”，这形成了 1 组或更多的传输通道。各个设备最少支持 1 传输通道(x1)的链接。也可以有 2，4，8，16，32 个通道的链接。这可以更好的提供双向兼容性。（x2 模式将用于内部接口而非插槽模式）PCI-E 卡能使用在至少与之传输通道相当的插槽上（例如 x1 接口的卡也能工作在 x4 或 x16 的插槽上）。一个支持较多传输通道的插槽可以建立较少的传输通道（例如 8 个通道的插槽能支持 1 个通道）。PCI-E 设备之间的链接将使用两设备中较少通道数的作为标准。一个支持较多通道的设备不能在支持较少通道的插槽上正常工作，例如 x4 接口的卡不能在 x1 的插槽上正常工作，但它能在 x4 的插槽上只建立 1 个传输通道（x1）。PCI-Express 卡能在同一数据传输通道内传输包括中断在内的全部控制信息。这也方便了与 PCI 的兼容。多传输通道上的数据传输采取交叉存取，这意味着连续字节交叉存取在不同的通道上。这一特性被称之为“数据条纹”，需要非常复杂的硬件支持连续数据的同步存取，也对链接的数据吞吐量要求极高。由于数据填充的需求，数据交叉存取不需要缩小数据包。与其它高速数传输协议一样，时钟信息必须嵌入信号中。在物理层上，PCI-E 采用常见的 8B/10B 代码方式来确保连续的 1 和 0 字符串长度符合标准，这样保证接收端不会误读。编码方案用 10 位编码比特代替 8 个未编码比特来传输数据，占用 20%的总带宽。有些协议（如 SONET）使用另外的编码结构如“不规则”在数据流中嵌入时钟信息。PCI-E 的特性也定义了一种“不规则化”的运算方法，但这种方法与 SONET 完全不同，它的方法主要用来避免数据传输过程中的数据重复而出现数据散射。第一代 PCI-E 采用 2.5 兆位单信号传输率，PCI-SIG 计划在未来版本中增强到 5~10 兆位。

（2）数据链接层

数据链接层采用按序的交换层信息包(Transaction Layer Packets,TLPs)，是由交换层生成，按 32 位循环冗余校验码(CRC，本文中用 LCRC)进行数据保护，采用著名的协议（Ack and Nak signaling）的信息包。TLPs 能通过 LCRC 校验和连续性校验的称为 Ack(命令正确应答)；没有通过校验的称为 Nak(没有应答)。没有应答的 TLPs 或者等待超时的 TLPs 会被重新传输。这些内容存储在数据链接层的缓存内。这样可以确保 TLPs 的传输不受电子噪音干扰。

Ack 和 Nak 信号由低层的信息包传送，这些包被称为数据链接层信息包（Data Link Layer Packet,DLLP）。DLLP 也用来传送两个互连设备的交换层之间的流控制信息和实现电源管理

功能。

（3）交换层

PCI Express 采用分离交换（数据提交和应答在时间上分离），可保证传输通道在目标端设备等待发送回应信息传送其它数据信息。它采用了可信性流控制。这一模式下，一个设备广播它可接收缓存的初始可信信号量。链接另一方的设备会在发送数据时统计每一发送的 TLP 所占用的可信信号量，直至达到接收端初始可信信号最高值。接收端在处理完毕缓存中的 TLP 后，它会回送发送端一个比初始值更大的可信信号量。可信信号统计是定制的标准计数器，这一算法的优势，相对于其他算法，如握手传输协议等，在于可信信号的回传反应时间不会影响系统性能，因为如果双方设备的缓存足够大的话，是不会出现达到可信信号最高值的情况，这样发送数据不会停顿。第一代 PCI-E 标称可支持每传输通道单向每秒 250 兆字节的数据传输率。这一数字是根据物理信号率 2500 兆波特除以编码率（10 位/每字节）计算而得。这意味着一个 16 通道（x16）的 PCI-E 卡理论上可以达到单向 250*16=4000 兆字节/秒（3.7G 兆字节/每秒）。实际的传输率要根据数据有效载荷率，即依赖于数据的本身特性，这是由更高层(软件)应用程序和中间协议层决定。PCI Express 与其它高速序列连接系统相似，它依赖于传输的鲁棒性（CRC 校验和 Ack 算法）。长时间连续的单向数据传输（如高速存储设备）会造成>95%的 PCI-E 通道数据占用率。这样的传输受益于增加的传输通道，但大多数应用程序如 USB 或以太网络控制器会把传输内容拆成小的数据包，同时还会强制加上确认信号。这类数据传输由于增加了数据包的解析和强制中断，降低了传输通道的效率。这种效率的降低并非只出现在 PCI-E 上。

3、PCI Express 总线的特点

（1）易于布线、减少串扰，多方式连接

PCI Express 导线数量比 PCI 减少近 75%，数据不需要同步，在同一系统内能够以不同频率运行，而且能够延伸到系统之外，采用专用线缆可将各种外设直接与系统内的 PCI Express 总线连接在一起。这是 PCI 无法做到的。

（2）PCI Express 数据传输速率快每个通道带宽为 2.5Gb/s（可提升到 5Gb/s），理论上最高连接带宽可达 8-10GB/s。

（3）兼容 PCI 和 PCI-X

（4）其它功能

PCI Express 接口标准可以支持不同的信令协议；采用先进电源管理技术，支持热插拔功能；可以对所有的接入设备进行实时监控；采用独特的纠错机制保证整个系统的稳定运行。

9.6.2 USB 总线

USB(Universal Serial Bus)称为通用串行总线，是在 20 世纪 90 年代中期由 Compaq、DEC、IBM、Intel、Microsoft 和 NEC 等多家美国和日本公司共同提出的新一代接口标准总线。1994 年，Intel、Digital、IBM、Microsoft、NEC、Northern Telecom 等几家世界著名的计算机和通信公司成立了 USB 论坛，花了近两年的时间形成了统一的意见，于 1995 年 11 月正式制定了 USB0.9 通用串行总线规范，1997 年，真正符合 USB 技术标准的外设出现了。1999 年初在 Intel 的开发者论坛大会上，与会者介绍了 USB2.0 规范，该规范的支持者除了原有的 Intel、Microsoft 和 NEC 等成员外，还有惠普、朗讯和飞利浦三个新成员。USB2.0 向下兼容 USB1.1，传输率将达到 480Mbps，还支持宽带数字摄像设备及下一代扫描仪、打印机及存储设备。USB 总线接口如图 **9-10** 所示。

图 9-10　USB 总线接口

USB 总线是为了适应微机系统应用的日益广泛，需要连接的外部设备不断增加，解决微机端口短缺而产生的。它和 IEEE1394 同样是一种连接外围设备的机外总线。从性能上看，最初 USB 总线在很多方面不如 IEEE1394，但是却拥有 IEEE1394 无法比拟的价格优势，在一段时期内，它和 IEEE1394 总线并存，分别管理低速和高速外设。随后由于廉价的 USB 总线具有热插拔的魅力，使得 USB 大量出现在主板上，出现在打印机、扫描仪、摄像头、优盘，甚至是显示器等设备上。

USB 是一个外部总线标准，用于规范电脑与外部设备的连接和通讯。USB 接口支持设备的即插即用和热插拔功能。

USB 使用一个四针的插头作为标准插头，采用菊花链形式可以把所有的外设连接起来。USB 接口可用于连接多达 127 种外设，如鼠标、调制解调器和键盘等。USB 自从 1996 年推出后，已成功替代串口和并口，并成为当今个人电脑和大量智能设备的必配的接口之一。

1、USB 总线的性能

USB 是一条串行总线，传统的串行端口是各自独立的，例如，键盘端口只能连接键盘，鼠标器端口只能连接鼠标器，它们相互间并不连通。但在 USB 总线上，各 USB 端口是互相关联的，除连接自己的外设外，也同时连接到 USB 提供的 4 根连线上，其中 2 根为信号传输线，2 根为电源连接线。但是，同其他总线的工作相似，在某一特定的时间，信号传输线只能与某一特定的外设相连通。也就是说，在同一时间内只能有一台外设获得 USB 的控制权，以确保各台外设传送的信号互不干扰。

区别于传统端口一个端口只能连接一台外设，USB 端口连接的外设可以有两类：一类是单个的外设，在 USB 术语中称为“功能单元”（Function）；另一类是“集线器”（Hub），它带有连接其他外设的 USB 端口，使 USB 可经过它再连接到其他外设上。由于集线器具有多路转换的功能，所以不论有多少台外设连到集线器上，在同一时刻只有一台外设可以通过集线器与 USB 相连。采用集线器之后，USB 的拓扑结构将呈现树状结构，其连接的外设总台数可以达到 127 台。

2、USB 总线的特点

特点一，可以热插拔，这就让用户在使用外接设备时，不需要重复“关机将并口或串口电缆接上再开机”这样的动作，而是直接在 PC 开机时，就可以将 USB 电缆插上使用；

特点二，标准统一，大家常见的是 IDE 接口的硬盘，串口的鼠标键盘，并口的打印机扫描仪，可是有了 USB 之后，这些应用外设统统可以用同样的标准与 PC 连接，这时就有了 USB 硬盘、USB 鼠标、USB 打印机，等等；

特点三，可接入多达 127 个设备，目前计算机外设越来越多，PC 机内有限的插槽和接口已经不能满足要求，USB 缓解了这一矛盾；

特点四，携带方便，USB 设备大多以“小、轻、薄”见长，对用户来说，同样 20G 的硬盘，USB 硬盘比 IDE 硬盘要轻一半的重量，在想要随身携带大量数据时，当然 USB 硬盘会是首要之选了；

特点五，可即插即用；

特点六，适用于低速外设的连接。

3、USB 的应用

随着计算机硬件飞速发展，外围设备日益增多，键盘、鼠标、调制解调器、打印机、扫描仪早已为人所共知，数码相机、MP3 随身听接踵而至，这么多的设备，如何接入个人计算机？USB 就是基于这个目的产生的。USB 是一个使计算机周边设备连接标准化、单一化的接口，其规格是由 Intel、NEC、Compaq、DEC、IBM、Microsoft、Northern Telecom 联系制定的。

USB1.1 标准接口传输速率为 12Mbps，但是一个 USB 设备最多只可以得到 6Mbps 的传输频宽。因此若要外接光驱,至多能接六倍速光驱，无法再高。而若要即时播放 MPEG-1 的 VCD 影片,至少要 1.5Mbps 的传输频宽,这点 USB 办得到,但是要完成数据量大四倍的 MPEG-2 的 DVD 影片播放，USB 可能就很吃力了，若再加上 AC-3 音频数据，USB 设备就很难实现即时播放了。

不过，并非所有的 Windows 系统都支持 USB。目前，Windows 系统中有许多不同的版本，在这些版本中，只有 Windows98 以上版本的系统对 USB 的支持较好，而其他的 Windows 版本并不能完整支持 USB。例如 Windows95 的零售版是不支持 USB 的，只有后来与 PC 捆绑销售的 Windows95 版本才支持 USB。

目前 USB 设备虽已被广泛应用，但比较普遍的却是 USB1.1 接口，它的传输速度仅为 12Mbps。举个例子说，当你用 USB1.1 的扫描仪扫一张大小为 40M 的图片，需要 4 分钟之久。 这样的速度，让用户觉得非常不方便，如果有好几张图片要扫的话，就得要有很好的耐心来等待了。

用户的需求，是促进科技发展的动力，厂商也同样认识到了这个瓶颈。这时， COMPAQ、Hewlett Packard、Intel、Lucent、Microsoft、NEC 和 PHILIPS 这 7 家厂商联合制定了 USB 2.0 接口标准。USB 2.0 将设备之间的数据传输速度增加到了 480Mbps，比 USB 1.1 标准快 40 倍左右，速度的提高对于用户的最大好处就是意味着用户可以使用到更高效的外部设备，而且具有多种速度的周边设备都可以被连接到 USB 2.0 的线路上，而且无需担心数据传输时发生瓶颈效应。

所以，如果你用 USB 2.0 的扫描仪，就完全不同了，扫一张 40M 的图片只需半分钟左右的时间，一眨眼就过去了，效率大大提高。

而且，USB2.0 可以使用原来 USB 定义中同样规格的电缆，接头的规格也完全相同，在高速的前提下一样保持了 USB 1.1 的优秀特色，并且，USB 2.0 的设备不会和 USB 1.X 设备在共同使用的时候发生任何冲突。

USB2.0 兼容 USB1.1，也就是说 USB1.1 设备可以和 USB2.0 设备通用，但是这时 USB2.0 设备只能工作在全速状态下(12Mbit/s)。USB2.0 有高速、全速和低速三种工作速度，高速是 480Mbit/s，全速是 12Mbit/s，低速是 1.5Mbit/s。其中全速和低速是为兼容 USB1.1 而设计的，因此选购 USB 产品时不能只听商家宣传 USB2.0,还要搞清楚是高速、全速还是低速设备。USB 总线是一种单向总线，主控制器在 PC 机上，USB 设备不能主动与 PC 机通信。为解决 USB 设备互通信问题，有关厂商又开发了 USB OTG 标准，允许嵌入式系统通过 USB 接口互相通信，从而甩掉了 PC 机。

随着 USB2.0 协议的推出，USB 总线的应用会更加广泛。USB 2.0 将设备之间的数据传输速度增加到了 480Mbps，比 USB1.1 标准快 40 倍左右，速度的提高对于用户的最大好处就是意味着用户可以使用到更高效的外部设备，而且具有多种速度的周边设备都可以被连接到 USB2.0 的线路上，而且无需担心数据传输时发生瓶颈效应。而且，USB2.0 可以使用原来USB 定义中同样规格的电缆，接头的规格也完全相同，在高速的前提下一样保持了USB1.1 的优秀特色。USB 2.0 设备在和 USB 1.X 设备共同使用的时候也不会发生任何冲突。

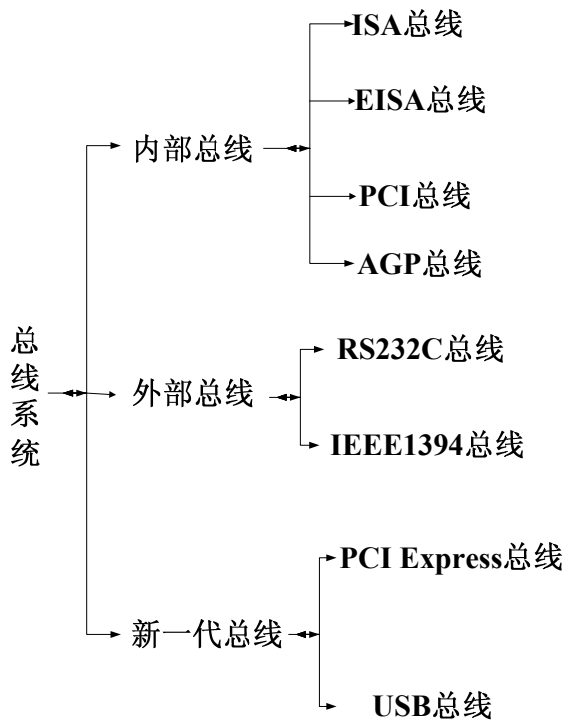
3、USB3.0 简介

英特尔公司（Intel）和业界领先的公司一起携手组建了 USB 3.0 推广组，旨在开发速度超过当今 10 倍的超高效 USB 互联技术。该技术是由英特尔，以及惠普（HP）、NEC、NXP 半导体以及德州仪器（Texas Instruments）等公司共同开发的，应用领域包括个人计算机、消费及移动类产品的快速同步即时传输。随着数字媒体的日益普及以及传输文件的不断增大——甚至超过 25GB，快速同步即时传输已经成为必要的性能需求。

USB 3.0 具有后向兼容标准，并兼具传统 USB 技术的易用性和即插即用功能。该技术的目标是推出比目前连接水平快 10 倍以上的产品，采用与有线 USB 相同的架构。除对 USB 3.0 规格进行优化以实现更低的能耗和更高的协议效率之外，USB 3.0 的端口和线缆能够实现向后兼容，以及支持未来的光纤传输。

从逻辑上说 USB 3.0 将成为下一代最普及的个人电脑有线互联方式，英特尔技术战略师 Jeff Ravencraft 说道：“数字时代需要高速的性能和可靠的互联来实现日常生活中庞大数据量的传输，USB 3.0 可以很好地应对这一挑战，并继续提供用户已习惯并继续期待的 USB 易用性体验。”

关　联



习 题

9.1 判断题

（1）总线是专门用于完成数据传送的一组信号线。（ ）

（2）无条件式的 I/O 是按先读状态口，再读数据口的顺序传送数据的。（ ）

（3）计算机使用总线结构主要优点是便于实现模块化，同时减少了信息传输线的数目。（ ）

（4）在计算机的总线中，地址信息、数据信息和控制信息不能同时出现。（ ）

（5）内部总线是指 CPU 内部连接各逻辑部件的一组数据传输线，它用三态门和多路开关来实现。（ ）

（6）USB 提供的 4 根连线中有 2 根信号线，每一条信号线可以连通一台外设，因此，在某一时刻，可以同时有 2 台外设获得 USB 的控制权。（ ）

（7）组成总线不仅要有传输信息的传输线，还应有实现总线传输控制的器件，它们是总线缓冲器和总线控制器。（ ）

（8）总线的发展是和 CPU 的发展紧密相连的，CPU 的速度提高后，总线的数据传输率如果不随之提高，势必妨碍整机性能的提高。（ ）

9.2 选择题

（1）现代计算机的运算器一般通过总线结构来组织，下述总线结构的运算器中，（ ）的操作速度最快，（ ）的操作速度最慢。

A、单总线结构 B、双总线结构 C、三总线结构 D、多总线结构

（2）把总线分成数据总线、地址总线、控制总线三类是根据（ ）来分的。

A、总线所处的位置 B、总线传送的内容 C、总线的传送方式 D、总线的传送方向

（3）CPU 与 I/O 设备间传送的信号有（ ）。

A、数据信息 B、控制信息 C、状态信息 D、以上三种都是

（4）将处理器、内存储及 I/O 接口连接起来的总线是（ ）。

A、片总线 B、外总线 C、系统总线 D、内部总线

（5）计算机中地址总线的作用是（ ）。

A、 用于选择存储单元 B、用于选择进行信息传输的设备

C、 指定存储单元和 I/O 设备接口电路的选择地址 D、 用于确定操作对象

（6）计算机使用总线结构便于增减外设，同时（ ）。

A、 减少了信息的传输量 B、提高了信息的传输量

C、 减少了信息传输线的条数 D、增加了信息传输线的条数

（7）在计算机中将各个主要组成部件连接起来，组成一个可扩充基本系统的总线称之为（ ）。

A、 外部总线 B、 内部总线 C、 局部总线 D、 系统总线

（8）状态信息是通过（ ）总线进行传送的。

A、数据 B、地址 C、控制 D、外部

（9）下列总线中，属于局部总线的是（ ）。

A、ISA B、EISA C、MCA D、PCI

（10）为协调计算机系统各部件工作，需有一种器件来提供统一的时钟标准，这个器件是（ ）。

A、总线缓冲器 B、总线控制器 C、时钟发生器 D、操作命令产生器

9.3 填空题

（1）在链式查询、计数器定时查询、独立请求三种总线控制判优方式中，响应时间最快的是_____方式；对电路故障最敏感的是_____方式。

（2）在单总线、双总线、三总线三种系统中，从信息流传送效率的角度看，_____的工作效率最低；从吞吐量来看，_____最强。

（3）连接在单总线上的设备均以_____或_____的形式申请使用总线。

（4）在单总线结构的计算机系统中，每个时刻只能有两个设备进行通信，在这两个设备中，获得总线控制权的设备叫_____，由它指定并与之通信的设备叫 _____。

（5）标准微机总线中，PC/AT 总线是 _____位总线，MCA 总线是 _____位总线，EISA 总线是_____位总线，PCI 总线是_____位总线。

（6）AGP 总线不同于通用的 PCI 局部总线，它是供 _____专用的。它在_____与系统内存之间提供了一条直接的访问途径。

（7）USB 串行接口通过使用_____，可以使一台 PC 机连接的外部设备数多达_____台。

（8）在计算机系统中根据总线所传输的信息内容的不同，总线可分为_____、_____和_____。任何类型计算机的总线都包含有这三种总线。

9.4 什么叫总线？为什么各种计算机系统中普遍采用总线式结构？

9.5 总线按照功能分类可分为哪几类，各完成什么功能？

9.6 总线的结构类型有哪些，都有什么特点？

第 10 章　并行处理和互连网络

【内容摘要】

并行处理（Parallel Processing）是计算机系统中能同时执行两个或更多个处理机的一种计算方法，是提高系统性能的主要手段，是信息处理的一种有效形式。互连网络已成为并行处理系统的核心组成部分，它对并行处理系统的性能起着决定性的作用。

【学习要点】

- 并行处理概念、基本结构和特点
- 典型 SIMD 计算机结构和应用
- 互连网络的概念和互连函数
- 静态和动态互连网络
- 互连网络消息传递机制和死锁

10.1　并行处理的概念

10.1.1 并行性

研究改进计算机系统结构的一个主要方面是如何开发出并行性。并行性（Parallelism）是指问题中具有可同时进行运算或操作的特性。如在同一时刻或同一时间间隔内完成两种或两种以上性质相同或不同的任务为并行性。开发并行性的目的是为了能予以并行处理，以提高解题效率。

并行性有两个含义：一是同时性（Simultaneity），是指两个或多个事件在同一时刻发生在多个资源中；二是并发性（Concurrency），指两个或多个事件在同一时间间隔内发生在多个资源中。

并行处理是一种有效的强调开发计算过程中并行事件的信息处理方式，是提高系统性能的主要手段之一。如在运算器中采用串行结构运算，每次进行一位运算，那么完成 n 位数据的运算要花费 n 个时间单位，如果采用并行结构，设置一个 n 位运算器，则用 1 个时间单位就可完成（理想状态下）。可以看出，在元器件速度相同的条件下，后者的速度几乎是前者的 n 倍。可见，并行处理能大幅度的提高计算机系统的运行速度。

11.1.2 并行性的等级和分类

从不同的角度看，并行性有不同的类型。

1、从计算机系统处理数据的角度

从计算机系统处理数据的角度看，并行性等级由低到高，分别是位串字串（串行单处理机，无并行性）、位并字串（传统并行单处理机）、位片串字并、全并行。

2、从计算机信息加工的各个步骤和阶段的角度

从计算机信息加工的各个步骤和阶段的角度，并行性等级可分为如下 4 种：

（1）存储器操作并行性

存储器操作并行性如并行存储器和相连处理机，可采用单体多字，多体交叉存取等方式，在一个存储周期内访问多个字。

（2）处理器操作步骤并行

处理器操作步骤并行如流水线处理机，指令处理器在执行取指令、分析指令、执行指令等过程中的并行。

（3）处理器操作并行

处理器操作并行如阵列并行处理机，为支持向量、数组运算、可以通过重复设置处理单元进行。

（4）指令、任务、作业的并行

这种并行称为高级并行，指令级以上并行是指多个处理机同时对多条指令及有关的数据进行处理。如多指令流多数据流多处理机、分布处理系统和计算机网络等。

3、从计算机系统中执行程序的角度

从计算机系统中执行程序的角度看，并行性等级由低到高，分别是指令内各微操作之间的并行、多条指令之间的并行、多个任务或进程之间的并行和多个作业或程序之间的并行。

4、从系统结构发展的角度

从系统结构发展来看，并行性等级由低到高，分别是高性能的单处理机、SIMD 并行处理机、多处理机和多计算机系统、非冯·诺依曼计算机。

按照 Flynn 分类法归纳的并行计算机体系结构图谱可如图 10-1 所示。

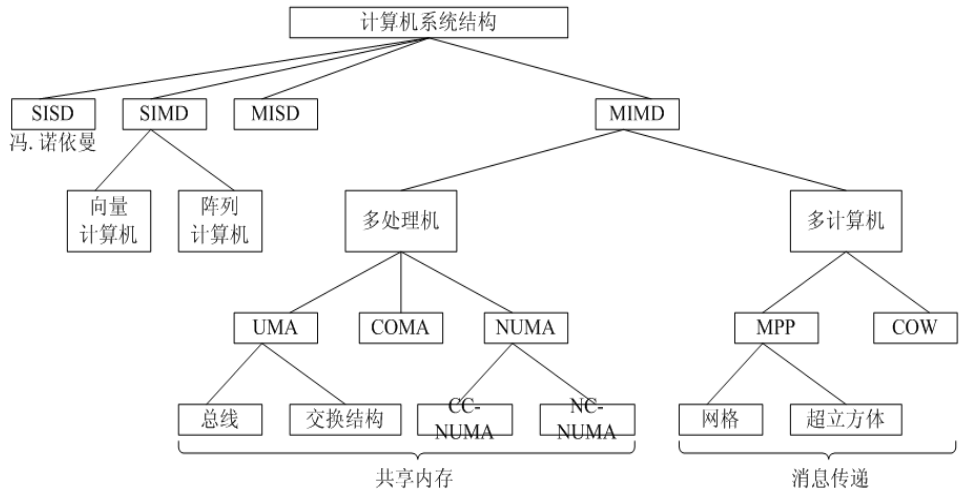


图 10-1 并行计算机的 Flynn 分类图

10.1.3 开发并行性的途径

提高计算机系统的并行性，可通过多种技术途径来实现。通常以时间重叠、资源重复和资源共享为开发并行性的三个主要途径。

1、时间重叠(Time Interleaving)

时间重叠是在并行性概念中引入时间因素，让多个处理过程在时间上相互错开，轮流重叠地使用同一套硬件设备的各个部分，以加快硬件周转而赢得速度，如指令内部各操作步骤采用重叠流水的工作方式。 一条指令的解释分为取指、 分析、执行三大步骤，分别在相应的硬件上完成。只要不出现相关，则每过一个△t 时间，就可以流出结果，从而加快了程序的执行速度。这种时间重叠技术原则上不需要增加更多的硬件设备就可以提高计算机系统的性能价格比。

2、资源重复(Resource Replication)

资源重复是在并行性中引入空间的因素。它是靠重复设置硬件资源来提高可靠性或性能，如通过使用两台或多台完全相同的处理器或计算机完成同样的任务来提高性能。典型的例子是双工系统、相连处理机和阵列处理机等。

3、资源共享

资源共享是用软件方法让多个用户共用同一套资源，通过提高系统资源的利用率来提高系统的性能和效率。典型的例子是多道程序分时系统、它是利用共享 CPU、主存资源以降低系统价格来提高设备利用率的一个实例。例子还包括计算机网络和分布处理系统等。

沿时间重叠途径，多个处理机进行流水线构成的多处理机，一般都是非对称型或异构型的；沿资源重复途径，构成的相联处理机和阵列处理机都是对称型或同构型的多处理机；沿资源共享途径发展的多处理机则既可以是同构型的，也可以是异构型的。

10.2　并行处理机基本结构

并行处理机也称阵列机，是采用资源重复的并行性措施实现。它是指将大量重复设置的多个处理单元 PE（Processing Element）按一定方式互连成阵列，在统一的控制部件 CU(Control Unit)控制下，对各自所分配的不同数据并行执行同一指令规定的操作，是操作并行的 SIMD(单指令流多数据流)计算机。它采用资源重复的措施开发并行性，是以 SIMD 方式工作的。这里的 PE 是指不带指令控制部件的算术逻辑运算单元。

10.2.1 并行处理机的两种典型结构

并行处理机通常由一个控制器 CU（Control Unit，CU）、N 个处理器单元 PE(Processing Element，PE)、M 个存储模块以及一个互连网络部件(Inter Connection Network，ICN)组成。根据其中存储器模块的分布方式，并行处理机可分为两种基本结构：分布式存储器的并行处理机和共享存储器的并行处理机。这两种结构的共同特点是在整个系统中设置多个处理单元，各个处理单元按照一定的连接方式交换信息，在统一的控制部件作用下，各自对分配来的数据并行地完成同一条指令所规定的操作。下面分别对这两种基本结构进行介绍。

1、分布式存储器结构并行处理机

分布式存储器结构并行处理机具备 4 个特点：①包含重复设置的多个同样的处理单元 PE，通过数据寻径网络以一定方式互相连接；②各 PE 都拥有自己的局部存储器 PEM (Processing Element Memory)，存放被分配的数据并只能被本处理单元直接访问；③在统一的阵列控制部件作用下，实现并行操作；④程序和数据通过主机装入控制存储器。由于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。其结构图如图 10-2 所示。

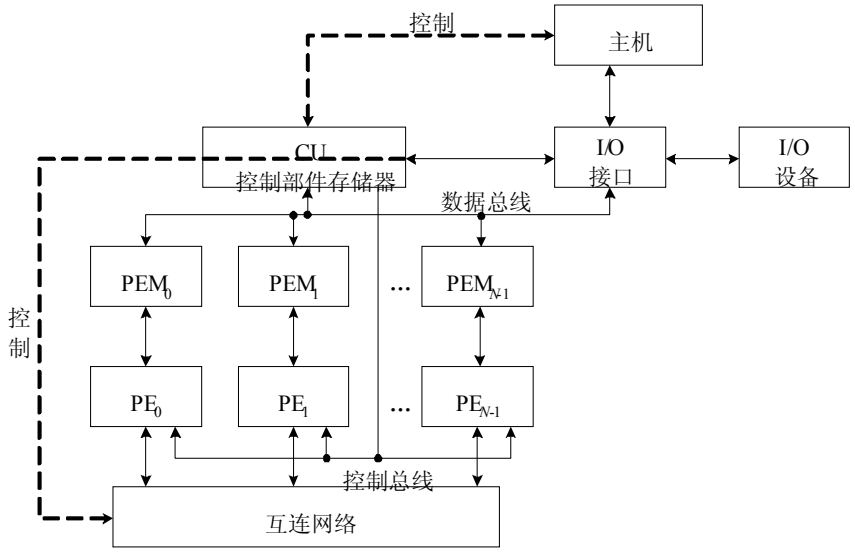


图 10-2 分布式存储器的并行处理机结构图

这种结构的并行处理机种处理单元的数目和存储体的数目是相同的，每个处理单元仅和自己的存储体之间相连，直接交换数据。在实现时，为了有效地进行高速处理，数据应在各存储体间合理分配，使各处理单元都可以依靠自身存储体中的数据进行运算。各数据单元之间交换数据是经过两条途径相互联系：一条是直接通过互连网络；另外一条是如果在某个存储体中存有各处理单元都需要的公共数据，则可以通过将处理单元局部存储体 PEM 读入控制单元，然后通过公共数据总线“广播”到全部处理单元中。在处理单元很多的并行处理机中，PE 之间的互连网络只能是有限而固定的连接。

ILLIAC-IV 是这种结构的 SIMD 机器，它由 64 个本地存储器的 PE 组成，PE 间通过 8×8

环绕连接网格实现互连,因而也称为阵列处理机，其处理速度达每秒 150×106 次 64 位浮点加法运算。各种 SIMD 机器的主要差别在于进行 PE 之间互相通信的数据寻径网络不同。目前的大部分并行处理机是基于分布式存储器模型的系统。

2、共享存储器并行处理机

共享存储器并行处理机结构有 M 个存储体构成统一的并行处理机存储器，经过互连网络 ICN 连接，为全部处理单元 PE_0~PE_{n-1} 所共享，其结构图如图 10-3 所示。

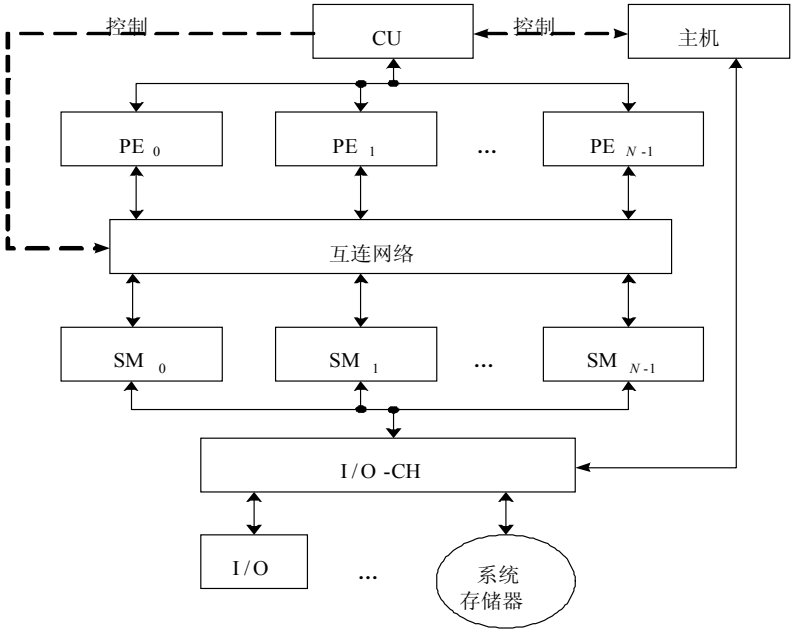


图 10-3 共享存储器的并行处理机结构图

为了使各处理单元能同时对向量的各个元素进行并行处理，一般都使存储体的数量 M 大于或等于处理单元的数量 n。同时，在存储体之间合理分配数据，使各处理单元数据能来自不同的存储体，从而最少化受存储器冲突的影响。从图 10-3 中可以看出，在这种结构中，互连网络是处理单元和存储器之间交换数据的通道。它提供多种连接方式，使每个处理单元之间的数据传送在大多数向量运算不受影响的情况下是非常重要的。同时也可以看到，这种互连网络是复杂的，因此，这种结构在处理单元数量不大的情况下是很理想的。如 Burroughs 公司和伊利诺斯大学在 1979 年研制的科学处理机 BSP，就属于这种结构，它有 16 个处理单元，17 个存储体，最高的向量运算速度可达 50×10^6 次浮点加法运算。

11.2.2 并行处理机的特点

并行处理机利用多个处理单元对向量或数组所包含的各个分量同时进行运算，从而易于获得很高的处理速度。与同样擅长于向量处理的流水线处理机相比，并行处理机有如下特点：

1、多处理单元的互连网络连接

互连网络是并行处理的最有特色的一个组成部分，它规定了处理单元的连接方式，决定了并行处理能适应的算法类别，对系统整体的各项性能指标产生了重要的影响。

2、依靠资源重复并行措施

并行处理机获得高速度的主要原因就是使用大量的处理单元对向量所包含的各个分量进行运算，每一个处理单元要负责多种处理功能，相当于向量处理机中的多功能流水线部件，但其效率要比单个的功能流水线低。因此，只有在硬件价格降低和系统结构不断改进，并行处理机才具有较好的性能价格比。

3、同时性

它的每个处理单元在同一时刻要同等地担负起各种运算功能，相当于向量处理机的多功

能流水线部件那样，但其效率(设备利用率)可能没有多个单功能流水线部件那样高。

4、运算速度高

主要是靠增大处理单元个数，与依靠缩短时钟周期地向量流水线处理机来说，速度提高的潜力要大得多。

正像并行处理机的特点描述一样，并行处理机主要是由处理单元代替了向量处理机中的流水线，用设备的重复设置达到高速运算的目的。事实上，SIMD 计算机正是属于多处理单元的这一类，下面将重点介绍 SIMD 计算机的结构与设计。

10.3 SIMD 计算机基本结构

SIMD 计算机属于多处理机单元范畴，它含一个控制单元，多个处理单元，取指令在控制单元的作用下进行，取出的指令经分析译码后送给个处理器协作完成任务，也就是说指令流是单独的，数据流是多倍的。下面着重以典型 SIMD 计算机为例介绍其发展进程、结构和设计。

10.3.1 SIMD 计算机模型

SIMD 计算机的抽象模型可以理解为：在同一个控制部件管理下，有多个处理单元，所以处理单元均收到从控制部件广播来的同一条指令，但操作对象是不同的数据。

H.J.Siegel 提出了 SIMD 计算机的操作模型，如图 10-4 所示。

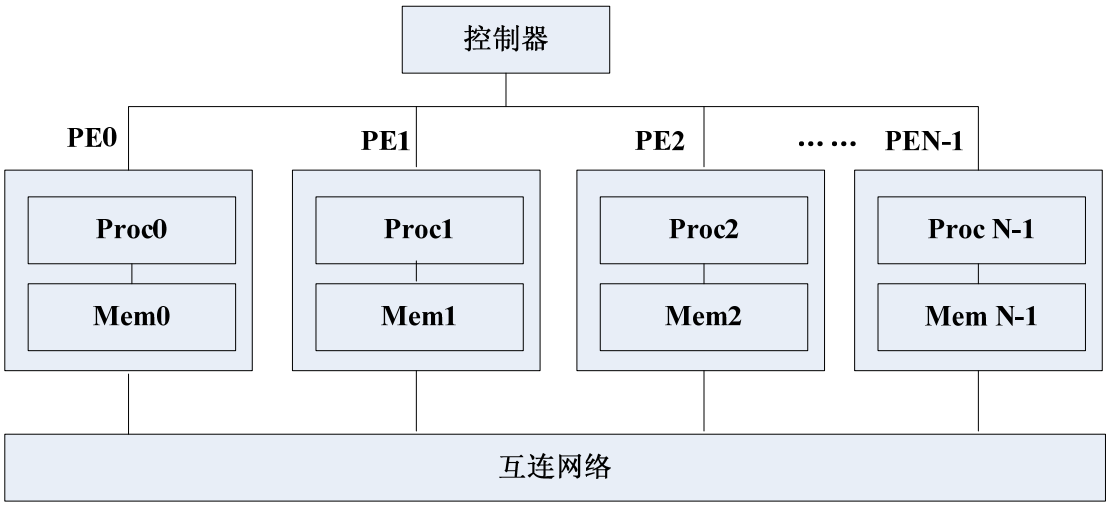


图 10-4 SIMD 计算机的操作模型

SIMD 计算机的操作模型可用五元组表示：M=(N,C,I,M,R)，其字母所代表的含义如下：

N 为机器的处理单元(PE)数。例如，IlliacIV 采用 64 个 PE，连接机(Connection Machine)CM-2 采用 65536 个 PE。

C 为由控制部件(CU)直接执行的指令集，包括标量和程序流控制指令。

I 为由 CU 广播至所有 PE 进行并行执行的指令集，它包括算术运算、逻辑运算、数据寻径、屏蔽以及其它由每个活动的 PE 对它的数据所执行的局部操作。

M 为屏蔽方案集，其中每种屏蔽将 PE 集划分为允许操作和禁止操作两种子集。

R 是数据寻径功能集，说明互连网络中 PE 间通信所需要的各种设置模式。

10.3.2 SIMD 计算机发展过程

Illiac IV 是最先采用 SIMD 计算机结构的计算机。它分成两个分支，一个是为用位片 PE 制造的 SIMD 计算机，如 Goodyear MPP、AMT/DSP610 和 TMC/CM-2，CM-5 是以 SIMD

模式运行的同步 MIMD 计算机；另外一个是用字宽运算 PE 的中粒度 SIMD 计算机，如 BSP 是 16 台处理机和 17 个存储模块同步工作的共享存储 SIMD 计算机，MasPar MP-1 是中粒度 SIMD 计算机。SIMD 计算机的发展过程如图 10-5 所示：

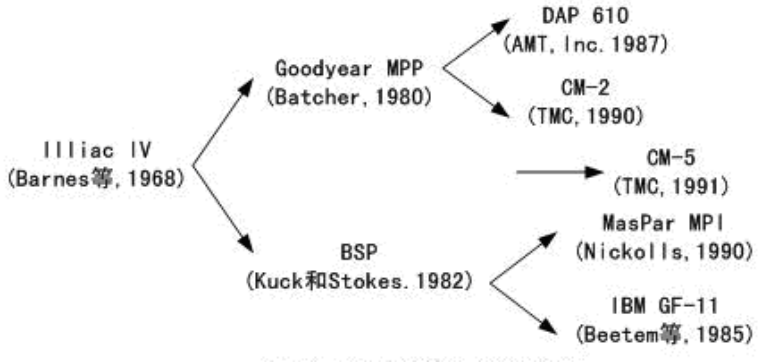


图 10-5 SIMD 计算机的发展过程图

10.3.3 Illiac IV 计算机

Illiac IV 阵列机是世界上最早采用 SIMD 结构设计的计算机，由美国宝来公司（Burroughs）和伊利诺斯大学合作 1965 年研制，并与 1972 年生产。Illiac IV 系统的原理总框图如图 10-6 所示。

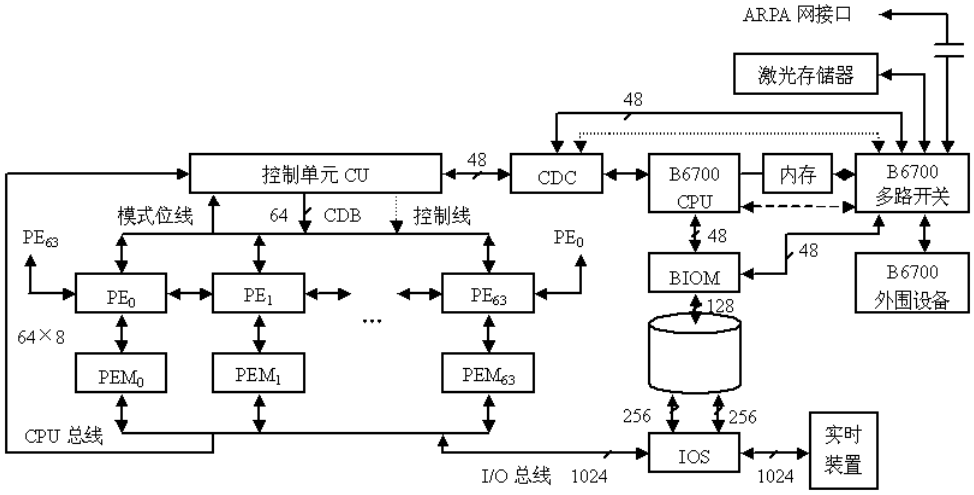


图 10-6 Illiac IV 的系统结构框图

总体由两大部分组成，即 Illiac IV 阵列和 Illiac IV 输入输出系统。具体来讲，它是有三种类型处理机联合组成的多机系统：一是专门对付数组运算的处理单元阵列（Processing element array）；二是阵列控制器（array control unit），它既是处理单元阵列的控制部分，又可以看作一台相对独立的小型标量处理机；三是一台标准的 Burrouths B6700 计算机，担负 Illiac IV 输入输出系统和操作系统管理功能。

1、Illiac IV 阵列

Illiac IV 阵列处理器采用分布式存储器结构。每个处理单元配有专用的存储器模块，共由 64 个处理单元、64 个处理单元存储器和存储器逻辑部件所组成。这个阵列的 64 个处理部件 PU0～PU63 排列成 8×8 的方阵，每一个 PUi 只和其东、西、南、北四个近邻 PUi+1（mod 64）、PUi-1（mod 64）、PUi+8（mod 64）和 PUi-8（mod 64）有直接连接。按照此规则，南北方向上同一列的 PU 两端相连成一个双向环，东西方向上每一行的东端 PU 与下一行的西端 PU 双向相连，最下面一行的东端 PU 则与最上面一行的西端 PU 双向相连，从而

8 行构成一个闭合螺线形状。因此，Illiac IV 的阵列结构又称为闭合螺线阵列。如图 10-7 所示。

这种连接方式即便于一维长向量（最多 64 个元素）的处理，又便于两维数组运算，从而缩短处理单元之间数据传送的路径距离。在这个阵列中，任意两个单元之间的数据传送步距不超过 7 步。一般来讲，这种闭合螺线结构的连接方式，有 n×n 个单元组成的阵列中，任意两个处理单元之间的最短距离不会超过（n-1）步。例如，从 PU_{10} 到 PU_{45} 的距离以下列路径为最短：

$$PU_{10}\rightarrow PU_{1}\rightarrow PU_{57}\rightarrow PU_{56}\rightarrow PU_{48}\rightarrow PU_{47}\rightarrow PU_{46}\rightarrow PU_{45}$$

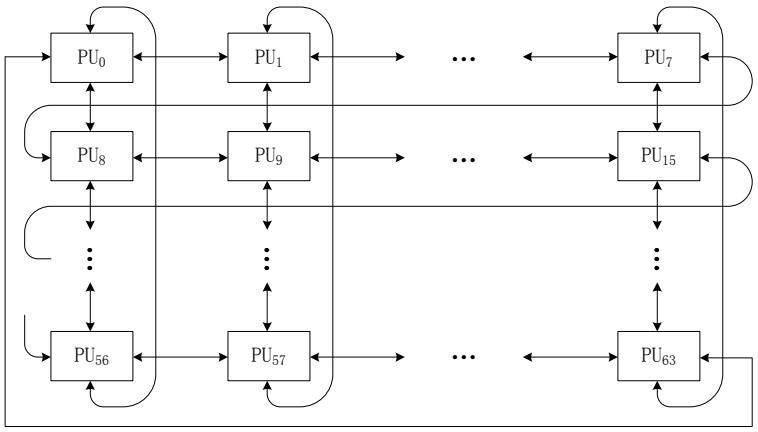


图 10-7 Illiac IV 各处理单元阵列结构的闭合螺线连接方式

处理单元数组处理的运算部分，它可对 64 位、32 位和 8 位操作数进行多种算术和逻辑操作，包括 48 位、24 位或 8 位定点运算。处理单元存储器 PEM 分属每一个处理单元，各有 2048×64 位的存储容量和不大于 350ns 的取数时间。64 个 PEM 联合组成阵列存储器，存放数据和指令。PE 和 PEM 之间经过存储器逻辑部件 MLU 相连，这些部件用于处理机与存储模块接口的逻辑电路，它包含存储器信息寄存器和有关控制逻辑线路，用于实现 PEM 分别和 PE、CU 以及 I/O 之间的信息传送。

2、阵列控制器

阵列控制器 CU 实际上是一台小型控制计算机。它除了对阵列的处理单元实行控制以外，如发控制信号、广播公共地址、广播公共数据，还能利用本身的内部资源执行一整套指令，用以完成标量操作，在时间上与各 PE 的数组操作重叠起来。因此，控制器的功能有以下五个方面：

第一，对指令流进行控制和译码，包括执行一整套标量操作指令；

第二，向各处理单元发出执行数组操作指令所需的控制信号；

第三，产生和向所有处理单元广播公共的地址部分；

第四，产生和向所有处理单元广播公共的数据；

第五，接收和处理由各 PE(如计算出错时)、系统 I/O 操作以及 B6700 所产生的陷阱中断信号。

IlliacIV 阵列控制器 CU 与处理单元阵列之间的信息联系有以下四条信息通路：

（1）CU 总线

处理单元存储器 PEM 经过 CU 总线把指令和数据送往阵列控制器，以 8 个 64 位字为一信息块。这里指令是指分布存放在阵列存储器中用户程序的指令；而数据可以是处理所需的公共数据，先将它们送到 CU，再利用 CU 的广播功能送到各处理单元。

（2）公共数据总线 CDB (Common Data Bus)

这是 64 位总线，用作向 64 个处理单元同时广播公共数据的通路。

（3）模式位线(mode bit line)

每一个单元都可以经过模式位线把它的模式寄存器(mode register)状态送到 CU 中来，送来的信息中也包括该处理单元的"活动"状态位。只有那些处于"活动"状态的处理单元才执行单指令流所规定的公共操作。

（4）指令控制线

处理单元微操作控制信号和处理单元存储器地址、读/写控制信号都经过约 200 根指令控制线由 CU 送到阵列处理单元 PE 和存储器逻辑部件 MLU 中来。

3、输入输出系统

Illiac IV 输入/输出系统由磁盘文件系统 DFS（Disk File System）、I/O 分系统和 B6700 处理机组成。

磁盘文件系统 DFS 是两套大容量并行读写磁盘系统及其相应的控制器。I/O 分系统又输入输出开关 IOS（Input Output Switch）、控制描述字控制器 CDC(Control Description Word Controller)和输入输出缓冲存储器 BIOM(Buffer of Input and Output Memory)三个部分组成 IOS 的功能有二：一是作为名副其实的开关，用以把 DFS 或可能连上的实时装置转接到阵列存储器，进行大批数据的 I/O 传送；二是作为 DFS 和 PEM 之间的缓冲，以平衡两边不同的数据宽度。CDC 的功能是对阵列控制器的 I/O 请求进行管理。BIOM 处在 DFS 和 B6700 之间，是为了取得二者之间传送频带的匹配用到的缓冲。

B6700 管理计算机的基本组成部分是：一般配置一个 CPU(另一 CPU 可选)、32K 字内存(最大可扩充至 512K 字)和经过多路开关控制的一大批外围设备(包括一台容量为 1012 位的激光外存储器以及 ARPA 网络接口)。B6700 的作用是管理全部系统资源，完成用户程序的编译或汇编，为 Illiac IV 进行作业调度、存储分配、产生入/出控制描述字送至 CDC、处理中断，以及提供操作系统所具备的其它服务等。

10.3.4 Burroughs BSP 计算机

BSP(BSP—Burroughs Scientific Processor)是美国 Burroughs 公司和伊里诺大学合作设计的用于科学计算的并行处理机，于 1979 年生产。与 Illiac IV 不同，这台计算机是采用共享集中式主存结构的计算机，是共享存储器 SIMD 结构的典型代表。BSP 计算机系统组成如图 10-8 所示。它由系统管理计算机 B7700/B7800 和 BSP 处理机两大部分组成 。

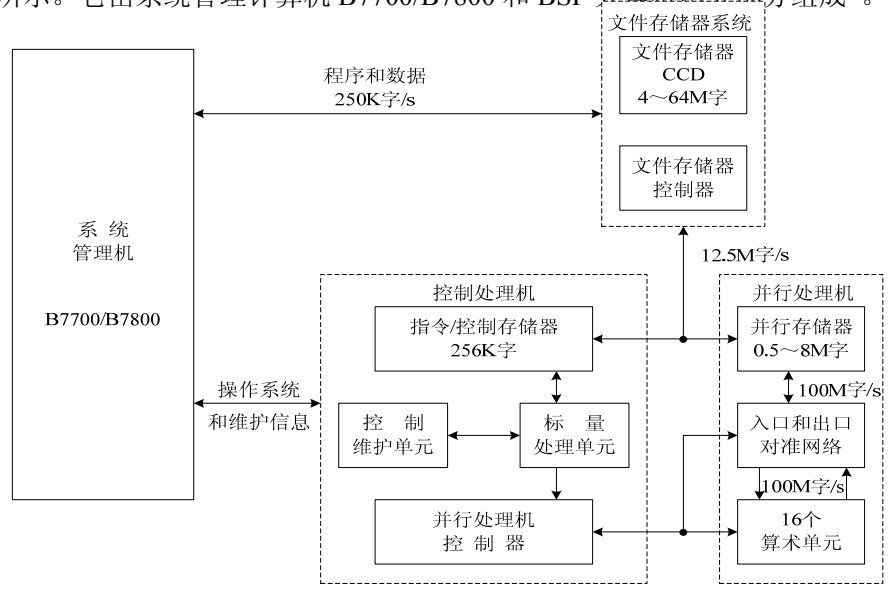


图 10-8 BSP 科学计算机系统结构图

BSP 不能够单独工作，而是作为管理机 B7700/B7800 的后台机工作的。BSP 大部分与分布式相同，区别是它的系统的存储器是由 K 个存储体(M_0～M_{k-1})集中在一起构成，经过互

连网络 ICN 为全部 N 个处理单元(PE_0～PE_{N-1})所共享。

这种结构形式中，为使各处理单元对长度为 N 的向量中的各个元素都能同时并行处理，存储体的体数 K 通常总是等于或多于处理单元的个数 N。为了各处理单元在访问主存时，尽可能避免发生分体冲突，也要求有合适的算法能够将数据按一定规律合理地分配到各个存储体中。与分布式存储器结构另一个不同之处在于互连网络 ICN 的作用不同。集中式主存的结构形式中，互连网络是用来连接处理单元和存储器分体之间的数据通路的，希望能让各个处理单元可有高速、灵活的方式连到不同的存储体上。因此有的并行处理机系统上将其称为对准网络(Alignment Network)。

BSP 系统采用了全面并行化操作，即把资源重复和时间重叠两种并行性技术结合起来，有人称为第二代并行处理机。BSP 科学处理机系统由系统管理机（又称系统资源机）和科学处理机两大系统构成。前者担负 BSP 编译，任务调度，数据通讯，外部设备管理等任务；而科学处理机本身又包括控制处理机、文件存储器及并行处理机三大部分。

1、BSP 处理机的组成

BSP 处理机可分为 4 个部分，即并行处理机、控制处理器、文件存储器和对准网络。

（1）并行处理机

并行处理机包含 16 个算术单元、由 17 个存储模块（与 16 最接近的质数）组成的一个无冲突访问的并行存储器和一套对准网络。

并行机中每个处理器以 160ns 的时钟周期进行向量计算。所有 16 个算术单元 AE 对不同的数据组（从并行处理机控制器广播来）进行同一种指令操作，大部分的算术运算能在 2 个时钟周期(320ns)内完成。进行向量运算的数据是存在 17 个并行存储器模块中，每个模块的容量可达 512 千字，17 个存储器模块的组织形成了一个无冲突访问存储器，它容许对任意长度以及跳距不是 17 倍数的向量实现无冲突存取。

（2）控制处理器

控制处理器是一台用于控制的计算机，其核心是标量处理单元，处理机的时钟频率是 1MHz。它除了用以控制并行处理机以外，还提供了与系统管理机相连的接口。标量处理机则处理存储在控制存储器中的全部操作系统和用户程序的指令，即指令/控制存储器。

（3）文件存储器

文件存储器是一个半导体辅助存储器，是一个高速大容量外存储器，是置于 BSP 直接控制下的唯一外围设备。BSP 的计算任务文件是从系统管理机加载到它上面，然后对这些任务进行排队，由控制处理机取出并加以执行。BSP 在运行过程中产生的输出文件和中间结果也都暂存于文件存储器中，由于读写速度较快，可很好的解决处理机和外设之间的带宽瓶颈。

（4）对准网络

对准网络包涵完全交叉开关以及用来实现数据从一个源广播至几个目的地以及当几个源寻找一个目的地址时能分解冲突的硬件。这就需要在算术单元阵列和存储器模块之间具备通用的互连特性，而存储模块和对准网络的组合功能则提供了并行存储器的无冲突访问能力。算术单元也利用输出对准网络来实现一些诸如数据压缩和扩展操作以及快速傅立叶变换算法等专用功能。

2、BSP 的并行存储器

BSP 并行存储器又称为质数存储系统，由 17 个存储模块组成，存储周期为 160ns。BSP 并行存储器的无冲突访问是它的一个独特的技术性能。其实现的硬件技术包括：质数个存储器端口（BSP 并行存储器的存储体数是一个质数 17）、存储器端口和 AE 之间的完全交叉开关（对准网络）、以及特殊的存储器地址生成机构。

3、BSP 的并行流水技术

BSP 系统采用了全面的并行性技术，它并不依靠提高时钟频率获得高速，而是依靠并行

性。在 BSP 中，存储器－存储器型的浮点运算是流水进行的。BSP 的流水线组织由 5 个功能级组成，尤其是并行处理机包括有 16 个处理单元、17 个存储器模块和 2 套互连网络（亦称对准网络）组合在一起，就形成了一条五级的数据流水线，使连续几条向量指令能在时间上重叠起来执行。其结构示意图如图 10-9 所示，五级的功能作用依次是：

①由 17 个存储器模块并行读出 16 个操作数；

②经对准网络 NW1 将 16 个操作数重新排列成 16 个处理单元所需要的次序；

③将排列好的 16 个操作送到并行处理单元完成操作；

④所得的 16 个结果经过对准网络 NW2 重新排列成 17 个存储器模块所需要的次序；

⑤写入存储器。

两套对准网络 NW1/2 的作用分别是在读与写存储器时，使得并行存储器中为保证无冲突访问而错开存放的操作数顺序能够与算术单元并行处理要求的正常顺序一致。整个流水线由统一的指令译码和控制部件进行控制。这种流水线结构是很新颖的，对提高系统处理效率起着很大的作用。

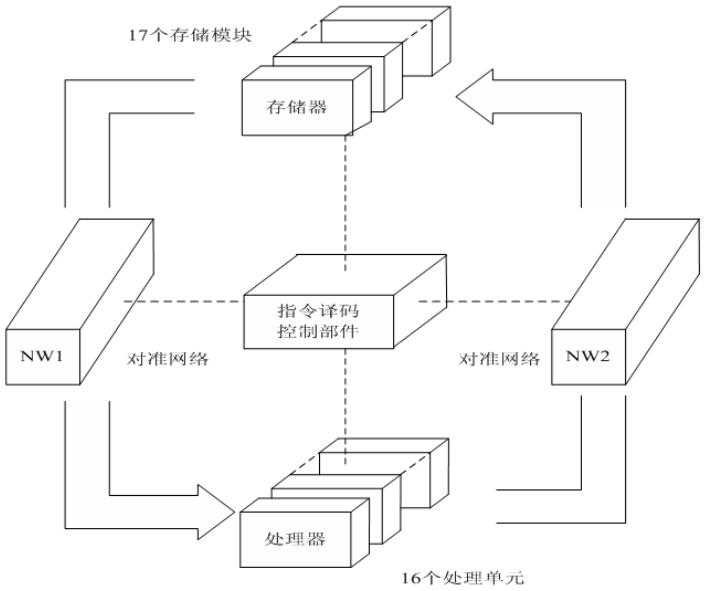


图 10-9 BSP 数据流水线结构示意图

BSP 还有一个高效率的 FORTRAN 编译程序，具有很强的向量化功能，对程序中隐含的并行性保证有较高识别率。向量程序不但能够处理明显的数组操作，还能处理线性递归，循环内部的条件分支等进程，并产生明显的加速效果。

10.3.5 CM-2 计算机

CM-2 是 Thinking Machines Co-operation 于 1987 年推出的 Connection Machine 系列机中的一台高档机，是一台细粒度的 SIMD 计算机，它由数千个位片 PE 组成，它的峰值处理速度超过 10Gflops，它的系统结构如图 10-10 所示。

所有程序从前端开始执行，当需要并行数据操作时，发送微指令到后端处理阵列。定序器(sequencer)分解这些微指令并且把它们广播给阵列中的所有数据处理器(data processor)。前端机和处理阵列之间有三条交换数据计算结果的通路：广播总线(broadcasting)、全局组合总线(global combining)和标量存储器总线(scalar memory bus)，其组成部分介绍如下。

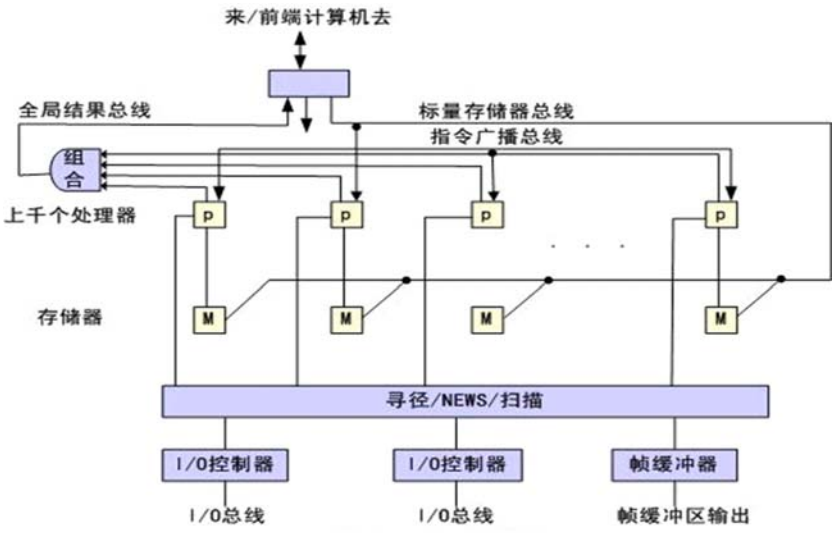


图 10-10 CM-2 的系统结构图

1、处理阵列

CM-2 是一台数据并行计算的后端机。处理阵列包含 4K 到 64K 个位片数据处理器(或 PE)，所有数据处理器都由定序器控制。定序器对来自前端机的微指令进行译码，然后把毫微指令广播到阵列中各个处理器。所有处理器可以同时访问它们的存储器，它们以锁步方式执行广播来的指令。

处理器之间通过寻径、NEWS 网格(NEWS gird)或扫描机构(scanning mechanism)相互交换数据。这些网络也与 I/O 接口相连。称为数据穹(data vault)的大容量存储器子系统与 I/O 相连，它可存储多达 60G 字节的数据。

2、寻径器、NEWS 网格和扫描机构

CM-2 处理机间能够快速高效的通信，主要是通过下列技术构造的互联网络来实现。

（1）寻径器

每个处理器芯片包含一个用于处理器之间数据寻径的专门硬件。把所有处理器芯片上的寻径器结点用线连在一起形成一个布尔 n-立方体。在 CM-2 最大配置时，所有处理器芯片上共有 4096 个寻径器结点，连成一个 12 维的超立方体。

（2）NEWS 网格

每个处理器芯片中的 16 个物理处理器可以排列成 8×2，1×16，4×4，4×2×2 或 2×2×2×2 等形式的网格。规定每个物理处理器有 64 个虚拟处理器。可以想象这 64 个虚拟处理器在芯片中排列成 8×8 网格。

（3）扫描机构

除了通过超立方体寻径器可以动态地重构 NEWS 网格外，CM-2 还有专门的硬件支持对整个 NEWS 网格的扫描或传播。这些都是很有效的并行操作，在整个阵列中进行快速的数据组合和传播。

3、输入输出系统

Connection Machine 不但强调计算的大规模并行性，还强调计算结果的可视化。2 到 16 条高速 I/O 通道用于数据和/或图象 I/O 操作。连接到 I/O 通道的外围设备包括数据穹、CM-HIPPI 系统、CM-IOP 系统和 VME 总线接口控制器，如图 0-13 所示。数据穹（Data Vault）是基于磁盘的海量存储系统，用来存放程序文件和大数据库。

CM-2 已经用于解决几乎所有 MPP 所面临的具有重大挑战性的应用问题。特别是 Connection Machine 系列已经用于借助相关反馈技术的文档检索、基于记忆的推理，如用在

医疗诊断系统 QUACK 中摸拟诊断疾病以及处理工作量很大的自然语言等。CM-2 的其它应用包括 SPICE 的 VLSI 电路分析和布线、计算流体动力学、信号/图象/视觉处理和集成、神经网络模拟和连接模型、动态规划，上下文无关文法分析、射线追踪图以及计算几何等问题。

10.4 SIMD 计算机的应用

本节主要结合 SIMD 计算机的特点给出几种基本的数据处理算法。

10.4.1 计算模型及有限差分

哈辛诺在 1986 年将物理计算模型分成连续模型和离散模型（粒子模型）。

连续模型是指所有计算的时间和空间是连续变化的物理量，且这些物理量是一定范围内的平均值。典型的参数如电荷密度、温度和压力等。粒子模型则将世界看成是由离散的粒子组成的，这些物理量反映单个粒子的当前状态，典型的参数如速度、力和动量等。两种模型只是对同一问题的两种不同观察方法而已。

连续模型和离散模型的主要区别表现在连续模型符合偏微分方程。按离散形式处理时，方程式中所有变量的变化都是其近邻变量的函数，远程变量没有直接的影响。先通过作用于近邻变量，近邻变量再作用于随后的近邻变量，由此向前非直接地作用于整个媒质，也就是通过局部作用而将作用传送到整个媒质。离散模型（粒子模型）允许粒子受远程粒子的影响。任何粒子在任何时刻都与模型中的其它粒子有关。

应用有限差分方法求解连续模型对于并行计算具有很大的吸引力。有限差分方法是求解偏微分方程的一种有效方法，它把一个有规则的网格履盖在整个模型域上，用网格点上变量值写出差分方程组以代替连续模型的偏微分方程来进行计算。作为连续模型，在解决物理问题时，我们考察描述平面域的拉普拉依方程。

$$\frac{\partial^2 U}{\partial x^2}+\frac{\partial^2 U}{\partial y^2}=0$$

将这个方程离散化，即在 x 和 y 方向上每隔一个相同的距离 h 取一个样值点，这个微分方程就可以用差分方程的形式表示。下面的就是二阶偏导数表示为差分形式，式中(x, y)为平面直角坐标，h 为网格间距。

$$\left.\begin{aligned}\frac{\partial^2 U}{\partial x^2}&=\frac{U(x+h,y)-2U(x,y)+U(x-h,y)}{h^2}\\ \frac{\partial^2 U}{\partial y^2}&=\frac{U(x,y+h)-2U(x,y)+U(x,y-h)}{h^2}\end{aligned}\right\}$$

把上述差分方程代入原方程，则可得有限差分计算公式：

$$U(x,y)=\frac{U(x+h,y)+U(x,y+h)+U(x-h,y)+U(x,y-h)}{4}$$

Illiac IV 的阵列结构特别适用于计算这种在网格上定义的有限差分函数。这是因为，根据拉普拉依方程，任一网格点(x, y)上的函数值可由其四周邻近点的函数值计算出来，这一要求正好反映了阵列处理机每一处理单元与其 4 个近邻连接的性质。实际计算时，应利用张弛法进行。每一网格点上的函数值用求其四邻平均值的方法计算，经多次迭代，逐次逼近其最终的平均值。网格边缘的函数值是已知的，由场域的边界条件决定；而对于内部各点的函数值，开始时可选择为零，然后根据拉普拉依方程多次迭代求 U(x, y)值，直至连续二次迭代所求值的差小于规定误差为止(已知迭代过程是收敛的)。

IlliacIV 在计算时，是把内部网格点分配给各个处理单元的。因此，上述计算过程可以

并行地完成，从而可几十倍地提高处理速度。由于实际问题中所遇到的内部网格点数目往往是很大的，因此需要将其分成许多子网格，然后才能在 Illiac IV 上求解。

10.4.2 阵列处理机的几种基本算法

1、矩阵加

在阵列处理机上，实现矩阵加的算法是最简单的一维数组运算。设 A 和 B 是 $n\times n$ 阶矩阵，A、B 相加的和矩阵为 C，它也是 $n\times n$ 阶矩阵。矩阵加的算法为

$$A + B = C;\ \text{其中}\ c_{ij} = a_{ij} + b_{ij}$$

2、矩阵乘

设 A 和 B 是 $n\times n$ 阶矩阵，A、B 的乘积矩阵 C 也是 $n\times n$ 阶矩阵。矩阵乘的传统串行算法为 $A\times B = C$；其中

$$c_{ij}=\sum_{k=0}^{n-1}a_{ik}b_{kj}$$

其中 $0\leqslant i\leqslant n-1$ 且 $0\leqslant j\leqslant n-1$。

3、累加和

累加和并行算法是将 N 个数的顺序相加过程变为并行相加过程。串行求和的 FORTRAN 程序如下：

```
    C(-1)＝0
    DO 10 I＝0, N
10  C(I)＝C(I-1)＋A(I)
```

在并行处理机上，采用递归加法，FORTRAN 程序如下：

```
    DO 10 I=0，log2N－1
10  A＝A＋SRL(A, 2**I) ；把 A 向量逻辑右移 2^i 个 PE，只需做 log2N 次加法。
```

例如，当 N=8 时，在 SISD 计算机要用 8 次加法时间。如果在并行处理机上，采用成对递归相加的算法，则只需 $\log_2 8=3$ 次加法时间就够了。首先，原始数据 A(I)，0<I<7，存放在 8 个 PEM 的 a 单元中，然后按照下面的步骤求累加和：

第一步 置全部 PE 为活动状态

第二步 全部 A(I), 0<I<7, 从 PE 的 a 单元读到相应 PE 的 RGA 中；

第三步 令 K＝0；

第四步 全部 PE 的(RGA)转送到 RGR；

第五步 全部 PE 的(RGR)经过互连网络向右传送 2k 步距；

第六步 $j=2^{k-1}-1$;

第七步 置 PE_0 至 PE_j 为不活动状态；

第八步 处于活动状态的 PE 执行(RGA)：＝(RGA)＋(RGR)操作；

第九步 k：=k+1

第十步 若 k<3, 则转回第四步，否则继续往下执行；

第十一步 置全部 PE 为活动状态；

第十二步 全部 PE 的(RGA)存入相应 PEM 的 a+1 单元中。

采用 SIMD 并行算法，运算速度提高，加速比为 $N/\log_2 N$ 倍，运算次数增加。从 N 次增加到 $N\cdot\log_2 N$ 次，效率降低，实际效率为 $1/\log_2 N$。例如当 N＝1024，速度提高 100 倍，运算参数增加 10 到倍，效率只有 1/10。如果 N＝220，即 100 万个数求和，速度可以提高 5 万倍。

10.5 互连网络的概念

互连网络是一种由开关元件按照一定的拓扑结构和控制方式将集中式系统或分布式系统中的结点连接起来所构成的网络，这些结点可能是处理器、存储模块或者其他设备，他们

通过互连网络相互连接并进行信息交换。互连网络已成为并行处理系统的核心组成部分，它对并行处理系统的性能起着决定性的作用。

10.5.1 基本概念和作用

在多计算机系统中，无论是处理机之间，还是处理器与存储器之间，都要通过互连网络实现信息交换。决定互连网络性能价格比的诸因素中，主要的是结构复杂性（反应成本）和通信频带及结构灵活性（反应性能）。

如果我们把处理单元或存储体看成结点（Node），互连网络则为输入和输出二组结点之间提供一组互连或映象（Mappings）。对于有 N 个输入结点和 M 个输出结点的情况，则从输入到输出的映象可定义出 N×M 种映象的互连网络，我们称之为广义互连网络（Generailsed Connection Network）。图 10-12 为画出了 3 个输入结点和 2 个输出结点的广义互连网络。它的映象关系可以分为两类：一对多的映象（图 10-11（a）），一对一映象（图 10-11（b））。如果我们限制只有一对一的情况，则共有 N！种映象。对于有 N！种映象的互连网络，我们称之为连接网络（Connection Network）或全排列网络。

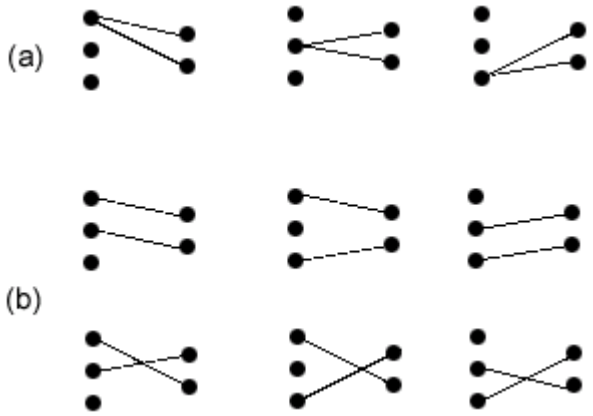


图 10-11 3 个输入结点和 2 个输出结点的网络映像

（a）一对多映像　　（b）一对一映像

实现广义互连网络的直观方法是全交叉开关网络（Full Crosser Network）。它是互连网络中最通用的一种形式，图 10-12 显示了具有 4 个输入和 4 个输出的全交叉开关网络。对于任一输入和输出之间，都有一个交叉点，每个交叉点实际代表一套开关，而且包含必要的用来分解多重访问冲突的硬件设备。这种网络由于任意一对输入和输出之间都有直接连接，因此通信频带宽、灵活性好。但由于交叉点的数量为 N×M 个，再考虑到总线的线数，整个交叉开关非常复杂。当 N 很大时，交叉开关的成本会超过 N+M 个处理单元和存储器的成本。因此，它只用于输入输出端数目较少的情况。如 BSP 用了两个这种网络，一个是 N=16，M=17；另一个是 N=17，M=16。

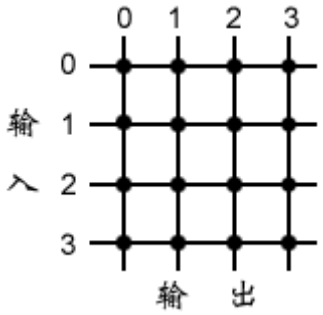


图 10-12 具有 4 个输入和 4 个输出的全交叉开关网络

互联网络主要涉及到链路、交换开关和网络接口电路等。其中链路也称为链或通道，它用来将计算机系统中两个硬件部件进行物理连接；交换开关也称为路由器，它用于建立交换

网络；网络接口电路也称为网卡。

互连网络是一种由开关元件按照一定的拓扑结构和控制方式构成的网络，用来实现计算机系统内部多个处理机或多个功能部件之间的相互连接。随着各个领域对高性能计算的要求越来越高，多处理机和多计算机系统的规模越来越大，处理机之间或处理单元和存储模块之间的通信要求和难度也越来越突出。所以互连网络已成为并行处理系统的核心组成部分，它对整个计算机系统的性能价格比有着决定性的影响。互连网络在多处理机系统中位置和作用如图 10-13 所示。

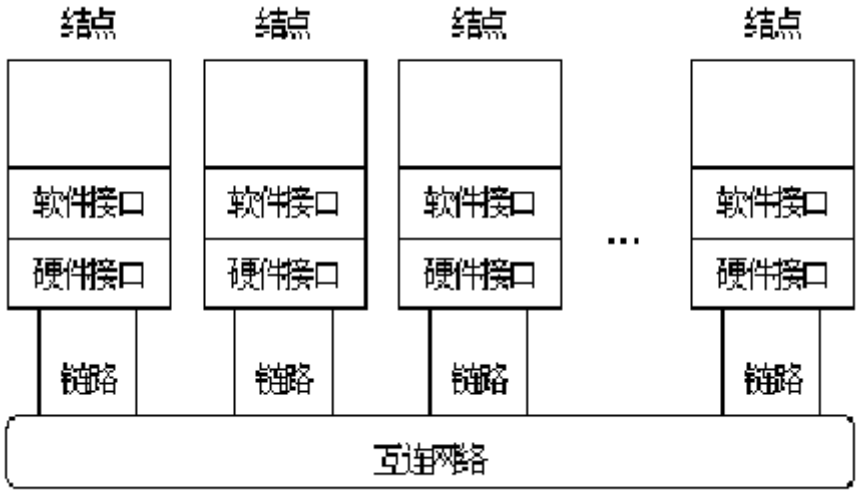


图 10-13 互连网络在多处理机系统中位置和作用

10.5.2 主要特性和性能参数

1、互连网络的结构特性参数

网络是用有向边或无向边连接有限个结点的图来表示。下面从图的角度定义几个用于估算复杂性、通信效率和网络价格的参数，即网络特性。

（1）网络规模

网络中结点数称为网络规模，它表示该网络所能连接的部件多少。

（2）结点度

与结点相连接的边（即链路或通道）数称为结点度，用 d 表示。在单向通道情况下，进入结点的通道数叫做入度，而从结点出来的通道数则称为出度。结点度就是二者之和。结点度反映了结点所需要的 I/O 端口数，也即反映了结点的价格。为了降低价格，应尽可能使它小。为构造可扩展系统，使构件能模块化，要求结点度保持恒定。

（3）距离

两结点之间相连的最少边数。

（4）网络直径

它是网络中任意两个结点之间距离的最大值。它是说明网络通信性能的一个指标。因此从通信的观点来看，网络直径应当尽可能地小。

（5）等分宽度

当某一网络被切成相等的两半时，沿切口的最小边数（通道）称为通道等分宽度，用 b 表示。于是线等分宽度就是 B=b×w，w 为通道宽度（用位表示）。因此，等分宽度是说明沿等分网络最大通信带宽的一个参数。网络的所有其它横截面都应限在等分宽度之内。

（6）结点间的线长

它是两个结点间的线的长度。它会影响信号的时延、时钟扭斜和对功率的需要。

（7）对称性

网络对称性指的是如果从网络任意结点看上去网络的拓扑结构都是相同的，便称该网络是对称的，否则网络是非对称的。对称网络较易实现，编程也较容易。

（8）数据寻径

数据寻径网络用来进行 PE 间数据交换的方式有以下几种：①对一通信也称点对点通信，仅有一个发送者和一个接收者；②对多通信包括广播和散射；③多对一通信包括聚集和归约，④多对多通信中最简单的形式是置换。如图 10-14 形象化的给出几种常见的数据寻径方式。

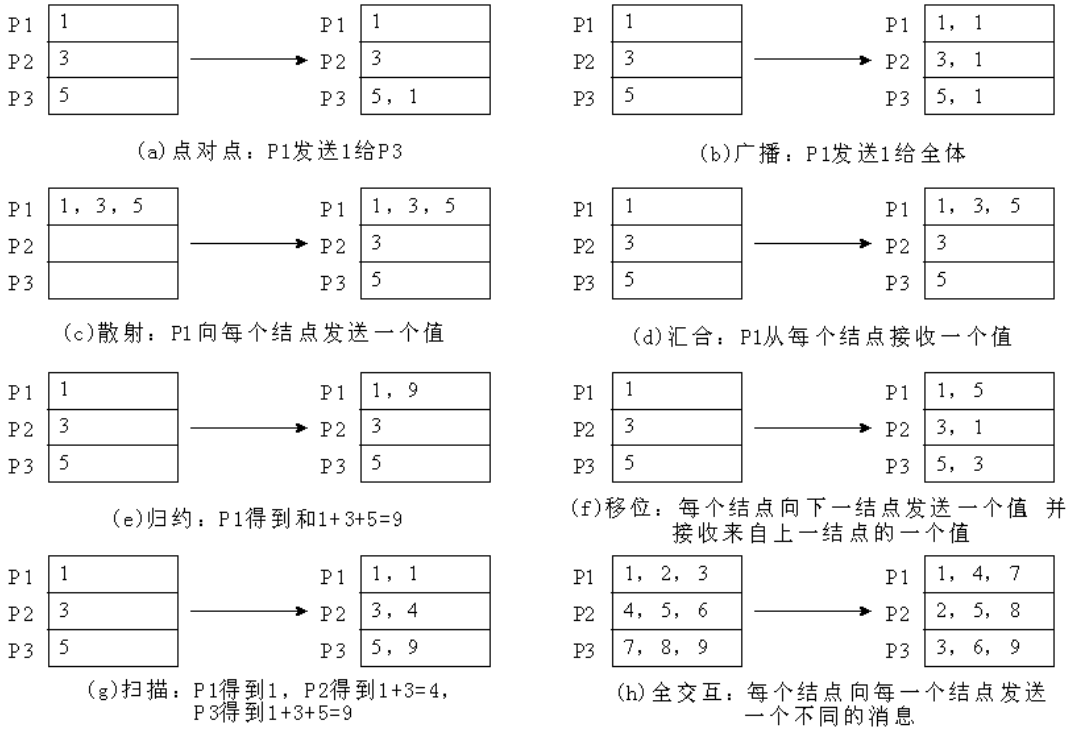


图 10-14 几种常见的数据寻径方式

2、网络的传输性能参数

下面以两台计算机互连的最简单的网络为例讨论网络的传输性能参数。图 10-15 是两台计算机连接的最简单网络，每台计算机有一个 FIFO 的数据队列。

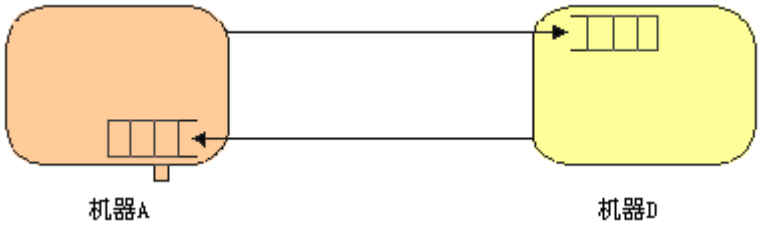


图 10-15 连接两台机器的简单网络

一台机器发送消息给另一台机器时，发送方的步骤如下：

步骤 1：应用程序把要发送的数据拷贝到操作系统的缓冲区。

步骤 2：操作系统根据要发送的数据计算出检查和，并把它加在消息中，同时启动超时计数器。

步骤 3：操作系统把缓冲区中的数据送到网络接口硬件并通知硬件开始发送消息。

消息包的接收和上述步骤正好相反，即：

步骤 1：系统把数据从网络接口硬件拷贝到操作系统缓冲区。

步骤 2：系统根据接收到的数据计算出检查和。如果计算出的检查和与发送过来的检查和匹配，则接收方发一个回答信号给发送方。如果不匹配，则删除这个消息，因为发送方在

超时计数器超时后会重发这个消息。

步骤 3：如果数据通过检查，系统把接收到的数据拷贝到用户地址空间并启动应用程序继续执行。

发送方接收到回答信号后，可以释放系统缓冲区的消息了。如果发送方的超时计数器已超时，那么它重发消息。从消息包的发送和接收步骤上看，一个消息从发送到接收经历的时间过程为：发送方的开销时间、消息在线路上的传播时间、接收方的开销时间。下面给出互连网络传输方面和时延有关的性能参数：

频宽（Bandwidth） 它是指消息进入网络后，互连网络传输信息的最大速率。它的单位是兆位/秒，而不用兆字节/秒。

传输时间（Transmission time） 消息通过网络的时间，它等于消息长度除以频宽。

“飞行”时间（Time of flight） 消息的第一位信息到达接收方所花费的时间，它包括由于网络中转发或其它硬件所起的时延。

传输时延（Transport latency） 它等于"飞行"时间和传输时间之和。它是消息在互连网络上所花费的时间，但不包括消息进入网络和到达目的结点后从网络接口硬件取出数据所花费的时间。

发送方开销（Sender overhead） 处理器把消息放到互连网络的时间，这里包括软件和硬件所花费的时间。

接收方开销（Receiver overhead） 处理器把到达的消息从互连网络取出来的时间，这里包括软件和硬件所花的时间。所以一个消息的总时延可以用下面公式表示：总时延＝发送方开销＋"飞行"时间＋传输时间＋接收方开销

这几个性能参数的关系如图 10-16 所示。

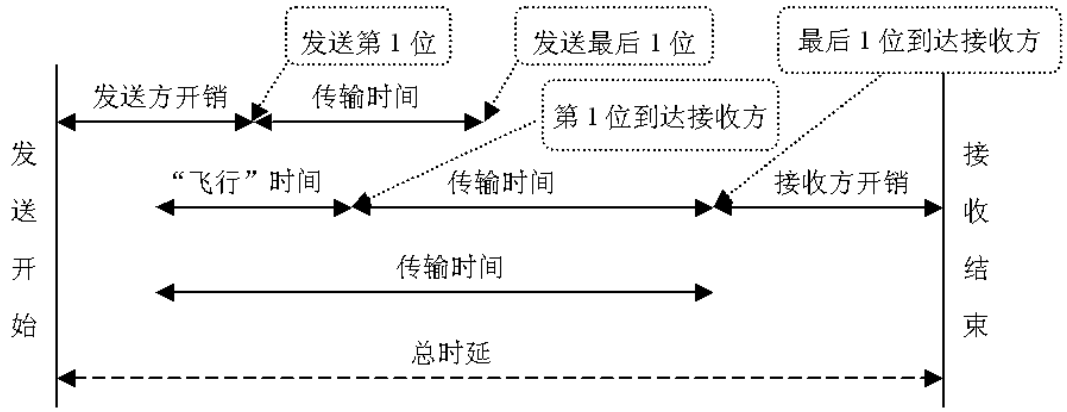


图 10-16 互连网络的传输性能参数间的关系

【例 10-1】假设一个网络的频宽为 10 兆位/秒，发送方开销和接收方开销分别等于 230 微秒和 270 微秒。如果两台机器相距 100 米，现在要发送一个 1000 字节的消息给另一台机器，试计算总时延。如果两台机器相距 1000 公里，那么总时延为多大？

解：光的速度为 299792.5 公里/秒，信号在导体中传递速度大约是光速的 50％，所以"飞行"时间可以计算出来了。那么相距 100 米时总时延为：

$$T=\text{发送方开销}+\text{“飞行”时间}+\frac{\text{消息长度}}{\text{频宽}}+\text{接收方开销}$$

$$=230\mu s+\frac{0.1Km}{0.5\times 299792.5Km/s}+\frac{1000\times 8\text{位}}{10\text{兆位/秒}}+270\mu s$$

$$=230\mu s+0.67\mu s+800\mu s+250\mu s$$

$$=1301\mu s$$

相距 1000 公里时的总时延为：

$$T=230\mu s+\frac{1000\times 10^6}{0.5\times 299792.5}\mu s+\frac{1000\times 8}{10}\mu s+270\mu s$$
$$=230\mu s+6671\mu s+800\mu s+270\mu s$$
$$=7971\mu s$$

10.5.3 互连函数

为了反映不同互连网络的连接特性，每种互连网络可用一组互连函数来描述。如果将互连网络的 N 个输入端和 N 个输出端分别用整数 0，1，…，N-1 来表示，则互连函数表示相互连接的输出端号和输入端号之间的一一对应关系。或者说，存在互连函数 f，在它的作用下，输入 i 应与输出 f(i)相连，其中 $0\le i\le N-1$。当互连网络用来实现处理器与处理器之间的数据变换时，互连函数也反映了网络输入数组与输出数组间对应的置换关系或称排列关系。所以互连函数有时也称为置换函数或排列函数。

表示互连函数通常用三种方法：函数表示法、输入输出对应表示法、图形表示法。

函数表示法用 x 表示输入端变量，用 f(x)表示互连函数。x 还常用几位二进制形式来表示，写成 $\mathbf{X_{n-1}X_{n-2}\cdots\cdots X_1X_0}$，互连函数则对应地表示为 $\mathbf{f(X_{n-1}X_{n-2}\cdots\cdots X_1X_0)}$。

输入输出对应表示法把互连函数表示为：$\begin{pmatrix}0 & 1 & \text{N-1}\\ f(0) & f(1) & f(N-1)\end{pmatrix}$这就是说 0 变换为 f(0)，1 变换为 f(1)，…，N-1 变换成 f(N-1)，f 是互连函数。例如，N＝8 均匀洗牌函数的这种表示形式为：$\begin{pmatrix}0&1&2&3&4&5&6&7\\0&2&4&6&1&3&5&7\end{pmatrix}$。

图形表示法是用输入输出的连接图表示关系。这种表示法较为直观，但有时显得较为繁琐，难以体现内在规律。因此一般结合函数表示法和输入输出对应表示法使用。

有一种特殊的互连函数，称为循环表示法，这种函数把互连函数 f(x)表示为：（$\mathbf{X_0X_1\cdots X_{j-1}X_j}$）,所谓"循环"表示，则代表对应关系为：$\mathbf{f(X_0)=X_1}$,$\mathbf{f(X_1)=X_2}$,…，$\mathbf{f(X_j)=X_0}$，其中 j+1 称为该循环的长度。

下面介绍常用的基本互连函数、它们的函数表达式和主要的特征。

1、立方体（cube）

立方体的每个顶点代表一个结点，结点的编号用二进制码（$C_2C_1C_0$）表示。如图 10-17 所示为 N＝8 的三维立方体结构。

立方体单级网络的互连函数实现二进制编号中第 k 位值不同的结点之间的连接。故三维的立方体单级网络有三种互连函数：$Cube_0$、$Cube_1$ 和 $Cube_2$，分别建立结点编号中 C_0 不同或 C_1 不同或 C_2 不同的结点之间的连结，其连接方式如图 10-18。一般情况下，一个 n 维立方体有 $N=2n$ 个结点，共有 n 种互连函数，分别由 n 位编号中的每一位的位求反来确定。即

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

其中，P_i 为入端标号的二进制代码第 i 位，且 $0\leqslant i\leqslant n-1$。当维数 $n>3$ 时，称为超立

方体(Hyper Cube)网络。对于 *n* 维立方体单级网络，要实现任意两个结点之间的连接，最多需使用 *n* 次不同的互连函数，因此 *n* 维立方体单级网络的最大距离为 *n*。

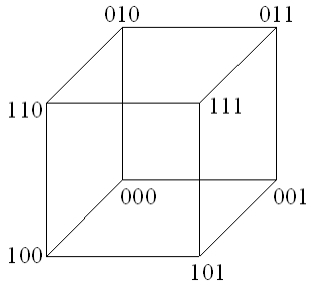


图 10-17　N＝8 的三维立方体结构

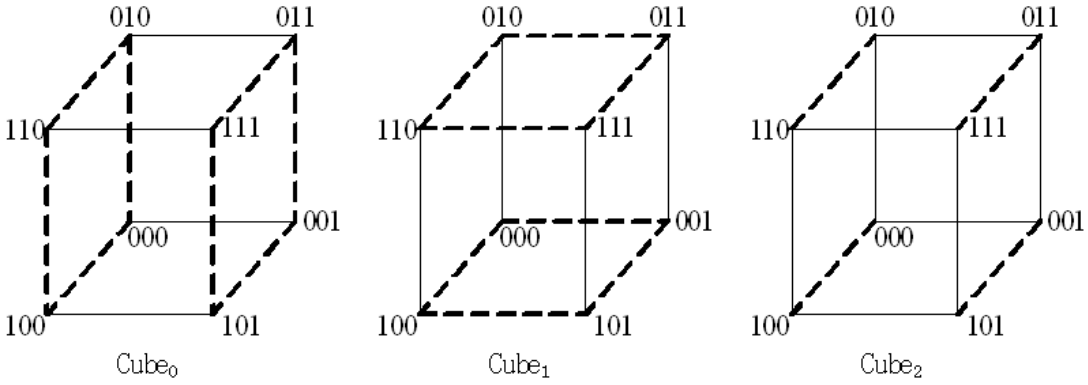


图 10-18 N＝8 的三维立方体三种互连方式

2、PM2I

PM2I 是加减 2i 的简称（plus-minus2i），如图 10-19 中给出了 PM2I 互连网络的部分连接图。

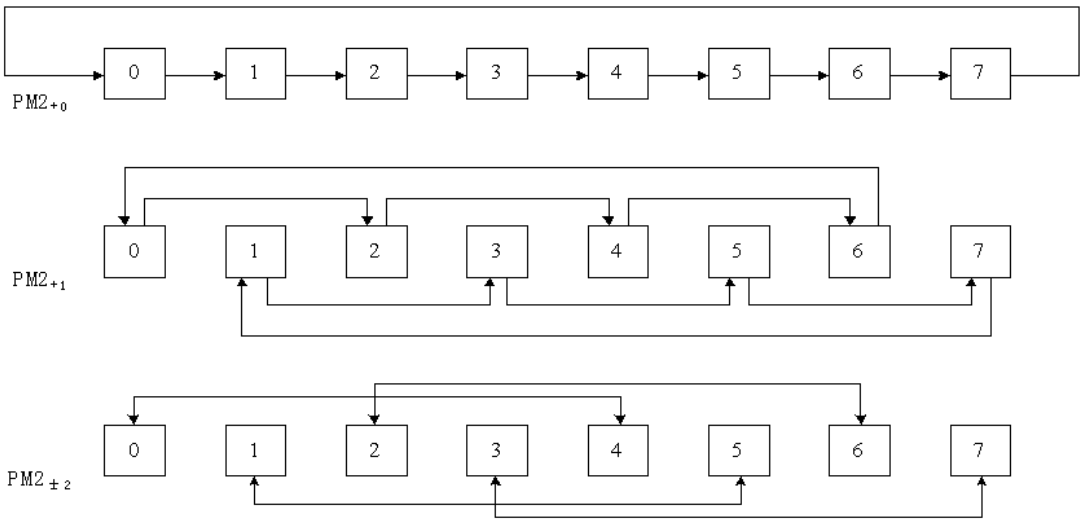


图 10-19　N＝8 的 PM2I 互连网络的部分连接图

PM2I 单级网络能实现 j 号结点与 j±2i mod N 号结点的直接相连，N 为处理器的个数，PM2I 单级网络的最大距离为［n/2］，n＝$\log_2$N。因此，它共有 2n 个互连函数，即

$PM2_{+i}$（j）＝$j+2^i$　mod N

$PM2_{-i}$（j）＝$j-2^i$　mod N

式中，0≤j≤N－1，0≤i≤n－1。如设 N＝8，则共有 2n＝6 个互连函数，各互连循环函数表示为：

$PM2_{+0}$：（01234567）

$PM2_{-0}$：（76543210）

$PM2_{+1}$：（0246）（1357）

$PM2_{-1}$：（6420）（7531）

$PM2_{\pm 2}$：（04）（15）（26）（37）

3、混洗交换(shuffle exchange)

混洗交换互连网络包含全混洗和交换两种互连函数,混洗交换又称为洗牌置换.

（1）全混洗(Perfect Shuffle)

这种连接规律是把全部按标号次序排列的处理器从当中分为数目相等的两半，然后洗牌达到理想的"全混"状态一样，这也是"混洗"这个名词的由来。正如人们在洗扑克牌时，先将整副牌分成两半，然后洗牌以达到最理想的状态一下，因此这种方法也称为均匀洗牌。全混洗的互连函数为σ（$P_{n-1}P_{n-2}\ldots P_1P_0$）＝ $P_{n-2}\ldots P_1P_0P_{n-1}$ 。

可见，全混洗是将输入端的二进制编码循环左移 1 位得到输出端二进制编码的方法，如图 10-20 为全混洗置换图。

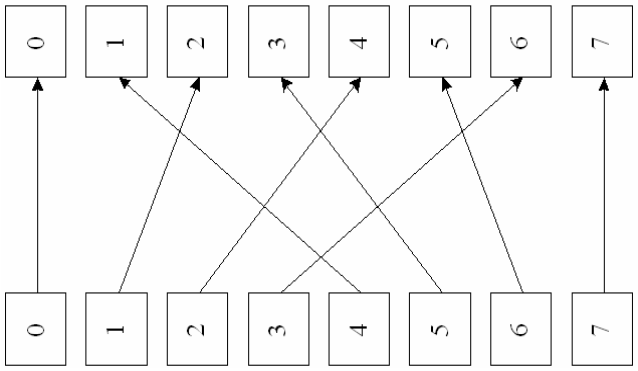


图 10-20　全混洗置换图

（2）交换（Exchange）

由于单一的全混洗互连网络不能实现二进制编号为全“0”和全“1”的结点与其他任何结点的连接，所以又增加了 $Cube_0$ 交换互连函数。同时采用了全混洗和交换的单级互连网络称为混洗交换单级互连网络。如图 10-21 所示为混洗交换单级互连网络图。

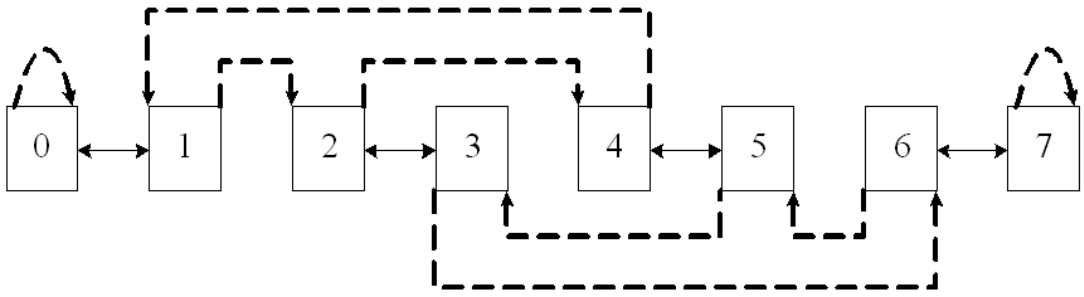


图 10-21 混洗交换单级互连网络图

4、蝶形(butterfly)

如图 10-22 是蝶形置换的图形表示。蝶形互连网络的互连函数为：Butterfly($P_{n-1}P_{n-2}\ldots P_1P_0$)= $P_0P_{n-2}\ldots P_1P_{n-1}$，它是将入端二进制编号的最高位和最低位互换位置即可得相应出端的二进制编号。

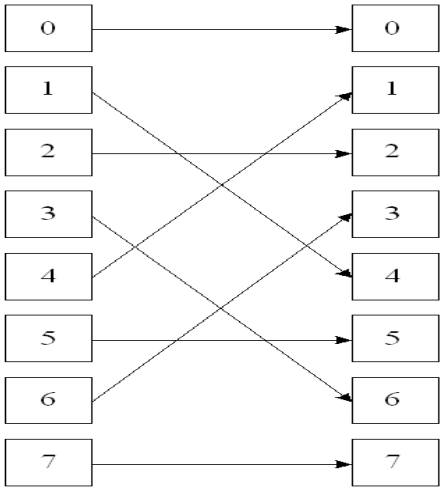


图 10-22 n＝8 蝶形置换的图形表示

5、恒等置换

相同编号的输入端与输出端一一对应互连所实现的置换即为恒等置换，其表达式为：

$$I(x_{n-1}\ x_{n-2}\ x_{n-3}\cdots\cdots\ x_1\ x_0)= x_{n-1}\ x_{n-2}\ x_{n-3}\cdots\cdots\ x_1\ x_0$$

即二进制编码完全相同，其中等式左边括号内的 $x_{n-1}\ x_{n-2}\ x_{n-3}\cdots\cdots\ x_1\ x_0$ 和等式右边的 $x_{n-1}\ x_{n-2}\ x_{n-3}\cdots\cdots\ x_1\ x_0$ 均为网络输入端和输出端的二进制地址编号。

6、交换置换

交换置换是将输出编号的二进制编号中第 0 位取反，得到的互连函数称为交换置换。其表达式为：

$$E(x_{n-1}\ x_{n-2}\ x_{n-3}\cdots\cdots\ x_1\ x_0)= x_{n-1}\ x_{n-2}\ x_{n-3}\cdots\cdots x_1\ \overline{x_0}$$

10.6 静态互连网络

静态互连网络是指在各结点间使用的直接链路，且一旦构成后就固定不变的网络。这种网络比较适合于构造通信模式可预测或可用静态连接实现的并行处理系统和分布计算机系统。集中式系统的多系统之间或分布式系统的多个计算机之间的固定连接通常采用静态拓扑结构，并工作于异步状态下，采用分组交换的方法进行通信。

10.6.1 静态互连网络结构

静态互连网络结构主要包含一维线性阵列、环和带弦环、循环移数网络、树形和胖树形、网格形与环形网格、超立方体和带环立方体、k 元 n 立方体等 7 种结构，下面分别做简单介绍。

1、线性阵列

线性阵列是一种一维的线性网络，其中 *N* 个结点用 *N*-1 条链路连成一行，如图 10-23 所示。首尾结点的度是 1，内部结点度为 2，端结点度为 1，直径为 *N*-1，等分宽度为 1，结构不对称。这种结构不同于总线结构，总线结构是通过时分切换使用多对结点时分进行通信，而线形阵列允许不同的结点对并发使用系统的不同通道。线性阵列是连接最简单的拓扑结构，当 *N* 很大时，通信效率很低。

2、环和带弦环

用一条附加链路将线性阵列的两个端结点连接在一起，就构成了环。每个结点的度都是相等的，都是 2。环可以单向工作，也可以双向工作。双向环有两个方向，当其中的一个单向环出现故障时，另一个环还可以继续工作。环是对称的，结点度为 2。双向环直径为[N/2]，单向环直径为 N。增加一条或两条附加链路，就可以得到结点度分别为 3 和 4 的带弦环。

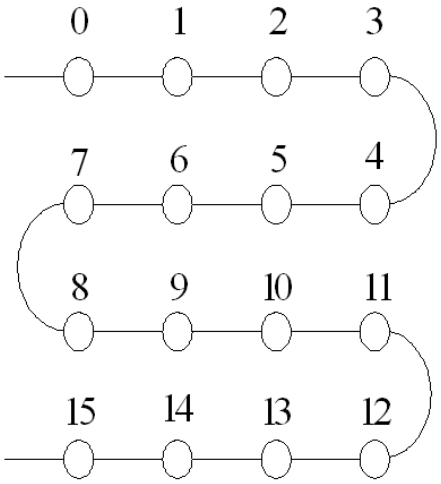


图 10-23 一维线形阵列连接结构

3、循环移数网络和全连接

循环移数网络是通过环形网络结构上增加“弦”的方法使直径减小的改进网络。其规则是将环上每个结点出发与距该结点距离为 2 的整数幂的结点之间都增加一条附加链路而构成的。全连接网络是环上任意两个结点间都有附加链路连接。这种全连接网络是一个对称的网络，当网络规模为 N 时，结点度为 N-1，网络直径为 1，链路数为 N(N-1)/2。

4、树形和星形

一棵 5 层 31 个结点的二叉树如图 10-24(a)所示。顶部的一个结点称为根，底部的 16 个结点称为叶了，其他的称为中间结点。除了叶子结点之外，每个结点都有两个孩子结点。一棵 k 层完全平衡的二叉树应有 N＝2k-1 个结点，最大结点度为 3，直径是 2(k-1)。从一个叶子结点到另外一个叶子结点的通信必须经过叶→根→叶的过程。由于各子树和叶子无横向的联系，因此这种结构的扩展性较好，只是直径较大。

树形另外一个应用是通信量不平衡问题。树形结构中叶子结点的通信量最小，而根结点的通信量最大。且随着层次的增加将呈现指数级增加，在所有的链路带宽相同的情况下，很容易造成网络阻塞。为了解决这个问题，即根结点瓶颈问题，1985 年 Leiserson 提出了胖树的概念。思想为通道带宽从叶结点往根结点上行方向逐渐增宽，越靠近根结点，链路带宽越大。

星形是树的一种特殊结构，是一种 2 层树，结点度为 N-1，网络直径为常数 2。图 10-24 (b)所示的是一个结点数为 9 的星形网络。

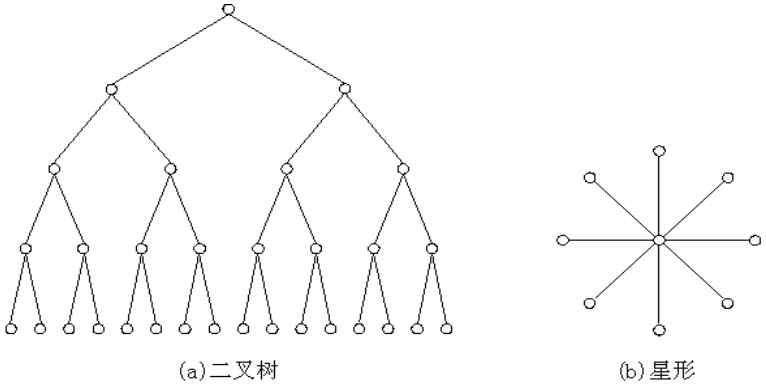


图 10-24 二叉树和星形网络连接结构

5、胖树形

二叉胖树结构如图 10-25 所示，胖树的链路数从叶结点往根结点上行方向逐渐增多。

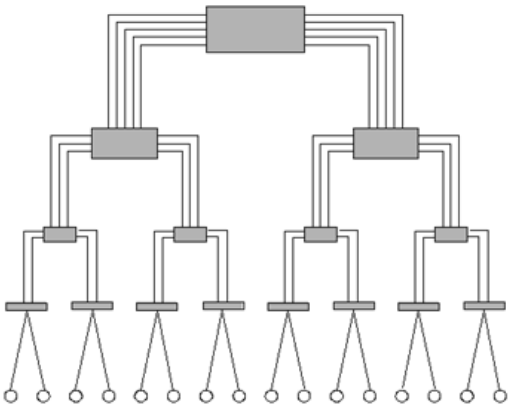


图 10-25　二叉胖树网络结构

互连网络结构还包含网格形、环网形、搏动式阵列、超立方体、带环立方体和 k 元 n-立方体网络等。搏动式阵列通常是为实现某种特定数据流算法而设计的一类多维流水线阵列结构。超立方体是一种二元 n 维立方体结构。带环立方体结构从超立方体结构改进而来，即将立方体的每个顶点结点用一个环形网络代替。k 元 n-立方体网络中的参数 k 是沿每个方向的结点数，n 是立方体的维数。这两个数与网络中结点数 N 的关系为 N＝kn。一般来说，低维 k 元 n-立方体称为环网，而高维二元 n-立方体则称为超立方体。

10.6.2 静态互连网络特性

大多数网络的结点度都小于 4，这是较为理想的。全连接网络和星型网络的结点度均太高，超立方体的结点度随 $\log_2 N$ 的增大而增大，当 N 值很大是，其结点度也很高。由于较高的结点度要求为结点提高较多的通信资源，因而在选择使用网络时，结点度不宜太高。静态连接网络特性如表 10-1 所示：

网络直径关系到消息传输时可能造成的时沿，一般要选择直径较小的网络。但是这存在两种例外情况，一是在一些情况下，人们可能更愿意使用平均距离来衡量一个网络可能造成的传输时沿，而不愿意用表示最大距离的直径，因为实际上可能只有极少量的通信在网络直径距离上进行的；另一种情况就是消息在网络中传输的机制，如果消息采用“虫蚀”方式（一种硬件寻址方式，它将包进一步分成更小的片，同一个包中所有的片不间断地以流水方式顺序地传送，只有头片知道包将发往何处）寻径，消息使用流水线机制传输，就像流水一样通过各结点、通信延迟几乎是固定不变的。但是，网络直径还是人们选择网络时经常要考虑的问题之一。

除了结点度和网络直径之外，链路数会影响网络的价格，等分宽带 B 将影响网络的带宽，对称性会影响网络的可扩展性和寻径效率。一般来讲，网络的总价格会随着结点度和链路数的增大而上升。

根据上述性能选择的简单规则，环形、网格、带环立方体和 k 元 n-立方体都具体一定的条件用来构造未来的大规模并行处理系统。

表 10-1　　静态连接网络特性

网络类型	结点度	网络直径	链路数	等分宽度	对称性	网络规模
线性阵列	2	$N-1$	$N-1$	1	非	N个结点
环形	2	$[N/2]$	N	2	是	N个结点
全连接	$N-1$	1	$N(N-1)/2$	$(N/2)^2$	是	N个结点
二叉树	3	$2(k-1)$	$N-1$	1	非	树高$k=\lceil \log_2 N \rceil$
星形	$N-1$	2	$N-1$	$[N/2]$	非	N个结点
2D网格	4	$2(n-1)$	$2N-2n$	n	非	$n\times n$网格，$N=n^2$
ILLIAC网	4	$n-1$	$2N$	$2n$	非	与$N=n^2$的带弦环等效
2D环网	4	$2\times[n/2]$	$2N$	$2n$	是	$n\times n$环网，$N=n^2$个结点
超立方体	n	n	$nN/2$	$N/2$	是	N个结点，$n=\log_2 N$(维数)
3–CCC	3	$2k-1+[k/2]$	$3N/2$	$N/(2k)$	是	$N=k\times 2^k$个结点环长 $k\geqslant 3$
k元n－立方体	$2n$	$n[k/2]$	nN	$2K^{n-1}$	是	$N=k^n$个结点

注：[　]表示取整，⌈　⌉表示上取整。

10.7　动态互连网络

动态连接网络可根据程序要求实现所需的通信模式。它不采用固定连接，而是沿着连接路径使用开关或仲裁器以提供动态连接特性。根据级间连接方式，动态连接网络有单级和多级两类。单级网络只有有限的几种连接，任意两个结点之间的信息传送可能须经过在单级网络中循环多次才能实现，故单级网络也称循环网络。多级网络由多个单级网络串联组合而成，以实现任意两个结点之间的连接。按照价格和性能增加的顺序，动态互连网络可分为总线互连方式、交叉开关互连方式、多级网络互连方式等。

10.7.1 动态互连网络的互连形式

1、总线互连方式的动态连接网络

总线互连方式，又称为公共介质，是实现动态连接最简单的一种结构形式。用一条公用系统总线将多个处理机、存储器模块和 I/O 部件通过各自的接口部件或是多个由 CPU、本地存储器和 I/O 部件所组成的计算机模块通过公共的接口部件互连起来。总线上的各模块以分时或多路转换方式通过总线在主设备与从设备之间传送信息。在多个请求情况下，总线仲裁逻辑每次只能将总线服务分配或重新分配给一个请求。如图 10-26 为总线连接的多处理机系统　。

总线上的各模块是通过争用或时分方式获得总线服务，因此总线也被称为争用总线或时分复用总线，与另外两种动态互连方式相比，总线系统价格较低，带宽较窄。但由于价格低的优势，总线的使用很广，有很多的工业标准和 IEEE 总线标准。

总线系统通常设置在印刷电路底板上，连接到总线上的其他设备往往通过总线插槽或电缆与总线相连。总线系统中需要解决的主要问题有总线仲裁、中断处理和 Cache 一致性等问题。

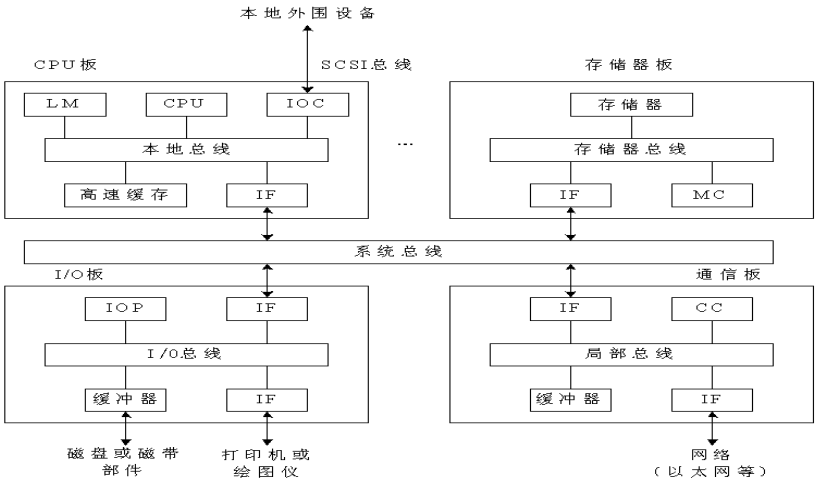


图 10-26 总线连接的多处理机系统

2、交叉开关互连方式的动态连接网络

交叉开关(crossbar)互连由一组纵横开关阵列组成，将横向的 p 个处理机 P 及 i 个 I/O 模块与纵向的 m 个存储器模块 M 连接起来，如图 10-27 所示。

行与列的交叉点上是由开关来控制其接通与否。交叉开关网络中的每一个开关只有两种状态："通"或"断"。如果 P_1 处理机要访问 M_3 存储模块，只要将 P_1 行线与 M_3 列线交叉点的开关置为"通"，即可实现 P_1 处理机和 M_3 存储模块的互连。这里的交叉点开关，实际上是一组受同一信号控制的开关，可能包括地址线、数据线及其必要的控制线的连接。

交叉开关网络的每一行中可以同时接通多个交叉点开关，阵列中的总线条数等于全部相连的模块数之和 m＋p＋i，且当 m≥p＋i 时，可以使每个处理机和 I/O 设备都能分到一套总线与某个存储器模块相连，从而大大加宽互连传输带宽和提高系统的效率。与总线互连中采用分时使用总线不同，交叉开关采用的是空间分配机制。

为了保证系统的正常工作，每条列线上只允许有一个开关处于接通状态，否则可能由于多个处理机接入同一个存储模块而形成冲突。但是反过来，如果系统设计中允许处理机访问多个存储模块，每一条行线上可能会有多个开关被接通。如果希望同时读写多个存储模块，必须采用空间上或时间上的分隔使用。

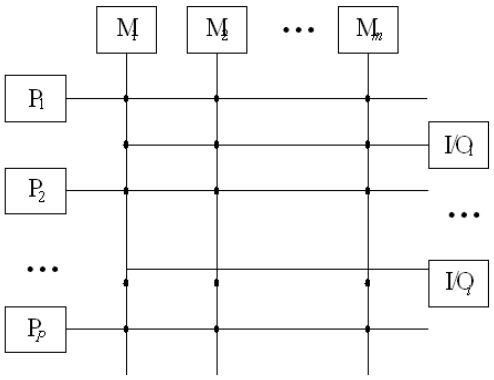


图 10-27 交叉开关互连方式

交叉开关网络是目前使用最为广泛的一种网络，其优点技术可以随工作进程随时改变开关的连接方式，以达到改变通信链路的目标；采用多级交叉开关网络虽然降低了硬件复杂性，但时延会随着网络级数的增加而增大。

3、多级网络互连方式的动态连接网络

多级网络互连是将多套单级互连网络通过开关模块串连扩展成多级互连网络（简称MIN）的方式。现在几乎所有的 MIMD 和 SIMD 计算机系统中都采用了多级互连网络结构，这样可以满足处理机之间灵活通信的需求。与单级网络相比，多级网络可以通过改变开关的控制方式灵活地实现各种连接，满足系统应用的需要 。如图 10-28 为由 $a\times b$ 开关模块和级间连接模式 ISC 构成的通用多级互连网络结构。

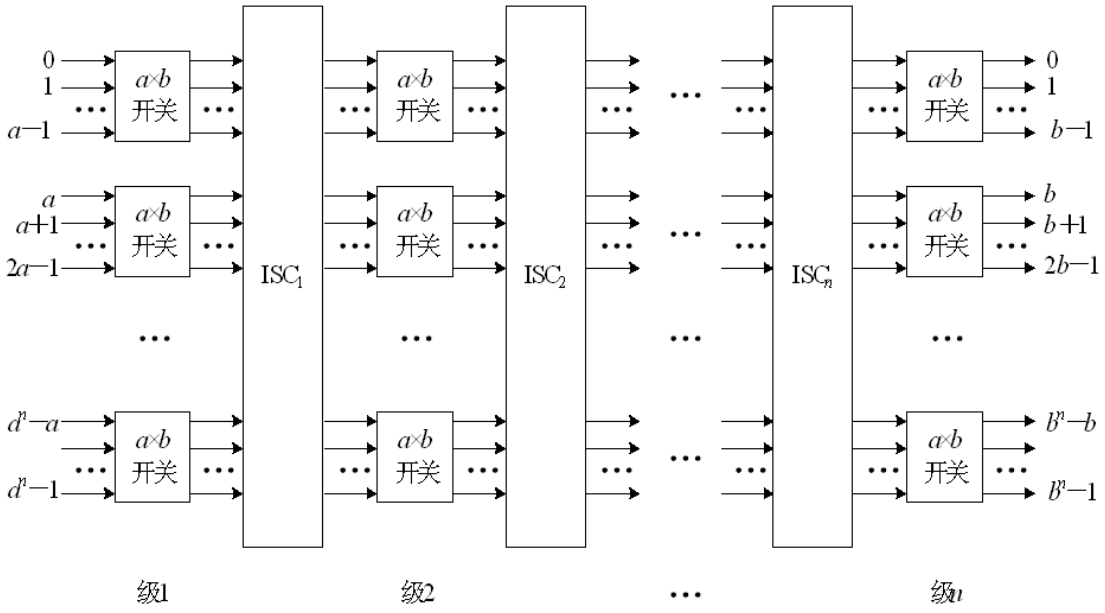


图 10-28 由 a×b 开关模块和级间连接模式 ISC 构成的通用多级互连网络结构

各种多级互连网络的区别就在于所用开关模块、级间连接（ISC）模式和控制方式上的不同，它们构成多级网络互连方式中的三个重要内容，从而形成各种不同的多级互连网络。

10.7.2 动态网络互连方式的比较

构成动态网络的总线、多级网络、交叉开关三种互连方式中，总线的造价最低，但其缺点是每台处理器可用的带宽较窄。总线所存在的另一个问题是容易产生故障，有些容错系统常采用双总线以防止系统产生简单的故障。由于交叉开关的硬件复杂性以 n^2 倍上升、所以其造价最为昂贵。但是交叉开关的带宽和路由性能最好。如果网络的规模较小，它就是一种理想的选择。多级网络则是两个极端之间的折衷。它的主要优点在于采用模块结构，因而可扩展性较好。然而，其时延随网络级数 $\log_n$ 的增加而上升。另外由于增加了连线和开关复杂性，价格也是一种限制因素。如表 10-2 汇总了构成动态网络的总线、多级网络、交叉开关的主要特性。

表 10-2　　动态网络互连方面的比较表

网络特性	总线系统	多级网络	交叉开关
单位数据传送的最小时延	恒定	$O(\log_n n)$	恒定
每台处理机的带宽	O(w/n)至 O(w)	O(w)至 O(nw)	O(w)至 O(nw)
连线复杂性	O(w)	$O(nw\log_n n)$	$O(n^2w)$
开关复杂性	O(n)	$O(n\log_n n)$	$O(n^2)$
连接特性和寻径性能	一次只能一对一	只要网络不阻塞，可实现某些置换和广播	全置换，一次一个
典型计算机	Symmetry S1,Encore Multimax	BBNTC－2000IBM RP3	Cray Y-MP/816Fujitsu VPP 500
评注	总线上假定有 n 台处理机；总线宽度为 w 位	n×nMIN 采用 k×k 开关，其线宽为 w 位	假定 n×n 交叉开关的线宽为 w 位

10.7.3 多级互连网络

多级互连网络是各种置换连接将各级开关连接在一起，形成的一个完整网络。从外部看，整个多级互连网络只有输入端和输出端，输入和输出端之间的功能也是某种特定的置换连接。

与单级网络相比，多级网络具有更多的优势。从直观上理解，多级网络可以通过改变开关的控制方式，灵活地变换所实现的连接，使其更能适应系统的需要。也正是因为网络所实现的环路具有可变性，所以多级互连网络是动态网络中非常重要的一员，在多处理机系统中得到了广泛的应用。

动态多级互连网络可分为阻塞网、可重排非阻塞网和非阻塞网三种类型。

在同时实现多对入端与出端之间的连接时，可能会引起开关和通信链路使用上的冲突。具有此类性质的互连网络称为阻塞网络。换句话来讲，是指一对以上输入端和输出端可同时实现互连的网络。在阻塞网络中，可能发生 2 个或 2 个以上的输入端对输出端的连接产生路径争用冲突。各种阻塞网络都能实现一些典型互连函数表示的连接，但不能实现任意的互连函数。一般来讲，阻塞网络所采用开关数量少，路径控制比较简单，所以所产生的时沿比较小，且能够实现某种互连函数所表示的置换。由于其简单，能够实现的互连函数也是比较有限。典型的阻塞网络有：Ω 网络、STARAN 网络、基准网络、间接二进制 n 方体网络、δ 网络和数据交换网络等。

可重排非阻塞网络是如果重新安排现有的连接，就可实现无阻塞的任意结点对的连接，从而满足一个新的结点对的连接请求。它在实现非阻塞网络连接时，往往需要通过重新安排开关的控制才能满足新的连接要求。典型网络有可重排 Close 网、Benes 二进制置换网络等。

非阻塞网络不必改变原来的开关状态就可满足任意输入端和输出端之间的连接请求。它同可重排非阻塞网络是不同的，可重排非阻塞网络要通过改变原来的开关状态来改变连接的路径，才能满足新的连接请求。多级 Close 网络就属于此类网络。

10.8 互连网络的消息传递机制

消息传递机制的研究在互连网络的研究中占有非常重要的地位。在多计算机系统中通过互连网络进行消息传递需要专门的硬件和软件的支持。在这一节将研究各种寻径方法，并分析它们的通信时延问题。我们还要引入虚拟通道的概念，考察消息传递网络产生死锁的各种情况，说明怎样用虚拟通道来避免死锁。为了实现无死锁的消息寻径，人们提出了确定的和自适应两种寻径算法。

10.8.1 消息寻径

1、消息格式

首先掌握消息、包和片的基本概念和相关联系。

消息（Message）：是在多计算机系统的处理结点之间传递包含数据和同步消息的信息包。它是一种逻辑单位，可由任意数量的包构成。包（Packet）：包的长度随协议不同而不同，它是信息传送的最小单位，64-512 位。片（Flit）：片的长度固定，一般为 8 位，它们的相互关系如图 10-29 所示。

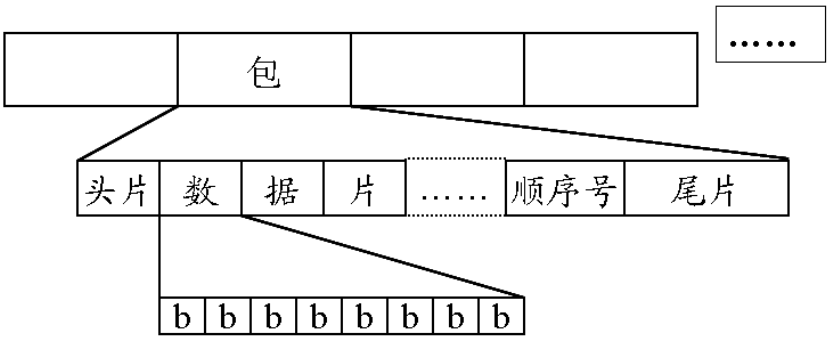


图 10-29 消息、包和片的相互关系图

消息寻径中的信息单位如图 10-30。消息是结点间通信的逻辑单位，它常常由任意数目的长度固定的包组成，因此它的长度是可变的。

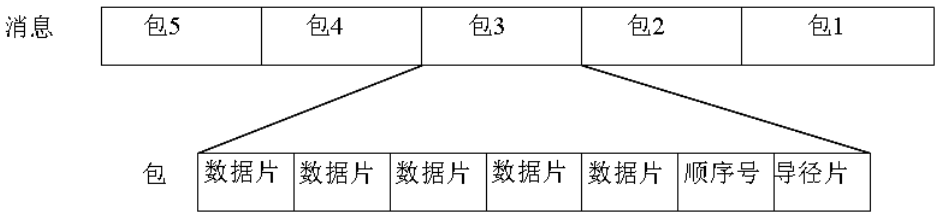


图 10-30 消息的组织方式

包是包含寻径目的地址的基本单位。由于不同的包可能异步地到达目的结点，因此每个包需要一个序号，以便把传递的消息重新装配起来。可以进一步把包分成一些固定长度的数据片。寻径信息（目的地址）和序号形成头片，其余的片是数据片。

在采用存储转发寻径方式的多计算机系统中，包是信息传送的最小单位。在采用虫蚀寻径网络的多计算机中，包可以进一步分成片。片的长度往往受网络大小的影响，256 个结点的网络需要片长为 8 位，即一个由 256 个结点组成的网络，常用 8 位作为一个片的长度。包的长度取决于寻径方式和网络的实现方法。典型的包的长度为 64－512 位。包中顺序号通常要占用 1－2 个片，这取决于系统在组织消息时所采用的策略和消息的长度。如果消息长度控制在 256 个包以内，那么只有 1 个片就足够了，反之就需要两个片。

包和片的大小还与通道频宽、寻径器设计以及网络流量密度等有关。

2、四种寻径方式

消息寻径方式可以分为两大类：线路交换和包交换。包交换是当前数据传输中效率最高，也是使用和研究得最多的方式。其中包交换又包括：存储转发寻径、虚拟直通寻径和虫蚀寻径等。下面逐一进行讨论。

（1）线路交换(circuit switch)

线路交换是一种以建立从源结点到目的结点间的直达物理通路为信息传输基础的交换方式。即在传递一个消息之前，先建立一条从源结点到目的结点的物理通路，然后再传递消息。如图 10-31 所示，图中的每一个带阴影的片是建立一个连接通路所需要的时间，经过 4 个时间片，打通了 4 个结点之间的通路，建立起到底目的结点之间的通道。传输开始后，4 个结点将无迟滞地传送全部信息。

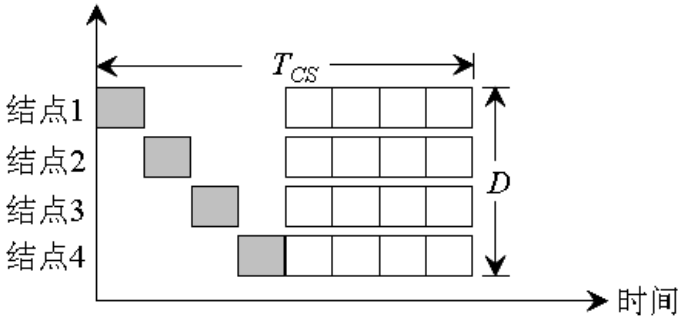


图 10-31 线路交换寻径示意图

线路交换寻经的传输时延用公式表示为 T = (Lt / B)×D + L / B，式中的 Lt 是在一个结点中建立路径所需要的小信息包长度，B 为带宽，D 为通路上总的结点数，L 为信息包的长度。

在并行计算机中的频繁的小信息包通信的这种方式下，由于在传递一个消息之前，需要频繁地建立从源结点到目的结点的物理通路，开销将会很大，这种寻径方式与以下的几种包交换(packet switch)的寻径方式相比这是一个很大的缺点。包交换的寻径方式以其较高的传输带宽和较低的平均传输时延，更适合于具有动态和突发特性的 MPP 数据传送。相应地，如果是短信息包传输时采用此种方式，系统效率不高，因为建立路径的开销占信息传输时间的比例就会上升。

（2）存储转发寻径(store and forward)

在存储转发网络中包是信息流的基本单位，每个结点有一个包缓冲区。包从源结点经过一系列中间结点到达目的结点。即先按寻径信息传送到最近的结点存储，再根据寻径信息从此结点出发传至下一结点，逐段传送—存储，直至到达目的结点。

当一个包到达一个中间结点时，它首先被存入缓冲区。当所要求的输出通道和接收结点的包缓冲区可使用时，然后再将它传送给下一个结点。存储转发方式可以根据网络各结点间的忙闲情况调整信息包的发送路径和时间，比较灵活，但是时延长是其最致命的缺点，特别是当中间结点数量大时更加突出。如图 10-32 所示，存储转发网络的时延与源和目的地之间的距离（段数）成正比。第一代多计算机系统采用这种寻径方式。

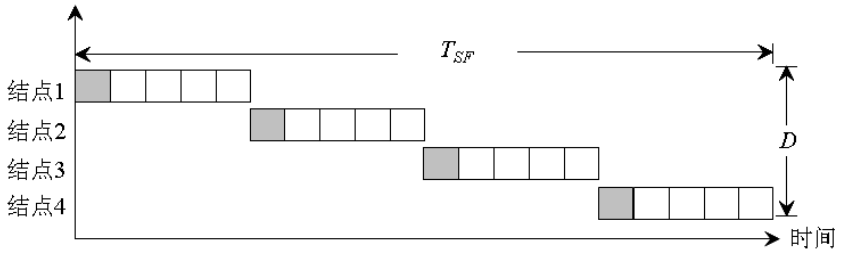


图 10-32 存储转发寻径示意图

存储转发的时延用公式表示为 T＝(L/B)×D+L/B=(D+1)×L/B。由图和公式可以看到，存储转发寻径有两个很大的缺点：一是包缓冲区大，不利于 VLSI 的实现；二是时延大，与结点距离成正比。

（3）虚拟直通(virtual cut through)

目前有一些多计算机系统采用的是虚拟直通的寻径方式。虚拟直通的寻径方式的思想是，为了减少时延，没有必要等到整个消息全部缓冲后再作路由选择，只要接收到用作寻径

的消息头部即可判断。其通信时延用公式表示为 T = (Lh / B)×D + L / B = (Lh×D + L) / B。式中 Lh 是消息中用于寻径那部分信息的长度。一般来说，L 的长度远较 Lh×D 更大，所以公式可以近似为 T＝L/B，可以看到此时通信时延与结点数无关，这相对于存储转发的寻径方式来说是一个非常大的改进，所以虚拟直通寻径的延时接近于线路交换。

然而，当出现寻径阻塞时，虚拟直通方式只有将整个消息全部存储在寻径结点中，直到寻径通道不阻塞时才能将消息发出，这就需要每个寻径结点都有足够的缓冲区来存储可能出现的最大的信息包，在这一点上，虚拟直通方式与存储转发的寻径方式是一样的，同样不利于 VLSI 的实现。因此，虚拟直通方式在最坏的情况下与存储转发方式的通信时延是一样的。由此出现了下面将要讨论的新的寻径方式虫蚀寻径方式，它改进了以上提到的缺点。

（4）虫蚀寻径(wormhole)

新型的多计算机系统很多采用的是虫蚀寻径方式，把包进一步分成更小的片，且与结点相连的硬件寻径器中有片缓冲区。而消息从源结点传送到目的结点传输就要经过一系列寻径器，如图 10-33 所示。

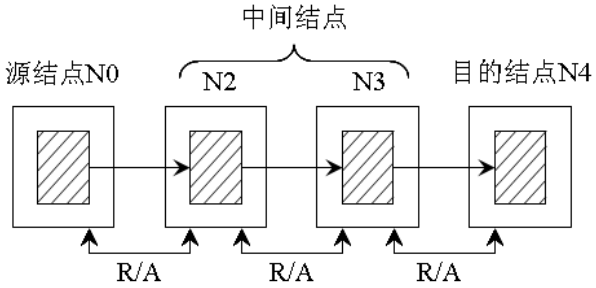


图 10-33 源结点传送到目的结点传输中的寻径器

同一个包中所有的片象不可分离的同伴一样以流水方式顺序地传送。包可以看作是一条蠕虫（一列火车），由蠕虫头（火车头、即头片）和被牵引的蠕虫的节（车厢，即数据片）组成。头中包中含有目的地址，因此只有头片知道包将发往何处。所有的数据片必须跟着头片。不同的包可以交替地传送，但不同包的片不能交叉，否则它们可能被送到错误的目的地。虫蚀寻径传输过程如图 10-34 所示。

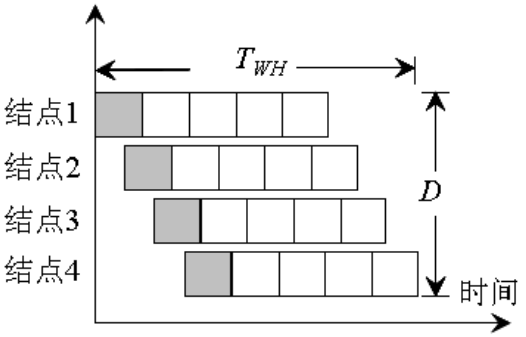


图 10-34 虫蚀寻径传输过程示意图

用头片直接开辟一条以输入链路到输出链路的路径的方法来进行操作。每个消息中的片以流水方式在网络中向前“蠕动”。每个片相当于虫的一个节，"蠕动"是以节为单位顺序地向前爬行。传输过程就像一条蠕虫一样往前爬行，因此这种方式称为虫蚀寻径。

当消息的头片到达一个结点 A 的寻径器后，寻径器根据头片的寻径消息立即做出路由选择：如果所选择的通道空闭且所选择的结点 B 的片缓冲区可用，那么这个头片就不必等待，直接通过结点 A 传向下一个结点 B。随后的其它数据片跟着相应地向前“蠕动”一步。当消息的尾片向前“蠕动”一步之后，它刚才所占有的结点就被放弃了。如果所选择的通道

忙或所选择的结点的片缓冲区不可用时，那么这个头片就必须在该结点的片缓冲区中等待，直到上述两者都可用时为止，其它数据片也在原来的结点上等待。此时，被阻塞的消息不从网络中移去，片也不放弃它所占有的结点和通道。

全部通信的延时公式为：$T = Tf \times D + L / B = (Lf / B) \times D + L / B = (Lf \times D + L) / B$。式中，Lf 是片的长度，Tf 是一个片经过一个结点所需的时间。通常，Lf×D 总是远小于 L，所以可以认为：$T = L / B$，即延时长短与结点数的多少基本无关。

可以看出，虫蚀寻径有以下的优点：

第一，每个结点的缓冲区较小，易于 VLSI 实现。

第二，较低的网络传输时延。所有的片以流水方式向前传输，采用了时间并行性。而存储转发方式的消息包整个地从一个结点"跳"到另一个结点，通道的使用是串行的，所以它的传输时延基本上正比于消息在网络中传输的距离。虫蚀寻径方式的网络传输时延正比于消息包的长度，传输距离对它的影响很小。

第三，通道共享性好，利用率高，对通道的预约和释放是结合在一起的一个完整的过程，有一段新的通道后将立即放弃用过的一段旧通道。

第四，易于实现选播和广播通信方式。允许寻径器复制消息包的片并把它们从其多个输出通道输出。

然而虫蚀寻径方式也有缺点，当消息的一个片被阻塞时，整个消息的所有片都将被阻塞在所在结点，占用了结点资源，因此需要采用虚拟通道的方式来避免由此引起一连串的阻塞。

虫蚀寻径方式也可以分为无缓冲和有缓冲两类，区别在于缓冲的大小。缓冲大有利于性能的提高，但会增加结点的复杂度。IBM SP2 采用的寻径方式就是带缓冲的虫蚀寻径方式，它采用共享的存储区来对输入/输出消息进行缓冲。

10.8.2 死锁和虚拟通道

虫蚀寻径多计算机网络的通信通道实际上由许多源和目的对共享。也就是说有多对通信对象使用同一条物理通道，而各结点之间建立起的连接只是一个逻辑通道，即一个虚拟的通道，图 10-35 形象地表示了物理与虚拟通道之间的关系。即从共享物理通道可以引出虚拟通道的概念。这一节我们将介绍这一概念并讨论它在避免死锁方面的应用。

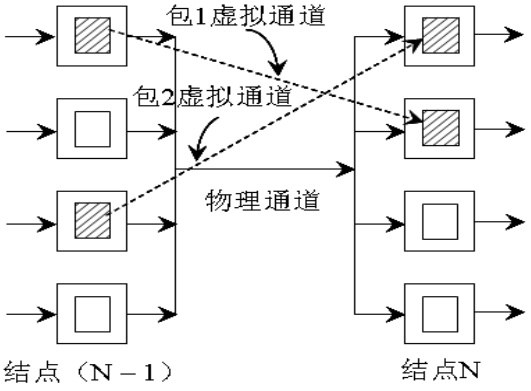


图 10-35 物理与虚拟通道之间的关系图

1、虚拟通道

虚拟通道是两个结点间的逻辑链，它由源结点的片缓冲区、结点间的物理通道以及接收结点的片缓冲区组成。下图 10-36 说明了四条虚拟通道共享一条物理通道的概念。

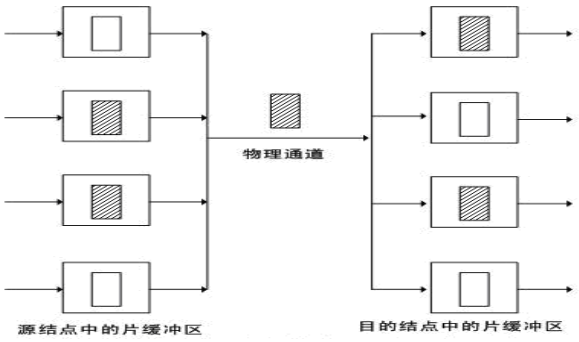


图 10-36 四条虚拟通道共享一条物理通道

源结点和接收结点各有 4 个片缓冲区。当物理通道分配给某对缓冲区时，这一对的源缓冲区和接收缓冲区形成了一条虚拟通道。换句话说，物理通道由所有的虚拟通道分时地共享。除了有关的缓冲区和通道之外，还必须用某些通道状态（如 R/A 信号）来表示不同的虚拟通道。源缓冲区存放等待使用通道的片，如上图中有两对片缓冲区已经分配给了两个包使用（用阴影部分标出），也就是说在一条物理通道上，建立了两个虚拟通道。接收缓冲区存放由通道刚刚传送过来的片。通道（电缆或光纤）是它们之间的通信媒介。

2、死锁的产生和避免

缓冲区或通道上的循环等待会引起死锁，如图 10-37 所示。它可以分两种情况，一种是缓冲区死锁，另外一种是通道死锁。

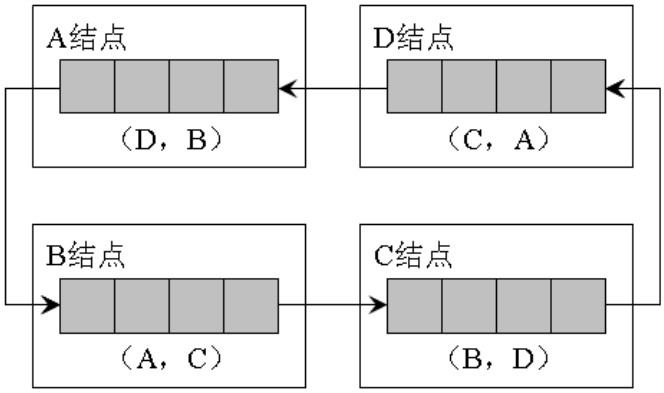


图 10-37 存储转发网络中所产生的缓冲区死锁

四个消息 A、B、C、D 包占用了四个结点的四个缓冲区，各自的消息包分别向 C、D、B、A 结点发送消息。开始时结点的缓冲区都是空的，所以 4 个结点都沿逆时针方向向下一个结点发生数据包。如图中所示的时刻，B 结点中存放是 A 结点要发送给 C 结点的数据包，记作（A，C）；C 结点中存放是（B，D）；D 结点中存放是（A，C）；A 结点中存放是（D、B）。此时，4 个缓冲区都已充满，按照握手信号的规定，这时 4 个结点都是向前一个结点发出禁止传输的信号。于是形成缓冲区的相关环，导致循环等待。除非扬弃某个消息包或者某个包的寻径出错，否则死锁不会解除。其死锁时示意图如 10-38 所示：

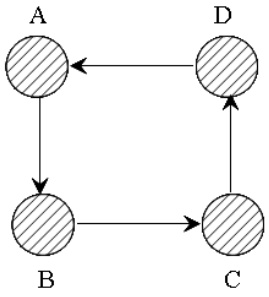


图 10-38 死锁时示意图

那么如何避免死锁呢？

由于死锁是循环等待造成的，因此可以用破坏循环的方法来打破死锁。解决死锁问题的

方法是增加虚拟通道，根据虚拟通道的含义，如果在结点上增加一个缓冲区，就相当于多了一条虚拟通道，图 10-39 为利用虚拟通道将通道相关图中的环路变成螺旋线来避免死锁的方法。

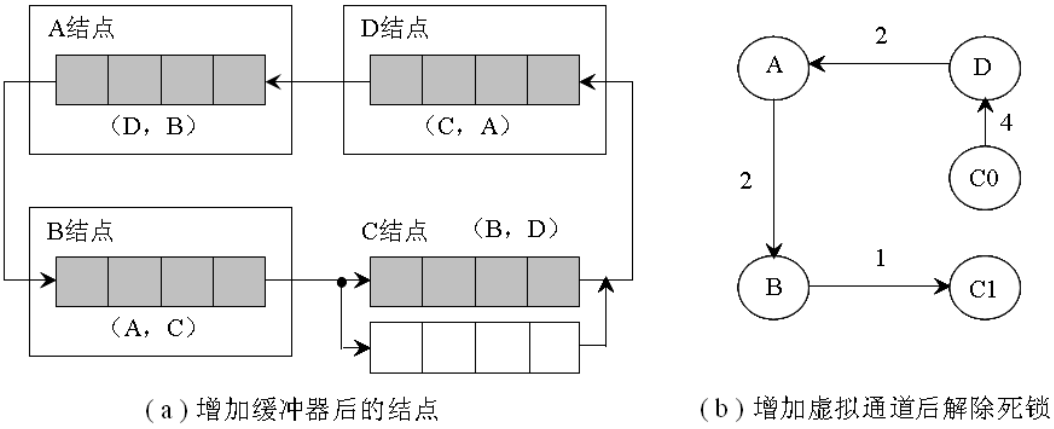


图 10-39 存储转发方式增加虚拟通道规避死锁示意图

下图 10-40 所示为四个通道之间因循环等待而出现了死锁。图中结点表示通道，带方向的箭头表示通道之间的依赖关系。图 10-40 a 表示为 4 个通道之间因循环等待而出现了死锁；图 10-40 b 表示为通道相关图，是一个环路。

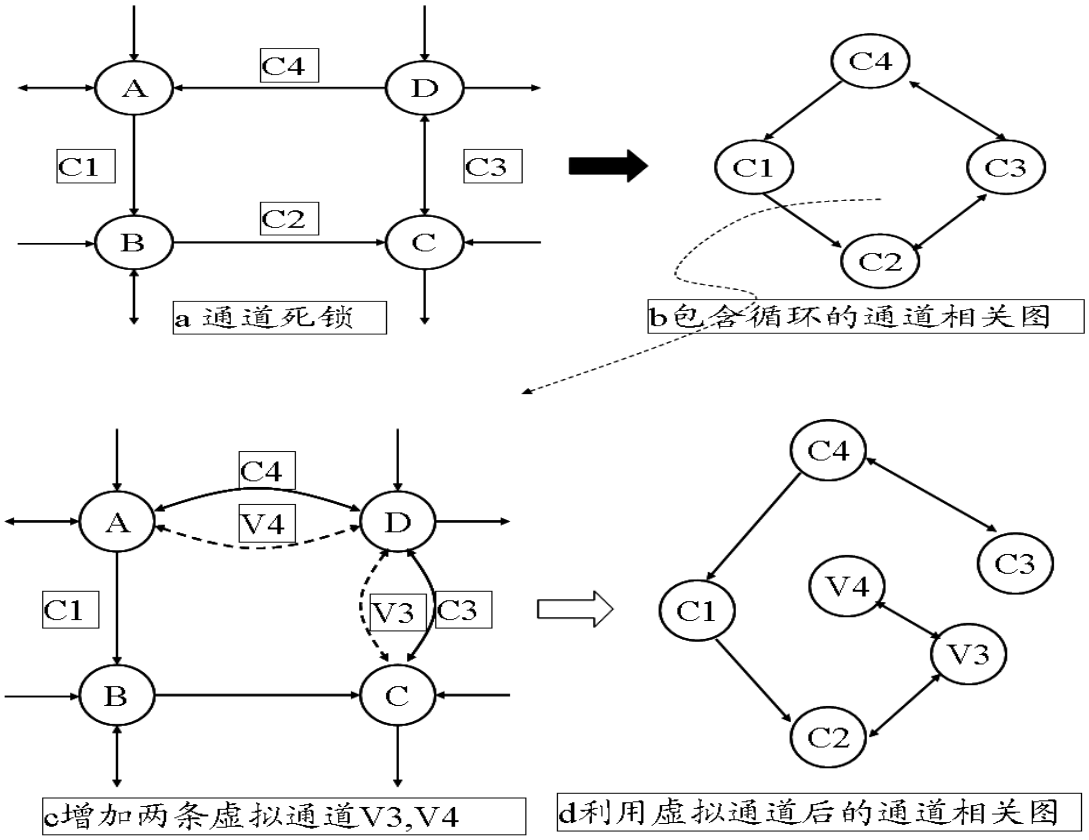


图 10-40 利用虚拟通道避免死锁

利用虚拟通道方法可以避免死锁。通过增加两条虚拟通道 V3 和 V4，如图 10-40c 所示，可以打破死锁循环，增加虚拟通道 V3 和 V4 可得到一张修改后的通道相关图如图 10-40d 所示，在使用通道 C2 之后不再使用 C3 和 C4。

通道多路复用可在片一级进行，如果包长度足够的话，那么也可以在包一级进行。

虚拟通道可以用单向通道或者双向通道实现。把两条单向通道组合在一起可以构成一条双向通道，这不仅增加了利用率而且还可使通道的频宽加倍。然而，双向通道中的仲裁要复杂一点。用双向通道互连的相邻结点需要专用的仲裁线，用它来控制信息流的方向。

实际上，和单向通道相比较，双向通道由于要做方向仲裁，因而增加了延迟，又由于控制复杂，因而还增加了成本。如果网络的流量不大，则双向通道的效率比较高。虚拟通道可能会使每个请求可用的有效通道频宽降低。确定虚拟通道数目时，需要对网络吞吐量和通信时延折衰考虑。实现数目很大的虚拟通道需要用高速的多路选择开关。

10.8.3 单播方式下的寻径

单播方式是多处理机系统中最常见的一种工作方式，其基本工作特点是实现点对点的通信。在对目的结点进行寻径的过程中，最重要的问题就是尽一切可能避免出现死锁。虽然在前面介绍了采用增加虚拟通道的方法来避免死锁，但是从更根本上看，死锁的形成在于结点之间相处循环的包冲突。在多处理机系统中，由于各处理机是以异步方式工作的，在错综复杂的通信需求中有极大的可能造成某结点的包冲突，一旦出现包冲突，并且同时满足循环堵塞的条件，那么死锁就会产生。所有必需有一套解决包冲突的方法，以避免出现更严重的冲突情况。

即当两个或更多的包在某个结点为竞争缓冲区或通道资源而发生冲突时，必须确定如何解决冲突的策略。这一节将考察各种不会引起拥挤或死锁现象的控制网络流量的策略。下面将以这些策略为基础，讨论为一对一通信设计的确定寻径算法和自适应寻径算法。

1、包冲突及其解决策略

通道流水线上的两个相邻结点之间要传送片时，必须具备以下三个条件。

条件 1：源缓冲区已存有该片；

条件 2：通道已分配好；

条件 3：接收缓冲区准备接收该片。

但是当两个包到达同一个结点时，它们可能会请求用同一个接收缓冲区或者要用同一个输出通道。即当一个结点同时面对两个源结点时，就可能出现两个包同时争用一个缓冲器的现象，这就发生了包冲突（又称为包阻塞）。

因此必须对两个问题做出仲裁：第一是把通道分配给哪个包？第二是如何处理没有分配到通道的包？为了解决上述问题，解决包冲突的结果有四种方法，如图 10-41 所示为解决包冲突的四种策略。

（1）虚拟直通寻径缓冲法

Kermani 和 Kleinrock(1979)提出了一种虚拟直通寻径(virtual cut-through routing)缓冲(buffering)方法。在冲突结点中设立一个包缓冲器，足以将一个结点的整个包接收到本结点，使本结点与前一个被拒收的结点间建立一个直通的途径，但是这个途径是虚拟的。而已分配通道的包仍按正常的工作途径获得通道。如图 10-41（a）所示，包 2 被暂时存放在一个包缓冲区。当通道可以使用时再传送包 2。

这种缓冲方法的好处是不会浪费已经分配的资源，但是需要一个能存放整个包的缓冲区。此外，通信路径上的包缓冲区不应形成循环。由于包缓冲区不可能做在寻径器芯片上，因此要用本地存储器作为包缓冲区，这会引起相当大的存储延迟。虚拟直通方法是存储转发和虫蚀两种寻径方法的折衷。当不发生冲突时，就如同虫蚀寻径方法一样工作。在最坏情况下，效果则与存储转发寻径方法相同。

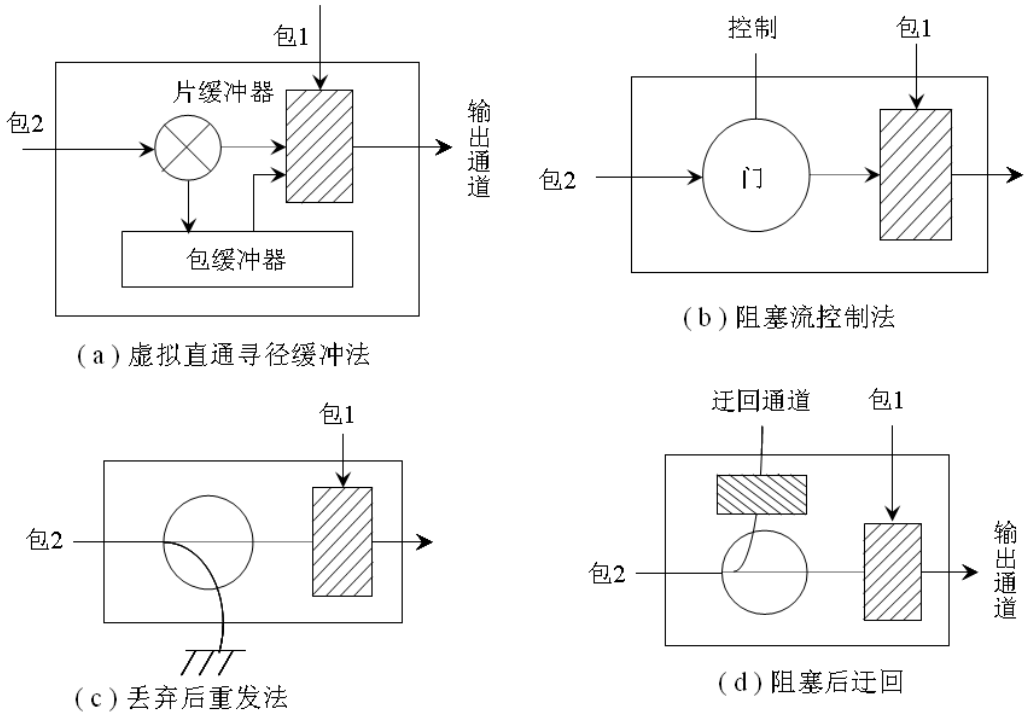


图 10-41 解决包冲突的四种策略

（2）堵塞流控制法

如图 10-41（b）所示，对于拒绝的包采用简单的阻断方式，纯粹的虫蚀寻径在出现冲突时就采用阻塞(blocking)策略。即图中第 2 个包被阻塞不再前进，但是并没有被放弃。它直到通路打开为止。

（3）丢弃后重发法

如图 10-41（c）所示的是丢弃(discard)策略，它简单地把被阻塞的包扬弃掉。即当出现冲突时，对于没有分配通道的包采取丢弃的方法，在通道允许后要求前结点重发。

（4）堵塞后迂回法

又称为绕道(detour)，如图 10-41（d）所示。它也是一种智能方法，当一个包被阻塞时，包被送到一条绕行的通道，使其有可能通过别的结点到底最终的目的结点。

阻塞策略的实现成本低，但可能出现分配给阻塞包的资源空闲的情况。丢弃策略可能会出现资源严重浪费，并且需要包重新发送和回答，否则被扬弃的包也许会丢失。现在已很少使用这种策略，因为包的输送率不稳定。BBN Butterfly 网络采用了这种扬弃策略。绕道寻径为包寻径提供了更大的灵活性。然而，绕道可能要用更多的通道资源才能到达目的地。而且绕行的包可能进入一个活锁（livelock）循环，这将浪费网络资源。Connection Machine 和 Denelcor HEP 采用了这种绕道策略。实际上，某些多计算机网络综合了以上某些流控制策略的优点，采用混合策略。、

2、确定寻径和自适应寻径

寻径可以分为确定和自适应两类。采用确定寻径时，通信路径完全由源和目的地址确定。换句话说，寻找的路径是预先唯一确定的，与网络的状况无关。自适应寻径与网络的状况有关，可能会有几条路径。这两种寻径都需要无死锁算法。所谓“自适应”的本意是指 2 个或 2 个以上消息包在同一结点中发生冲突时，网络具有自动选择迂回路径的一种能力。采用自适应寻径要特别注意避免死锁。虚拟通道的概念使实现自适应寻径更经济和更灵活。

10.8.4 广播方式下的寻径

多计算机网络中会出现四种通信模式：

（1）单播（unicast）模式对应于一对一的通信情况，即一个源结点发送消息到一个目的结点。

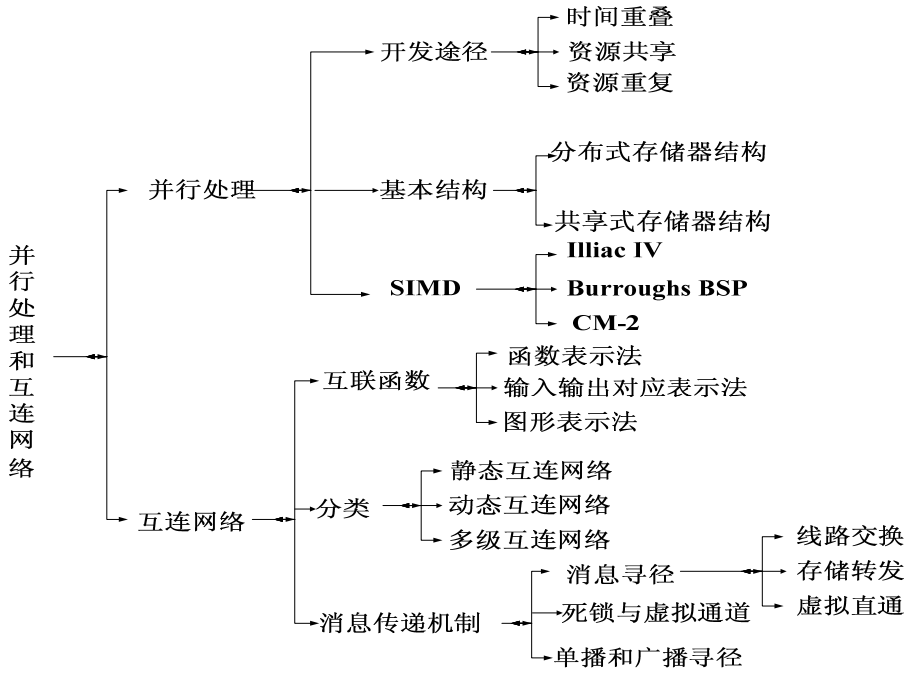
（2）选播（multicast）模式对应于一到多的通信情况，即一个源结点发送同一个消息到多个目的结点。

（3）广播（broadcast）模式对应于一到全体的通信情况，即一个源结点发送同一个消息到全部结点。

（4）会议（conference）模式对应于多到多的通信情况。

通道流量（channel traffic）和通信时延（communication latency）是描述效率常用的两个参数。通道流量可用传输有关消息所使用的通道数来表示。通信时延则用包的最长传输时间来表示。

关　联



习 题

10.1 解释下列概念

并行处理 时间重叠 资源重叠 资源共享 SIMD 并行处理机 共享存储器结构 分布式存储器结构 互联网络 互联函数 静态互联函数 动态互联函数 结点度数 网络直径 网络等分宽度 交叉开关 阻塞网络 非阻塞网络 存储转发寻径 虫蚀寻径 虚拟通道 自适应寻径

10.2 请从运算方式、体系结构方面比较 SIMD 与向量计算机的主要区别。

10.3 连续计算模型有何特点？为什么 SIMD 计算机比较适合此类计算？

10.4 在 Illiac IV 上实现 8×8 矩阵乘法，要求 J 和 K 的循环并行完成，数据在存储器中不准重复存放，请设计能使 64 个处理单元全并行工作的算法。

10.5 设有一个向量 A=（A_0，A_1，…，A_{15}），要计算其累加和 S=$\sum_{i=0}^{15} A_i$。在 SISD 计算机上用 Fortran 语言表示为：

```
      s=0.0
      do 10 I=0,15
   10 s=s+A(i)
```

这是一个串行程序。在 SISD 计算机上，它要用 16 次加法时间。如果在阵列机上采用递归相加算法，则只需要 $\log_2 16=4$ 次加法时间就够了。首先，原始数据 A(i),0≤ i ≤15，存放在 16 个 PEM 的 a 单元中，请写出阵列处理机上用成对递归相加的算法求和步骤。

10.6 BSP 计算机是一台由 16 个 PE 和 17 个共享存储器模块构成的 SIMD 计算机。试证明：如果跳数不是 17 的倍数，那么对任意长度向量的访问都不会出现访问存储器冲突。

10.7 设 16 个处理器的编号为 0、1、…、15，采用单级互联网络，当互联函数分别为

（1）$Cube_3$ （2）$PM2_{+3}$ （3）$PM2_{-0}$ （4）Shuffle

时，第 13 号处理器与哪一个处理器相连？

10.8 设编号分别是 0，1，2，…，F 的 16 个处理器之间，要实现下列结点对之间的通信：（B，1），（8，2），（7，D），（6，C），（E，4），（A，0），（9，3），（5，F）。试选择所用的互联网络类型、控制方式并画出该互联网络的拓扑结构和各级交换开关的状态图。

10.9 在一个 16×16 的Ω网络中，如果开关的控制信号为 1100，那么 9 号结点被连接到哪个结点上？

10.10 假定 8×8 的矩阵 A=(a_{ij})顺序存放在存储单元的 64 个单元中，用什么样的单级互联网络可实现对该矩阵的转置变换？总共需要传送多少步？

10.11 对于采用级控制方式的三级立方体网络，当第 i 级开关（0≤ i ≤2）为直送状态时，不能实现哪些结点之间的通信？为什么？当第 i 级开关为交叉状态时，不能实现哪些结点之间的通信？

10.12 Ω网络中每一个开关都是单元控制的，其连接情况有终端地址决定。请

（1）说明开关控制也可以有 T=S⊕D 来担任。

（2）给出实现广播连接的算法。

10.13 什么是死锁？发生死锁的原因有哪些？死锁与阻塞有什么区别？

第 11 章　多处理机与机群系统

【内容摘要】

本章讲述多处理机的结构和特点、机群系统以及多处理机的性能分析等内容。主要掌握互连网络的连接方式及其特点和结构，掌握典型的寻径算法，并对典型实例有一定的了解。

【学习要点】

- 多处理机系统特点与分类
- 多处理机系统的 cache 一致性问题
- 多处理机软件和并行算法和
- 机群系统

11.1　多处理机系统特点与分类

多处理机具有两台以上处理机，每台处理机可以带有本地 Cache、本地存储器、甚至 I/O 设备，它们都能独立执行各自的程序。多台处理机之间通过总线、纵横交叉开关、多级互连网络或高速的商品化网络实现互连。多处理机可以通过共享存储器，也可以通过消息传送系统来实现处理机间的通信。多台处理机在操作系统的控制下，实现资源的统一分配与调度。具有多任务处理，协同求解，提高速度的特点，并利用冗余，提高可靠性、适应性、可用性。为了不同的目的，使用不同的技术途径，可以发展出同构型、异构型、分布型等形式各异的多处理机系统。

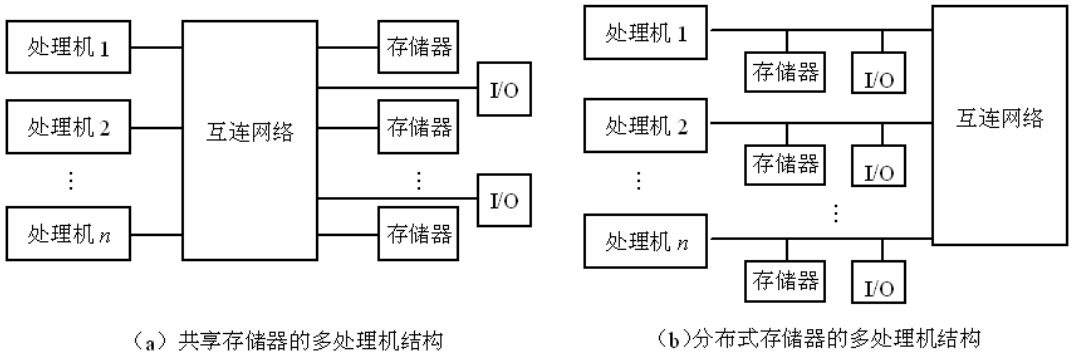
11.1.1 基本结构

多处理机是指两个或两个以上处理机(包括 PU 和 CU)，通过高速互连网络连接起来，在统一的操作系统管理下，实现指令以上级（任务级、作业级）并行。按照 Flynn 分类法，多处理机系统属于 MIMD 计算机，多处理机系统由多个独立的处理机组成，每个处理机都能够独立执行自己的程序。

多处理机有两种基本的结构：共享存储器结构和分布式存储器结构。这两种结构的多处理机都是通过底层的互联网络实现数据的交换和同步的，如图 11-1 所示。

共享存储器方案中，存储器和 I/O 设备是独立的子系统，为所有处理机所共享，这是实现信息交换和同步最简单的办法，任何两台处理机都可以通过共享存储器的单元实现通信。

分布式存储器结构每台处理机都有自己的存储器和 I/O 设备，处理机之间通过点对点的信息交换实现通信。整个存储器被分成多个模块，每个模块与一个处理机紧密结合，因此这种结构的存储模块也被称为是本地存储器。当处理机访问本地存储器时，不需要通过互联网络就可以直接进行，但是，系统内的任意一个处理机仍然可以通过互联网络访问系统中的任何一个存储器模块。



（a）共享存储器的多处理机结构　　（b）分布式存储器的多处理机结构

图 11-1 多处理机系统的两种基本结构

11.1.2 多处理机系统特点

多处理机属于多指令流多数据流(MIMD)计算机，它和单指令流多数据流(SIMD)计算机的并行处理机相比，有很大的差别。它们的差别归结底来源于并行性级别的不同：多处理机要实现任务一级的并行，不能再象 SIMD 计算机那样只能对多数据流执行同一指令操作。因此，在结构上，它的多个处理机要用多个指令部件分别控制，并且要有复杂的互连网络实现机间通信；在算法上，不限于数组向量处理，要挖掘和实现更多通用算法中隐含的并行性；在系统管理上，要更多依靠软件手段有效地解决资源管理，特别是处理机管理以及进程调度等问题。下面概括说明多处理机系统的特点。

1、结构灵活性

多处理机能适应更为多样的算法，具备更为灵活多变的系统结构以实现各种复杂的机间互连模式，同时还要解决共享资源的冲突问题。

2、程序并行性

在多处理机中，不限于解决数组向量处理问题，并行性存在于指令外部，即表现在多个任务之间，再加上系统通用性的要求，就使程序并行性的识别难度较大。因此，它必须利用多种途径，如算法、程序语言、编译、操作系统、以至指令、硬件等，尽量挖掘各种潜在的并行性，而且主要的责任不能放在程序员肩上。

3、并行任务派生

多处理机是处于多指令流操作方式，一个程序当中就存在多个并发的程序段，需要专门的指令来表示它们的并发关系以控制它们的并发执行，以便一个任务开始被执行时就能派生出可与它并行执行的另一些任务，这个过程称为并行任务派生。

4、进程同步

多处理机所实现的是指令、任务、程序级的并行。一般说，在同一时刻，不同的处理机执行着不同的指令。这就要求多处理机采取特殊的同步措施，才能使并发进程之间保持程序所要求的正确顺序。

5、资源分配和进程调度

多处理机执行并发任务，需用处理机的数目没有固定要求，各个处理机进入或退出任务的时刻互不相同，所需共享资源的品种、数量又随时变化。由于上述情况十分复杂，于是，就提出了一个资源分配和进程调度问题，这个问题解决的好坏对效率有很大的直接影响。

11.1.3 多处理机系统的 Cache 一致性问题

Cache 作为提高系统性能的一种技术手段在计算机系统中得到普遍的使用。在共享存储器的多处理机中，每台处理机都有自己的局部 Cache。这类多处理机在运行一个具有多个进程的程序时，各处理机可能会使用到共享存储器中的同一数据块，为维持多处理机的高速运行，这些共享数据块将被调入各处理机的局部 Cache 中。由于多个处理机是异步地独立操作，可能使共享存储器中同一数据块的不同 Cache 拷贝出现不一致，就可能危及系统的正常运行，这就是 Cache 的一致性问题。

1、产生 Cache 一致性问题的原因

出现 Cache 一致性问题的原因主要有 3 个：共享可写的数据、进程迁移、I/O 传输。

（1）共享可写数据引起的不一致

假设在写操作之前，Cache 的初始状态如图 11-2(a)所示。处理机 P_1 和 P_2 的局部高速缓冲存储器 $Cache_1$ 和 $Cache_2$ 中分别有共享存储器的某个数据 x 的副本。

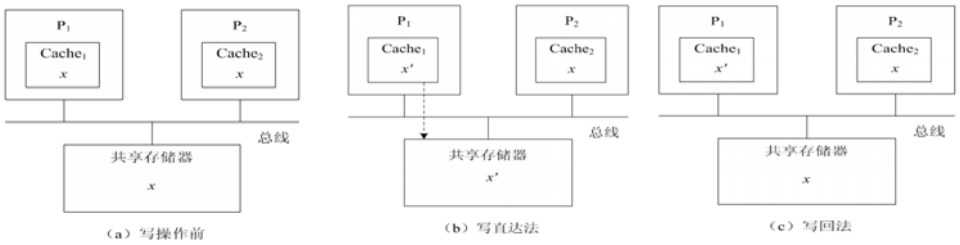


图 11-2 共享可写数据引起的 Cache 不一致性

若采用写直达法，如图 11-2 (b)所示，当处理机 P_1 改写 Cache1 中的数据为 *x′* 时，共享存储器中的相应数据也立即更新为 *x′*，这将导致 $Cache_1$ 中的数据与 $Cache_2$ 中所缓存的数据不一致。 若采用写回法，如图 11-2 (c)所示，当处理机 P1 改写 $Cache_1$ 中的数据为 *x′* 时，$Cache_1$ 的变化不会引起 $Cache_2$ 和共享存储器的变化，这将导致 $Cache_1$ 中的数据与共享存储器和 $Cache_2$ 中所缓存的数据不一致。

（2）进程迁移引起的不一致

在多处理机上，有时为了提高系统的效率而允许进程迁移，将一个尚未执行完而被挂起的进程调度到另一个空闲的处理机上去执行，使系统中各处理机的负荷保持均衡。但这样做可能会造成 Cache 一致性问题。

假设在迁移前 Cache 的初始状态仍如图 11-3(a)所示，处理机 P_1 的 $Cache_1$ 中有共享存储器中数据 *x* 的拷贝，而处理机 P_2 的 $Cache_2$ 中没有该数据。

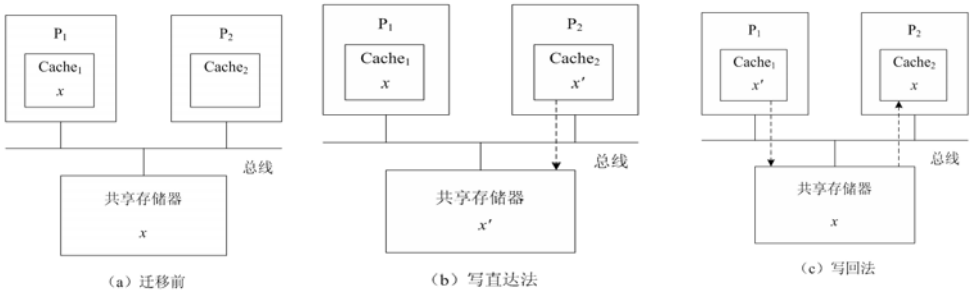


图 11-3 进程迁移引起的 Cache 不一致性

若采用写直达法，如图 11-3 (b)所示，当某进程从 P_1 迁移到 P_2 后，处理机 P_2 在执行此进程时需要使用数据 *x* 时，会将 *x* 所在块由共享存储器调入 $Cache_2$。假设进程运行时将 $Cache_2$ 的数据 *x* 改写为 *x′*，在共享存储器中的相应数据也立即更新为 *x′*，这将导致共享存储器和 $Cache_2$ 中的数据与处理机 P_1 中 $Cache_1$ 所缓存的数据的不一致。

若采用写回法，如图 11-3 (c)所示，当处理机 P_1 修改 $Cache_1$ 中的数据成 *x′* 时 $Cache_1$ 的变化不会引起共享存储器的变化。当某进程从 P_1 迁移到 P_2 后，处理机 P_2 在执行此进程时需要使用数据 *x* 时，会将 *x* 所在块由共享存储器调入 $Cache_2$。此时调入的数据 *x* 并非是已经修改过的 *x′*，这将导致共享存储器及 $Cache_2$ 与处理机 P_1 中 $Cache_1$ 所缓存的数据的不一致。

（3）I/O 操作引起的不一致

假设在执行 I/O 操作之前，Cache 的初始状态如图 11-4 (a)所示。处理机 P_1 和 P_2 的局部 Cache 中分别有共享存储器的某个数据 *x* 的拷贝。

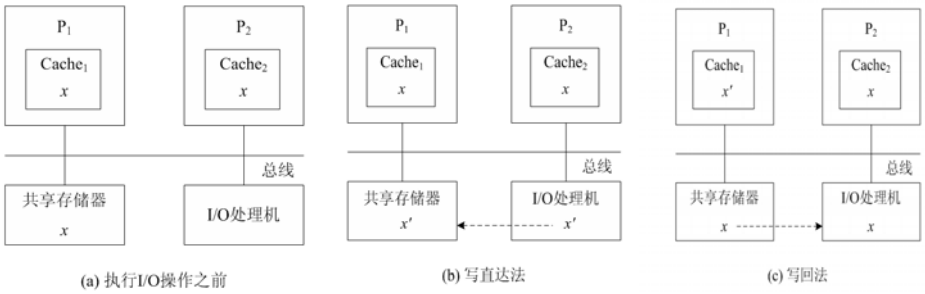


图 11-4　I/O 操作引起的 Cache 不一致性

若采用写直达法，当 I/O 处理机执行输入操作，直接将共享存储器中的数据 x 改写成 x’时，将导致 $Cache_1$ 和 $Cache_2$ 中的数据与共享存储器数据的不一致，如图 11-4(b)所示。若采用写回法，当处理机 P_1 将 $Cache_1$ 中的数据 x 改写为 x' 时，由于 $Cache_1$ 数据的修改不会立即引起共享存储器中数据 x 的更新，此时若此数据恰好被 I/O 处理机输出，就会造成输出错误，如图 11-4(c)所示。

2、解决 Cache 一致性问题的方法

解决多处理器维护 Cache 一致性的协议称为 Cache 一致性协议。实现 Cache 一致性协议的关键是跟踪共享数据块的状态。目前有两类 Cache 一致性协议，它们采用了不同的共享数据状态跟踪技术，分别适合于不同的系统结构。

（1）监听协议

每个 Cache 除了包含物理存储器中块的数据拷贝之外，也保存着各个块的共享状态信息。这是一种基于总线的一致性协议，各处理机通过监听总线上处理机与存储器之间的 Cache 操作事件，对各自局部 Cache 中的数据采取保持一致性的措施。监听协议有写无效策略和写更新策略两种策略来保持 Cache 一致性。

（2）目录协议

物理存储器中共享数据块的状态及相关信息均被保存在一个称为目录的地方。共享数据块的变化通过此数据块所在的各目录项，将一致性命令发给所有存放该数据块拷贝的 Cache。基于目录的协议的基本思想是：使用 Cache 目录来记录可以进入 Cache 的每个数据块的访问状态、该块在各个处理机的共享状态以及是否修改过等信息。把一致性命令只发给存有相应数据块拷贝的 Cache，从而支持 Cache 的一致性。各种目录协议的主要差别是目录存放什么信息以及如何维护信息。根据目录的结构特点，基于目录的协议可分为三类：全映射目录（full-map directory）、有限目录（limited directory）、链式目录（chained directory）。

（3）以软件为基础解决 Cache 一致性的方法

11.2　多处理机软件和典型的多处理机系统

11.2.1 并行算法

算法对于并行性的开发至关重要，算法必须适应具体的计算机结构。对表达式 $E_1=a+bx+cx^2+dx^3$ ，顺序算法与并行算法比较如图 11-5 所示。如果将运算过程用树形流程图来表示话，则运算的级数就是树的高度，用 T_p 代表；P 为所需处理机数目；S_p 为加速比，$S_p=T_1/T_p$；E_p 为效率，$E_p=S_p/P$ 。

对树进行变换来降低树高，减少运算级数，即可提高运算的并行性。树形结构可以用交换律、结合律、分配律来变换。先从算术表达式的最直接形式出发，利用交换律把相同的运算集中在一起；再利用结合律把参加运算的操作数（称原子）配对，尽可能并行运算，组成树高最小的子树；最后利用分配律，平衡各分支运算的级数，把这些子树结合起来，使总级数减至最少。

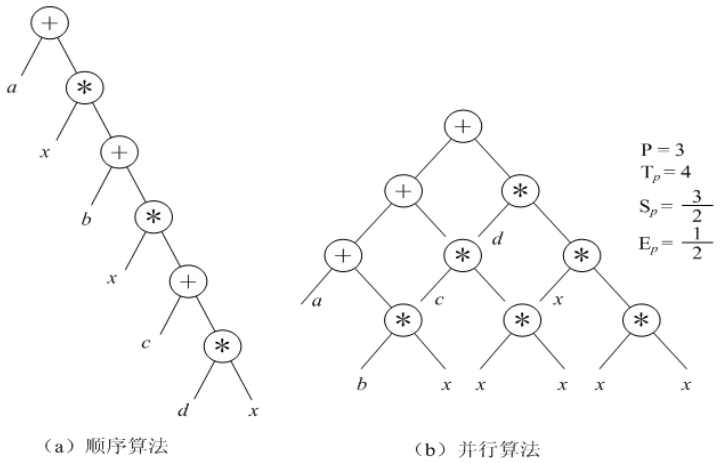


图 11-5 顺序算法与并行算法比较图

11.2.2 程序并行性分析

除算法以外，任务间能否并行在很大程度还取决于程序的结构形式。

假设一个程序包含 P_1、P_2、…、P_i、…、P_j、…、P_n 等 n 个程序段，其书写的顺序反映了该程序正常执行的顺序。为了便于分析，设 P_i 和 P_j 程序段都是一条语句，P_i 在 P_j 之前执行，且只讨论 P_i 和 P_j 之间的直接数据相关关系。

（1）数据相关

如果 P_i 的左部变量在 P_j 的右部变量集内，且 P_j 要从 P_i 取得运算结果来作为操作数，则称 P_j“数据相关”于 P_i。如

P_i　　A = B + D

P_j　　C = A * E

P_j 必须取 P_i 算得的 A 值作为操作数，相当于流水中发生的“先写后读”相关。

（2）数据反相关

如果 P_j 的左部变量在 P_i 的右部变量集内，且当 P_i 未取用其变量的值之前，不允许被 P_j 所改变，则称 P_i“数据反相关”于 P_j。如

P_i　　C = A * E

P_j　　A = B + D

在 A 的值未被 P_i 取用之前不能被 P_j 所改变，相当于流水中发生的“先读后写”相关。

（3）数据输出相关

如果 P_i 的左部变量也是 P_j 的左部变量，且 P_j 存入其算得的值必须在 P_i 存入之后。则称 P_j“数据输出相关”于 P_i。如

P_i　　A = B + D

P_j　　A = C + E

P_j 存入其算得的 A 值必须在 P_i 存入其结果 A 之后，相当于流水中发生的“写－写”相关。

对程序并行性的影响表现为下列几种可能的执行次序：写－读串行次序、读－写次序、可并行次序、必并行次序。

综上所述：两个程序段之间若有先写后读的数据相关，不能并行，只在特殊情况下可以交换串行；若有先读后写的数据反相关，可以并行执行，但必须保证其写入共享主存时的先读后写次序，不能交换串行；若有写－写的数据输出相关，可以并行执行 但同样需保证其写入的先后次序，不能交换串行；若同时有先写后读和先读后写两种相关，以交换数据为目的时，则必须并行执行，且要求读写完全同步，不许顺序串行和交换串行；若没有任何相关，或仅有右部源数据相同时，可以并行、顺序串行和交换串行。

11.2.3 并行程序设计语言

并行算法需要用并行程序来实现，而编写并行程序所用的程序语言中需要含有能明确表示并发进程的成分，这就要使用并行程序设计语言。并行进程的特点是多个进程在时间上重叠地执行，而并行程序设计语言必须便于具体描述这些并行关系。

1、描述程序并行性的语句

包含并行性的程序在多处理机上运行时，需要对并行任务的派生和汇合进行管理。

并行任务的派生和汇合通常是用软件手段来控制的。要在程序中反映出并行任务的派生和汇合关系，可以在程序语言中使用 FORK 语句来派生并行任务，用 JOIN 语句来实现对多个并发任务的汇合，读者如感兴趣，请参考相关资料。

2、并行编译

实现算术表达式的并行处理可以应用并行算法编制并行程序，还可以依靠并行编译程序。有一些编译算法，可以经过或不经过逆波兰式，直接从算术表达式产生能并行执行的目标程序。例如，对下列表达式：

Z = E + A * B * C/D + F

利用普通串行编译算法，产生三元指令组为：

1 *AB
2 *1C
3 /2D
4 +3E
5 +4F
6 =5Z

指令之间均相关，需 5 级运算。如采用并行编译算法可得

1 *AB
2 /2D
3 *12
4 +EF
5 +34
6 =5Z

其中，1，2 为第 1 级；3，4 为第 2 级；5，6 为第 3 级。分配给两个处理机，只需 3 级运算即可实现。

11.2.4 MPP 和 SMP

多处理机系统主要有四大类。第一类是多向量处理系统，以 CRAY YMP-90，NEC SX-3 和 FUJITSU VP-2000 等为代表；第二类是基于共享存储的多处理机系统，如 SGI Challenge 和 Sun SparcCenter 2000；第三类是基于分布存储的大规模并行处理系统（MPP），比如 Intel Paragon，CM-5，Cray T3D 等；第四类是机群系统。

1、MPP 大规模并行性并行处理

大规模并行性并行处理（massively parallel processing，MPP）系统的定义随着时间推移在不断地变化。按照当前的标准，具有几百或几千台处理机的任何机器就是大规模并行处理系统。显然，随着计算机技术的快速发展，对并行度的要求会愈来愈高。

MPP 系统最重要的特点是进行大规模并行处理。MPP 主要用于半导体建模、飞行动力学、气候模型、流体湍流、污染分析、人类染色体组、海洋环流、量子染色动力学等需要大量数值处理的领域。

我们以 1983 年美国的 128×128=16384 个微处理器(MPU)所组成规模并行处理机为例进行说明。此系统总结构如图 11-6 所示。其主要部分有 128×128 个处理机的阵列、驿站存储

器(Staging Memory)、阵列控制部件、宿主机。其中处理机的阵列由 2048 个处理机芯片组成，每个芯片上有 8 个处理机，组成 2×4 的子阵列，每个处理机附有一个 1024 位的 RAM，16384 个处理机可以并行工作

MPP 阵列的另一改进是在物理布局中增加了 4 列冗余的处理机（物理布局总数为 132 列）。当有处理机发生故障时，可以动态地进行重新组合，把存在故障的处理机旁路掉，因而极大地提高了系统的可靠性。 在阵列的上下各有一排 128 位的开关，分别用来输入输出数据。MPP 的时钟频率为 10*MHz*，每个时钟周期可以并行传送 128 位数据， 使最大传送速率可达到 1.28×10^9*bit/s*，即 160*MB/s*。

驿站存储器容量为 64*MB*，主要用来作为处理机阵列的输入输出缓冲器，它以 320*MB/s* 的速度从外部设备接收数据， 并能对数据进行压缩或重新安排格式的处理，使这些数据能更有效地在处理机阵列中进行处理。

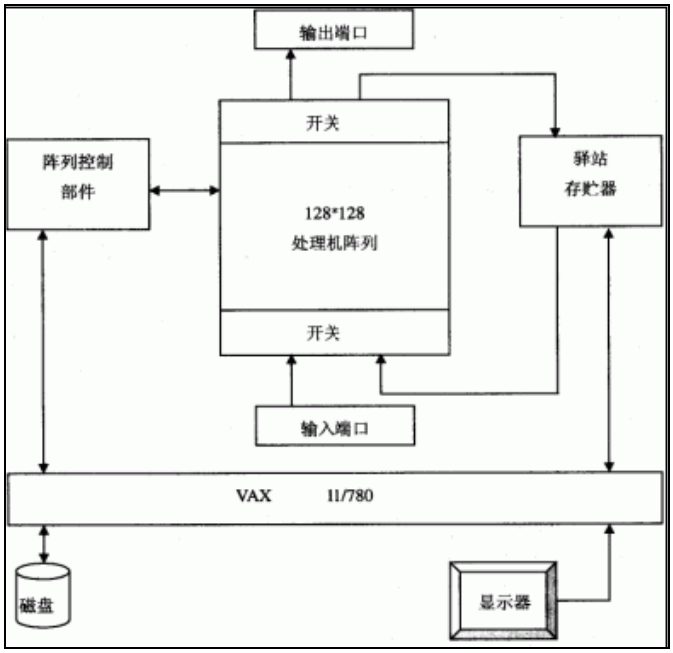


图 11-6 MPP 的系统结构图

阵列的控制部件实际是一台专用计算机， 执行 MPP 汇编语言，除了控制数据的输入输出通路外，还要执行存储在它的程序存储器中的应用程序，其功能包括进行循环的控制、子程序的调用及进行标量运算。MPP 系统用一台 VAX 11/780 机作宿主机， 整个系统在 VAX 11/780 机总的管理下工作的。PE 是位片式的处理机，它的算术运算用串行加法器按位串行进行，可处理任意长度的操作数。

MPP 的指令系统可分为三个子集：顺序指令、 并行指令和接口指令。与大多数 SIMD 计算机一样， MPP 还有一组接口指令，它们使数据在顺序部件和并行部件之间移动。 MPP 系统软件设计的基本准则是：用户像使用单机那样使用 MPP，但其性能却是单机系统的若干倍。

2、SMP 共享存储型多处理机

SMP 称为共享存储型多处理机(Shared Memory Multiprocessors)，也称为对称型多处理机(Symmetry Multiprocessors)。

共享存储系统拥有统一寻址空间，程序员不必参与数据分配和传输。早期的并行处理系统几乎都是基于总线的共享存储系统，它们的发展得益于两方面的原因：一个是微处理器令人难以置信的性能价格比，另一个是在基于微处理器的并行处理系统中广泛使用的 Cache

技术。这些因素使得将多个处理器放到同一条总线上，共享单一存储器成为可能，并通过Cache 将所有处理器访问存储器所需的存储带宽降低到可以接受的水平。Cache 一致性是通过监听协议实现的。这种实现方式虽然简单，但是阻碍了系统的扩展能力。

SMP 有三种模型，下面将分别作简单的介绍。

（1）UMA 多处理机

UMA 均匀存储器存取模型 (Uniform Memory Access)是指存储器被所有处理机均匀共享，所有处理机对所有存储单元具有相同的存取时间，每台处理机有局部 Cache，同时外围设备可以共享，如图 11-7 所示：

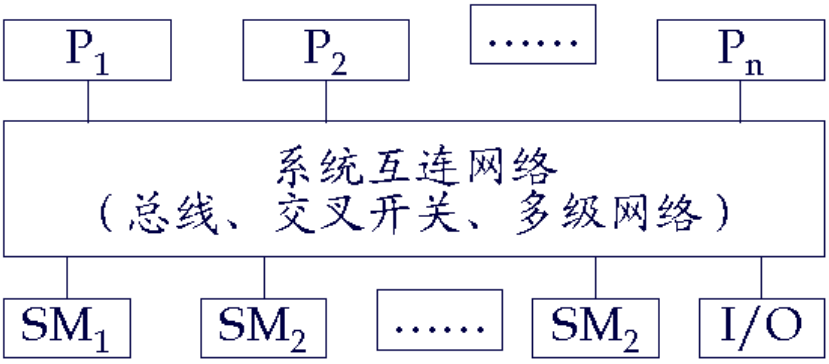


图 11-7 UMA 多处理机系统结构图

（2）NUMA 多处理机

NUMA 非均匀存储器存取 (Nonuniform Memory Access)模型是指存储器访问时间随存储单元的位置不同而变化。共享存储器在物理上是分布在所有处理机中的本地存储器，所有局部存储器地址空间的集合就组成了全局地址空间。

处理机访问本地存储器比较快，访问属于另一台处理机的远程存储器则比较慢，因为通过互连网络会产生附加的时间延迟，如图 11-8 所示：

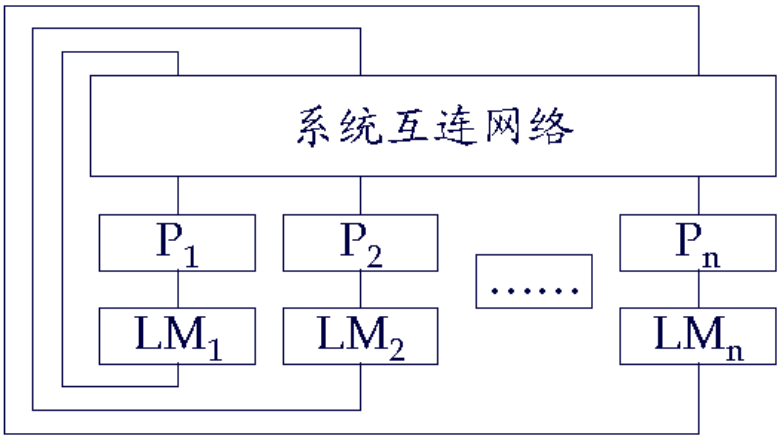


图 11-8 NUMA 多处理机系统结构图

（3）COMA 多处理机

COMA 只有 Cache 的存储器结构 (Cache-Only Memory Architecture) 模型是一种只用 Cache 的多处理机系统。实际上，COMA 模型是 NUMA 模型的一种特例，只是后者分布存储器换成了 Cache，同时在每个处理机结点上没有主存储器，全部 Cache 组成了全局虚拟地址空间。特点表现在远程 Cache 访问通过分布 Cache 目录进行，共享存储系统拥有统一的寻址空间，程序员不必参与数据分配和传输等。如下图 11-9 所示：

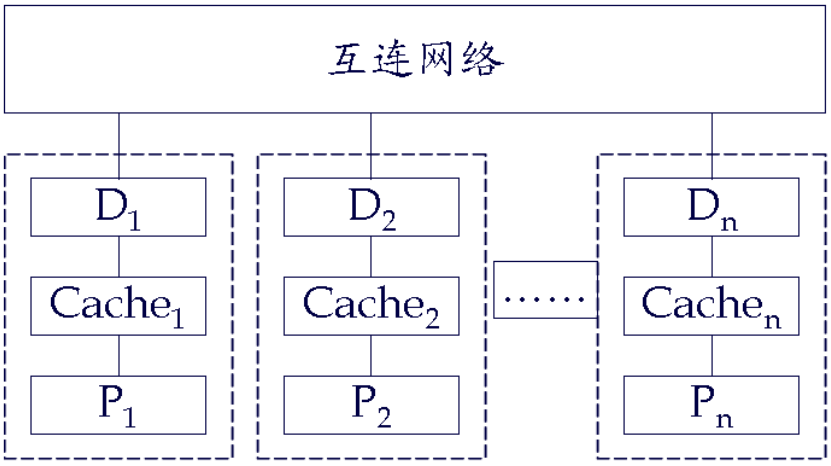


图 11-9 COMA 多处理机系统结构图

MPP 和 SMP 的区别如下所述：SMP (Symmetric Multi Processing),对称多处理系统内有许多紧耦合多处理器，在这样的系统中，所有的 CPU 共享全部资源，如总线，内存和 I/O 系统等，操作系统或管理数据库的复本只有一个，这种系统有一个最大的特点就是共享所有资源。 MPP (Massively Parallel Processing)，大规模并行处理系统，这样的系统是由许多松耦合的处理单元组成的，要注意的是这里指的是处理单元而不是处理器。每个单元内的 CPU 都有自己私有的资源，如总线，内存，硬盘等。在每个单元内都有操作系统和管理数据库的实例复本。这种结构最大的特点在于不共享资源。

既然有两种结构，那它们各有什么特点呢？采用什么结构比较合适呢？通常情况下，MPP 系统因为要在不同处理单元之间传送信息（请注意上图），所以它的效率要比 SMP 要差一点，但是这也不是绝对的，因为 MPP 系统不共享资源，因此对它而言，资源比 SMP 要多，当需要处理的事务达到一定规模时，MPP 的效率要比 SMP 好。这就是看通信时间占用计算时间的比例而定，如果通信时间比较多，那 MPP 系统就不占优势了，相反，如果通信时间比较少，那 MPP 系统可以充分发挥资源的优势，达到高效率。当前使用的 OTLP 程序中，用户访问一个中心数据库，如果采用 SMP 系统结构，它的效率要比采用 MPP 结构要快得多。而 MPP 系统在决策支持和数据挖掘方面显示了优势，可以这样说，如果操作相互之间没有什么关系，处理单元之间需要进行的通信比较少，那采用 MPP 系统就要好，相反就不合适了。

对于 SMP 来说，制约它速度的一个关键因素就是那个共享的总线，因此对于 DSS 程序来说，只能选择 MPP，而不能选择 SMP，当大型程序的处理要求大于共享总线时，总线就没有能力进行处理了，这时 SMP 系统就不行了。当然了，两个结构互有优缺点，如果能够将两种结合起来取长补短，当然最好了。

11.2.5 CM-5 系统

Connection Machine 系列的 CM-5 是 Thinking Machines 公司为实现 3T 性能目标（1*TFlops* 的计算速度、1*TByte* 主存储器和 1*TByte/s* 输入输出系统频带）而推出的 MPP 系统。

1、CM-5 的系统结构

CM-5 机器包含 32 到 16384 个处理器结点。每个结点有一台 32*MHz* 的 SPARC 处理机，32 兆字节的存储器和可以执行 64 位浮点和整数操作，速度为 128*Mflops* 的向量处理部件。系统采用若干台 SUN 公司的工作站计算机作为控制处理机。控制处理机的数目根据配置的不同而可从 1 台到几十台变化。每个控制处理机根据需要配有存储器和磁盘。系统的输入和输出通过高频宽的 I/O 接口与图形设备、海量辅助存储器以及高性能网络相连。与控制处理机相连的以太网提供低速 I/O 功能。配置在最大时的占地面积为 30 米×30 米，峰值速度可

望超过 1*Tflops*。

如图 11-10 所示，CM-5 系统有三个网络：数据网络、控制网络和诊断网络。数据网络和控制网络通过网络接口与处理结点、控制处理机和 I/O 通道相连。数据网络用于提供处理结点之间高性能点对点的数据通信。控制网络用于提供协同操作，包括广播、同步、扫描和系统管理功能。诊断网络允许从“后门”访问所有的系统硬件以便测试系统完整性，检测和隔离错误。

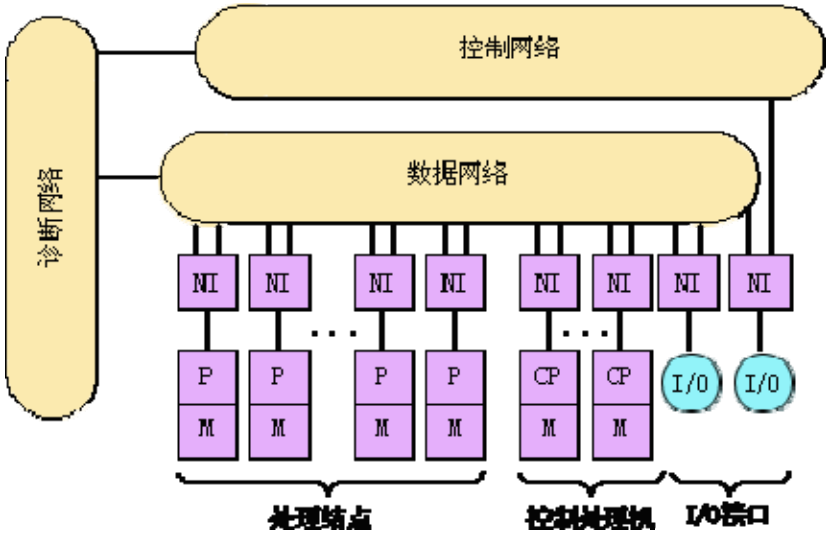


图 11-10 CM-5 计算机的系统结构

CM-5 系统结构是一种通用结构，它对大型复杂问题的数据并行处理，进而获得比较理想的结果。数据并行可用 SIMD 模式、多 SIMD 模式或者同步的 MIMD 模式实现。

2、CM-5 的网络结构

CM-5 的网络结构包括数据网络、控制网络和诊断网络。

（1）数据网络

数据网络是以 Leiserson 提出的胖树概念为基础设计的。胖树很象一棵真正的树，叶子越多时，它将变得越粗。处理结点，控制处理机和 I/O 通道都位于胖树的叶子上。树的内部结点都是开关。与一般的二进制树不同，从叶子到根，胖树的通道能力也随着增加。

（2）控制网络

该系统结构是一棵完全二叉树。所有的系统部件都在叶子上。可以给每个用户分区分配给一棵网络子树。处理结点位于子树的叶子处，控制处理机位于分区另外的叶子上。控制处理机执行代码中的标量部分，而处理结点执行数据并行部分。

（3）诊断网络

允许一组盒子按"超立方体编址"模式进行编址。一个专用的诊断接口用来对系统内部支持 JTAG(Join Test Action Group)标准的所有 CM-5 芯片以及所有的网络进行总体测试。它可以扫描访问支持 JTAG 标准的所有芯片以及可编程访问特定的非 JTAG 芯片。网络本身是完全可测试和可诊断的。它能够发现和排除机器的故障或掉电部件。

3、控制处理机和处理结点

控制处理机如图 11-11 所示，基本的控制处理机由 RISC 微处理器(CPU)、存储器子系统、带本地磁盘和以太网连接的 I/O 以及 CM-5 网络接口组成。它相当于一个流行的标准工作站类计算机系统。网络接口通过控制网络和数据网络将处理机与系统的其它部分相连。

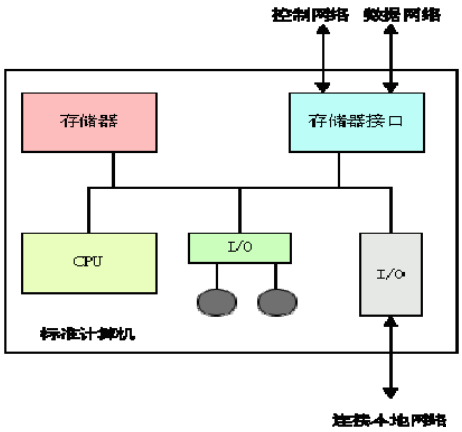


图 11-11 CM-5 控制处理机

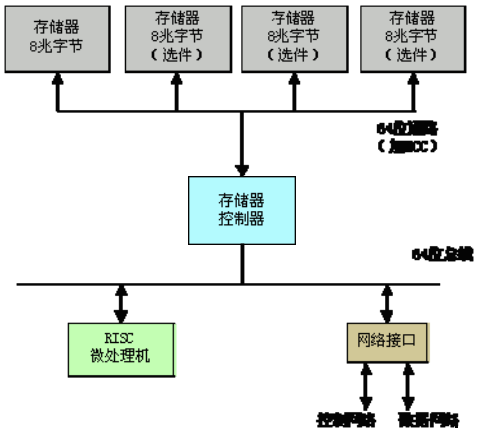


图 11-12 CM-5 处理结点基本结构图

图 11-12 给出了处理结点的基本结构。它是一台有存储器子系统的基于 SPARC 的微处理机。存储器子系统由存储器控制器和 8、16 或 32 兆字节的 DRAM 存储器组成。内部总线宽度为 64 位。

11.3.3 SGI Origin 2000 系列服务器

到 80 年代中期，对可扩展的多处理器系统的需求不断增长。基于总线的、cache 一致性、共享单一存储器的机器显然是不可扩展的。1996 年 SGI Origin 2000 系列服务器的推出，一种称为 S2MP 的并行计算机体系结构受到了广泛的注意。S2MP 全称为可扩展共享存储多处理(Scalable Shared-memory MultiProcessing)技术。S2MP 系统将大量高性能微处理器连接起来，共享一个统一的地址空间，较好地解决其他并行处理系统无法解决的问题。S2MP 是一种共享存储的体系结构，如下图 11-13 所示。

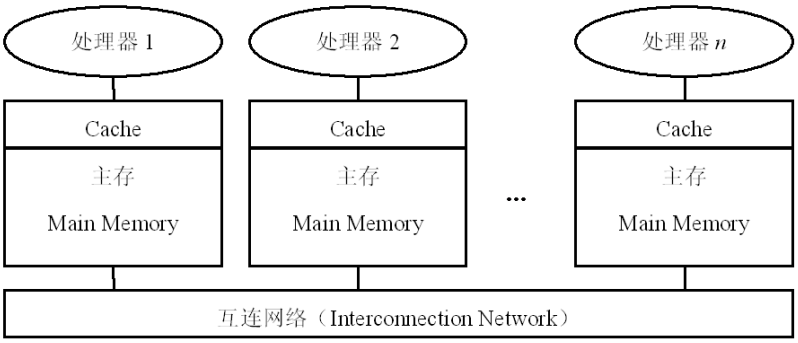


图 11-13 S2MP 系统的体系结构

和大规模的消息传递系统相比，它支持简单的编程模型，系统使用方便，是对 SMP 系统在支持更高扩展能力方面的发展。共享存储系统降低了通信的额外开销，因此系统也可以运行细粒度的应用。S2MP 着眼于扩展性能的研究，和传统的基于总线的共享存储并行处理系统相比，它不存在系统中可以连接的处理器数目的总线带宽的限制。

S2MP 作为大规模多处理系统，主要问题仍然是解决系统的可扩展和易编程能力。S2MP 系统采用分布式存储器技术，引入 cache，降低了访存时延。

S2MP 系统实际上是一种 NUMA 结构，每个结点由处理器和存储器两部分组成，存储器靠近处理器，而不是集中在某个地方，处理器可以访问本地存储器获取数据，NUMA 结构可以降低平均访存时延，并且随处理器数目的增加自动增加存储器带宽，也就是说存储带宽是可扩展的。由于在某个结点上访问本地存储器可以和其他结点板上的访存并行进行，系统的总带宽可以随着系统的规模而扩充。

SGI 公司将 Cray Research 子公司的开关网络技术应用到 S2MP 系统中，将 SMP、MPP

及工作站机群系统的优点结合起来，推出 Origin2000 系列可扩展服务器产品。Origin2000 系列服务器结构主要涉及到结点板、扩展连接方式、I/O 子系统、互连网络子系统、分布共享存储地址空间和系统时延。请读者自行参考相关信息。

11.3 机群系统

机群系统是指利用高速网络将一组计算机结点按某种结构连接起来，并在并行程序设计以及可视化人机交互集成开发环境支持下，统一调度、协调处理，实现高效并行处理的计算机系统。本节介绍机群系统的结构特点、关键技术及其提高通信系统的性能，并给出几种典型的机群系统。

11.3.1 机群系统的结构特点

从结构和结点间的通信方式来看，机群系统属于分布存储系统，主要利用消息传递方式实现各主机之间的通信，由建立在一般操作系统之上的并行编程环境完成系统的资源管理及相互协作，同时也屏蔽工作站及网络的异构性，对程序员和用户来说，机群系统是一个整体的并行系统。机群系统中的主机和网络可以是同构的，也可以是异构的。目前已实现和正在研究中的机群系统大多采用现有商用工作站和通用 LAN 网络，这样既可以缩短开发周期又可以利用最新的微处理器技术；大多数机群系统的并行编程环境也是建立在一般的 Unix 操作系统之上，尽量利用商用系统的研究成果，减少系统的开发与维护费用。

机群系统之所以能够从技术可能发展到实际应用主要是它与传统的并行处理系统相比有以下几个明显的特点：

（1）系统开发周期短

由于机群系统大多采用商用工作站和通用 LAN 网络， 使结点主机及系统管理相对容易，可靠性高。开发的重点在通信和并行编程环境上，既不用重新研制计算结点，又不用重新设计操作系统和编译系统，这就节省了大量的研制时间。

（2）用户投资风险小

用户在购置传统巨型机或 MPP 系统时很不放心，担心使用效率不高，系统性能发挥不好，从而浪费大量资金。而机群系统不仅是一个并行处理系统，它的每个结点同时也是一台独立的工作站，即使整个系统对某些应用问题并行效率不高，它的结点仍然可以作为单个工作站使用。

（3）系统价格低

由于生产批量小，传统巨型机或 MPP 的价格都比较昂贵，往往要几百万到上千万美元。工作站或高档 PC 机由于它们是批量生产出来的，因而售价较底。由近十台或几十台工作站组成的机群系统可以满足相当多数应用的要求，而价格却比较低。

（4）节约系统资源

由于机群系统的结构比较灵活，可以将不同体系结构，不同性能的工作站连在一起，这样就可以充分利用现有设备。单从使用效率上看，机群系统的资源利用率也比单机系统要高得多。UC Berkeley 计算机系 100 多台工作站的使用情况调查表明，一般单机系统的使用率不到 10%，而机群系统中的资源利用率可达到 80%左右。另一方面，即使用户设备更新，原有的一些性能较低或型号较旧的机器在机群系统中仍可发挥作用。

（5）系统扩展性好

从规模上说，机群系统大多使用通用网络，系统扩展容易；从性能上说，对大多数中、粗粒度的并行应用都有较高的效率。清华大学计算机系研制的可扩展机群系统上测试的结果表明 8 台工作站的加速比可以达到 5.83-7.9，并行处理的效率为 72.88%-99%。

（6）用户编程方便

机群系统中，程序的并行化只是在原有的 C、C++或 Fortran 串行程序中插入相应的通

信原语。用户使用的仍然是熟悉的编程环境，不用适应新的环境，这样就可以继承原有软件财富，对串行程序做并不很多的修改。

11.3.2 机群系统的关键技术

对于并行处理系统，人们希望要有较高的结点运算速度，系统的加速比性能接近线性增长，并行应用程序的开发要高效、方便。目前，机群系统大多采用商用高性能工作站或高档PC，结点的运算速度问题不是很突出，因而主要的研究方面是在提高系统的并行效率、使系统的使用更为方便上，包括建立高效的通信系统，有效地管理全局资源和提供友好的并行应用程序开发环境等。

1、高效的通信系统

机群系统一般使用通用局域网连接，目前常用的局域网技术大体可以分成两类，一类是共享介质网络，最常见是 10*Mbps* 或 100*Mbps* 的 Ethernet；另一类是开关网络， 比如 155Mbps/622*Mbps* 的 ATM，640*Mbps*/1.28*Gbps* 的 Myrinet 和 100*Mbps* 的交换式 Ethernet。对于共享介质网络，由于其聚合网络频带与单独链路频带是一样的，其性能会随网络负载的增加而下降，特别是对于某些负载比较集中的应用程序，这种影响会更明显，但是售价便宜，组成系统也相对容易，是组成中低档机群系统的一种较好的选择。而开关网络则相反，其聚合网络频带比单独的链路频率带要高得多，理论上讲是 N 倍；除开关的交换延迟影响外，性能不会随网络负载的增加而降底很多，开关网络的另一个优点是其可扩展性较好，由于 Wormhole、Cut-through 等交换技术的发展，交换延迟已经很低，与发送 / 接收端的开销相比要小得多。比如，Myrinet 开关的一次交换延迟小于 1*us*，一个中等规模的机群系统（16-32 台）的点-点的往返延迟仅有几十微秒。但是交换开关及相应接口卡的售价要高得多，组成机群系统的价格相对也比较高，对系统的普及有一定影响。

在不考虑网络负载的情况下，一般使用点-点的应用程序可见带宽和往返延迟来衡量通信系统的性能。应用程序可见带宽说明了网络的长消息包的传输性能，虽然由于网络技术的飞速发展，网络的物理链路越来越快，但是应用程序的可见带宽比起链路速度来要小得多，主要原因有网卡接口的硬件限制，协议处理开销和操作系统开销。例如，Myrinet 的物理链路是双向的 640*Mbps*，而在 TCP/IP 协议上点-点的应用程序可见带宽只有 38*Mbps*。往返延迟是 1 字节或 0 字节数据消息包的往返传输时间，它说明了网络短消息包的传输性能。新的网络技术大幅度地提高了传输速度，但往返延迟没有太大变化。

2、并行程序设计环境

随着 MPP 和机群系统等分布存贮结构并行系统的发展，开发出了 PVM，MPI，Express，P4 等基于 Message Passing 方式的并行程序设计环境，它为并行程序的设计和运行提供一个整体系统和各种辅助工具。功能包括提供统一的虚拟机、定义和描述通信原语、管理系统资源、提供可移植的用户编程接口和多种编程语言的支持。开发并行应用程序要比串行程序困难得多，它要涉及多个处理器之间的数据交换与同步，要解决数据划分、任务分配、程序调试和性能评测等问题，需要相应支持工具，比如并行调试器、性能评测工具、 并行化辅助工具，它们对程序的开发效率与运行效率都有重要作用。目前，提供工具较完善的系统有 FAUST, Express, TOPSYS 和 VIDE。

目前研制的机群系统大多支持 PVM 和 MPI，除了能适应广泛的硬件平台和编程方便等特点之外，它们都是免费软件，可以方便地进行再开发，有利于系统的推广与应用。正是由于它们都是免费软件，所以在支持语言、容错及工具等方面都不完善，许多研究机构和大学正在做这方面的研究。

3、多种并行语言的支持

并行程序设计语言是并行系统应用的基础，已有的机群系统大多支持 Fortran、C 和 C++, 实现的方法主要是使用原有顺序编译器链接并行函数库，比如 PVM，MPI，或者加入预编

译，比如 Multi-thread C, MPC++。目前机群系统并行程序设计语言的研究主要在三个方面：(1) 扩展原有顺序语言，提供广泛的并行语言支持，例如，清华大学可扩展机群系统的 ADA，MPC++。(2) 提供全新的并行语言，比如 Occam。(3) 研究自动化并行编译方法，直接将顺序程序编译成并行代码，目前比较成功的有 UIUC 的 Polaris, Stanford 的 SUIF 和复旦大学的 AFT。

4、全局资源的管理与利用

有效地管理系统中的所有资源是机群系统的一个重要方面，常用的并行编程环境 PVM, MPI 等对这方面的支持都比较弱，仅提供统一的虚拟机。主要原因是结点的操作系统是单机系统，不提供全局服务支持，同时也缺少有效的全局共享方法。UC Berkeley 的 NOW 项目中提出，在一般操作系统（Unix, Linux, Windows NT 等）之上建立一个全局 Unix: GLUnix 来解决机群系统中的所有资源管理，包括组调度、资源分配和并行文件系统。一般认为其中并行文件系统对提高系统的性能潜力最大，即所谓 Terabytes >> Teraflops, 就是说目前限制并行程序性能的因数主要来自 I/O 瓶颈，提高 I/O 性能的方法较之提高 CPU 速度更能增强并行系统的性能。

由于网络技术的发展，通信延迟越来越小，网络访问比本地磁盘访问要快得多。在 155*Mbits/s* 的 ATM 网络上，读取其他结点的内存 100*MBytes* 的时间是读取本地磁盘的五分之一。现在的工作站和高档 PC 机都配有相当多的内存(32*Mbytes* ~64*MBytes*)，整个机群系统的全部内存是一个很大的资源，利用其他结点的空闲内存作为本地结点的虚拟内存和文件缓存，可以节省相当多的访盘时间。据 UC Berkeley 的实验统计，对需要经常访盘的应用程序，使用这种方式可以比使用本地磁盘快 5~10 倍。

除了这几个主要方面的研究之外，还有许多特定应用方面的研究，比如，广播、多播等全局操作的高效实现、DSM 并行模型的支持、并行 I/O 的研究等。

5、机群负载平衡技术

在并行处理系统中，一个大的任务往往由多个子任务组成，这些子任务被分配到各个处理结点上并行执行，称之为负载。由于由各结点处理机结构不同，处理能力不同，使得各种子任务在其上运行时间和资源占有率不同。当整个系统任务较多时，各结点上的负载可能产生不均衡现象，就会影响整个系统的利用率。这就是负载不平衡问题，这个问题解决好坏直接影响到系统性能，因此它就成为并行处理中的一个重要问题。

为了充分利用高度并行的系统资源，提高整个系统的吞吐率，就需要负载平衡技术的支持，负载平衡技术的核心就是调度算法，即将各个任务比较均衡地分布到不同的处理结点并行计算，从而使各结点的利用率达到最大。

在机群系统上，负载平衡要解决的问题主要表现为系统资源使用不均和多用户系统资源分配问题。

11.3.3 提高通信系统的性能

通信子系统是并行计算机系统的重要组成部分，它完成系统中各结点之间数据传递的功能，其性能的好坏直接影响到并行计算的加速比和效率。这是因为并行计算时间是由各结点计算时间和结点间数据通信时间两部分组成，如果通信时间所占的比例过大，则必然使得并行计算的加速比下降，整个系统的效率也不会高。

并行机群系统是基于高性能工作站或高档微机和局域网（LAN）而发展起来的新系统，它是一个由若干微机或工作站通过普通 LAN 互联而成的松耦合计算机系统，这与大规模并行计算机（MPP）有较大差别，MPP 则通常是采用专用网络以紧耦合方式进行结点间的互联。与 MPP 相比，机群系统具有可扩展性好、性能/价格比高的特点，但网络带宽通常较低，如，使用广泛的传统以太网的带宽只有 10*Mb/s*，即使是新出现的一些高速网络的带宽也只不过上百 *Mb/s*，另外，局域网 LAN 所使用的通信协议通常是 TCP/IP 协议，这种协议处理

开销比较大，影响了网络硬件特性的发挥；而 MPP 的内部网络带宽通常都在数百 *Mb/s* 以上，而且是采用专用的通信协议，如 Paragon XP/S 的网络带宽可达到 1.4*Gb/s*，其通信协议是 NX 通信库。可见，机群系统中性能过低的通信系统会影响到整个并行计算效率的提高，因此要大力发展并行机群系统，一个关键的问题是对其通信技术进行深入的研究，以期大幅度地提高网络通信系统的性能。

1、影响通信系统性能的因素

机群系统下的通信子系统的性能是整个系统的薄弱环节，在介绍提高通信系统性能的方法措施之前，先从网络硬件、通信软件两方面分析影响通信系统性能的主要因素。

（1）网络带宽低

机群系统使用的网络是普通的局域网，而局域网的带宽通常都比较低，如传统以太网的带宽只有 10*Mb/s*。局域网的带宽之所以低，原因主要是局域网是为长距离的数据通信而设计的，由于通信距离较长，限制了通信速度的提高，因为信号的频率越高，它能够传输的距离也越短。另外一个原因是出于价格上的考虑。为了降低网络系统布线所需的成本，大多数 LAN 是共享一根信号总线进行数据传输，因此这也在很大程度上影响了网络系统的性能，特别是在网络负载较重时，由于各结点都要抢占信号总线，很容易造成通信阻塞，使得实际通信带宽比其最大带宽要小得多。

（2）传统 TCP/IP 协议的多层次结构带来了很大的处理开销

TCP/IP 协议是面向低速率、高差错和大数据包传输而设计的，它是一个多层次的软件结构，按自底向上的顺序划分，可分为四层：网络接口层、网络层（IP）、传输层（TCP）和应用层。由于协议层次多，在进行数据传输时，数据需要经过多次拷贝才能从应用层传递到网络接口或从网络接口传送到应用层，而多次的拷贝带来了很大的网络延迟时间。另外，在多层协议的实现中，各层还重复实现了很多相同的功能，比如从 IP 层到传输层都要进行差错控制、从网络接口层到应用层都要进行协议的处理机调度、从 IP 层到应用层都要进行流量控制、从 IP 层到应用层都要进行数据包组装和定序的缓冲。

这些冗余的功能虽然可确保数据的无差错传送，但确限制了数据及时提交给应用程序处理。可见，多层次的协议结构是造成通信瓶颈的主要原因之一，合并某些层次，删除冗余的处理，设计一种轻型通信协议，是提高通信性能的重要方法。

（3）协议复杂的缓冲管理增加了网络延迟

网络协议处理包括很多功能，如流量控制、差错控制、出错重发机制、拥塞控制等，而这些功能的实现都与缓冲管理密切相关。缓冲管理的作用是完成数据的分组和组装，缓冲区可看成一种网络资源，这种资源是有限的，对它的管理很重要。不过通常的缓冲管理机制都比较复杂，缓冲管理带来的网络延迟也很大，因此如何简化协议复杂的缓冲管理也是通信技术研究的主要内容。

（4）操作系统额外开销不可忽视

操作系统提供的系统调用和原语是网络协议实现的底层软件支持。在网络协议实现中涉及到上下文切换、调入/调出页面、启动 I/O 设备、中断响应等操作系统处理，有时这些开销可能比协议本身的处理开销还大。因此，要提高通信系统的性能，降低网络延迟，应当尽量减少网络协议对主机操作系统的服务请求，最大限度地使通信与计算重叠。

2、提高通信系统性能的方法措施

（1）采用新型高速网络，提高网络带宽

为了提高机群系统的网络带宽，必须采用新型的高速网络来取代 10*Mb/s* 以太网。由于多媒体应用、实时网络系统、大规模并行计算等应用对高速网络的需求，推动了网络技术的飞速发展，目前出现了多种新型的高速网络，如快速以太网、ATM、Myrinet。这些新型网络的传输速度是传统以太网的十倍或更高。由于高速网络的运用，使得影响通信系统性能的

瓶颈已从过去的网络硬件转移到网络通信软件上，因为虽然高速网络降低了网络的传输延迟，但并没有减少通信协议的处理开销。由于通信协议处理开销过大，在很大程度上阻碍了高速网实际性能的提高。

（2）设计新的通信协议，降低通信延迟

为了获得高带宽、低延迟的网络通信，必须对传统的通信协议作较大的修改，以克服传统协议的弊病，使高速网络的优越性能得以充分展现。主要方案有在用户空间实现通信协议、精简通信协议和利用 Active Message 通信机制。

为了减少操作系统的额外开销，一个重要的方法是在用户空间实现一个用户态的协议层，使得此协议层能够旁路操作系统的影响，直接对网络硬件设备进行操作，这样就可减少数据拷贝次数，提高通信效率；把协议实现放在用户空间的另一优点是可以减少操作系统调用的时间开销，而且通讯协议能够与用户的实际应用密切结合，可减少协议不必要的冗余，同时也不有损它的灵活性。不过，把通信协议放在用户空间实现，必须解决好两个问题：一个是多进程复用网络的问题；另一个是在没用核心参与的情况下，如何管理有限网络资源的问题。只有这样，用户态通信协议才能得以有效地实现。

前面的分析说明，通信的开销很大程度上是由于协议层次多、数据拷贝频繁引起的，另外，通用的网络接口和协议为满足各种用户的需求，增加了很多与数据传输无关的服务，这些服务也带来了额外的开销。而在并行机群系统中，有些功能是不必要的，完全可以进行精简，以降低通信开销。所谓精简，它包括两部分内容：一部分是功能的精简，就是删除不必要和冗余的功能；第二部分是协议层次的精简，合并各层的功能，使得通信协议变为一层，以达到减少数据拷贝次数的目的。比如，在操作系统 Solaris 2.4 中，通信协议由网络驱动程序、数据链路层、IP 层、TCP 层和 Socket 接口组成，由于数据链路层 DLPI 已经提供了不保证数据包无差错传送的基本数据通信功能，因此可以在它基础上实现一个保证数据可靠传送的模块以取代复杂的 TCP/IP 协议层和 Socket 接口，这样新的通信协议不论从结构上看，还是从功能上分析，都比原有的协议要简单得多。

通信协议的用户态实现以及协议的精简这两种方法都是针对传统通信协议在实现方法上所做的改进，而 Active Message 则是一种全新的通信机制，能够更为有效地提高通信系统的性能。

11.3.4 几种典型系统

目前国内外许多科研机构都在对机群系统下的通信技术进行深入的研究，如 UCB（University of California, Berkeley）提出的 NOW 计划，Cornell 大学研制的 U-Net 系统，清华大学提出的精简通信协议 RCP 等，如表 11-1 所示列举了在机群系统中实现的几种典型的通信子系统。

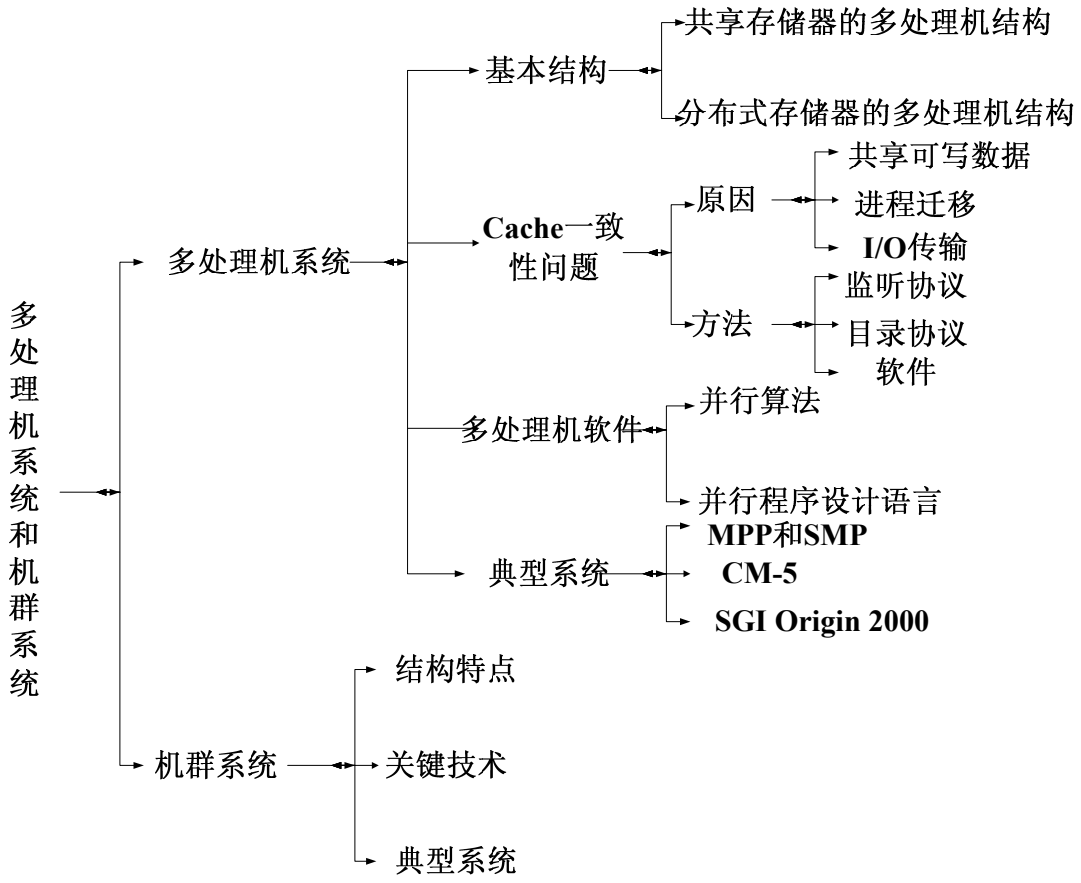
表 11-1　　典型通信机群子系统

系统名称	网络类型	实现技术	往返延迟时间(*ms*)	峰值宽度(*Mb/s*)
RCP(清华大学)	10Mbps Ethernet	精简通信协议	950(TCP 为 1438)	8.98(TCP 为 8.92)
FMP(清华大学)	1.28Gbps Myrinet	快速消息传递	24.8(TCP 为 220)	360(TCP 为 252)
Fast ocket(UCB)	640Mbps Myrinet	精简通信协议	80	96
FM(UIUC)	640Mbps Myrinet	Active Message	50	130
AM(UCB)	640Mbps Myrinet	Active Message	31.5	152
U-Net(Cornell)	155Mbps ATM	Active Message	65(TCP 为 1285)	120(TCP 为 80)
HPAM(UCB)	100Mbps FDDI	Active Message	29(TCP 为 2000)	96(TCP 为 43)

这些系统从实现技术上看，可以分成两类：一类是采用精简通信协议的方法；另一类是使用 Active Message 通信机制。对比这两类系统的性能可知，采用 Active Message 通信机制

实现的系统性能比用精简通信协议实现的系统一般来说要好一些。

关　联



习 题

11.1 解释下列概念

机群系统 共享存储多处理机 监听协议 基于目录的协议 Cache 一致性 MPP SMP S2MP UMA NUMA COMA

11.2 试比较多处理机系统与并行处理机系统在结构、工作方式及其应用范围方面的差同点。

11.3 共享存储器的多处理机系统有哪几种结构形式？画出它们的结构框图，并说明每种结构的主要特点。

11.4 多处理机系统为什么会出现 Cache 不一致的问题？两种解决不一致问题的方法各用在什么场合？

11.5 简要说明监听协议如何保证 Cache 的一致性。

11.6 比较集中共享与分布式存储器系统在结构上的特点，并说明它们各自的优缺点及适用场合。

11.7 简述 MPP 和 SMP 之间的区别。

11.8 什么是机群系统？发展机群系统的关键技术是什么？

11.9 提高机群系统的通信性能的途径有哪些？为什么要精简通信协议？

11.10 以 Origin 2000 为例，说明为什么分布式共享存储器系统有很好的可扩展性？

参考文献

1. Andrew S. Tanenbaum 著，刘卫东，徐恪译．结构化计算机组成原理．北京：机械工业出版社，2001
2. Gary B．Shell，Thomas J．Cashman，Misty E．Vermaat．Discovering Computers 2004，A Gateway to Information．THOMSON Course Technology，2003
3. 白中英．计算机组成原理(第三版)北京：科学出版社, 2001
4. 张功萱，顾一禾等．计算机组成原理. 北京：清华大学出版社，2005
5. 蒋本珊．计算机组成原理．北京：清华大学出版社，2003
6. 张五一，张道光．微型计算机原理与接口技术．郑州：河南科学技术出版社，2006
7. 贾金玲．微型计算机原理与接口技术．重庆：重庆大学出版社，2001
8. 王爱英．计算机组成与结构(第三版)．北京：清华大学出版社，2000
9. 郑玮民，汤志忠．计算机体系结构（第二版）．北京：清华大学出版社，1998
10. 马礼．计算机组成原理与系统结构．北京：人民邮电出版社，2004
11. 孙强南，孙昱东．计算机系统结构．北京：科学出版社，2000
12. Hennessy，J．L．等著，白跃彬译．计算机系统结构－量化研究方法（第四版）．北京：电子工业出版社，2007
13. 张钧良．计算机外围设备．北京：清华大学出版社，2005