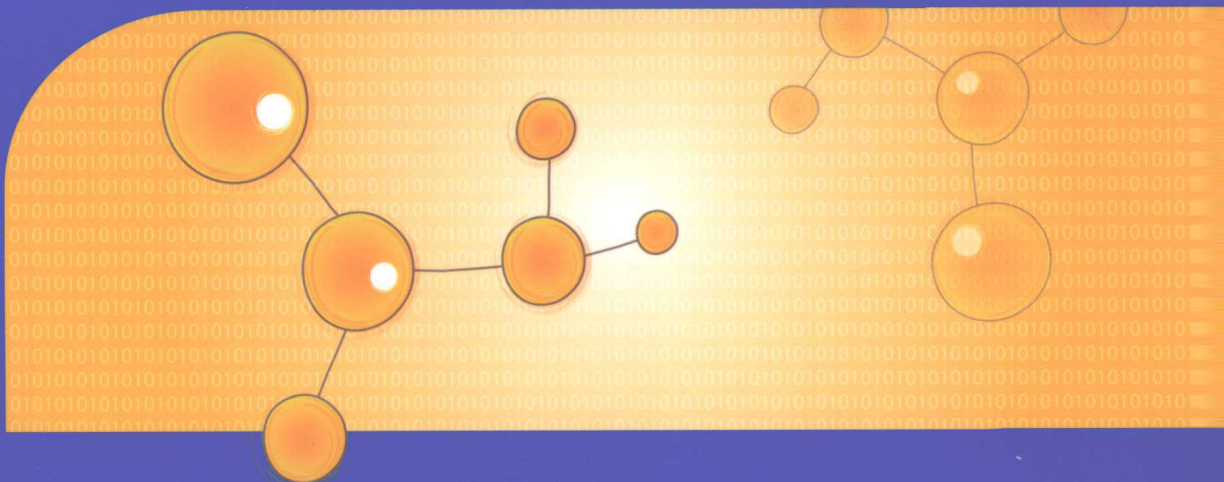




英特尔® 软件学院系列课程



多核多线程技术

Multicore Multithread Technology

英特尔® 亚太研发有限公司 组编

英特尔® 软件学院教材编写组 编



上海交通大学出版社

SHANGHAI JIAO TONG UNIVERSITY PRESS

内 容 提 要

本书就多核体系结构、芯片发展与系统软件,多性能并行程序,多线程程序的性能调优方法,多线程编程方法以及编程中的常见问题等作了综合讲述,处处体现了多线程编程理念与综合应用能力的培养。全书深入浅出,适合广大程序员和 IT 从事人员使用。

图书在版编目(CIP)数据

多核多线程技术/英特尔® 软件学院教材编写组
编. —上海:上海交通大学出版社,2011
英特尔软件学院系列课程培训教材
ISBN 978-7-313-06870-5
I. ①I… II. ①英… III. ①微处理器,
Intel 系列—程序设计—高等学校—教材 IV. ①TP332
中国版本图书馆CIP数据核字(2010)第197909号

多核多线程技术

英特尔® 软件学院教材编写组 编

上海交通大学出版社出版发行

(上海市番禺路951号 邮政编码200030)

电话:64071208 出版人:韩建民

上海崇明南海印刷厂印刷 全国新华书店经销

开本:787mm×960mm 1/16 印张:17.5 字数:318千字

2011年1月第1版 2011年1月第1次印刷

印数:1~5030

ISBN 978-7-313-06870-5/TP 定价:49.50元

版权所有 侵权必究

进入 21 世纪,信息技术和信息产业在全球范围内迅猛发展的势头更为强劲,如何尽快适应新技术和新应用带来的挑战,及时更新员工知识结构,并动态调整企业人才培养战略,已经成为广大科技公司迫切需要解决的问题之一。对于高等教育、职业教育等专业组织机构来说,则面临着紧跟企业前进步伐,准确接轨社会发展趋势,瞄准世界科技前沿水平,不断进行教育教学创新,提高学生实践能力,开拓学生知识视野的现实需求。在中国,尽管近年来已经在科技人才培养方面取得了长足的进步,但是就整体现状而言,尤其在知识更新和技术创新方面,距离完全满足社会的需求还存在着较大的发展空间。

英特尔公司历来关注技术的发展创新和科技人才的培养。英特尔®软件学院隶属于英特尔软件与服务事业部,作为英特尔公司专业的对外培训机构,为全球的软件开发人员提供了丰富的前沿技术培训课程。多年来,英特尔®软件学院一直致力于培训软件开发人员,与中国的软件开发人员共同发展,帮助其掌握和应用英特尔的最新技术及经验,提高软件开发技术水平,提升产品开发技能。目前,英特尔®软件学院在中国已经发展成为面向软件开发、项目管理及商业运营方向的优秀一站式培训服务基地。依托英特尔公司强大的师资力量,沿袭英特尔用户需求至上的传统,英特尔®软件学院已经与国内多家知名公司、大学和教育机构建立了长期稳定的



合作关系,迄今已有数万名工程师和大学教师参与了英特尔®软件学院的技术培训,并学以致用。

《英特尔®软件学院系列课程》由英特尔®软件学院牵头,联合国内的顶级高等学府合作编写。相信这本教材作为英特尔®软件学院的重点课程之一,在科技人才培养和知识创新方面必将发挥重要的作用。

**英特尔亚太研发有限公司总经理
英特尔公司软件与服务事业部中国区总经理
梁兆柱博士**

前 言

在多核体系结构出现以前的近 20 年里,程序员们已经习惯了由硬件的发展来自自然而然地获得程序性能的提高。即,每当出现一代新的体系结构平台,原有的程序无需修改,或者只需很少的修改,就可以轻而易举地获得由摩尔定律所带来的性能提升。但是,三个主要因素的日益凸显,使得这一免费午餐即将走到尽头:第一,系统建造者遇到了难以克服的物理阻碍——太多的发热量、太多的能量消耗以及过多的能量泄露,阻止了通过进一步提高时钟频率来提升性能的方法;第二,单个芯片上可以集成的引脚(pin)数目以及带宽限制,都意味着处理器与内存性能之间的差异只会越来越大;第三,为了解决以上两个问题,所在处理器体系结构上做的妥协将不足以使得单个处理器核心承担更高性能的计算需求。因此,要想在现在的多核体系结构上获得性能的提升,必须在原有的软件基础上作出大幅度的革新。这其中最主要的是利用多线程技术,充分利用好单个芯片上的多个计算核心,提高程序整体的计算吞吐量。

虽然在多核处理器出现之前,它一直属于一种比较深奥的理论,但多线程技术的出现已经有了几十年的历史。迄今为止,很多程序员都曾经在一些常见的多线程程序设计问题上经历过挫折。如何解决好这些问题?将是我们在本书中要与读者们共同探讨的重要问题。

在计算机产业,硬件和软件的发展一向是相辅相成、互相促进的,



多核这场硬件的革命自然也带动了软件的革命。在单核年代,程序员几乎很少考虑多线程编程的问题,因为即使是多线程的程序,在单核平台上也只能并发执行,不能并行执行,单线程程序和多线程程序在性能上的差异并不十分明显,相反,由于线程切换带来的开销,多线程程序的性能有时候还不如单线程程序。而在多核平台上,两者性能的差异非常明显,只要设计得当,多线程程序的性能将大大超越单线程程序。因此,程序员必须从以前的单线程思维中解放出来,投入到多线程程序设计中去。

多核程序设计已与程序员的开发工作密切相关,具备多核程序设计知识与能力已成为 21 世纪人才的基本素质之一,学会使用多核加快软件开发已是人们广泛而又迫切的现实需求。

高等学校是培养 21 世纪人才的主阵地,程序设计是高等学校计算机专业的主干课程,而多核程序设计是对程序设计课程的提升,是计算机专业与高性能计算有关专业的一门重要基础课程。大学生必须能在主流硬件平台上进行程序设计,利用多核的特点,这是程序员的基本素质之一。

本书第 1 章介绍了多核体系结构、芯片发展与系统软件;第 2 章综述了多线程并程序的性能分析方法;第 3 章针对 Intel 多核处理器介绍了

多线程程序的性能调优方法;第4章是多线程编程方法综述;第5章针对多线程编程中的常见问题进行了详细的分析并探讨了一些可能的解决途径;第6章介绍了 Unix/Linux 环境下利用 POSIX 标准线程库/接口进行多线程程序设计的具体接口和方法;第7章结合 Windows API 介绍了在 Windows 环境下进行多线程编程的基础知识和多线程编程的一些常用技术;第8章进一步探讨了一种以渐进方式,利用编译指导以及运行时库,将单线程程序逐步改造为多线程程序的 OpenMP 编程接口。纵观全书,处处体现出对多线程编程理念与综合应用能力的培养。

本书主要为从事程序设计的工程师、大学生编写。使他们在面对多核体系结构,以及需要多线程编程时有一个好的参考手册。也为高等学校计算机专业的师生进行多核多线程程序的教学提供了一本有价值的参考书。

由于作者水平有限,且多核处理器推出时日较短,书中难免存在一些缺点和错误,殷切希望广大读者批评指正。

编著者
2010年9月

目 录

| | |
|--------------------------------|----|
| 1 多核技术导论 | 1 |
| 1.1 微处理器发展史 | 1 |
| 1.1.1 计算机与微处理器 | 1 |
| 1.1.2 4位、8位与16位微处理器 | 2 |
| 1.1.3 32位微处理器 | 4 |
| 1.2 并行计算机 | 7 |
| 1.2.1 并行处理思想与弗林(Flynn)分类 | 7 |
| 1.2.2 超级计算机 | 8 |
| 1.3 片上多核处理器架构 | 10 |
| 1.3.1 多核芯片 | 10 |
| 1.3.2 片上多核处理器体系结构 | 11 |
| 1.3.3 典型多核芯片架构 | 12 |
| 1.4 操作系统对多核处理器的支持方法 | 14 |
| 1.4.1 调度与中断 | 14 |
| 1.4.2 输入输出系统 | 17 |
| 1.4.3 存储管理与文件系统 | 18 |
| 1.4.4 典型支持多核的操作系统 | 18 |
| 2 多线程并行程序性能分析方法综述 | 21 |
| 2.1 性能调优周期 | 21 |
| 2.1.1 搜集性能数据 | 23 |
| 2.1.2 分析数据并定位性能瓶颈 | 23 |



| | | |
|----------|----------------------------|-----------|
| 2.1.3 | 加速比性能定律 | 24 |
| 2.1.4 | 解决性能瓶颈 | 27 |
| 2.1.5 | 实现优化措施 | 29 |
| 2.1.6 | 测试 | 29 |
| 2.2 | 使用 Intel® VTune 进行性能分析 | 29 |
| 2.2.1 | 性能分析器功能与使用方法 | 30 |
| 2.2.2 | 性能分析器实验 | 36 |
| 2.3 | MKL 数学核心函数库 | 39 |
| 2.3.1 | MKL 数学核心函数库功能与特性 | 39 |
| 2.3.2 | MKL 数学核心函数库性能 | 41 |
| 2.3.3 | MKL 数学核心函数库实验 | 42 |
| 2.4 | Thread Checker 线程检查器 | 43 |
| 2.4.1 | 线程检查器功能与使用 | 44 |
| 2.4.2 | 线程检查器实验 | 46 |
| 2.5 | Thread Profiler 线程档案器 | 52 |
| 2.5.1 | 线程档案器功能与使用 | 53 |
| 2.5.2 | 线程档案器实验 | 54 |
| 3 | Intel 多核处理器上的性能调优方法 | 61 |
| 3.1 | 体系结构无关的性能调优方法 | 61 |
| 3.1.1 | 消除循环不变量 | 64 |
| 3.1.2 | 减少过程调用 | 64 |
| 3.1.3 | 消除不必要的内存存取 | 65 |
| 3.2 | 阻碍优化的因素 | 66 |
| 3.2.1 | 编译优化选项 | 66 |
| 3.2.2 | 变量别名 | 67 |
| 3.2.3 | 函数调用的边际效应 | 67 |
| 3.3 | 体系结构无关优化的小结 | 69 |
| 3.4 | 深入理解 Intel 多核处理器体系结构 | 70 |
| 3.4.1 | Intel 多核处理器微体系结构的显著特性 | 71 |
| 3.5 | Intel 多核处理器体系结构相关的优化 | 73 |
| 3.5.1 | Intel 多核处理器微体系结构中对代码优化的支持 | 73 |

| | |
|-----------------------------------|-----|
| 4 多线程编程方法综述 | 76 |
| 4.1 线程的基本概念 | 76 |
| 4.1.1 线程与进程的区别 | 77 |
| 4.1.2 用户级线程、核心级线程和硬件线程 | 78 |
| 4.1.3 线程的生命周期 | 79 |
| 4.2 线程的同步 | 79 |
| 4.2.1 竞争条件 | 80 |
| 4.2.2 临界区 | 80 |
| 4.2.3 信号量 | 81 |
| 4.2.4 锁 | 81 |
| 4.2.5 条件变量 | 83 |
| 4.2.6 线程的本地存储 | 84 |
| 4.2.7 介绍 TLS 的特性和使用方法 | 85 |
| 4.3 多线程编程模型 | 88 |
| 4.3.1 流水线 | 88 |
| 4.3.2 工作组 | 89 |
| 4.3.3 客户/服务器方式 | 90 |
| 4.4 多线程编程的原则及要点 | 90 |
| 4.4.1 静态负载平衡 | 91 |
| 4.4.2 动态负载平衡 | 91 |
| 4.4.3 负载平衡的难题 | 91 |
| 4.4.4 串行化方面的难题 | 92 |
| 5 多线程程序设计中的常见问题及解决途径 | 97 |
| 5.1 线程过多 | 97 |
| 5.2 数据竞争、死锁和活锁 | 100 |
| 5.2.1 数据竞争 | 100 |
| 5.2.2 死锁和活锁 | 103 |
| 5.3 竞争激烈的锁 | 105 |
| 5.3.1 优先级倒置 | 105 |
| 5.3.2 锁竞争激烈的解决方法 | 109 |
| 5.4 非阻塞算法 | 111 |
| 5.4.1 比较并交换 | 112 |



| | | |
|----------|-------------------------------|------------|
| 5.4.2 | 原子变量类 | 112 |
| 5.4.3 | 非阻塞算法的介绍 | 113 |
| 5.4.4 | ABA 问题 | 119 |
| 5.4.5 | cache 线乒乓现象 | 120 |
| 5.4.6 | 存储空间回收问题 | 120 |
| 5.4.7 | 一些建议 | 121 |
| 5.5 | 线程安全函数和库 | 121 |
| 5.5.1 | 理解可重入与线程安全 | 121 |
| 5.5.2 | 函数可重入化 | 122 |
| 5.5.3 | 函数线程安全化 | 125 |
| 5.6 | 存储问题 | 127 |
| 5.6.1 | 带宽 | 127 |
| 5.6.2 | cache 的利用 | 128 |
| 5.6.3 | 存储竞争 | 130 |
| 5.7 | Cache 相关问题 | 133 |
| 5.7.1 | 伪共享 | 133 |
| 5.7.2 | 存储一致性 | 134 |
| 5.7.3 | 当前 IA-32 体系结构 | 135 |
| 5.7.4 | Itanium 体系结构 | 137 |
| 5.7.5 | 高级语言 | 140 |
| 5.8 | 避免 IA-32 上的流水线停顿 | 140 |
| 5.9 | 面向高性能的数据组织 | 140 |
| 6 | Unix/Linux 多线程编程 | 142 |
| 6.1 | POSIX 的一些基本知识 | 142 |
| 6.2 | POSIX 线程库 | 144 |
| 6.2.1 | 创建线程 | 144 |
| 6.2.2 | 分离和接合线程 | 145 |
| 6.2.3 | 退出和取消线程 | 146 |
| 6.2.4 | 用户级线程和内核级线程 | 147 |
| 6.2.5 | 线程的属性 | 149 |
| 6.2.6 | 线程安全函数 | 153 |
| 6.2.7 | 线程特定数据 | 155 |

| | | |
|----------|--------------------------|------------|
| 6.2.8 | 一个 POSIX 多线程实例 | 158 |
| 6.3 | 线程通信 | 159 |
| 6.3.1 | 互斥量 | 159 |
| 6.3.2 | 条件变量 | 161 |
| 6.3.3 | 信号处理 | 165 |
| 6.3.4 | 读写锁 | 171 |
| 6.3.5 | 信号量 | 174 |
| 7 | Windows 多线程编程 | 180 |
| 7.1 | Windows 操作系统的一些基本知识 | 180 |
| 7.2 | Win32 API 的线程库 | 181 |
| 7.2.1 | Windows 操作系统中对进程概念的定义 | 181 |
| 7.2.2 | Windows 操作系统中对线程概念的定义 | 182 |
| 7.2.3 | 使用 Win32 线程 API | 182 |
| 7.3 | 线程间通信 | 192 |
| 7.3.1 | 互锁函数 | 193 |
| 7.3.2 | 临界段 | 196 |
| 7.3.3 | 使用内核对象的线程间通信 | 200 |
| 7.3.4 | 事件 | 202 |
| 7.3.5 | 互斥量 | 206 |
| 7.3.6 | 信号量 | 209 |
| 7.4 | 调度优先级 | 211 |
| 7.5 | 线程池 | 214 |
| 8 | OpenMP 多线程编程及性能优化 | 217 |
| 8.1 | OpenMP 编程简介 | 217 |
| 8.1.1 | OpenMP 多线程编程发展概况 | 217 |
| 8.1.2 | OpenMP 多线程编程基础 | 218 |
| 8.1.3 | 编写 OpenMP 程序的准备工作 | 221 |
| 8.2 | OpenMP 多线程应用程序编程技术 | 224 |
| 8.2.1 | 循环并行化 | 224 |
| 8.2.2 | 并行区域编程 | 236 |
| 8.2.3 | 线程同步 | 242 |



| | | |
|-------|-------------------------|-----|
| 8.3 | OpenMP 多线程应用程序性能分析..... | 255 |
| 8.3.1 | 影响性能的主要因素 | 256 |
| 8.3.2 | OpenMP 程序性能分析实例 | 258 |

1 多核技术导论

1.1 微处理器发展史

1.1.1 计算机与微处理器

1945年,美国研制了世界上第一台全自动电子数字计算机 ENIAC(Electronic Numerical Integrator and Calculator,即电子数字积分器和计算器),是美国奥伯丁武器试验场为了满足计算弹道需要而研制成的,于1946年2月交付使用,共服役9年。它采用电子管作为计算机的基本元件,每秒可进行5000次加减运算。它使用了18000只电子管、10000只电容、7000只电阻,体积3000 in³,占地170 m²,重量30 t,耗电140~150 kW。ENIAC机的问世具有划时代的意义,表明计算机时代的到来。在以后的40多年里,计算机技术发展异常迅速,在人类科技史上还没有一种学科可以与电子计算机的发展速度相提并论。

计算机的发展按照硬件工艺可以分为以下几代。

第一代(1946~1958年):电子管数字计算机。计算机的逻辑元件采用电子管,主存储器采用汞延迟线、磁鼓、磁芯;外存储器采用磁带;软件主要采用机器语言、汇编语言;应用以科学计算为主。其特点是体积大、耗电大、可靠性差、价格昂贵、维修复杂。它奠定了以后计算机技术的基础。

第二代(1958~1964年):晶体管数字计算机。晶体管的发明推动了计算机的发展,逻辑元件采用了晶体管以后,计算机的体积大大缩小,耗电减少,可靠性提高,性能比第一代计算机有很大的提高。主存储器采用磁芯,外存储器已开始使用更先进的磁盘;软件有了很大发展,出现了各种各样的高级语言及其编译程序,还出现了以批处理为主的操作系统,应用以科学计算和各种事务处理为主,并开始用于工业控制。

第三代(1964~1971年):集成电路数字计算机。20世纪60年代,计算机的

逻辑元件采用小规模集成电路 (Small Scale Integration, SSI)、中规模集成电路 (Medium Scale Integration, MSI), 计算机的体积更小型化、耗电量更少、可靠性更高, 性能比第二代计算机又有了很大的提高。这时, 小型机也蓬勃发展起来, 应用领域日益扩大。主存储器仍采用磁芯, 软件逐渐完善, 分时操作系统、会话式语言等多种高级语言都有新的发展。

第四代(1971年以后): 大规模集成电路数字计算机。计算机的逻辑元件和主存储器都采用了大规模集成电路 (Large Scale Integration, LSI)。所谓大规模集成电路是指在单片硅片上集成 1 000~2 000 个以上晶体管的集成电路, 其集成度比中、小规模集成电路提高了 1~2 个以上数量级。这时计算机发展到了微型化、耗电极少、可靠性很高的阶段。大规模集成电路使军事工业、空间技术、原子能技术得到发展, 这些领域的蓬勃发展对计算机提出了更高的要求, 有力地促进了计算机工业的空前大发展。

随着大规模集成电路技术的迅速发展, 计算机除了向巨型机方向发展外, 还朝着超小型机和微型机方向飞速前进。

微型机的核心是微处理器, 微处理器被誉为 20 世纪最伟大的发明之一。微处理器的设计很规则, 可以使工作更加简单, 其指令尽量简单, 格式统一, 寻址方式简单, 硬件单元也尽可能功能单一化、简单化。这样能够大大降低处理器设计的复杂性, 提高系统开发的效率。微处理器的功能单元和寄存器等应该在满足需求的情况下尽可能少, 因为电信号传输距离越远, 传输距离越长。因此, 寄存器太多将会延长时钟周期。微处理器使用的设计是合适的折中, 很多情况下, 简单和高性能并不能统一, 这时就需要合适的折中, 选择一个性能损失不大却又最简单的实现方案。

由于微处理器具有体积小、重量轻、功耗低、功能强、可靠性高、结构灵活、使用环境要求低、价格低廉等一系列特点和优点, 因此得到了广泛的应用, 使计算机真正进入到人类社会生产和生活的各个方面。计算机从过去只限于各部门、各单位少数专业人员使用普及到广大民众乃至中小学生, 成为人们工作和生活不可缺少的工具, 从而将人类社会推进到了信息时代。

1.1.2 4 位、8 位与 16 位微处理器

1971 年, 英特尔公司推出了世界上第一款微处理器 4004, 这是第一个可用于微型计算机的四位微处理器, 它包含 2 300 个晶体管。随后英特尔又推出了 8008, 1974 年, 8008 发展成 8080, 成为第二代微处理器。8080 作为代替电子逻辑电路的器件被用于各种应用电路和设备中, 如果没有微处理器, 这些应用就无法实现。

由于微处理器可用来完成很多以前需要用较大设备完成的计算任务,价格又便宜,于是各半导体公司开始竞相生产微处理器芯片。Zilog 公司生产了 8080 的增强型 Z80,摩托罗拉公司生产了 6800,英特尔公司于 1976 年又生产了增强型 8085。这些芯片基本没有改变 8080 的基本特点,都属于第二代微处理器。它们均采用 NMOS 工艺,集成度约 9 000 只晶体管,平均指令执行时间为 $1\sim 2\ \mu\text{s}$,采用汇编语言、BASIC、Fortran 编程,使用单用户操作系统。

1978 年,英特尔公司生产的 8086 是第一个 16 位的微处理器。很快,Zilog 公司和摩托罗拉公司也宣布计划生产 Z8000 和 68000。这就是第三代微处理器的起点。

8086 微处理器最高主频速度为 8 MHz,具有 16 位数据通道,内存寻址能力为 1 MB。同时英特尔还生产出与之相配合的数学协处理器 i8087,这两种芯片使用相互兼容的指令集,但 i8087 指令集中增加了一些专门用于对数、指数和三角函数等数学计算的指令。人们将这些指令集统一称之为 X86 指令集。

1979 年,英特尔公司又开发出了 8088。8086 和 8088 在芯片内部均采用 16 位数据传输,所以都称为 16 位微处理器,但 8086 每周期能传送或接收 16 位数据,而 8088 每周期只采用 8 位。因为最初的大部分设备和芯片是 8 位的,而 8088 的外部 8 位数据传送、接收能与这些设备相兼容。8088 采用 40 针的 DIP 封装,工作频率为 6.66 MHz、7.16 MHz 或 8 MHz,微处理器集成了大约 29 000 个晶体管。

8086 和 8088 问世后不久,英特尔公司就开始对它们进行改进,他们将更多功能集成在芯片上,这样就诞生了 80186 和 80188。这两款微处理器内部均以 16 位工作,在外部输入输出上,80186 采用 16 位,而 80188 和 8088 一样是采用 8 位工作。

1981 年,美国 IBM 公司将 8088 芯片用于其研制的 PC 机中,从而开创了全新的微机时代。也正是从 8088 开始,个人电脑(PC)的概念开始在全世界范围内发展起来。从 8088 应用到 IBM PC 机上开始,个人电脑真正走进了人们的工作和生活之中,它也标志着一个新时代的开始。

1982 年,英特尔公司在 8086 的基础上,研制出了 80286 微处理器,该微处理器的最大主频为 20 MHz,内部、外部数据传输均为 16 位,使用 24 位内存储器的寻址,内存寻址能力为 16 MB。80286 可工作于两种方式:一种叫实模式,另一种叫保护方式。在实模式下,微处理器可以访问的内存总量限制在 1 兆字节;而在保护方式之下,80286 可直接访问 16 兆字节的内存。此外,80286 工作在保护方式之下,可以保护操作系统,使之不像实模式或 8086 等不受保护的微处理器那样,在遇到异常应用时会系统停机。

IBM 公司将 80286 微处理器用在先进技术微机(AT 机)中,引起了极大的轰动。80286 在以下四个方面比它的前辈有显著的改进:支持更大的内存,能够模拟内存空间,能同时运行多个任务,提高了处理速度。最早 PC 机的速度是 4 MHz,第一台基于 80286 的 AT 机运行速度为 6~8 MHz,一些制造商还自行提高速度,使 80286 达到了 20 MHz,这意味着性能上有了重大的进步。

1.1.3 32 位微处理器

1985 年 10 月 17 日,英特尔划时代的产品 80386 DX 正式发布,其内部包含 27.5 万个晶体管,时钟频率为 12.5 MHz,后逐步提高到 20 MHz、25 MHz、33 MHz、40 MHz。80386 DX 的内部和外部数据总线是 32 位,地址总线也是 32 位,可以寻址到 4 GB 内存,并可以管理 64 TB 的虚拟存储空间。它的运算模式除了具有实模式和保护模式以外,还增加了一种“虚拟 86”的工作方式,可以通过同时模拟多个 8086 微处理器来提供多任务能力。虽然当时 80386 没有完善和强大的浮点运算单元,但配上 80387 协处理器,80386 就可以顺利完成许多需要大量浮点运算的任务,从而顺利进入了主流的商用电脑市场。另外,80386 还有其他丰富的外围配件支持,如 82258(DMA 控制器)、8259A(中断控制器)、8272(磁盘控制器)、82385(Cache 控制器)、82062(硬盘控制器)等。针对内存的速度瓶颈,英特尔为 80386 设计了高速缓存(Cache),采取预读内存的方法来缓解这个速度瓶颈,从此以后,Cache 就和 CPU 成为如影随形的东西。

1989 年,英特尔推出 80486 芯片。这款芯片首次突破了 100 万个晶体管的界限,集成了 120 万个晶体管,使用 $1\ \mu\text{m}$ 的制造工艺。80486 的时钟频率从 25 MHz 逐步提高到 33 MHz、40 MHz、50 MHz。80486 是将 80386 和数学协微处理器 80387 以及一个 8 KB 的高速缓存集成在一个芯片内。80486 中集成的 80487 的数字运算速度是以前 80387 的两倍,内部缓存缩短了微处理器与慢速 DRAM 的等待时间。并且,在 80X86 系列中首次采用了 RISC(精简指令集)技术,可以在一个时钟周期内执行一条指令。它还采用了突发总线方式,大大提高了与内存的数据交换速度。由于这些改进,80486 的性能比带有 80387 数学协微处理器的 80386 DX 性能提高了 4 倍。

1993 年,新一代 586 CPU 问世。为了摆脱 486 时代微处理器名称混乱的困扰,英特尔公司把自己的新一代产品命名为 Pentium(奔腾)以区别 AMD 和 Cyrix 的产品。Pentium 最初级的 CPU 是 Pentium 60 和 Pentium 66,分别工作在与系统总线频率相同的 60 MHz 和 66 MHz 两种频率下,没有倍频设置。早期的奔腾 75~120 MHz 使用 $0.5\ \mu\text{m}$ 的制造工艺,后期 120 MHz 频率以上的奔腾则改用